uc3m | Universidad Carlos III de Madrid

University Degree in Computer Science and Engineering
Academic Year 2017-2018

Bachelor Thesis

Development of an Android Medical Application to Classify Patient's Symptoms by Means of Machine Learning Algorithms

Author: Francisco Javier Lovelle Moraleda

Tutor: Dr. David Griol Barres

Leganés, September 22nd, 2017

Acknowledgements

With this project finalizes the university degree that allows me to continue advancing in my studies and professional life. Due to this reason, I would like to take this section to thank all those people who have helped me in the realization of this bachelor thesis and university degree.

I would like to thank my tutor in this bachelor thesis, David Griol Barres, for a valuable guidance and encouragement.

I would like to thank my friends, inside and outside the university degree, that have encouraged and helped me to surpass each of the milestones in the university degree.

I take this opportunity to sincerely thank my family for helping me choosing the studies that I love, giving me the courage to keep going in the most difficult times and celebrating each of the milestones in the university degree.

I also wish to thank all who directly or indirectly have helped throughout the university degree.

Abstract

The smartphone has been one of the most revolutionary inventions in this last decade and, with this technology, the population has changed the way it searches for information.

The field of medicine has also adapted to the arrival of this technology with the release of hundreds of applications that let the patients, not only communicate with doctor, but also help them recognize the diseases related to their condition.

However, even though there are a lot of types of medical applications of recognition of symptoms, the possibility of studying the symptoms of patients has not been studied enough through the analysis of the text introduced in a vocal manner with techniques of speech recognition and machine learning.

In this project is described the development of an application that allows the patient to classify the introduced symptoms with voice recognition libraries and machine learning techniques.

At the beginning of the project the introduction and objective of the project is proposed, besides presenting the necessary budget for the system to be elaborated.

Then, the different mobile platforms on the market, along with medical applications in these platforms and voice recognition libraries, are studied. Also, the technologies that have been used for the project development of the project, along with working environment used, are discussed

To continue, its detailed the data protection law that governs in Spain, where the reason behind the impossibility of publication of the system is detailed. Also, the software licenses of the technologies mentioned before.

After seeing the technologies that will be used in the development of the project and the limitations and guidelines that the data protection law imposes, the development of the system is related, where it is analyzed the use cases and requirements, to later present the architecture of the system.

Once the system is developed, the evaluation of the system is presented. With the help of personnel external to the system, it is proved for the system to show an average success rate higher than 50% at the time of classifying the symptoms of the patient through machine learning algorithms.

Keywords: Algorithm, Application, Machine learning, Library and Speech recognition.

Resumen

El teléfono inteligente ha sido una de las revoluciones más destacadas de esta última década y, con esta tecnología, ha cambiado la forma en la cual la población busca información.

El campo de la medicina también se ha adaptado a la llegada de esta tecnología sacando aplicaciones que permiten a los pacientes, no solo poder comunicarse con los doctores, pero también ayudar al paciente a reconocer enfermedades relacionadas con su condición.

Sin embargo, aunque existan muchos tipos de aplicaciones médicas de reconocimiento de síntomas, casi no se ha estudiado la posibilidad de estudiar los síntomas de los pacientes a través del análisis del texto introducido de manera vocal con técnicas de reconocimiento de voz y aprendizaje automático.

En este proyecto se desarrolla una aplicación que permite al paciente clasificar los síntomas introducidos con librerías de reconocimiento de voz a través de técnicas de aprendizaje automático.

Al principio del proyecto se plantea la introducción y objetivo del proyecto, además de presentar el presupuesto necesario para que el sistema sea elaborado.

Después se estudian las diferentes plataformas móviles existentes en el mercado, junto con aplicaciones médicas en estas plataformas y librerías de reconocimiento de voz. También se exponen las tecnologías que han sido usadas para la elaboración del proyecto además del entorno de trabajo utilizado.

A continuación, se detalla la ley de protección de datos que rige en España, y los motivos que restringen la publicación del sistema elaborado en este proyecto. También se presentan las licencias software de las tecnologías mencionadas anteriormente.

Tras haber visto las tecnologías que serán usadas y las limitaciones y directrices que impone la ley de protección de datos, se plantea el desarrollo del sistema, donde se analizarán los casos de uso y requisitos, para después presentar la arquitectura del sistema.

Una vez desarrollado el sistema, se pasará a ver la evaluación del mismo, a través de la colaboración de personal externo al proyecto, donde se comprobará de que, aun utilizando una pequeña base de datos, los resultados muestran una media de porcentaje de acierto superior al 50% a la hora de clasificar los síntomas del paciente con algoritmos de aprendizaje automático.

