KARELIA UNIVERSITY OF APPLIED SCIENCES
Information Technology

Decock Andy, Moreels Pieter-Jan,Zsombik Kenny

CONSOLIDATING STUDENT SERVICES WITH LIFERAY

Thesis
June 2014

Karelia UNIVERSITY OF APPLIED SCIENCES	**THESIS** **June 2014** **Degree Programme in Applied Informatics** Karjalankatu 3 FIN 80200 JOENSUU

Author(s)

Decock Andy, Moreels Pieter-Jan, Zsombik Kenny

Title

Consolidation of Student Services with Liferay

Abstract

This thesis describes some of the possibilities of Liferay. Liferay is an open-source portal framework, which can be used to create multipurpose websites. To begin the thesis project we first needed to analyse the capabilities of Liferay and to determine if it was a feasible framework for a student web portal, Only afterwards we were able to start creating a prototype.

As the first step of the thesis project we made an action plan which describes the project itself and the wishes and needs of the client. In this thesis report we then present Liferay and explain some important parts it uses. After that we also discuss the consolidation of Intalio with Liferay. This section will be followed by the development scheme. Including the theme and the portlets we created for the prototype. This section of the report also discusses about Vaadin. A portlet programming language based on Java, which makes it possible to create and manage complicated and complex JavaScript-based web components.

After the section that describes the development of the portal prototype, we also discuss how external data can be integrated into Liferay. Moreover Liferay and SharePoint are compared and analysed. In the concluding section of the thesis we highlight some problems we encountered or some remarks we noticed while creating the prototype.

.

Language	Pages Appendices Pages of Appendices
English	

Keywords

Electronic services , Liferay, Sharepoint, software development

Contents

LIST OF TABLES

LIST OF IMAGES

APPENDICES

Acronyms

URL	Uniform resource locator. Known as web address
HTTPS	Hypertext transfer protocol secure is a communication protocol for secure communication over a computer network. Which is commonly used on the internet
LTS	Long-term support. Is a product lifecycle management policy for computer software
HTML5	Hypertext Mark-up Language. Language used on the internet for presenting content
JSP	JavaServer Pages is a technology that helps software developers create dynamically generated webpages.
GNU	GNU's Not Unix is a Unix-like computer operating system
TCAT	Tomcat is an open source web server and servlet container
GB	Gigabyte
CPU	Central processing unit is the central part of a computer/server
DDR	Double data rate
BPMS	Business process management
DB	Database
AJP	Apache JServ Protocol is a binary protocol that can proxy inbound requests
GUI	Graphical user interface
UUID	Universally unique identifier is used on the internet to identify entities or objects
SDK	Software development kit
EJB	Enterprise JavaBeans is a server-side component architecture for modular construction of enterprise applications
WAR	Web application Archive is a JAR file used to distribute a collection of JavaServer Pages, Java Servlets, etc.
JAR	Java Archive is a package file format typically used to aggregate many Java class files and associated metadata en resources
Scr	Short of source
RWD	Responsive web design is a web design approach aimed at crafting sites to provide an optimal viewing experience.
GWT	Google web toolkit is an open source set of tools that allows web developers to create and maintain complex JavaScript front-end applications
UIDL	User interface description language is a JavaScript based language for expressing complex user interfaces
LSD	Last Ship Date is the a marketing term meaning the last date where software is delivered to a client
EOL	End of life. Term that means when the software is no longer supported by the developer
SMTP	Simple Mail transfer protocol is an Internet standard for electronic mail transmission
VPN	Virtual private network extends a private network across a public network such as the internet

1 INTRODUCTION

This thesis report explains the different architectural choices for the new student intranet of Karelia University of Applied Sciences (UAS). In this document two main options are discussed followed by a conclusion with our suggestion. The project consists of an analysis of the needs of Karelia UAS, regarding new portal site service. Then the report presents project were we create a prototype of for the portal site.

As mentioned before, there were two choices that we were exploring. The first option was Liferay which is an open source solution for making modern websites and portals. The reason why Liferay was being considered is simple: it is open source and there is an active community that can help whenever roadblocks occur. This community can help in the development of a completely new portal. On the other hand there is the pricing issue. Liferay offers two solutions; the Community and the Enterprise Editions. The software provided by Liferay is very similar for both editions. The difference lays in the price, support and out-of-the-box features.

The second option that we explored was SharePoint. This platform is offered by Microsoft. It is a very powerful platform but it has its limitations. Unlike Liferay, it is closed source software. This means that the default web parts cannot be changed. If a certain web part does not meet the functional requirements, it must be completely removed and rewritten. Therefore, before selecting a default delivered web-part, the buyer must be sure it will be adequate for his purpose. The prices are higher as well, which plays a important role in choosing the platform for the new student website.

2 ACTION PLAN

2.1 Project Background

2.1.1 Organization

We are three Belgian student from the Vives University of Kortrijk, who are in our final bachelor year of applied informatics. This is our final thesis and is commissioned by Karelia University of Applied Sciences (UAS). The university, which is also the home institution for our internship, is located in Joensuu North Karelia, which is a province of eastern Finland. The university offers various studies from Bachelor programmes to Master degrees. A few example the possible Bachelor degree studies are: Bachelor in Culture, Bachelor in Social Sciences, etc... The Master degree possibilities are: The degree Programme in the Development and Management of Health Care and Social services, Degree Programme in Technology Competence Management and The Degree Programme in Environmental Technology. Our tutors was Petri Laitinen from Karelia UAS.

2.1.2 Vision and Values

The vision of Karelia UAS is to offer high quality higher education. Karelia UAS is known for having an active role in regional development participating in research and development projects. Currently it is very important for educational institutions to be innovative and open to new ideas, and that is why Karelia UAS makes it possible for students and teachers from abroad to come and experience the Finnish way of life, while offering them a variety of international studies.

To ensure the future is guaranteed, they offer students seven fields of study where they can choose from. Those seven fields are divided into 18 degree programmes.

2.1.3 Project Rationale

The reason why this project was launched is because the current student services website is of Karelia UAS very fragmented and user-unfriendly. That is why the university requested for new portal website to which every available service could be centralized. The portal site must be easy to navigate, customizable and contain ing features of social networking and social media. The social features include interaction such as instant-messaging, which should be made possible between students and teachers. The customisable aspect will be easier to explain with an example: every student should be able to see his/her own personal calendar that includes all the courses s/he follows and other important events that apply to him/her. Since a few existing application modules are outsourced to another company, those modules must be integrated into the new portal website. The implementation of the existing modules must be user-friendly, which means that whenever the user has logged in, there will be no new login prompts.

When evaluating the site, we as exchange students, made some very important observations. For example, we noticed that although there was an option to choose the interface language, it did not affect much on what was displayed on the page. Without even changing the language in the drop down menu, we could clearly see that most of the text was displayed both in English and in Finnish. The use of two languages simultaneously reduces the effectiveness and makes the website less user friendly. Therefore, the new portal website should have a language selector that would show the page either in Finnish or in English. Moreover the current website had no uniform layout and therefore some pages completely change the placement of certain components. This reduces the usability of the website. In the new website it is obligatory to have uniform layout.

2.2 Goals and Project Description

2.2.1 Project Description

As mentioned above, the student portal that is currently in use at the university is very fragmented. A Lot of modules are on different websites, this result in the fact that people need to login multiple times to see certain things, like the grades for example. Because of this, the student portal is not very user friendly. That's why it is our goal to create a central website that includes all the current modules and will still be easy to navigate through.

In order to realize this we used an open source alternative for SharePoint, called Liferay. The main reason that Liferay is regarded as the main option is because of the price-quality factor. A Liferay portal website is a lot cheaper than a portal website made with competitor systems and the budget available for the university to spend is limited. In this document we will mainly discuss the difference between Liferay and SharePoint. We are aware that there are other big portal website developers out there, but discussing all of these would far exceed the scope of our project.

Since none of us have used or developed for Liferay, we needed to do some research, starting with making basic portlets and themes and work our way up and learn how to create a prototype. The main reason we chose to use Liferay is because it is cheaper than SharePoint, but also the fact that it is easier to extend and it has the ability to give a Facebook-like experience. The reason why the Facebook-like experience played a role in our choice is because the client requested to create a more personalized website where users have easy access to their own information and documents.

2.2.2 Strategic Objectives of the Client

By doing this project, the university's main goal is to receive a prototype of a centralized student portal website. This prototype has to include all the requested modules. Unlike the current websites they use, it is their goal to include all modules on one website and prevent multiple logins. Not only does it have to include all

current modules but it also has to be easy to navigate through and be user friendly. Aside from the prototype they would also like to receive a document that includes a comparison between multiple portal website systems, like Liferay and SharePoint. We created our thesis, the document you are currently reading, in a way that it also includes all the findings and analysis the Karelia UAS requested. Once the University of Karelia receives this document, they will start searching for investors.

2.3 Project Boundaries

Because of the vast size of this project, we had to set specific boundaries for making this project. Due to the limited time - three months - and the limited workforce and knowledge - three students -, this project would otherwise have been far too big to complete.

2.3.1 Analysis Current Workflow

In the current student web portal, there are a lot of different modules and most of them are located on different websites. This results in the fact that for every website the students get redirected to, they need to login again. Another problem is that each of these websites also have their own lay-out and navigation bar. Because the navigation bars have no relation with another site, it is impossible to return to the main site after being redirected unless you type the URL again. And a last major problem is the fact that most of the content is displayed in both English and Finnish. This gives us a lot of data that we do not need and makes it harder when we search for a certain module. There are a few things that are only in one language, the problem however is that these will only be able to be displayed in Finnish, even if we selected English as the displaying language.

We also noticed some alarming problems with the current setup. Upon closer inspection, we noticed that the current websites, both Moodle and Pakki, are not using a secure connection. This means that anyone who is keeping an eye on the network can get the passwords of people who are logging in, as well as any sensitive

data passing over the network. Due to this security threat, sensitive data can easily be stolen by a third party. This can be fixed by switching to a secure HTTPS connection.

2.3.2 Scope of the Project

As mentioned before, the scope of our project consists of creating a prototype for a new student portal website. This means that we will not provide a complete website but a test version. Not only will we deliver a prototype but we will also a document that includes our findings. These findings include information about the prototype itself, some administrative information, the ease to extend the website and some analysis we have done. The prototype included the integration of some existing modules we recreated in Liferay. These modules are a personalized calendar and the ability to view emails inside the website. Accompanied by these modules, we also provided an example of a general layout that can be used in the final website.

2.3.3 Boundaries

The project has to be finished in a limited time, in order to ensure the possibility, we needed to do detailed research on the current modules and decide which ones we were going to recreate in our prototype. After creating these modules, we had to make sure we tested them enough so that the other team members can continue their work without too much problems. To test our prototype we received two servers, one which was a normal personal computer, on which we installed Ubuntu Server LTS 14.04, together with Liferay. The other server was a Windows server with SharePoint installed. Since the university also requested a comparison between Liferay and SharePoint we needed to make sure we documented our research. This comparison also required some administrative information so that they will be able to decide which portal site suits their needs the best for the final website.

2.4 Description of Possible Solutions and Evaluation

2.4.1 Description of Solution

In order to ensure the project was completed within the limited time we needed to do a good analysis and learn to work Liferay as soon as possible. As mentioned before, we were working with Liferay, an open source alternative and competitor of SharePoint. Liferay allows us to program in a lot of different languages like Ruby, PHP, JavaScript and many others. For our project we mainly used a combination of JavaScript, Java, JSP and HTML5. However, before we can start the programming, needed to do some interviews. The interviewees we chose to interview were two students, teachers and our tutors. The interviews gave us a clear view of the needs that needed to be included in the new website. The analysis of the current websites was crucial. First of all this gave us an image of the improvements that were needed, and which modules the students and teachers use the most.

Once we had an overview of all the requirements and the all the modules used in the current portal we were able to start with the development our prototype. The available time was divided in several sprints. Those sprints usually took around three weeks. At the end of each sprint, we had a testing moment. This was needed because we wanted to ensure that during the development, no code is changed and that would result in a software crash. To make sure we made progress, we decided that the work should be divided into three. Each member worked on his own portlet. Once he successfully developed this on his own machine, he would compile it, and deploy it on the test server. The reason why we work like this, is because we wanted to minimize the risk that modules affected the integrity of the server. If for example the server should have crashed and we had to recover it, we would have lost precious time.

2.4.2 Expected Outcomes

As mentioned before, the university expected a prototype that included the existing modules. Of course this is a prototype and it would not have been possible to include all the current modules but by including some modules that have been adjusted to fit the new website, we will be able to tell what the difficulty will be when they create the final website. That's why the university expected to receive an analysis document as well. This document contains any information regarding the portlets we created. It also contains a comparison between Liferay and SharePoint and some administrative information.

2.4.3 Pros and Cons

Expected gains: Since we created a prototype for a new student portal, the university can get an idea on how to create their final website. Not only did they receive a prototype but also the analysis document. By using this document the university will be able to decide their approach on which portal website they will use.

Expected costs: Depending on what portal website they decide to use and which version, the cost range can vary a lot. This is because the community edition of Liferay is free but it is not suited for the university and they will likely have to take the enterprise edition. SharePoint is the other alternative but this will cost more than when they decide to use the enterprise edition of Liferay.

2.4.4 Risks Analysis

Of course when working on a project like this there are some risk involved. Here we will discuss some of those risks.

Risk: Liferay does not meet the requirements.

Description: It was possible that during the development we discovered that Liferay was not fit as for Karelia like we originally thought.

The probability of the event: There is an always a possibility this can happen, however the main reason this could happen is if we have not done a correct analysis before we started programing.

Insurance rule: In order to prevent this from happening we had to make sure that we did a complete analysis of the current student portal and had a detailed list of the requested requirements. And then we did an analysis of the possibility of the student portal in Liferay.

Risk: Prototype is not finished with all the different modules.

Description: There are multiple reasons this can happen. The first one is because of a bad planning and/or we are overconfident. A second reason could be because we promised too much finished modules.

The probability of the event: The reason this could happen is because we did some bad planning poker for some modules and given it insufficient development time. For that reason it could be possible that the schedule had to change and not all modules will be completed. Because none of us worked with Liferay before, the probability was fairly high.

Insurance rule: To prevent this from happening we needed to make sure we planned as good as possible. We also needed to make sure we checked the remaining time regularly and see if everything could be done on time. When we planned, we also needed to make sure we implemented a safety margin. This margin allowed us to work longer on the portlet than needed or gain time for the other portlets if we do not need this margin time. If necessary we also compensated by working overtime.

Risk: The prototype does not fit the requested style.

Description: There is a possibility that the layout we created does not look like what they expected it to look like.

The probability of the event: There is always a possibility that this can happen. However, the reason why this can happen is mainly due to too little communication with the client.

Insurance rule: Since the most likely reason this can happen is because too little communications, this could easily be prevented by having regular meeting regarding our progress and the lay-out of the website.

2.5 Project Organization

The stakeholders of the project consist mostly out of students and teachers. This is because the project is for students and teachers. The new centralized portal will mostly be used by them. Therefore students and teachers played an active part during the prototyping. Both gave us feedback on the layout and the different modules we create. On the other hand we also had the administration department who helped us during the development, since they have to manage all the class groups and teacher groups. However the most important stakeholder was the Karelia UAS as a whole, because if they are not happy with the project, they will not invest time to develop the final version of the website portal.

2.6 Scheduling

2.6.1 Basic Scheduling

Our developing team consists out 3 people:

- Zsombik Kenny
- Moreels Pieter-Jan
- Decock Andy

We also had a SharePoint server and a Liferay server at our disposal. We worked in iterations with regular meetings.

2.6.2 Project Control

The project control consists of regular meetings. At these meetings we discussed what we would do and what we wanted to accomplish in the upcoming iteration and possible problems and/or setbacks we had encountered in the last iteration.

2.6.3 Schedule

Date	Task	Executor
Week 1: 17 March 2014	Action plan	Entire Team
Week 2: 24 March 2014	Analysing current modules	Entire Team
Week 3: 31 March 2014	Liferay Theme Development	Entire Team
Week 4: 7 April 2014	Server Configuration Liferay	Kenny
	Calendar Portlet	Andy
	Theme Development	Pieter-Jan
Week 5: 14 April 2014	Server Configuration & benchmarking Liferay	Kenny
	Calendar Portlet	Andy
	Theme Development	Kenny
Week 6: 21 April 2014	Assignment Portlet	Kenny
	Calendar Portlet	Andy
	Theme Development	Pieter-Jan
Week 7: 28 April 2014	Mail Portlet	Andy
	Grades Portlet	Pieter-Jan
	Assignment Portlet	Kenny
Week 8: 5 May 2014	Mail Portlet	Andy
	Grades Portlet	Pieter-Jan
	Assignment Portlet	Kenny
Week 9: 12 May 2014	Assignment Portlet	Kenny
	Mail Portlet	Andy

	Grades Portlet	Pieter-Jan
Week 10: 19 May 2014	Configuration SharePoint server	Kenny
	Self-tuition SharePoint development	Andy, Pieter-Jan
Week 11: 26 May 2014	Development Assignment webpart SharePoint	Kenny
	Development Theme	Pieter-Jan
	Development Calendar	Andy
Week 12: 2 June 2014	Finalizing Project	Entire Team
	Thesis	Entire Team
Week 13: 9 June 2014	Thesis	Entire Team
	Administration task	Entire Team

Table 1: Schedule

2.7 Project Closure

The project will be closed at the end of our internship in June. The project will be seen as successful when we created a working prototype and we have a good analysis document.

