Masters in Informatics Engineering

Internship
Final Report

An E-learning Platform Development

Guilherme Filipe Gonçalves Palma Barão dos Santos

gbarao@student.dei.uc.pt

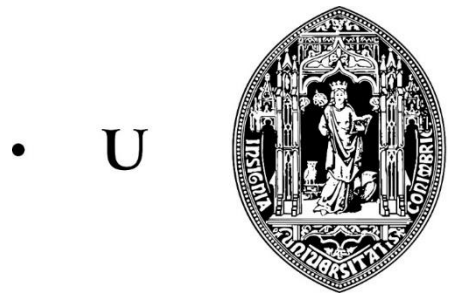
Supervisors:

Dr. Tiago Cruz

Eng. Serge Nunes

01 July, 2016

FCTUC DEPARTAMENTO
DE ENGENHARIA INFORMÁTICA
FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE DE COIMBRA



Masters in Informatics Engineering

Internship
Final Report

E-Learning Platform Development

2015/2016

Jury 1 : Dr. Nuno Laranjeiro

Jury 2 : Dr. Fernando Barros

Author:

Guilherme Filipe Gonçalves

Palma Barão dos Santos

2009111685

gbarao@student.dei.uc.pt

Supervisors:

Tiago Cruz

tjcruz@dei.uc.pt

Serge Nunes

sergenunes@konkrets.pt

January 22, 2016

DEPARTMENT OF INFORMATICS ENGINEERING

FACULTY OF SCIENCES AND TECHNOLOGY

Acknowledgements

I would like to thank my company supervisor engineer Serge Nunes for the opportunity and trust shown in me to integrate this project. I would also like to thank my department's supervisor, Professor Tiago Cruz for all the guidance provided, for having the patience to answer my never ending queue of questions and for having accepted to supervise my internship on a short notice. Last but not least I would like to thank my family, girlfriend and friends for all the support and comprehension in both the good and the bad moments.

Abstract

In order to complete my Master's degree in Informatics Engineering at the Faculty of Sciences and Technology of the University of Coimbra in 2015/2016, I had to complete a yearlong internship so, from an extensive list of options I ended up choosing Konkrets, Lda which is a consulting startup headquartered in Lousã, Coimbra. The company has an instructorship branch as well. However, the classes were hosted locally on a face-to-face basis which made it difficult for the practice to expand from a regional standpoint. Also, it was deemed necessary by the company's administration to reduce the available budget for physical materials and infrastructure to augment the overall profit.

Therefore, to assist the company achieve such goals it was decided that there was the need for the learning process to evolve from a face-to-face to a remote basis so that both issues could be resolved. So, for my internship I have developed an e-learning platform from scratch that would suit the company's needs as the free ones available didn't provide the functionalities required.

This report serves as documentation for the devising and development of such a platform while describing the necessary steps required to replicate the process. Initially the document focuses on providing the reader with an overview of both the company and the project as well as a state of the art of the platforms in the market followed by the process used throughout the project. In the middle sections lie a description of both functional and non-functional requirements alongside with the architecture that was chosen. Last but not least in the final chapters there is an overview of the final product as well as the verification and validation of the requirements.

Keywords: e-learning, platform, WebLink, scalability, rational unified process, deployment pipeline, accountability, spring MVC, Security, Model-view-controller, usability, Docker, Jenkins, streaming, blended-learning, interoperability, TokBox, version control software.

Table of Contents

Table of tables

Table of Figures

1. Introduction

1.1 Purpose of this Document

In order to complete my Master's degree in Informatics Engineering at the Faculty of Sciences and Technology of the University of Coimbra in 2015/2016, I was required to complete a one year long internship, so, from an extensive list of options, I ended up choosing Konkrets, Lda whose proposal was to devise and develop an e-Learning platform, entitled WebLink that would allow the company to expand their instructorship business by reducing the costs and mitigating the limitations that face-to-face teaching entails. So, for the duration of a school year, spanning from September 2015 to August 2016 I have been working at the company in order to complete the assignment at hand.

Therefore, the purpose of this document is to report on the activities carried out during said internship and serve as documentation for every step taken towards the goal of providing the client with the tool he needs. Every action, choice or decision is detailed in this report from the studying of the existing tools and platforms to the overview and benchmarking of the final product.

1.2 Document Outline

Each section of this document represents a step that was necessary in the devising and the development of WebLink, and should provide the reader with an insight on the decision making process throughout every phase of the project.

Section 2: This section provides the reader with some context as to why, and for whom, the platform was built and it is divided in four topics. The first one refers to the company itself and it should allow the reader not only to understand what their service consist of, but also to perceive what values and guidelines it tries to achieve. Subsequently there is an overview of their educational business branch that WebLink is aiming to enhance.

Section 3: This section is divided in three topics and describes the overview of the platform itself, allowing the reader to better understand the developed system. The first part

attempts to clarify what WebLink is supposed to achieve while pointing out some functionalities that it will implement without providing much detail for each one. In the second topic lies a description of the project scope, explaining what will and won't be implemented in a detailed manner, followed by the constraints imposed by either the client, the internship or technical limitations that might have been identified from the start.

Section 4: This section consists of a detailed explanation as to why I chose the rational unified process for this project, followed by a series of Gantt Maps predicting the timeframes to be used for each task. In this section the reader can also read about the infrastructure used in the development and software engineering techniques that were used. There is also a brief description of the software version control used and the different environments taken into account. Finally, there is a description of the delivery pipeline that was created to get new versions into production.

Section 5: This section is divided in five topics and it consists of a brief description about the e-Learning state of the art. The first topic contains a description of the history behind such a market, followed by an explanation of the most important methods of teaching used in the XXI century. Subsequently lies a summarization of the top most well-known e-Learning platforms and some of their most interesting features complemented by a table comparing the functionalities provided by them.

Section 6: This section provides a description on the requirements inherent to this project and also describes the actors that will use the platform once it is ready. It also describes the various non-functional requirements and explains the how they were prioritized. To find a detailed description of the functional requirements the reader will have to look into annex 1.

Section 7: This section provides a description of the architecture that was implemented for the platform and a description of the technologies that are being taken in account. In this section it will also be possible to understand the depth of the project and how all parts come together to form WebLink.

Section 8: This section provides the risk management and assessment that was done during the development of WebLink in order to identify and mitigate possible issues that could have arose during the project.

Section 9: This section provides further details on the tools and frameworks used during the development of WebLink. It focuses on providing an overview of the many techniques used in the development process.

Section 10: This section provides an overall description of how the end product looks like and how a user can navigate the platform. It focuses on the elements that are common to the whole platform instead of explaining each and every view. This section also displays the classroom view.

Section 11: This section provides a description regarding the clearance of both functional and nonfunctional requirements alongside with a description of the quality assurance. For the non-functional requirements it also presents the benchmarking experiments that have been developed to validate them.

Section 12: This section details some features that were not inserted in the scope of the project but have been considered interesting by the intern and should be considered for future development.

Section 13: This section concludes the document describing what was learned during the internship and what difficulties were experienced during the process. This section also describes the current state of the platform and what lies ahead for the intern until the internship is officially over.

Section 14: This section provides a reference listing for all materials that have been used in the development of this report.

1.3 Acronyms and Abbreviations

B-Learning	**Blended Learning**
RUP	Rational Unified Process
ERD	Entity Relationship Diagram
CBT	Cognitive Behavioral Therapy
CD	Compact Disk
TV	Television

ICT	Information and communication Technology
IEFP	Instituto do emprego e formação profissional
CCP	Certificado de competências pedagógicas
OpenACS	Open Architecture Community System
MIT	Massachusetts Institute of Technology
VLE	Virtual Learning Environment
LMS	Learning Management System
UHI	University of Highlands and Islands
FIT	Fraunhofer Institute of Technology
BSCW	Basic Support and Cooperative Work
FR	Functional Requirement
NFR	Non-Functional Requirement
WebRTC	Web Real Time Communications
W3C	World Wide Web Consortium
RTS	Real Time Streaming
VoIP	Voice over IP
API	Application Program Interface
ASR	Architecturally Significant Requirements
WCAG	Web Content Accessibility Guidelines
SR	Security Requirement
CSRF	Cross site request forgery
MVC	Model-View-Controller
JSON	JavaScript Object Notation
SVN	Subversion
IBM	International Business Machines
SSH	Secure Shell
WAR	Web Archive
REST	Representational State Transfer
POM	Project Object Model
HDFS	Hadoop File System

FS	File System
SQL	Structure Query Language
ELK	ElasticSearch, Logstash Kibana
SR	Security Requirement
UR	Usability Requirement
Dbm	Database manager
HQL	Hibernate Query Language
HTML	Hypertext markup language
CSS	Cascading Stylesheets
ETL	Extract, Transform and Load
DB	Database
XML	Extensible Markup Language
GNU	Gnu's not Unix
POJO	Plain Old Java Object
HTTP	Hypertext Transfer Protocol
URL	Uniform Resource Locator
MITM	Man-In-The-Middle
XSS	Cross-site-Scripting
R	Risk
IP	Internet Protocol
JPG	Joint photographic Experts Group
PDF	Portable Document Format
TLS	Transport Layer Security
MD5	Message Digest 5

***Table 1** -Acronyms and Abbreviations*

2. Context

2.1 The Company

Konkrets, Lda, the client for this project, is a company created in 2009 headquartered in Lousã, Portugal and whose vision is not only to be an integrated consultant but also a leader in the creation of value on the market it operates in. In order to accomplish that, they identify, evaluate, define and implement strategic solutions that can generate value for their clients. Alongside with consulting the company also has an educational business branch, which is the one the platform described throughout this document aims to improve (This branch is further detailed later on).

Although the company is still headquartered in Lousã it has been under a great process of expansion, having inaugurated two more offices since its creation, once in 2013 in Lousã and finally in Coimbra, in 2015.

2.2 The Business

Konkrets' educational business branch consists of a traditional teaching method (Trainees and instructors need to share the same space and time) in which potential clients can subscribe and attend courses for a fee in a face-to-face basis.

Even though this business model of teaching has been proven to be effective, it lacks the necessary scalability to expand both overseas and continentally due to the constant need for staff and infrastructure upgrading, so, in order to grow as a business, management decided there was the need to create a platform providing an online service for teaching, and that is why this project was created.

3. Project Overview

3.1. Description

The client required a platform that would allow the company to expand its educational business branch from a local service to a continental one with the objective of eventually being able to provide a service on a worldwide scale increasing income due to the higher number of clients. Alongside with the increase of students, the existence of such platform would result in the reduction of costs inherent to traditional teaching and consequently the overall profit of the entire teaching branch. The platform would reduce its costs by minimizing or even removing expenditures with:

- Personnel
- Infrastructure
- Physical materials

The number of e-learning platforms in the market is overwhelming however, I and the company couldn't find one that would suit every need of the client and even though it was possible to extend an open-source solution it was decided that it would be better to create a brand new platform modeled towards the needs of the company. Therefore, the objective of the internship became the devising and development of a secure and scalable e-learning platform named WebLink.

WebLink provides a remote and interactive service in which anyone could potentially be a client regardless of age or educational background. It allows instructors to teach both synchronously and non-synchronously (definition is given later on this document). The platform implements methods that support both a remote and blended learning due to legal issues.

Regardless of the learning method and user background WebLink provides the necessary tools for the creation and management of everything needed in an online course through intuitive means. Anyone who intends to take or teach a course can easily register to the platform and subscribe to one available upon online payment and if the user has enough permissions it can even create and manage its own course. Core functionalities such as the

creation of quizzes, classes, material uploading and visualization are all present in the platform.

WebLink also implements social functionalities both within and to the outside of the platform. Once a course is completed users can share their feats with their peers using social networks while further advertising the system that they are using. Also WebLink makes it possible for its users to connect with each other using a simple friend system, allowing them to talk between themselves and even exchange courses has a gift. A global instant message chat is also present in the platform allowing users to get to know new people and exchange experiences with other people.

When a course coordinator creates a course, he is prompted to make a choice between blended and a remote learning paradigm. Blended learning refers to a course that is not completely online and requires the presence of the student in the physical premises of the company for some of the classes due to a myriad of reasons. Remote, on the other hand, is a type of learning in which everything from registration to course certification can be done from afar. Within remote courses one can also differentiate between Synchronous and Asynchronous learning. While in the first, knowledge is transferred directly from an instructor to his students through conference calls, in the other, it is available using pre-recorded media content made available at course creation.

The platform also integrates with a previously built system available at the company named Sik Webforma, which is a financial management service created to keep track of everything going on at the company from financial transactions to worker management. Therefore, all platforms in production need to send information to Sik. It is also worth noting that the platform is able to handle a high number of clients at all times while preserving a usable performance.

3.2. Scope

All the Functional requirements are described in the requirements section of this document and every one with a priority *"Must have"* shall be implement for the project to be considered completed, *"Should have"* requirements shall also have a minimum implementation check of 75% while *"Could have"* requirements shall only be implemented if there is enough time left in the internship and regardless of their completion status the project shall be successful if the previous criteria are met. Finally *"Won't have"* requirements won't even be taken into account during the span of the internship and shall be implemented only in the future if the company decides to do so.

The description for all non-functional requirements is presented in a detailed manner on the requirements section and the following have been considered necessary by the company for the project WebLink:

- Security
- Scalability
- Usability
- Interoperability
- Accountability

Scalability is to be regarded as being in a prototype stage as it is untested in real cases and therefore it is considered unsafe in a production environment. While no other non-functional requirements will be taken into account in the future, functional requirements can be changed by scheduling meetings between the development team and the client due to the flexible nature of the process. However, the development team must agree with the changes considering the deadline imposed by the university, meaning that if the change requires more development time it is likely to be dropped.

3.3. Constraints

- Technical constraints:
  - The client specifically requested the platform to be built using a Tomcat/Java application server so the company's team can keep on managing WebLink even after it is completed and new features can be implemented in the future if needed.
  - The Client also required for the main database to be relational, and more specifically to use MySQL since it is the one the company engineering team has more practice using.
  - The platform shall run on machines using CentOS as an operative system in the company premises.

- Resource Constraints:
  - Effort on development: 16h a week during the first semester
  - Effort on development: 40h a week during the second semester
  - Final Deadline: July, 2016
  - Team size: 1

4. Methodologies

4.1 Life Cycle

I have decided to use a RUP [1] as the process for the development of WebLink since the project requires a semi-agile approach due to the large number of functional requirements and their volatile nature.

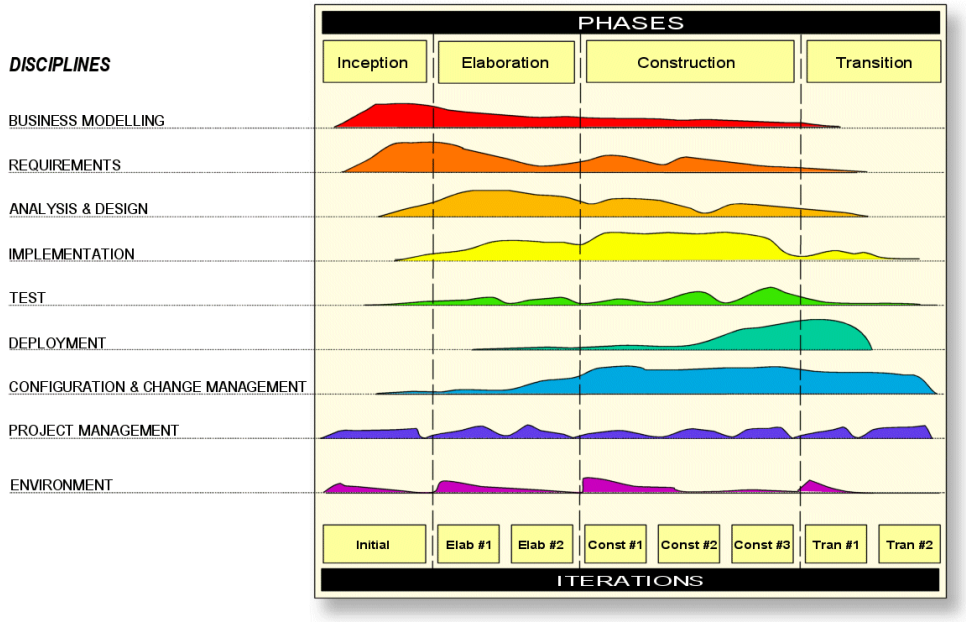


Figure 1** – RUP **[3]

Also, to help keeping track on the tasks being developed I have used Trello which is an online platform used in project management that allows users to create cards based on the *Kanban Board* from the *Kanban* methodology. A *Kanban Board* consists of a board divided in sections, each representing a phase in the RUP development process. In these sections cards are used to provide a visual aid, each card representing a task that needs to be developed. These cards are displayed in a timeline which prevent the development team from straying from the pre-defined implementation order. Due to the visual representation it is also

[1] Rational Unified Process

possible to better perceive the current state of the project and calculate more accurately how late each task actually is.

A RUP life cycle consists of four phases, each somewhat similar to a "mini - waterfall" since all phases of the waterfall are completed for a small part of the system and then iterated until that phase is production ready. Also at the start of each phase the previous ones are reviewed and new changes added. The four phases to the RUP process life cycle are as follows:

1. **Inception:** The primary objective for this phase is defining the outlines of the project, the scope, the stakeholders, the business models and describe the requirements on a high level and then iterate until production ready. In this phase no actual development is done, and is only a pre-game phase.
2. **Elaboration:** Consists of devising a development plan for the overall project, gathering and documenting the requirements and defining the architecture to be used in the implementation of the system. During the devising of the architecture a prototype should be built for the client or other stakeholders to provide feedback on the work being developed. Upon client feedback the prototype is improved and these steps iterated until usability is achieved, technical risks are mitigated and the prototype is deemed as ready. Although RUP has a semi agile approach, all the documentation is done formally, using use cases and UML to represent both the requirements and the architecture. When this phase is over any changes that might have come up regarding the inception phase are then implemented.
3. **Construction phase:** In this phase most of the functionalities and coding will be done. This phase is fairly technical in nature and it can be summarized the implementation of the architecture that was devised in the previous phase. Also as in the previous phases after an initial implementation there is a meeting with the client or any other stakeholders in order to assess if the project meets the all requirements, both functional and non-functional. Also by the end of this phase the previous ones are reviewed and changes implemented. It is also in this phase that the whole infrastructure is prepared for the deployment in the next phase.
4. **Transition phase:** This is the final phase in the life cycle in RUP, and consists of deploying the project and monitoring real life usage. Testing and benchmarking should

be done to eliminate as many defects as possible in the time frame and do any changes to the infrastructure. When this phase is over there is a meeting to assess the project as a whole and make sure everything is working and documented as intended. When this phase is over the project is deemed as complete and successful.

Rational unified process is a process created to provide teams with the necessary tools to develop complex projects while accommodating changes due to its incremental and iterative nature. WebLink is a big project therefore it might be subject to changes throughout the process, either because the client might change his mind about the required functionalities or because some requirements were not clear in early phases.

Also WebLink has a large number of functional requirements that need to be implemented in a usable way. For the required usability to be achieved iterative prototyping is required because what is obvious for the development team might not be to other users when they navigate the platform. So such a platform could not be built with a non-agile methodology because it would not accommodate the possibility of changes arising throughout the development.

Since the client asked for the project to be developed in Java it might also be worth mentioning that RUP projects are usually documented using component based architectures in which object oriented programming excels at during the implementation phase. Also risks and defects inherent to projects are usually identified and addressed during the early phases of the project due to its iterative nature.

4.2 Process Tools

4.2.1 Jenkins

Jenkins is a Java-written tool that provides continuous integration services for software development. I was created by Oracle and released in February, 2011 as a server-based service that should be running in non-production machines. It can execute Apache ant or Apache maven based projects as well as arbitrary shell scripts and Windows batch commands. Jenkins makes use of its slave nodes running on production servers to provide deployment capabilities on clustered environments.

Figure 2 *- Jenkins Logo*

Jenkins can be connected to a multitude of software version control software such as Github or Apache SVN[2] using Web hooks (this is explained in the GitHub section). Whenever a project is committed or rebased Jenkins is notified, building and testing it. If those tests are successful Jenkins uses his slave nodes to deploy the new version of software in the production environment. Jenkins also has a built in plugin system which allows anyone to create new plugins for it, so that it can be integrated with new software regularly. An example of Jenkins interface can be seen in the following image:

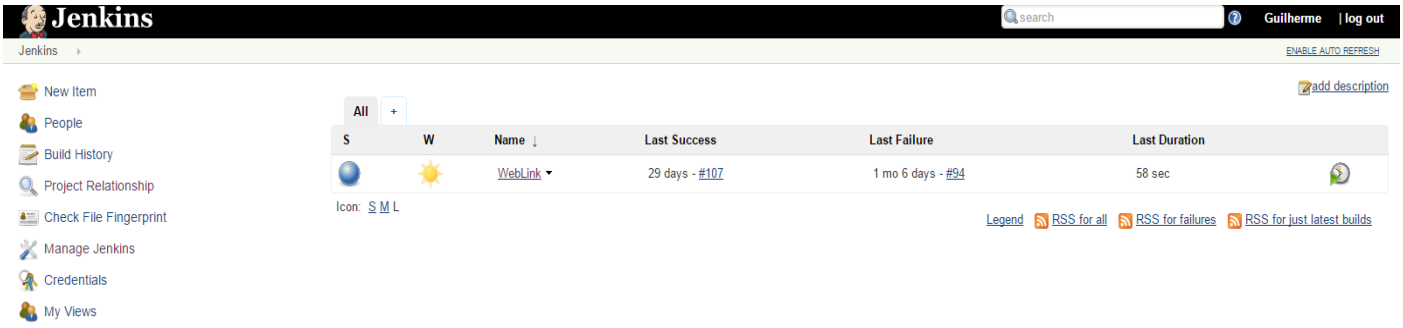


Figure 3 *- Jenkins Interface Example*

[2] Apache Sub versioning

4.2.2 Docker

The method in which the kernel of an operative system allows the existence of multiple isolated user-space instances instead of one is a virtualization technique named Operative-system-level virtualization. Each of those instances is named software container and on a Unix-like operative system this technology can be seen as an advanced implementation of the standard *chroot* mechanism **[7]**.

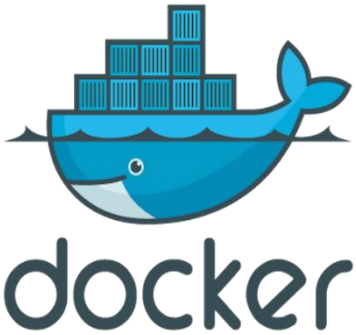


Figure 4 *- Docker logo*

This technology can be used to run both on bare metal or inside a virtual machine using a UNIX based operative system such as CentOS (the one used in the internship). These containers may run a different operative systems with a set start state and therefore it will behave the same independently of where it is running. It is worth noting that this technology has little to no overhead because programs in virtual partitions use the operating system's call interface and do not need to be subjected to emulation or be run in an intermediate virtual machine.

Should the reader be interested where did I get the data to support this claim, it is available at **[8]** which is a benchmark study done by IBM[3] in 2014 using Linux and Docker containers both on Virtual machines and on bare metal.

[3] International Business Machines

4.2.3 GitHub

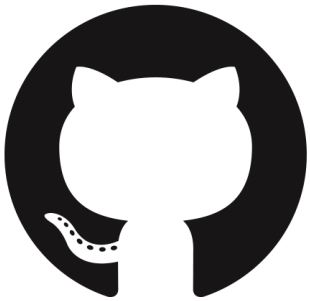


Figure 5 *- GitHub Logo*

Github is a web-based Git repository hosting service that offers distributed revision control, source code management and version control services. It is widely used by development teams all over the world to share and manage code versions and it is possible to integrate it with other software using the web hook system, in which it is possible to setup Github to send an HTTPS request to another application whenever some action occurs.

Git was built by Linus Torvalds (principal developer of the Linux kernel) to replace Apache SVN[4] which he disliked due the unnecessary complexity and is now the most used software version control software in the market. Git can be setup to integrate with most IDEs[5] which allow for the version control to be done with next to no effort while developing.

4.3 Software Engineering

4.3.1 Software Control

Even though I am the sole developer of WebLink, my supervisor Serge Nunes should have access to any code developed both FrontEnd and BackEnd so I considered to use either Apache SVN or GitHub as my repository and version control software. I ended up choosing GitHub since it is easier to integrate with continuous deployment tools and I have worked with it in the past.

Github is a distributed software control tool that can be integrated in most Java IDEs, and allows the user to push his project to an online repository that can be either public or private. During the development of this process I will be using a staging branch containing the features that are already tested and ready to be deployed, and a development branch that

[4] Apache Sub versioning
[5] Integrated development environment

contains incomplete or untested features. When the development branch is ready and tested it is merged with the staging branch.

