ANALYSIS, DESIGN AND IMPLEMENTATION OF A WEB PROTOTYPE OF SOCIAL NETWORKING FOR STUDENTS

Ana Belén García Cívico
EXAM WORK 2012
Computer Engineering

<table>
<thead>
<tr>
<th>Postadress:</th>
<th>Besöksadress:</th>
<th>Telefon:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Box 1026</td>
<td>Gjuterigatan 5</td>
<td>036-10 10 00 (vx)</td>
</tr>
<tr>
<td>551 11 Jönköping</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
This exam work has been carried out at the School of Engineering in Jönköping in the subject area Computer and Electrical Engineering. The work is a part of the three-year Bachelor of Science in engineering programme. The authors take full responsibility for opinions, conclusions and findings presented.

Examiner:

Supervisor: Magnus Schoultz

Scope: 15 credits (first cycle).

Date:
Abstract

The idea would be to unify it into a system where people from different countries can provide information on their hometowns, helping people who are interested in them. The system created is a collaborative Web system in which people can access and edit contents by entering information about their hometowns, and this information will help other people who decide to move to the different cities.
Keywords

Usability
Model-view-controller
Java
Social Networking
Students
Cities
Contents

1 Introduction ........................................................................................................... 7
  1.1 BACKGROUND .............................................................................................. 7
  1.2 PURPOSE AND RESEARCH QUESTIONS .................................................. 8
  1.3 DELIMITATIONS .......................................................................................... 8
  1.4 OUTLINE ..................................................................................................... 9

2 Theoretical background ....................................................................................... 10
  2.1 CONCEPT OF UsABILITY ......................................................................... 10
  2.2 JAVA ........................................................................................................... 10
  2.3 JSP .............................................................................................................. 11
  2.4 CSS ............................................................................................................ 11
  2.5 JPA ............................................................................................................. 12
  2.6 ORACLE ...................................................................................................... 12
  2.7 PATTERN MVC (Model-View-Controller) ............................................... 13
  2.8 NETBEANS ............................................................................................... 14

3 Method and implementation ............................................................................... 15
  3.1 INTRODUCTION ......................................................................................... 15
  3.2 STUDY OF EXISTING SOLUTIONS ........................................................... 16
  3.3 PHASES TO DEVELOP AND PLANNING ............................................... 18
  3.4 COST ESTIMATE ......................................................................................... 19
  3.5 USERS PROFILE ......................................................................................... 21
  3.6 REQUIREMENTS CAPTURE ...................................................................... 23
  3.7 TASK ANALYSIS: USE CASES ................................................................. 24
  3.8 TASK ANALYSIS: HTA DIAGRAM ......................................................... 41
  3.9 DESIGN OF THE SYSTEM INTERFACE .................................................... 44
  3.10 CLASSES DIAGRAM .............................................................................. 57

4 Findings and analysis .......................................................................................... 59
  4.1 INTRODUCTION .......................................................................................... 59
  4.2 IMPLEMENTATION DECISIONS ................................................................. 59
  4.3 VALIDATION OF THE PROJECT ................................................................. 60

5 Discussion and conclusions ............................................................................... 64
  5.1 DISCUSSION OF FINDINGS ...................................................................... 64
  5.2 CONCLUSIONS .......................................................................................... 65

6 References ........................................................................................................... 66
<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HTML</td>
<td>HyperText Markup Language</td>
</tr>
<tr>
<td>CSS</td>
<td>Cascading Style Sheets</td>
</tr>
<tr>
<td>CHSS</td>
<td>Cascading HTML Style Sheets</td>
</tr>
<tr>
<td>JSP</td>
<td>JavaServer Pages</td>
</tr>
<tr>
<td>API</td>
<td>Application Programming Interface</td>
</tr>
<tr>
<td>JPA</td>
<td>Java Persistence API</td>
</tr>
<tr>
<td>ISO</td>
<td>International Organization for Standardization</td>
</tr>
<tr>
<td>XML</td>
<td>eXtensible Markup Language</td>
</tr>
<tr>
<td>CERN</td>
<td>European Organization for Nuclear Research</td>
</tr>
<tr>
<td>W3C</td>
<td>World Wide Web Consortium</td>
</tr>
<tr>
<td>JDBC</td>
<td>Java Database Connectivity</td>
</tr>
<tr>
<td>MySQL</td>
<td>My Structured Query Language</td>
</tr>
<tr>
<td>PHP</td>
<td>Hypertext Pre-processor</td>
</tr>
<tr>
<td>LAMP</td>
<td>Linux Apache MySQL PHP</td>
</tr>
<tr>
<td>MVC</td>
<td>Model View Controller</td>
</tr>
<tr>
<td>IDE</td>
<td>Integrated Development Environment</td>
</tr>
<tr>
<td>DBMS</td>
<td>Database management system</td>
</tr>
<tr>
<td>JDK</td>
<td>Java Development Kit</td>
</tr>
<tr>
<td>HTA</td>
<td>Hierarchical Task Analysis</td>
</tr>
</tbody>
</table>
## List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3-1</td>
<td>Screenshot ERASMOOS</td>
<td>16</td>
</tr>
<tr>
<td>3-2</td>
<td>EVENT IN FACEBOOK</td>
<td>17</td>
</tr>
<tr>
<td>3-3</td>
<td>GANT CHART</td>
<td>18</td>
</tr>
<tr>
<td>3-4</td>
<td>BOUNDARY DIAGRAM</td>
<td>26</td>
</tr>
<tr>
<td>3-5</td>
<td>USE CASE: REGISTER STUDENT</td>
<td>27</td>
</tr>
<tr>
<td>3-6</td>
<td>USE CASE: IDENTIFICATION STUDENT</td>
<td>28</td>
</tr>
<tr>
<td>3-7</td>
<td>USE CASE: EDIT PROFILE</td>
<td>29</td>
</tr>
<tr>
<td>3-8</td>
<td>USE CASE: CREATE EVENT</td>
<td>30</td>
</tr>
<tr>
<td>3-9</td>
<td>USE CASE: CONSULT FRIENDS</td>
<td>31</td>
</tr>
<tr>
<td>3-10</td>
<td>USE CASE: SEND A MESSAGE</td>
<td>32</td>
</tr>
<tr>
<td>3-11</td>
<td>USE CASE: READ A MESSAGE</td>
<td>33</td>
</tr>
<tr>
<td>3-12</td>
<td>USE CASE: SEARCH</td>
<td>34</td>
</tr>
<tr>
<td>3-13</td>
<td>USE CASE: CONFIRM FRIEND REQUEST</td>
<td>34</td>
</tr>
<tr>
<td>3-14</td>
<td>USE CASE: CANCEL FRIEND REQUEST</td>
<td>35</td>
</tr>
<tr>
<td>3-15</td>
<td>USE CASE: ADD FRIEND</td>
<td>36</td>
</tr>
<tr>
<td>3-16</td>
<td>USE CASE: JOIN CITY</td>
<td>37</td>
</tr>
<tr>
<td>3-17</td>
<td>USE CASE: EVALUATE EVENT</td>
<td>38</td>
</tr>
<tr>
<td>3-18</td>
<td>USE CASE: CONSULT EVENTS</td>
<td>38</td>
</tr>
<tr>
<td>3-19</td>
<td>USE CASE: JOIN EVENT</td>
<td>39</td>
</tr>
<tr>
<td>3-20</td>
<td>USE CASE: DISJOIN EVENT</td>
<td>40</td>
</tr>
<tr>
<td>3-21</td>
<td>HTA</td>
<td>42</td>
</tr>
<tr>
<td>3-22</td>
<td>LOGIN SCREEN</td>
<td>44</td>
</tr>
<tr>
<td>3-23</td>
<td>HOME SCREEN</td>
<td>45</td>
</tr>
<tr>
<td>3-24</td>
<td>EDIT PROFILE SCREEN</td>
<td>46</td>
</tr>
<tr>
<td>3-25</td>
<td>INBOX SCREEN</td>
<td>47</td>
</tr>
<tr>
<td>3-26</td>
<td>DETAILS MESSAGE SCREEN</td>
<td>48</td>
</tr>
<tr>
<td>3-27</td>
<td>MY EVENTS SCREEN</td>
<td>49</td>
</tr>
<tr>
<td>3-28</td>
<td>DETAIL EVENT SCREEN</td>
<td>50</td>
</tr>
<tr>
<td>3-29</td>
<td>MY CITIES SCREEN</td>
<td>51</td>
</tr>
<tr>
<td>3-30</td>
<td>ADD EVENT SCREEN</td>
<td>52</td>
</tr>
<tr>
<td>3-31</td>
<td>CITY SCREEN</td>
<td>53</td>
</tr>
<tr>
<td>3-32</td>
<td>SEARCH SCREEN</td>
<td>54</td>
</tr>
<tr>
<td>3-33</td>
<td>MY FRIENDS SCREEN</td>
<td>55</td>
</tr>
<tr>
<td>3-34</td>
<td>DETAIL FRIEND SCREEN</td>
<td>56</td>
</tr>
<tr>
<td>3-35</td>
<td>CLASSES DIAGRAM</td>
<td>57</td>
</tr>
</tbody>
</table>
List of Tables

