

Títol: Implementation of a social Networks aggregation platform

Volum: 1

Alumne: Daniel Ariño Pla

Director/Ponent: Juan De Bravo Díez

Maria Carme Quer Bosor

Departament: Llenguatges i Sistemes Informàtics

Data: 23/03/2009

DADES DEL PROJECTE

Títol del Projecte: Implementation of a social Networks aggregation platform

Nom de l'estudiant: Daniel Ariño Pla

Titulació: Màster en Tecnologies de la Informació (MTI)

Crèdits: 30

Director/Ponent: Juan De Bravo Díez / Maria Carme Quer Bosor

Departament: Llenguatges i Sistemes Informàtics (LSI)

MEMBRES DEL TRIBUNAL (*nom i signatura*)

President: Pere Botella López

Vocal: Ferran Sabate Garriga

Secretari: Carles Farre Tost

QUALIFICACIÓ

Qualificació numèrica:

Qualificació descriptiva:

Data:

0. Content

0. CONTENT	4
1. INTRODUCTION	9
1.1. PROJECT DESCRIPTION.....	10
1.1.1- <i>Project Context</i>	11
History and development of Social Networks	11
1.1.2- <i>SociaLuna and Social Networks</i>	15
1.1.3- <i>IOBlog and SociaLuna</i>	15
1.2. PROJECT STRUCTURE.....	16
1.3. PROJECT TARGETS	17
1.4. CONTENT OF THIS REPORT	18
2. REQUIREMENTS	21
2.1. FUNCTIONAL REQUIREMENTS.....	21
2.1.1. <i>SociaLuna</i>	21
REST Services.....	22
Social networks connectors	23
Queues management.....	23
2.1.2. <i>IOBlog</i>	23
Read messages	24
Send messages	25
Profile management.....	26
2.2. NON-FUNCTIONAL REQUIREMENTS	27
<i>Security</i>	27
<i>Usability</i>	27
<i>Attractiveness and understandability</i>	28
<i>Maintainability</i>	28
<i>Efficiency</i>	28
3. TECHNOLOGIES AND ENVIRONMENT	29

- 3.1. DEVELOPMENT TECHNIQUES..... 30
- 3.2. TECHNOLOGIES 31
 - 3.2.1. *Front-End*..... 31
 - PHP + Zend Framework 31
 - Smarty..... 33
 - JavaScript + AJAX + jquery 34
 - 3.2.2. *Back-End*..... 35
 - Restlet 36
 - Spring..... 37
 - Hibernate..... 38
 - Apache Camel + Active MQ..... 39
- 4. WORKING METHODOLOGY (SCRUM) 40**
 - 4.1. SCRUM..... 40
 - 4.2. ROLES IN SCRUM..... 40
 - 4.3. ELEMENTS OR ARTEFACTS IN SCRUM..... 42
 - 4.3.1. *Product Backlog* 42
 - 4.3.2. *Sprint Backlog* 43
 - 4.3.3. *Sprint Burn-down*..... 43
 - 4.4. THE SCRUM PROCESS 44
 - 4.4.1. *Sprint* 44
 - 4.4.2. *Sprint Planning* 45
 - 4.4.3. *Daily Meeting*..... 46
 - 4.4.4. *Sprint Review* 46
 - 4.4.5. *Sprint Retrospective* 47
- 5. PROJECT TIME-LINE 48**
 - 5.1. SPRINT 3 49
 - 5.2. SPRINT 4 50
 - 5.3. SPRINT 5 51
 - 5.4. SPRINT 6 53
- 6. ECONOMIC STUDY OF THE PROJECT 54**

6.1. HUMAN RESOURCES	54
6.2. ENVIRONMENT RESOURCES.....	55
7. SOCIALUNA FRONT-END	56
7.1. USE CASES	57
7.1.1. <i>Login</i>	57
7.1.2. <i>Networks settings use cases</i>	58
Import social networks accounts	58
Change password	66
Delete Social Network.....	66
7.1.3. <i>Settings – Create groups</i>	67
Create group	68
Update group	68
Delete group.....	69
7.1.4. <i>Show messages</i>	70
Permanent post link	71
7.1.5. <i>Messages management</i>	71
Filter messages.....	72
Search friends / Filter friends	73
Messages visualization options	73
Mark messages as favourites	76
Mark messages as read / unread	78
Tag messages.....	78
7.1.6. <i>Send new message</i>	82
Send message to multiple networks	82
Reply messages.....	84
Fast post to network.....	85
7.2. UX MODEL DESIGN	86
7.3. IOBLOG STRUCTURE	88
7.3.1. <i>Model View Controller (MVC)</i>	88
7.3.2. <i>IOBlog directories</i>	90

7.3.3. <i>PHP Controllers</i>	91
PHP Zend config.ini	92
PHP Controller Structure.....	93
7.4. VIEW CONSTRUCTION	97
7.4.1. <i>Smarty web development</i>	97
Linking PHP Controllers and Smarty templates.....	98
7.4.2. <i>JavaScript web construction</i>	99
Loading messages page.....	102
7.5. INTERNATIONALIZATION (I18N)	104
8. SOCIALUNA BACK-END	105
8.1. SOCIALUNA ARCHITECTURE.....	105
8.2. SOCIALUNA SERVERS	109
9. CONCLUSIONS	110
10. FUTURE WORK	111
ANNEX 1. FINAL RESULT AND USER'S MANUAL.....	112
ANNEX 2. OTHER SOCIALUNA APPLICATIONS	126
GLOSSARY	127
BIBLIOGRAPHY.....	131
SPECIAL ACKNOWLEDGMENTS.....	133

1. Introduction

The current document is a Master's Final Project titled "SocialLuna: Implementation of a social networks aggregation platform". The project was developed by Daniel Ariño, student of Barcelona School of Informatics in collaboration with Telefonica R&D, company. Telefonica R&D is an important company of Telefonica Group specialized in research projects in the communication sector.

This project has been led by Juan De Bravo Diez, as a company representative, and it has been advised by Carme Quer, professor of LSI department in Barcelona School of Informatics.

The aim of this project is to develop an Internet application that introduces me to the web development technologies, so this will become a solid knowledge base to the web development techniques.

1.1. Project description

This project is part of an innovation project developed by Telefonica R&D. The project's goal is to develop the SocialLuna platform, which will be an aggregator of different social networks that exist in Internet.

Because of the huge trends in social networks on the Internet, some users of these networks need to access multiple pages to get all their social activity. This fact leads to think of a platform to aggregate social networks, as a great innovation project with good prospects.

The basic idea is that a user who is registered with different social networks may get unified information of all movements on their accounts in those networks by using the platform.

The advantages of having such a platform would be:

- ◆ no need to enter the reference information on each account, that provides a clear comfort to the user by centralizing all the information in a single application;
- ◆ be able to access different accounts in these networks and see all the movements occurred in them;
- ◆ send and reply messages actively to contacts without accessing into a specific network;

Due to this, SocialLuna will become an important communication tool for users of different social networks.

Since SocialLuna will be a platform, not a final service, it is necessary to create services to offer a specific functionality to the final user. Inside the SocialLuna project will be developed a web application called **IOBlog**, such implementation will bring the e-mail paradigm to social information, showing in inbox and outbox the information generated by the user's friends and the user himself.

The project will have two development lines: the creation of the platform SocialLuna (Backend) and the development of an application to get the maximum performance of this platform, the IOBlog tool (Front-End).

My work inside the project will be in the Front-End part where I will work to create an application that, using the services of SocialLuna, shows to final user all interesting information about his or her social networks.

1.1.1- Project Context

The project context is clearly framed within the scope of social networks and the revolution that these have caused to the Internet since its emergence. This revolution has led many companies to focus their business lines into the creation of new social networks as well as the development of applications to interact with them, and then, improve and make the users navigation easier inside their habitual social networks.

Therefore, to understand the growing interest in this field, it is necessary to know more about the emergence and development of social networks.

History and development of Social Networks

[9][10] History of social networks begins in the theory of six degrees of separation which, it argues that all the people of the planet are connected by no more than six people. This theory, originally from the Hungarian writer Frigyes Karinthy, was proposed in 1929 in a short story called Chains.

In the '50s, researchers from MIT and IBM were trying to demonstrate the theory mathematically, but twenty years later of trying to solve the problem, they were not yet capable of solving to their own satisfaction.

[11] It was already 1967 when the U.S. psychologist Stanley Milgram thought in a new way to test the theory. It consisted of an experiment in which randomly selecting several people in the Midwest U.S., they had to get postcards to a stranger located in Massachusetts (several miles away). The only information known by the senders was the name, occupation and approximate location of the recipient. With these data, they had to send the package to a person who thought he could hear directly to the recipient so that it will deliver in person; this person should do the same and so on until the package was delivered personally to their final destination.

Although participants in this experiment thought that the chain of intermediaries includes hundreds of people, final results showed that each packet was delivered to the addressee at an average of between 5 and 7 brokers.

But it was not until the beginning of the twenty-first century that those theories about social networks made the leap to the Internet, and around years 2001 and 2002 the firsts networking circles of friends online began to appear. Shortly after their appearance, these networks have evolved, at the same time that they ever with a larger number of users. This mass turned social networks into a desired and ambitious product for any company which facilitated the quickly creation of social networks like MySpace (2003) **[1]**, Orkut (2004) **[2]** from Google, Yahoo 360° (2005) **[3]** of Yahoo, and many others, as Flickr **[4]** of Yahoo, Twitter (2006) **[5]**, or one of the most recent, Facebook **[6]**, which already has millions of users and a major advertising investment from Microsoft.

Currently, there are a multitude of different social networks at different levels. In this way, we can observe international networks that bring together users around the world such as Facebook **[6]**, MySpace **[1]**, Orkut **[2]**, or Hi-5 **[7]** and other networks of local or national.

Image 1, located in the next page, shows the social network with more acceptances on the basis of the geographical area in which we find ourselves, in October 2008. It should be noted that only indicates the most popular in terms of number of users but in many cases the growth trends of many networks allow foresee a change in the not too distant future.

Noting the map of social networks popularity, we can conclude that the network with more popularity in the World Wide Web is Facebook **[6]** that is notable for being the only presence among the favourite of users on all continents. Another important key of Facebook paper in the Social Networks is that is the Social Network with a greatest growth in all the continents **[13]**. Orkut **[2]** of Google stands firm in leading social networks in Brazil and India, and MySpace **[1]** continues to dominate in the U.S. In Europe, there is much difference between different social networks of smaller areas. Even so, the spectacular growth of Facebook since its birth and his growth rates as compared to other networks provide an alignment with other networks in several countries.

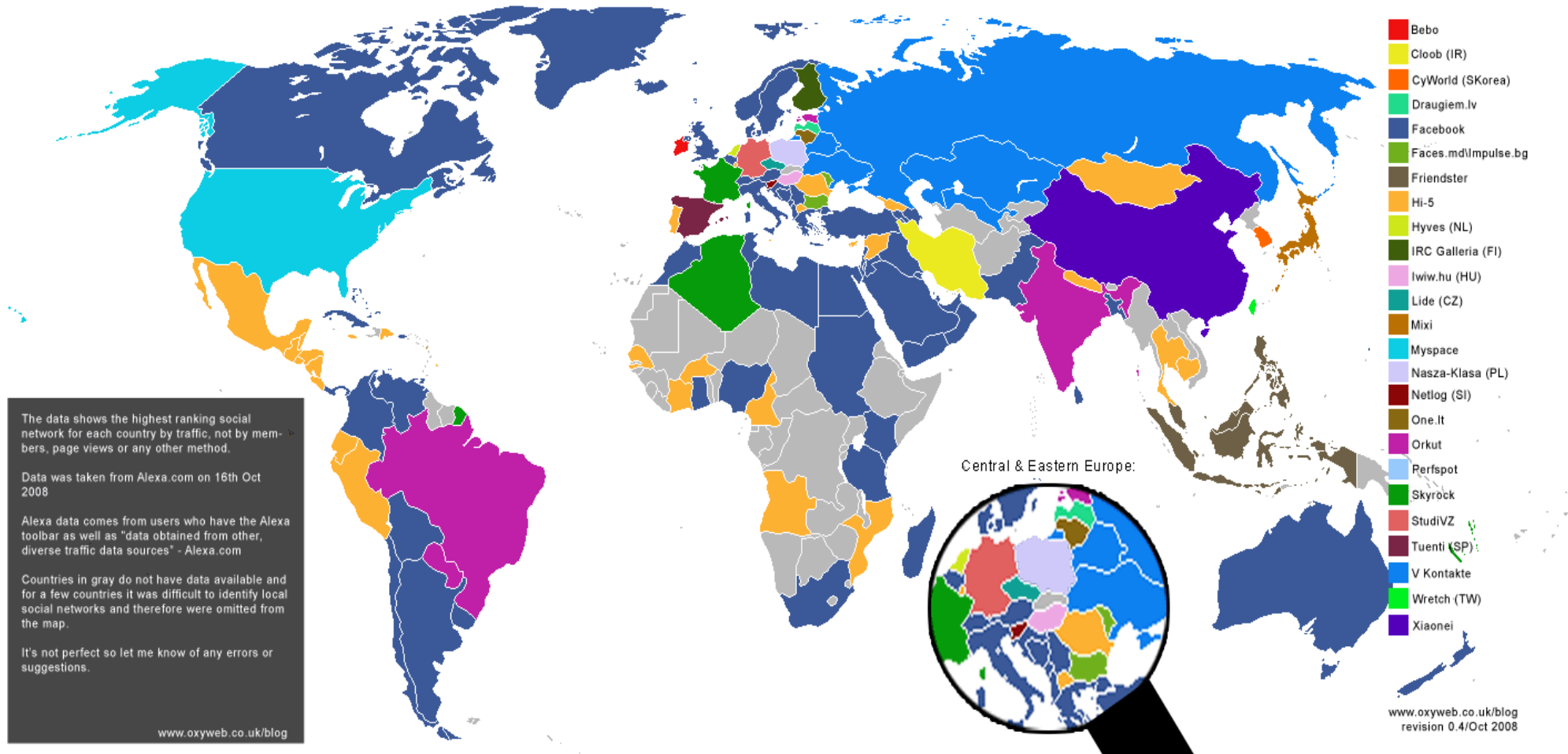


Image 1: popularity index of Social Networks on date 16th October 2008

1.1.2- SocialLuna and Social Networks

Considering the segmentation of social networks market appears SocialLuna, which aims to make easier the creation of applications that allow access to all of them under a single interface. In this way, it will help the end user of these networks to interact with them quickly and easily.

1.1.3- IOBlog and SocialLuna

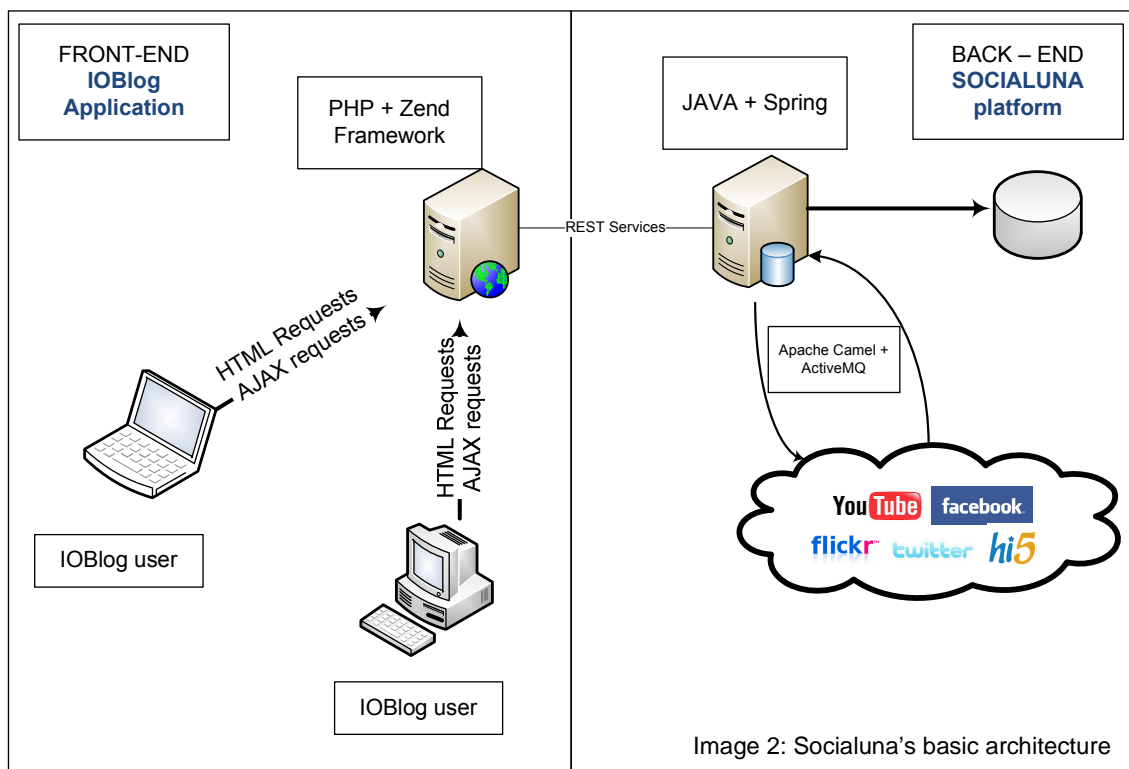
In order to bring the platform SocialLuna into a tangible product for the user, the web application IOBlog is being developed at the same time. As discussed above, this application will access to the information scattered in different social networks of the user using SocialLuna platform as a link with the different networks. With appropriate services, SocialLuna will provide all necessary information to IOBlog application, then IOBlog will submit to the user all his/her activity across his/her social networks, which has been imported into the application.

On the other hand, IOBlog will use the full potential of SocialLuna platform allowing the users to interact with various social networks, in which the user is a participant. In this way, the application will allow to send messages and comments to other networks to achieve a bilateral communication.

1.2. Project structure

SocialLuna project have a complex environment. On the one hand, it consists on the development of a platform that allows the interaction between multiple applications and external social networks. On the other hand, it also includes the development of IOBlog, a web application that aggregates different accounts in the same place.

So, from now on I will say “front-end” when I refer to IOBlog application and “back-end” when I mean SocialLuna platform.



In the project, I am mainly assigned to the development of the Front-End part. I am working in the development of a useful web application using SocialLuna services to aggregate user’s social networks accounts and proposing news services to improve the SocialLuna platform at the same time improving IOBlog application. In the next sections I will present all the project scope, detailing my work in the front-end part.

1.3. Project targets

Within developing process of the project, there are several objectives at both, the personal level and the project itself.

On a personal level, my main goal is to finish my university education process within a company of innovation as Telefonica R & D. With this project, I hope to gain professional experience and to feel comfortable with the methodology of work. At the same time, I hope to acquire knowledge about the technologies currently used in the development of web projects of this scale.

In project development, it had been established different targets through meetings with the end customer. These meetings are framed within an extreme programming methodology called Scrum, which it will be described later.

Some of these targets are:

- ◆ Achieve an integration platform for different social networks (initially it was only considered the social network Twitter and Flickr, and after it was included Youtube and Facebook).
- ◆ Get all the messages of user registered within the social networks.
- ◆ Allow sending messages to different social networks simultaneously.
- ◆ Import all contacts in all users' social networks to obtain information about them in the same web.
- ◆ Integrate IOBlog inside customer web portal.
- ◆ Develop a user-friendly interface for the IOBlog application.

1.4. Content of this report

Introduced the project context, I will enter in depth all project development phases. The project report is divided in ten chapters and two annexes:

- ◆ **Chapter 1 – Introduction:**

On this chapter I introduce the project context and I revise the state and evolution of social networks in Internet.

- ◆ **Chapter 2 – Requirements:**

Chapter 2 explains the different requirements to develop the application. On it, I emphasize on functional and non-functional requirements separating by front-end requirements or back-end requirements.

- ◆ **Chapter 3 – Technologies and environment:**

On this chapter I will enumerate the different technologies used in the development process, also separating between front-end technologies and back-end technologies.

- ◆ **Chapter 4 – Working methodology (SCRUM):**

Scrum is an agile programming management method used during all time to manage the project development. I will explain the main points of this management framework in chapter four.

- ◆ **Chapter 5 – Project time-line:**

In this chapter I will enumerate the different tasks assigned to each developing month and I will show contrast between estimated time and final time with graph used as scrum performance indicator.

◆ **Chapter 6 – Economic study of the project:**

Chapter 6 will analyze an estimation of the economic cost of all phases of project development.

◆ **Chapter 7 – SocialLuna Front-End:**

In chapter seven I will enter in depth in all my work developed during the project in the SocialLuna front-end application. I will start defining the uses cases of IOBlog application and I will continue explaining the web navigation with a UX Model Design. Once defined the web navigation I will explained the developing process linked to Zend Framework functions. On this developing process I will explain two different concepts in the web construction: the templates html creation using a template engine called smarty and the JavaScript web construction, more complex but with more facilities to manage the events of the user's navigation. Finally I will review the internationalization process to create different dictionaries to the web.

◆ **Chapter 8 – SocialLuna Back-End:**

To get a better understand of all project, is necessary explain the back-end structure of the application. In this chapter I will enter to define the back-end architecture developed by my fellow team.

◆ **Chapter 9 – Conclusions:**

Finally, I will present my personal conclusions about these months working in the project in chapter nine.

◆ **Chapter 10 – Future work:**

The latest chapter is dedicated to comment some possible features to develop in the application or explain important work to do to turn the application in a final product.

◆ **Annex 1 – Final result and user’s manual:**

The first Annex will show the final result of the work in different screenshots that will serve as user’s guide to use the application.

◆ **Annex 2 – Other SocialLuna Front-End applications:**

In the second Annex, I will comment other applications that have been carried out in the company at the same time, that use the SocialLuna platform backend.

2. Requirements

This section revises the main requirements of the project divided between the SocialLuna platform and the IOblog application. In the first point are explained the platform important use cases, and in the next section are defined the IOBlog use cases divided in functional requirements and non-functional requirements.

2.1. Functional requirements

These requirements are the set of features that will offer the whole platform SocialLuna and then they will be used by different applications, as for example IOBlog. In order to expose these requirements clearly, first of all, I will present the functional requirements of the whole SocialLuna platform and after that, the web application IOBlog requirements.

2.1.1. SocialLuna

SocialLuna platform will be a bridge between some different social networks and the web application IOBlog. Some features will be developed to manage this link. We can bring these features in three great features:

1. Creation of rests services,
2. Creation of connectors with the external social networks,
3. Management of queues

REST Services

REST services are a set of Web Services that allow communication between the Web server of the application and SocialLuna platform. In this manner whenever an application wants to interact with the platform, it will make it through requests for such services. There are several services available, a small group of these would be:

- ◆ Account information: Services that allow getting information about the user account, the user profile, or the social networks accounts imported to SocialLuna.
- ◆ In posts: These services are related with the received user messages. We can find several services that return a number of posts grouped by friends, networks, date...
- ◆ Out posts: Similar to the previous group but for sending user messages.
- ◆ Posts management: These useful services allow the application to offer to the final user some functions to manage his messages. Some functions, they can use, are selecting multiple posts as read or unread, mark them as favourites, or sending new messages and comments.
- ◆ Accounts management: These services are related to the management of the user social accounts, giving the option of import different social networks, allowing to change the user password in a network or to revalidate the token that some external social networks send to the platform to allow the offline access.
- ◆ Tagging messages: Finally, these services are for add to the platform the functions needed to assigning labels to user messages. Some services are:
 - create new labels,
 - delete labels,
 - apply labels to user posts, etc.