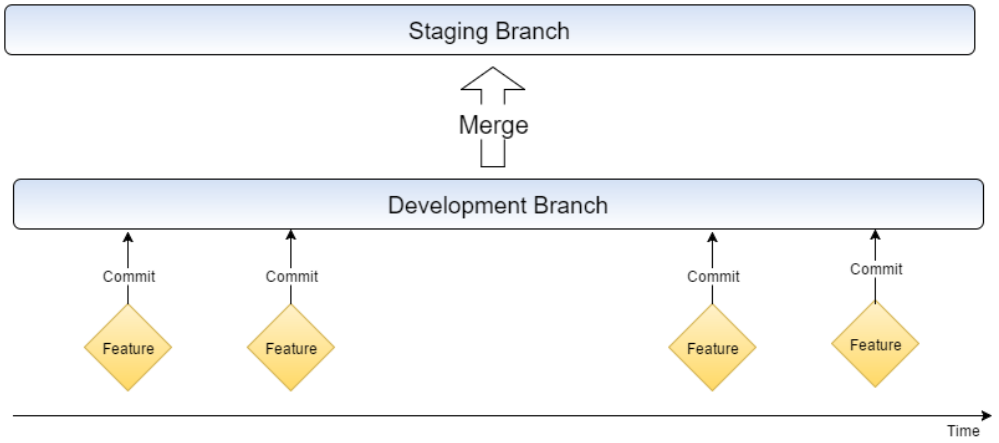


***Figure 6** - Github branching*

4.3.3 Continuous Delivery Pipeline

For the development of WebLink I have prepared a continuous delivery pipeline that allows the platform to be deployed into production upon the click of a button while minimizing the risks of failure and mitigating the consequences in case such failure occurs. In order to achieve this I have used both continuous integration and continuous deployment techniques and a myriad of tools that will be explained in this section.

If the application is as successful as the company expects, the platform must be prepared to increase the amount of available resources. This can be achievement by preparing the platform to scale horizontally[6]. Due to the statelessness of the REST[7] requests implemented, the load between Tomcat Server nodes can simply be balanced using HProxy.

Whenever a merge or rebase between the Github development branch and the staging branch occurs it sends a web hook signal to Jenkins over HTTPS. When Jenkins receives such signal it fetches the new project version and compiles it using the pom[8] file from my maven project and runs unit tests that make sure the core elements of the project are behaving as expect. If the tests return the expected results Jenkins builds the .war file and

[6] Increase the number of machines, to increase resources
[7] Representational State Transfer
[8] Project Object Model

proceeds to deployment automatically, otherwise the new version is dropped and Jenkins sends a notification with the details as to why the build failed. This is considered a continuous integration technique used to make sure the build provided is stable and behaving as expected to help prevent failures due to faulty merges and can be visualized in the following image.

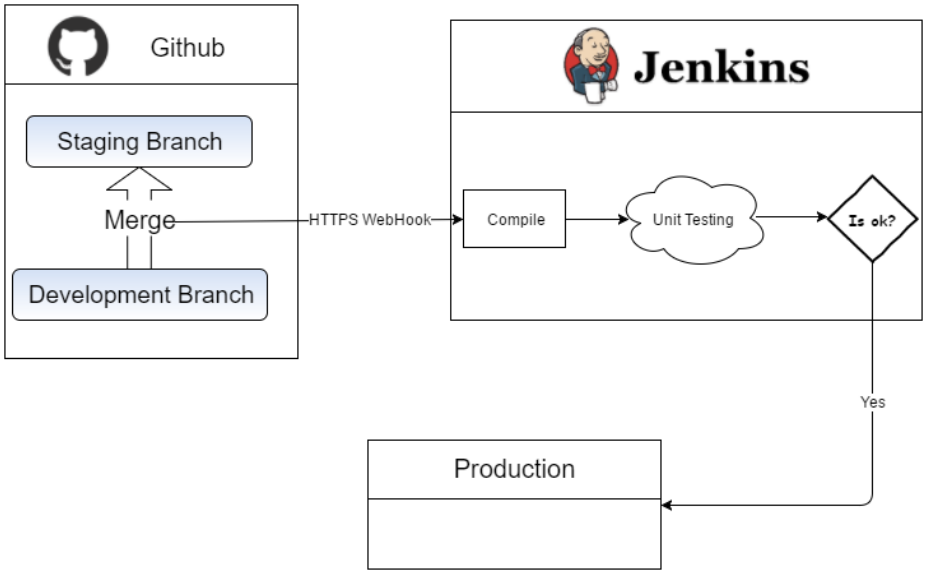


Figure 7- *Jenkins continuous integration*

After the build is completed and successfully tested, the new version will be deployed for production using two techniques:

- **Immutable Server:** Immutable server deployment is a technique that has a good synergy when using Docker containers. In this technique a server is never changed or updated, in fact it is completely removed and a new one is created from scratch using a base image that the developer knows to be working and what the base state is supposed to be[9]. This kind of approach is highly recommended since it prevents the hard maintenance required by a Snowflake Server in which you have to ensure the operating system and any other dependent software is patched to keep it up to date and configuration changes are regularly needed to tweak the environment so that it runs efficiently and communicates properly with other systems.

[9] These servers are known as Phoenix Servers

- **A/B Testing:** This technique was not developed during the internship since it is only useful for platforms already in production, however a description serving as proof of concept was devised in case the company decides to implement it later on. A/B Testing consists of a technique in which two server nodes or clusters are running different software versions (A and B). *A* being the stable version and *B* the new version to be tested. With the help of the load balancer it is possible to balance the load unequally for example, 90% for server *A* and 10% for server *B*. This way it is possible failures will be mitigated when uploading new versions, making sure that only a small percentage of requests actually use the new version, and making studies to know which version is actually better. Usually version *B* percentage of requests will gradually increase until it fully replaces version A. This is used by many major companies such as Facebook and Twitter to make sure the new version has the expected reception.

So when a new version is to be deployed, Jenkins will create a new container with the new version and replace the *B* version with. If the *B* version proves to be working as intended and the developer wants to make it the stable version, a script that will replace server *A* with the new version, should be used so that all requests go to the stable version.

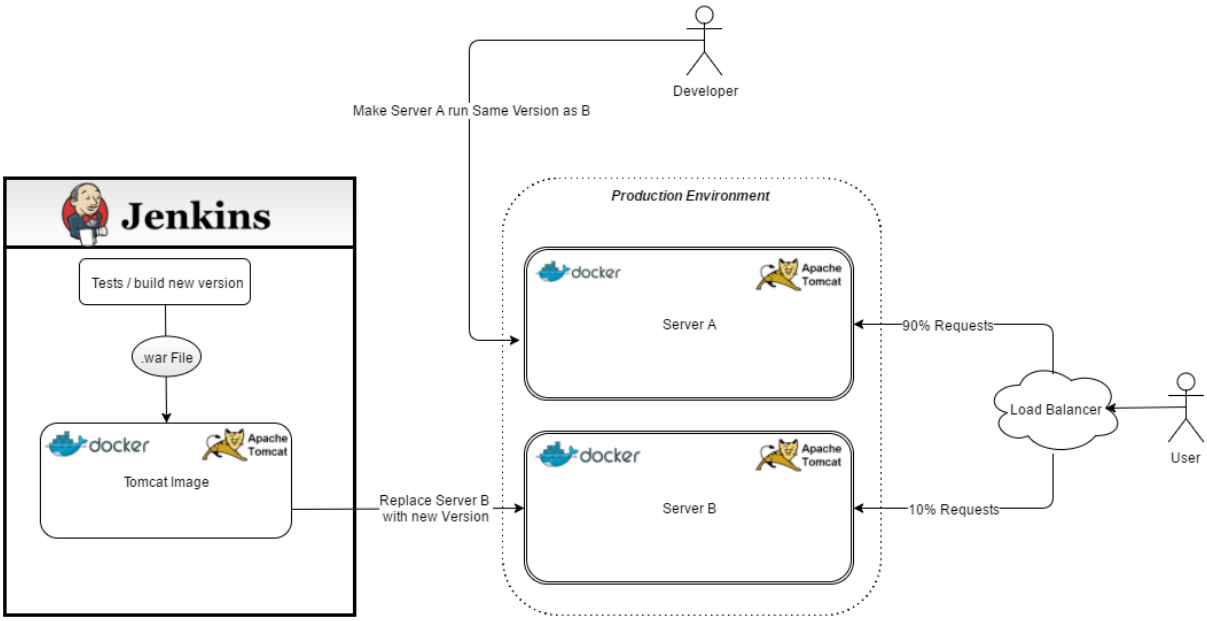


Figure 8 - *Continuous Delivery*

4.4 High-level Plan and Milestones preview

4.4.1 First Semester

The following Gantt describes the tasks developed in the first semester and in which timeframe they were developed.

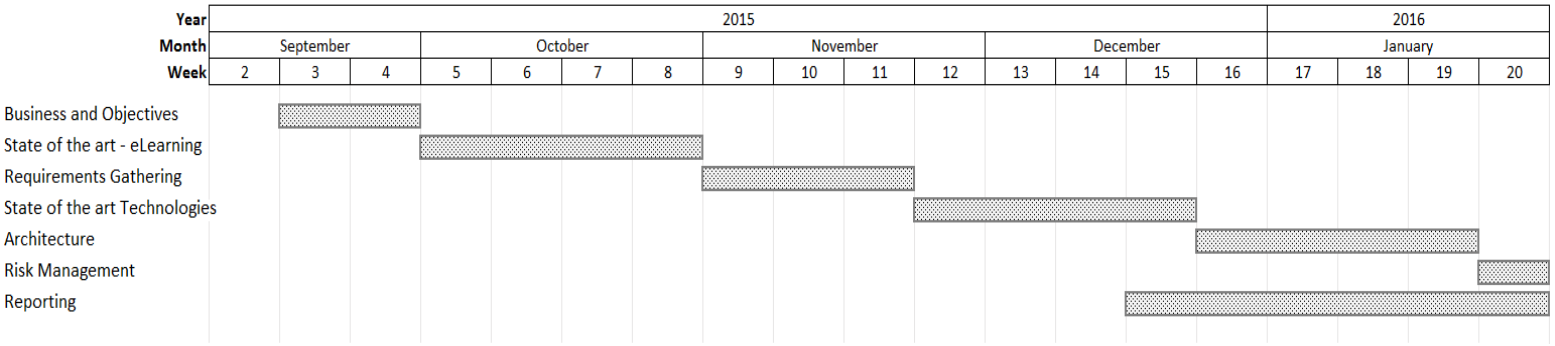


Figure 9 – *Gantt: First Semester Preview*

This Gantt represents how the tasks were distributed during the first semester. Since this was only created by the end of the first semester the timeframes represent how the project did happen instead of a prediction. The first phase of RUP was completed first by week 4, where all the business objectives were defined.

By week phase 2 of the RUP started and a thorough state of the art study regarding the available eLearning technologies took place followed by a meeting to assess the functional requirements the client expected in the platform. In the meeting it was decided a new platform was to be created instead of using any other available. Therefore there was a new state of the art study regarding the technologies that would be used in development such as the streaming technologies referred later on this document. Starting in week 12 architecture and risk assessment were devised while working on the report.

4.4.2 Second Semester

The following Gantts describe the tasks and timeframes of development prediction for the second semester followed by another image representing what timeframes actually happened.

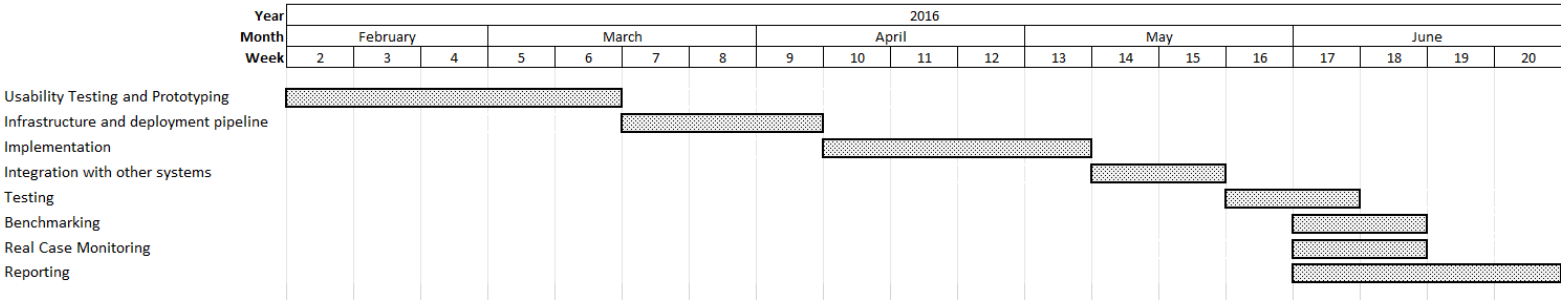


Figure 11 – *Gantt: Second Semester Prediction*

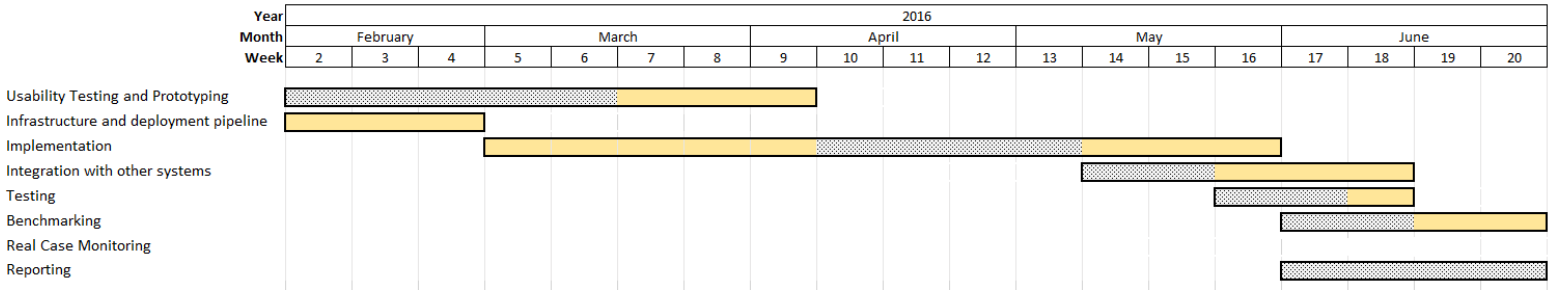


Figure 10 – *Gantt: Second Semester Reality*

When the first semester was concluded and the figure 11 Gantt was developed, it was expected that the first task of the second semester (Usability and prototyping) would be concluded by the second week of March. However, it was decided that it would speed the development further if some functionalities were developed in the meantime and therefore infrastructure setup, the delivery pipeline and functionalities were done while preparing high resolution prototypes. By the end of the first week of April Usability and prototyping were closed and phase 2 of RUP had come to an end.

The lack of experience designing frontend features delayed the project by around 3 weeks as it is visible in figure 12 which led to a serious delay in all subsequent tasks. As a result and as described in the requirement clearance section later on this document some requirements were not completed as expected. After the delivery of this report these functionalities and real case monitoring will still be done as expected.

5. e-Learning State of the art

5.1 e-Learning history

Remote education has been around for quite some time now and it dates back to, at least 1840 when Isaac Pitman taught his pupils by mailing tasks that they would send back upon completion, a few years later, in 1920 a new machine, somewhat similar to a typewriter, was built by Sidney Pressey with teaching in mind. It was composed of a simple window where a question would pop up and four buttons each containing a possible answer to the problem presented (Only one was correct). Upon choosing an option, the machine would register it in a piece of paper in the back, and present the testee with another question. This system, known today as multiple choice question, became a cornerstone of most online quizzes and surveys and also an important part of the e-learning industry.

E-Learning or electronic learning is an appellation that originally came up at a CBT[10] seminar held in Los Angeles, October, 1999. This designation comes from expressions such as *online learning* or *virtual learning* that were used to describe a way of learning using the new technologies. **[4]**

Due to the rivalry existing between many companies and constant advances in technology, new paradigms kept turning up in order to make learning easier leading to new systems with personalized and interactive materials like *CD-ROMs, TVs*, etc. and finally with the appearance of web 2.0 many companies invested in platforms for remote learning for profit, leading to a saturated market in which only the really innovative tools would be deemed to succeed.

[10] Cognitive Behavioral Therapy

5.2 Methods of learning

A learning method is referred to as the way in which learners or pupils receive instructions, knowledge or information from an instructor. There are several learning methods that have been described in literature, however the most relevant are the ones described by Dr. Dan Remenyi in 2007 **[2]**:

- **Traditional learning:** This refers to face to face sessions in which the teacher delivers course or subject material to students in the same place and in the same time. The teacher focuses on providing the learning information and the students' focus is to acquire it. Within a traditional classroom setting, knowledge delivery is limited to the available resources present at the school or classroom. Assessments depend on study notes given to students by the teacher (p.350).

- **E-Learning:** Refers to the use of ICTs[11] to transform and support the learning process ubiquitously (p.350). e-Learning can then be divided in a multitude of sub-methods, however, we are going to focus on the ones that require two types of actors: teachers and students which can be divided in two distinct categories:

    - Synchronous learning: Refers to a type of e-Learning in which the students and the teacher are required to be online at the same time even though they are not in the same space. Students participate in virtual classrooms with their teacher where the topics of the course are discussed. Student evaluation usually takes into account attendance rates and test grades.

    - Asynchronous e-learning: Refers to a type of e-Learning in which the students and the teacher are not required to be both online at the same time and in the same space. In this method the students learn from videos and materials that they can watch whenever they want. These materials are uploaded by the

[11] Information and Communications Technology

teacher and usually at the end of the course the students are evaluated based on what they learned through self-corrected testing.

- **Blended-Learning:** Also known as b-Learning makes use of a combination of both the previously referred methods, combining face-to-face meeting along with virtual and remote learning. This type of learning is usually used due to external factors such as laws, rules, employment policies, etc. For example, in Portugal the Pedagogical aptitude training course - IEFP certified trainer (CCP[12]), has two modules which have mandatory attendance due to external factors.

5.3 Popular e-Learning platforms

5.3.1 Moodle

Moodle is an Australian company founded in the early 90s that provides an e-Learning service with millions of registered users. The naming stands for *modular object-oriented dynamic learning environment* due to its flexibility. It is supposed to provide both instructors and trainees with a robust, secure and integrated system to create personalized environments and it currently harbors around 30 developers and gets funding from around 60 companies worldwide.

***Figure 12** - Moodle Logo*

Also Moodle is a free and open-source, cross-platform project under the GNU general public license, meaning that anyone can extend, adapt or modify Moodle for both commercial and noncommercial ends without any licensing fee. This PHP written platform implements a plugin system and an API that make the system quite flexible however its interface is often found confusing and cluttered due to the sheer number of functionalities. It is also worth to

[12] Certificado de competências pedagógicas

notice that it is used by the Open University in the United States and that it was the number two regarding the market share in 2013 only surpassed by Blackboard Learn.

5.3.2 Blackboard Learn

Figure 13 - *Blackboard Learn Logo*

Blackboard Learn previously known as the blackboard learning management system is a company founded in 1997 that provides course and content management systems. Right now blackboard is one of the most successful e-Learning platforms due to their unique and flexible approach to learning. Blackboard Learn originated from the merge of two previously existing e-Learning companies (CourseInfo LLC and Educause IMS) and faced quite a few legal issues due to faulty patent rights claims.

It consists of a service provided by Blackboard, Inc. and unlike Moodle it is not an open source solution even though it allows local hosting of services in a hosted system. Their business model consists of offering consulting to the companies using the application and renting local servers to run Blackboard servers in case a company doesn't want or can´t afford the equipment needed to host it. Although it only allows for an asynchronous way of learning it provides quite a bunch of innovative solutions and features.

Although the system is still considered one of the top e-Learning platforms in the market, people using the system have been complaining for years now due to the reliability of the system and due to a huge number of glitches and bugs that haven't been fixed yet, one of which causes the system to go offline for quite a long time compromising the system availability even though they advertise their hosted system to have 99.9% availability. Due to the previously described issues Blackboard has been losing a big number of customers to others platforms now since new systems have been built with mobile expansion in mind unlike Blackboard.

5.3.3. D2L

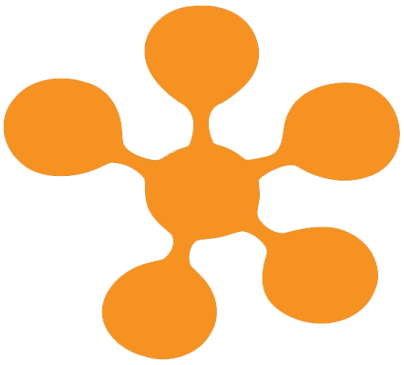


***Figure 14** - D2L Logo*

D2L previously known as Desire2Learn is a company created and headquartered in Ontario, Canada. It was originally founded in 1999 and has more than 800 developers working on their United States of America office. This company was one of the companies that was in the legal dispute with Blackboard Learn due to faulty patent claims that ended with both companies agreeing to license each other's software. The company claims to have more than 15 million users and was ranked third in 2013 with 11% market share right before Blackboard Learn and Moodle.

Unlike the previously described companies D2L only allows learning using their own website hosted in their own servers. The service is not open source and it is browser based, which makes it cross-platform and provides Synchronous way of learning and requires both instructors and students to subscribe to the website.

5.3.4 Dokeos

***Figure 15** - Dokeos Logo*

Dokeos is an e-Learning platform founded in 1999, built on top of an older, French platform called Claroline by users who desired an improved system since the previous one was becoming deprecated feature wise. Although Dokeos offers a lot of functionalities alongside with their open-source service it only allows for a free trial spanning for a month, then, their business model consists of charging the client for the extra modules/functionalities and a monthly fee for the platform itself.

Even though Dokeos might not be the most flexible platform in the market due its expensive payment model it is gaining popularity due to their slick interfaces and sheer number of functionalities. This platform is also web based and has been optimized to run on

smartphones/ Tablets as well as on your computer screen. It also offers some unique features that I haven't found in any other systems like their integrated web TV, gamification system and Dokeos Author that allows any user to develop further content for the platform allowing anyone to create either private or public sections.

5.3.5 Sakai

Sakai is a service-oriented java-based open source e-Learning platform created in 2004 by universities in the United States. Later on, quite a few companies joined Sakai as partners and developed tools based on their own products. It is mostly free, however extra tools or add-ons could be paid since any company can produce and sell them at will. Sakai is based on an asynchronous learning model and provides quite a few perks alongside with a nice and clean interface.

Figure 16 *- Sakai Logo*

Like Dokeos Sakai is quite a new platform and its popularity is still on the rise. It is estimated that over 300 e-Learning systems use Sakai. It is already in its tenth version and therefore the amount of coding produced can present a barrier to small teams that wish to create a new platform in top of it and already supports nineteen different languages.

5.3.6 ATutor

ATutor is an open source system supporting learning and content management and specifically considering accessibility and adaptability issues. It was first released in 2002 after two studies conducted that evaluated the accessibility of learning platforms to people with disabilities. Several features are planned for the near future, including a barrier free authoring tool and a streaming media server.

ATutor business model consists of free hostage until 3 courses are instanced following monthly payments increasing alongside with the number of courses. As most platforms in the market it is mostly focused on an asynchronous learning model and offers no support whatsoever for synchronous development. ATutor presents accessibility features that provide tools for their visually impaired / disabled users which is something unique that wasn't available in none of the previously documented platforms. It was the first e-Learning platform to actually implement all the accessibility specifications of the WCAG[13].

Figure 17 - *Atutor Logo*

As well as Dokeos, ATutor also has an authoring tool however, ATutor also allows the visually impaired to create content for themselves by listening to the entire interfaces with the help of a screen reader provided by them while the system can be accessed without the usage of a mouse. ATutor also presents interoperability with a large range of system such as mobile phones, personal data assistants (PDAs), text based browsers, etc. As L2D ATutor was also in legal disputes over patent claims with blackboard which lasts to this day.

5.3.7 EFront

EFront is an open source project that was developed in 2005, however, it was re-written from scratch, making essential changes to the core structure of the project and released under an open source license in 2007. In 2012 it got an award from eLearning! Magazine for being the best open source solution in the market.

Figure 18 - *EFront Logo*

There are two versions for EFront, a simpler one for free and a more complete one, which requires really expensive payments. It is one of the few open source tools in the market that actually supports both synchronous and asynchronous learning methods if you are using the paid version of the software. The EFront system can be hosted in either GNU/Linux,

[13] Web Content Accessibility Guidelines

Microsoft Windows or any other operative system that supports PHP and MySQL and it is said to have more than 50.000 users across 120 countries. The system is entirely web based and it is advertised as a robust, scalable, reliable and user friendly product.