TABLE 3-1. TASKS AND TEMPORARY PLANNING................................................................. 18
TABLE 3-2. COST ESTIMATE.................................................................................................. 20
TABLE 4-1. FILE PROPERTIES OF JAVA.................................................................................. 60
TABLE 4-2. TEST PLAN OF ACCESS ....................................................................................... 61
TABLE 4-3. TEST PLAN OF REGISTRATION ............................................................................ 62
TABLE 4-4. TEST PLAN OF EDIT PROFILE................................................................................ 62
### Introduction

#### 1.1 Background

All software must be reviewed by engineers over and over again so that the user is satisfied with the product. The needs of today, based on automating everyday tasks, evolve, usually improving the above, and furthermore appear new needs. Everything changes, and that is the purpose of this project: review what exists today and add new functionality, but this is not the only reason, but also it must be designed and constructed so that changes in the future, as computer engineers, just make us lose time. In software engineering terms, this can only be achieved by separating the business logic, interface and control program. So, I am going to delve into the Model View Controller architecture, which aims to do just that.

This project arises from a personal need, when at one point of my life I decided to study part of degree outside of my home town. After that decision, I searched for options that I had and which destinations offered at my university. Subsequently, I decided to get information from host universities and cities, as being outside of city which is the most familiar for you, everything is new and different; The people change around you and change the places you visit often. Knowing where we can find places, accommodations, venue for entertainment, areas for sports, supermarkets, and interest areas of the site. In addition, comments and ratings from other people living in that place, both students who are passing through the city like those of their place of habitual residence.

To find information about universities was, as far as it goes, easy. Because I agreed to the various sites that every university has and got information about them. But I had more trouble finding information about different cities and places of interest, quality of life, etc.

From all this, I thought not only in people who move outside our city one year, but also people who, for some reason, make their university studies the usual out of town.

Therefore, in our case, we will review the existing weaknesses in the platforms of support to students who decide to leave his hometown to continue his studies. Performing a detailed analysis and design what the potential user of the application needs.

As computer scientists or novice users, we require in our applications that the interaction between user and machine is simple and intuitive, with this task required in this project. Each phase of the software engineering process applied to this project will have a continuous basis usability evaluation to user profile application.

Being in the final year of degree, taking on a project with real and tangible needs is a challenge. Is essential to know deeply the study along with the requirements of usability for a computer engineering student, in a few months wills the labour market. Through this project I can improve both concepts in a way that was unattainable for me, and besides, I can enhance the study of technologies demanded in my area as: JAVA, HTML, JavaScript, CSS, Servlet and JSP.
1.2 Purpose and research questions

When a student changes city for various reasons, he/she searches information about cities and universities. And once this person is installed on the destination city, is interested in events taking place in the city, both academic and leisure events.

For example, the person seeking information about the cities, would like to see pictures of those cities, general information, find people who live in that city, what sights there are academics and entertainment. On the other hand there are people who organize this information, giving publicity to their city and are providing photographs, and all information necessary for a person accessing the city is informed fully of all that can be found in it.

Following, define the main objectives expected by potential users:

- The student who is looking for a city to live and study wants to know how the city is and then, when they live in, what events are held in that city.
- The person, who wants to publicize the city where he or she lives, wants to sell his best showing useful information and creating events that will be carried out in the city.

The two roles are interchangeable, because a person to update the information in his or her city may also be interested in other cities and seek information from them. On the other hand, a person who has used the system can enter information about the city where he lives, or their hometown.

1.2.1 Conclusions

Actually, I have not found any application that can cover everything I have mentioned. Therefore, I will develop an analysis, design and implementation of a prototype that can help people find interesting activities to do in their cities.

1.3 Delimitations

The application will be stored on a local server on the same site where the project does; it will not go up to an external server. This could be done as a possible improvement of the application.

Furthermore, due to the above, the management of the application cannot manage many users.

The web application will not be finished, because the estimated time for the project do not permitting it. Hence, some ideas will be left out. But, whatever is developed, it will work.

Moreover, to facilitate the development of the project I have relied on a very specific user, a student studying to be out of town. But, however, if the project in the future, have place in the business field, I will study extend the application to other users.
This happens considering that the application development is a prototype easily expandable and modifiable of a planned system, including its interface and its functionality of inputs and outputs.

1.4 Outline

This project is divided into 5 chapters. Whereof, the first consisting of the points prior to this, is where is explained the objectives, methodology, motivation of the project and a small study of existing solutions. With this last point, the introduction is completed; it is followed by 5 chapters whose contents are explained below.

The second chapter discusses the theoretical issues revolving around the project. This explains in detail the choice of programming languages and tools chosen.

In the third chapter explains the methods that have been chosen and all that that implies. In addition, there will be an explanation of how it has performed the work, including descriptions of experiments have been conducted and how the procedure was collected and processed the application data. Finally, also it includes the design tools that were used during this process.

The fourth chapter shows the results of the study and analysis of these results. Also includes a description of the system that is designed and analyzed its functions in respect to what was originally contextualized.

In the last chapter we find a discussion of the method. The choice of method will be discussed, and the strengths and weaknesses that have been experienced. Furthermore, the results will be analyzed and evaluated with regard to the proposal and research questions.
2 Theoretical background

Among the technologies available, we chose to use Java and JPA as programming language and persistence API, respectively. On the other hand, between the developments tools used include NetBeans. We will know that is each of those profits and benefits.