Social networks connectors

To allow the interaction with the external social networks that the platform shall integrate, the development of different connectors with each one is a requirement. It uses the API provided by each social network for these purposes. The first connectors to develop are the ones that will allow integrating Flickr and Twitter networks. Later on Facebook and Youtube will be added to these.

Queues management

Another important question to bear in mind is that the platform will need to be synchronized with the external networks and respects the limitations imposed by them. To manage these connections and not overcharge the social networks servers that Socialuna will access, a queues system will manage these external calls.

2.1.2. IOBlog

IOBlog application will focus on three main areas:

1. Read messages from all user imported social networks,
2. Send messages to these networks and reply or comment friend messages,
3. Create user configurations that allow the user the management of all his accounts and friends.

Read messages

An IOBlog user will have access to all his messages (send or received) in all the networks that the user has imported. In this way, within the reading of messages it can be distinguished different requirements:

- ◆ IN-blog, where it will be shown the received messages, the functionality required will be:
 - Order messages by date, user, network...
 - Filter messages to allow the user to view only the messages that he wants. Some filters that will be implemented:
 - Filter by date, network, friends...
 - Mark messages as favourites.
 - Mark messages as read or unread.
 - Reply messages.
 - Delete messages.
- ◆ OUT-blog, where it will be shown the send messages, the functionality required will be:
 - Order messages by date and network.
 - Filter messages by date and network.
 - Mark messages as favourites.
 - Delete messages.

Send messages

Another basic function to implement in a social networks aggregator is allowing the users to send messages to their social networks. IOBlog will have different ways to offer these functions to the final user:

- ◆ Fast post in a social network:
 - To send posts to the social networks that allows sending only text messages (initially Twitter). A fast post toolbar will be present in all windows of IOBlog application to allow the user send a message at the moment.
- ◆ Reply a post / comment:
 - Accessible from the IN-Blog, it will send direct replies to a message from another user or to add new comments at a photo of a friend.
- ◆ New message:
 - This option will allow sending a new message to any network that the user has imported inside our system. Another important feature will be the option of send a message to more than one network simultaneously. This new feature will provide greater convenience to the user who does not have to type the same message several times to send a message to different social networks.

Profile management

IOBlog users can manage their user profile in order to receive all the information in the manner that best suits their needs. Principal features that users can edit are:

- ◆ Import networks:

The user can import different accounts of their social networks to their SocialLuna account in order that the application will directly connect to their networks to pick all the movements that he makes.

- ◆ Page personalization:

The user can change some interface options of the page to make it more usable and suitable to their preferences. For example, it allows him to choose the number of messages per page, or to select option of being advised when making certain actions as mark messages as read.

- ◆ Groups management:

Another interesting feature is the creation of groups of friends. Users can group their contacts to obtain a useful user filter to search the messages of interested user friends. With this service, for example, users can create the group university with their university friends, or the group work...

- ◆ Friends management:

Similar to the previous options, users can create a special type of groups called friends matching. If a user has a friend in different social networks, she can create an aggregation of all friend social networks accounts. For example, user can aggregate the Flickr and twitter accounts of her friend assigning the name she wants.

2.2. Non-functional requirements

In this section, we can see some non-functional requirements that are present in all the project development. These requirements are assigned to evaluate the application under the standard ISO 9236-1.

Security

In all community applications, security is an important point to manage. In the SocialLuna's database we will have all information about users in their own social networks. This information will be public for other users or private in function of the privacy terms stated by the users in the external networks. It will be necessary to differentiate on their privacy level to show this information only to authorized users.

Another interesting point into SocialLuna security is the passwords storage in the database. The database will maintain not only the IOBlog password, but also the external social networks passwords to allow SocialLuna offline access to these networks. The passwords will be saved encrypted to protect them of possible unauthorized access.

It is also important to protect access to SocialLuna services calls; so that they cannot be accessed directly by a web a navigator call, but these will be access-controlled part of the application.

Usability

The IOBlog application, to which users will access, should be as simple and manageable as possible to facilitate navigation. All their functions must be clearly marked and visible to prevent that the user will lose through the page and decide to stop using it. In addition we must be especially careful when choosing the features that will be more useful to the user.

Attractiveness and understandability

Linked to the usability requirements, we can find the page layout or interface design. This design has to be attractive or pleasing to the user and understandable to avoid the user gets lost due to poor composition of the web page.

Maintainability

The system must be easily expandable so that in future new social networks may be added to the platform without this having a huge impact on the code. In this way, the code must be independent of the network as far as possible to facilitate possible expansions in the future. This changeability requirement is very important both in the SocialLuna platform as in the web application IOBlog.

Efficiency

Along with the model consistency, it is necessary to test the performance between client and server, several conditions in all web applications. This will perform many of the features using AJAX technology to avoid having to reload the entire page with each new user action. In addition, it will try to group small requests, like marking a message as read or as a favourite to send a joint request to the server instead of sending them one by one and unnecessarily burdening the network traffic.

3. Technologies and environment

The development environment is different in each part of the project. As we can see in image 4, the front-end is the part that corresponds to the development of a web application, and the back-end is the part that supports the communication with the social networks.

We can divide the front-end development in two parts:

- The part that is in charge of executing all the logic of the application and communications with SocialLuna platform. This part is developed using PHP and a PHP framework called Zend Framework and it is running under an apache server.
- The part that is responsible of showing the webpage to a user. This part is developed in html using a template engine called smarty and using JavaScript to minimize the number of calls to the server. To help us in the use of JavaScript we are using a JavaScript library called jquery that allows multiple useful functions to facilitate DOM management and AJAX calls.

Back-end is running under a Tomcat server and it is developed using JAVA under a framework called Spring. Mysql is the database that is used to manage all data needed by the platform. The queues are being managed by Apache ActiveMQ, a powerful open source Message Broker and Enterprise Integration Patterns provider.

The management of the versions of the software and documentation about the whole project is done using a Subversion repository. It provides a version control system used to maintain current and historical versions of files such as source code, WebPages and documentation.

All these technologies will be explained with more detail in the next sections.

3.1. Development techniques

[17] SocialLuna is developed using a TDD technique. TDD (Test-driven development) is a software development technique that uses short development iterations based on pre-written test cases that define desired improvements or new functions. Each iteration produces code necessary to pass that iteration's tests. Finally, the programmer or team refactors the code to accommodate changes. The key of TDD concept is that, preparing tests before coding facilitates rapid feedback changes.

The TDD cycle consists on the following steps:

1. Choose a new requirement: Choose the new feature to develop from the requirements list.
2. Add a test: each new feature begins with writing a test. This test must inevitably fail because it is written before the feature has been implemented.
3. Run all tests and see if the new one fails: This validates that the test harness is working correctly and that the new test does not mistakenly pass without requiring any new code.
4. Write some code: write simple code to pass the test. The code written is only designed to pass the test. To develop the code is used the principles of KISS (Keep It Simple, Stupid) that design simplicity should be a key goal and unnecessary complexity avoided.
5. Run the tests again: If pass all test cases, the code meets all the tested requirements.
6. Refactor code: Clean the code to delete duplicates without damage other features.
7. Run the tests again: Pass all tests to be confident that refactoring does not damage any other functionality.
8. Remove requirement from the list: Remove the functionality implemented from the pending requirements list.
9. Repeat: Repeat alls phases with a new requirement.

3.2. Technologies

As has explained in the previous section, multiple technologies were being used to develop the final software. In this point are explained one by one all these technologies.

3.2.1. Front-End

In this first chapter are explained all used technologies to develop the front-end application. These technologies are oriented to web development projects. They are PHP and Zend Framework, Smarty templates engine, and JavaScript language using jquery framework.

PHP + Zend Framework



PHP is a general-purpose scripting language that is especially suited for web development. PHP generally runs on a web server, taking PHP code as its input and creating web pages as output. PHP can be deployed on most web servers, many operating systems and platforms, and can be used with many relational database management systems. It is available free of charge, and the PHP Group provides the complete source code for users to build, customize and extend for their own use. **[18]**

[19] Zend framework is a set of classes focused on building more secure, reliable, and modern Web 2.0 applications & web services, and consuming widely available APIs from leading vendors. This is an open source and object-oriented framework implemented in PHP5. Some advantages that offer this framework are:

- ◆ MVC: Zend framework includes multiple classes to allow developer and web designers to their concerns and skills, making code implementation and design easily and clearly separated.

- ◆ Internationalization (i18n): Zend framework also includes some classes to localize a web application for a particular language and culture.
- ◆ Authentication, authorization and session management: Customizing data and protecting from access by unauthorized users
- ◆ Web and Web Services: Web services are an integral part of Zend Framework and this intend to be the nexus for an entire eco-system of Web Services and APIs providers.

Why this framework?

- ◆ Zend Framework helps the web developing using the MVC pattern.
- ◆ It offers multiple API's to connect with different social networks, ideal for a project of these characteristics.
- ◆ This framework is in an advanced and stable version (v 1.7).
- ◆ It is free, important in a project of a R&D Company.

Other similar frameworks in the market

- ◆ Symfony Framework - <http://www.symfony-project.org/>
- ◆ Kumbia PHP Framework - <http://www.kumbiaphp.com/blog/>
- ◆ Agavi - <http://www.agavi.org/>

Smarty



[20] Smarty is a template engine for PHP that facilitates a manageable way to separate application logic and content from its presentation. This is best described in a situation where the application programmer and the template designer play different roles, or in most cases are not the same people.

On this case, the designer can create the html labels in smarty templates using the variables accorded with the programmer. If one day the programmer need to change the application logic, this changes does not affect the template designer and the content will still arrive in the template exactly the same. Likewise, if the template designer wants to completely redesign the templates, this would require no change to the application logic.

In other hand, smarty is very fast and only compiles once. It is smart about recompiling only the template files that have changed and allow the easily create your own custom functions and variable modifiers, so the template language is extremely extensible. These and other features make Smarty a usefull engine to develop web applications.

Why Smarty?

- ◆ It helps to future changes in the view layer without affect to other layers.
- ◆ It is very fast and flexible.

Other similar templates engines in the market

- ◆ Savant - <http://phpsavant.com/>
- ◆ bTemplate - <http://www.massassi.com/bTemplate/>
- ◆ ETS (Easy Template System) - <http://ets.sourceforge.net/>

JavaScript + AJAX + jquery



[21] JavaScript is a scripting language widely used for client-side web development. It is a dynamic, weakly typed, prototype-based language with first-class functions. JavaScript was influenced by many languages and it was designed to look like Java, but be easier for non-programmers to work with. The use of JavaScript in IOBlog is very important and some pages are being created using this language in client-side. The reason that impulse to develop some pages of the application in JavaScript is the easily to introduce AJAX in the page.

```
{
  "firstName": "John",
  "lastName": "Smith",
  "address": {
    "streetAddress": "21 2nd Street",
    "city": "New York",
    "state": "NY",
    "postalCode": 10021
  },
  "phoneNumbers": [
    "212 555-1234",
    "646 555-4567"
  ]
}
```

code 1: JSON Example

[22] AJAX (asynchronous JavaScript and XML is a group of interrelated web development techniques used for creating interactive web applications or rich Internet applications. With Ajax, web applications can retrieve data from the server asynchronously in the background without interfering with the display and behaviour of the existing page. To do this data transfers between server and client we are using the JSON (JavaScript Object Notation). **[23]** The JSON format is often used for transmitting structured data over a network connection in a process called serialization. Its main application is in Ajax web application programming, where it serves as an alternative to the use of the XML format.

[24] To help us in JavaScript developing we are using a JavaScript framework called jquery. This library includes multiple functions that simplify HTML traversing, event handling, Ajax interactions and animating.

Why jquery framework?

- ◆ JQuery is a very extensive JavaScript library that simplifies the developing process.
- ◆ This library adds different animations to be used in web pages.

Other similar JavaScript frameworks in the market

- ◆ Prototype - <http://www.prototypejs.org/>
- ◆ Mootools - <http://mootools.net/>
- ◆ Dojo - <http://www.dojotoolkit.org/>

3.2.2. Back-End

Next of explain the Front-End technologies I will comment the technologies used to developed all Back-end functionalities. These technologies are Java language + Spring Framework to develop the platform, Restlet technology to translate the URL calls to java functions, Hibernate is used for the communication with data base. Finally, Apache Camel and Active MQ are used to manage the different queues that manage the connectors with external networks.

Restlet



[25] Restlet is an open source REST framework for the Java platform. This framework provides a concrete solution to build solid applications following the REST architecture style.

Representational state transfer (REST) is a style of software architecture for distributed hypermedia systems such as the World Wide Web.

The next image shows typical web architectures from a REST point of view. In it, ports represent the connector that enables the communication between components which are represented by the larger boxes. The links represents the particular protocol (HTTP, SMTP, etc.) used for the actual communication.

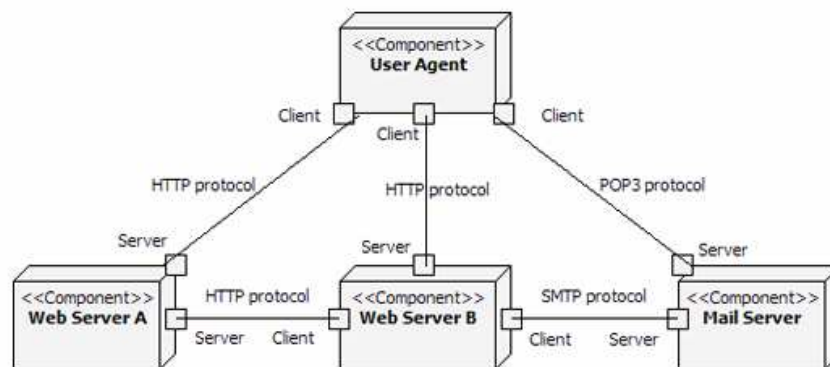


Image 3: REST Architecture

Why Restlet framework?

- ◆ It simplifies the translation from web services URL calls to Java functions.
- ◆ REST architecture has many advantages against SOAP architecture, as it is easy to build, lightweight (not a lot of xml mark-up) and the results are easy readable.

Other similar REST frameworks in the market

- ◆ Jersey - <https://jersey.dev.java.net/>

Spring



The Spring Framework is an open source application framework for the Java platform. By design, the framework offers a lot of freedom to Java developers yet provides well documented and easy-to-use solutions for common practices in the industry.

Some important features of Spring Framework are:

- ◆ **A flexible MVC web application framework** highly configurable that accommodates multiple view technologies. Spring middle tier can easily be combined with a web tier based on any other web MVC framework, like Struts.
- ◆ **A JDBC abstraction layer** that offers a meaningful exception hierarchy, simplifies error handling, and greatly reduces the amount of needed to write. The JDBC-oriented exceptions comply with Spring's generic DAO exception hierarchy.
- ◆ **Integration with Hibernate:** in terms of resource holders, DAO implementation support, and transaction strategies.

Why Spring framework?

- ◆ Adds a simplified layer against more complex technologies like JMS messages, data base access etc.
- ◆ Spring framework simplifies the application of MVC pattern to the web application.
- ◆ It is highly configurable and easy to use with a lot of different technologies (like hibernate).
- ◆ It allows the use of DI (Dependency Injection) pattern, resulting in more modular and testable system.

Other similar Spring frameworks in the market

- ◆ Aurora - <http://www.auroramvc.org/>
- ◆ Apache Struts - <http://struts.apache.org/>

Hibernate



[26] Hibernate is an object-relational mapping (ORM) library for the Java language, providing a framework for mapping an object-oriented domain model to a traditional relational database. Hibernate solves Object-Relational impedance mismatch problems by replacing direct persistence-related database accesses with high-level object handling functions.

Hibernate's primary feature is mapping from Java classes to database tables (and from Java data types to SQL data types). Hibernate also provides data query and retrieval facilities. Hibernate generates the SQL calls and relieves the developer from manual result set handling and object conversion, keeping the application portable to all supported SQL databases, with database portability delivered at very little performance overhead.

Hibernate provides transparent persistence for Plain Old Java Objects (POJOs). POJO is used to emphasize that the object in question is an ordinary Java Object, not a special object, and in particular not an Enterprise Java Bean. The only strict requirement for a persistent class is a no-argument constructor, not compulsorily public.

Hibernate provides a dirty checking feature that avoids unnecessary database write actions by performing SQL updates only on the modified fields of persistent objects.

Why Hibernate framework?

- ◆ It simplifies the mapping of attributes between a relational database and an object oriented application.
- ◆ It avoids the use of SQL queries and all related data base access common errors, working directly with objects instead of relational tables.

Other similar frameworks in the market

- ◆ iBATIS - <http://ibatis.apache.org/>

Apache Camel + Active MQ



[27] Apache Camel is a Spring based Integration Framework which implements the Enterprise Integration Patterns with powerful Bean Integration.

Apache ActiveMQ is the most popular and powerful open source Message Broker and Enterprise Integration Patterns provider. Apache ActiveMQ is fast, supports many Cross Language Clients and Protocols, comes easy to use Enterprise Integration Patterns and many advanced features while fully supporting JMS 1.1 and J2EE 1.4. Apache ActiveMQ is released under the Apache 2.0 License.

Some interesting features that ActiveMQ offers to SocialLuna Project are:

- ◆ Supports a variety of Cross Language Clients and Protocols from Java, C, C++, C#, Ruby, Perl, Python, PHP;
- ◆ REST API to provide technology agnostic and language neutral web based API to messaging;
- ◆ Supports very fast persistence using JDBC along with a high performance journal

Why Apache Camel and ActiveMQ?

- ◆ They help to manage the queues to communicate with external networks.
- ◆ They facilitate the control functions creation to control external errors.
- ◆ They add business logic to send and manage messages between the different parts of the system.

Other similar frameworks in the market

- ◆ FUSE - <http://fusesource.com/products/enterprise-camel/>

4. Working methodology (SCRUM)

[14][15][16] Into the project developing process, we are following an extreme programming methodology. This methodology is called Scrum and is explained in detail in this chapter.

4.1. Scrum

Scrum is an agile methodology/framework mainly oriented to the project management. This management is not based on monitoring a plan, but it is based on the continuous **adaptation to the circumstances of the project evolution**. Scrum conforms to the principles of the agile manifesto because:

It is an adaptable development method.

It is oriented to persons rather than processes.

It uses agile development: iterative and incremental.

4.2. Roles in Scrum

Within a process we can differentiate various roles that are distinguished primarily on set of responsibilities assigned. These roles are:

- ◆ **Scrum Master:** The Project Manager usually assumes this role and he is responsible for ensuring the smooth operation of the project, he is also responsible to:
 - Protect the team so it can work.
 - Improve the team's productivity and facilitate the creativity and access to necessary resources.
 - Facilitate a close collaboration of all the roles and functions, eliminating the potential barriers.
 - Fix the impediments that arise in individual sprints.
 - Ensure that the agile practices bring value to the team and the project.
 - Teach the customer how to achieve his goals using Scrum.

- ◆ **Product Owner:** It represents the people that are interested in the product. Usually is a person who works in the client company, which hires the product and owns the final product, but also this role can be assumed by a person's development team. It is responsible for:
 - The Product Backlog (view point 1.3.3 Artifacts of Scrum) where is defined in business language, and in a prioritized way, everything that is expected from the product (requirements, features, deliveries...).
 - Agreeing with the team and the Scrum Master the dates of the releases and its contents.
 - Look that during the development of Sprint, the requirements of the Sprint in progress cannot be changed.
 - Deciding whether the goal has been achieved or not when the Sprint finishes.
 -
- ◆ **Team members:** It is a group responsible for developing the project. This team is made up with about 2 to 7 members. When highly specialized profiles are needed within the team, there is a search and an integration process to train a person so he would be able to become a new member of the team. The responsibilities' team are:
 - The Sprint backlog.
 - Select the items of the Product Backlog that need to be done in the next iteration and estimate the effort with the Product Owner and the Scrum Master.
 - Define the goal of each Sprint with the Product Owner and the Scrum Master.
 - Divide the items chosen for the Sprint in tasks of technical working with granularity one-man-day.
 - Show the results to Product Owner at the end of the Sprint.

- ◆ **Agile Coach:** Supports the smooth functioning of the project and the use of agile practices that bring value to the team and the project.

4.3. Elements or artefacts in Scrum

There are two basic devices in a process of development through the Scrum methodology. These are the product backlog and the sprint backlog which will have an important role at various points of the iteration.

4.3.1. Product Backlog

The Product Backlog is the most fundamental artefact in Scrum as it directs what is built. The most effective and efficient team will fail if it builds the wrong system. The Product Backlog consists in a prioritized list of functional and non-functional requirements (new functionalities, improves, technology, bug reports...) along with a high level estimation for the amount of effort needed to turn each requirement into a complete element of the system. It represents everything that customers, users, or, interested people expect from the product. Any work to be done by the team has to be reflected in the backlog and it never gives completely; it is continually growing and evolving.

To start the development, it is necessary to have a vision of the goals the team wants to achieve with the product; this vision should be understood and known by all the team, and also, they need to have sufficient elements in the product backlog to carry out the first Sprint. The basic format of a product backlog should contain the following information:

- ◆ Unique identifier of the functionality or work.
- ◆ Functionality description.
- ◆ Prioritized system.
- ◆ A temporary estimation of the cost.

4.3.2. Sprint Backlog

The Sprint backlog is a list of tasks that defines a team's work for a Sprint. The list emerges during the Sprint planning. The tasks on the Sprint backlog are what the team has defined as being required to turn committed Product Backlog items into system functionality. Each task identifies who the responsible is for doing the work and the estimated amount of work remaining on the task on any given day during the Sprint.

It is useful because it breaks down the project into tasks of an adequate size for assessing progress on a daily basis, and it identifies risks and problems without the need of complex management.

4.3.3. Sprint Burn-down

The Sprint Burn-down chart gives the team a daily indication of their velocity and progress against the work they have committed to for the current Sprint.

For a Sprint in progress, the line shows the total of Work Remaining for all Sprint Backlog Items in this Sprint. The trend lines give an indication of whether the Sprint will achieve its objectives on time or not.

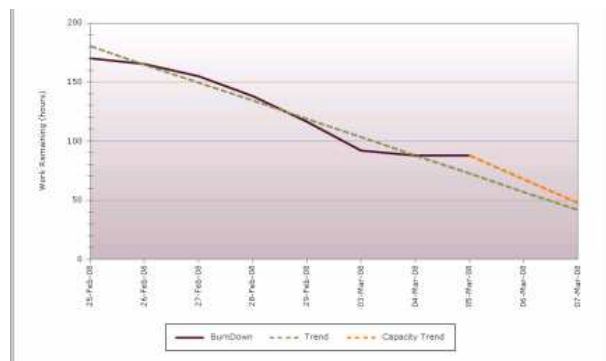


Image 4: Sprint Burn-down

For a Sprint that is complete, the chart shows how progress was viewed historically without a Capacity trend line.

4.4. The Scrum process

When all the involved roles are known in the process of Scrum, a person can begin to define and to develop the entire process and the steps that pass along its route.