Palabras clave: Algoritmo, Aplicación, Aprendizaje automático, Librería and Reconocimiento de voz.

Index

Index of Figures

Index of Tables

Chapter 1

Introduction

In this chapter, some facts of the mobile phone industry are introduced. Then, the main objective of the project that takes advantage of the technological situation is described. To continue, the development phases, the resources and the budget of the project is shown. At the end of the chapter the structure of the rest of the document is described.

1.1. Introduction

Smartphones have changed the way society communicates and gets information, making this device one of the most revolutionary creations of the last decade. Nowadays there are more than 2.32 billion people in the world using smartphones as a daily device and it is expected that by 2020 there will be more than 2.87 billion smartphone users (Statista, 2017a), making the smartphone a great device to communicate, inform and help more than one third of the world population (Statista, 2017b).

Due to the fast growth of the smartphone sector and given the close relation existing between technological progress and medicine, more and more people attempt to look for medical information on the internet through their smartphones. This phenomenon known as mHealth or m-Health cover fields like patient care and monitoring, health applications, and education and research articles (Ahuja, N., Ozdalga, A. and Ozdalga, E., 2012).

In order to find information using the smartphone there are two major input interfaces, hands and voice. While the first one operates through a traditional keyboard, the second one is based on the use of a microphone with help of speech recognition systems.

With the integration of the speech recognition libraries from Apple, Google and Microsoft, the three major smartphone operating system manufacturers, it is getting easier, faster and more natural to communicate and search for information through the smartphone at the same time the speech recognition libraries get better.

Speech recognition systems have improved so much in the last years that according to a recent study, they are now not only faster but also more accurate that traditional typing methods in multiple languages. The mentioned study found that speech recognition produced text three times faster and with a twenty percent lower error rate than typing in English language. The results were even more impressive in Mandarin Chinese, as the voice recognition worked almost three times faster and had an error rate sixty three percent lower than typing (Carey, B., 2016).

1.2. Objective

The medical applications that are already in the mobile market has surpassed more than a million downloads, however, the text classification combined with speech recognition techniques is an area not yet exploited in medical applications.

There have been studies of self-reported medical conditions and how the accuracy varies depending on the disease the user has (Smith, B. et al., 2008). This project aims to improve the accuracy of self-reported diagnostics with the help of the mobile devices each time more extended.

The objective of this project is to develop an application that lets the user introduce the symptoms through speech recognition libraries and shows the user the set of diseases he or she may have by analyzing the symptom with text classification techniques.

1.3. Development Phases

The workflow that the bachelor thesis has during its development can be divided in the subsequent major categories.

Planning

This first phase consists in the study of the area of interest and the initial statement of the requirements of the project.

The area of study in this project can be divided in 3 different fields.

- The study of the medical applications that are found in the Google Play Store that provides Google as a resource of applications for Android.
- The software libraries used to develop an application in Android along with its environment, the speech recognition library offered by Google.
- The text classification techniques that are used nowadays along with software libraries that provides with these techniques to the project.

To continue, a study to find technologies that facilitate the development of the project will be made.

To finalize, the requirements that will provide the application with its functionality are declared. These requirements must compliment the study made earlier in the planning phase.