2.1 Concept of Usability

Today if we want an application to be used must be very usable, so the user can perform his or her job without the interface is a barrier.

According to standard ISO 9241-11 [1] usability is defined as:

“Extent to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use.”

This is a centered definition in the concept of quality in use, in other words, it is referred how user carries out specific tasks in specific scenarios with effectiveness and efficiency.

2.2 Java

In this paragraph I am going to talk about Java language. Java is distinguished for their safety, simplicity and independence. This language will used in the web prototype.

Java is a programming language object – oriented with a similar syntax to C or C++, but if offers simplicity and hardness in the development.

The main feature of Java is that it is platform independent and can run on different architectures and operating systems without having to modify the program code. This independence is achieved because the language is supported by two key elements: the compiler and virtual machine. The compiler translates the program to a special format called bytecode, which is the format that is passed to the virtual machine.
2.3 JSP

JavaServer Pages (JSP) [2] is a Java technology which can generate dynamic content for web, as HTML or XML documents.

JSP allows the use the Java code through scripts.

Example of JSP:

```jsp
<%@ page errorPage="myerror.jsp" %>
<%@ page import="com.foo.bar" %>
<html>
<head>
<!--[if int serverInstanceVariable = 1;]>
...  
<% int localStackBasedVariable = 1; %>
```

2.4 CSS

Cascading Style Sheets (CSS) [3]: is a language designed for describing the appearance of documents written in a mark-up language such as HTML. With CSS you can control the colour of text, the style of fonts, the spacing between paragraphs, how columns are sized and laid out, what background images or colours are used, and a variety of other visual effects. One of the major benefits is that the same CSS can be used by more than one page, meaning that the style of an entire website can be adjusted without having to change each page individually.

The most common use for CSS is to style web pages, and in combination with HTML or XHTML (which is used to describe content) and JavaScript (which is used to add interactivity to a site), CSS is a very powerful tool.

The history of how CSS came to be isn’t actually all that relevant to CSS authors of today, so you can skip the next bit if you’re in a hurry. If, like me, you’re interested in the nitty-gritty, read on.

In the early days of the Web, nine different proposals were made to the World Wide Web Consortium, the main standards organization for the Web which is more commonly known as the W3C, for a style sheet language to help separate the visual appearance of a document from its content. In 1994, Cascading HTML Style Sheets was proposed by Håkon Wium Lie, now CTO of Opera Software (a company you’ll meet again later in this chapter), but at the time he was working at CERN with Tim Berners-Lee and Robert Cailliau, the two men who invented the World Wide Web. CHSS became CSS because CSS can be applied to more than just HTML, and in December 1996 the CSS level 1 Recommendation was published.

Since then, three more CSS specifications have been published by the W3C. CSS 2 became a recommendation in 1998, with CSS 2.1 (which fixes a few mistakes in 2), and CSS 3 currently existing as candidate recommendations.
Although CSS 3 is still under development, CSS 2.1 is likely to become a fully fledged recommendation in the near future and is well supported by all modern browsers. In this book, you will cover CSS 2.1 as it stands today and take a look at some of the new features in CSS 3 that you can use in browsers today.

In the rest of this chapter, you will learn the advantages of using CSS and then get started with your first Cascading Style Sheet.

### 2.5 JPA

JPA [4] is a persistence API that places an interface between business logic and database. Place the interface for the developer to access a database transparently.

JPA is an alternative to JDBC because JPA is considered a pure interface. With JPA do not have to create tables in the database as it does for us. Neither we have to remember how to insert a row into a table.

Simply indicate certain directives in the classes we want to persist through operations in Java, you can access, modify or delete data in the database.

Advantages that offer JPA are the following:

- Automatic creation of tables in the database from Java classes.
- Addition, modification and deletion of rows in the tables in the database from operations in Java.
- Addition, modification and deletion of attributes in the tables in the database automatically.
- Possibility of migration of the database instantly, because the tables are generated automatically.
- Ability to maintain the persistence of XML or files without modifying the code.

### 2.6 Oracle

Oracle [5] is an object – relational data base management system and its characteristics are:

- Transaction support
- Stability
- Scalability
- Platform support
2.7 Pattern MVC (Model-View-Controller)

The implementation of this prototype is based on the architectural pattern Model-View-Controller [6]. This is currently a very established standard for the development of software systems that require a user interface.

2.7.1 How the pattern arises MVC

The first document which speaks of the separation of the elements of a software system in a model, driver software and a manipulable view by user emerges in the Xerox PARC (Palo Alto Research Center) in 1978; an additional component is mentioned, called Publisher, part of the view.

The MVC pattern arises during development of an application (by a scientist who was staying at PARC) in which he tries to find a general solution to the problem of controlling a set of large and complex data by a user. To this end, devised a process in which the model represents the knowledge with which to work the system, the view was a visual representation of that model and the controller acting as a liaison between the user and the system. These ideas were written in a document published on 10 December 1979 under the title Model-Views-Controllers, i.e. MVC.

The first implementation of MVC architectural pattern itself was conducted in PARC for the programming language Smalltalk-80. From that moment began its popularization.

2.7.2 Benefits that provides MVC to design of software systems

Benefits that provide MVC are many. We will discuss the most important:

- The same system can have multiple views, depend the type of user that uses it. This prototype, for example, contains a view for users who access a web browser, but may have one for users accessing from a mobile device, and this would not cause a change in the model.

- The MVC pattern provides decoupling between model and view, so that you can switch views very simply without altering in any sense the model.

- The model has not knowledge about the view or the controller.

- When the system starts, it creates the controller and leaves the creation of the model and view or views that may exist in hands of controller, establishing the right connections between them. So it makes the model never know the view or controller.

Therefore, the MVC architectural pattern promotes the design of software systems: only the model manages the state of the system, only the view generated visual representations of that state so, and maintains a high degree of decoupling, enabling modification and even replacement of any of them without affecting the rest. This helps developers to quickly respond to change, because the application is easier to maintain.
2.8 NetBeans

In the market there are many products that facilitate the development of Java applications, tools consist of code editors, interface designers, package administrators, and so on. A large part of them are open source and multi-platform, so the choice of development tool will depend primarily on the needs of each particular project and, to a large extent on the preferences of each developer.

Finally it was decided to use the latest version of NetBeans [7] to implement the project for two reasons: be a free tool from the same manufacturer that the platform, and available for multiple operating systems.
3 Method and implementation

3.1 Introduction

This chapter will begin with a study of existing solutions. Then we will project planning time and cost. Then will document the analysis phase and finally define the most important part in the whole process of software engineering: the design of the prototype web.

For the prototype website to be used in future by the user, must make an analysis in the areas that a student seeking information from another city. By now we have some knowledge of what motivates a user, but we have not done an exhaustive study of it. The study begins by observing what need a user and what areas he/she frequently performed. This is accomplished with user profiles, analysis of requirements and scenarios.

Once we are clear what the user demand is a matter of abstracting the process a little more. This would get to define the known use cases and task analysis, completing the analysis phase.

The process continues with the design of screens and storyboards. Each use case is defined as a user screen and task analysis helps us to successfully complete navigation changes (storyboards) between screens. The design of screens and storyboards in this project is subject to a continued sense of usability, whose principles are explained in the second chapter of this project.

