

ALMA MATER STUDIORUM – UNIVERSITY OF BOLOGNA
CESENA'S CAMPUS

SCHOOL OF ENGINEERING AND ARCHITECTURE

MASTER OF SCIENCE IN COMPUTER SCIENCE AND ENGINEERING

Enterprise Social Networks: The Case of CERN

Thesis in
Distributed Systems

Supervisor
Prof. Andrea Omicini

Presented by
Marco Carlo Cavalazzi

Co-supervisor
Dr. Stefano Mariani

Advisor
Prof. Mario Bravetti

Session II
Academic Year 2015/2016

To my family, all over the world, and to meine schöne und clever Freundin, Beatrice. To Elena Vogliolo and Guido Langé, always there for me. To my Italian friends, that push me to become a better man without asking me to change who I am. To my professors, Marco Boschetti, Andrea Omicini, Mario Bravetti and Antonella Carbonaro, without whom all this would not have happened. To my great supervisor at CERN, Bruno Silva De Sousa, who inspires me to do more. A Jean-Alain Brulhart, qui m'a fait vivre Genève, and to all my "crazy" friends in Saint-Genis-Pouilly and Geneva, they made the experience even more worth it.

Table of Contents

Sommario.....	1
Abstract.....	3
Introduction.....	5
1. Web 2.0.....	9
1.1. Social Networks.....	13
1.1.1. Origins of Social Networks.....	14
1.1.2. What is a Social Network.....	18
1.1.3. How does it work?.....	22
1.2. Enterprise Social Networks (ESNs).....	23
2. Knowledge Management.....	27
2.1. What is “Knowledge”?.....	27
2.2. The Knowledge Worker.....	28
2.3. Knowledge Management and Knowledge Sharing.....	30
2.4. Barriers to Knowledge Sharing.....	32
3. Transition to the Social Organization.....	39
3.1. Introducing the Social Business.....	39
3.2. Key Enterprise 2.0 tools.....	42
3.3. ESNs Examples.....	45
3.4. Real-Life Stories.....	55
3.5. Benefits of Knowledge Management Systems (KMSs) and Enterprise Social Networks (ESNs).....	60
3.6. The Right ESN.....	64
4. CERN as a Social Organization.....	69
4.1. CERN.....	69
4.2. CERN Communications Strategy.....	71
4.3. Social at CERN.....	76

4.3.1. System Architecture	87
4.3.2. Deployment and Future Plans	89
5. Hands-on Social Development.....	91
5.1. Technical Student Programme.....	91
5.2. Social Mobile.....	91
5.2.1. Development	92
5.2.2. Testing.....	96
5.2.3. Problems Encountered and Limitations	97
5.3. Resource Planning Tool (RPT)	98
5.3.1. Development	99
5.3.2. Testing.....	108
5.3.3. Problems Encountered and Limitations	108
5.4. Social API.....	108
5.4.1. Development	111
5.4.2. Testing.....	156
5.4.3. Problems Encountered and Limitations	157
6. Conclusions	159
Appendix A.....	163
Appendix B.....	169
Appendix C.....	199
Works Cited.....	253

Sommario

I social network sono comunemente visti come una tendenza globale, che consente agli utenti di trovare altri con interessi simili, scrivere commenti, rispondere, esprimere apprezzamenti verso o condividere un contenuto, creare gruppi e organizzare eventi. Detto questo, c'è molto altro che può essere fatto per esprimere il vero potenziale dei social media. Al fine di migliorare il business, provvedendo a dare al personale, ai clienti e ai partner i migliori strumenti per cooperare e trarre valore da tutta la comunità, molte organizzazioni stanno prendendo l'iniziativa, creando gli Enterprise Social Networks. Un'attenta analisi dei casi di studio e delle statistiche mostra perché è importante perseguire questa strada. Al CERN, l'Organizzazione Europea per la Ricerca Nucleare, dove il numero di impiegati, studenti e volontari che ogni giorno cooperano sia in loco che attraverso la rete raggiunge le migliaia, è stato sviluppato un nuovo tipo di piattaforma, in grado di sfruttare la conoscenza collettiva del personale. La tesi descriverà il caso di studio del CERN per capire non solo perché è essenziale diventare un'organizzazione di tipo "social" ma anche come un ambiente simile può essere sviluppato. Negli ultimi capitoli verrà esaminato il mio contributo alla piattaforma, considerando il design per i dispositivi mobile, realizzato per far sì che l'ambiente si adatti a qualunque dimensione di schermo, uno strumento di gestione delle risorse integrato, che fornisce agli scienziati un mezzo per gestire facilmente il lavoro di tutti i giorni sugli acceleratori di particelle, e l'Application Programming Interface della piattaforma, che consente a chiunque abbia le credenziali di includere il contenuto dell'Enterprise Social Network all'interno di un sito web personale o di dipartimento, dando a tutti un modo ancora più semplice per partecipare.

Abstract

Social networks are commonly seen as a global trend that allows users to search and contact others with similar interests, write a post, reply, like or share content, create groups and organize events. This said, there is much more that can be done to exploit the full potential of social media. In order to improve the business, providing employees, customers and partners the best tools to cooperate and gain value from the whole community, many organizations are taking the matter in their own hands, using Enterprise Social Networks. Close analysis of case studies and comprehensive statistics shows why it is important to pursue this path. At CERN, the European Organization for Nuclear Research, where the number of employees, students and volunteers that everyday work in partnership both on site and through the network reaches the thousands, a new kind of platform has been deployed, able to exploit the social knowledge of the personnel. The thesis will describe the case study of CERN to understand not only why it is essential to become a social organization but also how a social environment can be developed. The last chapters will focus on examining my work on the platform, considering a mobile responsive design, realized to make the environment adapt to any screen size, an integrated resource planning tool, which gives the scientists the mean to easily manage the everyday work on the particle accelerators, and the platform's Application Programming Interface, which allows anyone with the right credentials to include content from the enterprise social network into a personal or departmental webpage, giving everyone an even easier way to participate.

Introduction

At CERN, the European Organization for Nuclear Research, the number of employees, students and volunteers that everyday work in partnership both on site and through the network reaches the thousands. In order to guarantee an efficient and effective communication between employees, companies and partners a variety of tools has been developed able to offer all the needed functionalities for the exchange of messages, documents and to have real-time conversations.

Despite these technologies very well do their jobs their number has become excessive and their functionalities, often similar, have led to an excessive amount of unnecessary communication in the workplace. We refer, as an example, to the increase of redundant messages not originally meant for a specific recipient but sent to groups of possibly interested people. In this case, the message might be useful to a colleague, but not necessarily to the whole team. In addition, we have to consider situations like the ones caused by the “reply all syndrome”, which happens when a number of people start a conversation over emails using the reply all broadcasting option, thus sending all the replies to everyone in the email distribution list. In this case, the number of messages increases exponentially. To address these and other problems a new kind of communication platform has to be deployed. The technology required has to make the users able to choose the communication channel to listen to and, at the same time, preserve and exploit the social knowledge of the team, making it easy to share the know-how, ask questions and reply to anyone in the network.

The goal of the thesis is to prove that a more suitable and effective approach is indeed possible, which can make an organization achieve better results in less time. Enterprise Social Networks (ESNs) provide the infrastructure needed to support the exchange of knowledge between employees, customers and partners to cooperate and gain value from the whole community. They give users a proper environment to cooperate and face every challenge together.

The 2015 McKinsey Global Survey on ESNs states:

“Where social tools are used, respondents say processes have changed notably as a result — particularly for developing customer insights and competitive intelligence, where 62 percent of respondents say the use of social technologies has significantly

changed the work flow. Executives also report that in the processes where social tools are used most often, tools tend to be integrated more deeply into day-to-day tasks — suggesting that companies must adjust the way they work to get the full value from these technologies. [...] At fully networked organizations — the companies seeing the greatest benefits from internal and external use of social technologies — executives report greater-than-average use of these tools in each process.” When asked about the future of social media, they state, “In the coming years, nearly all executives believe that social technologies could affect some key changes in structural and management processes. Their visions of social’s potential diverge, though, depending on the benefits their companies currently see. At internally networked organizations, executives believe the use of social could democratize decision-making. Fifty-one percent cite data-driven decisions as a likely change at organizations without constraints (compared with 33 percent of the total average), and 24 percent cite the use of internal markets and voting mechanisms to allocate resources (compared with 16 percent). At fully networked organizations, executives most often predict the organization’s formal hierarchy would become flatter or disappear altogether.” (McKinsey, 2015)

In order to properly understand and explore the topic, the thesis is organized as follows:

- The first chapter introduces the technology in all its many facets. It talks about the origins of social networks and their peculiarities, providing an explanation on what made them become a reality and how they work. It continues speaking about the differences between classic and enterprise social networks, giving an overview on their most important characteristics;
- The second chapter talks about the concepts of knowledge, how it can be managed and shared, explaining the barriers to knowledge sharing and what can be done to prevent those complications;
- Chapter 3 introduces the concept of social business. It discusses about the benefits of Knowledge Management Systems (KMSs) and ESNs, providing statistics and real-life examples to support the theory;
- The fourth chapter presents the CERN organization. It talks about its Communications Strategy and the reasons that brought to the creation of its

ESN, Social. It explains the system architecture and the planned improvements for the near future;

- Chapter 5 provides a detailed description of the work I have done on Social while taking part on its creation, with a thorough explanation of the code. It includes the development of its mobile design, an integrated Resource Planning Tool (RPT) and its Application Programming Interface (API), together with a clear description of its limitations in order to provide an objective view on the possibilities of the platform.

1. Web 2.0

Originally, the Web has been conceived as a way to visualize static documents linked through hypertext links built with the HTML programming language. It has been defined as a way of accessing information over the medium of the Internet. It is an information-sharing model that is built on top of the Internet. This approach is the so-called Web 1.0, pertaining the static web paradigm. From the data exchange point of view, it is a unilateral communication. The client requests a webpage and the server sends it back to the user, which will be able to read all its content and ask for another webpage through a hypertext link or using a different URL, but nothing more. With the advent of the Web 2.0 this has changed. The possibilities for the client are plenty now. The term "Web 2.0" usually refers to an evolutionary phase of the Internet and, in particular, of the World Wide Web. O'Reilly defines the Web 2.0 as "the commercial revolution in the IT (Information Technologies) sector due to the use of the Internet as a platform and the attempt to understand the rules of success on this new platform. The main principle consists in building networking applications that improve while operating. The more they are used the more they improve" (O'Reilly & Battelle, 2009). This phase has brought to the appearance of all those online applications that allow a strong website-user interaction (like Wikipedia, Facebook or YouTube).

The term makes its first appearance at the end of 2001, following the *dot-com bubble* burst. The term "dot-com bubble" identifies a phenomenon of the *new economy*, which in turn is the result of the transition from a manufacturing-based economy to a service-based one. The dot-com bubble, developed in the late twentieth century, comprehends many companies developed exploiting the surplus of funds generated from the venture capitals bound to the optimism that ruled the stock market in that period. The companies that managed to survive the end of the dot-com era are today's leading actors of the Web 2.0. Companies like Skype and YouTube.

The factors that facilitated the advent of the Web 2.0 phenomenon are many. We can say that the most important ones are the maturity and the level of development of the Internet and the realization that billions of people have now access to mobile devices

and technologies like Wi-Fi networks that make it as easy as possible for everyone to surf the web and participate.

A great example is Google Search, one of the most famous search engines that ranks all the data on the Internet. This service is strongly influenced by the number of accesses from the clients and increases its effectiveness and the quality of its results with its use. In fact, the more statistics are collected the higher will be the reliability of the data provided. The Web 2.0, in fact, is not a specific application or a particular brand, but it has to be considered as a group of approaches used to exploit the network in a new and innovative way. With the 2.0 version, the web becomes a development platform. For the companies the web is a business platform. For the marketers it is a communication platform. For the journalists it is a new media platform. For the technicians it is a new development platform and so on (McManus, 2005).

An important characteristic that defines the concept of Web 2.0 is represented by the active participation of the clients. Before its advent, in both the web and the real world the assets management was most of the times controlled by sector experts that collected and organized the data. Now the user can participate and become an active part that gives added value to the content. Another difference between the Web 1.0 and the 2.0 is the shift from the personal websites to blogs. A change that has simplified a lot the web for its users. If before it was necessary to understand and know how to write the Hypertext Markup Language (HTML) code for the pages, now anybody can publish his/her own material and give it a pretty design nonetheless. All of that without ever possessing any technical skill.

These new technologies allow the information to become independent from the person or the site that created them. It becomes possible for the user to mix and update the data collected for a particular purpose, contributing to the enrichment of those data.

The Web 2.0 is open source, meaning that it is a free source of information that allows to easily share knowledge and spread it, creating, at the same time, new job opportunities.

An important example of user participation is Wikipedia, a free encyclopaedia built collaboratively, where every user can update the information stored at any given time. It can be defined as “an open-architecture institute possible thanks to the lowering of the barriers for the publication of new content, the way in which the clients can connect

their ideas and the bandwidth available with the upgrading of the computers and the network” (Weinberger, 2007).

In an environment like this, we see people on the network become not just passive users but active and responsive elements of the web. This brings us knowing that the active participation of the users is a great example of democracy. Web services are services offered by an electronic device to another electronic device, communicating with each other via the World Wide Web for the purpose of exchanging data. They are a central node of the Web 2.0 that leaves out the concept of specific application. One must not consider the Web 2.0 as a well-defined application nor as a specific service. It can be thought as a group of websites, applications and resources that work together and are easily accessible for the client.

The transition from the software as a product to the software as a service implies an ongoing daily management, which, if omitted, can cause the termination of the software.

The most known applications in the Web 2.0 can be classified as follows:

<p style="text-align: center;">Level 3</p>	<p>It is the highest application level, which only exists while connected to the Internet and improves exploiting human connections. The more it is used the better it works.</p> <p>Some examples are eBay, Skype and AdSense.</p>
<p style="text-align: center;">Level 2</p>	<p>At this level we have applications that can work off-line but gain many advantages when on-line.</p> <p>We can think of examples like Flickr, which benefits from the sharing of photos and videos and from the tags used to identify the content.</p>

Level 1	<p>Level 1 applications can work off-line, but have technical characteristics that only work while connected.</p> <p>Examples comprehend Google Docs, which can synchronize the files modified while off-line only when connected, or iTunes with its music store.</p>
Level 0	<p>This level's applications can work without a live connection.</p> <p>Examples include Google Maps and Yahoo! Local, mapping applications that gain value with the contributions of the users.</p>

As stated before, up until the Web 2.0 the duty to collect and organize the content, both in the web and in the *Knowledge Management* (KM) environments was considered a job for the experts, while the final user could only read it while playing a passive role. The great news introduced with the Web 2.0 is the possibility given to the user to actively participate in the management and sharing of knowledge. This active involvement creates an added value to the information on the web, thanks to the provision of new ideas and new experiences.

Now that the users are able and willing to create value, both actively and passively, the enterprises create new systems to aggregate users' data that will later be used to build value as a collateral effect of the normal usage of the application.

With the advent of the Web 2.0 we witness a radical change in the classification systems. From the classic taxonomy (or science of classification), enforced from web programmers through the use of directories, to the new concept of *folksonomy*, a classification made from the users via keywords called "tag". These tags are chosen not from a list but created as needed. The tags are then associated to the information shared. They make the classification and the research of content possible and they are usually chosen according to personal criteria.

This application is particularly developed in the "social bookmarking", virtual bookmarks freely available and shared with the other members of the web community.

“Tagging”, the act of linking a tag to some piece of information, allows everyone to look for information through tags and obtain the lists of related topics that have been labelled with the same tag.

The web has now an architecture that allows everyone to gain from it. The users exploit the network for their own personal gain but, at the same time, contribute to the whole community adding value to the contents. When a user adds new material or new web pages, these are integrated in the web structure so that the other users will immediately be able to discover it and contribute to its development.

Again, Wikipedia is a great example of the Web 2.0 new features. A free encyclopaedia where each element can be written from a user and modified from another one at any given time, built in the hope that every user would contribute with reliable information. This “experiment” of trust in the final user allowed Wikipedia to enter in the top 10 of the most popular websites, representing a profound change in the way content is created.

The key to succeed in the Web 2.0 market is, thus, to master its collective intelligence using the contributions of the users and their interactions. Many companies already do this and this way they also manage to save on advertisements, using *viral marketing*, that is the word of mouth of the digital era, shared online for everyone to hear.

Summing up, the principles of the Web 2.0 are:

- The Web as a platform
- Services development
- Active participation of the users
- Improvement of the service with its use
- Collective intelligence

1.1. Social Networks

In this section, we are going to introduce the concept of Social Networks. Before trying to give a definition, let us walk through the various stages that brought to their creation.

1.1.1. Origins of Social Networks

Social media has become a ubiquitous part of the daily life, but its growth and evolution has been in the works since the late 80s. From primitive days of newsgroups, listservs and the introduction of early chat rooms, social media has changed the way we communicate, gather and share information, and have given rise to a connected global society.

Let us have a brief look into the history of social networks, starting from the various applications and services that came before them.

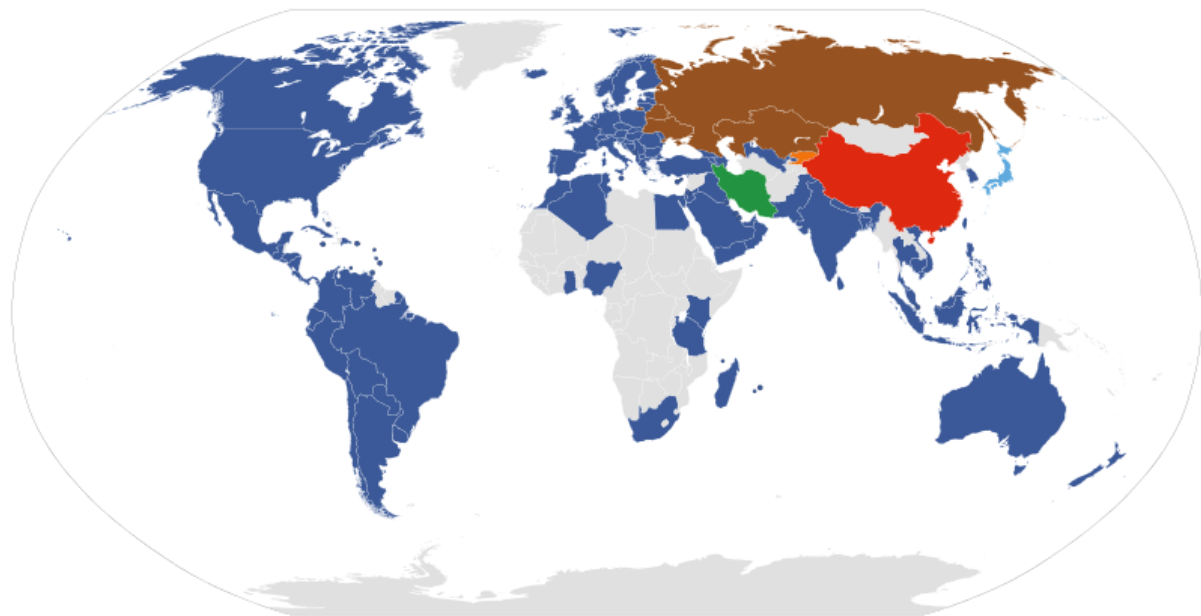
1978	<p>The first service related with Social Networks appeared on the scene in the 70s.</p> <p>“The Bulletin Board System (BBS) was the first collaborative tool available for the personal computer platform. The first BBS went up on Feb. 16, 1978 in the suburban Chicago home of Walt Christensen. (Carlson, s.d.). Once logged in, the user could perform functions such as uploading and downloading software and data, reading news and bulletins, and exchange messages with other users.</p>
1980	<p>Usenet is an internet service consisting of thousands of newsgroups. Established in 1980, it is one of the oldest forms of computer network communications still actively used today. Users can post to newsgroups and access articles from years ago.</p>
1980	<p>CompuServe broke new ground in 1980 as the first online service to offer real-time chat online with its CB Simulator, where CB stands for “citizens band radio”, often abbreviated as CB radio. By 1982, the company had formed its Network Services Division to provide wide-area networking capabilities to corporate clients. (CompuServe, 2016)</p>
1984	<p>Prodigy Communications Corporation (Prodigy Services Corp., Prodigy Services Co., Trintex) was an online service that offered its subscribers access to a broad range of networked services, including news, weather, shopping, bulletin boards, games, polls, expert columns, banking, stocks, travel, and a variety of other features.</p>

1988	<p>Internet Relay Chat Protocol (IRCP) is an application layer protocol that facilitates communication in the form of text.</p> <p>The chat process works on a client/server networking model. IRC clients are computer programs that users can install on their system. These clients communicate with chat servers to transfer messages to other clients (Oikarinen & Reed, 1993). IRC is mainly designed for group communication in discussion forums but also allows one-on-one communication via private messages (Kalt, 2000) as well as chat and data transfer, including file sharing (Wallace, 2004) .</p>
1996	<p>ICQ is an open source instant messaging computer program that was first developed and popularized by the Israeli company Mirabilis in 1996. The name ICQ stands for "I Seek You". The ICQ client application and service were initially released in November 1996 and the client was freely available to download. Users could register an account and would be assigned a number, like a phone number, for others to be able to contact them (DeCoursy, 2001).</p>
1997	<p>In 1997 Andrew Weinreich created SixDegrees. With SixDegrees we approach the first example of a real social network service website. Named after the six degrees of separation concept it allowed users to create an account and compile lists of "friends" or family members and search for other users with similar interests.</p>
1999	<p>In one of the first attempts at social networking we also have LiveJournal. LiveJournal is a community publishing platform, wilfully blurring the lines between blogging and social networking. Since 1999 LiveJournal has been home to a wide array of creative individuals looking to share common interests, meet new friends, and express themselves.</p> <p>LiveJournal encourages communal interaction and personal expression by offering a user-friendly interface and a deeply customizable journal. The service's individuality stems from the way highly dedicated users utilize the tools, along with the instinct for individual expression, to create new venues for online socializing (LiveJournal, 2016).</p>
2001	<p>Wikipedia is the first example of a successful social media that enabled the users to actively collaborate toward a common goal.</p>

	<p>“Wikipedia is a multilingual, web-based, free-content encyclopaedia project supported by the Wikimedia Foundation and based on a model of openly editable content.” (Wikipedia, s.d.)</p>
2003	<p>MySpace is a social networking website offering an interactive, user-submitted network of friends, personal profiles, blogs, groups, photos, music, and videos.</p> <p>The novelty here is the inclusion of music and videos in the network. Artists can upload their songs onto Myspace and have access to millions of people on a daily basis. Many artists became famous thanks to this. As a result, MySpace had a significant influence on pop culture and music.</p>
2004	<p>Facebook is the most known social network today. It is a for-profit corporation and online social networking service based in Menlo Park, California, United States.</p> <p>After registering on the site, users can create a user profile, add other users as "friends", exchange messages, post status updates and photos, share videos, use various applications (apps), and receive notifications when others update their profiles. Additionally, users may join common-interest user groups organized by workplace, school, or other topics, and categorize their friends into lists such as "People From Work" or "Close Friends". In groups, editors can pin posts to top. Additionally, users can complain about or block unpleasant people.</p>

Since 2004, many more have spawn to populate the web, so we can see that the efforts toward the creation of the perfect platform are not over yet.

Here we show the world map with the most popular networking sites by country:



- Facebook
 - Twitter
 - VKontakte
 - QZone
 - Odnoklassniki
 - Facenama
 - no data
- (Alexa, 2016)

Attesting to the rapid increase in social networking sites' popularity, by 2005 it was reported that Myspace was getting more page views than Google.

We can see from the timeline that the advent of social networks was neither a coincidence nor a single brilliant idea. Instead, we read that over the years many teams contributed incrementally to the development of what could have been the best way to make people connect and share information over the Internet.

Adapting to the new technologies and the new trends, companies from all over the world try to create innovative platforms able to reach out to customers willing to expand their horizons and discover new ways to express themselves.

1.1.2. What is a Social Network

In the previous paragraph, we have cited many social media trying to explain what kind of service each of them provides or provided to its users. In order to best express ourselves, be objective and ultimately give the most comprehensible idea of what exactly is a social network we will propose now some definitions given on this phenomenon in recent years.

“A social networking service (also social networking site, SNS or social media) is a platform to build social networks or social relations among people who share similar personal and career interests, activities, backgrounds or real-life connections.” (Buettner, 2016)

Social media is “a group of internet-based applications that build on the ideological and technological foundations of Web 2.0, and that allow the creation and exchange of UGC (User-Generated Content)”. (Kaplan & Haenlein, 2010)

“Internet-based software and interfaces that allow individuals to interact with one another, exchanging details about their lives such as biographical data, professional information, personal photos and up-to-the-minute thoughts.” (Investopedia, n.d.)

“Social Media is a new set of tools, new technology that allows us to more efficiently connect and build relationships with our customers and prospects. It is doing what the telephone, direct mail, print advertising, radio, television and billboards did for us up until now. But social media is exponentially more effective.” (Safko, 2012)

“A Social Network is a dedicated website or other application which enables users to communicate with each other by posting information, comments, messages, images, etc.” (Oxford Dictionaries, n.d.)

As we can see from the few definitions given up until now there are many opinions on what features a social media should or should not have. A review of the literature on

the topic can give us a more comprehensive view on the matter and sum up the main characteristics that define a social media as such:

Social media are computer-mediated tools that allow people, companies and other organizations to create, share, or exchange information, career interests, ideas, and pictures/videos in virtual communities and networks. The variety of stand-alone and built-in social media services currently available introduces challenges of definition; however, there are some common features:

1. social media are Web 2.0 Internet-based applications,
2. UGC (User-Generated Content) such as text, digital photo or digital video posts are the lifeblood of the social media organism,
3. users create their own profiles for the website or app, which is designed and maintained by the social media organization, and
4. social media facilitate the development of online social networks by connecting a user's profile with those of other individuals and/or groups.
5. Social media depend on mobile and web-based technologies to create highly interactive platforms through which individuals and communities share, co-create, discuss, and modify user-generated content.
6. They introduce substantial and pervasive changes to communication between businesses, organizations, communities, and individuals.

(Buettner, 2016) (Obar & Wildman, 2015) (Kaplan & Haenlein, 2010) (Ellison, 2007) (Kietzmann & Hermkens, 2011)

As explained, there are many social networks available on the web. Between them, there are some of everyone's knowledge by now, like Facebook, YouTube and Twitter, quoted as the most popular social networking sites of 2016 (eBizMBA Inc., 2016).



Mark Zuckerberg ✓

Segui
Messaggio
...

Diario
Informazioni
Amici
Foto
Altro ▾

Segui Mark per visualizzare i suoi post pubblici nella tua sezione Notizie.







72.390.750 persone seguono gli aggiornamenti

Segui

In breve

Making the world more open and connected.

- 🏠 Founder and CEO presso Facebook
- 🏠 Lavora presso Chan Zuckerberg Initiative
- 🎓 Ha studiato Informatica presso Harvard University
- 🏠 Vive a Palo Alto
- ❤️ Sposato con Priscilla Chan
- 📍 Di Dobbs Ferry
- 👤 Seguito da 72.390.750 persone

Foto





Mark Zuckerberg

17 h · Palo Alto, California, Stati Uniti d'America · 🌐

I just ran my 365th mile in my A Year of Running challenge.

This has been more fun than I expected -- mostly because a few friends are doing it with me. One friend's New Years challenge was to run one more mile than however many miles I run. I'm pretty competitive, so that'll keep me going.

It's inspiring to see so many people in our community join as well. Our Year of Running group has more than 100,000 people and every day people share amazing stories of how running helped ... Altro...

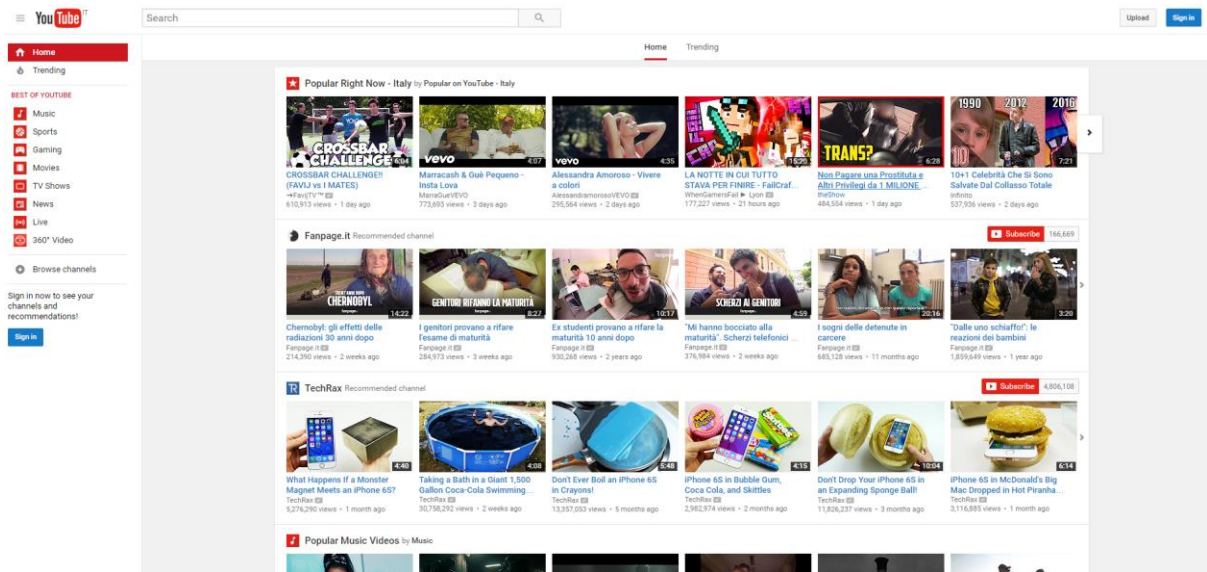


👍 Mi piace
💬 Commenta
➦ Condividi

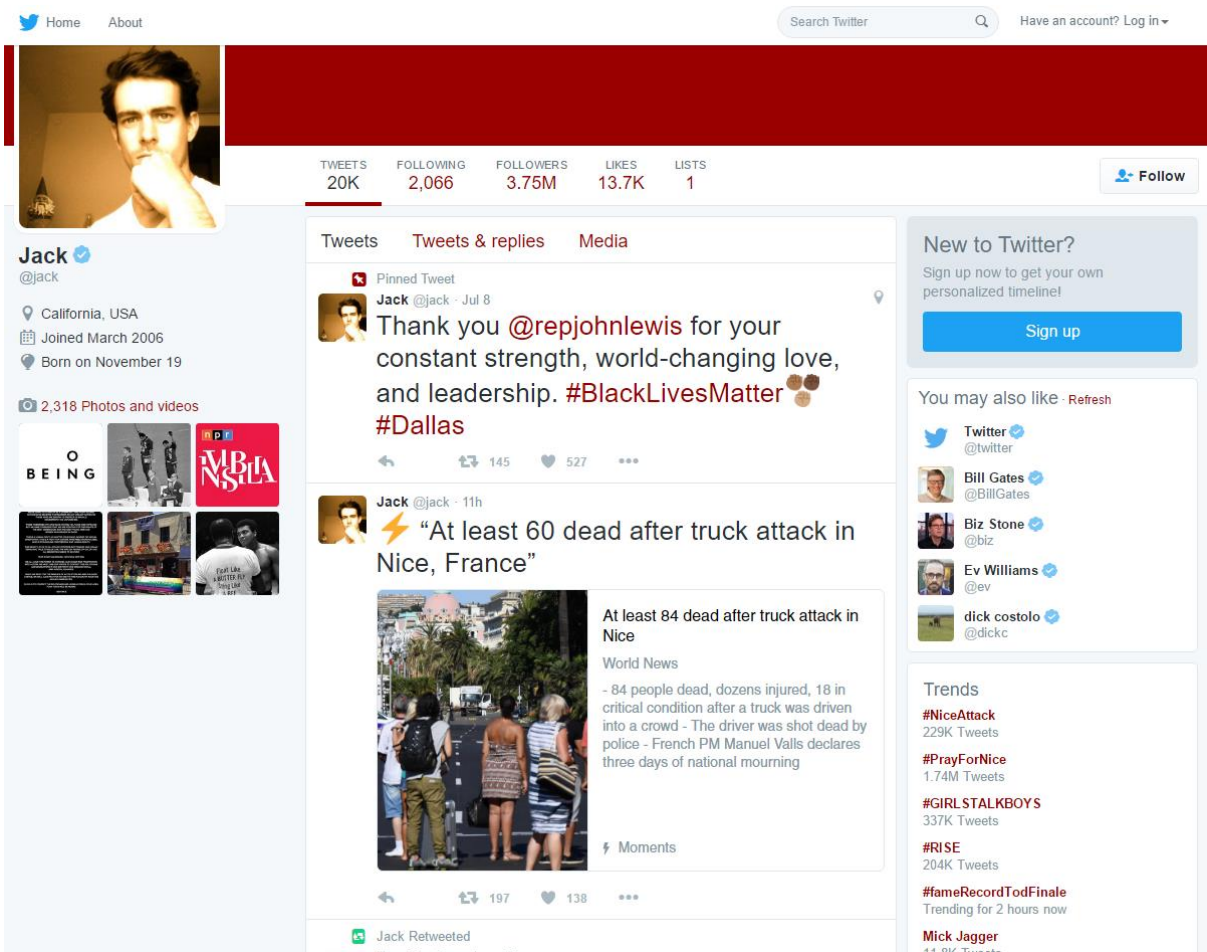
👤 Julie Zhuo e altri 195 mila

Commenti più in vista ▾

Facebook's page example (taken on July 15, 2016)



YouTube's page example (taken on July 15, 2016)



Twitter's page example (taken on July 15, 2016)

What is surprising about these platforms is that they are not only well known among teenagers but among every age range; even people of 70 years of age know and use them! This level of reach in the population makes them have an enormous amount of users on the network. Consider that Facebook claims to have 1.13 billion daily active users on average (stated on March 2016) (Facebook, 2016).

1.1.3. How does it work?

Social networks are based on the users' ability to create personal profiles. It is then possible for the users to provide some personal information regarding one's tastes and interests. Professional information is also useful in social network profiles; descriptions of specialties, areas of expertise and professional interests give opportunities to find colleagues with similar interests, experts in a particular area, etc... (Angehrn, et al., 2008).

Social presence is created not only by static information in profiles, but also by facilitating awareness about other people and their work. Awareness means an "understanding of the activities of others, which provides a context for your own activity" (Böhringer & Richter, 2009).

Once created the social profile, users are able to share text, images, sounds and videos regarding any argument they want. This is because user-generated content is the lifeblood of social media. It is through content that a user can discover other people with the same interests and create connections. These connections enable then the user to know more and more people, to the point that anyone is easy to reach. In this case, it is easy to think about the *six degrees of separation* concept, which says that everyone and everything is six or fewer steps away, by way of introduction, from any other person in the world. Concerning the social media, we know that the number of steps needed to reach someone can be reduced to one when the user is made able to search for a specific profile. In this kind of environment, it is also possible to create "groups" where a limited amount of users can talk about a specific subject and share knowledge with a selected set of people. Another service offered to the users, between the many, is the chance to create "events", where friends can be invited to join and participate in the discussions online before, during and after the occasion.

As we can see, there is no limit in the amount of services that can be offered to the users from a social media. Every time a new need is identified, a new tool can be developed in order to adjust the network to the users' needs.

Regarding the low-level implementation of a social network there is very little documentation on how exactly a social media works. What we know for sure is that many programming languages and platforms can be used to create a social network. Moreover, the differences between them are little when put in a condition to have to satisfy millions of users around the world. At that level of complexity it does not matter anymore if the language chosen is LAMP (Linux, Apache, MySQL, PHP/Python/Perl) or Ruby on Rails with Jabber/XMPP (Extensible Messaging and Presence Protocol), because as Blaine Cook, former Twitter architect, said "languages don't scale, architectures do." and at that level it is the architecture of the system that matters, with its scalability and its ability to perform well under stressful situations.

1.2. Enterprise Social Networks (ESNs)

"Ask just about anyone today about social media, and they will probably acknowledge using Facebook, knowing something about Twitter, and admit that social media are a widespread, perhaps even global, trend. Push them a bit further, and they will affirm that social media are genuinely significant somehow, but they might have a hard time pinning down exactly how or why. If you probe deeper yet and ask them if or how social media will transform the way businesses work, most people won't have a clear answer at all" (Hinchcliffe & Kim, 2012).

These doubts are entirely understandable, given how the digital world has virtually remade the means and tools of digital communication in just a few years. As the worldwide interactive marketing group-director of Coca-Cola, Michael Donnelly, said: "Business is changing right before our very eyes. We are in a world of empowered individuals with reliable, always-on, cross-media connectivity with a vivacious appetite for continuous improvement to win amongst global competition."

Operating a business through the social lens presents a profound new way of thinking. Social media can benefit an enterprise in many ways. When talking about ESNs, social

media have to be seen as a means to an end, not the end itself. Enterprise Social Networks enable “mass collaboration, in which a large and diverse group of people who may have no pre-existing connections pursues a mutual purpose that creates value” (Bradley & McDonald, 2011).

In the endless search for the customer needs and tastes, more and more companies are shifting to social networks to get the answers they need in order to be able to develop products that better reflect the preferences of their potential clients. Using analytics and business intelligence, companies can give sense to the big amount of data from the network, derive useful insight and glean value from the social media relevant to the business. Instead of thinking on passively looking at the habits and tastes of the customers, some companies bring them straight to the boardroom and let them have a voice on the details of the final product or service.

In addition, businesses now can “pick and choose new partners in an open marketplace, where business reputation and prior performance are shared and visible for all to see” (Hinchcliffe & Kim, 2012). Every time there are teams, the social approach can help. “What we are observing now is that social media have moved far beyond a means to stay in touch with old friends and colleagues. They have become how business gets done” (Hinchcliffe & Kim, 2012).

Operating a business through the social lens presents a profound new way of thinking. In the attempt to provide employees, customers and partners the best tools to cooperate and gain value from the whole community many companies are exploiting the power of social media taking the matter in their own hands. In order to keep the information shared safe and to personalize the experience given to the users the ESN came to life. A private social network built specifically with the purpose to serve the company, its employees, its partners and its clients. It is easy to imagine how the purpose in this kind of networks is not the sharing of one’s personal thoughts (like on Twitter) but to be available for everyone else involved, in case somebody would need a particular skill or piece of knowledge.

The goal of the companies here is not only to give customers a loud voice, to flatten the hierarchies making managers more approachable and to let the right person be easily reachable but to embed mass collaboration in “who” they are and “how” they work. They need to develop the right corporate skills to use this level of collaboration

again and again to deliver real business value, both inside and outside the enterprise, all along the value chain.

The main tenets of this new philosophy are:

- Anyone can participate
Nearly all aspects of business, regarding employees, partners, customers and everyone that can bring value to the community, will ultimately be open, social and participative. “In general, the more open the participation, the more superior the result.” (Hinchcliffe & Kim, 2012)
- Create shared value by default
Building value requires that, whenever possible, contributors automatically share content with the entire community in as close to real-time as possible. The reputation gained in the community by the author matters, as well as the resonance of his/her contribution with others. Individual additions of shared value may seem tiny at the moment, but when aggregated they build value exponentially.
- While participation is self-organizing, the focus is on business outcomes
Control in social businesses is ultimately embodied in those willing to participate and contribute. Instead of having a well-defined chain of command, a classic organizational hierarchy, the control processes of social businesses change dynamically according to its community.

Even if social networks and ESN use the same processes and similar tools, the goals are completely different. While classic social media goals are solely those of the individuals, in social business the purpose is specifically about productive outcomes shared from everyone involved.

We identified here few of the benefits of using a social approach. In the next chapters, we will give a broader view of the subject and we will answer some of the questions that might already arise like:

- ✓ With such seemingly uncontrolled processes how can work be done?
- ✓ How does a business maintain direction, focus, control and ownership of the results?
- ✓ How does a business define and solve problems while deriving business value from the community as a whole?

- ✓ Is it enough to provide the people with the necessary tools?
- ✓ Who is winning in social business and why?

2. Knowledge Management

“Social enterprising is based on effective knowledge management” (Keyes, 2013). It is, thus, important to spend some time explaining what can be considered as knowledge, what it means to manage knowledge and why it is so important.

2.1. What is “Knowledge”?

Every new piece of knowledge or theory is built on the background of previous knowledge.

Over half of the work in developed countries is knowledge work. In some industries, like in the finance field, more than three quarters of its workforce is dedicated to creating and managing high value information. Information that is now the heart of the world economy. As previously stated, from information comes knowledge, but what exactly can be defined as knowledge?

A number of meanings of the term *knowledge* were proposed from the ancient to the modern times. From the point of view of philosophy, knowledge can be considered as “justified true belief” (Plato), though this definition is now agreed by most analytic philosophers to be problematic because of the Gettier problems (Gettier, 1963), so a better definition is “well-justified true belief”. Another definition could be “certain understanding, as opposed to opinion” (Oxford Dictionaries, 2016). From the point of view of economic theory, knowledge is a “critical organizational resource that provides a sustainable and competitive advantage in a competitive and dynamic economy” (Davenport & Prusak, 1998). For an enterprise, “knowledge is a key strategic asset for organizations of all sizes” (Keyes, 2013).

Knowledge is defined in different ways depending on the context and purpose of the definition. Often, it is defined by distinguishing among knowledge, information and data (Alavi & Leidner, 2001).

As commonly accepted, data is referred to as raw numbers and facts, information as processed data, and knowledge as authenticated information. However, it is not obvious to know how to discern between information and knowledge.

The main aspects that help us distinguish between knowledge and information are the following:

1. Knowledge is dynamic, it is created in the social interaction between individuals and organizations and
2. Knowledge is context specific.

Without the social aspect and context, knowledge becomes close to just information (Nonaka, et al., 2000).

Knowledge can also be understood as personalized information (which may or may not be new, unique, useful, or accurate) related to facts, procedures, concepts, interpretations, ideas, observations, and judgments (Alavi & Leidner, 2001) (Nonaka, et al., 2000).

There are two kinds of knowledge when talking about social business: explicit and tacit. Explicit knowledge can be expressed in a formal language and can be shared in the form of data, scientific formulae, specifications, manuals and the like. It can be processed, transmitted and stored relatively easily. In contrast, tacit knowledge is highly personal and hard to formalize. It represents know-how and intuitive knowledge which is rooted to context, experience, practice and values. Subjective insights, intuitions and hunches fall into this category of knowledge. This is the kind of knowledge that can lead innovation and breakthroughs (Frost, 2013) (Nonaka, et al., 2000).

2.2. The Knowledge Worker

Knowledge workers are employees whose job is to “think for a living”, to use knowledge in order to solve a problem through creative thinking. Knowledge workers are considered to be the intellectual capital of a company and a key factor in its sustainable development. Managers must value the knowledge obtained by the employees and do all that is necessary to exploit it as much as possible.

For a knowledge worker working in a team, one of the problems that could slow down and mine the efficacy and the efficiency of the team is having different team members at different times during the development of the solution for a client. When this happens, there are three different kind of problems to consider:

1. Loss of knowledge. The longer someone stays on a team, the more knowledge he or she acquires about the project, the problem domain and the stakeholders. Losing a team member means losing all of his or her experience and knowledge. When dealing with this kind of problem the most common approach is to make sure that the team has at least some stable team members.
2. Thinking differently. When a group of people work for a long time together, they get to know each other and start to develop work patterns that make the team efficient. When a new team member arrives, the team might need some time to adapt. It is clear now that the commonly used solution to deal with the “loss of knowledge” problem is not enough. Using a knowledge management system and social networking technologies (e.g. knowledge bases, wikis, blogs, social networking groups etc.) would greatly accelerate a new team member’s trip along the learning curve.
3. Low commitment. When a worker knows that you are going to work for a short amount of time on a team, it might be difficult for him or her to share the enthusiasm and be totally committed to the team. Motivation is the key to solve this problem. Reward systems are an effective method to overcome low commitment. Both positive and negative rewards can be considered. While positive reward systems (e.g. public validations, days off, bonus pay etc.) are the most commonly used, negative reinforcement approaches should be considered as well, since they are the best way for an employee to understand in little time where he or she is doing something wrong and why.

Researchers see a strong, on-going linkage between knowledge workers and innovation, but the pace and manner of interaction have become more advanced (Tapscott & Williams, 2006). The many social media tools can drive more powerful forms of collaboration. Knowledge workers now engage in “peer-to-peer” knowledge sharing across company boundaries, forming networks of expertise. Some of these networks are even open to the public. While they share their concerns over copyright and intellectual property law being challenged in the marketplace, they

feel strongly that businesses must engage in collaboration to survive. They highlight the on-going alliance of public (government) and private (commercial) teams to solve problems, referencing the open source Linux operating system along with the Human Genome Project as examples where knowledge is being freely exchanged, with commercial value being realized.

Many researched knowledge workers' productivity and work patterns. Part of the research has involved the analysis of how, on average, knowledge workers spend their day. It has been noted that effective and efficient knowledge work relies on the smooth navigation of unstructured processes and the elaboration of custom and one-off procedures. "As we move to the 21st century business model, the focus must be on equipping knowledge workers with tools and infrastructure that enable communication and information sharing, such as networking, email, content management and increasingly, social media." (Palmer, et al., 2014).

In the next section we will talk more about knowledge sharing (or information sharing), analysing what exactly means "sharing" in this case and what kind of problems could arise in the process.

2.3. Knowledge Management and Knowledge Sharing

"Knowledge management can be defined as the processes which support knowledge collection, sharing, and dissemination. The expectations for knowledge management are that it should be able to improve growth and innovation, productivity and efficiency reflected in cost savings, customer relationships, decision making, innovation, corporate agility, rapid development of new product lines, employee learning, satisfaction and retention, and management decision. Interestingly, these are the same expectations for social enterprising." (Keyes, 2013)

It has been stated that the importance of Knowledge Management (KM) is no longer restricted to knowledge intensive firms in the high-tech industries but to all sectors of the economy (Teng & Song, 2011). Even companies in the traditional industries, such

as cement, can benefit greatly from KM (Zack, 2003). In essence KM is beneficial to all sectors, be it educational, banking, telecommunications, manufacturing or even the public sectors.

The management of knowledge has generated considerable interest in business and management circles due to its ability to deliver to organisations strategic results relating to profitability, competitiveness and capabilities enhancement. To state it more clearly “Organisations that effectively manage and transfer their knowledge are more innovative and perform better” (Riege, 2007). Successful organisations now understand why they must manage knowledge, develop plans to accomplish this objective and devote time and energies to these efforts.

Once understood the importance of having an effective knowledge flow between employees, teams and departments in the company, it is central to start talking in more detail about sharing the new knowledge produced during the everyday working hours. Limiting the efforts or not giving the proper consideration to knowledge sharing can have substantial effects on the organization. “It is estimated that an organization with 1,000 workers might easily incur a cost of more than \$6 million per year in lost productivity when employees fail to find existing knowledge and recreate knowledge that was available but could not be located. On average, 6% of revenue, as a percentage of budget, is lost from failure to exploit available knowledge.” (Keyes, 2013)

Even if it could seem useless at the moment of its definition, sharing and collecting knowledge are the activities that best carry out the interests of the company, its employees, its customers and its partners.

Looking at the definition of knowledge given by Davenport & Prusak as a critical organizational resource one easily realizes that this resource, as any other, has to be managed in the most optimal way to gain advantage of it. So what is the best way to create, retain and exploit knowledge in a company? Simply put, sharing it.

“Knowledge sharing is the means through which employees can contribute to knowledge creation, use and innovation to the competitive advantage of the organization. Knowledge sharing refers to the provision of information and know-how

to help others and to collaborate with others to solve problems, develop new ideas, or implement policies or procedures.” (Wang & Noe, 2009).

It has to be considered not only as the sharing of knowledge between individuals but also between teams, organizational units and organizations. In general, knowledge management is aimed at identifying and leveraging the collective and personal knowledge, know-how, experiences and judgments inside and outside organizations to bring additional value to organizations and help them compete. (Quaddus & Xu, 2012). In this definition, we can read the focus of knowledge sharing on both tacit and explicit knowledge. This is why most managerial practices and efforts are devoted to facilitating sharing of both types of knowledge.

The goal of knowledge sharing is to share the existing knowledge not just for future uses but also to create new knowledge more quickly. It regards a profound new way of thinking that requires openness and trust at first and that will lead to fast and effective improvements in the everyday work.

2.4. Barriers to Knowledge Sharing

Technology itself does not make organisations share knowledge but, if people are willing to share it, technology can increase the reach and scope of such exchanges. Developing a KM system in place is not going to make people utilise it, but the success of KM initiatives involves taking into account the socio-cultural factors that may inhibit people to willingly share their knowledge, such as:

- lack of trust,
- lack of time or
- fear of being judged
- concerns about loss of power/status.

Lack of trust between employees is a well-known issue that endangers relations and thus the efficiency of the whole company. In the case of knowledge sharing it makes the member of the team avoid sharing his or her knowledge because there is no trust in how that knowledge will be used or by who. To increase the level of trust between

workers it is useful to first have more information available about colleagues (Dignum & Eijk, 2005) and then have closer and more frequent communications (Cheng, et al., 2008). In this matter social networking tools help, providing more information about members of the company and facilities for communication between co-workers (Boeije, et al., 2009).

Lack of time is a common problem between workers that already spend a lot of time on the projects they are working on and feel sharing knowledge not wrong but simply as a waste of time, as if he or she is not paid to do so. In this case, it is necessary to make it clear to everyone, even writing it on the contracts if necessary, that part of the job is to define the notions learned and make them available for everyone to see. This way everyone will take some time at the end of a project and complete the task feeling rewarded from it.

The fear of being judged is usually felt by newcomers and, more in general, people that are not very familiar with the subject they are going to talk about while sharing their knowledge. It is completely normal to feel the pressure of the opinions of the co-workers, to consider the possibility to be mocked or ridiculed. Everyone can make mistakes. What is important is to learn from them. It is the company's duty to clarify that a behaviour that tries to diminish a colleague's image will not be tolerated.

Loss of power or status are the concerns of the senior members of an enterprise. The most experienced and knowledgeable elements of the company. This kind of employees are the ones that should best embrace the sharing attitude, since their contribution would greatly benefit the new members of the organization, making it easier for them to catch up and be ready to help when needed. This said, it is not easy for a person in a position of power to decide to share what made them reach that specific position. They may fear that sharing their knowledge would make them lose their job easily and the younger members of the company overtake them. In such a case, it is advisable to create an environment where it is clear that such an outcome is improbable and that the organization values very much the employee and its contribution to the global knowledge of the company. More specifically, such an outcome is possible only when the perceived benefits (some extrinsic motivation solutions, such as bonuses, presents, etc) are higher than the perceived costs of

sharing (e.g. time and efforts for contributing knowledge). One of the immediate ways to reduce perceived costs is to make it easier for people who share their knowledge to do this, also with the help of IT, as well as make knowledge sharing secure from the point of view of the loss of jobs or advantages (Cabrera & Cabrera, 2003).

To sum up, in order to deal with the issues stated above the main directions of efforts are the following:

- a) making knowledge visible and showing the role of knowledge in organizations,
- b) building a knowledge infrastructure, not only technical system, but also connections among people given space, time, tools and encouragement to interact and collaborate,
- c) developing knowledge-intensive and knowledge sharing culture in order to free employees from fear of losing their advantages when sharing their unique knowledge,
- d) be liberated from the fear of losing important intellectual assets, if valued colleagues leave the firm. (Yang & Chen, 2007)

The last point of the list explains in short one the most important features of knowledge sharing: the possibility for a work team to preserve the knowledge of one of its members if he or she leaves the team or the company. This way it is easier for the team to recover from the loss and learn what they should in order to undertake the work of the colleague.

In order to reach the goals set for knowledge management and realize an effective knowledge sharing culture the company has to provide its employees the right tools, education on how to use them and motivation to make it happen. Realizing the right tools requires the chiefs of the departments involved to meet and make a list of all the features that a proper tool must have in order to be both useful and easy to learn. The IT department will be of help in this matter, yet the real problem is not the realization of the tools, but giving the employees the motivation they need to use them. As we read in (Argote, et al., 2003): "Organizational settings in the field of knowledge management can impact an individual's ability to create, retain and share knowledge, as well as provide motives and opportunities or tools to do this".

It is clear by now that knowledge sharing is not necessarily synonymous with pro-social behaviour. Indeed, knowledge sharing may involve significant effort or sacrifice. For the most part knowledge-sharing barriers can be categorized into three dimensions:

- a. individual,
- b. organizational and
- c. technological.

A study of 1,180 staff members in the regional transport union of Palm Beach (Florida) determined that its culture was not conducive to knowledge sharing for a variety of reasons, including:

- absence of support systems,
- lack of training,
- lack of job security,
- lack of organizational culture,
- employee competition, and
- lack of recognition.

At the first element of the list, we find a huge technological problem. Having a good technical support is key to make it easier for everyone to share one's expertise and later create new knowledge starting from the one collected so far.

The second issue in the list could be resolved exploiting technological tools, yet the decision to make this kind of effort has to come from the organization itself. The management have to evaluate the need for a proper introduction to the sharing mechanisms for the employees. The third and fourth elements still are a responsibility of the company, that must focus on creating a culture that considers the needs of the employees while clarifying how useful for everyone is to work not as individuals or small teams but as a community that involves all of the people at the company, the customers and the enterprise's partners.

Competition is usually good for business. It pushes everyone to "go the extra mile" to become the best in what they do and be seen as such. As explained before, however, this kind of behaviour can lead to difficulties in sharing one's hardly achieved expertise. It is necessary to do what it takes to make the employees certain that they will not lose their advantage when sharing their unique knowledge.

The last point is an organizational problem as it is the company that has to recognize the efforts of its workers. The solution for this problem can be found in a technical approach that exploits social media tools to make it easy for both the worker and the management to recognize when someone is making a significant effort in following the sharing culture. This is possible through instruments that make it clear to everyone who is the author of the shared piece of knowledge and who are the people that liked or shared it. When number of “likes” or “share” is reached the software given the employee a virtual medal, visible on his or her profile, which symbolizes the achievement. It has been studied that this kind of tools greatly increases the willingness of the stakeholders to participate in the community.

Beyond the effects of the introduction of technological tools, it has been proven that there is also a relationship between group compatibility and knowledge sharing. The more compatible a person is with the group in terms of age, gender, and other factors, the more likely he or she is to practice knowledge sharing. Conversely, individuals who perceive themselves in a minority (e.g. gender, marital status, education, etc...) are less likely to participate in knowledge sharing. Of particular note is the finding that women participants require a more positive social interaction culture before they perceive a knowledge-sharing culture as positive. The list of compatibility variables includes more than just the obvious traits of age, gender, ethnicity, and educational level. Personality differences, communication skills, and individual values also factored into the equation. (Keyes, 2013)

Another study, from (Wang & Noe, 2009), has shown instead that socially isolated members or sub-groups are more likely to disagree with others and so contribute their unique knowledge within a heterogeneous team.

We now know that people that perceive themselves as a minority in the team are less likely to share their knowledge, but when everyone is a minority and the group is truly heterogeneous then competition arises and knowledge sharing happens. In this case, within a functionally diversified team, the acknowledgement of team members' expertise also helps increase participation in knowledge sharing (Thomas-Hunt, et al., 2003). This leads us to the understanding that, in order to obtain the knowledge sharing pattern required to help group decision-making processes in the organization,

work teams should either be formed of people with similar status, education, gender, age, skills and values or formed of a very heterogeneous set of team members.

In order to minimize the risks and increase the chances of success there are other steps that can be followed. The research has proven that a less centralized organizational structure can help facilitate the knowledge flow, as well as the open-space working environment (Yang & Chen, 2007). Another research suggests that the organizations should actively create opportunities for employee interactions to occur and employees' rank, position in the organizational hierarchy and seniority should be deemphasized to facilitate knowledge sharing (Argote, et al., 2003).

We will later talk about the fact that many, if not all, of the guidelines collected can be followed and made a reality in the organization using an ESN.

3. Transition to the Social Organization

The term Enterprise 2.0, as Social Organization, Social Business or Conversation Company, means introduction of the Web 2.0 infrastructure and relative tools by organizations (Levy, 2009). To explain the topic we start from the paper by A. P. McAfee (McAfee, 2006) in which he claims that the conventional systems for knowledge management are not enough or not suitable enough for successful knowledge sharing and knowledge creation process. He states that the “newly emerged technologies, such as blogs, wikis, instant messengers, social network tools, and folksonomies may be more effective for knowledge management tasks”, and calls a set of these technologies Enterprise 2.0.

3.1. Introducing the Social Business

Collaboration is the strategic factor required to compete in the global market. Knowledge sharing allows the stakeholders to exchange ideas and, by working together, obtain results that could not be achieved by working autonomously. The Internet has become a fundamental platform to connect people and organizations. Through the company’s work teams, it is possible to lower the costs and time to market (TTM) of a product or service thanks to the new vision of the world: a wider and connected environment, where personnel from multiple sectors can work on the same subject together. This approach allows the parts to develop ideas and projects quickly, managing to tap into the knowledge of a vast amount of people.

The challenge for the enterprises now is to get to the market quickly, having products and services that well respond to the needs of the customers. It is important to remember here that consumers nowadays are evermore informed, thanks to the web, and demanding. The network offers the companies the chance to renew their business models, not to modify what is done but how it is done. The Web 2.0 offers decentralized offices the chance to collaborate through the network, creating virtual work teams. The collaboration concept is shaping the way companies get the work

done, coming from a vertical integration system and a hierarchical company model to a system that focuses on coordination and cooperation. All of this is possible thanks to the Web 2.0 technologies, which allowed a significant reduction of the costs of coordination between enterprises. This new organizational concept brings many companies to collaborate and form a corporation able to better exploit the economies of scale and quickly answer to the demands of the market. Through cooperation, the corporation can reduce costs and increase its ability to innovate, making each of the companies more competitive on the market and able to follow its trends even when the skills of a single company would not be enough.

The Internet evolution allows not only the employees to better collaborate with the management and the company's partners but also with the stakeholders outside of the company, like customers and shareholders. The network gathers billions of people and potential customers that use blogs, chats and websites to interact, acquire information, buy online and cooperate with organizations (e.g. evaluating their products and advertisement campaigns) in a very easy, fast and inexpensive way. This opportunity of interaction can be welcomed and seen as a fortuitous new way to create value by the enterprises or it can be seen as a threat. It is up to the organizations to decide whether there is a way to exploit the potential of the web in their line of business or not.

The concept considered is self-organization, which is the idea that independent users can work together, willingly or not, and generate something valuable and original. This emergent phenomenon very often reveal itself as successful, yet difficult to control because of its inherent self-regulation and their lack of a hierarchical structure. The determination of the company's boundaries is an important step for an enterprise, which can obtain a significant advantage if it can identify the right mix between the skills that have to remain in the company and those that can be found outside its borders. Sometimes companies keep expertise that could be found outside. While years ago the enterprises focused on the development of products or services using only the company's resources, nowadays a growing number of enterprises work together to achieve those services. The competitive advantage will not concern a single activity but a set of activities run by various firms that, put together, are hard to imitate. While a single one can easily be reproduced, the competitors will have a hard

time obtaining the same benefits from this activity as those that it gives when placed in a system.

In this context, new figures are emerging, like agents and intermediaries that do nothing but drive other companies to the creation of the added value that can be found exploiting the web and the global market. These societies are usually small in the number of employees but handle large business volumes in every field.

Herbalife, a company that works in the health sector producing dietary supplements is an example. Herbalife is a multinational corporation distributed in more than 90 different countries has its shares traded on the New York Stock Exchange (NYSE: HLF) with net sales of \$4.5 billion in 2015 (Herbalife, 2016). The strength of this business is given by its ability to manage its resources, bringing their drugs development teams, composed of doctors and scientists, in laboratories in China and India while supervising their work from the corporation's headquarters. The production part is delegated to various satellite agencies located mostly in Asia and Europe while the retail sales are assigned to thousands of independent distributors. The crucial part handled by Herbalife is logistic, considered key from the management to ensure availability of the products at all times. This focus on the supply-chain and the deals with its partners grants Herbalife a net gain in respect to the centralized approach to the business.

Wishing to expand their market share and their profits many companies find themselves having to adapt their organization's architecture and their business model in order to achieve those results. The society will have to decide if to carry out improvements following a traditional approach focusing in growing its personnel and infrastructures and see its fixed costs or choose to merge with or acquire another company. The merge and acquisition of another company allows to scale the business rapidly. It lets two middle-level companies to share their knowledge in a single development platform and become leader in the market. This way they can avoid new fixed costs. Another alternative to the traditional growing approach is called "fast track business model". Many businesses, instead of following a more organic approach, choose to establish contacts that let them create new value and grow their market share. These enterprises usually acquire low-cost raw materials and outsource the production to some other company. They then exploit designers that offer their

services through the web and provide the company new ideas and new concepts for the future. Ultimately, another partner handles the logistics. In the end, this kind of corporations just need to manage the life cycle of a product employing a small number of qualified staff. Taking advantage of the web it is possible to handle the whole supply-chain in real-time reducing both development and production costs. Marketing and advertisements are usually very expensive when considering the traditional approach, especially when there is a product to sell but they require a limited budget when the potential of the network is exploited. Targeted advertising and the online reviews of the users create a powerful advertising campaign that counts on the word-of-mouth and keep the budget low. Small companies are so free from many fixed costs that burden the more traditional enterprises.

3.2. Key Enterprise 2.0 tools

The advantage of using the Web 2.0 is the possibility to exploit social networks to create a “lever” effect for problem-solving and information management. To be able to achieve this goal tools such as wikis, weblogs and microblogs, social tagging, RSS and social networks are introduced to the enterprise context. Here we give a more detailed overview on each of these concepts.

Wikis are sets of user-editable web pages that offer anyone the ability to easily create and edit pieces of content built collaboratively (Lazar, 2007). This tool came to wide popularity through sites such as Wikipedia.

As Andrew McAfee points out while talking about the benefits of wikis for the enterprises: “The main one they get out of it so far has been the ability to find not so much other pieces of information but other brains all the way across the community.” Beyond the written articles that one can find, “because everyone’s contributions [...] are attributed rather than anonymous, if you’ve done something smart, I can find not only what you’ve done, but I can find you. The point is, I would never have found you within the intelligence community without the new tools.” (McAfee, 2010)

Plus, the ability to easily create links between wiki pages enhances the knowledge sharing dimensions (Levy, 2009). Wikis are designed according to the eleven principles, summarized by (Wagner, 2004).

Principle	Explanation
Open	If a page is found to be incomplete or poorly organized, any reader can edit it as he/she sees fit. Wiki is based on open-source technology.
Incremental	Pages can cite other pages, including pages that have not been written yet.
Organic	The structure and text content of the site is open to editing and evolution.
Mundane	A small number of (irregular) text conventions will provide access to the most useful (but limited) page mark-up.
Universal	The mechanisms of editing and organizing are the same as those of writing, so that any writer is automatically an editor and organizer.
Overt	The formatted (and printed) output will suggest the input required to reproduce it. (For example, location of the page.)
Unified	Page names will be drawn from a flat space so that no additional context is required to interpret them.
Precise	Pages will be titled with sufficient precision to avoid most name clashes, typically by forming noun phrases.
Tolerant	Interpretable (even if undesirable) behaviour is preferred to error messages.
Observable	Activity within the site can be watched and reviewed by any other visitor to the site. Wiki pages are developed based on trust.
Convergent	Duplication can be discouraged or removed by finding and citing similar or related content.

(Wagner, 2004)

Weblogs (or blogs) are web pages for personal use written in the form of a diary. What distinguishes them from a personal website is that blogs are written continually in one page with different posts in chronological order. Blog entries can be commented by other authors and readers, and can be followed with the use of alerts like RSS

technologies. When weblogs and their authors are united in communities, they form social networks. (Levy, 2009).

Recently microblogs became widely popular primarily through the success of Twitter. A microblog is a smaller version of a blog, where authors have limited logs (for example, in Twitter each entry is limited to 140 symbols). Microblogs, like blogs, can bring to features like social networking activities but with a strong focus on mobility (Böhringer & Richter, 2009). Microblogging has also found its place in the enterprise environment. The most popular enterprise microblogging tool is Yammer (www.yammer.com). Its main focus is on inspiring people to share information on “What are you working on”, compared to the focus of Twitter on “What are you doing?”. As the Yammer webpage says: <Just by logging in and sharing “what are you working on” you’re growing your company’s Yammer network and building a knowledge base of information that will benefit your coworkers.>. (Böhringer & Richter, 2009), on the basis of a case study, concluded that microblogging helps creating awareness in a company to support collaboration, communication and coordination.

Tagging can be explained as a practice of attaching keywords to the content (text, media or documents) shared on content management websites. Those keywords are called *tags* and provide semantics to the content (Levy, 2009). Tagging is widely used in bookmarking to let users quickly mark and find later items of interest based on personal and others’ categorization of the content. Tags build personal user categorization systems called *folksonomies*, opposite to well-known taxonomies defined by organizations (Levy, 2009). Tagging has become a standard element of many blogs, wikis, websites and social networks.

Social Networking is the sum of every tool described above. It is based on the ability of users to create a personal profile on the web unique for a particular website or common (exploiting techniques such as OpenID, which allows the user to log in on a website using the profile set on a different one). Profiles are created so that interactions like contributing to wikis, social tagging and commenting on the blogosphere create relationships between people (Tapiador, et al., 2006).

Barnes (1954) defined social network as a social structure comprised of nodes (individuals or organizations) that are connected by one or more specific types of relations and this definition is still valid today. In general, social networks and their

analysis are important for determining the ways groups operate, how problems are solved and the extent to which people succeed in attaining goals (Lai & Turban, 2008). Being already very popular on the Internet, social networking tools are introduced in organizations. Profiling systems for employees, we see that the ability to author and comment documents and pieces of knowledge in knowledge management systems automatically creates relationships between people (Boeije, et al., 2009). During the years, a wide variety of tools has been developed to increase the usefulness of social networks. In the next paragraph we are going to analyse in more detail the kind of social networks actually in use.

3.3. ESNs Examples

At this point, we are going to introduce some examples of enterprise social networks in order to give a closer look on how an ESN can look like.

We will start talking about Salesforce. Salesforce is an enterprise customer relationship management (CRM) giant, which has improved its CRM services providing social networking capabilities. Its new Chatter service is available on Salesforce's real-time collaboration cloud. Users can use it to set up profiles and generate status updates, which might be questions, bits of information and/or knowledge or relevant hyperlinks. All of this is then aggregated and broadcasted to co-workers on their personal page. Essentially, an employee's personal page contains a flow of comments and updates regarding those in that particular network. Employees can also follow the rest of their colleagues from around the company, not just those in their own personal network, enabling cross-organizational knowledge sharing. Towards that end, Chatter also provides a profile database that users can tap into to find someone with the needed skills for a particular project. All of this is accessible via both desktop and mobile. Like Salesforce.com, many of well-known software companies have developed collaboration tools with similar features.



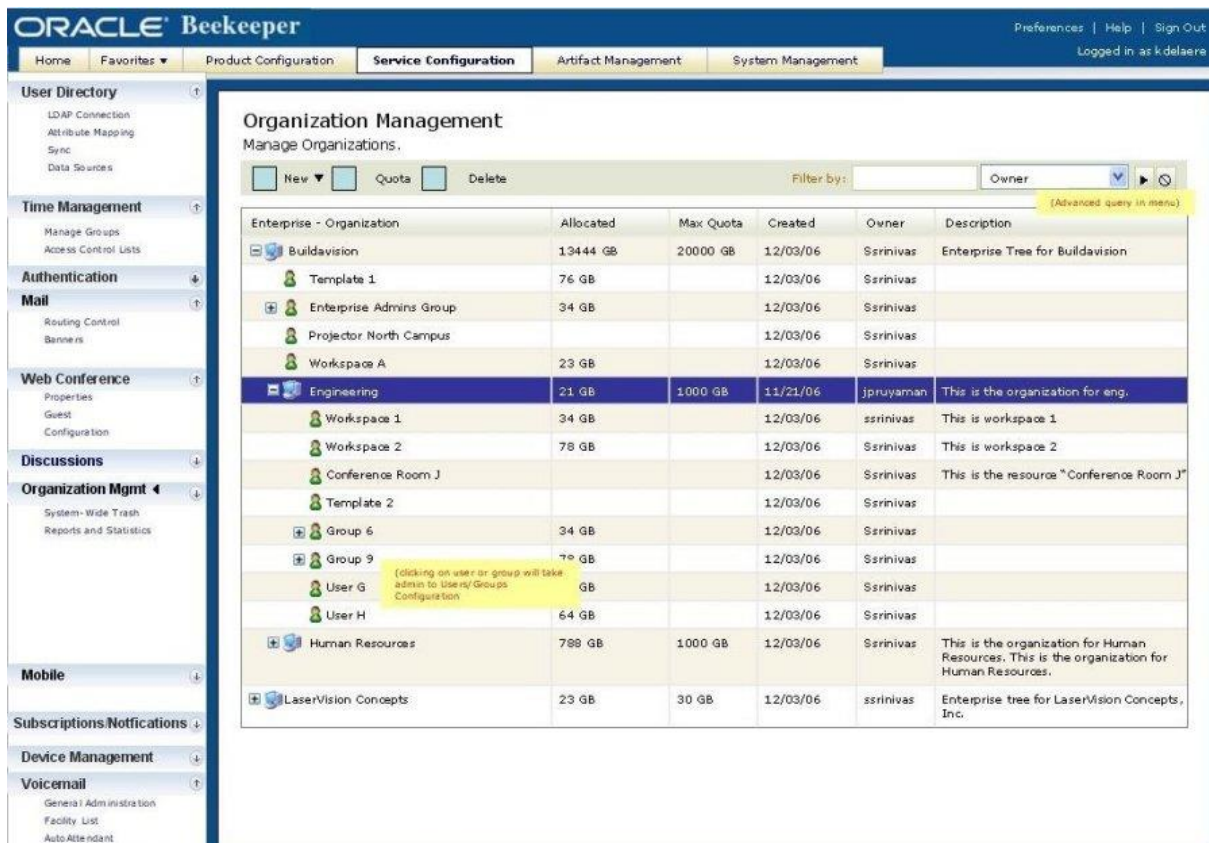
Chatter page example

Oracle's Beehive collaboration platform provides a suite of tools such as email, calendar, voicemail, instant messaging, group chat, Presence, web conferencing, audio conferencing, team workspaces, document sharing, and employee directory. It has support for mobile devices and can be integrated with Cisco, Avaya and Nortel infrastructures to deliver, as an example, voicemails and faxes to an email's inbox. Outlook can also be integrated in their software, providing access in particular to personal email, address book and calendar in order, as an example, to schedule appointments and deadlines. It also allows users to schedule conferences via Microsoft Outlook, Beehive Webmail or any standards-based CalDAV client. Calendaring Extensions to WebDAV, or CalDAV, is an Internet standard allowing a client to access scheduling information on a remote server. It extends WebDAV (HTTP-based protocol for data manipulation) specification and uses iCalendar format for the data.

