Bachelor’s Thesis

Web Development of a Social Network for Events

February 6\textsuperscript{th}, 2014

Software Engineering Specialization

Author: Ricardo Ramos González
Supervisor: Enric Mayol Sarroca
Acknowledgments

This project would not have been possible without the support of many people.

I would like to thank my supervisor Enric for his helpful suggestions, corrections and constructive feedback during the planning and development of this project.

Secondly, I would like to thank my friends and colleagues at the Barcelona School of Informatics. Thank to friends who have always been by my side during this journey.

Finally, I would also like to thank my family. Specially, thank to my parents and my brother for their support and encouragement throughout my career.

They have given me the strength to keep going.
List of contents

List of figures .................................................................................................................. 4
List of tables ..................................................................................................................... 5
1. Introduction .................................................................................................................. 6
   1.1. Abstract ................................................................................................................ 6
   1.2. Context ................................................................................................................. 7
   1.3. Purpose ................................................................................................................. 7
2. Analysis .......................................................................................................................... 8
   2.1. State of the art ..................................................................................................... 8
   2.2. Project Scope ..................................................................................................... 13
   2.3. Objectives .......................................................................................................... 14
   2.4. Requirements ...................................................................................................... 15
   2.5. Risks ..................................................................................................................... 19
   2.6. Methodology ....................................................................................................... 20
   2.7. Project planning ................................................................................................. 21
   2.8. Project budget ..................................................................................................... 26
   2.9. Sustainability and social responsibility ............................................................... 29
3. Specification .................................................................................................................. 31
   3.1. Iteration 1 ............................................................................................................ 31
   3.2. Iteration 2 ............................................................................................................ 45
   3.3. Iteration 3 ............................................................................................................ 50
   3.4. Complete conceptual model .............................................................................. 59
4. Design ............................................................................................................................ 61
   4.1. Technology .......................................................................................................... 61
   4.2. Architecture ....................................................................................................... 63
   4.3. Iteration 1 ............................................................................................................ 66
   4.4. Iteration 2 ............................................................................................................ 83
   4.5. Iteration 3 ............................................................................................................ 89
List of figures

Figure 1. A social network diagram ........................................................................................................ 8
Figure 2. Meetup logo ............................................................................................................................. 9
Figure 3. Eventbrite logo ........................................................................................................................ 10
Figure 4. Plancast logo ........................................................................................................................... 10
Figure 5. Doodle logo ............................................................................................................................. 10
Figure 6. Elgg logo .................................................................................................................................. 11
Figure 7. Diaspora logo ........................................................................................................................... 11
Figure 8. BuddyPress logo ....................................................................................................................... 11
Figure 9. Agile methodology schema ....................................................................................................... 20
Figure 10. Screenshot of Gantt diagram for initial planning ................................................................. 23
Figure 11. Example of a MVC request in CakePHP .................................................................................. 63
Figure 12. Example of associations and linking models on CakePHP ..................................................... 64
Figure 13. Create table example ............................................................................................................ 98
Figure 14. Belongs to and has many examples ...................................................................................... 99
Figure 15. Has and belongs to many example ....................................................................................... 99
Figure 16. Develop environment schema ............................................................................................... 113
Figure 17. Deployment schema .............................................................................................................. 114
Figure 18. Final Gantt diagram .............................................................................................................. 116
List of tables

Table 1. Estimated time for each task ................................................................. 22
Table 2. Hourly wage for human resources .......................................................... 26
Table 3. Human resources budget ...................................................................... 27
Table 4. Software resources budget .................................................................... 27
Table 5. Hardware resources budget ................................................................... 28
Table 6. Project budget ......................................................................................... 28
Table 7. Example methods find CakePHP .............................................................. 65
1. Introduction

1.1. Abstract

1.1.1. English
The project consists in the creation, configuration and management of a social network to organize and to share information about events. The users of this social network can be part of the events organization. They have the possibility to decide on different aspects of the events such as day, time, place, assigned tasks to organize the event and other relevant aspects. Moreover, the users will be able to interact with each other, creating groups and friend relationships. The technological platform on which the project is based is the web, a web site known for its accessibility and usability.

1.1.2. Catalan
El projecte consisteix en la creació, configuració i gestió d’una xarxa social per organitzar i compartir informació d’esdeveniments. Els usuaris d’aquesta xarxa social poden formar part de l’organització d’aquests esdeveniments. Tindran la possibilitat de decidir sobre diferents aspectes d’aquest esdeveniment com poden ser el dia, el lloc, la localització, assignar tasques per organitzar l’esdeveniment i d’altres aspectes rellevants. A més, els usuaris podran interaccionar entre ells creant grups i fent amics. La plataforma tecnològica en la que es basa el projecte és la web. Un lloc web caracteritzat per la seva accessibilitat la seva facilitat d’ús i la seva accessibilitat.

1.1.3. Spanish
El proyecto consiste en la creación, configuración y gestión de una red social para la organización y compartición de información sobre eventos. Los usuarios de esta red social pueden formar parte de la organización de estos eventos. Tendrán la posibilidad de decidir sobre diferentes aspectos de un evento como pueden ser el día, lugar, localización, asignar tareas para organizar el evento y otros aspectos relevantes. Además, los usuarios pueden interactuar entre ellos creando grupos y haciendo amigos. La plataforma tecnológica en la que se basa el proyecto es la web. Un sitio web caracterizado por su accesibilidad y facilidad de uso.
1.2. Context

This project is part of a final thesis degree at Barcelona School of Informatics supervised by Enric Mayol Sarroca. It is a project conceived and designed by the author of this document, Ricardo Ramos González. Therefore, it is a personal project guided by the supervisor.

The web is a good platform where a lot of social networks have grown and many of them are part of everyday life for millions of people. Specifically, this project is aimed at web users who want to have a web-based social network specialized in the dissemination and management of events.

In the future, the people who have really actively participated in this social network will be the ones to obtain potential benefits. Besides the non-economic benefits that users can obtain derived from the use of the social network website.

1.3. Purpose

As we have seen, the main idea of the project revolves around events. There are many types of them, but all of them have many features in common which respond to the following questions: when does an event take place? Where does it occur? Who is involved in the event? What is the purpose of event? And how is it organized? We are going to construct a social network where users can answer these questions.

Some examples of events could be the following ones: a group of friends who want to organize a barbecue and need to assign tasks and organize the lunch; an official event such as a conference at Barcelona Schools of Informatics and they want to promote and control the attendees of this conference; and finally, a simple event like a football match between friends.
2. Analysis

2.1. State of the art

Nowadays, social networks have become a revolution that changes the way in which people interact with others and gives the opportunity to expand and strengthen relations with the people around them.

2.1.1 What is a social network?

A social network can be defined as a group of people or entities that share interests and that are linked by common characteristics such as kinship, friendship and other kinds of relationship.

To understand and define a social network, we have to make a brief introduction to graph theory. A graph is an easy way to see the structure and the composition of a social network. It is a set of vertices and edges or arcs. Each edge is a line or arc joining two vertices of the graph or vertex to itself.

Applied to social networks, nodes are people or entities and the links between nodes are relationships between them. Therefore, nodes are interrelated and interact with each other forming an ordered set of elements with similar purposes. (Lozares, 1996). In figure 1, we can see an example of a social network diagram.

2.1.2 Web-based social network

Web-based social networks are virtual communities where users interact with people around the world who share interests or likes. It works as a communication platform that connects known people or people you want to meet, allowing them to share resources such as photos, videos, news, and so on. Moreover, users can access and manage these resources themselves.

The basic necessary technology to create a web-based social network, it is not much different from the technology that is used for any other website. On the one hand, we have a hosting server which contains the website logic and the database. On the other side there is the client side. Through a web browser, the client sends requests to the
server which immediately processes the response and displays it into a customer-friendly format to the user.

The technological combination widely used is LAMP which stands for the following open source software: Linux\(^1\) operating system, Apache\(^2\) web server, MySQL\(^3\) database management and PHP\(^4\) programming language, although sometimes it can be Python or Perl.

As from now in this section, when we talk about social networks, we will be referring to web-based social networks.

### 2.1.3 Classification of social networks

**Horizontal**

Social networks that do not have a specific type of users or that are not oriented to a specific activity. Some examples are Facebook, Twitter or Google+.

**Vertical**

On the other hand, vertical social networks are specialized in a type of users or specific activity. For example, LinkedIn, which is for people in professional occupations or YouTube which specialized in video sharing.

### 2.1.4 Events on social networks

Some horizontal networks like Facebook or Tuenti have widely used functionalities to manage events and it is an important part from the social network. However, in this section we are going to focus on specialized event-based networks such as:

- **Meetup**
  
  Meetup is an online social network portal that facilitates offline group meetings in various localities around the world. Meetup allows members to find and join groups unified by a common interest, such as politics, books, games, movies, health, pets, careers or hobbies.

---

\(^1\) Linux is an operating system that uses the Unix kernel. [http://www.linux.org](http://www.linux.org)

\(^2\) Apache web server is a free, secure and open source software. It is developed within the collaborative software project of the Apache Software Foundation. [http://www.apache.org/](http://www.apache.org/)

\(^3\) MySQL is a multithreaded, multi-user management database developed by MySQL AB as free software. [http://www.mysql.com/](http://www.mysql.com/)

\(^4\) PHP is an interpreted programming language, high performance, which along with HTML allows creating dynamic web pages. [http://php.net/](http://php.net/)
Eventbrite is an online ticketing service whose mission is to make it as easy as possible for people to host, discover and attend events. Eventbrite provides online resources for event management, including registration, ticket sales and event promotion.

Plancast helps to find out about events and other social activities. Plancast users share the most interesting things on their calendars, making this the place to find out about things to do, straight from the people doing them.

Doodle is not a social network; actually, it is a web tool to schedule events. It is a free internet calendar tool for time management, and coordination of events. Users are asked to determine the best time and date to schedule an event.

To summarize, these social networks or web tools have events as a central element of discourse. However, each of them gives a different approach to events.

In the case of Eventbrite, they put special emphasis on ticketing sales and promotion of events. The company provides a platform for this purpose and they obtain profits from the ticketing sales of users. Nevertheless, users can create and organize their own free events with no charge.

In contrast, Plancast and Meetup are closer to the idea of the project that we are going to develop. Meetup focuses on groups of people with similar interests. In both cases, the platforms have user profiles and they are able to create and share events.

None of the previously social networks goes beyond the creation and management of events and they give poor user interaction. Users cannot decide on various aspects of events, to simplify they can only join and sometimes promote events. The key of our project is the aim to provide a more flexible events management and a more usable interface for user.
2.2.5 Social network software

Once we have the idea of the social network that we want to build, we have to think about how we are going to reflect this idea on a website. To do this, we have to consider different software or frameworks that would be useful to achieve our objectives.