Finally would be one of the most delicate phases of the project: obtaining from system diagram of all the tools used so far in the design phase. This task has a relative difficulty, since it involves abstracting all the visual information gathered so far (screens and storyboards) to get the class diagram, allowing the programmer to code the prototype of an easier way.
3.2 Study of existing solutions

⇒ Erasmoos [8]

![Figure 3-1. Screenshot Erasmoos.](image)

Erasmoos is an online community for the International students, a place where to meet people before you go to your exchange experience abroad, where to share your experience and find information from the tips written by others.

This web application has some disadvantages compared to what we are looking. Are detailed below:

- We only used to contact people, who also travel to a particular city. We only have information from people who are or have been in that city, but no indigenous people of the place, who can give interesting information.

- Does not provide aid neither to get familiarized with cities or sites of interest. The purpose of the application is to meet people who have lived or are living the same experience but is not intended to know the city itself, only interact with people and meet new people. But its organization does not provide us information about places of interest in the city, which is one of the first things a person arises when moving from one city to another.

- Is an application made by a Spanish bank, which apart from giving you a social network to meet people from the same fate ERASMUS yours, as we discussed earlier, offers financing. Therefore this website is too specific.
Method and implementation

- It is intended for people who receive the Erasmus grant. As we said before, this portal provides funding for students receiving the scholarship ERASMUS and information for them, but there are people who move from city to study for many reasons. Not only abroad, if not within their own country. For example, people who want to pursue a career but in his Hometown University there, or wish to study a degree not found in his hometown and have to move in search of their interests.

  ⇒ Facebook [9]

![Create Event](facebook_create_event.png)

Figure 3-2. Event in Facebook.

Another system that can allow us to release information about events taking place in a particular city is Facebook, more specifically Facebook events. This tool gives us Facebook is creating events, which allows us to name the event, choose a date, location and additional details. But this has a number of disadvantages compared to what we are seeking to develop, which we will detail below:

- For the events we are invited only if friends invite us who we have in the social network.

- We cannot access all the events organized in a particular city, because as stated above can only be accessed by invitation of another person or if the event was created as public.

- With the events, we cannot get an overview of a city, only if we are invited to the event, obtain details of the event, but we will not obtain information from the sights of the city.
3.3 Phases to develop and planning

In the introduction to the project (see 1.4), phases were mentioned, and enumeration of specific tasks in the following planning.

Table 1 lists the tasks to be performed, the time devoted to it and the dependencies between them.

<table>
<thead>
<tr>
<th>Id</th>
<th>Task</th>
<th>Week</th>
<th>Depends</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Investigation of the MVC architectural pattern</td>
<td>2</td>
<td>-</td>
</tr>
<tr>
<td>2</td>
<td>Analysis and design of the prototype web</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>Implementation of prototype web</td>
<td>8</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>Testing and validation</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>5</td>
<td>Report</td>
<td>4</td>
<td>-</td>
</tr>
</tbody>
</table>

Table 3-1. Tasks and temporary planning.

The development of activity 5, Drawing from memory, was carried out in parallel to other project tasks.

The temporary ordination and task dependencies, is graphically illustrated in Figure 2, for the Gantt chart.

Figure 3-3. Gant chart.
3.4 Cost estimate

This section of the chapter we will make an estimate of project costs to develop. This information is very important, and it is an essential step in any engineering project. Used to study the feasibility of the project, and used to the knowledge of the investments required for their development.

Software resources we do not want that involve extra cost neither for developer nor customer, so the analysis will focus on developing search tools freely licensed.

We shall discuss the most interesting aspects of the budget, as summarized in Table 2.

- The total cost of the laptop is 8000 Kronor. However in the table is 1000 Kronor, because it takes into account the proportionate cost in the 6-month duration of the project, considering that the laptop had a useful life of 48 months.

- The total cost wireless mouse is 200 Kronor. However in the table is 50 Kronor, because it takes into account the cost proportional to the 6-month duration of the project, considering that the mouse had a useful life of 24 months.

- To calculate the number of hours at different stages of the project, we have relied on the section of the development and planning phases (See 3.3), where an estimated duration of each of the tasks for weeks.

- In one week, the worker performed a total of 30 hours of useful work (5 days * 6 hours / day)

- The gain corresponds to the margin of profit the company earns the realization of the project; it works not only to cover costs.
<table>
<thead>
<tr>
<th>Resource</th>
<th>Amount</th>
<th>Unit cost</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>HARDWARE</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Laptop</td>
<td>0,125</td>
<td>8000 kr</td>
<td>1000 kr</td>
</tr>
<tr>
<td>Wireless mouse</td>
<td>0,25</td>
<td>200 kr</td>
<td>50 kr</td>
</tr>
<tr>
<td><strong>SOFTWARE</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tomcat</td>
<td>1</td>
<td>0 kr</td>
<td>0 kr</td>
</tr>
<tr>
<td>JDK</td>
<td>1</td>
<td>0 kr</td>
<td>0 kr</td>
</tr>
<tr>
<td>NetBeans</td>
<td>1</td>
<td>0 kr</td>
<td>0 kr</td>
</tr>
<tr>
<td>Visual Paradigm 8.0 (Community Version)</td>
<td>1</td>
<td>0 kr</td>
<td>0 kr</td>
</tr>
<tr>
<td>Office suite</td>
<td>1</td>
<td>0 kr</td>
<td>0 kr</td>
</tr>
<tr>
<td><strong>SOFTWARE ENGINEERING</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Investigation of the MVC and usability</td>
<td>60 hour</td>
<td>500 kr/h (analyst)</td>
<td>30000 kr</td>
</tr>
<tr>
<td>Analysis and design</td>
<td>120 hour</td>
<td>500 kr/h (analyst)</td>
<td>60000 kr</td>
</tr>
<tr>
<td>Implementation of prototype web</td>
<td>240 hour</td>
<td>300 kr/h (programmer)</td>
<td>72000 kr</td>
</tr>
<tr>
<td>Testing and validation</td>
<td>60 hour</td>
<td>300 kr/h (programmer)</td>
<td>18000 kr</td>
</tr>
<tr>
<td>Report</td>
<td>120 hour</td>
<td>500 kr/h (analyst)</td>
<td>60000 kr</td>
</tr>
<tr>
<td><strong>OTHER TYPES OF COSTS</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bound and printed memory</td>
<td>3</td>
<td>500 kr</td>
<td>1500 kr</td>
</tr>
<tr>
<td>Internet</td>
<td>6 month</td>
<td>300 kr</td>
<td>1800 kr</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Costs</td>
<td></td>
<td></td>
<td>244350 kr</td>
</tr>
<tr>
<td>Benefit (20%)</td>
<td></td>
<td></td>
<td>48870 kr</td>
</tr>
<tr>
<td>Subtotal (benefit included)</td>
<td></td>
<td></td>
<td>293220 kr</td>
</tr>
<tr>
<td>VAT (25%)</td>
<td></td>
<td></td>
<td>73305 kr</td>
</tr>
<tr>
<td>TOTAL</td>
<td></td>
<td></td>
<td>366525 kr</td>
</tr>
</tbody>
</table>

Table 3-2. Cost estimate.
3.5 Users profile

User profiles allow obtaining better understanding of user and definition final goals, which later were transformed in functional requirements of the application.