On Beehive, the work team can create its own workspace starting from given templates and rely on various features like team wiki, calendar, team task management, discussion forums, contextual search and team announcements through an ad-hoc microblog and its RSS feeds. The most interesting feature for a team is the document library, which allows users to:

- lock a file, to stop others to modify it
- use check in and check out operations, that let a user download a copy of a file and later upload the new version in the library and merge it with its online counterpart
- set up specific privileges for each user
- have control over the workflow of the project and
- remote content sharing with Oracle UCM (Universal Content Management)

Beehive is then extendible with the use of scalable Oracle technology like, but not limited to, “Oracle Information Rights Management” (IRM) or “Oracle Secure Enterprise Search” (SES). Beyond Oracle’s extensions, the system that can also be personalized with its given RESTful (that follows the REpresentational State Transfer principle) APIs that exploit BPEL (Business Process Execution Language) and other common specifications like the with all the standard communication protocols (like IMAP, SMTP, CalDAV, iSchedule, WebDAV, XMPP, FTP, OMA Data Synchronization, PushIMAP, SIP, and VoiceXML). All of these compatibilities make the developers’ life easy when building an addition for the platform. “Oracle Beehive Mobile Communicator” is the name of the app for smartphones that help workers stay connected using IM (Instant Messages) while away from the computer. (Oracle, 2016)



Beehive page example

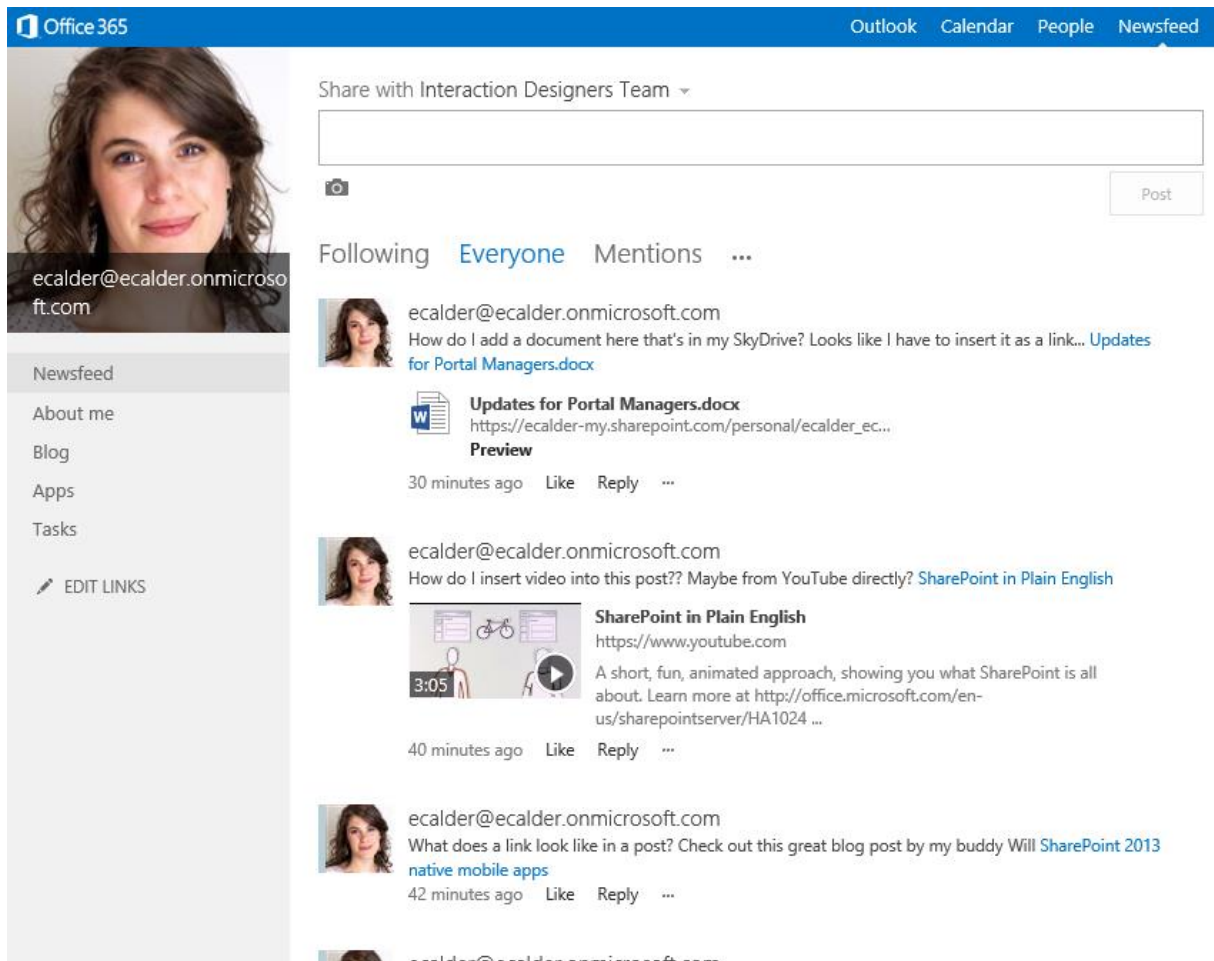
Microsoft's SharePoint is heavily used within many enterprises. It includes many features like the creation of personal websites, personal tasks, blogs and microblogs, team sites and community sites, which make it easy for users to find and connect with the people and content that matter to them and to share information and ideas.

The developers can add new social features or extend the features that are already available in SharePoint 2013. For example, you can create an app that lets you find and follow people who have a common interest, create a custom visualization of feed data, or publish custom activities to the feed.

SharePoint Server 2013 provides the following APIs that one can use to programmatically work with social feeds:

- Client object models for managed code
 - .NET client object model
 - Silverlight client object model
 - Mobile client object model
- JavaScript object model
- Representational State Transfer (REST) service

- Server object model



Newsfeed page example

While Oracle's Beehive tries very hard to make it easy for developers to personalize the environment and shows a platform similar to SharePoint, a significant difference in the latter is the presence of the "feeds". Regarding the handling of the feeds from all the profiles in the network, SharePoint gives the users the ability to choose between Yammer and Newsfeed.

SharePoint Newsfeed used to be the default option for social experiences in Microsoft's Office 365. It offers all the standard features such as the possibility to like, comment, share the published content and follow others in order to have their feeds straight in the homepage. It allows tagging and lets users cite each other in the feeds. Newsfeed includes also an app for Windows Phone and iPhone smartphones that lets employees to stay connected to your organization's social pulse while on the go.

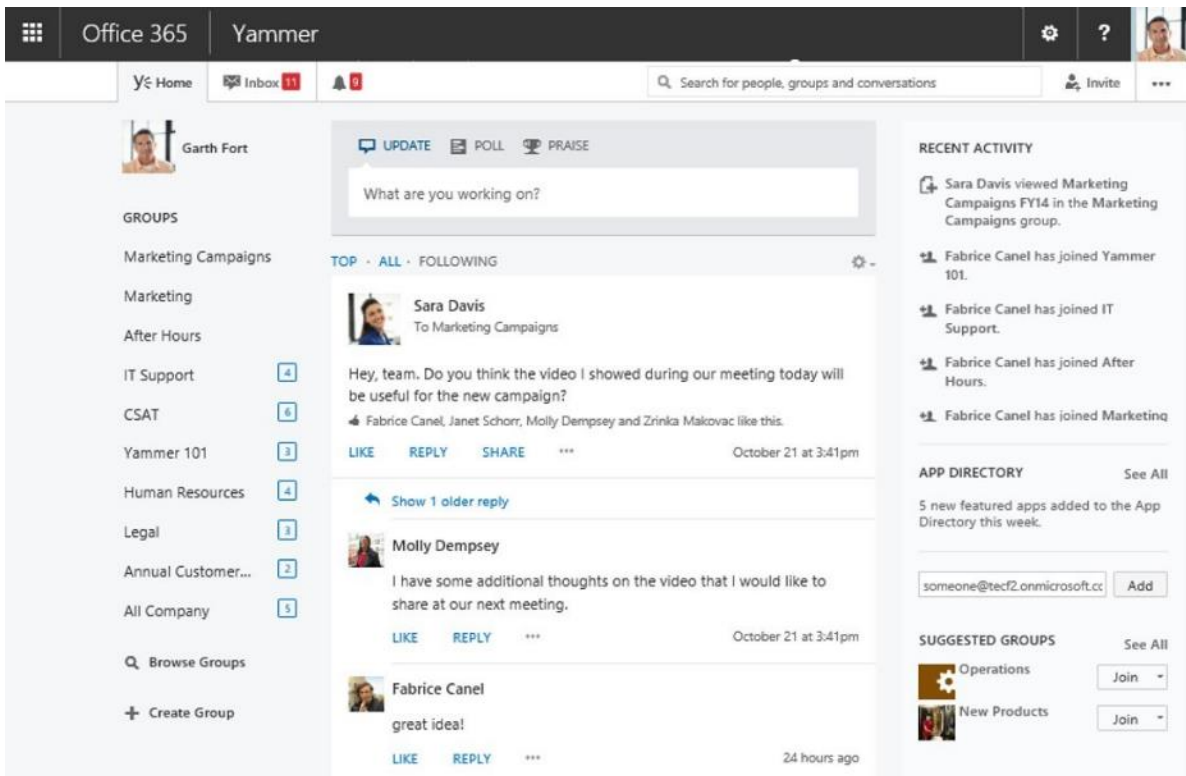
Yammer, instead, is a private, secure social network for an organization that will allow people to collaborate securely across departments and geographies. It is designed to inspire company-wide knowledge exchange and to increase team efficiency. Only people with a verified company email address can join the company's network.

To talk about Yammer we have to do a little digression talking about Twitter, a social networking app made famous by celebrities who tweet hourly updates on what they are doing (e.g., eating lunch, shopping, etc.). Twitter itself is not useful in a company's environment. It lacks to all the tools that an organization requires to control work teams or share knowledge. This said, to bridge the gap Twitter developed an enterprise social networking application called Yammer, bought in 2012 by Microsoft. With the ability to integrate with tools such as SharePoint, Yammer provides a suite of tools including enterprise microblogging, communities, company directory, direct messaging, groups, and knowledge base. Much of what Yammer offers is free with the basic service. With a fee, however, it provides niceties as security controls, admin controls, broadcast messages, enhanced support, SharePoint integration, keyword monitoring, and virtual firewall solution. Yammer can be used by the software development team to interactively discuss any aspect of a project.

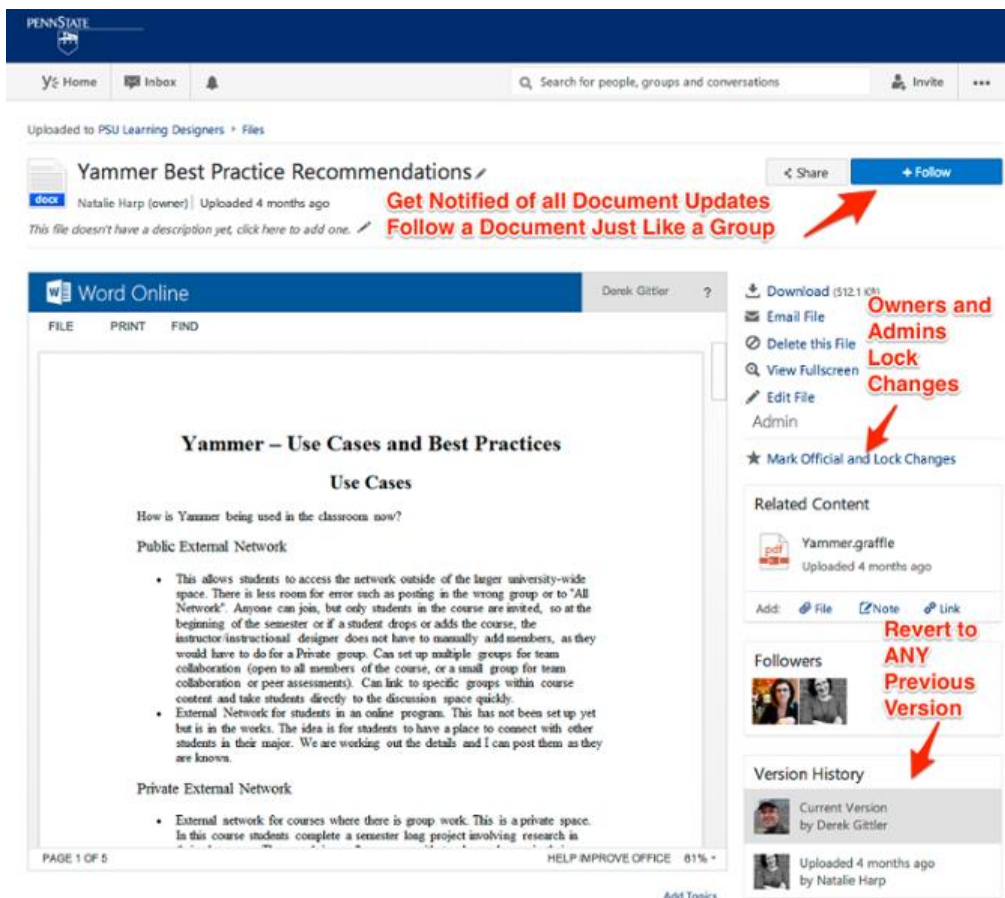
Project groups can use SharePoint in many ways:

- to write up personal research and make comments on others' research;
- to ask questions;
- to post links to resources that might be of interest to others in the group;
- to add details for upcoming events and meetings;
- to let each other know what they're up to;
- to add comments to other team members' information and pages and
- to record minutes of meetings in real time.

(Microsoft, 2016)



Yammer page example



Yammer document sharing page example

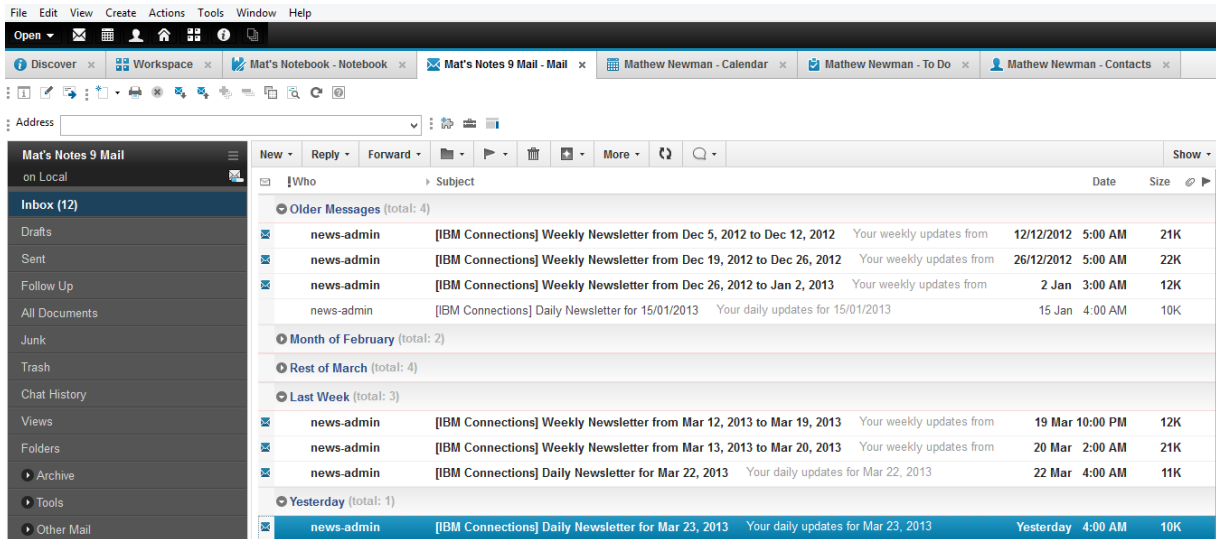
Microsoft's Skype for Business, formerly Lync, is another tool that Microsoft acquired and integrated in its SharePoint platform. Skype for Business not only works on desktops and mobiles but, most importantly, gives the users a way to communicate not only through messages and images, but via voice, video, or document share from anywhere with many people at a time, increasing even more the flexibility of the system.

(Microsoft, 2016)

One of the first companies to work in the collaborative market was Lotus, with its platform: Lotus Notes. Now owned by IBM, IBM Notes (formerly IBM Lotus Notes) brings together a wide array of tools: instant messaging, team rooms, discussion forums, and even application widgets. There is also a wide variety of free tools available, which can be adapted for one's purposes. IBM Notes does not use feeds like SharePoint but exploits emails to handle communication and focuses on improving what emails bring, together with plain text. In one email we can find:

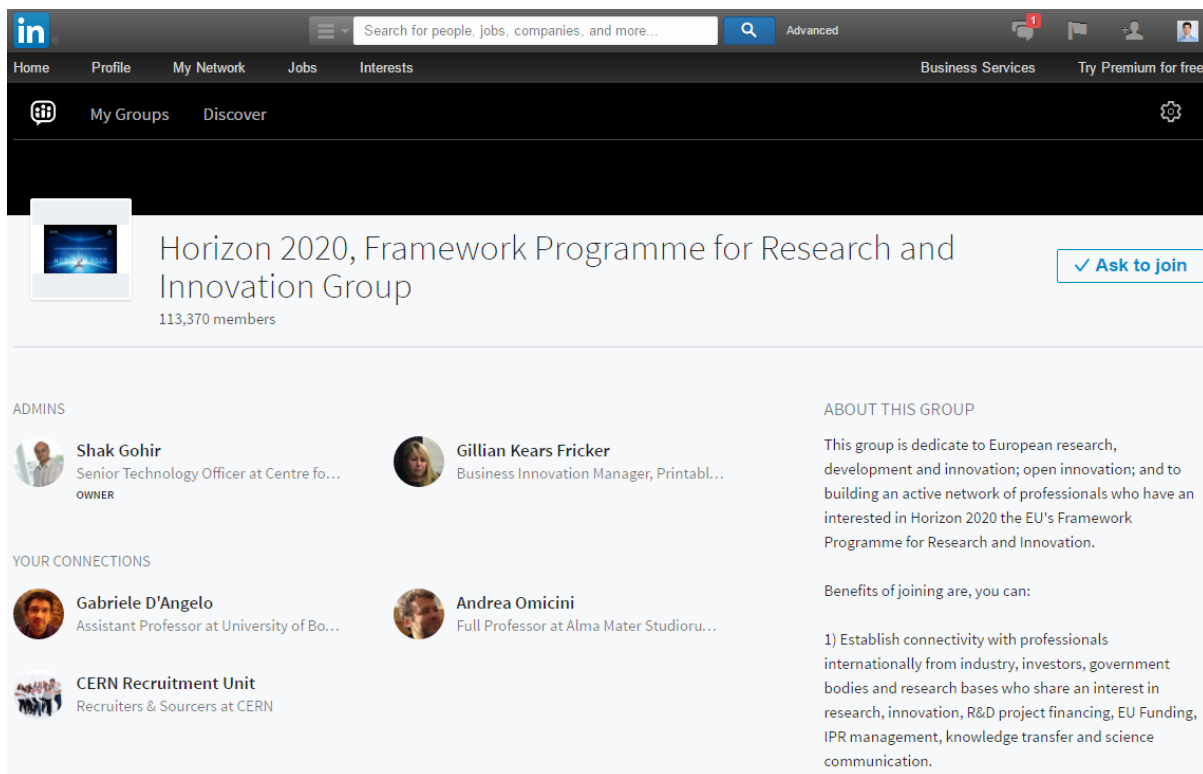
- all the information on the contact that sent it
- the text of the email where parts of it can be made as "live text", which enables the recipient to, for example, call phone numbers or open Google Maps to see the location cited in the email through a single click
- options to "like", "follow" or "share" a document received or that is on the network, in order to show appreciation, be notified when changes are made to the document or simply share the file. It is also possible to write a personal comment that will later be visible for all the community.

The email platform offers the user many features like drag-and-drop, which makes it easier to select the recipients and the files to attach. Plus, there is a comprehensive search option that allows the user to filter emails by the people involved as senders or recipients, subject of the email, date in which it has been sent or simply "any column", which covers everything else.



IBM Notes page example

Another great example of a social environment is LinkedIn, which has been widely used to provide networking capabilities for business people. A relevant feature is LinkedIn groups. A group can be created for any purpose. In order to join a group a user can either click on “Ask to join” on the group page or respond to an invitation from a group member or manager. Thus, project teams can make use of the already-developed facilities LinkedIn provides. Using this platform the employees can share knowledge and handle team work. This said, there is one big issue regarding LinkedIn: lack of confidentiality. Since it cannot be stored on the servers of the company, all the information shared online from every member of the enterprise can be seen (at least) from the LinkedIn system administrators, which may not be entrusted by the company to look at their classified data. While usually this may not pose a threat, often the information exchanged regards projects or initiatives that should remain visible only to the employees of the company, which sign confidentiality agreements before beginning their jobs. While for some organizations it could be just fine to make their efforts visible, others like pharmaceutical companies would not agree in sharing details on their researches.



LinkedIn groups example – Horizon 2020's LinkedIn group

One might think that the use of these sorts of ad-hoc discussion tools would degenerate into chaos. In truth, this rarely happens, even in a social network of anonymous users. The largest wiki of all, Wikipedia, is fairly resistant to vandalism and ideological battles. The reason for this is “the emergent behaviour of a Pro-Am (meaning professional and amateur) swarm of self-appointed curators.” This group of curators has self-organized what Anderson terms the most comprehensive encyclopaedia in history—creating order from chaos. This is what is called “peer production.” (Anderson, 2006)

These are only few of the many ESNs that are out there, trying to give the best platform possible to employees of any company. We have covered here only the main features of the ESNs considered, but it already shows the potential of social technologies.

3.4. Real-Life Stories

The first story we are going to investigate regards a German multinational software giant that makes enterprise software to manage business operations and customer relations: SAP SE (System analyses and Programme networking; Systems, Applications & Products in Data Processing).

In the early 2000s, SAP encountered increasing challenges in the ways it provided its network with information and its customers with support services. At the time, SAP used common support channels like email and phone but it did not satisfy the clients. Another issue was that potential customers were having trouble determining if SAP's complex software solutions would meet their needs.

In order to address the communication issues between SAP and its 170,000 clients a proper solution had to be found. Experts determined that improving the old channels, like adding more staff to existing support channels, would have had minimal impact. At the time, online communities were new to the market, used to connect with people with similar interests. Nevertheless, SAP had both the resources and the motivation to test these new concepts for its service issues. The goal was to enlist clients and other stakeholders to join the community to share ideas and solve problems. This way, not only SAP could have had a new platform to communicate with the clients in a more efficient way, but also customers could work together directly and exchange valuable knowledge. Mark Finnern, who went on to become an SAP community evangelist (a formally recognized champion of the service), said: "To make it work, we knew we would have to put the people in our company on the front line before customers would engage. It would be 90% of us and 10% of them at first. But we knew if we did that it would eventually be 10% of us and 90% of them." (Happe, 2010) It is essential for the social organizations to commit seriously in order to kick-start participation by customers and partners. Using this approach in just two years 100,000 customers joined. "By plugging customers into the process of creating reusable knowledge, every contribution made both SAP and the community much richer and more useful. What's more, the process was repeatable, scalable and relatively inexpensive compared to traditional customer support methods." (Hinchcliffe & Kim, 2012) The social network works every day, 24 hours a day, and delivers high-quality information to stakeholders

making the company save on support costs. Moreover, it improves customer retention, which is a crucial aspect for a company like SAP.

The second story regards another software giant that develops, manufactures, licenses, supports and sells computer software, consumer electronics and personal computers and services: Microsoft.

In 2009, a survey revealed that Microsoft's partners were less than satisfied with the company. Precisely, 62% of them expressed desire for a stronger support that would not include just periodical formal email announcements and occasional updates. Microsoft executives decided to take action and use the same social media used by others to organize quickly and effectively: Twitter and blogs. For each country, a new Twitter account was set up to best address the partners while speaking in their own language (e.g. <https://twitter.com/microsoftfrance> or <https://twitter.com/microsoftde>), fostering participation. Microsoft applied a fast read-and-respond strategy that aimed at answering as fast as possible to every question/problem. Satisfaction levels increased by 15% the first year and 17% in the second while phone calls for assistance dropped, substantiating the fact that the program was working. (Klier, 2011)

Many social business transformation stories exist outside the technology industry. Let us look at how rethinking existing business processes can affect one of the biggest consumer products multinational companies: Procter and Gamble Co. (P&G).

One of its best-known products, Old Spice, once a customer favourite, was losing market share, especially among young consumers. In response, Old Spice used advertising slogans like "If your grandfather hadn't worn it, you wouldn't exist", but it did little to increase sales. Since the traditional approach did not work, they had to figure out a new way to communicate with the customers. The idea they came up with required the use of both traditional methods and social media. Following this then-revolutionary concept, the brand launched a new campaign during the Super Bowl and on television, starring actor Isaiah Mustafa. The social media came into play when all the television commercials were posted on YouTube, with the @OldSpice Twitter account engaging with consumers in real-time. When somebody wrote to @OldSpice, Mustafa answered to those messages with new spots posted on YouTube and then referenced on Twitter. Advertisements that typically take weeks were produced in a

matter of hours, with a copywriter standing by, an actor and a warehouse full of costumes where one could shoot the ad.



One of the many cited tweets (starring Isaiah Mustafa)

“The wide reach across traditional media kick-started social media participation, which then led to compelling two-way conversations in social media between Old Spice and consumers.” (Hinchcliffe & Kim, 2012) On the very first day, it received 6 million views and in just six months achieved 1.4 billion views. The combined campaign reached half of the Internet over its lifetime (Schroeder, 2010). After years of declining sales, the new campaign helped increase the sales for Old Spice up by 107% (Griner, 2010). At the same time, Old Spice has become the #1 Most Viewed Sponsored YouTube Channel (P&G, s.d.).

A good example of *crowdsourcing*, which is the process of obtaining needed services, ideas, or content by soliciting contributions from a large group of people (Merriam-

Webster, 2016), is the “Goldcorp Challenge”. Goldcorp, a Canadian mining firm, was having difficulties finding more gold to prospect on its 55,000 acres in Ontario. In literal desperation, it opened up all its 400 megabytes of its valuable prospecting data to the geological community for help. Despite worries about loss of secrecy and looking foolish for not finding it themselves, they offered a \$575,000 in prize for successful recommendations. More than one thousand entities from over 50 countries applied unique and highly disparate methods to crunch the data, including applied mathematics, advanced physics, computer visualization and many other creative methods. The success rate was impressive. Over 80% of new targets provided from the community yielded useful finds. Ultimately, the challenge unearthed 8 million ounces of gold and catapulted the organization from a poorly performing company worth a mere \$100 million into a \$9 billion mining giant in a few years. (Tapscott & Williams, 2006) (Hinchcliffe & Kim, 2012)

In 2010, the leaders of NASA’s Marshall Space Flight Centre (MSFC) found themselves having to deal with some serious problems. The space shuttle was reaching its end of life and the plans for its replacement were not going anywhere while the Constellation program and the Ares rockets were being criticized from outside scientists and engineers. The Constellation program main goals were the "completion of the International Space Station (ISS)" and the "return to the Moon no later than 2020" with a crewed flight to the planet Mars as the ultimate goal. The Ares rockets, instead, were the program's booster rockets. The ones that had to be used to take the shuttle to its destination. (Connolly, 2006). So the shuttle and NASA’s programs for the future of space exploration were at risk when President Barack Obama announced a proposal to cancel NASA’s program in favour of privatization. As MSFC leaders began rethinking the nature and value of the space mission, recognized the importance of involving more people in the process of dealing with these fundamental questions. They began envisioning how community collaboration could help the mission and increase awareness of the value NASA and MSFC provide to the world. (Pettus & Bradley, 2009)

Jonathan Pettus, Chief Information Officer at MSFC, said:

“We believed that social media could have a significant impact on how we pursue our mission. That using it could help us collaborate in new ways to build rockets better, but you cannot just put the technology out

there and expect big results. It is not that easy. We are not all the way there yet, but we are moving forward towards our goals and our focus on vision and purpose has provided a foundation for continued progress.”
(Pettus, 2011)

Organizations like NASA began to use social media to exploit the advantages of community collaboration. Some pursue it sporadically, but some are considering involving the “social” part in their day-to-day operations. To give an example on how things improved during the years for NASA we can look at the New Horizons space probe that, on July 14, 2015, turned all its sensors to Pluto for a 20-hour long flyby. Before it went dark – no contact with Earth – New Horizons sent one last chunk of data home. Contained therein was the best picture of Pluto in history. When it received that image, the agency did something unique: NASA posted the image on Instagram. Doing so, NASA allowed people to share and comment not only on Instagram, but also on many other social networks all over the world, receiving hundreds of thousands of “likes” and comments. From putting the science team on Reddit for an AMA (Ask Me Anything) (NASA, 2015) to pulling questions from Twitter during live press briefings, the New Horizons mission reached out to millions of fans. Moreover, it is not just New Horizons, NASA’s social media strategy is ambitious and, most importantly, carefully planned.

Concerning the need to help its mission, NASA organized a series of call of proposals with the aim to complement its research and speed up the development of new technologies. They launched competitions for each piece of technology or theory needed to move forward in the reach for Mars, coordinating with internal and external stakeholders, including academia, industry and other government agencies. As an example, the “Sample Return Robot Competition” is going to grant a \$1.5 Million reward to whoever manages to build a robot that can locate, collect and return geologic samples on natural terrain without human control and within a specified time (NASA, 2016).

These innovations may enhance NASA's space exploration capabilities and could have applications on Earth. All these projects, built outside the agency, are going to save it both time and money. As stated by the presidential panel appointed for this purpose on 2009, “allowing companies to build and launch their own rockets and spacecraft to carry American astronauts into orbit would save money and also free up

NASA to focus on more ambitious, longer-term goals.” (Pasztor, 2010) This kind of initiatives surely encourages innovation and allows NASA’s employees to go forward and tackle the rest of the challenges that need to be addressed before launching the next mission.

It is paramount to have a clear vision regarding social media and community engagement that states the potential benefits for the organization. A vision statement, similar to the Pettus declaration, which expresses these concepts serves two main purposes. First, it articulates the belief of leadership in the importance and value of community based collaboration. Second, it concretely identifies significant opportunities for the firm where such collaboration can add value by helping the organization move closer to its goals (Bradley & McDonald, 2011).

In general, the vision on ESNs is that everybody inside and outside of an organization, from the CEO (Chief Executive Officer) to the newcomers, from the partners to the customers and the fans, can provide a solution to a company’s problem or have anyway a meaningful impact in a project’s development. The Internet has enabled everyone to learn everything. Hence, a 12-years-old boy or girl can now be more knowledgeable on a subject than a 30-years-old can just because the latter lacked the passion needed to study the subject in detail. Considering this, it becomes easy to understand the truth is this vision.

3.5. Benefits of Knowledge Management Systems (KMSs) and Enterprise Social Networks (ESNs)

The main principles when talking about social businesses, KMSs and ESNs are the following:

- I. *Harnessing collective intelligence* implies benefiting from the cumulative expertise of a group, rather than an individual, to make decisions (Lykourantzou, et al., 2010).

- II. *Authoring* is important to elicit the contribution of every person to collaborative efforts or products and the contribution of any kind, whether it is knowledge, insight, experience, a comment, a fact, an edit, a link and so on (McAfee, 2006).
- III. *Folksonomies* are user-generated classifications enabled by tagging. They reflect the information structures and relationships that people actually use, not a classification planned before. Besides, user tags reflect the popularity of subjects and identify knowledge pieces most used by employees. Since they are generated without control, they can be redundant. In any case, they create a one-level classification (McAfee, 2006).
- IV. *Reputation* of an author or an object (a wiki article, blog, etc...) is defined not by some set of characteristics but by the number of links directing to the object; number of followers, "likes" etc... In order to make this principle work many people must have the ability to build and share links, in general to express one's appreciation. This principle is highly dependent on the number of participants in a network.
- V. *Recommendations* are used to propose to users the items most relevant to their interests based on their previous behaviour (McAfee, 2006).
- VI. *Signals* (RSS, notifications) are useful to notify users when new content of interest, comment, new post or reply appears.
- VII. "*Wisdom of crowds*" effect implies that a large number of people making small contributions can create a quality product (Kittur & Kraut, 2008).
- VIII. *Network Externalities* or "network effect" means that the more users a system has the more valuable it becomes for every single user (Kim, et al., 2009).

If the principles stated above are respected, using KMSs together with ESNs creates many benefits for an organization. Hereunder are explained the most important ones.

1. Enabling better and faster decision making

When faced with the need to respond to a customer, solve a problem, analyse trends, assess markets, benchmark against peers, understand competition, create new offerings, to plan strategy and think critically one typically looks for the information and the resources needed to support these activities. By delivering relevant information at

the right time through structure, search, subscription, syndication and support, a knowledge management environment like the ESNs can provide the knowledge necessary to take well-thought decisions. The power of collaboration remains in bringing together a large number of people with diverse opinions and experiences that will allow the discovery of the best-fitting solutions.

2. Reusing ideas, documents, and expertise

Once the company develops an effective process, it will be desirable to share that knowledge and let others use the same process each time a similar problem arises, avoiding redundant efforts. "No one likes to spend time doing something over again" (Garfield, 2014). Just as the recycling of materials is good for the environment, reuse is good for organizations because it minimizes rework, prevents risks, saves time and money, keeps employees morale up and accelerates progress. This approach allows employees to learn how things are done, leads to predictable and high-quality results and enables large organizations to be consistent in how work is performed. Moreover, the reuse of knowledge allows the outcomes to be based on actual experiences, making the task easier for everyone involved.

3. Accelerating delivery to customers

Speed of execution is another important differentiator among competitors. All other things being equal, the company that can deliver sooner wins. Knowledge sharing, reuse and innovation can significantly reduce the time needed to deliver a proposal, product or service to a customer.

4. Avoiding making the same mistakes twice

George Santayana said, "Those who ignore history are doomed to repeat it." Knowledge management allows the sharing of the lessons learned not only about successes but also about failures. In order to do so the organization must have a culture of trust, openness, and reward for willingness to talk about what someone did wrong. The potential benefits are enormous. We can give many examples. Think of NASA, if it learns why a space shuttle exploded, it can prevent it to happen again and save lives. If FEMA (Federal Emergency Management Agency) learns what went wrong in responding to Hurricane Katrina, it can reduce the losses caused by future disasters. If engineers learn why highways and buildings collapsed during a previous

earthquake, they can design new ones to better withstand future earthquakes. (Garfield, 2014)

5. Communicating important information widely and quickly

There are two arguments that should be considered in this matter. First, when working in a team of 10 people it is easy to talk and share information with every other member. As the number of people increases, though, communication becomes increasingly difficult. It is fundamental to actively enable the organization to leverage its size. Second, since almost everyone today is an information worker either completely or partially, we all need information to do our jobs effectively. A problem that can arise today when reaching for information is called “information overload”: it refers to the difficulty a person can have understanding an issue and making decisions caused by the presence of too much information.

How can we get to everyone in the company information that is targeted, useful, and timely without drowning in a sea of email, having to visit hundreds of web sites, or reading through tons of printed material? Knowledge management helps address this problem through personalized portals, targeted subscriptions, RSS feeds, tagging, and specialized search engines. To best share the knowledge with the rest of the organization we can make use of tools like community discussion forums, training events, ask the expert systems, recorded presentations, podcasts and blogs.

6. Showing customers how knowledge is used for their benefit

In competitive situations, it is important to be able to differentiate yourself from other firms. Demonstrating to potential and current customers that the enterprise has widespread expertise and has ways of bringing it to bear for their benefit can help convince them to start or continue doing business with the company. Conversely, failure to do so could leave the company vulnerable to competitors who can demonstrate their knowledge management capabilities and benefits. (Garfield, 2014)

Between the researches on the subject, an interesting interview nicely explains some of the benefits of using an ESN:

“We made knowledge maps – knowledge areas and persons and put this in the system. I like it and I know that younger generation loves it, too. The older generation says they know everybody. But it is interesting to have this

discussion between the older and the younger people, saying “You know everybody, but I can’t get into your network” – “Then just call me”. But then he (an expert in some field) starts complaining that he is constantly being questioned about trivial things. Then he realizes that it is interesting to write it down. Besides, I can always say that if you have a question about manufacturing engineering ask “Jan”, because he knows everybody, who knows something about the question. But then Jan will start to complain for constantly being bothered by people with some questions. And then they agree on writing down who knows what.” (Gordeyeva, 2010)

3.6. The Right ESN

In the previous sections and chapters, we have talked about a wide variety of social networks and tools. The reason why we did so is to introduce the concept of Enterprise Social Networks and give a more comprehensive view on the subject. It is necessary now to state that even if we introduced many examples and considered many technologies, when talking about social media the focus should never be on technology but on conversations between people, since it is the conversations that have the power to influence opinions. Billions of conversations take place every day about new products, new promotions, the prices of the goods and services and the opinions (both good and bad) of the customers on a brand (Keller, 2007). These types of conversations strongly affect the opinions and the purchasing behaviour of the consumers. (InSites Consulting, 2012) Even if the majority of conversations of this kind takes place offline (InSites Consulting, 2009), online conversations have the advantages to be able to reach large audiences quickly, easily and cheaply. Numerous studies have underlined the relationship between positive conversations and good sales figures (Herr, et al., 1991). Being able to exploit this potential can increase people’s perception of the company and this will not only increase the sales, but also make it easier to recruit new talented staff.

The highest proportion of unused conversation potential can be found among customers and staff. Everyone will agree that satisfied customers are important. Satisfied customers who talk about the company are even more important. A

European survey of various sectors showed that 28 percent of customers were very satisfied with particular products or services they received, but did not talk with anyone about them (Bellenghem, 2012). In other words, almost a third of the consumers had a good experience of a product or service but said nothing about it to anyone. It is clear that the amount of unused conversation potential cited before can be quite alarming. Reducing the level of unused conversation potential makes a company have a wider reach and greater impact in everything it does, from the development stage to the customer relationship management.

In order to choose the ESN that best fits an organization, and that will likely increase conversations on their products and services, there are many factors to take into consideration.

First, does the company need its products and services to be developed and improved within a community of people or is it fine to use the traditional approach that limits a specific team of employees to work on them?

To limit the number of employees working on a project makes it easy to the team to work as a unit and have good communication. This said, it also limits the amount of innovation put in the development phase and it does not guarantee that all the knowledge needed to build a product or service is already in the team. New requirements may come up and new technologies be considered while brainstorming the new service or product and it would be useful to be able to easily spot who can help or manage to read a report on a previous project developed with that technology.

Second, does the company need a way to build knowledge retention and knowledge search or does it just want to improve the way it communicates the news to the consumers?

A simple Facebook page or Twitter account, possibly supported by an Instagram or YouTube channel, could be enough to engage the customers and raise the awareness on the brand. Yet Twitter's 140 characters cannot really be knowledge. There is not much that can be written there. One can put a question or an answer. Still, it facilitates conversation since writing a short message does not require major efforts. On the other hand, if the goal is to collect knowledge to be used in the future by any of the stakeholders, than a Twitter account will not be enough. In this case, a dedicated ESN can be the

right solution, since it helps making personal and corporate knowledge more visible and accessible for everybody.

Third, does the company need its partners to be able to contact any of the employees in order to get the best answers to their questions or is it preferable to have a single element in the organization to take care of this kind of public relations?

Without social tools, there would probably be a single person responsible for the client. If that person is an expert on the field it could already be enough. This said, the enterprise might want to speed up the learning curve of its employees and be sure that its workers can always find the answers they need for their client in little time. Social networking tools create social presence (which means having some information about each person on the network) and context for communication, so that a person's job, title and responsibilities are visible to everyone else. Thanks to this, both partners and colleagues are able to recognise when somebody is the right person to contact or not.

Fourth, is it important to the managers to talk with any other employee easily or do they prefer a more hierarchical arrangement, where a worker can speak only to his peers and its direct supervisor?

With a hierarchical approach, the work environment is a more predictable and manageable place where everyone just follows what the management says. In this kind of circumstances, only few people can take initiatives. Using dedicated Enterprise Social Networks flattens the traditional chain of command making any member of the company, including the managers and chiefs, more approachable. This allows everyone to obtain knowledge and get a feedback from any employee while highlighting the most useful interventions, which will receive more "likes" or similar tokens of appreciation. This way it is harder for the management to play their role, but it uses this wider reach to get better ideas from many more sources.

Fifth, how confidential is the information that is going to be exchanged on the network?

The traditional approach keeps using closed systems, like emails, that shield the company from involuntarily leaking data outside the intranet. Having a more open approach, on the other hand, gives the company all the benefits that we discussed earlier.

Questions like these help narrowing down the characteristics that an enterprise is looking for. Once those are known, a proper ESN can be chosen or even developed from scratches. It cannot be emphasized enough, though, that having a tool as an ESN does not mean that it is going to be used as much as expected. It is crucial when becoming a social organization to have a Vision and a Culture that supports it. Without these two elements, the best ESN is useless. Moreover, using a social approach does not simply mean that the enterprise gets another tool. It is necessary to rethink the way the company does business, because it is going to alter all the processes used to build the final products and services. “Social business amplifies partner activities by driving network effects and other ecosystem benefits, such as having the organization come together with all of its partner companies to market, sell, innovate, support or otherwise accomplish business objectives” (Hinchcliffe & Kim, 2012). This means that the company will have to take into account all the entities involved instead of deciding by itself.

4. CERN as a Social Organization

This chapter will introduce CERN and present Social, its enterprise social network.

4.1. CERN

At the end of the Second World War, European science was no longer world-class. Following the example of international organizations, a handful of visionary scientists imagined creating a European atomic physics laboratory. Raoul Dautry, Pierre Auger and Lew Kowarski in France, Edoardo Amaldi in Italy and Niels Bohr in Denmark were among these pioneers. Such a laboratory would not only unite European scientists but also allow them to share the increasing costs of nuclear physics facilities.

French physicist Louis de Broglie put forward the first official proposal for the creation of a European laboratory at the European Cultural Conference, which opened in Lausanne on 9 December 1949. A further push came at the fifth UNESCO General Conference, held in Florence in June 1950, where American physicist and Nobel laureate Isidor Rabi tabled a resolution authorizing UNESCO to "assist and encourage the formation of regional research laboratories in order to increase international scientific collaboration..."

At an intergovernmental meeting of UNESCO in Paris in December 1951, the first resolution concerning the establishment of a European Council for Nuclear Research (Conseil Européen pour la Recherche Nucléaire) was adopted. Two months later, 11 countries signed an agreement establishing the provisional council – the acronym CERN was born (CERN, 2016).



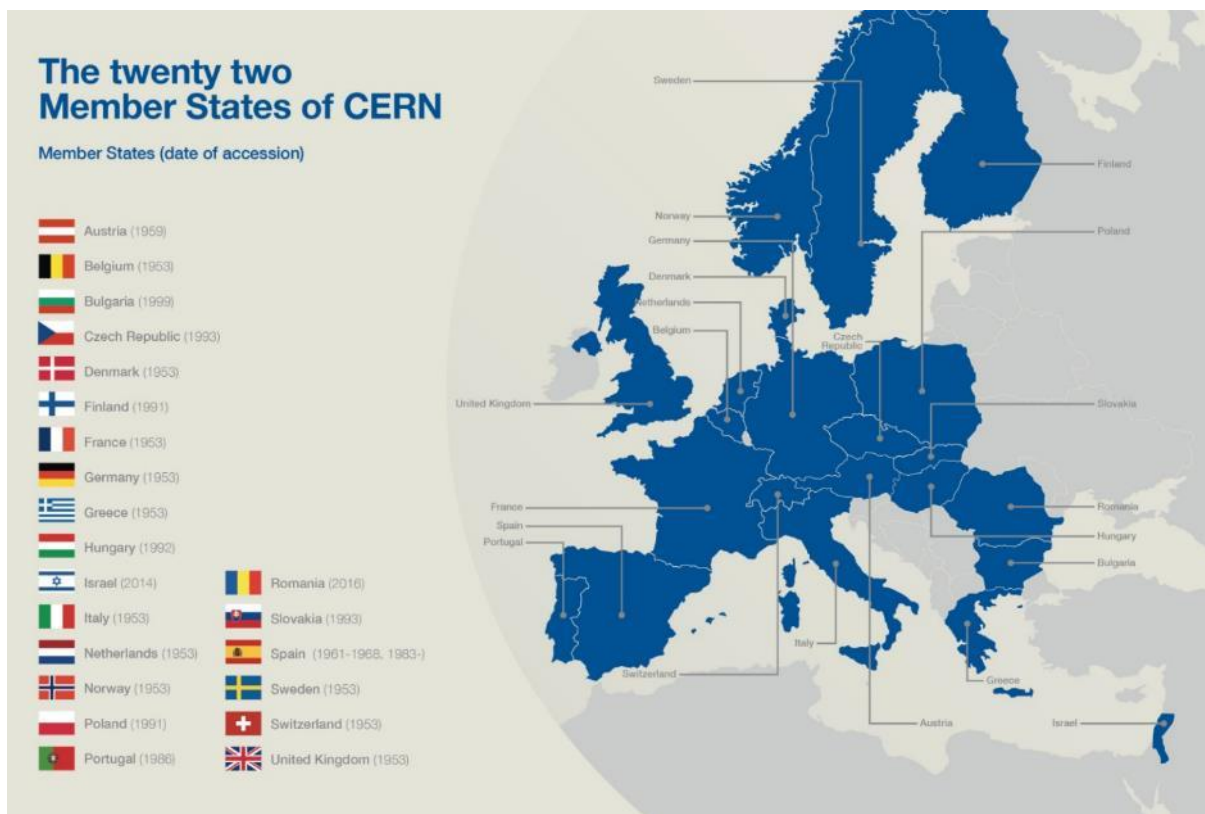
On 17 May, 1954, the first shovel of earth was dug on the Meyrin site in Switzerland under the eyes of Geneva officials and members of CERN staff.

At CERN, many experiments are carried out at the same time with one main goal, to understand the fundamental structure of the universe and push the boundaries of the human knowledge a little further. Physicists and engineers use state-of-the-art technology and most complex scientific instruments to study the basic constituents of matter – the fundamental particles. The particles are accelerated close to the speed of light and made to collide together in order to split and reveal the sub-particles they are made of. The process gives clues about how the particles interact and provides insights into the fundamental laws of nature.

The instruments used at CERN are purpose-built particle accelerators, like the Large Hadron Collider (LHC), and detectors. Accelerators boost beams of particles to high energies before the beams are made to collide with each other or with stationary targets. Detectors observe and record the results of these collisions. Approximately 600 million times per second, particles collide within the Large Hadron Collider (LHC). Each collision generates particles that often decay in complex ways into even more particles. Electronic circuits record the passage of each particle through a detector as

a series of electronic signals, and send the data to the CERN Data Centre (DC) for digital reconstruction. The digitized summary is recorded as a "collision event". Physicists must then sift through the 30 petabytes or so of data produced annually to determine if the collisions have thrown up any interesting physics (CERN, 2016).

Founded in 1954, the CERN laboratory sits astride the Franco-Swiss border near Geneva. It was one of Europe's first joint ventures and now has 22 member states.



CERN's member states

4.2. CERN Communications Strategy

On June 28 2010, Rolf Heuer, the Ex Director General (DG) of CERN, decided to invite the InterAction Collaboration to conduct a peer review of the communication, outreach and education activities at the CERN Laboratory. The purpose was "to develop internal communication, local communication and communication at the political level".

On November of the same year, after extensive presentations and discussion with CERN management and staff, the review committee delivered its final report at a

closeout that included the Director General and multiple members of the senior management. In the report, between the many, there is a specific section regarding “Publications, Web and Social Media” where the findings, comments and recommendations of the review on the subject are summarized. It is important to consider all these parts in order to fully understand the situation in 2010 and the recommendations given.

Findings:

- CERN publishes a very wide range of printed materials:
 - The CERN Courier is a much-respected traditional publication and targeted for the scientific community.
 - The Bulletin is very heavy to produce:
 - It is a hybrid publication: half of the contents are printed, and everything is posted on the Web.
 - The Bulletin does produce high-quality contents that are suitable for internal and external readers.
 - CERN prepares brochures in many languages. These brochures contain appropriate contents for first time visitors.
 - The Communication Group publishes the CERN Annual Report each year, which is a significant time commitment for the communication staff.
- The Web specialists in the CERN Communication Group are introducing a new content management system with the help of appropriate expertise from the IT division.
- A new domain name to represent CERN’s global status is being discussed: cern.org.
- CERN does have a very visible presence on Twitter: approximately 208,000 followers, “No. 1 in the scientific category”
- CERN provides timely updates of developing events. For example, CERN used the Web and Twitter to post regular updates from the First Physics event.
- The CERN Courier has a different domain name: cerncourier.com, operated by the Institute of Physics.

Comments:

- The staff is highly motivated to produce high quality articles:
 - Translating the articles into multiple languages takes a large amount of human resources and tends to slow down the process.
 - The contents and exact wording require delicate proof readings, given the diplomatic and scientific situation of CERN.
- An effort to make Web content more easily manageable is commendable.
- The target audiences for each medium are mixed:
 - No regular publications for non-technical audiences or for the general public exist.
 - Brochure content could be more coherent with the Web.
 - The CERN Courier is not visible on the CERN website.
 - Many articles in the Bulletin are good and interesting to internal and external readers, including journalists.
- The Bulletin might be much more effective if articles are posted and updated daily.
- Website coherency is an issue
 - No one is in charge of editorial management of the Web, which leads to the lack of long term and daily decision-making.

Recommendations:

- Push ahead with the reorganisation of the CERN website.
 - Consider adding contract support to the CERN team to support this goal.
- Use the reorganisation of the website to strengthen the brand image of CERN and to restructure how target audiences reach news, bulletins and the CERN Courier.
- Synchronise the printing process with Web publishing.
- Establish a coherent mechanism to share information among staff writers and editors.
- Better utilise the power of the Web to reach the general public:
 - Daily CERN news, scientific news
- Implement a process to evaluate the real time impact of publications, Web and social media.

- Develop a social media policy and plan.

(InterAction Collaboration, 2010)

As we can see, the focus is on creating a plan to handle the news and delivering the information to the right audiences at the right time exploiting the Web. In fact, “the role of communications is to plan strategically, manage and sustain an organization’s relationship with key audiences, taking responsibility for the organization’s reputation and thereby helping the leadership to achieve its strategic and operational goals” (CERN IR-ECO group, 2016). As such, communications is an integral part of management responsibility. Mainly, people with management-level accountabilities at CERN are the Director General and the Directorate, which assists the DG and runs the laboratory through a structure of departments. The Directorate is formed by the chiefs of the following areas:

- ❖ Accelerators and Technology, currently Frédérick Bordry
- ❖ Research and Computing, currently Eckhard Elsen
- ❖ Finance and Human Resources, currently Martin Steinacher
- ❖ International Relations, currently Charlotte Warakaulle

Between the many groups forming the International Relations sector, the Communications group (DG-CO) is the most interesting for the purpose of the thesis. Its mandate is to generate public engagement in science, to produce and distribute information, to foster community building and to build support for CERN and its missions. The key audiences are:

- The general public – to foster engagement with scientific issues
- The scientific community – to provide information about CERN's activities
- Science and technology decision makers – to promote CERN's activities
- The CERN community – to provide information and build motivation
- Local communities – to provide information and promote events, including activities for local schools

(CERN IR-ECO group, 2016)

The recommendations from the Interactive Collaboration have been of great help devising where to focus resources and time. In support to the review from the

Interactive Collaboration, the CERN communications group (IR-ECO) worked on redacting a more exhaustive document that could provide valuable knowledge to the directorate in order to take the necessary steps to further improve communications at CERN.

Presented in October 2011, the “CERN communications strategy 2012-2016” was a draft document that outlined the strategic vision for official communications at CERN and was the product of many months of consultation with key stakeholders by the communications group. It had been informed by independent research conducted by external partners, as well as the peer review process led by the InterAction Collaboration. Its purpose was to generate and secure sustained political, financial and popular support for CERN’s scientific and societal missions from all its stakeholder groups. One of the tools used to achieve that goal is Social media. They are a component of the CERN communication strategy and are used in order to disseminate information.

By 2014, the CERN Computer Security Team wrote the guidelines (CERN Computer Security team, 2014) regarding the use of Social Media from its contributors (i.e. staff members, fellows, apprentices, associates, users or students), who comment professionally or privately about their activities at CERN using Social Media. They refer to the CERN Code of Conduct (CERN, 2015) and explain what is considered a proper behaviour on the network. It talks about all the aspects that can interest a user when going online, like the use of the CERN’s logo, confidential information, intellectual property, how to handle differences in opinion and gives contacts for everyone who needs to ask a question or need an advice about the Social Media policy. This way everyone can have a reference on how to use the Media.

“Through the DG-CO group CERN devotes about 0.25% of its resources to the organization communications function with further resources being deployed in other Departments and Groups. At a time of unprecedented, global reputational potential for CERN, the current resource levels and structures entail reputational risks for the Organisation. These risks can also have a direct influence on the Organization’s budget and ability to operate. The communications strategy defines the messaging architecture, maps out target audiences, and formulates key messages and proof points. It also proposes a structural alignment of CERN’s communications functions in order to mitigate the risks and, just as importantly, to ensure that CERN is fit to meet

the communications requirements of its stakeholders in the second decade of the 21st century”. (CERN IR-ECO group, 2016)

4.3. Social at CERN

There are three main strands to the social media strategy:

1. **Begin a journey**

Key messages are disseminated by repackaging CERN’s online content for the different social media channels. Most social media content contains links forwarding back to the CERN website, starting a journey for the user to find out more.

2. **Foster engagement**

CERN’s presence on social media channels fosters engagement in the public and helps to form an online community of stakeholders interested in the laboratory and its work. The level of engagement, through the shares, likes and comments, on CERN information is regularly monitored.

3. **Retain positive sentiment**

Social media is a way to reach the public and to monitor sentiment towards the organization. By keeping the sentiment positive and handling the negative sentiment constructively, by responding as appropriate to questions or concerns, CERN’s strong brand identity is retained.

(CERN, 2016)

CERN began using the social media channels in 2008. By August 2014, its Twitter account had more than a million followers keen to find out news about the organization. “During the 4 July 2012 Higgs announcement CERN’s live tweets reached journalists faster than the press release and helped contribute to worldwide coverage of the particle discovery and an October 2013 study (Lüfkens, 2013) cited CERN as the most effective international organization on Twitter” (CERN, 2016).

Currently active on Twitter, Facebook, YouTube, Google+, Instagram and LinkedIn, CERN is actively working on engaging its fans and followers, providing insights in the everyday work of thousands of scientists and answering to every question on their

work. The most important questions and answers are collected and displayed on the CERN Media and Press Relations webpage, like:

- Is the Large Hadron Collider dangerous?

No. Although powerful for an accelerator, the energy reached in the Large Hadron Collider (LHC) is modest by nature's standards. Cosmic rays – particles produced by events in outer space – collide with particles in the Earth's atmosphere at much greater energies than those of the LHC. These cosmic rays have been bombarding the Earth's atmosphere as well as other astronomical bodies since these bodies were formed, with no harmful consequences. These planets and stars have stayed intact despite these higher energy collisions over billions of years.

- What happened with the LHC in 2015 and what does CERN plan to do in 2016?

The Large Hadron Collider (LHC) restarted at a collision energy of 13 teraelectronvolts (TeV) in June 2015. Throughout September and October 2015, CERN gradually increased the number of collisions, while remaining at the same energy. In November, as with previous LHC runs, the machine run with lead ions instead of protons until mid-December when it had its winter technical stop. The most powerful collider in the world was switched back on in March 2016, followed by a period of tests. After a period of commissioning, the LHC experiments began taking physics data for 2016. Over the coming months, the LHC operators plan to increase the intensity of the beams so that the machine produces a larger number of collisions. This will enable physicists to have a better understanding of fundamental physics. Towards the end of year the machine will be set up for a four-week run colliding protons with lead ions.

- Why is the Higgs boson referred to as the God particle?

The Higgs boson is the linchpin of the Standard Model of particle physics but experimental physicists were not able to observe it until the arrival of the LHC, nearly 50 years after the particle was first postulated. Leon Lederman coined the term 'the God particle' in his popular 1993 book 'The God Particle: If the Universe Is the Answer, What is the Question?' written with Dick Teresi. In their book, Lederman and Teresi claim the nickname originated because the publisher would not allow them to call it 'the Goddamn Particle' – a name that reflected the difficulty in observing the elusive boson. The name caught on

through the media attention it attracted but is disliked by both clerics and scientists.

(CERN Media and Press Relations, 2016)

Usually every answer carries one or more hyperlinks to direct the users to other web pages on the CERN site that can better explain the subject.

CERN ✓
@CERN
CERN, the European Organization for Nuclear Research, is the world's largest particle physics lab and has #RestartLHC for #13TeV collisions. French: @CERN_FR
Geneva
home.cern
Joined June 2008
603 Photos and videos

TWEETS 2,494 FOLLOWING 378 FOLLOWERS 1.46M LIKES 106 LISTS 6

Tweets Tweets & replies Media

CERN @CERN · Aug 11
1968: staff working inside the new telephone exchange at CERN
cern.ch/go/L8x6 #TBT

CERN's Twitter page

Since CERN is a public non-profit organization and does not produce any product or service to sell, its objective is not to make advertising campaigns like Old Spice nor to ask for suggestions from the public on how to build particle accelerators. The goal is

the sharing of knowledge on new discoveries and increase awareness on what kind of experiments are being carried on and why.

Being on many public social networks lets CERN take care of its fans and followers, but does not help its employees. In order to give the proper support to its personnel, it has to adopt a different system.

One approach uses emails to exchange knowledge. Primarily used for communication, thanks to its flexibility it now also serves as to-do list, personal information management tool, archive, mechanism to foster coordination and collaboration among colleagues and source for assigning and delegating tasks (Mark & Voids, 2012). In summary, its usage goes well beyond what it was originally built for, making us checking email about 36 times an hour (Renaud, et al., 2006).

Looking at the research about social media, we find that another possible approach considers ESNs. They started taking an important role inside organizations, with Gartner stating that, by 2016 50% of large organizations will have ESN and 30% of these will be considered as essential as email is today (Gartner, 2013). This clearly indicates that the private sector realized the potential benefits and, at the same time, the current workforce is getting more and more adapted to ESN, setting them as a natural and essential tool at the workplace. We also know that ESNs allow solving problems faster and better as knowledge becomes available and searchable. Instead of seeking colleagues, one can go directly to the ESN and search for an answer. If no results are found, questions can be posted very quickly on a lightweight and informal way without causing interruptions. In a sense, ESNs can become an “internal Google” to find relevant answers. ESNs empower people, everyone has an equal voice, it encourages people to speak up giving them an opportunity to make meaningful contributions with their skills and ideas, and again leveraging innovation. It increases engagement by humanizing the way in which people work (Li, 2012), opposing to the classic and formal way to communicate provided by email.

We can say that an experimental study has shown that people deprived of email multitasked less had longer focus and even lower levels of stress (Mark & Voids, 2012). Using an ESN would allow us to keep the email channel for formal communication while deferring informational emails to pull-oriented channels like the ESN or RSS feeds. Email is also not adapted for all kind of communications. For

example, the case of the “reply-all syndrome”, when every employee willing to answer an email replies to everyone and causes everyone to receive a huge number of emails to read. It can cause all kind of reactions on members of a distribution list. At the same time, how many times opportunities for a constructive dialog or opinion were missed just because email is not suited? Knowledge residing inside mailboxes is locked. The amount of knowledge that is inaccessible to others and eventually deleted (when the mailbox owner leaves the organization) is unmeasurable. ESN’s by their nature, as open communication platforms where everyone has an equal voice, can clearly unlock opportunities for collaboration and capture existing knowledge for everyone.

Choosing between ESN and RSS technologies, we notice that RSS feed is a pull communication channel, thus the information is published and then gathered by the consumers, but it does not offer social interaction features – e.g. comments, sharing or networking. (De Sousa, et al., 2015) These reasons brought to the adoption of an ESN.

Another aspect to consider when approaching social media is how to build the network. Many questions arise, like: Do we want to build the enterprise social network from scratches or do we want to use one that is already online, like Trello? Do we need a specific one? In case we want to have a private ESN, do we have to build it or not? Do we have any reason to distrust the ones available on the market? In case we want to use a software like Oracle’s Beehive to set up our private social network, what software should we use between the many?

Like many agencies, CERN considers much of the information exchanged between employees as classified, so in this case it is not possible to choose an existing social network as platform for the CERN social network. This is because all the information exchanged on social networks like Trello has to be stored on their servers, thus violating confidentiality. In particular, sharing specifics about the developed projects could give the hackers and crackers, individuals with extensive computer knowledge whose purpose is to breach or bypass internet security or gain access to software without paying royalties, which daily try to violate the CERN network useful material to exploit the hypothetical vulnerabilities of the system. The best solution is to maintain the data inside CERN at all times using a private ESN. At this point, the choice is between building the ESN from zero or adopt one of the many management platforms available on the market. The management at CERN decided that the best choice would be to follow the latter option and selected SharePoint to be the best platform.

On March 2014, the pilot service bringing social networking capabilities for CERN people started (De Sousa, 2014). Available at the address <https://social.cern.ch>, *Social* introduces a lightweight communication channel, which aims to become a central tool for people to follow and interact with information and at the same time enrich existing communication channels with social features. The goal is to achieve the potential benefits of an ESN by proposing rich profiles along with microblogging features to communicate and share with CERN people.

Social main components are:

- Blogs and Wikis
- Personal homepages
- Calendaring and Task Managing features
- Communities and Workgroups
- Microblogs and Instant Messaging
- Social Network
- Suggestion System and Social Voting
- OneDrive for Business

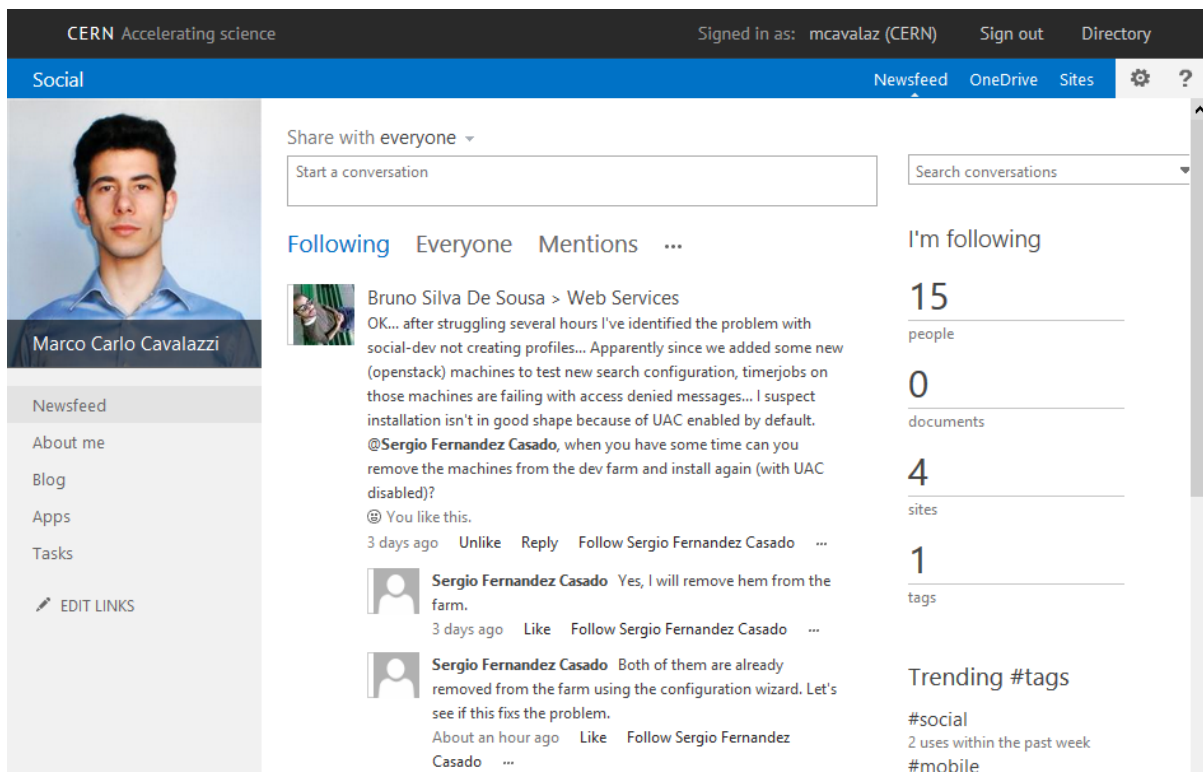
Blogs and Wikis are available for anyone who wants to share knowledge, create a personal website or start a wiki on a particular topic. Each of them can be accessed from Social and users are able to comment and share the content.

Profiles are pre-filled with basic contact information like phone number, email address and office location. Moreover, users can add their photo and information about themselves, like areas of knowledge, past projects, relevant experiences or interests that will help on the discovery of expertise on the network. This is extremely useful in this kind of environment because of the broad range of skills of the employees and it can help creating opportunities for new projects and collaborations. Social profiles also include the activities made on the platform (as messages shared with everyone), followed people, participation on communities or liked content. This makes profiles rich and constantly updated with the latest information depending on the person's activity. At the same time, activities automatically shared can be configured on the profile's settings in order to adapt to the preferences of the user. As a platform open to everyone at CERN, the variety of the content can be broad and thus not relevant to

all. The solution adopted makes it possible to follow content either by person or by hashtag, making the consumption easy and customizable. This allows the users to decide what content will be shown on their personal homepages after login.

Workgroups let team members express themselves on the work in progress, discuss about it and ask for suggestions from the colleagues while being sure that only the members will be able to see them and participate. This is useful in order to have a private section where only the messages regarding one project are posted and, at the same time, keep the information classified. Specific tools have been implemented for workgroups, like a calendar where the members can add their tasks and their deadlines, so that the rest of the team can program in advance the next steps.

Microblogging is at the heart of Social – known as the Newsfeed. Using the textbox available on the top of the homepage, one can very easily broadcast a message to everyone or a limited group of people. Hashtags can be used to add context and meaning to a post e.g. #chep2015, making it easier to find content or even get insights of current trends on what people is talking about. To catch the attention from someone the character @ appended by the person's name, can be used for mentions. Social interaction happens by adding comments to posts for conversation or giving "likes" for public validation and relevance.



Social Newsfeed example

SharePoint query suggestions are phrases that we want the search system to suggest to users as they start typing a query. They help make any research in the system quicker and help filling the blanks when one cannot remember the full name or there are more people with similar credentials. Social voting is another way to collect feedback from the network, where an opinion can be expressed using one of many options like the thumb up/thumb down system or the star system, where a user express their vote on a scale from one to five stars.

OneDrive for Business is the default document library in a user's "My Sites" section in SharePoint Server 2013 or SharePoint Online. The contents of the library can optionally be synchronized with one or more of the user's computers or devices. OneDrive for Business ensures that users' business files are stored in a central location. Storing business files in one location makes it easy for users to share and collaborate on documents. Using Office 365 for enterprises, one can also reduce the on-premises storage costs by moving the users' files to the cloud.

Documents

Welcome to your OneDrive for Business, the place to store, sync, and share your work. Documents are private until shared. [Learn more here.](#) [Dismiss](#)

[+](#) new document or drag files here

✓	Name	Modified	Sharing	Modified By
	Documents	... 43 minutes ago	🔒	■ Marco Carlo Cavalazzi
	Pictures	... 43 minutes ago	🔒	■ Marco Carlo Cavalazzi
	Presentations	... 43 minutes ago	🔒	■ Marco Carlo Cavalazzi
	Shared with Everyone	... 01 November, 2013	👥	■ Marco Carlo Cavalazzi

SharePoint OneDrive for Business page example

The features described above can also be used by CERN Service Accounts (CERN, 2015) making it possible for CERN services or events to be organized on Social to communicate with everyone. An example are the IT Lightning Talks (ITLT), which “are regular sessions of 5-10 short presentations (maximum 5 minutes long) on any topic related to computing technology or to the IT department with the goal to have a lightweight, informal and open communication channel, where everyone can share experience, seek advice, propose ideas, find others interested, brainstorm, team-up... and maybe create the next IT revolution!” (CERN, s.d.). ITLT use Social to publicize presentations and, at the same time, promote discussion on the presented topics.

Social also provides the grounds to build communities where people sharing similar interests can discuss about a topic, on a “questions and answers” oriented format, similar to discussion forums. A Social Community takes form on a dedicated website that can be created by CERN people using WebServices (CERN, s.d.). Those are easily customizable in terms of look and feel. Permissions access is very flexible, communities can be restricted to determined individuals, groups like CERN e-groups (CERN, s.d.) or even people outside CERN whom can sign-in with public service accounts like Facebook, Google or Microsoft account. There is a large set of features to make the administration of the communities effortless and adapted to the needs, like categories to organize discussions, badges, reputation settings for participants and options for moderation. (De Sousa, et al., 2015) A very popular example of a Social Community is the CERN Market (CERN, s.d.).



CERN Market

Search this site

Home

Classifieds list

Categories

Vehicles

Housing

Computing/TV/Video

Furniture/equipment

Baby items/babycare

Services/car sharing

Others

Members

Get updates

FAQ

Send feedback

Terms of use

Report abuse

The purpose of this site is to allow the CERN people to buy and sell personal objects, anything ranging from a house to a car to a set of snow chains. It cannot be used for commercial advertising. By accessing CERN Market and its related websites, services, applications or tools you are agreeing to the following [terms of use](#).

You must [sign in](#) with a valid account in order to read or post messages. CERN users should use their CERN account, others can use a public service account like Facebook, Google or Windows Live.

Join this community

What's happening

6637

members






926

discussions

935

replies

Top contributors

-  Daniel William Fitz
-  chabani malik
-  Fernando Carrasco
-  Rodolphe Rosol
-  Emin Ozturk

+ new classified

Recent What's hot My classifieds

- One-wheeled electric scooter, Moto Pogo**

Unique model, Canadian design.<http://www.gizmag.com/moto-pogo-one-whee...>
By Eduard Simioni | In [Vehicles](#) | Latest reply by Eduard Simioni | 10 minutes ago

5 replies
0 likes
- Selling new bedding 1 bed frame, 1 bed base, mattresses, pillows etc**

Pine wood bed frame 140x190, simple model needs assembling, €30, vacuum mo...
By jean west | In [Furniture/equipment](#) | 10 minutes ago

0 replies
0 likes
- Ducati Sport 750, black, year 2001**

Ducati Super-Sport 750, in excellent working condition and double disk Brembo fr...
By Eduard Simioni | In [Vehicles](#) | Latest reply by Eduard Simioni | 10 minutes ago

9 replies
1 like
- Solar powered mosquito killer for garden**

12 units, new in the box. Can work as zap killer or as light. Solar panel to rech...
By Eduard Simioni | In [Furniture/equipment](#) | 10 minutes ago

0 replies
0 likes
- Jaguar X-Type 2.0D British racing green, 5500 euro!**

160 000 km, water pump, steering pump and injection pumps replace last m...
By Eduard Simioni | In [Vehicles](#) | 11 minutes ago

0 replies
0 likes
- Large mosquito killer for garden house (220 Volt)**

Large 220 Volt lamp to kill mosquito. I have 5 available new in the box. Only 1...
By Eduard Simioni | In [Housing](#) | 13 minutes ago

0 replies
0 likes

CERN Market's Social Community page



Site Settings › Community Reputation Settings

- Home
- Documents
- Community
- Home
- Categories
- Members
- About
- Site Contents
- EDIT LINKS

Rating settings

Specify whether or not items in this list can be rated.

When you enable ratings, two fields are added to the content types available for this list and a rating control is added to the default view of the list or library. You can choose either "Likes" or "Star Ratings" as the way content is rated.

Allow items in this list to be rated?

Yes No

Which voting/rating experience you would like to enable for this list?

Likes Star Ratings

Member achievements point system

Within the community you can allow members to collect points based on their participations.

Enable member achievements point system

Specify the point values for the following activities

Creating a new post	<input type="text" value="10"/>
Replying to a post	<input type="text" value="10"/>
Member's post or reply gets liked or receives a rating of 4 or 5 stars	<input type="text" value="10"/>
Member's reply gets marked as 'Best Reply'	<input type="text" value="100"/>

Achievement level points

As members accumulate points, they can reach specific levels as milestones of achievement. Specify the number of points required for members to reach each achievement level.

Specify achievement levels

Level 1	More than	<input type="text" value="0"/>
Level 2	More than	<input type="text" value="100"/>
Level 3	More than	<input type="text" value="500"/>
Level 4	More than	<input type="text" value="2500"/>
Level 5	More than	<input type="text" value="10000"/>

Social Community Reputation and Badges settings page

Social Communities are directly connected with Social Newsfeed. Hashtags and mentions can also be used on community posts, thus if a followed hashtag is used it will be visible in the Newsfeed (or Yammer) page, even if the user is not aware that the community exists. This makes the Social Newsfeed (or Yammer) the single place where all the relevant content will be presented. At the same time, email based alerts are available on communities, as it can be useful for the owners or most active members. Email alerts can be triggered every time an activity occurs or be a summary scheduled either daily or weekly at specific times. Social Communities can be explored from a dedicated site exposing them by popularity.

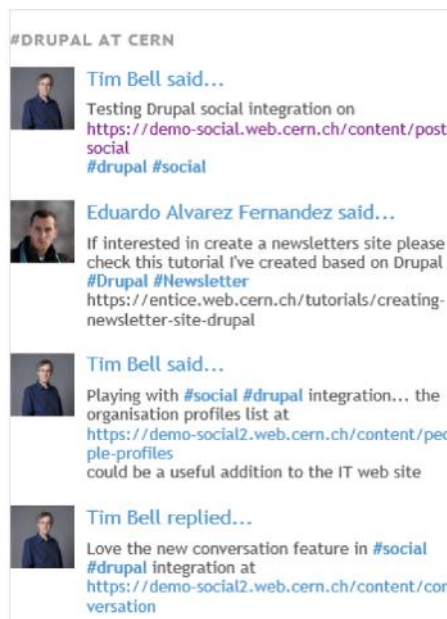
The integration of Social with the existing Web environment is very important in order to extend its usage to other contexts. Social is tightly integrated with the Collaboration Workspaces through the implementation of microblogging in the context of team collaborations for specific projects or services. This approach has shown very good

results in the context of the CERN WebServices team by increasing the communication flow between team members that are responsible for different platforms based on different technologies. This allowed to build better team spirit and create synergies between the functional elements of the service.