- **Elgg**
  Elgg is open source social networking software that provides individuals and organizations the components needed to create an online social network. It is free to download and use, licensed under the terms of the GNU General Public License\(^5\). In addition, Elgg runs on the LAMP platform. It has functionalities for setting up profiles, sharing files, adding friends, blogging, aggregating RSS, content tagging, and social graphs. (Sharma, 2008)

- **Diaspora**
  It is a free personal web server that implements a distributed social networking service. It built on Ruby on Rails\(^6\). In addition, Diaspora is an open source web server that allows implementing decentralized and private social networks as an alternative to Facebook and other social networks. It is currently in development.

- **Buddypress**
  BuddyPress is an open source social network tool. It is a WordPress\(^7\) extension that allows the addition of a social network to the existing installation. BuddyPress is essentially a set of plugins for WordPress. Each plugin adds a different characteristic, and each of them is responsible for a different functionality such as internal messaging between users, friendship, and others basic functionalities for a social network.

---

5 GNU General Public License (GNU GPL or GPL) is a free software license, which guarantees end users the freedoms to use, study, share, and modify the software. [http://www.gnu.org/copyleft/gpl.html](http://www.gnu.org/copyleft/gpl.html)

6 Ruby on Rails is an open source web application framework which runs on the Ruby programming language. [http://rubyonrails.org/](http://rubyonrails.org/)

7 WordPress is a free and open source blogging tool and a content management system (CMS) based on PHP and MySQL which runs on a web hosting service. [https://wordpress.org/hosting/](https://wordpress.org/hosting/)
Other examples of web software tools that can help us build our social network would be Drupal or Joomla. Both are open source content management system (CMS) which, with the appropriate modules, we could construct a social network.

As we have seen on this section, there are a lot of solutions to develop a social network and it seems that we do not need to start from scratch. However, we believe it should be created from scratch for the following reasons:

- A greenfield project brings the opportunity to create and adapt our social network to our liking with no dependency of the chosen framework.
- For the developer it is easier to maintain and extend because the developer knows all aspects of the software and how it works.
- Some of the software explained before have an inexistent or poor documentation; therefore it could derive into delays and inefficiency on the development of the social network.
- Finally, we believe it is a good opportunity to learn how to do a project from scratch. In addition, it will also be an opportunity to learn in depth about the technological difficulties behind a social network.
2.2. Project Scope

The project consists in the development of a website and it is not part of previous software. However, we are going to use different web frameworks to develop the web application.

The project will be divided into two parts. The first one will be the definition and implementation of the social network. This part includes the relationships between users, how they can interact with each other, register, login, make friends and manage groups. Users will be able to share their profiles with other users or they will be able to navigate to user profiles through events or groups. Certain complex functionalities for social networks such as chat and friends’ recommendations are beyond the scope of this project.

The second and most important part of this project is the definition and implementation of events logic. First, we have the basic part such as creation, administration, sharing and search of events. There is also a more complex part that includes how users interact with the event. Interactions like polls, assigned tasks to be completed by users, a wall to communicate between users and events search.

The project is addressed to an audience who is a regular user of social networks and is looking for a web tool to manage events in a fast, simple and intuitive way. Initially, the geographical scope of the project encompasses the entire national territory.

The main base technology of the project is web. However, we will use a framework to make a responsive web design for mobile devices, but it is not the main objective of this project.
2.3. Objectives

2.3.1 Main objective

Generally, the main objective of all web social networks is to facilitate communication and share information between users or organizations. But, specifically the main objective of this project is:

“To design and develop a website to support a social network oriented to events”

2.3.2 Objectives

In addition, there are the following specific objectives:

- To learn how a social network works internally.
- To offer a website which facilitates the organization of events.
- To develop a usable and useful website.

2.3.3 Technical competences

Below, the technical competences associated to this project:

- To develop, maintain and evaluate complex and/or critical software systems and services.
- To identify, evaluate and manage potential risks related to software building which could arise.
- To specify, design, implement and evaluate databases.
- To define and manage the requirements of a software system.
2.4. Requirements

2.4.1. Functional requirements

Requirement: User registration.
Description: The social network has to allow the registration of new users. The registration process can be filling out a registration form or via the following accounts from other social networks: Google, Facebook or Twitter account.

Requirement: User login.
Description: Users has to access the social network through a username and password or with the Google, Facebook or Twitter account.

Requirement: User profile.
Description: All system users must have a profile where they can share personal information such as name, city, age, email or profile picture.

Requirement: Users relationships.
Description: The system allows the relationship between users. They can add friends and they can create or join to group of friends.

Requirement: Event management.
Description: System users can create, edit and delete events. Title, description, time and location information is needed to create an event. This information is managed by one or more event administrators.

Requirement: Event search.
Description: The system must have a simple search engine to filter and find events by location, time, title and description.

Requirement: Invite people to an existing event.
Description: Event administrators can invite their friends or groups to attend the event.

Requirement: Join an event.
Description: Invited or non-invited users to an existing event can indicate this event as a participant, non-participant or unanswered.

Requirement: Communication wall for events.
Description: The system must have a public communication wall where participants can interact on event page.

Requirement: Surveys management for events.
Description: The event managers can create polls to ask about different aspects of the event.

Requirement: Task management for events.
Description: The system has to allow users to assign tasks. Tasks that are needed to carry out the event.

2.4.2. Performance requirements

Requirement: Response time
Description: The website has to respond to users requests as quickly as possible.

Requirement: Capacity
Description: The system must be capable of supporting a large number of concurrent users without a significantly affecting performance.

Requirement: Reliability and availability
Description: The system must be available at all times and it must be reliable.

Requirement: Scalability and extensibility
Description: The system has to be easy to escalate and to extend to higher demands.

2.4.3. Security requirements

Requirement: Data backups
Description: The system has to perform regular backups.

Requirement: Data privacy
Description: The system must ensure the privacy of the data that users introduce into the social network.

Requirement: Immunity
Description: The system must be prepared to prevent computer attacks such as SQL injection, DDoS, virus and spyware.

2.4.4. Maintenance and support requirements

Requirement: Maintainability
Description: For future updates, the development process of the social network must be well documented.

Requirement: User guide
Description: This project must have a user guide.

2.4.5. Look and feel requirements

Requirement: Website design
Description: The website design must be attractive and it must invite user to participate on it. Moreover, the design has to be useful and simple.

2.4.6. Usability and humanity requirements

Requirement: Usability
Description: The website must be easy to use and give feedback to the user at all times.

Requirement: Accessibility
Description: The system has to be accessible to users with a diverse range of sight and cognitive ability.
Requirement: Language  
Description: The website language must be English

Requirement: Localization  
Description: The system must be prepared to allow extending to other languages.

2.4.7. Operational and environmental requirements

Requirement: Responsive Web Design  
Description: The website design has to adapt for mobile browsers.

Requirement: Supported browsers  
Description: The website must be compatible with Google Chrome and Mozilla Firefox browsers mainly.

Requirement: Open-source  
Description: Software has to be developed with open-source tools and frameworks.

2.4.8. Cultural and political requirements

Requirement: Data Protection Law  
Description: The Spanish law of data protection (LOPD) must be ensured.

Requirement: Web standards  
Description: The website must satisfy the W3C web standards.
2.5. Risks

It is a greenfield project, and there are many risks that we must consider:

- Problems with different frameworks used to develop the website. Depending on the framework or programming language selected, some problems, such as configuration, compatibility or unexpected bugs can appear. To avoid this risk we are going to use widely known and mature technologies.

- Incorrect time estimation on the scheduling plan. Maybe some functionality takes more time than it was initially expected. In that case, we are going to classify and to prioritize them to prevent deadlocks between some complex functionalities.

- Developing a source code with less functionality than originally expected. This risk is derived from the incorrect time estimation. We must ensure that all expected functionalities are scheduled as realistically as possible.

- Low experience in software projects. The experience of the director of this project will be important to guide and recommend the best practices to achieve the specified objectives.

- Small community of users. We will give the possibility to register with other social network accounts. So that users can access the website with just one click.
2.6. Methodology

The most appropriate methodology to use for this project has to be an agile methodology. It is an appropriated development methodology for this type of project and moreover it is appropriate because we need to be as flexible as possible. Hence, flexibility will give us the opportunity to respond to changes suitably and avoid the risks detailed previously.

Before the first iteration there is an initial phase that includes: planning, feasibility study, analysis and design of the software.

Iteration has four stages: analysis of the sprint, implementation, testing and finally integration. The duration of the iteration must be about 3-4 weeks. Upon the completion of iteration, it generates a deliverable which corresponds to a specific functionality of the project. Once the functionality is tested and validated, it will be part of the final deliverable.

The tests aim to prove and validate functionality. We will use test accounts, invented events and other necessary data. In addition it will be tested with the isolated functionality and tested with the functionality integrated.

After the finalization of iteration we must analyze, revise and update the plan project if it is necessary. Moreover, we must prepare the next iteration.

8 Agile development methodology is a group of software development methods based on iterative and incremental development.
2.7. Project planning

2.7.1. Initial estimated project duration

“The project starts on June 25\textsuperscript{th}, 2013 and the estimated completion date is on January 6\textsuperscript{th}, 2014”

2.7.2. Considerations

As we have seen on the project scope section, we are going to use an agile methodology. Therefore, the initial planning could be revised or modified depending on the new requirements and needs that will appear during the development of the project.

The period from August 12\textsuperscript{th}, 2013 to September 1\textsuperscript{st}, 2013 both included, corresponds to summer holidays. Hence, the project will remain on hold during these three weeks.

2.7.3. Planning

Project planning and feasibility
This stage corresponds to the project management course; it includes project scope, project planning, project budget and initial state of the art.

Project analysis and design
The next stage after the project planning is the analysis and subsequent design. On the analysis, we are going to define accurately objectives, requirements, functionalities and use cases. In addition we will expand the state of the art and the study of different technologies we will apply on the project development.

Furthermore, the project design includes the definition of software architecture, diagrams, and so on; to sum up, we will apply all the software engineer skills learned on the degree to design the website.

Project iterations
Iteration has four phases: analysis of the iteration, implementation, test and integration. Each iteration depends on the previous iteration; in addition, it also depends on from the project analysis and design. The project will have the following iterations:
Iteration 0 - Initial configuration of the software architecture and software environment: this stage will be focused on the preparation and tuning up to start the software implementation.

Iteration 1 - Implementation of the social network: this iteration is essential for the project. Therefore, we are going to precisely analyze and study the best solution to develop the social network. This iteration includes functionalities such as register, login, user profile, friendship and groups.

Iteration 2 - Implementation of the events logic: this is the shorter iteration which includes implementation. The included functionalities are the following: creating events, updating them, delete them and finally join it and share it with friends or groups.

Iteration 3 - Implementation of the events interaction logic: finally, the last iteration includes the interaction functionalities with events: polls, communication, tasks and so on. Initially it will be considered as a single iteration, but depending on the difficulty of functionalities, it could be split into smaller iterations.

Final stage

Finally, once the web application is implemented and tested, we will complete the user manual and the thesis report. In addition, we will prepare the final presentation.

2.7.4. Estimated time

The following table represents the duration in hours of the activities described earlier. Furthermore, the total estimated duration of the project is included.