3.5.1 User profile: student

Overview

Any student, who wants to know another city or to give information of their own city.

Characteristics of the user

Students have a high knowledge in the use of computers. They are accustomed to using similar applications.

The average educational level is high school or college. Read and write without difficulty and have medium or high knowledge of another language (mostly English). The average age does not exceed 23 years.

His experience in using similar applications is high, since over 80% of students manage social networks. His experience with computers is high. Use computers regularly.

Goals

- **Final goals**
  - Meeting new cultures.
  - Practicing languages.
  - Be done to help people unfamiliar with the city.
  - Make known their city to others.
  - Meeting people from different places.
  - Know the city and information about places of interest for leisure, cultural or related to the university.
  - Keep informed of interesting events that take place in that city.
  - Do not feel lost in the destination.
  - Know the culture of the place where you will live.

- **Use goals**
  - Create an event that is not tedious.
  - The interface is not a barrier and everything is organized and ready for use at all times.
  - Student feels safe with every action you make.
Method and implementation

- Do not waste time answering questions from other students.
- Sometimes the user does not want to waste time and want to know very quickly what are the most important events that will be done the same day is viewing.

Usability requirements

The indigenous student has no problem when using an application, so they are very critical when evaluating an interface. This application will be used daily and a medium time, so you have to be highly usable and safe for them. The interface has to be direct, simple and flexible, always giving all safe to use.

Keep in mind that the native student cannot afford to waste much time when creating an event, see what messages has, and this should be taken into account especially when designing the interface.

The display shall be part exclusively devoted to tasks that students can do directly, so they do not get lost in navigation. Thus we get that in a few steps to get their purpose.
3.6 Requirements capture

3.6.1 Functional requirements analysis

The functional requirements allow, in a clear and concise way, define the ultimate goals of potential users of the system. With this, we get that from these requirements, we obtain the prototype use cases.

The functional requirements of the prototype software we are developing are:

- **Register in the application.** *It consists of what a particular type of user can register or not.*
  - A user himself can register in the system.
  - A user can register an event.

- **Application management.** *It consists of what a particular type of user can manage in the application or not.*
  - Access the system by identifying.
  - Create an event.
  - Join an event.
  - Evaluating an event.
  - Write a private message.
  - Make friends.
  - Edit profile.
  - Join one or more cities.

- **Consultations.** *Consists in the possibility that consultations can be made and these are displayed in a simple and direct way to the user.*
  - Consult the list of friends.
  - Consult the list of cities the user joined.
  - Consult the events user will go.
  - Consult the events of the cities that user has joined.
  - Consult private messages that others users have written to him.
  - Consult users.
  - Consult cities.

3.6.2 Non–functional requirements analysis

Non–functional requirements are those that restrict the functional requirements. These are just as important as functional requirements itself and may even become critical to system acceptance. These requirements specify properties of the system (platform, speed, performance…) and design of the graphic interface.
As we can guess at this point, the application is an architecture client – server. This means that the server will run transactions that one or more clients, simultaneously, request through an electronic device. We will explain the non – functional requirements of these two parts of the architecture.

### 3.6.2.1 Non – functional requirements. (Server).

- **Hardware**
  - Speed: The server must be fast enough to run the application in the shortest possible time and with greater reliability.
  - Memory: The server must dispose have sufficient RAM to do the operations the application and database request.
  - Storage. The server must have sufficient a storage capacity to store the database which works the application and allow the transactions between the two applications.

- **Software**
  - The server must have installed a manager of database (DBMS). Our application has been tested and designed with the DBMS of MySQL in a version that allow cipher keys. (MySQL version 4.0 or higher).
  - The server must have installed JDK 1.6 or higher for exist compatibility with the used libraries.
  - The server must have installed Apache Tomcat with a compatible version to JDK installed. All serve through Tomcat, because the application only uses JSP and Servlets.

### 3.6.2.2 Non – functional requirements. (Client)

- **Hardware**
  - Speed: The PC should be fast enough to run a web browser with JavaScript enabled. Today this is no problem, as any personal computer microprocessor is capable of fulfilling this task.

- **Software**
  - It should have of a compatible browser with JavaScript.
  - JavaScript should be enabled.

### 3.7 Task analysis: Use cases

#### 3.7.1 Introduction to use cases

A use case represents a situation or a interaction task of one or more user with application. Use cases are tasks meaningful, consistent and relatively independent that actors to perform in their daily work.

Use cases consist of the following elements:
Method and implementation

- Unique name.
- Use case diagram.
- Actors involved.
- Inlet conditions.
- Flow events.
- Output conditions.
- Exceptions.

In our system we have an actor (previously described in user profiles): student.

Now is the time to create the different use cases. When performing this action is important that each of the functional requirements already defined, appears in at least one use case, as we have stated previously.

We emphasize that there can be new use cases, which does not appear any of the requirements, because we are in a phase of refinement of the system.

3.7.2 Boundary diagram

A step towards the creation and description of the different use cases is to obtain a diagram that fully describes the system functionality. This is called boundary diagram.
Figure 3-4. Boundary diagram.
The use cases shown in a boundary diagram can be sufficiently accurate or conversely can be concretized in greater detail. When a use case is detailed you can use two kinds of relations:

- \text{<<\ extend \>>}: is a relationship that represents exceptional behaviour of the use case.
- \text{<<\ include\ >>}: is a relationship that represents the need for another use case.

Below are descriptions of the use cases and diagrams associated with each of a more precisely:

\textbf{3.7.3 Use case: Register student}

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{use_case_register_student.png}
\caption{Use Case: Register Student.}
\end{figure}

\textbf{Actors involved}: student.

\textbf{Inlet conditions}: -

\textbf{Flow events}:
1. Student fills mandatorily the following attributes: name, email, password, sex and birthday.
2. System registers the new user profile.

\textbf{Exit conditions}:
- Student has been registered in the system.

\textbf{Exceptions}:
1a. Student did not fill all fields and presses \textit{Sign up}. 


Method and implementation

- The system notifies the user and returns to step 1.

### 3.7.4 Use case: Identification student

![Diagram of the Identification Student use case](image)

**Figure 3-6. Use Case: Identification Student**

**Actors involved:** student.

**Inlet conditions:**
- Student has been registered correctly.

**Flow events:**
1. Student enters her/his credentials: e-mail and password.
2. The system successfully authenticates the student.

**Exit conditions:**
- Student accesses the system.

**Exceptions:**
1a. Student does not fill correctly the data and presses *Sign in*.
   - The system notifies the user and returns to step 1.
3.7.5 Use case: Edit profile

Figure 3-7. Use Case: Edit Profile.

A **ctors involved**: student.

**Inlet conditions**:
- Student has been identified correctly.

**Flow events**:
1. Student fills information that he/she wants updates.
2. The system successfully registers new profile of the user.

**Exit conditions**:
- Student has a new profile.

**Exceptions**:
3a. Student does not fill correctly the data and presses *Save*.
   - The system notifies the user and returns to step 1.
3.7.6 Use case: Create event

![Diagram of Create Event process]

**Figure 3-8. Use Case: Create Event**

**Actors involved:** student.

**Inlet conditions:**
- Student has been identified correctly.

**Flow events:**
1. Student accesses the option *Create Event*.
2. System displays a screen which user establishes the place, time and a description of event.
3. Student fills all attributes and press *Save*.
4. System registers the event.