By means of a REST API (CERN, 2014) a set of Drupal modules (CERN, 2014) is available to introduce social features and content to existing CERN websites. It is possible to embed information about a specific profile, show all conversations with a known hashtag or list the picture and contact details of all members of a known department or group at CERN (De Sousa, et al., 2015).



Posts from a specific profile



Posts with a specific hashtag (#drupal)



All members of a specific group (IT/OIS)

4.3.1. System Architecture

An application's design is the set of activities aimed at identifying the best solution in order to meet the functional (and non-functional) objectives expected by the customer and the end user. These activities can be of various kinds, be carried out at different times and different ways depending on the approach used. In general, they help the architect and the development team to take important decisions, often of a structural nature. Design shares with programming the tendency to use an abstract

representation of the information and the logical sequence of development steps, but the level of detail in the two cases is different. The design builds a representation of the software that considers many aspects. It focuses on the structure of the system and the existing relations between the constituent parts, identifies the logical operations that must be carried out and identifies the way in which the system can interact with the outside world.

The result of the design is the definition of the system architecture, meaning the structural organization of the system itself, which includes its software components, the externally visible properties of each of them (the interfaces) and the relationships between the parties. In this case, "software component" means any entity forming part of a system, at different levels of detail and granularity, from the simple application module (for example, a class in an application based on the object-oriented paradigm) to the complex subsystem (for example, a DBMS or an LDAP server) (Bass, et al., 2003). In line with the definitions given, we can say that every software system has its own architecture since each system can be viewed as an aggregate of its constituent parts and the relationships between them. In the simplest and trivial cases a system is composed of a single constituent element. In these situations, the architecture is not complex and is probably uninteresting. In more complex cases the software system is formed by a series of heterogeneous subsystems that interoperate with each other using more or less complicated mechanisms for communication, working with huge amounts of data and users. In these cases, steps need to be taken in order to guarantee the security and reliability of the system.

Developed using SharePoint 2013, the architecture of Social consists on a highly available set up with mirrored database servers (MS SQL Always On technology), with one Application Server running SharePoint specific services and three web front-end servers. The main reason that brings to this choice is related with high-availability. Downtime can happen because of hardware/software failure or during maintenance operations, which can require services/servers restarts. Therefore, it is crucial to have roles redundancy at both logical and physical level.

The SharePoint service at CERN is considered as critical not because of Social Web App but because of the Collaboration Workspaces (<https://espace.cern.ch>). People at CERN can store procedures and documentation in those workspaces (e.g. LHC operations, schedules, Fire Brigade operations, IT services procedures, Pension Fund

management, etc.), thus downtime must be avoided. On the other hand, SharePoint specific services are for example the Search Service or the User Profiles Application. Those are services that do not need to run on the web front-ends. For those services we do not have redundancy because in case of downtime those do not prevent the web applications to work.

In addition, in order to be able to develop new tools and improve Social's functionalities, a development environment has been created. Identical to the production environment, this setting allows the developers to test out thoroughly the new modifications before the final release.

4.3.2. Deployment and Future Plans

Opposed to classic deployment where the technology is made available, training is provided and then people are expected to use it, the deployment of ESN is 80% cultural change and 20% about technology. It should not be seen as a one-department initiative, but as part of a broader change at the organization level. Many obstacles can be expected during the roll out of ESN. It is predictable to see adoption drop-off from users after a grace period of time. This is why it is important to keep users engaged and, in order to do that, take one step at the time and carefully follow a plan. Part of the plans for the future of Social involve feeding the Newsfeed with content by adding more sources with relevant information. Simple examples like posting daily CERN restaurant's menu or migrating existing classifieds site CERN Market to Social Community had very positive effects and added new features to existing services. Bidirectional integration is available for other CERN Web platforms to allow users to share context-based information directly to Social. It is important to highlight that programmatic interfaces are easy to use and allow both consuming and feeding new data.

New features are also under development like the Social Feed that consists on a topic-based microblog feed. This will allow, for example, lightweight departmental and private discussions and will make the conversation open to external people. One example is available at <https://cern.ch/chep2015>. We also expect that, in part, Social Feeds will replace the heavy usage of mailing lists when the purpose is mostly non-critical information exchange. Finally yet importantly, the development of

comprehensive usage analytics to measure the engagement of CERN people or success of communication campaigns is also part of the plans (De Sousa, et al., 2015).

5. Hands-on Social Development

In this chapter, we are going to introduce the Technical Student Programme and the work I have done on Social while attending it.

5.1. Technical Student Programme

CERN gives every student the possibility to participate in one of projects at the laboratory. It is the case of the Technical Student Programme. Aimed for undergraduate in Applied Physics, Engineering or Computing looking for a practical training period or a place to complete their final project, the programme allows a student to spend 4 to 12 months at CERN during the course of the studies (Bachelor or Master). An extension of up to a maximum of 14 months may be given.

The technical student programme gives the students a broader view of the world, thanks to the mix of people with their own customs and traditions xcoming from all around the world. It allows the students to take part in the research field and work on a specific project. Most importantly, it is possible express an opinion on the project and its development while being taken in serious consideration. Furthermore, one can attend several seminars on a large number of subjects, expanding one's knowledge of the field of study.

The main jobs I completed for this thesis have been:

- ❖ The implementation of the design for Social Mobile
- ❖ The creation of a Resource Planning Tool (RPT)
- ❖ The creation of the Social API

5.2. Social Mobile

SharePoint 2013 offered the possibility to set up the Social environment while providing many integrated features to support it. In addition, “for smartphone mobile

devices SharePoint Server 2013 provides a lightweight, contemporary view browsing experience for users to navigate and access document libraries, lists, wikis, and Web Parts” (Microsoft, 2013). In fact, SharePoint 2013 offers the “SharePoint Newsfeed” app to work with Social using Windows phones, iPhones and iPads. Unfortunately, it does not cover other environments, like Android. In this kind of OSs, without proper management, the website is always shown to the users in the same manner, meaning that a smartphone would receive a web page content thought for a wide screen. Shrunk to fit the screen of a smartphone, a page like that becomes very small, showing small fonts, links and images. Since Android has a market share much larger than all the other smartphone operating systems (OS) combined, it has been necessary to develop a solution to this problem.

5.2.1. Development

In order to have a mobile-optimised implementation of Social for those environments, SharePoint gives two main options: Responsive Web Design (RWD) and Device Channels. Before describing the chosen approach, we will briefly explore what the options have to offer and run through some of their benefits and limitations.

RWD relies on grid layouts, media queries and CSS to alter the display of a web site based on the width of the browser accessing that site. The main benefit of RWD is that no matter what the device width is, the site will display in its optimal (or nearest-to-optimal) form. The method is also search engine friendly. Search engines prefer a URL to always render the same HTML and utilising RWD achieves this. RWD, however, does not come without its faults. Its biggest drawback is what enables that SEO-friendly approach – the fact that the HTML served is the same. This means that while images or sections may be hidden in the CSS, the resources will still be served to the device which is not an optimal approach when targeting low-bandwidth mobile devices (Menezes, 2014). It also provides a less flexible approach to targeting a given device allowing less options for modifying the display. On top of this, RWD can often be costly to implement in terms of the number of tweaks and regressions required when targeting different browser widths. Todd Baginski and Michael Sherman, in their SharePoint Conference session SPC390, stated that they anecdotally noted that 25%

of developer time was spent dealing with such requirements (Baginski & Sherman, 2014).

SharePoint 2013's Device Channels rely instead on targeting the device accessing the page and serving up customised HTML based on the device which will be rendering the content. This ensures that one can provide a completely customised user experience depending on whether the user is on a desktop, tablet or phone. One could even go as far as serving up different content depending on the type of device (iPad, Surface, Android for instance). This option comes, therefore, with many advantages. It negates the main drawback of the responsive designs – HTML can be served in a manner that completely optimises the page load for the device being targeted. A desktop version could be highly interactive and visual, serving large images where the mobile experience could be lightweight for improved performance. The disadvantages of this approach mirror the advantages of the responsive approach. As has been previously stated, search engines prefer the HTML being rendered at a given URL to be identical. However, device channels serve up different content at the same URL for different devices (Menezes, 2014).

In the end, it has been decided to create a mobile version of the website using the Responsive Web Design, using CSS to get the responsiveness needed for the mobile environment. This way the webpage can be personalized for both tablets and smartphones and we are able to configure the outcome as needed. To work with the RWD it is essential to activate or deactivate the following site features as specified:

- Mobile View – Deactivate
- Wiki Page Home Page – Deactivate
- Publishing – Activate

It is important to know that the “mobile view” did not help Social to be available on mobiles and that it is necessary to deactivate that feature as a prerequisite for using the RWD approach. After this, in order to have the “Master page” feature under “Settings → Look and Feel” on SharePoint 2013 it is indispensable to first activate the “SharePoint Server Publishing Infrastructure”, and then the “SharePoint Server Publishing”, which can be found at:

Settings → Site Collection Features → SharePoint Server Publishing Infrastructure;
Settings → Site features → SharePoint Server Publishing.

It is now possible to work with the “Master page” feature and we can assign personalized CSS files to it. The use of a customized CSS (appendix A) added the responsiveness needed for Social. Many parts of the page can be hidden and others moved and enlarged in order for the users to have the core functionalities of Social well visible and usable.

To have a responsive design, the most important rule used in the custom CSS file is “@media”. Using a rule like the following it is possible to define the rules that have to be used when the screen of the device has a maximum width of 750px:

```
@media only screen and (max-device-width:750px), media only screen and (max-width:750px) {...}
```

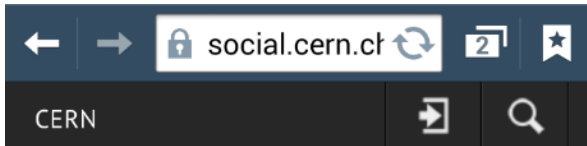
In order to have a responsive design that can adapt to smaller devices other arrangements have been done. The viewport is the user's visible area of a web page.

In this case, the viewport has been manually set, from the Master page, to:

```
<meta name="viewport" content="width=device-width, initial-scale=1.0">
```

This helps fixing the scale of the page and allowing it to have a consistent behaviour between devices while using the custom CSS file.

Special units have been used, like vw and vh instead of pt or px, to define for example the font-size, the width of the HTML divisions and the margins. Responsive units like vw allow us to have an element that adapts to the width of the device, while units like vh allow the element to adapt to its height. Exploiting this kind of CSS rules and measures it has been possible to adapt the normal view of the Social into a more mobile friendly one. After the development of the new website appearance, a documentation has been written for the users in order to explain how to have the best experience possible from any device.



About Tim Bell

IT/OIS

Email Tim.Bell@cern.ch
Phone +41227672695
Office 31 R-008

Tim Bell liked a post by Dan Noyes.
Yesterday at 17:13 Show conversation



Tim Bell

Playing with #social #drupal integration... the organisation profiles list at <https://demo-social2.web.cern.ch/content/people-profiles> could be a useful addition to the IT web site

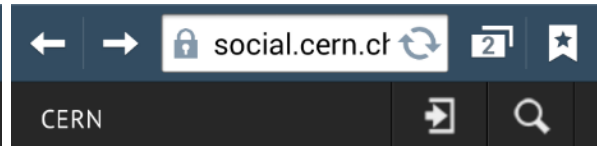
Bruno Silva De Sousa likes this.
Yesterday at 12:22 Like Reply Follow #drupal

Tim Bell liked a post by Marcos Fermin Lobo.
4 days ago Show conversation

Tim Bell liked a post by Marco Carlo Cavalazzi.
5 days ago Show conversation

Tim Bell replied to a post by Marco Carlo Cavalazzi.
5 days ago Show conversation

On the left, mobile Social design displaying Tim Bell's page; on the right, the feeds coming from the people and groups followed by the user



Share with everyone -

Following Everyone Mentions



Bruno Silva De Sousa > Web Services

Looks like webredir servers are using a Verisign certificate... is there any special reason or we can use *.cern.ch?
Yesterday at 09:44 Like Reply



Manuel Malo de Molina

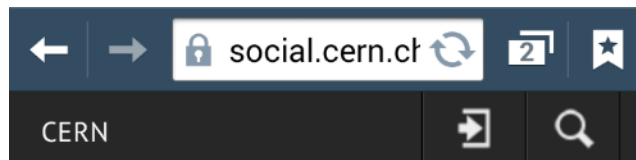
Maybe for "www.cern.ch", but not for "cern.ch" domain. We would need to renew this certificate before next month

3 hours ago 1 Like Follow Manuel Malo de Molina



Bruno Silva De Sousa > Web Services

OK... after struggling several hours I've identified the problem with social-dev not



About #social

[follow](#) this #tag

Relevant conversations



Dan Noyes said...

“ Great intro to #social by @Bruno Silva De Sousa - I guess it's up to all of us to make it a success now! ”

28 March
2 replies
7 likes



Bruno Silva De Sousa said...

“ test #social ”

28 March
0 replies
0 likes



Marco Carlo Cavalazzi said...

“ #SocialAPI is ready to go! Check it out at <https://demo-social.web.cern.ch/> #Drupal #social ”

18 July
0 replies
5 likes



Tim Bell said...

Mobile Social design displaying all the feeds having the #Social tag

5.2.2. Testing

The tests have first been carried out on an Android smartphone, but the resulting responsive design has been realized to work on any mobile device. The modifications have first taken place in the development environment, where it has been possible to test the behaviour of the page without bothering the production environment used by the people at CERN. Once satisfied with the adjustments, a number of colleagues helped to test the result in any sort of device, from Windows phones to Blackberries and iPads.

5.2.3. Problems Encountered and Limitations

During the development of the mobile version of Social many design problems arose, particularly when passing from testing the site on small devices like smartphones to testing it on tablets, which are bigger and demanded a revision of the design in order to adjust the given measures. Most of them regarded the outcome design, but some were more difficult to handle and demanded some study of SharePoint. An example regards the possibility for a user to click on a hashtag in a message in order to see visualized all the feeds containing the same tag. This became a problem because, even if they seem identical, the master page used when displaying the feeds containing the same hashtag is different from the main one used to display the feeds from followed people or groups, so the custom CSS file is applied only in the main one. This was the problematic piece of code used in the main master page to include the custom CSS file:

```
<SharePoint:AjaxDelta id="DeltaPlaceHolderAdditionalPageHead"
Container="false" runat="server">
    <asp:ContentPlaceHolder
id="PlaceHolderAdditionalPageHead" runat="server" />
    <link href="SiteAssets/CustomResponsive.css"
rel="stylesheet" type="text/css" ms-design-css-
conversion="no" />

    <SharePoint:DelegateControl runat="server"
ControlId="AdditionalPageHead"
AllowMultipleControls="true" />
    <asp:ContentPlaceHolder id="PlaceHolderBodyAreaClass"
runat="server" />
</SharePoint:AjaxDelta>
```

A very small correction in this case meant everything for SharePoint. Just adding a “/” at the beginning of the URL of the CSS file solved the issue, explaining to the environment that the CSS code had to be applied on every master page. This is the final code:

```

<SharePoint:AjaxDelta id="DeltaPlaceHolderAdditionalPageHead"
Container="false" runat="server">
  <asp:ContentPlaceHolder
id="PlaceHolderAdditionalPageHead" runat="server" />
  <link href="/SiteAssets/CustomResponsive.css"
rel="stylesheet" type="text/css" ms-design-css-
conversion="no" />

  <SharePoint:DelegateControl runat="server"
ControlId="AdditionalPageHead"
AllowMultipleControls="true" />
  <asp:ContentPlaceHolder id="PlaceHolderBodyAreaClass"
runat="server" />
</SharePoint:AjaxDelta>

```

The only limitation there is in this case is the absence of a link to the documents stored on SharePoint, so the user cannot access to the online files. It would be a useful improvement for the future.

5.3. Resource Planning Tool (RPT)

The engineers and physicists at CERN work every day to build, check and upkeep the high-end instruments used for the research in particle physics. To achieve that, they use a multitude of sophisticated pieces of equipment. For organizational purposes, the IT department has been asked to develop an online service that could allow the personnel to plan the use of the equipment. In particular, the requirements include:

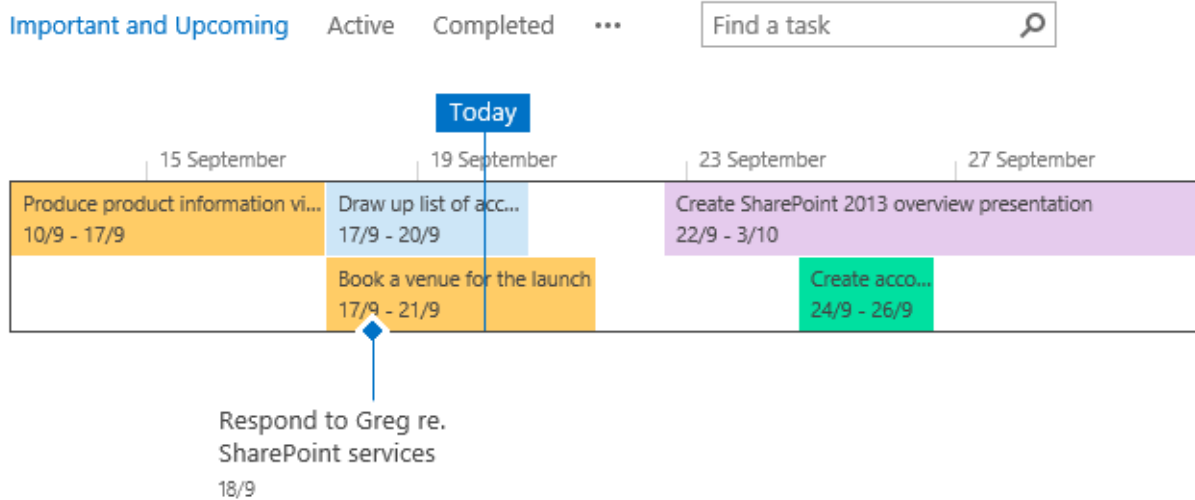
- ✓ The possibility to book one or more instruments and define a task for a set time period;
- ✓ The possibility to associate an activity with the tool needed for it, the project to which the activity is associated with and the employee that is going to carry out the task using the booked equipment;
- ✓ The possibility to group some instruments in the same category;

- ✓ The possibility, for the project managers, to assign a person to a certain activity and some equipment;
- ✓ The possibility to specify the amount of magnets to control;
- ✓ Having a visual representation of the schedule for each activity and each tool.

5.3.1. Development

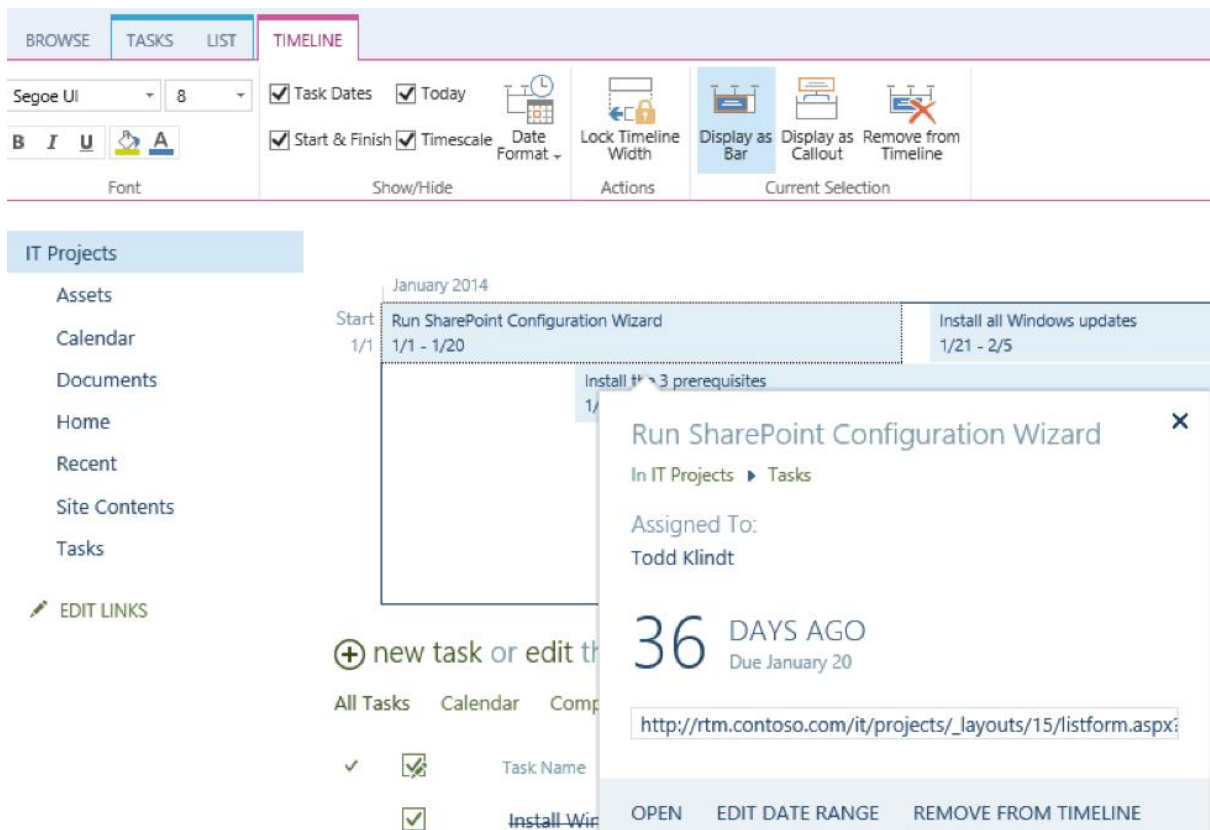
To develop the service it has been decided to integrate the new service in the social media at CERN. This way the social environment would expand becoming even more useful and anyone willing to add an entry to the Resource Planning Tool (RPT) would be able to do so without having to access to a secondary system.

Between the many features offered by SharePoint 2013 there is the possibility to create a scheduled list of tasks through the “Tasks” feature. The tasks and subtasks are then represented with start and end dates in a graphical timeline. The purpose of such a tool is to allow all the members of a workgroup to know their respective jobs and deadlines, in order to increase coordination and efficiency. Users have the ability to add or remove tasks or subtasks from the timeline and display them with various colours. The timeline allows the users to be easily be aware of the time period in which the tasks have to be executed. Each Task can be moved upside or downside and re-organized with a simple Drag&Drop action. Unfortunately, it is not possible to move the tasks laterally in order to adjust the task’s time frame. To do so, a user has to access the task’s “edit date range” page.



Standard SharePoint 2013 timeline example

When a task is selected, a “Timeline” tab appears in the ribbon at the top of the page. This tab lets a user configure the look and feel of each individual task and, as an example, make it be represented as a coloured bar or as a callout outside the timeline.



Standard SharePoint 2013 timeline example

Apparently, SharePoint does not offer a utility that can satisfy all of the requirements. Nevertheless, the Tasks list feature could be used as a starting point to develop all the functionalities asked. Two webpages will need to be modified. The first is the one provided to create a new task (creating a new task in the Tasks list on SharePoint) or modify one. Beyond the standard entries, this page will need to ask to the user extra information, including:

1. The name of the project
2. The name of the instrument
3. The available tools' categories, because a tool can be part of a group and the group name has to be visible
4. The activity of the project for which the equipment is needed
5. The amount of magnets to check (used in the particle accelerators).

Since it is important to insert the correct names for the project, the equipment category and the equipment booked, it has been decided to use drop-down menus to make it easier for the users to insert the right name choosing between all the possibilities already listed.

Columns	Type	Status
Project	Lookup	Required
Task Name	Single line of text	Optional
Start Date	Date and Time	Required
Due Date	Date and Time	Required
Equipment Category	Lookup	Required
Equipment Name	Lookup	Required
Assigned To	Person or Group	Optional
Amount of magnets	Number	Optional
Description	Multiple lines of text	Optional

The list of data asked when creating a new task

Moreover, the scientists required to be able to see the timeline while creating a new task. Since this is not possible when using the standard SharePoint interface, we need

to introduce a custom modification in order to retrieve the data from SharePoint and display a custom timeline above the “New Task” form. This way, a user can easily check when some equipment is available while defining the details of his or her task.

Talking about SharePoint 2013, we note that it implements Client Site Rendering (CSR), which is a term used to express a technology that allows the data to be transformed on the client side, rather than on the server. This means that it uses client-side technologies, such as HTML and JavaScript. This allows developers to style SharePoint elements using JavaScript, rather than having to write XSLT. In particular, the “clienttemplate.js” file is the SharePoint 2013 CSR framework core file intended to implement all JavaScript logic of CSR (Quinto, 2016). This means that in order to be able to communicate with SharePoint we need to wait until the core file is loaded in the page. To apply any modification to the webpage, the first function to call is “ExecuteOrDelayUntilScriptLoaded”, which is defined in SharePoint Foundation and executes the specified function if the specified file is loaded; otherwise, waits until the file is loaded before executing the function. Therefore, we need to call the function:

```
ExecuteOrDelayUntilScriptLoaded(registerRenderer,  
'clienttemplates.js');
```

Through this function, the webpage waits until the “clienttemplate.js” file is available and then uses the “registerRenderer” function to override the registered templates in the SPClientTemplates object and set our custom function to handle the visualization of the information on the page. The function is the following:

```
function registerRenderer()  
{  
    var ctxForm = {};  
    ctxForm.Templates = {};  
    ctxForm.OnPreRender = OnPreRenderDocItemTemplate;  
    ctxForm.OnPostRender = {};  
  
    SPClientTemplates.TemplateManager.RegisterTemplateOverrid  
es(ctxForm);  
}
```

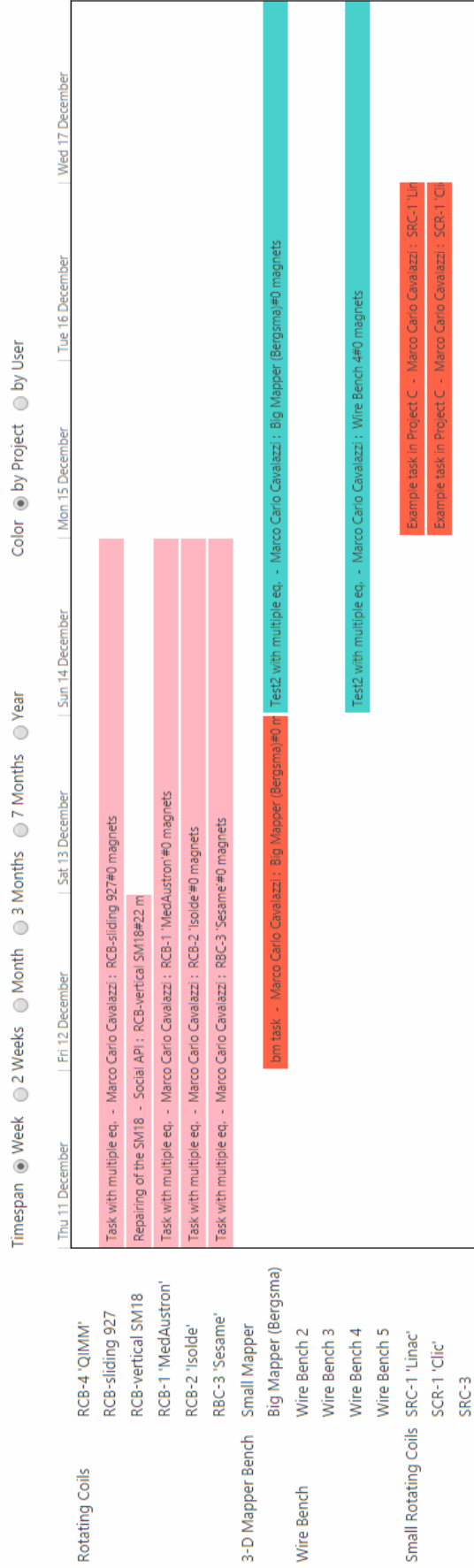
We can then retrieve the information regarding the Task lists creating a new instance of the SP.ClientContext object for our SharePoint site. To do this, we need to wait for another file to be loaded in the webpage: "sp.js", which "provides a subset of types and members in the Microsoft.SharePoint namespace for working with a top-level site and its lists or child Web sites" (Microsoft, n.d.). To be sure of that we can use the same function as before. Once the SP namespace is loaded, we can navigate the data and read any list we need, like "Equipment". We can continue to use this operation for each list and use the items retrieved to display a custom timeline with the equipment on the side, the dates horizontally and the activities, together with the amount of magnets used and the user to whom the task is assigned to, in coloured boxes inside the timeline. In this case, the Drag&Drop function has to be removed to prevent one activity to end up in the wrong row. We must avoid it, since it would mean that the user would need a different instrument to carry out that task.

At this point, we can personalize the timeline even more, displaying for example the tasks of one colour for each project or one colour for each user, so that a user could easily find his or her own activities. In addition, the time span of the timetable can be set as adjustable, spanning for example from few days to a year. Many details of the page have been handled this way, like the automatic refresh of the equipment's drop-down menu, where every time someone selects a category of tools only the instruments belonging to the chosen category become available in the equipment's drop-down menu.

To use the custom JS file (appendix B) we need to deactivate the "Minimal Download Strategy" feature for the site (from Settings → Manage site features). Then the file has to be added to the site using the "Miscellaneous" options of the existing SharePoint Web Part. The first thing to do is to load it on SharePoint, in a folder like "SiteAssets". Then, it has to be linked to the site using a URL like the following:
`~sitecollection/SiteAssets/CustomTimeline_newTaskForm.js`

Tasks

Example of the new custom timeline



Since we now have two dimensions in the timeline, dates and equipment, and SharePoint does not consider the possibility to have more than one dimension (the dates), we need to create a new control for the New Task page in order to properly check the availability of the chosen tool for the given dates before allowing the user to save the task. The function used to achieve this goal is:

```
function updateSaveButtonOnClickEvent() {
    var oldOnClickString =
        $("input[value='Save']").attr('onclick');
    var newOnClickString = 'if(consistencyCheckOnDates()){ ' +
        oldOnClickString + '; }else{alert("The selected equipment is
        not available in the chosen period. Please enter different
        dates.");}';

    // Updating the onclick event
    $("input[value='Save']").attr('onclick',
        newOnClickString);
}
```

Using this code, the “onclick” function of the “Save” button in the New Task form is modified to first, include a check on the chosen dates, and second, launch the old code to normally check the content of every input element compiled from the user. A new JS file can be created to separate the operations regarding the timeline from the ones regarding the New Task form. In this case, to include the JS file “newTaskForm.js” (appendix B) in the page we can create a new empty “Content Editor” Web Part and link the file as Content Link. The URL used will be like the following:

```
/timelineWebsite/SiteAssets/newTaskForm.js
```

This way all the requirements for this page are met.

We have discussed about adding the timeline to the “New Task” page. The second page that requires to be modified is the one containing the timeline and the list of tasks that have been uploaded. In order to show it in the main webpage as well we need to first hide the regular timeline unchecking the option in Web Part properties → Show

timeline. Then, we need to repeat the previous operation and add the JS file “customTimeline.js” (appendix B) to the Web Part through the JSLink in the Web Part properties → Miscellaneous with the link:

```
~sitecollection/SiteAssets/CustomTimeline.js
```

In the main webpage, beyond the timeline, there is shown a list of the saved tasks. The basic list view in SharePoint offers few elements and no classification. During the development of the RPT, the scientists asked to create a custom view for that list. Using a Custom List View, we can show the tasks in the main page differently. This way one can decide which property has to be displayed and where. In our case, it has been possible to create groups of tasks according to the project to which they belong. The new Custom List View is displayed in the next page.

As we can see, the tasks are grouped by project and the most important information is on the left while the less important one is shown on the right.

CustomTaskView All Tasks Calendar ...

Project	Task Name	Start Date	Due Date	Assigned To	Equipment Category	Equipment Name	ProjectColor	Amount of magnets
Project : Project A (4)								
Project A	Test task	... 20 November	26 November	Marco Carlo Cavalazzi	3-D Mapper Bench	AC mole #6	LightPink	0
Project A	Task on top	... 27 November	04 December	Marco Carlo Cavalazzi	Rotating Coils	RCB-4 'QIMM'	LightPink	0
Project A	Repairing of the SM18	... 02 December	Today	Social API	Rotating Coils	RCB-vertical SM18	LightPink	22
Project A	Task with multiple eq.	... Monday	Wednesday	Marco Carlo Cavalazzi	Rotating Coils	RCB-sliding 927; RCB-1 'MedAustron'; RCB-2 'Isolde'; RCB-3 'Sesame'	LightPink	0
Project : Project B (4)								
Project B	Task for proj B	... 20 November	20 November	Marco Carlo Cavalazzi	3-D Mapper Bench	Big Mapper (Bergsma)	MediumTurquoise	0
Project B	Rotating coils's second task	... 27 November	30 November	Marco Carlo Cavalazzi	Rotating Coils	RCB-sliding 927	MediumTurquoise	0
Project B	test after Task on top - Project B	... 05 December	6 days ago	Marco Carlo Cavalazzi	Rotating Coils	RCB-4 'QIMM'	MediumTurquoise	10
Project B	Test2 with multiple eq.	... Tuesday	Wednesday	Marco Carlo Cavalazzi	Wire Bench	Big Mapper (Bergsma); Wire Bench 4	MediumTurquoise	0
Project : Project C (2)								
Project C	Task Proj. C November	... 01 November	30 November	Marco Carlo Cavalazzi	Rotating Coils	RCB-vertical SM18	Red	0
Project C	Task January Proj. C	... 01 January, 2015	31 January, 2015	Marco Carlo Cavalazzi	Rotating Coils	RCB-4 'QIMM'	Red	0

Example of a Custom List View

5.3.2. Testing

The development of the Resource Planning Tool required many tests on different machines and browsers to be sure that the final outcome would be seen the same from every employee and every platform. The JS code has been tested thoroughly using several SharePoint accounts in order to check its behaviour in different conditions. Smoke tests have been carried out in order to seek for possible bugs while using inconsistent values, like using letters to tell the number of magnets required for an activity or requiring to use an equipment already booked for the selected dates.

5.3.3. Problems Encountered and Limitations

Few small problems were encountered while approaching the world of SharePoint's customization and have been the portability of the code for the many platforms used at CERN and browsers and finding the right information between the SharePoint's data structures, which did not present intuitive names.

5.4. Social API

In computer programming, an Application Programming Interface (API) is a set of subroutine definitions, protocols, and tools for building software and applications. An API specification can take many forms, but often include specifications for routines, data structures, object classes, variables, or remote calls.

Just as a graphical user interface (GUI) makes it easier for people to use programs, application programming interfaces make it easier for developers to use certain technologies in building applications. By abstracting the underlying implementation and only exposing objects or actions the developer needs, an API reduces the cognitive load on a programmer. While a graphical interface for an email client might provide a user with a button that performs all the steps for fetching and highlighting new emails, an API for file input/output might give the developer a function that copies a file from one location to another without requiring the developer to understand the file system operations occurring behind the scenes (Clarke, 2004).

APIs have many uses, depending on the context:

- Libraries and frameworks

In the first case, an API use can vary depending on the type of programming language involved. An API for a procedural language, such as Lua, could primarily consist of basic routines to execute code, manipulate data, or handle errors, while an API for an object-oriented language such as Java would provide a specification of classes and their class methods (de Figueiredo, et al., 1994) (Sintes, 2001). An API can also be related to a software framework. A framework can be based on several libraries implementing several APIs, but unlike the normal use of an API, the access to the behaviour built into the framework is mediated by extending its content with new classes plugged into the framework itself.

- Operating Systems

APIs that can specify the interface between an application and the operating system (Lewine, 1994). POSIX, for example, specifies a set of common APIs that aims to enable an application written for a POSIX conformant operating system to be compiled for another POSIX conformant operating system. Linux and Berkeley Software Distribution are examples of operating systems that implement the POSIX APIs (West & Dedrick, 2001).

- Remote APIs

Remote APIs allow developers to manipulate remote resources through protocols, specific standards for communication that allow different technologies to work together, regardless of language or platform. For example, the Java Database Connectivity API allows developers to query many different types of databases with the same set of functions, while the Java remote method invocation API uses the Java Remote Method Protocol to allow invocation of functions that operate remotely, but appear local to the developer (Bierhoff, 2009) (Wilson, 2001). Therefore, remote APIs are useful in maintaining the object abstraction in object-oriented programming; a method call, executed locally on a proxy object, invokes the corresponding method on

the remote object, using the remoting protocol, and acquires the result to be used locally as return value. A modification on the proxy object will also result in a corresponding modification on the remote object (Henning & Vinoski, 1999).

- Web APIs

Web APIs are the defined interfaces through which interactions happen between an enterprise and applications that use its assets. An API approach is an architectural approach that revolves around providing programmable interfaces to a set of services to different applications serving different types of consumers. (Rudrakshi, et al., 2014) When used in the context of web development, an API is typically defined as a set of Hypertext Transfer Protocol (HTTP) request messages, along with a definition of the structure of response messages, which is usually in an Extensible Markup Language (XML) or JavaScript Object Notation (JSON) format.

In the case of the Social API, it can be seen as a Web API that handles HTTP request messages and uses the JSON format in order to deliver information, but it can also be seen as a remote API, since it allows the user to read and manipulate remote resources. It does not only give an interface to interact with the resources, but implements all the functions needed to achieve that purpose.

The following list contains several examples of popular APIs:

1. Google Maps API

Google Maps API lets developers embed Google Maps on any webpage using a JavaScript interface. It is designed to work on both mobile and desktop devices.

2. Google YouTube APIs

These APIs let developers integrate YouTube videos and functionality into websites or applications. YouTube APIs include the YouTube Analytics API, YouTube Data API, YouTube Live Streaming API, YouTube Player APIs and others.

3. Flickr API

The Flickr API is used by developers to access the Flickr photo sharing community data.

4. Twitter APIs

Twitter offers two APIs. The REST API allows developers to access core Twitter data and the Search API provides methods for developers to interact with Twitter Search and trends data.

5. Amazon's Product Advertising API

Amazon's Product Advertising API gives developers access to Amazon's product selection and discovery functionality. It is commonly used to advertise Amazon products and monetize a website.

5.4.1. Development

The Social API (appendix C) is meant to be used from any of the CERN's employees to retrieve the feeds from the Social Network and add them to a webpage, which can be, for example, the experiment's webpage, like the one for ATLAS, the departmental webpage, like the one for Theoretical Physics, or even the CERN's homepage. It allows users to talk with the SharePoint server and show the data retrieved on any kind of webpage, taking also care of the design given to the information displayed. The Social API implements not only a superficial interface for the developers, but also the data structures and the functions needed to carry out the operations provided. The API has been developed using the agile approach. It has been built incrementally from the start, with each iteration producing a new functionality, instead of trying to deliver it all at once near the end. The functions implemented have been the sum of the efforts made to satisfy both the needs of the departments and the additional requests made from the employees, which added value to the final outcome.

The development of the Social API required the use of many technologies:

- JavaScript
- jQuery
- SharePoint 2013 REpresentational State Transfer (REST) interface
- HTML
- CSS

JavaScript is a high-level, dynamic, untyped, and interpreted programming language, standardized in the ECMAScript language specification (Flanagan, 2011). Alongside HTML and CSS, it is one of the three core technologies of World Wide Web content production; the majority of websites employ it and it is supported by all modern Web browsers without plug-ins. JavaScript is prototype-based with first-class functions, making it a multi-paradigm language, supporting object-oriented, imperative and functional programming styles (Ecma International, 2016). JavaScript is also used in environments that are not Web-based, such as PDF documents, site-specific browsers, and desktop widgets. Newer and faster JavaScript virtual machines (VMs) and platforms built upon them have also increased the popularity of JavaScript for server-side Web applications. On the client side, JavaScript has been traditionally implemented as an interpreted language, but more recent browsers perform just-in-time compilation. It is also used in game development, the creation of desktop and mobile applications, and server-side network programming with run-time environments such as Node.js.

jQuery is a fast, small, and feature-rich JavaScript library. It makes things like HTML document traversal and manipulation, event handling, animation, and Ajax much simpler with an easy-to-use API that works across a multitude of browsers. With a combination of versatility and extensibility, jQuery makes easier to both read and write JavaScript (jQuery Foundation, 2016). The reasons why we use jQuery over plain JavaScript can be explained with an example on how the two handle one simple operation. The purpose of the code in both cases will be to make a line of text in the HTML page change colour using its class name.

The JavaScript version:

```
var d = document.getElementsByClassName("goodbye");
var i;
for (i = 0; i < d.length; i++) {
    d[i].className = d[i].className + " selected";
}
```

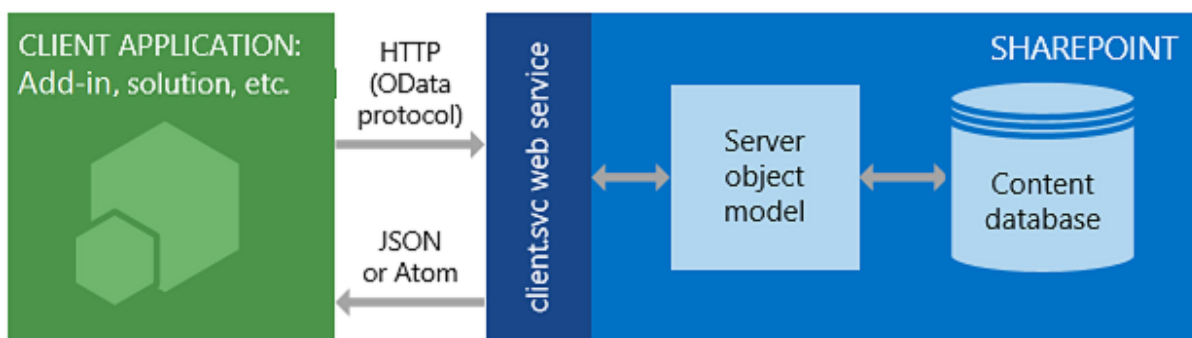
The jQuery version of the code obtaining the same result:

```
$(".goodbye").addClass("selected");
```

While JavaScript makes the code difficult to read even for a simple action like the one described, jQuery shortens the code while making it more readable. This, when writing

a high number of lines of code, allows us, and whoever will ever work on the API, to effortlessly understand how it works.

The SharePoint 2013 REpresentational State Transfer (REST) interface permits developers to interact remotely with SharePoint data by using any technology that supports REST web requests, like JavaScript. This means that developers can perform Create, Read, Update, and Delete (CRUD) operations from their SharePoint Add-ins, solutions, and client applications, using REST web technologies and standard Open Data Protocol (OData) syntax.



SharePoint REST service architecture

CRUD operations can be summarized in the following table:

Operation	Relative HTTP request	Keep in mind
Read a resource	GET	
Create or update a resource	POST	Use POST to create entities such as lists and sites. The SharePoint 2013 REST service supports sending POST commands that include object definitions to endpoints that represent collections. For POST operations, any properties that are not required are set to their default values. If you attempt to set a read-only property as part of a POST operation, the service returns an exception.
Update or insert a resource	PUT	Use PUT and MERGE operations to update existing SharePoint objects.

		<p>Any service endpoint that represents an object property set operation supports both PUT requests and MERGE requests.</p> <ul style="list-style-type: none"> • For MERGE requests, setting properties is optional; any properties that you do not explicitly set retain their current property. • For PUT requests, if you do not specify all required properties in object updates, the REST service returns an exception. In addition, any optional properties you do not explicitly set are set to their default properties.
Delete a resource	DELETE	<p>Use the HTTP DELETE command against the specific endpoint URL (Uniform Resource Locator) to delete the SharePoint object represented by that endpoint. In the case of recyclable objects, such as lists, files, and list items, this results in a Recycle operation.</p>

(Microsoft, 2015)

In order to be able to perform any CRUD operation we need to construct a fully qualified REST URL for the JavaScript calls to the SharePoint Server. To achieve that it is necessary to prepend `http://server/site/_api/` followed by the right URL fragment. In the table hereunder are few examples for URL endpoint fragments:

Description	URL endpoint	HTTP method	Body content
Retrieves the title of a list	<code>web/title</code>	GET	Not applicable
Retrieves all lists on a site	<code>lists</code>	GET	Not applicable
Retrieves a single 'list's metadata	<code>lists/getbytitle('listname')</code>	GET	Not applicable
Retrieves items within a list	<code>lists/getbytitle('listname')/items</code>	GET	Not applicable

Retrieves a specific property of a document. (In this case, the document title.)	<code>lists/getbytitle('listname')?select=Title</code>	GET	Not applicable
Creates a list	<code>lists</code>	POST	<pre>{ '_metadata':{'type':SP.List}, 'AllowContentTypes': true, 'BaseTemplate': 104, 'ContentTypesEnabled': true, 'Description': 'My list description', 'Title': 'RestTest' }</pre>
Adds an item to a list	<code>lists/getbytitle('listname')/items</code>	POST	<pre>{ '_metadata':{'type':SP.listnameListItem}, 'Title': 'MyItem' }</pre>

(Microsoft, 2015)

In the Social API, the URL to prepend is <https://social.cern.ch/api/>, followed by a specific URL fragment according to our needs that will specify the service we need and all the variables we need to pass to SharePoint and conditions we want to be applied on the results. An example is:

<https://social.cern.ch/api/search/query?querytext='tags:'+tagText+'&sourceid='459dd1b7-216f-4386-9709-287d5d22f568'&sortlist='created:1'>

We will explain the content of those URLs later on in this chapter.

HyperText Markup Language (HTML) is the standard markup language for creating web pages and web applications. With Cascading Style Sheets (CSS), and JavaScript, it forms a triad of cornerstone technologies for the World Wide Web (Flanagan, 2011). Web browsers receive HTML documents, where it is described the structure of the web page, from a server or from local storage and render them into multimedia web pages. HTML5 is the fifth major revision of the core language of the World Wide Web: the HTML. "In this version, new features are introduced to help Web application authors, new elements are introduced based on research into prevailing authoring practices, and special attention has been given to defining clear conformance criteria for user agents in an effort to improve interoperability." (W3C, 2014)

In the Social API many HTML5 features have been exploited, like the `<canvas>` element, to draw 2D and 3D tag clouds in the webpage, and the XMLHttpRequest API, which allows fetching synchronously or asynchronously parts of the page, allowing it to display dynamic content, varying according to the time, the situation on the ESN and user actions (Mozilla Developer Network, 2016).

Cascading Style Sheets (CSS) is a style sheet language used for describing the presentation of a document written in a markup language. The latter can be any kind of markup language. It can be HTML, but also XML(Extensible Markup Language) or XUL(XML User Interface Language) for example (Mozilla Developer Network, 2015).

CSS is designed primarily to enable the separation of document content from document presentation, including aspects such as the layout, colours, and fonts (W3C, 2010). This separation improves content accessibility, provides more flexibility and control in the specification of presentation characteristics, enables multiple HTML pages to share formatting by specifying the relevant CSS in a separate .css file and reduces complexity and repetition in the structural content. This separation of formatting and content makes it possible to present the same markup page in different styles for different rendering methods, such as on-screen or in print. It can also be used to display the web page differently depending on the screen size or device on

which it is being viewed. To display the information coming from Social correctly, a dedicated CSS file has been created to handle the design of the feeds displayed using the API. The Social API's CSS is included in the HTML page from the developers. Then, the API adds new elements to the webpage using specific attributes to let the CSS distinguish the feeds displayed using the API from the rest of the webpage.

The API has been implemented following the jQuery example. In order to use a function defined in the jQuery library a developer needs to write a prefix like "jQuery." or "\$." before calling the function. In the Social API, any function can be called using the prefix "socialAPI()", followed by the name of the function and its parameters. No data structure or function of the API is available outside it without the use of its prefix. This is an important feature for the API, since it will have to work in any kind of website, where probably many other CSS and JS files will be used. This way we manage to avoid conflicts with other resources, like having more than one JavaScript function with a particular name.

When using the Social API, we find ourselves having to incorporate data from various sources. We have to access data from the website domain and the SharePoint domain. For security reasons though, there are blocking mechanisms that prevent communication with more than one domain at a time. These security mechanisms are implemented in most browsers, making difficult or impossible to accomplish client-side calls across domains. In order to be able to work this way we need to exploit the SharePoint cross-domain library, which is a client-side library in the form of a JavaScript file (SP.RequestExecutor.js) that makes it possible to use Cross-Origin Resource Sharing (CORS) requests.

The most important functionalities provided from the API are:

Operation	Corresponding Social API function
Retrieve the feeds regarding only the people or groups followed by the user. If already displayed, refresh the content.	updateFollowedFeeds
Retrieve the feeds from Social coming from a specific profile. If already displayed, refresh the content.	updateFeedsFromProfile

Retrieve the feeds from Social regarding a specific hashtag, like “#HiggsBoson”. If already displayed, refresh the content.	updateFeedsWithSameHashtag
Retrieve the tags from Social and display them in a Tag Cloud.	loadTagCloud
Retrieve information about the people working in a specific department, group and section.	updateGroupInfo
Post a message on Social through the Social API.	postToMyFeeds

In order for the Social API to work properly, it is necessary to include few other files, together with the API, in the HTML webpage:

socialAPI.js	The Social API itself.
socialAPI.css	The CSS file that takes care of the design given to the information displayed using the Social API.
jquery.js	The jQuery library, needed from the API to work with jQuery.
SP.RequestExecutor.js	Needed to access SharePoint 2013 data using CORS requests.
tagcanvas.min.js	Used to display the 2D or 3D tag cloud in every browser except Internet Explorer.
excanvas.js	Necessary to display the 2D or 3D tag cloud when using Internet Explorer.

We will now present each function described above explaining the operations required for them to work. We will start from the “updateFollowedFeeds” function. Its input parameters are explained in the following table.

Name of the parameter	Description
whereToWrite	The name of the HTML section where the feeds will have to be written.
updateInterval	The update interval, in case we want the function to automatically refresh the feeds every tot seconds. In this case, the values as negative numbers, 0, “null” or “undefined” are used to express the will to avoid the auto-refresh behaviour.

numFeeds	The number of feeds to display in the page with this function.
flagDisplayReplies	This flag tells the function if the replies to the feeds have to be displayed or not.

In the function, some comments can be found to allow the reader to better understand the code. Each comment line starts with the double slash punctuation (`// comment`). A more throughout description of the code is given after the function. The function is the following:

```
function updateFollowedFeeds(whereToWrite, updateInterval,
numFeeds, flagDisplayReplies){
    // Sanitizing the input. encodeURI() is used instead of
    encodeURIComponent() when there has to be allowed the
    possibility to have hashtags.
    whereToWrite = encodeURI(whereToWrite);
    updateInterval = encodeURIComponent(updateInterval);
    if(numFeeds == null || numFeeds == undefined){ numFeeds =
0; }
    if(flagDisplayReplies == null ||
        flagDisplayReplies == undefined){ flagDisplayReplies =
true; }

    // Consistency checks.
    if( updateInterval===null ||
        updateInterval===undefined ||
        updateInterval<0){ updateInterval = 0; }

    if(whereToWrite[0] !== '#'){
        whereToWrite = '#' + whereToWrite;
    }

    // Saving the name of the parent HTML section of the feeds.
    var parentWhereToWrite = whereToWrite;
```

```

// Updating the global variables. These variables will be
necessary when the User needs to post a new message on
Social
followedFeedsWhereToWrite = whereToWrite;
followedFeedsUpdateInterval = updateInterval;
followedFeedsNumFeeds = numFeeds;
followedFeedsFlagDisplayReplies = flagDisplayReplies;

// Converting the time from seconds to milliseconds
if(updateInterval < 1000)
{ updateInterval = updateInterval*1000; }

var tempSectionID = whereToWrite.substring(1);
// Section check. If the HTML section is present in the
webpage we can move on, otherwise the function has to stop.
if( document.getElementById(tempSectionID) === null ){
    // Error. No HTML section found to display the feeds
    from Social.
    $(whereToWrite).html('<div class="feedsItem"> <p
id="text"> There has been a problem while
communicating with the server. <br/>Please try again
later refreshing the page. </p> </div>');
    console.log('Error while trying to write the followed
feeds. The section ID passed in input seems not to be
present in the webpage. ');
    return;
}

// If there are no feeds (there can be error message), so
we clean the section
if( $(whereToWrite + " .feedsItem").length == '' ||
$(whereToWrite + " .feedsItem").length == null ||
$(whereToWrite + " .feedsItem").length == undefined ){
    $(whereToWrite).html('');
}

```

```

        // Clearing the section of the feeds that the user is
        following.
    }

    // Adding a new wrapping section in the HTML page to make
    the SocialAPI's CSS file point only at this part of the
    webpage, in case many CSS files are used.
    if($(whereToWrite).html() == ''){
        // If there are no feeds in the section yet...
        var wrapSection = '<div class="socialAPIWrapClass">'+
            '<div
id="socialAPIFollowedFeeds">'+
            '</div>'+
            '</div>';
        $(whereToWrite).html(wrapSection);
    }

    // We need to re-authenticate on Social every time
    authenticateOnSocial();

    if(updateInterval > 0){
        clearInterval(followedFeedsUpdatesHandler);
        // Deleting the old automatic refresh of the feeds

        // Creating the new automatic refresh of the feeds.
        The followed feeds will be updated every
        "updateInterval" seconds. This variable will be
        necessary when the User needs to post a new message
        on Social.
        followedFeedsUpdatesHandler = setInterval(function(){
            updateFollowedFeeds(whereToWrite, updateInterval,
            numFeeds, flagDisplayReplies); }, updateInterval);
    }

```

```

// Updating the focused section that we will pass to
the following function the new ID, which is inside the
new wrapper div.
whereToWrite = '#socialAPIFollowedFeeds';
executeRestCallExtendedSix(myFeedManagerEndpoint +
"my/news", 'GET', null,
checkDataReceivedAndDisplayTheFeeds, onError,
whereToWrite, parentWhereToWrite, numFeeds, numFeeds,
flagDisplayReplies);
// Searches the feeds and passes them to the function
"checkDataReceivedAndDisplayTheFeeds()"
}else{
// If we reach this part of the code it means that the
function has to retrieve the feeds without the
automatic refresh.

if(followedFeedsUpdatesHandler != 'a'){
// If there is an active automatic update of the
feeds
clearInterval(followedFeedsUpdatesHandler);
// Deleting the old automatic refresh of the
feeds
followedFeedsUpdatesHandler = 'a';
}

whereToWrite = '#socialAPIFollowedFeeds'; //
Updating the focused section that we will pass to the
following function the new ID, which is inside the new
wrapper div.
executeRestCallExtendedSix(myFeedManagerEndpoint +
"my/news", 'GET', null,
checkDataReceivedAndDisplayTheFeeds, onError,
whereToWrite, parentWhereToWrite, numFeeds, numFeeds,
flagDisplayReplies);

```



```

        // searches the feeds and passes them to the function
        "checkDataReceivedAndDisplayTheFeeds()"
    }
}

```

As we can see, the function begins sanitizing the input and checking if some of the variables are in the right range of admissible values. Then, it checks if the HTML section passed in input actually exists in the page. This is important, since the function cannot put the feeds wherever it likes. There has to be a wrapper HTML section – like `<div id="feeds">` – that suggests the function where it is safe to display the feeds. If the HTML division is available, then a new `<div>` section is created inside with "socialAPIWrapClass" as its class. This way we know we are printing the feeds in the right spot and, at the same time, we make the CSS file included in the page, called "socialAPI.css" (appendix C), work only on the elements inside the new HTML section. After that, the user is authenticated on Social and the API executes a REST call to the SharePoint interface to ask for the feeds. The function "checkDataReceivedAndDisplayTheFeeds" is set to be executed in case of success, so it will be the one taking the data coming from SharePoint. The URL used for the request is formed by the variable "myFeedManagerEndpoint", which contains the address <https://social.cern.ch/api/social.feed/>, and the string "my/news", so we are using the SharePoint interface to the social feeds looking for the "news" of the user, which correspond to the most recent feeds regarding all the actors he or she follows on the network. If needed, we can use the "setInterval" function to set up the automatic refresh of the feeds. Otherwise, a button can be created to call the same function and have a manual update of the feeds.

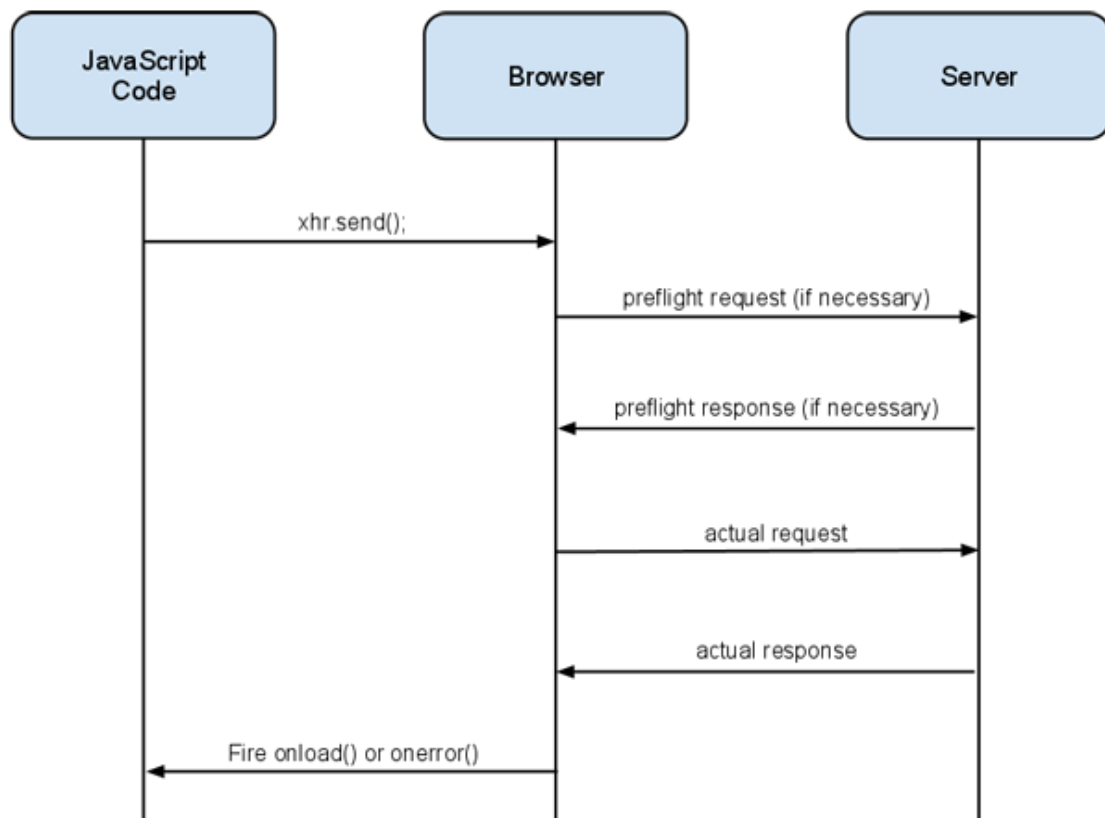
Once the data reach the "checkDataReceivedAndDisplayTheFeeds" function, the received JSON file is parsed and analysed, in order to understand if there has been any problem while communicating with the server, any internal error, if the request was not well formed, if there is a problem in the data received, or if the ESN service is down. If everything went well the feeds are passed to the "appendFeeds" function that will take care of adding them to the webpage.

The function described can be called using the following code:

```
socialAPI().updateFollowedFeeds("#feedsFollowed", 60);
```

The function displays the feeds followed from the current user and takes in input the number of seconds of interval after which these feeds will be automatically refreshed, if the number of seconds is 0 (zero) the feeds will not be automatically updated. With this approach, the logical level of the API can remain separated from its implementation.

Every CORS request made with JavaScript follows the same pattern, explained by the following diagram.



CORS flow of information (Hossain, 2013)

In the Social API, the CORS requests are handled by functions like “executeRestCallExtendedSix”, used in “updateFollowedFeeds”. The XMLHttpRequest element in it, necessary to proceed with the CORS request, is created in the function “createCORSRequest”. Both those functions can be found hereunder.

```

function executeRestCallExtendedSix(url, method, data, onSuccess,
onError, whereToWrite, id, numFeeds, numFeedsStillToGet, flag)
{

```

```

var xhr = createCORSRequest(method, url);

// Is the CORS request supported?
if (!xhr) {
    // If it is not supported we have to stop..
    throw new Error('CORS not supported');
}else{
    // ...otherwise we can continue.
    // Setting the function to be called in case of
    success.
    xhr.onload = function () {
        // passing the parameters and the results of the
        RESTcall to the function pointed by 'onSucc'.
        onSuccess(xhr.responseText,    whereToWrite,    id,
        numFeeds, numFeedsStillToGet, flag);
    };

    // Setting the function to be called in case of error.
    xhr.onerror = onError;

    // Sending the CORS request, with or without a body.
    if (data !== null && data !== undefined && data !==
    ''){
        xhr.send(data);
    }else{
        xhr.send();
    }
}
}

function createCORSRequest(method, url) {
    var xhr = new XMLHttpRequest();
    if ("withCredentials" in xhr) {

```

```

        // In case the XMLHttpRequest object has a
        "withCredentials" property, which only exists on
        XMLHttpRequest2 objects, we use this version of the
        "open" function.
        xhr.open(method, url, true);
    } else if (typeof XDomainRequest != "undefined") {
        // Otherwise, check if XDomainRequest is defined.
        // XDomainRequest only exists in Internet Explorer
        (IE).
        // It is IE's way of making CORS requests. In this
        case we need to use this other version of the "open"
        function.
        xhr = new XDomainRequest();
        xhr.open(method, url);
    } else {
        // Otherwise, CORS is not supported by the browser.
        xhr = null;
    }

    if(xhr != null){ // if the CORS is supported..
        // We prepare now the xhr element for the request,
        setting "withCredentials" to "true" and asking
        SharePoint for a JSON formatted file, with the
        "verbose" version of the reply, containing all the
        information we need.
        xhr.withCredentials = true;
        xhr.setRequestHeader("accept", "application/json;
        odata=verbose");
    }

    return xhr;
}

```

In particular, about the `.withCredentials` we know that standard CORS requests do not send or set any cookies by default. In order to include cookies as part of the request, the `XMLHttpRequest`'s `.withCredentials` property has to be set to `true`. In order for this to work, the server must also enable credentials by setting the `Access-Control-Allow-Credentials` response header to `"true"`. The `.withCredentials` property will include any cookies from the remote domain in the request, and it will also set any cookies from the remote domain. Note that these cookies still honour same-origin policies, so the JavaScript code cannot access the cookies from `document.cookie` or the response headers. They can only be controlled by the remote domain.

Before displaying any feed on the webpage, the message is checked to prevent any problem in the HTML page. This is because if some code is injected into a Social entry, it has to be properly sanitized when displayed in the webpage. The function written for this purpose exploits a regular expression to replace the following characters and thus sanitize a string from containing executable code: `<, >, ", ', ` , !, @, $, {, |, }, [,], \, ^`.

In order to do this, the characters are mapped with their correspondences and passed to the `"replace"` function. So, the input message is encoded to be correctly represented in an HTML webpage. To do this, we need the following function.

```
function myEscapeHTML(text) {
    var MAP = {
        '<': '&lt;', '>': '&gt;', '"': '&#34;', "'":
        '&#39;', '`': '&#96;', '!': '&#33;', '@': '&#64;',
        '$': '&#36;', '{': '&#123;', '|': '&#124;', '}':
        '&#125;', '[': '&#91;', ']': '&#93;', '\\': '&#92;',
        '^': '&#94;'
    };
    // Note that the single quote (') cannot be replaced with
    '&apos;', because it is not valid HTML 4. We have to use
    '&#39;'.

    // The message is returned to the calling function with
    the characters replaced by the strings defined in MAP.
```

```

return text.replace(/[\<\>\"'\\`!\@\$%\{\|\}\[\]\\\\^]/g,
function (a) { return MAP[a]; });
}

```

After this, we introduce the second function: “updateFeedsFromProfile”. The input parameters are explained in the following table.

Name of the parameter	Description
accountName	The name of the actor on Social (e.g. “Marco Carlo Cavalazzi” or “CERN Bulletin”)
whereToWrite	The name of the HTML section where the feeds will have to be written.
updateInterval	The update interval, in case we want the function to automatically refresh the feeds every tot seconds. In this case, the values as negative numbers, 0, “null” or “undefined” are used to express the will to avoid the auto-refresh behaviour.
numOfFeeds	The maximum number of feeds to display in the page with this function.
flagDisplayReplies	This flag tells the function whether we want the replies to the feeds to be shown or not.

In the function, some comments can be found to allow the reader to better understand the code. A more throughout description of the code is given after the function. The function is the following:

```

function updateFeedsFromProfile(accountName, whereToWrite,
updateInterval, numOfFeeds, flagDisplayReplies){
    // Consistency checks
    if( updateInterval===null || updateInterval===undefined ||
updateInterval<0)
    { updateInterval = 0; }
    if( numOfFeeds===null || numOfFeeds===undefined ||
numOfFeeds<0 || numOfFeeds>20){ numOfFeeds = 0; }
    if( flagDisplayReplies===null ||
flagDisplayReplies===undefined )
    { flagDisplayReplies = true; }
}

```

```

// Sanitizing the input (encodeURIComponent() is used instead of
encodeURIComponent() when there has to be allowed the
possibility to have hashtags.):
accountName = encodeURIComponent(accountName);
whereToWrite = encodeURIComponent(whereToWrite);
updateInterval = encodeURIComponent(updateInterval);
numOfFeeds = encodeURIComponent(numOfFeeds);

if(whereToWrite[0] !== '#'){
    whereToWrite = '#' + whereToWrite;
}

var tempSection = whereToWrite.substring(1); // It will be
the ID of the HTML section in which we want to write the
information without the hashtag as first character.
// Temporary variable used to store new elements inside
globalArrayOfProfiles.
var tempElement;
// Memorizing the main section
var parentWhereToWrite = whereToWrite;

// Converting the time from seconds to milliseconds
if(updateInterval < 1000){
    updateInterval = updateInterval*1000;
}

// Section check. If the HTML section is present in the
webpage we can move on, otherwise the function has to stop.
if( document.getElementById(tempSection) === null ){
    // Error. No HTML section found to display the
followed feeds on Social. Please add a <div
id="feedsFromProfile"> section.

```

```

        console.log('Error while trying to write the feeds
        from the specific profile. See function
        updateFeedsFromProfile().');
        $(whereToWrite).append('<div class="feedsItem"> <p
        id="text"> There has been a problem while
        communicating with the server. <br/>Please try again
        later refreshing the page. </p> </div>');
        return;
    }

    var innerWrap = "socialAPIFeedsFromProfile"+
    whereToWrite.substring(1) + accountName;
    var wrapSection = '<div class="socialAPIWrapClass">'+
        '<div id="'+ innerWrap +'>'+
        '</div>'+
        '</div>';

    // Adding a new wrapping section in the HTML page to make
    the SocialAPI's CSS file point only at this part of the
    webpage, in case many CSS files are used.
    if( $(whereToWrite +" .feedsItem").length == 0 &&
    $(whereToWrite +" .socialAPIWrapClass").length == 0 ){
    // If there is the HTML section and it is still empty...
        $(whereToWrite).html(wrapSection);
    }

    // We need to re-authenticate on Social every time
    authenticateOnSocial();

    try{
        if(updateInterval > 0){
            // The function "checkPresenceOfElement" returns
            -1 if the element is not in the array.

```



```

var                indexOfElement                =
checkPresenceOfElement(parentWhereToWrite,
globalArrayOfProfiles);
// If the element is inside the array, than we
have to clear the interval and pop the element
from the array before creating a new automatic
update interval.
if( indexOfElement > -1  &&  indexOfElement <
globalArrayOfProfiles.length ){
    // Stopping the old automatic refresh of the
    feeds
    clearInterval(globalArrayOfProfiles[indexOfElement].automaticUpdatesHandlersCode);

    // Creating the new automatic refresh of the
    feeds
    globalArrayOfProfiles[indexOfElement].automaticUpdatesHandlersCode
    =
    setInterval(function(){
    updateFeedsFromProfile(accountName,
parentWhereToWrite,                updateInterval,
numOfFeeds,                flagDisplayReplies);    },
updateInterval);
// The feeds will be updated every
"updateInterval" seconds

whereToWrite = '#' + innerWrap;

// Retrieving the feeds
executeRestCallExtendedSix(myFeedManagerEndpoint
+
"actor(item='cern\\"+accountName+"')/Feed"
,                'GET',                null,
checkDataReceivedAndDisplayTheFeeds,

```

```

        onError, whereToWrite, parentWhereToWrite,
        numOfFeeds,                                numOfFeeds,
        flagDisplayReplies); // getting the
        feeds and passing them to the function
        checkDataReceivedAndDisplayTheFeeds()
    }
else{
    // Creating the automatic refresh of the
feeds.

    var tempHandler = setInterval(function(){
        updateFeedsFromProfile(accountName,
        parentWhereToWrite,        updateInterval,
        numOfFeeds,        flagDisplayReplies);    },
        updateInterval);
    // The followed feeds will be updated every
    "updateInterval" seconds.
    tempElement = new updateObj(accountName,
    whereToWrite, tempHandler, updateInterval,
    numOfFeeds, flagDisplayReplies);
    globalArrayOfProfiles.push(tempElement);
    // Inserting the new element in the
    'globalArrayOfProfiles'.

    whereToWrite = '#' + innerWrap;

    // Retrieving the feeds.
    executeRestCallExtendedSix(myFeedManagerEn
    dpoint                                +
    "actor(item='cern\\"+accountName+"')/Feed"
    ,                                'GET',                                null,
    checkDataReceivedAndDisplayTheFeeds,
    onError, whereToWrite, parentWhereToWrite,
    numOfFeeds,                                numOfFeeds,
    flagDisplayReplies);

```

```

        // getting the feeds and passing them to the
        function
        checkDataReceivedAndDisplayTheFeeds()
    }
}
else{
    // If we are here it means that the function has
    to retrieve the feeds without automatically
    update them.

    var                indexOfElement                =
    checkPresenceOfElement(parentWhereToWrite,
    globalArrayOfProfiles);
    // The function "checkPresenceOfElement" returns
    -1 if the element is not in the array.

    // If the element is not inside the array, than
    we have to add it.
    if( indexOfElement === -1 ){
        tempElement = new updateObj(accountName,
        whereToWrite,    null,    0,    numOfFeeds,
        flagDisplayReplies);
        // Pushing the new element in the
        'globalArrayOfProfiles'.
        globalArrayOfProfiles.push(tempElement);
    }

    whereToWrite = '#' + innerWrap;

    // getting the feeds and passing them to the
    function. checkDataReceivedAndDisplayTheFeeds()
    executeRestCallExtendedSix(myFeedManagerEndpoin
    t + "actor(item='cern\\"+accountName+"')/Feed",
    'GET',                                null,

```

```

        checkDataReceivedAndDisplayTheFeeds,    onError,
        whereToWrite,    parentWhereToWrite,    numOfFeeds,
        numOfFeeds,    flagDisplayReplies);
    }
} catch(e) {
    $(whereToWrite).html('<div>There has been a problem
    while retrieving the feeds. <br/>Please try again
    later. </div>');
}
}
}

```

As we can see from the code, the function also begins sanitizing and controlling the input for possible problems. It checks the presence of the dedicated HTML section in the page, creates the new division (with class “socialAPIWrapClass”) for the feeds and authenticates on Social. At this point its behaviour becomes different from the previous function, because in this case we have to take into consideration the fact that more sections of the page can have this kind of feeds, each linked to a different profile on Social. In order to take care of every profile considered we need to store them in a global variable: “globalArrayOfProfiles”. This way we are able to take care of the automatic refresh of each section separately, launch or stop a single timer or all at once. After this, the CORS request is executed to get the feeds, using the URL [https://social.cern.ch/api/social.feed/actor\(item='cern\\'+accountName+\)/Feed](https://social.cern.ch/api/social.feed/actor(item='cern\\'+accountName+)/Feed). As we can see, the SharePoint interface used in this case requires a specific actor’s name to get his or her feeds and uses the variable “accountName” from the input parameters to take care of the issue. As in the previously examined function, in case of success the function “checkDataReceivedAndDisplayTheFeeds” takes the data in input and checks for exceptions, after which the “appendFeeds” function takes care of the displaying of the feeds. Even in this case we can set an automatic or manual update of the feeds.

The third function is “updateFeedsWithSameHashtag”. Its input parameters are explained in the following table.