4.4.1. Sprint

Sprint is a Scrum's term for an iteration which is a time-boxed period of time, typically 2 to 4 weeks, during which the Team works to turn the selected Product Backlog items it has selected into an increment of the potentially shippable product functionality.

The goal and activities for the Sprint are planned at the beginning of each Sprint in the Sprint Planning meeting. At the end of the Sprint, the team demonstrates what they

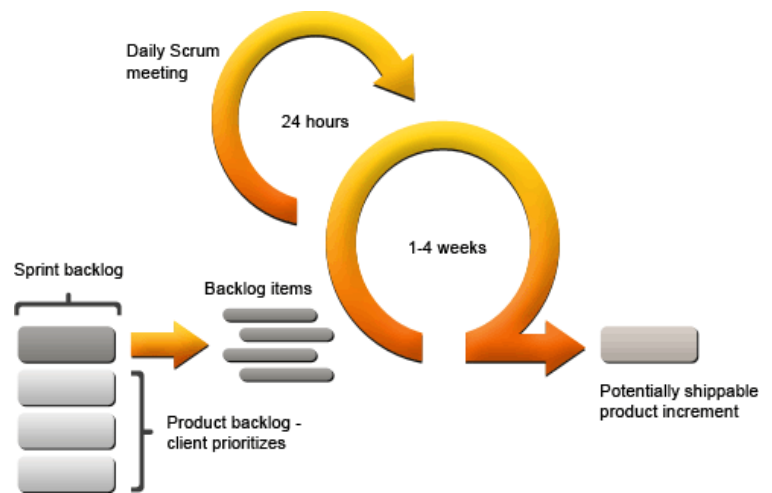


Image 5: Sprint cycle

had built in the Sprint Review and they analyze their own performance and decide how they can improve in the Sprint Retrospective.

The progress of the Sprint is tracked with the Sprint Backlog and the Sprint Burn-down chart.

4.4.2. Sprint Planning

The Sprint planning is an initial meeting of the whole team with the Scrum Manager and the Product Owner. At this meeting, the product owner presents the product backlog and the team selects what it believes it can build during the Sprint.

The Sprint Planning Meeting consists of two parts, each one lasting up to four hours:

- ◆ **Backlog selection:** the Product Owner presents the highest priority backlog to the development team. They decide how much can be turned into an increment potentially shippable of the product functionality during the next Sprint. The team will carry out the necessary questions, seek clarifications and make proposals, suggestions and amendments and alternatives which may involve changes in the Sprint Backlog. Once everything is cleared up, the team will define what the goal of the Sprint is, or in other words, which is the value that it will provide the product. The result of this first meeting is the Sprint Backlog.
- ◆ **Sprint workload planning:** the team defines the architecture and design of the functionality that it has selected, and then it defines the work, or tasks, to build that functionality during the next Sprint. When the tasks are defined, the team members catch these tasks having considerations such as their knowledge, interest and homogeneous distribution of labour. This second part should be seen as a team meeting in which all members must be there and they should divide the work into tasks, assigns and estimations. The Scrum Master works as driver or moderator.

4.4.3. Daily Meeting

Daily Meeting consists in a short status meeting that is time-boxed to 15 minutes and that is held daily by each Team. During the meeting the Team members synchronizes their work and progress and report any impediments to the Scrum Master for removal. In it, each member of the group should exhibit these three issues:

- ◆ Task where he worked yesterday.
- ◆ Task or tasks that will work today.
- ◆ If he is going to need something special or a disability expected to do their jobs.

At the end of the meeting, the Scrum Master begins to manage the possible needs or impediments identified.

4.4.4. Sprint Review

Sprint Review is a meeting that takes place when the Sprint is over. In this meeting, the team shows to the Product Owner, clients and customers the improvements that had been developed during the Sprint. With it, the Product Owner gets a review of the progress of the system and sees the increase in operation, in other hand, the team gets feedback, the key to evolve and provide value to the Product Backlog.

This is an informal meeting with informative purposes. It is not aimed to make decisions or criticize the increase. At the end of the meeting, the Scrum Master will close the date for the next Sprint planning.

4.4.5. Sprint Retrospective

This is a meeting facilitated by the Scrum Master at which the Team discusses the just concluded Sprint and it determines what could be changed so it might make the next Sprint more enjoyable and productive. The Sprint Review looks at "What" the team is building whereas the Retrospective looks at "How" they are building.

Anything that affects how the team builds software is open for debate, this might include: processes, practices, communication, environment, artefacts and tools.

This meeting is an important development and continuous improvement in the mechanism of the project cycle of life.

5. Project time-line

As I have explained in previous chapter, the project has been divided in Sprints of one month. After each Sprint the project has an empty week to finish pending tasks, fix known bugs, do the Sprint Review and the Sprint Retrospective and prepare the new Sprint (Sprint planning). In the next points I will show the tasks assigned to me in all the Sprint planning's. I will start by the Sprint 3 because is the Sprint where I enter in the team project. Nevertheless I haven't Sprint burn-down graph of this Sprint because it was a different Sprint with team summer holidays that prevented an effective Sprint monitoring.

5.1. Sprint 3

The Sprint 3 was between 15th July and 31st August. This sprint was longer than normal because the dates were in summer holidays and the development team was not complete.

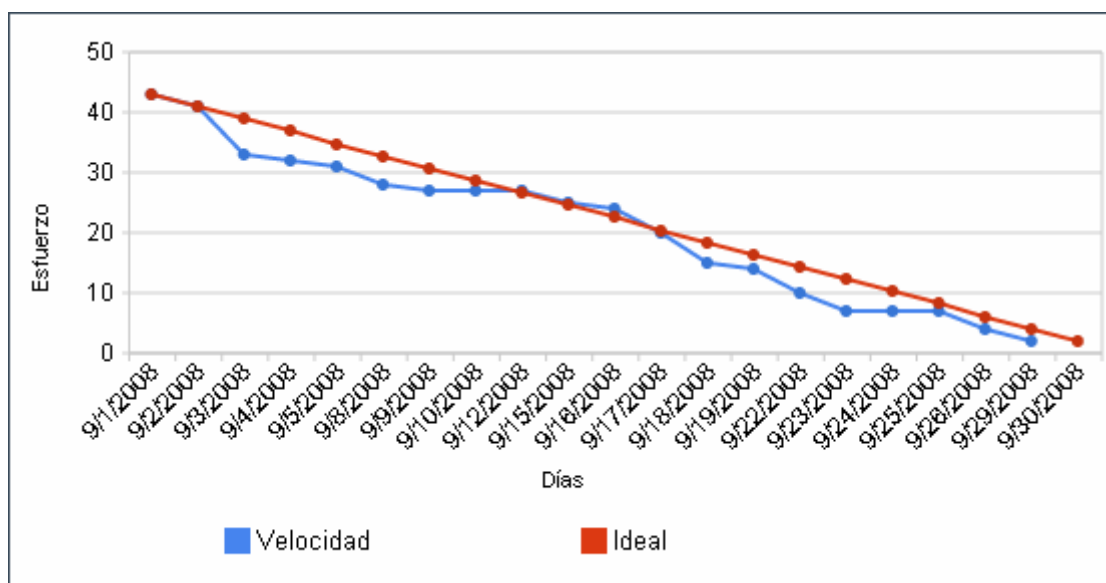
Task	Days estimated	Status
Study used front-end technologies	5	Complete
Training examples smarty	2	Complete
Create teaser page with smarty	2	Complete
Transform inblog design to smarty template	2	Complete
Obtain and show messages in a table	3	Complete
Import twitter and Flickr networks	3	Complete
Create filters	2	Complete
Training JavaScript examples	5	Complete
Transform html messages table to JavaScript object	5	Complete

5.2. Sprint 4

The fourth Sprint was between 1st September 2008 and 30th September 2008. My tasks on this Sprint were:

Task	Days estimated	Status
Front-end of send message to networks	3	Complete
Upload files (images & videos)	2	Complete
Show messages comments on Inblog	2	Complete
Error control in front-end	2	Complete
Mark all messages as read	1	Complete
Basic left menu on Inblog / Outblog	2	Complete
Friends searcher	2	Complete
Translation and internationalization	3	Complete
Import networks settings to smarty	1	Complete

The final result of all the spring (including front-end and back-end) was paint the next sprint burn-down graph:

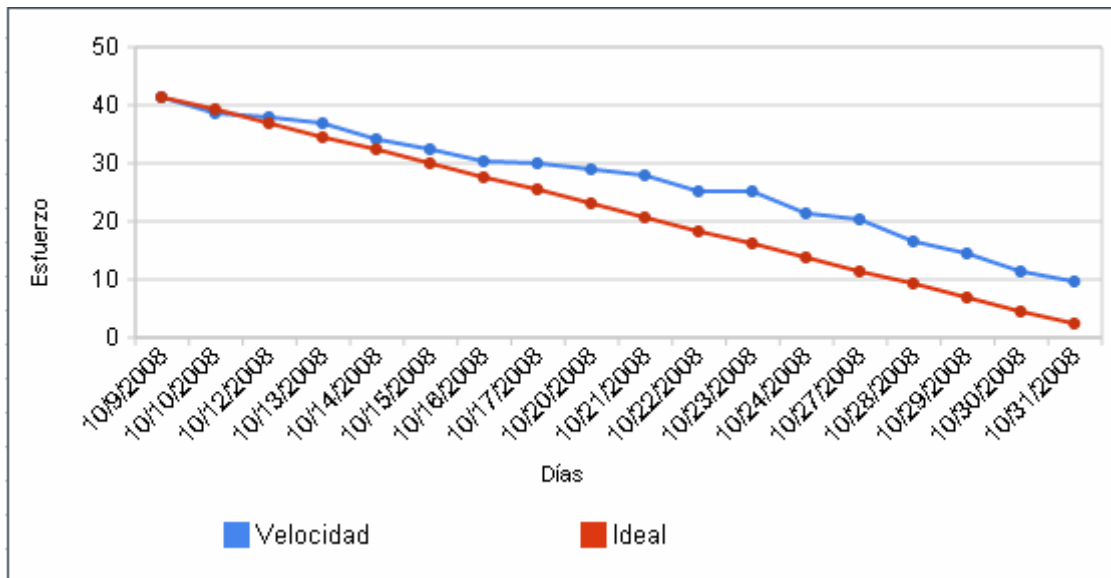


5.3. Sprint 5

Sprint five was developed between 9th October and 31st October from 2008. On this Sprint, the main features were the inclusion of three new social networks (youtube, facebook and hi5). Finally hi5 can not be added because the API was not allows all the requirements needed for SocialLuna platform:

Task	Days estimated	Status
Add facebook to import networks	1	Complete
Show facebook images and status on inblog	2	Complete
Add facebook to new message	2	Complete
Add hi5 to import networks	1	Impeded
Show hi5 images and status on inblog	2	Impeded
Add hi5 to new message	1	Impeded
Add youtube to import networks	1	Complete
Show youtube videos on inblog	2	Complete
Add youtube to new message	1	Complete
Changes on user profile page	2	Complete
Add message permalink page	2	Pending
Change the inblog page from smarty to JavaScript / AJAX	4	Complete

And the sprint burn-down graph of all team components is the shown in the image below:



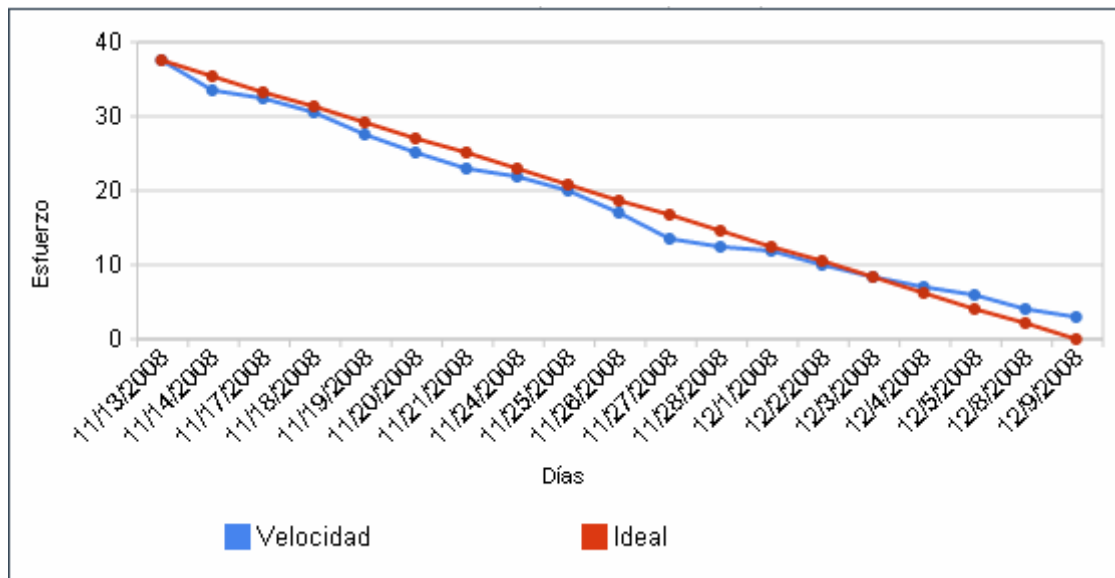
In the graph we can see that the team speeds were a little down of the ideal to the sprint planned. This is because the time spent on studying the hi-5 API was not productive in the development of the project and the tasks were blocked.

5.4. Sprint 6

The sprint six was developed between 13th November and 10th December of 2008.

Task	Days estimated	Status
Change youtube authentication from password to Auth_Sub	2	Complete
User Profile – Groups creation	4	Complete
Tagging messages	5	Complete
User Profile – Friends aggregation	4	Complete
Concurrence control in AJAX queries	2	Complete
End message Permalink page	1	Complete

The Sprint burn-down from the last Sprint was the following below:



6. Economic study of the project

In the next point I will summarize a brief study of the economic impact of the project. I will divide the economic study in two points: human resources and environment resources.

6.1. Human resources

The project SocialLuna was developed by four programmers and a project manager. Taking as reference the next hour prices by hour:

Worker	Price/hour	Resources	Hours/week	Hours/month	Total price
Programmer	20 €/ h	4	40 x 4 = 160 h	160 x 4 = 640	12800 €
Designer	20 € / h	0.5	20 h	20 x 4 = 80	1600 €
Project manager	35 € / h	1	40 h	40 x 4 =160	5600 €
Total month	-		-	-	20000 €
Total(4 months)	-		-	-	80000 €

The human resources project cost is around 20.000 € by month and the project was develop in four months. The total human costs approximation is **80.000 €**

6.2. Environment resources

The environmental resources cost is divided between the licenses cost of software and technologies used into project development process and the cost of computers and servers used to maintain and work in the project. I will assume the cost of the computers null because is company material shared with all the projects developed in the company.

In the table below I show the licenses used for each technology and the prize of its.

Software	License	Price
Apache Server	Apache License	0 €
Apache Tomcat Server	Apache License	0 €
Apache Camel	Apache License	0 €
SVN	Apache License	0 €
Tortoise SVN	GNU GPL	0 €
Hibernate	GNU GPL	0 €
Maven	Apache License	0 €
MySQL	GNU GPL	0 €
Spring	Apache License	0 €
jQuery	GNU GPL + MIT	0 €
Zend Framework	BSD License	0 €
NetBeans	CDDL	0 €
Eclipse	Eclipse Public License	0 €
Total		0 €

7. SocialLuna Front-end

In this section I will explain all the web application created using the SocialLuna platform. I will start with the design of the application use cases and I will continue explaining the most interesting parts of my work that has been developed in this part. After use cases I will show the web navigation of the application using a UX Model Design that illustrates the web page structure. After that, I will explain the different ways used to develop the pages starting by the MVC pattern and Zend Framework facilities into this way to create the web controllers. Finally I will explain the two different ways of create the views of the webpage: html templates using Smarty and JavaScript web construction.

7.1. Use cases

This section tries to expand all use cases that a user needs using IOBlog. I divide these use cases in blocks to obtain a better comprehension. The basic use cases blocks are:

- ◆ Login
- ◆ Settings
- ◆ View messages
- ◆ Messages management
- ◆ Send messages

7.1.1. Login



Image 6: Login Use Case

A non logged user enters to IOBlog application using a username and password. The authentication process access SocialLuna’s Data Base or the internal Company LDAP.

Actor – user	System
1 -User enters his/her username and password	2- The system authenticates the user and... 2.1 – System redirect user to home page if authentication is success. 2.1b – System alerts to the user that authentication is failed.

7.1.2. Networks settings use cases

These are the use cases that will allow users edit his social networks inside SocialLuna platform. These use cases allow:

- ◆ Import new networks to IOBlog application.
- ◆ Remove networks from IOBlog application.
- ◆ Modify networks in IOBlog application.

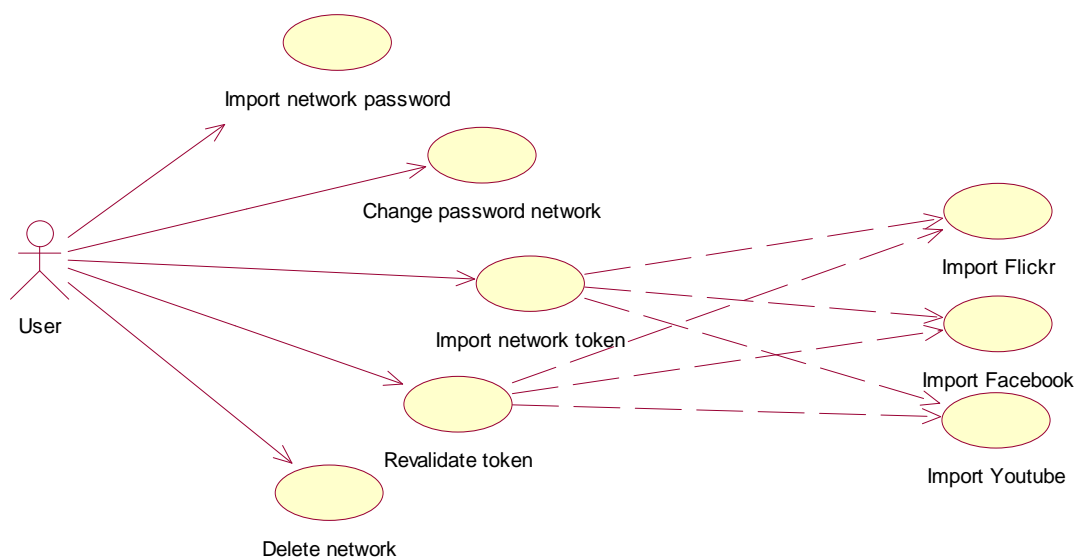


Image 7: Networks Use cases

Import social networks accounts

A user imports a personal social network to IOBlog to allow SocialLuna obtains his/her messages and show them in the application. Depending on the social network to import, these are two different import use cases.

- ◆ **Password Authentication**: SocialLuna only needs the username and password of the user into his social network. This case is less used because social networks try to avoid that users give their passwords to external applications. Only twitter API uses the authentication with username and password. Youtube allows authentication with password and token but is highly recommended the use of token authentication.

Actor – user	System
1- User enters the username and password of his social networks and click import.	2 – System saves this information to connect to social network. 3 – System redirects user to import page with a successfully message.

- ◆ **Token Authentication**: Another way to authenticate a user in a social network is getting a token from the social network that allows the application the access to the user account. According to the social network and the different APIs from each social network the authentication process is something different:

[28] Flickr authentication:

1. IOBlog is registered in Flickr as a Web Application that uses Flickr service. Flickr assigns to IOBlog an api_key (ID from application) and a shared key to allow the communication between Flickr and the web application.
2. When users try to import Flickr, IOBlog redirect their to Flickr authentication passing as URL parameters the api_key and the permissions to request (read, write, delete).
3. When the user enters in Flickr, they must give permission in Flickr to allow IOBlog application to access to his account.
4. Flickr redirects users to the web application returning a frob (a session key of one use only).
5. Using Flickr API, IOBlog transform this frob in a non-expire token (infinite session).
6. This token will grant SocialLuna the access to user's Flickr account to obtain new feeds.

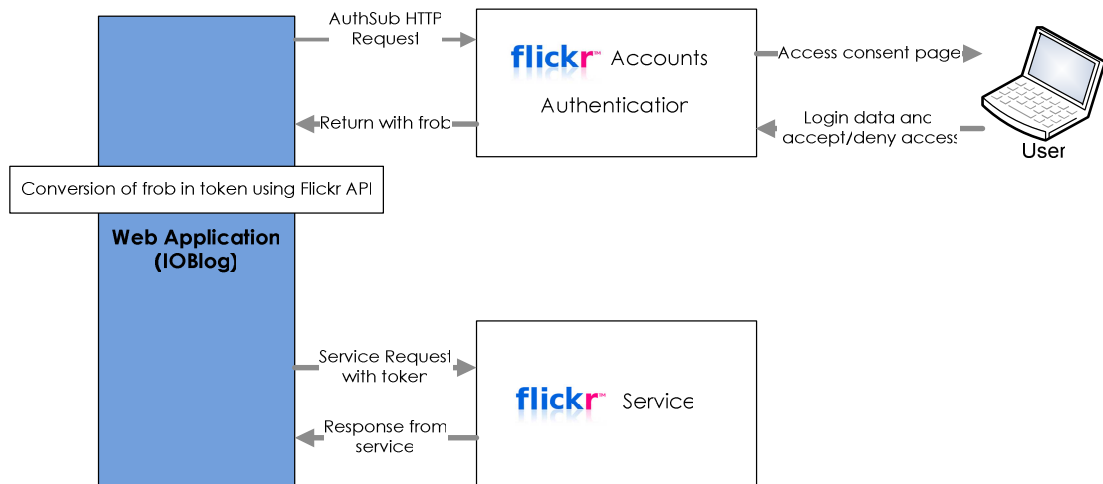


Image 8: Flickr web authentication

Actor – User	System	Actor - Flickr
1- User clicks in "go to Flickr to import my account".		
	2 – System redirects user to Flickr authentication page.	
3- User enters his username and password in Flickr.		
		4 – Flickr shows a message to allow IOBlog the access to his account.
5- User click in allow permissions.		
		6 – Flickr redirects to IOBlog returning a frob.
	7 – IOBlog transforms this frob in an infinite token using Flickr API and stores it.	
	8 – IOBlog redirects user to import page and show the message "Import successfully done".	

Alternative flow:

- 5 – User doesn't allow permissions to web application.
- 6 – Flickr redirects to call-back URL without frob.
- 7 – System ends without importing Flickr account.