5.3.8 Coursera

Coursera is an e-Learning tool released in 2012 created by Andrew Ng and Daphne Koller from Stanford University and whose business plan is to get Universities to teach courses in it for a fee. Since the courses hosted usually have university funding it is possible for the platform to sell some of their courses and provide some for free. Due to this system the platform is massive usually having thousands of people at a time taking classes even though they only support asynchronous teaching.

Figure 19 - *Coursera*

When Coursera was initially released it provided a free service however, lately, a fee is required for a certificate useful for employment to be issued. The platform has more than 13 million users and as of 2015 and by 2013 it was already worth more than 100 million dollars. They have 1000 different courses provided by 119 companies and their user base is in 190 countries.

Coursera is probably the most recent platform described in this report, and thanks to the fact that when they started developing the mobile market was already a possibility, they focused their system on that market which differentiates them from most other companies that have focused on slick web interfaces.

5.4 Open source platform comparison

In this section I will present a table showing the features implemented in each of the open source platforms.

	Payments	*Storage*	*Quiz/test Creation*	*Asynchronous Learning*	*Synchronous Learning*	*Free?*	*Gamification*	*Multi-Language Support*	*Calendar*	*Forum*
Moodle		X	X	X		X		X	X	X
Dokeos	X	X	X	X	X	1000$	X	X		
ATutor		X	X	X		200$		X		X
Sakai		X	X	X				X	X	
EFront	X	X	X		X	3000$	X		X	X
.LRN	X	X	X	X		X			X	
D2L		X	X		X			X	X	X
Coursera	X	X	X	X			X	X	X	

Table 2 - Open Source platform Comparison

5.5 Most interesting technologies for WebLink

Upon initial contact with the client it became obvious that the following learning methods described in 2.1 would be required in the platform:

- **Synchronous Learning –** The platform should provide support for a synchronous way a learning. The instructor of a Course shall have the possibility of both giving classes and joining conference calls. In a class the instructor responsible for the module will be the only one who can share its webcam and talk while the other users can only communicate using text chat, also the instructor may choose to stream either what is captured by its webcam or its screen to make presentations for example. Classes have a 1 to N topology regarding streaming because it was decided that any other way would cause more entropy to the room than necessary. On the other hand conference

calls support an N to N topology where every user shares both screen and sound. Conference calls shouldn't be used to present new subjects but rather to present questions to the instructor or for students to get together and solve exercises simultaneously.

- **Asynchronous Learning –** While the main aspect of WebLink is the support for asynchronous learning, it should be possible to create courses based on materials and quizzes alone without the need of any instructor to have an active duty. It was decided that supporting this type of learning was crucial because in the future if the platform is to be expanded it is impossible to provide synchronous learning to a massive amount of people. Also, this type of learning only requires a course to be created once and therefore reproduced in the future lowering the man-power and maximizing number of buys / cost to create.
- **Blended Learning -** As referred before some courses cannot support a full online course due to external factors, so it shall be possible in the platform to schedule mandatory classes in the company's premises and later sign in students manually in the platform for that class.

Also, after doing some research on the available e-Learning platforms described in 2.2 some of their technologies / tools / services stood out and are now requirements for the new platform being built. Firstly it is quite important for the platform to allow for online payments because if it gets to an expansion standpoint new employees would be required in order to have the necessary man-power to handle the financial influx, it is also worth noticing that not having this feature would also would limit the expansion to a country level.

Eventually if the platform is successful and the company wishes to expand further it might be necessary to implement a distributed file system technology (either in a cloud storage server, or locally depending on the existent hardware) so that the number of uploads and downloads of materials and other files can have further capacity and performance. Also in any learning method evaluation is an important part, so the ability for easy taking and creating of tests and quizzes with multiple types of questions is a must have alongside with homework and other evaluation support. This is one of the few characteristics all e-learning tools have in common. It would also be important for the platform to offer a multi-language support,

calendar and forums even though these are not likely to be implemented during the duration of the internship.

Accessibility features such as the ones presented by EFront could also prove to add quite a lot of value to such a platform however these kinds of functionalities require some degree of medical knowledge and expertise which I do not possess and the studying of which would not be possible while being able to successfully create the platform from scratch. It would also be nice to have gamification, which has proven to increase the interactivity between platform and clients while providing a sense of competition between the clients resulting in a higher number of buys.

Support for a user-created plugin system was considered but discarded due the already high number of both functional and non-function requirements however, such a feature would grand the platform some autonomy in its own development and creation of content and therefore should be considered for future implementation. Finally, an authoring tool could also be implemented in a future providing and value certainly captivating some new clients if done and advertised properly.

6. Requirements

This section describes the approach taken towards functional and non-functional requirements gathering. Here I will describe the actors needed in the platform and the functional / non-functional requirements of WebLink. These were gathered by meetings with the client in which the fixtures necessary for WebLink to function properly were discussed thoroughly. These requirements have been changing since the start of the project as described previously in the section regarding the rational unified process. When a change need arises, if there is enough time to accommodate it, the use case list is updated.

6.1 System Actors

The permissions in WebLink allow users to access different parts of the platform depending on the permissions each user has associated to its account. Only administrators have the necessary permissions to change the permissions of other users. Each of these permissions provides access to different parts of the platform.

When a user initially registers and confirms its email, it has the basic access which is User permissions, which allows the user to subscribe to website and use any features available to a student in the platform. A list and description of the different permissions associated to users in WebLink follows bellow:

Unregistered User class: Any user who wants to browse the platform even though he doesn't have an account is able to do so with limited permissions. As these users don't actually have the necessary clearance to enter the platform menus yet due to the fact that they still don't own an account, they can only access pages up to the login menu. They can read about the platform, see the courses that are on display in the homepage or fill a registration. When the registration is filled and the email confirmed they become part of the registered user class described below.

Registered User class: Once an unregistered user registers he will become a registered user. This class is automatically attained by completing the registration form and confirming the email they have registered with. With this permission a user can navigate through the application menus, buy and take courses.

Abstract User class: Although this class is not an actual class, it is used in the requirements in order to represent the actions that can be developed by both a registered and an unregistered user.

Instructor class: To attain this permission the user has to be granted it by an administrator. Having this permission on itself won't show any difference on the platform, however the user becomes available to teach courses to which he can be added by the coordinators. Whenever an instructor becomes listed to teach a class the instructor can enter the course they are teaching and add new materials, host classes or create homework and quizzes for the students attending to solve.

Coordinator class: This is the highest class bellow administrator and to attain it a user has to be promoted by an administrator. With this permission it is possible for a user to create, manage and delete actions, courses or modules. The coordinator also has the job to associate instructors to each action and make them visible when they are ready, opening the possibility to users to subscribe to the action.

Administrator class: This class is the highest class in WebLink and only trusted employees of the company should have it, only administrators can give other users the administrator permission. As user with this class can change the information that is specific to the platform. The administrators can promote, ban or demote other users easily and change the logos of the sponsors for the platform.

6.2 Functional Requirements

The functional requirements have been documented formally using use cases as defined in RUP and their priorities have been defined using the MoSCoW nomenclature which represents four different levels:

- *"Must have"*
- *"Should have"*
- *"Could have"*
- *"Won't Have"*

In this technique to describe requirement priority *"Must Have"* represent the functionalities that the system must have to function properly and without whom the system can't be put in production. "*Should Have*" requirements represent the ones that the client would like to see in the platform but that are not needed for the project to go into production even though they still add some value. "*Could have*" are usually requirements that are only to be implemented if the project goes well and there is enough time to develop some as they would add some value to the final product but it is not enough to deem them as necessary. Finally "*Won't have*" is pretty explanatory, it has been decided that these requirements won't feature in the software being developed for the time being due to the lack of value added and the amount of work and time needed to put for that requirement that doesn't make up for the low value.

The functional requirements for WebLink have been separated from the main document and added as annex 1 due to the sheer number of them and the formality in which they are described as each use case takes a lot of space.

6.3 Non-functional Requirements

6.3.1 Scalability

If the platform is as successful as it is expected and succeeds in improving Konkrets' learning branch from a regional standpoint to a global one it will require a resource increase plan. Currently the company expects WebLink to grow on a yearly basis, increasing the number of simultaneous users using the platform.

Therefore, even though scalability was only implemented as a prototype, it shall provide the company with a solid plan on how to deal with this increase in demand. Currently there are two ways to deal with the increase of popularity of a platform in terms of providing scalability:

- **Horizontal Scalability:** This process refers to the ability of a system to increase and decrease the number of nodes working together in an application layer to achieve the expected throughput. Nowadays it is possible to configure hundreds of small computers in a cluster instead of having to use a supercomputer. The action of adding a node to a cluster is known as scale in, while removing a node is known as scale out. Due to the lower cost inherent to this way to achieve scalability this was the one chosen in the development of WebLink.
- **Vertical Scalability:** This process refers to the possibility of adding more resources to a single node of the system. Typically it includes increase the number of CPUs or memory in a single computer. Increasing the number of resources is also known as scale up while removing them is known as scale down.

The company expects the platform to reach at least 500 courses taking place concurrently 5 years after the platform goes in production. Each course shall have an expected average of 30 students and each student is expected to visit the platform once a day for an average of 1 hour.

Average visitations per day = 1250 * 20 = 15.000

Average visitations per hour = 15.000 / 24 = 625

Average concurrent users = 1 * 625 = 625

***Equation 1** - Scalability Calculations*

The platform scalability goal is therefore to provide prototypes that the company can use as guidelines in the future and all the benchmarks and goals will be set on the predictions done for the number of users using the platform in 5 years of production. A description of each layer and the goals for each one is presented below:

- **Application layer:** Assuming the average request time of each user is 5 seconds, the platform is required to support at least 125 requests per second. Therefore, the platform must be prepared to support the usage of multiple Tomcat web server nodes. The requests for those nodes are to be balanced using round robin which is possible due to the statelessness of the REST requests implemented by WebLink. Later on this document, lies both the description and results for the benchmarking process for this layer.

- **Data layer:** As is described later on the architecture section the platform will be using two types of databases: relational and non-relational.
    - Relational databases will have mostly a read load and should provide throughput of at least 375 select operations assuming that each request reads data from the relational database 3 times. Later on this document lies a benchmark showing that there was no need to create a prototype scaling in this layer. If there were the need to increase the throughput in this layer I

would have used a replication approach due to the load being mostly read operations.

- On the other hand non-relational databases will be have a high write load since this type of storage is mostly used to store logs. The platform creates an average of 2 logs per request and therefore this database would require a throughput of at least 250 inserts per second. Later on this document a benchmark is done showing that a single node of ElasticSearch is able to support this load. If there were the need to scale in this layer I would have used a sharding strategy due to the load being composed mostly of insertion operations.

As referred before, these prototypes are not supposed to be used in production directly as there was not enough time to thoroughly test the platform while using these strategies. Also, for these prototypes to be completely successful there would be the need to either implement a distributed file system (HDFS[14], Gluster FS), or to create a remote web service for file fetching and storage running on a different node from those that are currently being used in production. These solutions could not be implemented during the internship due to the low amount of time left after development.

6.3.2 Security

The platform supports real monetary transactions therefore security is an important non-functional requirement. Before presenting the security requirements it is important to introduce some of the attack vectors that have been taken into account:

- **SQL Injection:** This attack vector consists of a code injection technique used to attack data-driven applications. In this vector an attacker inserts SQL

[14] Hadoop File System

statements within input fields to spoof identity, tamper with the existent data or even destroy it.

- **Man-in-the-middle:** Man in the middle or MITM for short, consists of an attack in which an attacker secretly relays and possibly alters the communication between two parties. These parties believe they are communicating directly and in fact all their messages are being intercepted by a third party.
    - **Active Eavesdropping:** An active eavesdropping attack is an attack in which the attacker is intercepting messages between two parties with the objective of spying on its contents.
    - **Tampering:** In this type of attack the man in the middle changes the contents of the messages between the two parties.
- **Cross-site-request-forgery:** Cross-site-request-forgery or CSRF for short is a malicious exploit in which an attacker tricks the victim into submitting malicious requests. These attacks usually target platforms that rely on authentication to attempt identity theft or purchases.
- **Session fixation:** Also known as session hijacking is an attack vector in which an attacker attempts to steal a session from another person using that person's session identifier.
- **Buffer overflow:** A buffer overflow attack consists of an attack vector in which the attacker attempts to corrupt the data values in memory addresses next to the destination buffer by surpassing the capacity of the buffer.
- **Brute force:** Brute force attack in an attack vector in which an attacker attempts to login using another users authentication by trying to guess it. These attacks usually rely on a dictionary in which all words available will be tested.

A description of each security requirement considered is available in Annex 3. Even though functional requirements were described using use cases it was considered that approach would not be necessary in these requirements.

6.3.3 Interoperability

Although WebLink is a standalone application and should work by itself, the company has the need for the application to connect with other applications they already have or could have in the future such as their management platform Sik. Sik is a managerial platform that is already in production that is used to manage personnel and finances of the company. This platform must have information regarding every transaction and courses available for accountability purposes.

The system shall then implement interfaces that allow for external connection to WebLink information using neutral technologies such as REST so that it is possible for external applications (used on the premises only) to fetch information regarding, users , transactions, courses, actions and modules. Also, the system has the need to implement interfaces to connect with other external services such as:

- Facebook
- Payments provider
- Streaming Server
- Email Service Provider

6.3.4 Accountability

The system should keep logs of all actions done by each user alongside with the state of the overall platform (errors, warning, etc.) for managerial reasons and to further improve application support in the future by using this data not only to improve the platform by fixing bugs and adding fixtures but also to provide users and administrators with analytics regarding the information gathered by the platform.

Also the logs stored in the non-relational database shall be used to create dashboards that could be embedded in other platforms in an easy manner. These dashboards shall contain information regarding user activity and the overall platform status. As this data is stored using JSON it shall not be manually parsed and there is the necessity for Sik to contain the generated dashboards.

6.3.5 Usability

The platform shall implement a lot of functionalities and these need to be accessible easily. Therefore, the human-computer interaction is obviously an important non-functional quality that is needed in order for the system to be usable. This feature won´t be described architecturally but will be achieved by doing usability tests using different people that have no knowledge of the platform, and until the level of usability of the platform is as intended the computer-human interface shall be changed over many iterations.

Even though this platform will be used by both potential students, administrators and instructors this non-functional requirement focuses only on the students. Students are the target audience for the platform and the ones that will be funding it on the long term and therefore they are ones whom satisfaction is mandatory. An inquiry shall be presented to the usability testers and a maximum average number of clicks defined as primary heuristic for usability tests.

The usability requirements are presented in Annex 4 and note that */weblink* is the URL extension to the page to which a user is redirected to upon login. These usability tests will be done in a face-to-face basis and the results will be presented later on this document. The enquiry is available in Annex 2 and is composed of multiple choice questions ranked using a 5 point Likert-scale.

The Likert scale is a psychometric scale used widely in the development of questionnaires. When responding to an item respondents are supposed to specify how they feel about a specific remark on a symmetric agree-disagree scale. Currently the usage of the Likert scale is the most widely used approach to scaling responses in survey research. For usability to be considered a success the platform shall score at least an average of 3 / 5 in the inquiries and no more than one usability requirements mentioned previously can be unsuccessful.

7. Architecture

7.1 Technologies

7.1.1 Streaming

In any e-Learning platform the ability to stream or share audio visual content is of critical importance and WebLink is no exception however, this task is really demanding due the large bandwidth necessary to provide this service.

Bandwidth is probably the most relevant issue when it comes to streaming since it is such an intense task for the server to handle. A platform allowing 640x480 pixels at 24fps requires 600-700 kbps transmission rate using modern encoding (h.264). Having a conference call with 20 participants will require the server to withstand receiving and transmitting to all of them meaning that for each conference call the server must have at least 280Mbps, being capped to 3 simultaneous conference calls for a Gbit bandwidth.

Konkrets, only has one physical server, which made their requirement of allowing for N to N conference calls while achieving scalability and using their infrastructure nearly impossible so, after a lot of consideration I proposed two ways to deal with this situation:

- **Rented Server Cluster:** The first proposed method was to use an external service that provided the necessary tools and infrastructure to support all the requirements proposed by the client, however Infrastructure as service is usually paid, which might lead the client to reject this proposal, even though it is the best option.

- **Scope Reduce:** The alternative to using an external infrastructure is to reduce the scope by eliminating the need for conference calls and support only 1 to N broadcasts during classes instead of the initially requested N to N topology.

7.1.1.1 WebRTC

WebRTC is an API definition drafted by W3C that supports browser-to-browser application for VoIP and RTS without the need of any internal or external plugins. This was originally developed as an open-source project by Google before it became the standard for browser-to-browser Real time communication and it is supported by all major browsers at the moment. There are a lot of frameworks for WebRTC. Some of these technologies will be presented in the next few sections.

7.1.1.2 TokBox

TokBox is a service that provides an infrastructure and framework for WebRTC that costs 50€ for the first 10.000 minutes of streaming followed by a series of ladders in which there is a cost per minute, this way the system costs will probably be lower than the initial 50€ and will only surpass that value if the platform is indeed successful. With each milestone the cost per minute will decrease.

This system provides all the necessary streaming tools necessary in the implementation of WebLink such as: N to N streaming using a cluster running on their end, Live messaging, screen sharing and support for all major browser platforms (Firefox, chrome, Safari and Opera). TokBox allows for any number of simultaneous conference rooms each up to 24 members in an N to N topology or up to 1000 members using a 1 to N topology.

If the company chooses to use a rented server cluster this is the technology that I would recommend as it uses JavaScript for the integration, it is not expensive since the cost is equivalent to the success of the platform and from all the tools studied it is the one with the best documentation and support platforms. In the end this technology was adopted for the development of WebLink.

7.1.1.3 OpenWebRTC

If the client ultimately decides to reduce the scope and support only for 1 to N broadcast sessions I would recommend the usage of OpenWebRTC which is a framework of WebRTC. This technology is open-source and free, however it does not provide an infrastructure in which the service could be used, it would require the company's server to run this service itself. It uses a node.js server and JavaScript clients.

7.1.2 Non-relational Database

A non-relational database or NoSQL database is a technology that allows the storage and retrieval of data without the need for tabular relations used in relational databases. These kind of databases implement a simpler design which aims increase their scalability while maintaining a high availability and performance. At the moment there are two major types of non-relational databases:

- Key-Value store
- Document store

Databases such has Oracle NoSQL Database, Redis, and dbm are Key-Value stores which is the simplest model. This model is somewhat similar to a Hash map where there is a key associated to each entry. Elastic Search and MongoDB are document stores and in this model each entry is a JSON[15] object containing information (Can be represented by another technology depending on the database implementation) and even though there is still a key associated to each document engines implement lookup functionalities that allow queries to do a full text search on the JSON documents. There are another types of NoSQL databases such as graph stores and wide column stores even though they are not as used as the ones described before.

Most functionalities in WebLink rely mostly on read operations from previously inserted information on a regular SQL database however, requirements such as accountability

[15] JavaScript Object Notation

will rely heavily on writing so, and in order to reduce the load existent in that database WebLink will make use of a NoSQL one as well. Also it allows for an easier scaling increasing the overall capacity of the platform. Since one of the objectives of this internship is to practice using new technologies I have decided to use a Document Store in the platform as I have had previous experience using Key-Value stores.

I considered to use either mongo DB or ElasticSearch from the databases that use a document model since they are the most well-known. At the start I was more inclined to using ElasticSearch due to the easy integration with business intelligence tools using the ELK architecture (Explained later on this document) however, I decided to go with one with the better performance.

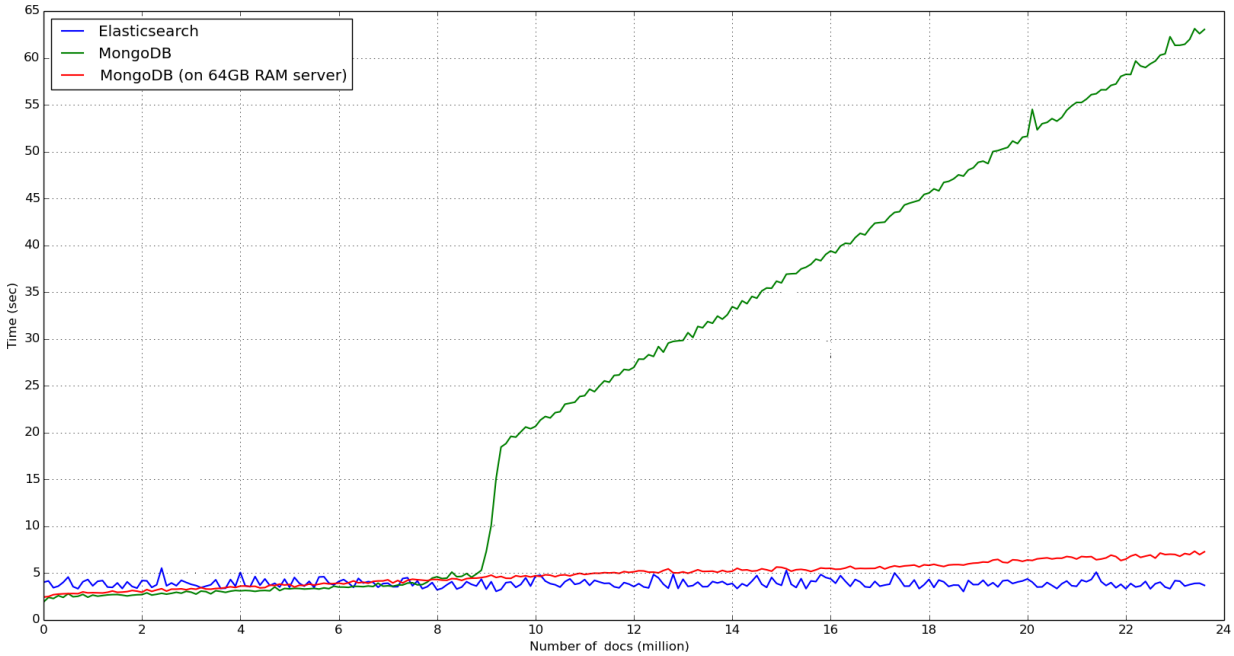


Figure 20** - MongoDB vs ElasticSearch performance inserting chunks of 100k documents and a total of 23M documents using 32GB of RAM. Benchmark from **[9]

As it is visible in the image above even though MongoDB performs better for a lower number of documents but with a higher number ElasticSearch is far superior. ElasticSearch therefore has both the performance to support the number of inserts alongside with the necessary integration tools to create a business intelligence solution for a production environment in an easy manner.

7.1.3 ELK Architecture

ELK architecture is an architecture used to provide simple business intelligence solutions using three simple pieces of software:

- **ElasticSearch:** Document oriented database and RESTful data indexer, it provides a clustered solution to store and make searches through sets of data.
- **Logstash:** Responsible for collect the data, make transformations like parsing – using regular expressions – adding fields, formatting as structures like JSON, etc and finally sending the data to various destinations, like a ElasticSearch cluster. Similar to the ETL [16]process without the need to extract.
- **Kibana:** Web-based application, responsible for providing a light and easy-to-use dashboard tool. This tool allows the administrator to create any query and then share it using a HTML iframe or just store it for later use.

***Figure 21** - ELK Architecture*

As referred before, accountability is a requirement for WebLink therefore it is important for the system to keep logs of user actions and software status however, logs alone are meaningless without tools to interpret and gather information from them. So I implemented a simple business intelligence solution that will provide the administrator of the system with up-to-date dashboards regarding the information logged using the ELK architecture. Even though I am using this architecture I cut out Logstash as logs will not be need any pre-parsing since they are standardized within the application and that would only

[16] Extract, Transform, Load

make the process slower, however, if more instances of the application are created with different logging technologies it is advised to use Logstash.

7.1.4 Hibernate

Hibernate is an object relational mapping framework for Java and it provides an abstraction for mapping an object-oriented domain model to a relational database. It solves mismatch issues by replacing direct, persistent database accesses with high-level object handling functions. This tool is free and it was release under the GNU Lesser general public License. It is configured using either a XML descriptor or java annotation files named entities each describing a table in a relational database. Relationships between multiple entities are also facilitated during development since they represent object relations. Also worth noting that hibernate has its own query language HQL (Hibernate query language) that is somewhat similar to SQL. Hibernate also provides an alternative to using HQL [17]which is criteria queries which are more oriented towards object oriented programming even though they are not as flexible.