Name of the parameter	Description
tag	The hashtag in which we are interested in (e.g. “#CERN”, “#Drupal” or “#LHC”)
whereToWrite	The name of the HTML section where the feeds will have to be displayed.
updateInterval	The update interval, in case we want the function to automatically refresh the feeds every tot seconds. In this case, the values as negative numbers, 0, “null” or “undefined” are used to express the will to avoid the auto-refresh behaviour.
numOfFeeds	The maximum number of feeds to display in the page with this function.
flagDisplayReplies	This flag tells the function whether we want the replies to the feeds to be shown or not.

In the function, some comments can be found to allow the reader to better understand the code. A more throughout description of the code is given after the function. The code is the following:

```
function updateFeedsWithSameHashtag(tag, whereToWrite,
updateInterval, numOfFeeds, flagDisplayReplies){
    // Consistency checks
    if( updateInterval===null || updateInterval===undefined ||
updateInterval<0){ updateInterval = 0; }
    if( numOfFeeds===null || numOfFeeds===undefined ||
numOfFeeds<0 || numOfFeeds>20){ numOfFeeds = 0; }
    if( flagDisplayReplies===null ||
flagDisplayReplies===undefined){ flagDisplayReplies =
true; }

    // Sanitizing the input (encodeURIComponent() is used instead of
encodeURIComponent() when there has to be allowed the
possibility to have hashtags.).
    // Splitting the input tags from one string to an array of
strings.
    var noSharpTagArray = tag.split(' ');
```

```

var noSharpTagString = '';
// This variable will be used for the innerWrap variable
only.
var noSpaceNoSharpTagString = '';
for(var i=0; i<noSharpTagArray.length; i++){
    if(noSharpTagArray[i][0] === '#'){
        noSharpTagArray[i] =
        noSharpTagArray[i].substring(1,
        noSharpTagArray[i].length);
    }
    noSpaceNoSharpTagString +=
encodeURIComponent(noSharpTagArray[i]); // Adding the tag only
    if( i < noSharpTagArray.length-1 ){
        // Adding the tag plus an empty space
        noSharpTagString +=
        encodeURIComponent(noSharpTagArray[i]) + ' ';
    }else{
        // Adding the last tag
        noSharpTagString +=
encodeURIComponent(noSharpTagArray[i]);
    }
}
whereToWrite = encodeURIComponent(whereToWrite);
updateInterval = encodeURIComponent(updateInterval);
numOfFeeds = encodeURIComponent(numOfFeeds);

if(whereToWrite[0] !== '#'){
    whereToWrite = '#' + whereToWrite;
}
if(document.getElementById(whereToWrite.substring(1))===n
ull){
    console.log("The HTML section appears not to exist.
    See updateFeedsWithSameHashtag() function.");
    return;
}

```

```

}

var tempSection = whereToWrite.substring(1,
whereToWrite.length);

if(updateInterval < 1000)
{
    // Converting the time from seconds to milliseconds
    updateInterval = updateInterval*1000;
}

var parentWhereToWrite = whereToWrite;

// Section's check. If the HTML section is present in the
webpage we can move on, otherwise the function has to stop.
if( document.getElementById(tempSection) === null ){
    // Error. No HTML section found to display the
followed feeds on Social. Please add a <div
id="feedsWithSameHashtag"> section.
    console.log('Error while trying to write the feeds
with the same hashtag. The section ID passed in input
is not present in the web page. ');
    return;
}

$(whereToWrite).html(''); // Clearing the feeds
displayed.

// We need to re-authenticate on Social every time
authenticateOnSocial();

// Adding a new wrapping section in the HTML page to make
the SocialAPI's CSS file point only at this part of the
webpage, in case many CSS files are used.

```

```

var innerWrap = "socialAPIFeedsWithSameHashtag"+
whereToWrite.substring(1, whereToWrite.length) +
noSpaceNoSharpTagString;
var wrapSection = '<div class="socialAPIWrapClass">'+
                  '<div id="' + innerWrap + '">'+
                  '</div>'+
                  '</div>';
$(whereToWrite).html(wrapSection);

try{
    // Activating the automatic refresh of the feeds
    if(updateInterval > 0){
        var index = checkPresenceOfElement(whereToWrite,
globalArrayOfHashtags); // Checking the
presence of the element inside the array.
// If the element is already present we can
simply modify the information about it.
if(index >= 0 && index <
globalArrayOfHashtags.length){
    clearInterval(globalArrayOfHashtags[index]
.automaticUpdatesHandlersCode); //
stopping the previously set automatic
updater
    var handlerCode = setInterval( function() {
updateFeedsWithSameHashtag(noSharpTagStrin
g, whereToWrite, updateInterval,
numOfFeeds, flagDisplayReplies); },
updateInterval);
// The feeds will be updated every
"updateInterval" seconds

    globalArrayOfHashtags[index].automaticUpda
tesHandlersCode = handlerCode;

```



```

        globalArrayOfHashtags[index].timeInterval
        = updateInterval;
    }else{
        // else: we have to add a new element to the
        array.
        var handlerCode = setInterval( function() {
            updateFeedsWithSameHashtag(noSharpTagString,
            whereToWrite, updateInterval,
            numOfFeeds, flagDisplayReplies); },
            updateInterval);
        // The feeds will be updated every
        "updateInterval" seconds.
        // Updating the global array for the timed
        updates.
        globalArrayOfHashtags.push(new
        updateObj(noSharpTagString, whereToWrite,
        handlerCode, updateInterval, numOfFeeds,
        flagDisplayReplies));
    }
}else{
    var index = checkPresenceOfElement(whereToWrite,
    globalArrayOfHashtags); // Checking the
    presence of the element inside the array
    // If the element is already present we can
    simply modify the information about it
    if(index === -1){
        // If the element is not yet in the array...
        globalArrayOfHashtags.push(new
        updateObj(noSharpTagString, whereToWrite,
        null, 0, numOfFeeds, flagDisplayReplies));
        // Adding the element to the array
    }
}
}

```

```

        // We want to write in the inner section.
        whereToWrite = "#" + innerWrap;
        // Retrieving the feeds with the same tag(s) and
        writing them in the section of the HTML page with
        ID='feedsWithSameHashtag'.
        retrieveFeedsWithSameTag(noSharpTagString,
        whereToWrite,      parentWhereToWrite,      numOfFeeds,
        flagDisplayReplies);
    }catch(e){
        $(whereToWrite).html('There has been an error while
        trying to write the feeds with the same hashtag.
        Please try again later.');
```


As in the other functions, here the code begins sanitizing and checking the input for possible consistency issues. It checks the presence of the dedicated HTML section in the webpage, authenticates on Social and creates the new division (with class “socialAPIWrapClass”) for the feeds. After this, the function is similar to the previous one, since we are facing the same possible issue: there can be more than one section with feeds with the same hashtag in the page and each can have feeds containing a different hashtag. As we can see from the code, the global variable “globalArrayOfHashtags” is used to solve the issue. It is important to notice that we can look for feeds with a variable number of hashtags in them. We can look for all the feeds with a single hashtag, like #IT, or multiple ones, like #SharePoint and #API. In the case with multiple hashtags, the API will retrieve all the feeds containing all of the specified hashtags in their text. Next, through the function “retrieveFeedsWithSameTag” it creates the CORS request for the SharePoint interface using the URL:

["https://social.cern.ch/ api/search/" + "query?querytext='tags:'+ tagText + '&sourceid='459dd1b7-216f-4386-9709-287d5d22f568'".](https://social.cern.ch/api/search/?query=querytext='tags:'+tagText+'&sourceid='459dd1b7-216f-4386-9709-287d5d22f568')


We can see that here is used the Search REST service. The tags, placed after the keyword “tags:”, will be a series of strings without any “#”, separated by a space. The “sourceid” used is a special string that lets SharePoint know that we are looking for

the tags from the Conversations on Social and that it should not look for tags in other areas like Collaboration Workspaces. The feeds are then passed to the “retrieveFeedsWithSameTagBodyFunction” function, which will take care of displaying the feeds on the page.

YOUR PROFILE ON SOCIAL




Bruno Silva De Sousa
 Can you heat up your brain for 60 hours for a good cause?
 This is a call for **#experts** in different technology fields to help 6 teams with participants coming from all over the world to work on **#humanitarian** challenges proposed by ICRC, UNHCR, OHCHR and many more. On 14th-16th of October we are organising a **#hackaton** in **#CERN - #IdeaSquare** and we are looking for people who'd like to apply their knowledge and skills and make a difference in the world. Register before 15th of June at www.theport.ch




THE Pori HUMANITARIAN HACKATHON
 14-16 OCTOBER 2016 @ CERN
 6 WEEKS PREPARATION
 3 DAYS ON-SITE PROTOTYPE DESIGN
 APPLICATION OPEN UNTIL 15 JUNE 2016
 WWW.THEPORT.CH

Tue Jun 07 2016 15:44:48 @ 4 Like Reply




Bruno Silva De Sousa
 17 percent of the time you spend at work is wasted on email **#productivity #mail**

Wed Apr 27 2016 14:14:23 @ 2 Like Reply



Sebastian Lopienski
 Certainly more :-)


Tue May 03 2016 16:23:36 @ 1 Like




Jose Benito Gonzalez Lopez
 17% only? I find it a rather conservative estimate :)

Fri Apr 29 2016 10:08:14 @ 2 Unlike


SOCIAL & DRUPAL TRENDS




Tim Bell said...
 Testing Drupal social integration on <https://demo-social.web.cern.ch/content/post-social>.
#drupal #social




Tim Bell said...
 Playing with **#social #drupal** integration... the organisation profiles list at <https://demo-social2.web.cern.ch/content/people-profiles> could be a useful addition to the IT web site




Tim Bell said...
#Social support now available in CERN **#drupal**
 ..
<http://entice.web.cern.ch/announcements/cern-social-module-published>



Eduardo Alvarez Fernandez said...
 Interested on integrate social.cern.ch content into your Drupal site? Check <https://demo-social.web.cern.ch> and soon will be available on the Drupal infrastructure **#drupal #social #socialAPI**



Marco Carlo Cavalazzi said...
#SocialAPI is ready to go! Check it out at <https://demo-social.web.cern.ch/>
#drupal #social



Dan Noyes said...
#social experts! Do tags have clean URLs? I'd like to link to **#social** tags from content on the CERN website with a link along the lines of 'Join the conversation on this topic'.

Examples of sidebars created on Drupal using the Social API.

On the left, it shows the feeds from the followed actors on Social; on the right, it shows the most recent feeds with both #Social and #Drupal hashtags (CERN, 2015)

The fourth function considered is “loadTagCloud”. Its input parameters are explained in the following table.

Name of the parameter	Description
whereToWrite	The ID of the HTML section where the feeds will have to be placed.
maxNumTags	The maximum number of tags to retrieve. Used to limit the tags displayed in the cloud canvas.
textColor	The colour of the text.
textBorderColor	In the 3D Tag Cloud it is possible to set the colour of the border of the text, which appears when the mouse is over the tag.
numDimensions	The number of dimensions to take into account (2= 2D Tag Canvas, 3=3D Tag Cloud).
weightFlag	It is possible to set this flag as “true” if we need the size of the text for each tag to be related to the frequency in which they are present on Social. This means that the more frequent a tag is used on the network the bigger will be displayed in the canvas (up to a maximum).
periodOfTime	The period of the time we are looking for ('lastDay', 'lastWeek', 'lastMonth', 'lastYear', 'allTime').

In the function, some comment lines can be found to allow the reader to better understand the code. A more throughout description of the code is given after the function:

```
function loadTagCloud(whereToWrite, maxNumTags, textColor,
textBorderColor, numDimensions, weightFlag, periodOfTime){
    // Section checks. If the HTML sections are presents in the
    webpage we can move on, otherwise the function has to stop.
    while(whereToWrite[0] === '#' && whereToWrite.length > 0){
        whereToWrite = whereToWrite.substring(1);
    }
    if( document.getElementById(whereToWrite) === null ){
        // Error. No HTML section found to display the
        followed feeds on Social. Please add a <div id="+
        whereToWrite +"> section.
```

```

        console.log('Error while trying to write the tags for
        the Tag Cloud. The HTML section appears not to exist.
        See the function loadTagCloud().');
        return;
    }

    var date = new Date(); // Reading today's date

    switch (periodOfTime) {
        case 'lastDay':
            date.setDate(date.getDate()-1);
            // Going back one day.
            break;
        case 'lastWeek':
            date.setDate(date.getDate()-7);
            // Going back one week.
            break;
        case 'lastMonth':
            date.setDate(date.getDate()-30);
            // Going back one month.
            break;
        case 'lastYear':
            date.setDate(date.getDate()-365);
            // Going back one year.
            break;
        case 'allTime':
            date = null;
            // We will retrieve all the tags ever used (with
            their number of occurrences).
            break;
        default:
            date = null;
            // We will retrieve all the tags ever used (with
            their number of occurrences).
    }

```

```

    }

    // This will be the URL used to retrieve the tags from
    Social.
    var querySiteToGetTheTags;
    if(date === null){
        querySiteToGetTheTags = querySiteToGetAllTheTags;
    }else{
        var day, month;

        month = date.getMonth() + 1;
        if(month < 10){ month = '0' + month; }
        // We want the 'month' string to have always two chars.

        day = date.getDate();
        if(day < 10){ day = '0' + day; }
        // We want the 'day' string to have always two chars.

        querySiteToGetTheTags = searchRestService +
        "query?querytext='ContentTypeId:0x01FD* write>=\""+
        date.getFullYear() + "-" + month + "-" + day + "
        00:00:01Z\" -ContentClass=urn:content-
        class:SPSPeople'&refiners='Tags'";
    }

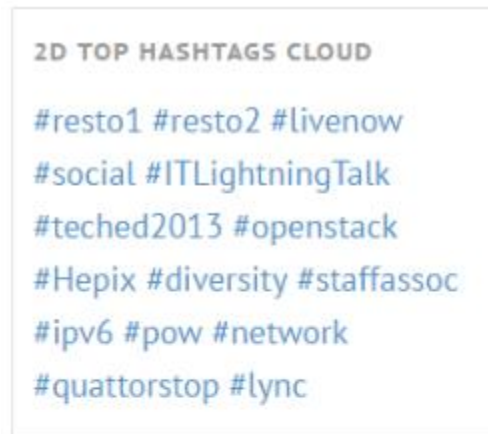
    executeRestCallExtendedSeven(querySiteToGetTheTags,
    'GET', null, drawUserTagsInCanvas, onError, whereToWrite,
    maxNumTags, textColor, textBorderColor, numDimensions,
    weightFlag);
    // Getting all the tags from Social and passing them to the
    function drawUserTagsInCanvas().
}

```

A the beginning of the function we can find the controls on the input, in particular on the HTML section. We then create the “date” variable, which will tell SharePoint from what point in time we need to examine the Social tags. The CORS request is then launched using the URL:

[“https://social.cern.ch/ api/search/” + “query?querytext='ContentTypeId:0x01FD*write>=\'"+ date.getFullYear\(\) +"-"+ month +"-"+ day +" 00:00:01Z\' - ContentClass=urn:content-class:SPSPeople'&refiners='Tags'”](https://social.cern.ch/api/search/?query=querytext='ContentTypeId:0x01FD*write>=\')

Even in this case the Search REST service is exploited. In the URL, the string “ContentTypeId:0x01FD*” specifically calls for trending tags, while the “write” keyword is used to get everything written from the date given in input to the present. Then the time string “00:00:01Z” is passed, stating that we need the feeds written from the first second of the given day. “ContentClass” is a mandatory property for SharePoint 2013 for this kind of requests and states that the content we are looking for does not regard the People on Social (Microsoft, 2015). Instead, we can read from the last part, “refiners='Tags'”, that our interest is for hashtags only. The feeds are then passed to the “drawUserTagsInCanvas” function, which will take the data from the server and take care of displaying the feeds on the canvas element in the HTML page.



Social API use example (CERN, 2015)

We will now present each function described above explaining the operations required for them to work. We will start from the “updateGroupInfo” function. Its input parameters are explained in the following table.

Name of the parameter	Description
whereToWrite	The name of the HTML section where the feeds will have to be written.
updateInterval	The update interval, in case we want the function to automatically refresh the feeds every tot seconds. In this case, the values as negative numbers, 0, “null” or “undefined” are used to express the will to avoid the auto-refresh behaviour.
numFeeds	The number of feeds to display in the page with this function.

flagDisplayReplies

This flag tells the function if the replies to the feeds have to be displayed or not.

In the function, some comments can be found to allow the reader to better understand the code. A more throughout description of the code is given after the function. The function is the following:

```
function updateGroupInfo(wheretowrite, department, group,
section, imageFlag, departmentFlag, numFeeds){
    // Sanitizing the input (encodeURIComponent() is used instead of
encodeURIComponent() when there has to be allowed the
possibility to have hashtags.):
wheretowrite = encodeURIComponent(wheretowrite);
department = encodeURIComponent(department);
group = encodeURIComponent(group);
section = encodeURIComponent(section);

// Resetting the global variable
numOfElementsAlreadyDisplayed = 0;

if(department === null || department === 'null' ||
department === undefined || department === '' ||
department.length < 1){
    $('#content').html('<div class="feedsItem"> <p
id="text"> There has been a problem while retrieving
the feeds. Please try again later. </p> </div>');
    return;
}
var groupString;
if(group == null || group == 'null' || group == undefined
|| group == ''){
    groupString = '';
}else{
    if(typeof(group) === 'string' && group.length > 1
&& group.length < 20){
```

```

        groupString = '/' + group;
    }else{
        groupString = '';
    }
}
var sectionString;
if(section == null || section == 'null' || section ==
undefined || section == ''){
    sectionString = '';
}else{
    if(typeof(section) === 'string' && section.length >
1 && section.length < 20){
        sectionString = ' Section:' + section;
    }else{
        sectionString = '';
    }
}

if(wheretowrite[0] !== '#'){
    wheretowrite = '#' + wheretowrite;
}

var tempElement;
var parentWheretowrite = wheretowrite;

// Clearing the section of the feeds I am following
$(wheretowrite).html('');

var tempSectionID = wheretowrite.substring(1,
wheretowrite.length);
// Section check. If the HTML section is present in the
webpage we can move on, otherwise the function has to stop.
if( document.getElementById(tempSectionID) === null ){

```

```

// Error. No HTML section found to display the
followed feeds on Social. Please add a <div
id="feedsFollowed"> section.
$(whereToWrite).append('<div class="feedsItem"> <p
id="text"> There has been a problem while
communicating with the server. <br/>Please try again
later. </p> </div>');
console.log('Error while trying to write the followed
feeds. The section ID passed in input seems not to be
present in the webpage.');
```

```

return;
}

// Adding a new wrapping section in the HTML page to make
the SocialAPI's CSS file point only at this part of the
webpage, in case many CSS files are used.
var wrapSection = '<div class="socialAPIWrapClass">'+
    '<div id="socialAPIIDepartment' +
    tempSectionID + "'>'+
    '</div>'+
    '</div>';
$(whereToWrite).append(wrapSection);








whereToWrite = '#socialAPIIDepartment' + tempSectionID;

var searchForGroupInfoSite = searchRestService +
"query?querytext='department:" + department + groupString +
sectionString+"'&sourceid='B09A7990-05EA-4AF9-81EF-
EDFAB16C4E31'";
// In the variable 'searchForGroupInfoSite', the code:
// sourceid='B09A7990-05EA-4AF9-81EF-EDFAB16C4E31'
```

```
// tells the Server that we are looking for People (possible
search options: Everything, People, Conversations,
Videos).

try{
    // Launching the function that executes the CORS
request.
    executeRestCallExtendedSeven(searchForGroupInfoSite,
    'GET', null, updateGroupInfoBodyFunction, onError,
whereToWrite, department, group, section, imageFlag,
departmentFlag, numFeeds);
}
catch(err){
    errorHandlerFunction(11, "There was a problem while
communicating with the Server.\nPlease try again
later.");
}
}
```

IT / CDA / FW TEAM

	MARIO REY REGULEZ IT/CDA mario.rey@cern.ch
	ADRIAN MONNICH IT/CDA adrian.moennich@cern.ch
	BRUNO SILVA DE SOUSA IT/CDA bruno.sousa@cern.ch
	VINCENT NICOLAS BIPPUS IT/CDA vincent.bippus@cern.ch
	DJILALI MAMOUZI IT/CDA Djilali.Mamouzi@cern.ch
	SEBASTIEN DELLABELLA IT/CDA Sebastien.Dellabella@cern.ch
	ALVARO GONZALEZ ALVAREZ IT/CDA alvaro.gonzalez.alvarez@cern.ch

Example of the Social API retrieving information on the People working in the IT/CDA/FW section (written as department/group/section) (CERN, 2015)

At this point, we will look at the “postToMyFeeds” function. This function has the purpose to post a new feed on Social using the user’s credentials. Its input parameters are explained hereunder.

Name of the parameter	Description
message	The message to post to Social.
inputFunction	The function that we need to launch as soon as the message is posted successfully.

A more throughout description of the code is given after the function presented here:

```
function postToMyFeeds(message, inputFunction) {

    // If no message is given in input... [ the function is
    // called as "postToMyFeeds();" ]
    if( typeof(message) !== "string" || message === null ||
    message === "" || message === undefined ){
        // Reading the message from the page.
        message =
        document.getElementById("textareaPostNewFeed").value
        ;
        // If the message is still null
        if(message === null || message === "" || message ===
        undefined){
            console.log("Error: No message passed in input.
            The new feed can not be created.");
            $('#nextToPostButton').html("<i> &nbsp; Please
            write some text first.</i>");
            // The "setTimeout" function will hide the
            message after 3 seconds.
            setTimeout("socialAPI().clearMessageToTheUser('
            nextToPostButton');", 3000);
            return;
        }
        else{
            // Removing any text eventually present in this
            section of the HTML file.
            $('#nextToPostButton').html("");
        }
    }

    if( inputFunction == null || inputFunction == undefined
    ){
        inputFunction = function(flag){
```

```

        if(flag){ alert("Message posted."); }
        else{ alert("There has been a problem while
posting the message. Please try again later.");
        }
    }
}

// Calling the function that will read the text from the
<p> HTML section and post it online.
executeRestCallExtended(formDigestUrl, "POST", null,
postMessage, onError, message, inputFunction);
}

```

This is only the first of the two CORS requests needed to upload the post online. It is necessary in order to get the “formDigest” value and be able to set the “X-RequestDigest” property in the RequestHeader of the second XMLHttpRequest to the formDigest value, like this:

```
xhr.setRequestHeader("X-RequestDigest", formDigest);
```

So, if the CORS request is successful, the “postMessage” function is called. It first reads the message written in the page. Checks if there actually is any message and reads the data from SharePoint to obtain the formDigest. Next, we need to consider that if there is any hashtag or web link in the message and we want them to be recognised as such, instead of simple text, then we need to substitute them with tokens, like {0} or {1}, and store their real values in a different location. Later, we will add their real values in the JSON file for the SharePoint service.

If there are already some parts of the text that could be exchanged from SharePoint as tokens, we need to modify those parts in order to avoid problems like unintended repetitions of tags in the post. Using the regular expression (RE) `/\{ [0-9]+\}/` we can identify the presence of any token-like string in the message. Those “fake tokens” are modified simply adding a blank space between the parentheses and the content, so that SharePoint will not see those as tokens and, at the same time, the message will still be clear.

At the same time, using JSON as the format to exchange data, it is important to avoid having more than one open or closed curly bracket at a time, like “{”, because it would cause problems while formatting the message for SharePoint. Even this issue can be solved adding a simple space between the parentheses.

At this point the function looks for any hashtag or link in the page, so that they can be correctly reported to SharePoint and be represented as such on Social. The words in the text are split into an array using the space as key character and examined one by one. Looking for hashtags, we just need to look for the “#” symbol, followed by a series of simple numbers or letters. If found, the hashtag is saved in a list and replaced by a token. In order to acknowledge the presence of web links we can use a regular expression that checks for every possible combination of letters, symbols and numbers that could represent a web link. The RE used is:

```
/^((ftp|https?):\/\/)?(www\.)?([\w\-\ ]{2,})([\.\ ][\w\-\ ]{2,})*([\.\ ][a-z]{2,})+([\\/][\w\+\-\ \?\. \&\%=\#\:\;\ (\)\ \~]{2,})*[\\/]?/i
```

Next, the JSON file is formed respecting the following structure (expressed in JavaScript), which includes the metadata needed from SharePoint to recognize its content:

```
" { 'restCreationData':{ " +
"   '__metadata':{
'type':'SP.Social.SocialRestPostCreationData'}, " +
"   'ID': null, " +
"   'creationData':{ " +
"
      '__metadata':{'type':'SP.Social.SocialPostCreationData'},
" +
"     'Attachment': null," +
"     'ContentItems': " + contentItemsString +
"     'ContentText':'" +message+
"', 'UpdateStatusText':false "+
"   } " +
" } }";
```

Now, the JSON file format varies slightly according to the presence/absence of hashtags and links or the presence/absence of an attachment, but it always follows the same design. In particular, we can see that, in this case, there is no attachment and the text of the message will go in the “ContentText” section, while the real values of the tokens will be written in the “ContentItems” division.

5.4.2. Testing

Software testing is an investigation conducted to provide stakeholders with information about the quality of the product or service under test (Kaner, 2006). Test techniques include the process of executing a program or application with the intent of finding software bugs (errors or other defects). Software testing involves the execution of a software component or system component to evaluate one or more properties of interest. In general, these properties indicate the extent to which the component or system under test:

- meets the requirements that guided its design and development,
- responds correctly to all kinds of inputs,
- performs its functions within an acceptable time,
- is sufficiently usable,
- can be installed and run in its intended environments, and
- achieves the general result its stakeholders desire.

The testing phase followed the agile approach used for the development of the API. Therefore, it took place iteratively for each of the functions implemented. In order to check the behaviour of the API and its CSS, the tests have first been carried out on a Windows machine using all of the most used browsers, like Mozilla Firefox, Google Chrome, Internet Explorer, Opera and Safari. Then, for compatibility purposes, older versions of the same environments have been tested out. Almost every environment required adjustments and it was often necessary to adopt ad hoc solutions.

After these tests, similar ones have been executed on Linux and Apple machines. This has been crucial because all these three systems are used in the everyday work at CERN.

5.4.3. Problems Encountered and Limitations

During the development of the API, a number of problems arose and many requests were presented during its development. It has been difficult to sort out the vast amount of information returned from SharePoint in JSON format in order to find information like the name of the author of a feed or the ID of the feed itself. As an example, to capture the array of tags retrieved using the Search REST service for the tag cloud we need to look deep in the JSON structure for the results. The resulting line of code is:

```
tags =  
result.d.query.PrimaryQueryResult.RefinementResults.Refiners.r  
esults[0].Entries.results;
```

In the code presented, the first “result” corresponds to the main section of the JSON file while every dot corresponds to opening a subsection in order to go deeper in the file structure and find what we are looking for. As we can see, there are many levels of information and it has been necessary to analyse them all. In a complex situation like this one, where no detailed documentation from Microsoft can be found, it has not been easy to find every piece of information required for the API functions to work.

In order to satisfy the requirements it has been essential to update the functions developed many times in order to make them adaptable to any kind of preference, from the section in which the feeds have to be displayed and the amount of seconds to wait before a refresh of the feeds to the colour of the text. In addition, the design is made responsive, so that it can adapt to the size of the HTML section in the webpage. If the feeds are displayed in a sidebar the text shrinks, while if the feeds are displayed in the centre of the page the text can have a larger font size and both the profile and attached pictures can take more room.

Bruno Silva De Sousa
we can also directly embed #social content on Drupal sites... Here a working example <https://eduardo-social.web.cern.ch/>
Fri Mar 28 2014 15:43:09 Like

Bruno Silva De Sousa
not for the moment... tags urls depend on query string parameters... but we can write a module to handle redirection from clean urls to social tags page
Fri Mar 28 2014 14:34:10 Like

Dan Noyes
That would be mighty useful. We could have a block on the Drupal side that pings people through to the appropriate related content on #social.
Fri Mar 28 2014 14:41:43 Like

Bruno Silva De Sousa
we can also directly embed #social content on Drupal sites... Here a working example <https://eduardo-social.web.cern.ch/>
Fri Mar 28 2014 14:45:10 Like

Dan Noyes
I think (!) I like that Drupal integration test, but it blurs the boundary between #social being a space for CERN people and a more public medium. Establishing trust is going to be crucial.
Fri Mar 28 2014 14:57:30 1 Like

Eduardo Alvarez Fernandez
the important part of this integration is that the data is always on social, no data is really imported to the drupal site. Its like facebook, twitter or linkedin widgets on web pages. Also I think after the presentation today it would be convenient the two ways of interact with social from drupal, 1- see content from social on drupal sites (widget) 2- Publish content from drupal on social, with predefined tags for example What do you think about it?
Fri Mar 28 2014 15:43:08 Like

Problem example: the post is repeated between its replies

As stated before, on Social it is possible to cite other authors in the feeds. The limitations of the API include the impossibility to cite other authors in the feeds posted through the API, writing for example “@Marco Carlo Cavalazzi” in order to notify the user to look at that particular feed. This has been a conscious choice made during the development process. The reason is that in order to get that kind of feature a user should cite the person with their correct name on Social and it is unmanageable to remember the name of all the people working at CERN. Now, this problem could be solved using the SharePoint suggestion system, which suggests possible names while typing them into the feed, but it would have introduced a high number of calls to the SharePoint service and it has been decided that since it is not a crucial feature it was not worth the risk of overloading the CERN’s network. It is possible to write that kind of text in a post, but it will not trigger a notification.

In addition, the API will not be able to work on old browsers, like Internet Explorer 7, especially because of the lack of support for CORS requests. A documentation has been redacted for the users and developers who need the Social API, which clarifies the problem and how to solve it.

6. Conclusions

The benefits of ESNs are undeniable. The extensive research on the subject proves that the possibilities of an ESN can provide a useful platform on which everyone can express opinions, ask questions and join discussions contributing to the process of knowledge sharing, which creates a self-reinforcing flow of information that contributes in making the organization ever more ready for future's challenges.

We have seen how the IT department can develop tools and improvements for the organization's ESN. More importantly, we know that when a new collaboration tool is necessary it can be integrated in the existing environment, making it a well-worth long-term investment.

Clearly, the biggest challenge for most knowledge management initiatives is the willingness of people to share knowledge with others both in their work groups and across groups, as the cultural shift is significant. Social business requires a minor revolution in thought and a steady evolution in cooperative action. In order for the social approach to succeed there should be an organizational need, a problem in daily processes or communications that can be solved by the introduction of new communication media. In addition, new technologies should be easy enough to use. Simplicity of social tools in both usage and installation facilitates the bottom-up initiatives of adoption. Furthermore, certain organizational settings, such as open enough culture, encouragement of innovation, clear intentions, policies and guidelines towards the social organization should be in place. It may seem difficult. However, as we have seen in many examples, there is no doubt that the potential benefits are worth the efforts. "Social business successes of well-known, market-leading organizations offer compelling evidence of the returns on this evolution of business" (Hinchcliffe & Kim, 2012).

In order to work as a social organization, the enterprise has also to make use of new success indicators, like:

- ◆ Financial returns, no more based on sole profit, but identified following the evolution during and after the change towards the conversation company;
- ◆ Savings, that become evident when the number of conversations increases and becomes less necessary to pay for expensive advertising;

- ♦ The number of extremely satisfied customers and employees, since it is by looking at those elements that the highest standards can be met;
- ♦ Conversations, making sure that the conversation potential is fully exploited;
- ♦ Knowledge integration, paying attention so that business is done in a way that follows the new philosophy.

At CERN, the goal is to provide a stimulating environment where people and newcomers can learn from each other. To provide a mean of communication and way to access to knowledge adapted this new era, in order to attract, engage and finally keep talents. This will help to foster innovation, which is, naturally, part of CERN's essence. ESNs empower people, everyone has an equal voice, it encourages people to speak up giving them an opportunity to make meaningful contributions with their skills and ideas, and again leveraging innovation. It increases engagement by humanizing the way in which people work (Li, 2012), opposing to the classic and formal way to communicate provided by email.

Social at CERN is being progressively promoted and it is expected an increase in its usage while existing communication channels are being moved to Social. The tools and improvements discussed in the last chapter are now used for the everyday operations. In particular, the Social API is now used in some Drupal modules. Those are important for the integration aspects of existing public facing websites (running on Drupal) with Social. Part of the plans and future of Social at CERN involve feeding the Newsfeed with content by adding more sources with relevant information. Simple examples like posting daily CERN restaurant's menu or migrating existing classifieds site CERN Market to the Social community had very positive effects and added new features to existing services. Bi-directional integration is available for other CERN Web platforms to allow users to share context-based information directly on Social. It is important to maintain programmatic interfaces easy to use to allow both consumption and feeding of data. New features are also under development like the Social Feed that consists on a topic-based microblog feed. This will allow, for example, lightweight departmental and private discussions, which can be opened to external people. It is also expected for the heavy usage of mailing lists to be replaced by Social Feeds when the purpose is mostly non-critical information exchange. Finally yet importantly, the development of comprehensive usage analytics to measure the engagement of CERN

people or success of communication campaigns is also part of the plans (De Sousa, et al., 2015).

The effective management of knowledge has been described as a critical ingredient for organisations seeking to ensure a sustainable and strategic competitive advantage. It has been brought out that processes and technology alone are not enough to drive an organisation, but its people and the knowledge that resides in them are an integral pivot in organisation's success. It is therefore essential for management in organisations to look for means to gain, maintain and leverage knowledge not for a brief period but on a regular basis.

The ESN empowers people, giving them the tools to share knowledge and, at same time, have available, at any time, all the knowledge present in the KMS of the company. This way the company can react faster to new problems and be ready to spot new business opportunities.

Appendix A

Here is displayed the code regarding the custom responsive design for Social Mobile.

File “CustomResponsiveness.css”:

```
/****** Custom SharePoint 2013 Responsive *****/
/* @media only screen and (max-device-width:750px), media only screen and (max-width:750px) { */
@media only screen and (max-device-width:750px), media only screen and (max-width:750px) { /* The order in which the conditions are written is IMPORTANT! */
    div.toolbar-wrapper(width:100.1% !important;);
    #cern-toolbar(height:110px !important;);

    #cern-toolbar h1{
        margin-top: 1em !important;
    }
    #cern-toolbar h1 a{
        font-size:3em;
        margin-top: 12%;
    }
    #cern-toolbar h1 span {
        display: none;
    }

    #cern-toolbar ul {
        border-right: 1px solid #000;
        -moz-box-shadow: 1px 0 0 #444;
        -webkit-box-shadow: 1px 0 0 #444;
        box-shadow: 1px 0 0 #444;
    }

    #cern-toolbar ul li{
        height:9em;
        padding:0 !important;
    }

    #cern-toolbar li {
        padding: 0;
        margin: 0 1em 0 1em !important;
        border-left: 2px solid #000;
    }

    #cern-toolbar li a {
        background-image: url("../layouts/15/images/cern/toolbar/toolbarsprite.png");
        background-repeat: no-repeat;
        height: 67px;
        width: 90px;
        -moz-border-radius: 0;
        -webkit-border-radius: 0;
        border-radius: 0;
        text-indent: -5000px;
        overflow: hidden;
        border-left: 2px solid #444;
    }

    #cern-toolbar .cern-account {
        background-position: 9px 0;
    }

    #cern-toolbar .cern-directory {
        background-position: 16px -107px;
        background-size: 175%;
        height: 120px;
        padding-left: 2em;
    }

    #cern-toolbar .cern-signout {
        background-position: 21px -221px;
        background-size: 175%;
        height: 120px;
        padding-left: 2em;
        margin-left: 0 !important;
    }

    #cern-toolbar .active .cern-account {
        background-position: -31px 0;
    }

    #cern-toolbar .active .cern-directory {
        background-position: -31px -40px;
    }

    #cern-toolbar .cern-accountlinks span {
        display: none;
    }

    #cern-toolbar .cern-multifactor {
        background-image: none;
        padding: 0;
    }
}

/****** Mobile (All Screens Up to 750px) *****/
```

```

/*
@media only screen and (max-device-width:480px), media only screen and (max-width:480px){
@media only screen and (max-device-width:750px), media only screen and (max-width:750px){
For the Master page use: <SharePoint:ScriptLink language="javascript" name="csToggle.js" OnDemand="true" runat="server" Localizable="false" />
*/

@media only screen and (max-device-width:750px), media only screen and (max-width:750px) { /* this condition is just for debugging, use the ones before this. */
/* General CSS. */
.ms-dialog body{background-image: none !important;}
.ms-dialog #s4-titlerow{display: none !important;}
.ms-dialog #contentBox{background-image: none !important;}

/* Global Body */
body{
    overflow: auto;
    background-image: none !important;
    background-color: rgba(255, 255, 255, 0.95);
    font-size: 4.5vw;
    margin-top:2.5em; /* Used to place the content below the CERN Toolbar. */
    height:110%;
}

#contentBox{
    margin-left:auto;margin-right:auto;
    width:90% !important;
}

div.desktopOrMobileVersion{ /* This is the parent section of the link to go to the Desktop version. It will appear at the bottom of the page. */
    float:left;
    width:100%;
    margin-left:-10px;
    padding:26px 11.2% 26px 0;
    text-align:center;
    background-color:#D8D8D8;
}

div.desktopOrMobileVersion a{ /* This is the link to go to the Desktop version.*/
    margin-left:9.5%;
    text-decoration:none;
}

/* Hiding the extra information on the TITLES of the web pages. */
#mysite-titlerow{ /* Modifying the title row of the webpage. E.g.: "About Marco Xyyyy". */
    font-size:0.7em !important;
    position: absolute;
    top: 0px;
    left: 0;
    width:31em;
    height:110px;
    overflow:hidden;
    word-wrap:break-word;
}

.ms-profile-editAndFollowLinks span.ms-textXLarge{display:none;}
.ms-profile-editAndFollowLinks{display:none;}

/* Correcting the displaying of the textarea used to write a new post on Social. */
div.ms-microfeed-focusBox.ms-microfeed-focusBoxNoFocus{outline: 1px solid #ababab}
div.ms-microfeed-focusBox.ms-microfeed-focusBoxInFocus{outline: 1px solid #ababab}

/* Resizing the "Following Everyone Mentions" string's area and font-size. */
span#ms-microfeed-titleViewSelectorPivotContainer{width:92vw !important;}
div.ms-microfeed-titlePivotControl span{font-size:6.5vw !important;}

/* Resizing the spaces between the strings "Following", "Everyone" and "Mentions". */
div.ms-microfeed-titlePivotControl span a{margin-right:4vw;} /* Following and Everyone */
div.ms-microfeed-titlePivotControl span a.ms-pivotControl-surfacedOpt[alt="Mentions"]{margin-right:0 !important;} /* Mentions */
div.ms-microfeed-titlePivotControl span a.ms-pivotControl-surfacedOpt-selected[alt="Mentions"]{margin-right:0 !important;} /* Mentions selected */

/* Changing the font-size of the options from the "everyone" drop down menu near "Share with". */
.ms-core-menu-list{font-size: 3em; max-height: 14em;}
/* Changin the size of the arrow on the right of "everyone". */
span.s4-clust.ms-viewselector-arrow.ms-menu-stdarw{height: 8px !important; width: 17px !important;} /* The parent section. */
span.s4-clust.ms-viewselector-arrow.ms-menu-stdarw img{ /* The image itself. */
    height: 1580px !important;
    width: 400px !important;
    margin-top: -345px !important;
}

/* Hiding the dots after the "Following Everyone Mentions" titles. */
.ms-pivotControl-overflowDot{display:none;}

/* Expanding width of the parent section of the feeds. */
#ms-feeddiv{width:22em;}

/* Extending the width of the feed. */
.ms-microfeed-text{max-width:90%;}
.ms-microfeed-likeImageParent{width:40px;height:40px; margin-bottom:-5px;} /* Enlarging the container. */
.ms-microfeed-likeImageParent img{ /* Enlarging the image. */
    width:400px !important;
    height:400px !important;
    top: 0px !important;
    left:-900% !important;/*!left:-360px;*/
}

.ms-microfeed-attachmentImage{ /* Expanding the size of the attachment images published with the feeds (and the replies). */
    width:500px;
    max-width: 500px;
    max-height:500px;
}

/* Adjusting the size of the profile pictures. */
.ms-peopleux-userImgWrapper{width:3em !important; height:3em !important;}
.ms-peopleux-userImg{width:3em !important; height:3em !important; max-width:3em !important; clip:rect(0px, 3em, 3em, 0px) !important;}

/* Adjusting the position of the profile pictures to make it align with the feed's author's name. */
div.ms-microfeed-userThumbnailAreaRootPadding, .ms-microfeed-userThumbnailAreaReplyPadding{padding-top: 0.1em;}
div.ms-microfeed-replyBody{margin-left:21%;}

```

```

/* Hiding the white stripe on the left of the profile picture, because it comes from a sprite image and it is thus not expandable. */
.ms-lmmlink{display:none;}

/* Adapting the right side of the feeds (with author and text) to consider the bigger user profile pictures. */
.ms-microfeed-rootBody{margin-left: 16%;}

/* Enlarging the images near the system informations like "Bruno is now following Marco." */
.ms-microfeed-iconImage{width:0.7em; height:0.7em;}
/* Mention feeds: we are expanding the size of the image displaying an "@" to the left at "@Mentioned by..." string. */
.ms-microfeed-activityImage{width:0.7em; height:0.7em;}
/* We also have to add some space below the string "Posts that mention you in this view." */
.ms-microfeed-viewDescription{margin-top: -20px; padding-bottom: 40px;}

/* Hiding the RIGHT SIDEBAR area on the right which contains the search area and shows below the number of people followed, the number of documents, sites and tags. */
/* In a user's page we want to show the details of the user, which are in the "#MiddleRightCell" section. */
#followedPeopleContainer{display:none;}
#searchInputBox{display:none;}
#WebPartWPQ6{display:none;} /* Section with the information about the number of followed people etc... */
#MiddleLRightCell #WebPartWPQ7{display:none;} /* Hiding the trending tags */

/* Resizing the textarea to POST A NEW FEED and refine the settings for the links below the textarea, like "Following", "People" and "Mentions". */
.ms-microfeed-microblogpart#ms-microblogdiv{color:black; margin-bottom:1em; max-width:none; min-width:0; width:90vw;}
#DeltaPlaceholderMain #MiddleLeftCell{width:100%;}
#ms-microfeed-titleViewSelectorPivotContainer{margin-right:0;}
.ms-microfeed-postBox{width: 98.5%; color: black;} /* This line makes the textarea and its container have the same dimensions. */
div.ms-microfeed-postBox.ms-textSmall.ms-microfeed-replyMentionHighlightDiv{height:93% !important;}

/* Enlarging the camera icon below it that is used to upload a picture with the feed. */
#ms-addImageButton_Span{width:50px; height:45px;} /* Enlarging the container. */
#ms-addImageButton_Span img{ /* Enlarging the image. */
    width:800px;
    height:800px;
    position:relative;
    top:-640px;
    left:-320px;
}

/* Enlarging the "X" button at the top right corner of every feed usable to delete them. It is visible when the user clicks (touches) the feed's area. */
button.ms-microfeed-button.ms-microfeed-deleteButton{margin-left:-30px;}
button.ms-microfeed-button.ms-microfeed-deleteButton span.ms-microfeed-deleteButtonImageParent{width:40px; height:40px; margin-bottom:-5px;} /* Enlarging the
container. */
button.ms-microfeed-button.ms-microfeed-deleteButton span.ms-microfeed-deleteButtonImageParent img{ /* Enlarging the image. */
    width: 900px !important;
    height:900px !important;
    top: -1030 !important;
    left:-1660 !important;
}

.ms-core-menu-root{display:none;} /* Hiding the dots at the right of the "Like" and "Reply" links below the feeds. */

/* Adapting the font of the "Post" button. */
#ms-postbutton{font-size:0.7em;}

/* Modifying how the links for "Following", "People" and "Mentions" are displayed. */
#ms-microfeed-titleViewSelectorPivotContainer{padding:0;}
#ms-titlebararea{margin-bottom:0 !important; padding-bottom:1.5em !important;}
#ms-titlebardiv{margin-bottom:0; padding:0; width:94%;}
#ms-feedthreaddiv{margin-top:-20px; padding-top:20px;}

@media only screen and (max-device-width: 319px), media only screen and (max-width: 319px){
    #ms-microfeed-titleViewSelectorPivotContainer{font-size: 1.27em; }
}

/* Top Links */
#suiteBar{display:none;}
.ms-core-suiteLinkList{
    #welcomeMenuBox, .ms-cui-TabRowRight{
    #suiteBarLeft,
    #suiteBarRight{
}

/* Top Links */
#ms-titlebararea{width:22em; padding-bottom:1.5em;}

/* Remove SharePoint Logo */
.ms-tableRow .ms-core-brandingBox{display: none;}

/* Bottom Section Container */
.ms-core-overlay{background-image: none; background-color: transparent !important;}
#s4-workspace{overflow: visible; height: 100% !important; width: 100% !important;}

/* Header Section */
#s4-titlerow{border-bottom: 0px #FFF solid !important; padding-top: 11px !important; background-color: transparent !important;}

/* Search */
.ms-mpSearchBox{
.ms-mpSearchBox, #searchInputBox{background-color: transparent;}
.ms-srch-sb{background-color:#FFF; border: none !important; border-radius:6px; padding: 5px;}
.ms-srch-sb>input{font-size: 1.2em !important;}

/* Logo */
#siteicon{
    float: none;
    padding-left: 0px;
    margin-top: 50px !important;
    margin-bottom: 0px !important;
    line-height:normal !important;
    text-align: center;
    height: 100%;
    margin-right: 0px;
    margin-left: 0px;
    width: auto;
    display: block;

```

```

    }

/* Navigation and Page Title */
.ms-breadcrumb-box{padding-top: 0px; margin: 10px 20px -35px 20px !important; width: auto; height: 100%;}

/* Main Navigation */
.ms-core-navigation{
.ms-core-listMenu-horizontalBox,
.ms-core-listMenu-horizontalBox ul,
.ms-core-listMenu-horizontalBox li,
.ms-core-listMenu-horizontalBox .ms-core-listMenu-item,
.ms-core-listMenu-horizontalBox > ul > li > table{display: block;}
.ms-core-listMenu-horizontalBox > .ms-core-listMenu-root > .ms-listMenu-editLink{
margin-left:0px;
}
.ms-core-listMenu-horizontalBox .ms-core-listMenu-selected,
.ms-core-listMenu-horizontalBox > ul > li > ul > li{
border: 1px #FFF solid !important;
margin: 15px 0px !important;
background-color: #FFF !important;
border-radius:6px;
font-size: 1em !important;
padding: 5px 5px 5px 5px !important;
}
.ms-core-listMenu-horizontalBox > ul > li > ul > li a{
padding: 0px !important;
}
.ms-core-listMenu-horizontalBox > ul > li > ul > li > ul .dynamic{
display: none;
}
.ms-core-listMenu-horizontalBox > ul > li > ul > li: hover{
background-color: #CCC;
}
.ms-core-listMenu-horizontalBox .ms-core-listMenu-selected{
background-color: transparent !important;
color: #FFF !important;
}
.ms-core-listMenu-horizontalBox .ms-core-listMenu-selected: hover{
background-color: #000 !important;
}
.ms-core-listMenu-horizontalBox a.ms-core-listMenu-item,
.ms-core-listMenu-horizontalBox a.ms-core-listMenu-item.ms-core-listMenu-selected,
.ms-core-listMenu-horizontalBox .ms-listMenu-editLink{
padding: 10px;
color: #000 !important;
}
.ms-core-listMenu-horizontalBox .ms-listMenu-editLink{
padding: 10px;
font-size: 2.0em !important;
display: none;
}
.ms-navedit-flyoutArrow{background-image: none !important;}
.ms-core-listMenu-horizontalBox a.ms-core-listMenu-item: hover,
.ms-core-listMenu-horizontalBox a.ms-core-listMenu-item.ms-core-listMenu-selected: hover,
.ms-core-listMenu-horizontalBox a.ms-core-listMenu-item.ms-listMenu-editLink: hover{
color: #000 !important;
}
.ms-core-listMenu-horizontalBox a.ms-core-listMenu-item.ms-core-listMenu-selected,
.ms-core-listMenu-horizontalBox a.ms-core-listMenu-item.ms-core-listMenu-selected: hover{
color: #FFF !important;
font-weight: bold;
}
}

/* Content Section Container */
#contentRow{
/* Left Navigation */
#sideNavBox{display: none;}
/* If we want only the profile picture visible, but everything else in the #sideNavBox hidden... */
/*#sideNavBox ms-profile-image{margin-bottom: -20px;}
#sideNavBox a.ms-uppercase{display:none;}
#sideNavBox a.ms-textLarge{display:none;}
#sideNavBox div#DeltaPlaceholderLeftNavBar{display:none;}*/

/* Content Area */
#contentBox{
min-width: 0px;
margin: 0px;
padding: 0px 10px;
border: 1px #FFF solid;
background-color: transparent;
border: none !important;
}

#layoutsTable td{display: inline-block !important; float: left !important; width: 100% !important;}
#layoutsTable .ms-wiki-columnSpacing{padding: 0px;}
#layoutsTable td td{display: table-cell !important; float: none !important; width: auto !important;}
.ms-rte-layoutszone-outer{display: block;}

.ms-promlink-root{display: none;}
.ms-microfeed-fullMicrofeedDiv{min-height:72vh;} /* The height of this section has to be minimum of this size, to let the "Desktop version" link be at the bottom
even if there are no feeds listed. */
.ms-hashtagProfile-mainColumn{min-height:73.3vh;} /* Feeds with same hashtag: the height of this section has to be minimum of this size, to let the "Desktop version"
link be at the bottom even if there are no feeds listed. */
.ms-microfeed-fullMicrofeedDiv, .ms-microfeed-siteFeedMicroBlogPart, .ms-microfeed-feedPart, .ms-microfeed-rootText, .ms-microfeed-replyArea, .ms-microfeed-newReplyDiv{min-
width: 0px !important;}
.ms-microfeed-message{padding-right: 0px;}
.ms-viewlists{
border: 0px #FFF solid;
background-color: transparent;
margin: 0px;
}
.lm_wb_webzone-title{
font-size: 1.5em !important;
}

```

```

border-bottom: 1px #FFF solid;
padding: 0px 0px 5px 0px;
}
.ms-webpart-titleText.ms-webpart-titleText,
.ms-webpart-titleText > a{
font-size: 1.3em !important;
font-weight: normal;
}
.ms-webpart-titleText.ms-webpart-titleText{
border-bottom: 1px #FFF solid;
padding: 0px 0px 5px 0px;
}
.lm_vb_webzone-content{padding: 5px 0px 0px 0px;}
.ms-headerCellStyleIcon, .ms-vb-imgFirstCell{display: none;}

/* Resizing the little icon image on the left of "Show all x replies" when a conversation has more than 2 replies. */
span.ms-microfeed-moreRepliesImageParent{width:28px;height:28px; margin-bottom:2px; margin-right:10px;}
span.ms-microfeed-moreRepliesImageParent img.ms-microfeed-moreRepliesImageDown{
width: 900px !important;
height:900px !important;
top: -2896% !important;
left:-904% !important;
}
span.ms-microfeed-moreRepliesImageParent img.ms-microfeed-moreRepliesImageUp{
width: 900px !important;
height:900px !important;
top: -1680% !important;
left:-2376% !important;
}
}

/***** CSS of the REPLY area *****/
.ms-microfeed-replyArea{padding-left:16.3%; max-width:16em;} /* Moving the whole reply area. */
.ms-microfeed-replyArea .ms-microfeed-userThumbnailAreaRootPadding, .ms-microfeed-replyArea .ms-microfeed-userThumbnailAreaReplyPadding{padding-top: 0px;}

.ms-microfeed-replyArea .ms-microfeedReplyContent{margin-left:3.5em;} /* Moving the textarea with post buttons. */
.ms-microfeed-replyArea .ms-microfeed-postButton{font-size:0.6em;} /* Changin the font size in the post button. */

.ms-microfeed-replyArea .ms-microfeed-focusBox{width:99.5%; margin-top:2px;} /* Enlarging the textarea. */
.ms-microfeed-replyArea .ms-microfeed-postBox{width: 98% !important; color: black;} /* Changing the reply's textarea. */
.ms-microfeed-replyArea .ms-microfeed-postBox.ms-textSmall.ms-helperText.ms-microfeed-rootOrReplyPostBox{height:64px;}

/* Enlarging the camera icon below it that is used to upload a picture with the feed. */
.ms-microfeed-replyArea span.ms-microfeed-addImageButtonImageParent{width:50px;height:45px;} /* Enlarging the container. */
.ms-microfeed-replyArea span.ms-microfeed-addImageButtonImageParent img{ /* Enlarging the image. */
width:800px;
height:800px;
position:relative;
top:-640px;
left:-320px;
}

/***** CSS for the "About" webpage of a User on Social *****/
/* Modifying the CSS of the title when the User looks at somebody else's page. */
#mysite-titlerow{margin-bottom:6em; margin-left:-5px;}
div.ms-profile-aboutMe div.ms-askMeAbout-aboutMe{text-indent:-9999px; margin-bottom:4em;} /* Hiding the text but not the whole section saying: "Tell others
about yourself and share your areas of expertise by editing your profile.". It is used in this case to add empty space between the title ("About Marco...") and the feeds. This section is
present only in the user's About webpage. */
div.ms-askMeAbout-valuesFiveOrLess{text-indent:-9999px; margin-bottom:4em;} /* Hiding the text but
not the whole section saying: "Tell others about yourself and share your areas of expertise by editing your profile.". It is used in this case to add empty space between the title ("About
Bruno...") and the feeds. This section is present only in the others About webpages. */
div.ms-profile-image{background-color:white;} /* div containing the image. */
img.ms-profile-image{height:190px; margin-left:5%;} /* the profile picture. */
h2#ms-currentFeedLabel{display:none;} /* Hiding the string that says like "Marco Xyyy activities...". */

.ms-profile-profileInfo{position:absolute; left:0.3em; top:1.9em; word-wrap:break-word; -ms-word-break: break-all;}
div#WebPartWP06_ChromeTitle{display:none !important;} /* Hiding the "In common" string next to the "About Marco ..." one.

.ms-askMeAbout-aboutMe{width:20em;} /* Enlargin the section with the "Tell others about yourself and..." text on top of the page, right below the
title. */
#ms-titlebararea{margin-bottom:1em;} /* Adding some space below the string "[User name]'s Activities". */
#WebPartWP05 #ProfileViewer_ValueProperties{display:none;} /* Hiding the "SHOW MORE" option below the information fields below the title. */

.ms-microfeed-seeMoreThreadsDiv{left:-52%;}

/***** Modifying the appearance of the pop-up window used to UPLOAD A PICTURE with the feed *****/
div.ms-dlgContent{width: 80% !important; height:11em !important; top: 20% !important; left: 10% !important;} /* Expanding the whole area. This section has the outer
white background. */
div.ms-dlgContent div.ms-dlgBorder{width:100% !important; height: 100% !important; margin-left:35px;} /* Expanding the sub-parent section. */
div.ms-erch-hover-postPersona{padding-right:20px;} /* Increasing the distance between profile image and feed text. */
div.ms-dlgContent iframe{min-width:95%; width:95% !important; height:390px !important; margin-top: -1em;} /* Expanding the parent section that contains the
"Browse", the "Upload" and the "Cancel" buttons. This section has the inner white background (try to increase the 'height' to see it). */
/* Resizing the "X" at the top-right corner of the pop-up window. */
div.ms-dlgContent div.ms-dlgTitle span#dlgTitleBtns{display:none;} /* Hiding hte "X" button on the top-right corner of the pop-up window. The User can use the "Cancel"
button to go back to the main page. */

html.ms-dialog body div#mysite-titlerow{display:none} /* If this section is shown it is placed from the browser on top of the button "Browse..." and it will make it
unclickable. */
div.ms-dlgContent table.ms-main input#profileimagepickerinput{height:2em;} /* Increasing the height of the "Browse" button used to select the picture to upload on Social. */

html.ms-dialog body{font-size: 2em !important;}
html.ms-dialog #pageStatusBar{font-size: 1em !important;} /* Resizing the font in the status bar that can appear if the User clicks ont he "Upload" button before having
chose an image. */

div.ms-dlgOverlay{width:100% !important;} /* Modifying the width of the overlay section that makes the background darker while the user chooses the image to upload. */
html.ms-dialog body input[type="file"]{font-size: 1em; width:98%;} /* Resizing the "browse" element. */
html.ms-dialog div.ms-core-form-bottomUploadButtonBox{margin-top: 50px}
html.ms-dialog body input[type="button"].ms-ButtonHeightWidth{ /* Resizing the "Upload" and "Cancel" buttons. */
font-size: 1em !important;
padding-top:0.5em;
padding-bottom:0.6em;
margin-left:50px;
}

```

```

html.ms-dialog div.desktopOrMobileVersion{display:none;} /* Hiding the "Desktop version" link at the bottom. */

/***** Modifying the appearance of the pop-up window used to UPLOAD A PICTURE with the feed *****/
div.ms-microfeed-confirmationDiv{max-width:80% !important; font-size:1.5em;}
div.ms-microfeed-confirmationDiv button{font-size:0.5em;}
div.ms-microfeed-confirmationDiv button.ms-microfeed-confirmationDivButton.ms-microfeed-cancelButton{margin-left:30px !important;}

/***** SPECIFIC HASHTAG: Modifying the appearance of the window used to show the feeds containing a specific hashtag (es. #social). *****/
body div.ms-hashTagProfile-mainColumn{width:100%; margin-top:3em; word-wrap:break-word; -ms-word-break: break-all;} /* Defining new rules for the main section (with the
feeds). */
div.ms-hashTagProfile-rightColumn{display:none;} /* Hiding the right column with "Add a related tag" and "SEE ALL". */
div.ms-hashTagProfile-mainColumn div.ms-srch-item{width:95vw !important;} /* Adjusting the width of the feeds. */
div.ms-hashTagProfile-mainColumn div.ms-srch-item-body{width:80%;} /* Adjusting the width of the main part of the feeds (the part without the profile
picture). */
div.ms-hashTagProfile-mainColumn div.ms-srch-item-metadataContainer{width:24%;} /* Adjusting the width of the info on each feed which are displayed at the bottom-
right corner of the feed's space. */
div.ms-hashTagProfile-mainColumn ul.cbs-List{width:3em !important;} /* This section expands its width when the user goes in all paging
pages except the first one. */
div.ms-srch-hover-outerContainer{display:none !important;} /* This section of the page would show more details on the feed when the user goes hover it with the mouse. In the
Mobile version this area is hidden. */

/* Enlarging the star icon at the left of the first string "(star)follow this #tag". */
div.ms-hashTagProfile-mainColumn a#HashTagProfile_FollowTagLinkOption1 span{height:40px !important; width:40px !important; margin-bottom:11px;} /* Enlarging the parent
span. */
div.ms-hashTagProfile-mainColumn a#HashTagProfile_FollowTagLinkOption1 span img.ms-hashTagProfile-followTag-img{height:240px !important; width:240px !important; margin-left:-
64px !important} /* Enlarging the image itself. */

div.ms-hashTagProfile-mainColumn li#PagingImageLink{display:none;} /* Hiding the small icons on the top-left of the page that are meant to allow the User to move
forward and back through the paging of the feeds. The User can anyway use a better designed link at the bottom centre of the page. */
div.ms-hashTagProfile-mainColumn span.ms-srchnav-quotatationopennglyph-span{width:22px; height:22px; margin-bottom: 7px;}
div.ms-hashTagProfile-mainColumn span.ms-srchnav-quotatationcloseglyph-span{width:22px; height:22px; margin-bottom: 7px;}
div.ms-hashTagProfile-mainColumn span.ms-srchnav-quotatationopennglyph-span img{width:200px; height:200px; left:-170px !important; top: -33px;}
div.ms-hashTagProfile-mainColumn span.ms-srchnav-quotatationcloseglyph-span img{width:200px; height:200px; left:-156px !important; top: -65px;}

div.ms-hashTagProfile-mainColumn .ms-srch-result #Paging{margin: 45px 0 30px 7%;} /* Centering the paging numbered links (1 2 3). */

/* end of mobile CSS */
}

```

Appendix B

The code below regards the custom Resource Planning Tool integrated into SharePoint.

File “customTimeline_newTaskForm.js”:

```
// Adding jQuery to the webpage
document.write('<script type="text/javascript" src="//code.jquery.com/jquery-1.11.0.min.js"></script>');

// Global variables that hold the names of the lists on SharePoint
var globalUsersListName = 'Users'; // The list containing the name of the User and the color to assign to it.
var globalEquipmentListName = 'Equipment';
var globalProjectsListName = 'Projects';
var globalTasksListName = 'Tasks';

// Calling the first function
ExecuteOrDelayUntilScriptLoaded(registerRenderer, 'clienttemplates.js'); // The first function launched

// The second function launched, after loading the file 'clienttemplates.js'.
function registerRenderer()
{
    var ctxForm = {};
    ctxForm.Templates = {};
    ctxForm.OnPreRender = OnPreRenderDocItemTemplate;
    ctxForm.OnPostRender = {};

    SPClientTemplates.TemplateManager.RegisterTemplateOverrides(ctxForm);
}

/***** Code regarding specifically the "New Task" form for a new task. *****/

// Function that will be called only once, when the page is loaded. It shrinks the Equipment list to consider only the elements related to the selected category.
function editNewTaskFormEquipment() {
    var listItemEnumerator = equipmentListItems.getEnumerator();

    var categories = document.querySelectorAll('[title="Equipment Category"]')[0];
    var categoriesIndex = categories.selectedIndex; // Chaching the HTML section with the categories' drop-down list
    var selectedCategory = categories[categoriesIndex].innerHTML; // Catching the selected element in the list
    categories.setAttribute("onchange", "onCategoryChange()"); // Setting an 'onchange' event that will ebtriggered everytime the chosen category changes.

    try{
        var removeButton = document.querySelectorAll('input[value="< Remove"]')[0];
        removeButton.setAttribute("onclick", "onCategoryChange()"); // Setting an 'onclick' event that will ebtriggered everytime some equipment is
removed from the list.
    }catch(e){}

    var equipmentDisplayedList = document.querySelectorAll('[title="Equipment Name"]')[0]; // Chaching the HTML section with the equipment' drop-down list
    if(equipmentDisplayedList == undefined || equipmentDisplayedList == null){
        equipmentDisplayedList = document.querySelectorAll('[title="Equipment Name possible values"]')[0];
    }
    equipmentDisplayedList.innerHTML = ''; // Clearing the equipment' list

    var item, equipmentName, category;
    var dropDownElement = ''; // The HTML string that will be put in the equipment' drop-down list
    while ( listItemEnumerator.moveNext() ) {
        item = listItemEnumerator.get_current().$5_0.$1h_0.$m_dict; // The current examined item

        equipmentName = item.Title; // The Equipment name. E.g.: "AC mole #6"
        equipmentCategory = item.Parent_x0020_Category.$2e_1; // The Category in which the equipment is included. E.g.: "3-D Mapper Bench"
        equipmentID = item.ID; // The value associated with the task name. E.g.: 62

        if( equipmentCategory == selectedCategory ){
            dropDownElement = '<option value="'+ equipmentID +'">' + equipmentName + '</option>';
            equipmentDisplayedList.innerHTML += dropDownElement;
        }
    }
}

function onCategoryChange(){
    var listItemEnumerator = equipmentListItems.getEnumerator();

    var categories = document.querySelectorAll('[title="Equipment Category"]')[0];
    var categoriesIndex = categories.selectedIndex; // Chaching the HTML section with the categories' drop-down list
    var selectedCategory = categories[categoriesIndex].innerHTML; // Catching the selected element in the list

    var equipmentDisplayedList = equipmentDisplayedList = document.querySelectorAll('[title="Equipment Name possible values"]')[0]; // Chaching the HTML section with the
equipment' list
    equipmentDisplayedList.innerHTML = ''; // Clearing the equipment' list

    var chosenEquipmentArray = new Array(); // This array will contain the name of the equipment chosen from the User (so equipment put in the area ont he right).
    var chosenEquipmentList = document.querySelectorAll('[title="Equipment Name selected values"]')[0]; // Chaching the HTML section with the equipment' drop-down list
    for( var i=0; i<chosenEquipmentList.length; i++){
        chosenEquipmentArray.push( chosenEquipmentList[i].text );
    }

    var item, equipmentName, category;
```

```

var dropdownElement = ''; // The HTML string that will be put in the equipment' drop-down list
while ( listItemEnumerator.moveNext() ) {
    item = listItemEnumerator.get_current().$$_0.$1h_0.$m_dict; // The current examined item

    equipmentName = item.Title; // The Equipment name. E.g.: "AC mole #6"
    equipmentCategory = item.Parent_x0020_Category.$2e_1; // The Category in which the equipment is included. E.g.: "3-D Mapper Bench"
    equipmentID = item.ID; // The value associated with the task name. E.g.: 62

    if( equipmentCategory == selectedCategory && chosenEquipmentArray.indexOf(equipmentName) == -1 ){
        dropdownElement = '<option value="'+ equipmentID +'" title="'+ equipmentName +'">' + equipmentName + '</option>';
        equipmentDisplayedList.innerHTML += dropdownElement;
    }
}

var equipmentArea = document.querySelectorAll('[title="Equipment Name selected values"]')[0];
for(var i=0; i<equipmentArea.length; i++){
    $(equipmentArea[i]).unbind();
    $(equipmentArea[i]).off();
    equipmentArea[i].onclick = onCategoryChange ;
}

/**** Modifying the category shown in the timeline. *****/
updateCategoryInTimeline();
}

// This function is very wimilar to "onQuerySucceededEquipment()", but it handles the case in which every call tot he Server has already been made, so that we just have to use the global
variables already available for us.
function updateCategoryInTimeline(){
    // Resetting the global variables
    globalEquipmentArray = new Array();
    globalCategoriesArray = new Array();
    globalCustomTimelineEquipmentHTMLstring = '';

    var equipmentList = '<div style="float:left; margin-top: 42px;">';
    var numRows = 0; // Variable used to know the amount of rows to display.
    var firstItem = true; // Boolean to treat differently the first item of the array. It needs a greater padding-top.
    var listItemEnumerator = equipmentListItems.getEnumerator();
    var innerListItemEnumerator = equipmentListItems.getEnumerator();
    var category = '';
    var firstOfEquipment = true;

    var ua = window.navigator.userAgent;
    var msie = ua.indexOf("MSIE ") > -1 || !navigator.userAgent.match(/Trident.*rv:11\./); // "True" if the Browser is IE (with support for IE 11).
    var firefox = ua.toLowerCase().indexOf('firefox') > -1; // Detects any version of Firefox. "True" if we are using Firefox;

    while (listItemEnumerator.moveNext() ) {
        var oListItem = listItemEnumerator.get_current();
        try{
            category = oListItem.get_item('Parent_x0020_Category').$2e_1;
            if(category != null && globalCategoriesArray.indexOf( category ) == -1 ){ // If we have not met this category before...
                globalCategoriesArray.push(category); // We add it to the 'globalCategoriesArray'

                // setting local variables
                var equipmentName, innerListItem;
                var tempEquipmentString = '';
                var firstOfCategory = true;
                innerListItemEnumerator = equipmentListItems.getEnumerator(); // Resetting the 'innerListItemEnumerator'

                // Reading the selected category from the drop-down list in the webpage
                var categories = document.querySelectorAll('[title="Equipment Category"]')[0];
                var categoriesIndex = categories.selectedIndex; // Chaching the HTML section with the categories' drop-
                down list
                var selectedCategory = categories[categoriesIndex].innerHTML; // Catching the selected element in the list

                // Seek for every equipment belonging to that category and add it to the HTML string.
                while(innerListItemEnumerator.moveNext() ){
                    var innerListItem = innerListItemEnumerator.get_current();
                    if(category != selectedCategory){ continue; }
                    if( category == innerListItem.get_item('Parent_x0020_Category').$2e_1 ){ // If the currently considered category
                        equipmentName = innerListItem.get_item('Title');
                        globalEquipmentArray.push(equipmentName); // Memorizing the name of the Equipment. We will need it later while
                        displaying the Tasks in the timeline

                        var stringHeight;
                        if(msie || firefox){ // If IE or Firefox...
                            stringHeight = 9;
                        }else{
                            stringHeight = 22;
                        }

                        if(firstOfCategory){ // If it is the first element of a category...
                            if(firstOfEquipment){ // If it is the first line of equipment to be written (in absolute)...
                                tempEquipmentString += '<p id="categoriesList" style="padding:0; margin-top:3px; margin-left:-110px; height:'+ stringHeight + 'px; -webkit-margin-before: 0em!important; -webkit-margin-after: 0em!important;"><span style="display:inline-block; margin-top:11px; width:120px;" title="'+
                                category + '">' + category + '</span> <span style="display:inline-block; margin-top:10px; width:200px; white-space:nowrap; text-overflow:ellipsis; overflow:hidden; margin-bottom: -3px;"
                                title="'+ equipmentName + '">' + equipmentName + '</span><p>';
                                firstOfEquipment = false;
                            }else{
                                tempEquipmentString += '<p id="categoriesList" style="padding:0; margin-left:-110px; height:'+
                                stringHeight + 'px; -webkit-margin-before: 0em!important; -webkit-margin-after: 0em!important;"><span style="display:inline-block; margin-top:11px; width:120px;" title="'+ category + '">'
                                + category + '</span> <span style="display:inline-block; margin-top:10px; width:200px; white-space:nowrap; text-overflow:ellipsis; overflow:hidden; margin-bottom: -3px;" title="'+
                                equipmentName + '">' + equipmentName + '</span><p>';
                            }
                            firstOfCategory = false; // This has to be done in any case
                        }else{
                            tempEquipmentString += '<p id="categoriesList" style="padding:0; margin-left:-110px; height:'+ stringHeight + 'px;
                            -webkit-margin-before: 0em!important; -webkit-margin-after: 0em!important;"><span style="display:inline-block; margin-top:11px; width:120px;" </span> <span style="display:inline-block;
                            margin-top:10px; width:200px; white-space:nowrap; text-overflow:ellipsis; overflow:hidden;" title="'+ equipmentName + '">' + equipmentName + '</span><p>';
                        }
                    }
                }
                numRows++; // Increasing the rows' counter (Used to set the height of the Timeline with the Tasks)
            }
        }
    }
}

```



```

        }
        equipmentList += tempEquipmentString;
    }
} catch(e) {}

}
equipmentList += '</div>';

// Passing the local variables' values to the global ones.
globalCustomTimelineEquipmentHTMLString = equipmentList;

// Re-populating the timeline
onQuerySucceededTasks();
}

/***** Timeline customization code: through this code we can display the timeline above the "New Task" form and then launch the code above. *****/

// The system tries to prerender 3 times.
// The first one is useless in Chrome and Firefox (it is useful in IE), it gives us no data from the Server, so we can avoid it. After that call on Firefox it works fine, while on Chrome
the system gives us the error:
// Uncaught Error: The collection has not been initialized. It has not been requested or the request has not been executed. It may need to be explicitly requested.
// but it does not matter. The code makes the third call that solves the problem even on Chrome.
var firstCallAlreadyMade = false;
function onPreRenderDocItemTemplate(renderCtx) {
    SP.SOD.executeOrDelayUntilScriptLoaded(loadContext, 'sp.js');
    function loadContext() {
        var ua = window.navigator.userAgent;
        var msie = ua.indexOf("MSIE ") > -1 || !!navigator.userAgent.match(/Trident.*rv:11\./); // "True" if the Browser is IE (with support for IE
11).

        if(msie){
            checkSituationAndLunch();
        }else{
            if(firstCallAlreadyMade == false){
                checkSituationAndLunch();
                firstCallAlreadyMade = true;
            }
        }
    }
}

function checkSituationAndLunch() {
    try{
        // This control has been implemented since for some actions SharePoint refreshes the webparts without refreshing the whole webpage.
        // We are talking about operations like expanding or collapsing a Group of Tasks.
        if ( document.getElementById("innerTimeline") ){ // If the timeline is already in the webpage...
            return; // Do not add code to the timeline.
        }
        // Reading the Equipment and the Categories from SharePoint
        retrieveEquipmentAndCategories();
    } catch(e) { return; }
}

// Retrieving information on every equipment and every category from SharePoint.
function retrieveEquipmentAndCategories() {
    /* In the "New Task" form this function is called many times and the variable "equipmentListItems" is re-written for many times.
    * This leads to a race condition when the first sequence of function tries in the code to read some data from it in order to display the "Equipment" and their "Categories",
    * thus causing, some times, to find the resource locked and so having as output the HTML section thought for the equipment empty. */
    if (this.equipmentListItems != null && this.equipmentListItems != undefined){
        return;
    }

    var siteUrl = document.URL; // It is going to be something like: https://espace2013-dev.cern.ch/test-Timeline/Lists/Tasks/AllItems.aspx
    // We want something like "https://espace2013.cern.ch/test-Timeline"

    // Correcting the URL (if necessary)
    if( siteUrl == null || siteUrl == undefined ){ siteUrl = window.location.href; }
    var numSlashes = 0;
    for( var i=0; i<siteUrl.length; i++){
        if(siteUrl[i] == '/'){
            numSlashes++;
            if(numSlashes == 4){
                siteUrl = siteUrl.substring(0, i);
                break;
            }
        }
    }

    var clientContext = new SP.ClientContext(siteUrl);
    var oList = clientContext.get_web().get_lists().getByTitle( globalEquipmentListName );

    var camlQuery = new SP.CamlQuery();
    camlQuery.set_viewXml('<View><Query><Where>' +
        '</Where></Query></View>');

    this.equipmentListItems = oList.getItems(camlQuery);
    clientContext.load(equipmentListItems);

    clientContext.executeQueryAsync(Function.createDelegate(this, this.onQuerySucceededEquipment), Function.createDelegate(this, this.onQueryFailed));
}

// Global variables for categories and equipment
var globalEquipmentArray = new Array(); // This global variable will keep the names of the Equipment to let the User change timespan in the timeline if needed, without
recalling the Server for this information.
var globalCategoriesArray = new Array(); // This global variable will keep the names of the Categories.