3 LIFERAY

Liferay is an open source portal platform. This platform uses the GNU Lesser General Public License. This means the framework is free to use, but in addition, it is also possible to buy a commercial license. Liferay is a powerful web platform. This platform offers a multitude of built-in and common used plug-ins. Thanks to its open source nature, there is a good and growing community. A lot of information about default portlets can be found on the Wiki pages of Liferay.

Image 1: Liferay Logo

One of the big advantages of the portal platform is that it has been written in Java, it can be deployed on any Operating system that can run Java. There is also no specified application server database. When you use Liferay you can fully choose what application server you want to use.

3.1 Community Edition vs Enterprise Edition

As mentioned before, Liferay exist in two versions. The first and free version is Community Edition. Contrary to other software vendors, the community is almost a full version of Liferay. It has all the core plug-in bundles, but some plug-ins, that are free in the Enterprise Edition, are not included in the Community Edition. Instead, for these you have to pay separately.

The question that you might ask is: “Why would someone pay, for software that is more or less complete in the free version?”

The difference will be explained by the following table:

	Community Edition	Enterprise edition
Software		
Developer Tooling	Eclipse IDE Plug-in	Liferay Developer Studio
Include Application, Widgets, and portlets	60+ Developer Sample Portlets	100's of include feature for workflow ,web forms, system integration
Exclusive Enterprise Plug-in	No	Yes
Complete Product Documentation	Yes	Yes
Community Innovations	Yes	Yes
License Type	Open Source License	Commercial Licenses
Access to Product Source Code	Yes	Yes
Indemnification	No	Yes
Incident Resolution Support		
Testing Cycles	Feature Quality	Enterprise/Production Quality
Enterprise-level Performance & Security Testing	No	Yes
Product Alerts and Notifications	No	Yes
Security Alerts	No	Yes
Patch Updates	No	Yes
Consolidated Service Packs	No	Yes
Emergency Hot Fixes	No	2 Levels: Gold/Platinum
Access to Customer Portal	No	Yes

Web-based Support	No	Up to 24x7 (One business day response time)
Phone Support	No	Up to 24x7 (As low as 1 hour response time)
Additional Feature		
Performance Monitoring	No	Yes
Analytics	No	Partial
Rules Engine, Integration	No	Yes
Advanced Cluster Communication Channels	No	Yes
Report Engine Integration	No	Yes
Enhanced Cache Replication	No	Yes
Liferay Tcat Server Edition	No	Available for Purchase
Liferay Sync	Singe Site	Multiple Sites
Workflow Designer	No	Yes
Liferay Workflow form	No	Yes

Table 2: Liferay CE vs EE

When comparing the versions of Liferay, you can clearly see that the community edition is optimal for developers. But if a company like the University of Karelia wants to use Liferay it is recommended to take the Enterprise editions. To guarantee the highest availability, a cluster of servers is required. With “Advanced Cluster Communication Channels”, you can assure that the downtime is limited. (Liferay Clustering, 2014) (Sezov, Hinkey, Kostas, Rao, & Hoag, 2013)

3.2 Hardware Requirements & Prototyping

This section of the chapter will first cover the minimum hardware requirements. The second part will explain the different testing rigs we used. There we will go over the impact of the hardware on the software performance.

3.2.1 Minimum Hardware Requirements

The website of Liferay does not state the minimum requirements. The main reason why no information is given on the website is because each configuration depends on the number of users. The configuration of the server depends also on the amount of request that is required to be processed. Therefore, we will discuss the minimal hardware requirements for a server that processes 90-100 requests per second.

The following table shows the minimum requirements for this server, running Liferay portal 6.2

	Minimal	**Good to have**
RAM	16 GB	32 – 64 GB
Processor	4 CPU cores or equivalent	8 CPU cores or equivalent

Table 3: Liferay Portal 6.2 System Requirements

As seen in the table above, Liferay portal 6.2 requires a lot of memory and processor cores to work stable. This configuration has been reported as working really stable and also be able to process at least 90-100 requests a second. (Saxena, 2010)

3.2.2 Testing Rigs

During our prototyping cycle we used 3 different environments. The environments can be found in the following table:

Operating System	App Server	Database	Virtualized
Windows 7 x64 ultimate	Apache Tomcat 7.0.42	Hypersonic	No
Ubuntu 12.04 x64 (Desktop version)	Apache Tomcat 7.0.42	MySQL 5.5	Yes (Virtual box)
Ubuntu 12.04 LTS x86 (server version)	Apache Tomcat 7.0.53	MySQL 5.5	No
Ubuntu 14.04 LTS X86 (server version)[1]	Apache Tomcat 7.0.53	MySQL 5.5	No

Table 4: Testing Rigs

On the first two test rigs we enabled developer's mode. The main reason why we enabled developer's mode on 2 out of the 3[1] systems was so that we could see changes to portlets and themes in real-time. On the last testing rig we tried to simulate an as real possible environment.

Since it is impossible to have homogenous hardware, we tested the different setups with different hardware. The reason why we also tested it on different hardware was to benchmark. Those benchmarks could indicate if a more powerful processor or more memory has a big influence on booting time, response time, etc.

3.2.3 Benchmarking

During the whole development of the portal website we tried to optimise the deployment of Liferay as much as possible. The hardware information of the different testing rigs is the following:

[1] New server release and upgrade to the latest version of Ubuntu Server.

	Windows 7	Ubuntu Virtualized	Server Ubuntu
CPU	Intel i7 3630QM	1 Core of Intel i5 2450M	AMD Athlon 3500+
Memory	8 GB DDR3	1,5 GB	2,0 GB DDR2
Hard disk	750 GB Sata2	15 GB VDI	80 GB Sata2

Table 5: Hardware Configuration

We can clearly see all 3 rigs are not fully optimised for the function of the server. Therefore the test result can be different when running a test on the final machine.

We benchmarked Liferay in multiple stages. While benchmarking, it was required to run Liferay and Intalio BPMS[1].

The first benchmarking was after a clean installation of Liferay and Intalio BPMS. Both were newly installed and only basic configurations as connection ports were edited.

Boot Time Liferay (in ms)	Boot Time Intalio (in ms)	Memory Usage Liferay	Memory Usage Intalio	Remarks
241465	163581	849 Mb	259 Mb	Side by side start-up
151640	80995	849 Mb	259 Mb	Liferay started first, then started Intalio
149276	76042	849 Mb	259 Mb	Intalio started first then Liferay

Table 6: Boot Time Fresh Install

We can clearly see that Intalio has almost no impact on the boot time of Liferay. The memory usage for the three boot sequences stayed the same at the moment of start-up. What was noticeable was the memory usage after 10 to 15 minutes. The associated Java process for Intalio was stable around 260 MB and 300 MB. On the other side we see that Liferay starts at 849 MB and goes up to around 1 GB. Once a user logs in we see that the memory usage keeps getting higher. The highest amount that was measured was 1.2 GB. This amount was used when 2 users were logged in and server was deploying 1 theme. Even the stability of the system was not great.

[1] See section Intalio BPMS for more information

When requesting a page, we could clearly notice a long response time. This was even worse when the server was idle for a long time.

The second stage of benchmarks was after we tweaked Liferay. The tweaking process of Liferay consisted of removing the sample data, re-index all search indexes, clear the direct servlet cache and run the garbage collector. Since the boot sequence does not influence the memory usage, it will not be taken into account in the following table. The boot time after the tweaking can be found in the next table:

Boot Time Liferay (in ms)	Boot Time Intalio (in ms)	Boot order
169010	88560	1. Liferay 2. Intalio
141648	74358	1. Liferay 2. Intalio
170829	89297	1. System reboot 2. Liferay 3. Intalio
169457	88451	1. Intalio 2. Liferay

Table 7: Boot Time After Tweaking

We can see that the tweaking did not improve the boot times. On the other hand we can clearly see that the stability of the server has improved. The memory usage of the server went down with 40-50%. Not only was the server stable, but the HTTP-requests were quicker. Therefore we can conclude that even if the boot time is unchanged, the tweaking was a success.

At the end of the prototype we ran a final benchmark. The results of the benchmark can be found in the following table:

Boot Time Liferay	Boot Time Intalio	Remarks
194097	N/A	Java default settings
259888	N/A	Boot after long time hibernate
81148	N/A	Restart Liferay

Table 8: Boot Time On Local Machine

3.3 Development Components

3.3.1 Hooks

Hooks are pieces of code that override the normal behaviour of the Liferay source code. Hooks are very useful if you want to change your portal website to act in a different way it normally does. For example, if you want your login portlet to check your credentials not by using the Liferay Database, but a specific Active Directory, you can use a Hook to override the current login portlet. Hooks should be deployed with care, however. Using multiple hooks on specific code might cause interference, and harm your setup.

3.3.2 Portlets

Liferay, just as most portal websites, makes use of the portlet system. Portlets are small applications that make up the main page of the portal website. By using portlets, websites can be built in a modular system. It also allow you to personalize the website to the needs of specific users, as content of portlets can be filled in dynamically, by making use of the vast database structure of Liferay.

3.3.3 Template

The template is a blueprint of the default layout of the theme. We use this template so we do not have to create the layout over and over again, for every page in the portal website. Instead, Liferay will use this template to create these pages for us.

Of course, we need to specify several parts in this template, if we want to make something useful of it. For example, in order to navigate to other pages, we need to implement the navigation bar. The code of navigation bar is contained in another file,

and we will not discuss the code here, but in the snippet of code below, you can see how we implemented in the theme template:

```
<div id="sidebar">
    #parse ("$full_templates_path/navigation.vm")
</div>
```

As you probably figured out, the code for the navigation bar is stored in the file navigation.vm. The variable *$full_templates_path* indicates the path where all the files from the template are located. By using this variable, we can be sure that Liferay can find the correct file, regardless of where it reads the theme from. We add this navigation bar to the template by using the code *#parse*.

As mentioned earlier, a template is some kind of blueprint for a theme, creating the parts that are the same for every page. Of course, we do not want every page to look the exactly same. Therefore, we also specify at least one content area on the template. This content area is a place where every page can has its own content. We define this content area with the code below:

```
<div id="contentwrapper">
        <div class="main wrapper clearfix">
                $theme.include($content_include)
        </div>
</div>
```

Just like in the previous snippet, you can see that we declared a specific * * tag. By using this tag, with its unique id, we can define a specific layout for the parts of the template. The main code of this snippet is the *$theme.include($content_include)* part, that adds the content area.

3.4 Setting up Liferay

When first setting up our Liferay environment, we had to set up some specific settings. (Grievous, 2013) Some of these things are necessary, others were just nice extras. The setup is customisable to some extent.

3.4.1 Choosing a Database

When developing a portal website in Liferay, a Hypersonic database is sufficient. (Wikipedia, 2014) Hypersonic is a perfect database for developing, because it is a very small and very fast database, which can be completely loaded into the memory. The advantages of this is the speed in which you can test your code, or the impact of your input data. However, this database is not what you want to go for when you run the portal website on the actual server. The maximum size of a Hypersonic database is far too small (around 1300KB) to be used in this scenario. Instead, Liferay offers a wide variety of databases that you can choose from.

- Berkeley DB
- IBM DB2
- JDBC Compatible
- Microsoft BI
- MySQL
- Oracle
- PostgresSQL
- Firebird
- Informix
- Microsoft SQL Server

It is also possible to connect your own database system, or database systems that are not in this list. To do this, you have to create your own java connector. This will enable Liferay to connect with your database.

3.4.2 Adding Roles

One of the necessary things to add, are roles. These roles are needed to set permissions, and setting user groups. The names of these roles, as well as the setup, are completely up to the administration and tech-department of the Karelia UAS. The roles we added were “Student”, “Teacher” and “Administration”.

These roles have several tasks in our portal website. The dockbar, for example, is only visible for people with the role of Administration. This, of course, is completely

customisable. Other examples are the Announcements and Alerts portlet. In these portlets, announcements and alerts can be created for specific user groups, roles or sites.

Adding roles can be done from the Configuration panel, under the tab “Users”, subcategory “Roles”.

3.4.3 Groups

Groups should be automatically generated for every class. We manually added a few groups, for demonstration purposes. We added a group for a class, to demonstrate how the alerts and announcement system could work, if all students are assigned to their group. Groups can also be used as identifiers in portlets.

Adding groups can be done from the Configuration panel, under the tab “Users”, subcategory “Groups”

3.4.4 E-mail Domain

Setting the e-mail domain is not a necessity, but is very useful for the everyday user. By default, the mail domain is “@Liferay.com”. This has an impact on, for example, the login portlet. The login portlet displays the mail domain automatically by default. Because of this, it is easier for users to log in.

Setting the e-mail domain can be done from the Configuration panel, under the tab “Configuration”

3.4.5 Pages

Our portal website makes use of a specific theme. This theme requires specific pages, in order to work correctly. An example of a page we certainly need to add is the profile page. These pages are required if you want to work with the current

theme. When creating these pages, we selected the “Hide from Navigation Menu” option when creating the page. This is not needed, but makes it a lot nicer in the navigation bar, as we have other buttons to navigate to those pages.

3.4.6 Terms of Use

We most likely want to change the fact that users have to agree to the terms and conditions when they first log in. Instead, students need to sign those when they apply for the school, so there is no point in making them accept again.

These settings can be changed from within the Configuration panel, under the tab “Configuration”, “Portal Settings”, “Users”

3.4.7 Custom Login Hook

In our portal website, we do not want strangers to create accounts on our website. You can disable a lot of options at the control panel, under portal settings, authentication. However, due to a bug, you cannot remove all the unnecessary options. We will explain this in more detail later in the document. However, we will have to create a hook to remove all the other options.

4 INTALIO BPMS

Image 2: Intalio Logo

Intalio BPMS is an open source Business Process Management Suite. This suite helps the user to manage the business processes. While using Intalio BPMS it is also possible to design and deploy complex business processes. Since it is delivered with a powerful and visual designer, a user can create a business process very intuitively and without

any problems. The service makes use of a reliable high performance process execution server; therefore Intalio BPMS does not require a dedicated server, but can easily be added on another server. But always keep in mind that it requires the default access settings. (Intalio, sd)

4.1 Available Versions

Like Liferay, Intalio BPMS exist in 2 versions. The free version Community edition and the paid version: Enterprise Edition. The differences of the 2 versions can be found in the following table:

	Community	**Enterprise**		
		Development	Gold	Platinum
Core Availability	80 % open source	100 % Open source		
Distribution	Binaries	Binaries + Source code		
Support	Online Community	Enterprise Service Level Agreement		
Maintenance	Manual Upgrades	Manual Upgrades	Automated Upgrades	
Process Designer	Yes	Yes	Yes	Yes
Process Server	Yes	Yes	Yes	Yes
Mobile	No	Yes	Yes	Yes
Process Reuse	No	Yes	Yes	Yes
BAM[1]	No	Yes	Yes	Yes
BRE[2]	No	Yes	Yes	Yes
Enterprise Connectors	No	Yes	Yes	Yes

[1] BAM: Business Activity Monitoring
[2] BRE: Business Rules Engine

Clustering	No	Yes	Yes	Yes
High-Availability	No	Yes	Yes	Yes
Certification	No	No	Yes	Yes
Enterprise Portal	No	No	Yes	Yes
Correction of Errors	No	No	Yes	Yes
Phone support	No	No	No	Yes
24x7 Support	No	No	No	Yes

Table 9: Intalio Version Comparison

4.2 Consolidation with Liferay

During the development process we consolidated the Intalio|BPMS service with Liferay. Since both applications use the same application server (Apache HTTP Server), tweaking was required.

To consolidate Liferay and Intalio we tested several options, but only one of these options was successful. The approaches we tried, are the following ones:

1. Liferay and Intalio without tweaking
2. Liferay using default settings while changing connection, shutdown and start-up ports of Intalio
3. Intalio with a bundled Apache tomcat server and Liferay standalone installed on dedicated Apache Tomcat Server
4. Liferay on tweaked dedicated server and Intalio running on a bundled server.

