AALBORG UNIVERSITET

Smartphone Software for Department of Computer Science

COURSE: Software System Engineering 3 - Distributed Systems

TITLE : Smartphone Software for Department of Computer Science

AUTHORS: Marina Fernández, Gurvan Herve, Joan Puig, Côme Saint-André

SUPERVISORS: Mads Christian Olesen, Arne Skou

DATE: December 18, 2010

Title : Smartphone Software for Department of Computer Science

Authors: Marina Fernández, Gurvan Herve, Joan Puig, Côme Saint-André

Supervisors: Mads Christian Olesen, Arne Skou

Date: December 18, 2010

Overview

The presented document provides an overview of the *Smartphone Software for Department of Computer Science* project development through out several phases: analysis of the project proposal, preliminary study of the concepts involved in the project, design decisions and modeling, implementation, experiments and conclusions obtained in the end, as well as a reflection on possible future system improvements.

The final version of the system, which was built after the design and implementation decisions made through out the development of the project, is a prototype mainly composed of an application for Android smartphones leaning on a distributed architecture in order to provide all its expected functionalities.

Thus, the system is based on a mixture of the two main distributed systems architectural models: client-server and peer-to-peer. The functionalities that lean on the client-server architecture are those whose data are relative to information of general interest and need to be always (ideally) available: consulting news, information and schedules for courses, frequently asked questions, maps and indoors positioning.

On the other hand, functionalities such as the chat system and the possibility of contacting with the students of a certain course depend on the availability of the users, so a peer-to-peer architectural model was developed to support them.

Regarding the system functionalities, the in-door localization and the chat system were determined as the most relevant ones. With the aim to provide those functionalities, the choice made was to implement an in-door positioning based on the RedPin model and, on the other hand, to make use of an already existing Java solution to build a chat system by means of multicast DNS and DNS Service Discovery: JmDNS.

CONTENTS

INTRODUCTION

0.1. Purpose of this Report

The aim of the presented document is to provide an overview of the *Smartphone Software for Department of Computer Science* project development, proposed by and directed to use at Aalborg University.

This report is, thus, addressed to our supervisors and all whom this project may concern.

0.2. Description of the Project

This project consists mainly of solving the problems that new students at the Department of Computer Science face in their daily life, such as looking for rooms and consulting schedules, by developing an application for a modern smartphone that new students will be directed towards. This way, students may be helped as best as possible by using the smartphones they have.

On the other hand, the application is not only focused on students, but also on teachers: in addition, it will allow them to contact those of their students who are on campus for last-minute announcements.

In order to make this possible, an application for a smartphone device will be developed, maybe an Android phone, which will lean on a network access points infrastructure in the department and, perhaps, in one or more servers that could have to be also developed in order to provide some services, if needed (e.g. a database server functionality), so that the application could display the information requested by students and teachers.

Application functionalities, design and implementation decisions to achieve those functionalities will be described in detail in subsequent chapters.

0.3. Aims of the Project

The main objective intended to be met is **to develop an application** accessible to those students and teachers in Department of Computer Science who have a smartphone, and which will allow all of them to consult information related to the department and the people who are there. Application functionalities will change according to the role held by the user in the department, that is, student role or teacher role: to be precise, the application will provide teachers an additional functionality as it will be described later in the presented document. Each user will be signed up in the system with a certain associated role which will be recognized by the application when a user logs in providing some identification data.

The application will be developed for running on a smartphone operating system, which could be likely Android (based on a Linux operating system), but this decision will be made after a preliminary study of the available operating systems for smartphones.

User interface will be graphical in order to make it user-friendly and intuitive.

It could be also necessary to develop additional services which the application would need to provide the expected functionalities, such as a database server, but this needs will be studied and discussed at the design phase.

Since this project is conceived for academic purpose, its development also intends to meet some learning goals (for example, to study the different existing network architectures), which are interesting to the development team but not to the reader of this report, so they are not going to be identified here.

0.4. Overview of the Project's Final Version

A general perspective of the design and implementation decisions made throughout the development of the project, and which can be seen on the final given in version, is provided in this section.

Mobile Operating System

Since the most important objective to be met was to implement an application for smartphones, as said in the previous section, the first decision that was needed to make was on what smartphone operating system it was going to be run. After a study of the existing possibilities was made, Android operating system was chosen, what implied to develop a Java-based application by using specific packages and libraries for Android platforms.

System Architecture

The architecture of the system can be seen from two points of view: the distributed systems architectural models, which have to be with the system topology, and the software architecture, which has to be with the identification of sofware components.

The system is based on a mixture of the two main distributed systems architectural models: client-server and peer-to-peer. The functionalities that lean on the client-server architecture are those whose data are relative to information of general interest and need to be always (ideally) available, and therefore not to depend on devices whose availability is unpredictable (such as the user's smartphones). So, the use of a server was chosen as the best alternative in order to provide functionalities such as consulting news, information and schedules for courses, frequently asked questions, maps and indoors positioning. On the other hand, functionalities such as the chat system and the possibility of contacting with the students of a certain course depend on the availability of the users, and their performance may be adversely affected by introducing a server (this would introduce an unnecessary level of indirection: the server would act as an intermediary). Therefore, a peer-to-peer architectural model was developed to support those functionalities.

Regarding the software architecture, features of the system such as it would be a distributed system, and that the application would allow user interaction, led to structure the software according to two patterns: *Model-Controller-View* and *Layers*, which will be discussed in Chapter 3.

Security

Requirements regarding security needed to be met by means of encryption and authenticating communication. In order to achieve this, the *Transport Layer Security* protocol was chosen although not implemented, because the time available did not allow to deal with its complexity.

However, it could be possible to provide a minimum level of database accessing protection by defining views and different user privileges, as well as to ensure the privacy of the passwords stored in the database by applying a hash function to them.

In-door Localization

This functionality was determined as one of the most relevant in the system, and, in order to provide it, the choice made was to implement an in-door positioning based on the Red-Pin model (which will be described in Chapter 2).

Chat System

This is another of the most relevant functionalities, and with the aim of providing it, the decision to use the *Extensible Messaging and Presence Protocol* (XMPP) or its extension *XEP-0174* was made first (this will be detailed in Chapters 2 and 3). However, taking into account the time given for developing the project, the best option in practice was looking for the already existing solutions to do *multicast DNS* and *DNS Service Discovery*. Since it was required to work on an Android platform, it was finally necessary to use an available Java solution: *JmDNS*, which will be discussed in Chapter 4.

0.5. Summary of the Report

The presented document intends to provide an overview of the whole development of the project, and to be rigorous in its theoretical contents.

It is divided into chapters, which are at the same time divided in sections, in the logical order of the development process. The content which each chapter will tackle is briefly described below:

Chapter 0: Introduction. This is the present chapter, and it intends to provide a brief presentation of the project and the objectives intended to be achieved.

Chapter 1: Software Requirements Specification. In this chapter, a classified list of the given requirements will be provide and an analysis of the proposed system which meets those requirements will be detailed in terms of external behaviour of the system (i.e. user-system interaction).

Chapter 2: Preliminary Study. This chapter gives a theoretical basis on topics related to the project which subsequent design decisions will lean on.

Chapter 3: Design. Design decisions made starting from the previous chapter are explained here, as well as a deployment and a design view of the overall system.

Chapter 4: Implementation and Experiments. This chapter provides a description of the implementation phase and a list of tests performed on the system and their results.

Chapter 5: Conclusion and Future Work. A reflection on the objectives achieved is shown in this chapter, just as possible system improvements.

Appendix A: Requirements Document. This section provides a more detailed view of the software requirements and of the use cases shown in Chapter 1.

Appendix B: Use Case Realizations Document. This section complements Chapter 3 by describing how use cases are realized.

CHAPTER 1. SOFTWARE REQUIREMENTS SPECIFICATION

The aim of this chapter is to show an analysis of the project proposal given by the Department of Computer Science of Aalborg University. This will be made by means of both UML[1] modeling language [RJB99a] and textual explanations.

The scheme of this chapter is partially based on ANSI/IEEE Std. 830 - 1984 [web84].

1.1. Introduction

The purpose of this section is to let the reader know the different contents of the presented chapter, as well as its structure.

Firstly, a brief overview of the system will be provided: as said in the introduction chapter, it will be composed of an application which will manage all the information needed to perform the expected functionalities, information that will be requested from sources external to the application, but which will compose the whole system architecture together with it.

In subsequent sections of this chapter, an analysis of the proposed system will be detailed throughout UML diagrams and textual explanations where necessary to complement those diagrams.

1.1.1. Product Scope

The intention of the proposed smartphone software is to provide some functionalities that are not given currently; that is, this system does not have the aim to replace nor extend any similar previously existing system, the goal is to design and build a new system which meet user's current needs. However, the proposed system will be built over an already existing network infrastructure (e.g., WiFi[2] access points) and stored data the user will request, such as news feed or the data stored in a database.

The main objective of the proposed system, as said in the introduction, is to make it easier for students to consult useful daily information as that about their lectures and the schedules and rooms assigned, news and last-minute announcements, just as allowing student-to-student and teacher-to-student communication.

[1]UML (Unified Modeling Language) is a standard modeling language for conceptual and physical representation of systems, mainly used in software systems. It is supported by the OMG (Object Management Group) and it provides a graphical language for visualizing, specifying, constructing and documenting a software system. [RJB99b, page 15]

[2]The word WiFi is usually used to refer to a range of connectivity technologies including Wireless Local Area Network (WLAN) based on the IEEE 802.11 standards. [web10i]

1.2. Overall Description

This section specifies the main functionalities of the entire system and it describes user interfaces and how the user will interact with the system. It also defines the main physical restrictions which determines how the system works or its installation and, finally, it explains system dependencies on its environment.

1.2.1. Product Perspective

Smartphone Software for Department of Computer Science is a software system which not intends to extend a similar previously existing system, but which depends on some services already existing, such as news feed and database.

It is a system which will be installed in smartphones for department use and which will make use of a network (could be the existing WiFi infrastructure) and maybe several services as those already mentioned (stored data, news RSS[3] feed).

Users located at the department owning a smartphone will be able to use the system through these devices and they will be directed towards the application when in the department.

1.2.2. Product Functions

The main functionalities which the proposed system provides, described as an overall view, are the following:

- Show department news.
- Show information and schedules for courses.
- Show a map of the campus.
- Show a map of the department and position a user on it.
- Show a FAQ.
- Allow teachers to make last-minute announcements to their students who are on campus at a certain moment.
- Provide an in-house chat system to users.

All these functionalities will be subsequently further described.

[3]"RSS (most commonly expanded as Really Simple Syndication) is a family of web feed formats used to publish frequently updated works (such as blog entries, news headlines, audio, and video) in a standardized format". [web10h]

1.2.3. User Characteristics

The system will be accessed by two different types of users: students and teachers. All of them will be able to interact with the system by using a smartphone when at the department. From this point forward, the word *user* will refer to both types of users (students and teachers).

As the system will be installed at the Department of Computer Science environment, all students and teachers who will access the system hold these roles within the mentioned department, that is, they all are students or teachers at the department.

Users do not need any previous training for being able to use the application, since the user interface must be intuitive and usable. However, they need to be familiar with the smartphone device they are going to use, a requirement which can be assumed all users who are going to access the system already meet if they are going to use their own device to do that.

1.2.4. General Constraints

The application will be developed for a smartphone operating system within an Object-Oriented paradigm.

In addition, the system has to be built over and using the existing hardware resources, which can not be extended.

Finally, the system has to be completed before the stipulated deadline.

1.2.5. Assumptions and Dependencies

Software requirements have been drawn from the project proposal document that was given at the beginning, and completed in subsequent meetings with the project supervisors. On the other hand, as said before, requirements are also related to the existing infrastructure in the department and the availability of the hardware devices which that infrastructure is composed of. Thus, any change in the physical infrastructure and/or in the project characteristics, will directly affect the requirements stated in this document.

1.3. Specific Requirements

This section describes the software requirements and how the system satisfies them.

1.3.1. Functional Requirements

Functional requirements describe the high-level functionality of the system, and a textual description of them will be given in this subsection. A more formal description of these functional requirements can be found in Appendix A, presented in tables whose correspondence with the functional requirements described in this subsection is also indicated here.

According to the expected functionalities mentioned before (see subsection 1.2.2.), the functional requirements are the following:

[FRQ-0001] Show news. The system will show the department's latest news to those users who want to consult them (see Appendix A, Table A.1).

[FRQ-0002] Show schedules for courses. If a user wants to check her schedule for the present day, or those for another day in the future for a certain course, the system shall allow her to request this information (see Appendix A, Table A.2).

[FRQ-0003] Add schedules to calendar. When a user consults a schedule for a course, the system will provide her the possibility to add it to her phone calendar (see Appendix A, Table A.3).

[FRQ-0004] Show courses information. A user could need to consult the available information about a course. The system will show this information when requested (see Appendix A, Table A.4).

[FRQ-0005] Show map of the department. In order to help users to find rooms, the system will show a map of the department when the user requests it (see Appendix A, Table A.5).

[FRQ-0006] Show map of the campus. In order to help users to find the different departments or other university services, the system will show a map of the campus when the user requests it (see Appendix A, Table A.6).

[FRQ-0007] Show FAQ. If the user needs some help about the application, the system will provide a set of *Frequently Asked Questions* (starting from now, the word *FAQ* will be used instead) she will be able to consult (see Appendix A, Table A.7).

[FRQ-0008] Show indoors position. When an user is consulting the map of the department, she will have the possibility to request from the system her position on it (see Appendix A, Table A.8).

[FRQ-0009] Contact students. The system shall allow a teacher to contact her students (i.e. students that are registered for her course(s)) who are on campus for last-minute notices (see Appendix A, Table A.9).

[FRQ-0010] Chat. The system will give its users the option to chat with other users at the department (see Appendix A, Table A.10).

1.3.2. Non-Functional Requirements

Non-functional requirements describe aspects of the system which are not directly related to the system functionality. Non-functional requirements are usually divided into several categories according to the aspect they concern, such as: usability, reliability, performance, design constraints, security, user documentation and help systems, and interfaces. The few non-functional requirements that were given can be classified as design constraints and security requirements, and a textual description of them will be provided in this subsection. A more formal description of these non-functional requirements can be found in Appendix A, presented in tables whose correspondence with the non-functional requirements described in this subsection is also indicated here.

Thus, the identified design constraints are the following:

[NFR-0001] Platform. The application will work on an operating system for smartphones (see Appendix A, Table A.11).

[NFR-0002] News feed. Department's news will be consulted from the department's RSS feed (see Appendix A, Table A.12).

And the identified non-functional requirements regarding system security are which follows:

[NFR-0003] Routing table protection. The system will protect the routing table against illegal operations (see Appendix A, Table A.13).

[NFR-0004] Reliable message delivery. The system will guarantee that messages will not be neither accessed, nor modified, and legitimate transmitter's identity will not be supplanted (see Appendix A, Table A.14).

[NFR-0005] Authorized users. Only authorized users will be able to access the chat system and contact students (see Appendix A, Table A.15).

[NFR-0006] Password protection. The system will store and send/receive this kind of data in an encrypted way to keep privacy (see Appendix A, Table A.16).

[NFR-0007] Connection loss detection. The system will realize that a user has lost her connection within the first subsequent minute (see Appendix A, Table A.17).

1.3.3. Use Case Model

The *use case view* models the functionality of the system as perceived by outside users, called *actors*. A *use case* is a coherent unit of functionality expressed as a transaction among actors and the system. The purpose of the use case view is to list the actors and the use cases and show which actors participate in each use case. [RJB99a, page 26]

1.3.3.1. General Description of Actors

Actors who interact with the proposed system are:

Student. This actor represents a general user at the Department of Computer Science, typically a student.

Teacher. This actor represents a teacher at the Department of Computer Science.

Not Authenticated User. This actor represents a user at the Department of Computer Science who is not logged in the system yet (and maybe will not be).

This actually is a hierarchical structure of actors where the most restrictive one in terms of the services she can request to the system is *Not Authenticated User* and the least one is *Teacher*, who inherits the set of accessible services from the more restrictive users than herself and extends it. This inheritance hierarchy of actors is shown in the subsequent use case diagram (see Figure 1.1).

1.3.3.2. Use Case Diagram

The analysis use case diagram for the proposed system is shown in Figure 1.1.

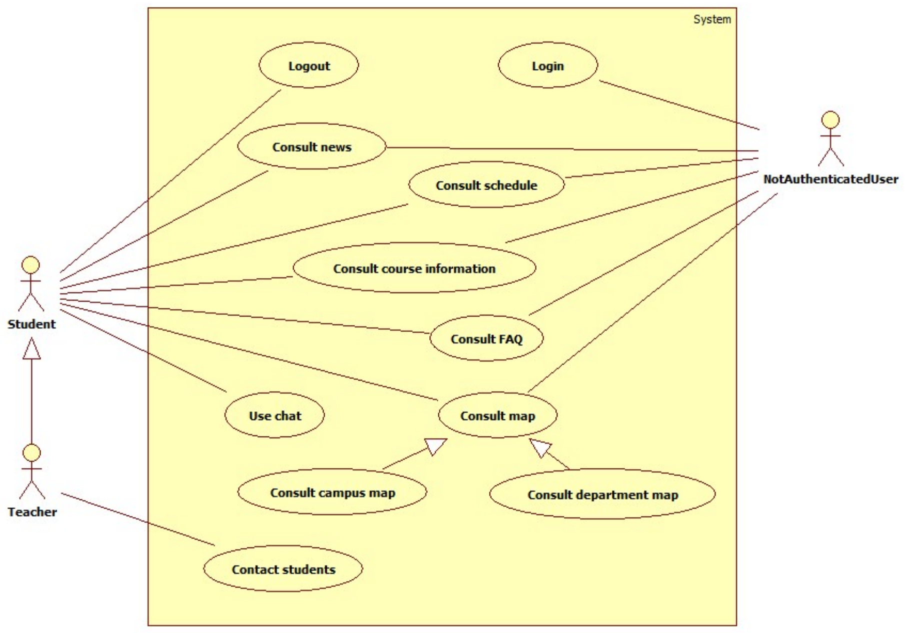


Figure 1.1: Analysis - Use Case diagram

1.3.3.3. Use Case Event Flows

Use cases shown in the previous use case diagram will be textually detailed in this section, so that the reader can know action sequences carried out by the system while interacting with actors, i.e. the external behaviour of the system. A more formal description of these use cases can be found in Appendix A, presented in tables whose correspondence with the event flows described in this subsection is also indicated here.

Use case event flows for the existing use cases are the following:

[UC-0001] Login. An user who is not authenticated yet in the system, can provide her username and password by using her smartphone to open a session. If the authentication data given by the user is not correct, the system will request for them again (see Appendix A, Table A.18). This login functionality shall allow the user to access the chat system and to contact students if user is a teacher.

[UC-0002] Logout. A student or a teacher who is logged in the system, can close her session by selecting this option (see Appendix A, Table A.19).

[UC-0003] Consult news. An user can consult the latest department's news by selecting the option on her smartphone. Then, the system will request the information and will show them to the user on her smartphone display (see Appendix A, Table A.20). Users do not need to be logged in the system to access this functionality.

[UC-0004] Consult schedule. An user who wants to consult the schedule for a course, can request them from the system. The system will show a list of the courses the user can consult, and then the user will be able to select one. Default schedules shown then by the system are for the current day, but the user will be able to select another period (e.g. a week, a month), and the system will show schedules for that period. In addition, a user will be able to add schedules to her smartphone calendar if she chooses that option (see Appendix A, Table A.21). Users do not need to be logged in the system to access this functionality.

[UC-0005] Consult course information. An user who wants to consult information for a course, can request it to the system. The system will show a list of the courses the user can consult, and then the user will be able to select one. Then, the system will display the requested information (see Appendix A, Table A.22). Users do not need to be logged in the system to access this functionality.

[UC-0006] Consult map. Realization of this use case is described in its sub-use cases realization (see Appendix A, Table A.23).

[UC-0007] Consult campus map. An user will be able to see a map of the campus by selecting that option. Then, the system shall display the map on the user's smartphone (see Appendix A, Table A.24). Users do not need to be logged in the system to access this functionality.

[UC-0008] Consult department map. An user will be able to see a map of the department by selecting that option. Then, the system shall display the map on the user's smartphone. In addition, when an user is consulting the map of the department, she will have the possibility to request the system her position on it. Then, the system shall obtain this position and shall display it on the map (see Appendix A, Table A.25). Users do not need to be logged in the system to access this functionality.

[UC-0009] Consult FAQ. An user who needs help with the application features, can request the system a FAQ. Then, the system will display a list of the existing topics, which the user will be able to select and read on her smartphone display (see Appendix A, Table A.26). Users do not need to be logged in the system to access this functionality.

[UC-0010] Use chat. A logged in user who is at the department, can access the chat system by selecting this option. Then, a list of the available users also using the chat system will be displayed. The user will be able to chat with them by writing in the general chat board, which is displayed to all users, or by selecting a certain user from the list, in such case the written message will be only displayed on that user's smartphone (see Appendix A, Table A.27).

[UC-0011] Contact students. A logged in teacher can select this option for giving a notice to her students. The system will show a list of her courses, and the teacher will be able to choose one from that list. Then, a written message can be sent to the students who enroled that course and who are at the department at that moment (see Appendix A, Table A.28).

1.3.4. Analysis Model

> The analysis model[4] should be a minimal representation of the system being modeled, sufficient to capture the essential logic of the system without getting into issues of performance or construction. [RJB99a, page 45]

1.3.4.1. Initial Class Diagram

> Classes can exist at several levels of meaning in a model, including the analysis, design and implementation levels. [...] An analysis-level class represents a logical concept in the application domain or in the application itself. [RJB99a, page 45]

The analysis class diagram for the proposed system is shown in Figure 1.2.

1.3.4.2. Classes Description

Classes shown in the previous class diagram will be textual detailed in this section, so that the reader can know which concepts each class represents and what their objects can do, i.e. the responsibilities of each class:

SmartphoneDisplay. This class represents a smartphone device through which a user will access the system and results for her requests will be displayed.

TeacherSmartphone. It represents the additional functionalities a teacher will be able to access via her smartphone.

PhoneCalendar. This class' responsibility is to store schedules in a structured way in the user's smartphone.

CSUser. This class represents a generic user at the Department of Computer Science, regardless of the role performed by her. It stores personal data of that user.

Student. This class represents an user performing a student role in the system.

Teacher. This class represents an user performing a teacher role in the system.

NewsFeed. It represents the RSS news feed of the department which will be accessed, and therefore, its responsibility is to store the department's news.

Course. This representation of a single existing course in the Department of Computer Science has the responsibility of storing its related information and keeping knowledge of its associated schedules.

[4]"A model is a complete description of a system at a given precision from one viewpoint. There may be several models of a system from various viewpoints - for example, an analysis model as well as a design model". [RJB99a, page 37]

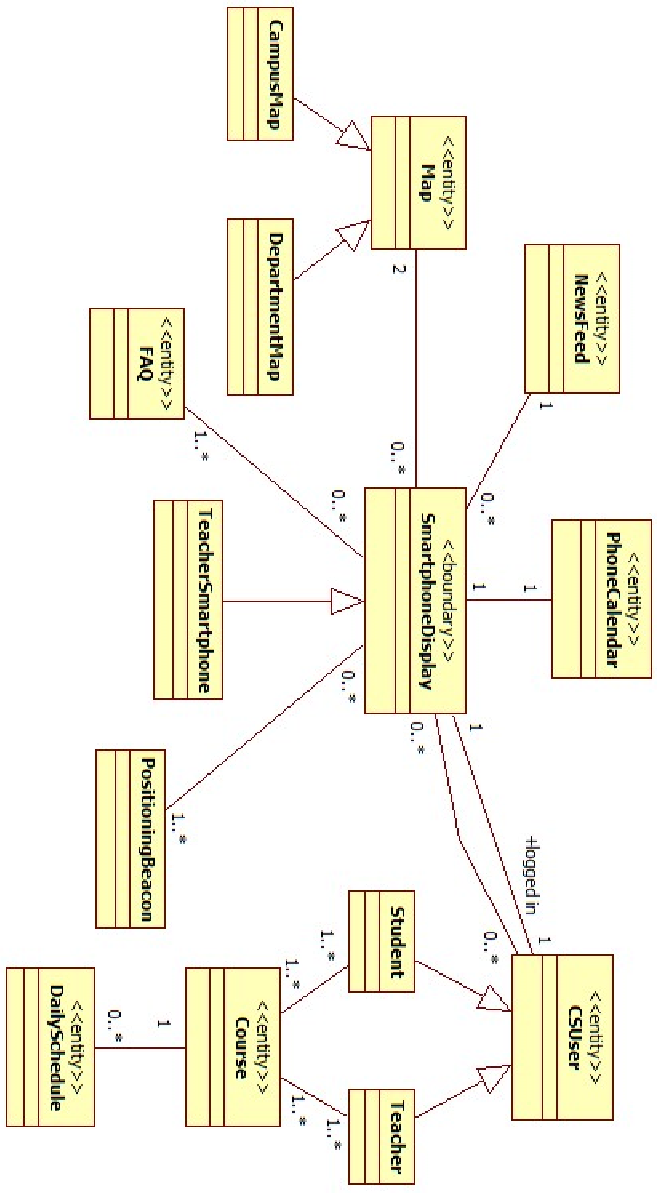


Figure 1.2: Analysis - Class diagram

DailySchedule. It represents the schedule for a certain day, associated to a certain course. Its responsibility is to keep that information.

Map. This class represents a map in the system, regardless of the kind of map.

DepartmentMap. This class represents a map of the department. Its responsibility is to keep the information needed by the application to show the map when requested.

CampusMap. This class represents a map of the University campus. Its responsibility is to keep the information needed by the application to show the map when requested.

FAQ. This class stores a *Frequently Asked Question* description.

PositioningBeacon. It represents a device which will provide the data needed to find out the position of a user at the department. Therefore, its responsibility is to keep that data and calculate the position based on that data.

1.3.5. User Interface

In this section, several screenshots and navigation of an initial user interface prototype will be shown. Providing some hand-made drafts of the user interface at this point complements the analysis of the proposed system and makes easier for the reader to understand what the system is expected to offer to users.

When designing the user interface, the goal of making it easy to use and intuitive should not be lost sight of. Thus, a main menu for the smartphone application could seem like shown in Figure 1.3. This menu should show a list of the main functionalities the system will have to provide, associating icons to each one to make easier to the user to quickly make an impression on what each functionality offers (e.g., an icon of a calendar could be used to represent *[UC-0004] Consult schedule* functionality).