**Exit conditions:**
- System stores new event.

**Exceptions:**
3a. Student cancels the operation.
   - System does not register new event and return *City*.
3b. Student does not fill correctly the data and presses *Save*.
   - The system notifies the user and returns to step 2.
3.7.7 Use case: Consult friends

Figure 3-9. Use Case: Consult friends

**Actors involved:** student.

**Inlet conditions:**
- Student has been identified correctly.

**Flow events:**
1. Student accesses option *my friends*.
2. System displays a list with friends of the student.

**Exit conditions:** -

**Exceptions:** -
3.7.8  **Use case: Send a message**

A diagram is shown with the title "System" and various flow events and conditions. The diagram includes a path for sending a message and conditions for extending the message process.

**Awards involved:** student.

**Inlet conditions:**
- Student has been identified correctly.

**Flow events:**
1. Student accesses a friend and presses *send a message*.
2. System displays a screen where the user sets the message text.
3. Student writes the message and presses *send*.
4. System sends the message to the right person.

**Exit conditions:**
- System stores the new message.

**Exceptions:**
3a. Student cancels the operation.
   - System does not register a new message and returns *Friend*.
3b. Student does not fill correctly the data and presses *Send*.
The system notifies the user and returns to step 2.

### 3.7.9 Use case: Read a Message

**Figure 3-11. Use Case: Read a Message**

**Actors involved:** student.

**Inlet conditions:**
- Student has been identified correctly.

**Flow events:**
1. Student accesses the option: *Messages*
2. System displays a list of messages, and specially marks unread messages.
3. Student chooses the message.
4. System displays the complete message.

**Exit conditions:** -

**Exceptions:** -
3.7.10 Use case: Search

![Use Case: Search](image)

Figure 3-12. Use Case: Search.

**Actors involved:** student.

**Inlet conditions:**
- Student has been identified correctly.

**Flow events:**
1. Student writes a text in search box.
2. System displays a results list.

**Exit conditions:** -

**Exceptions:** -

3.7.11 Use case: Confirm Friend Request

![Use Case: Confirm Friend Request](image)

Figure 3-13. Use Case: Confirm Friend Request

**Actors involved:** student.

**Inlet conditions:**
Student has been identified correctly.
The user has pending friend requests.

**Flow events:**
1. Student accesses the option *Friends.*
2. System displays a list with the pending friend requests.
3. Student presses *Accept* in one of them.
4. System registers a new friend on the user’s profile.

**Exit conditions:**
- System registers the new friendship in the system.

**Exceptions:** -

### 3.7.12 Use case: Cancel Friend Request

**Actors involved:** student.

**Inlet conditions:**
- Student has been identified correctly.
- The user has pending friend requests.

**Flow events:**
1. Student accesses the option *Friends.*
2. System displays a list with the pending friend requests.
3. Student presses *Cancel* in one of them.
4. System deletes the friend request of the list.

**Exit conditions:**
- System deletes the friend request of the system.
Exceptions: -

3.7.13 Use case: Add friend

Actors involved: student.

Inlet conditions:
- Student has been identified correctly.
- Student has done a search in the system.

Flow events:
1. Student chooses a person of the list and presses Add friend.
2. System notifies the appropriate person the friend request.

Exit conditions:
- System registers the friend request in the system.

Exceptions: -
3.7.14 Use case: Join city

![Diagram of the Join City use case](image)

Figure 3-16. Use Case: Join City

**Actors involved:** student.

**Inlet conditions:**
- Student has been identified correctly.
- Student has done a search in the system.

**Flow events:**
1. Student chooses a city of the list and presses *Join*.
2. System displays in the list of cities of the user the new city.

**Exit conditions:**
- System registers the city in the user profile.

**Exceptions:** -
3.7.15 Use case: Evaluate Event

Actors involved: student.

Inlet conditions:
- Student has been identified correctly.
- Student has event in her/his profile.

Flow events:
1. Student accesses the option Events and presses Like or Dislike in an event.
2. System increase in one Like or Dislike.

Exit conditions:
- System registers the assessment of user.

Exceptions: -

3.7.16 Use case: Consult Events

Figure 3-17. Use Case: Evaluate Event

Figure 3-18. Use Case: Consult Events
**Actors involved:** student.

**Inlet conditions:**
- Student has been identified correctly.

**Flow events:**
1. Student chooses the option *Events*
2. System displays a list of events of the cities that user is joined.

**Exit conditions:** -

**Exceptions:** -

### 3.7.17 Use case: Join Event

![Use Case: Join Event](image)

**Figure 3-19. Use Case: Join Event.**

**Actors involved:** student.

**Inlet conditions:**
- Student has been identified correctly.

**Flow events:**
1. Student chooses the option *Events*
2. System displays a list of events of the cities that user is joined.
3. Student chooses one of them and presses *Join*.
4. System stores this event to the list of events that user will go.

**Exit conditions:**
- System registers assistance of the user.

**Exceptions:** -
3.7.18 Use case: Disjoin Event

Figure 3-20. Use Case: Disjoin Event.

Actors involved: student.

Inlet conditions:
- Student has been identified correctly.
- Student is joined at least one event.

Flow events:
1. Student chooses the option My Events.
2. System displays a list of the events that the user will go.
3. Student chooses one event and presses Disjoin Event.
4. System deletes the event of the list of the user.

Exit conditions:
- System registers this action in the system.

Exceptions: -
3.8 Task analysis: HTA diagram

**Analysis task** is defined as:

The process of analyze how a user realizes his/her work. Furthermore, know the things they do, and things on which they act, information that they need to carry out, and goals that leads them to do this work.

The purpose of a user is not interacting with computer, but comply some goals. The interaction is the means by which that goal is met. Thus, knowledge of the user's task requires knowledge of their goals and motivations; we know this, for example, through the whole process of analysis we carry so far.

**Task analysis** is done at two levels. At one level the structure of each individual interaction task is determined, and a second level determines the architecture of all tasks of interaction as a whole.

The first level has already been covered with the use cases, where we know the dialogue between actor and system, i.e. the structure of the task. The second level is usually expressed as a diagram with the HTA.

**HTA diagram** expresses the information in a visual format easier to understand for a development team. Each task or subtask is represented by a rectangular box and is labeled with its job number in the upper left corner, followed by the name of the task in the next line.

The task hierarchy is represented by a horizontal line at the top of the boxes, while simple tasks or activities (i.e., those which are not subdivided) are represented by a horizontal line at the bottom of the box represents them.

The HTA diagram would look like (Figure):
Method and implementation

Figure 3-21. HTA
Plan 0:
- Do 1 and 2. Do 3, 4 and 5 in any order until complete.