```

```

var globalCustomTimelineEquipmentHTMLstring = '';
function onQuerySucceededEquipment() {
    // Resetting the global variables
    globalEquipmentArray = new Array();
    globalCategoriesArray = new Array();
    globalCustomTimelineEquipmentHTMLstring = '';

    var equipmentList = '<div style="float:left; margin-top: 42px;">';
    var numRows = 0; // Variable used to know the amount of rows to display.
    var firstItem = true; // Boolean to treat differently the first item of the array. It needs a greater padding-top.
    var listItemEnumerator = equipmentListItems.getEnumerator();
    var innerListItemEnumerator = equipmentListItems.getEnumerator();
    var category = '';
    var firstOfEquipment = true;

    var ua = window.navigator.userAgent;
    var msie = ua.indexOf("MSIE ") > -1 || !navigator.userAgent.match(/Trident.*rv:11\./); // "True" if the Browser is IE (with support for IE 11).
    var firefox = ua.toLowerCase().indexOf('firefox') > -1; // Detects any version of Firefox. "True" if we are using Firefox;

    while (listItemEnumerator.moveNext() ) {
        var oListItem = listItemEnumerator.get_current();
        try {
            category = oListItem.get_item('Parent_x0020_Category').$2e_1;
            if (category != null && globalCategoriesArray.indexOf( category ) == -1 ) { // If we have not met this category before...
                globalCategoriesArray.push(category); // We add it to the 'globalCategoriesArray'

                // setting local variables
                var equipmentName, innerListItem;
                var tempEquipmentString = '';
                var firstOfCategory = true;
                innerListItemEnumerator = equipmentListItems.getEnumerator(); // Resetting the 'innerListItemEnumerator'

                // Reading the selected category from the drop-down list in the webpage
                var categories = document.querySelectorAll('[title="Equipment Category"]')[0];
                var categoriesIndex = categories.selectedIndex; // Chaching the HTML section with the categories' drop-
                down list
                var selectedCategory = categories[categoriesIndex].innerHTML; // Catching the selected element in the list

                // Seek for every equipment belonging to that category and add it to the HTML string.
                while (innerListItemEnumerator.moveNext() ) {
                    var innerListItem = innerListItemEnumerator.get_current();
                    if (category != selectedCategory) { continue; }
                    if (category == innerListItem.get_item('Parent_x0020_Category').$2e_1 ) { // If the currently considered category
                        equipmentName = innerListItem.get_item('Title');
                        globalEquipmentArray.push(equipmentName); // Memorizing the name of the Equipment. We will need it later while
                        displaying the Tasks in the timeline

                        var stringHeight;
                        if (msie || firefox) { // If IE or Firefox...
                            stringHeight = 9;
                        } else {
                            stringHeight = 22;
                        }

                        if (firstOfCategory) { // If it is the first element of a category...
                            if (firstOfEquipment) { // If it is the first line of equipment to be written (in absolute)...
                                tempEquipmentString += '<p id="categoriesList" style="padding:0; margin-left:3px; margin-left:-110px; height:'+ stringHeight +'px; -webkit-margin-before: 0em!important; -webkit-margin-after: 0em!important;"><span style="display:inline-block; margin-top:11px; width:120px;" title="'+ category +'>'
                                + category +'</span> <span style="display:inline-block; margin-top:10px; width:200px; white-space:nowrap; text-overflow:ellipsis; overflow:hidden; margin-bottom: -3px;" title="'+ equipmentName +'>'
                                + equipmentName +'</span></p>';
                                firstOfEquipment = false;
                            } else {
                                tempEquipmentString += '<p id="categoriesList" style="padding:0; margin-left:-110px; height:'+ stringHeight +'px; -webkit-margin-before: 0em!important; -webkit-margin-after: 0em!important;"><span style="display:inline-block; margin-top:11px; width:120px;" title="'+ category +'>'
                                + category +'</span> <span style="display:inline-block; margin-top:10px; width:200px; white-space:nowrap; text-overflow:ellipsis; overflow:hidden; margin-bottom: -3px;" title="'+ equipmentName +'>'
                                + equipmentName +'</span></p>';
                            }
                            firstOfCategory = false; // This has to be done in any case
                        } else {
                            tempEquipmentString += '<p id="categoriesList" style="padding:0; margin-left:-110px; height:'+ stringHeight +'px; -webkit-margin-before: 0em!important; -webkit-margin-after: 0em!important;"><span style="display:inline-block; margin-top:11px; width:120px;" title="'+ category +'>'
                            + category +'</span> <span style="display:inline-block; margin-top:10px; width:200px; white-space:nowrap; text-overflow:ellipsis; overflow:hidden;" title="'+ equipmentName +'>'
                            + equipmentName +'</span></p>';
                        }
                    }
                    numRows++; // Increasing the rows' counter (Used to set the height of the Timeline with the Tasks)
                }
            }
            equipmentList += tempEquipmentString;
        }
    }
} catch (e) {}

equipmentList += '</div>';

// Passing the local variables' values to the global ones.
globalCustomTimelineEquipmentHTMLstring = equipmentList;

// Calling the next function for the retrieval of the Projects
retrieveProjects();
}

function retrieveProjects() {
    var siteUrl = document.URL; // It is going to be something like: https://espace2013-dev.cern.ch/test-Timeline/Lists/Tasks/AllItems.aspx
    // We want something like "https://espace2013.cern.ch/test-Timeline"

    // Correcting the URL (if necessary)
    if (siteUrl == null || siteUrl == undefined) { siteUrl = window.location.href; }
    var numSlashes = 0;
    for (var i=0; i<siteUrl.length; i++) {
        if (siteUrl[i] == '/') {
            numSlashes++;
        }
    }
}

```

```

        if(numSlashes == 4){
            siteUrl = siteUrl.substring(0, i);
            break;
        }
    }
}

var clientContext = new SP.ClientContext(siteUrl);
var oList = clientContext.get_web().get_lists().getByTitle( globalProjectsListName );

var camlQuery = new SP.CamlQuery();
camlQuery.set_viewXml('<View><Query><Where>' +
    '</Where></Query></View>');

this.projectListItems = oList.getItems(camlQuery);
clientContext.load(projectListItems);

clientContext.executeQueryAsync(Function.createDelegate(this, this.onQuerySucceededProjects), Function.createDelegate(this, this.onQueryFailed));
}

var globalProjectsArray = new Array(); // This global variable will keep the names of the Projects and their associated colors.
function onQuerySucceededProjects(){
    // Variables necessary to read the query results
    var listItemEnumerator = projectListItems.getEnumerator();
    var innerListItemEnumerator = projectListItems.getEnumerator();

    var project, color;
    while (listItemEnumerator.moveNext() ) {
        var oListItem = listItemEnumerator.get_current();
        try{
            project = oListItem.get_item('Title');
            color = oListItem.get_item('Color');
            globalProjectsArray.push({'projectName':project, 'projectColor':color});
        }catch(e){}
    }

    // Calling the next function for the retrieval of the Users
    retrieveUsers();
}

function retrieveUsers(){
    var siteUrl = document.URL; // It is going to be something like: https://espace2013-dev.cern.ch/test-Timeline/Lists/Tasks/AllItems.aspx
    // We want something like "https://espace2013.cern.ch/test-Timeline"

    // Correcting the URL (if necessary)
    if( siteUrl == null || siteUrl == undefined){ siteUrl = window.location.href; }
    var numSlashes = 0;
    for( var i=0; i<siteUrl.length; i++){
        if(siteUrl[i] == '/'){
            numSlashes++;
            if(numSlashes == 4){
                siteUrl = siteUrl.substring(0, i);
                break;
            }
        }
    }

    var clientContext = new SP.ClientContext(siteUrl);
    var oList = clientContext.get_web().get_lists().getByTitle( globalUsersListName );

    var camlQuery = new SP.CamlQuery();
    camlQuery.set_viewXml('<View><Query><Where>' +
        '</Where></Query></View>');

    this.userListItems = oList.getItems(camlQuery);
    clientContext.load(userListItems);

    clientContext.executeQueryAsync(Function.createDelegate(this, this.onQuerySucceededUsers), Function.createDelegate(this, this.onQueryFailed));
}

var globalUsersArray = new Array(); // This global variable will keep the names of the Users (saved in a list on SharePoint).
function onQuerySucceededUsers(){
    // Variables necessary to read the query results
    var listItemEnumerator = userListItems.getEnumerator();
    var innerListItemEnumerator = userListItems.getEnumerator();

    var user, color;
    while (listItemEnumerator.moveNext() ) {
        var oListItem = listItemEnumerator.get_current();
        try{
            user = oListItem.get_item('User');
            color = oListItem.get_item('Color');
            globalUsersArray.push({'userName':user, 'userColor':color});
        }catch(e){}
    }

    // Calling the next function for the retrieval of the Tasks
    retrieveTasksListItems();
}

// Retrieving information about each of the Tasks and adding them to the Timeline
function retrieveTasksListItems(){
    var siteUrl = document.URL; // It is going to be something like: https://espace2013-dev.cern.ch/test-Timeline/Lists/Tasks/AllItems.aspx
    // We want something like "https://espace2013.cern.ch/test-Timeline"

    // Correcting the URL (if necessary)
    if( siteUrl == null || siteUrl == undefined){ siteUrl = window.location.href; }
    var numSlashes = 0;
    for( var i=0; i<siteUrl.length; i++){
        if(siteUrl[i] == '/'){
            numSlashes++;
            if(numSlashes == 4){
                siteUrl = siteUrl.substring(0, i);
                break;
            }
        }
    }
}

```

```

    }
    var clientContext = new SP.ClientContext(siteUrl);
    var oList = clientContext.get_web().get_lists().getByTitle( globalTasksListName );

    var camlQuery = new SP.CamlQuery();
    camlQuery.set_viewXml('<View><Query><Where>' +
        '</Where></Query></View>');

    this.tasksListItems = oList.getItems(camlQuery);
    clientContext.load(tasksListItems);

    clientContext.executeQueryAsync(Function.createDelegate(this, this.onQuerySucceededTasks), Function.createDelegate(this, this.onQueryFailed));
}

function onQueryFailed(sender, args) {
    alert("There has been a problem communicating with the Server. Please try again later.");
    console.log('Request failed. ' + args.get_message() + '\n' + args.get_stackTrace());
}

// Reading the Tasks saved in the timeline
var addTasksToTimelineFlag = false; // This variable will tell the system if the function "addTasksToTimeline" has already been called at least once or not.
function onQuerySucceededTasks () {
    var numRows = globalEquipmentArray.length;
    var equipmentHTMLstring = globalCustomTimelineEquipmentHTMLstring;

    // Deleting the previously shown timeline (if present)
    try{
        var temp = document.getElementById('timelineArea');
        temp.parentNode.removeChild(temp);
    }catch(e){}

    // Catching the area for the timeline in the HTML code
    var timelineArea = $('div[id~="MSOZoneCell_"]')[0]; // Chatching the HTML section in which we want to add the customized timeline
    // e.g.: id="MSOZoneCell_WebPart...WP03"

    $(timelineArea).prepend('<div id="timelineArea"></div>');
    timelineArea = document.getElementById("timelineArea");

    // Modifying the CSS for the section to include the Rows on the Left of the Timeline.
    // (if dynamic width) Each character equals 0.7em, so the amount of space on the left has to be 0.7*maxNumCharacters.
    timelineArea.style.paddingLeft = "120px"; // (0.7*maxNumCharacters) + 'em'; // Making space on the left of the Timeline for the Rows' titles.
    timelineArea.style.height = ((22*numRows)+70) + 'px'; // Expanding the area including the timeline to push down the rest of the webpage (the list containing the Tasks).

    // Creating some radio buttons to enable the User to change the timespan of the timeline
    var radioButtonns = '<span id="timelineRadioButtons" style="margin-left:7px;">Timespan <input type="radio" onclick="addTasksToTimeline(7, 1060);" name="time span" value="Week"
checked>Week &nbsp;&nbsp;&nbsp;' +
        '<input type="radio" onclick="addTasksToTimeline(15, 1060, null);"
name="time span" value="2 Weeks">2 Weeks &nbsp;&nbsp;&nbsp;' +
        '<input type="radio" onclick="addTasksToTimeline(31, 1060, null);"
name="time span" value="Month">Month &nbsp;&nbsp;&nbsp;' +
        '<input type="radio" onclick="addTasksToTimeline(90, 1060, null);"
name="time span" value="3 Month">3 Months &nbsp;&nbsp;&nbsp;' +
        '<input type="radio" onclick="addTasksToTimeline(210, 1060, null);"
name="time span" value="7 Month">7 Months &nbsp;&nbsp;&nbsp;' +
        '<input type="radio" onclick="addTasksToTimeline(365, 1060, null);"
name="time span" value="Year">Year &nbsp;&nbsp;&nbsp;' +
        '</span>';
    radioButtonns = '<span id="colorRadioButtons">Color ' +
        '<input type="radio"
onclick="colorTasksInTimeline(\'colorByProject\');" name="color choice" value="Color the tasks by Project" checked>by Project &nbsp;&nbsp;&nbsp;' +
        '<input type="radio"
onclick="colorTasksInTimeline(\'colorByUser\');" name="color choice" value="Color the tasks by User">by User &nbsp;&nbsp;&nbsp;' +
        '</span>';

    // Adding the resources found in the list as rows in the Timeline
    var resourcesSection = document.getElementById("timelineRadioButtons");
    if( resourcesSection == undefined || resourcesSection == null ){ // If there is our custom Resources list and the rows have not yet been added...
        timelineArea.innerHTML = radioButtonns + equipmentHTMLstring + '<div id="timeline"></div>' + timelineArea.innerHTML;

        var timeline = document.getElementById('timeline');

        if(numRows > 1){
            timeline.style.height = ((22*numRows)-2) + 'px'; // Enlarging the height of the timeline in order to have one line for each Resource.
            // The last line will not need a white space below it. That's why we take out 2px fromt he result.
        }else{
            timeline.style.height = '20px'; // Enlarging the height of the timeline in order to have one line for each Resource.
        }
        var timelineWidth = 1060;
        timeline.style.width = timelineWidth + 'px'; // Manually setting the width of the timeline to override the behaviour of SharePoint, which would expand the
        timeline according to the width of the page.
        timeline.style.display = "inline";
    }

    // Reading the selected category from the drop-down list in the webpage
    var categories = document.querySelectorAll('[title="Equipment Category"]')[0];
    var categoriesIndex = categories.selectedIndex; // Chatching the HTML section with the categories' drop-down list
    var selectedCategory = categories[categoriesIndex].innerHTML; // Catching the selected element in the list

    /***** Adding our code to the timeline. *****/
    var numOfDayInTimeline=7;
    addTasksToTimeline(numOfDayInTimeline, timelineWidth, selectedCategory, 'colorByProject');

    /***** We are also ready to modify the equipment present in the drop-down list of the "New Task" form. *****/
    editNewTaskFormEquipment();
}

// This function will color the tasks in the timeline according to the equipment or the personnel.
function colorTasksInTimeline(colorRule){
    var timelineWidth = document.getElementById('timeline').style.width;
    var numOfDayInTimeline = 7; // Initializing the variable for the consistency check
}

```

```

var radioButtons = document.getElementById('timelineRadioButtons').getElementsByTagName('input');

var numOfDayArray = [7, 15, 31, 90, 210, 365]; // Defining the array containing the number of days considered for each possible time span

for(var i=0; i<radioButtons.length; i++){
    if(radioButtons[i].checked){
        numOfDayInTimeline = numOfDayArray[i];
        break;
    }
}

// Reading the selected category from the drop-down list in the webpage
var categories = document.querySelectorAll('[title="Equipment Category"]')[0];
var categoriesIndex = categories.selectedIndex; // Chaching the HTML section with the categories' drop-down list
var selectedCategory = categories[categoriesIndex].innerHTML; // Catching the selected element in the list

if(colorRule == 'colorByProject'){
    addTasksToTimeline(numOfDayInTimeline, timelineWidth, selectedCategory, 'colorByProject');
}else{
    addTasksToTimeline(numOfDayInTimeline, timelineWidth, selectedCategory, 'colorByUser');
}
}

// This function will add the tasks read from the Server to the timeline "manually" (instead of using the SharePoint's disposition.
function addTasksToTimeline(numOfDayInTimeline, timelineWidth, selectedCategory, colorRule){
    // Checking the input
    if( typeof(numOfDayInTimeline) == 'string' ){ numOfDayInTimeline = parseInt(numOfDayInTimeline); }
    if( typeof(timelineWidth) == 'string' ){ timelineWidth = parseInt(timelineWidth); }
    if(selectedCategory == null || selectedCategory == undefined){
        // Reading the selected category from the drop-down list in the webpage
        var categories = document.querySelectorAll('[title="Equipment Category"]')[0];
        var categoriesIndex = categories.selectedIndex; // Chaching the HTML section with the categories' drop-down list
        var selectedCategory = categories[categoriesIndex].innerHTML; // Catching the selected element in the list
    }

    var siteUrl = document.URL;
    // Correcting the URL (if necessary)
    if( siteUrl == null || siteUrl == undefined ){ siteUrl = window.location.href; }
    var numSlashes = 0;
    for( var i=0; i<siteUrl.length; i++){
        if(siteUrl[i] == '/'){
            numSlashes++;
            if(numSlashes == 4){
                siteUrl = siteUrl.substring(0, i);
                break;
            }
        }
    }

    var timeline = document.getElementById('timeline');

    /***** If the User clicks on the timeline radio buttons we have to read which radio button *****/
    if(colorRule == null){
        // Reading the color rule to apply to the tasks
        var radioButtons = document.getElementById('colorRadioButtons').getElementsByTagName('input'); // The radio buttons for the color rules
        if(radioButtons[0].checked){ // If the first radio button is checked...
            colorRule = 'colorByProject';
        }else{
            colorRule = 'colorByUser';
        }
    }

    /***** This piece of code will be used to refresh the timeline when selecting different time spans (e.g.: 1 week, 2 week, 1 month etc...). *****/
    // only if the timeline is set as "shown" from SharePoint. If "hidden" we can work without it.
    var indexOfTasks = timeline.innerHTML.indexOf('<div class="ms-tl-today"');
    if( indexOfTasks > -1){ // Using "timeline" as first variable in the next line does not work. We have to re-catch the HTML section.
        timeline.innerHTML = timeline.innerHTML.substring( 0, indexOfTasks ); // We are deleting the tasks that were in the
    }else{
        // If the "Today"'s flag is not present...
        indexOfTasks = timeline.innerHTML.indexOf('<div class="timeline-dates"');
        if(indexOfTasks > -1){
            timeline.innerHTML = timeline.innerHTML.substring( 0, indexOfTasks ); // We are deleting the tasks that were in the
        }
    }

    // Defining the dates to write on the X axis of the timeline
    var datesStrings = new Array();
    var today = new Date();
    var timeSpan; // It will signal to the createDateString() function the kind of string we want in output.

    if(numOfDayInTimeline == 7){ timeSpan = 'week'; }
    else{
        if(numOfDayInTimeline == 15){ timeSpan = 'twoWeeks'; }
        else{
            if(numOfDayInTimeline == 31){ timeSpan = 'month'; }
            else{
                if(numOfDayInTimeline == 90){
                    // Re-calculating the number of days int he three months according to the months considered.
                    numOfDayInTimeline = daysInMonth(today.getMonth(),today.getFullYear()); // Adding the days in this month
                    today.setDate(1);
                    today.setMonth( today.getMonth() -1 );
                    numOfDayInTimeline += daysInMonth(today.getMonth(),today.getFullYear()); // Adding the days in the previous
                }
                month
                today.setMonth( today.getMonth() +1);
                today.setMonth( today.getMonth() +1);
                numOfDayInTimeline += daysInMonth(today.getMonth(),today.getFullYear()); // Adding the days in the following
                month
                today = new Date(); // Resetting 'today'

                timeSpan = 'threeMonths'; }
            else{

```

```

        if(numOfDaysInTimeline == 210){ timeSpan = 'sevenMonths'; }
        else{
            if( numOfDaysInTimeline == 365){ timeSpan = 'year'; }
            else{ timeSpan = 'year'; }
        }
    }
}

// Adding the flag that signals today's date on the timeline
/*
if( timeSpan == 'week'){
    // "Today" label's code
    // I do not know why but SharePoint displays this label differently when adding it to the webpage the first time or some other time through Javascript and the
radio buttons.
    if(addTasksToTimelineFlag == false){ // If this is the first time that the function has been called...
        var todayLabel = '<div class="ms-tl-today" style="position:absolute; background-color:rgb(0, 114, 198); color:white; z-index=14; border-color:
rgb(0, 114, 198); height: 22px; top: 42px; left: -605px;"></div>' +
            '<div class="ms-tl-todayLabel" style="position:absolute; background-color:rgb(0,
114, 198); color:white; text-align:center; z-index=14; width: 49px; height: 20px; top: 23px; left: -628px; background-color: rgb(0, 114, 198);">Today</div>';
        timeline.innerHTML += todayLabel;
    }else{
        var todayLabel = '<div class="ms-tl-today" style="position:absolute; background-color:rgb(0, 114, 198); color:white; z-index=14; border-color:
rgb(0, 114, 198); height: 22px; top: -23px; left: 227px;"></div>' +
            '<div class="ms-tl-todayLabel" style="position:absolute; background-color:rgb(0,
114, 198); color:white; text-align:center; z-index=14; width: 49px; height: 20px; top: -42px; left: 204px; background-color: rgb(0, 114, 198);">Today</div>';
        timeline.innerHTML += todayLabel;
    }
}

if(timeSpan == 'twoWeeks'){
    // "Today" label's code
    var todayLabel = '<div class="ms-tl-today" style="position:absolute; background-color:rgb(0, 114, 198); color:white; z-index=14; border-color:
rgb(0, 114, 198); height: 24px; top: -24px; left: 106px;"></div>' +
        '<div class="ms-tl-todayLabel" style="position:absolute; background-color:rgb(0,
114, 198); color:white; text-align:center; z-index=14; width: 49px; height: 20px; top: -42px; left: 81px; background-color: rgb(0, 114, 198);">Today</div>';
    timeline.innerHTML += todayLabel;
}

if(timeSpan == 'month'){
    // "Today" label's code
    var todayLabel = '<div class="ms-tl-today" style="position:absolute; background-color:rgb(0, 114, 198); color:white; z-index=14; border-color:
border-color: rgb(0, 114, 198); height: 24px; top: -24px; left: 119px;"></div>' +
        '<div class="ms-tl-todayLabel" style="position:absolute;
background-color:rgb(0, 114, 198); color:white; text-align:center; z-index=14; width: 49px; height: 20px; top: -42px; left: 95px; background-color: rgb(0, 114, 198);">Today</div>';
    timeline.innerHTML += todayLabel;
}
}
*/

// Creating local variables
var horizontalSectionWidth = timelineWidth/numOfDaysInTimeline; // We need to keep it as float, because when it comes to the "year" time span we
have to keep the decimal values to keep the result for the left margin of every section correct.
var remainingHorizontalSpace = timelineWidth - (horizontalSectionWidth*numOfDaysInTimeline);
var minDate = new Date();
var maxDate = new Date();
if( timeSpan == 'week' ){
    // Saving the first date on the left and the last date on the right of the timeline.
    minDate.setDate( today.getDate() -1); // The 10% of the timeSpan regards the past
    maxDate.setDate( today.getDate() + numOfDaysInTimeline-2 ); // The 90% of the timeSpan regards the future (90% less the present
day)

    today.setDate( today.getDate() - 2 ); // Bringing the today's date back to the first day of the timeline less one.
    for( var i=0; i<numOfDaysInTimeline; i++){
        today.setDate( today.getDate() + 1 ); // Updating the date object that we want to pass to the function
createDateString()
        datesStrings.push( createDateString(today, timeSpan) ); //
Creating the dates strings passing to the function
    }
}

if( timeSpan == 'twoWeeks' || timeSpan == 'month' ){
    // Saving the first date on the left and the last date on the right of the timeline.
    minDate.setDate( today.getDate() - (Math.floor(numOfDaysInTimeline/10) ) ); // The 10% of the timeSpan regards the past
less the present day)
    maxDate.setDate( today.getDate() + (Math.floor(numOfDaysInTimeline/10*9) ) ); // The 90% of the timeSpan regards the future (90%
less one.

    today.setDate( today.getDate() - (Math.floor(numOfDaysInTimeline/10)+1) ); // Bringing the today's date back to the first day of the timeline
less one.
    for( var i=0; i<numOfDaysInTimeline; i++){
        today.setDate( today.getDate() + 1 ); // Updating the date object that we want to pass to the function
createDateString()
        datesStrings.push( createDateString(today, timeSpan) ); //
Creating the dates strings passing to the function
    }
}

if( timeSpan == 'threeMonths' ){
    // Saving the first date on the left and the last date on the right of the timeline.
    // minDate = the present month
    minDate.setDate(1); // The first day of the present month
    maxDate = new Date();
    maxDate.setMonth( maxDate.getMonth() +3 ); // To have the maxDate set on the last day of the next month we can forward by 2
months and then move 1 day back.

    maxDate.setDate(1); // Setting the date on the first day of the month
    maxDate.setDate( maxDate.getDate() - 1 ); // Going to the last day of the previous month for maxDate (two months ahead for
us).

    datesStrings.push( createDateString(minDate, timeSpan) );
    minDate.setMonth(minDate.getMonth()+1);
    datesStrings.push( createDateString(minDate, timeSpan) );
    minDate.setMonth(minDate.getMonth()-1); // Bringing the minDate back to its original date (for the future
operations that use this variable)
    datesStrings.push( createDateString(maxDate, timeSpan) );
}
}

```

```

        if( timeSpan == 'sevenMonths' ){
            // Saving the first date on the left and the last date on the right of the timeline.
            if(minDate.getDate() == 31){ minDate.setDate(30); } // We need this check because the next
function rewinds at most for 30 days (Javascript bug)

            minDate.setMonth( minDate.getMonth() -1 );
            minDate.setDate(1); // The result will be the first day of the previous month.
            maxDate = new Date();
            maxDate.setMonth( maxDate.getMonth() +6 ); // To have the maxDate set on the last day of the next month we can
forward by 2 months and then move 1 day back.

            maxDate.setDate(1); // Setting the date on the first day of the month
            maxDate.setDate( maxDate.getDate() -1 ); // Going to the last day of the previous month for maxDate (next month
for us).

            if(maxDate.getMonth() === 'January' ){ maxDate.setFullYear( minDate.getFullYear() +1 ) }

            datesStrings.push( createDateString(minDate, timeSpan) );
            minDate.setMonth(minDate.getMonth()+1);
            datesStrings.push( createDateString(minDate, timeSpan) );
            minDate.setMonth(minDate.getMonth()+1);
            datesStrings.push( createDateString(minDate, timeSpan) );
            minDate.setMonth(minDate.getMonth()+1);
            datesStrings.push( createDateString(minDate, timeSpan) ); // This month
            minDate.setMonth(minDate.getMonth()+1);
            datesStrings.push( createDateString(minDate, timeSpan) );
            minDate.setMonth(minDate.getMonth()+1);
            datesStrings.push( createDateString(minDate, timeSpan) );
            minDate.setMonth(minDate.getMonth()-5); // Bringing the minDate back to its original date (for
the future operations that use this variable)

        }else{
            if( timeSpan == 'year' ){
                // Saving the first date on the left and the last date on the right of the timeline.
                minDate = new Date(new Date().getFullYear(), 0, 1); // First of January
                maxDate = new Date(new Date().getFullYear(), 11, 31); // End of this Year

                datesStrings.push("January");
                datesStrings.push("February");
                datesStrings.push("March");
                datesStrings.push("April");
                datesStrings.push("May");
                datesStrings.push("June");
                datesStrings.push("July");
                datesStrings.push("August");
                datesStrings.push("September");
                datesStrings.push("October");
                datesStrings.push("November");
                datesStrings.push("December");
            }
        }
    }
}

minDate.setHours(0);
minDate.setMinutes(0);
minDate.setSeconds(0);
minDate.setMilliseconds(0);
maxDate.setHours(0);
maxDate.setMinutes(0);
maxDate.setSeconds(0);
maxDate.setMilliseconds(0);

// Creating the HTML code for the X axis of the timeline (the dates)
var datesAxisString = ''; // The dates for the X axis in the timeline
var separatorsString = ''; // The string with the small vertical separators between the dates on the timeline
var timelineNumColumns;
if(timeSpan == 'week'){ timelineNumColumns = 7; }
else if(timeSpan == 'twoWeeks'){ timelineNumColumns = 15; }
else if(timeSpan == 'month'){ timelineNumColumns = 31; }
else if(timeSpan == 'threeMonths'){ timelineNumColumns = 3; } // 3 months
else if(timeSpan == 'sevenMonths'){ timelineNumColumns = 7; } // 7 months
else if(timeSpan == 'year'){ timelineNumColumns = 12; }}}} // 12 months

var sectionWidthInTimeline = Math.floor(timelineWidth/timelineNumColumns)-5; // Adjusting the width counting the padding-left property in the space sections

for(var i=0; i<timelineNumColumns; i++){
    if(i==timelineNumColumns-1){
        datesAxisString += '<span style="white-space: nowrap; overflow: hidden; height: 100%; font-family: \'Segoe UI\'; font-size: 8pt; float: left;
text-align: left; padding-left: 5px; padding-bottom: 5px; width: '+ (sectionWidthInTimeline+remainingHorizontalSpace) +'px;">' + datesStrings[i] + '</span>';
        separatorsString += '<span style="white-space: nowrap; overflow: hidden; height: 100%; float: left; border-left-width: 1px; border-left-style: solid;
border-color: rgb(213, 213, 213); width: '+ (sectionWidthInTimeline+remainingHorizontalSpace+4) +'px;"></span>';
    }else{
        datesAxisString += '<span style="white-space: nowrap; overflow: hidden; height: 100%; font-family: \'Segoe UI\'; font-size: 8pt; float: left;
text-align: left; padding-left: 5px; padding-bottom: 5px; width: '+ sectionWidthInTimeline +'px;">' + datesStrings[i] + '</span>';
        separatorsString += '<span style="white-space: nowrap; overflow: hidden; height: 100%; float: left; border-left-width: 1px; border-left-style:
solid; border-color: rgb(213, 213, 213); width: '+ (sectionWidthInTimeline+4) +'px;"></span>';
    }
}

// The timeline is 1060px width, so we have to split this value in the number of subsections we want to create (es. number of days, hours of the day ecc...)
// We are adding 220px to the left margin to make space for the categories and equipment.
var timelineXaxis = '<div class="timeline-dates" style="width: '+ (timelineWidth+1) +'px; height:20px; margin-left:220px; left:0px; top:-20px; padding-right:28px;"> ' +
'<div style="padding: 0px; left: 0px; top: 0px; overflow: hidden; position:inherit; height:
19px; background-repeat: repeat-x;"></div> ' +
'<div style="white-space:nowrap; overflow:hidden; position: relative; color: rgb(119, 119, 119);
border-bottom-width: 1px; border-bottom-style: hidden; border-bottom-color: rgb(119, 119, 119); height: 20px; top: 0px; left: 0px; margin-left: 2px;"> ' +
    datesAxisString +
'</div> ' +
'<div style="white-space:nowrap; overflow:hidden; position:relative; color: rgb(119, 119, 119);
border-bottom-width: 1px; border-bottom-style: hidden; border-bottom-color: rgb(119, 119, 119); height: 10px; top: -16px; left: 0px; margin-left: 0px;">' +
    separatorsString +
'</div>' +
'</div>' +

```



```

/* This function returns the number of days contained in the considered month (considers also the leap years).
 * Input:
 * - the month and the year considered. */
function daysInMonth(month,year) {
    month += 1; // Month has to be 1 based -> [1,12] instead of the Javascript usual zero-based month -> [0,11]
    return new Date(year, month, 0).getDate();
}

/* This function creates the string that will display the date and time of each feed and reply.
 * Input:
 * - the date object of the feed or reply
 * - the string defining the considered time span. */
function createDateString(dateObj, timeSpan){

    var day = dateObj.getDay();
    var month = dateObj.getMonth();

    if(timeSpan == 'week' || timeSpan == 'twoWeeks'){
        switch(day){
            case 0: day="Sun"; break;
            case 1: day="Mon"; break;
            case 2: day="Tue"; break;
            case 3: day="Wed"; break;
            case 4: day="Thu"; break;
            case 5: day="Fri"; break;
            case 6: day="Sat"; break;
            default: day = "Mon"; break;
        }
    }

    if(timeSpan == 'week'){
        switch(month){
            case 0: month="January"; break;
            case 1: month="February"; break;
            case 2: month="March"; break;
            case 3: month="April"; break;
            case 4: month="May"; break;
            case 5: month="June"; break;
            case 6: month="July"; break;
            case 7: month="August"; break;
            case 8: month="September"; break;
            case 9: month="October"; break;
            case 10: month="November"; break;
            case 11: month="December"; break;
            default: month="January"; break;
        }
    }

    }else{
        switch(month){
            case 0: month="Jan"; break;
            case 1: month="Feb"; break;
            case 2: month="Mar"; break;
            case 3: month="Apr"; break;
            case 4: month="May"; break;
            case 5: month="Jun"; break;
            case 6: month="Jul"; break;
            case 7: month="Aug"; break;
            case 8: month="Sep"; break;
            case 9: month="Oct"; break;
            case 10: month="Nov"; break;
            case 11: month="Dec"; break;
            default: month="Jan"; break;
        }
    }

    }else{
        if(timeSpan == 'month'){
            switch(day){
                case 0: day="Sun";

```

```

        case 1: day="Mon";           break;
        case 2: day="Tue";           break;
        case 3: day="Wed";           break;
        case 4: day="Thu";           break;
        case 5: day="Fri";           break;
        case 6: day="Sat";           break;
        default: day = "Mon";        break;
    }

    switch(month){
    case 0: month="Jan";             break;
    case 1: month="Feb";            break;
    case 2: month="Mar";            break;
    case 3: month="Apr";            break;
    case 4: month="May";            break;
    case 5: month="Jun";            break;
    case 6: month="Jul";            break;
    case 7: month="Aug";            break;
    case 8: month="Sep";            break;
    case 9: month="Oct";            break;
    case 10: month="Nov";           break;
    case 11: month="Dec";           break;
    default: month="Jan";           break;
    }
}

else{
    if( timeSpan == 'threeMonths' || timeSpan == 'sevenMonths'){
        switch(month){
            case 0: month="January"; break;
            case 1: month="February"; break;
            case 2: month="March";    break;
            case 3: month="April";    break;
            case 4: month="May";      break;
            case 5: month="June";     break;
            case 6: month="July";     break;
            case 7: month="August";   break;
            case 8: month="September"; break;
            case 9: month="October";  break;
            case 10: month="November"; break;
            case 11: month="December"; break;
            default: month="January"; break;
        }
    }
}

var numberOfTheDay = dateObj.getDate(); // Returns the day of the month (from 1-31)
if(numberOfTheDay < 10){
    numberOfTheDay = '0' + parseInt(numberOfTheDay, 10); // This way if the month is the 5th it will be displayed as "05", instead of "5"
}

if(timeSpan == 'week' || timeSpan == 'twoWeeks'){
    return day+ ' ' +numberOfTheDay+ ' ' +month;
}

else{
    if(timeSpan == 'month'){
        return numberOfTheDay+ ' ' +month;
    }

    else{
        return month;
    }
}
}
}

```

File "newTaskForm.js":

```

<script src="//code.jquery.com/jquery-1.11.0.min.js"></script>
<script src="/_layouts/15/clientpeoplepicker.js"></script>
<script type="text/javascript">
    // This jQuery configuration is MANDATORY to work with "_spPageContextInfo".
    $(document).ready(function ()
    {
        // Updating the "Save" button in the "new task" form.
        updatesaveButtonOnClickEvent();

        if(window.location.pathname.endsWith("NewForm.aspx"))
        {
            if (SP.ClientContext != null) {
                SP.SOD.executeOrDelayUntilScriptLoaded(GetCurrentUser, 'SP.js');
            }
            else {
                SP.SOD.executeFunc('sp.js', null, GetCurrentUser);
            }
        }

        // This code adds the Username of the current user that is creating the new task to the "Assigned To" field (as a default content).
        function GetCurrentUser()
        {
            var userid = _spPageContextInfo.userId;
            var requestUri = _spPageContextInfo.webAbsoluteUrl + "/_api/web/getuserbyid(" + userid + ")";
            var requestHeaders = { "accept" : "application/json;odata=verbose" };
            $.ajax({ url : requestUri, contentType : "application/json;odata=verbose", headers : requestHeaders, success : onSuccess, error : onError});
        }

        function onSuccess(data, request)
        {
            var loginName = data.d.LoginName;
            SetUserFieldValue("Assigned To",loginName);
        }

        function onError(error)
        {
            //alert(error);
        }

        function SetUserFieldValue(fieldName, userName)
        {
            var _peoplePicker = $("div[title='" + fieldName + "']");
            var _peoplePickerTopId = _peoplePicker.attr('id');
            var _peoplePickerEditor = $("input[title='" + fieldName + "']");
            _peoplePickerEditor.val(userName);
            var _peoplePickerObject = SPClientPeoplePicker.SPClientPeoplePickerDict[_peoplePickerTopId];
            _peoplePickerObject.AddUnresolvedUserFromEditor(true);
        }

        // This function updates the content of the onclick string of the 'Save' button of the new task form.
        function updatesaveButtonOnClickEvent(){
            var oldOnClickString = $("input[value='Save']").attr('onclick');
            var newOnClickString = 'if(consistencyCheckOnDates()){ ' + oldOnClickString + '};else(alert("The selected equipment is not available in the chosen period. Please enter different dates.));';

            // Updating the onclick event
            $("input[value='Save']").attr('onclick', newOnClickString);
        }

    });

    // Global variable
    var siteUrl = document.URL; // It is going to be something like: https://espace2013-dev.cern.ch/test-Timeline/Lists/Tasks/AllItems.aspx
                                                                    //           We           want           something           like
    "https://espace2013.cern.ch/test-Timeline"

    // Global functions (available for the code in the webpage):

    /* INPUT VALIDATION
    * This function is activated when pressing the "Save" button in the form.
    * It checks if the dates entered in the form are valid (if the chosen period intersect with another task's period).
    * We need to have any equipment booked at most for one task on a certain date. */
    function consistencyCheckOnDates(){
        // Reading the dates entered in the form
        var formStartDate = $("input[title='Start Date']")[0].value;
        var formDueDate = $("input[title='Due Date']")[0].value;
        var formSelectedEquipmentName;
        var formSelectedEquipmentArray = new Array();

        formSelectedEquipmentName = $("select[title='Equipment Name']").find('option:selected').text(); // Reading the selected equipment from
the drop-down list.

        if(formSelectedEquipmentName == null || formSelectedEquipmentName == undefined || formSelectedEquipmentName == ''){
            formSelectedEquipmentName = $("select[title='Equipment Name selected values']")[0]; // Reading the selected equipment from
the drop-down list.

            for(var i=0; i<formSelectedEquipmentName.length; i++){
                formSelectedEquipmentArray.push( formSelectedEquipmentName[i].innerHTML );
            }
        }

        // Creating the Date objects in Javascript.

```

```

var year, month, day;
try{
    day = formStartDate.split('/')[0];
    month = formStartDate.split('/')[1];
    year = formStartDate.split('/')[2];

    formStartDate = new Date(year, (month-1), day, 0,0,0,0);

    day = formDueDate.split('/')[0];
    month = formDueDate.split('/')[1];
    year = formDueDate.split('/')[2];

    formDueDate = new Date(year, (month-1), day, 0,0,0,0);
}catch(e){ return false; }

var listItemEnumerator = tasksListItems.getEnumerator();
while ( listItemEnumerator.moveNext() ) {
var oListItem = listItemEnumerator.get_current();
var equipmentName;
try{
    equipmentName = oListItem.get_item('Equipment_x0020_Name');
}catch(e){ continue; } // In case of exception we allow the creation of the task.

var equipmentsArray = new Array();
if(equipmentName.$2e_1 == null || equipmentName.$2e_1 == undefined){
    equipmentsArray = equipmentName;
}

// If there is more than one equipment...
if(equipmentsArray.length > 0){
    // we check the dates regarding each of the equipments
    for(var tempIndex=0; tempIndex<equipmentsArray.length; tempIndex++){
        equipmentName = equipmentsArray[tempIndex].$2e_1;

        if( formSelectedEquipmentArray.indexOf(equipmentName) > -1 ){ // If the task is using
the chosen equipment...

            var startDate = oListItem.get_item('StartDate');
            var dueDate = oListItem.get_item('DueDate');

            // Checking if the chosen dates fall inside the period in which another task has to be executed.
            var startDateNotValid = formStartDate >= startDate && formStartDate <= dueDate; // If the formStartDate falls in the
period already chosen for another task -> True.
            var dueDateNotValid = formDueDate >= startDate && formDueDate <= dueDate;
            // If the formDueDate falls in the period already chosen for another task -> True.
            var periodNotValid = (formStartDate < startDate && formDueDate >= startDate) || (formDueDate > dueDate &&
formStartDate <= dueDate); // If the chosen period comprehends the period chosen for this task.
            // If the start date or the due date chosen are falling inside the chosen period we have to tell the User to select
different dates.
            if( startDateNotValid || dueDateNotValid || periodNotValid){
                return false;
            }
        }
    }
}else{
    if(equipmentName == formSelectedEquipmentName){ // If the task is using the
        var startDate = oListItem.get_item('StartDate');
        var dueDate = oListItem.get_item('DueDate');

        // Checking if the chosen dates fall inside the period in which another task has to be executed.
        var startDateNotValid = formStartDate >= startDate && formStartDate <= dueDate; // If the formStartDate falls in the period already
chosen for another task -> True.
        var dueDateNotValid = formDueDate >= startDate && formDueDate <= dueDate; // If
the formDueDate falls in the period already chosen for another task -> True.
        var periodNotValid = (formStartDate < startDate && formDueDate >= startDate) || (formDueDate > dueDate && formStartDate <=
dueDate); // If the chosen period comprehends the period chosen for this task.
        // If the start date or the due date chosen are falling inside the chosen period we have to tell the User to select different
dates.
        if( startDateNotValid || dueDateNotValid || periodNotValid){
            return false;
        }
    }
}

// If all the Tasks using the selected equipment have been examined and no one is using the equipment in the chosen period we can say that everything is ok.
return true;
}
}

</script>

```

File “customTimeline.js”:

```

// Adding jQuery to the webpage
document.write("<script type='text/javascript' src='//code.jquery.com/jquery-1.11.0.min.js'></script>");

// Global variables that hold the names of the lists on SharePoint
var globalUsersListName = 'Users'; // The list containing the name of the User and the color assigned to it (the color is saved as a
string).
var globalEquipmentsListName = 'Equipments'; // The list containing the name of the Equipments and their relative Category (lookup field)
var globalProjectsListName = 'Projects'; // The list containing the name of the Project and the color assigned to it (the color is saved as a string).
var globalTasksListName = 'Tasks'; // The list containing the name of the Tasks and their relative Project (lookup field)

// Calling the first function

```

```

ExecuteOrDelayUntilScriptLoaded(registerRenderer, 'clienttemplates.js'); // Telling to the webpage to launch our postTaskFormRenderer() function during the post rendering phase.

function registerRenderer()
{
    var ctxForm = {};
    ctxForm.Templates = {};
    ctxForm.OnPreRender = OnPreRenderDocItemTemplate;

    SPClientTemplates.TemplateManager.RegisterTemplateOverrides(ctxForm);
}

// The system tries to prerender 3 times.
// The first one is useless in Chrome and Firefox (it is useful in IE), it gives us no data from the Server, so we can avoid it. After that call on Firefox it works fine, while on Chrome
the system gives us the error:
// Uncaught Error: The collection has not been initialized. It has not been requested or the request has not been executed. It may need to be explicitly requested.
// but it does not matter. The code makes the third call that solves the problem even on Chrome.
var firstCallAlreadyMade = false;
function OnPreRenderDocItemTemplate(renderCtx) {
    SP.SOD.executeOrDelayUntilScriptLoaded(loadContext, 'sp.js');
    function loadContext() {
        var ua = window.navigator.userAgent;
        var msie = ua.indexOf("MSIE ") > -1 || !navigator.userAgent.match(/Trident.*rv:11\./); // "True" if the Browser is IE (with support for IE
11).

        if(msie){
            checkSituationAndLunch();
        }else{
            if(firstCallAlreadyMade == false){
                checkSituationAndLunch();
                firstCallAlreadyMade = true;
            }
        }
    }
}

function checkSituationAndLunch(){
    try{
        // This control has been implemented since for some actions SharePoint refreshes the webparts without refreshing the whole webpage.
        // We are talking about operations like expanding or collapsing a Group of Tasks.
        if( document.getElementById("innerTimeline") ){ // If the timeline is already in the webpage...
            return; // Do not add code to the timeline.
        }
        // Reading the Equipments and the Categories from SharePoint
        retrieveEquipmentsAndCategories();
    }catch(e){ return; }
}

/***** Timeline customization code: through this code we can display the timeline above the "New Task" form. *****/

// Retrieving information on every equipment and every category from SharePoint.
function retrieveEquipmentsAndCategories() {
    /* In the "New Task" form this function is called many times and the variable "equipmentListItems" is re-written for many times.
    * This leads to a race condition when the first sequence of function tries in the code to read some data from it in order to display the "Equipments" and their "Categories",
    * thus causing, some times, to find the resource locked and so having as output the HTML section thought for the equipments empty. */
    if (this.equipmentListItems != null && this.equipmentListItems != undefined){
        return;
    }

    var siteUrl = document.URL; // It is going to be something like: https://espace2013-dev.cern.ch/test-Timeline/Lists/Tasks/AllItems.aspx
    // We want something like "https://espace2013.cern.ch/test-Timeline"

    // Correcting the URL (if necessary)
    if( siteUrl == null || siteUrl == undefined ){ siteUrl = window.location.href; }
    var numSlashes = 0;
    for( var i=0; i<siteUrl.length; i++){
        if(siteUrl[i] == '/'){
            numSlashes++;
            if(numSlashes == 4){
                siteUrl = siteUrl.substring(0, i);
                break;
            }
        }
    }

    var clientContext = new SP.ClientContext(siteUrl);
    var oList = clientContext.get_web().get_lists().getTitle( globalEquipmentsListName );

    var camlQuery = new SP.CamlQuery();
    camlQuery.set_viewXml('<View><Query><Where>' +
        '</Where></Query></View>');

    this.equipmentListItems = oList.getItems(camlQuery);
    clientContext.load(equipmentListItems);

    clientContext.executeQueryAsync(Function.createDelegate(this, this.onQuerySucceededEquipments), Function.createDelegate(this, this.onQueryFailed));
}

// Reading the categories and equipments
var globalEquipmentsArray = new Array(); // This global variable will keep the names of the Equipment to let the User change timespan in the timeline if needed, without
recalling the Server for this information.
var globalCategoriesArray = new Array(); // This global variable will keep the names of the Categories.
var globalCustomTimelineEquipmentsHTMLString = '';
function onQuerySucceededEquipments(){
    var equipmentList = '<div style="float:left; margin-top: 42px;">';
    var numRows = 0; // Variable used to know the amount of rows to display.
    var firstItem = true; // Boolean to treat differently the first item of the array. It needs a greater padding-top.
    var listItemEnumerator = equipmentListItems.getEnumerator();

```



```

var innerListItemEnumerator = equipmentListItems.getEnumerator();
var category = '';
var firstOfEquipments = true;

var ua = window.navigator.userAgent;
var msie = ua.indexOf("MSIE ") > -1 || !!navigator.userAgent.match(/Trident.*rv:\d\.\d/); // "True" if the Browser is IE (with support for IE 11).
var firefox = ua.toLowerCase().indexOf('firefox') > -1; // Detects any version of Firefox. "True" if we are using Firefox;

while (listItemEnumerator.moveNext()) {
var oListItem = listItemEnumerator.get_current();
try{
category = oListItem.get_item('Parent_x0020_Category').$2e_1;
if(category != null && globalCategoriesArray.indexOf( category ) == -1 ){ // If we have not met this category before...
globalCategoriesArray.push(category); // We add it to the 'globalCategoriesArray'

// setting local variables
var equipmentName, innerListItem;
var tempEquipmentsString = '';
var firstOfCategory = true;
innerListItemEnumerator = equipmentListItems.getEnumerator(); // Resetting the 'innerListEnumerator'

// Seek for every equipment belonging to that category and add it to the HTML string.
while(innerListItemEnumerator.moveNext() ){
var innerListItem = innerListItemEnumerator.get_current();
if( category == innerListItem.get_item('Parent_x0020_Category').$2e_1 ){ // If the currently considered category
equipmentName = innerListItem.get_item('Title');
globalEquipmentsArray.push(equipmentName); // Memorizing the name of the Equipment. We will need it later while
displaying the Tasks in the timeline

var stringHeight;
if(msie || firefox){ // If IE or Firefox...
stringHeight = 9;
}else{
stringHeight = 22;
}

if(firstOfCategory){ // If it is the first element of a category...
if(firstOfEquipments){ // If it is the first line of equipments to be written (in absolute)...
tempEquipmentsString += '<p id="categoriesList" style="padding:0; margin-top:3px; margin-left:-110px; height:'+ stringHeight +'px; -webkit-margin-before: 0em!important; -webkit-margin-after: 0em!important;"><span style="display:inline-block; margin-top:11px; width:120px;" title="'+ category +'>' + category + '</span> <span style="display:inline-block; margin-top:10px; width:200px; white-space:nowrap; text-overflow:ellipsis; overflow:hidden; margin-bottom: -3px;" title="'+ equipmentName +'>' + equipmentName +</span></p>';
} else{
firstOfEquipments = false;
}
}else{
tempEquipmentsString += '<p id="categoriesList" style="padding:0; margin-left:-110px; height:'+ stringHeight +'px; -webkit-margin-before: 0em!important; -webkit-margin-after: 0em!important;"><span style="display:inline-block; margin-top:11px; width:120px;" title="'+ category +'>' + category + '</span> <span style="display:inline-block; margin-top:10px; width:200px; white-space:nowrap; text-overflow:ellipsis; overflow:hidden; margin-bottom: -3px;" title="'+ equipmentName +'>' + equipmentName +</span></p>';
}

}
firstOfCategory = false; // This has to be done in any case
}else{
tempEquipmentsString += '<p id="categoriesList" style="padding:0; margin-left:-110px; height:'+ stringHeight +'px; -webkit-margin-before: 0em!important; -webkit-margin-after: 0em!important;"><span style="display:inline-block; margin-top:11px; width:120px;" </span> <span style="display:inline-block; margin-top:10px; width:200px; white-space:nowrap; text-overflow:ellipsis; overflow:hidden;" title="'+ equipmentName +'>' + equipmentName +</span></p>';
}

numRows++; // Increasing the rows' counter (Used to set the height of the Timeline with the Tasks)
}
}

equipmentList += tempEquipmentsString;
}
}catch(e){}

equipmentList += '</div>';

// Passing the local variables' HTML string to the global ones.
globalCustomTimelineEquipmentsHTMLString = equipmentList;

// Calling the next function for the retrieval of the Projects
retrieveProjects();
}

function retrieveProjects(){
var siteUrl = document.URL; // It is going to be something like: https://espace2013-dev.cern.ch/test-Timeline/Lists/Tasks/AllItems.aspx // We want something like "https://espace2013.cern.ch/test-Timeline"

// Correcting the URL (if necessary)
if( siteUrl == null || siteUrl == undefined ){ siteUrl = window.location.href; }
var numSlashes = 0;
for( var i=0; i<siteUrl.length; i++){
if(siteUrl[i] == '/'){
numSlashes++;
if(numSlashes == 4){
siteUrl = siteUrl.substring(0, i);
break;
}
}
}

var clientContext = new SP.ClientContext(siteUrl);
var oList = clientContext.get_web().get_lists().getByTitle( globalProjectsListName );

var camlQuery = new SP.CamlQuery();
camlQuery.set_viewXml('<View><Query><Where> + '</Where></Query></View>');

this.projectListItems = oList.getItems(camlQuery);
clientContext.load(projectListItems);

clientContext.executeQueryAsync(Function.createDelegate(this, this.onQuerySucceededProjects), Function.createDelegate(this, this.onQueryFailed));
}

```

```

var globalProjectsArray = new Array(); // This global variable will keep the names of the Projects and their associated colors.
function onQuerySucceededProjects() {
    // Variables necessary to read the query results
    var listItemEnumerator = projectListItems.getEnumerator();
    var innerListItemEnumerator = projectListItems.getEnumerator();

    var project, color;
    while (listItemEnumerator.moveNext()) {
        var oListItem = listItemEnumerator.get_current();
        try {
            project = oListItem.get_item('Title');
            color = oListItem.get_item('Color');
            globalProjectsArray.push({'projectName':project, 'projectColor':color});
        } catch (e) {}
    }

    // Calling the next function for the retrieval of the Users
    retrieveUsers();
}

function retrieveUsers() {
    var siteUrl = document.URL; // It is going to be something like: https://espace2013-dev.cern.ch/test-Timeline/Lists/Tasks/AllItems.aspx
    // We want something like "https://espace2013.cern.ch/test-Timeline"

    // Correcting the URL (if necessary)
    if (siteUrl == null || siteUrl == undefined) { siteUrl = window.location.href; }
    var numSlashes = 0;
    for (var i=0; i<siteUrl.length; i++) {
        if (siteUrl[i] == '/') {
            numSlashes++;
            if (numSlashes == 4) {
                siteUrl = siteUrl.substring(0, i);
                break;
            }
        }
    }

    var clientContext = new SP.ClientContext(siteUrl);
    var oList = clientContext.get_web().get_lists().getByTitle(globalUsersListName);

    var camlQuery = new SP.CamlQuery();
    camlQuery.set_viewXml('<View><Query><Where>' +
        '</Where></Query></View>');

    this.userListItems = oList.getItems(camlQuery);
    clientContext.load(userListItems);

    clientContext.executeQueryAsync(Function.createDelegate(this, this.onQuerySucceededUsers), Function.createDelegate(this, this.onQueryFailed));
}

var globalUsersArray = new Array(); // This global variable will keep the names of the Users (saved in a list on SharePoint).
function onQuerySucceededUsers() {
    // Variables necessary to read the query results
    var listItemEnumerator = userListItems.getEnumerator();
    var innerListItemEnumerator = userListItems.getEnumerator();

    var user, color;
    while (listItemEnumerator.moveNext()) {
        var oListItem = listItemEnumerator.get_current();
        try {
            user = oListItem.get_item('User');
            color = oListItem.get_item('Color');
            globalUsersArray.push({'userName':user, 'userColor':color});
        } catch (e) {}
    }

    // Calling the next function for the retrieval of the Tasks
    retrieveTasksListItems();
}

// Retrieving information about each of the Tasks and adding them to the Timeline
function retrieveTasksListItems() {
    var siteUrl = document.URL; // It is going to be something like: https://espace2013-dev.cern.ch/test-Timeline/Lists/Tasks/AllItems.aspx
    // We want something like "https://espace2013.cern.ch/test-Timeline"

    // Correcting the URL (if necessary)
    if (siteUrl == null || siteUrl == undefined) { siteUrl = window.location.href; }
    var numSlashes = 0;
    for (var i=0; i<siteUrl.length; i++) {
        if (siteUrl[i] == '/') {
            numSlashes++;
            if (numSlashes == 4) {
                siteUrl = siteUrl.substring(0, i);
                break;
            }
        }
    }

    var clientContext = new SP.ClientContext(siteUrl);
    var oList = clientContext.get_web().get_lists().getByTitle(globalTasksListName);

    var camlQuery = new SP.CamlQuery();
    camlQuery.set_viewXml('<View><Query><Where>' +
        '</Where></Query></View>');

    this.taskListItems = oList.getItems(camlQuery);
    clientContext.load(taskListItems);

    clientContext.executeQueryAsync(Function.createDelegate(this, this.onQuerySucceededTasks), Function.createDelegate(this, this.onQueryFailed));
}

function onQueryFailed(sender, args) {
    alert("There has been a problem communicating with the Server. Please try again later.");
    console.log('Request failed. ' + args.get_message() + '\n' + args.get_stackTrace());
}

```

```

}

// Reading the Tasks saved in the timeline
var addTasksToTimelineFlag = false; // This variable will tell the system if the function "addTasksToTimeline" has already been called at least once or not.
function onQuerySucceededTasks() {
    var numRows = globalEquipmentsArray.length;
    var equipmentsHTMLstring = globalCustomTimelineEquipmentsHTMLstring;

    // Catching the rows of the timeline in the HTML code
    var timelineArea = $('div[id^="MSOZoneCell_"]')[0]; // Chatching the HTML section in which we want to add the customized timeline
    // e.g.: id="MSOZoneCell_WebPart...WPQ3"

    $(timelineArea).prepend('<div id="timelineArea"></div>');
    timelineArea = document.getElementById("timelineArea");

    // Modifying the CSS for the section to include the Rows on the Left of the Timeline.
    // (if dynamic width) Each character equals 0.7em, so the amount of space on the left has to be 0.7*maxNumCharacters.
    timelineArea.style.paddingLeft = "120px"; // (0.7*maxNumCharacters) + 'em'; // Making space on the left of the Timeline for the Rows' titles.
    timelineArea.style.height = ((22*numRows)+70) + 'px'; // Expanding the area including the timeline to push down the rest of the webpage (the list containing the Tasks).

    // Creating some radio buttons to enable the User to change the timespan of the timeline
    var radioButtons = '<span id="timelineRadioButtons" style="margin-left:7px;">Timespan <input type="radio" onclick="addTasksToTimeline(7, 1060);" name="time span" value="Week" checked>Week &nbsp;&nbsp;&nbsp;' +
        '<input type="radio" onclick="addTasksToTimeline(15, 1060, null);" name="time span" value="2 Weeks">2 Weeks &nbsp;&nbsp;&nbsp;' +
        '<input type="radio" onclick="addTasksToTimeline(31, 1060, null);" name="time span" value="Month">Month &nbsp;&nbsp;&nbsp;' +
        '<input type="radio" onclick="addTasksToTimeline(90, 1060, null);" name="time span" value="3 Month">3 Months &nbsp;&nbsp;&nbsp;' +
        '<input type="radio" onclick="addTasksToTimeline(210, 1060, null);" name="time span" value="7 Month">7 Months &nbsp;&nbsp;&nbsp;' +
        '<input type="radio" onclick="addTasksToTimeline(365, 1060, null);" name="time span" value="Year">Year &nbsp;&nbsp;&nbsp;' +
        '</span>';
    var colorRadioButtons = '<span id="colorRadioButtons">Color ' +
        '<input type="radio" onclick="colorTasksInTimeline(\"colorByProject\");" name="color choice" value="Color the tasks by Project" checked>by Project &nbsp;&nbsp;&nbsp;' +
        '<input type="radio" onclick="colorTasksInTimeline(\"colorByUser\");" name="color choice" value="Color the tasks by User">by User &nbsp;&nbsp;&nbsp;' +
        '</span>';

    // Adding the resources found in the list as rows in the Timeline
    var resourcesSection = document.getElementById("timelineRadioButtons");
    if (resourcesSection == undefined || resourcesSection == null) { // If there is our custom Resources list and the rows have not yet been added...
        timelineArea.innerHTML = radioButtons + equipmentsHTMLstring + '<div id="timeline"></div>' + timelineArea.innerHTML;

        var timeline = document.getElementById("timeline");

        if (numRows > 1) {
            timeline.style.height = ((22*numRows)-2) + 'px'; // Enlarging the height of the timeline in order to have one line for each Resource.
            // The last line will not need a white space below it. That's why we take out 2px from the result.
        } else {
            timeline.style.height = '20px'; // Enlarging the height of the timeline in order to have one line for each Resource.
        }
        var timelineWidth = 1060;
        timeline.style.width = timelineWidth + 'px'; // Manually setting the width of the timeline to override the behaviour of SharePoint, which would expand the
        timeline according to the width of the page.
        timeline.style.display = "inline";
    }

    /***** Adding our code to the timeline. *****/
    var numOfDayInTimeline=7;
    addTasksToTimeline(numOfDayInTimeline, timelineWidth);
}

// This function will color the tasks in the timeline according to the equipment or the personnel.
function colorTasksInTimeline(colorRule) {
    var timelineWidth = document.getElementById("timeline").style.width;
    var numOfDayInTimeline = 7; // Initializing the variable for the consistency check
    var radioButtons = document.getElementById("timelineRadioButtons").getElementsByName("input");

    var numOfDayArray = [7, 15, 31, 90, 210, 365]; // Defining the array containing the number of days considered for each possible time span

    for (var i=0; i<radioButtons.length; i++) {
        if (radioButtons[i].checked) {
            numOfDayInTimeline = numOfDayArray[i];
            break;
        }
    }

    if (colorRule == "colorByProject") {
        addTasksToTimeline(numOfDayInTimeline, timelineWidth, "colorByProject");
    } else {
        addTasksToTimeline(numOfDayInTimeline, timelineWidth, "colorByUser");
    }
}

// This function will add the tasks read from the Server to the timeline "manually" (instead of using the SharePoint's disposition.
function addTasksToTimeline(numOfDayInTimeline, timelineWidth, colorRule) {
    // Checking the input
    if (typeof(numOfDayInTimeline) == 'string') { numOfDayInTimeline = parseInt(numOfDayInTimeline); }
    if (typeof(timelineWidth) == 'string') { timelineWidth = parseInt(timelineWidth); }

    var siteUrl = document.URL;
    // Correcting the URL (if necessary)
    if (siteUrl == null || siteUrl == undefined) { siteUrl = window.location.href; }
    var numSlashes = 0;
    for (var i=0; i<siteUrl.length; i++) {
        if (siteUrl[i] == '/') {

```

```

        numSlashes++;
        if(numSlashes == 4){
            siteUrl = siteUrl.substring(0, 1);
            break;
        }
    }
}

var timeline = document.getElementById('timeline');

/***** If the User clicks on the timeline radio buttons we have to read which radio button *****/
if(colorRule == null){
    // Reading the color rule to apply to the tasks
    var radioButtons = document.getElementById('colorRadioButtons').getElementsByName('input'); // The radio buttons for the color rules
    if(radioButtons[0].checked){ // If the first radio button is checked...
        colorRule = 'colorByProject';
    }else{
        colorRule = 'colorByUser';
    }
}

/***** This piece of code will be used to refresh the timeline when selecting different time spans (e.g.: 1 week, 2 week, 1 month etc...). *****/
// only if the timeline is set as "shown" from SharePoint. If "hidden" we can work without it.
var indexOFtasks = timeline.innerHTML.indexOf('<div class="ms-tl-today"');
if( indexOFtasks > -1){ // Using "timeline" as first variable in the next line does not work. We have to re-catch the HTML section.
    timeline.innerHTML = timeline.innerHTML.substring( 0, indexOFtasks ); // We are deleting the tasks that were in the
}else{ // If the "Today"'s flag is not present...
    indexOFtasks = timeline.innerHTML.indexOf('<div class="timeline-dates"');
    if(indexOFtasks > -1){
        timeline.innerHTML = timeline.innerHTML.substring( 0, indexOFtasks ); // We are deleting the tasks that were in the
    }
}

// Defining the dates to write on the X axis of the timeline
var datesStrings = new Array();
var today = new Date();
var timeSpan; // It will signal to the createDateString() function the kind of string we want in output.

if(numOfDaysInTimeline == 7){ timeSpan = 'week'; }
else{
    if(numOfDaysInTimeline == 15){ timeSpan = 'twoWeeks'; }
    else{
        if(numOfDaysInTimeline == 31){ timeSpan = 'month'; }
        else{
            if(numOfDaysInTimeline == 90){
                // Re-calculating the number of days in the three months according to the months considered.
                numOfDaysInTimeline = daysInMonth(today.getMonth(),today.getFullYear()); // Adding the days in this month
                today.setDate(1);
                today.setMonth( today.getMonth() -1 );
                numOfDaysInTimeline += daysInMonth(today.getMonth(),today.getFullYear()); // Adding the days in the previous
                month

                today.setMonth( today.getMonth() +1);
                today.setMonth( today.getMonth() +1);
                numOfDaysInTimeline += daysInMonth(today.getMonth(),today.getFullYear()); // Adding the days in the following
                month

                today = new Date(); // Resetting 'today'

                timeSpan = 'threeMonths'; }
            else{
                if(numOfDaysInTimeline == 210){ timeSpan = 'sevenMonths'; }
                else{
                    if( numOfDaysInTimeline == 365){ timeSpan = 'year'; }
                    else{ timeSpan = 'year'; }
                }
            }
        }
    }
}

// Adding the flag that signals today's date on the timeline
/*
if( timeSpan == 'week'){
    // "Today" label's code
    // I do not know why but SharePoint displays this label differently when adding it to the webpage the first time or some other time through Javascript and the
    radio buttons.

    if(addTasksToTimelineFlag == false){ // If this is the first time that the function has been called...
        var todayLabel = '<div class="ms-tl-today" style="position:absolute; background-color:rgb(0, 114, 198); color:white; z-index=14; border-color:
        rgb(0, 114, 198); height: 22px; top: 42px; left: -605px;"></div>' +
        '<div class="ms-tl-todayLabel" style="position:absolute; background-color:rgb(0,
        114, 198); color:white; text-align:center; z-index=14; width: 49px; height: 20px; top: 23px; left: -628px; background-color: rgb(0, 114, 198);">Today</div>';
        timeline.innerHTML += todayLabel;
    }else{
        var todayLabel = '<div class="ms-tl-today" style="position:absolute; background-color:rgb(0, 114, 198); color:white; z-index=14; border-color:
        rgb(0, 114, 198); height: 22px; top: -23px; left: 227px;"></div>' +
        '<div class="ms-tl-todayLabel" style="position:absolute; background-color:rgb(0,
        114, 198); color:white; text-align:center; z-index=14; width: 49px; height: 20px; top: -42px; left: 204px; background-color: rgb(0, 114, 198);">Today</div>';
        timeline.innerHTML += todayLabel;
    }
}

}else{
    if(timeSpan == 'twoWeeks'){
        // "Today" label's code
        var todayLabel = '<div class="ms-tl-today" style="position:absolute; background-color:rgb(0, 114, 198); color:white; z-index=14; border-color:
        rgb(0, 114, 198); height: 24px; top: -24px; left: 106px;"></div>' +
        '<div class="ms-tl-todayLabel" style="position:absolute; background-color:rgb(0,
        114, 198); color:white; text-align:center; z-index=14; width: 49px; height: 20px; top: -42px; left: 81px; background-color: rgb(0, 114, 198);">Today</div>';
        timeline.innerHTML += todayLabel;
    }else{
        if(timeSpan == 'month'){
            // "Today" label's code
            var todayLabel = '<div class="ms-tl-today" style="position:absolute; background-color:rgb(0, 114, 198); color:white; z-index=14; border-color:
            border-color: rgb(0, 114, 198); height: 24px; top: -24px; left: 119px;"></div>' +

```

```

background-color:rgb(0, 114, 198); color:white; text-align:center; z-index=14; width: 49px; height: 20px; top: -42px; left: 95px; background-color: rgb(0, 114, 198);">Today</div>;
        timeline.innerHTML += todayLabel;
    }
}
*/

// Creating local variables
var horizontalSectionWidth = timelineWidth/numOfDaysInTimeline; // We need to keep it as float, because when it comes to the "year" time span we
have to keep the decimal values to keep the result for the left margin of every section correct.
var remainingHorizontalSpace = timelineWidth - (horizontalSectionWidth*numOfDaysInTimeline);
var minDate = new Date();
var maxDate = new Date();
if( timeSpan == 'week' ){
    // Saving the first date on the left and the last date on the right of the timeline.
    minDate.setDate( today.getDate() -1); // The 10% of the timeSpan regards the past
    maxDate.setDate( today.getDate() + numOfDaysInTimeline-2); // The 90% of the timeSpan regards the future (90% less the present
day)

    today.setDate( today.getDate() - 2 ); // Bringing the today's date back to the first day of the timeline less one.
    for( var i=0; i<numOfDaysInTimeline; i++){
        today.setDate( today.getDate() + 1 ); // Updating the date object that we want to pass to the function
createDateString()
        datesStrings.push( createDateString(today, timeSpan) ); //
Creating the dates strings passing to the function
    }
}
else{
    if( timeSpan == 'twoWeeks' || timeSpan == 'month'){
        // Saving the first date on the left and the last date on the right of the timeline.
        minDate.setDate( today.getDate() -(Math.floor(numOfDaysInTimeline/10)) ); // The 10% of the timeSpan regards the past
        maxDate.setDate( today.getDate() + (Math.floor(numOfDaysInTimeline/10*9)) ); // The 90% of the timeSpan regards the future (90%
less the present day)

        today.setDate( today.getDate() - (Math.floor(numOfDaysInTimeline/10)+1) ); // Bringing the today's date back to the first day of the timeline
less one.

        for( var i=0; i<numOfDaysInTimeline; i++){
            today.setDate( today.getDate() + 1 ); // Updating the date object that we want to pass to the function
createDateString()
            datesStrings.push( createDateString(today, timeSpan) ); //
Creating the dates strings passing to the function
        }
    }
    else{
        if( timeSpan == 'threeMonths' ){
            // Saving the first date on the left and the last date on the right of the timeline.
            // minDate = the present month
            minDate.setDate(1); // The first day of the present month
            maxDate = new Date();
            months and then move 1 day back.
            maxDate.setMonth( maxDate.getMonth() +3 ); // To have the maxDate set on the last day of the next month we can forward by 2
months and then move 1 day back.

            maxDate.setDate(1); // Setting the date on the first day of the month
            maxDate.setDate( maxDate.getDate() -1 ); // Going to the last day of the previous month for maxDate (two months ahead for
us).

            datesStrings.push( createDateString(minDate, timeSpan) );
            minDate.setMonth(minDate.getMonth()+1);
            datesStrings.push( createDateString(minDate, timeSpan) );
            minDate.setMonth(minDate.getMonth()-1); // Bringing the minDate back to its original date (for the future
operations that use this variable)
            datesStrings.push( createDateString(maxDate, timeSpan) );
        }
        else{
            if( timeSpan == 'sevenMonths' ){
                // Saving the first date on the left and the last date on the right of the timeline.
                if(minDate.getDate() == 31){ minDate.setDate(30); } // We need this check because the next
function rewinds at most for 30 days (Javascript bug)

                minDate.setMonth( minDate.getMonth() -1 );
                minDate.setDate(1); // The result will be the first day of the previous month.
                maxDate = new Date();
                forward by 2 months and then move 1 day back.
                maxDate.setMonth( maxDate.getMonth() +6 ); // To have the maxDate set on the last day of the next month we can
forward by 2 months and then move 1 day back.

                maxDate.setDate(1); // Setting the date on the first day of the month
                maxDate.setDate( maxDate.getDate() -1 ); // Going to the last day of the previous month for maxDate (next month
for us).

                if(maxDate.getMonth() === 'January' ){ maxDate.setFullYear( minDate.getFullYear() +1) }

                datesStrings.push( createDateString(minDate, timeSpan) );
                minDate.setMonth(minDate.getMonth()+1);
                datesStrings.push( createDateString(minDate, timeSpan) );
                minDate.setMonth(minDate.getMonth()+1);
                datesStrings.push( createDateString(minDate, timeSpan) );
                minDate.setMonth(minDate.getMonth()+1);
                datesStrings.push( createDateString(minDate, timeSpan) ); // This month
                minDate.setMonth(minDate.getMonth()+1);
                datesStrings.push( createDateString(minDate, timeSpan) );
                minDate.setMonth(minDate.getMonth()+1);
                datesStrings.push( createDateString(minDate, timeSpan) );
                minDate.setMonth(minDate.getMonth()-5); // Bringing the minDate back to its original date (for
the future operations that use this variable)
            }
            else{
                if( timeSpan == 'year' ){
                    // Saving the first date on the left and the last date on the right of the timeline.
                    minDate = new Date(new Date().getFullYear(), 0, 1); // First of January
                    maxDate = new Date(new Date().getFullYear(), 11, 31); // End of this Year

                    datesStrings.push("January");
                    datesStrings.push("February");
                    datesStrings.push("March");
                    datesStrings.push("April");
                    datesStrings.push("May");
                    datesStrings.push("June");
                    datesStrings.push("July");
                    datesStrings.push("August");
                }
            }
        }
    }
}