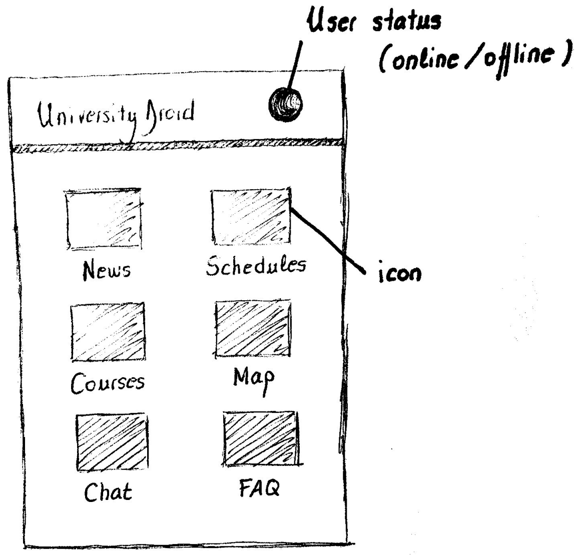


Figure 1.3: User interface - Draft of the main menu

Some of the functionalities can be grouped together: *[UC-0007] Consult campus map* and *[UC-0008] Consult department map* could be both accessed by clicking on the "Map" icon represented in Figure 1.3, and *[UC-0011] Contact students* could be accessed by clicking the "Chat" icon, which should allow to access *[UC-0010] Use chat* first.

An icon representing the user status (online: logged in, offline: logged out) could be also displayed, as shown in the top right corner in Figure 1.3. A submenu could display the access to those *[UC-0001] Login* and *[UC-0002] Logout* as can be seen in Figure 1.4, whose state is represented by the user status icon mentioned before. The icon to access both functionalities can be named as "Settings", which is a standard in most of these devices.

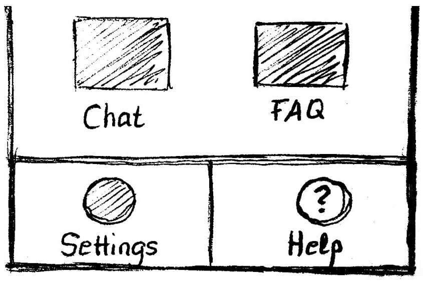


Figure 1.4: User interface - Draft of the submenu accessed from the main menu

Then, when selecting "Settings", a login dialog similar to which is shown in Figure 1.5 could be displayed. The login view should be composed of the standard form requesting "username" and "password" and buttons to submit the form ("OK" button) or close the dialog without submitting data ("Cancel" button), since it is usually the well-known way to log in a system by most of users.

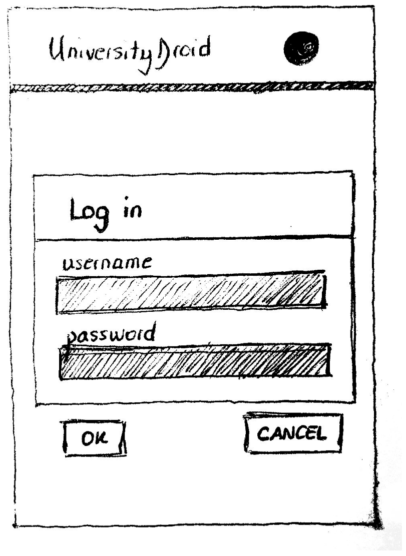


Figure 1.5: User interface - Draft of the login dialog

One of the most interesting functionalities the system will have to provide is *[FRQ-0008] Show indoors position*, whose behaviour was described as part of *[UC-0008] Consult de-*

partment map. As said before, a map of the department will be displayed when clicking on the ”Map” icon in main menu. Then, a pointer could be also shown to indicate the user’s position on the map, as shown in Figure 1.6.

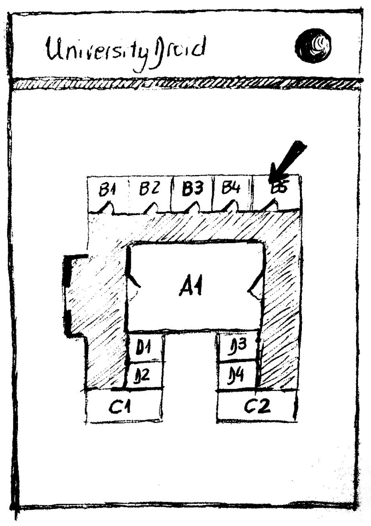


Figure 1.6: User interface - Draft of the department map with the user’s location pointer

Other of the most interesting functionalities is *[UC-0010] Use chat*. Figure 1.7 shows a first user interface design for the chat system: the top half of the screen could be used to show the messages user’s will be able to send. In the bottom half, a text box for writing those messages and sending them by clicking a ”Send” button and a list of the logged in users could be placed.

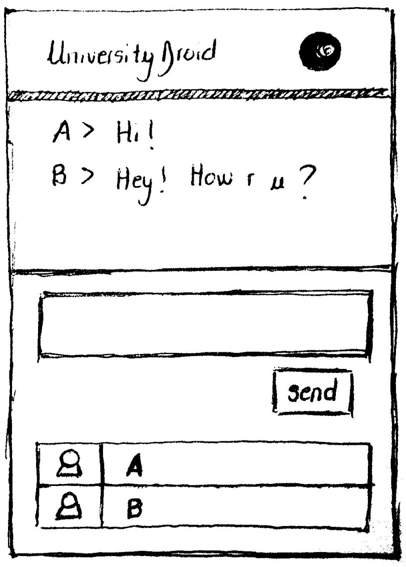


Figure 1.7: User interface - Draft of the chat system

Finally, about the rest of the functionalities (news, courses information, schedules and FAQ), they could be displayed in a similar way: text or tables showing the requested information should be placed below the heading with the title of the application (”University-Droid”) and the user status icon, which has been stated as fixed heading.

CHAPTER 2. PRELIMINARY STUDY

This chapter presents a theoretical study of the concepts needed to make the design decisions which will lead to solve how this project will meet the requirements described in the previous chapter. Those requirements have described a problem to be solved, and a problem can often be solved in several ways. Even if only one way for solving the problem exists, the techniques for achieving it needs to be previously studied. For that reason, it is necessary to find how to solve the problem in the most suitable way, taking into account the available resources for that.

This preliminary study will be carried out in two phases: a first one which deal with concepts and technologies that will be the basis for the overall system, and a subsequent one which already focuses on the most complex specific functionalities the system will have to provide.

Therefore, what is going to be firstly made is a research of those aspects which the system will be based on, such as operating systems for mobile devices, distributed systems and security issues.

Once the basis for our system is studied, the subsequent task will be finding methods and techniques to provide each individual complex functionality the system is expected to offer. Thus, in the rest of the chapter, the following issues are going to be dealt with:

- *[FRQ-0008] Show indoors position* functionality: In-door localization methods.
- *[FRQ-0010] Chat* functionality: Chat systems.

2.1. Mobile Devices Operating Systems

Today there are a lot of operating systems for mobile devices, the most popular being: iPhone/iOS [App10], Symbian [Sym10], Android [web10a], etc.

Figure 2.1 shows the market of smartphones. The most used OS is Symbian, but as shown in Figure 2.2, developers prefer to use iPhone, Android and iPad instead of Symbian.

According to [Com10], the operating system for smartphones with the best market growth in the US[1] is Android (see Figure 2.4[2]) even with the new launch of iPhone 4, and according to [Mil10], in 2010, Android will be the platform used by most developers to create new applications (see Figure 2.3).

The most relevant OS in the market will be described: Symbian, iPhone and Android.

[1]The US market can be considered an indicator for the behavior of the market around the world.

[2]Notice that Figure 2.4 does not show Symbian OS because of the few developers interested in developing software for this platform, so that, the figure only takes into account the platforms with the best projections.

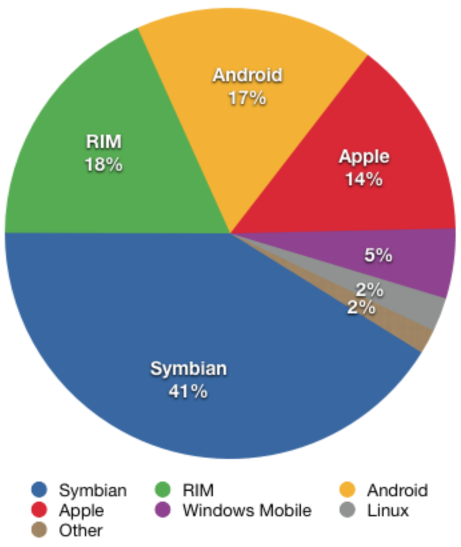


Figure 2.1: Smartphone OS market. Source [Com10]

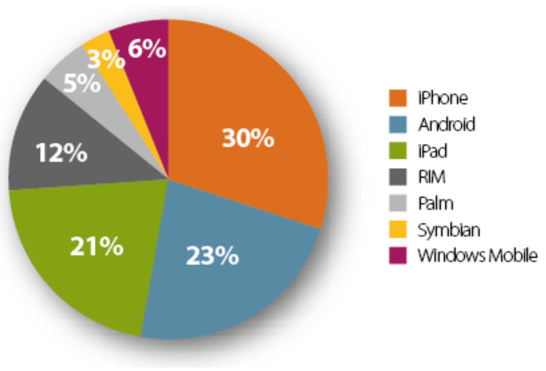


Figure 2.2: Mobile app platforms for which publishers are developing in 2010. Source [Mil10]

2.1.1. Symbian

This operating system is the most used and it was originally developed by Symbian Ltd. In 2008 Symbian Ltd. was acquired by Nokia and an independent non-profit organization called Symbian Foundation [Sym10] was created.

Symbian is one of Nokia's mobile Operating systems for mobile devices and smartphones, and it offers developers a lot of languages to develop applications: C++, Qt, Python, Web Runtime, Java, and a few others.

Notice that Figure 2.1 shows the main OS for smartphones used without taking into account the version. Symbian has a lot of versions for smartphones and most of them do not offer the features many developers require such as WiFi, GPS, etc.

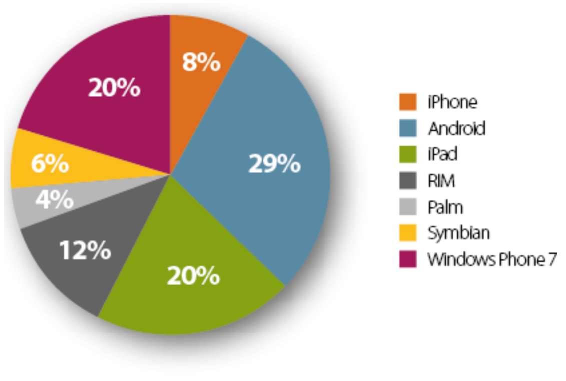


Figure 2.3: New app platforms publishers plan to support in 2011. Source [Mil10]

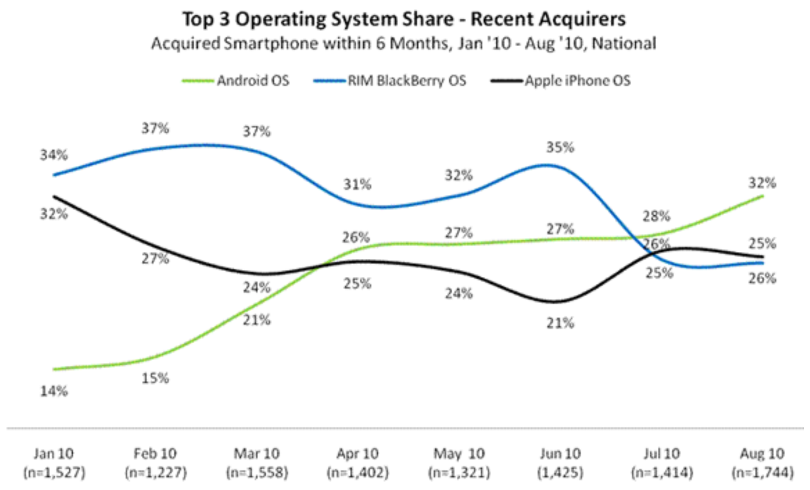


Figure 2.4: Android most popular operating system in US. Source [Com10]

2.1.2. iPhone/iOS

iPhone is the smartphone of Apple Inc. The First iPhone was introduced on January 9, 2007. The latest version (iPhone 4) runs the iOS4 operating system which is also the same operating system used on devices like iPod Touch, iPad and Apple TV.

iOS 4 is based on Darwin OS[3] that is found in Mac OS X [Inc10] and is programmed using C and C++. But this operating system only runs on Apple devices, making it impossible to run the system on other phones such as those from Nokia, Samsung, etc.

To develop an application for iOS 4, Xcode and the iOS SDK (Software Development Kit) are needed. Xcode is an Apple Inc. suite to develop software for the iOS. The iOS SDK allows the developer to write software for iOS and provide some tools, such as the "iPhone simulator" (Figure 2.5).

[3]Open source POSIX (Portable Operating System Interface for Unix. see [IEE03]) released by Apple Inc. and it is compatible with the Single UNIX Specification version 3 (SUSv3) and POSIX UNIX applications and utilities (see [Bra03])

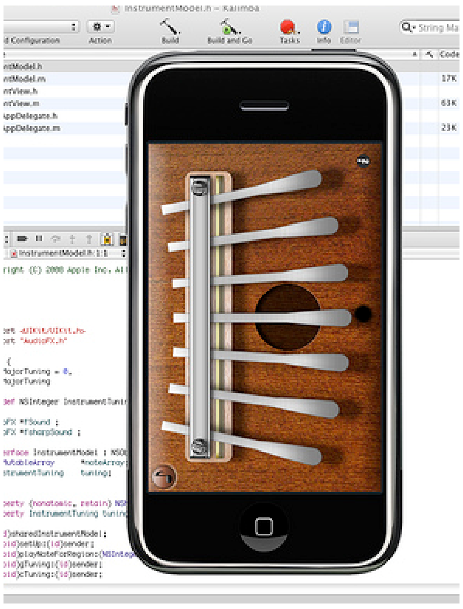


Figure 2.5: iPhone Simulator.

2.1.3. Android

Android (based upon a modified version of the Linux kernel) is an open source operating system for mobile devices initially developed by Android Inc., but in 2005 Google Inc. purchased Android Inc. and made it possible to install Android OS on all kinds of smartphone.

According to [web10g], there is a large community writing applications (“apps”) for Android, currently there are over 70,000 apps available for Android with some estimates saying 100,000 have been submitted.

Developers use Java as the main language (but native language can also be used) to control the device via Google-developed Java libraries (SDK, Source Development kit) [web10c] or using C with NDK (Native Development Kit) [web10b].

According to [web10e], Android has the following features:

- Application framework enabling reuse and replacement of components
- Dalvik virtual machine optimized for mobile devices
- Integrated browser based on the open source WebKit engine
- Optimized graphics powered by a custom 2D graphics library; 3D graphics based on the OpenGL ES 1.0 specification (hardware acceleration optional)
- SQLite for structured data storage
- Media support for common audio, video, and even image formats (MPEG4, H.264, MP3, AAC, AMR, JPG, PNG, GIF)
- GSM (Global System for Mobile Communications) Telephony (hardware dependent)

- Bluetooth, EDGE, 3G, and WiFi (hardware dependent)
- Camera, GPS, compass, and accelerometer (hardware dependent)
- Rich development environment including a device emulator, tools for debugging, memory and performance profiling, and a plugin for the Eclipse IDE (integrated development environment).

The Android architecture is divided into the following five sections (Figure 2.6):

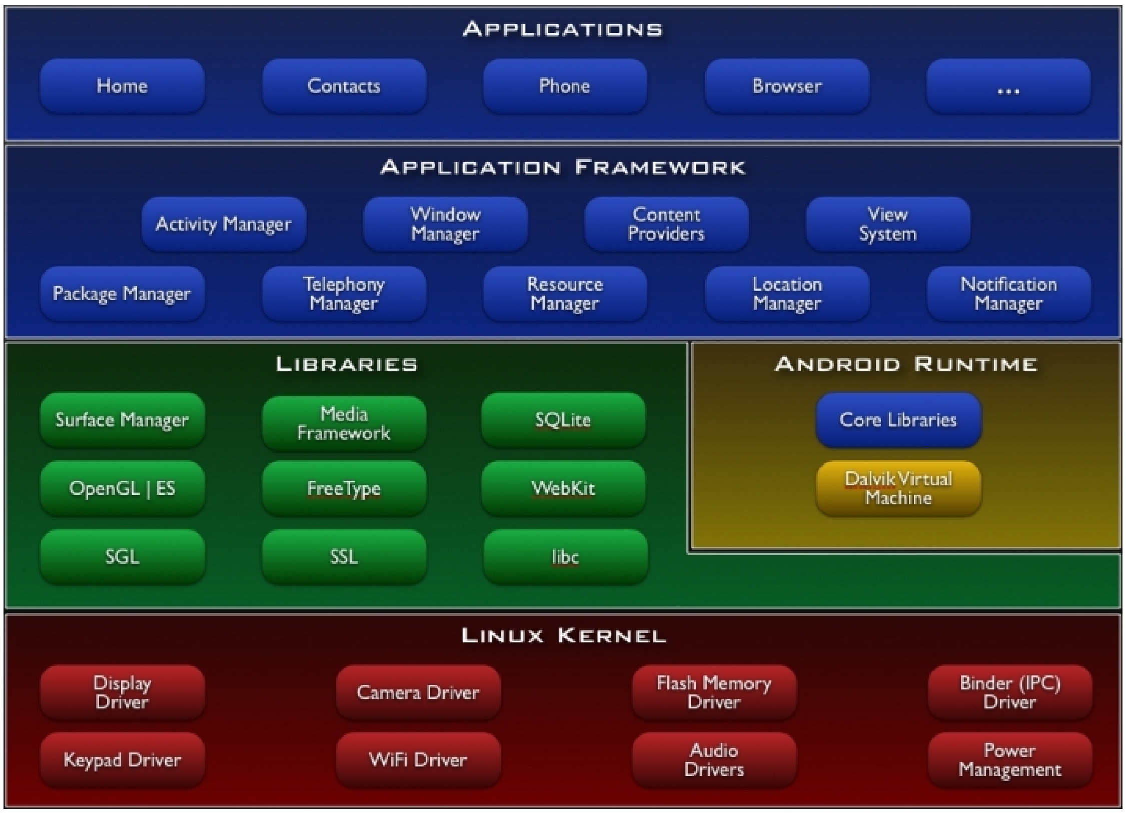


Figure 2.6: Android architecture. Source [web10e].

- **Applications:** Android gives developers the possibility to interact with core applications like mail, SMS program, maps, contacts and others. All the applications are written using the Java programming language.
- **Application Framework:** The main application framework is available to all developers, as a consequence is easy to set alarms, run background services, access local information, add notifications to the status bar, and more. In fact, other applications can publish their own capabilities and any other application may then make use of those capabilities (subject to security constraints enforced by the framework).
- **Libraries:** the developer can also use the included set of C/C++ libraries using the Android Application framework. These libraries include: SQLite, 3D libraries, SGL (2D libraries), LibWebCore (web browser engine), Media Libraries (to process video, audio and image data), etc.

- **Android Runtime:** Here it is the Dalvik VM (Virtual Machine), which runs the programs developed using Java and which is optimized to run with the Android OS.
- **Linux Kernel:** And finally, the kernel of the operating system. Android relies on Linux version 2.6 to manage the essential services like security, memory management, process management, network stack, and driver model

Many tools exist to develop applications for Android. Such tools are available from the official website (see [web10a]):

- **Android SDK:** This package provides the developer with all the necessary tools (emulator, framework, debugger, and more) to develop applications for Android. Figure 2.7 shows the emulator.

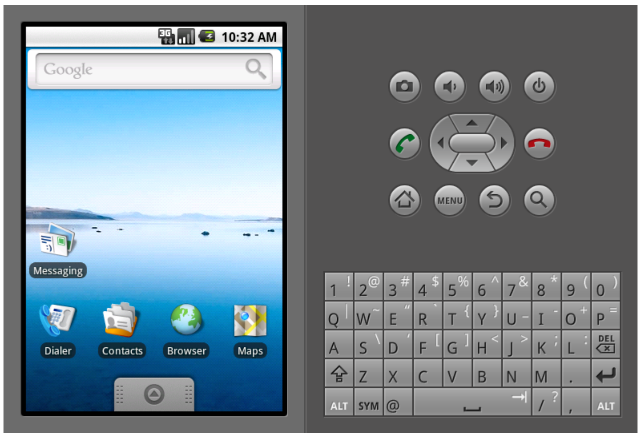


Figure 2.7: Android emulator.

- **Eclipse ADT plug-in:** ADT (Android Development Tools) is an Eclipse [web10f] plug-in to allow a better integration of the Android SDK with the IDE.

2.2. Distributed Systems

According to [Uni10], a distributed system is:

> An application that executes a collection of protocols to coordinate the actions of multiple processes on a network, such that all components cooperate together to perform a single or small set of related tasks.

2.2.1. Example of a Distributed system

In this section an example of a simple distributed system is explained. Imagine that a user wants to access a web page. If there is only one user connected to the server, the system is enough, but what happens if there is one user one day using dynamic[4] content, 20 tomorrow at the same time, 400 in two days, 160,000 in three days, and the amount of users that connect keep growing?

An administrator will need more than one computer, and these computers have to work as one to provide the information. Then the new web page needs a distributed system to provide the information correctly.

Figure 2.8 shows the scenario of a distributed system, where many users want to access a service, offered by the server cloud.

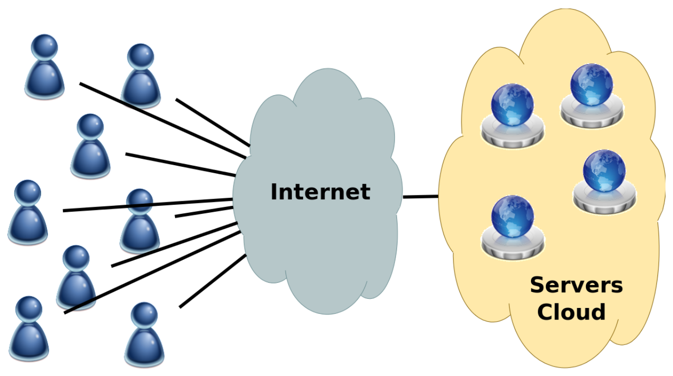


Figure 2.8: Distributed System example.

Using this strategy, the web page can continue grow, and if it needs more servers, the distributed system will allow more servers to be added

2.2.2. Main characteristics of a distributed system

Knowing the base of a distributed system, then the main features that all distributed systems have can be explained.

A distributed system can be huge, but all systems share some problems. The classical design issues are [Nes99]: 1) scalability, 2) heterogeneity, 3) objects (representation, encoding and translation), 4) resource management, 5) protection (security and privacy), 6) naming, 7) error control, 8) synchronization, 9) and measurement, testing and debugging.

1. **Scalable:** The system has the ability to either handle growing amounts of work in a graceful manner or to be enlarged [Bon00].

2. **Heterogeneity:** Not all the nodes are the same, for this reason, the system has to take into account these differences (different processing speeds, bandwidth, availability, administrators, software etc).

3. **Objects (representation, encoding and translation):** The distributed Systems handle a lot objects (moving, creating, coping, modifying, etc.), so that, the sys-

[4]Depending of the kind of content (dynamic or static) that the server provide, the quantity of users supported at the same time varies. See [CAMZ] to get more information.

tem needs a distributed programming model (like Java, CORBA, etc.) to access the objects and present them to the client.

4. **Resource management:** The system has to manage all resources efficiently (like CPU, process, threads, memory, bandwidth, etc) and try to use it fairly.

5. **Security:** The three pillars of information security are the following [Olz10]:

   - Confidentiality: concerned with making sure the wrong people can't see sensitive information
   - Integrity: ensuring all data, whether medical, business, or financial, is accurate
   - Availability (continuity): keeping the bad guys away from access they can use to take down a system or entire network (i.e., killing one or more critical business processes)

   For example, the security of a distributed system can be achieved by securing the processes and the channels used for their interactions and by protecting the objects (see Figure 2.9) that they encapsulate against unauthorized access [GC05, page 57-61].

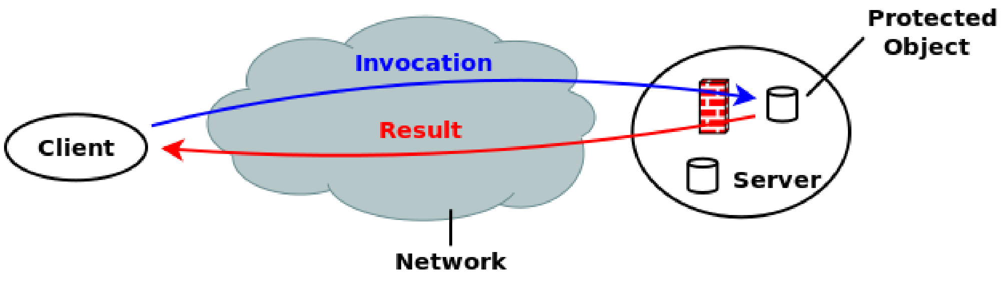


Figure 2.9: Protecting objects

   Section 2.3. shows more information about the available security protocols.

6. **Naming:** Naming is used to denote entities (like hosts, files, users, etc.) in a distributed system, but to operate on an entity an access point is needed to get the information to localize the entity. Figure 2.10 shows a simple example of a client looking for an entity.

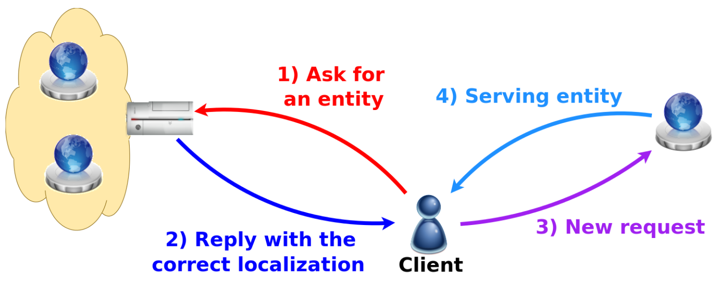


Figure 2.10: Finding an entity.

7. **Error Control:** In a distributed system, a lot of elements can cause errors: a firewall that drops the current connection, a router or link failure, broken hard disk, etc. The distributed system has to be able to detect these problems (for example, using out-of-band channels to notify a responsible technician or system administrator) and recover the service as soon as possible.