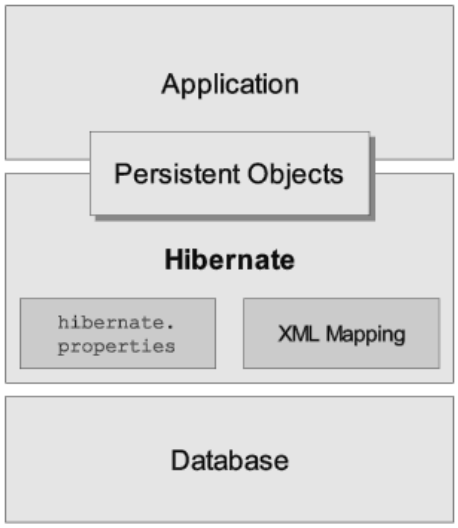


Figure 22 - *Hibernate Architecture*

Hibernate provides transparent persistence for POJOs [18] and the only requirement for a persistent class is a no-argument constructor, not necessarily public. Collections of data

[17] Hibernate Query Language
[18] Plain old Java Objects

objects are usually stored using Lists or Sets (Java 5+ generics). Related objects can also have cascade operations which will make sure all objects are updated upon change making it easier to ensure integrity.

7.1.5 MVC

MVC or model-view-controller is an architectural pattern invented by Xerox Parc. in the 70s, initially present in a SmallTalk-80**[12]** that is used to implement user interfaces on computers. This framework became extremely popular for designing web applications and is now used widely in a variety of projects. MVC architecture basically consists of dividing the application in three distinct components (or groups of components):

- **Model:** The model is the cornerstone of the application and it directly handles the logic for the application data. Often model objects are used to retrieve data (and Store) from a database.
- **View:** The view part of an application is responsible for the handling of the data to be displayed. Usually these are created by the data retrieved by the model.
- **Controller:** Controllers are responsible for user interaction and typically they get data input by the user and send it to the model.

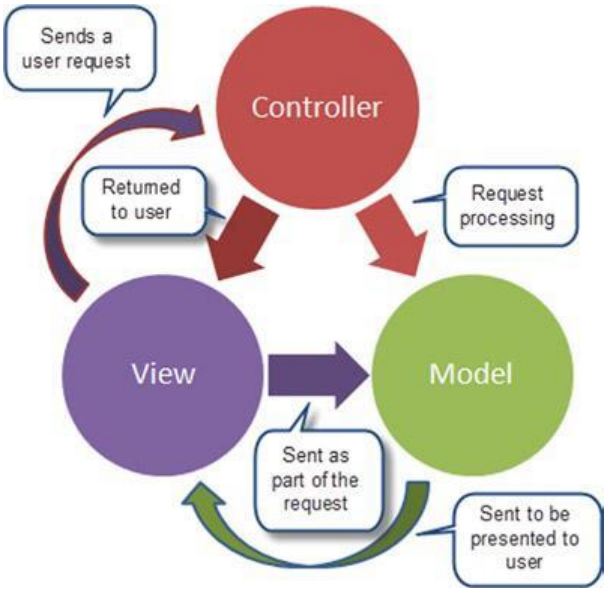


Figure 23** - MVC Architecture retrieved from **[12]

The previous image represents the basic interactions between the three components of the Model-view-controller architecture. As the reader can see, it separates different parts of an application which has two important benefits:

- **Multiple View Support:** Provides support for multiple views for the same data at the same time due to the decoupling from models and controllers. Nowadays this architectural pattern allows an application to easily create new visual ports (Web, Android, etc...) without having the need to change any business logic behind it.
- **Change accommodation:** Accommodates change since a change in one of the components doesn't require the remaining ones to change as well. For example, if there is the need to change the views the model won't need to be changed.

7.2 Quality Attributes

The quality attributes are represented as a utility tree. The table representing this is available in Annex 5.

7.2 System Context

WebLink will be operated through both human and automated means therefore it implements interfaces that facilitate interoperability. Clients and other users can operate the platform by using the front End designed while connections to other software is available using neutral integration points such as HTTP/Rest or Soap requests.

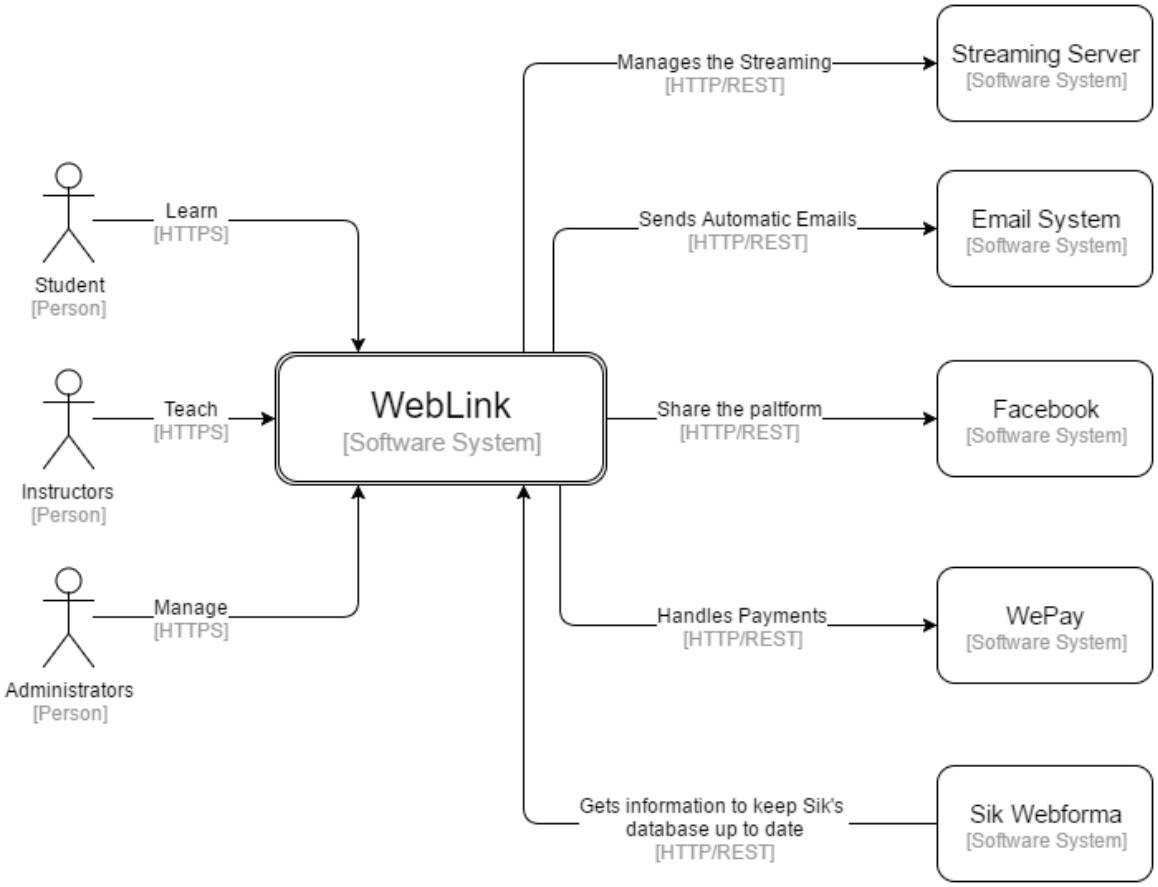


Figure 24: *System Context view*

The system shall be used by all the actors described previously in the requirements section and Sik Webforma which is the management system that Konkrets already has in production. The clients can connect only using HTTPS and will do so by using the front end made available to them by the platform. Sik Webforma needs to fetch information regarding the courses being taken, transactions or the users that have subscribed to the platform so the system implements a neutral integration point that will facilitate the gathering of such

information. WebLink also integrates with Social Networks such as Facebook and it uses their API to do so. WebLink also connects to the TokBox the streaming service described previously so that classes and conference calls can take place in the system without overloading the bandwidth of the server in which the platform is running.

There is also the need to integrate the platform with WePay so that it is possible for the system to validate course purchases and transactions .Finally WebLink also integrates with an emailing service provider so that notifications and automatic emails can be sent. Specifically for this internship I used Gmail, however it is likely that the company shall replace that with their own emailing service once they have it.

7.3 System Containers

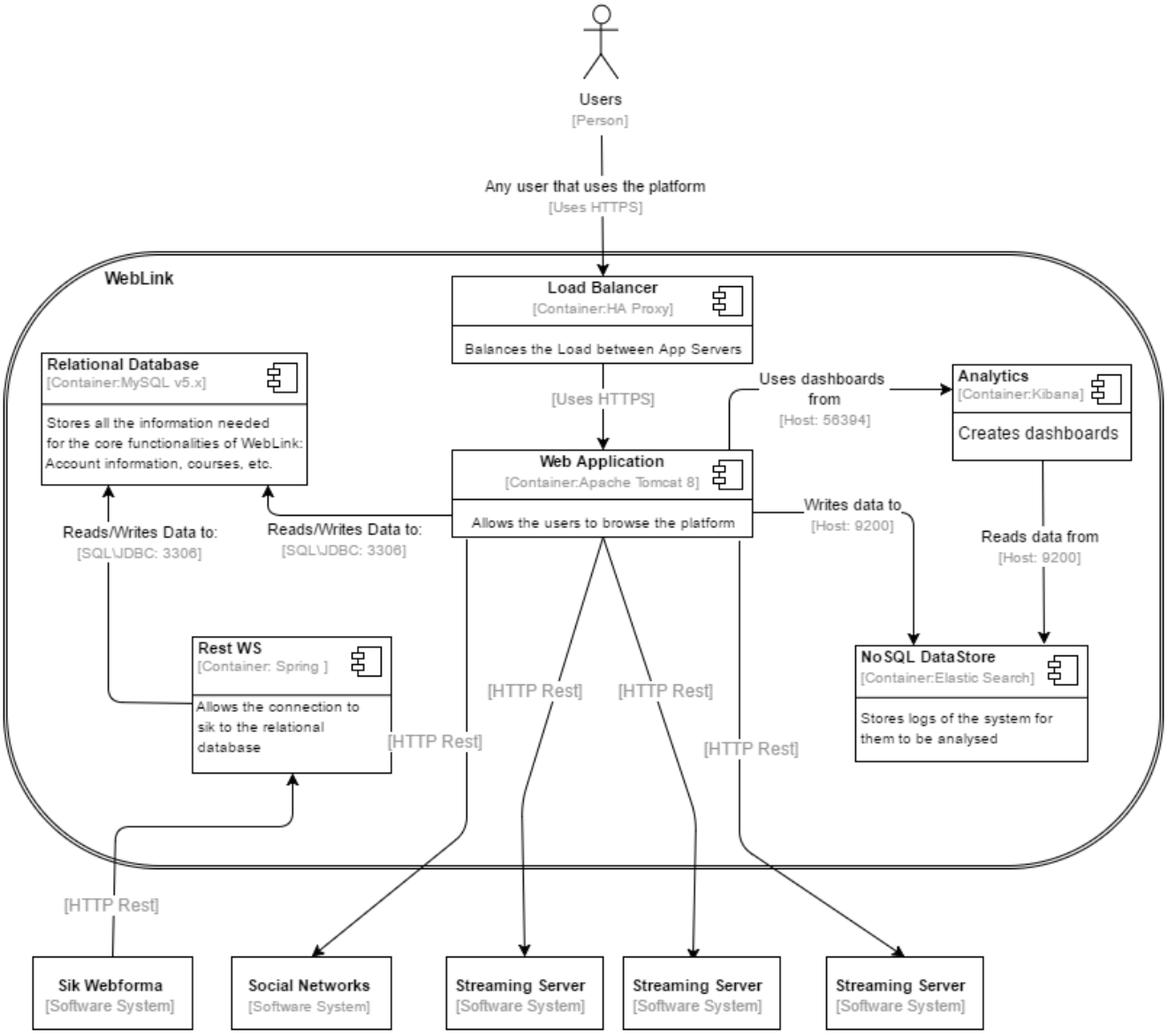


Figure 25: *System Containers view*

The previous image explains how the platform will connect to the multiple containers in the system. When a user connects to the system a load balancer (HA Proxy) will redirect the user to the correct application server. Figure 26 only displays an application server container however, it actually represents a series of tomcat servers whose requests will be load balanced by HA Proxy. This is possible because of the data-driven nature of platform which results on requests to be completely stateless. A more detailed representation of that

connection can be seen in the following image that depicts how HProxy balances the load between the multiple servers:

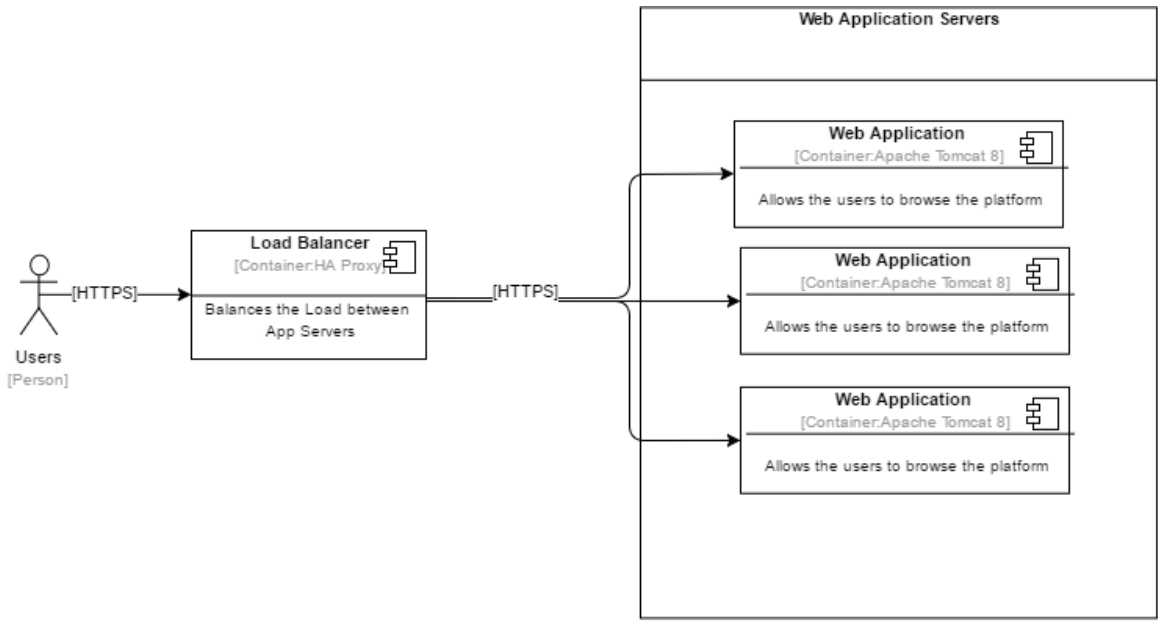


Figure 26 - *Load Balancing Tomcat*

In Figure 26 we can see that a user can make a request to HA Proxy which will distribute the load between N application servers. The distribution uses a round robin algorithm to define where the next request should be redirected to.

Back to the containers view. It is possible to observe that the application server connects four different external systems. It connects to social networks (Facebook) using their API so it is possible to post your achievements on the users wall. Using rest the application server also connects to TokBox to handle the streaming. Finally it also connects to the emailing service provider and a payment handler.

As referred before the system makes use of two distinct databases, using two different paradigms. The relational database is MySQL and it is responsible for the storage of the data important for the platform to function and it can be used by both the application server and a REST web service. The web service is the bridge between the platform and Sik which is the management system already in production at Konkrets. Sik can then retrieve data from the platform by making requests to it. Also worth to note that both the application server and the web service connect to the database using hibernate.

The other database that is used in the image is ElasticSearch that can work together with Kibana to provide the platform with dashboards or data in the format of JSON regarding pre-defined queries on the information. As referred before in the ELK architecture section Logstash was left out since all logs are coming from the same source.

7.5 Web Application Component

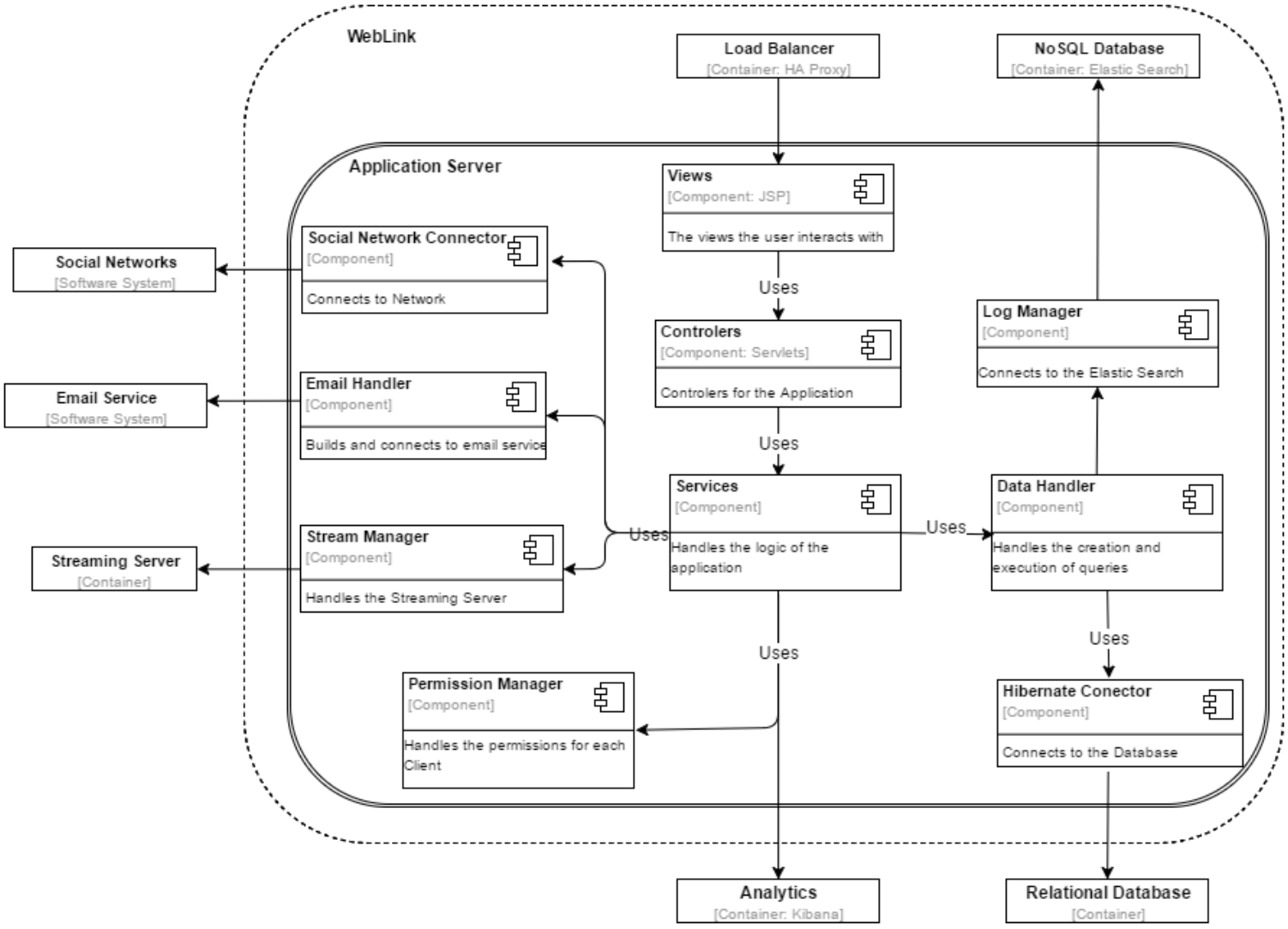


Figure 27: *Web Application Components view*

The Application container will use a MVC model and it is primarily composed of 4 components. The views which is what the user sees and interacts with. Controllers which handle which view will be return for each request, the services which handle the logic of the application and finally the data handler which is responsible for the creation and execution of queries. For more description on what each of these components does read the section about Spring MVC in the development section.

The stream manager is responsible for handling who should receive each stream using the TokBox API. Email handler is responsible for the creation of email templates and using the API for the emailing service. The social network connector is responsible for the connection with Facebook.

Services can use Kibana to request a set of data or a dashboard to place on the front end of the platform. Permission manager is the component responsible for the authorization of the system. Log manager and Hibernate connector are the components that connect directly to the databases.

8. Risk Management

8.1 Risk Assessment

	Description	*Likelihood*	*Consequences*
RM1	Project Scope might be too extensive and therefore finishing the project might not be possible during the span of the internship.	High	High
RM2	Changes to requirements might create additional work due the process in use and volatility of the project.	Medium	Medium
RM3	Testing the platform in a real production environment might not be possible due to the limited time in the internship.	High	Low
RM4	Company might decide not to invest on a remote streaming server cluster seriously compromising the scalability of the platform due to limited bandwidth.	Low	High

Table 3 *- Risk Assessment*

8.2 Risk mitigation

In case R1 becomes a certainty the project scope shall be reduced by meeting with the client to re-evaluate requirement prioritization and in a more critical scenario in which the scope can't be reduced any further the internship shall be extended to the special phase in September in order to achieve completion.

In case R2 happens and a set of requirements needs to get changed those will only be accepted if the overall amount of remaining development needed is less or equal to the one remaining in the internship.

Due to the limited time inherent to the internship and the large scope that was set from the start R3 is likely to occur in which case, prototypes regarding production strategies shall be developed or designed but not actually implement in a real environment serving only as guidelines for future reference.

If the company decides an investment in a remote streaming server cluster is no-go (R4) then I shall implement a solution using a local server however, scalability shall be dropped from the project as it shall be impossible for the application to actually scale much due to the limitation of the bandwidth existent.

9. Development

9.1 Build Automation

Build automation is a technique that provides dynamic compiling of the source code into binary code, packaging of the binary code and the running of automated tests. This is necessary to implement continuous delivery techniques and has been proven to greatly improve the quality of the product while reducing development time by eliminating *"bad builds"*. This technique also reduces the amount of time lost getting dependencies and accelerates the process of linking multiple projects. The most well-known build automation tools are Apache Ant, Apache Maven and Gradle. For this project I have chosen to focus on Maven because it is the most used nowadays and therefore should be the most useful one to know in the near future.

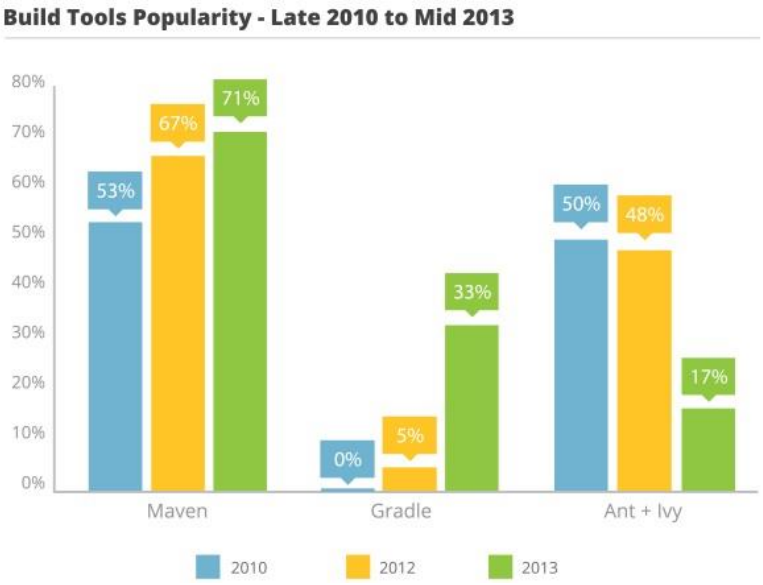


Figure 28 *- Build Automation tools popularity comparison retrieved from* ***[10]***

Maven is a build automation tool used mostly in Java projects but it can also be used to manage projects in other languages. Maven focuses both on the description of how the software is built and its dependencies. Maven uses XML to describe the project and it dynamically downloads the necessary Java libraries from a centralized repository. Maven is also used to map Junit tests that the project should clear on the build phase to be deemed successful.

When using Maven the description of the project should lie in the root directory within a file named *pom.xml*. In the development of WebLink I decided it would be helpful to use a build automation tool for the following reasons:

- **Jenkins Integrations:** The first reason a build automation tool was deemed necessary was the integration of the project with Jenkins because it allows it to compile and build the project using the information available at the *pom.xml.* This way there is also a description on how should the project be built and which Junit tests to run for the build to be considered successful and therefore deployed.
- **Dependency control:** By using Maven the needed for the project are automatically downloaded from the centralized Maven repository. Also, if the developer wants to upgrade any dependency Maven makes it simple by replacing the old version Java libraries with the new version, by simply changing the version number for that dependency in the *pom.xml*.
- **Structure convention:** Using Maven the project is sure to respect a conventional structure which makes it easy for anyone to understand and eventually make necessary changes. The Structure of the project is represented in image 28.