4.2.1 Liferay and Intalio without Tweaking

During the installation of both services, no errors were given. But during the start-up of both application servers, we discovered that only one of the services was running properly. Whenever we started both processes, the log of one of them would show the exception that the socket was already in use. This clearly showed us that at least one process needed to be tweaked.

4.2.2 Liferay on Default Connection Settings, Intalio Tweaked

In another attempt to fix the problem, we kept the default ports and connection settings in Liferay, but changed the settings for Intalio, so it would use a different connection port, AJP 1.3 connector[1] and shutdown port (Apache Tomcat 7, 2014). While booting, everything started like it should. The log-in and manipulating service of Liferay were working perfectly, but whenever we tried the same action on the Intalio service, we found out that it was impossible to log-in. When we watched the log-file of our Intalio service, we could clearly see that the response of the BPMS database was abruptly interrupted. This showed us that the two bundled Apache Tomcat servers were interfering with each other.

4.2.3 Installing Liferay Standalone on separate Application Server and Running Intalio on Bundle Application Server

Since the previous two options, would not work properly, it was time to use the standalone version of Liferay and try to make them work next to each other. After installing Apache Tomcat server on the server, we deployed Liferay on our webserver. When successfully installed we booted up both services. When we started both services we received the same exception as in the first option.

[1] Communication component in Tomcat

4.2.4 Dedicated Apache tweaked, Intalio Default Configuration

In our testing environment we changed the shutdown port to 8010, connection port to 8084 and AJP 1.3 Connector to 8012. The default ports are: Shutdown on 8009, Connection on port 8080 and the AJP 1.3 Connector on 8009. It is imperative to change all three ports, because else one of the Apache tomcat servers would throw an exception and shut down. We added the three newly changed ports to our firewall and tried to start up the services.

When starting up no exception or errors were shown in the log-files. When both applications were online, we tested the connection to both services. Liferay log-in and manipulation service were working flawlessly. After we were sure that everything in Liferay was working, we tried to approach Intalio. The log-in service was working like expected and we could even run a few test queries. This proved to us that both services were working properly without conflicting with each other.

4.3 Intalio Bundle with Liferay

There is a version of Intalio directly bundled with Liferay. But the bundled version is not the newest one. The two versions of the services are: 6.0.4.010 for tempo (Intalio) and 5.2.3 CE of Liferay. The delivered version of Liferay is deprecated and also is no longer available for customers. This means that certain portlets will not work in this version of Liferay. Therefore it is not recommended to use this bundle. To avoid any problem in the near future, it is better to use the latest version of both services. If integration is required, it is better to create a portlet which can use Intalio.

4.4 Accessing Intalio through Liferay

Since the bundle version of Liferay and Intalio are outdated, this means that the available portlet will not work with the current version. Therefore it is required to create a portlet, which will make it possible to access Intalio directly in Liferay. If for

some reason, the choice is not to create a portlet, then it will be possible to access Intalio, but it will not be possible to transfer the business process models to Liferay.

5 DEVELOPMENT

5.1 Theme

The theme is the centrepiece of the Liferay. This is what the user will see when he visits the website. All the content and portlets will be displayed here. We will explain a bit more about portlets later in this chapter.

There are multiple pieces that make up the theme. They all have their separate function, but are all considered part of the same theme. (Liferay, 2011)

5.1.1 Dockbar

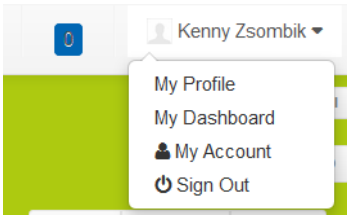


Image 3: User Dockbar

The dockbar is meant as a way for the user to navigate to several parts of the website. For example, the user can navigate to his dashboard, profile and account from here. He also gets the option to log out. In our case, we do not want normal users to be able to change things like their profile, since that's information only the administrators from the school are allowed to change. In our Liferay project, we hid the dockbar for normal users, and manually implemented the functions that students are allowed to access in the page.

The dockbar is also used to modify the page layout and the website settings. These features are only accessible by administrators. In our theme, we enabled the dockbar for administrators, so they can modify the pages as they see fit, and manage the website from their own account.

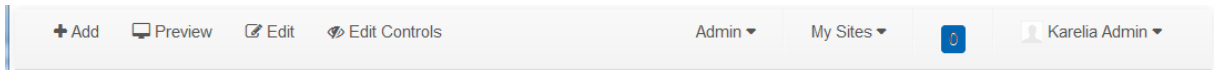


Image 4: Administrator Dockbar

5.1.2 Navigation Bar

The navigation bar, also called the navbar, is used to navigate through the several pages the website consists of. The difference with the dockbar is that navbar lets you navigate through pages of the same type, while the dockbar lets you navigate through the different types. These types are:

Public pages

These are pages that every user can see. They are accessible for every user by navigating there using the navigation bar, or by using the URL.

Private pages

Private pages are pages that are only visible to the creator. Other users will not see this page in their navigation bar, and they cannot access it by using the URL either.

Dashboard

The dashboard is a private page that every user can manage for himself. This has both its advantages and its disadvantages, in our scenario. The advantages are that the student can add the portlets he likes, and removes those he does not need. He also has the possibility to move them around as he sees fit. The disadvantage, however, is that with this many options, the user can ruin his own dashboard, and it is a bit difficult for most users to set it up again. There are also a lot of options they do not need, or should not be allowed to access.

Profile

The profile is a public page that is, again, completely customizable by the user. This is the page is accessible by anyone, so users can use this as their own place to write some things and place portlets as they see fit. This option is not needed for our student website, so we decided not to use it.

5.2 Portlets

In this section we will explain the portlets we developed as part of the prototype more in-depth. Screenshots of these portlets can be found in Appendix III.

5.2.1 Mail Portlet

The mail portlet is a portlet that proves that, with a single login, it is possible to access several different modules. With this portlet, users can read and send e-mails from their profile page. For this portlet, we took an existing portlet, from the Liferay website, and made our own adaptations. The major adaptation was to limit the users in adding e-mail accounts. Since this stays a school environment, we do not want users to add just any e-mail account. Therefore, we made the portlet in a way that the user information gets pulled from the Liferay database. Both the user's name and email address get taken from the database, and information like the incoming and outgoing connection can be set by the system administrator. This way, they will be the same for all students. The only thing the portlet cannot get from the database is the user's password. The reason for this is because Liferay runs a hashing algorithm[1] on the password, before saving it in the database. That's why, if a user wants to check his e-mail, he has to add his e-mail account to the portlet. You get the option to save your password, so that you do not have to do this every time again, but the choice is entirely up to the user.

The mail portlet was developed using Liferay JSP. During the development of this portlet, we encountered several difficulties. One of those difficulties was redirecting a user to another page within the same portlet. The problem is that you cannot just redirect to a specific URL, since there are different portlets on one page. To achieve this, we had to use a specific technique of retrieving the portlet-specific URL, and building upon this. The code snippet below will help with illustrating this.

```
LiferayPortletURL renderURL = PortletURLFactoryUtil.create(request,
    portletDisplay.getId(), plid, PortletRequest.RENDER_PHASE);
renderURL.setWindowState(WindowState.NORMAL); //to set windowState
renderURL.setPortletMode(PortletMode.VIEW); //to set portletmode
renderURL.setParameter("jspPage", "/accountAdd.jsp"); //the page within the
    portlet
```

[1] An encryption algorithm which is considered practically impossible to invert

In the code above you can see three remarkable properties of the *renderURL* object. The first one is the *setWindowState()* method. It is possible to make portlets go in Full Screen mode or make it hover above the page, like a pop-up. By setting this variable to *WindowState.NORMAL*, we make this portlet look like a normal portlet; in other words, make it only use the size it needs.

Second important method we can see in the code snippet is the *setPortletMode()*. This method is used, as the name suggests, to display a specific mode of a portlet. Most portlets by default have a view-, edit- and help-mode. You can choose what mode should be displayed by setting the *PortletMode*.

The last, and for us probably the most important method is the *setParameter()* method. This method is used to give a specific parameter a value of choice. In our case, we need to set the *jspPage* parameter. This parameter adds a specific page to our *renderUrl*. It is this parameter that enables us to navigate to a specific page within our portlet.

Since Liferay works with portlets, and you have the possibility to add multiple portlets on one page, Liferay cannot know, while building a portlet, whether a certain name is already used in another portlet. This can prove a big issue, because this means you could have, for example, several <form> tags. The problem is that the data from these forms can, if this occurs, be faulty, because of input fields in other portlets. For this reason, Liferay makes use of the <portlet:namespace/> tag. This tag will add the name of a specific portlet to a name, ensuring that only one object can have a certain name.

5.2.2 Grades Portlet

Another portlet we developed was the grades portlet. This was just a prototype with set variables. The idea was that we did not want to lose too much time, as progression in development of the portal website was slow at times.

The general idea behind the portlet is that students see only their grades. When on the profile page, they are able to see all their grades, with the most recent grades on top. With this portlet, you can also view your grades from the course page. There, only the grades from the course will be visible.

5.2.3 Student Assignments Portlet: Introduction

This part will handle over the portlet for student assignments. This part is divide in three. The first part will illustrate the database structure we used. This will be followed by the explanation of the Service builder, and how it translated the service.xml into usable model-, transaction- and persistence classes. And finally the GUI components and the used methods will be explained.

The final and bug free version of the “Student Assignment” portlet took around 2.5 weeks to develop. The time planned for the portlet was normally 1.5 week. The reason why there was more time spent on this portlet then expected can be explained by the multiple faulty approaches.

5.2.4 Student Assignments Portlet: Database

To build upon the Liferay database, we must use the service.xml file. In this file we declare all entities and the different columns. For our prototype there are 4 entities. Those entities will result in a table in our database. The tables we declared are the following ones:

1. Task
2. ClassGroup
3. Student
4. DoneTask

Each entity has its own attributes and finders. Finders are declared in the service.xml. These methods return an object or a collection of objects. If declared it will result in the creation of methods *FindByXYZ()* in the Local or Remote persistence class, when the service builder has completed his creation job.

The attributes and finders for each tables are the following:

Task:

Attribute name	Type	Explanation
Uuid[1]	Long	Universal unique identifier which will make the task unique.
Id	Long	When stored, each the task will receive an ID. Its purpose is to speed up update and deletion queries.
UserId	Long	Identifier of the creator of the task.
Username	String	Email of the user who created the task. Purpose: finder
classgroupId	Long	Class group ID of the class for whom the task is intended to.
Duedate	Date/time	The last possible day and time a student can turn in the assignment
Description	String	Description of the task.

Table 10: Database Table Task

Finders for table Task:

Finder name	Return type	Explanation
Username	Collection	Return a collection of Task-objects that were created by a certain user
ClassGroupId	Collection	Return a collection of Task-objects for a certain classgroup

Table 11: Finders Task

classGroup:

Attribute name	Type	Explanation
Id	Long	Identifier of a class group
Name	String	Name of the class group
teacherId	Long	The Id of the teacher who is responsible for the class group

Table 12: Database Table classGroup

[1] (Rouse, 2014)

Finders for classGroup:

Finder name	Return type	Explanation
Name	ClassGroup	Return a ClassGroup-object for a given name
Teacher	Collection	Return all ClassGroup-object where Teacher with ID is assigned to

Table 13: Finders classGroup

student:

Attribute name	Type	Explanation
Id	Long	Identifier of the student
Name	String	Name of the student. Purpose: finder
Firstname	String	First name of the student. Its purpose is to quick find students where the first name is equal to a given name.
uuidFromTable	String	Uuid from the table Users_, this will identified a student and will make the record unique in combination with the classgroupId
classgroupId	long	A student can be in multiple class groups, therefore the combination of UUIDFromTable and classgroupID, will give all classgroups for a student.

Table 14: Database Table student

Finders for table student:

Finder name	Return type	Explanation
Name	Collection	Return a collection of Student-objects with a given name
FirstName	Collection	Return a collection of Student-objects with a given first name
UuidFromTable	Collection	Return a collection of Student-objects with a given Uuid.
ClassgroupId	Collection	Return a collection of Student-objects based on a ClassgroupId

Table 15: Finders student

doneTask:

Attribute	Type	Explanation
Id	Long	Unique ID of a task
studentUUID	String	Student that turned in a task
taskUUID	String	Which task the student turned in

Table 16: Database Table doneTask

Finders for doneTask:

Finder name	Return type	Explanation
StudentUUID	Collection	Return a collection of DoneTask-objects for a given StudentUUID
TaskUUID	Collection	Return a collection of DoneTask-objects for a given TaskUUID.

Table 17: Finders For doneTask

Beware

When developing the database tables for the portlet, we saw that the usage of foreign keys was non-existent. The reason why Liferay does not implement foreign keys is the performance loss that occurs when a modify- or a delete query gets triggered. The moment the call is made to change or delete a record in the database, the software must first check the constraints. For example: you are deleting a class group in the table Class group. The software must first check if the class group is empty. If not you have to modify all students who are part of the class group. Once all the modification are made, it will be possible to delete the record.

For our portlet it would take around a few milliseconds. But in the real world it would be a lot slower. This difference can be explained by the number of records that need to be changed. In our prototype we only have a maximum of 100 records. For organisations like Karelia University of Applied Science (UAS) the amount of records can easily exceed 4000 records.

5.2.5 Student Assignments Portlet: Service Builder

Once the entities are declared in the service.xml, we started the service builder. Since we are using Eclipse as developing tool, we have to specify the build type when creating the project. For developing our portlets we chose the Ant (Liferay-plug-ins-sdk) build type.

With the help of the Ant builder we can easily create a service for our portlet. The reason why you should always use the service builder is simple: the Service Builder is a tool built by Liferay to automate the creation of interfaces and classes that are used by a given portal or portlet. This includes code for EJBs[1], Spring, Persistence and Model. The second reason why you should use the Service Builder is the following: once the Service builder is used, the portal-service.jar can be accessed by WAR's that are deployed outside the class path of the main code base.

The classes that are generated by the service builder should never be modified, with exception of the classes that ends with Impl. The abbreviation Impl, that the Service Builder uses, comes from the word implementation. If these classes are edited, only run the service builder and it will be propagated through the whole portlet.

Once the building processes is finished, the only thing that has to be done is refreshing the project. The refresh is needed, because if this is forgotten, no classes will appear in the project. Therefore every time the "build service" is called, you have to refresh.

5.2.6 Student Assignments Portlet: GUI Components and Methods

Since this portlet is role based, we decided to create 2 views based on the view. A view for the teacher where he can add assignments and a view for the student. The student view will show a calendar with a table underneath it. This table will show all assignments the student has to do. This section will be divided in 2 parts. The first part will handle the teacher's graphical user interface and the second one will handle the student's graphical user interface.

[1] EJB: Enterprise JavaBeans

- Teacher Graphical User interface

The component exist of tab views. The first tab view is to create a new assignment, the second one is to assign teachers to their class groups and the last one is to assign students to the different classes. Due the absence of an active directory we implemented two extra features. In the version the teacher normally should have, only the “new assignment” should be available, since the class group assignment and student assignment will be delegated to the active directory.

The first component that is added to the tab-view, is the assignment module. This module makes it possible to post assignments for a specific class group. When you see the layout, you can clearly see there is not much the user can do wrong to make the portlet crash. But during the development we struggled multiple times with exceptions. Those exception were mainly due bad coding. For example during the data retrieving from the database, it happened multiple times that we got an exception that a certain GUI[1] component could not be drawn. The reason why the component could not be draw was because the list that was retrieved from the database was of the object type *List*. To assign a data source to a dropdown list, the input data must be converted to a Vaadin container[2]. Once the conversion is done you can simply attach the container to the data source.

Since we added the table *classGroup* to our database, we could easily implement the function to retrieve a list of class groups where the logged in users is responsible for. To retrieve the logged in user we implemented the *PortletRequestListener* interface. The result was that we could override 2 methods. In order to retrieve the user that logged in, we only need to implement the following method: *onRequestStart()*. Once we have the user we can easily request his id. And with the finder that was created, while generating the database, we can request a list of all the class groups, from which he is part of. It is also possible that a teacher logs in and sees the message “No classgroups available”. This means the teacher has no class groups assigned to him. When this message is shown, there is no possibility for the teacher to create assignments. The reason why this was implemented is to protect the integrity of the database. Another security measure we added to the portlet is that every input field is required. With the help of Vaadin you can easily make an input field required. As long the user has not entered any text, an asterisk is shown. This means that the text-field

[1] GUI: graphical user interface
[2] Vaadin container: is the highest containment level of a Vaadin data model, for containing items.

or other input component was empty. For that reason the buttons in the portlet will not work. As soon as all required fields are filled in, it will be possible to use the buttons like store, edit or delete.