8. **Synchronization:** This is one of the most difficult problems in distributed systems. If the information is stored in more than one computer, this information has to be the same. So if a client is connected to the server A and then is connected to the server B, the information of both servers has to be the same.

9. **Measurement, testing and debugging:** The implementation of distributed applications requires measurement, testing and debugging to ensure applications behave correctly and perform well. But this is not an easy task, because this service must not affect the system and must take into account hundreds of elements that can fail.

2.2.3. Distributed System Architectural Models

As stated in [CDK00, pages 30–31]:

> An architectural model defines the way in which components of systems interact with one another and the way in which they are mapped onto an underlying network of computers. The overall goal of an architectural model is to ensure that its components structure will meet present and likely future demands on it. Major concerns are to make the system reliable, manageable, adaptable and cost-effective.

A set of several architectural models for distributed systems can be found, where client-server and peer-to-peer are the main ones as said in [GC05, page 35].

2.2.3.1. Client-server

According to [GC05, page 35], this is the architecture that remains the most widely employed. As the name of the architecture indicates, there are two types of processes: server processes and client processes. Server processes are allocated in separate host computers and they manage shared resources. Client processes in other host computers interact with server processes individually with the aim of accessing their shared resources (see Figure 2.11).

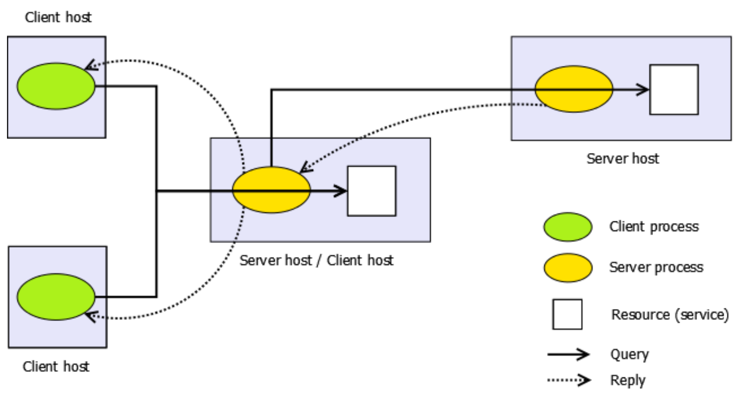


Figure 2.11: Client-server architecture

In other words, server processes provide a service. A client process that requires that service issues queries and waits for responses. Also servers can perform as clients of other servers, as shown in Figure 2.11.

2.2.3.2. Peer-to-peer

In the peer-to-peer architecture model all processes play similar roles, with equivalent responsibilities, interacting cooperatively as peers to perform a distributed activity or computation without any distinction between clients and servers [GC05, page 35]. In fact, processes in a peer-to-peer architecture are both clients and servers at the same time, and each process can send and receive requests to the same interaction interface. In this model, code in the peer processes maintains consistency of application-level[5] (see Table 2.1) resources and synchronizes application-level actions when necessary. The pattern of communication will depend on application requirements. [CDK00, page 36]

The system has to take into account that each peer connected to the network has different characteristics and each connection is very volatile, their owners don't guarantee any service.

2.2.3.3. Overlay Network

If a computer needs to send data, it will use IP routing[6]. But the problem of this protocol is that the computer which sends the data, doesn't know the path that the data will follow.

According to [And01], an overlay network is:

[5]The application layer is the OSI layer closest to the end user, which means that both the OSI application layer and the user interact directly with the software application. This layer interacts with software applications that implement a communicating component. Such application programs fall outside the scope of the OSI model. Application layer functions typically include identifying communication partners, determining resource availability, and synchronizing communication. Source [CIS99]

[6]IP routing is a set of protocols that determine the path that data follows in order to travel across multiple networks from its source to its destination (see [CIS] for more information)

	Data unit	Layer	Function
Host layers	Data	7. Application	Network process to application
		6. Presentation	Data representation,encryption and decryption,convert machine dependent data to machine independent data
		5. Session	Interhost communication
	Segments	4. Transport	End-to-end connections and reliability,Flow control
Media layers	Packet	3. Network	Path determination and logical addressing
	Frame	2. Data Link	Physical addressing
	Bit	1. Physical	Media, signal and binary transmission

Table 2.1: OSI layers. Source [CIS99]

a "virtual" network created on top of an existing network. The nodes in the overlay network use each other as routers to send data. The connections between the overlay nodes are carried over the underlying network; in this case, the Internet. Overlay networks have the potential to improve the reliability perceived by applications because they permit aggressive path exploration and maintenance.

The following list explains the main characteristics of a overlay network [YL04] [MC02] [QL02] [EKLMP05]:

- **Topology:** how the peers are connected to each other in the overlay network. There are some topologies like: star, ring, tree, mesh, hybrid (see figure 2.12).

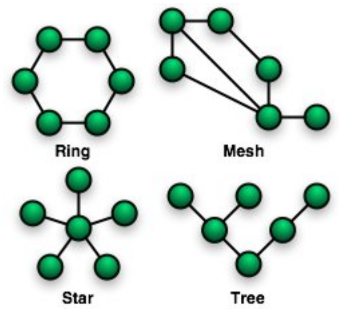


Figure 2.12: Some overlay network topologies

- **Localization:** the method to localize the information. The localization of the resources could be centralized or distributed (flood, hash, etc.)
  - **Centralized:** a super-peer or server exist that provides a list to the peers with all the available resources. This is easy to implement but it is not very scalable.

- **Distributed:** the resources list is distributed over the network, each peer has a piece of the list. The resource list is obtained by combining all the pieces. This is the most complicated to implement topology, but it is more scalable. Some of these architectures are detailed below:
  - **Flooding:** when a node is looking for some information, it asks all its neighbours and each neighbour repeats the process until a given TTL (Time To Live). To add a node to the network, it has to know at least one connected peer. The problem is that this method generates a lot of traffic.
  - **Routing:** each node contains an identifier table for its neighbours, but the table does not contain information about the content of its neighbours. When a node receives a request, it tries to calculate what is the best route, and sends it only once. This process continues until the TTL expires. If the same content is stored in more than one node, the node who starts the search, will only receive just one reply (see Figure 2.13)

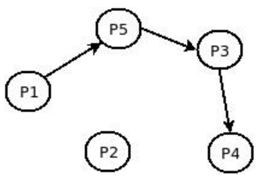


Figure 2.13: Routed request

  - **Distributed Hash Tables (DHT):** in this architecture a global table is created, but it is divided, so each node has a piece of the table. This method has an advantage, it allows any kind of data to be distributed uniformly across the entire network.

- **Network management:** network management are the methods to control all the nodes (basically insert and remove). In the centralized topologies it is easier, but less scalable, on the other hand, the DHT topologies are more complex but are more scalable.

- **Data transfer:** how the data is transferred between the nodes. There are two possibilities:
  - **Whole data transfer:** when a node is downloading a file, this data is being downloaded as a unique block, therefore if they lose the connection, the whole file must be downloaded again.
  - **Block transfer:** the main file is split in a lot of blocks (or chunks) and the client can obtain each chunk from different sources, so if the client loses the connection with a node, it doesn't need to download the whole file again.

Table 2.2 shows a comparison between IP and Overlay Routing.

Table 2.3 shows the main pros and cons of the peer-to-peer architectures.

	IP	Overlay
Scale	IPv4 is limited to 2^{32} address table nodes and IPv6 is limited to 2^{128}, but a lot of this space is preallocated according to administrative requirements	Peer-to-peer systems can address more objects. The GUID name space is very large a flat ($> 2^{128}$) allowing it to be much more fully occupied
Load Balancing	Loads on routers are determined by network topology and associated network patterns	Object locations can be randomized and hence traffic patterns are divorced from the network topology
Network dynamics (addition/deletion of objects/nodes)	IP routing Tables are updated asynchronously on best-efforts basis with time constants on the order of 1 hour	Routing tables can be updated synchronously or asynchronously with fractions of a second delays
Fault Tolerance	Each IP address maps to exactly one target node	Messages can be routed to the nearest replica of a target object
Security and anonymity	Addressing is only secure when all nodes are trusted. Anonymity for the owners of addresses is not achievable	Security can be achieved even in environments with limited trust. A limited degree of anonymity can be provided.

Table 2.2: Distinctions between IP and overlay routing for peer-to-peer applications. Source [GC05, page 400].

2.2.4. IP Broadcast

This section explains IP (Internet Protocol) Broadcast (RFC 919 [Mog84])which is a way to implement peer-to-peer broadcasts on a local network.

IP Broadcast uses the broadcast address to send messages to all the hosts on the local network. The broadcast address is a logical address that tags all the nodes. Figure 2.14 shows an example of the broadcast traffic, where the router only sends the package to the nodes in the network (LAN), it does not send the packet to the Internet.

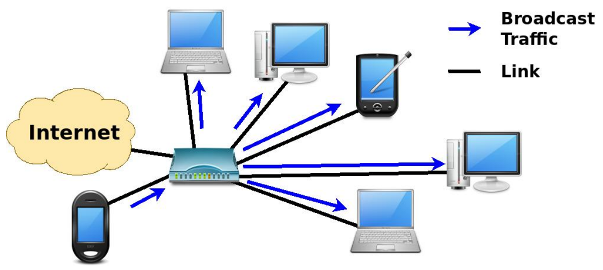


Figure 2.14: Broadcast traffic example.

Some applications that use IP broadcast are BOOTP (RFC 951 [CG85]) and DHCP (RFC 2131 [Dro97]) clients, which use IP broadcast to find and send requests to their respective servers.

P2P architecture	Pros	Cons	Examples
Distributed	No central server, higher fault tolerant, simpler architecture	More network resource consumption, low scalability	Gnutella, Freenet
Centralized	Less network resource consumption, high scalability	Central server dependent, less fault tolerant	Napster

Table 2.3: Pros and cons of different peer-to-peer architectures. Source [LK02]

This is a good method to find nodes and send information in a efficient way in a Local Area Network, but it has a problem:

- The network has to allow broadcast messages.
- There is no guarantee of sending a message without errors because it can not use a virtual circuit connection (like TCP, see RFC [VC74]).
- The size of the package is limited so the information contained is also limited.

But it allows for the sending of information in an efficient way and finds hosts on the network.

The broadcast address (for an IPv4 host) can be obtained by performing a bitwise logical OR operation between the bit complement of the subnet mask and the host's IP address.

For example, to broadcast a packet to an entire IPv4 subnet using the private IP address space 192.168.1.0/24, which has the subnet mask 255.255.255.0, the broadcast address is: 192.168.1.0 | 255.255.255.0 = 192.168.1.255. So the IP 192.168.1.255 it will point all hosts.

2.3. Security protocols

According to [CER]:

> Computer security is the process of preventing and detecting unauthorized use of your computer. Prevention measures help you to stop unauthorized users (also known as "intruders") from accessing any part of your computer system. Detection helps you to determine whether or not someone attempted to break into your system, if they were successful, and what they may have done.

Imagine user Alice wants to contact the user Bob, but a third user called Eve, accesses to the communication data (see Figure 2.15). It is complicated to prevent Eve from capturing

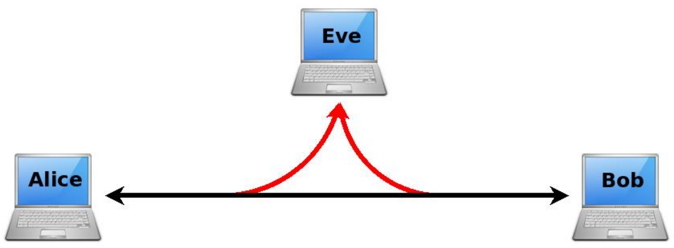


Figure 2.15: Alice and bob diagram.

the traffic, so the solution is to encrypt the data to guarantee authentication, privacy and confidentiality.

To cipher the data there are two different ways [Foua]:

1. Conventional cryptography also known as symmetric cryptography, requires the sender and receiver to share a key: a secret piece of information that may be used to encrypt or decrypt a message. If this key is secret, then nobody other than the sender or receiver may read the message. If Alice and Bob know a secret key, then they may send each other private messages. The task of privately exchanging a key before communicating, however, can be problematic.

2. Public key cryptography also known as asymmetric cryptography, solves the key exchange problem by defining an algorithm which uses two keys, each of which may be used to encrypt a message. If one key is used to encrypt a message then the other must be used to decrypt it. This makes it possible to receive secure messages by simply publishing one key (the public key) and keeping the other secret (the private key).

The problem of the first one is that a secure channel is needed to share the key, but this problem does not exist with the second one. Most of the protocols use the second method to solve the problem of the secured channel, like Secure Sockets Layer (SSL) and Transport Layer Security (TLS).

2.3.1. Secure Sockets Layer

SSL is a cryptographic protocol that provides security for communications over a network. This protocol was originally developed by Netscape and Version 1.0 was never publicly released; version 2.0 was released in February 1995 and SSL version 3.0 was released in 1996.

This protocol uses public key cryptography (asymmetric cryptography) to cipher the channel and get the key to cipher the data. To use a public key and a private key, a certificate is used, but each certificate need the signature of a CA (Certificate Authority).

Figure 2.16 shows the handshake sequence. Notice that depending on the configuration of the server the handshake can change. The handshake is used to share the certificates.

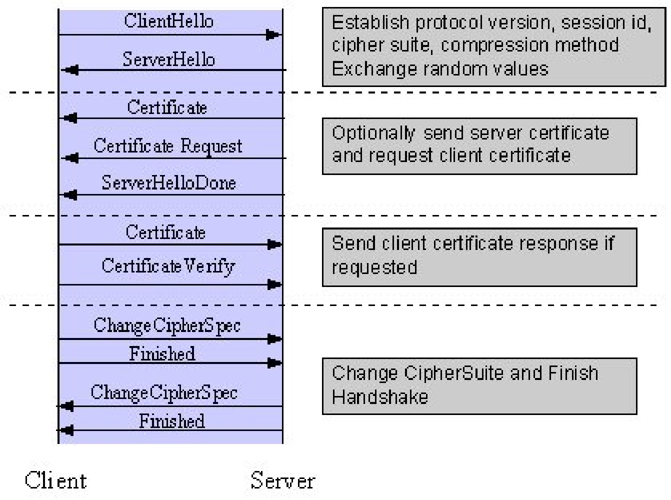

ClientHello
ServerHello
Establish protocol version, session id, cipher suite, compression method Exchange random values
Certificate
Certificate Request
ServerHelloDone
Optionally send server certificate and request client certificate
Certificate
CertificateVerify
Send client certificate response if requested
ChangeCipherSpec
Finished
ChangeCipherSpec
Finished
Change CipherSuite and Finish Handshake
Client
Server

Figure 2.16: Simplified SSL handshake sequence. Source [Foua]

During the handshake they choose the Cipher Suite (supportable by both of them), establish and share a session key, and optionally authenticate the server to the client and/or authenticate the client to the server.

The cipher suite is defined by the following components[Foua]:

- **Key Exchange Method:** defines how the shared secret symmetric cryptography key used for application data transfer will be agreed upon by client and server. SSL 2.0 uses RSA[RRA] key exchange only, while SSL 3.0 supports a choice of key exchange algorithms including the RSA key exchange when certificates are used, and Diffie-Hellman[AJMV97] key exchange for exchanging keys without certificates and without prior communication between client and server.
- **Cipher for Data Transfer:** SSL uses the conventional cryptography algorithm (symmetric cryptography) described earlier for encrypting messages in a session. There are nine choices, including the choice to perform no encryption:
  - No encryption
  - Stream Ciphers: RC4[PP] with 40-bit keys and RC4 with 128-bit keys
  - CBC[7] Block Ciphers: RC2 with 40 bit key, DES with 40 bit key, DES with 56 bit key, Triple-DES with 168 bit key, Idea (128 bit key) and Fortezza (96 bit key)
- **Message Digest:** the choice of digest function determines how a digest is created from a record unit. SSL supports the following: No digest (Null choice), MD5 (a 128-bit hash) and Secure Hash Algorithm (SHA-1) (a 160-bit hash)

  The message digest is used to create a Message Authentication Code (MAC) which is encrypted with the message to provide integrity and to prevent against replay attacks

[7]CBC refers to Cipher Block Chaining, which means that a portion of the previously encrypted cipher text is used in the encryption of the current block. “DES” refers to the Data Encryption Standard [Sch96], which has a number of variants (including DES40 and 3DES_EDE). “Idea” is one of the best and cryptographically strongest available algorithms, and “RC2” is a proprietary algorithm from RSA DSI [Sch96]

2.3.2. Transport Layer Security

TLS is the successor of SSL and is an IETF standards track protocol, last updated in RFC 5246 (see [Die08]). The differences between SSL and TLS are not dramatic, but they are significant enough that TLS 1.0 (RFC 2246 [Die99]) and SSL 3.0 do not interoperate.

Version 1.1 (RFC 4346 [Die06]) added protection against Cipher block chaining (CBC) [Wikb] attacks and support for IANA registration of parameters.

The version 1.2 use SHA-256 instate of MD5/SHA-1, enhancement to the client's and server's ability to specify which hash and signature algorithms will be accepted and Advanced Encryption Standard (AES)[Wika] CipherSuites were added (more changes in [Wikc] and [Die08]).

2.4. In-door localization

Due to the development of technologies, it is now possible to use a cell-phone to know the whereabouts of the user. A wide-spread technology doing that is Global System Positionning (GPS) nevertheless it is not well-recommended for in-door positionning due to the high-level of obstacles in buildings [STK05, page24-40].

So, we have to focus to find solutions for in-door positioning and there are two ways to calculate a cell-phone's position:

- build up a signaling system and a network infrastructure to measure the cell-phone's position
- use an existing wireless network to work out the location of the cell-phone

The former is well-recommended if the developpers want to control physical specification and the quality of the location results by adjusting the density of the sensor or designing a very small wearable tag. The latter will be used if the developpers want to avoid expensive and time-consuming infrastructure's deployment [LDBL07, 1067-1080].

2.4.1. Wireless technologies network

Currently, two main technologies exist as wireless network infrastructure:

- Wireless Local Area Network (WLAN)
- Bluetooth

Wireless Local Area Network

WLAN, also called WiFi (Wireless Fidelity), which protocol name is IEEE 802.11, is a technology using the air-environment to send and receive data between members of the same network. It operates in the 2.4 GHz ISM (Industrial, Scientific and Medical) band and owns a range of 50-100 meters with a bit rate of 11, 54 or 108 Mbps (Mega bites per second) [LDBL07, 1067-1080].

Bluetooth

Bluetooth, which protocol name is IEEE 802.15, is a low cost, low power and short range radio technology. It operates at the 2.4 GHz ISM band and owns a range of 5-10 meters with a maximum bit rate of 1 Mbps [Gha02].

2.4.2. Localization methods

Now that the two main infrastructures are known, the methods to locate a cell-phone will be discussed.

According to [LDBL07, 1067-1080], the localization methods could be split up in 2 categories:

- Triangulation
- Fingerprinting

The former uses the geometric properties of triangles to estimate the position of the cell phone. The latter estimates the cell-phone position by comparing the measurements of the mobile, from the target to a base station, with a database storing previous fingerprints.

2.4.2.1. Triangulation

In triangulation, two ways to calculate the target location can be distinguished, one using the distance between the target and the reference station and one using the angle between the target and the reference station. The following list shows some of these methods, the first three measure the location via the time and the last one via the angle [LDBL07, 1067-1080].

- Time of Arrival (ToA)
- Round Trip of Flight (RToF)
- Received Signal Phase Method (RSPM)
- Angle of Arrival (AoA)

Most of these methods use time to compute the location causing a problem: clock synchronization.

Clock Synchronization

Often in distributed systems, synchronised algorithms are used which work in "rounds", it means that at the same moment, nodes do some calculation and after this period, exchange theirs results at the same time. Due to that, it requires a perfect clock synchronization, if the clocks are not the same, some nodes will receive data later or earlier and disturb the running of the alogrithm thus the results [Sim90, 84-96].

Due to design, it is very hard to have two or more hardwares sharing the same clock. Moreover, during running time, some drifts appears causing a lost of accuracy. So in distributed systems is not possible to have clock synchronization whereas you can use a relative clock which is checked and re-calibrated if there are drifts. [KM10]

Time of Arrival

Time of Arrival determines the time when the cell-phone sends a message and when the base station receives it. After, the distance is calculated using the following relationship:

$$d_i = (t_i - t_0) \times c \tag{2.1}$$

where d_i represents the distance between the base station and the cell-phone i, t_0 when the cell-phone i sent the message and t_i when the base station received it, c represents the speed of light ($3.10^8 \mathrm{m/s}$). The distance is the radius of the circle where the cell-phone can be. Thus, 2 more base stations are required to locate the node. The figure 2.17 shows that, each base station draws a circle where the cell-phone may be. The intersection of these 3 circles is the mobile's location. So this method requires at least 3 based stations to have an optimal location of the cell-phone. Moreover, to have coherent measurement and accuracy, the base stations and the mobile have to be synchronized [STK05, 24-40].

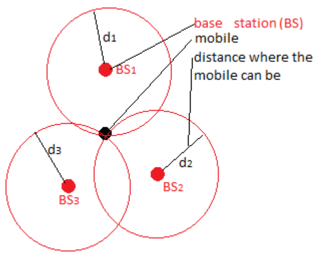


Figure 2.17: Example of Time of Arrival Location

Round Trip of Flight

Round Trip of Flight measures the time-of-flight, called round trip time (RTT), to and from the cell-phone by sending a data packet to the base station and receiving an immediate acknowledgement from the same base station. The equation 2.2 calculates the rtt.

$$rtt = t_{local}^{in} - t_{local}^{out} \tag{2.2}$$

The equation 2.2 is used several times to calculate a mean of the rtt (mrtt) and used to calculate the distance between the cell-phone and the base station according to the equation 2.3

$$d = \frac{(mrtt) \times c}{2} \tag{2.3}$$

t_{local}^{out} represents the time when the data leaves the cell-phone, t_{local}^{in} the time when the acknowledgement arrives, *d* the distance, *c* the speed of light.

RToF requires more relative synchronization between the cell-phone and the based station than ToA. Moreover, it is difficult to know the exact delay time caused by the responder [GH05, page768-779].

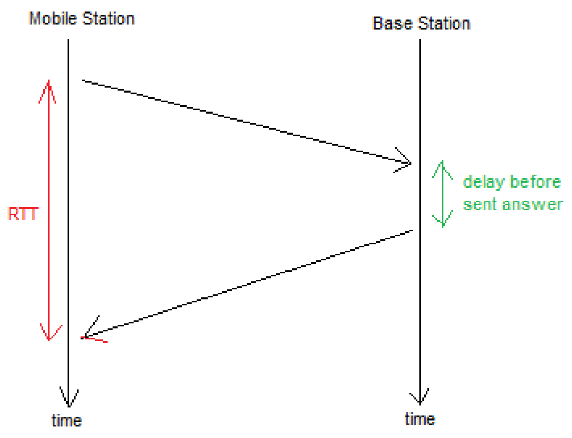


Figure 2.18: Example of Round Trip of Flight

Received Signal Phase Method

Received Signal Phase Method uses the phase difference to estimate the range of the cell-phone. All base stations emit pure sinusoidal signals of the same frequency. When the delay between the 2 signals is known, the distance could be computing using the following formula:

$$d = (c \times \phi)/(2 \times \pi f) \tag{2.4}$$

where *d* represents the distance, *c* the velocity of light, ϕ the delay in degree and *f* the frequency of the signal. This method requires a pure sinusoidal signal of the same frequency with 0 phase offset and clear LoS [LDBL07, 1067-1080].

Angle of Arrival

Angle of Arrival uses the calculation of different angles to determinate the location of the cell-phone. The angles can be calculated by measuring the phase difference or the power spectral density. When the angles are calculated, the distance between the cell-phone and each base station can be calculated using the trigonometric relationships. AoA requires at least 2 base stations and these base stations have to be antenna array [STK05, 24-40]. Figure 2.19 describes this idea, the angles a_1 and a_2 can determine the distance between BS_1 and BS_2 and the position of the target.

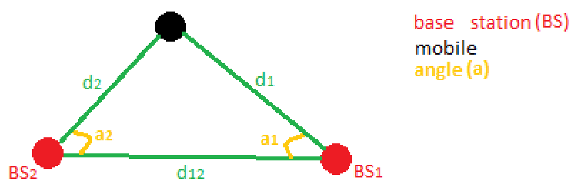


Figure 2.19: Example of Angle of Arrival Location

2.4.2.2. Fingerprinting

Fingerprinting localization requires 2 stages.

1. offline stage
2. online stage

During the offline stage, the signal strength from nearby base stations is collected and stored in a database. During the online stage, the current signal strength is measured and used to estimate the cell-phone location using algorithms. There is a lot of way to do that, two of them are explained in this subsection [LDBL07, 1067-1080].

- k-Nearest-Neighbor (kNN)
- RedPin

k-Nearest-Neighbor

k-Nearest-Neighbor is a method used to classify elements. A current element is classified by matching its k neighbors, using a metric, from a training set. Figure 2.20 illustrates this idea. We have three classes $(c1, c2, c3)$ of elements from a training database and one unknown element (u). We choose $k = 5$ and the Euclidian distance as metric. We can see that four neighbors of the five neighbors required are from the class $c1$. So u is assigned to the class c1 [GO02].

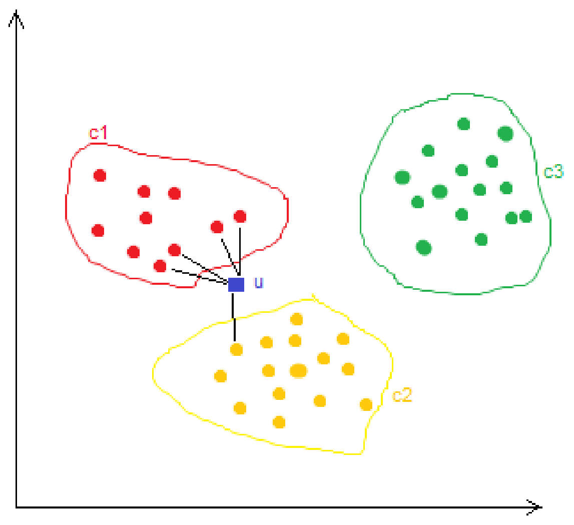


Figure 2.20: Example of kNN classification

For instance, the RSS (Received Signal Strength) could be used as metric. The values of the RSS could be stored in a database combined with its matching room. Once a cell-phone wants to be located, it compares its current RSS with all the database. The 3, for example, most closest RSS of the database to the current RSS are selected. The room which has the maximum of instance determines the location of the cell-phone [LDBL07, 1067-1080].