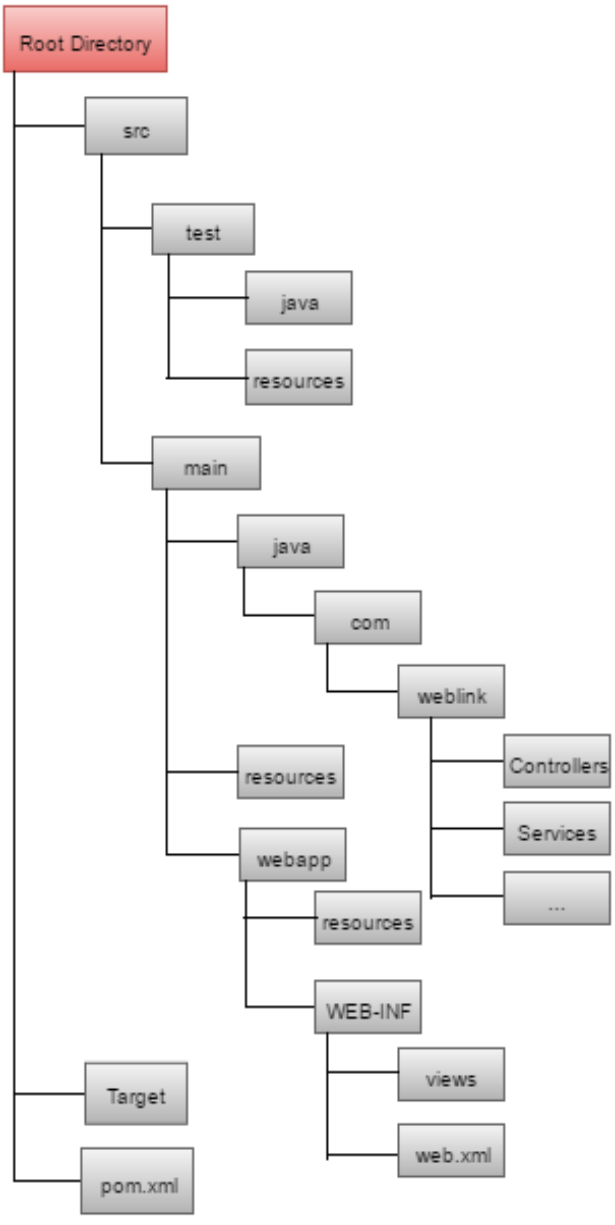


Figure 29 *- WebLink Structure*

9.2 Application framework – Spring MVC

An application framework consists of an abstraction in which software providing generic functionality can be selectively changed by additional user-produced code. Therefore, an application framework has the objective of promoting a standard structure for applications and make the creation of new functionalities rather easy while maintaining code integrity making it easier for future changes in the development team.

In the past I have only used the Struts2 framework therefore, and since the objective of this internship is to learn how to use new tools and techniques, I decided to use the most popular framework available (If Struts2 were the most popular I would choose the second instead) which happens to be Spring MVC as shown in the following image:

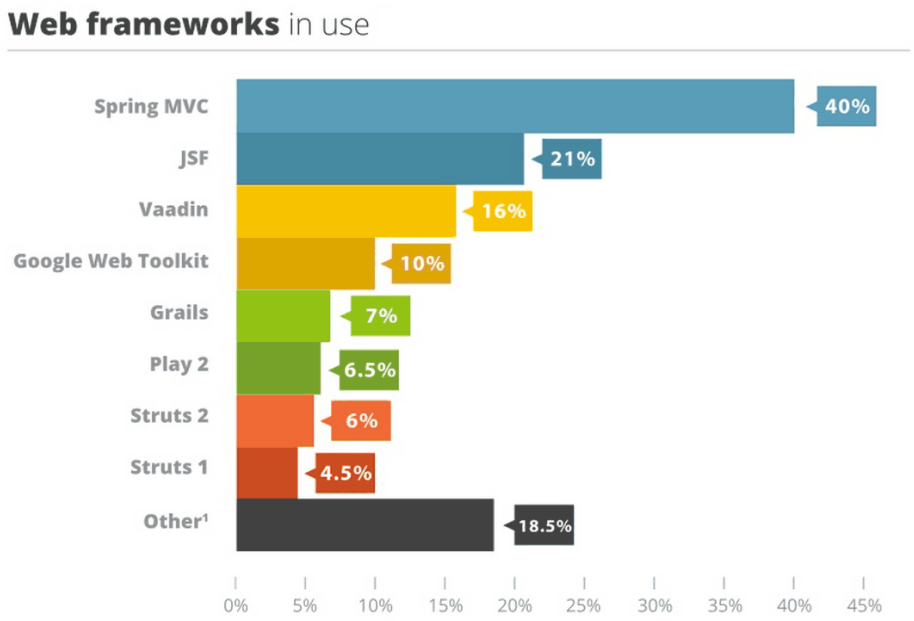


Figure 30 - *Build Automation tools popularity comparison retrieved from* ***[11]***

The Spring MVC architecture provides a model-view-controller architectural paradigm and is designed around a Dispatcher servlet which is responsible for the handling of all HTTP requests and responses. After receiving an HTTP request, the dispatcher consults a handler mapping container that will decide which controller shall handle the request.

Whenever a controller gets a request sent to it by the dispatcher servlet, one or multiple service methods will be executed and the name of the view to be resolved returned, these are responsible for handling the business logic and executing methods from the data

access object layer that will be responsible for doing operations on the database using hibernate. Finally the dispatcher servlet will consult with a view resolver to pick up the defined view for the request and then passes the model data to the view which is finally rendered on the browser. This process is detailed in the following image:

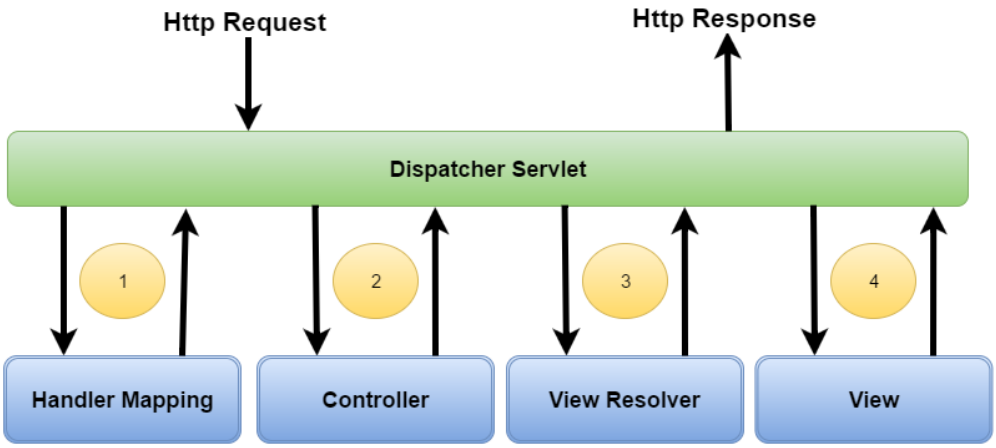


Figure 31 *- Spring MVC Architecture*

9.2.1 Inversion of Control Container

One of the cornerstones of the spring framework is the usage of an Inversion of Control (IoC) container. Inversion of control also known as dependency injection (DI) is a technique in which objects are configured and managed using reflection (Reflection is the ability of objects examining, introspecting and modifying its own structure and behavior at runtime).

The IoC container manages Java objects since they are instantiated using a *BeanFactory.* Components instantiated by such container are named beans and it is the container that manages the bean's scope and lifecycle events. The advantage of such technique is that components are then loosely coupled which allows the developer to code for abstractions and creates more testable and reusable code.

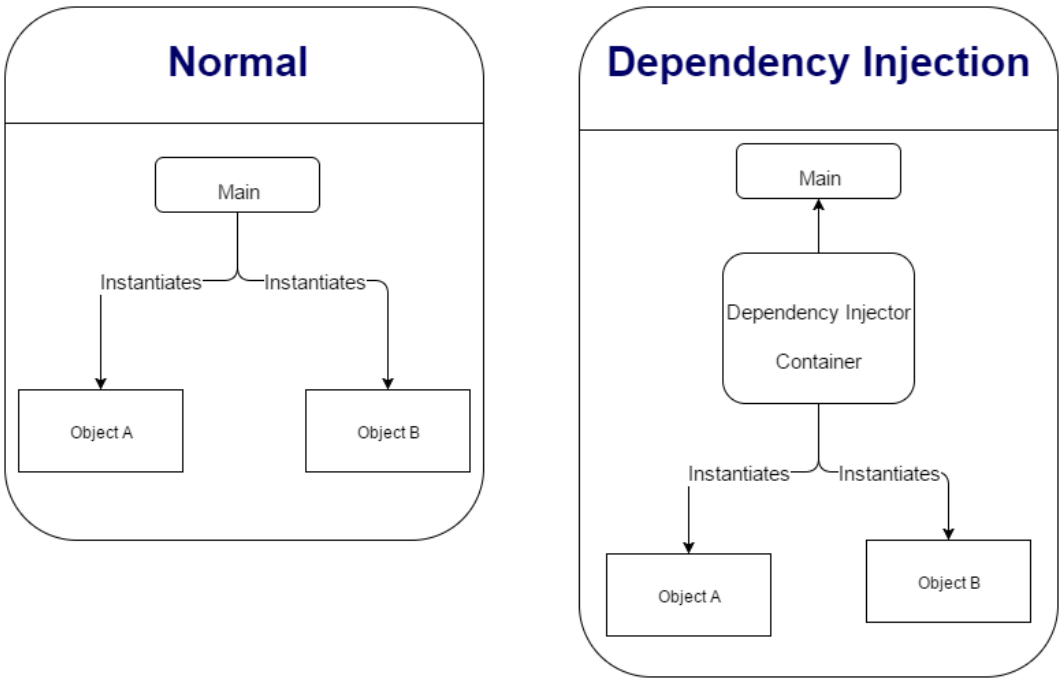


***Figure 32** - Dependency Injection Diagram*

As you can see in the previous image, in a normal scenario (not using DI) the main class would be the one dictating which classes it would depend on. On the other hand, in the right scenario, the class does not know which dependencies it needs, and the dependency injector informs the classes which modules it is supposed to use. This Spring MVC technique can be configured both by loading XML files or detecting specific Java annotations on the configuration classes. These data sources will contain the bean definitions that provide the information required to create the beans.

This technique is not mandatory when using Spring MVC since dependencies can also be looked up instead. This alternative is a pattern in which the caller asks the container for an object with a specific name or a specific type. Somewhat similar to the lookup system used by EJBs (Enterprise Java Beans).

9.2.2 Data Access Abstraction

The backend of a Spring MVC application usually is composed of three different layers to provide the connection between controller and model:

- **Controllers:** Usually Servlets 3 are used using the @Controller Java annotation. In this layer a controller gets an http request and calls service methods to complete that request.
- **Services:** A service in Spring MVC is declared using the @Service Java annotation and it consists of a service that can be looked up (Similar to the usage of EJBs). This layer is responsible for the business logic of the application and it does not connect directly with the database. Usually DAO layer and service layer are bundled together however, they represent different logical entities and therefore separating them will provide some modifiability because a change in the DAO layer won't affect the business rules.
- **Data Access Objects (DAO):** This final layer of Spring MVC is declared using the @Repository Java annotation and it consists of a Java Service that can be looked up. This layer is responsible for the connection to the database using the hibernate bean. The objective of this layer is aimed at making it easy to work with data access technologies as JDBC or Hibernate in a consistent way. The existence of this layer allows one to switch between the aforementioned persistent technologies fairly easily without having to worry about catching exceptions that are specific to each technology.

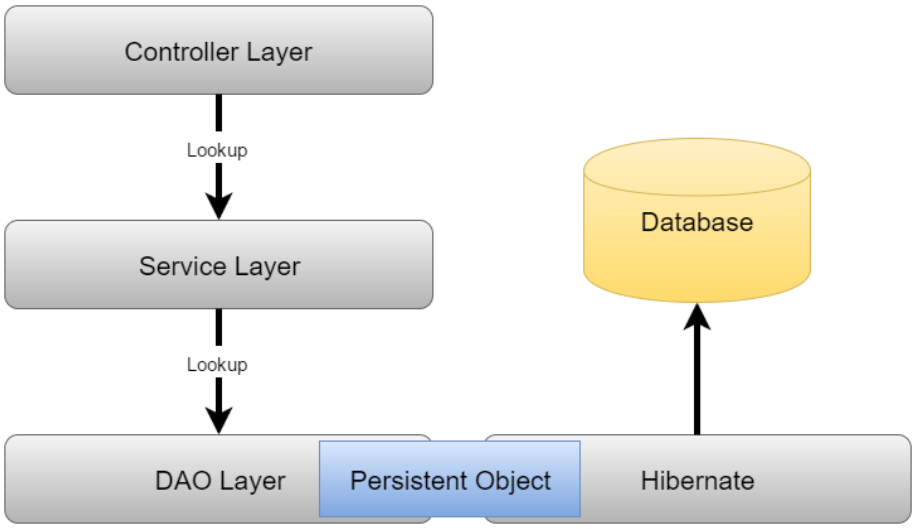


Figure 33 *- Spring MVC Layers*

Figure 34- *Configuration using servlets 3 Code Snippet*

9.3 FrontEnd

For the FrontEnd part of this platform I have used the following technologies, tools and techniques:

- **HTML:** Hyper Text Markup Language or HTML was used to define the web pages.
- **CSS:** Cascading Style sheets or CSS was used to define the presentation of the HTML pages.
- **JavaScript:** JavaScript was used for a few reasons. First one was to validate user input for type, content and length before the request reaches the backend so that the user can be notified if there is any identified issue with the data provided. JavaScript was also used alongside with AJAX to make some pages more dynamic providing the platform with functionalities without having to reload. This technology was also used to integrate with other systems such as Facebook, TokBox and WePay. Finally, JavaScript was also used to add animations, generate popups, etc.
- **Bootstrap:** Bootstrap is a web development framework that provides previously developed CSS and JavaScript. This tool was used because there wouldn't have been enough time on the internship to create all the required CSS and JavaScript from scratch.
- **Ajax:** Ajax or Asynchronous JavaScript and XML is a technology that allows the views to communicate with server side scripts by exchanging information using standardized formats such as JSON. This technology makes use of the XMLHttpRequest object and was used to reduce the number of reloads necessary for some actions within the platform.
- **Apache Velocity:** Apache Velocity is a Java-based templates engine. This technology was used for the automatic emailing system used in the platform. These emails would require some information to be dynamic rather than the similar for every user and with this technology the HTML source code doesn't need to be embedded in the Java code which provides higher modifiability to the email templates visually.

10. Final Product Presentation

In this section lies an overall description of the achieved product with the objective of providing the reader with a general view of how the platform operates. It was decided that a comprehensive detailing of every view would not be of benefit due to the large number of different screens inherent to the platform which would make this chapter a lot bulkier.

The following image depicts the screen to which a user is redirect upon login. This screen was chosen due to its simplicity and lack features to explain the overall usage of the platform and how to navigate it.

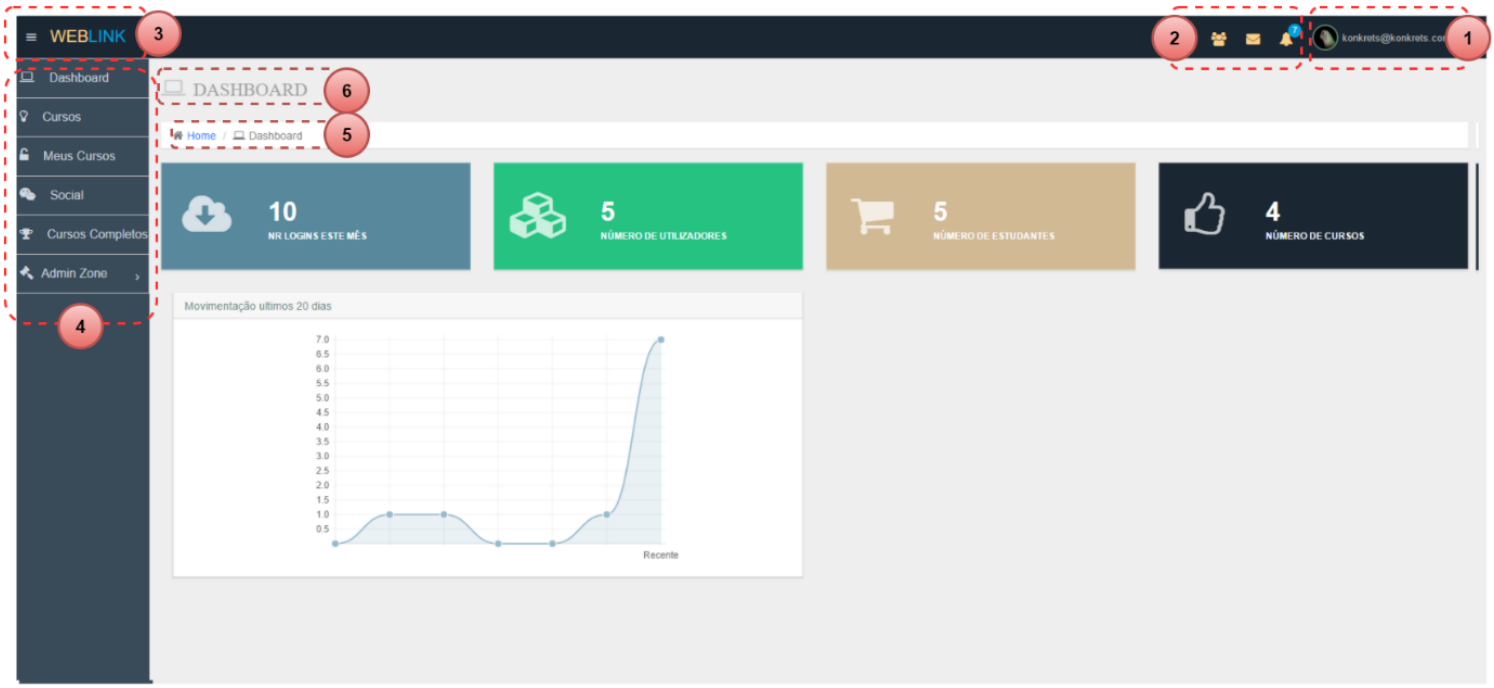


***Figure 35** - Dashboard view. Platform skeleton overview*

In all pages of the platform (but homepage and classroom) there are six elements present. Each number of the following listing represents a mapping in the image above and a part of the skeleton of the platform:

1. A button that has both the user chosen avatar and email. This button is a dropdown that allows the user to enter his profile information (Where the user can delete or change the information regarding his account and upload a new avatar). It also has an option to logout and a link to received personal messages.

2. Notification buttons that show how many friend requests, notifications and personal messages a user has. Each of these buttons is a dropdown in which a user can check details on the notification.
3. In the top left of the screen there is the logo of the platform (which is a link redirecting to this page) and a button that allows the user to both hide and show the left menu bar.
4. Menu bar with links to the many pages of the platform. Some of them such has admin zone are expandable into further options.
5. Breadcrumbs. Original crumb is the home page and the second one this dashboard page.
6. Title of the page.

The aforementioned elements are a constant throughout all the pages of the platform and represent the platform skeleton or template. Some of the views throughout the platform allow the user to open popups so that there doesn't have to be page redirects all the time. The following image is an example of these popups in terms of structure:

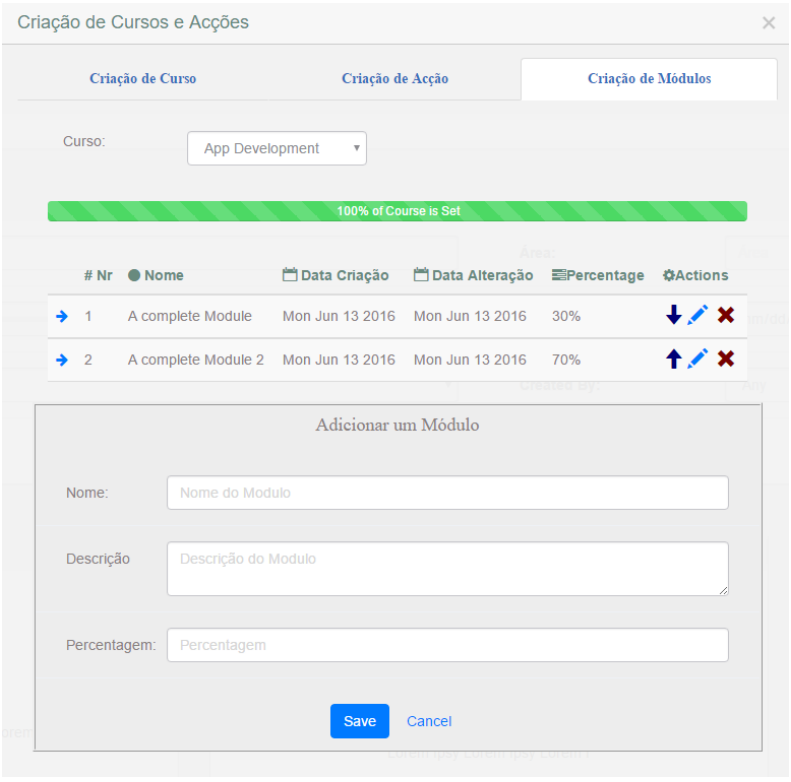


***Figure 36** - Popup example. Creation and management of modules.*

The previous image is a popup in which the user can create courses, actions and modules. The selected tab is used for the creation of modules for a course. All these popups are in fact hidden divs using JavaScript and they can all be dragged around. These popups can also be close by either clicking outside the box or using the X button to close them.

As referred before the platform does a user input verification both client and Server side. Client side informative popups were create to confirm choices or to inform that the user has made an error in an input. Also these popups are used to let the user know when a task is well successful:

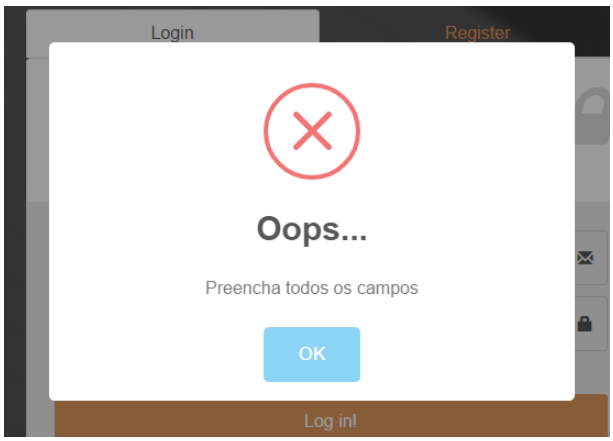


Figure 37 *- Warning message*

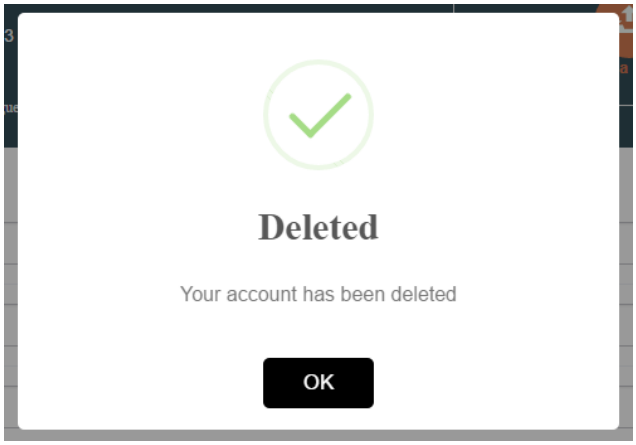


Figure 38 *- Success Message*

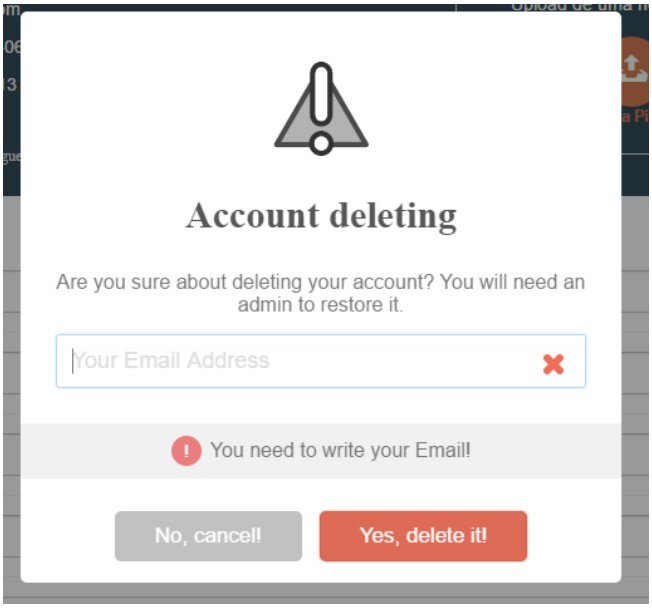


Figure 39 *- Confirmation message*

The previous images are examples of the different types of informative popups that give feedback to the user about the input. The warning message is used to tell the user something went wrong with the input, confirmation messages are used in critical decisions such as account deletion to make sure the user is aware of what he is about to do and finally success messages are used to let the user know their actions have been successful. The examples shown are from a failed login and account deletion confirmation and success feedback.