```

```

        datesStrings.push("September");
        datesStrings.push("October");
        datesStrings.push("November");
        datesStrings.push("December");
    }
}

}

}

minDate.setHours(0);
minDate.setMinutes(0);
minDate.setSeconds(0);
minDate.setMilliseconds(0);
maxDate.setHours(0);
maxDate.setMinutes(0);
maxDate.setSeconds(0);
maxDate.setMilliseconds(0);

// Creating the HTML code for the X axis of the timeline (the dates)
var datesAxisString = ''; // The dates for the X axis in the timeline
var separatorsString = ''; // The string with the small vertical separators between the dates on the timeline
var timelineNumColumns;
if(timeSpan == 'week'){ timelineNumColumns = 7; }
else if(timeSpan == 'twoWeeks'){ timelineNumColumns = 15; }
else if(timeSpan == 'month'){ timelineNumColumns = 31; }
    else if(timeSpan == 'threeMonths'){ timelineNumColumns = 3; } // 3 months
        else if(timeSpan == 'sevenMonths'){ timelineNumColumns = 7; } // 7 months
            else if(timeSpan == 'year'){ timelineNumColumns = 12; }}}}} // 12 months

var sectionWidthInTimeline = Math.floor(timelineWidth/timelineNumColumns)-5; // Adjusting the width counting the padding-left property in the space sections

for(var i=0; i<timelineNumColumns; i++){
    if(i==timelineNumColumns-1){
        datesAxisString += '<span style="white-space: nowrap; overflow: hidden; height: 100%; font-family: \'Segoe UI\'; font-size: 8pt; float: left;
text-align: left; padding-left: 5px; padding-bottom: 5px; width: '+ (sectionWidthInTimeline+remainingHorizontalSpace) +'px;">'+ datesStrings[i] +'</span>';
        separatorsString += '<span style="white-space: nowrap; overflow: hidden; height: 100%; float: left; border-left-width: 1px; border-left-style: solid;
border-color: rgb(213, 213, 213); width: '+ (sectionWidthInTimeline+remainingHorizontalSpace+4) +'px;"></span>';
    }else{
        datesAxisString += '<span style="white-space: nowrap; overflow: hidden; height: 100%; font-family: \'Segoe UI\'; font-size: 8pt; float: left;
text-align: left; padding-left: 5px; padding-bottom: 5px; width: '+ sectionWidthInTimeline+'px;">'+ datesStrings[i] +'</span>';
        separatorsString += '<span style="white-space: nowrap; overflow: hidden; height: 100%; float: left; border-left-width: 1px; border-left-style:
solid; border-color: rgb(213, 213, 213); width: '+ (sectionWidthInTimeline+4) +'px;"></span>';
    }
}

// The timeline is 1060px width, so we have to split this value in the number of subsections we want to create (es. number of days, hours of the day ecc...)
// We are adding 220px to the left margin to make space for the categories and equipments.
var timelineXaxis = '<div class="timeline-dates" style="width: '+ (timelineWidth+1) +'px; height:20px; margin-left:220px; left:0px; top:-20px; padding-right:28px;">'+
'<div style="padding: 0px; left: 0px; top: 0px; overflow: hidden; position:inherit; height:
19px; background-repeat: repeat-x;"></div>'+
'<div style="white-space:nowrap; overflow:hidden; position: relative; color: rgb(119, 119, 119);
border-bottom-width: 1px; border-bottom-style: hidden; border-bottom-color: rgb(119, 119, 119); height: 20px; top: 0px; left: 0px; margin-left: 2px;">'+
    datesAxisString +
'</div>'+
'<div style="white-space:nowrap; overflow:hidden; position:relative; color: rgb(119, 119, 119);
border-bottom-width: 1px; border-bottom-style: hidden; border-bottom-color: rgb(119, 119, 119); height: 10px; top: -16px; left: 0px; margin-left: 0px;">'+
    separatorsString +
'</div>'+
'</div>'+
'<div id="innerTimeline" style="margin-left:220px; border:1px solid black; margin-top:13px;
position:relative;"></div>';

// Adding the dates on the X axis to the timeline
timeline.innerHTML += timelineXaxis;
// Reading height and width of the timeline
var timelineHeight = timeline.style.height;
var timelineWidth = timeline.style.width;

timeline = document.getElementById("innerTimeline");
// Adjusting the height of the inner timeline
timeline.style.height = timelineHeight; // Enlarging the height of the timeline in order to have one line for each Equipment
timeline.style.width = timelineWidth;

var listItemEnumerator = taskListItems.getEnumerator(); // Resetting the enumerator
var authorsArray = new Array(); // The array containing the names of the creators of the tasks
var text = ''; // The HTML code of the task to add to the timeline

while( listItemEnumerator.moveNext() ){
    var oListItem = listItemEnumerator.get_current();

    var author = oListItem.get_item('Author');
    var project = oListItem.get_item('Project');
    var taskTitle = oListItem.get_item('Title');
    var assignedTo = oListItem.get_item('AssignedTo');
    var startDate = oListItem.get_item('StartDate');
    var dueDate = oListItem.get_item('DueDate');
    var equipmentCategory = oListItem.get_item('Equipment_x0020_Category');
    var equipmentName = oListItem.get_item('Equipment_x0020_Name');
    var amountOfMagnets = oListItem.get_item('Amount_x0020_of_x0020_magnets'); // The string "x0020" is a space in the SharePoint's list's property.
    var taskId = oListItem.get_item('ID');
    var contentTypeId = oListItem.get_item('ContentTypeId').$c_1;
    var listId = taskListItems.get_path().get_$1O_0().$f_1; // Here we get a string with a lot of information. E.g.: "740c6a0b-85e2-48a0-a494-
e0f1759d4aa7:site:5224dfee-cb44-4a8b-ada7-ed36f701eb5f:web:35c8c320-4179-49d9-9bd6-325be8036e6b:list:08e544cf-2b93-4378-b3bd-16ca91cae1e9"
    var idIndex = listId.indexOf("list:"); // Reading where the ID of the list
starts in the string.

    var equipmentsArray = new Array(); // This array will contain, if necessary, the array of equipments. If not used its length will be
= 0.

    // Taking only the "listId" and Formatting it for the EditItem2() method that will be created later.
    listId = listId.substring(idIndex + 5, listId.length); // Keeping only the ID of the list.
    listId = listId.toUpperCase().replace(/-/g, '%2D');

```



```

    }else{
        if(startDate >= minDate && dueDate <= maxDate){
            taskDaysGap = dueDate.getTime() - startDate.getTime()+1; // The +1 is necessary, otherwise sometimes a day is
            // lost, ending up counting e.g. 2 days gap instead of 3.
            taskDaysGap = Math.ceil(taskDaysGap / (1000 * 3600 * 24)); // Getting the difference in days
            taskWidth = horizontalSectionWidth * taskDaysGap; // +horizontalSectionWidth is
            // necessary to add one plus section's width to make the items display correctly.
            if(timeSpan == 'week'){ taskWidth -- 1; }
            else( if(timeSpan == 'twoWeeks'){ taskWidth --3; }
            else( if(timeSpan == 'threeMonths'){ taskWidth --3; }
            else( if(timeSpan == 'sevenMonths'){ taskWidth -- 4; }
            )
            )
            if(dueDate.getDay() === maxDate.getDay() && dueDate.getMonth() === maxDate.getMonth() && dueDate.getFullYear()
            === maxDate.getFullYear()){ if(timeSpan == 'week'){ taskWidth += 2; } } // If the item fits in the timeline but finishes
            // At least one of the dates is inside of the timeline. Checking which one.
            if(startDate >= minDate){
                taskDaysGap = maxDate.getTime() - startDate.getTime()+1; // Using the max date present on the
                // timeline as "dueDate"
                taskDaysGap = Math.ceil(taskDaysGap / (1000 * 3600 * 24)); // Getting the
                // difference in days
                taskWidth = horizontalSectionWidth * taskDaysGap;
                if(timeSpan == 'week'){ taskWidth += 1; }
                else( if(timeSpan == 'twoWeeks'){ taskWidth --4; }
                //if(timeSpan == 'month' || timeSpan == 'twoWeeks') -> no need to add value in these
                // cases
                )
            }else{
                if(dueDate <= maxDate){
                    taskDaysGap = dueDate.getTime() - minDate.getTime(); // Using the min date
                    // present on the timeline as "startDate"
                    taskDaysGap = Math.ceil(taskDaysGap / (1000 * 3600 * 24)); //
                    // Getting the difference in days
                    taskWidth = (horizontalSectionWidth * taskDaysGap) + horizontalSectionWidth -1;
                    // display correctly.
                    // +horizontalSectionWidth is necessary to add one plus section's width to make the items
                    // display correctly.
                    if(timeSpan == 'twoWeeks'){ taskWidth --1; }
                    else( if(timeSpan == 'threeMonths'){ taskWidth -- 4; }
                    )
                }
            }
        }
        // In any case we want the width to lose a pixel, because in the "New Task" form we increase by one the 'leftPadding' of each of them.
        // Except when the task takes the whole timeline.
        if( taskWidth != parseInt( timelineWidth.substring(0, timelineWidth.length-2) )){
            taskWidth--;
        }
        if(taskWidth < 2){ taskWidth = 2; } // Setting a minimum width

        // Calculating the space to be added on the left of the task in the timeline
        var leftMargin = 0; // It is the minimum distance from the left border in order for the Task to appear in the
        // innerTimeline (because it has 'absolute' positioning.
        if( startDate > minDate ){
            var daysDifference = startDate.getTime() - minDate.getTime();
            daysDifference = Math.ceil(daysDifference / (1000 * 3600 * 24));
            leftMargin += Math.floor(horizontalSectionWidth * daysDifference)+1;
            if(timeSpan == 'month'){
                leftMargin -- 2;
                taskWidth += 1;
            }else{
                if(timeSpan == 'twoWeeks'){
                    leftMargin -- 2;
                }else{
                    if(timeSpan == 'threeMonths'){
                        leftMargin -- 6;
                        taskWidth += 1;
                    }else{
                        if(timeSpan == 'sevenMonths'){
                            leftMargin -- 7;
                        }
                    }
                }
            }
        }
        if(leftMargin < 0) { leftMargin = 0; } // Bounding the value of 'leftMargin'
    }

    // We want the dates to be in the format "22/10". We convert now the Date objects in strings to display in the HTML element.
    startDate = startDate.getDate() +'/' + (parseInt(startDate.getMonth()+1));
    dueDate = dueDate.getDate() +'/' + (parseInt(dueDate.getMonth()+1));
    taskWidth = Math.floor(taskWidth); // Re-adjusting the width, in case there are decimal numbers in it.

    var backgroundColor = 'white'; // Setting the default background color
    if(colorRule == 'colorByProject'){ // If we want every project to have a different background color...
        // We have to find the related project in the 'globalProjectsArray' and use the linked color.
        var tempLength = globalProjectsArray.length;
        for(var x=0; x<tempLength; x++){
            if(globalProjectsArray[x].projectName == project){
                backgroundColor = globalProjectsArray[x].projectColor;
                break;
            }
        }
    }else{
        // Each User will have a different color associated with it and so its Tasks.
        // We have to find the related user in the 'globalUsersArray' and use the linked color.
        var tempLength = globalUsersArray.length;
        var tempAssignedTo = assignedTo.replace(/&nbsp;/g, '').substring(1);
        for(var x=0; x<tempLength; x++){
            if(globalUsersArray[x].userName.$2e_1 == tempAssignedTo){
                backgroundColor = globalUsersArray[x].userColor;
                break;
            }
        }
    }

    text = '<div class="ms-tl-bar" tabIndex="0" style="position:absolute; cursor:pointer; margin-bottom:2px; width: '+ taskWidth +'px;
    height: 20px; top: '+ distanceFromTopInTheTimeline +'px; left: '+ leftMargin +'px; background-color: '+ backgroundColor +'; white-space:nowrap; overflow:hidden;"> +

```



```
if(timeSpan == 'week'){
    switch(month){
        case 0: month="January";           break;
        case 1: month="February";          break;
        case 2: month="March";              break;
        case 3: month="April";              break;
        case 4: month="May";                break;
        case 5: month="June";                break;
        case 6: month="July";                break;
        case 7: month="August";              break;
        case 8: month="September";          break;
        case 9: month="October";            break;
        case 10: month="November";          break;
        case 11: month="December";          break;
        default: month="January";           break;
    }
}
}
else{
    switch(month){
        case 0: month="Jan";                break;
        case 1: month="Feb";                break;
        case 2: month="Mar";                break;
        case 3: month="Apr";                break;
        case 4: month="May";                break;
        case 5: month="Jun";                break;
        case 6: month="Jul";                break;
        case 7: month="Aug";                break;
        case 8: month="Sep";                break;
        case 9: month="Oct";                break;
        case 10: month="Nov";                break;
        case 11: month="Dec";                break;
        default: month="Jan";                break;
    }
}
}
}
else{
    if(timeSpan == 'month'){
        switch(day){
            case 0: day="Sun";               break;
            case 1: day="Mon";               break;
            case 2: day="Tue";               break;
            case 3: day="Wed";               break;
            case 4: day="Thu";               break;
            case 5: day="Fri";               break;
            case 6: day="Sat";               break;
            default: day = "Mon";             break;
        }
    }
    switch(month){
        case 0: month="Jan";                 break;
        case 1: month="Feb";                 break;
        case 2: month="Mar";                 break;
        case 3: month="Apr";                 break;
        case 4: month="May";                 break;
        case 5: month="Jun";                 break;
        case 6: month="Jul";                 break;
        case 7: month="Aug";                 break;
        case 8: month="Sep";                 break;
        case 9: month="Oct";                 break;
        case 10: month="Nov";                 break;
    }
}
```

```

        case 11: month="Dec";           break;
    default: month="Jan";              break;
    }
}
else{
    if( timeSpan == 'threeMonths' || timeSpan == 'sevenMonths'){
        switch(month){
            case 0: month="January";    break;
            case 1: month="February";   break;
            case 2: month="March";       break;
            case 3: month="April";       break;
            case 4: month="May";         break;
            case 5: month="June";        break;
            case 6: month="July";        break;
            case 7: month="August";      break;
            case 8: month="September";   break;
            case 9: month="October";     break;
            case 10: month="November";   break;
            case 11: month="December";   break;
            default: month="January";    break;
        }
    }
}

var numberOfTheDay = dateObj.getDate(); // Returns the day of the month (from 1-31)
if(numberOfTheDay < 10){
    numberOfTheDay = '0' + parseInt(numberOfTheDay, 10); // This way if the month is the 5th it will be displayed as "05", instead of "5"
}

if(timeSpan == 'week' || timeSpan == 'twoWeeks'){
    return day+ ' ' +numberOfTheDay+ ' ' +month;
}
else{
    if(timeSpan == 'month'){
        return numberOfTheDay+ ' ' +month;
    }
    else{
        return month;
    }
}
}
}

```

Appendix C

Hereunder is displayed the code regarding the Social API.

File “socialAPI.js”:

```
/*
 * Version 5.2
 * - Now when some feeds are already displayed and new feeds are retrieved from the Server the function only adds the new ones, without refreshing the whole HTML section.
 * - Improved the displaying of the feeds with same hashtag.
 */

// Creating the global variable socialAPI.
// It is made global to let other libraries and code inside the HTML page use it.
var socialAPI;

// The API:
(function ($) {
    /******
     * DEFINITION OF LOCAL VARIABLES
     * *****/
    // DEVELOPMENT LINKS
    /*
    var socialWebsite = "https://social-dev.cern.ch/";

    var requestExecutorSite = "https://social-dev.cern.ch/_layouts/15/AppWebProxy.aspx"; // Site used to authenticate to the social network
    var searchRestService = "https://social-dev.cern.ch/_api/search/";
    var formDigestUrl = "https://social-dev.cern.ch/_api/contextinfo";
    var myFeedManagerEndpoint = "https://social-dev.cern.ch/_api/social.feed/"; // From this site we can derive every other for post, delete and get
    feeds

    var apiEndpoint = "https://social-dev.cern.ch/_api/";
    var querySiteToGetAllTheTags = searchRestService + "query?querytext='ContentTypeId:0x01FD' -ContentClass=urn:content-class:SPSPeople'refiners='Tags'";
    // This variable is used only if the Canvas with tags is displayed
    */

    // PRODUCTION LINKS
    var socialWebsite = "https://social.cern.ch/";

    var requestExecutorSite = "https://social.cern.ch/_layouts/15/AppWebProxy.aspx"; // Site used to authenticate to the social network
    var searchRestService = "https://social.cern.ch/_api/search/";
    var formDigestUrl = "https://social.cern.ch/_api/contextinfo";
    var myFeedManagerEndpoint = "https://social.cern.ch/_api/social.feed/"; // From this site we can derive every other for post, delete and get feeds
    var apiEndpoint = "https://social.cern.ch/_api/";
    var querySiteToGetAllTheTags = searchRestService + "query?querytext='ContentTypeId:0x01FD' -ContentClass=urn:content-class:SPSPeople'refiners='Tags'";
    // This variable is used only if the Canvas with tags is displayed

    var hashtagCheckTimer; // This timer will check if any feed with the same hashtag has been retrieved from
    the Server. If not, it will display a message to the User.

    var globalArrayOfProfiles = new Array(); // Array of objects like (accountName, whereToWrite, tempHandler, updateInterval, numOffeeds)
    var globalArrayOfHashtags = new Array(); // Array of objects like (noSharpTagString, whereToWrite, handlerCode, updateInterval, numOffeeds)
    var globalArrayOfSingleConversations = new Array(); // Array of objects like (URL, whereToWrite)

    var followedFeedsWhereToWrite = ''; // global variable that stores the section containing the followed feeds
    var followedFeedsUpdateInterval; // global variable that stores the update interval for the followed feeds
    var followedFeedsNumFeeds = 0; // global variable that stores the maximum number of feeds to display
    var followedFeedsFlagDisplayReplies = false; // global variable that stores the boolean var that says if to display the replies or not.
    var followedFeedsUpdatesHandler = 'a'; // global variable that keeps the number of the automatic feeds updates handler

    var errorHandlerFunction = function() { alert("Error while making the CORS request."); }; // This variable will have the pointer to the function that will
    eventually handle the exceptions while making the CORS requests.

    jQuery.support.cors = true; // Used for createCORSRequest()

    /******
     * DEFINITION OF FUNCTIONS
     * *****/
    // This function authenticates the User on Social (transparently to the User)
    function authenticateOnSocial(inputFunction) {
        var executor = new SP.RequestExecutor(requestExecutorSite);

        executor.executeAsync(
            {
                url: formDigestUrl,
                method: "GET",
                headers: {
                    "Accept": "application/json; odata=verbose",
                    "Access-Control-Allow-Origin": "*",
                },
                dataType: "json",
                error: function (xhr, ajaxOptions, thrownError) {
                    // This function will be executed always. It is not an actual 'error' situation.

                    try {
                        // After the authentication completes we use the function passed in input, that will contain the
                        calls for any other function on Social

                        if (inputFunction !== null && inputFunction !== undefined) {
                            inputFunction();
                        }
                    } catch (e) { console.log("Error: input function parameter in the authentication function is not valid."); return; }
                }
            }
        );
    }
});
```

```

    }
    };
}

// This function executes a REST call and passes all the data retrieved to the 'onSucc' function specified by the calling function.
function executeRestCall(url, method, data, onSuccess, onError) {
    var xhr = createCORSRequest(method, url);

    if (!xhr) {
        throw new Error('CORS not supported');
    }
    else{
        xhr.onload = function () {
            onSuccess(xhr.responseText);
        };
        xhr.onerror = onError;
        if (data !== null && data !== undefined && data !== '') {
            xhr.send(data);
        }else{
            xhr.send();
        }
    }
}

// This function executes a REST call and passes all the data retrieved to the 'onSucc' function specified by the calling function.
function executeRestCallExtended(url, method, data, onSuccess, onError, writeToWrite, id) {
    var xhr = createCORSRequest(method, url);

    if (!xhr) {
        throw new Error('CORS not supported');
    }
    else{
        xhr.onload = function () {
            onSuccess(xhr.responseText, writeToWrite, id); // passing the parameters and the results of the RESTcall to the 'onSucc' pointed
function
        };
        xhr.onerror = onError;
        if (data !== null && data !== undefined && data !== '') {
            xhr.send(data);
        }else{
            xhr.send();
        }
    }
}

function executeRestCallExtendedFour(url, method, data, onSuccess, onError, writeToWrite, id, numFeeds) {
    var xhr = createCORSRequest(method, url);

    if (!xhr) {
        throw new Error('CORS not supported');
    }
    else{
        xhr.onload = function () {
            onSuccess(xhr.responseText, writeToWrite, id, numFeeds); // passing the parameters and the results of the RESTcall to the
'onSucc' pointed function
        };
        xhr.onerror = onError;
        if (data !== null && data !== undefined && data !== '') {
            xhr.send(data);
        }else{
            xhr.send();
        }
    }
}

function executeRestCallExtendedFive(url, method, data, onSuccess, onError, writeToWrite, id, numFeeds, tagText) {
    var xhr = createCORSRequest(method, url);

    if (!xhr) {
        throw new Error('CORS not supported');
    }
    else{
        xhr.onload = function () {
            onSuccess(xhr.responseText, writeToWrite, id, numFeeds, tagText); // passing the parameters and the results of the
RESTcall to the 'onSucc' pointed function
        };
        xhr.onerror = onError;
        if (data !== null && data !== undefined && data !== '') {
            xhr.send(data);
        }else{
            xhr.send();
        }
    }
}

function executeRestCallExtendedSix(url, method, data, onSuccess, onError, writeToWrite, id, numFeeds, numFeedsStillToGet, flag) {
    var xhr = createCORSRequest(method, url);

    if (!xhr) {
        throw new Error('CORS not supported');
    }
    else{
        xhr.onload = function () {
            onSuccess(xhr.responseText, writeToWrite, id, numFeeds, numFeedsStillToGet, flag); // passing the parameters and the
results of the RESTcall to the 'onSucc' pointed function
        };
        xhr.onerror = onError;
        if (data !== null && data !== undefined && data !== '') {

```

```

        xhr.send(data);
    }else{
        xhr.send();
    }
}

function executeRestCallExtendedSeven(url, method, data, onSuccess, onError, whereToWrite, id, textColor, textBorderColor, numDimensions, weightFlag) {
    var xhr = createCORSRequest(method, url);

    if (!xhr) {
        throw new Error('CORS not supported');
    }
    else{
        xhr.onload = function () {
            onSuccess(xhr.responseText, whereToWrite, id, textColor, textBorderColor, numDimensions, weightFlag); // passing the
parameters and the results of the RESTcall to the function pointed by 'onSuccess'.
        };

        xhr.onerror = onError;
        if (data !== null && data !== undefined && data !== ''){
            xhr.send(data);
        }else{
            xhr.send();
        }
    }
}

function createCORSRequest(method, url) {
    var xhr = new XMLHttpRequest();
    if ("withCredentials" in xhr) {
        // Check if the XMLHttpRequest object has a "withCredentials" property.
        // "withCredentials" only exists on XMLHttpRequest2 objects.
        xhr.open(method, url, true);
    } else if (typeof XDomainRequest != "undefined") {
        // Otherwise, check if XDomainRequest.
        // XDomainRequest only exists in IE, and is IE's way of making CORS requests.
        xhr = new XDomainRequest();
        xhr.open(method, url);
    } else {
        // Otherwise, CORS is not supported by the browser.
        xhr = null;
    }

    if(xhr !== null){
        // if the CORS is supported...
        xhr.withCredentials = true;
        xhr.setRequestHeader("accept", "application/json; odata=verbose");
    }

    return xhr;
}

function onError() {
    errorHandlerFunction();
}

// This function retrieves the value associated to an element inside an array with elements like (name, value), if that element is present.
function getValue(key, results) {
    try {
        var postItem = jQuery.grep(results, function (e) {
            if (e.Key === key){
                return e;
            }
        })[0].Value;

        return postItem;
    }
    catch (err) {
        return null;
    }
}

// Function that draws on the screen the news feeds.
function checkDataReceivedAndDisplayTheFeeds(data, whereToWrite, parentWhereToWrite, numOfFeedsTotal, numFeedsStillToGet, flagDisplayReplies) {
    var tempCheck;
    if(whereToWrite[0] === '#'){
        // 'whereToWrite' can be something like "#feedsFollowed". To use the function 'getElementById' we have to skip the '#'. We do that using the
'substring' function.
        tempCheck = document.getElementById(whereToWrite.substring(1)); // Checking the existence of the div section in the html code. If
(tempCheck = null) then no section has been found.
    }else{
        tempCheck = document.getElementById(whereToWrite);
        whereToWrite = '#' + whereToWrite;
    }

    // If the section foreseen for the feeds (section id = whereToWrite) exists, then...
    if(tempCheck !== null)
    {
        try{
            var result = JSON.parse(data); // parsing the data obtained from the social network
        }catch(e){
            $(whereToWrite).append('<div class="feedsItem"> <p id="text"> There has been a problem while communicating with the server.
<br/>Please try again later refreshing the page. </p> </div>');
            console.log("There was a problem while communicating with the Server.\nSee checkDataReceivedAndDisplayTheFeeds() function.");
            return;
        }

        // If the result object exists but has errors it means we made a bad request, than we have to stop the execution and signal it to the developer
in the console log
        if(result.error){
            try{
                if(result.error.message.value.indexOf("Internal error code: 83") > -1){ // The User does not exist on Social

```

```

        $(whereToWrite).append('<div class="feedsItem"> <p id="text"> Your account has not been found on
Social. <br/>Please visit https://social.cern.ch and create the account first. </p> </div>');
    }else{
        // Other error
        $(whereToWrite).append('<div class="feedsItem"> <p id="text"> There has been a problem while
communicating with the server. <br/>Please try again later refreshing the page. </p> </div>');
    }
    console.log("Bad request.\nPlease review the checkDataReceivedAndDisplayTheFeeds() function.");
    return;
}
}catch(e){console.log("Exception thrown in function showUserInfoInFeedsWithSameTagBodyFunction()"); return;}
}

// Consistency check : if no information has been retrieved...
if(result.d === null || result.d === undefined)
{
    // Printing the "problem" message on the screen
    $(whereToWrite).append('<div class="feedsItem"> <p id="text"> Network problem. Please try again later. </p> </div>');
}
else
{
    var feeds;
    // Reading the feeds when there are no feeds can cause an exception, so we use a try/catch section.
    try{
        feeds = result.d.SocialFeed.Threads.results; // capturing the feeds
        feeds.sort(function(a,b){ // This function sorts the elements of the array according to the date of creation
            of the feeds. The most recent one will be the first of the array.

            var dateA = new Date(a.RootPost.CreatedTime);
            var dateB = new Date(b.RootPost.CreatedTime);

            return dateB-dateA;

        });

        // If no feeds are found... (it is an array, so we can check the length)
        if(feeds.length == 0){
            // Printing the "no feed" message on the screen
            var uniqueId = 'networkProblemsWarning'+Date.now(); // It is important to have a unique id,
            otherwise the function 'fadeOut' will work only once.
            $(whereToWrite).append('<div class="feedsItem" id="'+uniqueId+'"> <p id="text"> No (more) feeds
available. </p> </div>'); // Printing the "problem" message on the screen

            $("#"+uniqueId).delay(5000).fadeOut('slow'); // This function will hide the warning after 5 seconds.
            setTimeout(function(){ $("#"+uniqueId).remove(); }, 7000); // This function will remove the
            warning from the HTML code after 7 seconds.
        }
        else{ //else: every feed found is printed
            appendFeeds(feeds, whereToWrite, parentWhereToWrite, numOfFeedsTotal, numFeedsStillToGet,
            flagDisplayReplies); // Displaying the new feeds
        }
    } catch(err) { $(whereToWrite).html("A problem occurred while reading the feeds. \nThe server is probably under maintenance.
\nPlease try again later."); }
}
// else: no feeds are written now.
}

function updateGroupInfo(whereToWrite, department, group, section, imageFlag, departmentFlag, numFeeds){
    // Sanitizing the input (encodeURIComponent() is used instead of encodeURIComponent() when there has to be allowed the possibility to have hashtags.);
    whereToWrite = encodeURIComponent(whereToWrite);
    department = encodeURIComponent(department);
    group = encodeURIComponent(group);
    section = encodeURIComponent(section);

    numOfElementsAlreadyDisplayed = 0; // Resetting the global variable

    if(department === null || department === 'null' || department === undefined || department === '' || department.length < 1){
        $('#content').html('<div class="feedsItem"> <p id="text"> There has been a problem while retrieving the feeds. Please try again later. </p>
</div>');
        return;
    }
    var groupString;
    if(group == null || group == 'null' || group == undefined || group == ''){
        groupString = '';
    }else{
        if(typeof(group) === 'string' && group.length > 1 && group.length < 20){
            groupString = '/' + group;
        }else{
            groupString = '';
        }
    }
    var sectionString;
    if(section == null || section == 'null' || section == undefined || section == ''){
        sectionString = '';
    }else{
        if(typeof(section) === 'string' && section.length > 1 && section.length < 20){
            sectionString = ' Section:' + section;
        }else{
            sectionString = '';
        }
    }
    if(whereToWrite[0] !== '#'){
        whereToWrite = '#' + whereToWrite;
    }
    var tempElement;
    var parentWhereToWrite = whereToWrite;

    $(whereToWrite).html(''); // Clearing the section of the feeds I am following

    var tempSectionID = whereToWrite.substring(1, whereToWrite.length);

```



```

// Section check. If the HTML section is present in the webpage we can move on, otherwise the function has to stop.
if( document.getElementById(tempSectionID) === null ){
    // Error. No HTML section found to display the followed feeds on Social. Please add a <div id="feedsFollowed"> section.
    $(whereToWrite).append('<div class="feedsItem"> <p id="text"> There has been a problem while communicating with the server. <br/>Please try again
later. </p> </div>');
    console.log('Error while trying to write the followed feeds. The section ID passed in input seems not to be present in the webpage.');
```

```

    return;
}

// Adding a new wrapping section in the HTML page to make the SocialAPI's CSS file point only at this part of the webpage, in case many CSS files are used.
var wrapSection =
    '<div class="socialAPIWrapClass">'+
        '<div id="socialAPIDepartment" + tempSectionID + ">'+
        '</div>'+
    '</div>';

$(whereToWrite).append(wrapSection);

whereToWrite = '#socialAPIDepartment' + tempSectionID ;

var searchForGroupInfoSite = searchRestService + "query?querytext='department:' +department + groupString + sectionString+'&sourceid='B09A7990-05EA-4AF9-81EF-EDFAB16C4E31'";
EDFAB16C4E31'';

// In the variable 'searchForGroupInfoSite', the code:
// sourceid='B09A7990-05EA-4AF9-81EF-EDFAB16C4E31'
// tells the Server that we are looking for People (possible search options: Everything, People, Conversations, Videos).

try{
    executeRestCallExtendedSeven(searchForGroupInfoSite, 'GET', null, updateGroupInfoBodyFunction, onError, whereToWrite, department, group, section,
imageFlag, departmentFlag, numFeeds);
}
catch(err){ errorHandlerFunction(11, "There was a problem while communicating with the Server.\nPlease try again later."); }
}

// This function displays the elements found using "updateGroupInfo()".
function updateGroupInfoBodyFunction(data, whereToWrite, department, group, section, imageFlag, departmentFlag, numFeeds){
    var peopleArray = JSON.parse(data);
    try{
        peopleArray = peopleArray.d.query.PrimaryQueryResult.RelevantResults.Table.Rows.results; // Reading people's data.
    }catch(e){
        console.log("There has been a problem while communicating with the Server. Please check updateGroupFeedsBodyFunction() function.");
        $(whereToWrite).append('<div class="feedsItem"> <p id="text"> There has been a problem while communicating with the Server. Please try again
later. </p> </div>');
    }

    if(peopleArray.length < 1){
        var uniqueId = 'networkProblemsWarning'+Date.now(); // It is important to have a unique id, otherwise the function 'fadeOut' will work
only once.
        $(whereToWrite).append('<div class="feedsItem" id="'+uniqueId+'"> <p id="text"> No (more) people found for the department. </p> </div>');
        // Printing the "problem" message on the screen
        $("#"+uniqueId).delay(5000).fadeOut('slow'); // This function will hide the warning after 5 seconds.
        setTimeout(function(){ $("#"+uniqueId).remove(); }, 7000); // This function will remove the warning from the HTML code after 7 seconds.

        return;
    }

    var numOfFeedsToDisplay = 0;
    if( numFeeds < peopleArray.length && numFeeds > 0){
        numOfFeedsToDisplay = numFeeds;
    }else{
        numOfFeedsToDisplay = peopleArray.length;
    }

    var pictureString = '';
    var profilePictureUri = '';
    var departmentString = '';
    var personalSite = '';
    for( var i=0; i<numOfFeedsToDisplay; i++){

        if(departmentFlag){
            departmentString = getValue("Department", peopleArray[i].Cells.results); // Reading the Department from the first person
            if( departmentString.substring(0, department.length) != department ){
                continue;
            }
            departmentString = '<div> <p>Department: '+ departmentString +'</p> </div>';
        }

        if(imageFlag){ // If we want to show the image we update the pictureString
            personalSite = getValue("Path", peopleArray[i].Cells.results);
            profilePictureUri = getValue("PictureURL", peopleArray[i].Cells.results);

            if( profilePictureUri == null || profilePictureUri == undefined || profilePictureUri == 'null' || profilePictureUri == '' ){
                // If there is no URL for the profile picture we display the anonymous profile image.
                pictureString = '<div class="picSection"> ' +
                    '<div id="picSection">' +
                    '<a href="'+
personalSite +'" target="_blank">' +
                    '' +
                    '</a>' +
                    '</div>' +
                    '</div>';
            }else{
                pictureString = '<div class="picSection">' +
                    '<div id="picSection">' +
                    '<a href="'+
personalSite +'" target="_blank">' +
                    '' +
                    '</a>' +
                    '</div>' +
                    '</div>';
            }
        } // else: the 'pictureString' variable will be an empty string.
    }
}

```

```

var name = getValue("PreferredName", peopleArray[1].Cells.results);
var email = getValue("WorkEmail", peopleArray[1].Cells.results);

$(whereToWrite).append('<div class="feedsItem">' +

                                                                    pictureString +
                                                                    '<div class="notPicSection">' +
                                                                    '<h2 id="author">' <a
                                                                    href="" + personalSite + "" target="_blank">' + name + '</a> </h2>' +
                                                                    departmentString +
                                                                    '<p>email: ' <a
                                                                    href="mailto:' + email + "">' + email + '</a></p>' +
                                                                    '</div>' +
                                                                    '</div>');

}

// If there are less than 10 elements displayed -> we call for more
if( document.getElementsByClassName("notPicSection").length < 10 ){
    moreGroupElements(whereToWrite, department, group, section, imageFlag, departmentFlag, numOfFeedsToDisplay);
    return;
}

// Adding a button to retrieve more elements
$(whereToWrite).append('<a id="moreFeedsButton" whereToWrite.substring(1) + "" class="moreFeedsButton" href="javascript:socialAPI().moreGroupElements({#39;'+
whereToWrite +'&#39;, &#39;'+ department +'&#39;, &#39;'+ group +'&#39;, &#39;'+ section +'&#39;, &#39;'+ imageFlag +'&#39;, &#39;'+ departmentFlag +'&#39;, &#39;'+ numOfFeedsToDisplay
+'&#39;})"> Show more elements <br/> <br/> </a>');

}

var numOfElementsAlreadyDisplayed=0; // Global variable
function moreGroupElements(whereToWrite, department, group, section, imageFlag, departmentFlag, numFeeds){
    numOfElementsAlreadyDisplayed += parseInt(numFeeds); // Updating the global variable
    $('#moreFeedsButton'+ whereToWrite.substring(1) ).remove(); // Removing the old "more elements" link

    var groupString;
    if(group == null || group == 'null' || group == undefined || group == ''){
        groupString = '';
    }else{
        if(typeof(group) === 'string' && group.length > 1 && group.length < 20){
            groupString = '/' + group;
        }else{
            groupString = '';
        }
    }

    var sectionString;
    if(section == null || section == 'null' || section == undefined || section == ''){
        sectionString = '';
    }else{
        if(typeof(section) === 'string' && section.length > 1 && section.length < 20){
            sectionString = ' Section:' + section;
        }else{
            sectionString = '';
        }
    }

    var searchForGroupInfoSite = searchRestService + "query?querytext=" + department + department + groupString + sectionString + "&startrow=" +
numOfElementsAlreadyDisplayed + "&sourceid=" + "B09A7990-05EA-4AF9-81EF-EDFAB16C4E31";
/* In the variable 'searchForGroupInfoSite', the code:
* sourceid="B09A7990-05EA-4AF9-81EF-EDFAB16C4E31"
* tells the Server that we are looking for People (possible search options: Everything, People, Conversations, Videos).
*/

    try{
        executeRestCallExtendedSeven(searchForGroupInfoSite, 'GET', null, updateGroupInfoBodyFunction, onError, whereToWrite, department, group, section,
imageFlag, departmentFlag, numFeeds);
    }
    catch(err){ errorHandlerFunction(11, "There was a problem while communicating with the Server.\nPlease try again later."); }

}

function conversationObj(URL, sectionID){
    this.URL = URL;
    this.sectionID = sectionID;
}

/* This function will retrieve the info about one particular thread. The one that can be found at the URL passed in input.
* Input:
* - whereToWrite: the id of the HTML section in which we want to display the feed.
* - url: the URL at which the feed can be found (loading it in a normal browser). */
function updateSingleFeed(whereToWrite, url){
    if(whereToWrite[0] != '#'){
        whereToWrite = '#' + whereToWrite;
    }

    // Updating global variables
    globalArrayOfSingleConversations.push(new conversationObj(url, whereToWrite));

    // Checking the presence of the HTML section in the webpage:
    if(document.getElementById(whereToWrite.substring(1)) == null){
        console.log("No HTML section found. Please check the ID of the HTML section passed in input to the function updateSingleFeed().");
        return;
    }

    var parentWhereToWrite = whereToWrite;

    $(whereToWrite).html(''); // Clearing the HTML section.

    // Adding a new wrapping section in the HTML page to make the SocialAPI's CSS file point only at this part of the webpage, in case many CSS files are used.
    var wrapSection = '<div id="" + parentWhereToWrite.substring(1) + "" class="" + parentWhereToWrite.substring(1) + "">' +
        '<div class="socialAPIWrapClass">'+

```

```

id="socialAPISingleFeed">'+
                                                                    '<div
                                                                    class="socialAPISingleFeed"
                                                                    '</div>'+
                                                                    '</div>'+
                                                                    '</div>';
                                                                    $(whereToWrite).append(wrapSection);
                                                                    whereToWrite = '#socialAPISingleFeed';
                                                                    executeRestCallExtendedFour(formDigestUrl, 'POST', null, updateSingleFeedBodyFunction, onError, parentWhereToWrite, whereToWrite, url); //
calling for the formDigest to make the request
                                                                    }
                                                                    // Function called from updateSingleFeedInfo if the CORS request succeeds. It will use the formDigest coming from the Server to call for the data about a single feed. The
response will be passed to the showUserInformationInFeedsWithSameTagBodyFunction() function, that will display correctly the feed in the webpage.
                                                                    function updateSingleFeedBodyFunction(data, parentWhereToWrite, whereToWrite, url){
                                                                    try{
                                                                    var result = JSON.parse(data); // Parsing the data obtained from the social network
                                                                    var formDigest = result.d.GetContextWebInformation.FormDigestValue;
                                                                    }catch(e){
                                                                    console.log("There was a problem while communicating with the Server.\nSee updateSingleFeedInfoBodyFunction() function.");
                                                                    $(whereToWrite).html("There was a problem while communicating with the Server. Please try again later.");
                                                                    return;
                                                                    }
                                                                    // If the result object exists but has errors it means we made a bad request, than we have to stop the execution and signal it to the developer in the console
log
                                                                    if(result.error){
                                                                    console.log("Error: "+ result.error.message.value +"\nPlease review the updateSingleFeedInfoBodyFunction() function.");
                                                                    try{
                                                                    if(result.error.message.value.indexOf("Internal error code: 83") > -1){ // The User does not exists on Social
                                                                    $(whereToWrite).html('<div class="feedsItem"> <p id="text"> Your account has not been found on Social. <br/>Please
visit https://social.cern.ch and create the account first. </p> </div>');
                                                                    }else{ // Other error
                                                                    $(whereToWrite).html('<div class="feedsItem"> <p id="text"> There was a problem while communicating with the
Server. Please try again later. </p> </div>');
                                                                    }
                                                                    }
                                                                    return;
                                                                    }catch(e){console.log("Exception thrown in function showUserInformationInFeedsWithSameTagBodyFunction()"); return;}
                                                                    }
                                                                    try{
                                                                    var id = url.split("ThreadID=");
                                                                    id = id[id.length-1];
                                                                    }catch(e){
                                                                    console.log("Error: "+ result.error.message.value +"\nPlease review the updateSingleFeedInfoBodyFunction() function.");
                                                                    $(whereToWrite).html("There was a problem while reading the ID of the feed. Please try again later.");
                                                                    return;
                                                                    }
                                                                    }
                                                                    var xhr = createCORSRequest("POST", myFeedManagerEndpoint + "post"); // Creating CORSRequest to Like the feed
                                                                    xhr.onload = function(){ showUserInformationInFeedsWithSameTagBodyFunction(this.responseText, whereToWrite, parentWhereToWrite,
parentWhereToWrite.substring(1));
                                                                    };
                                                                    xhr.onerror = console.log("CORS request encountered an error.\nSee updateSingleFeedInfoBodyFunction() function.");
                                                                    xhr.withCredentials = true;
                                                                    xhr.setRequestHeader("X-RequestDigest", formDigest);
                                                                    xhr.setRequestHeader("content-type", "application/json; charset=utf-8; odata=verbose");
                                                                    var data = "{ 'ID':'"+id+"' }"; // Including the ID of the feed we want to analyze.
                                                                    xhr.send(data); // Sending the information
                                                                    }
                                                                    }
                                                                    // Function that retrieves the feeds from the section Everyone on Social and print them in the webpage.
function updateFeedsFromEveryone(whereToWrite, updateInterval){
                                                                    /// Sanitizing the input (encodeURIComponent() is used instead of encodeURIComponent() when there has to be allowed the possibility to have hashtags.):
                                                                    whereToWrite = encodeURI(whereToWrite);
                                                                    updateInterval = encodeURIComponent(updateInterval);
                                                                    var tempSectionID;
                                                                    if(whereToWrite[0] === '#'){
                                                                    tempSectionID = whereToWrite.substring(1, whereToWrite.length);
                                                                    }else{
                                                                    tempSection = whereToWrite;
                                                                    whereToWrite = '#' + whereToWrite;
                                                                    }
                                                                    // Section check. If the HTML section is present in the webpage we can move on, otherwise the function has to stop.
                                                                    if( document.getElementById(tempSectionID) === null ){
                                                                    // Error. No HTML section found to display the followed feeds on Social. Please add a <div id="feedsFromEveryone"> section.
                                                                    return;
                                                                    }
                                                                    moreFeedsButtonPressed = false; // If the User has asked for a manual update then the automatic updates can be re-activated
                                                                    $('#feedsSectionName').html("#feedsFromEveryone"); // Writing the name of the section. It will be read from checkDataReceivedAndDisplayTheFeeds().
                                                                    executeRestCall("https://social.cern.ch/_api/social.feed/actor(item=@v)/feed?@v='https://espace2013.cern.ch/it-dep-ois/newsfeed.aspx'", 'GET', null,
checkDataReceivedAndDisplayTheFeeds, onError); // getting the feeds and passing them to the function checkDataReceivedAndDisplayTheFeeds()
                                                                    }
                                                                    }
                                                                    // Function used to automatically update the feeds every tot seconds.
function updateFollowedFeeds(whereToWrite, updateInterval, numFeeds, flagDisplayReplies){
                                                                    // Sanitizing the input. encodeURI() is used instead of encodeURIComponent() when there has to be allowed the possibility to have hashtags.
                                                                    whereToWrite = encodeURI(whereToWrite);
                                                                    updateInterval = encodeURIComponent(updateInterval);
                                                                    if(numFeeds == null || numFeeds == undefined){ numFeeds = 0; }
                                                                    if(flagDisplayReplies == null || flagDisplayReplies == undefined){ flagDisplayReplies = true; }
                                                                    // Consistency checks
                                                                    if( updateInterval===null || updateInterval===undefined || updateInterval<0){ updateInterval = 0; }

```

```

if(whereToWrite[0] != '#'){
    whereToWrite = '#' + whereToWrite;
}

var tempElement;
var parentWhereToWrite = whereToWrite;

// Updating the global variables. These variables will be necessary when the User needs to post a new message on Social
followedFeedsWhereToWrite = whereToWrite;
followedFeedsUpdateInterval = updateInterval;
followedFeedsNumFeeds = numFeeds;
followedFeedsFlagDisplayReplies = flagDisplayReplies;

if(updateInterval < 1000){ updateInterval = updateInterval*1000; } // Converting the time from seconds to milliseconds

var tempSectionID = whereToWrite.substring(1);
// Section check. If the HTML section is present in the webpage we can move on, otherwise the function has to stop.
if( document.getElementById(tempSectionID) === null ){
    // Error. No HTML section found to display the feeds from Social.
    $(whereToWrite).html('<div class="feedsItem"> <p id="text"> There has been a problem while communicating with the server. <br/>Please try again
later refreshing the page. </p> </div>');
    console.log('Error while trying to write the followed feeds. The section ID passed in input seems not to be present in the webpage.');
```

```

    return;
}

// If there are no feeds (there can be error message), so we clean the section
if( $(whereToWrite + " .feedsItem").length == '' || $(whereToWrite + " .feedsItem").length == null || $(whereToWrite + " .feedsItem").length == undefined ){
    $(whereToWrite).html(''); // Clearing the section of the feeds I am following
}

// Adding a new wrapping section in the HTML page to make the SocialAPI's CSS file point only at this part of the webpage, in case many CSS files are used.
if($(whereToWrite).html() == ''){ // If there are no feeds in the section yet...
    var wrapSection = '<div class="socialAPIWrapClass">'+
                                                                '<div id="socialAPIFollowedFeeds">'+
                                                                '</div>'+
                                                                '</div>';
    $(whereToWrite).html(wrapSection);
}

authenticateOnSocial(); // We need to re-authenticate on Social every time

if(updateInterval > 0){
    clearInterval(followedFeedsUpdatesHandler); // Deleting the old automatic refresh of the feeds

    // Creating the new automatic refresh of the feeds. The followed feeds will be updated every "updateInterval" seconds.
    followedFeedsUpdatesHandler = setInterval(function(){ updateFollowedFeeds(whereToWrite, updateInterval, numFeeds, flagDisplayReplies); },
updateInterval); // This variable will be necessary when the User needs to post a new message on Social

    whereToWrite = '#socialAPIFollowedFeeds'; // Updating the focused section that we will pass to the following function the new ID, which is
inside the new wrapper div.
    executeRestCallExtendedSix(myFeedManagerEndpoint + "my/news", 'GET', null, checkDataReceivedAndDisplayTheFeeds, onError, whereToWrite,
parentWhereToWrite, numFeeds, numFeeds, flagDisplayReplies); // searches the feeds and passes them to the function "checkDataReceivedAndDisplayTheFeeds()"
}
else{
    // If we are here it means that the function has to retrieve the feeds without automatically update them.

    if(followedFeedsUpdatesHandler != 'a'){ // If there is an active automatic update of the feeds
        clearInterval(followedFeedsUpdatesHandler); // Deleting the old automatic refresh of the feeds
        followedFeedsUpdatesHandler = 'a';
    }

    whereToWrite = '#socialAPIFollowedFeeds'; // Updating the focused section that we will pass to the following function the new ID, which is
inside the new wrapper div.
    executeRestCallExtendedSix(myFeedManagerEndpoint + "my/news", 'GET', null, checkDataReceivedAndDisplayTheFeeds, onError, whereToWrite,
parentWhereToWrite, numFeeds, numFeeds, flagDisplayReplies); // searches the feeds and passes them to the function "checkDataReceivedAndDisplayTheFeeds()"
}
}

// This is an object representing a User profile on Social and the section in the webpage in which we want its feeds be displayed.
function updateObj(keyValue, sectionID, automaticUpdatesHandlersCode, timeInterval, numOfFeedsToRetrieve, flagDisplayReplies){
    this.keyValue = keyValue;
    this.sectionID = sectionID;
    this.automaticUpdatesHandlersCode = automaticUpdatesHandlersCode;
    this.timeInterval = timeInterval;
    this.numOfFeeds = numOfFeedsToRetrieve;
    this.flagDisplayReplies = flagDisplayReplies;
}

/* This function check the presence of an object with one element which is equal to the sectionID in input in an array.
* Input:
* - the ID of the HTML section in which the feeds have to be displayed. It is the string that the function will try to find in the elements in the array
* - the array in which the function will look into.
* Output:
* - If the element has been found the function will return the index of the element in the array
* - else: it will return -1
*/
function checkPresenceOfElement(sectionID, array){
    var length = array.length; // This line of code will allow us to read to length of the array only once (and not at each cycle in the "for" statement),
speeding up the execution of the code.

    for(var i=0; i<length; i++){
        if( array[i].sectionID === sectionID ){
            return i;
        }
    }

    return -1;
}

```

```

function feedsToDisplayObj(sectionID, numFeedsAlreadyDisplayed){
    this.sectionID = sectionID;
    this.numFeedsAlreadyDisplayed = numFeedsAlreadyDisplayed;
}

// Function that retrieves the feeds from the page of an Actor on Social and print them in the webpage.
// The name of the actor is read from the html page, from a field invisible to the User.
function updateFeedsFromProfile(accountName, whereToWrite, updateInterval, numOfFeeds, flagDisplayReplies){
    // Consistency checks
    if( updateInterval===null || updateInterval===undefined || updateInterval<0){ updateInterval = 0; }
    if( numOfFeeds===null || numOfFeeds===undefined || numOfFeeds<0 || numOfFeeds>20){ numOfFeeds = 0; }
    if( flagDisplayReplies===null || flagDisplayReplies===undefined ){ flagDisplayReplies = true; }

    // Sanitizing the input (encodeURIComponent() is used instead of encodeURIComponent() when there has to be allowed the possibility to have hashtags.):
    accountName = encodeURIComponent(accountName);
    whereToWrite = encodeURI(whereToWrite);
    updateInterval = encodeURIComponent(updateInterval);
    numOfFeeds = encodeURIComponent(numOfFeeds);

    if(whereToWrite[0] != '#'){
        whereToWrite = '#' + whereToWrite;
    }

    var tempSection = whereToWrite.substring(1); // It will be the ID of the HTML section in which we want to write the information without the hashtag as first
character.
    var tempElement; // Temporary variable used to store new elements inside 'globalArrayOfProfiles'.
    var parentWhereToWrite = whereToWrite; // Memorizing the main section

    if(updateInterval < 1000){ updateInterval = updateInterval*1000; } // Converting the time from seconds to milliseconds

    // Section check. If the HTML section is present in the webpage we can move on, otherwise the function has to stop.
    if( document.getElementById(tempSection) === null ){
        // Error. No HTML section found to display the followed feeds on Social. Please add a <div id="feedsFromProfile"> section.
        console.log('Error while trying to write the feeds from the specific profile. See function updateFeedsFromProfile().');
        $(whereToWrite).append('<div class="feedsItem"> <p id="text"> There has been a problem while communicating with the server. <br/>Please try again
later refreshing the page. </p> </div>');
        return;
    }

    var innerWrap = "socialAPIFeedsFromProfile"+ whereToWrite.substring(1) + accountName;
    var wrapSection = '<div class="socialAPIWrapClass">'+
                                                                '<div id="'+ innerWrap +'>'+
                                                                '</div>'+
                                                                '</div>';

    // Adding a new wrapping section in the HTML page to make the SocialAPI's CSS file point only at this part of the webpage, in case many CSS files are used.
    if( $(whereToWrite + ".feedsItem").length == 0 && $(whereToWrite + ".socialAPIWrapClass").length == 0 ){ // If there is the HTML section and it
is still empty...
        $(whereToWrite).html(wrapSection);
    }

    authenticateOnSocial(); // We need to re-authenticate on Social every time

    try{
        if(updateInterval > 0){
            var indexOfElement = checkPresenceOfElement(parentWhereToWrite, globalArrayOfProfiles); // The function returns
-1 if the element is not in the array.
            // If the element is inside the array, than we have to clear the interval and pop the element from the array before creating a
new automatic update interval.
            if( indexOfElement > -1 && indexOfElement < globalArrayOfProfiles.length ){
                // Stopping the old automatic refresh of the feeds
                clearInterval(globalArrayOfProfiles[indexOfElement].automaticUpdatesHandlersCode);

                // Creating the new automatic refresh of the feeds
                globalArrayOfProfiles[indexOfElement].automaticUpdatesHandlersCode = setInterval(function(){
updateFeedsFromProfile(accountName, parentWhereToWrite, updateInterval, numOfFeeds, flagDisplayReplies); }, updateInterval); // The followed feeds will be updated every
"updateInterval" seconds

                whereToWrite = '#' + innerWrap;

                // Retrieving the feeds
                executeRestCallExtendedSix(myFeedManagerEndpoint + "actor(item='cern\\"+accountName+"')/feed", 'GET', null,
checkDataReceivedAndDisplayTheFeeds, onError, whereToWrite, parentWhereToWrite, numOfFeeds, numOfFeeds, flagDisplayReplies); // getting the feeds and passing them to the function
checkDataReceivedAndDisplayTheFeeds()
            }
            else{
                // Creating the automatic refresh of the feeds
                var tempHandler = setInterval(function(){ updateFeedsFromProfile(accountName, parentWhereToWrite, updateInterval,
numOfFeeds, flagDisplayReplies); }, updateInterval); // The followed feeds will be updated every "updateInterval" seconds
                tempElement = new updateObj(accountName, whereToWrite, tempHandler, updateInterval, numOfFeeds,
flagDisplayReplies);

                globalArrayOfProfiles.push(tempElement); // Inserting the new element in the 'globalArrayOfProfiles'

                whereToWrite = '#' + innerWrap;

                // Retrieving the feeds
                executeRestCallExtendedSix(myFeedManagerEndpoint + "actor(item='cern\\"+accountName+"')/feed", 'GET', null,
checkDataReceivedAndDisplayTheFeeds, onError, whereToWrite, parentWhereToWrite, numOfFeeds, numOfFeeds, flagDisplayReplies); // getting the feeds and passing them to the function
checkDataReceivedAndDisplayTheFeeds()
            }
        }
        else{
            // If we are here it means that the function has to retrieve the feeds without automatically update them.

            var indexOfElement = checkPresenceOfElement(parentWhereToWrite, globalArrayOfProfiles); // The function returns
-1 if the element is not in the array.
            // If the element is not inside the array, than we have to add it.
            if( indexOfElement === -1 ){
                tempElement = new updateObj(accountName, whereToWrite, null, 0, numOfFeeds, flagDisplayReplies);
                globalArrayOfProfiles.push(tempElement); // Inserting the new element in the 'globalArrayOfProfiles'
            }
        }
    }
}

```

```

        whereToWrite = '#' + innerWrap;

        executeRestCallExtendedSix(myFeedManagerEndpoint + "actor(item='cern\\\\" + accountName + "')/Feed", 'GET', null,
checkDataReceivedAndDisplayTheFeeds, onError, whereToWrite, parentWhereToWrite, numOfFeeds, numOFFeeds, flagDisplayReplies); // getting the feeds and passing them to the function
checkDataReceivedAndDisplayTheFeeds()
    }
}
catch(e){
    $(whereToWrite).html('<div>There has been a problem while retrieving the feeds. <br/>Please try again later. </div>');
}
}

// Function that retrieves the feeds with the same hashtag on Social and print them in the webpage.
function updateFeedsWithSameHashtag(tag, whereToWrite, updateInterval, numOFFeeds, flagDisplayReplies){
    // Consistency checks
    if( updateInterval===null || updateInterval===undefined || updateInterval<0){ updateInterval = 0; }
    if( numOFFeeds===null || numOFFeeds===undefined || numOFFeeds<0 || numOFFeeds>20){ numOFFeeds = 0; }
    if( flagDisplayReplies===null || flagDisplayReplies===undefined){ flagDisplayReplies = true; }

    // Sanitizing the input (encodeURIComponent() is used instead of encodeURIComponent() when there has to be allowed the possibility to have hashtags.):
    var noSharpTagArray = tag.split(' '); // Splitting the input tags from one string to an array of strings.
    var noSharpTagString = '';
    var noSpaceNoSharpTagString = ''; // This variable will be used for the innerWrap variable only.
    for( var i=0; i<noSharpTagArray.length; i++){
        if(noSharpTagArray[i][0] !== '#'){
            noSharpTagArray[i] = noSharpTagArray[i].substring(1, noSharpTagArray[i].length);
        }
        noSpaceNoSharpTagString += encodeURIComponent(noSharpTagArray[i]); // Adding the tag only
        if( i < noSharpTagArray.length-1 ){
            noSharpTagString += encodeURIComponent(noSharpTagArray[i]) + ' '; // Adding the tag plus an empty space
        }else{
            noSharpTagString += encodeURIComponent(noSharpTagArray[i]); // Adding the last tag
        }
    }
    whereToWrite = encodeURIComponent(whereToWrite);
    updateInterval = encodeURIComponent(updateInterval);
    numOFFeeds = encodeURIComponent(numOFFeeds);

    if(whereToWrite[0] !== '#'){
        whereToWrite = '#' + whereToWrite;
    }
    if(document.getElementById(whereToWrite.substring(1)) === null){
        console.log("The HTML section appears not to exist. See updateFeedsWithSameHashtag() function.");
        return;
    }

    var tempSection = whereToWrite.substring(1, whereToWrite.length);

    if(updateInterval < 1000){ updateInterval = updateInterval*1000; } // Converting the time from seconds to milliseconds

    var parentWhereToWrite = whereToWrite;

    // Section's check. If the HTML section is present in the webpage we can move on, otherwise the function has to stop.
    if( document.getElementById(tempSection) === null ){
        // Error. No HTML section found to display the followed feeds on Social. Please add a <div id="feedsWithSameHashtag"> section.
        console.log('Error while trying to write the feeds with the same hashtag. The section ID passed in input is not present in the web page.');
```

```

globalArrayOfHashtags.push(new updateObj(noSharpTagString, whereToWrite, null, 0, numOfFeeds,
flagDisplayReplies)); // Adding the element to the array
}
}

whereToWrite = "$" + innerWrap; // We want to write in the inner section. */
// Retrieving the feeds with the same tag(s) and writing them in the section of the HTML page with ID='feedsWithSameHashtag'.
retrieveFeedsWithSameTag(noSharpTagString, whereToWrite, parentWhereToWrite, numOfFeeds, flagDisplayReplies);
} catch(e) {
$(whereToWrite).html('There has been an error while trying to write the feeds with the same hashtag. Please try again later.');
```

```

}

// This function returns the correspondent handler for the 'whereToWrite' section ID.
function findMyHandler(whereToWrite){
var length;

if(whereToWrite === followedFeedsWhereToWrite){
return followedFeedsUpdatesHandler;
} else {
length = globalArrayOfProfiles.length;
for(var i=0; i<length; i++){
if(whereToWrite === globalArrayOfProfiles[i].sectionID){
return globalArrayOfProfiles[i].automaticUpdatesHandlersCode;
}
}

// If the element has not yet been found...
length = globalArrayOfHashtags.length;
for(var i=0; i<length; i++){
if(whereToWrite === globalArrayOfHashtags[i].sectionID){
return globalArrayOfHashtags[i].automaticUpdatesHandlersCode;
}
}

// If the handler has not been found...
return -1;
}

// This function returns the correspondent handler for the 'whereToWrite' section ID.
function findMyUpdateInterval(whereToWrite){
var length;

if(whereToWrite === followedFeedsWhereToWrite){
return followedFeedsUpdateInterval;
} else {
length = globalArrayOfProfiles.length;
for(var i=0; i<length; i++){
if(whereToWrite === globalArrayOfProfiles[i].sectionID){
return globalArrayOfProfiles[i].timeInterval;
}
}

// If the element has not yet been found...
length = globalArrayOfHashtags.length;
for(var i=0; i<length; i++){
if(whereToWrite === globalArrayOfHashtags[i].sectionID){
return globalArrayOfHashtags[i].timeInterval;
}
}

// If the handler has not been found...
return -1;
}

/*
* This function displays the feeds into the webpage.
* Input:
* - the feeds to display
* - the ID of the HTML section where to display the feeds
* - the ID of the parent HTML section where to display the feeds
* - the maximum number of feeds to display
*/
function appendFeeds(feeds, whereToWrite, parentWhereToWrite, numFeedsTotal, numFeedsStillToGet, flagDisplayReplies){
var numFeedsToDisplay;
if( (typeof numFeedsTotal) === 'string'){ numFeedsTotal = parseInt(numFeedsTotal); }
if( (typeof numFeedsStillToGet) === 'string'){ numFeedsStillToGet = parseInt(numFeedsStillToGet); }
if( (typeof flagDisplayReplies) === 'string'){
if(flagDisplayReplies == "false"){
flagDisplayReplies = false;
} else {
flagDisplayReplies = true;
}
}

if( feeds.length == 0 ){
// Printing the "no feed" message on the screen
var uniqueId = 'networkProblemsWarning'+Date.now(); // It is important to have a unique id, otherwise the function 'fadeOut' will work
only once.
$(whereToWrite).append('<div class="feedsItem" id="'+uniqueId+'"> <p id="text"> No (more) feeds available. </p></div>'); //
Printing the "problem" message on the screen
$("#"+uniqueId).delay(5000).fadeOut('slow'); // This function will hide the warning after 5 seconds.
setTimeout(function(){ $("#"+uniqueId).remove(); }, 7000); // This function will remove the warning from the HTML code after 7 seconds.
return;
}

if(numFeedsTotal == null || numFeedsTotal == undefined || numFeedsTotal <= 0 || numFeedsTotal > 20){
numFeedsToDisplay = 100; // The highest number of feeds will be 100.
}

```

```

    }
    else{
        numFeedsToDisplay = Math.min(feeds.length, numFeedsStillToGet);
    }

    var numFeedsDisplayed = 0;

    var picturePresentFlag = false;
    // Reading if there is already a "picSection" inside the "whereToWrite" section of the page.
    if( document.getElementById(whereToWrite.substring(1)).getElementsByClassName('picSection').length > 0 ){ picturePresentFlag = true ; }

    // If we don't want to display the replies we want to show the profile Picture and Name ONCE. Now:
    if(feeds.length > 0 && (!flagDisplayReplies) && (!picturePresentFlag)){
        var participants = feeds[0].Actors.results;
        var accountName, profilePicUri;
        if( participants[feeds[0].OwnerIndex].AccountName === null || participants[feeds[0].OwnerIndex].AccountName === undefined ){
            accountName = participants[feeds[0].OwnerIndex+1].AccountName.split("\\");
            profilePicUri = participants[feeds[0].OwnerIndex+1].ImageUri; // Reading the URI of the profile picture of the User.
            owner = participants[feeds[0].OwnerIndex+1].Name;
        }else{
            accountName = participants[feeds[0].OwnerIndex].AccountName.split("\\");
            profilePicUri = participants[feeds[0].OwnerIndex].ImageUri; // Reading the URI of the profile picture of the User.
            owner = participants[feeds[0].OwnerIndex].Name;
        }
        accountName = accountName[accountName.length-1];
        var personalAboutPage = socialWebsite + "Person.aspx?accountname=CERN%5C" + accountName;
        // Consistency check. If no picture is found -> use the anonymous profile picture.
        if( profilePicUri === null || profilePicUri === undefined || profilePicUri === "" ){
            profilePicUri = socialWebsite + "_layouts/15/images/PersonPlaceholder.42x42x32.png?rev=23";
        }

        var group = ''; // The variable that will (if present) store the title of the group in which the User posted the
feed.
        if(participants[0].ActorType === 2){
            group = participants[0].Name;
        }
        var groupString = ''; // The string that will tell the group in which the User posted the feed
        if(group !== ''){
            groupString = '&nbsp;<span>' + group + '</span>'; // output example: " > IT/OIS"
        }

        if( participants[feeds[0].OwnerIndex].IsFollowed ){
            var authorString = '<span id="author"> <a href="'+personalAboutPage+'" target="_blank"> ' + owner + ' </a> </span> ' + groupString;
        }else{
            var authorString = '<span id="author"> <a href="'+personalAboutPage+'" target="_blank"> ' + owner + ' </a> </span> ' + groupString;
        }

        $(whereToWrite).append('<div class="feedsItem" id="beginFeedsSection"> ' +
                                                                    '<div class="picSection"> ' +
                                                                    '<a
                                                                    <img
                                                                    </a> ' +
                                                                    '</div> ' +
                                                                    '<div class="notPicSection"> ' +
                                                                    '<p> ' + authorString +
                                                                    '</p> ' +
                                                                    '</div> ' +
                                                                    '</div>');
    }else{
        // We still apply our initial div section to allow the code write the new feeds at the beginning of the section without refreshing the whole
section.
        if( $(whereToWrite).html() == '' ){ // If there are no feeds displayed yet...
            $(whereToWrite).append('<div id="beginFeedsSection"> </div>');
        }
    }

    // Reading the most recent feed's ID from the ones already displayed.
    var addingFeedsFlag = false;
    var dateOfTheLatestFeedAlreadyDisplayed;
    try{
        var mostRecentId = $(whereToWrite + "#feedId").html(); // reading the IDs of the feeds already displayed.
        if( typeof(mostRecentId) == 'string' && mostRecentId != null && mostRecentId != undefined && mostRecentId != '' ){
            addingFeedsFlag = true; // There are already feeds in the page. We are adding feeds.
        }
        dateOfTheLatestFeedAlreadyDisplayed = $(whereToWrite + ".feedsItem .date")[0].innerHTML; // Reading the date of the first feed in the HTML
section which is the date of the earliest feed retrieved so far.
        dateOfTheLatestFeedAlreadyDisplayed = new Date(dateOfTheLatestFeedAlreadyDisplayed); // Re-creating the Date obj from the
information found in the HTML section
    }catch(e){}

    // If there are already some feeds displayed...
    if(addingFeedsFlag == true ){
        // We want to read the feeds in the array that are not yet displayed. Since the array is ordered having the most recent one at the beginning we
will copy the feeds that are not yet displayed until we find the array coming from the Server the last one displayed from the last session.
        var tempIndex = 0;
        var length = feeds.length;
        var tempArray = new Array();
        // We want to take only the feeds that are not in the webpage already.
        while( tempIndex < length && $(whereToWrite).html().indexOf(feeds[tempIndex].id) == -1 ){
            tempArray.push(feeds[tempIndex]);
            tempIndex++;
        }
        feeds = tempArray;

        if(feeds.length == 0){
            var uniqueId = 'networkProblemsWarning'+Date.now(); // It is important to have a unique id, otherwise the function
'fadeOut' will work only once.
            $(whereToWrite).append('<div class="feedsItem" id="'+uniqueId+'"> <p id="text"> No new feeds available. </p> </div>'); //
Printing the "problem" message on the screen
            $("#"+uniqueId).delay(5000).fadeOut('slow'); // This function will hide the warning after 5 seconds.
        }
    }
}

```



```

seconds.
        setTimeout(function() { $("#"+uniqueId).remove(); }, 7000); // This function will remove the warning from the HTML code after 7
        return;
    }

    try{
        var dateOfTheLatestFeedRetrieved = new Date(feeds[0].RootPost.CreatedTime);
        // If we created a correct Date object and there are some more feeds coming from Social but those have been already displayed...
        if( dateOfTheLatestFeedAlreadyDisplayed instanceof Date && dateOfTheLatestFeedRetrieved >= dateOfTheLatestFeedAlreadyDisplayed
    ){
        feeds = feeds.reverse(); // Inverting the array of feeds to let the function add the oldest one first using
    "insertAfter".

        var howToWriteFeeds = "prepend";
    }
    else{
        // We are dealing with older feeds, we want them to be displayed after the ones already in the webpage
        var howToWriteFeeds = "append";
    }
    }catch(e){ var howToWriteFeeds = "append"; }

    // Foreach feed
    var i = 0, thread, participants;
    while( numFeedsDisplayed < numFeedsToDisplay && i < feeds.length ){

        thread = feeds[i]; // Capturing the i-th feed
        i++; // Increasing the index

        // If the feed has already been displayed we can analyze the following one
        participants = thread.actors.results; // Reading the creators of the feed
        var group = ''; // The variable that will (if present) store the title of the group in which the User posted the
        feed.

        if(participants[0].ActorType === 2){
            group = participants[0].Name;
        }

        // If the feed is on a page like IT/OIS that the author will be the one following the Owner.
        var owner;
        var accountName;
        var tempIndex;
        var profilePicUri;
        // Reading the name of the owner of the feed
        if( participants[thread.OwnerIndex].AccountName === null || participants[thread.OwnerIndex].AccountName === undefined ){
            owner = participants[thread.OwnerIndex+1].Name;
            accountName = participants[thread.OwnerIndex+1].AccountName.split("@");
            tempIndex = thread.OwnerIndex+1;
            profilePicUri = participants[thread.OwnerIndex+1].ImageUri; // Reading the URI of the profile picture of the User.
        }
        else{
            owner = participants[thread.OwnerIndex].Name;
            accountName = participants[thread.OwnerIndex].AccountName.split("@");
            tempIndex = thread.OwnerIndex;
            profilePicUri = participants[thread.OwnerIndex].ImageUri; // Reading the URI of the profile picture of the User.
        }

        accountName = accountName[accountName.length-1]; // From something like "CERN/mcavalaz" we save "mcavalaz".
        var personalAboutPage = socialWebsite + "Person.aspx?accountname=CERN%5C" + accountName;

        var dateTimeFeed = new Date(thread.RootPost.CreatedTime); // The Date construct allows the User to automatically
        see the local time on the webpage

        var dateString = createDateString(dateTimeFeed);

        var threadId = thread.Id; // This is important to memorize on the html page. It will not be shown to the User, but it will become very useful
        for the other functions (e.g.: for the Replies).

        // Consistency check. If no picture is found -> use the anonymous profile picture.
        if( profilePicUri === null || profilePicUri === undefined || profilePicUri === "" ){
            profilePicUri = socialWebsite + "_layouts/15/images/PersonPlaceholder.42x42x32.png?rev=23";
        }

        // Checking the presence of attachments, like image, to the feeds.
        var att = thread.RootPost.Attachment;
        var attachmentUri = null;
        // If there is any attachment...
        if( att !== null && att !== undefined ){
            // and it is an image (image -> AttachmentKind = 0)...
            if( att.AttachmentKind === 0 ){
                attachmentUri = att.Uri; // Memorizing the URI of the image attachment
            }
        }
        var attachmentString;
        if( attachmentUri === undefined || attachmentUri === null || attachmentUri === '' ){
            attachmentString = '';
        }
        else{
            attachmentString = '<p>
        }

        var actorId; // This variable will be used for the feeds from other Users.
        var likeCounterString; // The string stating the number of people that likes the post
        var text = thread.RootPost.Text; // The text of the message

        // If the feed retrieved is only a message from the system like "Marco is now following Eduardo" LikerInfo will be undefined or null, therefore...
        if(thread.RootPost.LikerInfo === undefined || thread.RootPost.LikerInfo === null)
        {
            continue;
        }
        // If we reach this line the feed will be displayed.
        numFeedsDisplayed++; // Increasing the number of feeds displayed.

        text = myEscapeHTML(text); // Preventing code injection!
    }
}