RedPin

RedPin is a cooperative localization system for cell-phone developped by Philipp Bolliger [Bol08, 55-60]. RedPin is based on symbolic identifier, it means that a location is not given by the geographic coordinates of the cell-phone. For example, it gives the label of a room when the user requests for its localization. Moreover, RedPin is said as a cooperative system, when an user enters into a room, the system looks to find the location of the cell-phone and if it is not finding, it offers the possibility to the user to enter the label of the room he is and the matching measurements, severals measurements for one location are allowed. RedPin consists of two basic components:

- a Sniffer
- a Locator

The *Sniffer* collects and gathers information about the different wireless devices to create a fingerprint. In fact, RedPin measures the RSS of the active GSM cell, the RSS of all the WiFi access points and hotspots.

The *Locator* stores the fingerprint with its matching room and contains the algorithm to locate a cell-phone. It is located on a server that offers several services for the cell-phone that access to this server. For instance, there is one service that allows the cell-phone to store fingerprints into a database. Another service can retreive in the database the fingerprint, so the location, of the cell-phone that best matches with the measurements taken by the cell-phone. To compare different measurements, a distance measure had been defined. When two identifiers, for example the BSSID of wifi access points, occur in both of the measurements that are comparing, this matching is added to the distance measure. Moreover for each pair of matching, a contribution, based on the measured RSS, is added to the distance. Finally, to return the location, the *Locator* takes the distance which is smaller than the threshold.

2.5. Chat Systems

One of the main features of the application is the possibility to chat with other people from the university. Many kinds of protocols exist such as XMPP, IRC, MSNP or Global Index P2P.

2.5.1. Global Index P2P

Global Index P2P (GI P2P) is the protocol used by Skype. This method is a mix between peer-to-peer and Voice Over Internet Protocol (VoIP). Although it allows to communicate directly without a server and thus to exchange information faster, Skype uses one to identify the users. Another advantage of this technique is that it has resources sharing so that users with a faster connection can unload part of the traffic from users with a slower one. As this is a proprietary protocol we can not use this one but we could use something based on the same idea probably also using a server for users to log in.

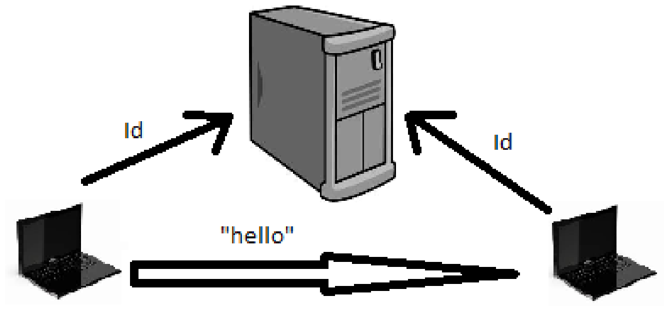


Figure 2.21: Example of Global Index P2P

Figure 2.21 shows that two clients need to log in a server first and then communicate directly together without using that server. The system would be the same with more than two users.

2.5.2. MSNP

Microsoft Notification Protocol is a protocol developed by Microsoft and used by Windows Live Messenger or Pidgin over TCP. Also being a proprietary protocol little is known about it. A main server receives every single message and then forwards it to the right client, which allows to filter and to moderate the contents of the messages.[ZP07]

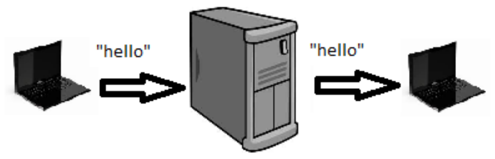


Figure 2.22: Example of Microsoft Notification Protocol

Figure 2.22 shows how two users need to go through a main server to exchange a message.

2.5.3. IRC

Internet Relay Chat is one of the most public common communication protocols using TCP. It uses a network (or tree) of servers to send text messages from a client to another. To login a user connects to any server and can communicate with someone else via a channel i.e. a list of connected users. One of the major drawbacks of that protocol is that a misbehaving server can prevent messages to go from one user to another. Another problem is that the characters being coded on only 8 bits two clients using different languages can not communicate together. However this protocol has one strength compared to many others, it is that a message travels through the network once and only once. [SoIT00, pages 62-65]

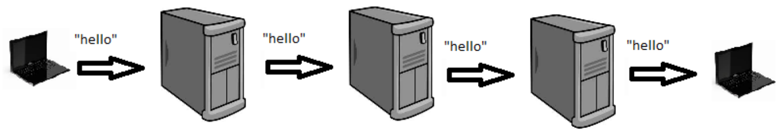


Figure 2.23: Example of Internet Relay Chat

Figure 2.23 shows how a message from a user to another has to go through a branch of the server tree.

2.5.4. XMPP

Extensible Messaging and Presence Protocol is a client server based messaging protocol using XML. The server can either be private or linked together inside the Jabber network, the main XMPP network. This protocol works similarly to the e-mail exchange, every user has a Jabber ID composed of the username and the domain. Two users communicate with each other directly through their respective servers. XMPP has many advantages. First, it is an open standard so it can be used for free. Then the message can be encrypted offering a good protection. Moreover it is very flexible thanks to the XMPP Extension Protocols (XEP) allowing to develop a client with many functionalities. Finally, as it is one of the most used protocols, it is easy to find support and help.[SAST09]

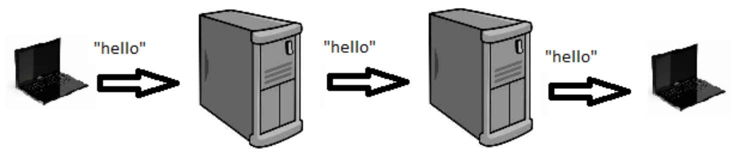


Figure 2.24: Example of Extensible Messaging and Presence Protocol

Figure 2.24 shows that a message from a client go through its server then through the receiver's server and finally arrive at the receiver.

2.5.5. Link-Local XMPP

The extension for XMPP XEP-0174, also known as Link-Local XMPP is a messaging protocol that defines how to establish an XMPP-like chat over a local or a wide-area network using the principles of zero-configuration and no server. It discovers clients using a mDNS-based service. Any entities can then start a serverless conversation using XML streams.

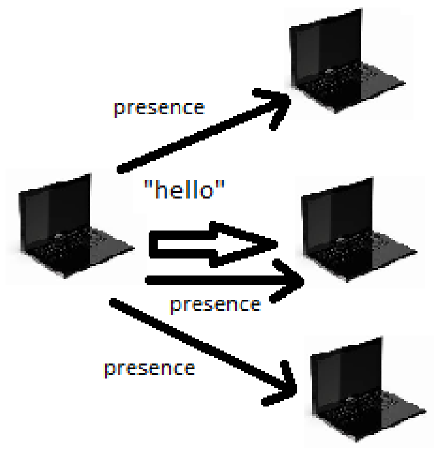


Figure 2.25: Example of Link-Local XMPP

Figure 2.25 shows that a message from a client goes straight to the receiver without any server.

2.5.5.1. Overview

The XEP-0174 [Foub] defines a series of actions that must be taken in order to do an XMPP-like client-to-client chat where the user can be seen as both the client and its own server.
First the client has to show it is connected. In order to do that it publishes a Domain Name System (DNS) record of type presence (_presence._tcp.local.) to a multicast DNS (mDNS) address giving the IP address at which the machine will listen, a record mapping the client to its machine via the desired port and a pointer saying that the service communicates over TCP or UDP. The client can also publish a DNS TXT record giving some optional information such as: name, status, whereabouts and so on.
The mDNS protocol is part of the Zero Configuration Networking (Zeroconf) allowing to create automatically an IP network without specific configuration or server. From developpers point of view it works in a very similar way than DNS allowing them to do Zeroconf without having to learn a new protocol.
Then the user can start looking for other hosts thanks to a DNS-based Service Discovery (DNS-SD). To do so the chat client asks for a pointer of type presence, querying for its mapping to the IP address they are listening to. In the same time the client finds out the TXT record containing additional information.
Afterwards a chat can be initiated by opening an XML stream to the IP address and port previously discovered. Messages can then be exchanged until both clients close the stream.
Finally when the user wants to disconnect, the client sends a ”Goodbye” mDNS packet so everyone else can stop querying about his or her presence.

2.5.6. Publishing DNS Records

As previously said when a client wants to advertise for its presence it must publish four mDNS records:

- A Pointer record (PTR) that should look like
  _presence._tcp.local. PTR user@machine._presence._tcp.local
- The address record where the IP address can be either an IPv4 or an IPv6
  machine.local. A ip-address
- The SRV record
  user@machine._presence._tcp.local <tt$l$ > SRV <priority> <weight> port-number machine.local.
- TXT record with the optional information written as a series of key-value pairs or a single zero byte if the user doesn’t want to publish anything.

For instance, if university_droid on the machine android-device wants to connect via the port 5562 with the IPv4 adress 10.2.1.187 the following report will be published :

```
_presence._tcp.local. PTR university_droid@ android-device._presence._tcp.local.
university_droid@ android-device._presence._tcp.local. SRV 5562 myPC.local.
myPC.local. A 10.2.1.187
```

The TXT record will look like something like that :

```
university_droid@ android-device._presence._tcp.local. IN TXT
”status=avail”
”1st=universitydroid”
```

Once someone connects to the hotspot the chat client sends a mDNS querie for services of type presence. To do so it first ask for a PTR record that match the ”_presence._tcp.local.” then for each service it querries the SRV record, mapping them to the A record before finding the optional information in the TXT record. After that the user can start an XML stream with anyone founded.[SCa]
To open an XML stream the initiator has to open a TCP connection at the IP adress and port previously discovered. If university_droid discovered a service named Alice@myComputer and wants to initiate the chat, his client has to send :

```
¡stream:stream
xmlns=’jabber:client’
xmlns:stream=’http://etherx.jabber.org/streams’
from=’university_droid@ android-device’
to=’Alice@myComputer’
version=’1.0’¿
```

The version flag indicates that the client supports stream features of XMPP 1.0.

Afterwards the other clients answer in the same way :

```
¡stream:stream
xmlns=’jabber:client’
xmlns:stream=’http://etherx.jabber.org/streams’
from=’Alice@myComputer’
to=’university_droid@ android-device’
version=’1.0’¿
```

At that point the XML stream is set and is ready to be used to send messages. To do so the XMPP message needs to be written like that :

```
¡message from=’university_droid@ android-device’ to=’Alice@myCoputer’¿
¡body¿Hello.¡/body¿
¡/message¿
```

Once one of the users wants to stop chatting the client has to close the stream:

```
¡/stream:stream¿
```

The other client then answers with the same message.

Finally to stop beeing seen as present the user sends a PTR goodbye record so the other clients to remove it from their presence list.

CHAPTER 3. DESIGN

Once a preliminary study of the different existing possibilities of designing the system is made, the most suitable options to provide the expected functionalities can be now chosen, taking also into account the resources that are available.

Therefore, the aim of this chapter is to explain what design decisions are going to be made and the reasons which lead to these decisions.

Moreover, a design model for the system will be also shown in this chapter. The design model will later guide the implementation of the system and it will be based on those design decisions, which are going to be described firstly in the subsequent section.

3.1. Design Decisions

As said before, at this point it is necessary to make design decisions based on the preliminary study described in the previous chapter. The way those decisions are going to be dealt with in this chapter, is following the order of the concepts seen in the previous chapter, also making design decisions for some functionalities which were not described explicitly in the preliminary study, but which can be derived from the concepts studied.

3.1.1. Mobile Operating System

According to Figure 2.1, the most suitable option is Symbian, because is the most widely used OS, but notice that most of the smartphones with Symbian use an old OS version so the real amount of smartphones that use the new Symbian OS version is much lower (there are only four smartphones available with Symbian 3 [Sym10]).

Android OS uses Java as the main developing language, so that the learning curve is small therefore the development of an Android app. is easier. On the other hand, to use all the features that Android and Symbian have, it is possible to use Windows, Mac or Linux, but with iOS, to get all the features Mac is needed.

Android and Symbian OS are open source but iOS is not open source.

For these reasons, Android OS has been chosen as the target operating system.

3.1.2. Distributed System Architecture

The mobile devices will be connected to a wifi network, so that it is possible to take advantage of the wifi network features

It is important to know that the network will not be very big (less than 200 users), so that a lot of decisions has been taken according to it.

This section describes the main process for use at the peer-to-peer network: Join, get an UI (unique Identifier), remove node and maintain the routing table.

3.1.2.1. Join node

If a node want to join the network, it has to do two steps: Logs in the central sever using its credentials and publishes its presence using MDNS.

When a peer logs in the server, it gets all the valid users from the server to checks it when a new user join the network.

To join the network, the peer send its presence using MDNS (see section 2.5.5). As the peer has to be authenticated using a server, the other peers will check the identity of the new host, if the user is in the list of existing user the others peers will add the new user to the chat list, if not the users will report the issue to the central server, just to know if this is a new user created moments ago, or if it is a malicious user. If it is a new user, the server will sent again the list with all the existing users, if not, the administrator will be notify.

After the user has been added, the other peers will try to authenticate the user (see section 3.1.3.) to be sure that this user is who it say.

3.1.2.2. Unique Identifier

To get the UI (Unique Identifier) in the WLAN (Wireless Local Area Network) the peer only needs to get the MAC (Media Access Control address)[IEE01] address and the user name of the client.

3.1.2.3. Remove node

All the nodes in the network send keep-alive messages to notify the other hosts that they are still connected. This TTL of the keep alive messages it should be short, because of the dynamically of the users. By default the TTL of the MDNS is an hour, so for the users of this application it should be approximately 5 minutes.

The MDNS also implement the “good bye” messages, so if a node want to leave the network, it only has to send the “good bye” message.

3.1.2.4. Routing Table

The MDNS provide a list of hosts have published their services, so it’s is possible to use this routing table. The problem is that all users (authenticated or not) can be registered in this list. To avoid it, the application combines the MDNS list with another list of valid users downloaded from the server.

So the system uses two list (one for the hosts in the network and one for the valid users) and combines both to offer the the authentication service. Basically the routing table contains: IP, host name, port and UID.

3.1.3. Security

To reach the security requirements the following features are needed: encrypted and authenticated communication.

Communication between all the peers should be encrypted. The protocol chosen for this purpose is TLS. So that a know central server is needed to authenticate the user's certificate. The problem of this protocol is that does not provide security for broadcast messages. Figure 3.1 shows the process:

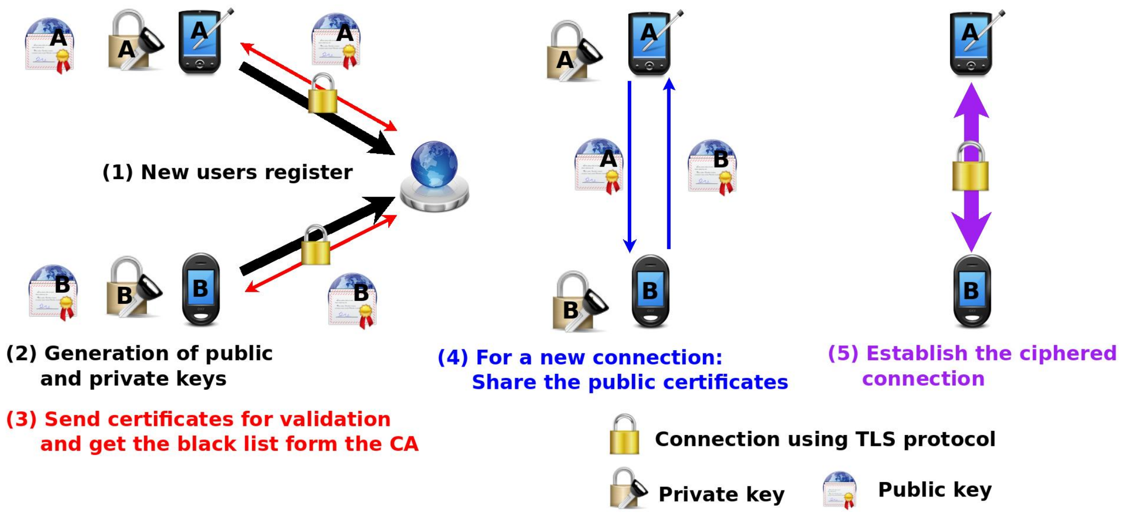

(1) New users register
(2) Generation of public and private keys
(3) Send certificates for validation and get the black list form the CA
(4) For a new connection: Share the public certificates
(5) Establish the ciphered connection
Connection using TLS protocol
Private key
Public key
A
B

Figure 3.1: TLS connection example.

1. When a new user is created, it has to build its own private and a public key and send the public key to the CA (Certificate Authority) to sign it. Then the CA send to the new user the certificate and the black list certificates. The user trust the CA because the public key of the CA is stored in the application.

2. If two users want to establish a communication they send to each other the certificates and verify this certificates if it is signed by the CA and with the black list sent by CA.

3. If the certificates are correct, then the users establish a new ciphered connection.

4. The users need an updated black list[1], so the clients has to check the server black list periodically.

Using TLS system guarantees authentication and communications confidentiality, over the LAN using cryptography.

[1]Baned users.

To guarantee that the routing table received is the correct one, the new user will check the certificate (checking the CA of the certificate and checking the black list if the user is already log in), and if it is valid the routing table is accepted, if not, the new user has to send a broadcast message to report the issue with the information of the malicious user (IP, MAC address and user ID) and the other peers will send again the routing table and they do not take in to account the messages sent by the malicious user. But to add an user to the black list, the server has to receive more than one report from different users and the administrator has to check the account to avoid the malicious users to add other user to the black list.

If this malicious user has already been log in, the system will notify the administrator, and the administrator has to take the decision about the user. If the administrator block the user, the account and the certificate will be added to the black list.

3.1.3.1. Security and Permissions in Android

Android is a multi-process system in which each application runs in its own process. Most security between applications and the system is enforced at the process level through standard Linux facilities (kernel of the main Android core). [web10d]

Each Android process runs into what is called a *sandbox*, which provides a safe runtime environment. By default, no application is allowed to perform any operations or behave in a way which can negatively affect other applications, the user or the system itself. The only way to skip these restrictions enforced by the Android environment, is through the explicit statement of a permit authorizing to carry out a certain action which is usually forbidden.

Furthermore, in Android it is mandatory that each application is digitally signed by a certificate whose private key is that of the application developer. There is no need to link to a certificate authority, the only role of this certificate is to create a trust relationship between applications. By signing, the application has attached its authorship.

As the decision of developing an application for an Android environment was made, at implementation phase it would be necessary to take into account those security issues whose overview has just been provided, and which was taken from [web10d].

3.1.4. Course Information Access

To get the data of the RSS, Schedule, information of courses and the FAQ, there are two options:

1. Directly downloading from a central server
2. Get the information from another peer using peer-to-peer

Table 3.1 shows a comparison between the previous mentioned options.

	Directly from the server	Peer-to-peer
Get information	to get the information, the client has to connect to a central server, and request the xml files, if the server is down, the client can not access to this information, but if there are any peer with this information or there are not any peer, the client can get the information.	If a peer is down, the peer can get the xml files from the other peers, as long as the users were able to contact with other peers
Load Balancing	It is possible to scale the server with other kind of architectures, like clusters, proxies, etc.	if the numbers of users increase, the peer-to-peer architecture provide to the system a scalable network
Network dynamics (addition/deletion of objects/nodes)	To add a node, the system administrator only have to upload the file in to the server, and all the peers will able to access the file. In this case, the new peers does not affect the availability of the file	If the administrator want to upload a file to the network has to add the file in one peer, and wait to the other peers if they want to download the file. If one peer left the network and is the last one who has the file, all the other peers can not access to this file until a peer with the file join the network.
Fault Tolerance	If the server is down, nobody can download the file	If more than one peer has has the file and one peer left the network, the file is still available, but if only one peer has the file and left the network, the file is unavailable

Table 3.1: Distinctions between get the files from a server or from the peer-to-peer network

The existing infrastructure has been considered to design the course information data access, so that the project will get all the information from the existing sources (web server of the university).

3.1.5. In-door Localization

Two ways for indoor-positionning have been seen in the section 2.4 : build a signaling system plus a network infrastructure or use an existing wireless network. In this report, an existing wireless network is used due to the lack of time to build a signaling system and a network infrastructure. In particular, the wireless network of the computer science departement is used.

As the computer science departement is not equipped with hotspots, the WLAN technology is used instead of the Bluetooth technology.

The following table shows an overview of the techniques seen in section 2.4.

Techniques	Accuracy	Specific requirements
Time of Arrival	Around a meter	Accurate clock synchronisation Needs at least 3 base stations to find the location
Round Trip of Flight	Around a meter	Relative clock synchronisation Time delay Needs at least 3 base stations
Received Signal Phase Method	Around a meter	Requires pure sinusoidal signal with 0 phase offset
Angle of Arrival	Around a meter	Antenna array Requires at least 2 base stations
k-Nearest-Neighbors	Around a room	Training phase of build the database At least one base station
RedPin	Around a room	No training phase to build the database At least one base station

As shown by the previous table, triangulation methods offer an accurate location but have precise requirements, such as clock synchronization or the number of base stations that are required to compute the distance. So a fingerprint method is used as if it offers a less accurate location. The name of the room is enough in the case of this report to locate a cell-phone due to the fact it is not required to have a meter accuracy. The RedPin method is used instead of a kNN method because it is more concreate and more information are given about this method.

To sum up, the indoor-localization used the following features:

Network architecture	Wireless technologies	Localization methods
Existing network (Computer Science WiFi network)	WLAN	RedPin

Adaption of RedPin

An adaptation of RedPin had been done for this report for severals reasons. First of all, the application have to provide a help to the user if he is lost so the indoor-localization feature is not cooperative. In addition, some malicious users can enter a false room for a set of measurements into the database thus false the result of the location algorithm. As hotpots are not available in the Computer Science department, these measurements to find a location are not taking into account. RedPin uses also the RSS from the active GSM cell but no GSM signals were received by devices that were used in application, this signal is not taking into account.

To sum up, a training phase to collect the necessary information of the access points is required due to the indoor-localization feature is not cooperative and the measurements used to find a location come from the WiFi access points. The indoor-localization processing could be divided into 4 systems:

- a system to collect the WiFi information of a location
- a system to store the WiFi information of all the locations
- a system to find the location when requesting, accessing to the WiFi information storage
- a system to ask for its location with its current measurements

Collect WiFi information

This system needs to collect all the nearby access points information, namely the BSSID of the access point and the RSS received by the cell-phone from a BSSID, of a room. Figure 3.2 shows this assumption, *0d:ff:02:c4:e2:23* is the BSSID of access point 1 and -*82* is the RSS of access point 1.

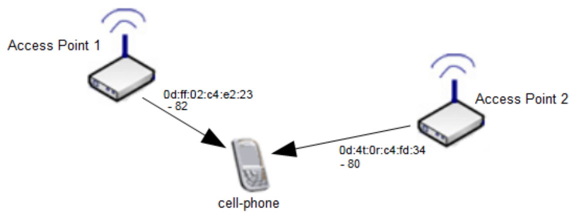


Figure 3.2: Example of collecting access points information with the cell-phone

Store WiFi information

The information collecting by the cell-phone needs to be store. A database is used to that and the stored information are:

- the name of the room
- the BSSID of one access point
- the RSS of the matching access point

The following table shows that. Each row matches to a set of the name of the room, the bssid of one access point and its matching RSS.

Room	BSSID	RSSID
room1	0d:ff:02:c4:e2:23	- 82
room1	0d:4t:0r:c4:fd:34	- 80

More information about the database is given in the database design section.

Find location

A location is found by comparing measurements of the cell-phone and the database and take the one that has the best matching. To do that, this system needs to access to the database and receive the measurements take by the cell-phone. A server is lauched to wait the connection from a cell-phone whose wants to know its location. First of all, when the server accepts the connection form the cell-phone, it receives all the measurements of the cell-phone, then access to the database to compare. When the comparing processing is done, it returns the requested location to the cell-phone.

To compare the current measurement and the database, a distance measurement is computing. This distance is based on comparing each BSSID of the table for each BSSID sends by the cell-phone. When two BSSID match, it is added to the distance. In addition, a contribution signal is computed, consisting of the difference between the two RSS, and added to the distance. When all the BSSID of one room of the database had been compared, the distance measurement is added to this room. When all the rooms have a distance, the one that has the maximum distance, so the best match, is set as the room of the measurements send by the cell-phone and return to the cell-phone. Figure 3.3 shows this processing.

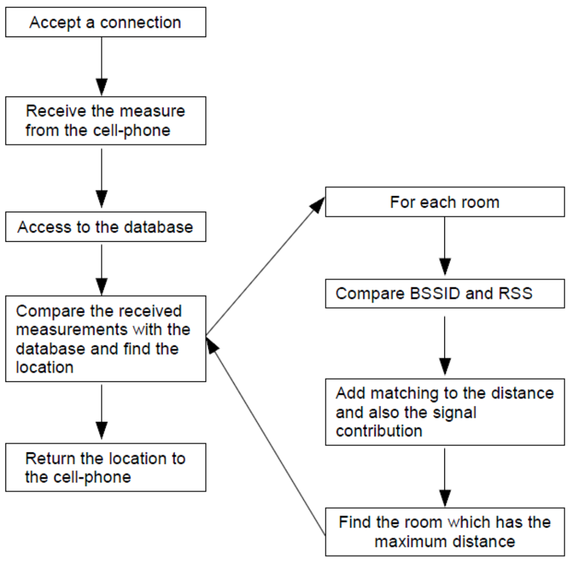

Accept a connection
Receive the measure from the cell-phone
Access to the database
Compare the received measurements with the database and find the location
Return the location to the cell-phone
For each room
Compare BSSID and RSS
Add matching to the distance and also the signal contribution
Find the room which has the maximum distance

Figure 3.3: Diagram of the processing of the server

Ask for the location

This system collects the information, BSSID and RSS, of its nearby access point, send it to the location system and wait for the result. When it has the result, it shows the room on a map.

3.1.6. Chat System

Due to the specification of the project not all the protocols can be used. In order not to pay royalties the protocol needs to be free. Then to ease its developement it would be nice to have a protocol many people use with a large comunity of developpers. Even it is not necessary, it would a plus if the client could be flexible and secured. Moreover, as many international students come in the computer science department the protocol needs to be able to manage a large number of characters for non latin alphabets. Finally, it is obvious that it needs to work on Android.