Execution

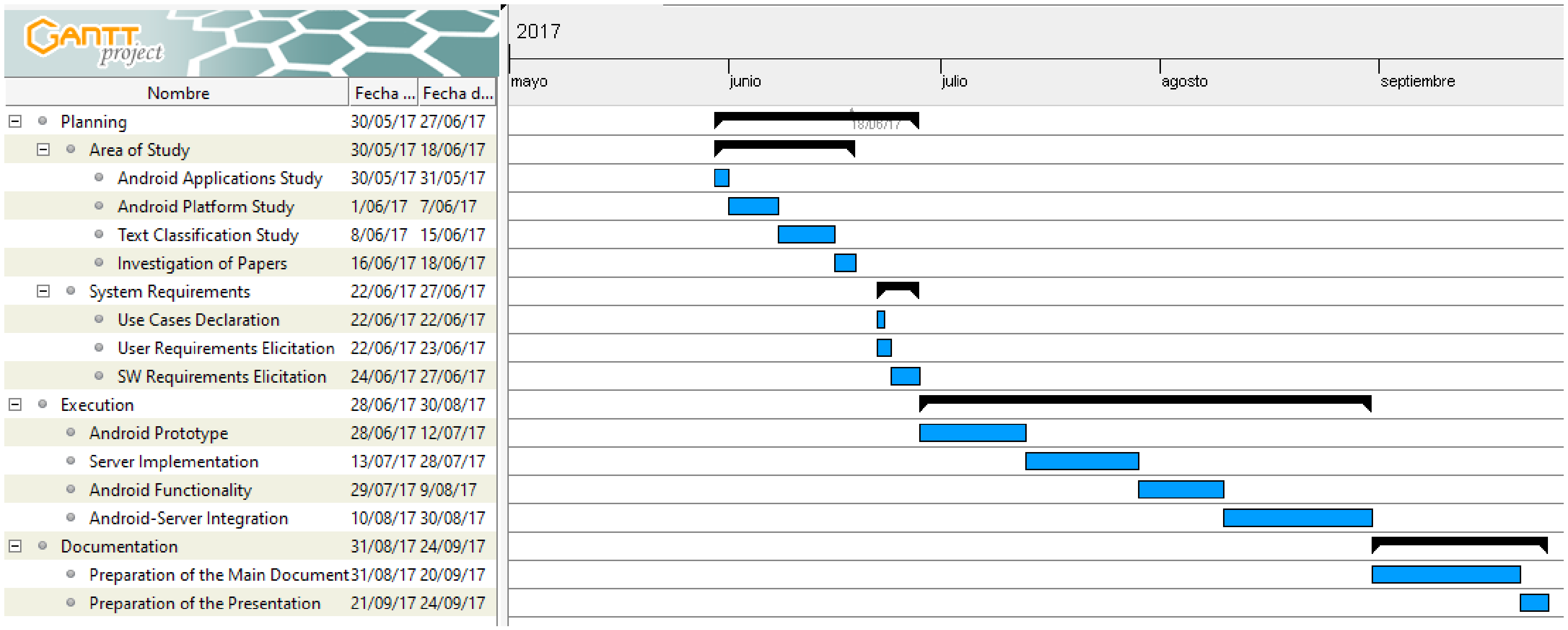
In this second phase the architecture, design and development of the application that fulfills the requirements defined in the planning phase is made along with the evaluation of the software implemented.

However, as the model that has been selected to develop the project is the prototyping model defined in the chapter 5.2., some requirements defined in the planning phase could be modified in this second phase.

Documentation

The final phase of the project is dedicated to build the documentation of the project, that involves the elaboration of this document and the presentation of the defense.

To better show the temporal planning that has been done during the development of this project, a Gantt diagram made with the free and open source program GanttProject is shown below in the Figure 1.



Nombre	Fecha ...	Fecha d...
Planning	30/05/17	27/06/17
Area of Study	30/05/17	18/06/17
Android Applications Study	30/05/17	31/05/17
Android Platform Study	1/06/17	7/06/17
Text Classification Study	8/06/17	15/06/17
Investigation of Papers	16/06/17	18/06/17
System Requirements	22/06/17	27/06/17
Use Cases Declaration	22/06/17	22/06/17
User Requirements Elicitation	22/06/17	23/06/17
SW Requirements Elicitation	24/06/17	27/06/17
Execution	28/06/17	30/08/17
Android Prototype	28/06/17	12/07/17
Server Implementation	13/07/17	28/07/17
Android Functionality	29/07/17	9/08/17
Android-Server Integration	10/08/17	30/08/17
Documentation	31/08/17	24/09/17
Preparation of the Main Document	31/08/17	20/09/17
Preparation of the Presentation	21/09/17	24/09/17

Figure 1. Gantt diagram with the development of the project

1.4. Resources

For the project, a set of resources has been necessary to implement and document this bachelor thesis.

Hardware

For the development of the project the laptop has been used as the server of the system to reduce the final budget. The hardware used to develop the system is the following.

- Laptop Toshiba Satellite L850.
- Mobile device Motorola Gen. 4.
- USB cable.

Software

The programs and tools used in this project are aimed to reduce the final budget of the system and to make use of the experience of the developers with the programs they feel comfortable with, without sacrificing development speed. These programs and tools used are the following.

- Android SDK.
- Android Studio.
- Debian.
- Google Drive.
- LibreOffice.
- Mozilla Firefox.
- MySQL.
- OpenJDK.

- Weka.
- Vim.
- Gantt Project

1.5. Budget of the Project

In this section, the total budget the project should count with is exposed given the total time spent in the project shown in section 1.3. and the resources used for the creation of this project presented in section 1.4.

1.5.1. Human Resources Costs

With the number of days worked in project, the average number of hours worked per day in the project and the average salary of a computer engineer, the total net salary of a person can be calculated.