```

```

text = formatText(text, parentWhereToWrite);

/*****
 *      Checking the existence of people that like the message.
 *****/
// If the number of people who likes this post is 0 (no-one)
if(thread.RootPost.LikerInfo.TotalCount == 0 || thread.RootPost.LikerInfo.TotalCount === null || thread.RootPost.LikerInfo.TotalCount ===
undefined){
    likeCounterString = ""; // If nobody liked the feed nothing particular is shown
}else{
    // If someone liked the post... e.g. 23 liked the post -> (smile 23)
    likeCounterString = "<span class=\"smile\"></span> <b> " + thread.RootPost.LikerInfo.TotalCount + "</b> &nbsp;";
}

// If the User already likes the feed...
if(thread.RootPost.LikerInfo.IncludesCurrentUser){
    var likeString = '<a onclick="socialAPI().unlikeFeedFunction({#39;'+threadId+' '+whereToWrite+'#39;}"
href="javascript:void(0);"> Unlike </a>';
}else{
    var likeString = '<a onclick="socialAPI().likeFeedFunction({#39;'+threadId+' '+whereToWrite+'#39;}" href="javascript:void(0);">
Like </a>';
}

try{
    // To 'Unfollow' that person the User will need the 'actorId'.
    actorId = participants[thread.OwnerIndex].Id;
}catch(e){}

var updateInterval = findMyUpdateInterval(whereToWrite);

var deleteString;
if( whereToWrite === "#socialAPIFollowedFeeds"){
    if( participants[thread.OwnerIndex].IsFollowed ){
        // therefore it will not be possible for the User to Delete this feed.
        // Instead, the User will be able to 'Unfollow' that person.
        deleteString = '<a href="javascript:void(0);" id="deleteFeed" onclick="socialAPI().unfollowPerson({#39;'+owner+'-
-'+actorId+'-'+participants[tempIndex].AccountName.replace("\\", "\\")+'-'+whereToWrite+'-'+updateInterval+'#39;}"> <b> X </b> </a>';
    }else{
        deleteString = '<a href="javascript:void(0);" id="deleteFeed"
onclick="socialAPI().deleteFeed({#39;'+threadId+'#39;}"> <b> X </b> </a>';
    }
}else{
    deleteString = '';
}

var groupString = ''; // The string that will tell the group in which the User posted the feed
if(group != ''){
    groupString = '&nbsp;<span>> ' + group + '</span>'; // output example: " > IT/OIS"
}

// If the creator of the thread is followed it means that it is NOT the User
if( participants[thread.OwnerIndex].IsFollowed ){
    var authorString = '<span id="author"> <a href="'+personalAboutPage+'" target="_blank"> ' + owner + ' </a> </span> ' + groupString
+ deleteString;
    var actorString = '<p id="feedId" class="actorId'+threadId+'">'+actorId+'</p>';
}else{
    var authorString = '<span id="author"> <a href="'+personalAboutPage+'" target="_blank"> ' + owner + ' </a> </span> ' + groupString
+ deleteString;
    var actorString = ''; // There is no need of the actorId if the feed is from the User itself
}

// If the feeds are to be displayed without replies we want to show only once the profile Picture and Name
var profilePicString;
if(!flagDisplayReplies){
    profilePicString = '<div>'; // Hiding totally the picture and the name of the author, allowing at the same time the text section
of the feed to be shown.
}else{
    // Standard format of the picture section
    profilePicString = '<div class="picSection"> ' +
target="_blank">' +
src="'+profilePicUri+' id="profilePicture" /> ' +
'<img
'</a>' +
'</div> ' +
'<div class="notPicSection"> ' +
'<p> ' + authorString + '</p>';
}

var conversationUri = thread.Permalink; // This will be the link to the conversation on Social

var openConversationString = '';
var replySectionString = '';
if(!flagDisplayReplies){ // In case we don't want to see the replies we will not show the "Reply" button also, and we will instead show the
link to the conversation for every feed to let the Users reply if they want to.
    openConversationString = '<p class="openConversationLink"><a href="javascript:void(0);"
onclick="socialAPI().moreRepliesFunction({#39;'+ thread.Permalink +'#39;});"> > Open conversation </a></p>';
}else{
    replySectionString = '<a onclick="socialAPI().showReplySection({#39;'+parentWhereToWrite+' '+whereToWrite+' '+threadId+'#39;}"
href="javascript:void(0);"> Reply </a> </span>';
// If more than 2 replies are present for this thread only 2 will be shown now.
// A button is created for the User to see the other replies if needed.
if(thread.TotalReplyCount >= 3){
    // Adding an extra element to the replies section.
    // This button will allow the User to ask for more replies.
}
}
}

```

```

        openConversationString = '<div class="openConversationLink">a href="javascript:void(0);"
onclick="socialAPI().moreRepliesFunction(&#39;'+ thread.Permalink + '&#39;);"> > Open entire conversation </a></div>';
    }
    else{
        openConversationString = '';
    }
}

var strOutput = '<div class="feedsItem" id="feedsItem'+threadId+'"> ' +
    '<div class="table">' +
        profilePicString +
        '<p id="feedText'+threadId+'"> ' + text
+ '</p>' +
        attachmentString +
        '<span class="noWrapString"><span
class="date" id="date'+threadId+'"> ' + dateString + '</span> &nbsp;&nbsp;&nbsp;&nbsp;</span> <span class="slideLeft">&nbsp;&nbsp;</span> ' +
        '<span class="noWrapString">' +
id="feed'+threadId+'"><span id="likeCounter'+threadId+'"> ' + likeCounterString + '</span> ' + likeString + '&nbsp;&nbsp;&nbsp;&nbsp;' + replySectionString + '</span>' +
        '</span>' +
class="feedId'+threadId+'">' + threadId + '</p>' +
        actorString +
        '</div>' +
        '</div>';

// If we are writing new feeds (there are already some in the HTML web section).
if( howToWriteFeeds == "prepend" ){
    var newFeedsSection = $(whereToWrite
#beginFeedsSection");//document.getElementById(whereToWrite).getElementsByById('beginFeedsSection'); // Reading the div section at the beginning and inside of the HTML section
"whereToWrite"

    if(!flagDisplayReplies){ // Adding a separation line between the feeds only if there have to be no replies.
        $('<hr style="border-top: dotted 1px; margin:0; padding:0 0 8px; clear:both; height:0;"
/>').insertAfter(newFeedsSection);
    }
    // Displaying the feed
    $(strOutput).hide().insertAfter(newFeedsSection).fadeIn(800 + (i*120)); // The (i*120) helps creating a cool
effect so that the feeds are displayed fading in one after another, instead than fading in all at the same time.
}
else{
    // We are adding the feeds in the section for the first time
    // Displaying the feed
    $(strOutput).hide().appendTo(whereToWrite).fadeIn(800 + (i*120)); // The (i*120) helps creating a cool effect so that
the feeds are displayed fading in one after another, instead than fading in all at the same time.

    if(!flagDisplayReplies){ // Adding a separation line between the feeds only if there have to be no replies.
        $(whereToWrite).append('<hr style="border-top: dotted 1px; margin:0; padding:0 0 8px; clear:both; height:0;" />');
    }
}

// Appending first the hidden reply textarea
var tempParentWhereToWrite;
if( parentWhereToWrite[0] === '*' ){ tempParentWhereToWrite = parentWhereToWrite.substring(1); }
else{ tempParentWhereToWrite = parentWhereToWrite; }
var thisFeedSection = $(whereToWrite) #feedsItem'+threadId.replace(/(:|\.|\/|\/g, "\\$1"); // Catching this feed's section
// Appending the reply section that will be shown when pressing the "reply" button of a feed.
$('<div class="textAreaReply" id="textAreaReply'+threadId+'"> ' +
    '<p id="textAreaReplySection"> <textarea placeholder="" wrap="hard" id="textAreaReply" class="textAreaReply"
tempParentWhereToWrite + threadId + "></textAreaReply" /> ' +
    '<p class="replyButtonsGroup"> <input type="button" value="Reply" id="replyButton" class="uploadMessage"
tempParentWhereToWrite + threadId + " onclick="socialAPI().createReply(&#39;'+tempParentWhereToWrite+' '+whereToWrite+' '+threadId+'&#39;)" /> ' +
    '</div>').insertAfter(thisFeedSection);
// To include also the "Clear text area" button, use this code: <input type="button" value="Clear text" id="replyButton"
class="clearMessage'+threadId+' onclick="clearReplyText(&#39;'+tempParentWhereToWrite+' '+whereToWrite+' '+threadId+'&#39;)" />

// We are now hiding the textAreaReply section. This has to be done here and not in the CSS because otherwise it will not work well in IE
(even IE11), causing the whole page to crash if Enter is pressed while the cursor is inside the textbox (my personal comment: <the "good" old IE>).
var elem = getElementInsideContainer(whereToWrite, "textAreaReply" + tempParentWhereToWrite + threadId); // Getting the element of the
'textAreaReply' just appended to the 'whereToWrite' section.
$(elem).hide();

if(!flagDisplayReplies){ // If we don't want to display the replies
    continue; // we can stop here and go on to the next feed.
}

var replies = thread.Replies.results; // Catching the eventually present replies of this thread

// We now look at the replies for this particular post. If any is found it is showed to the User.
// The User is able to delete answers written by other people.
if(replies.length > 0){
    var parts;
    var prefix = "http://";
    var index;

    for(var j=0; j < replies.length; j++) {
        /* The Server gives us the replies in inverted chronological order (most recent feed first),
        * but we want to display them like on Social, so oldest first
        * --> then we will use [ replies.length -1 -j ] as index for the single reply to be displayed. */
        var reply = replies[ replies.length -1 -j ]; // Capturing a single
reply

        var creatorOfTheReply = thread.actors.results[reply.AuthorIndex]; // Reading the creator of the reply

        // Checking the presence of attachments, like image, to the feeds.
        att = reply.Attachment;

```

```

attachmentUri = null; // Resetting the attachment Uri
// If there is any attachment...
if( att !== null && att !== undefined ){
    // and it is an image (image -> AttachmentKind = 0)...
    if( att.AttachmentKind === 0 ){
        attachmentUri = att.Uri; // Memorizing the URI of the image attachment
    }
}
var attachmentString;
if( attachmentUri === undefined || attachmentUri === null || attachmentUri === '' ){
    attachmentString = '';
}else{
    attachmentString = ' </p>';
}

// "accountName" and "personalAboutPage" has already been declared before, so we will not use the keyword "var".
accountName = creatorOfTheReply.AccountName.split("\\");
accountName = accountName[accountName.length-1];
personalAboutPage = socialWebsite + "Person.aspx?accountname=CERN%5C" + accountName;

dateTime = new Date(reply.CreatedTime);
dateString = createDateString(dateTime);

replyId = reply.Id; // This is important to memorize on the html page. It will not be shown to the User, but it
will become very useful for the other functions (e.g.: for the Replies).

profilePicUri = creatorOfTheReply.ImageUri; // Reading the URI of the profile picture of the User.
// Consistency check. If no picture is found -> use the anonymous profile picture.
if( profilePicUri === null || profilePicUri === undefined || profilePicUri === "" ){
    profilePicUri = socialWebsite + "_layouts/15/images/PersonPlaceholder.42x42x32.png?rev=23";
}

text = reply.Text;
text = myEscapeHTML(text); // Preventing code injection!
text = text.replace(/\n/g, "<br>"); // Replacing all the new line character ('\n') with
the equivalent in HTML.

text = formatText(text, whereToWrite); // This function will adapt the text to our needs

/*****
 * Checking the existence of people that like the message.
 *****/
// If the number of people who likes this reply is 0 (no-one)
if(reply.LikerInfo.TotalCount === 0){
    likeCounterString = ""; // If nobody liked the reply nothing particular is
shown
}else{
    // If someone liked the post... e.g. 23 liked the reply -> (smile 23)
    likeCounterString = "<span class='smile'></span> <b>" + reply.LikerInfo.TotalCount + "</b>";
}

// If the User likes the reply we will show the 'Unlike' button
if(reply.LikerInfo.IncludesCurrentUser){
    likeString = '<span <a onclick="socialAPI().unlikeReplyFunction({#39;'+threadId+' '+j+'
'+whereToWrite+'&#39;)" href="javascript:void(0);"> Unlike </a> </span>';
}else{
    // Otherwise we show the 'Like' button
    likeString = '<span <a onclick="socialAPI().likeReplyFunction({#39;'+threadId+' '+j+'
'+whereToWrite+'&#39;)" href="javascript:void(0);"> Like </a> </span>';
}

if( whereToWrite === "#socialAPIFollowedFeeds"){
    deleteString = '<span id="deleteReply"> <a href="javascript:void(0);" id="deleteReply"
onclick="socialAPI().deleteReply({#39;'+replyId+'&#39;})"> X </b> </a> </span>';
}else{
    deleteString = '';
}

/* id = var specifying the feed.
 * j = var specifying the reply.
 * id j = the reply number j of the feed number id. */
var replyStr = '<div class="replyItem" id="replyItem'+threadId+' '+j+'"> ' +
'<div class="table"> ' +
'<div ' +
'<a ' +
'href="'+personalAboutPage+'" target="_blank"> ' +
' ' +
'</a> ' +
'</div> ' +
'<div ' +
class="noPicSection"> ' +
'<p>
<span id="author"> <a href="'+personalAboutPage+'" target="_blank">'+ creatorOfTheReply.Name + '</a> </span> '+ deleteString + ' </p> ' +
'<p>
id="replyText'+threadId+' '+j+'">'+ text + '</p> ' +
attachmentString +
'<span ' +
class="noWrapString"><span class="date">'+ dateString + '</span> &nbsp;&nbsp;&nbsp;&nbsp;</span> <span class="slideLeft">&nbsp;&nbsp;&nbsp;</span> ' +
'<span ' +
class="noWrapString">' +
'<span id="reply'+threadId+' '+j+'"><span id="likeCounter'+threadId+' '+j+'">'+ likeCounterString + '</span>'+ likeString + '</span> ' +
'</span>' +

```



```

var gap;
if(socialWebsite === "https://social-dev.cern.ch/"){
    gap = '<a href="'+ socialWebsite + 'search/Pages/conversationresults.aspx?k=23'+ tag.substring(1, tag.length) +'>' target="_blank"><strong>'+ tag
+ '</strong></a>';
}
}else{
    // else: we are in the Production environment
    gap = '<a href="'+ socialWebsite + 'search/Pages/results.aspx?k=23'+ tag.substring(1, tag.length) +'>' target="_blank"><strong>'+ tag
+ '</strong></a>';
}
}
// var functionInputStr = ""+ tag +"'," + whereToWrite +"", 0"; // we have to adopt this method to pass the two input variable to the next function.
// var gap = '<a onclick="socialAPI().updateFeedsWithSameHashtag('+ functionInputStr +')" href="javascript:void(0)"><strong>'+ tag + '</strong></a>';

var textLength = text.length;

var textBeforeTheTag = text.substring(0, index);
var textAfterTheTag = text.substring(index+tag.length, text.length);
var textAndLengthOfString = new textObj(textBeforeTheTag + gap + textAfterTheTag, gap.length - tag.length);

return textAndLengthOfString;
}

function textObj(text, length){
    this.text = text;
    this.gap = length;
}

// This function checks if there is a substring inside a longer string
function compareSubstring(str, startIndex, numberOfCharsToCheck, strToCompare){
    var index = startIndex;
    var j=0;

    while( j < numberOfCharsToCheck ){
        if( str[index] !== strToCompare[j] ){
            return false;
        }
        // else: the character is the same

        index++;
        j++;
    }
    // all the characters found are equal

    return true; // the strings are equal
}

/* This function creates the string that will display the date and time of each feed and reply.
* Input:
* - the date object of the feed or reply */
function createDateString(dateObj){
    var day = dateObj.getDay();

    switch(day){
        case 0: day="Sun"; break;
        case 1: day="Mon"; break;
        case 2: day="Tue"; break;
        case 3: day="Wed"; break;
        case 4: day="Thu"; break;
        case 5: day="Fri"; break;
        case 6: day="Sat"; break;
        default: day = "Mon"; break;
    }

    var month = dateObj.getMonth();

    switch(month){
        case 0: month="Jan"; break;
        case 1: month="Feb"; break;
        case 2: month="Mar"; break;
        case 3: month="Apr"; break;
        case 4: month="May"; break;
        case 5: month="Jun"; break;
        case 6: month="Jul"; break;
        case 7: month="Aug"; break;
        case 8: month="Sep"; break;
        case 9: month="Oct"; break;
        case 10: month="Nov"; break;
        case 11: month="Dec"; break;
        default: month="Jan"; break;
    }
}

```



```

// else: The button is in a section feedsFromProfile...

// Example of website to refer to:
https://social.cern.ch/_api/social.feed/actor(item='cern\actorName')/feed(OlderThan=@v)?@v=datatime'2014-01-20T07:52:40.55679532'
executeRestCallExtendedSix(myFeedManagerEndpoint +
"actor(item='cern\actorName')/feed(OlderThan=@v)?@v=datatime'+@date'+", 'GET', null, moreFeedsBodyFunction, onError, whereToWrite, parentWhereToWrite, numFeedsToDisplay,
numFeedsStillToGet, flagDisplayReplies); // searches the feeds and passes them to the function "moreFeedsBodyFunction()"
}
}

function moreFeedsBodyFunction(data, whereToWrite, parentWhereToWrite, numFeedsToDisplay, numFeedsStillToGet, flagDisplayReplies){
try{
var result = JSON.parse(data); // parsing the data obtained from the social network
}catch(e){
var uniqueId = 'networkProblemsWarning'+Date.now(); // It is important to have a unique id, otherwise the function 'fadeOut' will work
only once.
$(whereToWrite).append('<div class="feedsItem" id="'+uniqueId+'"> <p id="text"> Network problem. Please try to refresh the page later. </p>
</div>'); // Printing the "problem" message on the screen
$("#"+uniqueId).delay(5000).fadeOut('slow'); // This function will hide the warning after 5 seconds.
setTimeout(function(){ $("#"+uniqueId).remove(); }, 7000); // This function will remove the warning from the HTML code after 7 seconds.

console.log("There was a problem while communicating with the Server.\nSee moreFeedsBodyFunction() function.");
return;
}
// If the result object exists but has errors it means we made a bad request, than we have to stop the execution and signal it to the developer in the console
log
if(result.error){
var uniqueId = 'networkProblemsWarning'+Date.now(); // It is important to have a unique id, otherwise the function 'fadeOut' will work
only once.
$(whereToWrite).append('<div class="feedsItem" id="'+uniqueId+'"> <p id="text"> Network problem. Please try to refresh the page later. </p>
</div>'); // Printing the "problem" message on the screen
$("#"+uniqueId).delay(5000).fadeOut('slow'); // This function will hide the warning after 5 seconds.
setTimeout(function(){ $("#"+uniqueId).remove(); }, 7000); // This function will remove the warning from the HTML code after 7 seconds.

console.log("Bad request.\nPlease review the moreFeedsBodyFunction() function.");
return;
}

// Consistency check : if no information has been retrieved...
if(result.d === null || result.d === undefined)
{
var uniqueId = 'networkProblemsWarning'+Date.now(); // It is important to have a unique id, otherwise the function 'fadeOut' will work
only once.
$(whereToWrite).append('<div class="feedsItem" id="'+uniqueId+'"> <p id="text"> Network problem. Please try to refresh the page later. </p>
</div>'); // Printing the "problem" message on the screen
$("#"+uniqueId).delay(5000).fadeOut('slow'); // This function will hide the warning after 5 seconds.
setTimeout(function(){ $("#"+uniqueId).remove(); }, 7000); // This function will remove the warning from the HTML code after 7 seconds.

return;
}
// else...

// Stopping the automatic retrieval of new feeds that would hide the "extra" feeds that the User may be reading at the moment.
// To refresh the feeds and reactivate the automatic retrieval (of the feeds) the User may click on the "News" button on the right top of the page, or refresh
the whole web-page.
try{
var handler = findMyHandler(whereToWrite);
if(handler !== -1){
clearInterval(handler);
}

var button = getElementInsideContainer(whereToWrite, "moreFeedsButton");
button.parentNode.removeChild(button); // Removing the "Show more posts" button, if there is.
}catch(err){}

var feeds = result.d.SocialFeed.Threads.results; // capturing the feeds

// If no feed is found.. (it is an array, so we can check the length)
if(feeds.length === 0){
// Printing the "no feed" message on the screen
var uniqueId = 'networkProblemsWarning'+Date.now(); // It is important to have a unique id, otherwise the function 'fadeOut' will work
only once.
$(whereToWrite).append('<div class="feedsItem" id="'+uniqueId+'"> <p id="text"> No more feeds available. </p> </div>'); // Printing the
"problem" message on the screen
$("#"+uniqueId).delay(5000).fadeOut('slow'); // This function will hide the warning after 5 seconds.
setTimeout(function(){ $("#"+uniqueId).remove(); }, 7000); // This function will remove the warning from the HTML code after 7 seconds.

return;
}
else{
//else: every feed found is printed

appendFeeds(feeds, whereToWrite, parentWhereToWrite, numFeedsToDisplay, numFeedsStillToGet, flagDisplayReplies); //
Appending the new feeds to the previous ones.

}
}

// This function opens the thread considered, with all of its replies, in a new tab in the browser.
// It opens the website memorized in the "permalink" variable into a new tab.
function moreRepliesFunction(permalink){
window.open(permalink, '_blank');
}

// This function deletes the feed considered (the i-th feed)
// 'id' is the unique id of the feed that has to be deleted
function deleteFeed(id){

```



```

var x = confirm("Are you sure you want to get rid of this conversation?");
if(x === false) { return; } // If the User clicks on 'No' then -> do nothing; else: continue

$('#tempID').html(id); // Memorizing the id into an invisible html field

executeRestCall(formDigestUrl, 'POST', null, deleteFeedFunction, onError);
}

// This function follows "deleteFeed(i)".
// If the feed has any reply the feed is deleted and the replies are deleted with it.
function deleteFeedFunction(data){

    try{
        var result = JSON.parse(data); // parsing the data obtained from the social network
    }catch(e){
        console.log("There was a problem while communicating with the Server.\nSee deleteFeedFunction() function.");
        return;
    }
    // If the result object exists but has errors it means we made a bad request, than we have to stop the execution and signal it to the developer in the console
    if(result.error){
        console.log("Bad request.\nPlease review the deleteFeedFunction() function.");
        return;
    }

    var formDigest = result.d.GetContextWebInformation.FormDigestValue;
    var threadId = $('#tempID').html(); // reading the id of the message to delete

    // Starting the request for the deletion of the feed.
    // Deleting the feed any reply is discarded with it.
    var xhr = createCORSRequest("POST", myFeedManagerEndpoint + "post/Delete"); // Creating CORSRequest to Delete the message
    xhr.onload = function () {
        // After the operation the User has to see the feed disappear from the page
        var el = document.getElementById('feedsItem'+threadId);
        el.parentNode.removeChild(el);

        // var firstReply = $('#replyItem'+threadId+' 0');
        var firstReply = document.getElementById('replyItem'+threadId+' 0');

        if( firstReply !== null){
            var elReply;

            // If there is even only one reply to this feed we need to refresh the page to make the replies disappear
            for(var i=0; (document.getElementById('replyItem'+threadId+' '+i) !== null; i++){
                // We do not need to call the function "deleteReply()", because deleting the feed will also delete its replies.
                // We just need to delete the replies from the html page.
                elReply = document.getElementById('replyItem'+threadId+' '+i);
                elReply.parentNode.removeChild(elReply);
            }

        }

        // If no more feeds are displayed... (if even the first feed (feed[0]) has been deleted...)
        if (isEmpty($("#socialAPIFollowedFeeds"))) {
            $("#socialAPIFollowedFeeds").append('<div class="feedsItem"> <p id="text"> No feeds available </p> </div>');
        }

    };
    xhr.onerror = function (e1, e2, e3) {
        errorHandlerFunction(2, 'There has been an error while deleting the feed. Please try again later.');
```

```

}
// If the result object exists but has errors it means we made a bad request, than we have to stop the execution and signal it to the developer in the console
log
if(result.error){
    console.log("Bad request.\nPlease review the deleteReplyFunction() function.");
    return;
}

var formDigest = result.d.GetContextWebInformation.FormDigestValue;
var replyId = $('#tempReplyIndex').html(); // reading the index of the reply on Social

var xhr = createCORSRequest("POST", myFeedManagerEndpoint + "post/Delete"); // Creating CORSRequest to Delete the message
xhr.onload = function () {
    // After the operation the User has to see the feed disappear from the page
    manuallyUpdateAllTheFeeds(); // Updating all the feeds displayed
};
xhr.onerror = function (e1, e2, e3) {
    errorHandlerFunction(3, "There has been an error while deleting the reply.");
};
xhr.withCredentials = true;

xhr.setRequestHeader("X-RequestDigest", formDigest);
xhr.setRequestHeader("content-type", "application/json; charset=utf-8; odata=verbose");

var data = "{ 'ID':'"+replyId+" }"; // Including the ID of the feed we want to 'Delete'

xhr.send(data); // Sending the 'Delete' request
}

/* Function that realizes the unfollow operation.
 * If a User wants to delete a feed posted from another User he has to "unfollow" that User. Otherwise the feed will remain as is.
 */
function unfollowPerson(temp){

    var temp2 = temp.split("- ");

    // Reading the actor name of the followed User and its nickname on Social
    var actorName = temp2[0];
    var actorId = temp2[1];
    var whereToWrite = temp2[3];
    var updateInterval = temp2[4];
    temp = temp2[temp2.length-1].split("\\"); // 'temp' should become from "cern\\name" an array like ["cern,name"]
    var accountName = temp[temp.length-1];

    // Is the User sure?
    var x = confirm("Would you like to stop following " + actorName + " and no longer receive this person's updates in your feed?");
    if(x == false){ return; } // If the User clicks on 'No' then -> do nothing; else: continue

    // Saving variables in invisible fields into the html page
    $('#accountName').html(accountName); // Memorizing the account's name into an invisible html field
    $('#tempID').html(actorId); // Memorizing the account's id into an invisible html field
    $('#feedsSectionName').html(whereToWrite);
    $('#messageToUpload').html(updateInterval);

    // Calling the Server to get the formDigest and then calling the function 'unfollowPersonFunction()' to call the "unfollow" operation.
    executeRestCallExtendedFour(formDigestUrl, 'POST', null, unfollowPersonFunction, onError);
}

function unfollowPersonFunction(data){
    try{
        var result = JSON.parse(data); // parsing the data obtained from the social network
    }catch(e){
        console.log("There was a problem while communicating with the Server.\nSee unfollowPersonFunction() function.");
        return;
    }
    // If the result object exists but has errors it means we made a bad request, than we have to stop the execution and signal it to the developer in the console
    log
    if(result.error){
        console.log("Bad request.\nPlease review the unfollowPersonFunction() function.");
        return;
    }

    var formDigest = result.d.GetContextWebInformation.FormDigestValue;

    var actorId = $('#tempID').html();
    var accountName = $('#accountName').html();
    var whereToWrite = $('#feedsSectionName').html();
    var updateInterval = $('#messageToUpload').html();

    /* This is the main code for unfollowing a User. It tells:
     * - the domain/username attributes of the User to unfollow
     * - the id of the User to unfollow
     */
    var xhr = createCORSRequest("POST", apiEndpoint + "social.following/stopfollowing(ACTORType=0,AccountName='cern\\"+ accountName +"',Id='"+actorId+"')");
    // Creating CORSRequest to Stop Following the Actor
    // ActorType 0 is for the Users, 1 is for the Documents, 2 for sites and 3 for Tags.

    xhr.onload = function () {
        // After the operation the User has to see the Actor's feeds disappear from the page.
        // We refresh the feeds to hide the ones from the Actor that the User has just stopped following:
        updateFollowedFeeds(whereToWrite, updateInterval); // updating the followed feeds
    };
    xhr.onerror = function (e1, e2, e3) {
        errorHandlerFunction(4, "'Stop follow' operation error. Please try again later.\n\nIf the problem persists for more than 24 hours please contact
the IT Services.");
    };
    xhr.withCredentials = true;

    xhr.setRequestHeader("X-RequestDigest", formDigest);
    xhr.setRequestHeader("content-type", "application/json; charset=utf-8; odata=verbose");
}

```

```

        var data = null;

        xhr.send(data);          // Sending the 'Stop Following' request
    }

/* Function that implements the mechanism of 'Like' (about feeds)
 * Input - one string containing:
 * - the ID of the feed
 * - the HTML section in which the feeds are displayed.
 */
function likeFeedFunction(idwhereToWrite) {
    var id = idwhereToWrite.split(' ')[0];
    var whereToWrite = idwhereToWrite.split(' ')[1];
    executeRestCallExtended(formDigestUrl, 'POST', null, likeFeedBodyFunction, onError, id, whereToWrite);
}
// This function shows that the pressing of the 'Like' button has been handled successfully and changes the code to show the 'Unlike' button (with its associated onclick
function)
function likeFeedBodyFunction(data, id, whereToWrite){
    try{
        var result = JSON.parse(data);          // parsing the data obtained from the social network
    }catch(e){
        console.log("There was a problem while communicating with the Server.\nSee likeFeedBodyFunction() function.");
        return;
    }
    // If the result object exists but has errors it means we made a bad request, than we have to stop the execution and signal it to the developer in the console
log
    if(result.error){
        try{
            if(result.error.message.value.indexOf("Internal error code: 83") > -1){ // The User does not exists on Social
                alert("We couldn't get data from Social. Please visit https://social.cern.ch to set up your profile first.");
            }else{
                // Other error
                console.log("Bad request.\nPlease review the likeFeedBodyFunction() function.");
            }
            return;
        }catch(e){console.log("Exception thrown in function likeFeedBodyFunction()"); return;}
    }

    $(whereToWrite).html(''); // Emptying the HTML section to force the update...() function to re-display every feed (with the new .

    var formDigest = result.d.GetContextWebInformation.FormDigestValue;

    var xhr = createCORSRequest("POST", myFeedManagerEndpoint + "post/Like"); // Creating CORSRequest to Like the feed
    xhr.onload = function () {
        // If the code reaches this part the operation was a success... and the message has been liked.
        manuallyUpdateAllTheFeeds(); // and we call the manual update of the feeds to let the User see the new ones.
    };
    xhr.onerror = function (e1, e2, e3) {
        errorHandlerFunction(5, 'There has been an error while trying to like the feed. \nPlease try again later.');
```

```

        manuallyUpdateAllTheFeeds(); // and we call the manual update of the feeds to let the User see the new ones.
    };
    xhr.onerror = function (e1, e2, e3) {
        errorHandlerFunction(6, "There has been an error while trying to unlike the feed. \nPlease try again later");
    };
    xhr.withCredentials = true;

    xhr.setRequestHeader("X-RequestDigest", formDigest);
    xhr.setRequestHeader("content-type", "application/json; charset=utf-8; odata=verbose");

    var data = "{ 'ID':'"+id+"' }"; // Including the ID of the feed we want to 'Like'

    xhr.send(data); // Sending the 'Like'
}

// This function executes a 'Like' to a reply of a post
function likeReplyFunction(inputString){
    var id = '';
    for(var i=0; i<inputString.split(' ').length-1; i++){
        id += inputString.split(' ')[i] + ' ';
    }
    id = id.substring(0,id.length-1); // Taking away the extra space at the end of the string
    var whereToWrite = inputString.split(' ')[inputString.split(' ').length-1];

    try{
        // Let's find the ID of the reply
        var replyId = document.getElementById("replyId"+id).innerHTML;
    }catch(e){ replyId = id; }

    executeRestCallExtended(formDigestUrl, 'POST', null, likeReplyBodyFunction, onError, replyId, whereToWrite);
}
// Function that executes after likeReplyFunction(). Launched from executeRestCall(formDigestUrl, 'POST', null, likeReplyBodyFunction, onError); if the operation is successful.
function likeReplyBodyFunction(data, id, whereToWrite){
    try{
        var result = JSON.parse(data); // parsing the data obtained from the social network
    }catch(e){
        console.log("There was a problem while communicating with the Server.\nSee likeReplyBodyFunction() function.");
        return;
    }
    // If the result object exists but has errors it means we made a bad request, than we have to stop the execution and signal it to the developer in the console
    if(result.error){
        try{
            if(result.error.message.value.indexOf("Internal error code: 83") > -1){ // The User does not exists on Social
                alert("We couldn't get data from Social. Please visit https://social.cern.ch to set up your profile first.");
            }else{ // Other error
                console.log("Bad request.\nPlease review the likeFeedBodyFunction() function.");
            }
            return;
        }catch(e){console.log("Bad request.\nPlease review the likeReplyBodyFunction() function."); return;}
    }

    $(whereToWrite).html('');

    var formDigest = result.d.GetContextWebInformation.FormDigestValue;

    var xhr = createCORSRequest("POST", myFeedManagerEndpoint + "post/Like"); // Creating CORSRequest to Like the feed
    xhr.onload = function () {
        manuallyUpdateAllTheFeeds();
    };
    xhr.onerror = function (e1, e2, e3) {
        errorHandlerFunction(7, 'Error while trying to like a feed.');
```

```

log
// If the result object exists but has errors it means we made a bad request, than we have to stop the execution and signal it to the developer in the console
if(result.error){
    try{
        if(result.error.message.value.indexOf("Internal error code: 83") > -1){ // The User does not exists on Social
            alert("We couldn't get data from Social. Please visit https://social.cern.ch to set up your profile first.");
        }else{ // Other error
            console.log("Bad request.\nPlease review the likeFeedBodyFunction() function.");
        }
        return;
    }catch(e){console.log("Bad request.\nPlease review the unlikeReplyBodyFunction() function."); return;}
}

$(whereToWrite).html('');

var formDigest = result.d.GetContextWebInformation.FormDigestValue;

var xhr = createCORSRequest("POST", myFeedManagerEndpoint + "post/Unlike"); // Creating CORSRequest to Unlike the feed
xhr.onload = function () {
    manuallyUpdateAllTheFeeds();
};
xhr.onerror = function (e1, e2, e3) {
    errorHandlerFunction(8, 'Error while trying to unlike a reply.');
```

```

var fakeTokensArePresent = message.match(/\{[0-9]+\}/); // The '+' means that we are considering only positive integers.
if( fakeTokensArePresent != null ){
    message = message.replace(/\{/g,"{ ");
    message = message.replace(/\}/g," }");
}

// Using more than one curly bracket per time can bring problems while composing the message for Sharepoint.
// The simple solution uses a space between every couple of brackets. Now the brackets do not bother the tags and do not bother while composing the message.
if( message.match("{}") ){
    message = message.replace(/\{\}/g," { ");
}
if( message.match("}") ){
    message = message.replace(/\}\}/g," } ");
}

/*****
 * Looking for websites and tags inside the message
 *****/
var sitesAndTagsArray = new Array();
var tempTag = "";
var dataItemNumber=0; // number of read tags used for the tokens to apply in the message
var tempToken;

/* Checking the existence of links to websites inside the text of the message. */
// Now we try to find possible URL links inside the text.
var parts = message.split(" "); // Splitting the message using the spaces ( URLs don't have spaces )
var i=0;

for( var x=0; x<parts.length; x++ ){
    var afterUrl = '', beforeUrl = '';
    while(parts[x][0] === '(' && parts[x].length > 2){
        beforeUrl += '(';
        parts[x] = parts[x].substring(1);
    }
    while(parts[x][parts[x].length-1] === ')' && parts[x].length > 2){
        afterUrl += ')';
        parts[x] = parts[x].substring(0, parts[x].length-1);
    }

    if ( validateURL(parts[x]) ){
        sitesAndTagsArray.push(new socialDataItemObj(parts[x], 4));
        tempToken = "["+dataItemNumber+"]";
        parts[x] = beforeUrl + tempToken + afterUrl; // Replacing the website string into the message with a token like "{0}" or "{1}"

        dataItemNumber++; // increasing the number of read tags.
        continue;
    }
    // else: it is not a URL, so we can check for tags

    i=0;
    while( i < parts[x].length ){ // It HAS TO recalculate the length everytime because it could happen that two or more tags are written one after
another without spacing.

        // Making sure that it is a tag (#something) and it is not the HTML code for e.g. curly brackets ("&#123;" and "&#125;")
        if(parts[x][i] === '#' && isOnlyLetterOrNumber(parts[x][i+1]) && ( parts[x][i+1]!=='1' && parts[x][i+2]!=='2' &&
parts[x][i+3]!=='3' && parts[x][i+4]!=='4') && ( parts[x][i+1]!=='1' && parts[x][i+2]!=='2' && parts[x][i+3]!=='3' && parts[x][i+4]!=='4') && (i+1) < parts[x].length){
            // The first element is a '#'
            tempTag += parts[x][i];
            i++; // moving on

            // From now on only letters and number will be accepted as part of the tag

            // Reading the tag
            while( i < parts[x].length && isOnlyLetterOrNumber(parts[x][i]) ){

                tempTag += parts[x][i]; // copying the i-th character of the message into "tempTag"
                i++;
            }

            sitesAndTagsArray.push(new socialDataItemObj(tempTag, 3)); // Copying the tag into 'sitesAndTagsArray'
            tempToken = "["+dataItemNumber+"]";
            parts[x] = parts[x].replace(tempTag, tempToken); // Replacing the tag string into the message with a
token like "{0}" or "{1}"

            if(dataItemNumber < 10){
                // if there are less than 10 tags...
                i = i - tempTag.length + 2; // Since the message has been modified we have to move
the cursor according to the new string to continue examining the text from the right point.
            }else{
                if(dataItemNumber < 100){
                    // if there are less than 100 tags but more than 9...
                    i = i - tempTag.length + 3;
                }
                else{
                    // There are more than 99 tags? Maybe there is a problem. Stopping the execution.
                    return;
                }
            }

            tempTag = ""; // Resetting 'tempTag'
            dataItemNumber++; // increasing the number of read tags.
        }

        i++;
    }
}

/*****
 * Re-assembling the message
 *****/
message = "";
for( x=0; x<parts.length; x++){
    message += parts[x] + ' ';
}

```

```

}

// Converting all the single quotes (') and backslashes (\) in the message adding and extra backslash to each char to let Javascript to read them correctly.
// This is done because otherwise there would be a problem during the creation of the post in the 'try' section a few rows below this line.
message = message.replace(/\\/g, "\\"); // To be able to post single backslashes we have to double each one of them
message = message.replace(/'/g, "\\'"); // "\\'" is the right replacement to be able to post single quotes

if( sitesAndTagsArray.length > 0 ){
    // We will now create the string to put inside the 'ContentItems' section of the data inside the CORSRequest
    var contentItemsString = '{ "results": [ { "__metadata": { "type": "SP.Social.SocialDataItem" }, "Text": ""';
    i=0;
    while( i<sitesAndTagsArray.length ){
        // if we are analysing the last tag...
        if( i === sitesAndTagsArray.length-1 ){
            contentItemsString += sitesAndTagsArray[i].value;
            if(sitesAndTagsArray[i].itemType == 4){
                contentItemsString += ", \"Uri\": \""+sitesAndTagsArray[i].value;
            }
            contentItemsString += ", \"ItemType\": "+ sitesAndTagsArray[i].itemType + ' ] ]';
        }else{
            contentItemsString += sitesAndTagsArray[i].value;
            if(sitesAndTagsArray[i].itemType == 4){
                contentItemsString += ", \"Uri\": \""+sitesAndTagsArray[i].value;
            }
            contentItemsString += ", \"ItemType\": "+ sitesAndTagsArray[i].itemType + ' }, { "__metadata": { "type":
"SP.Social.SocialDataItem" }, "Text": ""';
        }
        i++;
    }

    // Now the message is well-formed.
    // We try to post it.
    try
    {
        var xhr = createCORSRequest("POST", myFeedManagerEndpoint + "my/Feed/Post");
        xhr.onload = function () {
            if(this.status == 200){ // If the operation succeeds... than the feed has been uploaded.
                if(inputFunction != undefined && inputFunction != null){
                    inputFunction(true);
                }

                updateFollowedFeeds(followedFeedsWhereToWrite, followedFeedsUpdateInterval); //

            }else{ // We sent the request correctly but there has been a problem
                var response = this.responseText;
                if(response.indexOf("Internal error code: 83")){ // If the response from the Server says
                    alert("We couldn't get data from Social. Please visit https://social.cern.ch to set
up your profile first.");
                }else{ // Generic error
                    if(inputFunction != undefined && inputFunction != null){
                        inputFunction(false);
                    }
                }
            }
        };
        xhr.onerror = function (e1, e2, e3) {
            errorHandlerFunction(9, 'There has been an error while uploading the message. \nPlease try again later.');
```

```

        xhr.onload = function () {
            if(this.status == 200){ // If the operation succeeds... then the feed has been uploaded.
                if(inputFunction != undefined && inputFunction != null){
                    inputFunction(true);
                }
            }

            updateFollowedFeeds(followedFeedsWhereToWrite, followedFeedsUpdateInterval); //

...and we call the manual update of the feeds (to show the new one in the webpage)
        }else{ // We sent the request correctly but there has been a problem
            var response = this.responseText;
            if(response.indexOf("Internal error code: 83")){ // If the response from the Server says
                alert("We couldn't get data from Social. Please visit https://social.cern.ch to set
up your profile first.");
            }
        }else{ // Generic error
            if(inputFunction != undefined && inputFunction != null){
                inputFunction(false);
            }
        }
    }
};
xhr.onerror = function (e1, e2, e3) {
    errorHandlerFunction(9, 'There has been an error while uploading the message. \nPlease try again later.');
```

that the problem is that the user has not been found...

up your profile first.");

```

    if(inputFunction != undefined && inputFunction != null){
        inputFunction(false);
    }
}
};
xhr.withCredentials = true;

xhr.setRequestHeader("X-RequestDigest", formDigest);
xhr.setRequestHeader("content-type", "application/json; charset=utf-8; odata=verbose");
// Creating the data for the post
var data = " { 'restCreationData':{ " +
    "
    'creationData':{ " +
    "
    'ID': null, " +
    "
    'restCreationData':{ " +
    "
    'ContentText':'" +message+
    "
    " } " +
    " } }";

xhr.send(data); // Uploads the message

}
catch(err)
{
    errorHandlerFunction(9, 'There has been an error while uploading the message. \nPlease try again later.');
```

'_metadata':{'type':'SP.Social.SocialPostCreationData' }, " +

'UpdateStatusText':false " +

```

    if(inputFunction != undefined && inputFunction != null){
        inputFunction(false);
    }
}
return;
}
}
}

function socialDataItemObj(value, itemType){
    this.value = value;
    this.itemType = itemType;
}

/* This function deletes the text into the inner html section
 * Input:
 * - id = the Id of the section (it is used to to find the text to delete)
 */
function clearMessageToTheUser(id){
    try{
        document.getElementById(id).innerHTML = "";
    }
    catch(e) { }
}

/* This function checks if the passed element is a letter or a number.
 * If so, it returns true, false otherwise.
 * Input:
 * - temp = a symbol
 */
function isOnlyLetterOrNumber(temp){
    // Checking the ASCII character using the ASCII table at http://www.asciitable.com/
    if( (asc(temp) >= 48 && asc(temp) <= 57) || (asc(temp) >= 65 && asc(temp) <= 90) || (asc(temp) >= 97 && asc(temp) <= 122) || (asc(temp) >= 128 && asc(temp) <=
255) ){
        return true;
    }else{
        return false;
    }
}

// This function returns the corresponding number of the input symbol in the ASCII table.
function asc(String){
    return String.charCodeAt(0);
}

/* This function shows the Reply textbox that will be used to send a reply to a feed.
 * Input:
 * - inputString: one string containing two sub-strings separated using a space character.
 * First there is the name of the section where the feed is written.
```



```

clearInterval(timer); // Terminates itself
    }
    },1);
}

/* Function that clears the text of the reply that the User is writing (in case the User would like to empty the textbox and re-write the reply)
 * Input:
 * - inputString: one string containing two sub-strings separated using a space character.
 *           First there is the name of the section where the feed is written.
 *           Next the id of the feed considered.
 */
function clearReplyText(inputString){
    // Reading the input
    var parentSection = inputString.split(' ')[0]; // the id of the section in which the feeds are displayed.
    if(parentSection[0] === '#'){
        parentSection = parentSection.substring(1, parentSection.length); // Eliminating the '#' at the beginning. This is necessary for the
'getElementInsideContainer()' function
    }
    var whereToWrite = inputString.split(' ')[1]; // the id of the parent section where to find the threadId passed in input (see next line);
    if(whereToWrite[0] === '#'){
        whereToWrite = whereToWrite.substring(1, whereToWrite.length); // Eliminating the '#' at the beginning. This is necessary for the
'getElementInsideContainer()' function
    }
    var id = inputString.split(' ')[2]; // The 'threadId'

    var objText = getElementInsideContainer(parentSection, "textareaReply" + parentSection + id); // the textbox for the reply
    objText.value = "";
}

// This function creates a Reply
function createReply(inputString) {
    // Reading the input
    var parentSection = inputString.split(' ')[0]; // the id of the section in which the feeds are displayed.
    if(parentSection[0] === '#'){
        parentSection = parentSection.substring( 1 ); // Eliminating the '#' at the beginning. This is necessary for the
'getElementInsideContainer()' function
    }
    var whereToWrite = inputString.split(' ')[1]; // the id of the parent section where to find the threadId passed in input (see next line);
    if(whereToWrite[0] === '#'){
        whereToWrite = whereToWrite.substring( 1 ); // Eliminating the '#' at the beginning. This is necessary for the 'getElementInsideContainer()'
function
    }
    var threadId = inputString.split(' ')[2];

    // Reading the text of the Reply
    var text = getElementInsideContainer(whereToWrite, "textareaReply"+ parentSection + threadId).value; // Getting the text
    if(text === null || text === "" || text === undefined){ // consistency check
        errorHandlerFunction(1, "No text present. Please write some text first.");
        return;
    }

    $('#'+whereToWrite).html(''); // Clearing the HTML section to force the update...() function to re-display every feed and its replies.

    /* TO BE IMPLEMENTED: (sending an image with the message)
    var attachmentUri = getElementInsideContainer(parentSection, "replyButtonUploadFile"+threadId).value; // Getting the URI of the attachment
    if(attachmentUri !== null || attachmentUri !== undefined){
        $('#tempTagText').html(attachmentUri); // Copying the URI of the attachment into the invisible p section with
'id=messageToUpload' in the HTML file
    }
    else
    {
        $('#tempTagText').html(""); // Setting the attachment to empty string
    }
    */

    // We are now ready to post the reply... calling the executeRestCall() function.
    executeRestCallExtendedFive(formDigestUri, "POST", null, postReply, onError, text, threadId, parentSection, whereToWrite);

    // The algorithm continues in the postReply() function
}

/* Function that is demanded to post a Reply message on the Social Network.
 * The message is read from the <p id="messageToUpload"> section from the html file.
 */
function postReply(data, message, postId, parentSection, whereToWrite) {
    try{
        var result = JSON.parse(data); // parsing the data obtained from the social network
    }catch(e){
        console.log("There was a problem while communicating with the Server.\nSee postReply() function.");
        return;
    }
    // If the result object exists but has errors it means we made a bad request, than we have to stop the execution and signal it to the developer in the console
log
    if(result.error){
        try{
            if(result.error.message.value.indexOf("Internal error code: 83") > -1){ // The User does not exists on Social
                alert("We couldn't get data from Social. Please visit https://social.cern.ch to set up your profile first.");
            }else{ // Other error
                console.log("Bad request.\nPlease review the postReply() function.");
            }
            return;
        }catch(e){console.log("Exception thrown in function postMessage()"); return;}
    }
    var formDigest = result.d.GetContextWebInformation.FormDigestValue;

    // TO BE IMPLEMENTED: var attachmentUri = $('#tempTagText').html(); // and the URI of the attachment
    var attachmentUri = '';

```

```

// Since the tags are written as tokens like {0} or {1} we have to make any kind of text which has
// numbers (only numbers) between curly parentheses be coded in html to avoid unintended repetitions
// of tags in the post.
var fakeTokensArePresent = message.match(/\{[0-9]+\}/); // The '+' means that we are considering only positive integers.
if( fakeTokensArePresent != null ){
    message = message.replace(/\{/g, "{");
    message = message.replace(/\}/g, "}");
}

// Using more than one curly bracket per time can bring problems while composing the message for Sharepoint.
// The simple solution uses a space between every couple of brackets. Now the brackets do not bother the tags and do not bother while composing the message.
if( message.match("{") ){
    message = message.replace(/\{/g, "{ ");
}
if( message.match("}") ){
    message = message.replace(/\}/g, " }");
}

/*****
 * Looking for tags and websites inside the message
 *****/
var sitesAndTagsArray = new Array();
var tempTag = "";
var dataItemNumber=0; // number of read tags used for the tokens to apply in the message
var tempToken;
/* Checking the existence of links to websites inside the text of the message. */
// Now we try to find possible URL links inside the text.
var parts = message.split(" "); // Splitting the message using the spaces ( URLs don't have spaces )
var i=0;

for( var x=0; x<parts.length; x++ ){
    var afterUrl = '', beforeUrl = '';
    while(parts[x][0] === '{' && parts[x].length > 2){
        beforeUrl += '{';
        parts[x] = parts[x].substring(1);
    }
    while(parts[x][parts[x].length-1] === '}' && parts[x].length > 2){
        afterUrl += '}';
        parts[x] = parts[x].substring(0, parts[x].length-1);
    }

    if ( validateURL(parts[x]) ){
        sitesAndTagsArray.push(new socialDataItemObj(parts[x], 4));
        tempToken = ("+"+dataItemNumber+"");
        parts[x] = beforeUrl + tempToken + afterUrl; // Replacing the website string into the message with a token like "{0}" or "{1}"

        dataItemNumber++; // increasing the number of read tags.
        continue;
    }
    // else: it is not a URL, so we can check for tags

    i=0;
    while( i < parts[x].length ){ // It HAS TO recalculate the length everytime because it could happen that two or more tags are written one after
another without spacing.

        // Making sure that it is a tag (#something) and it is not the HTML code for e.g. curly brackets ("##123;" and "#125;")
        if(parts[x][i] === '#' && isOnlyLetterOrNumber(parts[x][i+1]) && ( parts[x][i+1]!=='1' && parts[x][i+2]!=='2' &&
parts[x][i+3]!=='3' && parts[x][i+4]!=='4') && ( parts[x][i+1]!=='1' && parts[x][i+2]!=='2' && parts[x][i+3]!=='5' && parts[x][i+4]!=='6') && (i+1) < parts[x].length){
            // The first element is a '#'
            tempTag += parts[x][i];
            i++; // moving on

            // From now on only letters and number will be accepted as part of the tag

            // Reading the tag
            while( i < parts[x].length && isOnlyLetterOrNumber(parts[x][i]) ){

                tempTag += parts[x][i]; // copying the i-th character of the message into "tempTag"
                i++;
            }

            sitesAndTagsArray.push(new socialDataItemObj(tempTag, 3)); // Copying the tag into 'sitesAndTagsArray'
            tempToken = ("+"+dataItemNumber+"");
            parts[x] = parts[x].replace(tempTag, tempToken); // Replacing the tag string into the message with a
token like "{0}" or "{1}"

            if(dataItemNumber < 10){
                // if there are less than 10 tags...
                i = i - tempTag.length + 2; // Since the message has been modified we have to move
the cursor according to the new string to continue examining the text from the right point.
            }else{
                if(dataItemNumber < 100){
                    // if there are less than 100 tags but more than 9...
                    i = i - tempTag.length + 3;
                }
                else{
                    // There are more than 99 tags? Maybe there is a problem. Stopping the execution.
                    return;
                }
            }

            tempTag = ""; // Resetting 'tempTag'
            dataItemNumber++; // increasing the number of read tags.
        }
        i++;
    }
}

/*****
 * Re-assembling the message
 *****/

```



```

'UpdateStatusText':false " +
                                                                    "
                                                                    'ContentText':" + message + "',
                                                                    "
                                                                    } " +
                                                                    " }";
                                                                    " }";

                                                                    xhr.send(data); // Uploads the message
                                                                    }
                                                                    catch(err)
                                                                    {
                                                                    {
                                                                    errorHandlerFunction(10, "There has been an error while uploading the reply. \nPlease try again later.");
                                                                    return;
                                                                    }
                                                                    }
                                                                    }
                                                                    else
                                                                    {
                                                                    /*.....
                                                                    * else: there are some TAGS and/or WEBSITES but no ATTACHMENT
                                                                    *.....*/

                                                                    // Trying to post the message
                                                                    try
                                                                    {
                                                                    var xhr = createCORSRequest("POST", myFeedManagerEndpoint + "post/Reply");
                                                                    xhr.onload = function(){
                                                                    // If the code reaches this part the operation was a success... and the message has been sent.
                                                                    if(this.status == 200){ // If the operation succeeds... than the feed has been uploaded.
                                                                    if(wheretowrite == "socialAPISingleFeed"){ // If it is a conversation we update
                                                                    // Looking for the section in the global array to retrieve the URL to
                                                                    var length = globalArrayOfSingleConversations.length;
                                                                    var tempElement;
                                                                    for(var i=0; i<length; i++){
                                                                    if(globalArrayOfSingleConversations[i].sectionID ==
                                                                    tempElement
                                                                    -
                                                                    globalArrayOfSingleConversations.splice(i, 1); // Removing that element from the global array (because it will be re-inserted in the updateSingleFeed() function)
                                                                    updateSingleFeed(tempElement[0].sectionID, tempElement[0].URL);
                                                                    }
                                                                    }
                                                                    }else{
                                                                    manuallyUpdateAllTheFeeds(); // and we call the manual update of the
                                                                    feeds to let the User see the new ones.
                                                                    }
                                                                    }else{
                                                                    // We sent the request correctly but there has been a problem
                                                                    var response = this.responseText;
                                                                    if(response.indexOf("Internal error code: 83")){ // If the response from
                                                                    the Server says that the problem is that the user has not been found...
                                                                    alert("We couldn't get data from Social. Please visit
                                                                    https://social.cern.ch to set up your profile first.");
                                                                    }
                                                                    }else{
                                                                    // Generic error
                                                                    alert("There has been a problem while posting the reply. Please try
                                                                    again later.");
                                                                    }
                                                                    }
                                                                    };
                                                                    xhr.onerror = function (e1, e2, e3) {
                                                                    errorHandlerFunction(10, 'There has been an error while uploading the reply. \nPlease try again
                                                                    later.');
```

```

        xhr.onload = function () {
            // If the code reaches this part the operation was a success... and the message has been sent.
            if(this.status == 200){ // If the operation succeeds... than the feed has been uploaded.
                if(wheretowrite == "socialAPISingleFeed"){ // If it is a conversation we update
                    // Looking for the section in the global array to retrieve the URL to
                    var length = globalArrayOfSingleConversations.length;
                    var tempElement;
                    for(var i=0; i<length; i++){
                        if(globalArrayOfSingleConversations[i].sectionID ==
                            tempElement
                        ){
                            globalArrayOfSingleConversations.splice(i, 1); // Removing that element from the global array (because it will be re-inserted in the updateSingleFeed() function)
                            updateSingleFeed(tempElement[0].sectionID, tempElement[0].URL);
                        }
                    }
                }else{
                    manuallyUpdateAllTheFeeds(); // and we call the manual update of the
                    feeds to let the User see the new ones.
                }
            }else{
                // We sent the request correctly but there has been a problem
                var response = this.responseText;
                if(response.indexOf("Internal error code: 83")){ // If the response from
                    the Server says that the problem is that the user has not been found...
                    alert("We couldn't get data from Social. Please visit
                    https://social.cern.ch to set up your profile first.");
                }else{
                    // Generic error
                    alert("There has been a problem while posting the reply. Please try
                    again later.");
                }
            }
        };
        xhr.onerror = function (e1, e2, e3) {
            errorHandlerFunction(10, "There has been an error while uploading the reply. \nPlease try again
            later.");
        };
        xhr.withCredentials = true;
        xhr.setRequestHeader("X-RequestDigest", formDigest);
        xhr.setRequestHeader("content-type", "application/json; charset=utf-8; odata=verbose");
        // execute post
        var data = " { 'restCreationData':{ '__metadata':{ 'type':'SP.Social.SocialRestPostCreationData'}, " +
            "'ID':'" + postId + "',"+
            " 'creationData':{
                'Attachment':
                'ContentText':'" +
            } " +
            " } }";
        xhr.send(data); // Sending the Reply
    }catch(err)
    {
        errorHandlerFunction(10, "There has been an error while uploading the reply. \nPlease try again later.");
        return;
    }
    }
    else
    {
        /*****
        * else: there are no TAGS & no ATTACHMENT
        *****/
        // Trying to post the message.
        try
        {
            var xhr = createCORSRequest("POST", myFeedManagerEndpoint + 'post/Reply');
            xhr.onload = function () {
                // If the code reaches this part the operation was a success... and the message has been sent.
                if(this.status == 200){ // If the operation succeeds... than the feed has been uploaded.
                    if(wheretowrite == "socialAPISingleFeed"){ // If it is a conversation we update
                        // Looking for the section in the global array to retrieve the URL to
                        var length = globalArrayOfSingleConversations.length;
                        var tempElement;
                        for(var i=0; i<length; i++){
                            if(globalArrayOfSingleConversations[i].sectionID ==
                                tempElement
                            ){
                                globalArrayOfSingleConversations.splice(i, 1); // Removing that element from the global array (because it will be re-inserted in the updateSingleFeed() function)
                                updateSingleFeed(tempElement[0].sectionID, tempElement[0].URL);
                            }
                        }
                    }else{
                        manuallyUpdateAllTheFeeds(); // and we call the manual update of the
                        feeds to let the User see the new ones.
                    }
                }else{
                    // We sent the request correctly but there has been a problem
                    var response = this.responseText;
                    if(response.indexOf("Internal error code: 83")){ // If the response from
                        the Server says that the problem is that the user has not been found...
                        alert("We couldn't get data from Social. Please visit
                        https://social.cern.ch to set up your profile first.");
                    }else{
                        // Generic error
                        alert("There has been a problem while posting the reply. Please try
                        again later.");
                    }
                }
            }
        }
    }
}

```

```

    }
  };
  xhr.onerror = function (e1, e2, e3) {
    errorHandlerFunction(10, 'There has been an error while uploading the reply. \nPlease try again
later.');
```

```

  };
  xhr.withCredentials = true;

  xhr.setRequestHeader("X-RequestDigest", formDigest);
  xhr.setRequestHeader("content-type", "application/json; charset=utf-8; odata=verbose");
  // execute post
  var data = " { 'restCreationData': { '__metadata': { 'type': 'SP.Social.SocialRestPostCreationData' }, " +
    " 'ID': '" + postId + "'," +
    " 'creationData': {
    " 'ContentText': '" + message +
    " } " +
    " } }";

  xhr.send(data); // Sending the Reply

} catch (err)
{
  errorHandlerFunction(10, "There has been an error while uploading the reply. \nPlease try again later.");
  return;
}
}
}

// This function allows both the API and the User to manually update the feeds in the web page.
function manuallyUpdateAllTheFeeds() {
  var tempElement;
  var i=0;

  // Every REST call is made only if the correspondent section exists in the HTML code.
  if( document.getElementById(followedFeedsWhereToWrite.substring(1, followedFeedsWhereToWrite.length)) != null ){
    clearInterval(followedFeedsUpdatesHandler);
    updateFollowedFeeds(followedFeedsWhereToWrite, followedFeedsUpdateInterval, followedFeedsNumFeeds, followedFeedsFlagDisplayReplies);
  }

  // For each profile that has to be read we retrieve the data from the globalArrayOfProfiles...
  var tempGlobalArray = new Array();
  var length = globalArrayOfProfiles.length;
  for(i=0; i<length; i++){
    tempElement = globalArrayOfProfiles.pop();
    if(tempElement.sectionID[0] === '#'){ tempElement.sectionID = tempElement.sectionID.substring( 1 ); }
    tempGlobalArray.push(tempElement); // Saving the globalArrayOfProfiles in a temporary array to prevent 'race conditions' that could
happen while updating one feed and trying to read the next one.
  }
  // ...and we use those data to call for an update.
  for(i=0; i<length; i++){
    tempElement = tempGlobalArray.pop();
    if( document.getElementById(tempElement.sectionID) != null ){
      try{
        if(tempElement.automaticUpdatesHandlersCode != null){
          clearInterval(tempElement.automaticUpdatesHandlersCode);
        }
        updateFeedsFromProfile(tempElement.keyValue, tempElement.sectionID, tempElement.timeInterval,
tempElement.numOfFeeds, tempElement.flagDisplayReplies);
      } catch(e) { console.log('There has been a problem updating the feeds of the account: ' + tempElement.keyValue); }
    }
  }

  // For each hashtag that has to be used to retrieve feeds with the same hashtag we retrieve the data from the globalArrayOfHashtags.
  tempGlobalArray = new Array();
  length = globalArrayOfHashtags.length;
  for(i=0; i<length; i++){
    tempElement = globalArrayOfHashtags.pop();
    if(tempElement.sectionID[0] === '#'){ tempElement.sectionID = tempElement.sectionID.substring( 1 ); }
    tempGlobalArray.push(tempElement); // Saving the globalArrayOfHashtags in a temporary array to prevent 'race conditions' that could
happen while updating one feed and trying to read the next one.
  }
  // ...and we use those data to call for an update.
  for(i=0; i<length; i++){
    tempElement = tempGlobalArray.pop();
    if( document.getElementById(tempElement.sectionID) != null ){
      try{
        if(tempElement.automaticUpdatesHandlersCode != null){
          clearInterval(tempElement.automaticUpdatesHandlersCode);
        }
        updateFeedsWithSameHashtag(tempElement.keyValue, tempElement.sectionID, tempElement.timeInterval,
tempElement.numOfFeeds, tempElement.flagDisplayReplies);
      } catch(e) { console.log('There has been a problem updating the feeds containing the tag: ' + tempElement.keyValue); }
    }
  }

  // Updating every section with a Single Conversation
  tempGlobalArray = new Array();
  length = globalArrayOfSingleConversations.length;
  for(i=0; i<length; i++){
    tempElement = globalArrayOfSingleConversations.pop();
    if(tempElement.sectionID[0] === '#'){ tempElement.sectionID = tempElement.sectionID.substring( 1 ); }
    tempGlobalArray.push(tempElement); // Saving the globalArrayOfSingleConversations in a temporary array to prevent 'race conditions'
that could happen while updating one feed and trying to read the next one.
  }
  // ...and we use those data to call for an update.
  for(i=0; i<length; i++){
    tempElement = tempGlobalArray.pop();
    if( document.getElementById(tempElement.sectionID) != null ){
      try{

```

```

        updateSingleFeed(tempElement.sectionID, tempElement.URL);
    }catch(e){ console.log('There has been a problem updating the feeds containing the tag: ' + tempElement.keyValue); }
    }

    // The automatic update of the feeds is re-activated during the execution of the called functions.
}

// Function that retrieves the feeds with the tag written from the User in the textarea.
// (this has nothing to spare with the tags in the Canvas construct)
function findTaggedFeeds(tag, whereToWrite){
    // Retrieving tag's name
    // If no tag is given in input... [ the function is called as "findTaggedFeeds();" ]
    if( typeof(tag) !== "string" || tag === null || tag === "" || tag === undefined ){
        tag = document.getElementById("textareaRetrieveTags").value;

        if(tag === null || tag === undefined || tag === ""){ return; } // consistency check
    }
    // If no 'whereToWrite' section ID is given in input... [ the function is called as "findTaggedFeeds();" ]
    if( typeof(whereToWrite) !== "string" || whereToWrite === null || whereToWrite === undefined || whereToWrite === "" ){
        whereToWrite = '#feedsWithSameTag';
    }

    // if(tag[0] !== '#') { tag = '#' + tag; } // We need the text with the # symbol at the beginning.

    /* If necessary, to make these feeds automatically updated use this code:
    var tempIndex = checkPresenceOfElement(whereToWrite, globalArrayOfHashtags);
    if(tempIndex >= 0){
        globalArrayOfHashtags.push(new updateObj(tag, whereToWrite, null, 0, numOffFeeds, flagDisplayReplies)); // Adding a new item to the
    globalArrayOfHashtags, to let the automatic updates be know what to look for after tot seconds.
    } */

    retrieveFeedsWithSameTag(tag, whereToWrite);
}

/* Function that makes the REST call to retrieve the tags from Social that will be displayed in the Tag Cloud.
* Input:
* - whereToWrite: the ID of the HTML section in which the tags have to be displayed;
* - maxNumTags: the maximum number of tags to retrieve;
* - textColor: in the 3D Tag Cloud it is possible to set the color of the text (e.g.: '#3861aa');
* - textBorderColor: in the 3D Tag Cloud it is possible to set the color of border of the text that appears when the mouse is over the tag (e.g.: '#3861aa');
* - numDimensions: the number of dimensions to take into account. (2= 2D Tag Canvas, 3=3D Tag Cloud);
* - weightFlag: it is possible to set the size of the text of each tag accordingly to the frequency in which they are present in Social;
* - periodOffTime: the period of the time we are looking for ('lastDay', 'lastWeek', 'lastMonth', 'lastYear', 'allTime'). The case does not matter.
*/
function loadTagCloud(whereToWrite, maxNumTags, textColor, textBorderColor, numDimensions, weightFlag, periodOffTime){
    // Section checks. If the HTML vsections are presents in the webpage we can move on, otherwise the function has to stop.
    while(whereToWrite[0] !== '#' || whereToWrite.length > 0){
        whereToWrite = whereToWrite.substring(1);
    }
    if( document.getElementById(whereToWrite) === null ){
        // Error. No HTML section found to display the followed feeds on Social. Please add a <div id="+ whereToWrite +"> section.
        console.log('Error while trying to write the tags for the Tag Cloud. The HTML section appears not to exist. See the function loadTagCloud().');
        return;
    }

    var date = new Date(); // Reading today's date

    switch(periodOffTime){
        case 'lastDay':
            date.setDate(date.getDate()-1); // Going back one day
            break;
        case 'lastWeek':
            date.setDate(date.getDate()-7); // Going back one week
            break;
        case 'lastMonth':
            date.setDate(date.getDate()-30); // Going back one month
            break;
        case 'lastYear':
            date.setDate(date.getDate()-365); // Going back one year
            break;
        case 'allTime':
            date = null; // We will retrieve all the tags ever used (with their number of occurrences)
            break;
        default:
            date = null; // We will retrieve all the tags ever used (with their number of occurrences)
    }

    var querySiteToGetTheTags; // This will be the URL used to retrieve the tags from Social
    if(date === null){
        querySiteToGetTheTags = querySiteToGetAllTheTags;
    }
    else{
        var day, month;

        month = date.getMonth() + 1;
        if(month < 10){ month = '0' + month; } // We want the 'month' string to have always two chars.

        day = date.getDate();
        if(day < 10){ day = '0' + day; } // We want the 'day' string to have always two chars.

        querySiteToGetTheTags = searchRestService + "query?querytext='ContentTypeId:0x01FD' write=>'" + date.getFullYear() + "-" + month + "-" + day + "
00:00:01Z\" -ContentClass=urn:content-class:SPSPeople&refiners='Tags'";
    }

    executeRestCallExtendedSeven(querySiteToGetTheTags, 'GET', null, drawUserTagsInCanvas, onError, whereToWrite, maxNumTags, textColor, textBorderColor,
numDimensions, weightFlag);
// getting all the tags of Social and passing them to the function drawUserTagsInCanvas()

```



```

$(whereToWrite).append('<div class="socialAPIWrapClass" id="socialAPIWrapClassTagCloud" + whereToWrite.substring(1, whereToWrite.length) + ""></div>');
whereToWrite = "#socialAPIWrapClassTagCloud" + whereToWrite.substring(1, whereToWrite.length);

// If we want a 3-dimensional canvas...
if(numDimensions == 3){
    // Creating the canvas in the HTML code
    var canvasID = 'socialAPITagsCanvas' + whereToWrite.substring(1, whereToWrite.length);
    $(whereToWrite).append('<canvas id="'+ canvasID +' " height="'+ $(whereToWrite).width() + " width="'+ $(whereToWrite).width() + "'>' +
        '<p>' +
        'If you are reading
this, your browser does not support the canvas tag. ' +
        'Please try again using
a different browser.' +
        '</p>' +
        '</canvas>'); // This section is hidden if no tags
are retrieved.

/*
// To use the colours in the canvas uncomment this section and set 'both' for the "weightMode" property.
// Create Linear Gradient to apply colors to the tags related to their weights
canv = document.getElementById('socialAPITagsCanvas');
// To realize colored tags into the canvas we create a gradient that will be relate to the weights of each tag (the number of times the tag has
been used on Social)

var gradient = {
    0: '#f00', // red
    0.33: '#ff0', // yellow
    0.66: '#0f0', // green
    1: '#00f' // blue
};

// use getContext to use the canvas for drawing
var ctx = canv.getContext('2d');
var linearGrad = ctx.createLinearGradient(0,0,0,150);
linearGrad.addColorStop(0, '#fff000');
linearGrad.addColorStop(0.3, '#00ff00');
linearGrad.addColorStop(0.6, '#0000ff');
linearGrad.addColorStop(1, '#00f'); */

try {
    TagCanvas.interval = 20;
    TagCanvas.textFont = 'Impact,Arial Black,sans-serif';
    TagCanvas.textHeight = 25; // Height in pixels
    TagCanvas.outlineThickness = 2;
    if(textColor === null || textColor === undefined || textColor === ''){
        TagCanvas.textColour = '#3861aa';
    }else{
        TagCanvas.textColour = textColor;
    }

    if(textBorderColor === null || textBorderColor === undefined || textBorderColor === ''){
        TagCanvas.outlineColour = '#3861aa';
    }else{
        TagCanvas.outlineColour = textBorderColor;
    }

    TagCanvas.maxSpeed = 0.07;
    TagCanvas.minBrightness = 0.25;
    TagCanvas.depth = 0.8;
    TagCanvas.pulsateTo = 0.2;
    TagCanvas.pulsateTime = 0.75;
    TagCanvas.initial = [0.03,-0.03]; // Initial spin of the sphere
    TagCanvas.decel = 0.98; // Controls the deceleration when the mouse leaves the
canvas area

    TagCanvas.reverse = true; // Sets the way the mouse moves the sphere

    if(weightFlag){
        TagCanvas.weight = true;
        TagCanvas.weightFrom = 'data-weight';
        TagCanvas.weightMode = 'size'; // The weights are emphasized with the size of the
text (the greater the weight the bigger the size)

        // ! The options for 'SizeMin' and 'SizeMax' have to be both set to work.
        TagCanvas.weightSizeMin = 11;
        TagCanvas.weightSizeMax = 46;
    }

    TagCanvas.fadeIn = 800; // Let the canvas fade in when loaded

    TagCanvas.hideTags = true; // This function hides the tag elements from the webpage (same effect as display:none)

    TagCanvas.Start(canvasID, 'socialAPIWeightenedTags'+parentWhereToWrite.substring(1, parentWhereToWrite.length));
} catch(e) {
    // Something went wrong, showing the User an error message:
    document.getElementById(whereToWrite.substring(1, whereToWrite.length)).innerHTML = '<p>' +

    'If you are reading this, your browser does not support the canvas tag. ' +

    'Please try again using a different browser.' +

    '</p>';

    console.log('Error while setting the Tag Cloud 3D. Please debug function createTagsCanvas().');
}

} else{
    // In this case, it will be a 2-dimensional graphic (list-like).

    // Hiding the tags
    $('#socialAPIWeightenedTags'+parentWhereToWrite.substring(1, parentWhereToWrite.length)).hide();

    // Reading the tags
    var tagsForTagCloud2D = $('#socialAPIWeightenedTags'+parentWhereToWrite.substring(1, parentWhereToWrite.length));