GI P2P and MSNP are not open so their is not a large community of developpers and are not flexible, plus MSNP does not allow encription. Thus neither of them can be used for our project.

So only IRC and XMPP (and the XEPs) can be used as they are both open and so has many developers. However, only XMPP is flexible allows encryption and multi-bytes coded characters. The following table sum up the diferent chat protocols as well as their properties.

	GI P2P	MSNP	IRC	XMPP	XEP-0174
Open			X	X	X
Many developers			X	X	X
Flexible				X	X
Encryption	X			X	X
Multi-Bytes Characters	X	X		X	X
Available on Android	X	X	X	X	X
Peer to Peer					X

According to this table it is obvious the good choice is using XMPP or its extension XEP-0174.

3.2. Design Model

The design model is based on the analysis model, and can be considered to be just a refinement and elaboration thereof. [AN02, page 251]

The design model deals with the structures and behaviour described by the analysis model in depth, in order to approach them to the real implementation of the system. In other words, design can be defined as the process that models the solution domain.

3.2.1. Aims of the Design

The objectives intended to be met at the design phase, as described in [AN02, pages 249–256], are the following:

- To understand the issues relating to non-functional requirements and restrictions relating aspects such as programming languages, reusable components, operating systems and distribution and concurrency technologies.
- To create a starting point for subsequent implementation activities.
- To be able to decompose the implementation work into manageable parts that can be carried out by different persons, taking into account possible concurrency.
- To design simple and usable interfaces. This point is an important one, since a simple interface will allow a better control of the application by the different users.

3.2.2. System Architecture

> Architecture describes the strategic aspects of how the system is broken down into components, and how those components interact and are deployed on hardware. [AN02, page 29]

To do this, the existing architectural patterns[2] will provide the base. Identification and application of architectural patterns make it easier to develop the system and to obtain a more reusable design.

There are two characteristics of the system that will guide its architecture: the fact that the system would be a distributed one, and that the application would allow user interaction (that is, it would be an interactive system).

Focusing on the user interaction feature, the application of the *Model-Controller-View* pattern (MVC) will be useful. This pattern divides an interactive application into three components [BMR$^+$07, pages 125–129]:

- **The Model**, which contains the data that the application need and the main functionalities.
- **The View**, which displays information to users.
- **The Controller**, which handles user input and updates Model or View if changes are made.

[2]"A pattern for software architecture describes a particular recurring design problem that arises in specific design contexts, and presents a well-proven generic scheme for its solution. The solution scheme is specified by describing its constituent components, their responsibilities and relationships, and the ways in which they collaborate". [BMR$^+$07, page 8]

Views and controllers together comprise the user interface. A change-propagation mechanism ensures consistency between the user interface and the model [BMR+07, pages 125–129]. The application of this pattern has several benefits, as shown in [BMR+07, pages 141–143], and one of the most relevant in our system is that it allows multiple views of the same model, i.e, the model is independent of user interface (as a consequence, it allows to port an application to a new platform without affecting its functional core). On the other hand, the controller is responsible for isolating the model and view the details of the protocol used for requests.

The MVC pattern can be combined with the *Layers* pattern, which helps to structure those applications that can be decomposed into groups of subtasks in which each group is at a particular level of abstraction [BMR+07, page 31], as it is, for example, the OSI 7-Layer Model. Three layers can be identified in our system:

- **Presentation layer.** It contains the user interfaces and it is the responsible of collecting the data given by the user. This layer interacts with the business layer to request the information that it is going to show the user or to send the collected data.
- **Business layer.** This layer contains those classes which were identified at the analysis phase, and it is where the logic of the application is handled. On the one hand, as said before, it interacts with the presentation layer to send processed data to display or collect those which have been entered by the user, and on the other hand with the persistence layer for sending or recovering data to/from storage .
- **Persistence layer.** It is responsible for the data storage, interacting with the business layer. The data storage will be handled by a database.

Thus, each layer is weak-coupled with the other two, but its components are cohesive. Components of a layer can only interact with those of their same layer or those of a lower layer. The main benefit derived from this features of the Layers pattern is that it facilitates components being reusable and keeping the effect of changes in the layer that is changed. [BMR+07, pages 48–51]

As it can be seen, there is some correspondence between the components of the MVC pattern and the Layers pattern, as shown in Figure 3.4.

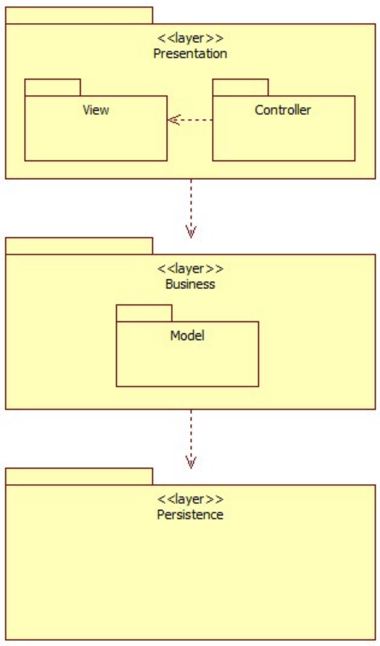


Figure 3.4: Design - MVC and Layers

3.2.2.1. System Topology

In this section, a view of the system deployment will be given. According to [AN02, page 19], a deployment view:

> Models the physical deployment of components onto a set of physical, computational nodes such as computers and peripherals. It allows you to model the distribution of components across the nodes of a distributed system.

The best way to show that system deployment is by means of a deployment diagram (see Figure 3.5). In the diagram, a representation of the different nodes[3] involved in the system is shown. There are two main types of nodes: clients/peers and server. Clients/peers are the smartphones where the Android app will be allocated, and, as can be figured out, sometimes performs the role of a client (when requesting the server information like schedules, for example), and sometimes performs the role of a peer (e.g, when chatting with another user). Server is where services such as RSS feed, database server and Web server are allocated.

[3]"A node is a run-time physical object that represents a computational resource". [RJB99a, page 94]

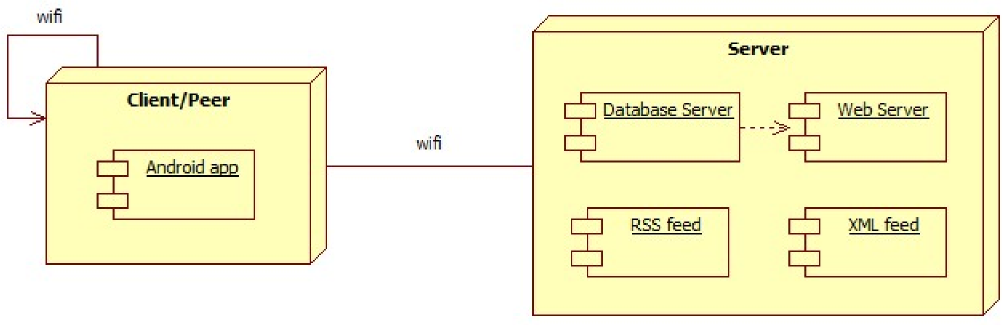


Figure 3.5: Design - Deployment diagram

Actually, there is not an only client/peer, but several of them conforming a peer-to-peer topology. All of the peers, as said before, can behave as clients when interacting with the server (see Figure 3.6).

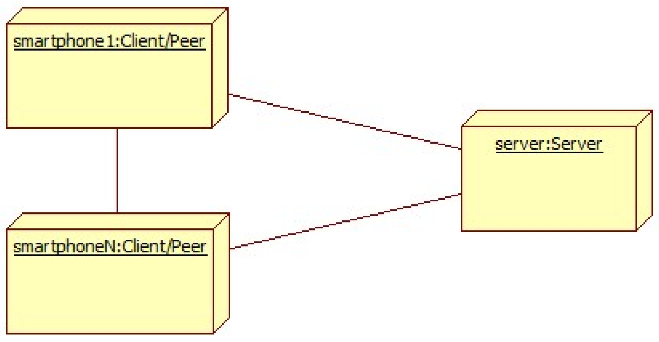


Figure 3.6: Design - Nodes intercommunication

3.2.2.2. Use Case View. Use Case Realizations

An use case realization shows how instances of the identified classes can interact to realize system behavior specified by a use case [AN02, page 99]. This is shown by means of interaction diagrams.

Interaction Diagrams

A list of the use cases whose realization will be shown is provided here, together with references to the figures which provide those sequence diagrams. Only the most relevant sequence diagrams are going to be shown in this chapter: to consult the rest of them, see Appendix B (figures in Appendix B are also referenced in the list). However, some trivial use cases realizations are not included neither in this chapter nor in Appendix B:

Use Case	**Reference**
[UC-0001] Login	Figure B.1
[UC-0002] Logout	-
[UC-0003] Consult news	Figure B.2
[UC-0004] Consult schedule	Figure B.3
[UC-0005] Consult course information	Figure B.4
[UC-0006] Consult map	-
[UC-0007] Consult campus map	-
[UC-0008] Consult department map	Figure B.5
[UC-0009] Consult FAQ	-
[UC-0010] Use chat	Figure 3.7
[UC-0011] Contact students	Figure 3.8

Sequence diagrams for *[UC-0010] Use Chat* and *[UC-0011] Contact students* use cases are shown therefore in this chapter, in Figure 3.7 and Figure 3.8 respectively.

Design Class Diagram

> A design-level class represents the decision to package state information and the operations on it into a discrete unit. It captures the key design decision, the localization of information and functionality to objects. Design-level classes contain both real-world content and computer system content. [RJB99a, page 45]

Starting from the initial class diagram shown in chapter 1 and according to the sequence diagrams presented before, the class diagram at the design phase can be completed as shown in Figure 3.9.

As can be seen in the diagram, the initial class diagram have been extended by adding some new classes:

ChatManager. This class has the responsibility of perform the chat functionality.

DataRequestManager. It handles data requests from the user, either if the user wants to update information and if the user wants to consult it.

TeacherController. This class' responsibility is to allow teachers to contact their students.

PositionManager. It manages to do the calculations needed to get an indoor position and to show it on the department map.

3.3. Database Design

So that the smartphone application can provide all the functionalities described in previous chapters, a data storage system is necessary to maintain and provide part of the data that

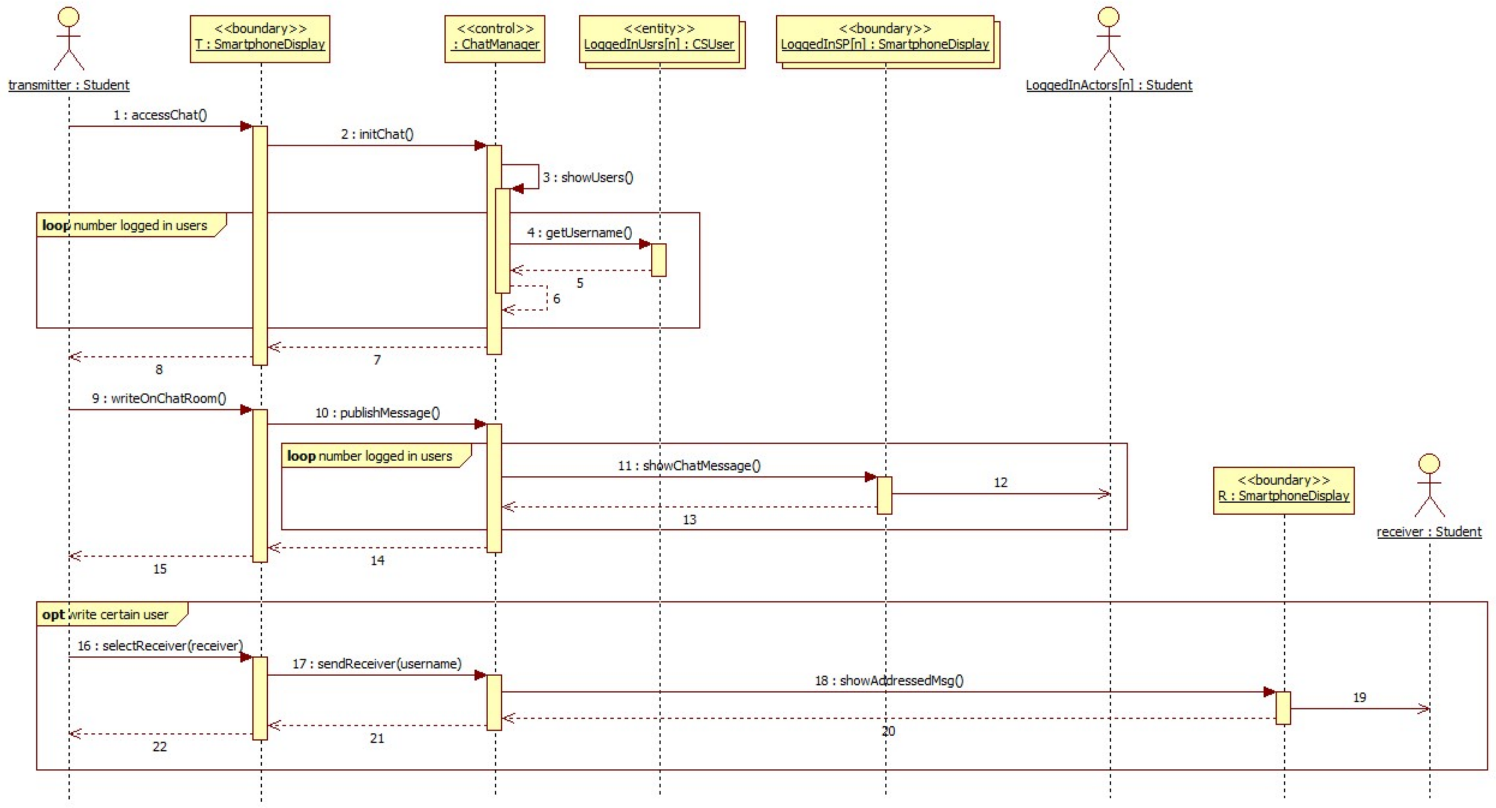

transmitter : Student
<<boundary>> T : SmartphoneDisplay
<<control>> : ChatManager
<<entity>> LoggedInUsrs[n] : CSUser
<<boundary>> LoggedInSP[n] : SmartphoneDisplay
LoggedInActors[n] : Student
<<boundary>> R : SmartphoneDisplay
receiver : Student
1 : accessChat()
2 : initChat()
3 : showUsers()
loop number logged in users
4 : getUsername()
5
6
7
8
9 : writeOnChatRoom()
10 : publishMessage()
loop number logged in users
11 : showChatMessage()
12
13
14
15
opt write certain user
16 : selectReceiver(receiver)
17 : sendReceiver(username)
18 : showAddressedMsg()
19
20
21
22

Figure 3.7: Design - UC-0010 sequence diagram

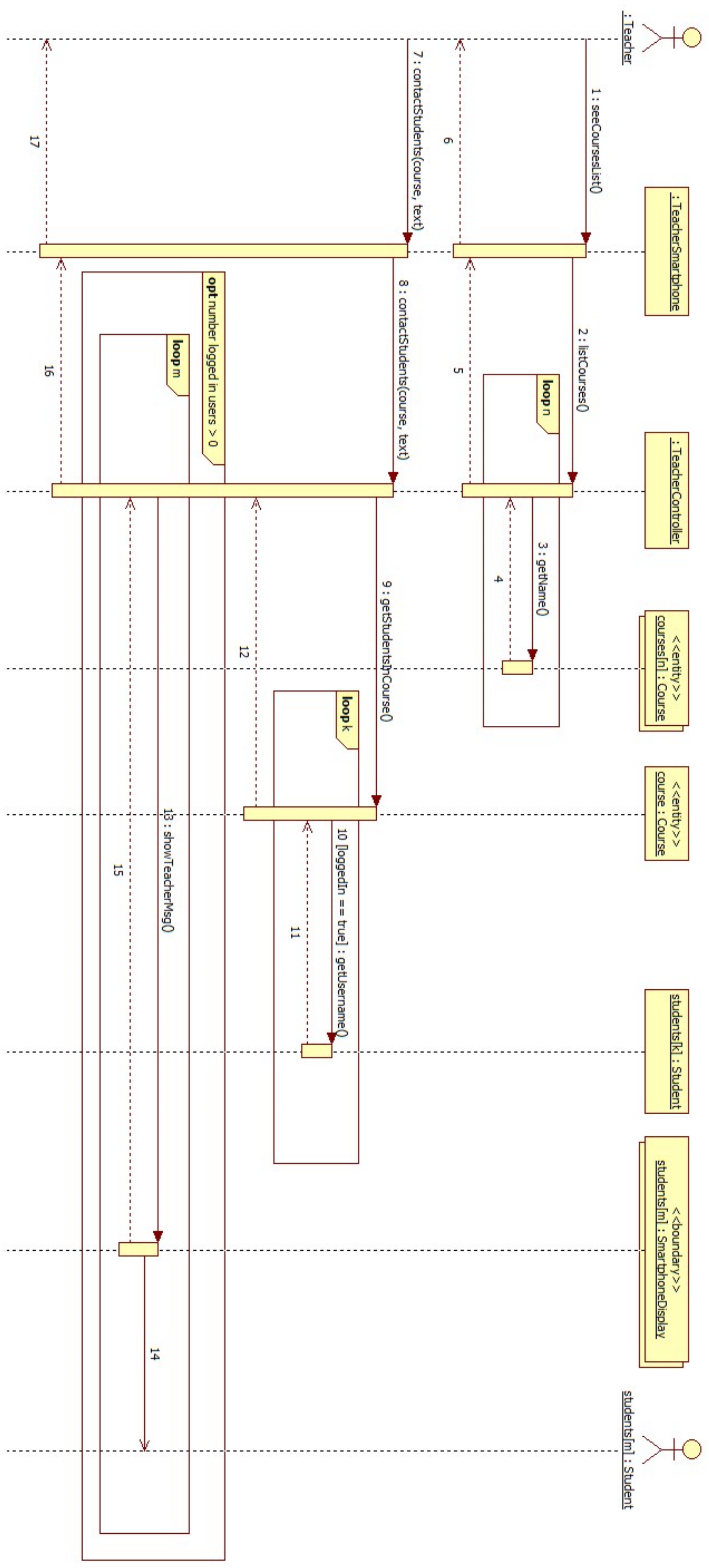


Figure 3.8: Design - UC-0011 sequence diagram

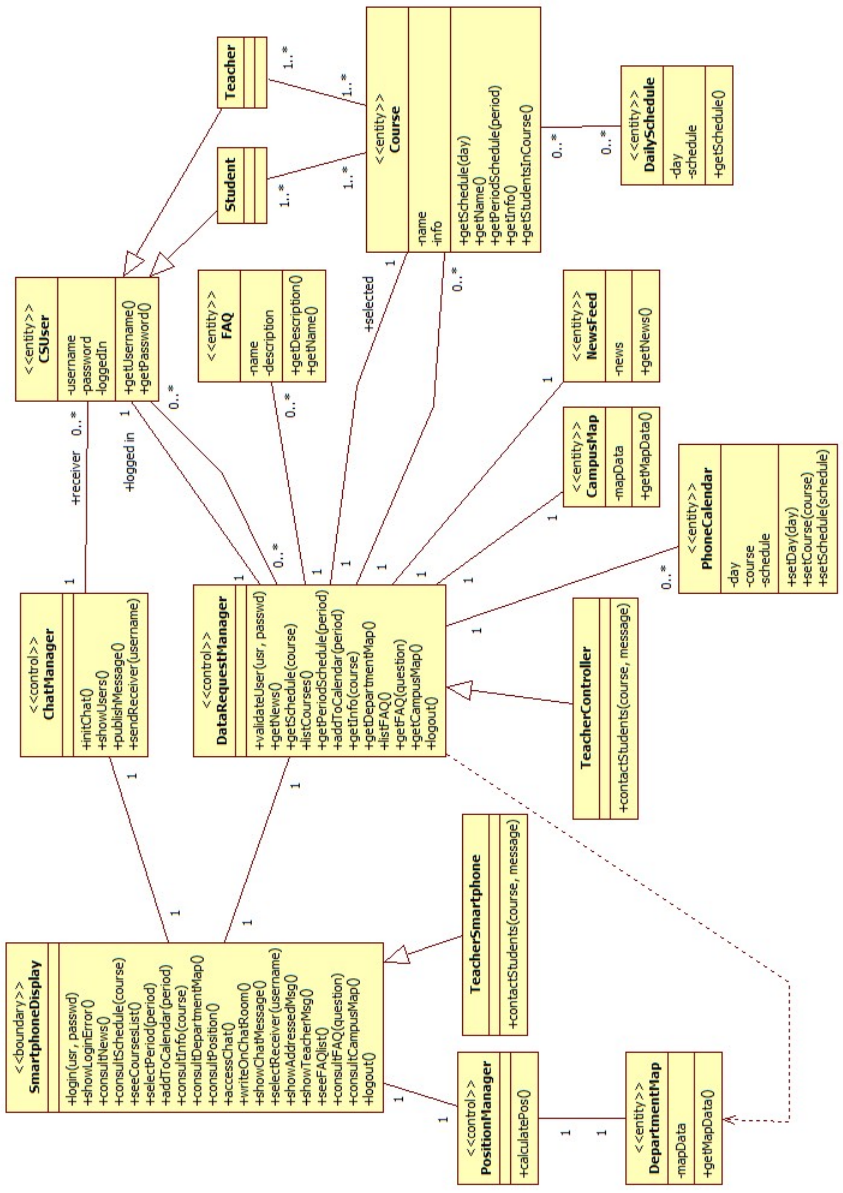


Figure 3.9: Design - Class diagram

the application would show to its users and which, in addition, allows to update that data when needed. This is what will be achieved by building a relational database.

According to [EN06, page 2]:

> A database is a logically coherent collection of data with some inherent meaning. [...] A database is designed, built, and populated with data for a specific purpose. It has an intended group of users and some preconceived applications in which these users are interested.

The aim of the presented chapter is to give a detailed view of the database design separately from the application design, since it constitutes the base over which the application will work.

To carry out this design, both sets of initial specifications of our system and main database design principles will be considered.

3.3.1. Database Design Phases

A database design process can be divided into several phases, each one of which can be carried out by means of specific methods [EN06]:

- **Conceptual Design.** It starts from the requirements specification and obtains a *Conceptual Schema* which is independent of the Database Management System[4](DBMS) and represents the highest level of abstraction.
- **Logical Design.** It starts from the Conceptual Schema and obtains the *Logical Schema*, which suits to the DBMS that is going to be used.
- **Physical Design.** It starts from the Logical Schema and obtains a *Physical Schema*. The aim of this phase is to obtain the most efficient implementation of the logical schema possible.

Only conceptual an logical design phases will be explained in the presented document, physical design will not be dealt with in this project.

3.3.2. Conceptual Design: the Entity-Relationship Model

The starting point is going to be the Entity-Relationship diagram, which intends to provide an overview of the database structure that will be detailed later. This diagram, shown in Figure 3.10, omits entities attributes in order to make it easier to be understood by means

[4]"A Database Management System (DBMS) is a collection of programs that enables users to create and maintain a database". [EN06, page 2]

of a representation of only the identified entities and the existing relationships between them. Attributes will be subsequently described.

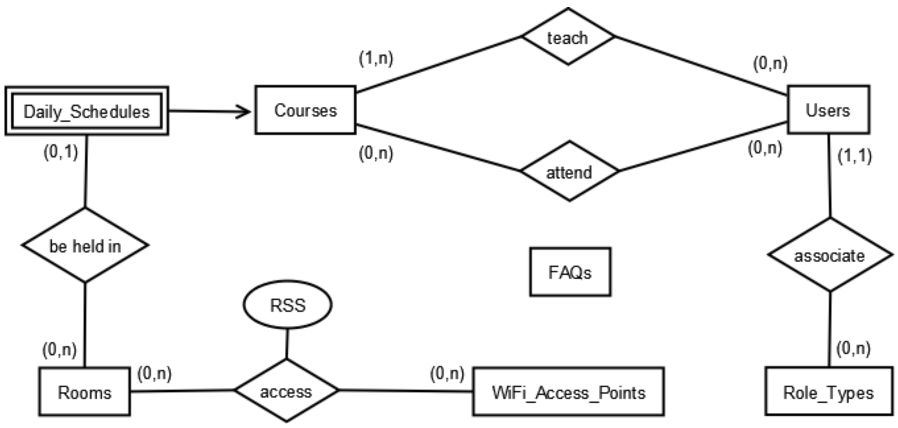


Figure 3.10: Entity-Relationship diagram

A more detailed description of entities and their relationships will be given with the aim of complementing the information shown in the diagram and, thus, making the proposed structure more clear:

Firstly, the focus will be on the USERS entity. A relationship called *associate* (1,n) exists between the mentioned entity and ROLE_TYPES entity, which establishes the kind of role each user is going to perform in the system. ROLE_TYPES has been identified as a separated entity and not as an attribute of USERS to give the possibility to extend role types in a easier way in the future (e.g. by adding an administrator role).

On the other hand, the entity COURSES is found, which has two relationships with USERS: *teach* (n,n) when the user is performing the role of a teacher in the system, and *attend* (n,n) when the user is a student.

In addition, a DAILY_SCHEDULES entity we can be found, which is a weak entity of COURSES, because its existence depends on the existence of the COURSES entity, i.e. there is no sense in keeping a schedules entity if its associated course does not exist anymore. DAILY_SCHEDULES has also a relationship with another entity that has been identified as ROOMS. That relationship is called *be held in* (1,n), and represents the fact that a certain course is going to be held in a certain room at the scheduled time.

A FAQS entity can be also be identified, which represents frequent asked questions in the system and is not related to the rest of the entities because it does not exist a semantic relation between them.

Finally, several entities regarding indoor positioning can be recognized as well: the entity ROOMS, which has already been mentioned, is connected to another entity WIFI ACCESS POINTS through a relationship called *access* (n,n). This means that, for a certain room, several (or just one or none) WiFi access points can be accessed from there, and vice versa (a certain WiFi access point can be accessed from several rooms). Relationship *access* (n,n) has an attribute RSS.