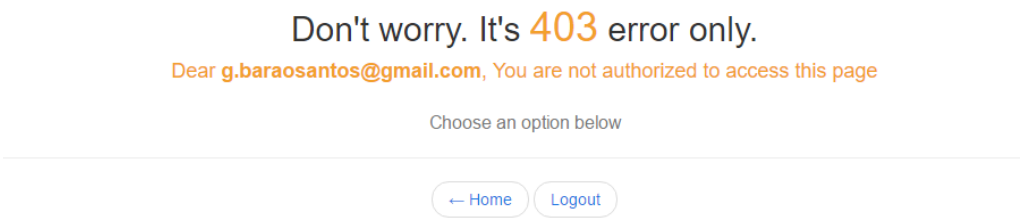


Figure 40 *- Error management example. 403 error*

Previously on this document, on the security section, it was also described that errors shouldn't provide the user with too much information regarding its cause. Therefore if the following errors occur they will show a message such as the one visible in the previous image (different colors are used to map different errors):

- **Error code 400:** Shown with the color blue, this error represents a bad HTTP request. These errors occur due to the server inability to process a client request.
- **Error 403:** Shown with the yellow color, this error occurs when a user tries to access a page to which he has no privileges to. In the image example g.baraosantos@gmail.com is trying to access an administration zone to which he shouldn't have access.
- **Error 405:** Shown with the green color, this error occurs when a request method is not supported by the application. For example, a GET request on a form which requires data to be presented via POST.

Error 404 was not described in the listing above because the visual presentation has been treated differently. An error 404 also known as not found occurs when a client tries to request a resource that does not exist either due to a bad link, user manual URL writing or future implementation. When an error 404 occurs it is shown to the user in the following manner:

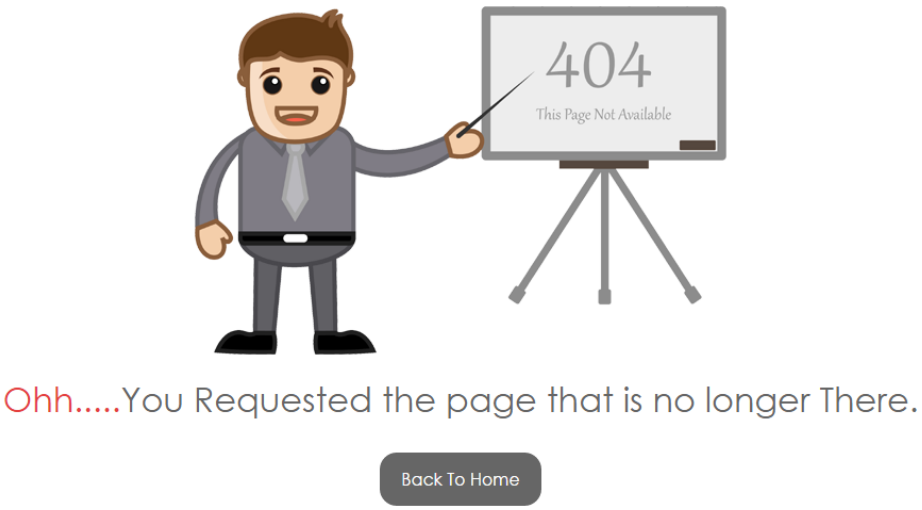


Figure 41 *- 404 Error*

Another thing that should be reference in this section is the fact that the application not only allows the client to download materials but also to consult some them on the page itself. Materials such as sound files and video files can all be consulted on the spot without the need to download.

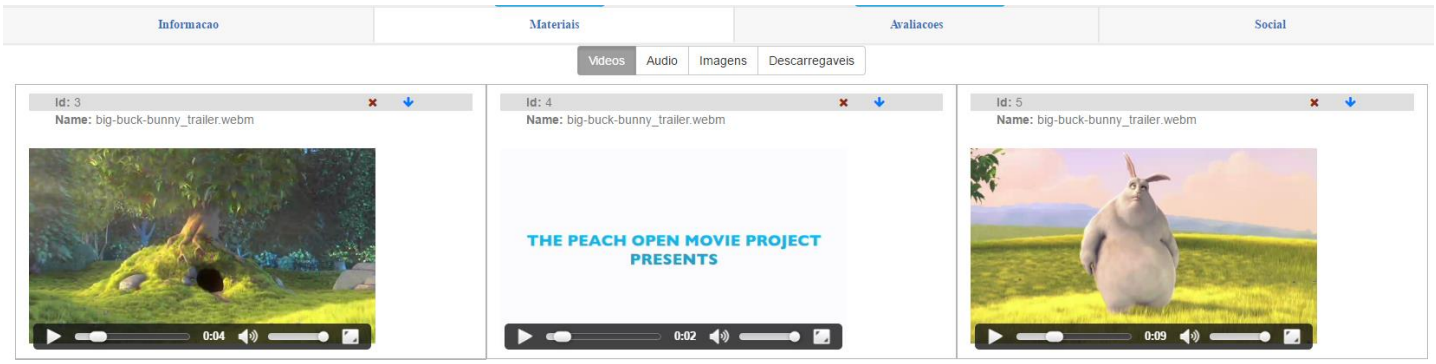


Figure 42 *- Interactive materials. Video Example*

The final view I am going to describe in this section is the classroom view. The classroom is always open during the duration of a course. I tried to give a different, more simplistic view to the classroom, so that students wouldn't get distracted with more functionalities than those actually needed during classes and so that it wouldn't look like just another part of the application. The following screenshot was taken using an account with instructor privileges during class.

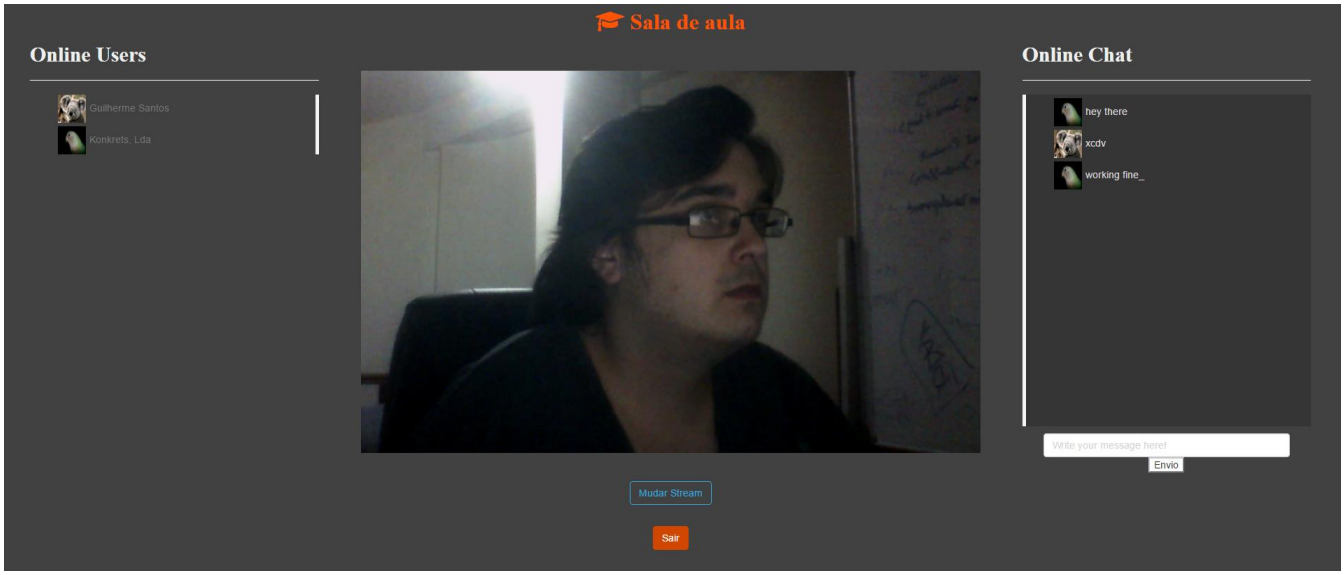


Figure 43 *- Classroom: Instructor view*

The classroom is divided in four areas. One that shows the online users to the left. To the right lies the instant message chat in which users can interact between themselves. In the center of the view is a 720p stream of the instructor and finally on the bottom lie two buttons: The orange one is available to all users and allows that user to leave the classroom while the second one is only visible to the instructor and allows him to change the video input between webcam and screen sharing. Even though the application does support screen sharing the instructor has to download and install a browser plugin beforehand. This video input change allows the instructor to make presentations using power points for example or show videos during a class in an easy manner.

Finally it is also worth noting that the stream can also be muted. If a user scrolls over the stream a microphone icon shows up. If that icon is clicked the stream can be changed between muted and not muted.

11. Requirement Validation

11.1. Used Tools

11.1.1. Junit

Unit testing is a software testing method in which programmers can create independent test cases. Usually unit testing is assisted by frameworks that allow the creation of mock objects and fake data to simulate a real environment. These types of tests are widely used by development teams all over the world to make sure functionalities are working as expected. From the most used frameworks that support Unit testing for Java projects I have decided to use Junit because it is one of the most popular. A study was done back in 2013 where 30.000 projects from GitHub were analyzed supports the previous claim:

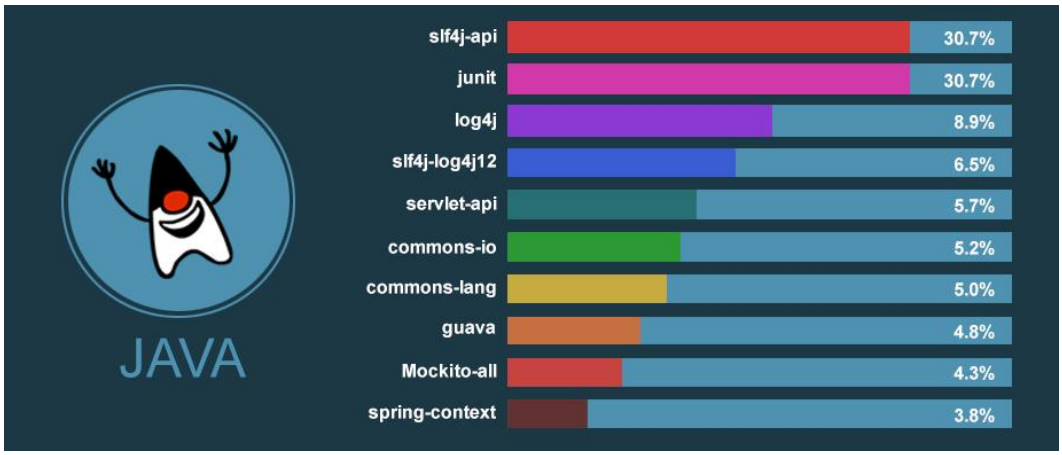


Figure 44 - *Most used Java third party software* ***[14]***

11.1.2. Spring Security

Spring security is a framework that can be used alongside with Spring MVC to promote standardized security measures. As Spring MVC, Spring Security can also be configured using servlets 3.0 and focuses on providing both authentication and authorization functionalities to Java based Applications.

Spring security also provides the necessary methods and techniques to prevent attacks like fixation attacks, CSRF and many others while maintaining just enough extendibility to meet custom requirements and implement new functionalities.

11.1.3. Let's Encrypt

Let's encrypt is an open source certificate authority that provides the generation of free X.509 certificates needed in TLS[19] and therefore HTTPS. Let's encrypt is a company created by a partnership between the University of Michigan and the Mozilla Foundation that has the objective of automating the validation, signing, installation and validation of certificates used in secure websites.

11.1.4. BCrypt

Many authentication schemes depend on secret passwords. Unfortunately, the length and randomness of user-chosen passwords remain fixed over time. In contrast, hardware improvements constantly give attackers increasing computational power. As a result, password schemes such as the traditional UNIX user-authentication system are failing with time. **[15]**

BCrypt is a salting algorithm that uses the Blowfish algorithm. Blowfish is an algorithm designed with the objective of taking user input passwords and encrypting them using a 64-bit lock cypher. BCrypt uses a 128-bit salt and encrypts a 192-bit magic value. Another popular hashing algorithm is MD5, however see how it performs when benchmarked against BCrypt using an i5 processor in the image to the right.

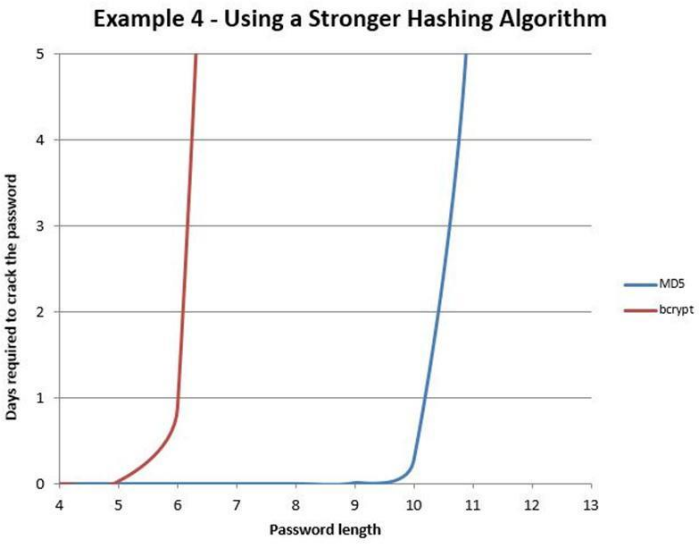


Figure 45** - BCrypt vs MD5 **[16]

[19] Transport Layer Security

11.2. Functional Requirements

The functional requirements were described previously in this document and expanded using use cases in Annex 1. The following table represents the clearance for the *"must have"* requirements:

TOTAL	MUST HAVE FUNCTIONAL REQUIREMENT LIST	COMPLETED STATUS
31	FR1.1 ; FR1.2 ; FR1.3 ; FR1.4 ; FR1.5 ; FR1.7 ; FR2.5 ; FR2.6 ; FR2.9 ; FR2.10 FR2.11 ; FR2.13 ; FR 2.17 ; FR2.18 ; FR3.4 ; FR3.7 ; FR4.1 ; FR4.2 ; FR4.3 FR4.4 ; FR4.5 ; FR4.6 ; FR4.7 ; FR4.9 ; FR4.10 ; FR5.7 ; FR5.8 ; FR5.12 FR5.13 ; FR5.14 ; FR5.15	✓
0		✘
5	FR2.14 ; FR 5.1 ; FR5.2 ; FR5.3 ; FR5.4	● [20]

Table 4 *- Must have requirements clearance*

In the previous table it is worth noting that there were 36 requirements and a clearance percentage to the date of 86%. The remaining requirements will still be developed until the end of the internship as 100% clearance is required for *"Must Have"* functional requirements. The next table depicts the clearance for *"Should Have"* requirements.

TOTAL	SHOULD HAVE FUNCTIONAL REQUIREMENT LIST	COMPLETED STATUS
10	FR1.6 ; FR1.8 ; FR2.1 ; FR2.2 ; FR2.3 ; FR2.4 ; FR2.16 ; FR5.16 ; FR5.17 FR3.3	✓
2	FR5.5 ; FR5.6 ;	✘

Table 5 *- Should have requirement Clearance*

[20] To be implemented until the end of the internship

In the previous table it is worth noting that there were 12 requirements and a clearance percentage of 83%. Enough should have requirements have been developed for the platform to be considered successful as described previously on the scope of the project (The scope predicted a clearance check of at least 75%). The next table describes the clearance of the *"could have"* functional requirements:

TOTAL	COULD HAVE FUNCTIONAL REQUIREMENT LIST	COMPLETED STATUS
3	FR2.7 ; FR2.8 ; FR2.12	✓
4	FR1.9 ; FR2.17 ; FR3.1 ; FR4.8[21] ;	✗

***Table 6** - Could Have Functional Requirement clearance*

As described in the scope there was not the need to develop any *"could have"* requirements if there was not enough time left in the internship however, from the total of 7, there was a clearance percentage of 43%. Finally as for *"Won't Have"* functional requirements none of them was implemented from a total of 6. The requirements that had this priority are: FR3.2; FR3.5; FR3.6; FR5.9; FR5.10; FR5.11.

11.3. Non-Functional Requirements

11.3.1. Scalability

11.3.1.1. Tomcat Benchmark

Apache AB is a benchmarking tool that tests web service's throughput by sending requests in bulks and expecting the response. The values for the time it took to connect, process and wait are discriminated in the results. Each node running a tomcat application server is running in a Docker container inside a virtual machine with 4 GB of RAM and using an Intel Xeon CPU E5-2650 V3 processor with 1 core at 2.3GHz.

[21] It was decided this functionality was not necessary and courses would be unique, changes would occur at an action level. Requirement got changed to could have and not implemented.

I have taken 20 measurements for each test. For each measurement the value used is the average throughput of 20.000 requests batched in groups of 200 concurrent requests. The measurements were taken from inside the virtual machine that contains the load balancer to reduce any entropy that would result of using a remote client.

The requested page was the Login Menu since there is no database operation needed for that request. The values obtained are the following:

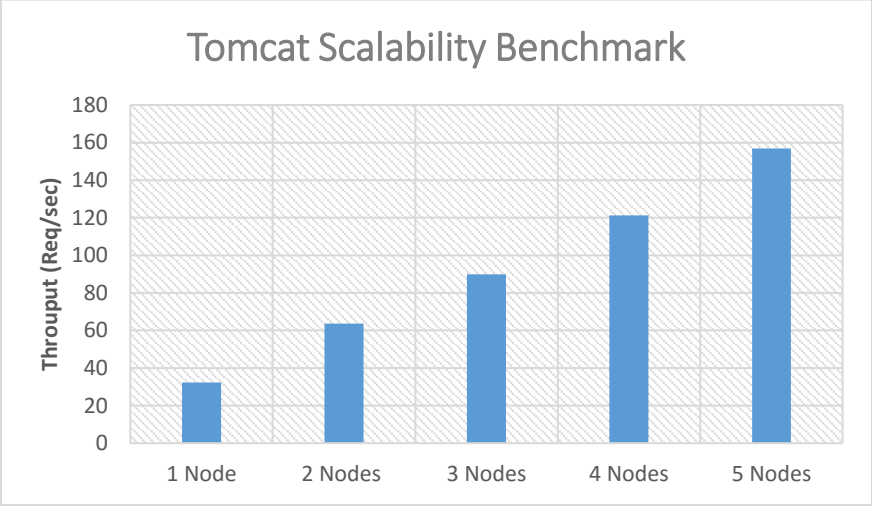


***Figure 46** - Tomcat Scalability Benchmark*

As seen in the previous image the base number of requests for a single tomcat node is only above 30 requests per second, however by adding 4 more nodes the scalability target values are achieved (125 request per second). The values used to create this chart are in the following table:

Number of Nodes	*Average(Req/sec)*	*Standard Deviation*	*Confidence Interval*
1	33.32	1.585	±0.58
2	63.69	3.122	±1.15
3	89.88	4.01	±1.47
4	121.23	5.34	±1.96
5	156.80	5.81	±2.14

***Table 7** - Application Server Scalability Benchmark Results*

As will be shown in the following benchmarks to the data layer of the application this application layer is the one bottlenecking the system in the near future and therefore the only one whose prototypes for scalability were built and benchmarked. Therefore, the system shall accommodate at least the expected 625 concurrent users and support as much as 125 requests per second. Adding more tomcat server nodes will increase the throughput and the number of active users concurrently in the system.

11.3.1.2. MySQL Benchmark

The relational database node is running in a Docker container inside a virtual machine with 4 GB of RAM and using an Intel Xeon CPU E5-2650 V3 processor with 1 core at 2.3GHz. For this benchmark 50.000 measurements were taken using Sysbench.

Populated the course the database with 100.000 entries and did measurements using a select operation to get one of those entries. The benchmarking took place it the same machine as the database to reduce any entropy caused by remote connections. The following results were obtained:

Target	*Average(ms/read)*	*Standard Deviation*	*Confidence Interval*
MySQL Database	1.33	13.32	±0.31

***Table 8 -** MySQL Read Benchmark*

Therefore, we achieve an average throughput of 751.88 selects per second which clears the scalability target set in the requirements session (375 reads per second). There was not the need to scale in this layer of the system because the only layer that can't withstand the number of requests expected in five years is the application server layer. If there was the need to scale the platform further I would use a replication strategy since the load is mostly composed of read operations.

11.3.1.3. ElasticSearch Benchmark

The non-relational database node is running in a Docker container inside a virtual machine with 4 GB of RAM and using an Intel Xeon CPU E5-2650 V3 processor with 1 core at 2.3GHz. For this benchmark 50.000 measurements were taken using a python script that would make inserts in the database.

The benchmarking took place it the same machine as the database to reduce any entropy caused by remote connections. Each measurement consists of inserting a document in the non-relational database. The following results were obtained:

Target	*Average(ms/doc)*	*Standard Deviation*	*Confidence Interval*
ElasticSearch Database	2.34	32.10	±0.89

Table 9 - *ElasticSearch Document Insert Benchmark*

Therefore we an average throughput of 751.88 selects per second which clears the scalability target set in the requirements session (250 writes per second). There was not the need to scale in this layer of the system because the only layer that can't withstand the number of requests expected in five years is the application server layer. If there was the need to scale the platform further I would use a replication strategy since the load is mostly composed of read operations.

11.3.2. Usability

11.3.2.1. Usability tests

To test the usability of the platform usability tests took place after the primary prototyping stage was completed. Usability testing is a user-centered technique to evaluate a product by having its human-computer interaction tested by real life users. This technique provides direct input on how other people, that are not familiar with the product interact with the system and how easily can they use the functionalities that have been implemented on the platform.

As a result of the nature of the project it was decided that the only heuristic that would be tested would be the average number of clicks to complete a given task. Average time to complete the task was also considered but it was dropped since it would be redundant and highly dependent on the number of wrong clicks.

Due to the lack of time only 10 people served as testers for the platform and therefore there was not enough data to undisputedly claim that the platform has indeed achieved usability as a whole due to the large confidence interval for 95%. However, the tests were successful and the results can be seen in the following graph:

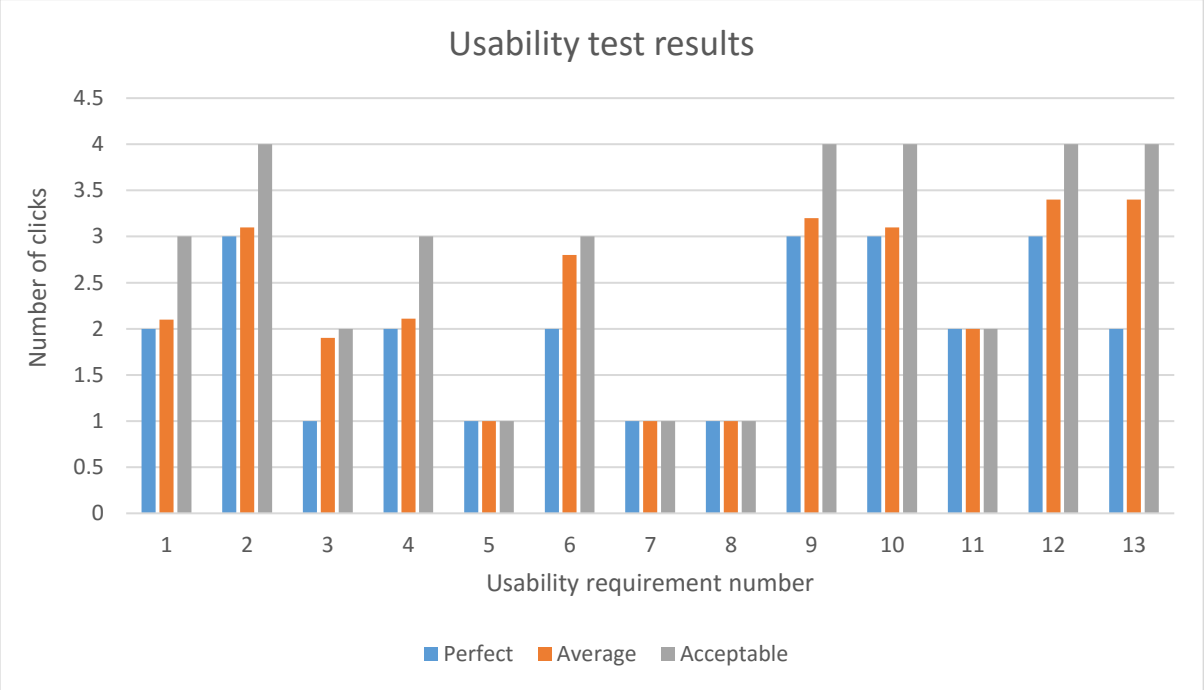


***Figure 47** - Usability test results*

As it can be seen in the previous graph, the average number of clicks for all tasks presented to the user is below the acceptable maximum number of clicks defined previously in the requirements section. The data used to create the graph can be gone over in the following table:

Usability Requirement	*Perfect Number of clicks*	*Acceptable average number of clicks*	*Average number of clicks*	*Standard deviation*	*Confidence Interval (95%)*
UR 1	2	3	2.1	0.316	± 0.196
UR 2	3	4	3.1	0.316	± 0.196
UR 3	1	2	1.9	0.876	± 0.542
UR 4	2	3	2.1	0.782	± 0.484
UR 5	1	1	1	0	± 0
UR 6	2	3	2.8	0.789	± 0.489
UR 7	1	1	1	0	± 0
UR 8	1	1	1	0	± 0
UR 9	3	4	3.2	0.422	± 0.261
UR 10	3	4	3.1	0.316	± 0.196
UR 11	2	2	2	0	± 0
UR 12	3	4	3.4	0.516	± 0.320
UR 13	2	4	3.4	1.174	± 0.728

Table 10 - *Usability tests data*

As the reader can see in the previous table usability requirements 3, 6 and 13 aren't statistically significant enough to claim that usability was successfully achieved for the 95% confidence interval. However, in the requirements section the objective was described as achieving an average bellow the acceptable number of clicks and that objective was successful for the tests that have been conducted.