<table>
<thead>
<tr>
<th>Task</th>
<th>Estimated time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Planning and feasibility</td>
<td>80 hrs.</td>
</tr>
<tr>
<td>Analysis and design</td>
<td>80 hrs.</td>
</tr>
<tr>
<td>Iteration 0 – initial configuration</td>
<td>40 hrs.</td>
</tr>
<tr>
<td>Iteration 1 – social network</td>
<td>110 hrs.</td>
</tr>
<tr>
<td>Iteration 2 – events logic</td>
<td>60 hrs.</td>
</tr>
<tr>
<td>Iteration 3 – events interaction logic</td>
<td>150 hrs.</td>
</tr>
<tr>
<td>Final stage</td>
<td>80 hrs.</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>600 hrs.</strong></td>
</tr>
</tbody>
</table>

*Table 1. Estimated time for each task*

On the following section, we can see the different scheduled tasks.
2.7.5. Gantt diagram

Figure 10. Screenshot of Gantt diagram for initial planning
2.7.6. Action plan

The agile methodology gives us the opportunity to adapt and revise this initial planning. Estimated time for each task has been calculated with a wide margin. However, if for any reason a task takes more time than the initially estimated, the estimated hours will be expanded and the next tasks will be adapted to this unexpected situation. On the other hand, if a task has been finished early, the next task will begin immediately.

Moreover, the final stage is scheduled to be complete in three weeks, more than sufficient time to complete this task. Therefore, we will have a margin of one or two weeks to readapt the planning if it is necessary. At the end of each iteration or task, we will schedule a follow-up meeting with the director of the project, to analyze the state of the project. In addition, at the start of the iteration 3 a follow-up report monitoring will be delivered to the director, a report which will be evaluated.

To summarize, the average time per week dedicated to this project is between 30-40 hours. Hence, if we follow the planning precisely, we can ensure the project completion. However, if we foresee that the project duration will be more than the expected or there are appears delays due to illness, deadlocks, expansion of functionalities, or others; we will increase the hours per week spent to the project or we will streamline tasks to make them in the shortest possible time.

2.7.7 Resources

To achieve and complete accurately this plan we are going to use these resources:

- **Hardware:**
  - Laptop ASUS A53SV-SX538V. It will be used as a main hardware to develop the entire project.
  - Notebook ASUS eeePC 1011CX. Notebook used for testing the website on small screens and occasionally to develop the project.

- **Software:**
  - Windows 8 Professional. Included operative system in hardware resources.
  - Microsoft Word 2013. Software used to complete the documentation of the project. (Planning and feasibility, analysis and design, final stage and if it is necessary it will be used during the entire project to modify documentation).
  - Microsoft Project 2013. Used to schedule the project planning.
- Microsoft Visio 2013. A diagramming and vector graphics. It will be used at analysis and design task.

- Astah. UML modeling tool. (Planning and feasibility, analysis and design)

- Adobe Reader PDF. Used to visualize PDF documents.

- Sublime Text. Source code text editor. Used in iterations to develop the website.

- GIMP. It is an image retouching and editing tool. It will be used for the frontend development of the website (iterations 1 to 3).
2.8. Project budget

After the project planning, in this document we are going to define the project budget. Each task or resource has a cost; we will divide these costs in human resources, software resources and hardware resources. Therefore, it is direct costs to develop the project.

This project budget will not consider indirect costs such as internet connection, electricity bill or the price of renting an office.

2.8.1. Estimated budget

“The estimated cost of this project is 22,859,21 €”

In the sections below we are going to analyze this estimated cost precisely.

2.8.2. Human resources

We are going to consider the following roles adopted during the different activities of the project and the corresponding salary per hour.

<table>
<thead>
<tr>
<th>Role</th>
<th>Salary per hour</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project Manager</td>
<td>50 €</td>
</tr>
<tr>
<td>Requirements Engineer</td>
<td>40 €</td>
</tr>
<tr>
<td>Software Developer Engineer</td>
<td>30 €</td>
</tr>
<tr>
<td>Software Tester</td>
<td>20 €</td>
</tr>
</tbody>
</table>

Table 2. Hourly wage for human resources

On the table below we can see the estimated cost for each task or iteration, and the total amount of the human resources budget.

<table>
<thead>
<tr>
<th>Task</th>
<th>Estimated hours</th>
<th>Estimated cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>Planning and feasibility</td>
<td>80 hrs.</td>
<td>4,000 €</td>
</tr>
<tr>
<td>Project Manager</td>
<td>80 hrs.</td>
<td>4,000 €</td>
</tr>
<tr>
<td>Analysis and design</td>
<td>80 hrs.</td>
<td>3,700 €</td>
</tr>
<tr>
<td>Project Manager</td>
<td>50 hrs.</td>
<td>2,500 €</td>
</tr>
</tbody>
</table>
2.8.3. Software resources

The useful life for each software resource is three years. The cost includes VAT. With the useful life, we are going to calculate the amortization of the software for the project duration. These licenses have been previously acquired; however we are going to include them on the budget. The costs of software resources are the following:

<table>
<thead>
<tr>
<th>Item</th>
<th>Cost</th>
<th>Real cost imputed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Windows 8 Professional x2</td>
<td>279,99 €</td>
<td>93,33 €</td>
</tr>
<tr>
<td>Microsoft Office 2013</td>
<td>119 €</td>
<td>19,83 €</td>
</tr>
<tr>
<td>Microsoft Project Standard 2013</td>
<td>769 €</td>
<td>128,17 €</td>
</tr>
<tr>
<td>Microsoft Visio Standard 2013</td>
<td>399 €</td>
<td>66,50 €</td>
</tr>
<tr>
<td>Astah professional</td>
<td>149 €</td>
<td>24,83 €</td>
</tr>
<tr>
<td>Sublime Text</td>
<td>55 €</td>
<td>9,17 €</td>
</tr>
<tr>
<td>GIMP</td>
<td>- €</td>
<td>- €</td>
</tr>
<tr>
<td>Total</td>
<td>2,050,98 €</td>
<td>341,83 €</td>
</tr>
</tbody>
</table>
2.8.4. Hardware resources

Considering that the cost for each item includes VAT and also a useful life of four years. Moreover, the hardware will be available for six months to develop and test the project. The real hardware costs imputed for the duration of this project are the following:

<table>
<thead>
<tr>
<th>Item</th>
<th>Cost</th>
<th>Real cost imputed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Laptop ASUS A53SV-SX538V.</td>
<td>660 €</td>
<td>82,50 €</td>
</tr>
<tr>
<td>Notebook ASUS eeePC 1011CX</td>
<td>279 €</td>
<td>34,88 €</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>939 €</strong></td>
<td><strong>117,38 €</strong></td>
</tr>
</tbody>
</table>

*Table 5. Hardware resources budget*

2.8.5. Total budget

<table>
<thead>
<tr>
<th>Concept</th>
<th>Cost</th>
<th>Real cost imputed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Human resources</td>
<td>22.400,00 €</td>
<td>22.400,00 €</td>
</tr>
<tr>
<td>Software resources</td>
<td>2.050,98 €</td>
<td>341,83 €</td>
</tr>
<tr>
<td>Hardware resources</td>
<td>939,00 €</td>
<td>117,38 €</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>25.389,98 €</strong></td>
<td><strong>22.859,21 €</strong></td>
</tr>
</tbody>
</table>

*Table 6. Project budget*

2.8.6. Budget monitoring

At the end of each task, we are going to analyze and control the cost of the project. Specifically, we are going to monitor the iterations. We will calculate and compare the real project cost with real hours dedicated to the project. In this manner, we will be able to react to unforeseen situations. In addition, we will define a performance indicator (KPI) to have an estimation of the relation between estimated cost and real cost.

The estimated cost of this project is economically feasible; we have a project planning and realistic costs. Moreover, we have a budget monitoring to control these costs. However, it is difficult to calculate the net present value because we do not have the necessary information to calculate it, for example the cash flows.
2.9. Sustainability and social responsibility

2.9.1. Social impact

We are going to build a social network; therefore the social aspect is an important aspect to consider. Thereby, we will explain in a few lines what provides this social network to society.

Web-based social networks technology offers added value, allowing users to create groups and to socialize. The immediacy of communication, transmission of views and content, it makes them a powerful social tool. But we must not forget that misuse of social networks can lead to problems such as loss of privacy, access to inappropriate content, harassment, and so on.

Specifically, our social network will help users organize themselves to hold events or meetings between friends, so that participants can be involved directly or indirectly. Used correctly and in these times of great social change, this social network could be a good instrument for the organization of social movements for example.

On the other hand, an example of misuse of our social network could be an event which many people attend without the consent of its creator or creators. This situation has already occurred before\(^9\), causing serious problems where police had to intervene.

To summarize, all these aspects are taken into account when developing our social network, so we will offer a useful social network with a good service and also a social network which safeguards its users.

2.9.2. Environmental impact

We believe that the environmental impact does not suppose an important aspect in this project. Since the web servers where we are going to host the website, will be outsourced to a professional hosting company and therefore their servers will be optimized to minimize the environmental impact. However, if we decide to host the website in our own servers, we would have to consider various aspects such as energetic consumption, bandwidth, and so on.

In both cases, the application will be as efficient as possible trying to use algorithms that do not require a big data processing or do not spend a high amount of time to process information. In this manner, for example we could reduce the CPU time used to process this data to reduce the electric consumption.

\(^9\) [http://www.telegraph.co.uk/technology/facebook/8012043/Girl-14-fears-21000-party-guests-after-Facebook-invite-blunder.html]
2.9.3. Economic impact

We can assure that the project is economically feasible, since the necessary resources to carry out the project have already been accurately detailed; besides an action plan to avoid increasing costs has also been taken into account.

Moreover, the project is committed to the use of open source software to cheapen costs, since in the current global economic situation and the Barcelona School of Informatics more specifically, cost reduction is an important aspect to consider. Although the use of non-open source software to complete the project has been initially considered in the budget, if it is possible we are going to use the software licenses which the Barcelona School of Informatics has, to decrease costs some more.
3. Specification

Having done the analysis and understood the project requirements, in this section we will detail what the system will be able to do or not. As we followed an agile methodology, the section is divided into the three iterations detailed before.

Each iteration involves functionalities of a part of the system; therefore we are going to define the conceptual model, use cases and operation contracts of the system that affects this iteration separated, following the agile methodology. This strategy will be followed on design section too.

At the end of this section we will define the complete conceptual model.

3.1. Iteration 1

3.1.1. Conceptual model
### Constraints

- Each class must have a unique identifier (id).
- User `username` must be unique.
- User `social_provider` and `social_id` must be unique.
- User `user_from` and `user_to` must be different users.
- For one User and a Group only must to exist a GroupUser instance.
- The friend relationship must be bidirectional, two users are considered friends if one are friend of each other and vice versa.

#### 3.1.2. Use cases

Use case: **User register**

**Actor:** Unregistered user

**Precondition:** The unregistered user does not have an account.

**Main scenario:**

1. The user enters in to the public homepage.
2. The user fills the registration form with username, password and password confirm.
3. The user clicks on the signup button.
4. The system saves user’s data and redirects to user homepage.