3.3.2.1. Attributes

Once entities and relationships have been described, each entity's attributes can be detailed:

- **ROLE_TYPES entity**
  - **ID.** Identifier of the type of role.
- **USERS entity**
  - **Username.** Identifier of the user.
  - **Password.** User's access key to the system, secret and unique.
  - **Name.** Name and surname of the user.
- **COURSES entity**
  - **Name.** Identifier of the course.
  - **Info.** Link to the course's related information.
- **DAILY_SCHEDULES entity**
  - **Date.** Discriminator of the schedule (thus, it can be repeated by other schedules).
  - **StartingTime.** Starting hour on a certain date.
  - **EndingTime.** Ending hour for a certain date.
- **FAQS entity**
  - **Question.** Identifier of the FAQ.
  - **Description.** Answer to the question.
- **ROOMS entity**
  - **ID.** Identifier of the room.
- **WIFI_ACCESS_POINTS entity**
  - **BSSID.** Identifier of the WiFi access point. It means *Basic Service Set Identifier* and it is represented by a MAC address.

As shown in Figure 3.10, it also exists a relationship with attributes:

- **ACCESS relationship**
  - **RSS.** The *Received Signal Strength* for a certain related room-WiFi access point pair.

3.3.3. Logical Design: the Relational Model

At this phase, the proposed Entity-Relationship model is transformed into the Relational model, which shows the final design of the database. To do this, the main relational database design principles are taken into account in order to avoid common problems, such as information redundancy and lack of consistency.

By applying those principles, tables which compose the system's database are obtained. Figure 3.11 shows an overview of the structure of tables and the relationships between them. Note that relationships that become tables have been renamed: *teach* relationship as **Teaching_Courses**, *attend* relationship as **Attending_Courses**, and *access* relationship as **Room_Information**.

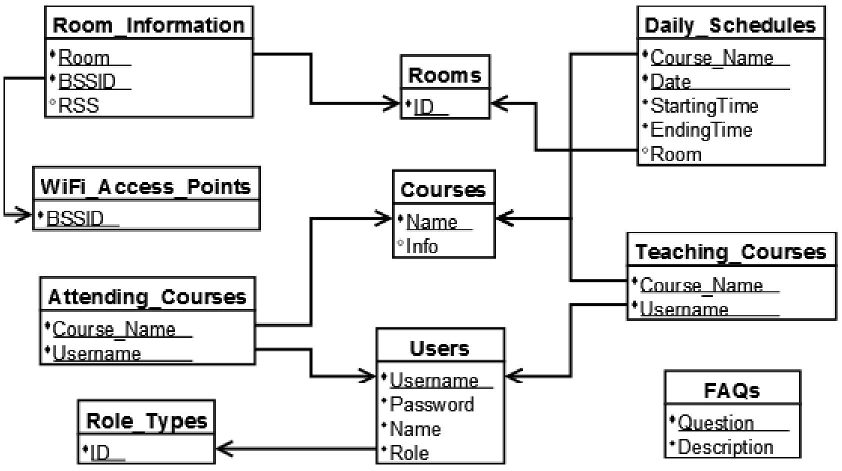


Figure 3.11: Representation of tables and foreign keys

3.3.3.1. User Views and Privileges

For greater security of the data stored in our database, it is necessary to establish some control over access and manipulation by the different users of the system and, this way, to ensure a minimum privacy of the data. Setting restrictions on tables and users directly on the database level it can be also considered a security measure, since it avoids the database to be accessed as *root* user (typically a default user in database systems), and with all privileges, by any user, even directly from the application. This is an interesting point, since connecting all application users to the database from the application as a user granted all privileges could leave the door open to manipulate the application source code in order to access and use the whole database.

Aiming this, different user privileges levels will be set, and those levels will be more or less restricted in accordance with the user type assigned to each person in the system. Therefore, four types of users will be identified:

- Anonymous user
- Student user

- Teacher user
- Administrator user

Note that those are the types of users that will be defined in the DBMS. It is necessary to make difference between types of users in the DBMS (for accessing the database and maybe manipulating it) and types of users in the application (for accessing certain application functionalities), although there will be correspondence between some of them (as using user type names as “student” and “teacher” in both DBMS and application may suggest).

Therefore, views[5] and privileges for each type of user on each table of the database will be described:

***Anonymous* user.** This type of user is granted SELECT privilege on the whole database, except for *Attending_Courses* table, rows in table *Users* referring to students, and *Password* field in *Users* table for those rows in the table referring to teachers.

***Student* user.** Users of this type are granted SELECT privilege on the whole database, except for *Password* field in *Users* table when not accessing their own password, and rows in *Attending_Courses* table which are also about other users.

***Teacher* user.** These users are granted SELECT privilege on the whole database, except for *Password* field in *Users* table when not accessing their own password, and rows in *Attending_Courses* table which refers students attending courses that are not their own courses.

***Administrator* user.** This type of user is granted all privileges on the whole database.

[5]A view is a derived table from other tables and that does not necessarily exist in physical form. [EN06, page 218]

CHAPTER 4. IMPLEMENTATION

Once the design of the system is made, its software components can be built. Main activities at this phase, and which are described in this chapter, are implementation and experiments, in that order.

> Implementation is about transforming a design model into executable code.
> [AN02, page 350]

The analysis and the design were described within an Object-Oriented paradigm. This encouraged the implementation being also carried out in an Object-Oriented way, since it made the transition from design to implementation process easier. In addition, this paradigm has advantages such as its wide diffusion and the high number of tools available, as it is intrinsically a very organized approach to programming.

Starting from the system design shown in the previous chapter, where the system was shown from different points of view, implementation can also be split into and described through different aspects of the system.

4.1. Programming Languages and Tools

As the decision of developing an Android application was made, that implied the use of Java as a base programming language. However, it was necessary to complement Java libraries with the *Android SDK* package, that provides tools and APIs to develop applications on the Android platform [web10c], and to use the *Eclipse* IDE[1] (versions 3.5 *Galileo* or 3.6 *HeliOS*) as a recommendation since it allows you to install an *Android Development Tools (ADT)* plugin available that provides an integrated environment in which to build Android applications [web10c].

About database building, the use of SQL (Structured Query Language) was needed, but taking into account some particularities of the DBMS chosen: *MySQL* (version 5.1.41), because of being the most widespread in web services and free software. For being the database accessed as remote, an *Apache* server (version 2.2.14) was installed and run, as well as a *PHP* engine (version 5.3.1), that led to the use of PHP language as well. It was necessary to access the remote MySQL database by means of a Web service using PHP as a middle layer because there are not MySQL client libraries available for the Android development environment.

[1] Integrated Development Environment

4.2. Distributed System Architectural Models

As shown in the design chapter, both main architectural models (i.e. client-server and peer-to-peer) are present in the proposed system, since each of them is more suitable to provide each functionality than the other.

4.2.1. Client-Server

Taking as starting point the design system deployment diagram shown in Figure 3.5, it can be made more specific to show how the client-server model has been built and deployed for this system. As it can be seen in Figure 4.1, clients (smartphones running the Android application needed to be developed, *UniversityDroid*) request data using the HTTP protocol to a Web server (Apache) that, by means of PHP scripts, retrieves that data from the database (managed by MySQL). Users will interact with the Android app by using their smartphones.

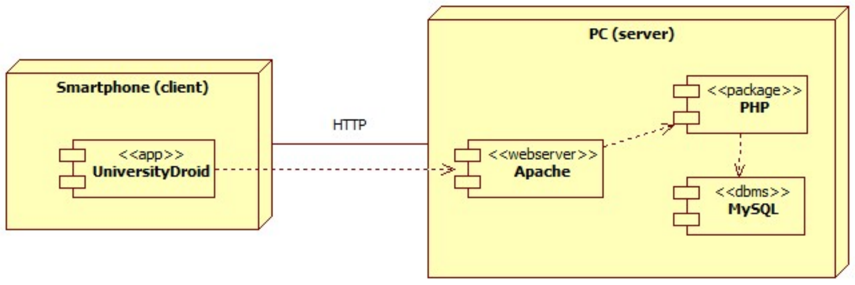


Figure 4.1: Client-server deployment

Regarding secure communication, it must be pointed out that the time available to develop this project did not allow to implement the security decisions made in Chapter 3 (see section 3.1.3.).

Most of the expected functionalities in the system are carried out leaning on this client-server architecture. The particular way each of those functionalities (which are going to be referred to in terms of the use cases) are implemented and their working are described below.

[UC-0001] Login

As mentioned before, the Android app acts like a client of the remote Web server. In general terms, these are the steps followed in the client-server intercommunication in order to allow an user to get logged in:

- If it is the first time an user is using the app, the user status icon in the top right corner of the main menu looks in grey (offline status) and the chat function disabled, as shown in Figure 4.2. To log in, the user needs to go to the log in dialog of the app shown in Figure 4.3. If both username and password and introduced and ”Log in” button pushed, then the app sends that data to the remote Web server via HTTP.

- At the server side, data is retrieved by means of a PHP script using a GET method. That PHP script connects to the database as a user defined in the DBMS and called ”login”, that was specially created to allow a user of the app to log in the system, and it is exclusively used for this (it was only granted select privilege on tables *Role_Types* and *Users*), since the user is not authenticated in the system yet. A query with the username and password provided is made in order to check if the user is a valid one.

- If the user is valid, the PHP script returns the role of the user (”T” for a teacher, ”S” for a student), and then the chat function becomes enabled (remember it was said in Chapter 1 that a user will need to be logged in to access the chat system –see section 1.3.3.3.), and the user status icon changes to green (online status) as shown in Figure 4.2. If she is not, the PHP script returns a code to indicate that and the app notifies the user of it.

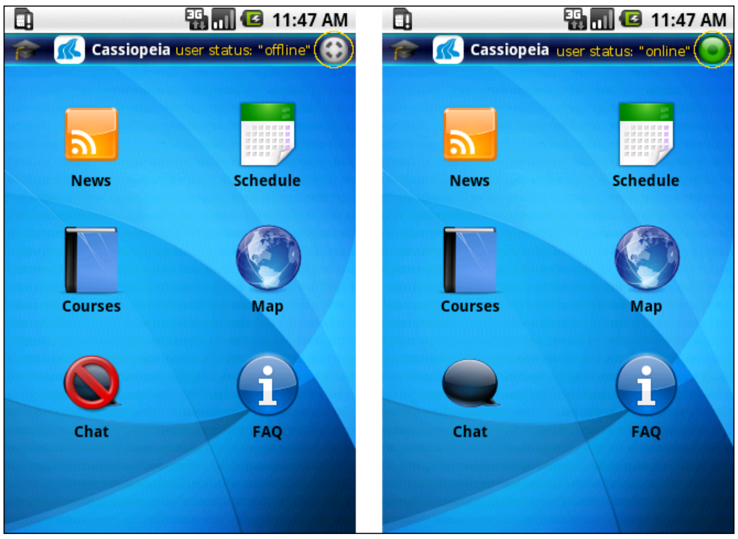


Figure 4.2: UniversityDroid - Main menu screenshots

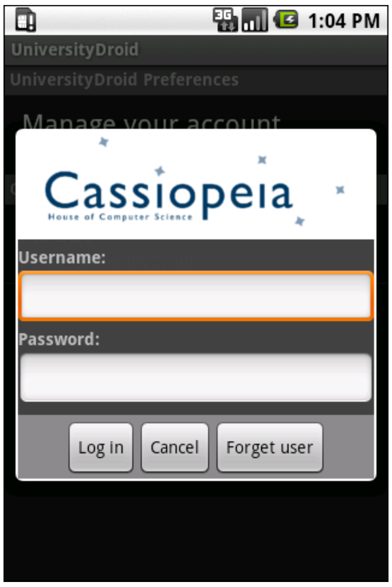


Figure 4.3: UniversityDroid - Login dialog screenshot

There is no need to open and close sessions in the system or to make use of cookies as it is usually done when requesting Web services, because the data retrieved from the remote database is not going to be updated frequently. Once a username and a password have been checked and determined as valid, they are stored in the Android app (as well as the role of the user that is received), what also avoids that users have to log in everytime they are launching the app. Subsequent times the user runs the app, her status will be online and the chat option enabled from the beginning. However, the user can change her account settings (i.e. username and password) whenever she wants, and then the new data provided will be checked and stored if correct as well.

About security issues for this functionality, passwords are stored in the database after their MD5 hash have been calculated, as well as they are calculated by the app before send them to the server via HTTP.

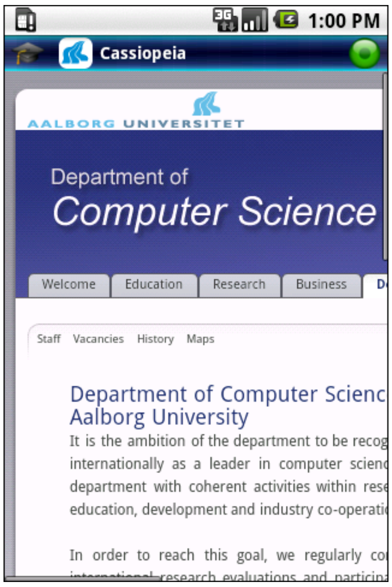
[UC-0002] Logout

This option is not explicitly accessible in the app, that is, there is not any ”Log out” option as there is to log in, but it exists the possibility to simply erase username and password from the log in dialog (see Figure 4.3). Then the previous stored username and password are cleared, what implies that a user becomes again an anonymous one. As no session variable nor any cookie is being used, there is no need to access the server, so this functionality itself is not leaning on the client-server model (though a previous login functionality must had been successfully completed before, and it is leaning on the client-server model).

[UC-0003] Consult news

When selected this option in the app’s main menu, a web browser activity is launched.

The department's website is requested via HTTP and displayed on the smartphone, as it can be seen in Figure 4.4.



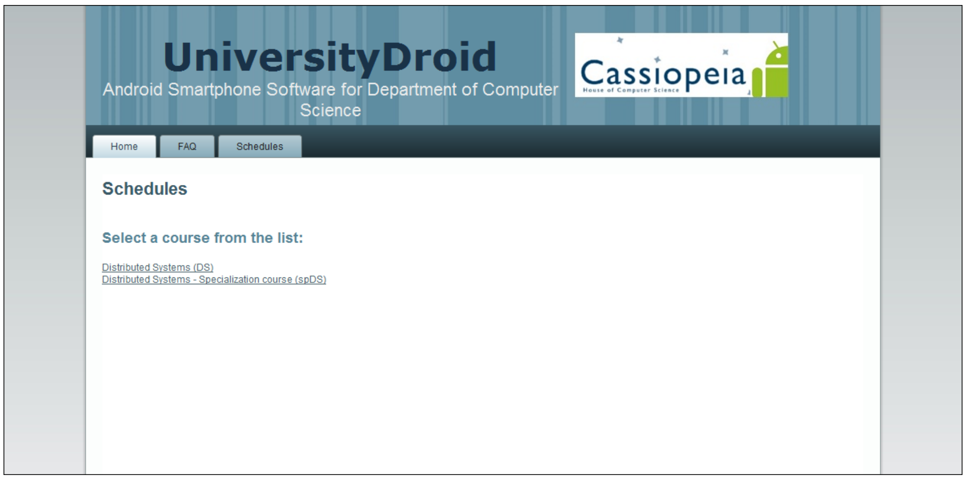
Department of Computer Scienc
Aalborg University
It is the ambition of the department to be recog
internationally as a leader in computer scienc
department with coherent activities within rese
education, development and industry co-operati

In order to reach this goal, we regularly co

Figure 4.4: UniversityDroid - News view screenshot

[UC-0004] Consult schedule

Analogous to the foregoing on the working of the consulting news functionality, when the ”Schedule” option in the main menu is selected, a web browser activity is also launched and schedules are requested by means of the HTTP protocol. However, the URL provided this time (`http://beyondar.com/universitydroid/web/courses.php`) is not addressing a previously existing website as it was when requesting the department's news, but it addresses a website at our server that was necessary to build in order to show the information stored in our database. The database is now accessed as anonymous user (see section 3.3.3.1.), since the information that is retrieved in this functionality is of general interest and a user does not need to be logged in the system to get it. A view of the website requested from a PC web browser is shown in Figure 4.5.


UniversityDroid

Android Smartphone Software for Department of Computer Science

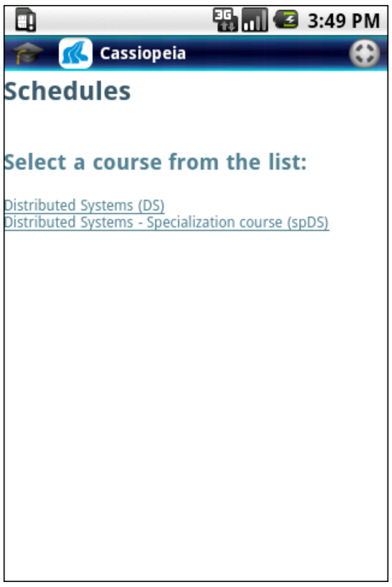
Schedules

Select a course from the list:

Distributed Systems (DS)
Distributed Systems - Specialization course (spDS)

Figure 4.5: PC Web browser - Website screenshot

When that website is requested from the app web browser activity, it is displayed on the smartphone as shown in Figure 4.6.



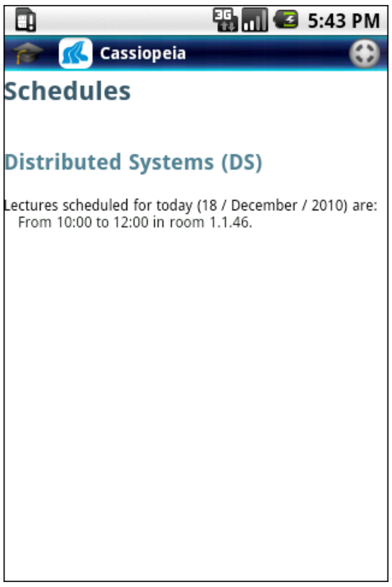
Schedules

Select a course from the list:

Distributed Systems (DS)
Distributed Systems - Specialization course (spDS)

Figure 4.6: UniversityDroid - Schedules courses list view screenshot

Then, if the user selects a course from the list, schedules for the current date and for the selected course are retrieved from the database by means of PHP and then showed as it can be seen in Figure 4.7 (or a message for warning the user there are no lectures for that course if the course has not been scheduled for the current date).



Schedules

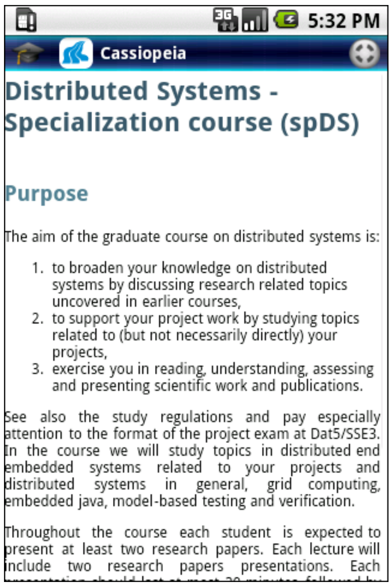
Distributed Systems (DS)

Lectures scheduled for today (18 / December / 2010) are:
From 10:00 to 12:00 in room 1.1.46.

Figure 4.7: UniversityDroid - Schedules view screenshot

[UC-0005] Consult course information

For providing this functionality, the first idea was requesting the department's intranet URL (`https://intranet.cs.aau.dk/education`), since it already shows all the information available about the existing courses. However, the Android platform does not allow to install any certificate to make that website considered as trusted (note that the protocol to request the mentioned URL is HTTPS and not HTTP), and, thus, it can not be displayed on the smartphone. Therefore, another web page was developed in order to show courses information, and the process is similar to that was described before: the app launches a web browser activity when clicking on the "Courses" icon in the main menu and requests the URL of the developed site (`http://beyondar.com/universitydroid/web/courses_info.php`), via HTTP. After the server replies, the website is displayed on the smartphone as shown in Figure 4.8.



Distributed Systems - Specialization course (spDS)

Purpose

The aim of the graduate course on distributed systems is:

1. to broaden your knowledge on distributed systems by discussing research related topics uncovered in earlier courses,
2. to support your project work by studying topics related to (but not necessarily directly) your projects,
3. exercise you in reading, understanding, assessing and presenting scientific work and publications.

See also the study regulations and pay especially attention to the format of the project exam at Dat5/SSE3. In the course we will study topics in distributed end embedded systems related to your projects and distributed systems in general, grid computing, embedded java, model-based testing and verification.

Throughout the course each student is expected to present at least two research papers. Each lecture will include two research papers presentations. Each

Figure 4.8: UniversityDroid - Courses information view screenshot

In this case, the database is also accessed as an anonymous user (similar to schedules, the information about courses is of general interest and users do not need to be logged in to consult that information).

[UC-0007] Consult campus map

The most suitable way to show a map of the whole main campus is by means of the existing *Google Maps* application for mobile devices. The Android development environment supports the use of a Google API to allow an Android app to access Google apps such as Google Maps. In this project, a Google API version 1.5 is used (API level 3).

The access to the campus map from the app has been designed in such way the user needs to click on the ”Maps” icon in the main menu. The map that is then shown is not the campus map, but the department map (see the subsequent description of *[UC-0008] Consult department map* to get further information about that functionality). The reason why the map of the campus is not shown first is because the map that provides the most relevant information to the user is the map of the department. In other words, as the user is going to use the application when she is at the department, it is going to be more helpful for her to see her location inside the department than the location of the department within the main campus. So, the user will need to push the ”MENU” button of her smartphone to see the option called ”Map” (see Figure 4.9), which leads to the campus map by clicking on it.

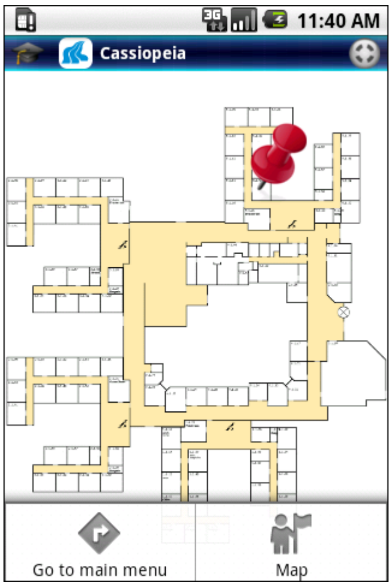


Figure 4.9: UniversityDroid - Access to the campus map view screenshot

Then, the app launches a map activity which makes use of Google Maps to show a map of the campus by means of the Global Positioning System (GPS). In such map, the location of both the department and the user is pointed at by means of two icons which the launched activity adds to the requested map: a blue flag to point at the location of the department and a green circle to point at the location of the user, as can be seen in Figure 4.10.

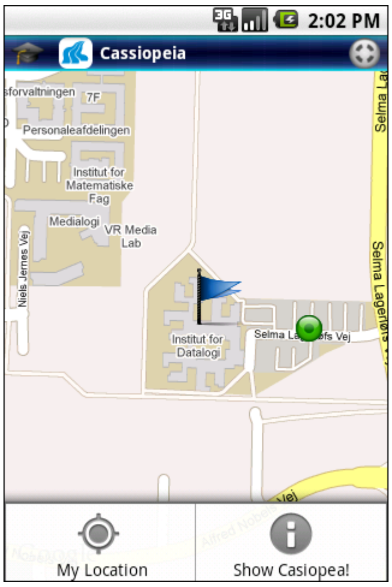


Figure 4.10: UniversityDroid / Google Maps - Campus map view screenshot

[UC-0008] Consult department map

As it was detailed in the previous *[UC-0007] Consult campus map* functionality description, the app shows the map of the department when clicking on the ”Map” icon in the main menu. Then the app launches an activity that is responsible for drawing the map while the the location of the smartphone inside the department is calculated by following the fingerprinting method, as decided in the previous chapter (see section 3.1.5.).

When the activity is launched, the BSSID and RSS of the nearby access points are collected and send to the server. This server is a basic Java application which waits until a client connects. When a client requests a connection, a thread is created to receive the information sends by the cell-phone, access to the room_information database, execute the location algorithm and return the name of the room to the cell-phone. The coordinates of the room on the map are stored in the application.

The room_information database is filled with a simple Android application whose ask for the name of the room where is the cell-phone and then the results of the collecting are sent to the database room_information on the Web server using a INSERT method by the mean of a PHP script which connects as an anonymous user. Figure 4.11 shows the interface of that collect application.

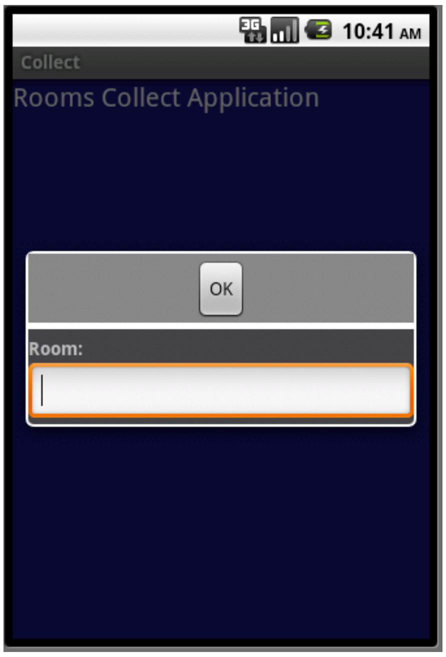


Figure 4.11: Interface of the application to collect the BSSID and RSS of the nearby access points during the training phase

The system was implemented but not tested.

Finally, the department map with the user's position pointer looks like shown in Figure 4.12.

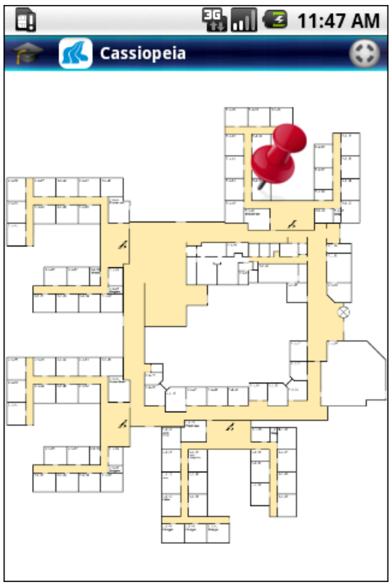
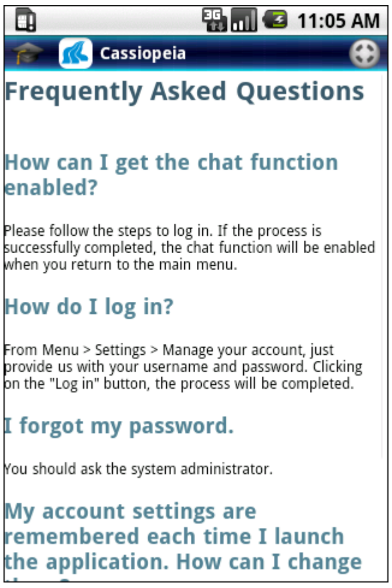


Figure 4.12: UniversityDroid - Department map view screenshot

[UC-0009] Consult FAQ

This functionality follows the same process than the followed in *[UC-0003] Consult news* and *[UC-0004] Consult schedule*: a web browser activity is launched within the application, and then it requests the URL `http://beyondar.com/universitydroid/web/faq.php` to the Web server. The FAQ website displayed then looks like shown in Figure 4.13.