[29] Facebook authentication:

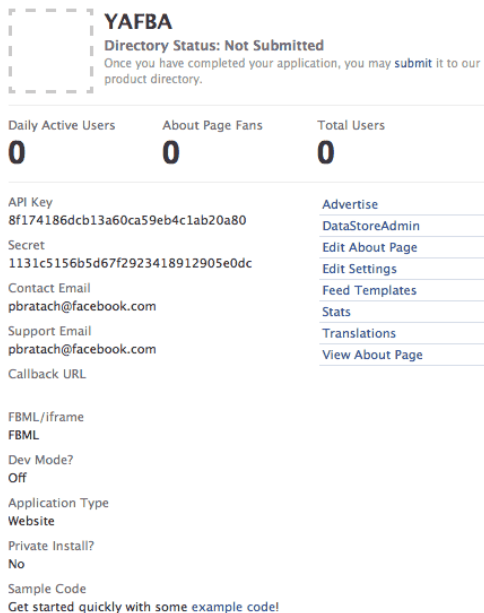


Image 9: Facebook application example

1. Like Flickr, to access Facebook accounts is previously necessary to register the web application and obtain an application ID and secret key.
2. The firsts steps into facebook authentication are similar to the steps seen in Flickr: the user goes to the Facebook page, logs in and accepts or denies the access to his account to an external web application.

3. The difference between other social networks and Facebook is that this last is most restrictive with the permissions granted to external applications and the basic permission allows access only while user is logged into facebook network. **[30]** To get other permissions is necessary to redirect the user to facebook to demand for extended permissions. IOBlog application needs three different extended permissions to a complete functional work:

- i. Offline access: this extended permission allows SocialLuna Platform to access the user account when he or she is not logged in. This is the basic permission needed and the only one demanded for the import use case.
 - ii. Status Update: extended permission needed to send an update of user status from an external application.
 - iii. Image Upload: extended permission needed to send a new image to facebook from an external application.
 - iv. Status Update and Image Upload will be demanded when a user tries to send an image or update the status from IOBlog and these permissions are not already granted.
4. When a user allows basic permissions to IOBlog, this redirects the user again to facebook to obtain offline access needed to update user feeds.
 5. With the user permissions, IOBlog sends the token to SocialLuna platform to store it and redirects the user to import page showing a confirmation message.

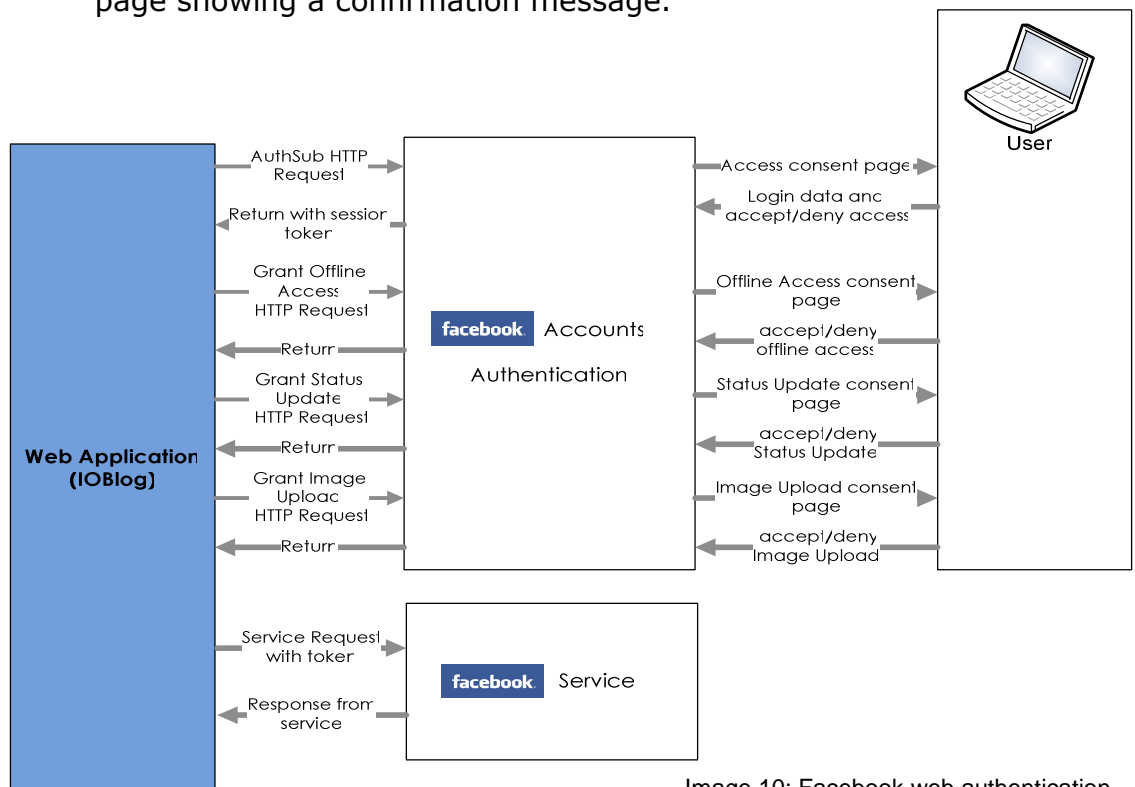


Image 10: Facebook web authentication

Actor – User	System	Actor - Facebook
1- User clicks in "go to facebook to import my account".		
	2 – System redirects user to facebook page.	
3- User enters her username and password.		4 – Facebook shows a message to allow SocialLuna the access to her account.
5- User clicks in allow permissions.		
	7 – IOBlog redirects user to facebook and demand offline access.	6 – Facebook redirects to IOBlog returning a session token.
9 – User allows offline access to IOBlog.		8 – Facebook shows the alert to demand offline access
	11 – IOBlog sends the token to SocialLuna to save the imported account	10 – Facebook redirects to IOBlog.
	12 – IOBlog redirects user to import page and show the message "Import successfully done".	

Alternative flow:

5b, 9b – User doesn't allow permissions to web application.

6b – Facebook redirects to call-back URL without session token.

7b – System ends without import facebook account.

[31] Youtube authentication:

Youtube authentication is similar to Flickr authentication. Youtube authentication uses the standard authentication authSub. The main difference with other authentication methods is that it is not necessary to register the application in Google accounts although it is advisable. Google offers three levels of registration to external web applications:

- ◆ Unregistered: application is not recognized by Google. The Access Request page, which prompts web application users to either grant or deny access for the application, displays a caution highlighted in yellow: "Non-registered, not secure. This website has not registered with Google. We recommend that you continue the process only if you trust this destination."
- ◆ Registered: application is recognized by Google. The Access Request page displays this warning: "Registered, not secure. This website is registered with Google to make authorization requests, but has not been configured to send requests securely. We recommend you continue if you trust this destination."
- ◆ Registered with enhanced security: registered applications with a security certificate on file can use secure tokens. The Access Request page displays this message: "Registered, secure. This website is registered with Google to make authorization requests."

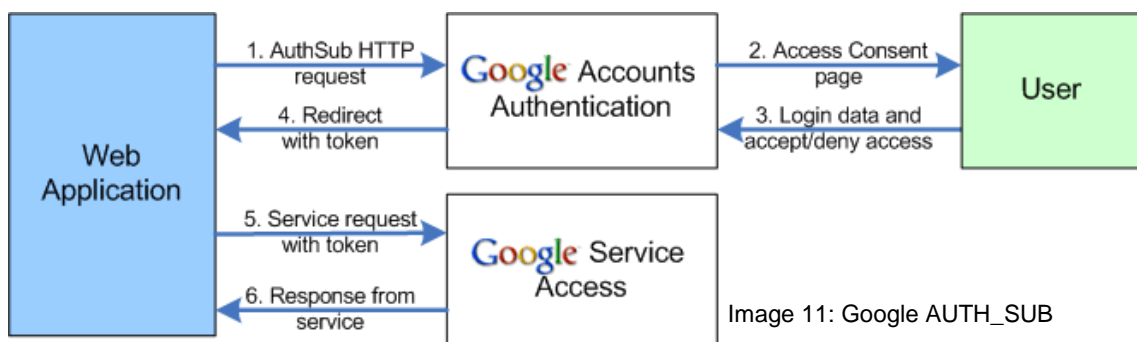


Image 11: Google AUTH_SUB

Actor – User	System	Actor - Youtube
<p>1- User clicks in "go to youtube to import my account".</p>		
	<p>2 – System redirects user to youtube authentication page.</p>	
<p>3- User enters his username and password in youtube.</p>		
		<p>4 – Youtube shows a message to allow IOBlog the access to his account with a security alert if the application is not registered.</p>
<p>5- User click in allow permissions.</p>		
		<p>6 – Youtube redirects to IOBlog returning a token.</p>
	<p>7 – IOBlog transforms this token in an infinite token with Gdata API and stores it.</p>	
	<p>8 – IOBlog redirects user to import page and show the message "Import successfully done".</p>	

Alternative flow:

- 5b – User doesn't allow permissions to web application.
- 6b – Facebook redirects to call-back URL without session token.
- 7b – System ends without import youtube account.

Change password

If the user changes the password to log into a social network with password authentication, SocialLuna will not access to his/her account because the password stored will be obsolete. When this happens, IOBlog warns the user to change the password stored in SocialLuna to allow the communication with the social network. In the profile page, the user will manage his/her accounts and will have the option to change the passwords stored in SocialLuna platform to connect with social networks.

Actor – user	System
1- User clicks in “change your password” hover the social network to change the password.	
	2 – System shows to user a form to insert the new password to the same account.
3 – User inserts and repeats his password and click in change button.	
	4 – System saves this information to connect to social network.

Delete Social Network

A user has the option to delete the Social networks from which he/she does not want to continue receiving updates into IOBlog. When a users delete a social network from the application, they lose all the messages of the network stored on IOBlog.

Actor – user	System
--------------	--------

1- User clicks in delete icon of a social network to delete the network from the system.

2 – System shows an alert, advising that this change will cause the loss of his messages in this network.

3 – User clicks in continue deleting button.

4 – System delete the social network

7.1.3. Settings – Create groups

Another option from users is the creation of groups that create more efficient filters to the messages search. This way, a user will create different groups for each friend and later search only messages from users within a group.

The different options that a user can do in profile groups are:

- ◆ Create empty group,
- ◆ Update group,
- ◆ Delete group

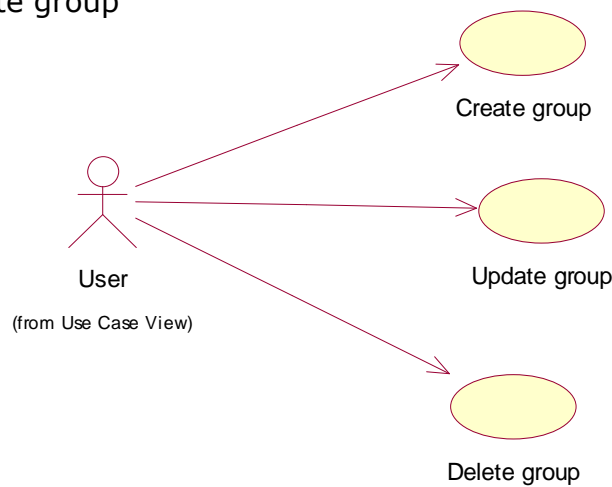


Image 12: Groups use cases

Create group

Users can create a new empty group at any time. They only have to fill up the form field with the name of the new group and click in create group button.

Actor – user	System
<p>1- User fill the form field with the group name and click in create group button.</p>	<p>2 – System creates group if this not exist from this user and return to groups profile page.</p>

Alternative flow:

2b – If the group exists, system will do nothing and will return to groups profile page.

Update group

Users can add or remove friends into a group to update the current status of the group. To add friends, they only have click in the friend’s name and this will move to the group form and the name will be deleted from friends list. To remove a friend from a group, the user clicks over the friend’s name in the group list. This friend will be removed from friends list and will be added into the friends list out of the group.

<u>Actor – user</u>	<u>System</u>
1 - User clicks over group name in group list.	2 – System fills the friends in group list with the friends in the group selected.
	3 – System removes from “friends out of group” list the friends in group selected.
4 – User clicks in a friend in “friends out of group” list.	5 – System removes this friend from the list and adds in the list with friends in group.
6 – User clicks in a friend in group list.	7 – System swaps this friend between lists.
8 – User clicks in update group	9 – System updates and save the new group configuration.

Alternative flow:

User can repeat steps 4 and 6 any time.

Delete group

Finally, users can delete created groups. To delete groups, the user only has to click in the delete logo next to the group name in the list.

Actor – user	System
1 - User clicks in delete logo to delete a group.	2 – System asks for confirmation.
3 – Users confirm to delete the group.	4 – System removes the group from the user account.

Alternative flow:

3b - User clicks in not delete group.

4b - System do nothing.

7.1.4. Show messages

The main use case of IOBlog is the visualization of posts in all user networks in the same interface. Users will see in their inblog page the messages received into their social networks and in their outblog page the messages sent by the user to each network.

Other uses cases will allow users organize the information to show, in each moment, only the interesting information to each user.



Image 13: Show messages Use Case

Actor – user	System
--------------	--------

1 - User enters into inblog or outblog page.

2 - System shows to user his messages into selected folder.

Users can also view more information selecting the message to view. The system will show all the information related with the post as images, comments, etc.

Actor – user	System
--------------	--------

1 - User clicks over a message post

2 - System shows to user an extended view with all information related with the message.

Permanent post link

Permanent link is a unique page for each user's message with all message information, comments, images ... To access this page, the user has to click over "View all Comments" in post view table.

Another way to view a post in this page is inserting the URL of the message manually in the web navigator. If the user in the message page is a non logged user, the options of reply message or post a comment will not be accessible.

All messages are visible to all Internet users because all information collected by SocialLuna is public in user's networks, although the URL is created encrypting to md5 different posts parameters, so the access is very difficult the access to third parties.

7.1.5. Messages management

For achieving good service is necessary to offer tools to allow the management, organization and presentation of all the user's messages. Different use cases have been created to offer all these functions to the final user. These use cases are:

- ◆ Filter messages,
- ◆ Sort messages,
- ◆ Messages visualization,
- ◆ Mark messages as favourites,
- ◆ Mark messages as read / unread,
- ◆ Tag messages.

Filter messages

Filter messages allow users seeing only specific posts according to search criteria. They can filter by:

- ◆ An specific network,
- ◆ A friend,
- ◆ A previous assigned tag,
- ◆ A previous created group or
- ◆ The time period of the message.

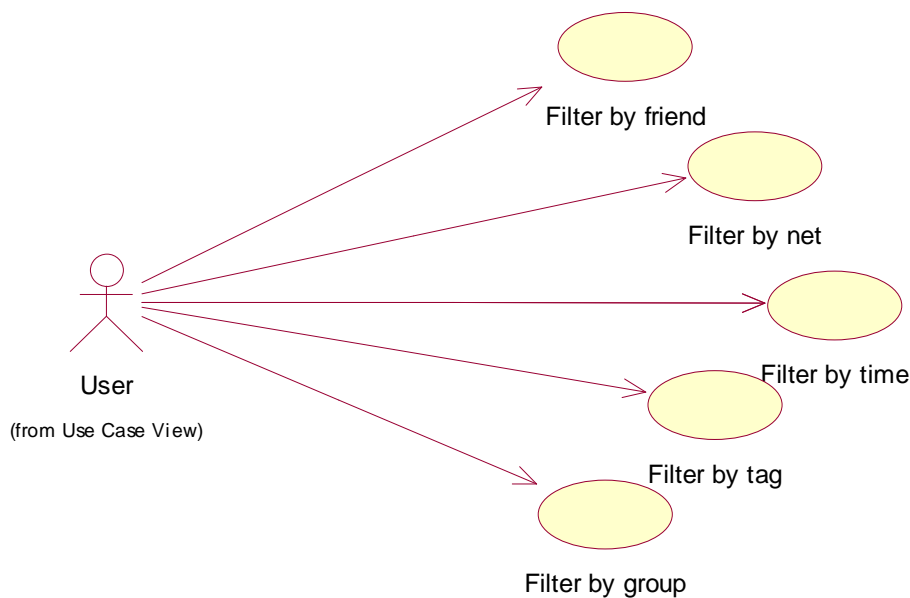


Image 14: Filter use cases

Actor – user	System
--------------	--------

1 – User clicks over a filter (network, friend, tag, group or time).

2 – System shows to user his messages applying the selected filter. If no messages result from the query, system shows the messages “No messages found”.

Search friends / Filter friends

Related with the previous filter messages by friends and to facilitate the search of friends to filter, IOBlog offers a powerful engine to find a friend into the user friend's list. On the other hand, users can filter the friends list viewing all their friends or showing only the friends with unread messages.

Actor – user	System
1 – User writes a character into input text.	
	2 System shows into friends list the first five results that contain the input text string ignoring case sensitive.
3 – User writes or deletes new characters into input text.	

Messages visualization options

Other useful feature to assist the messages view to final users is a sorting function. This function allows the user to arrange the messages by network friend, viewing only unread messages or all messages. The different uses cases for IOBlog application are:

- ◆ Sort by (network, friend, date)
- ◆ View (all messages, only unread)
- ◆ Order messages (ascendant, descendent)

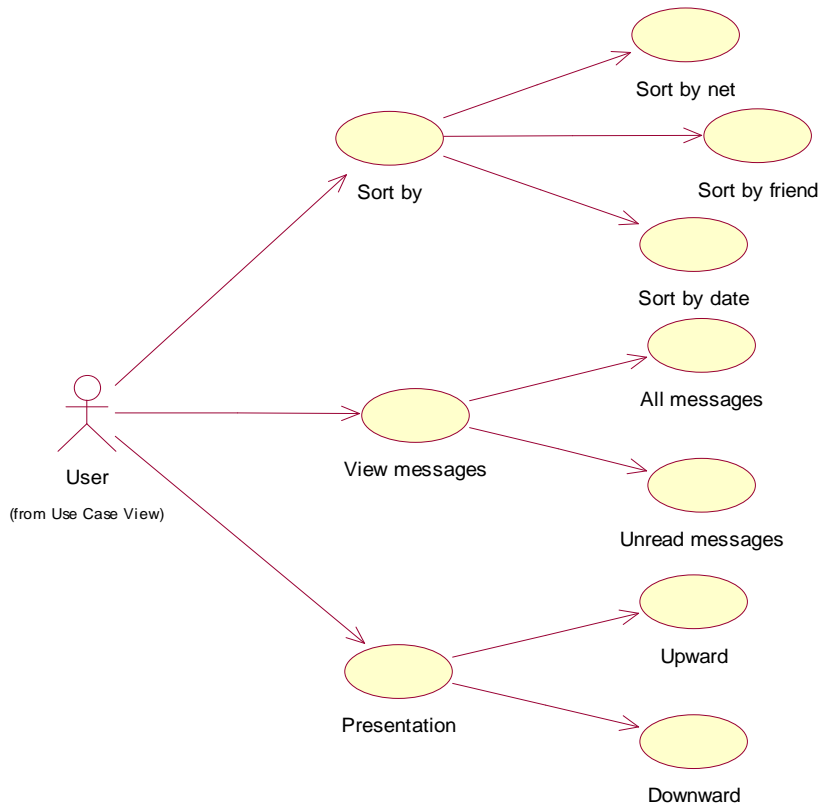


Image 15: Message visualization use

Sort by (network, date, friend)

“Sort by” use case reloads the messages table showing the same messages in different order. If user clicks “sort by date” the messages will be shown from newest to latest in creation date. If user clicks over “Sort by friend” the messages will be sorted in alphabetical order from A to Z. Finally, if user clicks over “Sort by net” the messages will be grouped alphabetically by net name.

Actor – user	System
--------------	--------

1 – User clicks over sort menu and choose the subject to sort the messages.

2 – System recharges messages table showing the messages in the selected order.

View (all messages, only unread)

View use case is another filtering message function located in the messages table. This use case reloads the messages table showing all the messages that meet the selected criteria or the unread subset messages only.

Actor – user	System
1 – User clicks over view menu and choose the option to filter messages to show into messages table.	2 – System recharges messages table showing the messages with the new filter selected.

Order view

Finally, users can sort the messages in the table in ascending or descending order.

If messages are sorted in descending order, it is only possible to click into ascending order and if messages are sorted in ascending order, only is possible to reorder the messages table in descending order.

Actor – user	System
1 – User clicks over ascending / descending arrows.	2 – System recharges messages table showing the messages in selected order.

Mark messages as favourites

Users can highlight their favourite messages and then find them more easily. These messages can be marked one by one or selecting all the checkbox of the messages and using the option into table menu "mark as favourite".

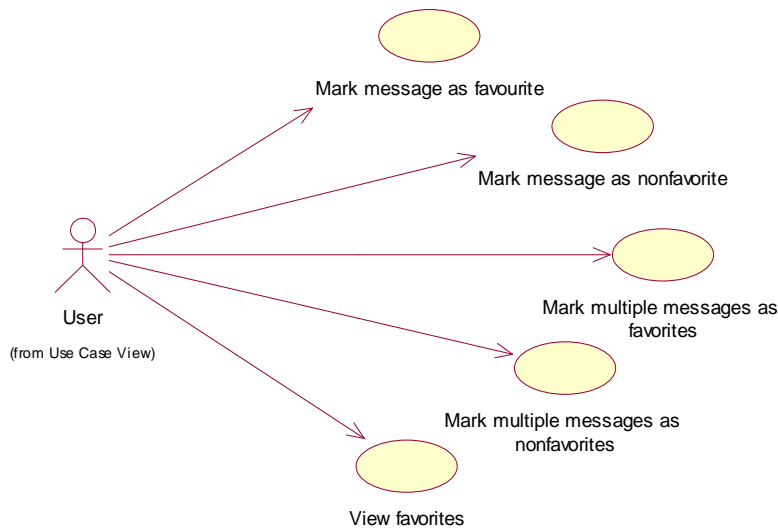


Image 16: Message management use cases

Mark message as favourite / no favourite

Actor – user	System
--------------	--------

1 - User clicks in star icon on message row.

2 - System toggles his state between favourite or non favourite.

Mark multiple messages as favorite / non favorite

Actor – user	System
<p>1 - User checks messages to change their favourite status and clicks "mark as favourite" into action menu.</p>	
	<p>2 – System mark all messages selected as favourites. If some messages were favourites, these messages don't change the status.</p>
<p>1b - User checks messages to change their favourite status and clicks "mark as non favourite" into action menu.</p>	
	<p>2b – System mark all messages selected as non favourites. If some messages were non favourites, these messages don't change the status.</p>

View favourites

This use case shows to users their favourite messages into messages table. View use case

Actor – user	System
<p>1 – User clicks over "favourites" button.</p>	
	<p>2 – System shows only favourite messages into messages table.</p>

Mark messages as read / unread

Each time a user clicks over an unread message; this message will be marked as read. In the same way, users can mark multiple messages as read or unread

Tag messages

An IOBlog user can manage his/her messages creating and assigning tags to classify and filter future searches. Possible use cases to tag messages are:

- ◆ Create a new tag.
- ◆ Assign an existent tag to selected posts.
- ◆ Remove a tag from selected posts
- ◆ Delete an existent tag.