- The total number of days the project has lasted has been 115 days, as shown in the Gantt diagram of the section 1.3. of this chapter. However, taking out weekends and the days not worked from the 19th to 21st of June, the total number of days worked in the project are 81.
- The average number of hours worked per day without weekends has been of 4 hours per day.
- Given the resolution of the 30th of December of 2016, "*Resolución de 30 de diciembre de 2016, de la Dirección General de Empleo, por la que se registra y publica el Convenio colectivo del sector de empresas de ingeniería y oficinas de estudios técnicos.*", the salary of a level 2 professional is of 17,544.24€ for 1800 hours, giving as a result 9.7468€ per hour.

To calculate the total salary per person per project, the following formula has to be applied.

$$\text{Days dedicated to the project} \times \text{Average hours per day to the project} \times \text{Salary per hour}$$

With a total of 81 days, 4 hours of work each day, and a salary of 9.7468€ per hour, it is obtained a total of 3,157.53€.

However, the Social Security costs dictated by the Spanish regulation have to be added to the total salary by increasing the previous result by a 31.55%.

The total cost of the project in human resources is **4,154.30€**.

1.5.2. Equipment Costs

For the equipment costs only the hardware resources are studied given that the software resources are free and do not sum up to the total costs of the project.

The hardware resources costs are shown below in Table 1.

Resource	Cost
Laptop Toshiba Satellite L850	749.95€
Mobile device Motorola Gen. 4	179.95€
USB Cable	4.95€

Table 1. Hardware resources costs

Each product of the table has a depreciation time assigned to it by the Spanish tax agency. The equipment to process information like laptops and mobile devices have 4 years of depreciation time, while cables have 15 years.

To calculate the total costs of the equipment the following formula must be applied.

$$\frac{\text{Cost of the product}}{\text{Depreciation time}} \times \text{Time of the project}$$

The Table 2 shows the total costs of the equipment.

Resource	Initial Cost	Time of the project in months	Total warranty in months	Total cost per product
Toshiba Satellite L850	749.95€	3.77	48	58.91€
Motorola Gen. 4	179.95€		48	14.13€
USB Cable	4.95€		180	0.10€
Total cost				73.14€

Table 2. Total equipment cost

The total cost of the project in equipment is **73.14€**.

1.5.3. Total Costs

To get the total budget needed for the implementation of the project, the human resources and the equipment costs have to be combined. Also, the 20% of the human resources should be sum up to cover indirect costs. Then, a 21% of increase to cover the value added tax as shown in the Table 3.

Description	Cost
Human resources	4,154.30€
Equipment	73.14€
Indirect costs	830.86€
Total cost (without V.A.T.)	5058.30€
Value Added Tax	1,062.24€
Total cost (with V.A.T.)	6,120.54€

Table 3. Total cost of the project

The total cost of the project is **6,120.54€**.

1.6. Economic Impact of the Project

Due to the lack of protection of the user data, the system is not yet prepared to be released by reason of the "*Ley Orgánica 15/1999, de 13 de diciembre, de Protección de Datos de Carácter Personal.*" law in the Spanish regulation. Given that the system is not yet prepared to be released, the only economic impact of this project is to reduce the costs of a future implementation of the complete system.

1.7. Social Impact of the Project

Due to the lack of protection of the user data, the system is not yet prepared to be released by reason of the "*Ley Orgánica 15/1999, de 13 de diciembre, de Protección de Datos de Carácter Personal.*" law in the Spanish regulation. Given that the system is not yet prepared to be released, the only social impact of this project is to reduce the research time of a future bachelor thesis.

1.8. Structure of the Document

In this first chapter of the project it has been shown how the smartphones have revolutionize the industry over the last decade and how new interfaces of communication are being used in everyday life. Then the final objective for this project has been introduced and, finally, the planning, technologies, budget and economic and social impact of the project have been shown.

In the second chapter of the project, the main mobile operating systems that are currently available are seen along with medical applications that are nowadays popular among users of the different mobile operating systems platforms. Also, some of the most known speech recognition APIs used by developers are presented.

In the third chapter of the project, the technologies used in the project are described and a justified reason behind the choice of the technologies selected is exposed.

In the fourth chapter, the Spanish regulation is shown. In this chapter is described the different laws that restricts the design of the system along with reason behind the impossibility of releasing the final product of the project. Also, the software licenses of the technology used is described.

In the fifth chapter is where the solution to the problem described in this first chapter is implemented. In this fifth chapter the use cases, user and software requirements, architectural design and class diagrams are presented.

In the sixth chapter, an evaluation of the system is shown, where the accuracy of the system is exhibited.