The method that is used to retrieve all the class groups of a teacher is: *findByTeacher(Long teacherID)*. The method is not directly accessible in the *ClassgroupUtil*. Therefore it is needed to reference it in the *ClassGroupLocalServiceImpl*. Once referenced, the only thing that is needed in order to use the *findByTeacher(Long teacherID)* method, is to run the service builder and refresh our build in Eclipse.

Since Vaadin is event-based we can easily implement some code that the application has to run whenever the button is pressed. The code that is hidden from the user is quite complex. Since the code is written in Java, it took us around 3 man days to make it work. But if we would have done the same with Liferay JSP, it would take us far more time. What our code does is first check if the date is in the right format. Once it is checked it will retrieve the data that is written in the input fields. Once this is done, the portlet will create a confirmation window. This window will ask the teacher if he's sure to post it, and also if the information he has entered is correct. Once he accept the confirmation dialog, the system will check if the assignment is unique. This means check if the assessment was not already created by the same person with the same description. If the test does not report any errors, the assignment will be available for the students of the class group. During the development we deliberately chose to allow teachers to post assignment before the day of today. In some cases it is possible that a teacher post an assignment that is already due. In our school it sometimes happened that a teacher posted an assignment. Because he is the teacher, he needs to give grades for the test. Let's explain this with an example: a teacher is giving a test on a sheet of paper. The test takes place on the 5th of June. By the time all the sheets of paper are corrected it is the 10th of June. To avoid misunderstandings, the teacher has to create an assignment but with due date the 5th of June. If the teacher would not be able to post an assignment on the 5th, then we would not be able to guarantee the integrity of the stored information.

In our portlet there are 2 flaws. One of the shortcomings is the following one: the component does not get repaint. If you want to repaint the whole component you

have to refresh the page. This could lead to confusion for the user of the portlet, because he could think that the software bugged and that the assignment is not posted. The second shortcoming of Vaadin is that whenever a notification is send, the browser will remove it as soon the user does an action. This action can be a key press or even if the mouse is moved. This could also lead to confusion. Most people like to read the notifications a website displays. But if in a fraction of a second you see a big block appearing and disappearing, the user could think the assignment is not posted. In the normal flow, the user will retry to store the assignment. But since we implemented a verification, the user will receive a message telling him that the assignment was already posted. This can lead into a loss in usability.

The second component is the assignment of the teacher to a class group. This component is not mandatory. But since there was no active directory available, we had to implement this component to assign teachers to class groups. In our prototype a teacher is assigned to a class group. By doing this, the responsible teacher is the only one who can post assignments. This ensure that teachers cannot post deviating assignments.

In this component there is one list that needs to be filled. After it is filled, we need to link it as a data source to the Teacher dropdown list. The program code used to get a list of teachers is the following one:

```
private Container getAllTeachersSelect()
{

long companyId = CompanyThreadLocal.getCompanyId();
List <User> allTeachers = new ArrayList<User>();
BeanItemContainer<User> allTeachersSelect = null;

try
{
        Role role = RoleLocalServiceUtil.getRole(companyId, "Teacher");
        allTeachers =UserLocalServiceUtil.getRoleUsers(role.getRoleId());
} catch (PortalException | SystemException e)
{
        window.showNotification("Retrieving Teachers error");
}finally
{
```

```
        allTeachersSelect = new BeanItemContainer<User>(User.class,allTeachers);
}
        return allTeachersSelect;
}
```

Since Liferay uses *companyID*, to store roles we have to ask Liferay the company id of the site where the portlet is located. Once we have the company id we can ask the *RoleLocalServiceUtil* to get the Role with *companyID* and name "Teacher". Once we have the role, we request to get all the users with this certain role. Once this is done we can return the container. The container will then be set as a data source of the dropdown list. To make it more readable for the user we need to set an *itemCaptionPropertyId*. The best readable and most comprehensive caption is the one where only the full name is displayed.

The second graphical input component that is available on our main component is a text field. The user has to input a class name. For example the name of a certain class is “Math - First year”. If by mistake the user should forget to type in a name or select the responsible teacher, the software will give an appropriate warning. Once the user has successfully selected a teacher and a class group, he will be able to send it to the server. On the server it will first convert the selected teacher into a User-object. In second stage of the storing procedure it will get the class group name. Once information is processed, the application will create a class group. During the development we came across an issue. When we were trying to store the information in the database, we received a message that the *primaryKey* was duplicated. To solve the issue, we had to set the *PrimaryKey* first. The setting of the *primaryKey* could be done by making a call on the method *increment* that is available in the *CounterLocalServiceUtil*-class. Once the information is stored in the database the user receives a notification telling that the teacher is assigned to the class.

The last and also not mandatory component we created for the prototype, is the possibility to add a student to a class. This component is not needed when you are using an active directory. What the component does is simple. First it will fill up the two dropdown lists. The first one is the students list. Like in the previous component, we will first use a standard method of Liferay. The code to retrieve a list of students is almost the same as in *getAllTeachersSelect()*, except that the role-name is “Student”.

Once the container is created, we only need to set the newly created data source to the dropdown list. The second list that needs to be filled, is the class group list. Since the class group table is not a default table, we cannot use default call methods of Liferay. We have to use the methods that are available in the Local class group utility service. While looking at the available methods in the class we could not find a method that returns a list of all class groups. Since a note clearly clarifies that modifying the *ClassGroupLocalServiceUtil* is discouraged, we had to check if the method is available in other java-classes. While looking in the persistence class of class group, we discovered there was a method *findAll()*. This method return a list of all class groups. To use this method we have to declare the method in the Local service. Once it is declared we have to re-run the service builder. At this point the service builder will propagate the method through the whole service. This means that we can do a call on the *findAll()*-method that is now available in the *ClassGroupLocalServiceUtil*. The reason why we cannot call the method that is available in the persistence class directly, is because the utilities classes guarantees that the Spring session has been initialized.

To insure that the software would not crash, we implemented multiple security measures. One of them is to disable the store button in the portlet if one of the lists is empty. The portlet first checks if the list is empty. If the list is empty it will no longer show a dropdown list but a label telling which list empty is.

- Student Graphical user interface

In contrary to the teacher graphical user interface, this interface has no input fields. The only thing we can see as a student is the following: a little calendar and underneath it a table with a list of all the assignments to be turned in. The table is ordered by due date. Since the table is only shown whenever assignments are available, we added a changeable view. When there is no assessment, we show a label that shows the following information “No assignments available”. We had to implement the changeable interface because if there were no assignments in the database for our student the component would throw a null pointer exception. And the webpage would return the message “Portlet is temporally unavailable”. Therefore to avoid this crash we implemented the dynamic components.

Retrieving and displaying the list with only useable information was not that easy. During the prototyping we saw that we had to take multiple exceptions into account. The exceptions we took into account for creating the table are the following ones:

1. Show assignments where the due date is already passed
2. Show assignments that are meant for the student
3. Do not show assignments that are already submitted

To get the list of assignments for a student, we need to get the list of all the class groups where he's in. Once we have this list we will check for every class group if there are any assignments. After we filled a list with all the assignments of all his class groups, we start extracting the assignments into a new list where only the assignments are kept that are not yet submitted or where the due day is passed. Once the remaining assignments fulfil all requirements, we transform the list to a user friendly representation. To do so, we created a new model where only the following information is stored: Due date, description and the class group. Thanks to the newly created model, only the important information is available for the table. Since we filtered all the information before sending it to the client we can say it is more secure to operate like this. If we want to avoid to filter the information, there is the possibility to hide columns. The security threat that occurs when we work like this, is that it is possible to unhide the hidden columns by changing a bit of code in the browser. When the newly created container was set as data source of our table, we found out that the date was not showed properly. To make the representation of the date user-friendly, we had to specify the format property value of the table cells. The format property value for the table is the following one:

```
tabel_Upcommings = new Table("Upcomming events :"){
/**
 * format Date cells into Day/Month/Year
*/
private static final long serialVersionUID = 304711426464704404L;


@Override
protected String formatPropertyValue(Object rowId, Object colId,Property
property) {
	Object v = property.getValue();
	if (v instanceof Date) {
		SimpleDateFormat df = new SimpleDateFormat("dd/MM/yyyy");
		return df.format(v);
	}
	return super.formatPropertyValue(rowId, colId, property);
}};
```

When analysing the code above, you can see that first of all we are overriding *FormatPropertyValue* of a table. Whenever the table is created, it will check if the instance of the object in specified row and column equals an instance of *Date*. If so it will create a *SimpleDateFormat* and will return it. This improves the readability of the table.

5.3 Vaadin

Vaadin is a Java based framework. This framework is used for building modern web applications. These applications are responsive and easily manageable.

5.3.1 Vaadin Version & Change Log

For the development of those portlets we used version 7.1.15 of Vaadin. We deliberated chose this version of Vaadin, because it was the most up to date. Some major security issues of Vaadin 7.1.11 were fixed. The most important security fixes that were resolved with Vaadin version 7.1.15 are the following:

Escaping of OptionGroup item icon URLs

The issue affects *OptionGroup* with item icons. Proper escaping of the src-attribute on the client side was not ensured when using icons for *OptionGroup* items. This could potentially, in certain situations, allow a malicious user to inject content, such as javascript, in order to perform a cross-site scripting (XSS) attack.

In order for an application to be vulnerable, user provided input must be used to form a URL used to display an icon for an *OptionGroup* item, when showing that *Option Group* to other users.
The vulnerability has been classified as moderate, due to its limited application.

Escaping of URLs in Util.getAbsoluteUrl()

The client side *Util.getAbsoluteUrl()* did not ensure proper escaping of the given URL. This could potentially, in certain situations, allow a malicious user to inject content, such as javascript, in order to perform a cross-site scripting (XSS) attack.

The method is used internally by the framework in such a manner that it is unlikely this attack vector can be utilized in practice. However, third party components, or future use of the method, could make an attack viable.
The vulnerability has been classified as moderate, due to its limited application.

Closing modal window

Other changes that where important for our project was the closing system of a modal window. In the previous version, there was a bug where modal windows sometimes did not closing properly. When the *onClose* trigger was send, the window was unregistered but it was still visible. This lead to requiring a page refresh.

5.3.2 Advantages

- Native support for a variety of browsers

In today's society we use a variety of browsers, not all browsers are equal. The most used browser today is Google Chrome, followed by Mozilla Firefox and Internet explorer. Vaadin 7 has native support for all most recent desktop browsers. The list of supported browsers can be found in the following table:

Browser	Version
Google Chrome	23 or newer
Internet Explorer	8 or newer
Mozilla Firefox	17 or newer
Opera	12 or newer
Safari	6 or newer

Table 18: Browser Support Vaadin

Vaadin does not only support desktop browsers but browsers for mobile devices are also supported. These days, practically everybody has a smartphone. A lot of students use their smartphone on school to check their grades or class schedules.

Therefore it is a great choice to use Vaadin to build portlets, because its supports Google, Apple and Microsoft based mobile browsers. The list of supported mobile browsers can be found in the following table:

Browser	Version
Android	2.3 or newer
IOS	5,6,7 or newer

Table 19: Mobile Browser Support Vaadin

- Java based programming language

Because Vaadin make use of the Google web toolkit for rendering the resulting webpages, this helps the developer to create and maintain complex JavaScript front-end applications, in contrast to the javascript based Language that Liferay uses as default programming language. This means, for developers who are used to program in Java - like us -, that they can use a framework that is based on a language they have mastered.

Vaadin is a framework which incorporates event-driven programming and makes use of widgets. This means that it is a programming model that is closer to GUI software development. This means that pages can be more reactive, and that the users can experience the website more interactively.

- Tooltips & required field

In Vaadin, it is possible to give Graphical components tooltips. Those tooltips will be displayed whenever a user hover over a component. Those tooltips can easily give a user extra information about a button or a text field. For example, instead of using a button with a specific text, we use an image to mimic the action of the button. Not every user will be able to understand what the button does. Therefore it is recommended to add a tooltip, which will explain the action of the button.

Defensive programming is the key to protect portlets from crashes. Vaadin implements a required field method. The required field method can be added to any user input related component. For example: A portlet consist of a three textboxes and a store button. A user has to input some basic information as a name, surname and social security number. The portlet will not initiate the Spring session as long the 3 required fields are filled in and the button is clicked. This will guarantee that empty values will not be transferred to the server, which could lead in to an application crash.

- Event driven language

Nowadays, websites needs to be responsive. This responsiveness will increase the user experience. Therefore, a responsive web design is needed. Implementing RWD[1] directly in Liferay is possible. But to do so the developer has to be an expert in JavaScript. Since Vaadin is based on Java, it is far easier to create a responsive web. Whenever a user clicks a button, text field or table element, there is the possibility to send an event. This event can be used to, for example, automatically fill in labels.

5.3.3 Disadvantages

- **Redeploying portlets is needed after updating Vaadin version**

Whenever there is a new version of Vaadin and it is installed, the widget-set needs to be recompiled. As a result, all portlets that uses Vaadin need to be redeployed. This can be time consuming for a portal that uses a lot of portlets.

When we started developing with Liferay, the version was 7.1.15. At the end of May, Vaadin released a new version, 7.2.0. This means that whenever we upgraded Vaadin, we needed to redeploy every application. The updating of Vaadin and redeploying of Vaadin portlets roadmap can be found in the section “Roadmap to a new Vaadin version”.

5.3.4 Possible Extensions

In this section we will explain a few possible extensions for the Assignment portlet. These extensions are not required but for students and teachers it is a nice to have.

- Integration of Upload plug-in

Since there is no implementation of an upload functionality to the “Student Assignment” portlet. There is the possibility to extend with one of the upload portlets of Vaadin. The reasoning behind this thought is simple. A teacher can post an

[1] Responsive Web Design

assignment where a student of a class group has to turn in a task. The student can simply turn it in by uploading his task. After the due date of the task has passed, the teacher can download all the turned in files and retrieve a list of all student who did not turn it in. This extension would mean that the teacher no longer has to check the email for possible submissions.

- Grading plug-in

Instead of using a remote system to grade student, there is the possibility to directly integrate it in Liferay. For the grading system there is the possibility to create a Vaadin plug-in. The only thing that is needed is to expand the database. The grading plug-in would first check if the teacher has made a task. If there is a task available, and need to be graded, the system will request of all the students who are in that class and need grading. Once the grading is done and set available, the student will be able to see his grade for a certain task

5.3.5 Development Issues

- **Development from the point of view of the user**

Since the database implantation and service can directly be built by the “Service Builder” of Liferay the first version was developed from the point of view of the user. We made a responsive user interface. This interface was developed in a way that user had as much freedom as possible. The user could redesign the layout of the portlet without any problem and without compromising key elements of the software. When the multiple views of the user interfaces where developed, we started implementing List viewers and domain classes

- Domain class explanation

For this portlet we had three domain classes:

1. ClassGroup
2. Task
3. User

The attributes of each of those classes can be found in the following table. Each of these attributes are also explained.

classGroup:

Attribute	Type	Explanation
Id	Long	Unique identifier for every class group object, that will be stored in the database
name	String	Name of the class group
teacherID	Long	Identifier of the Teacher that is responsible for class

Table 20: classGroup Rejected Development

task:

Attribute	Type	Explanation
id	Long	Unique Identifier for every Task object, that will be stored in the database
userId	Long	User id of the task creator
username	String	Name of the task creator. This is used to implements finders
classgroupID	Long	Identifier of the class group.
dueDate	Date	Until when will be the task be seen
description	String	Description of the task

Table 21: task Rejected Development

student:

Attribute	Type	Explanation
Id	Long	Unique identifier of a student, based on the uuid of Liferay

firstName	String	First name of the student, used for finder method
lastName	String	Last name of the student, used for finder method
Class GroupId	Long	Class group identifier. A student is assigned to a class

Table 22: student Rejected Development

- Domain class remarks

Since those portlets are prototypes, we tried to develop easy to understand classes. In reality will those classes more elaborated.

While using Liferay we saw that we could create roles and groups. But for the portlet those assignments were not sufficient. Therefore we chose to create a few users. And declare 2 Liferay groups. Student en Teachers. The permission of usage of the application is based on this group. If a user is not in one of those two groups, the application will give an appropriated error message.

Whenever a user is created in Liferay, he will receive an UUID (universal unique identifier). This is a 128-bit unique identifier that will identify an object or entity on the Internet. For that reason we decide to use this identifier as an ID for declaring a Student object.