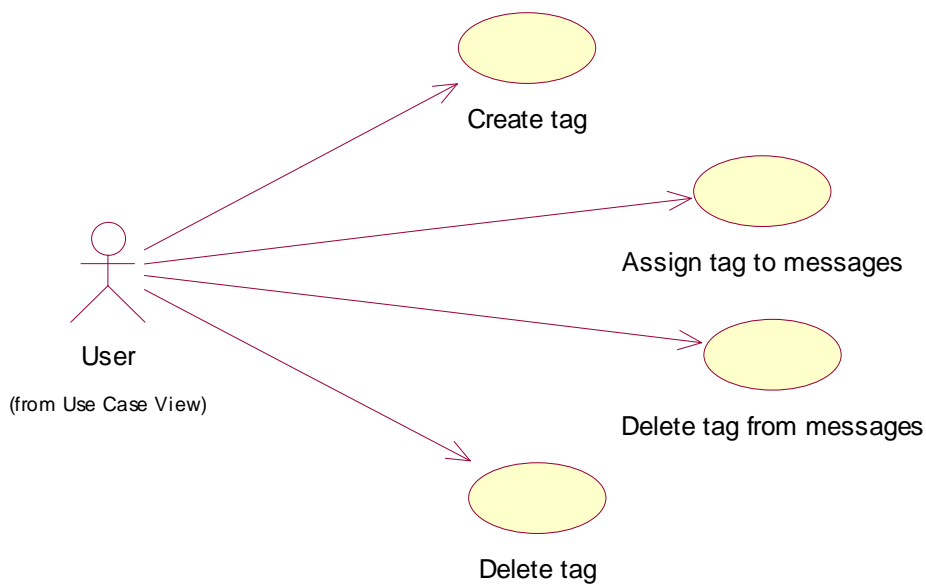


Image 17: Tags use cases

Actor – user	System
--------------	--------

1 - User clicks tags menu

2 - System shows different options related with tagging messages.

3 - User inserts new tag name into text field and clicks in "new" button.

4 - System creates the tag and shows this tag into tag list.

Create tag

Users can create tags or labels to use and apply them to different posts. The use case create tag creates a new tag related to user without any message associated.

Actor – user	System
--------------	--------

1 - User clicks tags menu

2 - System shows different options related with tagging messages.

3 - User inserts new tag name into text field and clicks in "new" button.

4 - System creates the tag and shows this tag into tag list.

Alternative flow:

4b - If created tag by the user exists, system does nothing.

Delete tag

Users also can delete created tags. If a tag is associated to multiple messages, the deletion of this association will be broken and all posts with this tag will loose the label.

Actor – user	System
1 - User clicks tags menu	2 – System shows different options related with tagging messages.
3 – User clicks in delete icon next to tag name.	4 – System shows an alert advising that all posts with this tag associated will lost the tag.
5 – User clicks in delete button	6 – System deletes the tag, remove it from tags list and delete all labels from tagged posts.

Alternative flow:

5b – User clicks in don't delete button.

6b – System does not delete the tag and does not remove the labels in associated posts.

Assign tag to messages

With a tag created, users can assign these tags to different messages. With these user associations, IOBlog will allow users filter the messages to show in main page according user preferences.

Actor – user	System
<p>1- User selects all messages checkboxes to tag and clicks hover tags menu</p>	
	<p>2 – System shows different options related with tagging messages.</p>
<p>3 – User clicks a tag name into tag list.</p>	<p>4 – System tags all posts without the tag previously assigned to each post.</p>

Remove tag from messages

Finally users can delete associations between user’s tags and post previously assigned.

Actor – user	System
<p>1- User selects all messages checkboxes to remove a tag and clicks hover tags menu</p>	
	<p>2 – System shows different options related with tagging messages and shows into remove tags list all possible tags to delete in function of tags assigned to messages selected.</p>
<p>3 – User clicks a tag name into remove tags list.</p>	<p>4 – System remove the tag form all selected posts.</p>

7.1.6. Send new message

Finally, the last basic use case to interact with external social networks is offering functions to send messages to all their networks. To achieve these features, we raise the following use cases:

- ◆ Send message to networks (one or multiple networks at once)
- ◆ Reply friend posts
- ◆ Fast update of network status

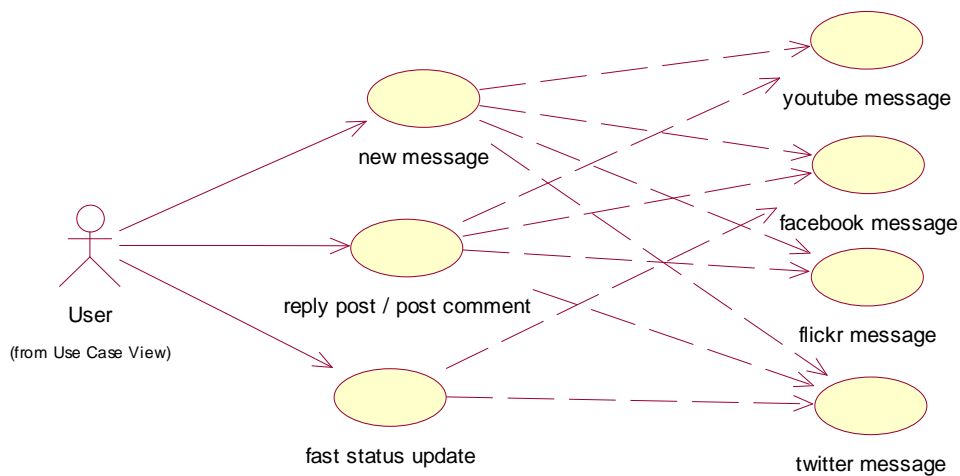


Image 18: Send messages use cases

Send message to multiple networks

IOBlog must be able to send messages to user social networks. In this use case, users choose the type of message to send. They can choose between a plain text (update status), upload an image or upload a video. When they have chosen the message type, IOBlog will show the networks available for this type of message according to user imported networks and networks APIs.

Actor – user	System
<p>1- User goes to new message page to send a new message.</p>	<p>2 – System shows a form radio buttons to select the type of content of the message.</p>
<p>3 – User selects the type of content of the message between text, image or video.</p>	<p>4 – System shows the available networks to send the message and the form fields necessities to create the message (title, text, image, video...). If user has facebook imported but has not given permissions to SocialLuna to update images or text, system shows an alert with a link to allow SocialLuna send messages to Facebook.</p>
<p>5 – User selects the network or networks to send the message.</p>	<p>6 – System shows the buttons send and clear.</p>
<p>7 – User fills the form fields and clicks send message button.</p>	<p>8 –System sends the message to selected networks and redirect user to Inblog page showing a message reporting that the message has successfully sent.</p>

Alternative flow

7b – User clicks button “Clear form”.

8b – System deletes content from all form fields.

Reply messages

Another option to send a message to a network is reply a message to a friend. These replies depend from networks where the messages have been posted. A reply in twitter is a new post on twitter with the text “@friend_name_to_reply” at the beginning of the post. In an image or video post, reply a post consists in sending a comment for the image or video.

Actor – user	System
1- User clicks over a message	
	2 – System shows to user all message information with a comments preview.
3 – User clicks “reply” or “post a comment” button.	
5 – User writes the message and click “send”.	4 – Systems do visible a input text form to write the reply.
	6 – System sends the message to the message network. If it was a image message this reply will be an image comment to friend post.

Fast post to network

Finally, users can send fast messages independently of IOBlog page that they have into screen. These messages will be status updates into the social network selected. The social networks available are twitter and facebook and only if the user has imported these networks previously.

To send a message on this way, user must click over header page and the text form will get focused.

Actor – user	System
1- User clicks over page header	
	2 – System deletes the message “Update your status” and put the focus over text input.
3 – User writes the status message to send.	
	4 – System updates the chars counter with each key pulsation.
5 – User clicks “Update my status” button or press enter key.	
	6 – System sends the message to the selected network.

Alternative flow

5b - User press escape key or clicks out of textbox area.

6b – Form losses the focus and appears again the message “Update your status” if text area was empty.

7.2. UX Model design

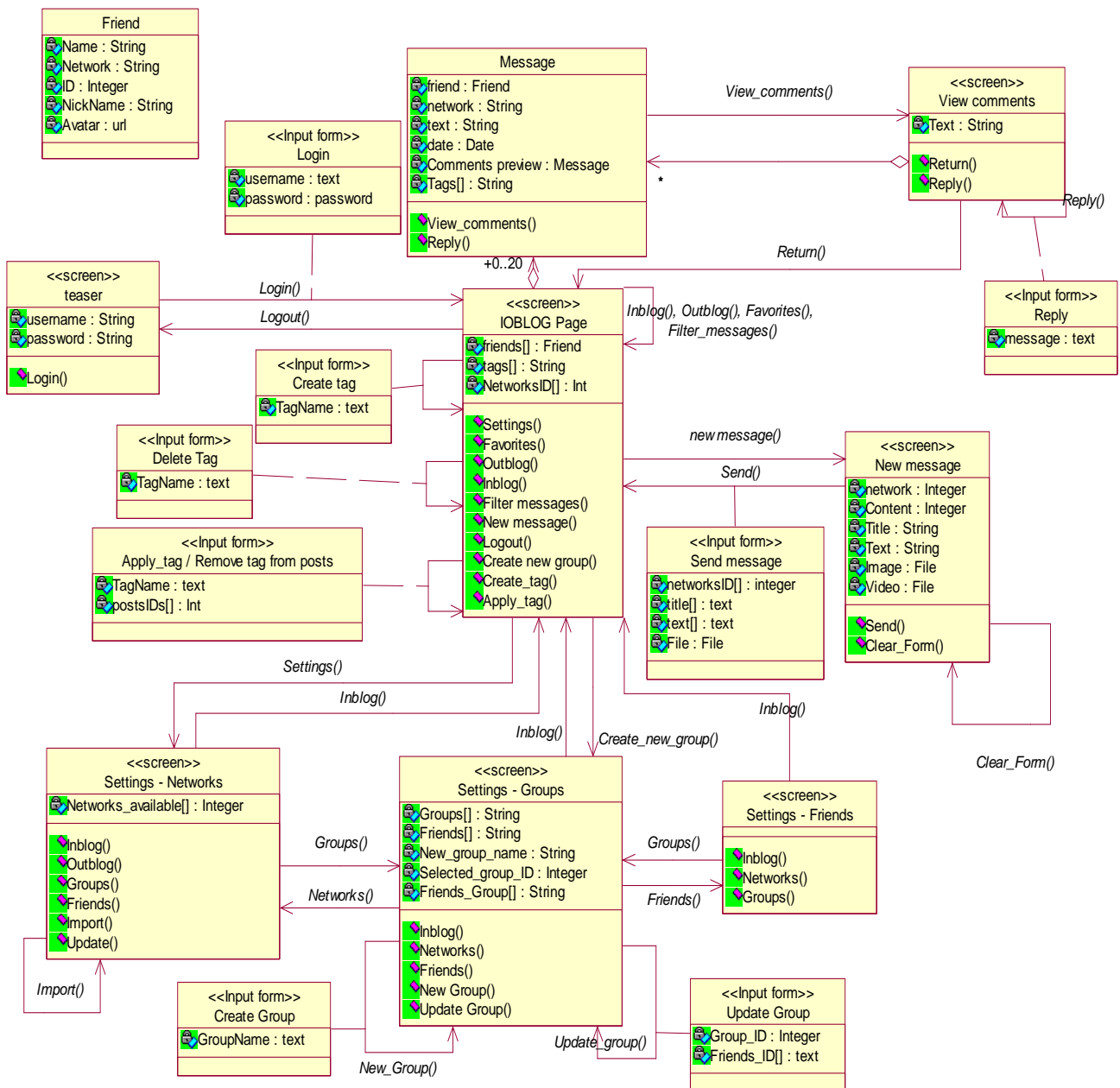


Image 19: UX Model

This UX diagram shows the map navigation in IOBLOG application. When a user goes to main page, the first page in appears is the teaser page. In this page the user has to enter into login form her username and password to authenticate that is a valid user.

When user authenticates, goes to IOBlog page. In this page, users can see all the messages in all their social networks and search messages applying filters or tags without moving of IOBlog page. To do any function over posts (as tagging posts) or to create or delete tags is necessary to send information in a form with the tag name or the selected posts to focus the action.

When user goes to edit, IOBlog settings, application redirects to new page with networks settings. In this page, user can manage her imported accounts and import new accounts. From this page, user can click over different settings options and go to new pages to edit her groups and friends preferences. Also, user can return to IOBlog from all previous described pages.

Again at IOBlog, users can see their messages directly in IOBlog page or can enter in a detailed message page where they can see the message and all comments. At same time they can reply or post a comment directly to network of the post.

Finally users can go to another web page to send a message to their imported accounts. In this page they can send messages to different networks at once and then return to IOBlog page.

7.3. IOBlog structure

In this section I will explain the general structure of the IOBlog web application IOBlog developed using SocialLuna platform. I will start explaining the Model View Controller pattern used to design the web application. Next I will enter to explain the main directories of the applications and the files that you can find there. With all the files located on the project context, I will explain the PHP controllers that manage the user interface. Finally, in the next point, I will explain the view development techniques using Smarty and JavaScript.

7.3.1. Model View Controller (MVC)

Model View Controller (MVC) is an architectural pattern that allows isolating the business logic from the user interface. This isolation results in an application easier to modify and allows changes in business logic without affecting the user interface view or vice versa.

The functions of the MVC pattern components are:

- ◆ **Model:** Is the responsible to enter the database to get data or manipulate it. In IOBlog applications, the model is in charge of calling the web services of the SocialLuna platform to get or update the data.
- ◆ **Controller:** Catches the user input events and executes all the business logic associated to the event. If it needs to access the data, it calls the model. Also, it changes the view variables that will be accessible from the view to show the dynamic information to the user.
- ◆ **View:** Shows to user the application interface and waits to a user input event.

In the image below you can see the basic structure of the MVC pattern and a sequence diagram of the interaction between user application events and MVC pattern:

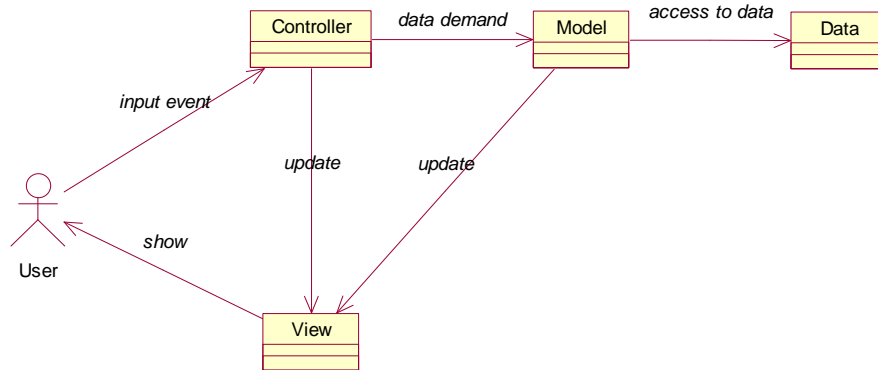


Image 20: MVC design pattern

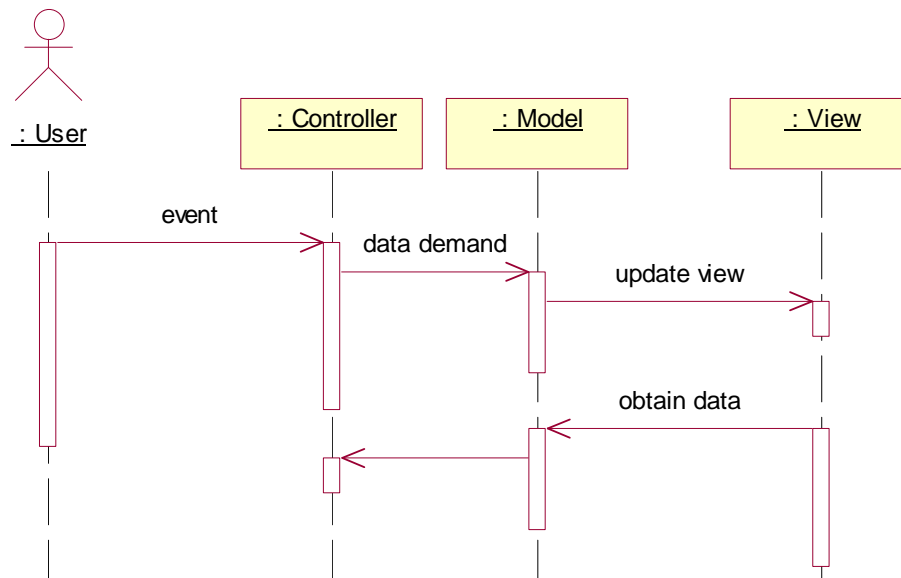


Image 21: MVC pattern – Sequence diagram

This pattern is commonly used on web applications assuming the view as the html code, the controller as the dynamic part of the web managed in the web server and the model as the access to the data base.

There are many frameworks that help the developers to apply the Model View Controller Pattern into their web applications. The framework used to help us in the IOBlog development process has been the Zend Framework, which runs under free BSD license.

7.3.2. IOBlog directories

Zend Framework, used to develop IOBlog application, offers multiple functions to maintain a great web project clean and tidy under the MVC pattern. The project is structured in several folders. The most significant folders are:

- ◆ /app/controllers:

This folder includes all PHP files with the logic to manage all user petitions. When it is necessary to use platform web services, calls to functions developed in models folder.

- ◆ /app/models:

Includes PHP files that implement all the calls to SocialLuna web services and returns the platform responses.

- ◆ /app/views/layouts:

It includes smarty templates with the basic web page structure. The basic webpage structure consists in a header div, a middle content div and a footer div.

- ◆ /app/views/scripts:

This folder includes a folder for each web page in IOBlog. In these folders there are the necessary templates to create the content page. In some cases, these templates are very small because some web pages are created in JavaScript files. In profile settings case, the web page is clearly html smarty templates created and we can found several templates for each part of the same screen.

- ◆ /webroot:

Webroot is the public directory where all elements accessible from the web can be found. Because of that, in this path there are all the needed files to create the client-side webpage such as JavaScript objects or CSS Style Sheets.

- ◆ /webroot/css:
It includes the CSS files to add style to all the web pages.
- ◆ /webroot/js:
Includes the JavaScript files with needed create some web pages. The JavaScript construction will be explained in the following sections.
- ◆ /webroot/gfx/img:
Includes all the necessary images used in web pages design.
- ◆ /libs:
The folder /libs include all the required libraries from the Zend Framework libraries, the social networks APIs or the project creation libraries.

7.3.3. PHP Controllers

PHP Controllers have the server logic of all the web pages.

There is a general abstract controller called **SocialLunaController** that extends the abstract Zend class `Zend_Controller_Action`. All the specific controllers will extend **SocialLunaController**. This controller assigns the basic attributes in all the pages such as general CSS files includes or general JavaScript files.

Another controller is created for each other action (web page) in the application. **IOBlogController** includes all needed logic to get user friends, user messages, networks and all information showed in IOBlog page (INBlog, Outblog or New message page). At once, **ProfileController** manages all the user settings options and **AuthController** has the actions to authenticate users on login.

PHP Zend config.ini

Config.ini file is a configuration file to the web responses. On this file are stored different variables to use in the application such as the social networks API keys, paths of log files, upload files directories or web services base path.

In this file, are also defined the instructions to redirect the actions and paths of the web application to a PHP controller.

```
; /login
; /logout
; /signup
; /teaser
route.auth.route           = ":action"
route.auth.reqs.action     = "login|logout|signup|teaser"
route.auth.defaults.controller = auth

; /ioblog application
route.ioblog.route        = "ioblog"
route.ioblog.defaults.controller = "ioblog"
route.ioblog.defaults.action   = "index"
```

Code 2: **Redirect actions configuration**

This lines are assigning the paths /login, /logout, /signup and /teaser to the actions loginAction, logoutAction, signupAction, and teaserAction in the authController. Also it is defined the route /ioblog to the controller ioblog and the indexAction as the default action to execute when a user enters at this point.

PHP Controller Structure

All controllers are developed following the same template:

- ◆ **Init** function is the first function on loading and it is used to assign specific page scripts or style sheets and assign them to the view layout. This layout is the sample template and it is located in folder /app/layout.

```
public function init()
{
    parent::init(); // Executes the parent init function

    // Assign CSS to use in the webpage of the controller
    $this->view->headLink()->appendStylesheet
        ('/css/ioblog.css?rev='.App::REVISION, 'screen');
    (...)
    // Assign JavaScript files to load with the webpage
    $this->view->headScript
        ('file','/js/table.js?rev='.App::REVISION);
    (...)
    // Assign view layout to render webpage
    $this->view->layout()->setLayout('main-full');
}
```

Code 3: **Basic controller init function**

- ◆ **IndexAction** is the main function to load the page. This function is executed when a user reloads the web page associated to this controller without any specific function declared. For example, if a user reloads the web page SocialLuna/ioblog the indexAction of IOBlogController is executed. If the web page reloaded is SocialLuna/profile, indexAction of ProfileController will be executed. On IndexAction are allocated the variables that are needed to create the view and control all different ways depending of the user selection. A simple example will be:

```

public function indexAction(){
    /* Getting GET parameter 'blog' */
    $blog = $this->request->get('blog');
    switch ($blog)
    {
        case 'inblog':
            //load inblog messages
            break;
        case 'outblog':
            // load outblog messages
            break;
        default:
            // load default vaiables
    }
}
    
```

Code 4: Index action example

- ◆ **myfuncAction** is the function executed when the controller is loaded with a specific function declared. For example myfuncAction in IOBlog controller will be executed when loading the page SocialLuna/ioblog/myfunc. This functions are often called on form submits to execute a specific action and at the end redirects user to indexAction.

```

//HTML Code
<form action="/ioblog/new-message" method="post">
    (...)
</form>

/*
 * On submit
 * PHP InblogController
 */
public function newMessageAction(){
    (...)
}
    
```

Code 5: Actions example

- ◆ **myfuncAjaxAction** is executed with the call to myfunc from a JavaScript file. This function can return data to the JavaScript function that calls it or redirects to an IndexAction after executing the assigned work. When it returns data to JavaScript file, this data is encoded to JSON format. An example of JavaScript call is:

```
/*
 * JavaScript function
 */
SL.myfunction = function (){
    var params ={
        // Parameters passed to AJAX call
    };

    jQuery.getJSON ("/ioblog/myfunction", params, function(data)
    {
        // Function executed on AJAX call return with
        // data result
    });
}

/*
 * PHP function on ioblogController
 */
public myfunctionAjaxAction(){

    // Get needed data
    $data = getData();

    //Disable ViewRenderer as we just want to output JSON
    $this->_helper->viewRenderer->setNoRender();
    $this->_helper->layout->disableLayout();
    // Encode data to JSON format to return to JavaScript function
    $jsonData = Zend_Json::encode($data);
    $this->getResponse()->setBody($jsonData);
}
```

Code 6: Ajax Actions example

In the following image can be seen the basic interaction between the user actions in front-end and the calls to back-end services.