The final chapter of the document, the seventh chapter, summarizes what have been seen in the document along with some final thoughts. Also, future work is presented to improve system presented.

After the main chapters of the document are presented, an Annex with an extended description of the Android platform is displayed.

At the end of the document a glossary with abbreviations and principal terms used in the document, and a bibliography with the resources to the documents used in the development of this project are set out.

Chapter 2

State of the Art

In this chapter, it is shown the context in which the project is located.

To begin, the mobile operating systems that hold most of the market share are shown.

Then the applications of evaluation of patients through self-diagnosis with most downloads found in the mentioned mobile operating systems are list.

At the end of the chapter the speech recognition method is described and some popular libraries available to the public providing this technology are listed.

2.1. Mobile Operating Systems

The mobile operating systems are operating systems aimed to run in devices with low power consumption like smartphones, tablets, smartwatches and even some small laptops. However, with the increasing computing power in the mobile devices, some personal computer operating systems features have been included over the last years along with some characteristics that are nowadays essential in mobile operating systems like touchscreen, cellular network and Wi-Fi.

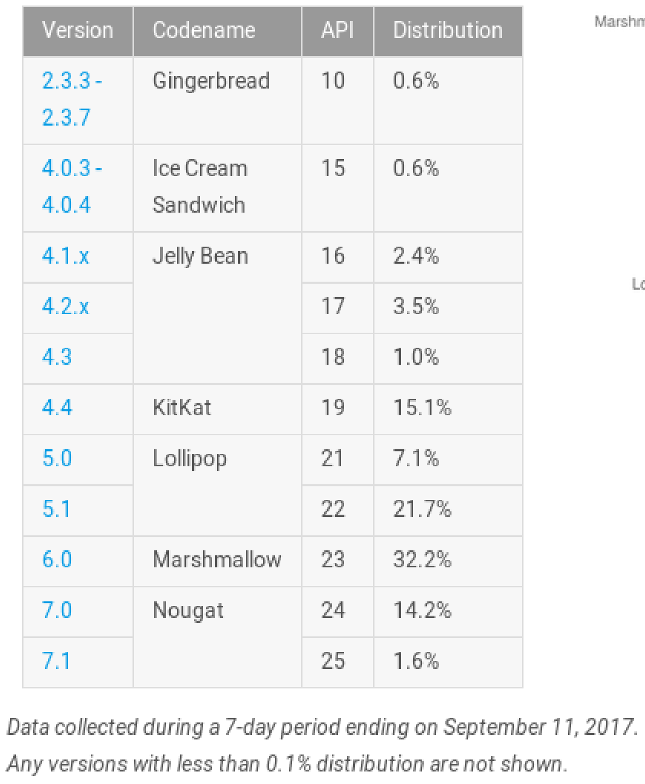
Over the last decade three operating systems have risen to occupy more than the 99% of the mobile operating systems market share, Android by Google, iOS by Apple and Windows Mobile by Microsoft (Vincent, J., 2017).

In the next section, the three operating systems that currently have the highest percentage of market share (Android, iOS and Windows Mobile) are briefly described.

2.1.1. Android

Android is an open source mobile operating system implemented from the Linux kernel by the Android Inc company. Bought by Google in 2005, the Android operating system has already more than the 80% of the global market share (Vincent, J., 2017).

Android has eight major releases, with the last one, Android 8.0 Oreo, being unveiled the 21st of August. However, as shown in Figure 2, even though Android 8.0 is the latest mobile operating system made by Google, the information provided by Google shows that Android 6.0 Marshmallow and Android 5.1 and 5.0 Lollipop are the mobile operating systems with more users in the Android platform (Google Inc, 2017).



Version	Codename	API	Distribution
2.3.3 - 2.3.7	Gingerbread	10	0.6%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	0.6%
4.1.x	Jelly Bean	16	2.4%
4.2.x		17	3.5%
4.3		18	1.0%
4.4	KitKat	19	15.1%
5.0	Lollipop	21	7.1%
5.1		22	21.7%
6.0	Marshmallow	23	32.2%
7.0	Nougat	24	14.2%
7.1		25	1.6%

Data collected during a 7-day period ending on September 11, 2017.
Any versions with less than 0.1% distribution are not shown.