- List view classes

The main reason why those classes were created, was because they would represent all the domain classes in a user friendly way. Those list views were responsible for sorting the various list we had implemented in an ascending way. The other responsibility for those classes was sorting the information that was useful for a user. An example of a useless information is the *graceLoginCount* from the table *Users_*. The information that is stored in the cell of *graceLoginCount*, is just a number. This number is important for the software since it is a password policy rule. But for the user it is insignificant. Therefore, we use the list view to remove such information from the view of the user

- Data transaction classes

The core classes of the portlet are the data transaction classes. Those classes make it possible to retrieve, store, update and remove information from the database. In this version we tried to access the database without using the build-in “Service

builder". This led to multiple connection-loss while saving information to the database.

The reason why the connections were dropped can be explained by the missing Spring session. Therefore, the portlet could no longer guarantee the security and integrity of data in the database.

5.3.6 Roadmap to Updating and Redeploy Vaadin

To update Vaadin to a newer version Vaadin, a site administrator has to log in. On the administration site he need to have the following plug-in installed: "Liferay Control Panel Plug-in for Vaadin". This plug-in is not needed to update the version, but it will make the process less time consuming.

Before updating Vaadin, the portlet will give a list of all portlets that need to be redeployed. This list will be used in the second part of redeploying the portlets.

The updating process takes some times, it all depends on the speed of the server. By the speed we mean, the processor clock speed, ram speed and hard drive writing speed. In our case we used a Laptop with 4GB ram and a Core I7 M430. The process of updating Vaadin took around 5 minutes. Once this is done recompiling the widget-set is needed.

In contrary to the updating process, this process takes more time. On our hardware it takes around 30 – 45 minutes. The reason why the widget-set is needed is because it is a GWT module that ties together client side component implementations used by a particular Vaadin application. It is the part of the client side terminal that creates components from UIDL messages generated by the servicer side.

For our Vaadin portlet we used the default widget-set. Therefore it is needed to recompile it after a new version of Vaadin is installed on the server.

Once the recompiling of the widget-set is completed, redeploying the Vaadin based portlets is needed. To do so the only thing that is needed, is to remove the portlet from the Application manager in Liferay and redeploy the WAR[1] file.

[1] WAR file or Web application Archive: is a JAR file used to distribute a collection of JavaServer pages.

6 INTEGRATION OF EXTERNAL DATA SOURCES

It is important for the University of Karelia to be able to use external services. An example of the external services that will be used is the SoleTM from Solenovo. The application makes it possible to generate time tables without problems. To avoid data integrity lost, Solenovo implemented a protection. If you want to access the time table and you are not allowed to edit it, you have 2 possible ways. The first one is to log in directly on the website where the timetables are hosted. Since we want to remove the log in part, we have to reject this solution. The second one is requesting the timetable with an URL. In URL that is retrieved, all the information of the schedule is available. To add the time table directly to Liferay we have two other possible options:

- Create a portlet with Liferay JSP
- Create a portlet with Vaadin

During the development, we agreed that we preferred to develop portlets with Vaadin. The reason why we advise to use Vaadin, is because it is not Liferay version dependant. For example if a new version of Liferay comes out, the only thing that needs to be done is to check if Vaadin is installed. If it is installed, you can just redeploy all the Vaadin based portlets without having to recompile them or rewrite them.

Since the portlet will not use the database of Liferay, there is no need to run the ANT Service builder. Instead, we can build a custom service, that request the timetable for the log in user. Once the custom service received the URL with all the information in it. The portlet will display it in a calendar based view. To improve the usability of the portlet, we advise to implement a popup method. Whenever a student clicks on an event in his calendar, he receive more information about the event.

The reason why we are sure that it is possible, is because there is a company called Nuxeo, which created a database which is accessible with the same principle as Solenovo. They created a Vaadin portlet who request and process the information, and shows it in a Liferay Solution.

7 CAPEX AND OPEX

7.1 CAPEX

CAPEX, or Capital Expenditures, is a term used to indicate the expenses of fixed assets that last longer than a year, in order to increase the future benefits. They are costs like machines, that only have to be bought once, or in our case, the development of the portal website, something that also only happens once. CAPEX includes both the expenses of new assets as expenses to upgrade or build upon current assets.

7.1.1 Hardware

Like we described earlier in this document, the hardware requirements are pretty high, if you want a high performance server. This server can be either on premise, meaning a server will have to be bought, and run in the university itself. This means that there will be a pretty expensive start-up cost. Another result of this is that the university will have to pay the electric bill for the server, as well as the extra internet traffic.

If the university, however, decides to run the server at another location, for example at another company, there will be a regular payment for this service.

7.1.2 Operating System

Depending on whether you run the server locally or in a cloud-based solution, you might have to invest in buying the operating system for the portal website software to run on.

Liferay

When it comes to Liferay, the choice of your operating system is almost endless, as long as your operating system supports Java. This means that there is also the option to use Linux operating systems. Most Linux distributions are free, however, some of the high security enterprise distributions require some kind of license. There is also the choice between Command Line Interface server and Graphical User Interface. This greatly affects the ease of use, and will influence the skill required to maintain the system.

SharePoint

In order to run SharePoint, a Microsoft server is required. The advantages of having a Microsoft operating system is the fact that there is a lot of documentation available on the internet, and that you have a central point of contact, in case something goes wrong. De disadvantage, of course, is the price tag that comes along with this. However, in case of SharePoint, you do not have a lot of other options.

7.1.3 Development

Liferay

Because the development of a Liferay portal website is a pretty tough job, the development will probably have to be outsourced to another company. This is the case for both the development of the theme and the individual portlets, as all of them will have to work together. It is possible to create the portlets yourself, but it is not an advisable thing to do, because if you make mistakes, and deploy it on the server, the whole server, including the data in your database are at risk of being corrupted. The problem with development of Liferay portals and portlets is the fact that, even though the code is very similar to normal Java and JSP, there are a lot of custom rules and tags that need to be kept in mind while developing. If you're not used to them, it is very easy to lose a lot of time during the development.

If the university later decides that the current portlets are not sufficient, new portlets can be developed and added. This should also be taken into consideration, as this might be a big money drain.

SharePoint

The development of a SharePoint portal website is done using ASP.NET and VB.NET. These programming languages are equally well known to Java and JSP, and are relatively easy to program in. Of course, because you're still programming for a portal website, there are some little tweaks in the code here and there as well, but the amount of these exceptions is a lot smaller then in Liferay, and the consequences these mistakes can have are far less. This makes development for SharePoint a lot easier then development for Liferay. This means that development can be done by people at school, and even by students. However, the deployment of these portlets is a lot harder, and requires knowledge of the PowerShell scripting language. As a result, in order to set up the server and deploy the portlets, an experienced user is required.

7.2 OPEX

OPEX, or operational expenses, are recurring expenses that are needed to keep a system functioning. In a lot of cases, this is disregarded at first, but this can be a decisive factor in the purchase of a certain system or setup.

7.2.1 Licenses

Liferay

For an Enterprise like the Karelia University of Applied Science (UAS), the Enterprise Edition (EE) is more or less mandatory. This is because they cannot risk the portal site being down for a long period of time. This means that the university will need a license for this Enterprise edition. Of course, it is possible for the university to choose for the Community Edition (CE) as well, but this is a risk.

For the Enterprise Edition a license is needed. There is no information available online about the price of these licenses, because they are dependent on the setup of the customer. In order to get a Liferay Enterprise Subscription Quote, you can send

an e-mail to the Liferay team, where you include your setup, the amount of CAL's you need, the planned redundancy, and so on. We sent an e-mail to the sales department of Liferay, explaining that we were students doing our thesis, and explained our project, but we did not receive an answer by the time this document got published.

SharePoint

We took a look into the different licensing systems SharePoint uses, and saw there is a wide variety of different licensing options you can choose from. One of these licenses is the “SharePoint Online for Office 365” license. At first glance, this looks very interesting, since this is a complete cloud-based solution. In other words, there would be no need to have a server at the Karelia UAS to run SharePoint. However, there are several problems with this solution. For example, the maximum storage amount per person is 500 Megabytes. This can be an issue when students start to upload their assignments to the SharePoint server.

The counterpart option is to run SharePoint on premise. There are three options when it comes to this. One of these, the SharePoint Foundation edition, we can disregard, as this does not nearly give us enough options. Beside this, there is the choice between the SharePoint Standard edition and the SharePoint Enterprise edition. This choice is up to the administrative and IT department of the Karelia UAS. There are several tools and features in the Enterprise edition that are not in the standard edition. So in order to make this decision, the costs and benefits should be compared; a thing we do not have enough insight in the structure of the university for.

Another small, but noteworthy issue is the list of the blocked file types. For the most part, any issues with blocked file types can be avoided by packaging the blocked file types into a .zip or a .rar file. However, if this would prove to be an issue, taking a look at the list of the blocked file types is advised. This list is included in the bibliography.

The things that will influence the costs of your SharePoint setup are listed below:

Name	Description
CAL	Client Access License (one per user). There is a difference between the standard and the enterprise CAL's, as well as intra- and internet CAL's.
SQL server	The amount of SQL servers connected to your SharePoint environment
WFE server	The amount Web Front-End servers. These are the servers that you access by HTTP, either from the intranet or the internet

Table 23: Sharepoint Costs

As you can see, with prices like this, the total budget grows very quickly. Using an online tool to get a rough estimation of the costs of 5000 CAL's, 1 SQL server and 1 WFE server, we received a price tag of $700.977 for the package with the standard CAL's, and $1.145.576 for the enterprise CAL'S. Given the current exchange rates, this would roughly translate to respectively €513.840 and €839.745. (Microsoft, 2014)

7.2.2 People

The amount of people and skills required to keep the portal website up and running depends on several factors. If the server runs on premise, someone with knowledge of the server's hardware is needed. Also, there should at least be two persons in the organisation who know everything about the server. There are two reasons for this: availability and redundancy. As an organisation, you cannot afford the portal website being down while your technician is on a holiday, and nobody else is capable of solving the problem. On the other hand, having two technicians also ensures that there is less of a risk to lose the portal website, if something happens to the technician, or he decides to quit his job.

Another decision that will influence your requirements of the people, is the type of portal website you choose.

Liferay

For the maintenance of the server, a person is needed who knows some basic things about Liferay. The reason being that, for example, when a user forgets his password, the administrator can reset the password, or if the server is not working correctly, the administrator can quickly intercept.

As an administrator, navigating through the administration page is very intuitive, although it requires some knowledge about IT. This job can easily be done by one or two dedicated administrators, as the Liferay server does not need a lot of attention when it is running in a normal state.

It is also advisable that this person knows a thing or two about the Java programming language, as well as JSP and JavaScript. As mentioned earlier in this document, Liferay uses a customized version of these programming languages, so a minimal training will be required. However, for someone who mastered at least two of the three programming languages, this should not prove to be a difficult task.

Another attention point is the database. Because the choice of the database is up to the IT department of the university, we cannot focus on a specific database in particularly. But regardless of the database type, the people responsible for the portal website should be familiar with the database system, so they can at least trace problems and perform some basic queries to fix them.

Since Liferay is cross-platform, we cannot predict the operating system it will be running on, but this should also be taken into account when choosing people. The main difference of course would be the choice of a Microsoft versus a Linux operating system, as these are the two most popular options when it comes to servers.

SharePoint

For the maintenance of a SharePoint server, it is very advisable to have somebody who knows the VB.NET and ASP.NET programming language, as well as Microsoft SQL Server. Since SharePoint can only run on Windows Server, this person should also be familiar with this operating system. This person should also be experienced in using the PowerShell scripting language, as this is how portlets are added to the SharePoint web portal. He or she should also be familiar with concepts like Services, the registry and ports.

The SharePoint administrator interface is also less intuitive then the Liferay administrator interface. Most parts of the administration are accessible from the web interface on port 80. However, for some more advanced configurations you need to log on to the server using another port. This can make it really confusing for people who are not very tech-savvy.

7.2.3 Server Maintenance

To keep the server up and running, there will be some costs. These are either, as explained earlier, costs for electricity, and maybe hardware changes, or a fee to let another company host the server.

Assuming the server would run on premise, these costs will for the main part consist of electricity and internet connectivity, as well as the space and maintenance for this server. The advantages of having the server on premise is that you have the possibility to alter and configure the servers however you want.

If the university decides to outsource the maintenance of the server, the costs of electricity, internet connectivity and maintenance will all be reduced into the form of a monthly fee. The main advantage is that this way, the maintenance of the server is greatly reduced. The main disadvantage is that the server administration possibilities are reduced this way.

7.3 Conclusion

When you compare Liferay to SharePoint, the main difference is the price tag. Liferay and SharePoint are more or less equally powerful as a portal website software, but they work in a different way, and offer a very different way of product support. Both the start-up and maintenance costs of a Liferay portal website environment are cheaper than those of its competitors.

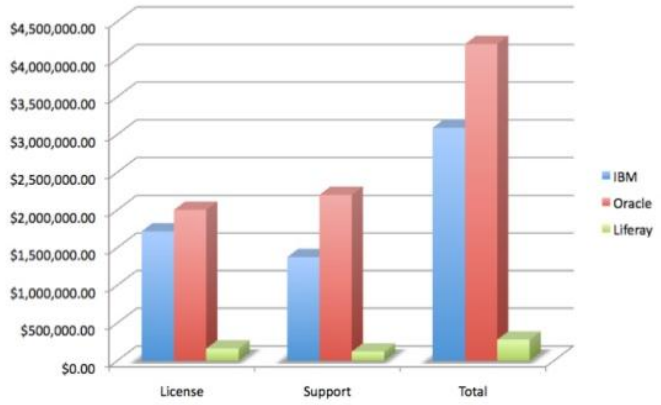


Image 5: Costs Liferay and Competitors

In Image 5 you can see a graph of the price, both license and support. This image shows Liferay compared to the competitors IBM and Oracle, and not SharePoint, but SharePoint follows the price trend of IBM and Oracle, compared to Liferay. These prices are for normal customers, and for the usage of an Internet license.

8 LIFERAY BUG & DEVELOPMENT ISSUES

8.1 Side Navigation

During the development of the sidebar navigation, we discovered our first issue. We already had some difficulties with integrating the sidebar navigation in the theme, as it is a separate file, and not part of the actual theme. The good thing about this is that we could change the navigation file as we wanted, and it would not influence the actual theme. The only thing we changed in this navigation file was the formatting. Without formatting, everything is displayed horizontally; something we do not want. Instead, as we made use of the Semantic UI framework for the collapsible menu system. Later we found out that Liferay uses bootstrap for its interface, which might have caused the problem that we had.

When we finished the modifications of the navigation.vm file and ran it on the development server, we encountered a strange phenomenon. On first sight, everything looked well; the navigation bar formatted as we wanted it to. The JavaScript did not work yet, but that was normal, as we did not implement it yet. The strange thing, however, was that the administrative bar to add portlets or make changes in the page structure were non-responsive. This, we found very strange, as we did not change any code or any files except the navigation.vm file, which should not have the ability to make this happen. When we started to strip down the content of the navigation.vm file to identify the problem, we found out that a certain CSS setting caused this to happen. Later we figured out that this might have been caused by conflicting CSS rules in Bootstrap and Semantic UI (Semantic UI, 2014).

8.2 Language Drop Down List

There is also a problem with the language selector. The problem is that the language goes back to default every time a user logs in, and cannot be changed there. We looked into this, and it seems that language is stored in a cookie. The problem is that the cookie does not get saved in the language selector, and when you manually edit this cookie (for example from English to Finnish), this works on the login page, but gets reset to default (English) every time the user logs in. The language restore can be explained by the saved settings in the database. In the default table user, we can see that the field “languageId” is not updated. It never changed, even if we change the language in the drop down list. Due to this, we browsed to several forums, this confirmed our suspicions. This seems to be a well-known bug in Liferay. This bug does not exist on the private pages in Liferay, but only on the public pages. (Liferay Forums: Language issue, 2011)

8.3 SSL Heartbleed

Since the discovery of the Hearthbleed bug in SSL, it is required to upgrade Openssl. (Wikipedia, 2014) We found out that the upgrade process of Openssl in our test-environment was not successful. The different upgrade process we used are the following:

- Upgrading Openssl with latest version on the website
- Upgrading Openssl with aptitude
- Upgrading kernel of Operating system

After trying those 3 methods we could remark that Openssl was upgraded on our host, but Apache Tomcat was still using Openssl 1.0.1. (Hearthbleed, 2014)

8.4 Navigation Arrows Not Shown in the Calendar

After deploying the default calendar module we discovered a few bugs. The first bug was the navigation arrows for changing months were not displaying. (Choi, 2013)

This is a well-known bug in Liferay. To fix this bug, we had to edit multiple default classes from Liferay. The reason why this bug emerges is because instead of using CSS to draw the navigation arrows, they used images. Those images did not exist. Therefore, whenever the calendar portlet is displayed, the server responds with the HTTP error 404 when the images are requested. To fix this error, we have to edit the following classes:

- Calendarnavigator.js
- Calendar navigator.css & calendarnavigator-skin.css
- Calendarnavigator-core.css

Once all the files are edited and saved, reboot the application server and navigation arrows will be displayed. (lundgren, 2014)

8.5 Portlets Working, but Throwing Exceptions

During our prototyping we made one portlet based on an existing portlet. But when we deployed both on our server, we saw that the server was throwing exceptions. These exceptions were thrown because both portlets were running simultaneously. Both portlets were asking for the same resources and the connection of the first Portlet did not successfully close. Therefore the second portlet was throwing an exception. An exception that is thrown must always be caught. But with the type of the thrown exception was impossible to catch. The exception was of the *ClassCastException* type. This means that the object we were trying to cast was not an instance of that type of object.