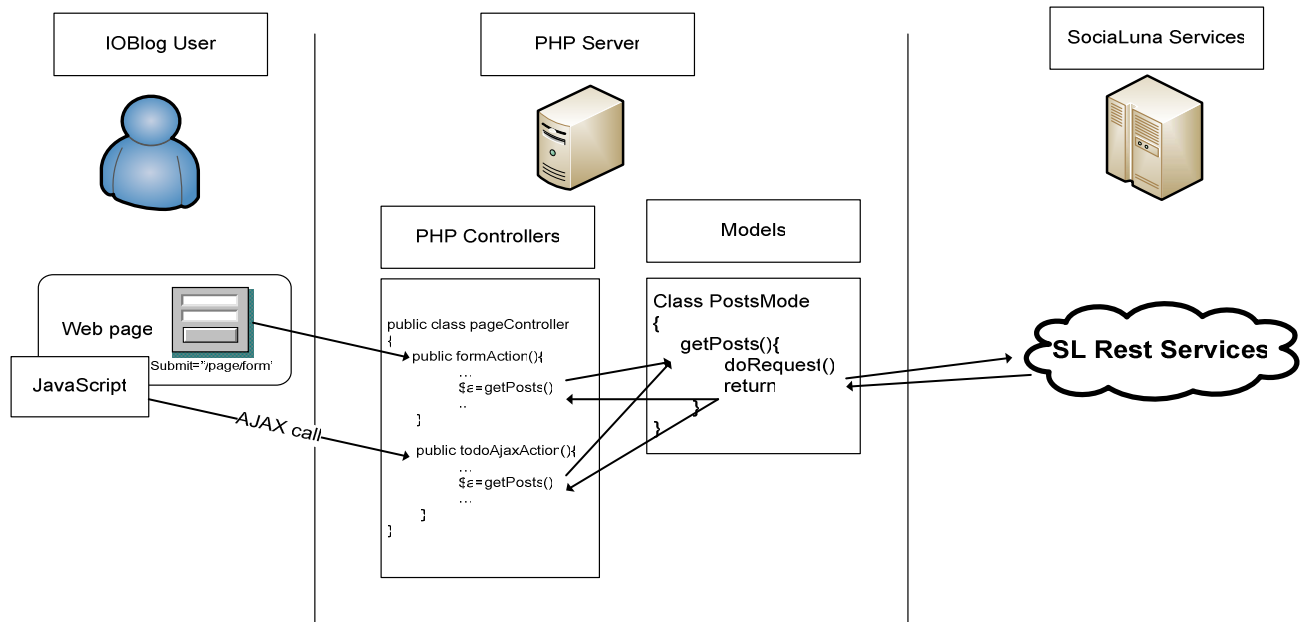


Image 22: calls to server diagram

The image shows how all the user actions go to the appropriate controller action. In these actions, the controller gets the parameters passed using GET method on the URL request or using POST method in a form request and executes the necessary logic of the action to do. If the action needs data from SocialLuna platform calls to a specific method in a model class. This function of the model will call the correct web service and will return the needed data.

7.4. View construction

Two methods were used in the development process of the application view. Initially, the project was developed entirely using Smarty, a template engine for PHP that facilitates a manageable way to separate application logic and content from its presentation. Later, with the introduction of many AJAX calls, the content of the website was developed using JavaScript to help handle all the possible events at the same time which gave more dynamism to the website. In this chapter I will explain the two ways of constructing webpage the content.

7.4.1. Smarty web development

Developing a webpage using Smarty is a simple way to create a graphic interface without dependencies on the logic part. Smarty needs the variables passed between the controller and the view to create the webpage using html code.

Smarty is extremely fast and allows the programmer the creation of custom functions, modification of variables, unlimited nesting of sections, if statements, etc.

A `“.tpl”` file (Smarty template) contains a simple html webpage with some instructions enclosed within delimiters. By default these are `{,}` but they can be changed.

The most used smarty instructions into project development are:

- ◆ **{if}...{else}...{/if}**: conditional instruction.
- ◆ **{section} ... {/section}**: looping over arrays of data.
- ◆ **{foreach} ... {/foreach}**: looping over associative arrays.
- ◆ **{assign}**: Modify variable.
- ◆ **{\$this->partial(“tpl file”)}**: Load the tpl file specified.

Linking PHP Controllers and Smarty templates

A view layout must be assigned to the Smarty template to link a Front Controller in PHP. As has been seen before this assignation it is done on init function, assigning to the layout variable of the object view the name of the smarty template in layouts folder. If the action has no layout assigned, Zend Framework will search the default template in the scripts folder.

On load the layout template with all project generic parts in screen as the header or the footer content page. The specific webpage content is in the index.tpl file at folder “/scripts/controller_name”.

Furthermore, it is often necessary to pass variables between the controller and the view to render these data into webpage and to render or not some parts of the view. To pass variables between views controllers and Smarty templates using php Zend framework is as easy as assign in the controller the value to the variable var_name into the view object of the controller (“\$this->view->var_name”). Assigning variables to the view component, these were accessible into Smarty templates as { \$var_name }.

For example, to render the settings page that has a tabs menu with different settings the system works as follows:

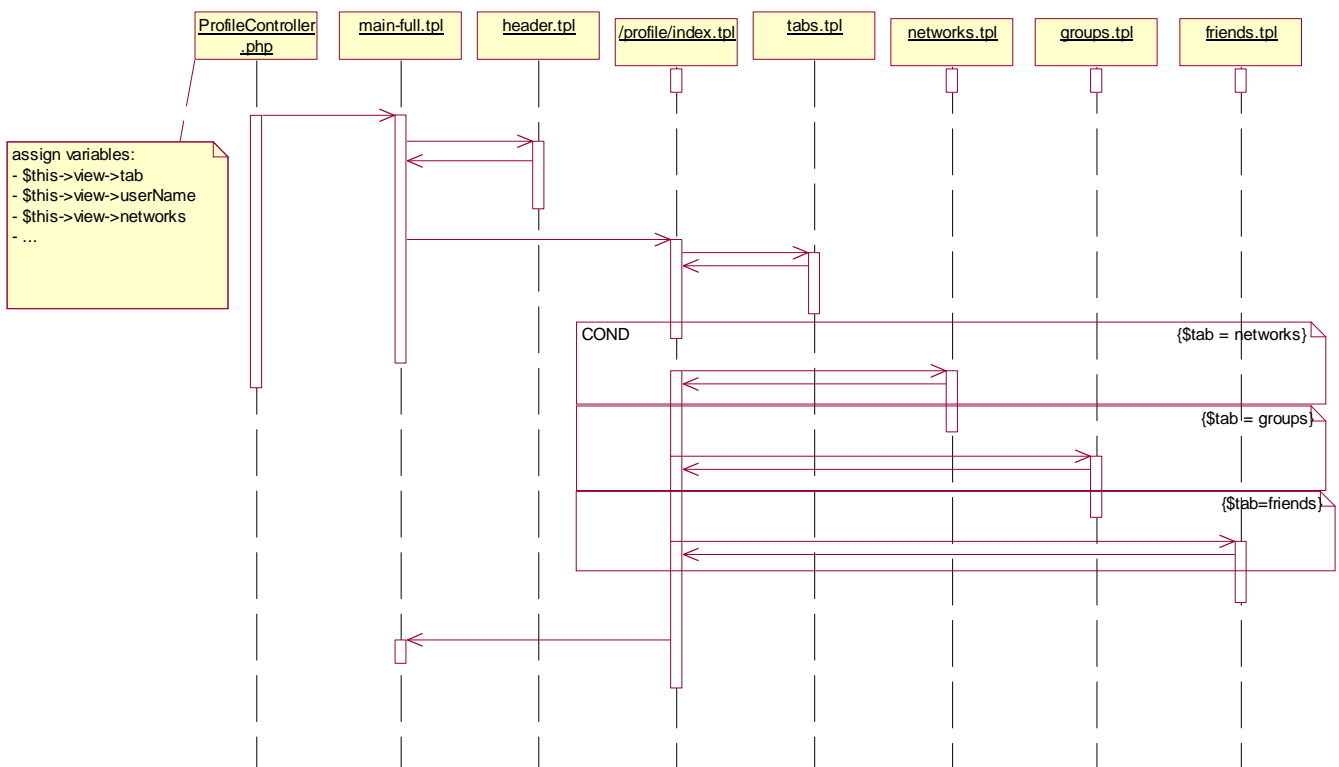


Image 23: Template loading example

7.4.2. JavaScript web construction

As it has been described on technologies introduction, JavaScript is an interpreted programming language widely used for client-side web. As other object-oriented languages, JavaScript allows inheritance but use prototypes instead of classes for defining object properties.

The most important features of this language that are used in the development process are:

- ◆ **Dynamic:** JavaScript is a weakly typed language. It means that as in most scripting languages, types are associated with values, not variables. For example, a variable `x` could be bound to a number and then later rebound to a string. JavaScript supports several types to test the type of an object.
- ◆ **First-class functions:** these are objects themselves because they have properties and can be passed around and interacted with like any other object

```
A = function (){
    B(_privatefunction);
    _privatefunction(){
        ...
    }
}
B = function (funconComplete){
    ...
    funconComplete();
}
```

Code 7: JavaScript classes example

- ◆ **Inner functions and closures:** Inner functions (functions defined within other functions) are created each time the outer function is invoked, and variables of the outer functions for that invocation continue existing as long as the inner functions still exist, even after that invocation is finished. A **closure** is a function that is evaluated in an environment containing one or more bound variables. When called, the function can access these variables.

```

A = function (){
    var x = 15;
    var y = 10;
    B = function (){
        var x = 20; // Creates new x variable in this closure
        y = 5 // modifies variable y on function A
        var sum = x+y; // 20 + 5
        alert (sum) // shows 25
    }
    B();
    var sum = x+y; // 15 + 5
    alert (sum) //Shows 20
}
    
```

Code 8: Inner functions and closures

- ◆ **Prototypes:** JavaScript uses prototypes instead of classes for defining object properties, including methods, and inheritance. It is possible to simulate many class-based features with prototypes in JavaScript.

```

SL = SL || {};
SL.Table = function (page,url) {...} //Constructor function
    /* Init function */
SL.Table.prototype.init=function (div) {
    this._setPage(0);
}
    /* Private functions */
SL.Table.prototype._renderHeader () {...}
SL.Table.prototype._renderRow () {...}
...
SL.Table.prototype.refresh (data) {...}
SL.Table.prototype._setPage(page){
    this._page = page; //Object var
}
    
```

Code 9: JavaScript prototypes

- ◆ **Functions as object constructors:** Functions double as object constructors along with their typical role. Prefixing a function call with **new** creates a new object and calls that function with its local **this** keyword bound to that object for that invocation. The function's **prototype** property determines the new object's prototype.

```
//Printing messages table
var div = document.createElement('DIV');
div.id='table_div';

//Creating a new object table to insert in the page
this.ioblog_messages = new SL.Table(this, '/ioblog/render-messages?');
this.ioblog_messages.init(div);
this.ioblog_messages.refresh(data);
```

Code 10: Messages table construction

Construct a webpage in JavaScript consists on creating some JavaScript objects with class properties and functions that render and control all events over the object. The basic structure is a constructor function that creates the JavaScript object and a set of functions to manipulate the DOM from the object attributes. This way to create a webpage completely changes the traditional way of construct html pages using this language only to add small user interactions.

In the project exists five basic JavaScript files with different render and event handler functions:

- ◆ **ioblogPage.js**: Builds the basic page structure (left menu, right info, middle info) of the messages visualization page.
- ◆ **table.js**: Renders the messages table in the middle content page.
- ◆ **message.js**: Builds the message extended view (permalink page) of a selected message.
- ◆ **new.js**: Renders the page to send a new message to the social networks imported.
- ◆ **functions.js**: Generic JavaScript functions to use in different pages and non JavaScript pages.

Loading messages page

The JavaScript `ioblogPage.js` is loaded from the Smarty template `index.tpl` passing the needed variables to the `init` function. The process to load the main page is the one shown in the image below:

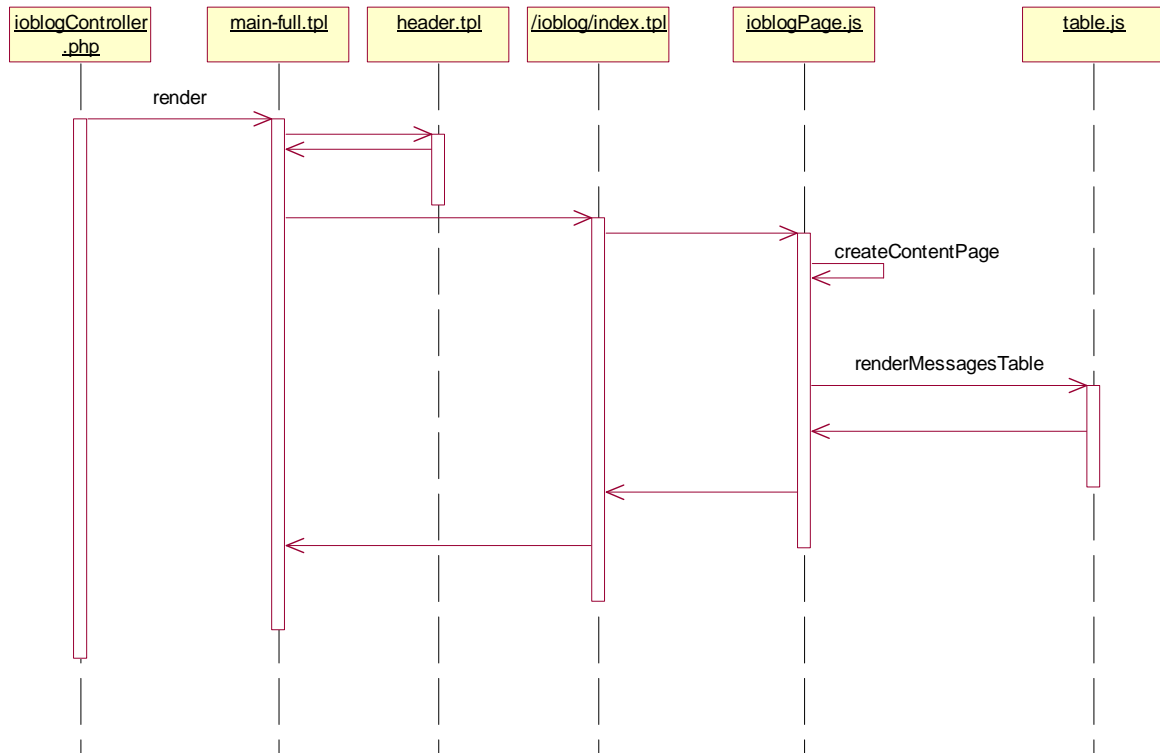


Image 24: Loading JavaScript page

The process to load the `index.tpl` is the same as described above, but once there, the template will create the object script that would be responsible for creating DOM elements of the website and the functions needed to manage the events over these elements.

When the page is loaded at first time, the controller is the responsible to get the messages that are shown in the table. However, when the user clicks over a filter to recharge the table, the JavaScript `table` object is the responsible for making an AJAX call to the controller demanding for the new messages to show. While the new messages are loading, changes the graphic table to add dynamism to the user demand.

The image below shows the different components that make up the main page of IOBlog:

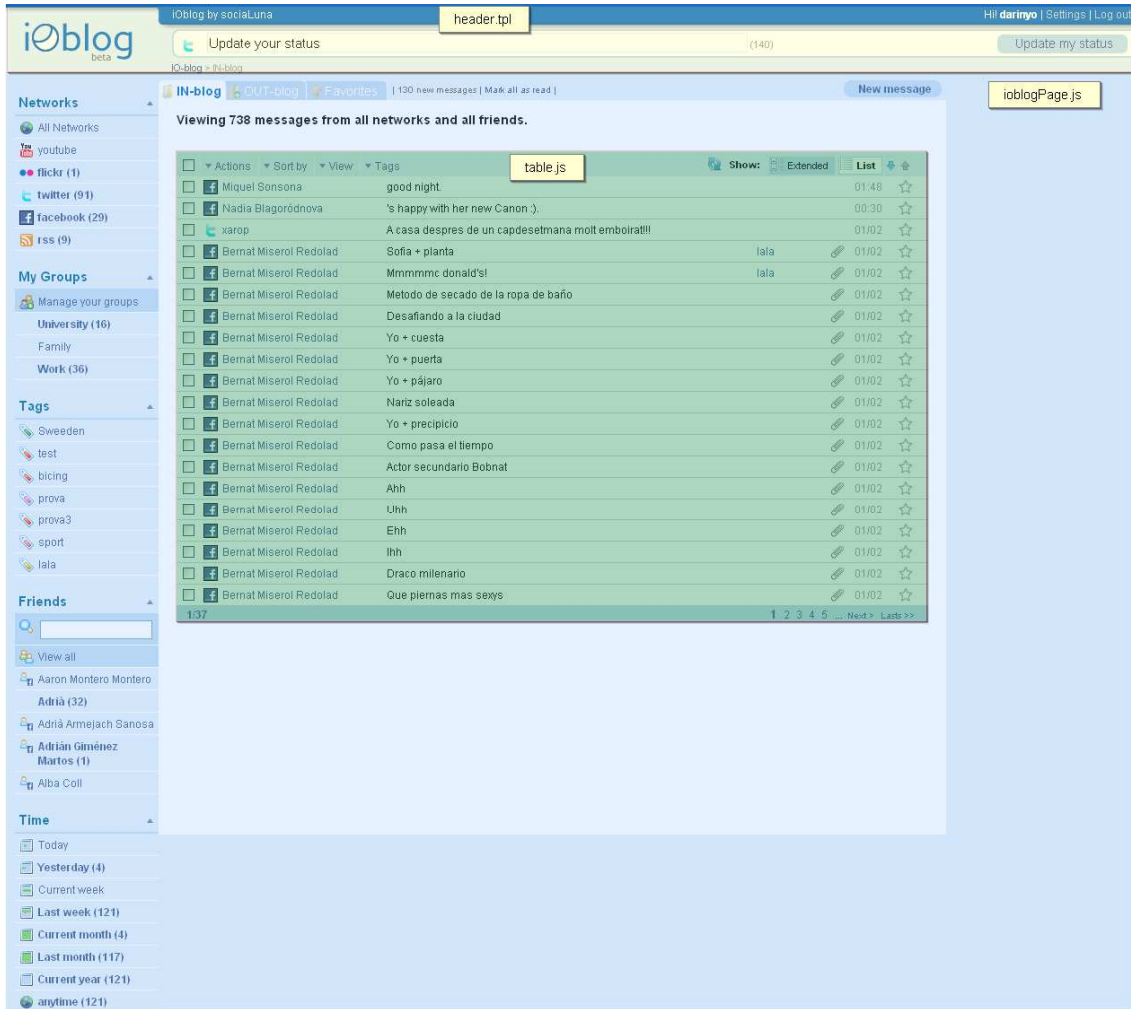


Image 25: IOBlog page components

The header is loaded by file header.tpl, loaded by main-full.tpl. After loading the header, main-full calls to index.tpl that will load the JavaScript file ioblogPage.js. This is responsible of loading all the page content and the left menus and of creating the object messages table calling to JavaScript file table.js.

7.5. Internationalization (I18N)

To create different dictionaries from all strings in the webpage, it was used an application called Poedit. This application allows capturing all strings in a web page, inserting a translation and saving a dictionary. After, a dictionary can be chosen to print the strings in the language desired. To do the translation, Zend framework has a function named `t`. When you write a string in this function, the framework goes to selected dictionary to get the translation. If this translation not exists, it shows the default language.

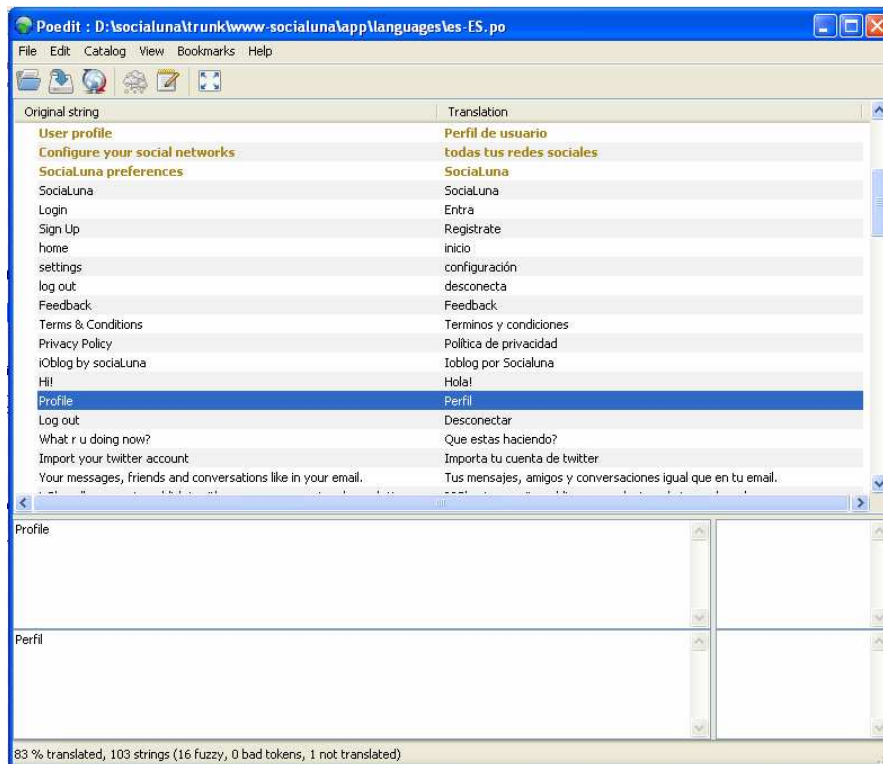


Image 26: POEdit

8. SocialLuna Back-End

In this section I will describe SocialLuna platform to get a better understanding about all project scope. I will focus this description in explain the architecture used to create the platform and all the components that form the platform. Finally, I will explain the servers used to run the application and the interaction between these servers.

8.1. SocialLuna Architecture

The first point that I would like to explain is the platform structure and its operation process.

All user requests from the services created over SocialLuna platform are received by a web server. If it is necessary to access external networks (import profile, sending a new message, etc.), it sends a message to the connector through a system of asynchronous queues. This connector will be responsible of sending the request and receive the reply from the external network. This will avoid bottlenecks caused problems outside SocialLuna.