11.3.2.2. Usability Questionnaire

As referred before, to test the usability of the platform all 10 users that participated in the usability tests also filled a questionnaire to determine their overall satisfaction with the system. This questionnaire is available in annex 2 and depicts a series of remarks that the tester could evaluate using 5 point Likert-scale bars in order to establish the how he felt towards the platform.

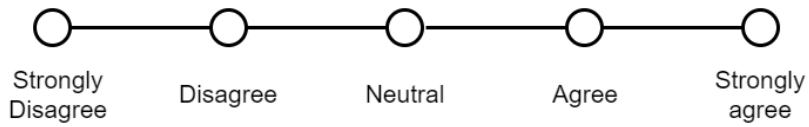


***Figure 48** - 5 point Likert-scale example*

As seen in the previous image, in a 5 point Likert-scale a value of 1 is given to strongly disagrees and the a value of 5 to strongly agrees. For usability to be considered a success each question shall require a minimum score of 3 out 5. The results can be consulted in the following graph:

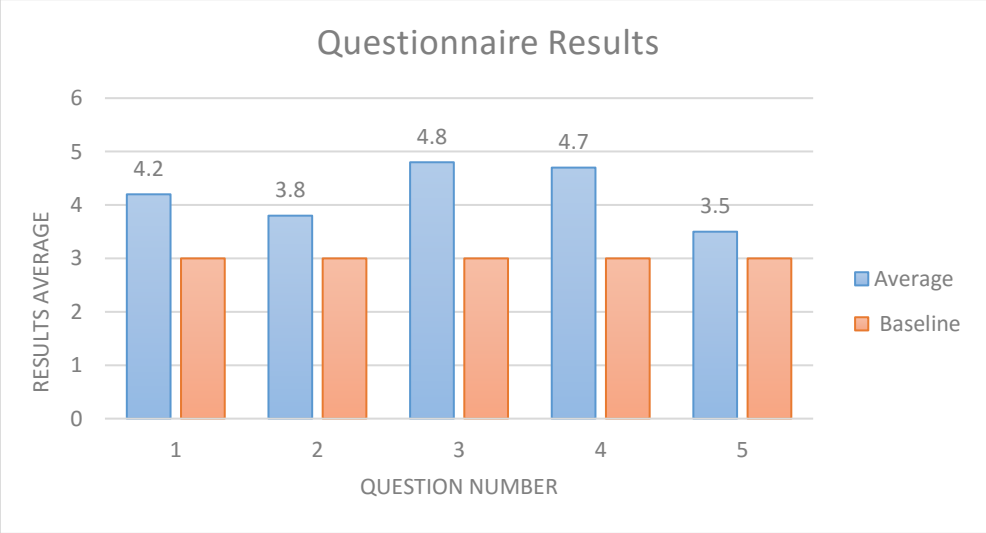


***Figure 49** - Inquiry graph results*

As it can be seen in the previous graph, the average score for all questions is above the baseline (3 out of 5). The data that was used to generate the graph presented can be seen in the following table:

Question	*Average score*	*Standard deviation*	*Confidence Interval (95%)*
1	4.2	0.876	± 0.542
2	3.8	0.788	± 0.489
3	4.8	0.421	± 0.261
4	4.7	0.483	± 0.299
5	3.5	0.707	± 0.438

***Table 11** - Data from usability questionnaire*

As in the usability tests not enough tests were executed for the results to be statistically significant for the 95% confidence interval. However, the average score was higher than the defined baseline.

Since both elements defined to study the usability of the system have been considered successful for the scope defined, usability will now be closed for the remainder of the internship. After the internship is over more tests should carried out to eliminate the existent uncertainty due to the rather low number of tests.

11.3.3 Security

In this section lies a description of how the various security requirements defined previously have been implemented in the system. First of all it is worth noting that alongside with Spring MVC, Spring Security has also been used to standardize the way some requirements were achieved.

From the security requirements described in previously only SR 21 was not implemented due to its low priority (Won't Have). In the following table lies a description of every security requirement, its completion status and how it was achieved:

11.3.4 Accountability

To achieve accountability I have implemented the ELK architecture defined previously without the usage of Logstash due to usage of custom logging. The logs are generated using the Jackson framework for Java to create JSON objects and then stored in the elastic search database.

These logs possess common dimensions that can be used to create the dashboards later on Kibana. The dimensions common to all log types are a level which is used to determine the importance of the entry and it can range from informational to error. Also each entry stores information regarding the user who generated the event and the date of when it was generated. A type dimension is also used to define which type of entry that log is since the entries can range from user login to platform failures. The facts stored depend on the entry type since each type has its own set of variables that have been defined as critical to store.

11.3.5. Interoperability

To achieve interoperability a spring based REST service was created separately from the platform to retrieve the data required by Sik. Sik can connect to this service from within the secured network of the company and request data regarding courses, user purchases or action details.

11.4 Quality Assurance

Due to the large number of functionalities inherent to WebLink it was impossible to actually develop unit tests that would cover the full functional scope of the platform. Therefore, it was decided that only critical sections of the platform would have unit tests defined and the remaining should be manually tested.

Manual testing is a quality assurance technique to find defects in which the developer plays the role of end user and tests the functionalities one by one experimenting with different input to make sure the platform outputs correctly.

The functional requirements that have been defined as critical are the ones linked directly with the overall security and functioning with the system. Unit tests were developed using Junit to make sure features such as the platform using HTTPS only and rejecting HTTP connections, making sure that after login users had the expected permissions or making sure that malformed input was not accepted.

12. What could have been done more?

12.1 Plugin System

It would have been interesting to implement a plugin system and plugin management system for the platform in which user could develop functionalities. Some platform such as Moodle implement this feature however, it would have been too much to develop single handedly alongside with the features already included in the time span of the internship.

I decided it would be interesting to document this possible future upgrade in this report though. The plugin system management platform would allow a user to create the backend and front end for features that would be incremented in the left sidebar of the platform leading to a new path/area in the platform. The coding could then be compressed and sent to a platform responsible for validating and distributing the user developed content. Any instance running WebLink could then use the aforementioned platform to get or buy new plugins.

12.2 Auto scaling

This feature would certainly have been interesting to develop for academic purposes on the scope of the scalability of the project. However, due to the limited time of the internship it wasn't developed. The clusters would scale in when certain pre-defined heurists were reached. For example when the amount of requests per second had gotten to let's say 85% of the maximum allowed for the number of nodes in production, the platform would automatically create a new node and dynamically change the load balancer so requests would also be sent to that node.

This feature would require the implementation of an orchestration manager for containers such as Kubernetes. Kubernetes in an open-source system for automating deployment, scaling and management of containerized applications. If this functionality were implemented Jenkins would only be responsible for the continuous integration of the platform relegating all continuous deployment tasks to Kubernetes instead. Also Kubernetes provides its own load balancing service

12.3 Distributed file system

As referred before for the platform to be production ready scalability wise it would have been needed the implementation of either a distributed file system or a service responsible for the storage on a centralized location. From these two options I would have picked a distributed file system because it would not only have been more interesting from an academic standpoint but also because it would improve the platforms scalability and availability greatly. Even though there was not enough time for this to be implemented this feature is recommend when the platform goes into production.

From the available technologies to achieve this I would have picked either HDFS (Hadoop file system) or GlusterFS. However, when doing my research I came up with the following benchmark comparing HDFS and GlusterFS and apparently the second performs way better than the previous one.

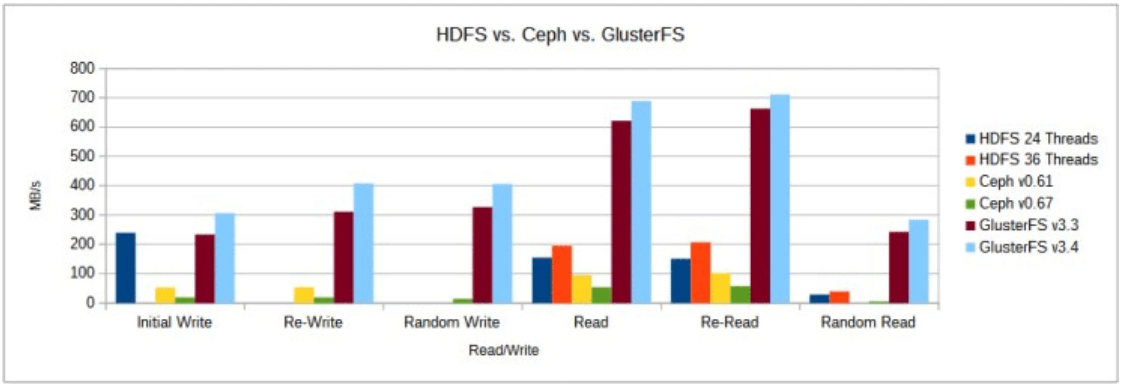


Figure 50** - HDFS vs Ceph vs GlusterFS benchmark **[13]

12.4 Accessibility Features

Implementing accessibility features would have brought enormous value to the platform especially if the platform were compliant with WCAG[22] standard. It would also have greatly increase my overall knowledge about accessibility features. However, with the already large amount of requested features there wouldn't have been time to implement this.

[22] Web Content Accessibility Guidelines

12.5 Performance and availability

Performance measures and availability should be implemented in the near future preferably before the platform actually goes into production. Even though the platform does not rely on heavy algorithmic complexity there was not enough time to tweak frontend libraries to enhance performance.

Also, availability should also be benchmarked and improved upon so there is at least 99.9% uptime. One of the measures that need to be done to achieve this has already been prototyped which redundancy of containers.

13. Conclusion

13.1. Current Standing

At this point the platform is almost production ready and it is expected for it to be in production by the start of August. There are still a few functional requirements that need to be cleared and tested for the platform to be completely ready. These requirements were not developed during the expected time due to a miscalculation regarding the time necessary to write this report and benchmark the scalability prototypes.

As for non-functional requirements, even though scalability only achieved a prototyping stage all of the remaining have to be considered a success considering the metrics defined in the start.

13.2. Future Work

During the next few weeks I will be implementing the remaining functional requirements and producing further unit test for the platform. The payment functionality, that even though implemented, is using fictional currency and therefore, details to merge with a real bank account need to be cleared.

Finally when the platform goes into production ill fix any bugs that might exist during a trial course while guiding the development team through the project developed so that they can take over when the internship is over.

13.3. Primary Issues

At the start of the internship there was some uncertainty towards exactly what was expected of the internship which slowed the drawing of the requirements. It was also one of the reasons as why RUP was chosen since there was the need to accommodate possible changes to the requirements even in late stages of development. Another issue was the

uncertainty towards which streaming methods to be used and that lasted until half of the development process.

Not having a designer working with me also greatly delayed the project since the development of the frontend was not as fast as expect due to the lack of experience on that area.

Also, the whole quality assurance part of the project was not as easy as I would expect because I had no experience whatsoever in that area and due to the large number of functionalities and features inherent to the platform it was not trivial to decide which ones should be formally tested since there was not enough time to implement tests for all.

Finally, the large scope of the project allied with the fact that I was the sole developer for all functionalities and features from scratch made it hard to complete all assigned tasks due to the limited time available even though in the end most of it was a success in my opinion.

13.4. Acquired Experience

In the end the project was without a doubt an enormous contribute towards my academic education especially in software development whose area I learned plenty of techniques necessary for the future. Also my frontend evolved from nearly inexistent to a level of proficiency that I feel comfortable with (even though this was not the primary objective of the internship).

Also studied a lot of tools necessary for operations that I had never heard of due to the lack of academic focus on this area. Finally my definition of what a real project is like also got sharper and clearer which will certainly be of benefit in the short and long term.

14. References

[1] Brown, Simon. (2015). The Art of Visualizing Software Architecture Retrieved from: https://leanpub.com/visualising-software-architecture

[2] Remenyi, Dan. (2007). 8th European Conference on e-Learning Retrieved from: https://core.ac.uk/download/files/51/66914.pdf

[3] Kroll, P. And Kruchten, P. (2003). The Rational Unified Process made easy. A Practitioner's guide to the RUP Retrieved from: http://www.temida.si/~bojan/IPIT_2014/literatura/Rational_Unified_Process_Made_Easy.pdf

[4] Cross, J. (2004). An informal history of e-learning Retrieved from: http://www.internettime.com/Learning/articles/xAn%20Informal%20History%20of%20eLearning.pdf

[5] Oracle (2014). Delivering Enterprise-class communication with WebRTC from: http://www.oracle.com/us/industries/communications/oracle-enterprise-web-rtc-wp-2132263.pdf

[6] Hauger, D. and Kock, M. (2008). The art of adaptativity in e-Learning platforms Retrieved from: http://users.informatik.uni-halle.de/~lwa07/abis07/Hauger.pdf

[7] Hogg, S. (2014, 05 26). *Software Containers: Used More Frequently than Most Realize.* Retrieved from NetworkWorld: http://www.networkworld.com/article/2226996/cisco-subnet/software-containers--used-more-frequently-than-most-realize.html

[8] Felter, W. and Ferreira, A. and Rajamony, R. and Rubio, J.(2014, 07 21). An Updated Performance Comparison of Virtual Machines and Linux Containers Retrieved from: http://domino.research.ibm.com/library/cyberdig.nsf/papers/0929052195DD819C85257D2300681E7B/$File/rc25482.pdf

[9] Maréchal, L.(2015, 04 23). MongoDB vs ElastiSearch the quest of the holy performances Retrieved from: http://domino.research.ibm.com/library/cyberdig.nsf/papers/0929052195DD819C85257D2300681E7B/$File/rc25482.pdf

[10] White, O.(2014, 08 21). The ultimate Java Build Tool Comparison: Gradle, Maven, Ant + Ivy from: http://zeroturnaround.com/rebellabs/java-build-tools-part-2-a-decision-makers-comparison-of-maven-gradle-and-ant-ivy/

[11] White, O.(2015, 01 06). Top 4 Java Web Frameworks Revealed; Real Life Usage Data of Spring MVC, Vaadin, GWT and JSF from: http://zeroturnaround.com/rebellabs/top-4-java-web-frameworks-revealed-real-life-usage-data-of-spring-mvc-vaadin-gwt-and-jsf/

[12] Model, M.(2004, 10 13). Model View Controller History retrieved from: http://zeroturnaround.com/rebellabs/top-4-java-web-frameworks-revealed-real-life-usage-data-of-spring-mvc-vaadin-gwt-and-jsf/

[13] XRayMachines.(2015, 01). GlusterFS vs Cepth vs HDFS retrieved from: http://www.xraymachines.info/article/794938075/glusterfs-vs-ceph-vs-hdfs/

[14] Weiss, T.(2013, 11 20). We analysed 30.000 GitHub Projects - Here are the top 100 Libraries in Java, JavaScript and Ruby retrieved from: http://blog.takipi.com/we-analyzed-30000-github-projects-here-are-the-top-100-libraries-in-java-js-and-ruby/

[15] Provos, T. and Maziéres, D. (1999, 06 11) A Future-Adaptable Password Scheme retrieved from: https://www.usenix.org/legacy/event/usenix99/provos/provos.pdf

[16] Cristofaro, F. (2013) The exponential nature of password cracking costs retrieved from: http://resources.infosecinstitute.com/the-exponential-nature-of-password-cracking-costs/

Annex 1 – Functional Requirements: Use Cases

Class 1 - Abstract User

ID	FR 1.1
Title	Registration
Use Case Diagram	Diagram 1
Description	The User shall be able to register to the online platform. This registration shall be free.
Dependencies	Null
Input	• Complete name • Email • Password • Date of birth • Nationality • Address • Postal code
Output	• New Account • Confirmation email
Exceptions	• Email already used • Passwords or emails do not match • Wrong input type • Malformed email • Empty Fields • Invalid Date of Birth
Priority	Must Have

Table 12 - *FR1.1: Registration*

ID	FR 1.2
Title	Login
Use Case Diagram	Diagram 1
Description	Any registered user shall be able to login to the platform for as long as he owns an account.
Dependencies	FR 1.4
Input	• Email • Password
Output	Login Confirmation

Exceptions	• Wrong Password • Account does not exist
Priority	Must Have

Table 13 – *FR 1.2: Login*

ID	FR 1.3
Title	Password Recovery / Reset
Use Case Diagram	Diagram 1
Description	Any registered user shall be able to recover his password if he doesn't remember it.
Dependencies	FR 1.4
Input	• Email • New Password
Output	Password updated
Exceptions	• Invalid Password • Account does not exist
Priority	Must Have

Table 14 - *FR 1.3: Password Recovery / Reset*

ID	FR 1.4
Title	Email Confirmation
Use Case Diagram	Diagram 1
Description	The User shall be able to confirm his email upon registration.
Dependencies	FR 1.1
Input	Click an Email Link
Output	Email Confirmed
Exceptions	• Invalid Link • Account does not exist
Priority	Must Have

Table 15 - *FR 1.4: Email Confirmation*

ID	FR 1.5
Title	Course list browse
Use Case Diagram	Diagram 1

Description	The User shall be able to browse the available courses.
Dependencies	Null
Input	Null
Output	Course List
Exceptions	• No Course available
Priority	Must Have

Table 16 - *FR 1.5: Course List Browse*

ID	FR 1.6
Title	Filter course list
Use Case Diagram	Diagram 1
Description	The User shall be able to filter the list of available courses.
Dependencies	Null
Input	• Theme • Price • Start date
Output	Filtered course List
Exceptions	• No Course available • Malformed attribute selection • Selected attribute is non-existent
Priority	Should Have

Table 17 - *FR 1.6: Filter course List*

ID	FR 1.7
Title	See details of a course
Use Case Diagram	Diagram 1
Description	The User shall be able to see the details of a course.
Dependencies	Null
Input	• Course
Output	• Price • Description • Video • Module Listing • Language

Exceptions	• Course does not exist • Missing attributes
Priority	Must Have

Table 18 - *FR 1.7: See Course Details*

ID	FR 1.8
Title	Browse about page
Use Case Diagram	Diagram 1
Description	The User shall be able to browse the about page.
Dependencies	Null
Input	Null
Output	• What this platform is • Who is responsible for this platform • Other information
Exceptions	• Missing attributes
Priority	Should have

Table 19 - *FR 1.8: Browse about Page*

ID	FR 1.9
Title	Idiom Change
Use Case Diagram	Diagram 1
Description	The User shall be able to change the idiom of the platform.
Dependencies	Null
Input	Selected Language
Output	• Platform changes language
Exceptions	• Selected language is not available
Priority	Could Have

Table 20 - *FR 1.9: Idiom Change*

Class 2 - Registered User

ID	FR 2.1
Title	Friend Invite
Use Case Diagram	Diagram 2
Description	Any registered user shall be able to add other users, known to him, as friends.
Dependencies	FR 1.2
Input	User to be added
Output	Friend Request
Exceptions	Invalid User
Priority	Should have

Table 21 - *FR 2.1: Friend Invite*

ID	FR 2.2
Title	Accept friend Request
Use Case Diagram	Diagram 2
Description	Any registered user shall be able to respond to friend requests with a positive answer
Dependencies	FR 2.1
Input	Positive Answer
Output	New friend
Exceptions	Invalid request
Priority	Should Have

Table 22 - *FR 2.2: Accept Friend Request*

ID	FR 2.3
Title	Reject friend Request
Use Case Diagram	Diagram 2
Description	Any registered user shall be able to respond to friend requests with a negative answer
Dependencies	FR 2.1
Input	Negative Answer
Output	Null
Exceptions	Invalid request
Priority	Should Have

Table 23 - *FR 2.3: Reject Friend Request*

ID	FR 2.4
Title	Remove friend
Use Case Diagram	Diagram 2
Description	Any registered user that owns friends in the platform shall be able to remove them
Dependencies	FR 2.2
Input	Friend to remove
Output	Confirmation
Exceptions	Invalid friend
Priority	Should Have

***Table 24** - FR 2.4: Remove friend*

ID	FR 2.5
Title	Change personal information
Use Case Diagram	Diagram 2
Description	Any registered user shall be able to change the myriad of information inherent to his account
Dependencies	FR 1.2
Input	• Complete Name • Email • Password • Date of birth • Nationality • Address • Postal code • Avatar
Output	Change confirmation
Exceptions	• Password too small • Invalid date of birth • Invalid file upload for avatar change • Malformed Email • Passwords don't match • Invalid postal code
Priority	Must Have

***Table 25** - FR 2.5: Change personal information*

ID	FR 2.6
Title	Consult event calendar
Use Case Diagram	Diagram 2
Description	Any registered user shall be able to consult the calendar with the events from the platform
Dependencies	FR 1.2
Input	Null
Output	Calendar
Exceptions	Null
Priority	Must Have

Table 26 - *FR 2.6: Consult Event Calendar*

ID	FR 2.7
Title	Chat: Send
Use Case Diagram	Diagram 2
Description	Any registered user shall be able to send text in real time to a friend
Dependencies	FR 2.2
Input	• Text • Friend
Output	Text Message
Exceptions	• Text is Null • Friend Non-existent
Priority	Could have

Table 27 *- FR 2.7: Chat: Send*

ID	FR 2.8
Title	Chat: Receive
Use Case Diagram	Diagram 2
Description	Any registered user shall be able to receive text messages directed to him
Dependencies	FR 1.2
Input	Null
Output	Text Message
Exceptions	Null
Priority	Could Have

Table 28 - *FR 2.8: Chat: Receive*

ID	FR 2.9
Title	Browse action List
Use Case Diagram	Diagram 3
Description	Any registered user shall be able to browse the list of actions that will take place
Dependencies	FR 1.2
Input	Course
Output	Action List
Exceptions	• Course is inexistent • No action is available
Priority	Must Have

Table 29 - *FR 2.9: Browse action list*

ID	FR 2.10
Title	See Action Details
Use Case Diagram	Diagram 3
Description	Any registered user shall be able to see the details inherent to an action
Dependencies	FR 1.2
Input	Action
Output	• End / Start Date • Evaluation method • Who is teaching • Event calendar
Exceptions	• Inexistent action • Missing attributes
Priority	Must Have

Table 30- *FR 2.10: See action details*

ID	FR 2.11
Title	Buy Action
Use Case Diagram	Diagram 3
Description	Any registered user shall be able to use his credit card to buy actions that haven't started yet.
Dependencies	FR 1.2
Input	• Action • Account Number • CC Number • Card Expiration date

	• Name associated to the account
Output	New modules of an action become available
Exceptions	• Inexistent action • Wrong credit card information • Not enough funds • Wrong number of modules
Priority	Must Have

Table 31 *- FR 2.11: Buy action*

ID	FR 2.12
Title	Offer action
Use Case Diagram	Diagram 3
Description	Any registered user shall be able to an action before it starts to a friend.
Dependencies	• FR 2.2 • FR 2.12
Input	• Action • Friend
Output	• Send Confirmation • Redeem code
Exceptions	• Inexistent action • Action has already started • Inexistent Friend • Invalid number of modules
Priority	Could Have

Table 32 *- FR 2.12: Offer action*

ID	FR 2.13
Title	Completed Details
Use Case Diagram	Diagram 3
Description	Any registered user shall be able to see the details on a completed course
Dependencies	FR 2.17
Input	Completed Course
Output	• Grade • Grade per module List • Completion date

	• Pass / Fail
Exceptions	• Course does not exist • Attributes missing
Priority	Must have

***Table 33** - FR 2.13: Completed Details*

ID	FR 2.14
Title	Be graded in course
Use Case Diagram	Diagram 3
Description	Any registered user shall be grade upon course completion
Dependencies	FR 1.2
Input	Completed Course
Output	• Grade • Grade per module List • Completion date • Pass / Fail
Exceptions	• Course does not exist • Attributes missing
Priority	Must Have