**Extensions:**

2a. External accounts

   1. The user clicks on the login with... (Google, Facebook or Twitter account) button.
   2. The user goes to the corresponding login form of the social network and accepts permissions.
   3. Once signed in, the external social network redirects to step 4.

4a. Invalid data

   1. The system shows a dialog message with the information.
   2. Go to step 3.

---

**Use case:** User login

**Actor:** User

**Precondition:** The user has an account.

**Main scenario:**

1. The user enters into the public homepage.
2. The user enters his username and password.
3. The user press the login button
4. The system saves user’s data and redirect to user homepage.

**Extensions:**

2a. External accounts

   1. The user clicks on the login with... (Google, Facebook or Twitter account) button.
   2. If user is logged into the corresponding external social redirects to step 4
   3. Otherwise the user log in into the external social network and redirects to step 4.

3a. Invalid data

   1. The system shows a dialog message with the information.
   2. Go to step 2.

---

**Use case:** User logout

**Actor:** User

**Precondition:** The user is logged in to the social network.
Main scenario:
1. The user clicks on the logout link.
2. The system logs out user and it redirects to the public homepage.

Use case: View user profile
Actor: User
Precondition: The user is logged in to the social network.
Main scenario:
1. The user enters in to the user profile page.
2. The system shows information about user such as name, surname, email, gender, city, birthday and a profile photo, in addition the friends list.

Use case: Edit user profile
Actor: User
Precondition: The user is logged in to the social network.
Main scenario:
1. The user enters in to the user profile page.
2. The user presses the edit profile button.
3. The user Fill the fields with information such as name, surname, email, gender, city, birthday and a profile photo.
4. The user clicks on the save button.
5. The system saves user’s data and redirects to edit user profile page.

Extensions:
4a. Invalid data
   1. The system shows a dialog message with the information.
   2. Go to step 3.

Use case: Change username
Actor: User
Precondition: The user is logged in to the social network.
Main scenario:
1. The user enters in to the user profile page.
2. The user presses the edit profile button.
3. The user fills the fields with the new username and password.
4. The user clicks on the save button.
5. The system saves user’s data and redirects to edit user profile page.
Extensions:
4a. Wrong password or exists username
   1. The system shows a dialog message giving feedback to user.
   2. Go to step 3.

Use case: Change password
Actor: User
Precondition: The user is logged in to the social network.
Main scenario:
1. The user enters in to the user profile page.
2. The user presses the edit profile button.
3. The user fills the fields with the current password and the new password with another field password to match.
4. The user clicks on the change username button.
5. The system saves user’s data and redirects to edit user profile page.

Extensions:
4a. Wrong current password or new password does not match
   1. The system shows a dialog message giving feedback to user.
   2. Go to step 3.
Use case:  Add friend
Actor:  User
Precondition:  The user is logged in to the social network.
Main scenario:
1. The user enters in to a user profile.
2. The user presses the add friend button.
3. The system sends a petition to the owner of this profile.
Use case: Respond friend request  
Actor: User  
Precondition: The user is logged in to the social network and has friend requests.  
Main scenario:
1. The user enters in to his homepage or in to the profile of the possible friend.  
2. The system shows received friend request with two options.  
3. The user accepts or declines the friend request clicking the corresponding button.  
4. The system saves the friend relationship or mark as rejected relationship between users.

Use case: Cancel sent friend request  
Actor: User  
Precondition: The user is logged in to the social network and sent a friend request to user.  
Main scenario:
1. The user enters in to his homepage or in to the profile of the possible friend.  
2. The system shows sent friend request with the cancel option.  
3. The user cancels the sent friend request.  
4. The system cancels the friend relationship between users.

Use case: Stop being friends  
Actor: User  
Precondition: The user is logged in to the social network and is friend of the user.  
Main scenario:
1. The user enters in to the friend profile.  
2. The system shows a button to stop being friend.  
3. The user presses the button.  
4. The system deletes the friend relationship between users.

Use case: Create a group  
Actor: User  
Precondition: The user is logged in to the social network.  
Main scenario:
1. The user enters in to his homepage.  
2. The user clicks on create a new group button.
3. The user fills the fields to give a name to the group, a brief description, the privacy and add from friends list the components of the group.
4. The user presses the create button.
5. The system saves the information group and invited members receive notification on his personal page and assign creator as administrator of this group.

Extensions:

4a. Invalid data
   1. The system shows a dialog message with the information.
   2. Go to step 3.

---

Use case: View group
Actor: User
Precondition: The user is logged in to the social network.
Main scenario:
1. The user enters in to the group page.
2. The system shows information about group such as name, a brief description and the list of members.

---

Use case: Edit group information
Actor: User group admin
Precondition: The user is logged in to the social network, takes part from group and is the administrator of the group.
Main scenario:
1. The user enters in to the group page info.
2. The user clicks on edit group information button.
3. The user fills the fields to update name to the group, brief description and the privacy.
4. The user presses the create button.
5. The system saves the information group.

Extensions:

4a. Invalid data
   1. The system shows a dialog message with the information.
   2. Go to step 3.

---

Use case: Add users to a group
Actor: User group admin
**Precondition:** The user is logged in to the social network, takes part from group and is the administrator of the group.

**Main scenario:**
1. The user enters in to the group page info.
2. The user clicks on edit group information button.
3. The user selects users from his friends list.
4. The user presses the add user button.
5. The system add user to the group and invited members receive notification on his personal page.

**Extensions:**
4a. Unknown user or user is already in to the group.
   1. The system shows a dialog message with the information.
   2. Go to step 3.

---

**Use case:** Assign admins to a group

**Actor:** User group admin

**Precondition:** The user is logged in to the social network, takes part from group and is the administrator of the group.

**Main scenario:**
1. The user enters in to the group page info.
2. The user clicks on edit group information button.
3. The user selects users from his friends list.
4. The user presses the assign admin button.
5. The system assigns user as administrator and list this group as administrator on the new user admin homepage.

**Extensions:**
4a. Unknown user or user is already admin of the group.
   1. The system shows a dialog message with the information.
   2. Go to step 3.

---

**Use case:** Join group

**Actor:** User

**Precondition:** The user is logged in to the social network.

**Main scenario:**
1. The user enters in to public group page info.
2. The user clicks on join group button.
3. The system add user to this group.

---

**Use case:** Respond join group request
Actor: User
Precondition: The user is logged in to the social network and has a group request.
Main scenario:
1. The user enters in to public group page info who has invited or in to his homepage.
2. The user clicks on accept or decline join group button.
3. The system add user to this group or deletes the invitation.

Use case: Leave a group
Actor: User
Precondition: The user is logged in to the social network and takes part from group.
Main scenario:
1. The user enters in to the group page info.
2. The user clicks on leave group button.
3. The system deletes the user of the current group.

3.1.3. Behaviour model

User register
Context: register(username: string, password: string)
Pre: Fields username and password are not empty.
The User with username does not already exist.
Post: Creates a user with the given username and password. An empty user profile is created too.

User login
Context: login(username: string, password: string)
Pre: Fields username and password are not empty.
The user with username exists.
Post: The system logs in the user.

User logout
Context: logout()
Pre: The user is logged in to the system.
Post: The system logs out the user.
**View user profile**

**Context:** view_profile(username: string): User

**Pre:** The user with *username* exists.

**Post:** The system returns user with profile information identified by *username*.

**Edit user profile**

**Context:** edit_profile(name: string, surname: string, email: string, gender: string, city: string, birthday: date, photo: file)

**Pre:** The *birthday* is a valid date.
The *email* is a valid email address.

**Post:** The user profile information is updated in the system.

**Change username**

**Context:** change_username(username: string, new_username: string, password: string)

**Pre:** Fields *username*, *new_username* and *password* are not empty.
A User with *username* exists.
A User with username *new_username* does not exist.

**Post:** If *password* matches, the user *username* is changed to the new username *new_username*.

**Change password**

**Context:** change_password(username: string, password: string, new_password: string)

**Pre:** Fields *username*, *password* and *new_password* are not empty.
A User with *username* exists.

**Post:** If *password* matches, the username for the user is changed to *new_username*.

**Add friend**

**Context:** add_friend(user_from: integer, user_to: integer)

**Pre:** user_from and user_to are not empty.
user_from and user_to are different.
Post: The system creates one instance of Friendship with ids of user user_from, user_to and status pending. User_from is the id of the user who sends the friend request and user_to who receives request.

Accept friend
Context: accept_friend(user_from: integer, user_to: integer)
Pre: user_form and user_to are not empty.
user_from and user_to are different.
Post: The system creates or update one instance of Friendship with ids of user user_from, user_to with status accepted, where user_from is the id of the user who accepts friend and user_to who sent request. Also, the system updates the other instance with status accepted, where user_from is the user_to of the created Friendship and vice versa, user_to is the user_from of the created Friendship.

Reject friend
Context: reject_friend(friendship_id: integer, user_from: integer, user_to: integer)
Pre: user_form and user_to are not empty.
user_from and user_to are different.
A Friendship identified by friendship_id.
Post: The system updates Friendship identified by friendship_id with status rejected. User_from is the id of the user who sent the friend request and user_to the id of user who rejects it.

Cancel friend request
Context: cancel_friend_request(friendship_id: integer, user_from: integer, user_to: integer)
Pre: user_form and user_to are not empty.
user_from and user_to are different.
A Friendship identified by friendship_id exists.
Post: The system delete Friendship identified by friendship_id.

Stop being friends
Context: delete_friendship(user_from: integer, user_to: integer)
Pre: user_form and user_to are not empty.
user_from and user_to are different.
A Friendship with user_from -> user_to exists.
A Friendship with user_to -> user_from exists.

Post: The system deletes the Friendships identified by user_from -> user_to and user_to -> user_from.

---

Create a group

Context: create_group(name: string, description: string, isPublic: boolean)
Pre: Fields name and isPublic are not empty.
Post: A group is created with name, description and privacy isPublic.

---

View group

Context: view_group(group_id: integer): Group
Pre: The group with group_id exists.
Post: The system returns group identified by group_id with group information and list of members.

---

Edit group information

Context: edit_group(group_id: integer, name: string, description: string, isPublic: boolean)
Pre: A Group identified by group_id exists.
Fields name and isPublic are not empty.
Post: The Group identified by group_id is updated with name, description and privacy isPublic.

---

Add users to a group

Context: send_group_requests(group_id: integer, users_ids: set(integer))
Pre: A Group identified by group_id exists.
Users_ids have at least one user id.
Post: For each user id of users_ids and group_id, the system creates one instance of GroupUser with regular role and status pending.

---

Assign admins to a group

Context: assign_administrators(group_id: integer, users_ids: set(integer))
Pre: A Group identified by group_id exists.
Users_ids have at least one user id.
All users identified by user id are member of group.
Post: For each user id of user_ids and group_id, the system updates GroupUser with admin role.

---

**Join group**

Context: \( \text{join\_group}(\text{group}\_id: \text{integer}, \text{user}\_id: \text{integer}) \)

Pre: A Group identified by group_id exists and it is a public group. User identified by user_id exists and is not member of group.