The next picture shows in more detail the structure of SocialLuna platform:

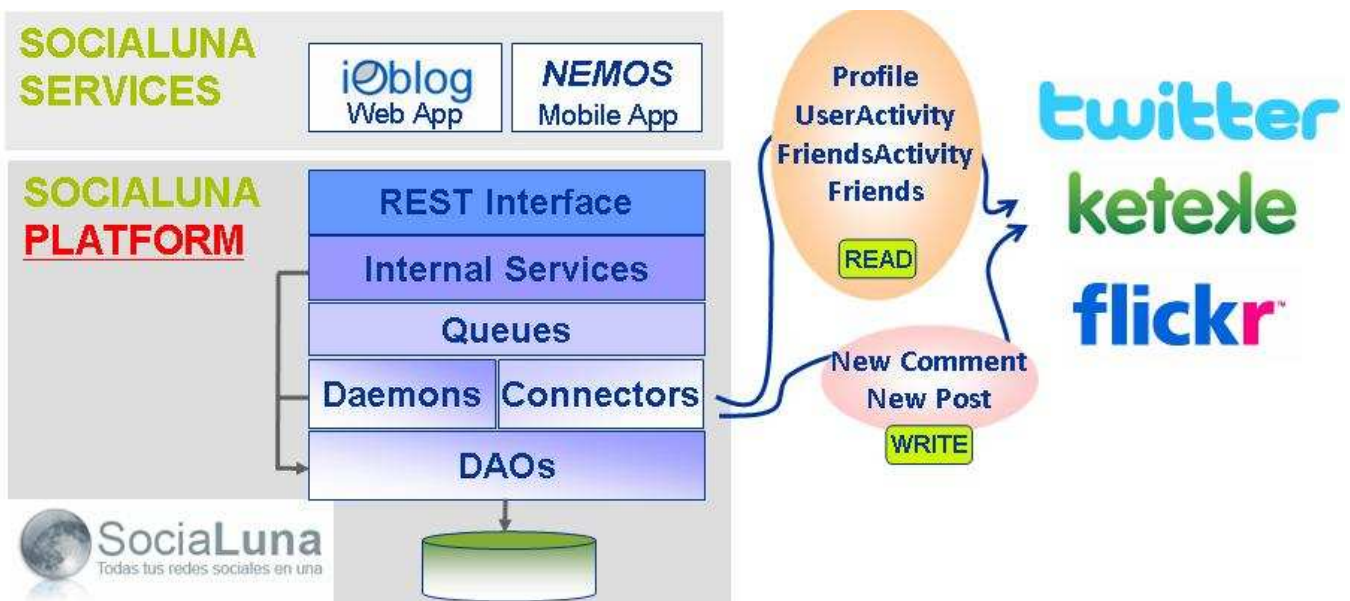


Image 29: SocialLuna architecture (I)

The basic blocks of SocialLuna platform architecture are:

- ◆ **Rest interface**: HTTP interface to access to SocialLuna functionality. It is the interface used to access the service capabilities of the platform. Using a REST syntax allows to easily identify the resource being accessed.
- ◆ **Internal Services**: REST layer is only used as a wrapper and is supported by the internal services layer to perform the necessary tasks. A service includes one or more functions grouped by subject (user profile, posts management, networks management, etc.).
- ◆ **Queues**: asynchronous queues that serve as a means of communication with the connectors that send and receive information from external networks.
- ◆ **Daemons**: processes that run periodically and are responsible to import user data that have been created/modified in the external networks in the last time interval.
- ◆ **Connectors**: processes that connect to external networks when they receive a signal in the form of asynchronous message through the platform queues.
- ◆ **DAOs**: access to Data Base layer.

The complete SocialLuna platform architecture is shown in the next image:

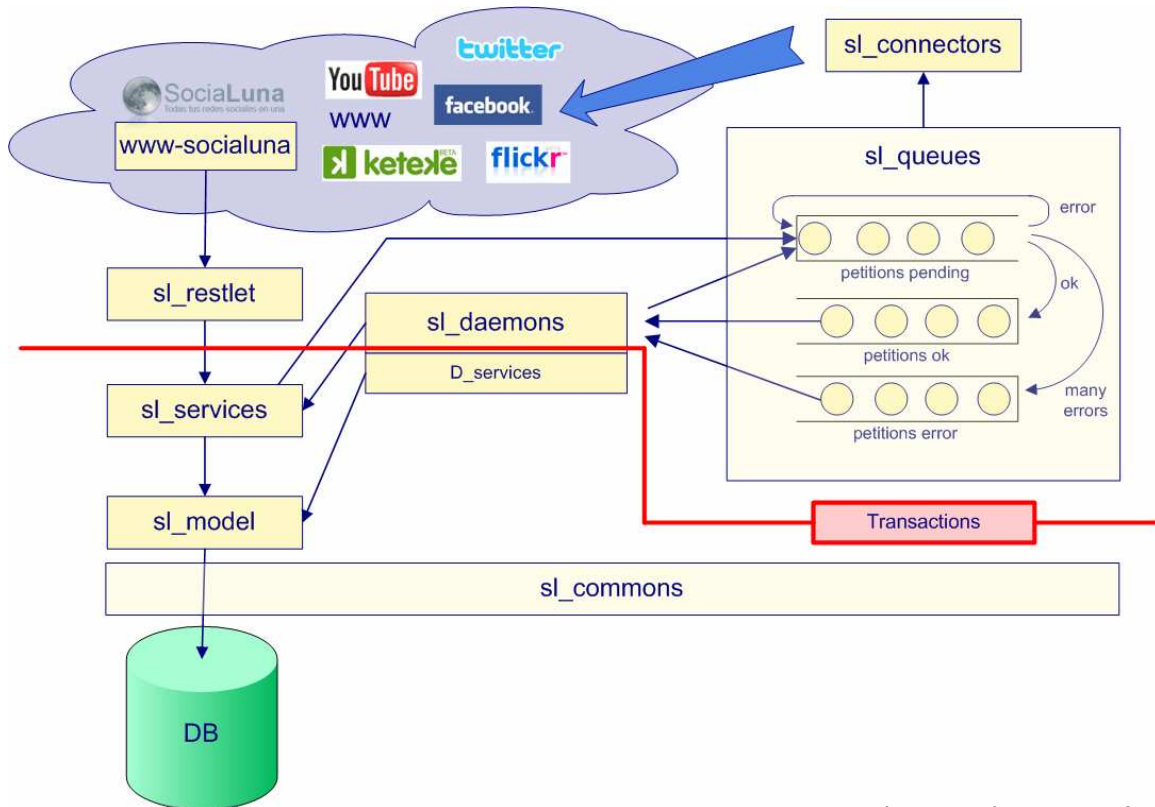


Image 30: Socialluna architecture (II)

At previous image, we can see all extended architecture of SocialLuna platform.

The enter point to the platform system is on **www-socialuna**. This part of the diagram is the front-end application that interacts with the platform. This front end application can be the IOBlog discussed above or any other application with a similar purpose. In Annex 2 there are explained other applications developed in the company over SocialLuna platform.

The layer **sl_restlet** has two functions. When this layer receives a request from a front-end application is responsible to translate this call to an understanding system call or, in other words, translate a URL path to his correct service system call. When the service called return the data to return to application, this layer is the responsible of translating these data to the format desired by the application. This format can be JSON or XML.

SI_services layer is the layer with all business logic. In this layer are implemented all functions that control the input to the system and execute the needed action in any case. If the service demanded requires connecting to an external social network, it will push into **sl_queues** the petition. If the service requires save or restore data into the data base it will call to the **sl_model** layer.

The layer responsible of interacting with the data base is the **sl_model** layer. In this layer are defined all calls to the data base under hibernate mapping. In this layer are launched the queries to the data base to get the data or insert or modify new data into system data base. Also includes the necessary logic to create the data base and the POJOs (Plain Old Java Objects).

SI_daemons has functions that run periodically. These functions normally connect with the external networks to get some user information (new posts, user profile, new friends...). Also is the receiver of all the changes in the queues states to add a new task on pending queue if a task has failed or return the data returned by the network to the service who called it.

SI_queues is the layer responsible to manage the different queues of the platform running on an **Apache ActiveMQ queues server**. Socialuna has three queues for each connector associated. The queue pending has all requests to the external network pending to be executed. When the request is made, the task can go to other two queues depending on the outcome. If the request fails, the task goes to the error queue to analyze the error and create a new pending task if necessary. However if the query returns correctly, the tasks goes to the complete queue that will return the data received.

Finally, **sl_connectors** has all the functions needed to interact with an external network. In some cases these connectors are open API's published by the network.

8.2. SocialLuna servers

SocialLuna platform is running under different servers. In image below I show these servers running over the SocialLuna architecture graph:

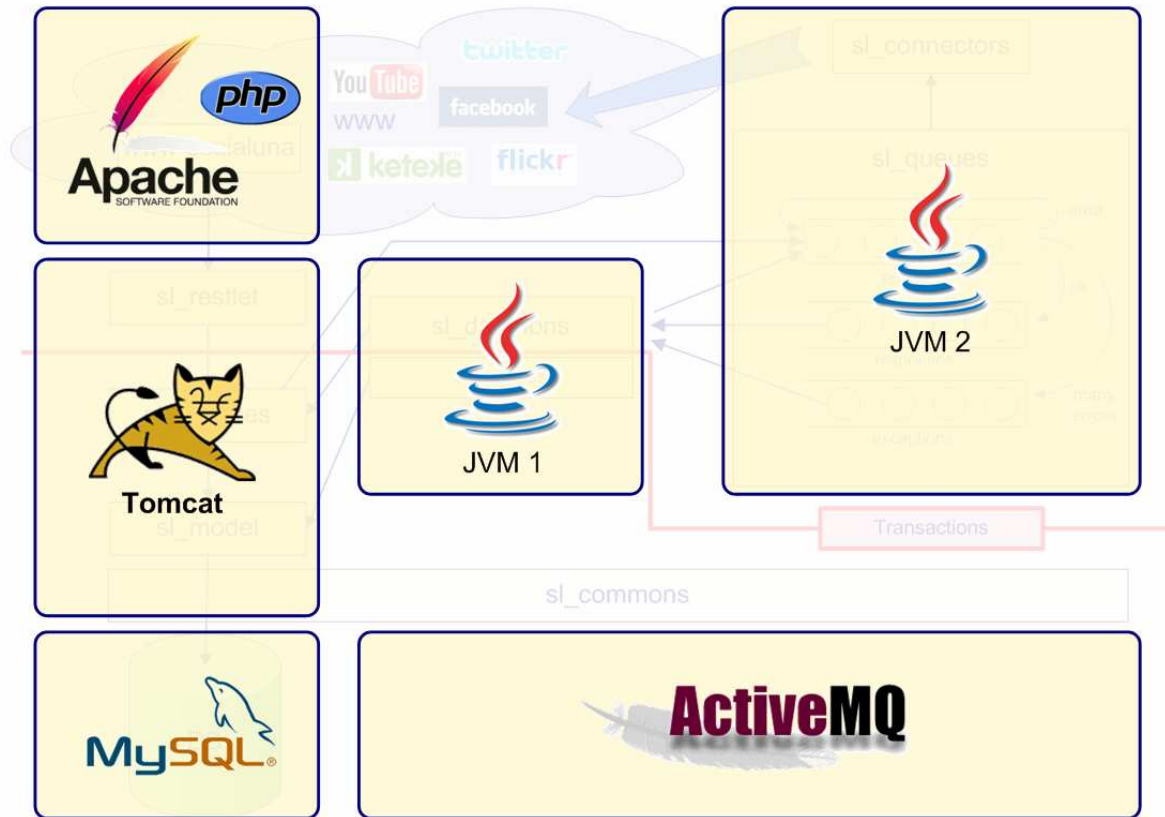


Image 31: Socialluna architecture (III)

The enter point to the platform is an Apache server that manage the Front-End application. The petitions to Back-End platform enter by a tomcat server. The daemons are running in a Java Virtual Machine (JVM1), in other hand, connectors and queues are running in a different one (JVM2). Therefore, these could be on different servers.

All messages between the systems are managed by Apache Camel and ActiveMQ. The servers write and read the messages in the ActiveMQ server to pass the information between the platform elements.

9. Conclusions

In the current Internet situation, users navigate in several social networks and, in many cases a user is registered in many of them. These users commonly need to sign in into multiples sites to retrieve all their social activity. SocialLuna solves this problem showing to the users all their social activity altogether allowing at the same time to reply and interact with these networks.

This situation also drives firms to invest in different projects related to social networks and SocialLuna is one of them. Every year there are new social networks and it is important that the platform allows future additions to increase the scope of the applications under the platform.

At this moment, the project is a real application with all its basic use cases implemented and running. Nevertheless, the project needs further development to become a final web application. Some of these will be explained in the next point.

As personal conclusions:

On these months in Telefonica R&D I have been working on web development that is one of the most active areas within the IT sector due to the number of potential users. This work has helped me to gain experience in an important company of the sector, and at the same time, to deal with a real project in a company.

Finally, my stay in the company has helped me to learn a lot about web technologies currently used in commercial projects. I have also learned several techniques of projects management inside the agile methods such as Scrum.

10. Future work

The future work to further improve the application is endlessly. I'm currently working to improve the navigation between Ajax calls enabling the default navigator back and forward actions. It is also necessary to improve the usability in some use cases to get a better user experience. At the same time, the number of social networks in Internet is very extensive and SocialLuna works as an aggregator. Adding new social networks to SocialLuna always provide an important enhancement to the platform, and the applications under the platform.

The next Social Networks to be imported on IOBlog application are keteke network and MoviStar Mobile Forum. The first social network is a social network of Telefonica similar to facebook or tuenti. MoviStar Mobile Forum will allow to MoviStar clients import this service to send from IOBlog SMS or MMS messages to their mobile contacts.

Other possible feature to add to SocialLuna is the grouping of image albums in a single post to avoid the overfreight of users' inbox when his/her friends upload albums of many images to their social networks.

Finally to do of IOBlog and SocialLuna a commercial application is needed to invert some time in the study of the application general performance with a large number of users, and work to reduce the waiting time using caches and other techniques for this purpose.

Annex 1. Final result and user’s manual

The final result of IOBlog application is a web page that using SocialLuna platform allows to users the aggregation of their social networks. At this moment, the social networks allowed are Flickr, twitter, facebook and youtube. In the next pages I show the final result of these months developing the applications as a simple user manual with screenshots of the application.

Teaser page:

As all web pages IOBlog has a teaser page where you can introduce your username and password to authenticate your account and enter to the application.



Image 32: IOblog login screen

Dashboard page:

When you authenticate your account as first time will appear a simple web page introducing to the most important functions that you will see in the application.

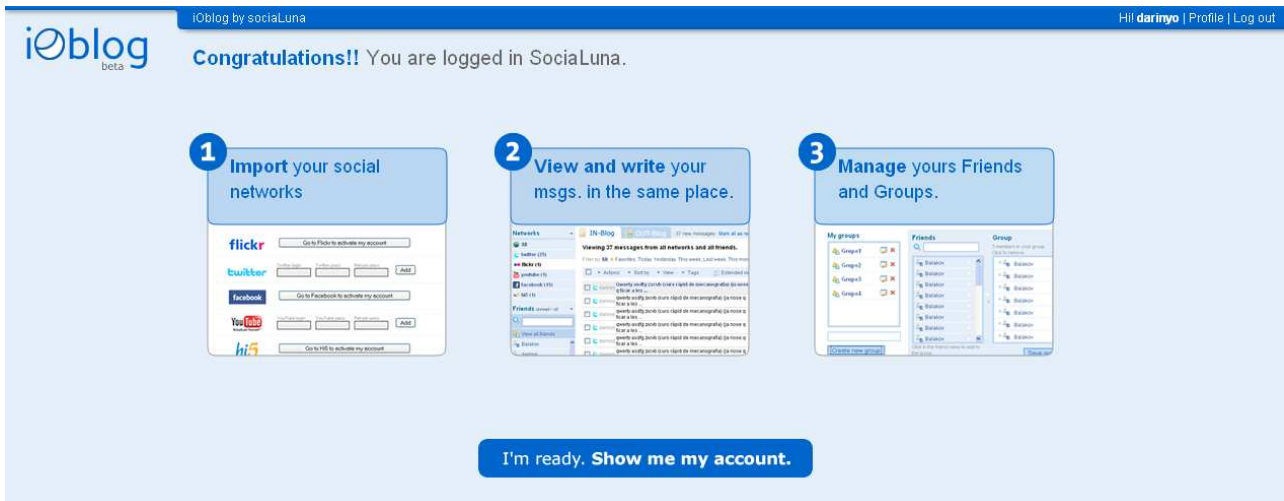


Image 33: IOblog dashboard

Importing social networks:

When you don't have any social network imported, IOBlog redirects to the import page to start including your social networks and thus, start using the application.

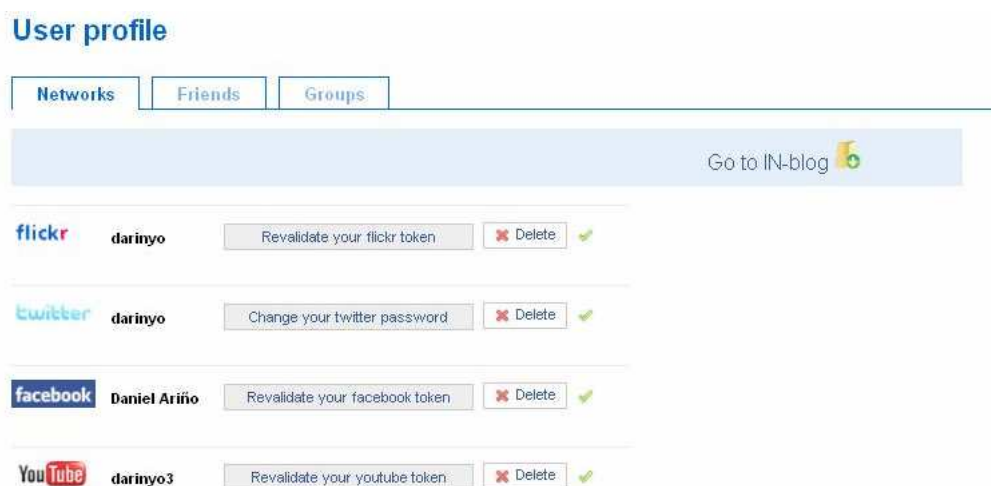


Image 34: Networks imported

Networks management:

Once you have imported your social networks, you can remove these networks or change the password at any time. As is explained in the use case, change the password of a social network only change the password on SocialLuna platform and doesn't change it in the external network. The passwords should always match in SocialLuna and external networks; otherwise IOBlog will show an authentication error and will help you to introduce the correct password.

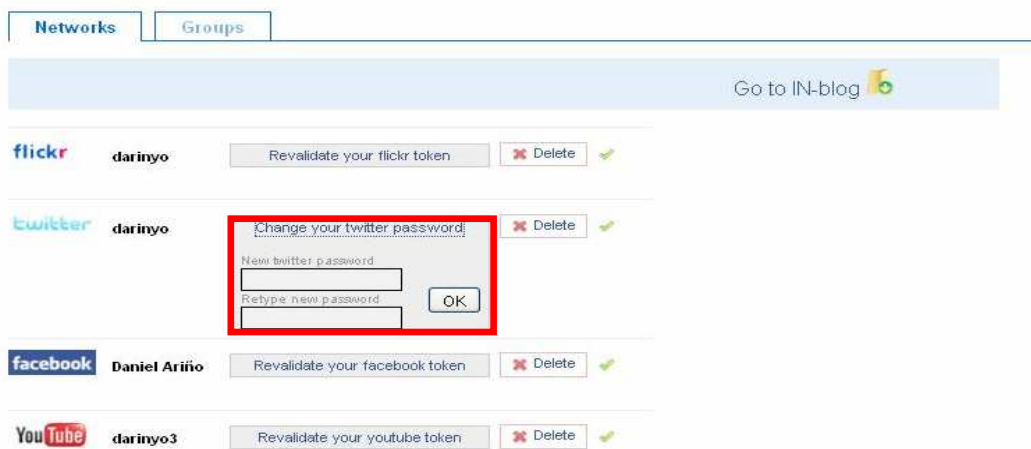


Image 35: Chanaina Social network

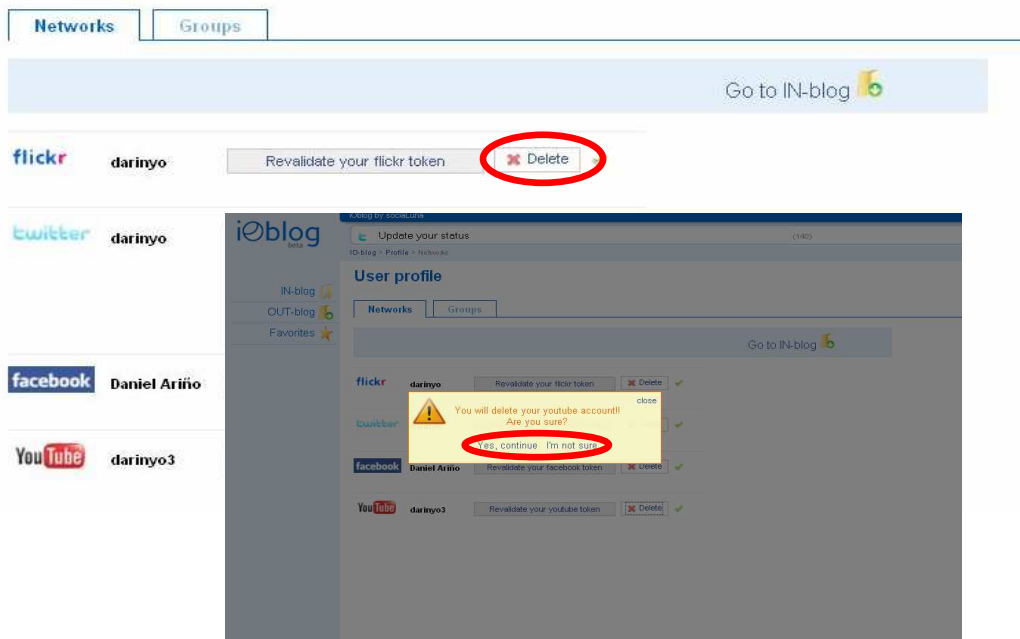


Image 36: Deleting a network from Socialuna

Viewing your messages:

Once imported your social networks, you can start to navigate by your inblog to see the messages of your friends in these networks and outblog to see your messages in the networks.



Image 37: Messages table in INBLOG40

You can choose between two different table visualizations to view your messages in the way that you can see more easily all the information of your interest.

The first view is a list view with all basic information in one line to do the table more compact and easy to read.

The other view is an extended view with more details and the friend avatar to do more easy to see at users the difference between all their messages.

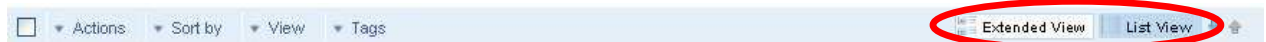
<input type="checkbox"/>	juandebravo	learning how to develop a hi5 application		15:41	☆
<input type="checkbox"/>	Diana Pascual	echa de menos la siesta...		14:41	☆
<input type="checkbox"/>	SocialToad	probando adobe air		13:34	☆
<input type="checkbox"/>	Adrià Julià Lundgren	parece que la inspiración ya vino .. no es de las mejores pero mejor q ...		11:50	★
<input type="checkbox"/>	xarop	parece que la inspiración ya vino .. no es de las mejores pero mejor q ...		11:50	☆
<input type="checkbox"/>	juandebravo	manifestación contra la masacre en Gaza		09:31	☆

Image 38: Messages table (List View)

<input type="checkbox"/>		13/01/2009 15:41:02 from twitter by juandebravo learning how to develop a his application	
<input type="checkbox"/>		13/01/2009 14:41:27 from facebook by Diana Pasual echa de menos la siesta...	
<input type="checkbox"/>		13/01/2009 13:24:34 from twitter by SocialToad probando adobe air	
<input type="checkbox"/>		13/01/2009 11:50:10 from facebook by Adrià Julià Lundgren parece que la inspiración ya vino .. no es de las mejores pero mejor que estarse parado ...	
<input type="checkbox"/>		13/01/2009 11:50:09 from twitter by xarop parece que la inspiración ya vino .. no es de las mejores pero mejor que estarse parado ...	
<input type="checkbox"/>		13/01/2009 09:31:44 from flickr by juandebravo manifestación contra la masacre en Gaza	

Image 39: Messages table (Extended View)

To toggle the selected view, you can click the buttons in table header with names "**Extended view**" and "**List view**".




To view more information about a single post, you can click over your interest post. This post will open showing under it some information and other options to do with the post as reply to user, **post a comment** or **view all its comments** in a permanent link to share it with friends.

juandebravo fsdfd test 09/01

xumi_clot Hate / Love 5 28/11/2008

Hate / Love



5 comments:

19/01/2009 11:06:25 by Nanoniano
dudu á!