***Table 34** - FR 2.14: Be Graded in Course*

ID	FR 2.15
Title	Re-try Failure
Use Case Diagram	Diagram 3
Description	Any registered user shall be able to re-try a course he failed
Dependencies	FR 2.17
Input	Completed Course
Output	• Account Number • CC Number • Card Expiration date • Name associated to the account
Exceptions	• Action does not exist • Wrong credit card information • Insufficient Funds
Priority	Could Have

Table 35 - *FR 2.15: Re-Try Failure*

ID	FR 2.16
Title	Share Success
Use Case Diagram	Diagram 3
Description	Any user that successfully completed a course shall be able to post it on Facebook
Dependencies	FR 2.17
Input	• Completed Course • Facebook Login
Output	Facebook Post
Exceptions	• Facebook login failure • Inexistent or failed course
Priority	Should Have

Table 36 - *FR 2.16: Share Success*

ID	FR 2.17
Title	Certificate Request
Use Case Diagram	Diagram 3
Description	Any user that successfully completed a course shall be able to request a certificate that is sent using CTT.
Dependencies	FR 2.17
Input	Completed Course
Output	Request
Exceptions	Inexistent or failed course
Priority	Must Have

Table 37 - *FR 2.17: Certificate Request*

ID	FR 2.18
Title	Send Certificate
Use Case Diagram	Diagram 3
Description	Konkrets, Lda shall send certificates upon requests
Dependencies	FR 2.20
Input	Request
Output	Physical certificate
Exceptions	No Request
Priority	Must Have

Table 38 - *FR 2.18: Send Certificate*

Class 3 – Admin & Master Admin

ID	FR 3.1
Title	Modify Company Logo
Use Case Diagram	Diagram 4
Description	A user of class Master Admin shall be able to change the logo of the company
Dependencies	FR 1.2
Input	New Logo
Output	Logo Changes
Exceptions	Invalid Logo
Priority	Could have

Table 39 - *FR 3.1: Modify Company Logo*

ID	FR 3.2
Title	Modify Platform Theme
Use Case Diagram	Diagram 4
Description	A user of class Master Admin shall be able to change the theme of the platform
Dependencies	FR 1.2
Input	Selected Theme
Output	Theme Changes
Exceptions	Invalid Theme
Priority	Won't Have

Table 40 - *FR 3.2: Modify Platform Theme*

ID	FR 3.3
Title	Add Sponsor
Use Case Diagram	Diagram 4
Description	A user of class Master Admin shall be able to add new sponsors to the platform
Dependencies	FR 1.2
Input	New Sponsor Logo
Output	Updated Sponsor List
Exceptions	Invalid Sponsor
Priority	Should Have

Table 41 - *FR 3.3: Add Sponsor*

ID	FR 3.4
Title	User Promotion
Use Case Diagram	Diagram 4
Description	A user of class Admin or superior shall be able to promote users to any permission status up to admin.
Dependencies	FR 1.2
Input	• User • New permission level
Output	User gets new permission level
Exceptions	• Invalid User • Invalid Permission Level • Not Enough permissions to promote
Priority	Must Have

Table 42 - *FR 3.4: User Promotion*

ID	FR 3.5
Title	External Application
Use Case Diagram	Diagram 4
Description	Any registered user shall be able to apply for an external coordinator position
Dependencies	FR 1.2
Input	Resume
Output	Application
Exceptions	• Invalid Resume
Priority	Won't Have

Table 43 - *FR 3.5: External Application*

ID	FR 3.6
Title	Respond External application
Use Case Diagram	Diagram 4
Description	A user of class Master admin shall be able to accept or reject external coordinator applications
Dependencies	FR 3.5
Input	Yes or No
Output	New coordinator or Null
Exceptions	Invalid Application

Priority	Could Have

Table 44 – *FR 3.6: External Application Response*

ID	FR 3.7
Title	Be Promoted
Use Case Diagram	Diagram 4
Description	Any Registered user shall be able to be promoted
Dependencies	FR 3.3
Input	Null
Output	New permissions
Exceptions	Invalid Permission
Priority	Must Have

Table 45 - *FR 3.7: Be Promoted*

Class 4 – Coordinator & Instructor

ID	FR 4.1
Title	Course Creation
Use Case Diagram	Diagram 5
Description	A user of class Coordinator or superior shall be able to create a new course
Dependencies	FR 3.7
Input	• Theme • Duration • Price • Language • Course Type (Asynchronous, Synchronous or B-Learning)
Output	New Course
Exceptions	Invalid Inputs
Priority	Must Have

Table 46 - *FR 4.1: Course Creation*

ID	FR 4.2
Title	Create Module

Use Case Diagram	Diagram 5
Description	A user of class Coordinator or superior shall be able to create a new modules for an existing course
Dependencies	FR 4.1
Input	• Theme • Description • Duration • Evaluation Percentage
Output	Module
Exceptions	Invalid Inputs
Priority	Must Have

Table 47 - *FR 4.2: Create Module*

ID	FR 4.3
Title	Create Action
Use Case Diagram	Diagram 5
Description	A user of class Coordinator or superior shall be able to create an action for a course
Dependencies	FR 4.1
Input	• Start Date • Discount • Instructor for each module
Output	New Action
Exceptions	• Invalid Inputs • Inexistent Instructor
Priority	Must Have

Table 48 - *FR 4.3: Create Action*

ID	FR 4.4
Title	Confirm Action
Use Case Diagram	Diagram 5
Description	A user of class Coordinator or superior shall be able to say an action is ready to start at the predicted time
Dependencies	FR 4.3
Input	Null
Output	Action becomes ready and available for subscription

Exceptions	Invalid Action
Priority	Must Have

***Table 49** - FR 4.4: Confirm Action*

ID	FR 4.5
Title	Delete Course
Use Case Diagram	Diagram 5
Description	A user of class Coordinator or superior shall be able to delete existing courses
Dependencies	FR 4.1
Input	Course
Output	Confirmation
Exceptions	Invalid Course
Priority	Could Have

***Table 50** - FR 4.5: Delete Course*

ID	FR 4.6
Title	Delete Action
Use Case Diagram	Diagram 5
Description	A user of class Coordinator or superior shall be able to delete existing actions
Dependencies	FR 4.2
Input	Action
Output	Confirmation
Exceptions	Invalid Action
Priority	Must Have

***Table 51** - FR 4.6: Delete Action*

ID	FR 4.7
Title	Delete Module
Use Case Diagram	Diagram 5
Description	A user of class Coordinator or superior shall be able to delete modules of an existing course
Dependencies	FR 4.3
Input	Module

Output	Confirmation
Exceptions	Invalid Module
Priority	Must Have

Table 52 - *FR 4.7: Delete Module*

ID	FR 4.8
Title	Alter Course
Use Case Diagram	Diagram 5
Description	A user of class Coordinator or superior shall be able to change the attributes of an existing course
Dependencies	FR 4.1
Input	• Theme • Duration • Price • Language • Course Type (Asynchronous, Synchronous or B-Learning)
Output	Updated Course
Exceptions	Invalid Inputs
Priority	Could Have

Table 53 - *FR 4.8: Alter Course*

ID	FR 4.9
Title	Alter Action
Use Case Diagram	Diagram 5
Description	A user of class Coordinator or superior shall be able to change the attributes of an existing action
Dependencies	FR 4.3
Input	• Start Date • Discount • Instructor for each module
Output	Updated Action
Exceptions	Invalid Inputs
Priority	Must Have

Table 54 - *FR 4.9: Alter Action*

ID	FR 4.10
Title	Alter Module
Use Case Diagram	Diagram 5
Description	A user of class Coordinator or superior shall be able to change the attributes of an existing module
Dependencies	FR 4.2
Input	• Theme • Description • Duration • Evaluation Percentage
Output	Updated Module
Exceptions	Invalid Inputs
Priority	Must Have

Table 55 - *FR 4.10: Alter Module*

Class 5 – Instructor & Student

ID	FR 5.1
Title	Create Quiz
Use Case Diagram	Diagram 6
Description	A user of class Instructor or above that is assigned to modules of an action, shall be able to create interactive quizzes.
Dependencies	FR 4.3
Input	• Date • Percentage of the module • Module
Output	Empty Quiz
Exceptions	• Module does not exist • Invalid inputs
Priority	Must Have

Table 56 *- FR 5.1: Create Quiz*

ID	FR 5.2
Title	Create question for a quiz
Use Case Diagram	Diagram 6

Description	A user of class Instructor or above that is assigned to modules of an action, shall be able to create questions, for previously created quizzes
Dependencies	FR 5.1
Input	• Type • Content • Percentage of the quiz grade • Quiz
Output	Updated quiz
Exceptions	• Invalid Inputs • Nonexistent quiz
Priority	Must Have

Table 57 - *FR 5.2: Create Question for quiz*

ID	FR 5.3
Title	Grade a quiz question
Use Case Diagram	Diagram 6
Description	A user of class Instructor or above that is assigned to modules of an action, shall be able to grade questions from quizzes with open answers
Dependencies	FR 5.4
Input	• Solved Quiz • Percentage
Output	Graded quiz
Exceptions	• Invalid Inputs • Nonexistent quiz
Priority	Must Have

Table 58 - *FR 5.3: Grade a quiz question*

ID	FR 5.4
Title	Solve a quiz
Use Case Diagram	Diagram 6
Description	Any user subscribed to an action shall be able to solve the quizzes made available
Dependencies	• FR 5.1 • FR 2.3
Input	• Answers • Quiz
Output	Solved quiz
Exceptions	• Invalid Inputs

	• Nonexistent quiz
Priority	Must Have

Table 59 - *FR 5.4: Solve a Quiz*

ID	FR 5.5
Title	Create Homework
Use Case Diagram	Diagram 6
Description	A user of class Instructor or above that is assigned to modules of an action, shall be able to create homework assignments that require the delivering of a file.
Dependencies	FR 4.3
Input	• Statement • Due Date • Percentage of the module • Module
Output	Homework
Exceptions	• Invalid Input • Invalid Module
Priority	Should Have

Table 60 - *FR 5.5: Create Homework*

ID	FR 5.6
Title	Grade homework
Use Case Diagram	Diagram 6
Description	A user of class Instructor or above that is assigned to modules of an action, shall be able to grade delivered homework.
Dependencies	FR 4.3
Input	• Solved Homework • Percentage of the homework
Output	Graded Homework
Exceptions	• Invalid Inputs • Nonexistent homework
Priority	Should Have

Table 61 *- FR 5.6: Grade homework*

ID	FR 5.7
Title	Upload Materials
Use Case Diagram	Diagram 7
Description	A user of class Instructor or above that is assigned to modules of an action, shall be able to upload support material
Dependencies	FR 4.3
Input	• File • Module
Output	Support material File
Exceptions	• No such file • Inexistent module
Priority	Must Have

***Table 62** - FR 5.7: Upload Materials*

ID	FR 5.8
Title	Download Materials
Use Case Diagram	Diagram 7
Description	Any user subscribed to an action shall be able to download the previously upload materials
Dependencies	• FR 5.8 • FR 2.3
Input	• Support Material File • Module
Output	File
Exceptions	• No such file • No such module
Priority	Must Have

***Table 63** - FR 5.8: Download Materials*

ID	FR 5.9
Title	Create forum topic
Use Case Diagram	Diagram 7
Description	Any user of class instructor or superior assigned to an Action shall be able to create a forum topic
Dependencies	FR 4.3
Input	• Action • Topic Name
Output	Topic

Exceptions	• Nonexistent action • Invalid topic name
Priority	Won't Have

***Table 64** - FR 5.9: Create forum topic*

ID	FR 5.10
Title	Create forum thread
Use Case Diagram	Diagram 7
Description	Any user subscribed to modules of an action shall be able to create a thread on the action forum under a previously created topic
Dependencies	• FR 5.10 • FR 2.3
Input	• Topic • Thread Name • Text
Output	Thread
Exceptions	• Nonexistent Topic • Invalid Name or text
Priority	Won't have

***Table 65** - FR 5.10: Create forum thread*

ID	FR 5.11
Title	Create forum post
Use Case Diagram	Diagram 7
Description	Any user subscribed to modules of an action shall be able to create a post on the action forum under a previously created thread
Dependencies	FR 5.11
Input	• Thread • Text
Output	Post
Exceptions	• Nonexistent Thread • Invalid Text
Priority	Won't Have

***Table 66** - FR 5.11: Create forum post*

ID	FR 5.12
Title	Start a class
Use Case Diagram	Diagram 7
Description	Any user of class instructor or superior shall be able to start a class of an action
Dependencies	FR 4.3
Input	• Action • Text Messages
Output	• Streams • Text Messages
Exceptions	• Invalid Action • Broken Streams • Wrong text messages
Priority	Must Have

Table 67 - *FR 5.12: Start a class*

ID	FR 5.13
Title	Join a class
Use Case Diagram	Diagram 7
Description	Any user subscribed to an action shall be able to join a class that has already started
Dependencies	FR 2.3
Input	• Action • Class • Text Messages • Streams
Output	Text Messages
Exceptions	• Invalid Action • Broken Streams • Wrong text messages
Priority	Must Have

Table 68 - *FR 5.13: Join a class*

ID	FR 5.14
Title	Join Conference Room
Use Case Diagram	Diagram 7
Description	Any user subscribed to an action shall be able to join the conference room associated with that action.
Dependencies	• FR 5.13 • FR 5.14
Input	• Class • Student
Output	Null
Exceptions	• Invalid Class • Invalid Student
Priority	Should Have

Table 69 – *FR 5.14: Expel a student*

ID	FR 5.15
Title	Leave a class
Use Case Diagram	Diagram 7
Description	Any user subscribed to an action shall be able to leave a class that has already started
Dependencies	FR 5.14
Input	Null
Output	Leaving class
Exceptions	Null
Priority	Must Have

Table 70 - *FR 5.15: Leave a class*

ID	FR 5.16
Title	Present materials
Use Case Diagram	
Description	Any user subscribed to an action shall be able to view videos and audio without having to download the content
Dependencies	FR 5.9
Input	Video or audio
Output	Stream
Exceptions	No content found
Priority	Should Have

Table 71 - *FR5.16: Present Materials*

ID	FR 5.17
Title	Change Steam Input
Use Case Diagram	
Description	Any user teaching a class shall be able to change the input of the stream between webcam and screen.
Dependencies	FR 5.16
Input	Webcam or Screen
Output	Stream
Exceptions	Invalid input
Priority	Should Have

Table 72 - *FR5.17: Change Stream Input*

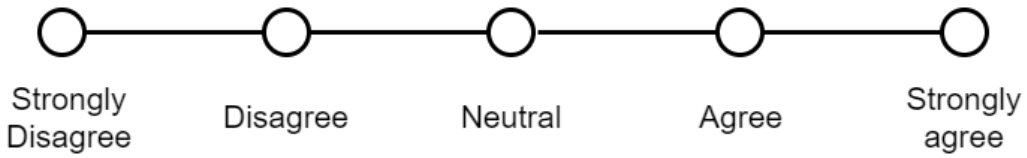
Annex 2: Usability Questionnaire

1. Introduction

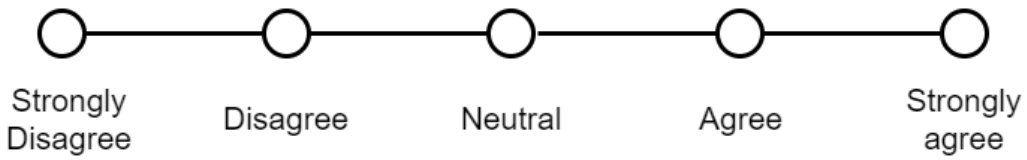
This questionnaire was devised in order to determine user satisfaction when using the platform WebLink. It is composed of remarks followed by 5 point Likert-scale bars that shall be answered by the inquired. The goals of this usability test include establishing a baseline of user performance and validating the human-interface interaction that was implemented. Tester identity shall remain anonyms and the answer to the questions shall only be presented in a form of statistical analysis during the defense of the internship.

2. Questionnaire

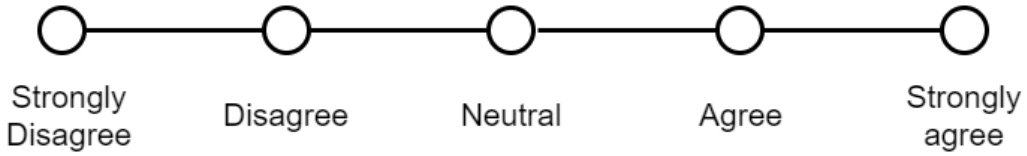
1. The platform is easy to navigate in. The menu tabs are easy to understand.



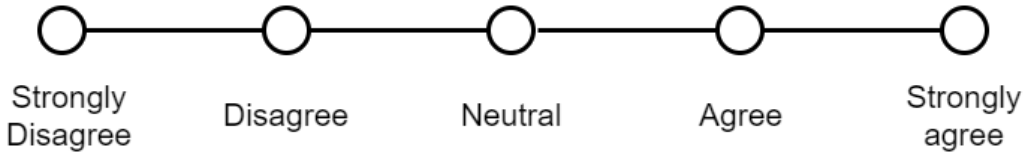
2. The proposed tasks were easy to achieve.



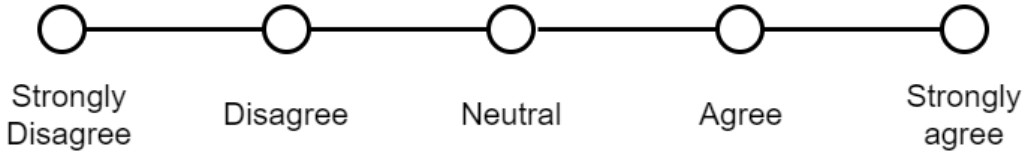
3. Classroom and conference call rooms were easy to find and utilize.



4. Whenever a mistake inputting data happened you were clarified about what that mistake was.



5. As a student it was easy to enroll in a course and access its materials.



Annex 3: Security Requirements

	Description	*Priority*
SR 1	Any connection from a client to the platform shall only be possible using a HTTPS connection to provide a reasonable protection against some man-in-the-middle attacks such as active eavesdropping or tampering.	M
SR 2	Database queries shall be protected against the usage of SQL injection.	M
SR 3	Each user input shall be checked to make sure if the request is not malformed both server and client-side.	M
SR 4	Input shall be checked both server and client-size for null values and size (make sure it is between expected length bounds) to prevent buffer overflow attacks.	M
SR 5	Input variable types shall also be checked both client and server side to prevent any errors or abuses.	M
SR 6	Input shall be parsed for content whenever that is possible. Value ranges must be checked, and inputs such as emails shall be parsed to make sure the string is within the expected format.	M
SR 7	The platform shall provide an authentication service using a username and a password.	M
SR 8	The platform shall encrypt the password before storing it in the database using a randomly generated salt hash.	M
SR 9	All connections within the platform to other system such as databases or external system shall be done using a HTTPS channel.	M
SR 10	Each Http session shall have a set lifetime of 30min of inactivity before expiring to prevent session overflow.	C
SR 11	The platform shall implement different levels of authorization therefore it shall implement an authorization clearance service upon authentication.	M
SR 12	Each request shall have a CSRF generated token to prevent cross-site-request-forgery	S

SR 13	A user shall not be able to authenticate more than one time simultaneously using the same login information either by removing previously existent connections, or preventing a new login.	S
SR 14	Any user input that may contain sensitive information (such as passwords) shall be replaced by symbols during input (for example bullet points) to prevent identity theft using social engineering techniques.	S
SR 15	Http errors 400, 403, 404 and 405 shall be escaped and the user provided with a custom page that reveals no sensitive information regarding the error.	C
SR 16	Multiple authentication failures shall block the IP of the caller for 5min before he can authenticate again in the platform to prevent brute force attacks.	C
SR 17	Upon registration the user shall be presented with a reCaptcha to make it harder for automated registrations in the platform and subsequent abuses.	M
SR 18	The host of the platform shall block users by IP that try and fail 5 times to SSH the virtual machine in which the platform is running.	C
SR 19	Any file input by users shall be checked and not accepted if the extension is not known to the platform or its size surpasses the defined maximum allowed (500 Megabytes).	M
SR 20	The platform shall not accept weak passwords (required to have at least 6 characters and include at least a letter, a capitalized letter and a non-capitalized letter).	S
SR 21	If a user tries to access unauthorized content, after three attempts within a set period of time (1 hour) administrators shall be notified via email, so that the matter can be looked into.	W

Table 73 - *Security Requirements Specification*

Annex 4: Usability Requirements

ID	*Description*	*Maximum Number of Average Clicks*
UR 1	Login starting in homepage	3
UR 2	Register starting in homepage	4
UR 3	Browse Courses starting in /weblink	2
UR 4	Check who the instructors are for module 1 of the app development course starting in course listing	3
UR 5	Buy the course app development starting in course information	1
UR 6	Enter bought course starting in /weblink	3
UR 7	Enter classroom starting inside course	1
UR 8	Enter conference room starting inside course	1
UR 9	View video entitled big buck bunny starting inside course	4
UR 10	Download PDF entitled test.pdf starting inside course	3
UR 11	Check Calendar for quizzes starting inside course	3
UR 12	Request certificate for course starting from /weblink	4
UR 13	Enter a new avatar for your profile starting from /weblink	4

Table 74 - *Usability Requirements Specification*

Annex 5: Quality Attributes

QUALITY ATTRIBUTE	ATTRIBUTE REFINEMENT	ASR
SCALABILITY	Tomcat Horizontal Scalability	The system shall be able to support at least 125 request per second. The system shall be prepared to expand further if needed.
	MySQL Replication	MySQL shall be able to handle at least 375 reads per second.
	ElasticSearch Horizontal Scalability	ElasticSearch Shall support at least 250 inserts per second.
ACCOUNTABILITY	Logging	The system shall keep logs of all the actions done by each user and the different states of the platform whenever a change occurs.
	Log Analysis	The system shall implement a solution that facilitates the analysis of logs generated by the application and creates dashboards to display that information.
INTEROPERABILITY	Sik Integration	The System shall provide integration points for Sik Webforma to be able to connect and get information from the platform.
	Email Integration	The platform shall be integrated with an email service provider so that the application can send automatic emails.
	Facebook Integration	The system shall integrate with Facebook. Allowing users to post courses they have finished in their wall.

	TokBox Integration	The system shall integrate with TokBox to use it as a streaming server.
	Payment Integration	The system shall integrate with a payment service for payment authentication and transaction management of funds.
SECURITY	Secure Sockets	The system shall use HTTPS to provide a more secure connection interface and prevent eavesdropping.
	User Authorization	The System Shall validate user permissions upon request
	Input Validation	Every user input data validation shall be previously checked Client and Server side.

Table 75 - *Utility Tree - Quality attributes*

Annex 6: Security Requirements Validation

	Solution	*Priority*	*Clearance*
SR1	To implement this requirement SSL certificates have been created using letsEncrypt as certificate Authority. When a request to port 8080 is received the platform redirects the client to the port 8443.	M	✓
SR2	Parameters aren't concatenated with HQL queries directly. By using the function *.setParameter()* potentially dangerous symbols are escaped making the platform SQL Injection proof.	M	✓
SR3	This has been implement server side using a validator layer and client side by doing verifications using JavaScript beforehand.	M	✓
SR4	Every input is checked for these properties both client and server side.	M	✓
SR5	Every input type is verified both client and server side.	M	✓
SR6	Every input is checked for these properties both client and server side whenever possible.	M	✓
SR7	The platform does provide authentication using a Login menu. Requesting a user name and a password.	M	✓
SR8	Spring security was used to salt the password using bcrypt.	M	✓
SR9	All the connections to the outside of the platform are also done using HTTPS	M	✓
SR10	Spring MVC allows the creation of listeners that detect when a session is created. These sessions were given an expiration date of 30min.	M	✓
SR11	Spring security implements a standard for authorization and authentication that has been used to deal with this.	M	✓
SR12	Whenever a request is done, a new token is created to prevent cross-site-request-forgery.	S	✓
SR13	Spring Security authentication provider keeps a list of the active user sessions. This list is parsed and if the user tries to	S	✓

	authenticate with an already signed in account, the previous one shall be kicked out.		
SR14	Using Html password input type to complete this.	S	✓
SR15	HTTP errors are caught in the application configuration file and custom views were developed.	C	✓
SR16	Spring security keeps a log of every authentication attempt. Using this information it is possible to block users that have tried too many times to login. The IP of the caller is stored within the session object.	S	✓
SR17	Google reCaptcha was used to achieve this on registration page.	S	✓
SR18	IPtables were used to achieve this.	C	✓
SR19	There is a list of files that can be accepted by the platform which include mp3, webm, png, jpg, pdf and zip. If the file doesn't have one of these extensions it will not be accepted.	M	✓
SR21	Implemented.	S	✓
SR22	Not Implemented.	W	✗

***Table 76** - Security Requirements Clearance*