Post: The system creates GroupUser with user_id, group_id, regular role and status accepted.

---

**Respond join group request**

Context: \( \text{response\_join\_group}(\text{group}\_id: \text{integer}, \text{user}\_id: \text{integer}, \text{status}: \text{RequestStatus}) \)

Pre: A Group identified by group_id exists. User identified by user_id exists and has a group request.

Post: The system update GroupUser with user_id, group_id setting status to status, and role to regular.

---

**Leave a group**

Context: \( \text{leave\_group}(\text{group}\_id: \text{integer}, \text{user}\_id: \text{integer}) \)

Pre: A Group identified by group_id exists. Users_ids have at least one user id. User identified by user_id is member of group.

Post: For each user id of user_ids and group_id, the system updates GroupUser with admin role.
3.2. Iteration 2

3.2.1. Conceptual model

Constraints

- Each class must have a unique identifier (id).
- Inherited constraints from iteration 1.
- For one User and Event only must to exist an EventUser instance.
3.2.2. Use cases

Use case: Create event
Actor: User
Precondition: The user is logged in to the social network.

Main scenario:
1. The user enters in to user homepage and click on the create event button.
2. The user fills the event creation form with name, description, start date, end date, a checkbox to mark as all day event, the location helped by a map, the privacy and a selector to invite users from friend list or groups list.
3. The user clicks on the create event button.
4. The system saves event’s data, sends invitations to users and redirects to user homepage.

Extensions:

3a. Invalid data
   1. The system shows a dialog message with the information.
   2. Go to step 2.

3b. Start date is greater than end date
   1. The system shows a dialog message with the information.
   2. Go to step 2.
Use case: View event
Actor: User
Precondition: The user is logged in to the social network.
Main scenario:
1. The user enters in to user homepage.
2. The user clicks on event name from calendar or from incoming events list.
3. The system shows information about event such as name, description, start date, end date, situation, the location, privacy and lists of users: attending, maybe, awaiting reply and not attending, in addition all related data such as messages, polls and tasks.

Use case: Edit event
Actor: User event admin
Precondition: The user is logged in to the social network and is the administrator of the event.
Main scenario:
1. The user enters in to the event page and click on the edit event button.
2. The user fills the event creation form with name, description, start date, end date, a checkbox to mark as all day event, the location helped by a map, the privacy and a selector to invite users from friend list or groups list, in addition a selector to assign administrators.
3. The user clicks on the update event button.
4. The system updates event’s data, sends invitations to users and redirects to user homepage.

Extensions:
3a. Invalid data
   1. The system shows a dialog message with the information.
   2. Go to step 2.

3b. Start date is greater than end date
   1. The system shows a dialog message with the information.
   2. Go to step 2.
Use case: Delete event
Actor: User event admin
Precondition: The user is logged in to the social network and is the administrator of the event.
Main scenario:
1. The user enters in to the event page and click on the delete event button.
2. The system shows a confirm dialog to delete event.
3. The user click on confirm button.
4. The system deletes event’s data and all related data from event.

Use case: Response event invitation
Actor: User
Precondition: The user is logged in to the social network and has event invitation
Main scenario:
1. The user enters in to the event page or on invitations list from user homepage
2. The user clicks one of the possible buttons: attending, not attending and maybe attending.
3. The system updates the invitation with the new status.

3.2.3. Behaviour model

Create event

Context: create_event(name: string, information: string, isPublic: boolean, address: string, latitude: integer, longitude: integer, startDate: date, endDate: date, allDay: boolean, friends_id: set(integer), group_members_ids: set(integer))
Pre: StartDate and endDate are valid dates. endDate must be greater than startDate.
Post: The event is created in the system and it sends the invitation to the corresponding users.

Edit event

Context: edit_event(event_id: integer, name: string, information: string, isPublic: boolean, address: string, latitude: integer, longitude: integer, startDate: date, endDate: date, allDay: boolean, friends_id: set(integer), group_members_ids: set(integer))
Pre:  
*StartDate and endDate are valid dates.*
*endDate must be greater than StartDate.*
The event with event_id exists.

Post:  
The event with event_id is updated in the system and it sends the invitation to the corresponding users.

Delete event  
Context:  
delete_event(event_id: integer)

Pre:  
The event with event_id exists.

Post:  
The event with event_id is deleted from the system and all related data.

View event  
Context:  
view_event(event_id: integer)

Pre:  
The event with event_id exists.

Post:  
The system return the information of event with event_id with all related data.

Response event invitation  
Context:  
response_event_invitation(event_id: integer, user_id: integer, status: string)

Pre:  
The event with event_id exists.
The user with user_id exists.
The user with user_id has been invited to event with event_id

Post:  
The system updates the invitations to event with the status status.
3.3. Iteration 3

3.3.1. Conceptual model

**Constraints**

- Each class must have a unique identifier (id).
- Inherited constraints from iteration 1 and 2.
- One user is assigned to a task if user is on attending list of the event that belongs to the task.
- A user can only vote once in a poll.
- `/vote_count` is a calculated field and is calculated counting the number of PollVote instances.
3.3.2. Use cases

Use case: Post message
Actor: User
Precondition: The user is logged in to the social network.
Main scenario:
1. The user enters in to event page.
2. The user goes to wall section.
3. The user enters a message on the corresponding field.
4. The user presses the send button.
5. The system saves message and shows on wall section.

Extensions:
3a. Empty message
1. The system shows a dialog message with the information.
2. Go to step 3.

Use case: Delete message
Actor: User
Precondition: The user is logged in to the social network.
Main scenario:
1. The user enters in to event page.
2. The user goes to wall section.
3. The user clicks on delete button of a message who posted before or clicks any message if the user is the administrator of the event.
4. The system shows a confirm dialog.
5. The user clicks on accept button.
6. The system deletes message.

Use case: Add poll
Actor: User event admin
Precondition: The user is logged in to the social network and is the administrator of the event.
Main scenario:
1. The user enters in to event page.
2. The user goes to poll section.
3. The user fills fields with question, due date, allow multiple choices or not and add answers for this question.
4. The user presses the create poll button.
5. The system saves poll.

Extensions:
3a. Empty fields
   1. The system shows a dialog message with the information.
   2. Go to step 3.

Use case: Edit poll
Actor: User event admin
Precondition: The user is logged in to the social network and is the administrator of the event.
Main scenario:
1. The user enters in to event page.
2. The user goes to polls section.
3. The user clicks on edit poll button.
4. The user update fields question, answers, due date, allows multiple choices and, in addition adds new answers.
5. The user presses the edit poll button.
6. The system updates poll.

Extensions:

4a. Empty fields
   1. The system shows a dialog message with the information.
   2. Go to step 3.

---

Use case: Vote on a poll
Actor: User
Precondition: The user is logged in to the social network.
Main scenario:

1. The user enters in to event page.
2. The user goes to polls section.
3. The user selects one or multiple options for one question.
4. The user clicks on vote button.
5. The system updates poll results.

---

Use case: Delete poll
Actor: User
Precondition: The user is logged in to the social network and is the administrator of the event.
Main scenario:

1. The user enters in to event page.
2. The user goes to polls section.
3. The user clicks on delete button of a poll.
4. The system shows a confirm dialog
5. The user clicks on accept button.
6. The system deletes message.

---

Use case: Add task
Actor: User event admin
Precondition: The user is logged in to the social network and is the administrator of the event.
Main scenario:
1. The user enters in to event page.
2. The user goes to tasks section.
3. The user fills task information with title, description, the type of task (before event or during event) and assigns users from attending list.
4. The user presses the assign task button.
5. The system saves task with status opened and informs assigned users on their homepage.

Extensions:
3a. Empty fields
   1. The system shows a dialog message with the information.
   2. Go to step 3.

Use case: Edit task
Actor: User event admin
Precondition: The user is logged in to the social network and is the administrator of the event.
Main scenario:
1. The user enters in to event page.
2. The user goes to tasks section and click on edit button of one task.
3. The user fills task information with title, description, the type of task (before event or during event) and assigns users from attending list.
4. The user presses the assign task button.
5. The system updates tasks information and informs assigned users on their homepage.

Extensions:
3a. Empty fields
   1. The system shows a dialog message with the information.
   2. Go to step 3.

Use case: Delete task
Actor: User event admin
Precondition: The user is logged in to the social network and is the administrator of the event.
Main scenario:
1. The user enters in to event page.
2. The user goes to tasks section and click on delete task button of one task.
3. The system shows a confirm dialog.
4. The user presses accept button.
5. The system deletes the task.

---

**Use case:** Change task status  
**Actor:** User  
**Precondition:** The user is logged in to the social network and is the administrator of the event.  
**Main scenario:**  
1. The user enters in to event page.  
2. The user goes to tasks  
3. The user change status (close or open task button) of one task who is assigned or any task if the user is the administrator of the event  
4. The system change status task.

---

**Use case:** Find event  
**Actor:** User  
**Precondition:** The user is logged in to the social network  
**Main scenario:**  
1. The user enters into find event view.  
2. The user fills one field with keywords to find events and filters by date.  
3. The system shows results on a list with name of event, description, date and location.  
**Extensions:**  
3a. No results  
   1. The system shows a message with no results with some tips to search accurately.  
   2. Go to step 2.
3.3.3. Behaviour model

Post message

Context: post_message(event_id: integer, user_id: integer, message: string)

Pre: Event with event_id exists.
User with user_id exists.
The parameter message is not empty.
If event identified by event_id.

Post: The message is saved for event event_id and posted by user with user_id.

Delete message

Context: delete_message(message_id: integer, event_id: integer, user_id: integer)

Pre: Message with message_id exists.
Event with event_id exists.
User with user_id exists.
User with user_id is the owner of message identified by message_id or is the administrator of the event identified by event_id.

Post: The message is deleted from system.

Add poll

Context: create_poll(question: string, due_date: date, options: set(string), multiple_choices: boolean, event_id: integer, user_id: integer)

Pre: Event with event_id exists.
User with user_id exists and is administrator of event with event_id.
Field question is not empty.
Field due_date is a valid date.
There are at least two options.

Post: The poll and their options for event with event_id and created by user with user_id are saved on the system.

Edit poll

Context: edit_poll(poll_id: integer, question: string, due_date: date, options: set(string), multiple_choices: boolean, event_id: integer, user_id: integer)

Pre: Poll with poll_id exists and is a poll of event identified by event_id.
Event with event_id exists.
User with user_id exists and is administrator of event with event_id.
Field question is not empty.
Field `due_date` is a valid date.
There are at least two options.

Post: The poll for event with `event_id` and modified by user with `user_id` is updated on the system.

---

**Vote on a poll**

**Context:**

```python
vote_poll(poll_id: integer, poll_option_ids: set(integer), event_id: integer, user_id: integer)
```

**Pre:**

- Poll with `poll_id` exists and is a poll of event identified by `event_id`.
- Event with `event_id` exists.
- User with `user_id` exists.
- There is at least one id on `poll_option_ids`.

**Post:**

The system saves for each poll option id on `poll_options_ids` a poll vote with `poll_id` and `user_id`.