ClassCastExcption means that the code has attempted to cast an object to a subclass of which is not an instance. Related to the exception we got in our portlet. We can say that both portlets used the same resources. Since the resources where single instance and that both portlets were trying to access them and manipulate them. This would lead into throwing an exception, which would lead into a working portlet, but with a security bug. Therefore whenever a portlet is created based on an existing portlet. It is better to hook them or create them from the scratch.

8.6 Liferay Sync will not Detect EE Version

Liferay Sync is a simple drag-and drop file sharing system. Liferay Sync is designed to work on Windows, Mac OS, Android and IOS. The reason why Liferay Sync so attractive it is because it centralizes documents and files. It even has a build in log for document versioning.

When there are no public pages in the website, Liferay Sync will not be able to detect an Enterprise Edition of Liferay. This can be very annoying, as Liferay Sync enables you to use the principal of versioning and revision tracking, making it easy to track an individual work for a group project.

The bug has been reported to Liferay but is still in progress to be fixed. Therefore it is recommended to always have at least one public page. (Ju, 2013)

8.7 Rebuilding Plug-ins and Portlets after Liferay Upgrade

Whenever a new version of Liferay is released, all plug-ins and portlets must be rebuilt and recompiled. If source code is not available this means that certain portlets have to be rebuilt from scratch. If this is not done, portals and plug-ins will not work anymore. Therefore it must always be kept in mind that upgrades can boost the performance of the system, but it can be that the upgrade is a setback in terms of productivity and usability.

8.8 End-of-Life and Last-Ship-Date

End-of-Life and Last-ship-date are two important dates for a company. Whenever the LSD occurs this means it will no longer be shipped nor installed for the client. This means whenever a catastrophic error happens, the client is forced to upgrade to a newer version. As mentioned in “8.7 Rebuilding plug-ins, portlets after upgrade Liferay”, this upgrade is risky. Therefore it is recommended to prepare for an EOL and LSD.

8.9 Page Non-Responsive, but No Errors Shown

It often happens when you're developing in Liferay that you'll encounter some problems while testing. One of the most common problems that you will encounter is the fact that sometimes parts of the website are not responding, or behaving like they should, but you get no output in the console. If you do not know what is wrong, you have to look for the problem manually. This can be cause a huge waste of time.

8.10 Default Login Portlet

In our case, we do not want to allow strangers to create accounts on our website, so we need to disable that option. You can remove these options in the configuration panel, but you cannot remove the openID login option. We do not want this option either, so we created a custom login hook, that removes all sign-up options, as well as all sign-in options that are not the native Liferay e-mail and password combination.

8.11 Blocked Ports

While developing our prototype, we stumbled upon the problem of blocked ports. The problem occurred while developing the mail portlet. Because we could only test the portlet with our own e-mail addresses, we had to use a G-Mail address. The problem here was that the ports for the SMTP service were blocked. Luckily, port 53 was open, allowing us to create a VPN connection, and thus allowing us to access G-Mail's SMTP servers this way.

9 REMARKS

This section will clarify the multiple remarks we have concerning Liferay and SharePoint. Most of the remarks were found during development and testing. Those remarks must be taken into account whenever someone decides to use Liferay and use their own development team.

9.1 Language Influence

Whenever a language like Hebrew is chosen, the layout of the portal is completely changed. The reason is because in Hebrew, the read-direction is not left-to-right, as in most other languages, but the other way around. Therefore it is recommended to only set only languages that use the same reading direction, or keep this in mind when you implement your theme. (Liferay, 2012)

9.2 Test Hooks Completely Before Deploying on Hosting Server

Hooks are commonly used in Liferay. Those are changes in default delivered portlets or applications. These simplify a lot of work, but during the development of the prototype we struggled multiple times with hooks. During deployment of hooks we found out that certain hooks change core files. If for any reason the hook is not working as expected, a complete restore of the changed files is required. The best way to execute this operation is by keeping a backup of the original files on a third-party device or back-up folder on the application server.

Before deploying the hooks to the application server, we recommend to deploy the hook in a test-environment. This will minimize the risks of completely ruining your server.

9.3 Development on Remote Server

First we developed directly on our local machine. But for an optimal performance it is required to have a fairly large amount of system memory. Most of our machines were not efficient enough. Therefore we decide to develop using a remote server. For developing on a remote server it is required to activate the developers' mode and install a plug-in called Remote server plug-in. This plug-in can be downloaded for free from the Liferay website.

9.4 Java Regular Cleanup

Since Liferay uses Java, there can be an issue if no cleanup is done on regular basis. The major problem with Java is that when updated it could leave an older version still running on the hosts. The reason why this is implemented in Java is because a lot of companies use Java. And when a new version of Java comes out certain modules no longer work. And therefore older versions can be found on a host. This is great for companies, because their software is still working perfectly, but this is bad for security. Because a lot of Java updates are patching security holes.

To fix this problem, you can either uninstall the older version, or change the PATH variables to the newer version of Java. (Hartsock, 2011)

9.5 Attention Points for SharePoint

Unlike Liferay, SharePoint is a closed software. (Flemming, 2011) This means that the freedom related to tweaking limited is. In this section we will discuss some major problems with SharePoint. Those remarks should be taken into account before deciding to use SharePoint as a portal website.

9.5.1 Browser Limitation

Not every browser will be able to work with a SharePoint based portal website. This is due the required plug-in: Silverlight. Can be downloaded for free for most commonly used browsers. But when you try to download Silverlight for Google Chrome, it will not work. If you want to make use of Microsoft Silverlight in Google Chrome you have to activate it. To activate it, you need to complete the following 5 steps:

1. Download Microsoft Silverlight
2. Install Microsoft Silverlight
3. Navigate in google chrome to chrome:plugins
4. Enable the plug-in of Microsoft Silverlight
5. Restart Google Chrome

9.5.2 Database Limitation

During an older project we came across the list view threshold limitation. When trying to show a list with more than 5000 elements in it, the user will receive a message telling the list cannot be shown. The error message the user will receive is the following one: “This view cannot be displayed because it exceeds the list view threshold (5000 items) enforced by the administrator. (KILIÇ, 2014) To view items, try selecting another view or creating a new view. If you do not have sufficient permissions to create views for this list, ask your administrator to modify the view so that it conforms to the list view threshold.”

To fix the problem there are 2 possible solutions. The first one is: Increase the threshold value to a limit that will never be reached. But this is dissuaded. The reason why we are against this is simple: when the value is increased over 5000 elements, SharePoint will completely lock the table. This means that it is impossible for another user to insert, change or delete rows in that table during that time-lapse.

The second solution is by applying the techniques for managing large lists. (Microsoft, 2014) It is a course of Microsoft for advanced SharePoint administrators. After completing the course the SharePoint administrator will be able to do:

- Learn what the List View Threshold is, and understand its benefits
- Create an index so that you can see more information in a view
- Create folders to better organize your large list
- Use Datasheet view for fast filtering and sorting of a large list

9.5.3 Backing up SharePoint

To make sure no information is lost during a catastrophic incident, backups must be made on regular base. In SharePoint there are 2 ways of creating a backup. The first one is a full back-up. The full back-up will store each created page on a remote storing device. This can be done automatically. But experience showed us that the process of backing-up is long. And during that time, the SharePoint server must be shutdown.

The second way of preparing for an incident, is by backing-up the database, and in case something would happen due to an incident, we just restore the database and let SharePoint rebuild every page. This process of back-up is less time consuming. And the downtime is minimized.

9.5.4 White Screen of Death

The white screen of death of SharePoint, can be very frustrating for system administrators. When the white screen of death will possibly occur is known. When a server is newly installed, and for some reason one of the core components, failed to install or load properly, that's when the white screen of death will appear. The symptoms are easily determined. When you are trying to connect to the SharePoint server, you will only see a white screen. If you look in the logs, you can clearly see that nothing is happening. The best way to fix it, is by removing and unregistering all services related to SharePoint and reinstall SharePoint on the server.

10 CONCLUSION

Liferay is a powerful and easy to use open source solution, with the best price/quality ratio. Therefore we can say that Liferay is the most appropriated solution for Karelia University of Applied Sciences. It offers a centralized portal website, where implementation of actual service is possible. However, for security purposes, we advise to discard Pakki and other homebrew software, and upgrade Moodle and patch the security leaks. Those services are insecure and can be breached by a simple man-in-the-middle-attacks or by packets sniffing. The transfer to the new system is possible in a time-lapse of two years. Since Liferay offers the possibility to implement an “Active Directory”, we advise to keep the current setup. The AD can be used to assign the permissions and user authentication possibilities.

The reason why it would take around two years to develop the student service portal, is because the development of the portlets is custom fit. This means that even if some portlets can be used from Liferay or Vaadin, there is a chance that they need to be changed in the source code to be useful for the deployment of Karelia UAS. Therefore the Karelia UAS has to establish trust with the development company. If the University of Karelia decides to upgrade to a newer version of Liferay and all portlets are written in the default Liferay JSP-language, the source code of each created portlet must be changed to match the new version. This can be avoided by collaborating with the company which develops Vaadin portlets for Liferay. In that case, whenever a new version of Liferay is installed on the server the only thing the system-administrator has to do is to check if Vaadin is installed. Moreover if a new version of Vaadin is available, there is no need to change the source code of the portlet. After the upgrade of Vaadin, there is only the need to recompile the widget-set and to redeploy all the Vaadin portlets.

At last, we would like to recommend that users such as “System Administrators” follow the training given by Liferay. With the course, administrators will know how to install, manage and recover a deployed solution. They will be the key persons to ensure that the downtimes are minimized and that the performance of the portal website is always as high as possible.

REFERENCES

Apache Tomcat 7. (2014, March 25). Retrieved May 6, 2014, from The AJP Connector: http://tomcat.apache.org/tomcat-7.0-doc/config/ajp.html

Choi, S. (2013, October 09). *Issues Liferay*. Retrieved April 22, 2014, from Lifreay: https://issues.liferay.com/browse/LPS-40772

Flemming, R. (2011, July 20). *Ten of the best SharePoint University websites*. Retrieved June 11, 2014, from Microsoft Education Blog: http://blogs.msdn.com/b/education/archive/2011/07/21/ten-of-the-best-sharepoint-university-websites.aspx

Grievous, G. (2013, oktober 13). *Youtube*. Retrieved March 18, 2014, from How to install liferay on Ubuntu 12.4 using tomcat and MySQL: https://www.youtube.com/watch?v=3uLWkqCg6kg

Hartsock, D. (2011, november 30). *You should junk your java!* Retrieved april 28, 28, from daves computer tips: http://www.davescomputertips.com/you-should-junk-your-java/

Hearthbleed. (2014, April 09). *Hearthbleed*. Retrieved April 14, 2014, from Hearthbleed bug: http://heartbleed.com/

Intalio. (n.d.). *Intalio*. Retrieved March 28, 2014, from Intalio: http://www.intalio.com/products/bpms/overview/

Ju, D. (2013, January 13). *SYNC-754*. Retrieved April 24, 2014, from Liferay Issues: https://issues.liferay.com/browse/SYNC-754

Liferay. (2011, June 7). *Liferay Forums*. Retrieved April 15, 2014, from Liferay: http://www.liferay.com/community/forums/-/message_boards/message/9205158

Liferay. (2012, July 3). *Liferay Forums*. Retrieved May 15, 2014, from Liferay: http://www.liferay.com/community/forums/-/message_boards/message/14687552

Liferay. (2012, Jully 6). *Liferay Forums*. Retrieved April 15, 2014, from Liferay: https://www.liferay.com/community/forums/-/message_boards/message/14736008

Liferay Clustering. (2014, June 10). Retrieved June 10, 2014, from Liferay: https://www.liferay.com/documentation/liferay-portal/6.1/user-guide/-/ai/liferay-clusteri-2

Liferay Forums. (2011, September 7). Retrieved April 15, 2014, from Liferay: http://www.liferay.com/community/forums/-/message_boards/message/10703163

lundgren, m. (2014, February 23). *Images should not be used to draw triangles. we have css*. Retrieved April 22, 2014, from github: https://github.com/marclundgren/yui3/commit/d4220bbee1b995e71eff63429d23c3fd285c995a

Microsoft. (2014, June 11). *Office.Microsoft*. Retrieved June 11, 2014, from SharePoint list 5: Techniques for managing large lists: http://office.microsoft.com/en-001/sharepoint-server-help/overview-RZ101874361.aspx?section=1

Microsoft. (2014, June 9). *TN SharePoint On Premise*. Retrieved June 2014, 9, from Technet.Microsoft: http://technet.microsoft.com/en-US/library/jj819267.aspx#bkmk_FeaturesOnPremise

Microsoft. (2014, June 9). *Types of files that cannot be added*. Retrieved June 9, 2014, from Office.Microsoft: http://office.microsoft.com/en-us/office365-sharepoint-online-small-business-help/types-of-files-that-cannot-be-added-to-a-list-or-library-HA101907868.aspx

Olin Business School. (2014, June 11). *Homepage*. Retrieved June 11, 2014, from Olin Business School: http://www.olin.wustl.edu/EN-US/Pages/default.aspx

Rouse, M. (2014, may 27). *search soa*. Retrieved from UUID (Universal Unique Identifier): http://searchsoa.techtarget.com/definition/UUID

Saxena, A. (2010, October 26). *Liferay Portal*. Retrieved April 25, 2014, from blogspot: http://abhishek-liferay.blogspot.fi/2010/10/hardware-requirements-and-sample-of.html

Semantic UI. (2014, June 11). *Semantic UI*. Retrieved June 11, 2014, from Semantic UI: http://semantic-ui.com/

Sezov, R. J., Hinkey, J., Kostas, S., Rao, J., & Hoag, C. (2013). Using Liferay Portal. In R. J. Sezov, J. Hinkey, S. Kostas, J. Rao, & C. Hoag, *Using Liferay Portal : A Complete Guide* (p. 683). Liferay Press.