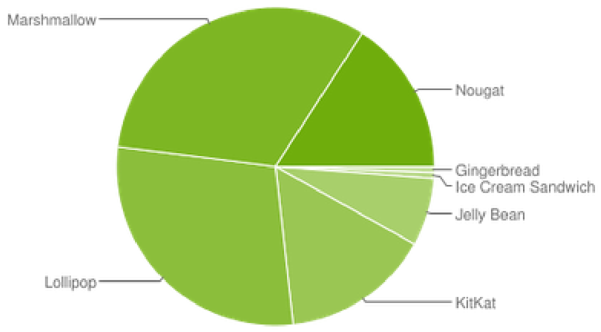


Figure 2. Google mobile operating systems percentage of users

Android does not only run in mobile devices but also in tablets, chromebooks, smartwatches, televisions, handheld game devices and single board computers.

For more information about the Android mobile operating system structure, its components and the SDK go to the Annex A.

2.1.2. iOS

iOS is the operating system that was created and developed by Apple Inc to launch along with its first smartphone “iPhone” in 2007. Nowadays iOS has more than the 15% of the global market share (Vincent, J., 2017).

The latest version of iOS is iOS 11. Available from the 19th of September of 2017, iOS 11 attempts to replace iOS 10 by making it compatible from the iPhone 5s to the very new iPhone X (Apple Inc, 2017a).

At the current moment, given that iOS is not currently in the market, the version of iOS with most percentage of users is iOS 10 with a 89% of users, followed by iOS 9 with a 9% of users as Figure 3 shows (Apple Inc, 2017b).

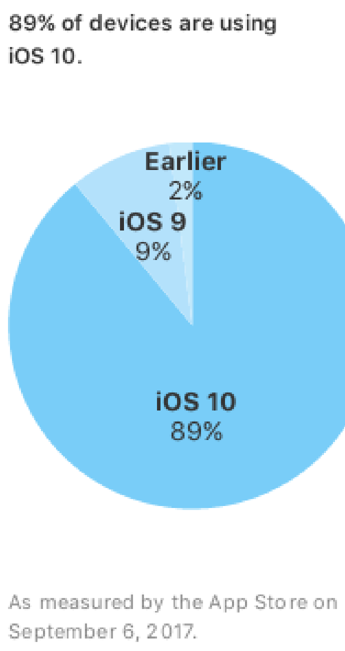


Figure 3. Apple mobile operating systems percentage of users

2.1.3. Windows Mobile

During the past years Microsoft has been looking forward to the convergence of the mobile and desktop devices. The successor of Windows Phone 8.1, Windows 10 Mobile is the current mobile operating system developed by Microsoft that aims to achieve this final unification, where the Windows 10 devices such PCs, mobile devices and Xbox run an universal platform that allows the user to run the same application no matter the device.

However, the Microsoft mobile operating system is not able to surpass the 1% market share, positioning the operating system at the same level BlackBerry one year before (Vincent, J., 2017).

Even though Windows 10 mobile is the latest mobile operating system from Microsoft, Windows Phone 8.1 is still the most used mobile operating system from Microsoft with more than 80% of users as shown in Figure 4 (Statista, 2016).