---

**Delete poll**

**Context:**

```python
delete_poll(poll_id: integer, event_id: integer, user_id: integer)
```

**Pre:**

- Poll with `poll_id` exists and is a poll of event identified by `event_id`.
- Event with `event_id` exists.
- User with `user_id` exists and is administrator of event with `event_id`.

**Post:**

The system deletes the poll with `poll_id` and all related data.

---

**Add task**

**Context:**

```python
create_task(title: string, description: string, due_date: date, assigned_users: set(integer), isDuringEvent: boolean, event_id: integer, user_id: integer)
```

**Pre:**

- Event with `event_id` exists.
- User with `user_id` exists and is administrator of event with `event_id`.
- Fields `question`, `description` and `due_date` are not empty.
- Field `due_date` is a valid date.
- There is at least one assigned user.

**Post:**

The task and users assigned for event with `event_id` and created by user with `user_id` are saved on the system; task is created with status opened by default.
Edit task

Context:
edit_task(task_id: integer title: string, description: string, due_date: date, assigned_users: set(integer), isDuringEvent: boolean, event_id: integer, user_id: integer)

Pre: Task with task_id exists and is a task of event identified by event_id. Event with event_id exists. User with user_id exists and is administrator of event by event_id. Fields question, description and due_date are not empty. Field due_date is a valid date. There is at least one assigned user.

Post: The task and users assigned for event with event_id and modified by user with user_id are updated on the system.

Delete task

Context:
delete_task(task_id: integer, event_id: integer, user_id: integer)

Pre: Task with task_id exists and is a task of event identified with event_id. Event with event_id exists. User with user_id exists and is administrator of event with event_id.

Post: The system deletes task identified by task_id and all related data.

Change task status

Context:
change_task_status(task_id: integer, event_id: integer, user_id: integer, status: boolean)

Pre: Task with task_id exists and is a task of event identified with event_id. Event with event_id exists. User with user_id exists and is administrator of event with event_id or is an assigned user of task identified by task_id.

Post: The system updates task identified by task_id with status to status.

Find events

Context:
search(text: string, from_date: date, to_date: date)

Pre: Field text is not empty. Fields from_date and to_date are valid dates.

Post: The system returns events searching on name, information or location using keywords on text and filtering by date.
3.4. Complete conceptual model
**Constraints**

- Each class must have a unique identifier (id).
- User *username* must be unique.
- User *social_provider* and *social_id* must be unique.
- User *user_from* and *user_to* must be different users.
- For one User and a Group only must to exist a GroupUser instance.
- The friend relationship must be bidirectional, two users are considered friends if one are friend of each other and vice versa.
- For one User and Event only must to exist an EventUser instance.
- One user is assigned to a task if user is on attending list of the event that belongs to the task.
- A user can only vote once in a poll.
- `/vote_count` is a calculated field and is calculated counting the number of PollVote instances.
4. Design

4.1. Technology

4.1.1. Frontend

HTML

HTML is the acronym of HyperText Markup Language. It is a widely used web standard and defines, through tags, a basic structure for defining content of a website, such as text, images, and so on. It is a standard controlled by the W3C, an organization dedicated to the standardization of almost all technologies related to the web.

CSS

Cascading Style Sheets are used for describing the look and formatting of a HTML document such as layout, colours and fonts. As HTML, it is also standardized by W3C organization.

Bootstrap

Bootstrap is a framework[^10] for quickly creating websites from the client side. It includes tools to help web developer, such as HTML and CSS templates for typography, forms, buttons, navigation and other interface components. Also it includes JavaScript extensions. Moreover, the grid system layout included on is responsive, to make use of different resolutions and types of devices such as smartphones, tablets and PCs with low and high resolution.

jQuery

jQuery is free, open source software, licensed under the MIT license[^11] JavaScript library. Nowadays it is the most popular JavaScript library; additionally it includes a large number of plugins. jQuery's syntax is designed to facilitate navigation on a document, select DOM elements, create animations, and manage events.

4.1.2. Backend

PHP

PHP (PHP Hypertext Preprocessor) is an interpreted programming language designed to create dynamic web pages from the server side, and is an object-oriented language. It is...

[^10]: It is a conceptual and technological structure defined, usually with specific devices or software modules, which can be the basis for the organization and software development support.

[^11]: Free software license originating at the Massachusetts Institute of Technology.
especially suited for web development because PHP code can be embedded into HTML. It is free software released under the PHP License.

CakePHP

As we can see on their web “CakePHP makes building web applications simpler, faster and require less code”. Based on PHP, CakePHP is a framework that facilitates interaction with the database because it uses ORM technique (Object Relational Mapping) and also it follows the MVC pattern (Model View Controller). On the architecture chapter this patterns will be precisely detailed.

CakePHP is licensed under the MIT license. It is a permissive free software license, which permits reuse within proprietary software. Also it includes security built-in tools such as input validation, CSRF protection, Form tampering protection, SQL injection prevention, and XSS prevention.

This framework has a lot of documentation available on the web; moreover it has a large community of developers.

Finally, for the project, we are going to use some CakePHP plugins such as ExtAuth and cakephp-upload, one it is a plugin for external authentication and the other one it is to manage images uploads.

MySQL

MySQL, the most widely used open-source relational database management system. It is a multithreaded, multi-user management database developed by MySQL AB available under the terms of the GNU General Public License. MySQL can run on UNIX, Windows and Mac OS. To administrate MySQL we are going to use phpMyAdmin, a free open-source web browser tool.

Apache server

Apache is an HTTP open source web server available on Unix, Microsoft Windows and Macintosh systems. Apache is perfectly suited to our needs because it includes required modules for our implementation, especially PHP and MySQL modules. Released under the Apache License, Apache is open-source software.

---

12 Acronym of Hypertext Transfer Protocol is designed to enable communications between clients and servers. It works as a request-response protocol between a client and server.
4.2. Architecture

4.2.1. Model View Controller
CakePHP is based on MVC, a widely used pattern on software engineering. The main purpose of this pattern is to separate user interface and the application logic. Therefore, applying this pattern, we ensure that making changes on one layer does not affect the other layer.

![Diagram of MVC in CakePHP](http://book.cakephp.org)

Firstly, the client send a petition to the system, secondly the dispatcher analyses this petition and routes to the corresponding action on the controller. Afterwards, the controller executes and processes the corresponding logic for this petition communicating with models if necessary. Once the models responds, the controller prepares the output data to send it to the corresponding view. Finally, the view shows this output data to the client.

Model

A model represents the application data, the information. Models should be responsible to manage this data allowing add, remove, update and change operations.

CakePHP uses an Object Relational Mapping to define the mapping with database tables. ORM consists in relate models of the application with the equivalent tables in the database: the model itself represents a table, each instance of the model a register, and attributes represent table fields.
Following the “model and database conventions” of cakePHP, automatically we can map without any configuration and just naming the tables and table fields on the database correctly.

Some conventions to follow are:

- Model classnames are singular and CamelCased. For example: User, AdminUser.
- Tables name corresponding to these models should be plural and underscored. For example: users and tasks_users.
- Foreign keys are recognized by default as the singular name of the related table followed by _id.
- Join tables between models should be named after the model tables they will join in alphabetical order. For example tasks_users rather than users_tasks.
- All tables with which CakePHP models interact require a singular primary key to uniquely identify each row

On figure below we can see an example to define models and their associations.

```php
class User extends AppModel {
    public $hasOne = 'Profile';
    public $hasMany = array('Recipe' => array('conditions' => array('Recipe.approved' => '1'), 'order' => 'Recipe.created DESC'));
    public $hasManyToMany = array('MemberOf' => array('className' => 'Group'), );
}

class Profile extends AppModel {
    public $belongsTo = 'User';
}
```

Figure 12. Example of associations and linking models on CakePHP

CakePHP comes with default functions to handle data from database automatically. To save or update one model or several models just calling to save method and structuring the data properly, CakePHP will be capable to save, update or delete this model and all related models if necessary. Similarly, CakePHP has methods to obtain data, with 'find' or 'findByFieldname' methods for example. On the next table we can see some retrieving data examples.
<table>
<thead>
<tr>
<th><strong>findBy Example</strong></th>
<th><strong>Corresponding SQL Fragment</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>$this-&gt;Product-&gt;findByOrderStatus('3');</td>
<td>Product.order_status = 3</td>
</tr>
<tr>
<td>$this-&gt;User-&gt;findByUsernameAndPassword('jhon', '123');</td>
<td>User.username = 'jhon' AND User.password = '123';</td>
</tr>
<tr>
<td>$this-&gt;User-&gt;findByEmailOrUsername('jhon', 'jhon');</td>
<td>User.email = 'jhon' OR User.username = 'jhon';</td>
</tr>
<tr>
<td>$this-&gt;Cake-&gt;findById(7);</td>
<td>Cake.id = 7</td>
</tr>
<tr>
<td>$this-&gt;User-&gt;findByLastName('psychic', array(), array('User.user_name =&gt; 'asc'));</td>
<td>User.last_name = 'psychic' ORDER BY User.user_name ASC</td>
</tr>
</tbody>
</table>

Table 7. Example methods find CakePHP

**View**

This layer is responsible for layout and display information to the user. Also this layer is responsible to send user actions to the controller.

The view layer in CakePHP can be made up of a number of different parts: layouts, elements and helpers. A layout contains presentation code that wraps around a view, element contains portions of code to reuse on layouts and finally helpers are encapsulated view logic.

**Controller**

The controller layer contains the business application logic. It provides model data to the view and processes user actions from view.

Controllers on CakePHP can include methods which referred to actions. Commonly, controllers are used to manage the logic for a single model.

Following the CakePHP names convention, controller class names are plural, CamelCased, and end in Controller. For example a controller for User model should be named as UsersController. Then automatically, for example, a request for http://www.example.com/users/index maps to a call on the index() method of the UsersController.
4.3. Iteration 1

4.3.1. Domain model

[Diagram of domain model including classes like User, Profile, GroupUser, Friendship, and enums for Gender, Role, RequestStatus, SocialProvider, showing relationships and attributes such as username, password, social_id, social_provider, social_access_token, status, role, etc.]
4.3.2. Data model

Constraints

- **Users**
  - Field `username` is unique.
  - Fields `(social_id, social_provider)` are unique.

- **Friendships**
  - Fields `(user_from, user_to)` are unique.

- **GroupsUsers**
  - Fields `(user_id, group_id)` are unique.
4.3.4. Sequence diagrams

User register
User login

In case of login with external service, a callback method will be the responsible of collect user info to login or register user if necessary.