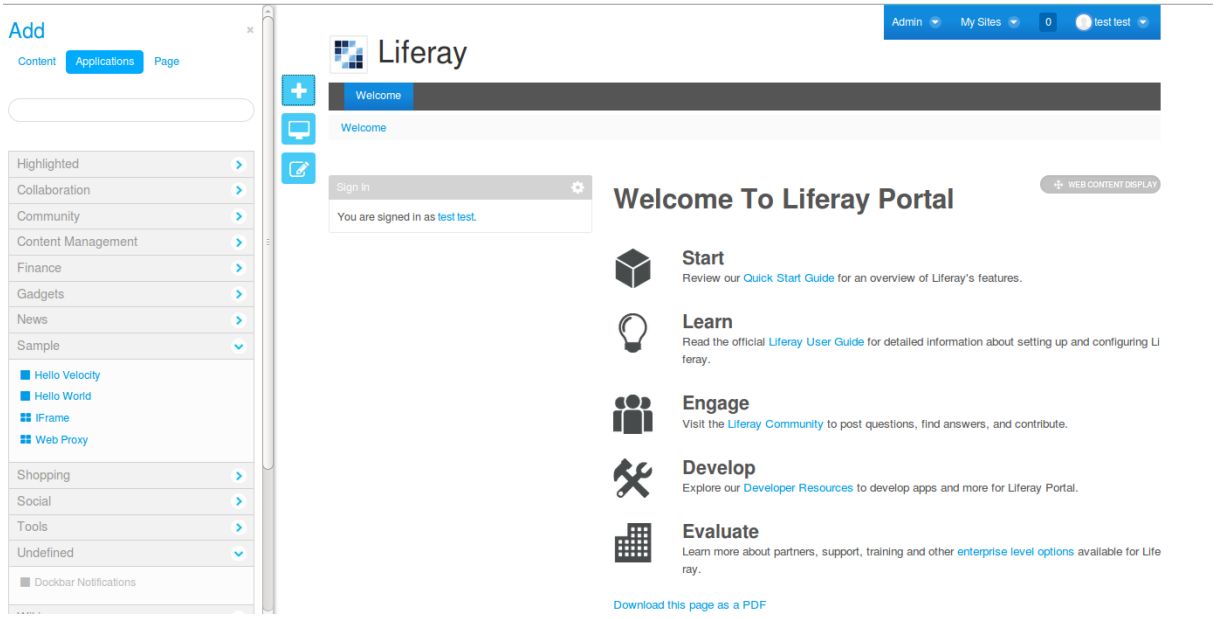
Stackoverflow. (2014, June 11). *Stackoverflow*. Retrieved June 11, 2014, from Exceeds the list view treshold 5000 items in SharePoint 2010: http://stackoverflow.com/questions/6093955/exceeds-the-list-view-threshold-5000-items-in-sharepoint-2010

Wikipedia. (2014, April 17). *Wikipedia*. Retrieved June 6, 2014, from HSQLDB: http://en.wikipedia.org/wiki/HSQLDB

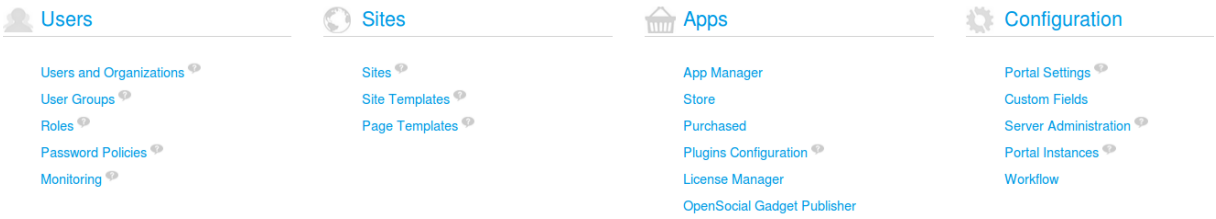
Wikipedia. (2014, April 9). *Wikipedia Hearthbleed*. Retrieved April 14, 2014, from Wiki: http://en.wikipedia.org/wiki/Heartbleed

Appendix I

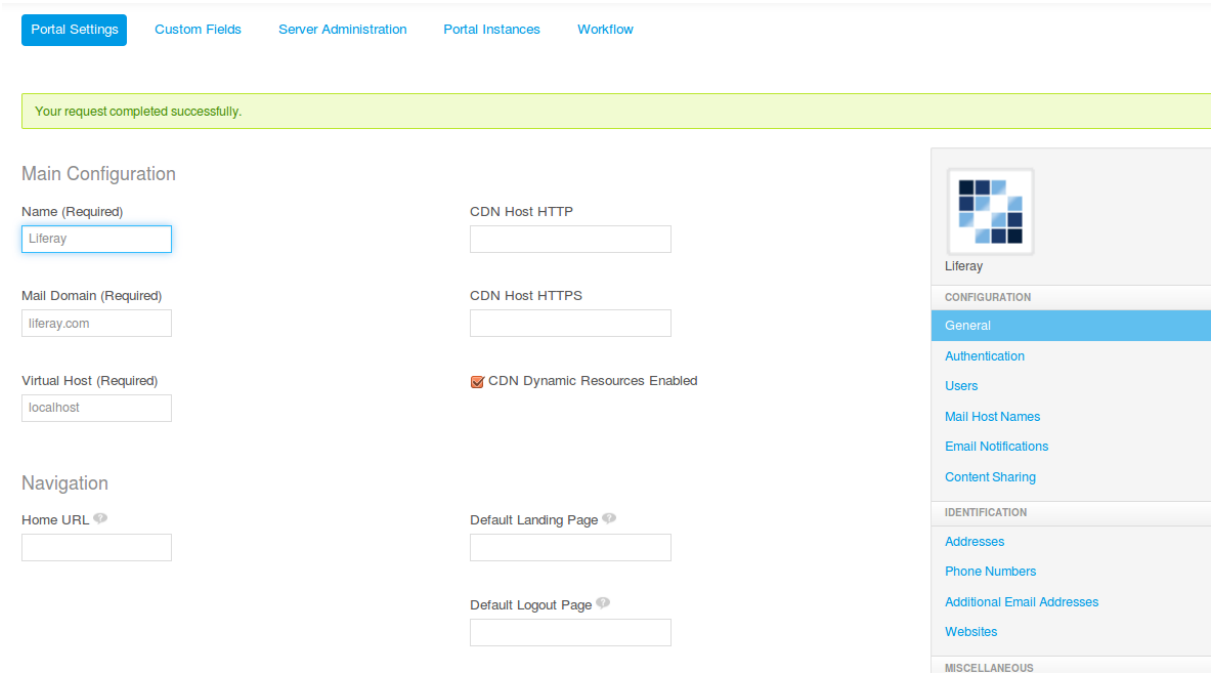
Default view of the Liferay homepage, with the portlet bar.



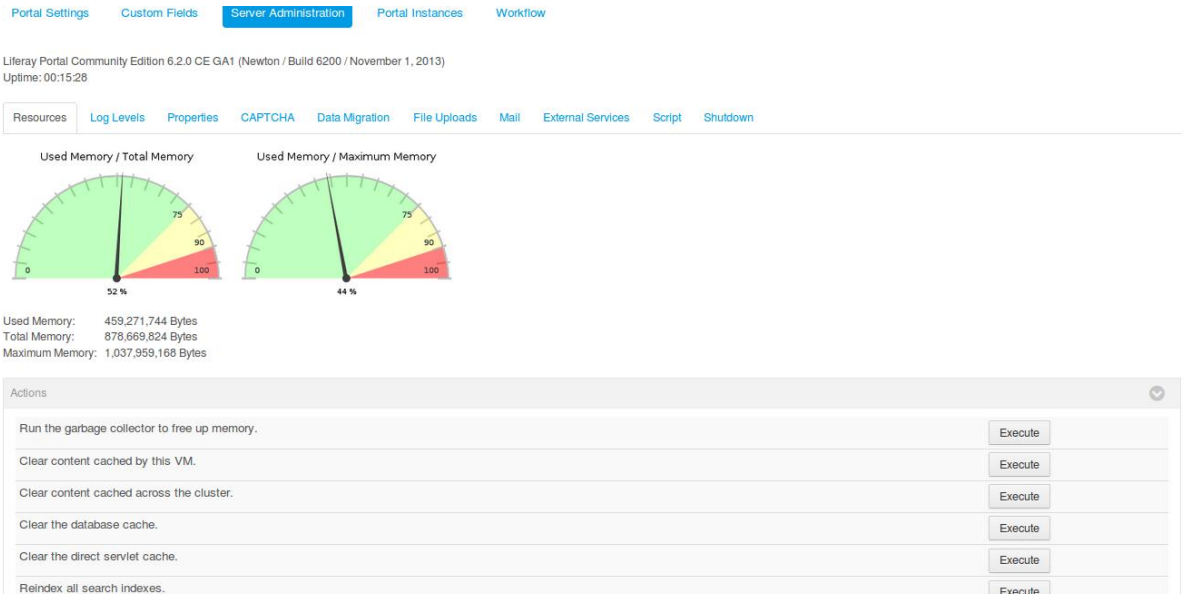
Liferay Configuration Panel



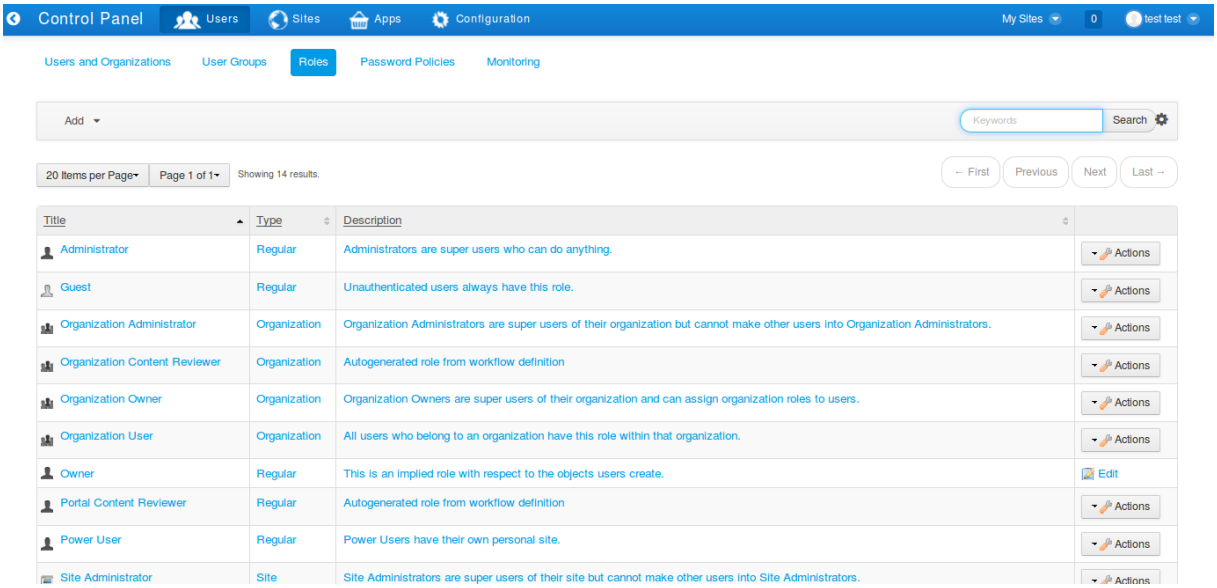
View of the Liferay default configuration