Plan 2:
- Do 2.1, if credentials are incorrect return 2.1

Plan 3:
- Do 3.1, if attributes are incorrect return 3.1

Plan 4:
- Do 4.1, 4.2, 4.3, 4.4 and 4.5 in any order
  
  Plan 4.2:
  - Do 4.2.1, 4.2.2 and 4.2.3 in any order

  Plan 4.3:
  - Do 4.3.1, 4.3.2 and 4.3.3 in any order

  Plan 4.4:
  - Do 4.4.1, 4.4.2 and 4.4.3 in any order

  Plan 4.5:
  - Do 4.5.1, 4.5.2 and 4.5.3 in any order
    
    Plan 4.5.1:
    - Do 4.5.1.1 and 4.5.1.2 in any order

    Plan 4.5.2:
    - Do 4.5.2.1, 4.5.2.2 and 4.5.2.3 in any order

We can see that is perfectly expressed the architecture of the interaction tasks of the student role. We observe how tasks are connected to each other through the hierarchy.

The plans describe in what order and under what conditions subtasks are performed. It is a very important task because from these we can determine much of the navigation paths in our application.
3.9 Design of the system interface

Once the analysis phase, the next phase of software engineering is the design. With the design seeks to obtain the class diagram of the prototype, which translates to a programming language allowing the deployment of the prototype web usable.

Before obtaining the class diagram should make the screen design and navigation between them (storyboards). Then there will be a most laborious task throughout this project: get the class diagram from the screen design. Later specify how, and what matters now is the next step: designing the system interface, consisting of design screens and storyboards.

3.9.1 Screens design

To perform the preliminary design of the user interface is possible to use different tools, one of which prototyping screens, either freehand drawings, cutting and pasting pieces of paper which represent components, or by using digital media. In my case, screen design will integrate with digital media, using Photoshop.

3.9.1.1 Login Screen

![Login Screen](image)

Figure 3-22. Login screen.
If the user is not registered, they are asked to fill in the second form. If it is registered user he/she have to fill in the first form.

### 3.9.1.2 Home Screen

![Home Screen](image)

**Figure 3-23. Home screen.**

The appearance of the Web application in most cases is the same. Above we can see a navigation bar at the top of the page. Here we can find a link that leads to principal page of user, a text box which we can search both, people and cities, profile access and log out.

The left navigation bar, you can see the profile picture and the name of the user, and there is a series of navigation options. In the middle we have the dynamic changes of the web that changes throughout the user navigation.

Users can access their messages, their events, theirs friends and their cities.
3.9.1.3  Edit Profile Screen

Figure 3-24. Edit profile screen.

Student accesses to this page when has pressed profile on the navigation bar at the top right.

Here, user has the option to change his/her name, email, password, birthday or city. In addition, the user can change her/his profile picture.
3.9.1.4 **Inbox Screen**

![Inbox Screen](image)

**Figure 3-25. Inbox screen.**

The user can view a list of all his/her messages sorted by user pressing the Inbox option on left navigation bar.
3.9.1.5 Details Message Screen

User can view the detail of message, and this is the thread of conversation he/she has with other user. Furthermore, user can send another message from this screen.

Figure 3-26. Details message screen.
3.9.1.6 My Events Screen

The user can view a list of all his/her events pressing the Inbox option on left navigation bar. Furthermore user can vote these events or changes her/his mind and not go to the event, thus will be removed from this list.
3.9.1.7 Detail Event Screen

Here, user can view details of an event, and if it is interesting she/he can join the event or vote it.
3.9.1.8 My cities Screen

The user can view her/his cities, with the option to remove it from the list or to access events clicking the name of the city.

Figure 3-29. My cities screen.
3.9.1.9  **Add Event Screen**

![Add Event Screen](image_url)

**Figure 3-30. Add event screen.**

User can add an event to the city which he/she is browsing. User must fill in the details shown on the screen and this event will appear in the list of events.
3.9.1.10 City Screen

Here, user can view the events of a particular city. And he/she can join event or vote event.

Figure 3-31. City screen.
3.9.1.11 Search Screen

User has option to search people or cities.

Figure 3-32. Search screen.
3.9.1.12 My friends Screen

The user can view her/his friends.

Figure 3-33. My Friends Screen.
3.9.1.13 Details Friend Screen

The user can view details of a friend.

Figure 3-34. Detail friend screen.
3.10 Classes diagram

3.10.1 Classes diagram

Class diagrams are used to show the static structure of the modeled system. Class diagrams can consist of classes, interfaces, packages, relations and even instances as objects or links.

Is a powerful design tool, it is a powerful design tool that helps developers to plan and establish the structure of the system before writing any code.

Class diagrams have the follow components:

- Classes: These are the fundamental components of class diagrams. Its general notation is a rectangle divided into three sections: The first is the name of the class, the following attributes and the last operations.

- Relations: a semantic connection between elements. There are four main types of relations:
  - Generalization is a relation of specialization.
  - Association: a structural relationship. It reflects a relationship between two independent classes that remains during the life of objects or at least for a time. There are two types: aggregation and composition.
  - Implementation: is a contractual relationship in which a class specifies a contract that another class guarantees that will comply.
  - Dependence: is a relationship of use, namely the implementation of a class depends on another.

As we know, the design scheme that will be used shall be a Model-View-Controller (MVC). Undoubtedly, and as we pointed earlier, the most important message provided by this design scheme is to keep the model as independent as possible of the view and controller, so you can change either without modification in other components.

Following is the class diagram:

![Figure 3-35. Classes diagram.](image)

We analyze each of the packages of our class diagram:
• **Model Package**: Implements the application logic, i.e. it stores all data, the status of the application and has the methods that manipulate these data.

• **Controller Package**: controller layer will perform all the control of the system and allow the user to perform any operations on it. This package constantly will use classes of Model package to obtain and save information, and will be responsible for operating with the data for display in the view.

• **View Package**: Displays a usable interface to the user, containing the data received from the model or controller.
Findings and analysis

4.1 Introduction

The ultimate goal of software development process, followed by the previous points as analysis and design, is to get a functional system, for which we must proceed to a concrete implementation of it, translating class diagrams and interaction in code that can be executed on a computer.

It makes no sense to explain the implementation carried out, nor is it appropriate to describe the source code since this is a simple translation of the class design made. As indicated in the following paragraphs are implementation decisions and the evaluation tests of the prototype itself.

4.2 Implementation decisions

Encode the prototype has been relatively easy because most of the process is based on translating the class diagram obtained in Chapter 3. However, there have been situations in which it has been necessary to take concrete decisions to achieve the purpose of implementing the prototype of the best. Among the implementation decisions taken include the following:

Using a database that allows encrypted passwords for users of the system.

Use support information-sharing sessions between Servlet and JSP system, checking that there are no illegal accesses to it: The Java support sessions to share information without using the GET and POST methods. That is, we can add any attribute to the session, without having to send it through the methods described above, and consult at any time. Not only is this the advantage of using sessions, perhaps the most important is the ease with which it can detect illegal access to the application. If for example in our application a user logs on, only have access to JSP and Servlet files allowed by the system for that type of user, if you try to access a JSP file that is not part of your session, the system detected quickly and throws an exception that prevents access to the system that way.

Handle exceptions common to have a processing different from the rest: There are accesses to the database exceptions occur because the data requested does not exist. We have created a class for each of the exceptions that can be controlled, so that if it occurs at some point, we are notified via the message specified in the operation toString() for each one of them.

I created a class to access the database for each of the existing tables: JPA creates tables in the database automatically, and as we described above, adding, modifying or deleting items from a table is made from Java operations independent of the database being used. But all these operations should be encapsulated in classes that enable decoupling the code.