User logout
View user profile

```
alt
[username]

1: profile(username)

2: findByName(username)
user

3: user(id)

loggedUserId

4: findById(loggedUserId)
user

if (user) throws NotFoundException("user not found");

5: set("user", user)
```
Edit user profile

1. update_profile(data)

2. user(id)
   - user.id
   - loggedUser.id

3. findByid(loggedUser.id)
   - user
   - data[Profile][id] = user[User][profile.id]

4. save(data)
   - profile.updated

5. selfFlash(message, 'success')
   - message = 'The user profile has been updated'

6. selfFlash(message, 'error')
   - message = 'The user profile could not be updated. Please try again.'
Change username

1: change_username(data)
   - data
     - username
     - password

2: user(id)
   - logged_userid

3: check_password(logged_userid, password)
   - correct_password

4: exists_username(username)
   - exists_username

all: [exists_username && correct_password]
   - data[User.id] = logged_userid

5: save(data)
   - username_updated

all: [username_updated]
   - message = 'The user profile has been updated'
   - type = 'success'

all: [username_updated]
   - message = 'The user profile could not be updated. Please, try again.'
   - type = 'error'

[exists_username]
   - message = 'The username entered already exists. Please, choose other username.'
   - type = 'error'

[!correct_password]
   - message = 'Your current password is incorrect. Please, try again.'
   - type = 'error'

6: setFlash(message, 'error')
Change password

*In this case the sequence diagram is similar to the diagram for "change username".*

Add friend
Accept friend

```
1. accept_friend(data)
   data
   user_to

2. user['id']
   logged_user_id

   data[Friendship][User_from] = logged_user_id
   data[Friendship][status] = 'accepted'

3. findByUserFromAndUserTo(user_from, user_to)
   friendship

alt

1. [friendship]
   <<create>>
   4. create()
   f: Friendship

   5. save(data)
      created_or_updated_frienship

   data[Friendship]['id'] = friendship[Friendship]['id']

   6. save(data)
      created_or_updated_frienship

alt

1. [created_or_updated_frienship]
   7. findByUserFromAndUserTo(user_to, user_from)
      friendship

   friendship[Friendship][status] = 'accepted'

   8. save(friendship)

message: 'The received friend request has been accepted successfully'
type: 'success'

message: 'The received friend request could not be accepted. Please, try again.'
type: 'error'

9. setFlash(message, type)
```
Reject friend

1. reject_friend(data)

2. user(id)
   loggedUserId

3. findByUserFromAndUserTo(user_from, loggedUserId)
   friendship
   friendship[Friendship][status] = 'rejected'

4. save(friendship)
   saved

alt [saved]

[1 saved]

message = 'The received friend request has been rejected successfully'
   type = 'success'

[1 saved]

message = 'The received friend request could not be rejected. Please, try again.'
   type = 'error'

5. setFlash(message, type)
Cancel friend request

This sequence diagram is similar to the diagram for "cancel friend request". In this case, the operation deletes both instances of friendship.

Stop being friends
Create a group

1. create(data)

2. user(id)
   - loggedUserId
   - data[Group][created_by] = loggedUserId
   - data[Group][modified_by] = loggedUserId

3. create()

4. save(data)

5. create()

6. save(groupUser)

7. setFlash(message, type)

message = "The group has been created successfully. You are the administrator."
type = 'success'

message = "The group could not be saved. Please, try again."
type = 'error'
View group

1. view(groupid)
2. user(Id)
  .loggedUserId
3. findById(groupid)
   .group
   .existsGroupInvitation(loggedUserId, group["Group"],"id")
   .userInvited

   if (userInvited && !group["Group"].isPublic())
       throws MethodNotAllowedException("You are not allowed to view this group.")

5. set('group', group)
Edit group information

1. edit(group_id)
2. user(id)
   - loggedUserId
3. findGroupById(group_id)
   - group
4. isAdmin(loggedUserId, group[groupId])
5. save(data)
6. setFlash(message, type)

message: 'The group information has been updated successfully'
type: 'success'

message: 'The group information could not be updated. Please, try again.'
type: 'error'
Add users to a group

Assign admins to a group

This sequence diagram is similar to the diagram for "add users to groups". In this case, the operation updates GroupUser instances with status admin.
Join group

Respond join group request

*This sequence diagram is similar to the diagram for "add users to groups". In this case, the operation updates GroupUser instance with status accepted.*
Leave a group

1. leave_group(group_id)

2. user("id")
   loggedUser.id

3. findByUsernameAndGroupId(loggedUser.id, group_id)
   groupUser

4. delete(groupUser)
   deleted

message = "You have left the group successfully."
  type = "success"

message = "There was a problem leaving the group. Please, try again."
  type = "error"

5. setFlash(message, type)
4.4. Iteration 2

4.4.1. Domain model
4.4.2. Data model

**Constraints**

- **Users**
  - Field username is unique.
  - Fields (social_id, social_provider) are unique.

- **Friendships**
  - Fields (user_from, user_to) are unique.

- **GroupsUsers**
  - Fields (user_id, group_id) are unique.

- **EventsUsers**
  - Fields (user_id, event_id) are unique.
4.4.4. Sequence diagrams

Create event

This sequence diagram is similar to the diagram for "create event". In this case, if user logged id admin, the operation updates Event and EventUser instances. In addition, a userIdsAdmin array is included to update corresponding instances with role admin.

Edit event
Delete event

1. delete(eventId)

2. user(userId)
   loggedUserId

3. isAdmin(loggedUserId, eventId)

   if (!isAdmin) throws MethodNotAllowedException("You are not allowed to delete this event")

4. findById(eventId)

5. delete(event)

6. setFlash(messages, type)

   message = "You have delete the event successfully!"
   type = 'success'

   message = "There was a problem deleting event. Please, try again!"
   type = 'error'
View event

1. view(eventId)

2. user['id']
   loggedUserId

3. findById(eventId)

   event

   if (event) throws NotFoundException('Event not found')

   4. existsEventInvitation(loggedUserId, event[Event]['id'])

   invited

   if (invited && event[Event]['isPublic']) throws MethodNotAllowedException('You are not allowed to view this event')

5. set('event', event)
Response event invitation

EventController

1. view(eventId, status)

2. user(id)

3. findById(eventId)

4. existsEventInvitation(loggedUserId, eventId)

5. invited

6. [limited && event[Event][isPublic]]

7. create

8. save(eventUser)

9. setFlashMessage(message, type)

Auth

Event

EventUser

Session

web development of a Social Network for Events

Bachelor's Thesis

Barcelona School of Informatics
4.5. Iteration 3

4.5.1. Domain model
Constraints

- **Users**
  - Field `username` is unique.
  - Fields (social_id, social_provider) are unique.

- **Friendships**
  - Fields (user_from, user_to) are unique.

- **GroupsUsers**
  - Fields (user_id, group_id) are unique.

- **EventsUsers**
  - Fields (user_id, event_id) are unique.
4.5.4. Sequence diagrams

Post message

![Sequence diagram for post message](image-url)
Delete message

1. delete_message({messageId}, eventId)

2. user(id)
   
   loggedUser

3. findMessage({messageId}, {loggedUser}, {eventId})
   
   message

4. delete(message)
   
   deletedMessage

af

[deletedMessage]

message = 'You deleted the message successfully.'

{type} = 'success'

if deletedMessage

message = 'There was a problem deleting the message. Please, try again.'

{type} = 'error'

5. setFlash(message, type)
Add poll

This sequence diagram is similar to the diagram for "add poll". In this case, the system updates information and adds new options if necessary.

Edit poll

This sequence diagram is similar to the diagram for "add poll". In this case, the system updates information and adds new options if necessary.
Vote on a poll

Delete poll

This sequence diagram is similar to the diagram for "delete message". In this case, the system deletes poll, poll options and poll votes.
Add task

This sequence diagram is similar to the diagram for "add task". In this case, the system updates information and reassigns users to task if necessary.

Edit task

This sequence diagram is similar to the diagram for "add task". In this case, the system updates information and reassigns users to task if necessary.

Delete task

This sequence diagram is similar to the diagram for "delete message". In this case, the system deletes task and all related data.
Change task status

Find events

We will use a cakephp plugin to find events. The sequence diagram in this case would simply pass the search criteria and set the fields where we want to filter. Finally, system returns results to the corresponding view.
4.6. Navigational map

To simplify diagram:

- All views have a navigation bar where user can navigate to user homepage, logout, view and edit personal profile, event search and create group or event.

- Logout redirects to public homepage.

- Users can navigate through events or groups to user profiles.
5. Implementation

Once specified and designed the project, the next step is implementation. In the three iterations we have followed the next implementation process: creation of the corresponding tables, model definition, controllers and finally user interface layout.

5.1. Database

As CakePHP is conceived and the reason that is a good framework for agile development, the first step to start implementing any functionality is the design of the database. Following the CakePHP name conventions described on design phase just we must to create the corresponding tables and their relations defined on previous section.

```
CREATE TABLE events(
    id INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(50) NOT NULL,
    information VARCHAR(500),
    isPublic TINYINT(1) NOT NULL DEFAULT 0,
    address VARCHAR(100),
    startDate DATETIME,
    endDate DATETIME,
    allDay TINYINT(1) NOT NULL DEFAULT 0,
    latitude DECIMAL(10, 8),
    longitude DECIMAL(11, 8),
    created DATETIME DEFAULT NULL,
    modified DATETIME DEFAULT NULL,
    created_by INT NOT NULL, FOREIGN KEY (created_by) REFERENCES users(id),
    modified_by INT NOT NULL, FOREIGN KEY (modified_by) REFERENCES users(id)
);
```

Figure 13. Create table example

In figure 13, we can see an example of create table with MySQL syntax. The fields created and modified does not defined initially on specification. These fields are managed automatically by CakePHP, and they are updated depending on insert or update operation.

5.2. Models

The next step is the definition of the models. We must create for each model a file with model name with PHP extension on the model folder. CakePHP without extra configuration, it maps the model with the corresponding table. Within the model we can define relationships with other models and create the necessary method.
In figures 14 and 15, we have most typical examples of association to other models. For example, through these associations when we retrieve one event, we can obtain related data such as attending lists, admins list, the user who modified or created event, polls, tasks and so on.

5.3. Controllers

Once created models and their associations, the next step is to build controllers and their actions methods. We must to create a file with the name of the model in plural followed by ‘Controller’ also with php extension as we done it with models.
Now we can define methods to manage requests (GET or POST)\textsuperscript{13} from view layer. By default all public methods in a controller are an action, and accessible from a URL and this methods handle processing and generating a response to view layer.

The request data is defined on the variable $this->request->data, and the parameters specified on the method parameters. For example a view form which make a POST on the next URL: \url{http://myweb.com/events/edit/8}, CakePHP routes to events controller on method edit with event id as parameter and also with the request data. Once again, cakephp makes work easier for the developer.

5.4. User interface

And finally, the implementation of the layer view. Each public method within the controller has automatically associated a view with the same name as method and a ctp extension file. These files must contain HTML code with embedded PHP to process the response of the controller and show it to user. It has to create in the ‘View’ folder and within the folder with the name of the associated model controller.

For the design of the user interface we have followed mostly used web usability criteria, with particular emphasis on:

- Easy of learning and memorability.
- Clean, simple and clear interface.
- Apply different sizes and colours to prioritize links or buttons.
- Facilitate data entry. For example, using lists, drop-down menus, autosuggest inputs, and so on.
- System feedback to user.
- Use of de facto standards web.