```

```

try{
    tagsForTagCloud2D = tagsForTagCloud2D[0].childNodes;
} catch(e){
    $(whereToWrite).append('<p>There is a problem communicating with the Server. <br/>Please try again later. </p>');
    console.log('No readable tags found for Tag Cloud 2D. Please debug function createTagsCanvas().');
}

var weight = 1; // Variable containing the number of occurrences of the tag (the max number will be 10000, see function
drawUserTagsInCanvas(). Minimum font-size = 0.8em.

try{
    var length = tagsForTagCloud2D.length;
    if(weightFlag){
        var minWeight=1000, maxWeight=0;
        // Reading the min and max weight present
        for(var i=0; i<length; i++){
            weight = $(tagsForTagCloud2D[i].childNodes[0]).data('weight');
            if( weight<minWeight ){ minWeight = weight; }
            if( weight>maxWeight ){ maxWeight = weight; }
        }
        var weightGap = maxWeight - minWeight;
        if( weightGap < 1 ){
            $(whereToWrite).append('<p>There is a problem displaying the tags. <br/>Please try again later.
</p>');
            return;
        }
    }

    // Displaying the tags
    for(var i=0; i<length; i++){
        tagName = tagsForTagCloud2D[i].textContent;
        link = tagsForTagCloud2D[i].childNodes[0].href;
        if(weightFlag){
            // weight = tagsForTagCloud2D[i].childNodes[0].dataset.weight;
            weight = $(tagsForTagCloud2D[i].childNodes[0]).data('weight');
            weight = (parseFloat(weight) - minWeight) / weightGap; // This way we have the percentage in
            // Then we want the 0% to be 0.8em font-size and the 100% to be 1.8em font-size
            weight = weight + 0.8; // The font in em will be 0.8 (the minimum size) + [a value from 0 to
            // Building the tag string to print on the web page
            var tagString = '<a href="'+ link +' " target="_blank"> <span style="font-size:'+weight+'em;"+ tagName
            // We increase the font-size accordingly to the data-weight of the tag.

            // Displaying the tag
            $(whereToWrite).append(tagString);
        }
    }
} catch(e){
    $(whereToWrite).append('<p>There is a problem displaying the tags. <br/>Please try again later. </p>');
    console.log('There is a problem displaying the tags. Please debug function createTagsCanvas().');
}

}

/* Function that calls the server to retrieve the feeds with the same tag.
 * Input: the text of the tag to search.
 */
function retrieveFeedsWithSameTag(tagText, whereToWrite, parentWhereToWrite, numOffeeds, flagDisplayReplies){
    if(parentWhereToWrite === '' || parentWhereToWrite === null || parentWhereToWrite === undefined){ parentWhereToWrite = whereToWrite; }
    if(numOffeeds < 0 || numOffeeds > 19 || numOffeeds === null || numOffeeds === undefined){ numOffeeds = 0; }

    // To retrieve the posts with this tag we use the "id" inside the following REST call:
    // var filter = 'path:' + socialWebsite.substring(0, socialWebsite.length-1); // The 'filter' is used to filter the results and
    // receive only the ones coming from Social
    var searchForTagPostsSite = searchRestService + "query?querytext='tags:'+ tagText +'&sourceid='459ddb7-216f-4386-9709-287d5d22f568'&sortlist='created:1'";
    // The code '459ddb7-216f-4386-9709-287d5d22f568' means 'Retrieve only Conversations'

    try{
        executeRestCallExtendedSeven(searchForTagPostsSite, 'GET', null, retrieveFeedsWithSameTagBodyFunction, onError, whereToWrite,
        parentWhereToWrite, numOffeeds, tagText, flagDisplayReplies, null); // searches the tags and display the tagged feeds grouped
    }
    catch(err){ errorHandlerFunction(11, "There was a problem while communicating with the Server.\nPlease try again later."); }

}

function retrieveFeedsWithSameTagBodyFunction(data, whereToWrite, parentWhereToWrite, numOffeeds, tagText, flagDisplayReplies, variableNotUsed){
    if(whereToWrite === null){ return; } // consistency check
    if(whereToWrite[0] !== '#'){
        whereToWrite = '#' + whereToWrite.toString();
    }

    try{
        var result = JSON.parse(data); // parsing the data obtained from the social network
    } catch(e){
        var uniqueId = 'networkProblemsWarning'+Date.now(); // It is important to have a unique id, otherwise the function 'hide' will work only
        once.
        $(whereToWrite).append('<div class="feedsItem" id="'+uniqueId+'"> <p id="text"> Network problem. Please try to refresh the page later. </p>
</div>'); // Printing the "problem" message on the screen
        $("#"+uniqueId).delay(5000).fadeOut('slow'); // This function will hide the warning after 5 seconds.
        setTimeout(function(){ $("#"+uniqueId).remove(); }, 7000); // This function will remove the warning from the HTML code after 7 seconds.

        console.log("There was a problem while communicating with the Server.\nSee retrieveFeedsWithSameTagBodyFunction() function.");
        return;
    }

    // If the result object exists but has errors it means we made a bad request, than we have to stop the execution and signal it to the developer in the console
    log
    if(result.error){

```

```

        var uniqueId = 'networkProblemsWarning'+Date.now(); // It is important to have a unique id, otherwise the function 'hide' will work only
        $(whereToWrite).append('<div class="feedsItem" id="'+uniqueId+'"> <p id="text"> Network problem. Please try to refresh the page later. </p>
    </div>'); // Printing the "problem" message on the screen
        $("#"+uniqueId).delay(5000).fadeOut('slow'); // This function will hide the warning after 5 seconds.
        setTimeout(function(){ $("#"+uniqueId).remove(); }, 7000); // This function will remove the warning from the HTML code after 7 seconds.

        console.log("Bad request.\nPlease review the retrieveFeedsWithSameTagBodyFunction() function.");
        return;
    }

    // Consistency check : if no information has been retrieved...
    if(result.d === null || result.d === undefined)
    {
        var uniqueId = 'networkProblemsWarning'+Date.now(); // It is important to have a unique id, otherwise the function 'hide' will work only
        once.
        $(whereToWrite).append('<div class="feedsItem" id="'+uniqueId+'"> <p id="text"> Network problem. Please try to refresh the page later. </p>
    </div>'); // Printing the "problem" message on the screen
        $("#"+uniqueId).delay(5000).fadeOut('slow'); // This function will hide the warning after 5 seconds.
        setTimeout(function(){ $("#"+uniqueId).remove(); }, 7000); // This function will remove the warning from the HTML code after 7 seconds.

        return;
    }
    // else...

    // Stopping the automatic retrieval of new feeds that would hide the "extra" feeds that the User may be reading at the moment.
    // To refresh the feeds and reactivate the automatic retrieval (of the feeds) the User may refresh the web-page.
    try{
        var handler = findMyHandler(whereToWrite);
        if(handler !== -1){
            clearInterval(handler);
        }
        // If it is the first step in displaying the feeds there will not be any button yet. It will be added in the printArrayOfFeedsWithSameTag()
        function

        var button = getElementInsideContainer(whereToWrite, "moreFeedsButton");
        button.parentNode.removeChild(button); // Removing the "moreFeeds" button
    }catch(err){}

    var primaryQueryResultsFound = false;
    var secondaryQueryResultsFound = false;
    var primaryArray;
    var secondaryArray;

    try{
        // Trying to read the results from the PrimaryQueryResult
        primaryArray = result.d.query.PrimaryQueryResult.RelevantResults.Table.Rows.results;

        if( primaryArray.length > 0 && primaryArray !== "" && primaryArray !== undefined && primaryArray !== null){
            primaryQueryResultsFound = true;
        }else{
            primaryQueryResultsFound = false;
        }
    }catch(err){ }

    try{
        // If the PrimaryQueryResult does not contain the results then the SecondaryQueryResults will
        secondaryArray = result.d.query.SecondaryQueryResults.results[0].RelevantResults.Table.Rows.results;

        if( secondaryArray.length > 0 && secondaryArray !== "" && secondaryArray !== undefined && secondaryArray !== null){
            secondaryQueryResultsFound = true;
        }else{
            secondaryQueryResultsFound = false;
        }
    }
    }catch(err){ }

    // If no results have been found then we can stop the function
    if(!primaryQueryResultsFound && !secondaryQueryResultsFound){
        // Telling the User that no feed has been found
        var uniqueId = 'networkProblemsWarning'+Date.now(); // It is important to have a unique id, otherwise the function 'hide' will work only
        once.
        $(whereToWrite).append('<div class="feedsItem" id="'+uniqueId+'"> <p id="text"> No more feeds found with this tag. You may try again later. </p>
    </div>'); // Printing the "problem" message on the screen
        $("#"+uniqueId).delay(5000).fadeOut('slow'); // This function will hide the warning after 5 seconds.
        setTimeout(function(){ $("#"+uniqueId).remove(); }, 7000); // This function will remove the warning from the HTML code after 7 seconds.

        return; // Terminating the function
    }
    // else: some results have been found -> continue

    // If there is nothing in the primary array...
    if(!primaryQueryResultsFound)
    {
        // We print the secondary array of feeds found
        try{
            printArrayOfFeedsWithSameTag(secondaryArray, whereToWrite, parentWhereToWrite, numOffeeds, tagText, flagDisplayReplies);
        }catch(e){
            // Telling the User that no feed has been found
            var uniqueId = 'networkProblemsWarning'+Date.now(); // It is important to have a unique id, otherwise the function 'hide'
            will work only once.
            $(whereToWrite).append('<div class="feedsItem" id="'+uniqueId+'"> <p id="text"> There has been a problem while reading the feeds.
            Please try again later. </p> </div>'); // Printing the "problem" message on the screen
            $("#"+uniqueId).delay(5000).fadeOut('slow'); // This function will hide the warning after 5 seconds.
            setTimeout(function(){ $("#"+uniqueId).remove(); }, 7000); // This function will remove the warning from the HTML code after 7
            seconds.

            return;
        }
    }
}

```

```

    }
    else
    {
        // If the primaryArray is not null...
        // We check the secondary array. If it is empty...
        if(!secondaryQueryResultsFound)
        {
            // Printing feeds from the 'primaryArray'
            try{
                printArrayOfFeedsWithSameTag(primaryArray, whereToWrite, parentWhereToWrite, numOfFeeds, tagText,
                flagDisplayReplies);
            }catch(e){
                // Telling the User that no feed has been found
                var uniqueId = 'networkProblemsWarning'+Date.now(); // It is important to have a unique id, otherwise the
                function 'hide' will work only once.
                $(whereToWrite).append('<div class="feedsItem" id="'+uniqueId+'"> <p id="text"> There has been a problem while
                reading the feeds. Please try again later. </p> </div>'); // Printing the "problem" message on the screen
                $("#"+uniqueId).delay(5000).fadeOut('slow'); // This function will hide the warning after 5 seconds.
                setTimeout(function(){ $("#"+uniqueId).remove(); }, 7000); // This function will remove the warning from the HTML
                code after 7 seconds.
            }
            return;
        }
    }
    else
    {
        // If both the arrays are useful we merge the two of them and call the printArrayOfFeedsWithSameTag() function passing the merged
        array.
        primaryArray = primaryArray.concat(secondaryArray); // Merging the two arrays into 'primaryArray'
        // Printing feeds found
        try{
            printArrayOfFeedsWithSameTag(primaryArray, whereToWrite, parentWhereToWrite, numOfFeeds, tagText,
            flagDisplayReplies); // Examining the two arrays at once.
        }catch(e){
            // Telling the User that no feed has been found
            var uniqueId = 'networkProblemsWarning'+Date.now(); // It is important to have a unique id, otherwise the
            function 'hide' will work only once.
            $(whereToWrite).append('<div class="feedsItem" id="'+uniqueId+'"> <p id="text"> There has been a problem while
            reading the feeds. Please try again later. </p> </div>'); // Printing the "problem" message on the screen
            $("#"+uniqueId).delay(5000).fadeOut('slow'); // This function will hide the warning after 5 seconds.
            setTimeout(function(){ $("#"+uniqueId).remove(); }, 7000); // This function will remove the warning from the HTML
            code after 7 seconds.
        }
        return;
    }
}

/* This function examines the content of the array and prints it on the screen in the 'whereToWrite' section.
* input:
* - array: the array to examine
* - whereToWrite: section ID of the area where the feeds have to be displayed
* - parentWhereToWrite: section ID of the parent area where the feeds are going to be displayed
* - numFeeds: the maximum number of feeds to display
* - tagText: the name of the tag with which we are requesting the feeds
*/
function printArrayOfFeedsWithSameTag(array, whereToWrite, parentWhereToWrite, numFeeds, tagText, flagDisplayReplies){
    clearTimeout(hashtagCheckTimer); // This line stops the timer that is going to check the situation of the section that should contain the feeds
    with same hashtag

    var arrayOfReadFeedsLinks = new Array(); // This array will contain only the 'original path' of the feeds already read and displayed.

    // Ordering the feeds in chronological order, from the most to the least recent one.
    // This operation is necessary because sometimes the results are coming from two concatenated arrays (each of them is ordered, but concatenated they may be no
    more).

    // array = array.sort();
    array = array.sort(function(a,b){
        var dateA = new Date(getValue("Created", a.Cells.results));
        var dateB = new Date(getValue("Created", b.Cells.results));

        return dateB - dateA; // Orders the feeds from the one with the most recent date to the one with the least recent date.
    });

    var itemResults;
    var originalPath;
    var parentLink;
    var numFeedsToDisplay;
    if(numFeeds === null || numFeeds === undefined || numFeeds <= 0){
        numFeedsToDisplay = array.length;
    }
    else{
        numFeedsToDisplay = Math.min(array.length, numFeeds);
    }

    var i = 0;
    while(i < numFeedsToDisplay && i < array.length) {
        itemResults = array[i].Cells.results;

        parentLink = getValue("ParentLink", itemResults); // The originalPath is not UNIQUE. In Sharepoint a single feed can
        have multiple 'OriginalPaths' between the feed and the replies. This is why we use multiple elements to determine if the feeds has already been displayed or not.
        originalPath = getValue("OriginalPath", itemResults); // The originalPath is not UNIQUE. In Sharepoint a single feed can
        have multiple 'OriginalPaths' between the feed and the replies. This is why we use multiple elements to determine if the feeds has already been displayed or not.
        // originalPath = getValue("RootPostUniqueID", itemResults); // Works only in the Development environment

        // Redundancy check - if this feed has already been displayed, we skip it.
        // The path can be different within the same feed, so we have to check the text and creation date.
        if( arrayOfReadFeedsLinks.indexOf(parentLink) == -1){ // If the originalPath has never been read before...

```

```

class=""originalPath""></div>;

                                document.getElementById( whereToWrite.substring(1, whereToWrite.length) ).innerHTML += '<div id=""originalPath""

try{
                                executeRestCallExtendedFive(formDigestUrl, 'POST', null, showUserInformationInFeedsWithSameTag, onError,
whereToWrite, parentWhereToWrite, originalPath, flagDisplayReplies);
                                arrayOfReadFeedsLinks.push(parentLink);
                                }catch(e){
                                }

                                i++;
                                }

// If there is the chance that there are more feeds to retrieve...
if( numFeedsToDisplay < array.length ){
    // Adding an extra element to the feeds section.
    // This button will allow the User to ask for more feeds.

    var latestTimeFeed = new Date(getValue("Created", array[numFeedsToDisplay-1].Cells.results));
    var dateTimeString = latestTimeFeed.toJSON(); // Converting the dateTime of the last feed printed to give the function the correctly
formatted string, that will work for the REST calls.
    $(whereToWrite).append('<a id="moreFeedsButton" class="moreFeedsButton" href="javascript:socialAPI().moreFeedsFunction(4#39;'+ dateTimeString
+'#39;, 4#39;'+ whereToWrite +'#39;, 4#39;'+ parentWhereToWrite +'#39;, 4#39;'+ tagText +'#39;, 4#39;'+ numFeedsToDisplay +'#39;,4#39;'+ flagDisplayReplies +'#39;)'> Show more posts
</a>');
}

// Activating a function that will check the situation of the page after 6 seconds.
// If, after that time, the page is still empty we assume that no feeds have been retrieved. Thus, we show a message to the User explaining the situation.
setTimeout(function(){
    // If every feed found has no text they are not displayed and the "whereToWrite" section will remain empty. In this case:
    if( $(whereToWrite).html() === null || $(whereToWrite).html() === "" || $(whereToWrite).html() === undefined){
        $(whereToWrite).append('<div class="feedsItem"> <p id="text"> No feeds found with this tag.<br/>Please try again later.</p>');
    }
},12000);

}

function showUserInformationInFeedsWithSameTag(data, whereToWrite, parentWhereToWrite, originalPath, flagDisplayReplies){
    try{
        var result = JSON.parse(data); // Parsing the data obtained from the social network
        var formDigest = result.d.GetContextWebInformation.FormDigestValue;
    }catch(e){
        console.log("There was a problem while communicating with the Server.\nSee showUserInformationInFeedsWithSameTag() function.");
        $(whereToWrite).html("There was a problem while communicating with the Server. Please try again later.");
        return;
    }
    // If the result object exists but has errors it means we made a bad request, than we have to stop the execution and signal it to the developer in the console
    if(result.error){
        console.log("Error: "+ result.error.message.value +"\nPlease review the showUserInformationInFeedsWithSameTag() function.");
        $(whereToWrite).html("There was a problem while communicating with the Server. Please try again later.");
        return;
    }

    var xhr = createCORSRequest("POST", myFeedManagerEndpoint + "post"); // Creating CORSRequest to Like the feed
    xhr.onload = function(){ showUserInformationInFeedsWithSameTagBodyFunction(this.responseText, whereToWrite, parentWhereToWrite, originalPath,
flagDisplayReplies); };
    xhr.onerror = console.log("CORS request encountered an error.\nSee showUserInformationInFeedsWithSameTag() function.");
    xhr.withCredentials = true;

    xhr.setRequestHeader("X-RequestDigest", formDigest);
    xhr.setRequestHeader("content-type", "application/json; charset=utf-8; odata=verbose");

    var data = "{ 'ID':'"+originalPath+"'}"; // Including the ID of the feed we want to analyze. We can also use the URL of the feed as ID.

    xhr.send(data); // Sending the information

}

function showUserInformationInFeedsWithSameTagBodyFunction(result, whereToWrite, parentWhereToWrite, originalPath, flagDisplayReplies){
    /*****
    * Reading the data received
    *****/
    var thread;
    try{
        result = JSON.parse(result); // parsing the data obtained from the social network
        // If the result object exists but has errors it means we made a bad request, than we have to stop the execution and signal it to the developer
        if(result.error){
            setTimeout(function(){
                var innerDivs, numDivs, contentFlag = false;
                var uniqueId = 'networkProblemsWarning'+Date.now(); // It is important to have a unique id, otherwise the
function 'fadeOut' will work only once.

                try{
                    innerDivs = document.getElementById(whereToWrite.substring(1)).getElementsByName("div");
                    numDivs = innerDivs.length;
                }catch(e){ numDivs = 0; } // We will not go in the following 'for' cycle

                for(var i=0; i<numDivs; i++){
                    if( innerDivs[i].innerHTML !== "" ){ // if there is some content in the 'div' section
                        contentFlag = true;
                    }
                }
                // we put the flag to 'true'
                break;
            }
            // and stop the cycle
        }
        // This cycle also avoids that more than one 'error' message is shown to the User.

        // If every feed found has no text they are not displayed and the "whereToWrite" section will remain empty for the
User. In this case:
        if( !contentFlag ){

```



```

    }
    else{
        authorIndex = 0;
        groupString = '';
    }

    var postAuthorName = thread.actors.results[authorIndex].Name; // The name of the User that posted the feed

    var profileImageUri = thread.actors.results[authorIndex].ImageUri; // The profile image of the Author of the feed
    // If the User has not yet a profile image a default one is visualized
    if( profileImageUri === null || profileImageUri === undefined || profileImageUri === '' ){
        profileImageUri = socialWebsite + '_layouts/15/images/PersonPlaceholder.42x42x32.png?rev=23';
    }
    profileImageUri = '';

    var personalAboutPage = thread.actors.results[authorIndex].PersonalSiteUri; // This is the Uri of the User's homepage on Social.

    var text = thread.RootPost.Text;
    // Consistency check - if the feed has no text the function ends.
    if( text === null || text === undefined || text === "" )
    {
        return;
    }
    else
    {
        var matchString = "<c0>CERN</c0> Accelerating science <ddd/>";
        var tempIndex = text.indexOf(matchString); // If there is a string like 'matchString' in the text the output will be the index of that
        substring.

        if( tempIndex !== -1){
            text = text.substring(0, tempIndex) + text.substring(tempIndex + matchString.length, text.length);
        }

        text = myEscapeHTML(text); // Preventing code injection!
        text = formatText(text, parentWhereToWrite); // This function will adapt the text to our needs
    }

    var attachmentUri;
    try{
        attachmentUri = thread.RootPost.Attachment.Uri;
    }catch(e){}
    var attachmentString;
    if( attachmentUri === undefined || attachmentUri === null || attachmentUri === '' ){
        attachmentString = '';
    }else{
        attachmentString = '<p>  </p>';
    }

    // Redundancy check - if this feed has already been displayed, we skip it.
    if( document.getElementById('profileImageSection'+ threadId) ){
        return;
    }

    var strOutput = '<div class="feedsItem" id="feedsItem"> ' +
        '<div class="table"> ' +
        '<div class="picSection">' +
        '<a href="'+personalAboutPage+' " target="_blank"> ' +
        '<span id="picSection" ' +
        'profileImageUri +
        '</span>' +
        '</a>' +
        '</div>' +
        '<div class="notPicSection">' +
        '<p><span id="author"> <a
        href="'+personalAboutPage+' " target="_blank"> ' + postAuthorName + ' </a> </span> ' + groupString + '</p>' +
        '<p>' + text + '</p>' +
        attachmentString +
        '<p id="likeString'+ threadId +' "> ' +
        likeString +
        '</p>' +
        '</div>' +
        '</div>' +
        '<div id="'+ repliesSectionID +' "></div>';

    $(strOutput).hide().appendTo(newWhereToWrite).fadeIn(800);

    // Appending the reply section that will be shown when pressing the "reply" button of a feed.
    $(newWhereToWrite).append('<div class="textbox"'+ tempParentWhereToWrite + threadId +' ">' +
        '<p id="textAreaReplySection"> <textarea
        placeholder="" wrap="hard" id="textAreaReply" class="textAreaReply'+ tempParentWhereToWrite + threadId +' "></p>' +
        '<p class="replyButtonsGroup"> <input type="button"
        value="Reply" id="replyButton" class="uploadMessage'+ tempParentWhereToWrite + threadId +' " onclick="socialAPI().createReply($#39;'+parentWhereToWrite+' '+newWhereToWrite.id+'
        '+threadId+'$#39;)"> </p>' +
        '</div>');

    var containerID; // To have the container ID we have got to check if 'newWhereToWrite' is an object pointing to the HTML section or just the string
    containing the ID of the section.
    if(whereToWrite !== "#socialAPISingleFeed" && whereToWrite !== "socialAPISingleFeed"){
        // In this case the reply button is already hidden from the CSS and we want to also hide the textarea.

        if( typeof(newWhereToWrite) !== "string" ){
            containerID = newWhereToWrite.id;
        }
        else{
            containerID = newWhereToWrite;
        }

        // We are now hiding the textAreaReply section. This has to do be done here and not in the CSS because otherwise it will not work well in IE
        (even IE11), causing the whole page to crash if Enter is pressed while the cursor is inside the textbox.
        var elem = getElementInsideContainer(containerID, "textbox" + tempParentWhereToWrite +threadId); // Getting the element of the 'textbox'
        just appended to the 'whereToWrite' section.
        $(elem).hide();
    }

```



```

}else{
    // The textarea is visible and we want to also show the 'Reply' button.

    if( typeof(newWhereToWrite) !== "string" ){
        containerID = newWhereToWrite.id;
    }
    else{
        containerID = newWhereToWrite;
    }

    // We are now hiding the textAreaReply section. This has to do be done here and not in the CSS because otherwise it will not work well in IE
    (even IE11), causing the whole page to crash if Enter is pressed while the cursor is inside the textbox.
    var elem = getElementInsideContainer(containerID, "uploadMessage" + tempParentWhereToWrite +threadId); // Showing the "Reply" button that is
    normally hidden through CSS.
    $(elem).css("display", "inline");
}

// If we don't want to display the replies...
if(whereToWrite == "#socialAPISingleFeed" || whereToWrite == "socialAPISingleFeed" || !flagDisplayReplies){
    return; // Skipping the displaying of the replies. We want to see only the feeds.
}

/*****
 * Displaying the information of the REPLIES
 *****/
// Displaying the replies
var repliesSection = document.getElementById(repliesSectionID);
var numberOfRepliesToShow = 0;
var replyPicUri;
var replyActorName;
var text;
var attachmentUri;
var numberOfReplies = 0;
try{
    numberOfReplies = thread.Replies.results.length;
}
catch(e){}

if(numberOfReplies > 0){
    var replies = thread.Replies.results; // Catching the replies

    for(var y=numberOfReplies-1; y>=0; y--){
        authorIndex = replies[y].AuthorIndex; // Reading the index of the author inside the array of Actors in the thread
        var id = replies[y].ID; // The ID of the reply

        // Reading all the information about the reply
        replyPicUri = thread.actors.results[authorIndex].ImageUri;
        // If the User has no profile image...
        if(replyPicUri === null || replyPicUri === undefined || replyPicUri === ''){
            replyPicUri = socialWebsite + "_layouts/15/images/PersonPlaceholder.42x42x32.png?rev=23"; // ...we set the default
            image.
        }

        replyActorName = thread.actors.results[authorIndex].Name; // Reading the name of the author of the reply
        personalAboutPage = thread.actors.results[authorIndex].PersonalSiteUri; // The Uri of the User's homepage on
        Social

        text = replies[y].Text; // Reading the content of the reply
        text = myEscapeHTML(text); // Preventing code injection!
        text = formatText(text, parentWhereToWrite); // This function will adapt the text to our needs

        var attachmentUri;
        try{
            attachmentUri = replies[y].Attachment; // Reading the attachment URI. If present it will be displayed,
            otherwise its HTML section will self-remove itself.
        }
        catch(e){}
        var attachmentString;
        if( attachmentUri === undefined || attachmentUri === null || attachmentUri === '' ){
            attachmentString = '';
        }
        else{
            attachmentString = '<p>  </p>';
        }

        dateString = createDateString(new Date(replies[y].CreatedTime)); // creating the string with the date of last
        modification of the feed
        likeString = createLikeReplyString(replies[y].LikerInfo.TotalCount, replies[y].LikerInfo.IncludesCurrentUser, id,
        newWhereToWrite); // Creating the string about the likes of the feed

        // Writing the reply on the web page
        //repliesSection.innerHTML +
        var strOutput = '<div class="replyItem">' +
        '<div
        class="table">' +
        '<div class="picSection">' +
        '<a href="'+personalAboutPage+'" target="_blank">' +
        '<div id="picSection">' +
        '' +
        '</div>' +
        '</a>' +
        '</div>' +
        '<div class="notPicSection">' +
        '<p> <span id="author"> <a href="'+personalAboutPage+'" target="_blank">' + replyActorName + ' </a> </span> </p>' +

```



```

}

/* This function check the text for the presence of link to webpages.
 * If any link is found it is substituted with an <a> HTML statement to make the User able to go on that website using a new tab.
 */
function checkForWebpages(text){
    // Now we try to find possible URL links inside the text.
    var parts = text.split(" "); // separate input by spaces ( URLs don't have spaces )
    text = ""; // Resetting text
    var prefix = "http://";
    var linkFlag = false;
    // Attempt to validate each string as URL.
    // If it is an URL it is converted and then appended to the "text" string.
    // else: it is simply appended to the "text" string.
    for (var index = 0; index < parts.length; index++) {
        var afterUrl = '', beforeUrl = '';
        while(parts[index][0] === '(' && parts[index].length > 2){
            beforeUrl += '(';
            parts[index] = parts[index].substring(1);
        }
        while(parts[index][parts[index].length-1] === ')' && parts[index].length > 2){
            afterUrl += ')';
            parts[index] = parts[index].substring(0, parts[index].length-1);
        }

        try{
            linkFlag = validateURL(parts[index]);
        }catch(err){ linkFlag = false; }

        if( linkFlag ){
            // If it is a valid URL then replace with anchor...
            // If the URL does not have the "http://" at the beginning we have to add it on the reference, otherwise it will point to a page
            inside our website, which is not the target.

            if( parts[index][0] !== 'h' && parts[index][1] !== 't' && parts[index][2] !== 't' && parts[index][3] !== 'p' )
            {
                text += beforeUrl + "<a href=\"" + prefix + parts[index] + "\" target=\"_blank\"" + parts[index] + "</a>" + afterUrl
            }
            else
            {
                text += beforeUrl + "<a href=\"" + parts[index] + "\" target=\"_blank\"" + parts[index] + "</a>" + afterUrl + " ";
            }
        }
        else
        {
            if( index < (parts.length-1) )
            {
                // It was not a valid URL. Appending the text as it is:
                text += parts[index] + " ";
            }
            else
            {
                // This is the last element, no space has to be added at the end.
                text += parts[index];
            }
        }
    }

    return text;
}

/* This function check the text of one feed for the presence of tags.
 * If any tag is found it is substituted with an <a> HTML statement to make the User able to call for feeds with that tag.
 */
function checkForTags(text, whereToWrite){
    var tempTag = "";
    var tempStrToCompare = '<a ';
    var tempStrToCompareLength = tempStrToCompare.length;
    var tempRes = false;

    var x=0;
    // The field "text.length" has to be left as it is, because the length of the text changes everytime we find a tag, therefore it has to be retrieved dynamically.
    while( x < text.length){

        tempRes = compareSubstring(text, x, tempStrToCompareLength, tempStrToCompare);
        // If 'tempRes' is false the string "<a href" has not been found. Otherwise...
        if(tempRes === true)
        {
            // The string "<a href" has been found.
            // We have a link inside our message. We have to go forward in the message to the end of the link and continue our work from
            there.

            x += 3; // We can move beyond the '<a ' string
            while( x < text.length ){

                x++;
                while(text[x] !== '<' && x < text.length){
                    x++;
                }
                // Now text[x] = '<'

                if( text[x+1] === '/' && text[x+2] === 'a' && text[x+3] === '>' ){
                    x += 4; // Goes beyond the "</a>" tag
                    break;
                }
            }
        }

        // Making sure that it is a tag (#something) and it is not the HTML code for curly brackets ("&#123;" and "&#125;")
        if(text[x] === '#' && isOnlyLetterOrNumber(text[x+1]) && ( text[x+1] !== '1' && text[x+2] !== '2' && text[x+3] !== '3' && text[x+4] !== '4' ) && (
            text[x+1] !== '1' && text[x+2] !== '2' && text[x+3] !== '5' && text[x+4] !== '6' ) && ( text[x+1] !== '3' && text[x+2] !== '9' && text[x+3] !== '9' ) && (x+1 < text.length){

```

```

// The first element is a '#'
tempTag += text[x];
x++; // moving on

// From now on only letters and number will be accepted as part of the tag

// Reading the tag
while( x < text.length && isOnlyLetterOrNumber(text[x]) ){

    tempTag += text[x]; // copying the x-th character of the text into the j-th position in "tempTag"

    x++;

}

var newTextObj = tagReplace(text, x-tempTag.length, tempTag, whereToWrite);
text = newTextObj.text;

x += newTextObj.gap-1; // The index is just beyond the tag. We have to add the string length and the length of the tag
again. -1 because the x will be increased by one at the end of the while cycle.

tempTag = ""; // Resetting 'tempTag'

}

x++;

}

return text;

}

/*****
 * CREATION OF EVENT LISTENERS
 *****/
/* No event listener is needed right now. If needed, use this function as example:
window.onload = function ()
{
    var elem;

    elem = document.getElementsByClassName('likeFeed'); // Retrieves all the elements with class = 'likeFeed'
    if(elem){
        var temp;
        var threadId;
        for(var i=0; i < elem.length; i++){
            temp = elem[i].parentElement.parentElement.parentElement;
            threadId = temp.lastChild.innerHTML; // Reading the threadId from the span invisible field in the HTML page

            temp.lastChild.addEventListener('click', function(){ socialAPI().likeFeedFunction(threadId, whereToWrite) }, false);

        }

        elem = document.getElementsByClassName('unlikeFeed'); // Retrieves all the elements with class = 'unlikeFeed'
        if(elem){
            var temp;
            var threadId;
            for(var i=0; i < elem.length; i++){
                temp = elem[i].parentElement.parentElement;
                threadId = temp.lastChild.innerHTML;

                // should I add the 'onclick' to the <a> section? But this would mean not having a onEvent situation...
                temp.lastChild.addEventListener('click', socialAPI().unlikeFeedFunction(threadId, whereToWrite), false);

            }

        }

    }
}
*/

/*****
 * DEFINITION OF THE SOCIALAPI ELEMENT
 *****/
// This element retrieves the elements from the web-page to which we would like to apply the changes.
var SocialAPI = function(){ return; };

/*****
 * DEFINITION OF THE SOCIALAPI MAIN FUNCTION
 *****/
socialAPI = function() {
    return new SocialAPI();
}
// This line allows the Developer to call the prototyped functions (see below) from outside this environment simply writing something like:
// socialAPI().nameOfPrototypedFunction(inputVariable);

/*****
 * DEFINITION OF THE PROTOTYPED FUNCTIONS
 *****/
// Exposing the prototype object via socialAPI.fn so methods can be added later
socialAPI.fn = SocialAPI.prototype = {
    // API methods
    // Main methods
    authenticateOnSocial: function(inputFunction){
        authenticateOnSocial(inputFunction);
    },
    updateFollowedFeeds: function(whereToWrite, updateInterval, numFeeds, flagDisplayReplies){
        updateFollowedFeeds(whereToWrite, updateInterval, numFeeds, flagDisplayReplies);
    },
    updateFeedsFromProfile: function(accountName, whereToWrite, updateInterval, numFeeds, flagDisplayReplies){
        updateFeedsFromProfile(accountName, whereToWrite, updateInterval, numFeeds, flagDisplayReplies);
    }
};

```

```

    },
    updateFeedsWithSameHashtag: function(tag, whereToWrite, updateInterval, numOfFeeds, flagDisplayReplies){
        updateFeedsWithSameHashtag(tag, whereToWrite, updateInterval, numOfFeeds, flagDisplayReplies);
    },
    updateGroupInfo: function(whereToWrite, department, group, section, imageFlag, groupNameFlag, numFeeds){
        updateGroupInfo(whereToWrite, department, group, section, imageFlag, groupNameFlag, numFeeds);
    },
    updateSingleFeed: function(whereToWrite, url){
        updateSingleFeed(whereToWrite, url);
    },
    manuallyUpdateAllTheFeeds: function(){
        manuallyUpdateAllTheFeeds();
    },
    findTaggedFeeds: function(tag, whereToWrite){
        findTaggedFeeds(tag, whereToWrite);
    },
    loadTagCloud: function(whereToWrite, maxNumTags, textColor, textBorderColor, numDimensions, weightFlag, periodOffTime){
        loadTagCloud(whereToWrite, maxNumTags, textColor, textBorderColor, numDimensions, weightFlag, periodOffTime);
    },
    postToMyFeeds: function(inputMessage, inputFunction){
        postToMyFeeds(inputMessage, inputFunction);
    },
    // Other methods
    clearMessageToTheUser: function(id){
        clearMessageToTheUser(id);
    },
    moreFeedsFunction: function(dateTime, whereToWrite, parentWhereToWrite, accountName, numFeedsToDisplay, numFeedsStillToGet, flagDisplayReplies){
        moreFeedsFunction(dateTime, whereToWrite, parentWhereToWrite, accountName, numFeedsToDisplay, numFeedsStillToGet, flagDisplayReplies);
    },
    moreGroupElements: function(whereToWrite, department, group, section, imageFlag, departmentFlag, numFeeds){
        moreGroupElements(whereToWrite, department, group, section, imageFlag, departmentFlag, numFeeds);
    },
    moreRepliesFunction: function(link){
        moreRepliesFunction(link);
    },
    deleteFeed: function(id){
        deleteFeed(id);
    },
    unfollowPerson: function(inputString){
        unfollowPerson(inputString);
    },
    likeFeedFunction: function(threadId){
        likeFeedFunction(threadId);
    },
    unlikeFeedFunction: function(threadId){
        unlikeFeedFunction(threadId);
    },
    showReplySection: function(mixedInput){
        showReplySection(mixedInput);
    },
    createReply: function(str){
        createReply(str);
    },
    deleteReply: function(id){
        deleteReply(id);
    },
    likeReplyFunction: function(id){
        likeReplyFunction(id);
    },
    unlikeReplyFunction: function(id){
        unlikeReplyFunction(id);
    },
    setErrorHandler: function(func){
        errorHandlerFunction = func;
    }

    // We can write more methods here, each using 'return this', to enable chaining.
};

})(jQuery);

```

File “socialAPI.css”:

```

/* Social API's CSS. Version 1.2
* - Created a new rule that overrides a rule in the webpage which limits the max-width of a textarea at 50em.
*/

/*Fix img size problem from CERN theme*/
.socialAPIWrapClass img{
    width: 100%;
}

.socialAPIWrapClass{
    font-size: 18px;
}

.socialAPIWrapClass a{
    color: #4d94cc; /* To avoid blue underlined links */
    text-decoration: none;
}

.socialAPIWrapClass a:hover{
    color: #4d94cd;
}

```

```

.socialAPIWrapClass a:focus{
    color: #256ca4;
}
.socialAPIWrapClass a.link{
    color: blue;
}

.socialAPIWrapClass button{
    background-color: #297CCF;
    border: 0px;
    color: white;

    padding-top: 4px;
    padding-bottom: 2px;
    padding-left: 7px;
    padding-right: -2px;
}
.socialAPIWrapClass button:hover{
    background-color: #246fba;
}

.socialAPIWrapClass input{
    background-color: #297CCF;
    border: 0px;
    color: white;

    padding-top: 4px;
    padding-bottom: 2px;
    padding-left: 7px;
    padding-right: -2px;
}
.socialAPIWrapClass input:hover{
    background-color: #246fba;
}

/* misc */
.socialAPIWrapClass .left {
    float: left;
}
.socialAPIWrapClass .right {
    float: right;
}

.socialAPIWrapClass h1{
    color: #414141;
    font-size: 2.8rem;
    font-weight: bold;
    line-height: 3rem;
    font-family: "FF Sans", Verdana, Tahoma, "DejaVu Sans", sans-serif;
}

/* updates */
.socialAPIWrapClass .feedsItem .replyItem {
    border-bottom: 6px solid #FFF;
}
.socialAPIWrapClass .label {
    margin-top: 20px;
    margin-bottom: 23px;
    border: 3px solid #212121;
    overflow: hidden;
    text-align: center;
    padding: 0px 10px 0px 10px;

    -webkit-transition: color 0.2s linear, background 0.2s linear;
    -moz-transition: color 0.2s linear, background 0.2s linear;
    -ms-transition: color 0.2s linear, background 0.2s linear;
    -o-transition: color 0.2s linear, background 0.2s linear;
    transition: color 0.2s linear, background 0.2s linear;
}
.socialAPIWrapClass .label:hover{
    background: #212121;
}

.socialAPIWrapClass .label #title{
    font: normal 2em "Lucida Sans Unicode", sans-serif;
    line-height: 40px;
    padding-top: 10px;
}

.socialAPIWrapClass .label #subTitle{
    font: normal 1em "Lucida Sans Unicode", sans-serif;
    line-height: 15px;
    padding-bottom: 10px; /* This line of code is necessary to make every part of the label as a link. Without it the bottom of the label is not a link. */
}

.socialAPIWrapClass #reply{
    font-size: 0.75em;
}

.socialAPIWrapClass #feed{
    font-size: 0.75em;
}

.socialAPIWrapClass #myCanvasContainer{
    vertical-align: top;
    display: inline-block;
    width: 45%;
}

```

```

.socialAPIWrapClass #feedsWithSameTagSphere{
  vertical-align: top;
  display: inline-block;
  width: 53%;
}

.socialAPIWrapClass #findTaggedFeedsArea{
  vertical-align: top;
  display: inline-block;
  width: 30%;
}

.socialAPIWrapClass #feedsWithSameTag{
  vertical-align: top;
  display: inline-block;
  width: 68%;
}

.socialAPIWrapClass .feedsItem {
  display: table;
  table-layout: fixed;
  word-wrap: break-word;
  font: normal 0.73em "Trebuchet MS", sans-serif;
  color: #565656;
  padding-bottom: 10px;
  overflow: hidden; /* necessary to hide text in excess */
  border-bottom: 1px solid #FFF;

  width: 100%;

  -webkit-transition: color 0.2s linear;
  -moz-transition: color 0.2s linear;
  -ms-transition: color 0.2s linear;
  -o-transition: color 0.2s linear;
  transition: color 0.2s linear;
}

.socialAPIWrapClass .feedsItem: hover{
  color: black;
}

.socialAPIWrapClass .feedsItem: hover #deleteFeed {
  filter: alpha(opacity=100);
  opacity: 1;
}

.socialAPIWrapClass .table{
  display: table-row;
}

.socialAPIWrapClass .picSection{
  display: table-cell;
  width: 50px;
}

.socialAPIWrapClass .feedsItem #picSection{
  padding-top: 3px;
  min-width: 38px;
}

.socialAPIWrapClass .feedsItem .notPicSection{
  display: table-cell;
  width: 100%;
  padding-left: 10px;
}

/* Now we put the name of the Author and the text near the user's picture starting at the right height. */
vertical-align: top;
}

.socialAPIWrapClass .notPicSection p{
  margin-top: 0;
  margin-bottom: 8px;
}

.socialAPIWrapClass .notPicSection p a{
  word-break: break-all;
  display: inline-block;
  -ms-word-break: break-all; /* For IE */
}

.socialAPIWrapClass .feedsItem #deleteFeed{
  filter: alpha(opacity=0);
  opacity: 0;

  float: right;
  font-size: 1.3rem;
  font-family: Arial;
  /* The 'X' at the top right of the feed used to delete the feeds changes colour gradually. */
  transition: color 0.2s linear, opacity 0.2s linear, filter 0.2s linear;
  -webkit-transition: color 0.2s linear, opacity 0.2s linear, filter 0.2s linear;
  -moz-transition: color 0.2s linear, opacity 0.2s linear, filter 0.2s linear;
  -o-transition: color 0.2s linear, opacity 0.2s linear, filter 0.2s linear;
}

.socialAPIWrapClass .feedsItem span#author{
  font-size: 1.2em;
  margin: 0px;
  padding: 0px;
}

.socialAPIWrapClass .feedsItem #authorOfFeedsWithSameTag{
  font-size: 1.39em;
}

.socialAPIWrapClass .feedsItem .date {
  color: #777777;
  font-size: 0.85em
}

.socialAPIWrapClass #textAreaReplySection{
  margin-left: 60px;
}

.socialAPIWrapClass textarea{
  max-width: none !important;
}

.socialAPIWrapClass #textAreaReply{

```



```

}

/* content */
.socialAPIWrapClass .label p {
    margin: 4px 0 10px 0;
}

.socialAPIWrapClass input{
    padding:5px;
    margin-top:5px;
}

.socialAPIWrapClass #tool{
    margin: 0 0 10px 0;
    background: #F0F0F0;
    padding: 5px;
    border: 1px solid black;
}

.socialAPIWrapClass #moreFeedsButton{
    clear: right;
    display: block;
    text-align: center;

    margin-top: 10px;
    margin-bottom: 10px;

    margin-right: auto;
    margin-left: auto;
}

.socialAPIWrapClass #moreFeedsButton:active{
    padding-top: 6px;
    padding-bottom: 4px;
    padding-left: 7px;
    padding-right: -2px;
}

.socialAPIWrapClass input#moreRepliesButton{
    clear: right;
    display: block;
    text-align: center;
    color: #454545;

    border-color: #a3a3a3;
    border-style: solid;
    border-width: 1px;

    margin-top: 10px;
    margin-bottom: 10px;

    background-image: -webkit-gradient(
        linear,
        left top,
        left bottom,
        color-stop(0, #F5F5F5),
        color-stop(1, #D4D4D4)
    );
    background-image: -o-linear-gradient(bottom, #F5F5F5 0%, #D4D4D4 100%);
    background-image: -moz-linear-gradient(bottom, #F5F5F5 0%, #D4D4D4 100%);
    background-image: -webkit-linear-gradient(bottom, #F5F5F5 0%, #D4D4D4 100%);
    background-image: -ms-linear-gradient(bottom, #F5F5F5 0%, #D4D4D4 100%);
    background-image: linear-gradient(to bottom, #F5F5F5 0%, #D4D4D4 100%);

    /* To have rounded edges uncomment here:*/
    /*-webkit-border-radius: 5px; /* For Safari, etc. */
    /*-moz-border-radius: 5px; /* For Mozilla, etc.*/
    /*border-radius: 5px; /* CSS3 Feature */

    margin-right: auto;
    margin-left: auto;
}

.socialAPIWrapClass input#moreRepliesButton:hover{
    border-color: #7ba7e;

    background-image: -webkit-gradient(
        linear,
        left bottom,
        left top,
        color-stop(0, #95DE95),
        color-stop(1, #CCFFCC)
    );
    background-image: -o-linear-gradient(top, #95DE95 0%, #CCFFCC 100%);
    background-image: -moz-linear-gradient(top, #95DE95 0%, #CCFFCC 100%);
    background-image: -webkit-linear-gradient(top, #95DE95 0%, #CCFFCC 100%);
    background-image: -ms-linear-gradient(top, #95DE95 0%, #CCFFCC 100%);
    background-image: linear-gradient(to top, #95DE95 0%, #CCFFCC 100%);
}

.socialAPIWrapClass input#moreRepliesButton:active{
    padding-top: 6px;
    padding-bottom: 4px;
    padding-left: 7px;
    padding-right: -2px;

    background: #95db95;
}

.socialAPIWrapClass #genericButton{

```

```

margin: 10px 0 10px 0;
}

.socialAPIWrapClass #profilePicture{
width: 50px;
/* While the width is fixed, the 'height' property will adjust automatically */
max-height: 50px; /* Bounding the height property. */
}

.socialAPIWrapClass #profileReplyPicture{
width: 38px;
/* While the width is fixed, the 'height' property will adjust automatically */
max-height: 38px; /* Bounding the height property. */
overflow: hidden;
}

.socialAPIWrapClass .smile{
display: inline-block; /* This line makes us able to give a width and height to the span section. */
width: 11px;
height: 11px;
background: url("https://social.cern.ch/_layouts/15/images/socialcommon.png?rev=23") -11px -1px;
}

.socialAPIWrapClass #attachmentImage{
/* Same limits as on Social. */
max-height: 300px;
max-width: 300px;
}

.notPicSection #author a {
word-break: normal;
word-wrap: break-word;
}

```

Works Cited

Alavi, M. & Leidner, D. E., 2001. *Review: Knowledge Management and Knowledge Management Systems: Conceptual Foundations and Research Issues.*, s.l.: MIS Quarterly.

Alexa, 2016. *The top 500 Sites on the Web.* [Online] Available at: <http://www.alexa.com/topsites/countries> [Accessed 30 08 2016].

Anderson, C., 2006. *The Long Tail: Why the Future of Business Is Selling Less of More.* s.l.:Hyperion.

Angehrn, A. A., Luccini, M. A. & Maxwell, K., 2008. *InnoTube : A video-based connection tool supporting collaborative innovation*, Fontainebleau, France: Centre for Advanced Learning Technologies (CALT).

Argote, L., McEvily, B. & Reagans, R., 2003. Managing Knowledge in Organizations: An Integrative Framework and Review of Emerging Themes.. *Management Science*, 49(4), pp. 571-582.

Baginski, T. & Sherman, M., 2014. *Creating Internet facing web sites with SharePoint on-premises or in the cloud.* [Online] Available at: <https://channel9.msdn.com/Events/SharePoint-Conference/2014/SPC390> [Accessed 25 08 2016].

Bass, L., Clements, P. & Kazman, R., 2003. *Software Architecture in Practice.* 2nd ed. s.l.:Addison-Wesley Professional.

Bellenghem, S. V., 2012. *The Conversation Company.* s.l.:Kogan Page Limited.

Benkler, Y., 2006. *The Wealth of Networks: How Social Production Transforms Markets and Freedom.* London: Yale University Press.

Bierhoff, K., 2009. *API Protocol Compliance in Object-Oriented Software*. [Online] Available at: <https://www.cs.cmu.edu/~kbierhof/thesis/bierhoff-thesis.pdf> [Accessed 29 08 2016].

Boeije, R., Vries, P. D., Kolfschoten, G. L. & Veen, W., 2009. *Knowledge Workers and the Realm of Social Tagging*. In "Proceedings of the 42nd Hawaii International Conference on System Sciences", s.l.: IEEE.

Boeije, R., Vries, P. D., Kolfschoten, G. L. & Veen, W., 2009. *Knowledge Workers and the Realm of Social Tagging*. *Proceedings of the 42nd Hawaii International Conference on System Sciences*. s.l., IEEE.

Böhringer, M. & Richter, A., 2009. Adopting Social Software to the Intranet: A Case Study on Enterprise Microblogging. In: *Mensch und Computer*. Berlin: s.n., pp. 1-10.

Böhringer, M. & Richter, A., 2009. Adopting Social Software to the Intranet: A Case Study on Enterprise Microblogging.. *Proceedings Mensch und Computer*, pp. 1-10.

Bradley, A. J. & McDonald, M. P., 2011. *The Social Organization*. s.l.:Harvard Business Review Press.

Buettner, R., 2016. *Getting a Job via Career-oriented Social Networking Sites: The Weakness of Ties*. *49th Annual Hawaii International Conference on System Sciences*. Kauai, Hawaii, IEEE.

Cabrera, A. & Cabrera, E. F., 2003. Knowledge-Sharing Dilemmas. *Organization Studies*, 23(5), pp. 687-710.

Carlson, D., n.d. *Bulletin Board Systems*. [Online] Available at: <http://iml.jou.ufl.edu/carlson/history/bbs.htm> [Accessed 30 08 2016].

CERN Computer Security team, 2014. *Social Media guidelines*. [Online] Available at: <https://security.web.cern.ch/security/rules/en/social-media.shtml> [Accessed 24 08 2016].

CERN IR-ECO group, 2016. *Communications strategy 2012-2016 - extracts*. [Online]
Available at: <http://communications.web.cern.ch/strategy>
[Accessed 12 08 2016].

CERN IR-ECO group, 2016. *Welcome page*. [Online]
Available at: <http://communications.web.cern.ch/>
[Accessed 12 08 2016].

CERN Media and Press Relations, 2016. *CERN answers queries from social media*.
[Online]
Available at: <http://press.cern/backgrounders/cern-answers-queries-social-media>
[Accessed 15 08 2016].

CERN, 2014. *CERN Social integration*. [Online]
Available at: <https://entice.web.cern.ch/projects/cern-social-integration>
[Accessed 15 08 2016].

CERN, 2014. *Social API*. [Online]
Available at: <https://cern.ch/social-api>
[Accessed 15 08 2016].

CERN, 2015. *Account Management - Account Types*. [Online]
Available at: <https://account.cern.ch/account/Help/?kbid=011010>
[Accessed 13 08 2016].

CERN, 2015. *CERN Code of Conduct*. [Online]
Available at: <http://hr-dep.web.cern.ch/content/cern-code-conduct>
[Accessed 24 08 2016].

CERN, 2015. *CHEP 2015 Social feed*. [Online]
Available at: <https://cern.ch/chep2015>
[Accessed 15 08 2016].

CERN, 2015. *Social API demo on Drupal*. [Online]
Available at: <https://demo-social.web.cern.ch/>
[Accessed 29 08 2016].

CERN, 2016. *About CERN.* [Online]
Available at: <http://home.cern/about>
[Accessed 11 08 2016].

CERN, 2016. *CERN Social Media.* [Online]
Available at: <http://communications.web.cern.ch/social-media>
[Accessed 12 08 2016].

CERN, 2016. *Origins.* [Online]
Available at: <http://timeline.web.cern.ch/timelines/the-history-of-cern/overlay#1949-12-09> 00:45:00
[Accessed 11 08 2016].

CERN, n.d. *CERN Market.* [Online]
Available at: <https://social.cern.ch/community/cern-market/SitePages/CommunityHome.aspx>
[Accessed 24 08 2016].

CERN, n.d. *CERN Web Services.* [Online]
Available at: <https://webservices.web.cern.ch/webservices/>
[Accessed 13 08 2016].

CERN, n.d. *E-groups.* [Online]
Available at: <https://cern.ch/egroups>
[Accessed 24 08 2016].

CERN, n.d. *IT Lightning Talks (ITLT).* [Online]
Available at: <http://information-technology.web.cern.ch/about/meeting/it-lightning-talks-itlt>
[Accessed 13 08 2016].

CERN, n.d. *Web Services.* [Online]
Available at: <https://webservices.web.cern.ch/webservices/>
[Accessed 24 08 2016].

Cheng, W., Hailin, L. & Hongming, X., 2008. Does Knowledge Sharing Mediate the Relationship between Trust and Firm Performance?. In: *International Symposiums on Information Processing*. Moscow: IEEE, pp. 449-453.

Clarke, S., 2004. *Measuring API Usability*. [Online] Available at: <http://www.drdoobs.com/windows/measuring-api-usability/184405654> [Accessed 29 08 2016].

CompuServe, 2016. *About CompuServe*. [Online] Available at: <http://webcenters.netscape.compuserve.com/menu/about.jsp> [Accessed 30 08 2016].

Connolly, J. F., 2006. *Constellation program overview*. [Online] Available at: http://www.nasa.gov/pdf/163092main_constellation_program_overview.pdf [Accessed 09 08 2016].

Davenport, T. H. & Prusak, L., 1998. *Working Knowledge: How Organizations Manage What They Know..* Boston: Harvard Business School press.

de Figueiredo, L. H., Ierusalimschy, R. & Filho, W. C., 1994. *The design and implementation of a language for extending applications*, s.l.: TeCGraf Grupo de Tecnologia em Computacao Grafica.

De Sousa, B. S., 2014. *Social Networking for CERN*. [Online] Available at: <http://indico.cern.ch/event/298144/contribution/0/material/slides/0.pdf> [Accessed 12 08 2016].

De Sousa, B. S., Wagner, A., Ormancey, E. & Grzywaczewski, P., 2015. *Benefits of Enterprise Social Networking Systems for High Energy Physics community*, Geneva: CERN.

DeCoursy, L., 2001. *ICQ Surpasses 100 Million Registered Users*. [Online] Available at: <http://www.timewarner.com/newsroom/press-releases/2001/05/09/icq-celebrates-100-million-registered-users> [Accessed 30 08 2016].

Dignum, V. & Eijk, R. M., 2005. *Towards a Model to Understand the Influence of Trust in Knowledge Sharing Decisions*. Utrecht, The Netherlands, Workshop On Trust - AAMAS Conference.

eBizMBA Inc., 2016. *Top 15 Most Popular Social Networking Sites*. [Online] Available at: <http://www.ebizmba.com/articles/social-networking-websites> [Accessed 31 08 2016].

Ecma International, 2016. *ECMAScript 2016 Language Specification*. [Online] Available at: <http://www.ecma-international.org/publications/files/ECMA-ST/Ecma-262.pdf> [Accessed 29 08 2016].

Ellison, N. B., 2007. Social Network Sites: Definition, History, and Scholarship. *Journal of computer-mediated communication*, 13(1), pp. 210-230.

Facebook, 2016. *Stats*. [Online] Available at: <http://newsroom.fb.com/company-info/> [Accessed 31 08 2016].

Flanagan, D., 2011. *JavaScript: The Definitive Guide*. 6th ed. s.l.:O'Reilly & Associates.

Frost, A., 2013. *The Different Types of Knowledge*. [Online] Available at: <http://www.knowledge-management-tools.net/different-types-of-knowledge.html> [Accessed 10 08 2016].

Frost, A., 2013. *The Different Types of Knowledge*. [Online] Available at: <http://www.knowledge-management-tools.net/different-types-of-knowledge.html> [Accessed 10 08 2016].

Garfield, S., 2014. *15 Knowledge Management Benefits*. [Online] Available at: <https://www.linkedin.com/pulse/20140811204044-2500783-15-knowledge-management-benefits> [Accessed 09 08 2016].

Gartner, 2013. *Gartner Says 80 Percent of Social Business Efforts Will Not Achieve Intended Benefits Through 2015*. [Online] Available at: <http://www.gartner.com/newsroom/id/2319215> [Accessed 13 08 2016].

Gettier, E. L., 1963. *Is Justified True Belief Knowledge? Analysis*, Minnesota: University of Minnesota.

Gordeyeva, I., 2010. *Enterprise 2.0: theoretical foundations of social media tools influence on knowledge sharing practices in organizations*, Twente: University of Twente.

Griner, D., 2010. *Hey Old Spice haters, sales are up 107%*. [Online] Available at: <http://www.adweek.com/adfreak/hey-old-spice-haters-sales-are-107-12422> [Accessed 09 08 2016].

Griner, D., 2010. *Hey Old Spice haters, sales are up 107%*. [Online] Available at: <https://www.google.it/url?sa=t&rct=j&q=&esrc=s&source=web&cd=1&cad=rja&uact=8&ved=0ahUKEwiq K-BlrTOAhUEVhoKH4YA2AQQFgqeMAA&url=http%3A%2F%2Fwww.adweek.com%2Fadfreak%2Fhey-old-spice-haters-sales-are-107-12422&usq=AFQjCNGZPw4ekZSS6o-qVpSmKEUsITty4g> [Accessed 09 08 2016].

Happe, R., 2010. *Managing The Social Ecosystem – A SAP Case Study*. [Online] Available at: <https://www.communityroundtable.com/best-practices/managing-the-social-ecosystem-an-sap-case-study/> [Accessed 08 08 2016].

Henning, M. & Vinoski, S., 1999. *Advanced CORBA Programming with C++*. s.l.:Addison-Wesley.

Herbalife, 2016. *About Herbalife*. [Online]
Available at: <http://company.herbalife.com/>
[Accessed 4 August 2016].

Herr, P., Kardes, F. & Kim, J., 1991. Effects of word-of-mouth and product-attribute information on persuasion: an accessibility-diagnostics perspective.. *Journal of Consumer Research*, 17(4), pp. 454-462.

Hinchcliffe, D. & Kim, P., 2012. *Social Business by Design*. s.l.:Jossey-Bass.

Hossain, M., 2013. *Using CORS*. [Online]
Available at: http://www.html5rocks.com/en/tutorials/cors/?redirect_from_locale=it
[Accessed 29 08 2016].

InSites Consulting, 2009. *"ConAir" study*, s.l.: s.n.

InSites Consulting, 2012. *Social Media Around the World*. [Online]
Available at: <http://www.slideshare.net/InSitesConsulting/social-media-around-the-world-2012-by-insites-consulting>
[Accessed 09 08 2016].

InterAction Collaboration, 2010. *Peer review of CERN communications*. [Online]
Available at: http://communications.web.cern.ch/system/files/docs/CERN_comms_peer_review_2010.pdf
[Accessed 16 08 2016].

Investopedia, n.d. *Social Media*. [Online]
Available at: <http://www.investopedia.com/terms/s/social-media.asp#ixzz4EPiNysnv>
[Accessed 31 08 2016].

Jacques, B. & Chui, M., 2010. *The Use of Web 2.0 in Businesses*. [Online]
Available at: <http://www.ft.com/cms/s/0/c93e7bba-04a4-11e0-a99c-00144feabdc0.html#axzz1f7uQlRhZ>
[Accessed 08 08 2016].

jQuery Foundation, 2016. *jQuery*. [Online] Available at: <https://jquery.com/> [Accessed 29 08 2016].

Kalt, C., 2000. *RFC2810: Internet Relay Chat: Architecture*. [Online] Available at: <https://tools.ietf.org/html/rfc2810> [Accessed 30 08 2016].

Kaner, C., 2006. *Exploratory Testing*. Orlando, FL, Florida Institute of Technology, Quality Assurance Institute Worldwide Annual Software Testing Conference.

Kaplan, A. M. & Haenlein, M., 2010. Users of the world, unite! The challenges and opportunities of social media. *Business Horizons*, 53(1), p. 61.

Keller, E., 2007. *Opening speech*. Womma Summit, s.n.

Keyes, J., 2013. *Enterprise 2.0 - Social Networking Tools to Transform Your Organization*. s.l.:CRC Press - Taylor & Francis Group.

Kietzmann, J. & Hermkens, K., 2011. Social media? Get serious! Understanding the functional building blocks of social media. *Business Horizons*, Volume 54, pp. 241-251.

Kim, D. J., Hall, S. P., Yue, K. & Gates, T., 2009. Global Diffusion of the Internet XV: Web 2.0 Technologies, Principles, and Applications: A Conceptual Framework from Technology Push and Demand Pull Perspective.. *Communications of the Association for Information Systems*, Volume 24, pp. 657-672.

Kittur, A. & Kraut, R. E., 2008. Harnessing the Wisdom of Crowds in Wikipedia: Quality Through Coordination.. *Proceedings of the ACM 2008 conference on Computer supported cooperative work*, pp. 37-46.

Klier, K., 2011. *From Community to Kinship: Online Communities that Drive Business Impact*. Atlanta, USA, s.n.

Lai, L. S. & Turban, E., 2008. Groups Formation and Operations in the Web 2.0 Environment and Social Networks.. *Group Decision and Negotiation*, 17(5), pp. 387-402.

Lazar, I., 2007. Creating Enterprise 2.0 From Web 2.0.. *Business Communications Review*, 37(8), pp. 14-16.

Levy, M., 2009. WEB 2.0 implications on knowledge management.. *Journal of Knowledge Management*, 13(1), pp. 120-134.

Lewine, D. A., 1994. *POSIX Programmer's Guide*. [Online]
Available at: [ftp://gamma.sbin.org/pub/doc/books/OReilly -
POSIX Programmers Guide.pdf](ftp://gamma.sbin.org/pub/doc/books/OReilly-_POSIX_Programmers_Guide.pdf)

[Accessed 29 08 2016].

Li, C., 2012. *Making the Business Case for Enterprise Social Networks*. [Online]
Available at: <http://www.altimetergroup.com/2012/02/making-the-business-case-for-enterprise-social-networks/>

[Accessed 12 08 2016].

LiveJournal, 2016. *About LiveJournal*. [Online]
Available at: <http://www.livejournal.com/about>

[Accessed 30 08 2016].

Lüfkens, M., 2013. *How do International Organisations Tweet in 2013?*. [Online]
Available at: <http://twiplomacy.com/blog/how-do-international-organisations-tweet/>

[Accessed 12 08 2016].

Lykourantzou, I. et al., 2010. CorpWiki: A self-regulating wiki to promote corporate collective intelligence through expert peer matching. *Information Sciences*, 180(1), pp. 18-38.

Mark, G. & Volda, S., 2012. *Pace not Dictated by Electrons: An Empirical Study of Work Without Email*. [Online]

Available at: https://www.ics.uci.edu/~gmark/Home_page/Research_files/CHI%202012.pdf

[Accessed 13 08 2016].

McAfee, A., 2010. *Andrew McAfee on Enterprise 2.0 Today*. [Online]

Available at: <http://www.forbes.com/2010/04/07/mcafee-enterprise-2-point-0->

[leadership-managing-mitsloan.html](#)

[Accessed 3 August 2016].

McAfee, A. P., 2006. *Enterprise 2.0: The Dawn of Emergent Collaboration.*, s.l.: MIT Sloan Management Review..

McKinsey, 2013. *Evolution of the networked enterprise: McKinsey Global Survey results.* [Online]

Available at: <http://www.mckinsey.com/business-functions/business-technology/our-insights/evolution-of-the-networked-enterprise-mckinsey-global-survey-results>

[Accessed 13 09 2016].

McKinsey, 2015. *Transforming the business through social tools.* [Online]

Available at: <http://www.mckinsey.com/industries/high-tech/our-insights/transforming-the-business-through-social-tools>

[Accessed 13 09 2016].

McManus, R., 2005. *Web 2.0 Definition and Tagging.* [Online]

Available at: http://www.readriteweb.com/archives/web_20_definiti.php

[Accessed 01 02 2005].

Menezes, M., 2014. *Responsive Web Design vs Device Channels in SharePoint 2013.*

[Online]

Available at: <https://spmatt.wordpress.com/2014/04/28/responsive-web-design-v-device-channels-in-sharepoint-2013/>

[Accessed 25 08 2016].

Merriam-Webster, 2016. *Definition of Crowdsourcing.* [Online]

Available at: <http://www.merriam-webster.com/dictionary/crowdsourcing>

[Accessed 10 08 2016].

Microsoft, 2013. *What's new for mobile devices in SharePoint 2013.* [Online]

Available at: <https://technet.microsoft.com/en-us/library/fp161352.aspx>

[Accessed 25 08 2016].

Microsoft, 2015. *Get to know the SharePoint 2013 REST service*. [Online]
Available at: <https://msdn.microsoft.com/en-us/library/office/fp142380.aspx>
[Accessed 29 08 2016].

Microsoft, 2015. *SharePoint Power Searching Using ContentClass*. [Online]
Available at: <https://blogs.msdn.microsoft.com/mvpawardprogram/2015/02/16/sharepoint-power-searching-using-contentclass/>
[Accessed 29 08 2016].

Microsoft, 2016. *Office 365 enterprise social experience: Yammer and Newsfeed*. [Online]
Available at: <https://support.office.com/en-us/article/Office-365-enterprise-social-experience-Yammer-and-Newsfeed-21954c85-4384-47d4-96c2-dfa1c9d56e66>
[Accessed 07 08 2016].

Microsoft, 2016. *Skype for Business*. [Online]
Available at: <https://products.office.com/en/skype-for-business/>
[Accessed 07 08 2016].

Microsoft, 2016. *Social and collaboration features in SharePoint 2013*. [Online]
Available at: <https://msdn.microsoft.com/en-us/library/office/jj163280.aspx>
[Accessed 07 08 2016].

Microsoft, n.d. *SP namespace*. [Online]
Available at: <https://msdn.microsoft.com/en-us/library/office/jj246996.aspx>
[Accessed 26 08 2016].

Mozilla Developer Network, 2015. *CSS Developer Guide*. [Online]
Available at: <https://developer.mozilla.org/en-US/docs/Web/Guide/CSS>
[Accessed 29 08 2016].

Mozilla Developer Network, 2016. *HTML5*. [Online]
Available at: <https://developer.mozilla.org/en-US/docs/Web/Guide/HTML/HTML5>
[Accessed 29 08 2016].

NASA, 2016. *Teams to Compete for \$1.5 Million in Fifth Year of NASA's Sample Return Robot Competition.* [Online] Available at: http://www.nasa.gov/directorates/spacetech/centennial_challenges/sample_return_robot/teams-compete-for-15-million-in-2016-sample-return-robot-competition.html [Accessed 09 08 2016].

NASA, M. S., 2015. *We're NASA Mars scientists. Ask us anything about today's news announcement of liquid water on Mars..* [Online] Available at: https://www.reddit.com/r/IAmA/comments/3mq1wl/were_nasa_mars_scientists_ask_us_anything_about/ [Accessed 09 08 2016].

Nonaka, I., Toyama, R. & Konno, N., 2000. *SECI, Ba and Leadership: a Unified Model of Dynamic Knowledge Creation. Long Range Planning*, s.l.: Elsevier Science Ltd..

O'Reilly, T. & Battelle, J., 2009. *What is Web 2.0.* San Francisco, CA, Web 2.0 Summit.

Obar, J. A. & Wildman, S., 2015. Social media definition and the governance challenge: An introduction to the special issue. *Telecommunications policy*, 39(9), p. 745–750.

Oikarinen, J. & Reed, D., 1993. *RFC1459: Internet Relay Chat Protocol.* [Online] Available at: <https://tools.ietf.org/html/rfc1459> [Accessed 30 08 2016].

Oracle, 2016. *Beehive feature list.* [Online] Available at: <http://www.oracle.com/technetwork/middleware/bee hive/overview/bee hive-feature-list-128862.pdf> [Accessed 07 08 2016].

Oxford Dictionaries, n.d. *Social Network.* [Online] Available at: <http://www.oxforddictionaries.com/definition/english/social-network> [Accessed 31 08 2016].

P&G, n.d. *The “Smell Like a Man, Man” Campaign*. [Online] Available at: https://www.pg.com/en_US/downloads/innovation/factsheet_OldSpice.pdf [Accessed 09 08 2016].

Palmer, N., Swenson, K. & Carlsen, S., 2014. *Empowering Knowledge Workers*. s.l.:Future Strategies Inc.

Pasztor, A., 2010. White House Decides to Outsource NASA Work. *The Wall Street Journal*, Issue <http://www.wsj.com/articles/SB10001424052748704375604575023530543103488>.

Pettus, J., 2011. *Telephone conversation with Anthony J. Bradley on May 11*. [Sound Recording] (NASA MSFC).

Pettus, J. & Bradley, A. J., 2009. *NASA MSFC Social Media Strategy Workshop*, s.l.: Gartner Inc..

Porter, M., 1996. *What is Strategy?*. s.l.:Harvard Business Review.

Quaddus, M. & Xu, J., 2012. Adoption and diffusion of knowledge management systems: field studies of factors and variables.. *Knowledge-Based Systems*, Volume 27, pp. 18-28.

Quinto, J., 2016. *SharePoint 2013 Client Side Rendering: register templates overrides in detail*. [Online] Available at: <http://josharepoint.com/2016/01/14/sharepoint-2013-client-side-rendering-register-templates-overrides-in-detail/> [Accessed 26 08 2016].

Renaud, K., Ramsay, J. & Hair, M., 2006. You've Got E-Mail! Shall I Deal With It Now?. *Internation Journal of Cumputer-Human Interaction*.

Riege, A., 2007. Actions to overcome knowledge transfer barriers in MNCs. *Journal of*, 11(1), pp. 48-67.

Rudrakshi, C. et al., 2014. *API-Fication, Core building Block of the Digital Enterprise*, s.l.: HCL Technologies.

Safko, L., 2012. *The Social Media Bible: Tactics, Tools, and Strategies for Business Success*. 3rd ed. s.l.:Wiley.

Schroeder, S., 2010. *Old Spice: The Archetype of a Successful Social Media Campaign*. [Online]

Available at: <http://mashable.com/2010/07/15/old-spice-social-media-campaign/>
[Accessed 09 08 2016].

Sintes, T., 2001. *Just what is the Java API anyway?*. [Online]

Available at: <http://www.javaworld.com/article/2077392/java-se/just-what-is-the-java-api-anyway.html>

[Accessed 29 08 2016].

Swenson, E., 2013. *Responsive Design for SharePoint 2013*. [Online]

Available at: <http://erikswenson.blogspot.it/2013/06/responsive-design-for-sharepoint-2013.html>

[Accessed 25 08 2016].

Tapiador, A., Fumero, A., Salvachua, J. & Aguirre, S., 2006. A Web Collaboration Architecture.. *International Conference on Collaborative Computing: Networking, Applications and Worksharing*, p. 12.

Tapscott, D. W. A. D., 2006. *Wikinomics*. s.l.:Tantor Media.

Tapscott, D. & Williams, A. D., 2006. *Wikinomics: How Mass Collaboration Changes Everything*. s.l.:Portfolio.

Teng, J. T. C. & Song, S., 2011. An exploratory examination of knowledge-sharing behaviours: solicited and voluntary.. *Journal of Knowledge Management*, 15(1), pp. 104-117.

Thomas-Hunt, M. C., Ogden, T. Y. & Neale, M. A., 2003. Who's really sharing? Effects of social and expert status on knowledge exchange within groups.. *Management Science*, 49(4), p. 464-477.

W3C, 2010. *What is CSS?*. [Online]
Available at: <https://www.w3.org/standards/webdesign/htmlcss>
[Accessed 29 08 2016].

W3C, 2014. *HTML5 specification: W3C Recommendation*. [Online]
Available at: <https://www.w3.org/TR/html5/>
[Accessed 29 08 2016].

Wagner, C., 2004. Wiki: a technology for conversational knowledge management and group collaboration.. *Communications of the Association for Information Systems*, Volume 13, pp. 265-289.

Wallace, W., 2004. Instant Messaging and Online Chat Rooms: Internet Relay Chat (IRC). In: *Steal this File Sharing Book*. 1st ed. San Francisco, CA: No Starch Press, pp. 61-67.

Wang, S. & Noe, R. A., 2009. *Knowledge sharing: A review and directions for future research.*, s.l.: Human Resource Management Review.

Weinberger, D., 2007. The real difference between the two 2.0s. *KM World*, 16(2).

West, J. & Dedrick, J., 2001. Open Source Standardization: The Rise of Linux in the Network Era. *Knowledge, Technology & Policy*, 14(2), p. 88–112.

Wikipedia, n.d. *Wikipedia:About*. [Online]
Available at: <https://en.wikipedia.org/wiki/Wikipedia:About>
[Accessed 30 08 2016].

Wilson, M. J., 2001. *Get smart with proxies and RMI*. [Online]
Available at: <http://www.javaworld.com/article/2076234/soa/get-smart-with-proxies-and-rmi.html>
[Accessed 29 08 2016].

Yang, C. & Chen, K., 2007. Can organizational knowledge capabilities affect knowledge sharing behavior?. *Journal of Information Science*, Volume 33, pp. 95-109.

Zack, M. H., 2003. Rethinking the knowledge-based organization.. *Sloan Management Review*, 44(4), pp. 37-71.

All rights on the images displayed belong to their rightful owners. No permission is given from the author of the thesis to use those images for any purpose. The code displayed cannot be used for commercial purposes without the prior written authorization from the author.

The opinions expressed do not necessarily reflect the position of CERN.