Frequently Asked Questions

How can I get the chat function enabled?

Please follow the steps to log in. If the process is successfully completed, the chat function will be enabled when you return to the main menu.

How do I log in?

From Menu > Settings > Manage your account, just provide us with your username and password. Clicking on the "Log in" button, the process will be completed.

I forgot my password.

You should ask the system administrator.

My account settings are remembered each time I launch the application. How can I change

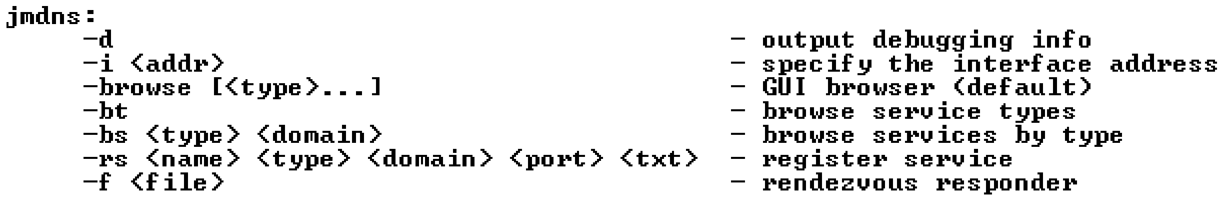
Figure 4.13: UniversityDroid - FAQ view screenshot

Those FAQ shown on the website are also retrieved from our database by using PHP as anonymous user.

4.2.2. Peer-to-Peer

[UC-0010] Use chat

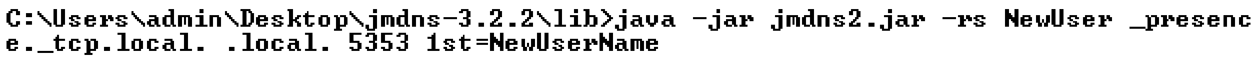
As explained before Link-Local XMPP uses mDNS to advertise for presence and to look for other devices. This technique, explained on the IETF website [SCb], uses datagrams comparable to the ones used for DNS. Every single device on the network keeps its own DNS list and when a clients wants to know the IP adress of an other client from its name, the latter answer with its IP adress. All those actions are made on the mDNS IP adress, i.e. 224.0.0.251 via the port 5353 in User Datagram Protocol (UDP).
The easiest way to proceed was to look for existing solution to do multicast DNS and DNS Service Discovery. Many aplications were developed by other people. However, due to the specifications of the project (developing for Android) it had to be in Java. After searching on the Internet the best solution available for that appeared to be JmDNS, an easy-to-use pure-Java implementation that runs on most JDK1.6 compatible virtual machines. It was developped by Arthur van Hoff and released under the Apache 2.0 license. [vH]
Notwhithstanding JmDNS is in Java, many adaptations had to be done. Indeed some packages and some methodes can't be used on Android and some modules are different. Moreover the JmDNS debug window is useless for a client application. Thus the program had to be modified and adapted to work on an Android device.
In order to test the mDNS software, tests were first run on computers. To do so Pidgin and the Zeroconf/Bonjour plugin that allows to perform Link-Local XMPP. Once installed JmDNS could be tested using command lines respecting the syntax shown on Figure 4.14.



```
jmdns:
     -d                                          - output debugging info
     -i <addr>                                   - specify the interface address
     -browse [<type>...]                         - GUI browser (default)
     -bt                                         - browse service types
     -bs <type> <domain>                         - browse services by type
     -rs <name> <type> <domain> <port> <txt>     - register service
     -f <file>                                   - rendezvous responder
```

Figure 4.14: List of JmDNS Commands

For instance to advertise the presence of a new user the command line shown on Figure 4.15 has to be written :



```
C:\Users\admin\Desktop\jmdns-3.2.2\lib>java -jar jmdns2.jar -rs NewUser _presenc
e._tcp.local. .local. 5353 1st=NewUserName
```

Figure 4.15: Adding a new user with mDNS

Where

- -rs means registring a new service.
- NewUser is the name of the user.
- _presence._tcp.local. is the type of the brodcast, i.e registering a new user on the local area network via tcp.

- .local. is the domain where the mDNS record has to be published. Here the record is sent only to the clients connected to the same hotspot.
- 5353 is the port that has to be used to contact that new user.
- 1st=NewUserName is the optional TXT report that allows to send additional information. Here only the nickname of the user is sent.

Afterwards the new user is detected by Pidgin and added to the contact list as shown on Figure 4.16.

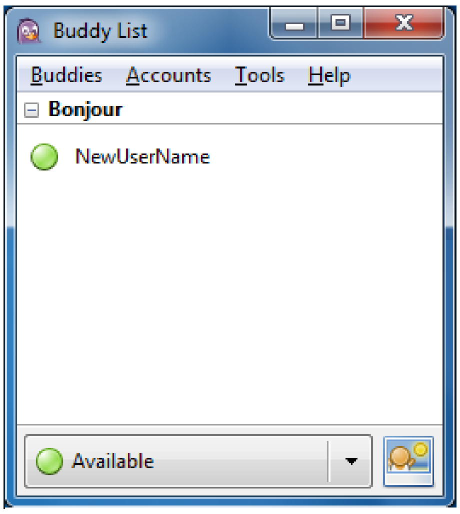
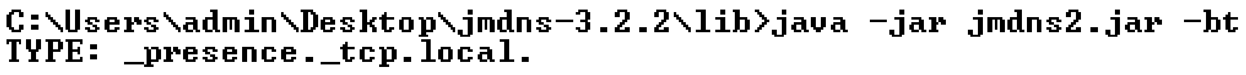


Figure 4.16: Contact list on Pidgin

Moreover if JmDNS is asked to listen to the type broadcasted it sees that there is one of type presence. Figure 4.17



```
C:\Users\admin\Desktop\jmdns-3.2.2\lib>java -jar jmdns2.jar -bt
TYPE: _presence._tcp.local.
```

Figure 4.17: List of the service types provided by JmDNS

Once the tests realised and the principle of JmDNS understood it was possible to adapt it for Android. The same kind of tests were made in order to verify its behavior on the smartphones.
However some problem came up when trying to use JmDNS on the phones. The first one is that some manufacturers changed the wpa_supplicant file, in charge of geetting identification on a network, of some of their phones, resulting in an impossibility to listen to mDNS records as explained in the Android forums [?] and [?] . One solution would be to replace the wpa_supplicant but it would be long and hasardous to develope and it would require a device with a root access. Thus, only the HTC Hero could be used for the tests.
In order to use JmDNS the multicastlock needs to be set. It is done by adding a line in the AndroidManifest.xml file :

```
¡uses-permission android:name=”android.permission.CHANGE_WIFI_MULTICAST_STATE” /¿
```

And by aquiring it in the code :

```
WifiManager wifi = getSystemService( Context.WIFI_SERVICE );
MulticastLock lock = wifi.createMulticastLock(”fliing_lock”);
lock.setReferenceCounted(true);
lock.acquire();
```

Finally the lock must be released when not used anymore. Other locks have to be set, in order to use the chat application, like the Internet one.

```
lock.release()
```

[UC-0011] Contact students

Figure 4.18 shows the chat activity for the application:

a) To login, the application can take around ten seconds, so it shows a dialog to let know the user that the application is working.

b) This screen show a list with all the users connected using the MDNS and authenticated with the server.

c) If there is a chat session opened, the application shows a bubble.

d) This screenshot shows a conversation between two users. The icon of the user is blue, it means that the displayed conversation is from this user.

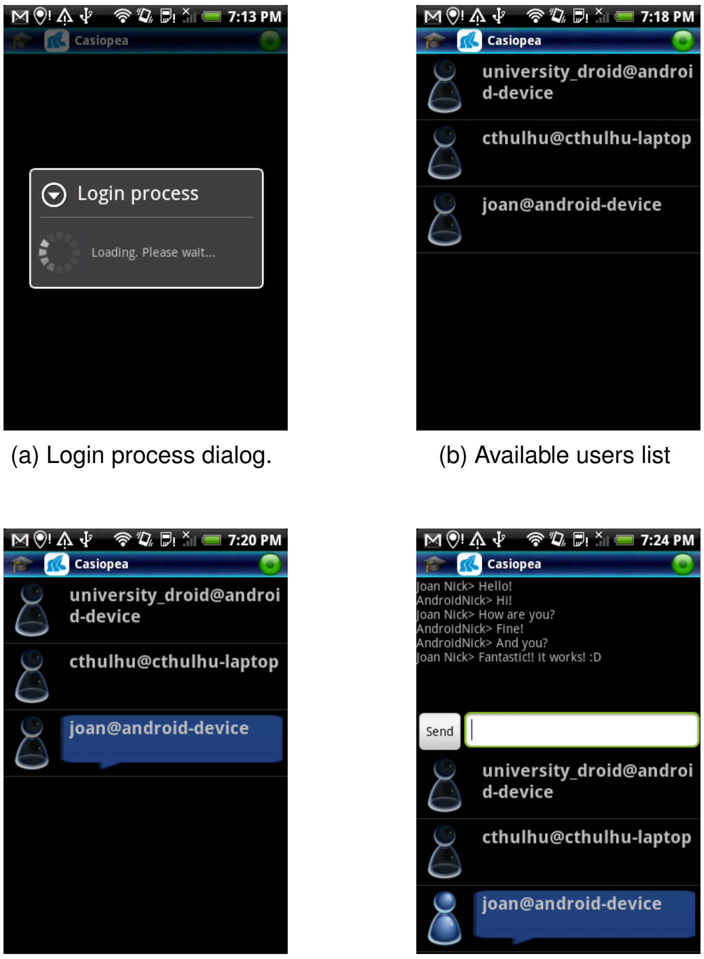
The communications between the users it's supposed to use SSL protocol and XMPP (see Section 3.1.3. and 3.1.), but due to the lack of time, this prototype establish a TCP connection between the users in the specified port (the port is provided by the MDNS protocol information).

When a new TCP connection is established, each node transfer its own UI (see section 3.1.2.2.) and each peer check it in the existing users list. If the user exist, the connection will be accepted, otherwise will be dropped.

To finalize the session chat, the users only has to close the TCP connection, but the previous messages will be stored, so it is possible to retrieve the previous messages.

4.3. Software Architecture

In this section the short explanation of the code is given. Mainly, the project is divided in seven modules: Main user interface, access the database, building and department maps,



Joan Nick> Hello!
AndroidNick> Hi!
Joan Nick> How are you?
AndroidNick> Fine!
AndroidNick> And you?
Joan Nick> Fantastic!! It works! :D

(a) Login process dialog. (b) Available users list

(c) Chat session opened (d) Chating with an user

Figure 4.18: UniversityDroid - Chat activity screenshots.

wifi, chat and some utilities.

Main user interface

This module manage the main user interface for the application and the interface of the following features: Access website content (like schedule, FAQ, news and courses information). This module is the one that check the user credentials and create a session for chat. Figure 4.19 shows the UML diagram for this module.

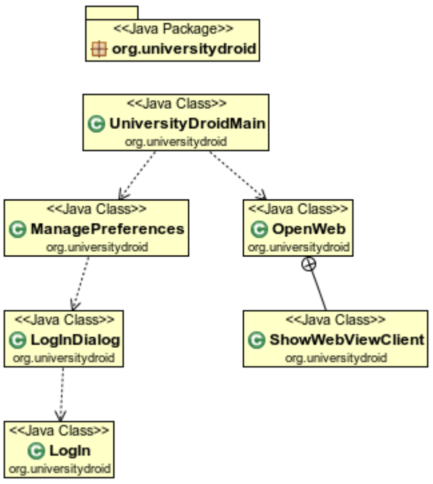


Figure 4.19: UML diagram - Main user interface

To access the schedule, FAQ, news and courses information, the activity OpenWeb class is used, which open a web browser to show the HTML information provided by the server (see Figures 4.13, 4.2, 4.4, 4.6, 4.7 and 4.13).

Access the database

To access the database this module is used. Android doesn't have API to access the database (see Section 4.1.), so this module establish a connection with a external PHP server to access the database. This PHP server request the database for the needed information and transfer the data using XML, then the Android device only has to read the data and process and store it (see Section 4.2.1.). Figure 4.20 shows the UML diagram for the database module.

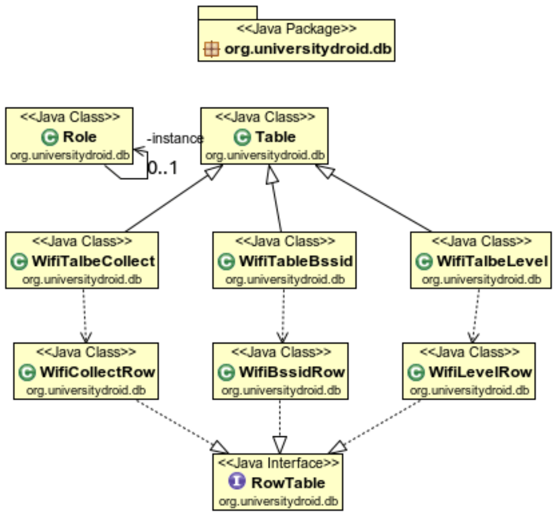


Figure 4.20: UML diagram - Database managment

Maps

This module draws the building and the campus maps. For the campus view, it uses the GPS sensor (see Figure 4.10) and the Google API. [Cod]. For the building map, it uses the wifi sensor and a class to draw the map and the pointer (see Figure 4.12). Figure 4.21 shows the UML diagram for the maps module

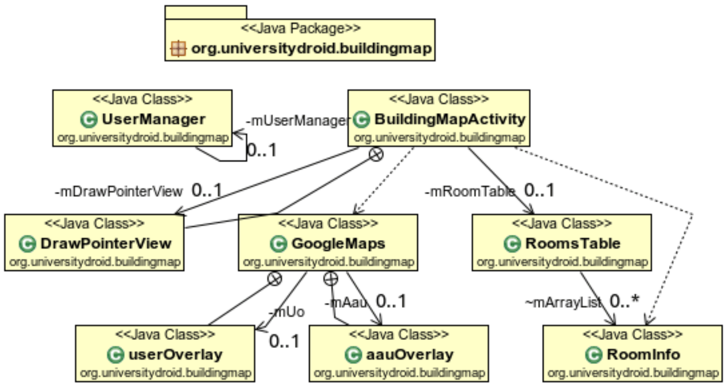


Figure 4.21: UML diagram - Maps

Wifi

This module calculate the position according to the methods explained in the Section 3.1.5. Figure 4.22 shows the UML diagram for the wifi module.

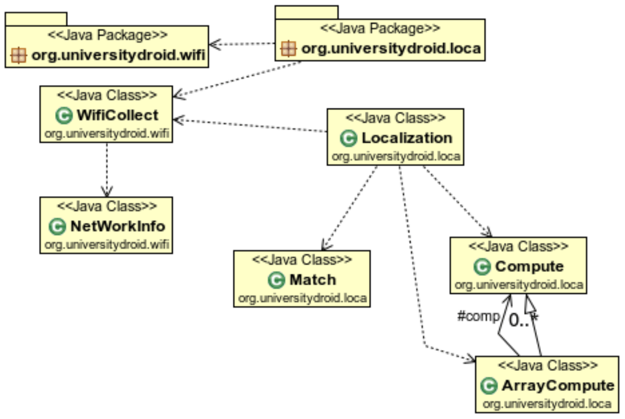


Figure 4.22: UML diagram - Wifi

Chat

This module manages the distributed system architecture and the chat system.For the Distributed, the module uses the Jmdns library and for the chat system, in stead of XMPP, a plain text is used due to the lack of time for the implementation. When the service has been registered in the overlay-network, the chat module opens a TCP port to manage the chat connections. Figure 4.23 shows the UML diagram for chat module.

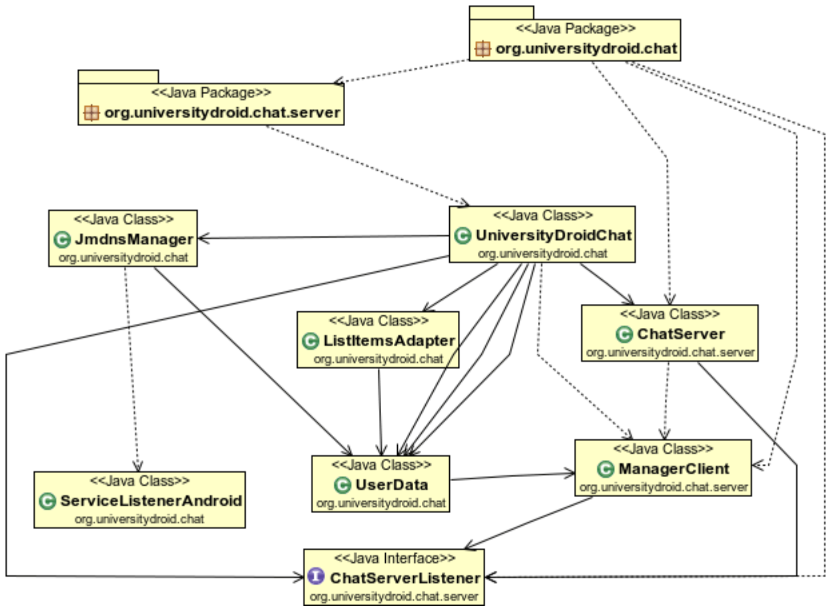


Figure 4.23: UML diagram - Chat

Utilities

This module manage the user session with the server and create the request for the XML data stored in the server, like the wifi tables and user information. The Figure 4.24 shows the UML diagram for utilities module.

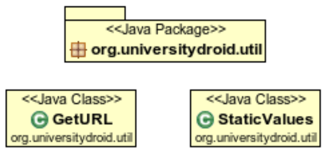


Figure 4.24: UML diagram - Utilities

CHAPTER 5. CONCLUSION AND FUTURE WORK

Once the prototype version of the project (final version in this report) has been finished, an overall assessment of it can be done, as well as a reflection of the improvements that could be incorporated to this final prototype of the system.

5.1. Milestones Assessment and Conclusions

What objectives initially set have been achieved successfully can be now listed:

- The application was finally developed meeting all the requirements agreed between the students and supervisors.
- The application presents the modularity needed to allow correct later reuse and extension, which will in turn allow to follow the future lines of work that will be described in the subsequent section.
- Despite the difficulties encountered in the implementation phase, such as having misunderstood the in-door localization model on which RedPin is based and having found problems in managing multicast in one of the devices given to test the app, contingency plans to provide alternative solutions could be implemented and finally the project could be finished before the stipulated deadline.

As conclusion, it can be established that the mixture of different distributed systems architectural models (client-server and peer-to-peer) seems to be successful in that each of the functionalities provided by the system is deployed on the architectural model that offers the best performance for that functionality.

Regarding the Android platform, it was found that it is a very versatile environment for developers (despite of the problems in managing multicast in several versions of the platform as has just been mentioned), since it is open source and provides Google APIs, what, undoubtedly, offers a wide range of possibilities and added functionalities when developing applications for smartphones.

About the in-door localization functionality, this report has allowed a discovery of some of the existing methods that are used to locate a cell-phone in a building. A method had been implemented but due to a lack of time, the suggested algorithm was test quickly and could not be improved.

5.2. Future Lines of Work

This prototype doesn't implement security, so, using Java libraries it is possible to implement SSL see Section 2.3.

Using XMPP instead of plain text would also allow to use profile pictures and to easily add features thanks to the XEPs.
Using XEPs other interesting properties could be added to the chat system such as channels between more than two users, allowing file transfers or video and voice chat.

Due to the amount of problems with multicast (see Section 4.2.2.), if the application has to be used by all the students, it's necessary to find a solution to this issue. In this section a complementary method is proposed. Exist the possibility to add more cooperating methods for the distributed network:

- A list of devices (with the kernel version, manufacturer and Android version) that support multicast is needed, so when new user that doesn't support multicast launch a request, the users that can receive the petition, will establish a point to point connection (using TCP) to share the table. Then the user has to update the table periodically, to have up to date the host table (see Section 3.1.2.4.).
- If in the network there are not any user that support multicast, there are two possibilities: Scan the local network looking for users connected to share the routing table, or the other solution is to add a central server to provide the routing the host table.

[UC-0001] Login

MD5 hash function applied when passwords are sent through the network and stored in the database, use always the same seed: that is, for a certain string, the result of applying MD5 hash to it is always the same (in other words, a certain password looks always the same).

One way to solve that vulnerability is make the server generate a random string (seed), and send it to the client that wants to log in the system. Then, what the client should do is hashing the password that is going to send for authentication twice: a first time by means of the MD5 function, and a second time by means of the seed which has received. Once the password is received at the server side, the server, knowing the seed sent to the client, can undo the double-hashed password string. Now, if the password is intercepted while crossing the network, it would be harder to decrypt it, since the seed used is not known.

[UC-0004] Consult schedule and [UC-0005] Consult course information

This information has been considered as data of general interest, since there is no need to be logged in the system to consult it, and it is retrieved as anonymous user in the database. However, both sets of information, schedules and courses, could be also retrieved depending on the user who is requesting for them when that user is logged in the system. That is, for example, a user logged in the system could have the option to consult directly only the courses of her interest (i.e. the courses she is attending).

[UC-0008] Consult department map

As mentioned in the previous conclusions section, the algorithm to calculate an in-door position could not be improved. So it needs to be done in a future work. Moreover, instead of storing the coordinates of a room into the application, it will be better to use a database which stores the name of the room and its coordinates. The server will return the coordinates of the room to the cell-phone instead of the name of the room.

Another way to improve the accuracy will be to take several measurements in one room, because the values of the RSS of an access point fluctuates a lot during the day.

BIBLIOGRAPHY

[AJMV97] Paul C. van Oorschot Alfred J. Menezes and Scott A. Vanstone. Handbook of applied cryptography. 1997. Available online: `http://www.cacr.math.uwaterloo.ca/hac/`.

[AN02] J. Arlow and I. Neustadt. *UML and the Unified Process. Practical Object-Oriented Analysis & Design*. 2002.

[And01] David G. Andersen. Resilient overlay networks. 2001.

[App10] Apple. Iphone. `http://www.apple.com/iphone/`, October 2010.

[BMR$^+$07] F. Buschmann, R. Meunier, H. Rohnert, P. Sommerlad, and M. Stal. *Pattern-Oriented Software Architecture: a System of Patterns*. 2007.

[Bol08] P. Bolliger. Redpin-adaptive, zero-configuration indoor localization through user collaboration. In *Proceedings of the first ACM international workshop on Mobile entity localization and tracking in GPS-less environments*, pages 55–60. ACM, 2008.

[Bon00] André B. Bondi. Characteristics of scalability and their impact on performance. pages 195–203, 2000.

[Bra03] The Open Brand. Apple inc. - unix 03. `http://www.opengroup.org/openbrand/register/brand3555.htm`, May 2003.

[CAMZ] Alan L. Cox Sameh Elnikety Romer Gil Karthick Rajamani Emmanuel Cecchet Cristiana Amza, Anupam Chanda, Julie Marguerite, and Willy Zwaenepoel. Specification and implementation of dynamic web site benchmarks.

[CDK00] G. Coulouris, J. Dollimore, and T. Kindberg. *Distributed Systems: Concepts And Design*. 3rd edition, 2000.

[CER] CERT. Home network security. `http://www.cert.org/tech_tips/home_networks.html`.

[CG85] Bill Croft and John Gilmore. Bootstrap protocol (bootp). RFC 919, September 1985. `http://tools.ietf.org/html/rfc951`; accessed November 6, 2010.

[CIS] CISCO. Ip routing. `http://www.cisco.com/en/US/tech/tk365/tsd_technology_support_protocol_home.html`.

[CIS99] CISCO. Osci model. `http://www.cisco.com/en/US/docs/internetworking/technology/handbook/Intro-to-Internet.html`, July 1999.

[Cod] Google Code. Google apis add-on. `http://code.google.com/intl/es/android/add-ons/google-apis/reference/index.html`.

[Com10] The Nielsen Company. Android most popular operating system in u.s. among recent smartphone buyers. http://blog.nielsen.com/nielsenwire/online_mobile/, October 2010.

[Die99] T. Dierks. The tls protocol. version 1.0. RFC 2246, January 1999. http://tools.ietf.org/html/rfc2246; accessed November 6, 2010.

[Die06] T. Dierks. The tls protocol. version 1.1. RFC 2246, April 2006. http://tools.ietf.org/html/rfc4346; accessed November 6, 2010.

[Die08] T. Dierks. The transport layer security (tls) protocol. version 1.2. RFC 5246, August 2008. http://tools.ietf.org/html/rfc5246; accessed November 6, 2010.

[Dro97] R. Droms. Dynamic host configuration protocol. RFC 2131, March 1997. http://tools.ietf.org/html/rfc2131; accessed November 6, 2010.

[EKLMP05] JON CROWCROFT ENG KEONG LUA and STEVEN LIM MARCELO PIAS, RAVI SHARMA. A survey and comparison of peer-to-peer overlay network schemes. 2005.

[EN06] R. Elmasri and S. B. Navathe. *Fundamentals of Database Systems*. 4th edition, 2006.

[Foua] The Apache Software Foundation. Ssl/tls strong encryption: An introduction. http://httpd.apache.org/docs/2.0/ssl/ssl_intro.html.

[Foub] XMPP Standrds Foundation.

[GC05] Tim Kindberg George Coulouris, Jean Dollimore. *Distributed Systems. Concepts And Design*. 4th edition, 2005.

[GH05] A. G
"unther and C. Hoene. Measuring round trip times to determine the distance between WLAN nodes. *NETWORKING 2005*, pages 768–779, 2005.

[Gha02] Dr. Ibrahim A. Ghaleb. Bluetooth - Connect Without Cables. Technical report, Electrical Engineering Department Faculty of Engineering Alexandria University, 2002.

[GO02] Ricardo Gutierrez-Osuna. Introduction to Pattern Recognition. Technical report, Wright State University, 2002.

[IEE01] IEEE. Ieee 802.3: Lan/man csma/cde (ethernet) access method. IEEE 802.3, IEEE, 2001. http://standards.ieee.org/getieee802/802.3.html; accessed October 31, 2010.