To improve usability we also have followed the same pattern design for all views. Thereby we have been very consistent and uniform designing them. For example: colours pattern, styles, sizes, elements disposition and so on. Below, we are going to present some screenshots of website design.

\textsuperscript{13} HTTP requests methods, commonly used methods for a request-response between a client and server. GET requests from a specified resource and POST submits data to be processed by specified resource.
Public homepage

User homepage
User profile

Luke profile information

- Name: Luke Skywalker
- Birthday: 12/04/1977
- Email: luke@starwars.com

Friends

- Han Solo
- Leia Organa
- Chewbacca
- C3PO
- Master Yoda
- Show More...

Groups

- Empty group list
Groups

Group of friends
- Name: Jedi
- Created: 22/01/2014 15:20 by Luke Skywalker
- Modified: 22/01/2014 15:23 by Luke Skywalker
- Privacy: Private

3 Members
- Luke Skywalker
- Master Yoda
- Obi-Wan K...

1 Admins
- Luke Skywalker
Edit event
View event

Babercue in Tatooine

Information | Wait | Posts | Tasks
---|---|---|---

Description

When
09/02/2014

Where
Tatooine

Privacy
Private event

Created
22/01/2014 16:24 by Luke Skywalker

Modified

Location

3 Attending

Luke Skywalker
Han Solo
Leia Organa

1 Maybe

Master Yoda

2 Awaiting reply

Obi-Wan Kenobi
C3PO

1 Not attending

R2D2

1 Admins

Luke Skywalker
Babercue in Tatooine

Messages

- Rwwoooaarrrr
  Posted by chewbacca - 22/01/2014 16:39

- But who's going to fly it, kid! You?
  Posted by Han Solo - 22/01/2014 16:36

- Ten thousand? We could almost buy our own ship for that!
  Posted by Luke Skywalker - 22/01/2014 16:36

- Well, that's the trick, isn't it? And it's going to cost you something extra. Ten thousand in advance.
  Posted by Han Solo - 22/01/2014 16:36
Web Development of a Social Network for Events
Babercue in Tatooine

Tasks to be done before event
- Repare x-wing fighter
  The canyon wall rushes by in the background, making this delicate task seem even more precarious.
  Due date: 09/02/2014 17:00
  Assigned to: Luke Skywalker

- Clean Millennium Falcon

Tasks to be done during event
- Cook with lightsaber
Find events

Responsive homepage design
Featured UI plugins for usability

Autocomplete.

Datetime picker.

Map location drag and drop.
Other components

System feedback.

You have successfully registered. Welcome to Inwigo

The username entered already exists. Please, choose other username.

Invalid username or password, please try again

Dropdown options.
5.5. Source package
5.6. Develop environment

To develop the project we have used XAMPP. It is a free and open source cross-platform web server solution stack package. The name comes from the acronym for X (any operating systems), Apache, MySQL, PHP and Perl. XAMPP allows web developers to test and develop their work on their own computers without any access to Internet. It is really simple to install and use just we should download, extract and start using.

![Develop environment schema](image)
5.7. Deployment schema

The deployment scheme it should have the project to be accessible on the web would be next.

Develop environment is for local working to individual developers or teams. The integration environment is oriented to simulate the production environment and test changes done before in develop environment. Once tested and validated these changes, the next step is to deploy on production environment where users will be able to access.
6. Testing

6.1. Functional testing

Following the agile methodology, the testing phase has been constant throughout the project development. Each newly developed functionality has been tested, revised and modified if necessary. Thus, from the beginning we were able to build the website ensuring that the software operates correctly.

At the end of every iteration and just as planned, the corresponding part of the developed system has been tested. Each use case and their operation contracts included on the iteration were thoroughly tested to ensure the compliance of operation contracts. During the test phase, there have appeared some new requirements and contracts. These changes have been updated on the documentation properly.

The most common errors have been the following:

- Incorrect data structure on post methods to save or update automatically data of one model and specially related data for this model.
- Control the correct view layout when we need to show a lot of information.
- Some minor difficulties on integration of plugins for cakePHP and jQuery.

Finally, once an iteration has finished and at the iteration’s follow-up meeting with the supervisor, we tested again all use cases during the demonstration of the functionalities. Moreover, supervisor gave some feedback to improve usability and possible new features.

6.2. Performance testing

This project has been created and tested on local development environment. Consequently, the performance testing has not been done because we need to prove it on production environment and check how the website responds to a simulation of huge demand with the project properly deployed on a server.
7. Conclusions

7.1. Project plan changes

At the time of initial project planning there was not a specific delivery date, and we decided to calculate an approximate delivery date for early January. After summer holidays, final dates were published: December and February. We decided to choose the second one. We decided February because compared to the initial plan; the new plan has given us almost an additional month to complete the project. Therefore, the completion date for the project is on January 30th, 2014 and the presentation is on February 6th, 2014.

\[\text{Figure 18. Final Gantt diagram}\]

This replanning did not mean a change in the time spent on each phase. It helped to distribute the initial calculated hours decreasing the dedicated hours per week on the project. Hence, the project budget has not been modified.

7.2. Conclusions

Most of the objectives and requirements defined initially in the project have been reached, summarized:

- Users can register into the system to add friends and create groups of friends.
• Registered users can create and manage events with name, description, date, location with the assistance of a map and they can control the attendance lists.

• They can assign tasks to other users, create polls to ask other users and post messages to communicate between users.

• Users are able to find events.

• We have added new features, especially on frontend part such as calendar for events, maps to locate events; visual date time picker and auto fill inputs to enter users easily.

In addition to the specific objectives of the project we had also specified technical competences. Below, we are going to discuss their achievement for each competence.

- **To define and manage the requirements of a software system.** From the beginning, we have defined a number of functional and non-functional requirements. Attainable requirements which we have been working and refining throughout the project, with special emphasis on the functional requirements.

- **To identify, evaluate and manage potential risks related to software building which could arise.** The project is economically quantified, measured in time and managed with resources available. We have built a plan based on agile methodology which has permitted us to be flexible avoiding potential risks.

- **To specify, design, implement and evaluate databases.** As we have seen throughout this thesis, one of the keys to developing a successful project based on CakePHP is the specification, designing and developing of a database. In the course of the entire project CakePHP conventions has been taken into account and have been followed to construct the database.

- **To develop, maintain and evaluate complex and/or critical software systems and services.** We have successfully developed and tested a complex project fulfilling the main objectives, defining its architecture and functionalities.

Personally, the construction of this project from the beginning has been a challenge. It has not been the first project in which I have participated, but it has been the first one that I had to define and build from scratch.

In first place, choosing an agile methodology for the project has been a good decision. Dividing the project into three parts was positive because it helped me appreciate the
progress of the project more clearly and other positive aspects of agile methodologies such as flexibility and adaptability. Furthermore, the decision of creating the social network from scratch was a good decision too because finally, we have built the social network as we wanted and without dependence of other software. Moreover, we have learned the basic principles of building a social network.

To sum up, we have built a functional and useful web based social network oriented to events. But this is just the beginning, on the next section we are going to give some keys to the next step: the release of the social network.

7.3. Future work

This section is aimed at giving some suggestions for future improvements and to make the project fully operational. Some improvements such as:

- Expanding social network functionalities with chat, friend recommendation, password recovering, email notifications and so on.

- Groups with a wall, group photo and group photo albums. Moreover, associating events to groups.

- Just as with groups, for events we should extend with event photo and event photos albums and create repeating event.

- Creating dependency between tasks to manage a workflow.

- Associating poll results to events for example creating a poll to ask about the location of an event and automatically update the event location with the result.

Besides the technical improvements commented before, we also need to create an online marketing plan including a good branding name to promote the social network over the internet. This online marketing plan should include a redesign of the website to make it more attractive to the user with two different versions, one for desktop and another for mobile devices.
8. References


9. Appendix – user manual

Register and login

Registration and login can be made from the homepage.

Regular registration

To sign up for the social network simply fill the following mandatory fields:

- Username.
- Password, with password confirmation.

If the username already or password does not match, the system shows a message informing the corresponding error.
Regular login

To log in the system fill the next mandatory fields:

- Username.
- Password.

If the username and password does not match, the system shows a message

External login

The system gives the option to log in easily from other social networks. First time you log in through an external social network, you will be asked for permission to login the system. Once accepted the permissions you will be capable of login just clicking the “Log in with...” button.
Logout

At the top right of the navigation bar, the system displays different options, one of them is the logout button, selecting this option you will be logged out from the system.

User profile

Edit profile

You can access on edit profile view through your profile or from the navigation bar. You can edit your profile information, change username and change your password.

To edit profile information you must fill the following fields:

- Name.
- Surname.
- Email (valid email address).
- Gender.
- City.
- Birthday (valid birthday).
- A profile photo (file size < 2MB).

Also, you can change your username to other non-existing username and change your current password.
View profile

You can access to your profile from navigation bar. Moreover you can access to other users profile throughout direct url or navigating from user friends list, events, groups and so on.

Friends

Making friends

You can add friends just visiting the corresponding user profile and clicking the “Add friend” button. The system will send a friend request to user homepage. You can cancel the friend request from your homepage or from user profile.

If you are the user who receives the friend request you can accept or reject friendship from your user homepage or from user profile, just clicking the corresponding button.
Friends list
You can see your friends list and friends list entering other user profile.

Groups
Creating and editing group
To edit profile information you must fill the following fields:

- Name (mandatory).
- Description.
- Privacity (public or private).

Moreover you can invite your friends to this group typing or selecting from input. The user will receive the invitation on his user homepage. Similarly, you can assign member users to manage this group.

**View group.**

You can access to a group view if the group is public. In case of a private group you can access if you have an invitation or you are member of this group. In your homepage there are listed the groups which you are member.

If you are the administrator of a group you can delete it from the group view and clicking on the delete button. Also you can leave group.
Joining group.

You can join to a public group (from user homepage or group view) without invitation or you can join to a group (public or private) with invitation sent by the administrator.

Events

Creating and editing event

To edit or create event you must complete the following information:

- Name.
- Description.
- Start date, end date or all day event.
- Location.
- Privacy.
- Users to send invitations.

The event invitation request will be sent to the corresponding user homepage. On edit mode you can assign administrators.
View event

You can access to an event page if event is public or on the other hand you have an invitation. The page contains four tabs one for event information, wall, polls and tasks.
To post a message just type the text in the input and click on the send button. If you want to delete the message posted, click on ‘x’.

In the polls tab, you can give your vote once for each poll. If you are an administrator you can add new polls, modify it and delete.
To edit or create a poll, you must complete the following information:

- Question.
- Options (at least two options).
- Due date (valid date).
- Type (multiple choice or unique).

You can assign tasks to users; on tasks tabs, there are shown the tasks list. Also, you can create, edit, and delete tasks.
To edit or create task you must complete the following information:

- Title.
- Description.
- Assigned users.
- Due date.
- Task type (before or during event).

Find events

You can find public events where you can be part of them. You are able to filter by date and keywords. To access to the find event page just navigate from navigation bar.
User homepage

In the user homepage you can manage events, friends, groups and tasks.