19/01/2009 11:04:07 by Nanoniano
dudu!

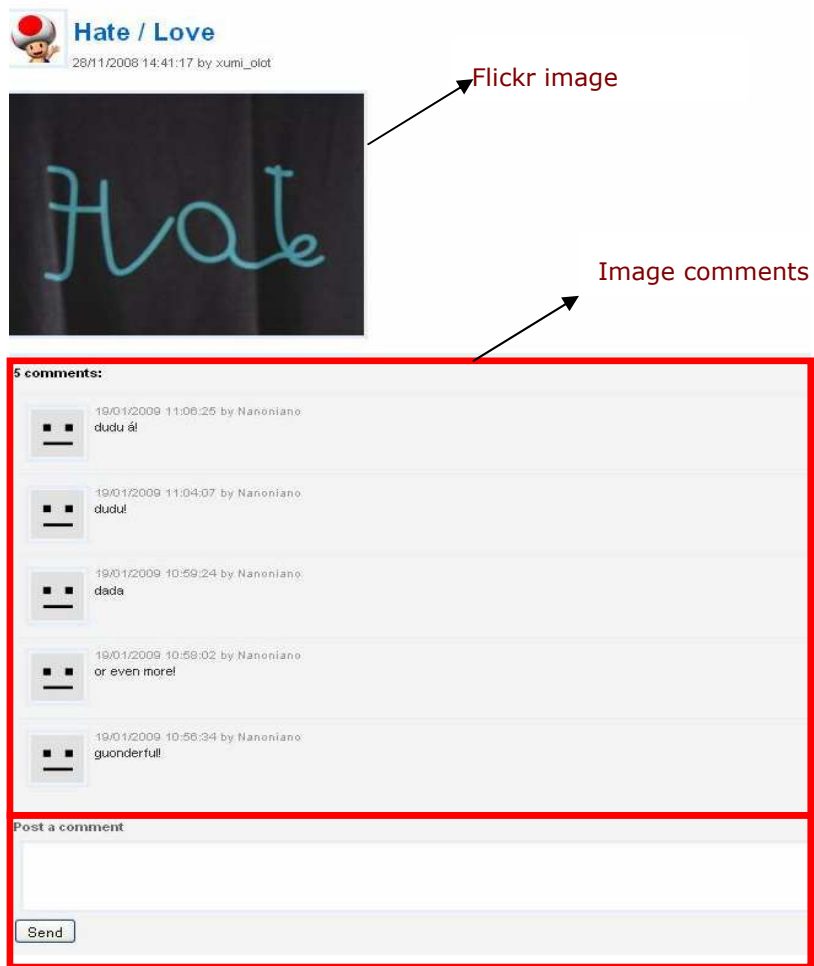
[View all comments](#)

[Post a comment](#)

<input type="checkbox"/>		One of your friends has comment this post	07/11/2008	
<input type="checkbox"/>		One of your friends has comment this post	3 07/11/2008	
<input type="checkbox"/>		One of your friends has comment this post	1 06/11/2008	

Image 40: Opened post information

When you click in the link "view all comments", a new page will be loaded with the post information, all posts comments and a text area to send a comment or reply to user message owner.



Flickr image

Image comments

Post a comment

Image 41: Permalink page

Manage your posts:

IOBlog have multiple functionalities to help you to manage your posts and to find quickly and easily all the messages that interest you. IOBlog have different filters to see in the messages table only the messages of your interest each time. The

filters created are **filter by friend** (allows view only the messages of a friend), **filter by network** (allows view only the messages of a unique network), **filter by date** (allows view the messages from a date), **filter by group** (allows view the messages of the friends included in a predefined group) or **filter by tag** (allows view the messages tagged with an specific tag).

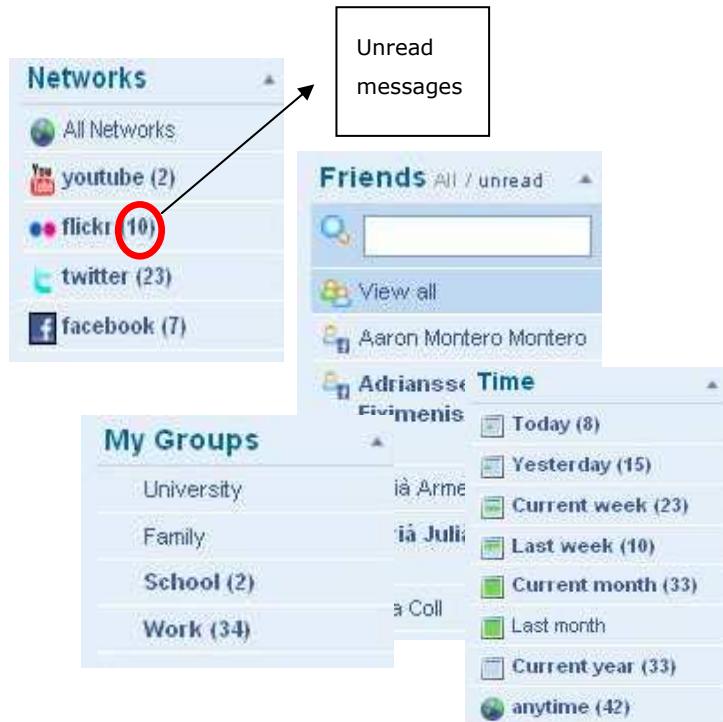


Image 42: Message filters

If you have imported all your social networks, is possible that the number of friends will be very high. To help you to find a friend, IOBlog have a searcher that filter the friends showed in the list showing only the friends that contains the string written on the text area.

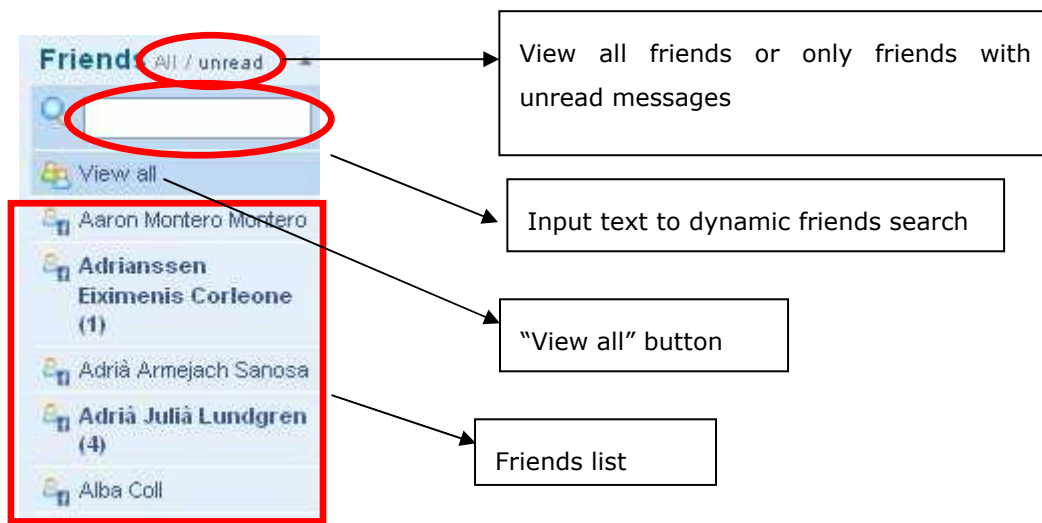


Image 43: Friends filter menu

By default, the friends list shows only the five first found friends. To show all your friends in the list, click over the button **“View all”**. The buttons **“all / unread”** allows to filter the friends list with all friends or only the friends with unread messages.

Other option to found your messages is order the messages in the table. You can order these messages by network, received date and friend message owner. To order the messages you only need to click in the menu **sort by** on the table header.



Image 44: Sort messages options

In the same way you can click in the **"view"** menu to select view all messages or view only the unread messages.



Image 45: Message View Options

Manage your favourite messages:

You can mark your favourite messages to get a quick access to them. To mark a message as favourite you only need to click over the star at right of the message. If the star is empty indicates that this post is not a favourite post. Otherwise, if the star is full, this post is a favourite post. Another way to mark as favourite different posts at the same time is selecting the messages checkboxes, going to actions menu and selecting the option mark as favourite.



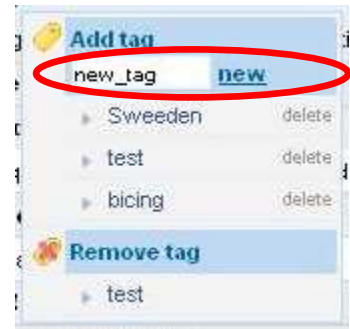
Image 46: Favourite messages



Image 47: Mark as favourite (multipost)

Tagging your messages:

Another option to manage the posts is the creation an association of tags with the messages. To create the tags you only need to click over **"tags"** menu, enter a tag name in the text area appeared and press enter key or click over **"new"** button.



To assign a tag to your posts you only need to select the posts checkboxes that you would like to apply the tag and click over the tag name in the tags menu.

Image 48: Tags menu

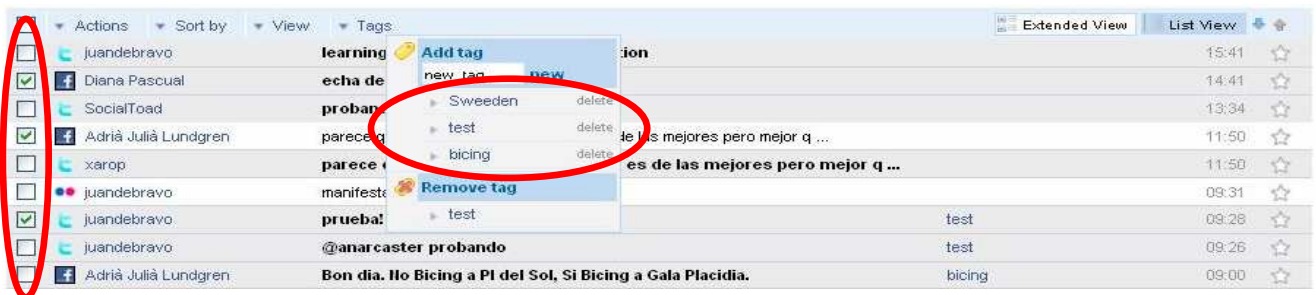


Image 49: Tags assignation

To remove a tag from a post you only need to open the tags menu with the posts that you would like to remove the tag selected. In the "Remove tag" menu will appear the tags of the selected posts. Clicking over a tag you will remove this tag from the selected posts.



Image 50: Remove assigned tags

Finally, you can delete a tag. To delete a tag you only have to click over delete button at right of the tag name. A message will appear advising that all the posts with the tag assigned will lost the association with the tag. Click on continue deleting to delete the tag or cancel to not delete the tag.

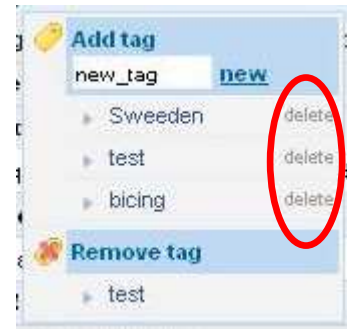


Image 51: Tags menu

To view all the messages tagged with a label click in the tag name on the filter at left menu or over the tag name in a message post. To view all your tags created, create new tags or remove existing tags you can also go to menu settings and click over tags tab. There you will see all your tags with the ten latest posts from each.

Create your groups:

Also in the menu settings you will find the tab "groups". There you can create groups of friends. With these groups you will have more effective filters to find at each moment the posts of your friends.

To create a new group you only have to enter the group name on the text area and click on create new group. To add friends to a selected group, click over the friend and this will change to the list of friends in the group. You have a search engine as described above to find your friends more easily. To save the group, click on the button "save group".

User profile

Manage groups

Create and manage groups of friends. ?

Friend's searcher

Selected group

My groups

- Work
- School
- University
- Family

Friends

- TiTo Español Gamon
- Victor Flores
- Bernard Edg Dca
- Daniel Morales
- Josep Llansana Riba
- Quim Vilà
- Eric Palau Erena
- Ferran Giral

Group

Click to remove from the group

Work

- xarop
- Gemma Hornos
- tidbcn
- diluvi
- xumi_clot
- pepitogrillo
- nanoniano

Click in the friend name to add to the group.

Save group

Create new group

List of friends

List of friends in group

Save group

mage 52: Groups update

To remove a group, click on the icon X, next to the group name. A warning message will appear. Click "continue" to remove the group or cancel to don't remove.

Note: Remove a group doesn't remove the friends inside the group, only remove the association between theirs.

Manage groups

Create and manage groups of friends. ?

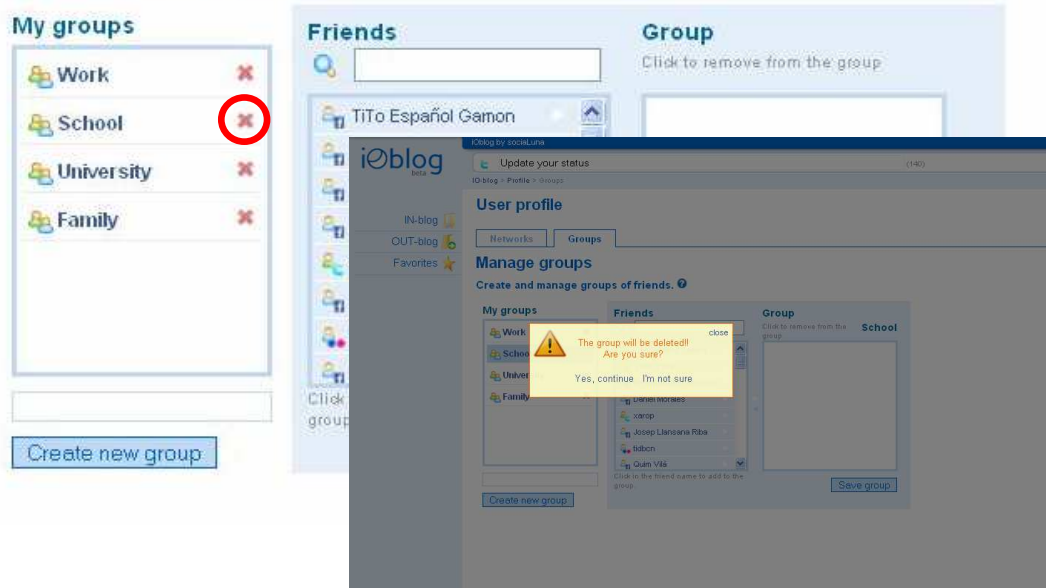


Image 53: Groups delete

Send messages to your networks:

On IOBlog exists different ways to send a message to your networks. A fast way to upload your status on twitter will appear in the header page when you will have imported twitter network into IOBlog. This way is fast and easy, you only have to write your status on the text area and press enter key or click on "update my status" button.

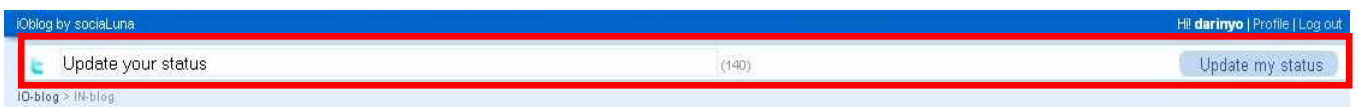


Image 54: Fast post message

To send a message to all your networks go to "new message". There you can write a new post to send to one or more networks. First of all you want to select the type of message (text, image or video). IOBlog will show the available networks to this type of message. Select the networks where you want to send the message, field the form fields and click in "send the message" button

[30] To update facebook status or upload images to facebook is necessary to obtain extended permissions from facebook. When IOBlog loads the page to send a new message obtains from facebook the permissions of SocialLuna in user facebook account. If no permissions have been given to send the message, it shows and alert with a link to facebook to obtain the needed permissions. We decided to check the permissions when the user tries to send a message to facebook, instead of asking for these permissions when the user imports the account, because the import steps would be too long and too many steps will be necessary.

New Message

The screenshot shows a 'New Message' form with several sections:

- Select type of post:** Radio buttons for Text, Photo, and Video. An annotation points to this section with the text 'Selecting message type'.
- Post in:** Checkboxes for flickr and facebook. An annotation points to this section with the text 'Selecting network or networks to send the message'.
- Warning:** A yellow box with a warning icon and the text 'You must allow IOBlog to upload photos on Facebook' and a 'Go to Facebook' button. An annotation points to this box with the text 'Facebook is not available because user has not allowed permissions to Socialuna to update images to Facebook.'
- Form fields:** Input fields for 'Title', 'Text', and 'Upload image' (with an 'Examinar...' button). An annotation points to this entire section with the text 'Form fields to upload'.

Image 55: New message page

Annex 2. Other SocialLuna Applications

Other projects in Telefonica R&D are being developed under SocialLuna platform. These projects are different front-end application to different supports.

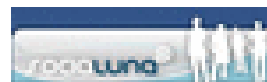
Nemos is a mobile phone application that using SocialLuna platform allows users the access to their Social Networks, download and upload content and geolocate these content.



Image 56: Nemos application



Image 57: Socialluna air application



Other application for **SocialLuna** is an Adobe Air client developed with Adobe Flex. This application has similar use cases that IOBlog application, but unlike this, is a desktop application instead of a web application.

Glossary

Apache License

Apache License is a free-software license authored by Apache Software Foundation that allows the use of the source code for the development of free and open source software as well as proprietary software

Apache Software Foundation

Apache Software Foundation is a non-profit corporation to support Apache software projects, including Apache HTTP Server or Active MQ.

API (Application Programming Interface)

An API is a set of functions and procedures that offers a service to be used by other service as abstract layer.

BSD License

BSD license represent a family of permissive free software licenses, original for the Berkeley Software Distribution (BSD). The licenses have a few restrictions compared to other licenses as GPL.

CDDL (Common Development and Distribution License)

CDDL is a free software license produced by Sun Microsystems. The Free Software Foundation considers it a free license incompatible with GNU GPL. Some projects under this license are Solaris Project or NetBeans.

Copyleft

Licensing group which aims to ensure that each person receiving a copy of a work can in turn use, modify and redistribute the work and derived versions of it.

CSS (Cascading Style Sheet)

CSS is a style sheet language used to describe the presentation of a document written in mark-up language. It's most common application is to style web pages written in HTML and XHTML.

DI (Dependency Injection)

Dependency Injection is an architecture pattern, object oriented, in which objects are injected in a class instead create the objects in the same class. To implement this pattern is needed a DI Container and Object POJO. The container injects to each object, the needed objects defined in a configuration file. Usually, this container is an external framework; in this case, the framework selected is Spring.

DOM (Document Object Model)

DOM is a platform for representing HTML or XML documents as well as an API for querying, traversing and manipulating such elements.

EPL (Eclipse Public License)

EPL is an open source software license used by the Eclipse Foundation for its software. EPL is designed to be a business friendly free software license, and features weaker copyleft provisions than contemporary licenses such as GPL.

Framework

A framework is a re-usable design for a software system. This may include support program, code libraries, a scripting language or other software to help develop and glue together the different components of a software project.

GNU GPL (GNU General Public License)

GNU GPL is a widely used free software license for GNU project. The GPL is the most popular example of the type of strong copyleft license that requires derived works to be available under the same copyleft.

HTML (Hypertext Mark-up Language)

HTML is the predominant mark-up language in Web pages, designed to structured texts and present these as hypertext format.

LDAP (Lightweight Directory Access Protocol)

LDAP is an application protocol for querying and modifying directory servers running under TCP/IP.

MIT License

MIT license is a free software license originating at Massachusetts Institute of Technology (MIT). This license allows reusing the software as free software or as proprietary software allowing doesn't release the changes on the original software.

POJO (Plain Old Java Object)

A POJO is an ordinary Java Object and not a special object. As designs using POJOs have become more commonly-used, systems have arisen that give POJOs some of the functionality used in frameworks and more choice about which areas of functionality are actually needed. Hibernate and Spring are examples.

Template

A web template is a tool used to separate content from presentation in web design. Web templates can be used to set up any type of website.

Webpage

A webpage is resource of information that is suitable for the World Wide Web and can be accessed through a web browser. This information is usually in HTML or XHTML format and provides navigation to other web pages via hypertext links.

XML (Extensible Mark-up Language)

XML is a metalanguage extensible labelled developed by the World Wide Web Consortium (W3C) that can define the grammar of specific languages. XML is proposed as a standard for exchanging structured information between different platforms.

BIBLIOGRAPHY

Social networks:

- [1] MySpace - <http://www.myspace.com>
- [2] Orkut - <http://www.orkut.com>
- [3] Yahoo 360º - <http://360.yahoo.com>
- [4] Flickr - <http://www.flickr.com>
- [5] Twitter - <http://www.twitter.com>
- [6] Facebook - <http://www.facebook.com>
- [7] Hi5 - <http://hi5.com>

Social networks history:

- [8] <http://www.maestrosdelweb.com/editorial/redessociales/>
- [9] http://en.wikipedia.org/wiki/Social_network
- [10] http://es.wikipedia.org/wiki/Redes_sociales
- [11] http://en.wikipedia.org/wiki/Small_world_experiment
- [12] <http://www.oxyweb.co.uk/blog/socialnetworkmapoftheworld.php>
- [13] <http://www.techcrunch.com/2008/08/12/facebook-is-not-only-the-worlds-largest-social-network-it-is-also-the-fastest-growing/>

Scrum, agile methods:

- [14] <http://www.scrumforteamssystem.com/processguidance/v2/Scrum/Scrum.aspx>
- [15] Flexibilidad con Scrum - Juan Palacio
(<http://www.navegapolis.net/content/view/694/61/>).
- [16] Scrum & XP from the Trenches - Henrik Kniberg

Project technologies

- [17] <http://es.wikipedia.org/wiki/Tdd>
- [18] <http://en.wikipedia.org/wiki/PHP>
- [19] <http://framework.zend.com/about/overview>
- [20] <http://www.smarty.net/>
- [21] <http://es.wikipedia.org/wiki/JavaScript>
- [22] <http://en.wikipedia.org/wiki/AJAX>
- [23] <http://es.wikipedia.org/wiki/JSON>
- [24] <http://en.wikipedia.org/wiki/JQuery>
- [25] <http://www.restlet.org/about/>
- [26] <http://www.hibernate.org/>
- [27] <http://activemq.apache.org/camel/>

User authentication

- [28] Flickr: <http://www.flickr.com/services/api/auth.howto.web.html>
- [29] Facebook:
http://wiki.developers.facebook.com/index.php/Authorizing_Applications
- [30] Facebook: http://wiki.developers.facebook.com/index.php/Extended_permission
- [31] Youtube: http://code.google.com/intl/es-ES/apis/youtube/developers_guide_protocol.html#AuthSub_Authentication

Special acknowledgments

I like to spend the last page to thank everyone who made this project possible. I would like to thank specially:

- ◆ All team project (Edraí, David, Juan and Pepe) for help me in all I need with the project.
- ◆ Ivan for introduce me in JavaScript and Zend world.
- ◆ Carme and Juan for tutoring the project
- ◆ Karla, Joan, Judit and Valentina for help me with English revisions