[IEE03] IEEE. Ieee posix certification authority. http://standards.ieee.org/regauth/posix/, 2003.

[Inc10] Apple Inc. Mac os x. http://www.apple.com/macosx/, 2010.

[KM10] G. Kremer and N. Melot. Mobility and synchronizations in wireless sensor networks. Technical report, TR LabSTICC/UBO, 2010.

[LDBL07] H. Liu, H. Darabi, P. Banerjee, and J. Liu. Survey of wireless indoor positioning techniques and systems. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, 37(6):1067–1080, 2007.

[LK02] Siu Man Lui and Sai Ho Kwok. Interoperability of peer-to-peer file sharing protocols. 2002.

[MC02] Y. Charlie Hu Antony Rowstron Miguel Castro, Peter Druschel. Topology-aware routing in structured peer-to-peer overlay networks. 2002.

[Mil10] Millennialmedia. State of the apps industry report. Octover 2010. `http://www.millennialmedia.com/research/stateoftheapps/appsthanks/`.

[Mog84] Jeffrey Mogul. Broadcasting internet datagrams. RFC 919, 1984. `http://tools.ietf.org/rfc/rfc919.txt`; accessed October 31, 2010.

[Nes99] Dan Nessett. Massively distributed systems: Design issues and challenges. 1999.

[Olz10] Tom Olzak, January 2010.

[PP] Souradyuti Paul and Bart Preneel. Analysis of non-fortuitous predictive states of the rc4 keystream generator.

[QL02] Edith Cohen Kai Li Scott Shenker Qin Lv, Pei Cao. Search and replication in unstructured peer-to-peer networks. 2002.

[RJB99a] J. Rumbaugh, I. Jacobson, and G. Booch. *The Unified Modeling Language. Reference Manual*. 1st edition, 1999.

[RJB99b] J. Rumbaugh, I. Jacobson, and G. Booch. *The Unified Modeling Language. User Guide*. 1st edition, 1999.

[RRA] A. Shamir R.L. Rivest and L. Adleman. A method for obtaining digital signatures and public-key cryptosystem.

[SAST09] P. Saint-Andre, K. Smith, and R. Tronçon. *XMPP: The Definitive Guide*. O'Reilly Media, Inc., 2009.

[SCa] Marc Krochmal Stuart Cheshire.

[SCb] Marc Krochmal Stuart Cheshire.

[Sch96] Bruce Schneier. Applied cryptography. 1996.

[Sim90] B. Simons. An overview of clock synchronization. *Fault-Tolerant Distributed Computing*, pages 84–96, 1990.

[SoIT00] C. Simpson and ERIC Clearinghouse on Information & Technology. Internet Relay Chat. *Educational Media and Technology Yearbook*, pages 62–65, 2000.

[STK05] AH Sayed, A. Tarighat, and N. Khajehnouri. Network-based wireless location: challenges faced in developing techniques for accurate wireless location information. *Signal Processing Magazine, IEEE*, 22(4):24–40, 2005.

[Sym10] Symbian. Symbian. http://www.symbian.org/, October 2010.

[Uni10] Google Code University. Introduction to distributed system design. http://code.google.com/edu/parallel/dsd-tutorial.html, October 2010.

[VC74] Carl Sunshine Vinton Cerf, Yogen Dalal. Specification of internet transmission control program. RFC 675, December 1974. http://tools.ietf.org/html/rfc675; accessed November 6, 2010.

[vH] Arthur van Hoff.

[web84] Ansi/ieee std. 830-1984. http://www.cvc.uab.es/shared/teach/a20363/IEEE.pdf, 1984.

[web10a] Android. http://www.android.com/, 2010.

[web10b] Android ndk. http://developer.android.com/sdk/ndk/index.html, 2010.

[web10c] Android sdk. http://developer.android.com/sdk/index.html, 2010.

[web10d] Android security and permissions. http://developer.android.com/guide/topics/security/security.html, November 2010.

[web10e] Android: What is android? http://developer.android.com/guide/basics/what-is-android.html, 2010.

[web10f] Eclipse ide. http://www.eclipse.org/, October 2010.

[web10g] The h open source. http://www.h-online.com/open/news/item/Report-Android-Market-surpasses-100-000-apps-1050505.html, Agust 2010.

[web10h] Wikipedia. RSS. http://en.wikipedia.org/wiki/RSS, December 2010.

[web10i] Wikipedia. Wi-Fi. http://en.wikipedia.org/wiki/Wi-Fi, December 2010.

[Wika] Wikipedia. Advanced encryption standard. http://en.wikipedia.org/wiki/Advanced_Encryption_Standard.

[Wikb] Wikipedia. Block cipher modes of operation. http://en.wikipedia.org/wiki/Cipher_block_chaining.

[Wikc] Wikipedia. Transport layer security. http://en.wikipedia.org/wiki/Secure_Sockets_Layer.

[YL04] Li Xiao Lionel M. Ni Xiaodong Zhang Yunhao Liu, Xiaomei Liu. Location-aware topology matching in p2p systems. 2004.

[ZP07] O. Zheng and J. Poon. Security Analysis of Microsoft Notification Protocol. 2007.

APPENDIX

APPENDIX A. REQUIREMENTS DOCUMENT

A.1. Functional Requirements

FRQ-0001	**Show news**
Version	1.0 (14/09/2010)
Dependencies	None
Description	The system shall show the user news from the department.
Comments	Cited in the original project proposal document.

Table A.1: FRQ-0001

FRQ-0002	**Show schedules for courses**
Version	1.0 (14/09/2010)
Dependencies	None
Description	The system shall allow the user to consult schedules for the courses she selects.
Comments	Cited in the original project proposal document.

Table A.2: FRQ-0002

FRQ-0003	**Add schedules to calendar**
Version	1.0 (14/09/2010)
Dependencies	[FRQ-0002] Show schedules for courses
Description	The system shall allow the user to add the schedules she is consulting to her phone calendar.
Comments	Cited in the original project proposal document.

Table A.3: FRQ-0003

FRQ-0004	**Show courses information**
Version	1.0 (14/09/2010)
Dependencies	None
Description	The system shall allow the user to consult information about courses or show links to this information.
Comments	Cited in the original project proposal document.

Table A.4: FRQ-0004

FRQ-0005	**Show map of the department**
Version	1.0 (14/09/2010)
Dependencies	None
Description	The system shall allow the user to consult a map of the department.
Comments	Cited in the original project proposal document.

Table A.5: FRQ-0005

FRQ-0006	**Show map of the campus**
Version	1.0 (14/09/2010)
Dependencies	None
Description	The system shall allow the user to consult a map of the campus.
Comments	Cited in the original project proposal document.

Table A.6: FRQ-0006

FRQ-0007	**Show FAQ**
Version	1.0 (14/09/2010)
Dependencies	None
Description	The system shall allow the user to consult a FAQ.
Comments	Cited in the original project proposal document.

Table A.7: FRQ-0007

FRQ-0008	**Show indoors position**
Version	1.0 (14/09/2010)
Dependencies	[FRQ-0005] Show map of the department
Description	The system shall allow the user consult her indoors position.
Comments	Cited in the original project proposal document.

Table A.8: FRQ-0008

FRQ-0009	**Contact students**
Version	1.0 (14/09/2010)
Dependencies	None
Description	The system shall allow the teacher to contact the students in her course that are on campus.
Comments	Cited in the original project proposal document.

Table A.9: FRQ-0009

FRQ-0010	**Chat**
Version	1.0 (14/09/2010)
Dependencies	None
Description	The system shall allow the user to have in-house chat with other logged in users who are at the department.
Comments	Cited in the original project proposal document.

Table A.10: FRQ-0010

A.2. Non-Functional Requirements

NFR-0001	**Platform**
Version	1.0 (14/09/2010)
Dependencies	None
Description	The system shall run on a smartphone operating system.
Comments	Established at an early phase, in agreement with supervisors.

Table A.11: NFR-0001

NFR-0002	**News feed**
Version	1.0 (14/09/2010)
Dependencies	[FRQ-0001] Show news
Description	The system shall consult news from the department's RSS feed.
Comments	Cited in the original project proposal document.

Table A.12: NFR-0002

NFR-0003	**Routing table protection**
Version	1.0 (06/11/2010)
Dependencies	None
Description	The system shall include a protection technic to avoid the routing table being modified by unauthorized elements (users and software).
Comments	Not cited in the original project proposal document.

Table A.13: NFR-0003

NFR-0004	**Reliable message delivery**
Version	1.0 (06/11/2010)
Dependencies	[FRQ-0009] Contact students [FRQ-0010] Chat
Description	The system shall guarantee that: • messages sent by a certain user, are actually from that user • messages will not be intercepted (accessed) by an unauthorized entity • legitimate messages are not modified by an unauthorized entity
Comments	Not cited in the original project proposal document.

Table A.14: NFR-0004

NFR-0005	**Authorized users**
Version	1.0 (06/11/2010)
Dependencies	[FRQ-0009] Contact students [FRQ-0010] Chat [NFR-0004] Reliable message delivery
Description	The system shall confirm that a user who wants to access certain functionalities, is a reliable user.
Comments	Not cited in the original project proposal document.

Table A.15: NFR-0005

NFR-0006	**Password protection**
Version	1.0 (07/11/2010)
Dependencies	[NFR-0005] Authorized users
Description	The system shall protect user passwords by means of a cryptographic system in order to provide a minimum level of privacy for them.
Comments	Not cited in the original project proposal document.

Table A.16: NFR-0006

NFR-0007	**Connection loss detection**
Version	1.0 (08/11/2010)
Dependencies	None
Description	The system shall detect when a user looses her connection to the system in less than 1 minute since that happened.
Comments	Not cited in the original project proposal document.

Table A.17: NFR-0007

A.3. Use Case Event Flows

UC-0001	**Login**	
Dependencies	None	
Description	The system shall behave as described in the following use case when the user wants to access to the system.	
Precondition	The user is not logged in.	
Ordinary sequence	**Step**	**Action**
	1	Actor Not Authenticated User (ACT-0003) chooses the login option on her smartphone.
	2	The system requests the user a username and a password.
	3	Actor Not Authenticated User (ACT-0003) inserts the required username and password.
	4	The systems checks that username and password are correct and shows the user her corresponding interface according to her role (teacher or student), and the use case finishes.
Postcondition	The user is logged in.	
Exceptions	**Step**	**Action**
	4	If the user inserts a wrong username or a wrong password, the system shows an error message and requests again the user a username and a password, then this use case resumes.
Comments	Use case identified in an early phase, not cited in the original project proposal document.	

Table A.18: UC-0001

UC-0002	**Logout**	
Dependencies	None	
Description	The system shall behave as described in the following use case when the user wants to close her session.	
Precondition	The user is logged in.	
Ordinary sequence	**Step**	**Action**
	1	Actor Student (ACT-0001) chooses the logout option on her smartphone.
	2	The system closes the session and shows the login view, and the use case finishes.
Postcondition	The user is logged out.	
Comments	Use case identified in an early phase, not cited in the original project proposal document. This use case can be performed by both student and teacher.	

Table A.19: UC-0002

UC-0003	**Consult news**
Dependencies	[FRQ-0001] Show news
Description	The system shall behave as described in the following use case when the user wants to consult the department news.
Precondition	-
Ordinary sequence	**Step** / **Action** 1 — Actor Student (ACT-0001) chooses the option to consult the news. 2 — The system requests for the information needed and shows it to the user, and the use case finishes.
Postcondition	-
Comments	This use case can be performed by both student and teacher.

Table A.20: UC-0003

UC-0004	**Consult schedule**
Dependencies	[FRQ-0002] Show schedules for courses [FRQ-0003] Add schedules to calendar
Description	The system shall behave as described in the following use case when the user wants to consult schedules for a course.
Precondition	-
Ordinary sequence	**Step** / **Action** 1 — Actor Student (ACT-0001) chooses the option to consult schedules. 2 — The system shows a list of the courses the user can consult. 3 — Actor Student (ACT-0001) chooses a course. 4 — The system shows the user the schedule for the present day. 5 — If user selects the option to add the consulted schedule to calendar, the system adds the schedule to her phone calendar, and the use case finishes.
Postcondition	-
Alternative sequence	**Step** / **Action** 4 — If the user wants to consult the schedule for the following days, the system shows the schedules for the requested days, then this use case resumes.
Comments	This use case can be performed by both student and teacher.

Table A.21: UC-0004

UC-0005	Consult course information
Dependencies	[FRQ-0004] Show courses information
Description	The system shall behave as described in the following use case when the user wants to consult information about a course.
Precondition	-
Ordinary sequence	**Step** / **Action** 1: Actor Student (ACT-0001) chooses the option to consult information about courses. 2: The system shows a list of the courses the user can consult. 3: Actor Student (ACT-0001) chooses a course from the list. 4: The system requests the information needed and shows it to the user, and the use case finishes.
Postcondition	**-**
Comments	This use case can be performed by both student and teacher.

Table A.22: UC-0005

UC-0006	Consult map
Dependencies	-
Description	The system shall behave as described in the following use case when the user requests the system to consult a map.
Precondition	-
Ordinary sequence	**Step** / **Action** 1: Actor Student (ACT-0001) requests the system a map. 2: The system asks the user what type of map she wants to consult. 3: Actor Student (ACT-0001) selects the type of map she wants to consult. 4: The system shows the requested map, and the use case finishes.
Postcondition	-
Comments	This use case can be performed by both student and teacher.

Table A.23: UC-0006

UC-0007	**Consult campus map**
Dependencies	[FRQ-0006] Show map of the campus
Description	The system shall behave as described in the following use case when the user requests the system to consult a map of the campus.
Precondition	-
Ordinary sequence	**Step** / **Action** 1: Actor Student (ACT-0001) requests the system a map. 2: The system asks the user what type of map she wants to consult. 3: Actor Student (ACT-0001) selects the map of the campus. 4: The system shows the map of the campus, and the use case finishes.
Postcondition	-
Comments	This use case can be performed by both student and teacher.

Table A.24: UC-0007

UC-0008	**Consult department map**
Dependencies	[FRQ-0005] Show map of the department [FRQ-0008] Show indoors position
Description	The system shall behave as described in the following use case when the user requests the system to consult a map of the department.
Precondition	-
Ordinary sequence	**Step** / **Action** 1: Actor Student (ACT-0001) requests the system a map. 2: The system asks the user what type of map she wants to consult. 3: Actor Student (ACT-0001) selects the map of the department. 4: The system shows the map of the department. 5: If the user wants to consult her position on the map, the system requests for the information needed and shows the user her position, and the use case finishes.
Postcondition	-
Comments	This use case can be performed by both student and teacher.

Table A.25: UC-0008

UC-0009	Consult FAQ
Dependencies	[FRQ-0007] Show FAQ
Description	The system shall behave as described in the following use case when the user wants to consult the FAQ.
Precondition	-
Ordinary sequence	**Step** / **Action**
	1: Actor Student (ACT-0001) chooses the option to consult the FAQ.
	2: The system shows an index of existing FAQ.
	3: Actor Student (ACT-0001) selects the FAQ she wants to consult.
	4: The system shows the detailed FAQ requested, and the use case finishes.
Postcondition	-
Comments	This use case can be performed by both student and teacher.

Table A.26: UC-0009

UC-0010	Use chat
Dependencies	[FRQ-0010] Chat
Description	The system shall behave as described in the following use case when the user wants to chat with other users at the department.
Precondition	The user is logged in.
Ordinary sequence	**Step** / **Action**
	1: Actor Student (ACT-0001) chooses the option to access the chat.
	2: The system shows the chat view and a list of the logged in users.
	3: If user wants to write on general chat, the system displays the message written by the user on general chat board.
	4: If user wants to write another certain user who is also logged in the chat, the system displays the message written by the user only for the addressed one, and the use case finishes.
Postcondition	-
Comments	This use case can be performed by both student and teacher.

Table A.27: UC-0010

UC-0011	**Contact students**
Dependencies	[FRQ-0009] Contact students
Description	The system shall behave as described in the following use case when a teacher wants to contact the students in her course.
Precondition	The user is logged in.
Ordinary sequence	**Step** / **Action**
	1 — Actor Teacher (ACT-0002) selects the option to contact students.
	2 — The system shows a list of the teacher's course.
	3 — Actor Teacher (ACT-0002) selects a course from the list.
	4 — The system shows a list of the students in that course who are at the department at that moment.
	5 — If there is any student at the department, actor Teacher (ACT-0002) writes a message to be sent.
	6 — The system shows the message to the addressed students, and the use case finishes.
Postcondition	-
Exceptions	**Step** / **Action**
	4 — If there is not any student at the department, actor Teacher (ACT-0002) cannot send a message, then this use case is cancelled.

Table A.28: UC-0011

APPENDIX B. USE CASE REALIZATIONS DOCUMENT

B.1. Use Case Realizations: Sequence Diagrams

B.1.1. [UC-0001] Login

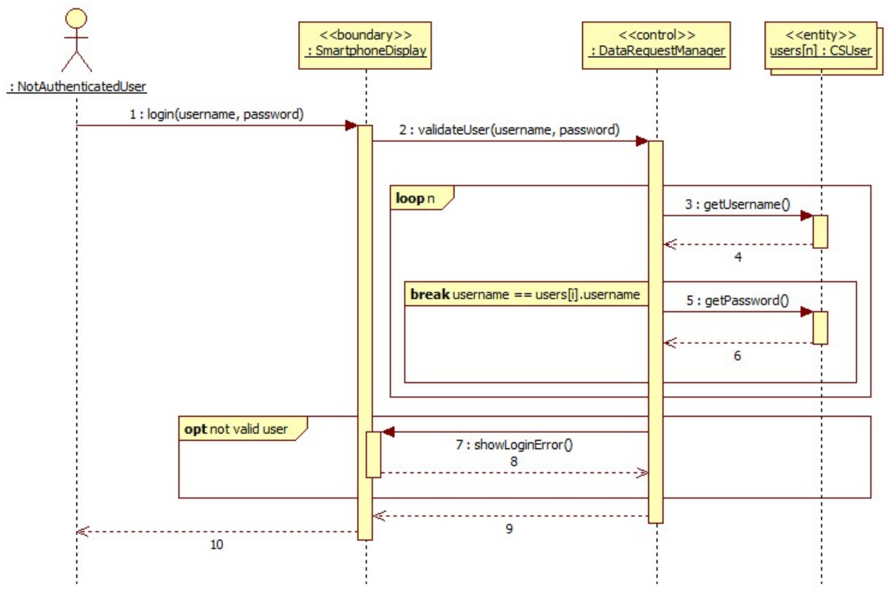


Figure B.1: Design - UC-0001 sequence diagram

B.1.2. [UC-0003] Consult news

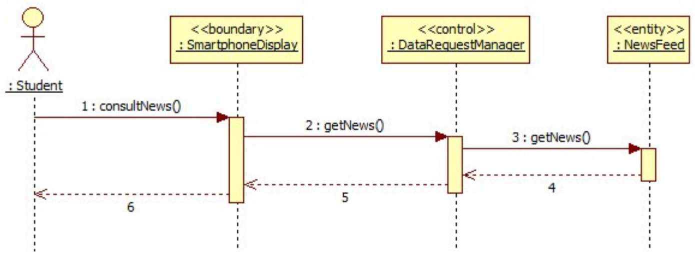
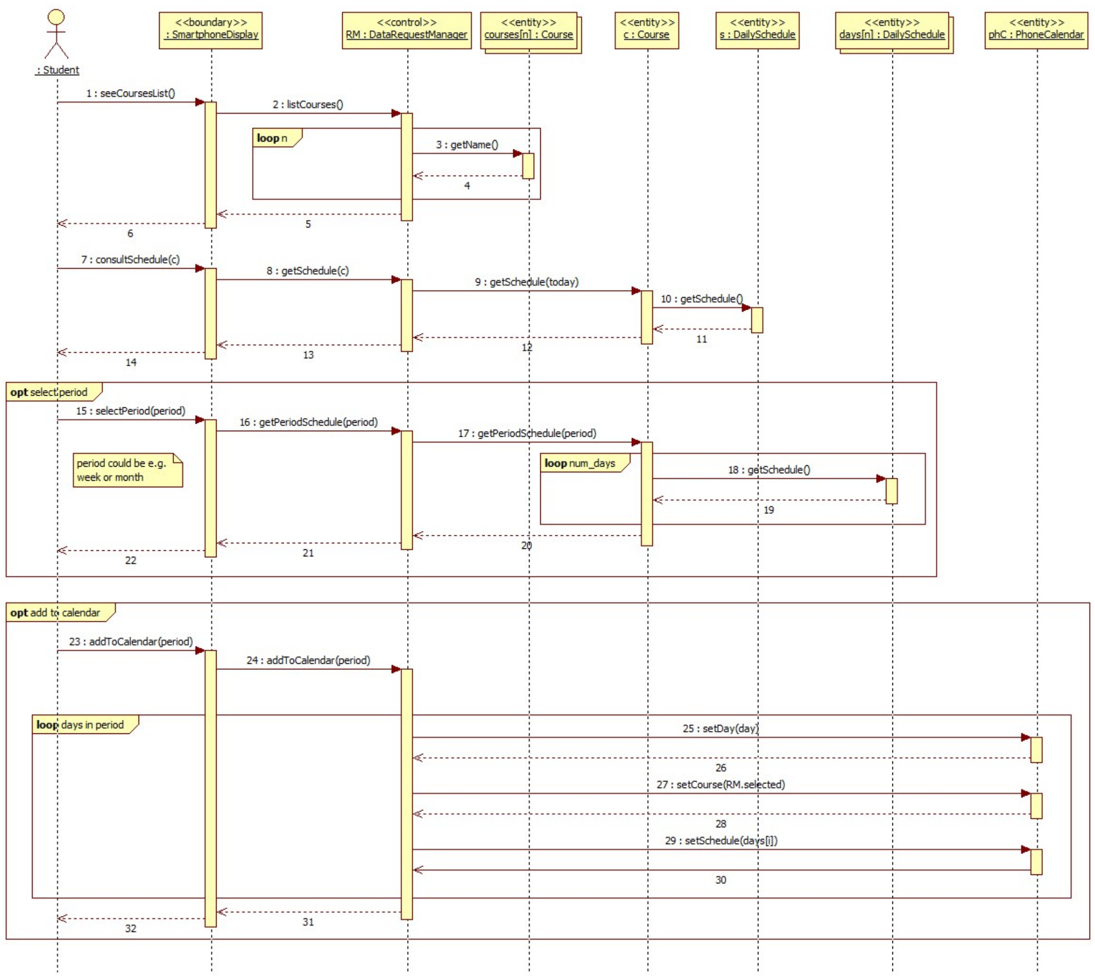


Figure B.2: Design - UC-0003 sequence diagram

B.1.3. [UC-0004] Consult schedule



: Student
<<boundary>> : SmartphoneDisplay
<<control>> RM : DataRequestManager
<<entity>> courses[n] : Course
<<entity>> c : Course
<<entity>> s : DailySchedule
<<entity>> days[n] : DailySchedule
<<entity>> phC : PhoneCalendar
1 : seeCoursesList()
2 : listCourses()
loop n
3 : getName()
4
5
6
7 : consultSchedule(c)
8 : getSchedule(c)
9 : getSchedule(today)
10 : getSchedule()
11
12
13
14
opt select period
15 : selectPeriod(period)
16 : getPeriodSchedule(period)
17 : getPeriodSchedule(period)
period could be e.g. week or month
loop num_days
18 : getSchedule()
19
20
21
22
opt add to calendar
23 : addToCalendar(period)
24 : addToCalendar(period)
loop days in period
25 : setDay(day)
26
27 : setCourse(RM.selected)
28
29 : setSchedule(days[i])
30
31
32

Figure B.3: Design - UC-0004 sequence diagram

B.1.4. [UC-0005] Consult course information

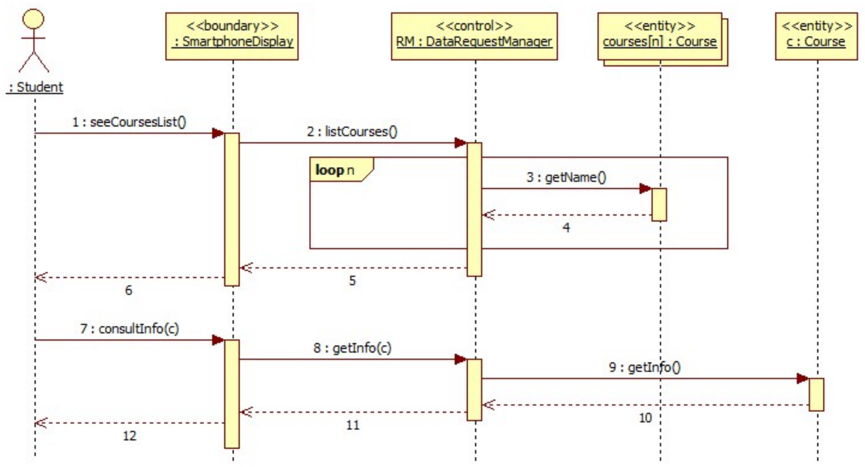


Figure B.4: Design - UC-0005 sequence diagram

B.1.5. [UC-0008] Consult department map

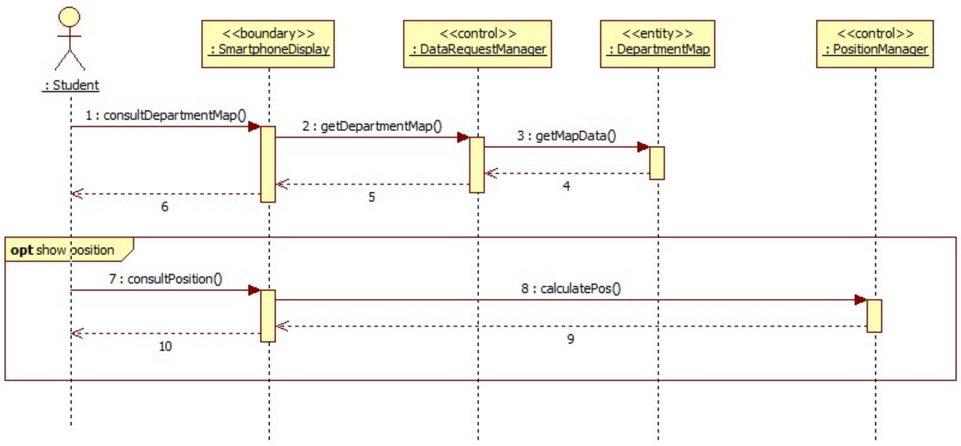


Figure B.5: Design - UC-0008 sequence diagram