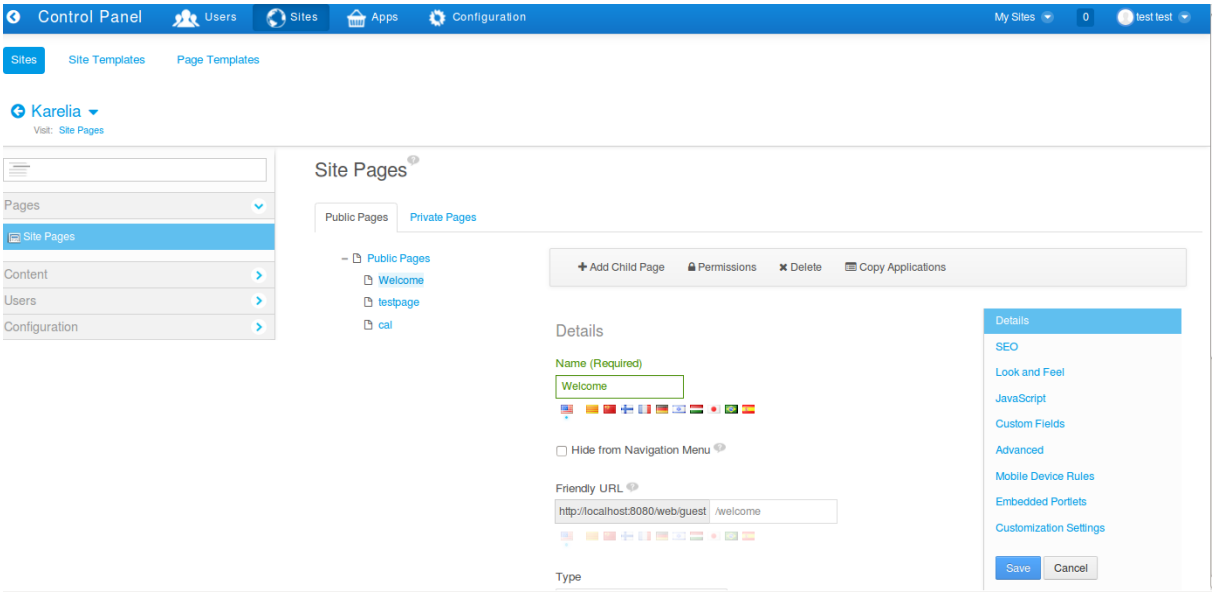
Liferay Server administration



Liferay roles

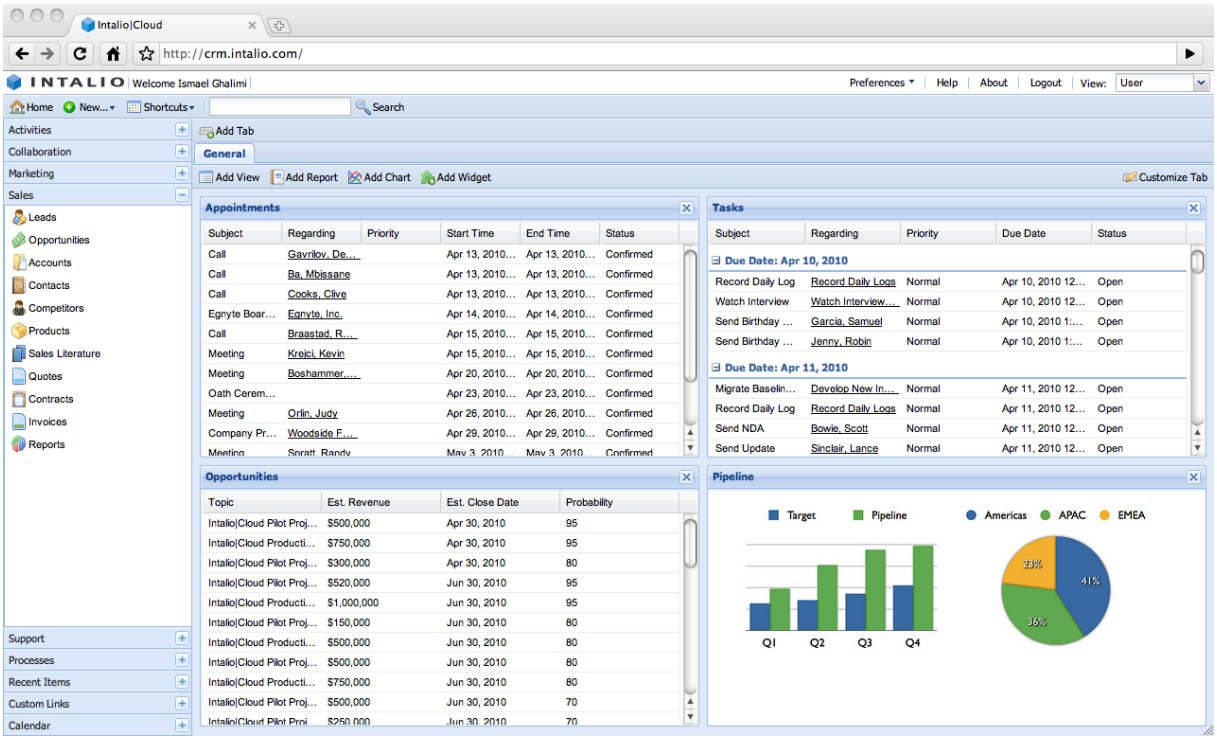


Liferay Page Management

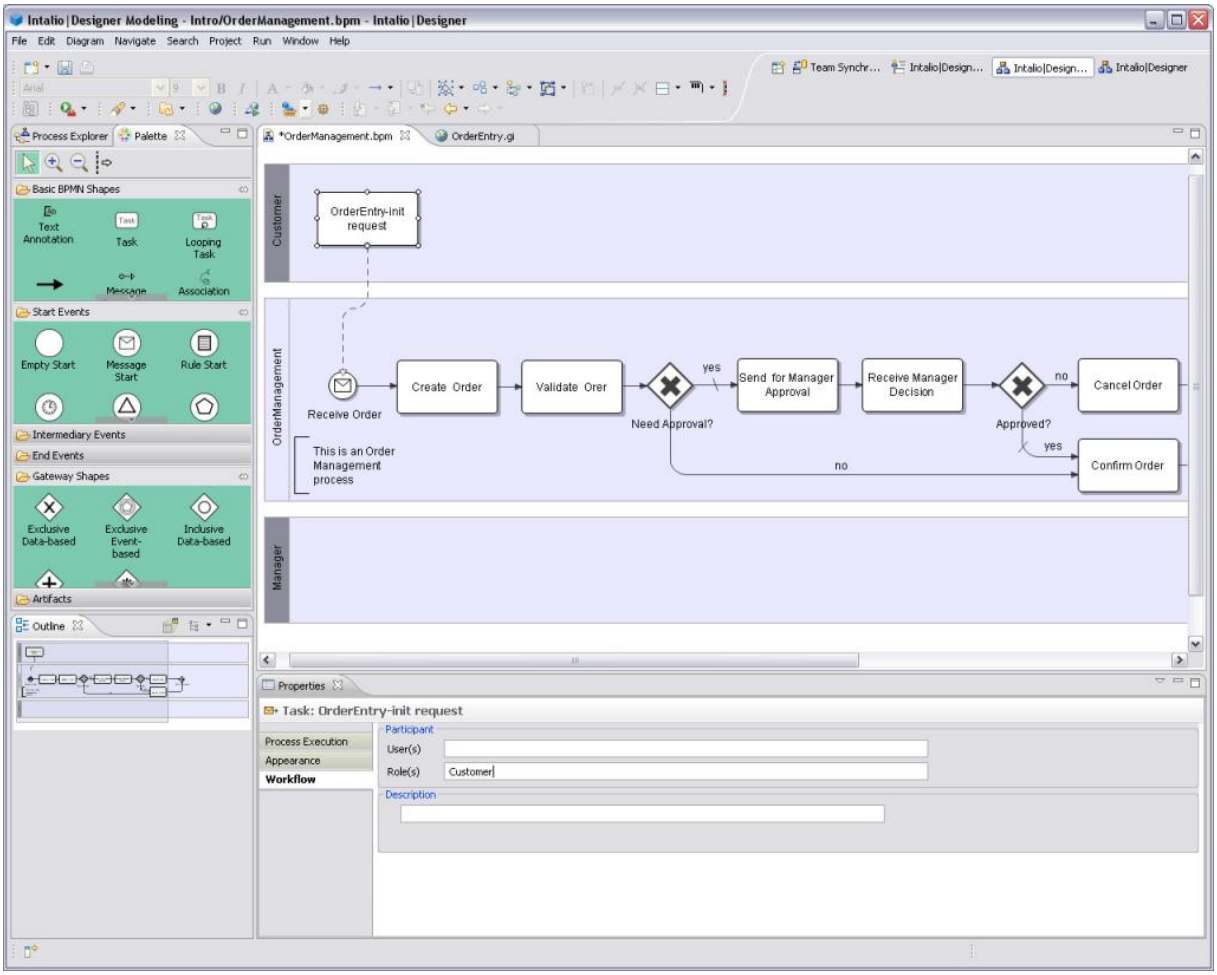


Appendix II

Intalio dashboard

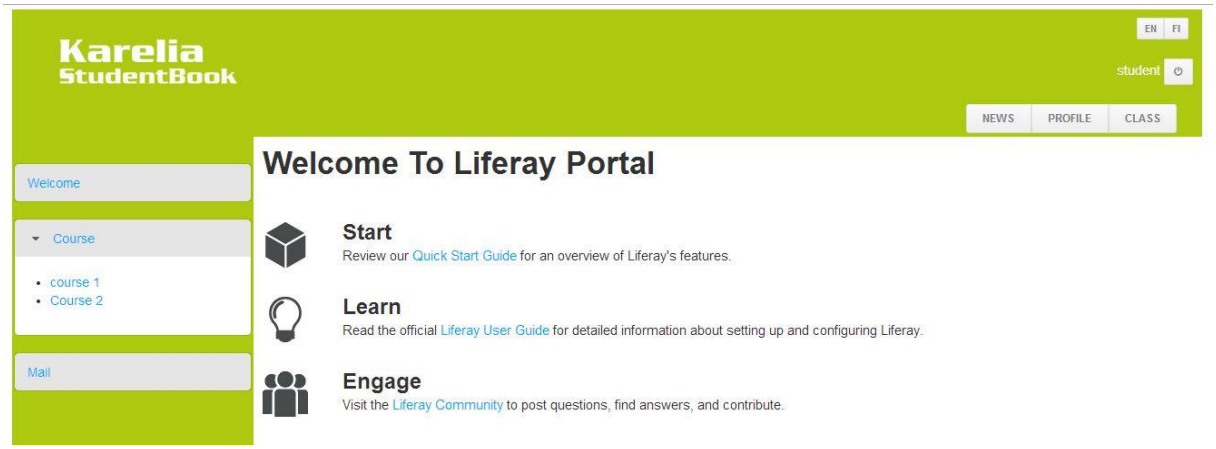


Intalio Business Process Management System

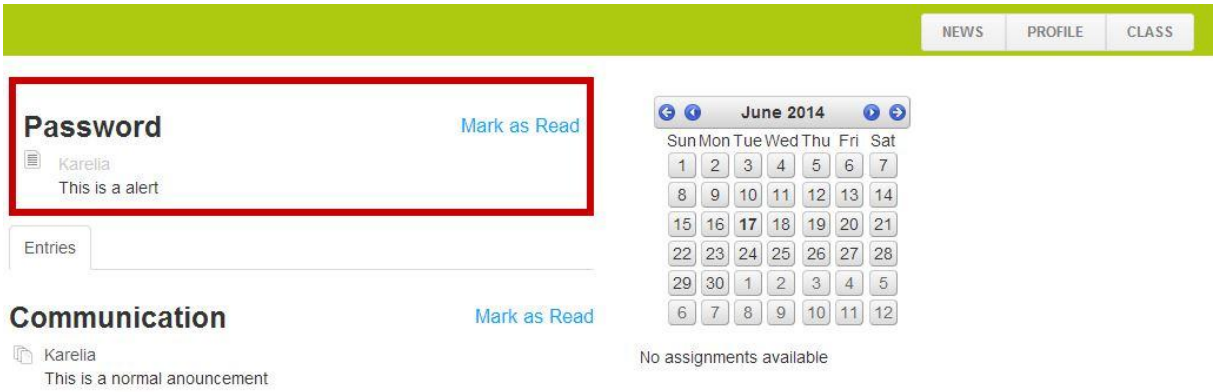


Appendix III

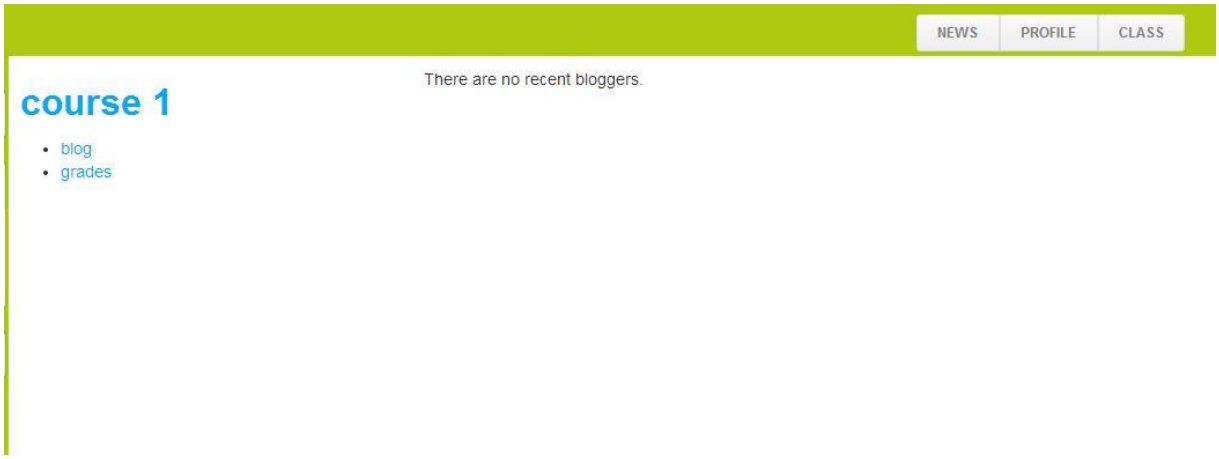
Prototype: Student view login



Prototype: Announcement and assignments

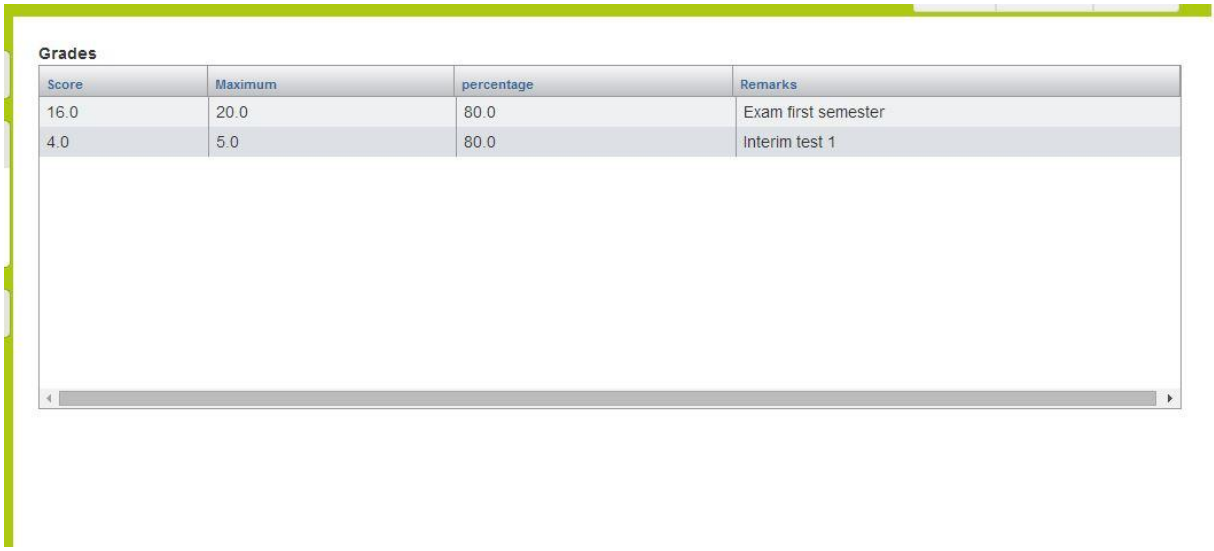


Prototype: Course view



Prototype: grades all courses

Prototype: grade 1 course



Grades

Score	Maximum	percentage	Remarks
16.0	20.0	80.0	Exam first semester
4.0	5.0	80.0	Interim test 1

Prototype: Assignment posted by teacher

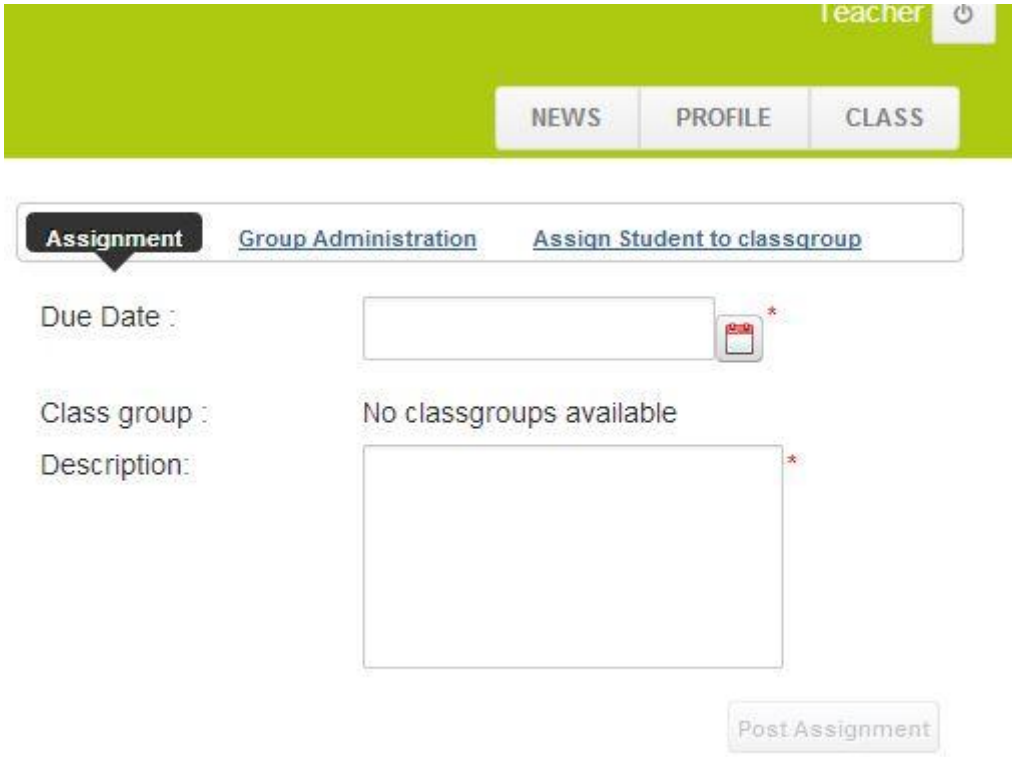
June 2014

Sun	Mon	Tue	Wed	Thu	Fri	Sat
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	**17**	18	19	20	21
22	23	24	25	26	27	28
29	30	1	2	3	4	5
6	7	8	9	10	11	12

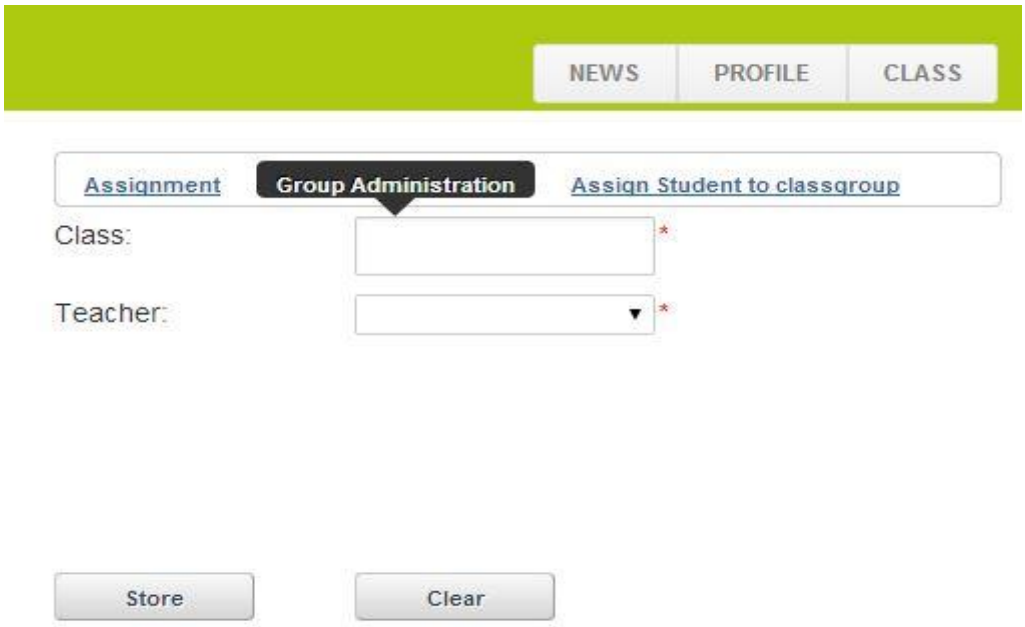
Upcomming events :

Due date:	Task description:	Class:
27/06/2014	Test on the 27the of june	Class 1

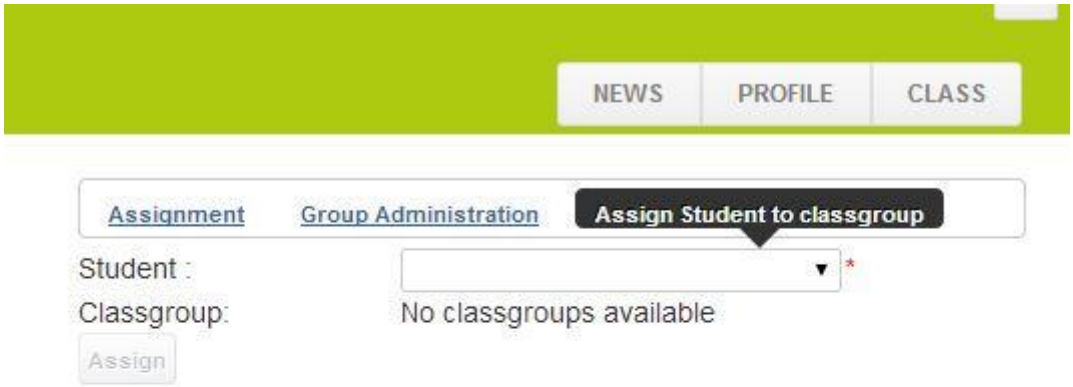
Prototype: Assignments teacher module



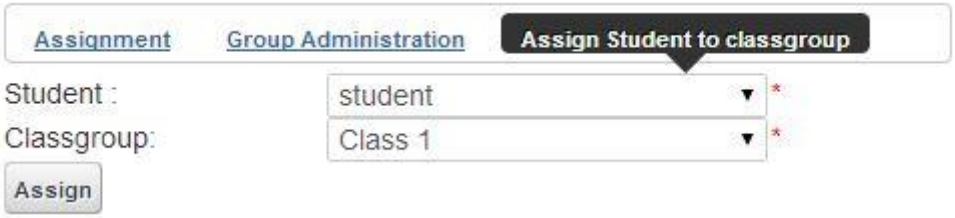
Prototype: non mandatory administration view



Prototype: non mandatory student assign to group module



Prototype: assign Student to class group after class creation



Prototype: Confirmation dialog assignment poster