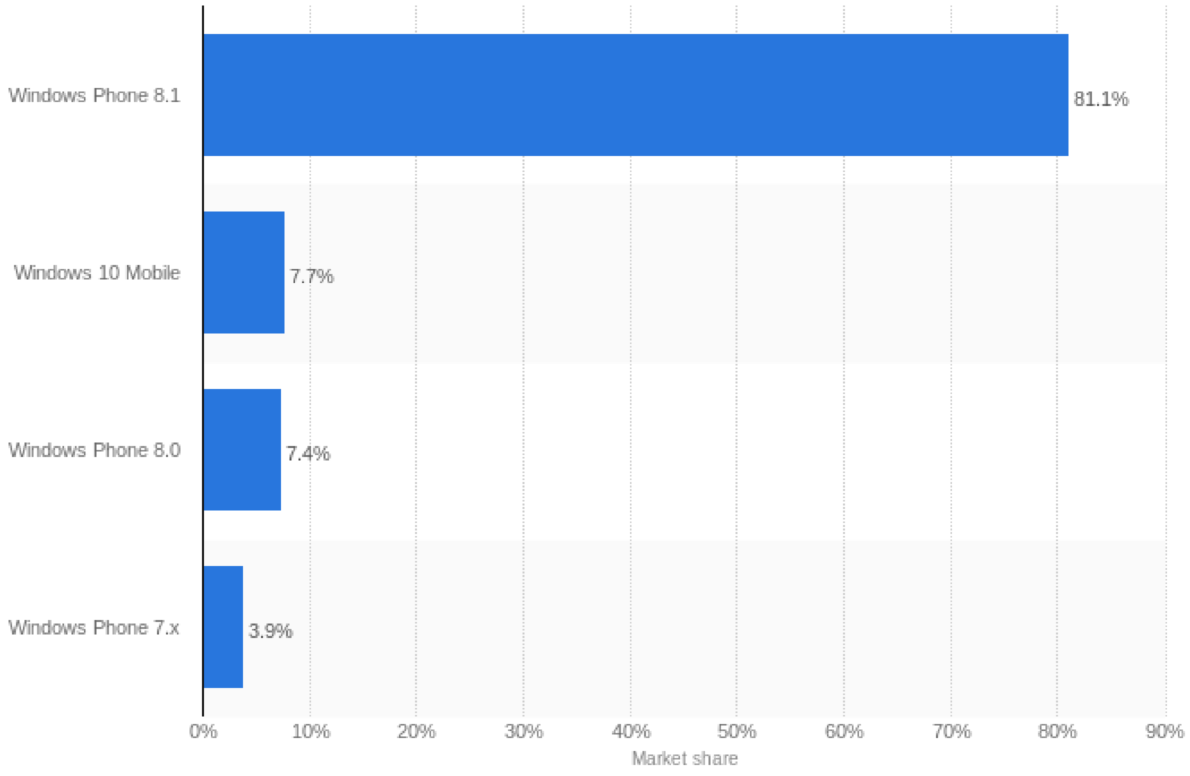


Figure 4. Microsoft mobile operating systems percentage of users

2.2. Patient Evaluation Applications

In the Android and iOS application stores can be found lots of medical applications, in this section are listed the applications of evaluation of patients through self-diagnosis most downloads in both platforms.

2.2.1. WebMD

WebMD is an application that provides the user with health information and decision support tools that lets the user analyze its symptoms to improve its health state. The Figure 5 shows the initial screen of the WebMD application.

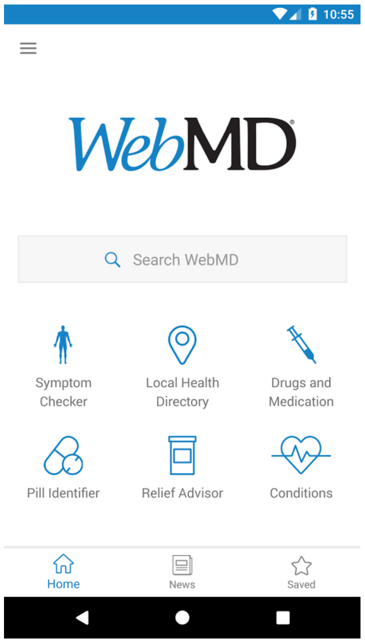


Figure 5. *WebMD* Android application main menu

The main features of the *WebMD* application are the following.

- **Symptom checker.** The patient can select the part of the body troubles it in order to show possible conditions.
- **Conditions.** The user can find medical information with diseases, treatments and symptoms.

- **Drugs and treatments.** A database with information about pills, drugs and supplements can be accessed by the user to access information about them.
- **Identification of pills.** A tool to identify the prescripted pills and drugs is provided to the user.
- **Local health listings.** A set of doctors, pharmacies and hospitals near the user can be accessed from the application.
- **Data storage.** The user can save all the information about conditions, drugs and articles to later read them anywhere and anytime.

The interface of the *WebMD* application opts to let the user introduce the symptoms of the patient by selecting the body part that troubles the patient as shown in Figure 6.

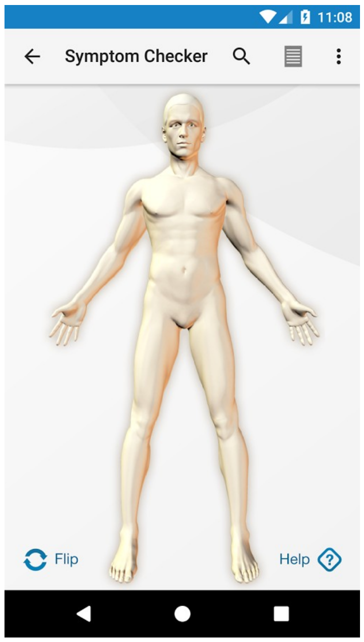


Figure 6. WebMD Android application symptom introduction

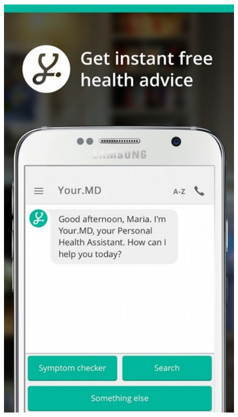
WebMD Health Corp. is the company that has implemented the *WebMD* application for Android and for iOS. *WebMD Health Corp*. provides its users with credible information, supportive communities and reference materials about health subjects (WebMD Health Corp., 2014).