Images are saved in a folder at the root of the project, rather than the database: The database stores the path of each of the profile pictures. Is more efficient to store the path in the database that a complete picture. If we had not done that way, it would have been very inefficient consult friends, cities or events in the database.
We used a library of the editorial "O'Reilly" to upload files (images, etc...) through a form [10]: We used a library of the editorial "O'Reilly" to upload files (images, etc...) Through a form [12]: This study uses a library that allows you to upload to the server in a comfortable manner and in a few lines code, files that are in the user's hard drive. For example, to upload profile photos.

I used the JAVA files: "properties" for if in the future I want to add a new language to the application, it is easy.

Table 4-1. File Properties of JAVA

4.3 Validation of the project

4.3.1 Black box testing

This type of testing is performed on the software interface and seeks to demonstrate that: [11]

- Software functions are operational.
- That input is properly accepted.
- Output is correctly produced.
- The integrity of external information is maintained.

Black-box testing uncovers errors of the following categories:

- In-correct or missing functions.
- Interface errors.
Findings and analysis

- Errors in the data structures or external data base access.
- Performance errors.
- Initialisation and termination errors.

The type of black-box testing will be conducted in this project will be equivalence partitioning method (That is to use a small subset of all possible inputs to find the highest likelihood of errors).

Thus, as a black box testing is based on the software interface for each Web page can contain any problems with the values supplied by the user, will make black-box testing. All those pages that do not need to supply values shall be exempt from testing because there is no possibility of interface errors.

The components of the view and their respective black-box testing are exposed as follows:

4.3.1.1 Site “Access”

Test plan

<table>
<thead>
<tr>
<th>Inlet conditions</th>
<th>Valid classes</th>
<th>Invalid classes</th>
</tr>
</thead>
<tbody>
<tr>
<td>$Unregistered user</td>
<td>1. False</td>
<td>2. True</td>
</tr>
<tr>
<td>$key = $password</td>
<td>3. True</td>
<td>4. False</td>
</tr>
</tbody>
</table>

Table 4-2. Test plan of Access

Valid classes (Correct identification is expected):
- Registered user (1).
- $key = ‘hello’ and $password = ‘hello’ (3).

Invalid classes (It is expected an error message):
- Unregistered user (2).
- $key = ‘hello’ and $password = ‘bye’ (4).

4.3.1.2 Site “Registration”

Test plan

<table>
<thead>
<tr>
<th>Inlet conditions</th>
<th>Valid classes</th>
<th>Invalid classes</th>
</tr>
</thead>
<tbody>
<tr>
<td>$Full_name</td>
<td>1. Not empty string</td>
<td>2. Empty string</td>
</tr>
<tr>
<td>$email</td>
<td>3. Not empty string</td>
<td>4. Empty string</td>
</tr>
<tr>
<td>$repeat_email</td>
<td>5. Not empty string</td>
<td>6. Empty string</td>
</tr>
<tr>
<td>$password</td>
<td>7. Not empty string</td>
<td>8. Empty string</td>
</tr>
</tbody>
</table>
Table 4.3. Test plan of Registration

Valid classes (Correct identification is expect):
- $full_name = ‘Iván Pérez’ (1).
- $email = ‘example@gmail.com’ (3).
- $repeat_email = ‘example@gmail.com’ (5).
- $password = ‘hello’ (7)
- $email1 = ‘example@gmail.com’ and $email2 = ‘example@gmail.com’ (9).

Invalid classes (It is expected an error message):
- $full_name = ‘’ (2).
- $email = ‘’ (4).
- $repeat_email = ‘’ (6).
- $password = ‘’ (8).
- $email1 = ‘example@gmail.com’ and $email2 = ‘hello@gmail.com’ (10).

4.3.1.3 Site “Edit profile”

Test plan

<table>
<thead>
<tr>
<th>Inlet conditions</th>
<th>Valid classes</th>
<th>Invalid classes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Size($picture_profile)</td>
<td>1. True</td>
<td>2. False</td>
</tr>
<tr>
<td>100 KBs</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Extension($picture_profile)</td>
<td>3. True</td>
<td>4. False</td>
</tr>
<tr>
<td>‘jpg’</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$full_name</td>
<td>5. Not empty string</td>
<td>6. Empty string</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$email1 = $email2</td>
<td>7. True</td>
<td>8. False</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$old_key = $old_password</td>
<td>9. True</td>
<td>10. False</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$new_key1 = $new_key2</td>
<td>11. True</td>
<td>12. False</td>
</tr>
</tbody>
</table>

Table 4.4. Test plan of Edit profile

Valid classes (Correct identification is expect):
- $picture_profile = ‘picture_low_resolution.jpg’ (1, 3).
- $full_name = ‘Iván Pérez’ (5).
- $email1 = ‘example@gmail.com’ and $email2 = ‘example@gmail.com’ (7).
Findings and analysis

- $\text{old\_key} = \text{‘hello’}$ and $\text{old\_password} = \text{‘hello’} $ (9).
- $\text{new\_key2} = \text{‘bye’}$ and $\text{new\_key2} = \text{‘bye’} $ (11).

Invalid classes (It is expected an error message):

- $\text{picture\_profile} = \text{‘picture\_high\_resolution.jpg’} $ (2).
- $\text{picture\_profile} = \text{‘picture\_low\_resolution.odt’} $ (4).
- $\text{full\_name} = \text{‘’} $ (6).
- $\text{email1} = \text{‘example@gmail.com’}$ and $\text{email2} = \text{‘hello@gmail.com’} $ (8).
- $\text{old\_key} = \text{‘hello’}$ and $\text{old\_password} = \text{‘bye’} $ (10).
- $\text{new\_key1} = \text{‘bye’}$ and $\text{new\_key2} = \text{‘hello’} $ (12).
5 Discussion and conclusions

5.1 Discussion of findings

These are the results from the project:

- A web prototype has been created and in this prototype works the functional requirements which were established in the analysis phase.

- Most usability principles have been taken into account when creating the site prototype.

- The project has been carried out following with precision the planning and performing the steps in the software engineering: analysis, design, implementation and testing.

- A report has been done which defines the whole process which has been carried out in the project.

- Improvements have been identified in the project, which have emerged over the realization of the same.
5.2 Conclusions

We must highlight the importance of implementing a prototype like this, because it allows students to do so is not difficult to adapt to a new city.

But the work could be further continued because every application is susceptible of being enhanced and much more if it is a prototype. As indicated below some extensions:

- Upload the application to a server, and publish the application.
- Creating an interface for mobile devices, at least for the more important actions of system.
- Creation of a system to recover passwords.
- Adding a new role to the system: Administrator.
- Using AJAX for dynamic queries is faster.
- Ability to notice of notifications via email.
- Addition of a new module for the user to ask questions about a city.
- Identify any user-centric improvements and implement them, always trying to bring that part to the future achievement of a final system.

On a personal basis I have to emphasize that the realization of this project has helped me to implement many of the methodologies and skills acquired during my years of education, and others I have had to acquire during their development. The most positive thing is I get improvement in my ability to create user–centered designs. I think it is very good for an engineer, and always try to get this in the future.

Finally, I would like to turn this into a real project, because I think it is a not covered need in the society.
6 References


[5] Oracle (base de datos)