2.2.1. Your.MD: Health Care Assistant

Your.MD: Health Care Assistant is an application that through text analysis helps the user evaluate the symptoms previously introduced through the keyboard. The main features of the application are the following.

- **Symptom checker.** Through a chat, the user can introduce the symptoms to let the symptom classifier of *Your.MD* check the disease you might have. *Your.MD*, after classifying the symptoms sends the user a battery of questions for it to respond.
- **Health tracking.** The application lets the user monitor its health over time using of charts.
- **Find health information.** If the user does not want to introduce its symptoms, it can search information by selecting the different diseases and checking information about it and possible treatments.
- **Doctor search.** *Your.MD* lets the user find a doctor to agree in a medical appointment.
- **Data storage.** The user can store information that has been retrieved to check it later.

The interface of the *Your.MD: Health Care Assistant* application lets users introduce the symptoms with the keyboard through a chat like interface as shown in Figure 7.



Good afternoon, Maria. I'm Your.MD, your Personal Health Assistant. How can I help you today?

Figure 7. Your.MD: Health Care Assistant symptom checker

2.2.2. Ada – Your Health Companion

Ada is an application that, as *Your.MD: Health Care Assistant* does, also opts to evaluate the symptoms of the patient introduced through the Android keyboard by looking if those symptoms are in the database. However, unlike *Your.MD* application, *Ada* does not provide analysis through text classification and requires an email account to start using its services. The main features *Ada* provides are following.

- **Symptom checker.** As *Your.MD* did, *Ada* provides a chat interfaces for the user to introduce its symptom. Once the main symptom has been selected by the user through the chat interface, *Ada* provides a battery of questions the user must respond to evaluate the user condition.
- **User data storage.** The user can store data like its name, age, height, weight, and data as its medicines and allergies for the application to provide the user better results.
- **User monitorization.** The application lets the user track its evolution over time with charts.

In the same way Your.MD interface worked, Ada also provides a chat interface for the user for the introduction of symptoms and condition retrieval as shown in Figure 8. However, Ada provides a much cleaner interface, with less buttons to navigate.

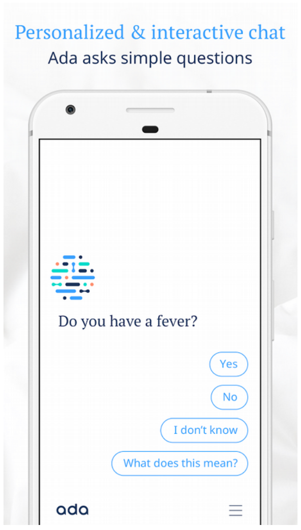


Figure 8. Ada – Your Health Companion Symptom Checker

2.3. Speech Recognition

Speech recognition is the field in charge of all the technologies and methods that let the user use the voice as a computer input interface. The speech recognition main purpose is to translate the spoken phrases into text.

As shown in Figure 9, speech recognition software processes the stream of data that is received from the microphone, and process the stream into values that will be introduced into the speech recognition software classifier to retrieve the text string the user has introduced through its voice.

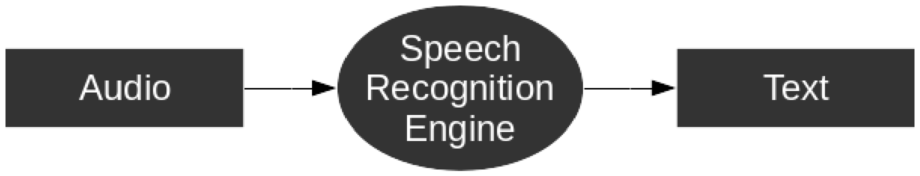


Figure 9. Speech recognition engine input and output

Over the years have appeared a lot of libraries that let the developer introduce speech recognition libraries inside their programs and applications. Among them all, the three that stand out the most are the following.

2.3.1. Google Cloud Speech API

The Google Cloud Speech API, with capacity to recognize over 110 languages, is a free speech recognition library that lets the users convert audio to text. It does use neural network techniques to transcribe the audio the user introduces trough the microphone in real time (Google Inc, n.d.a).

Because the Google Cloud Speech API uses an external server to convert the audio stream to text, it is continuously improving by recollecting data from all the request made to the server.

Also, because the server that converts the audio to text is external to the device, it is easier to support a great number of devices.

The Google Cloud Speech API is the default speech recognition software in the Android platform.

2.3.2. Bing Speech API

Microsoft has developed a speech recognition library that lets the user covert audio to text in real time. The Bing Speech API is free for the first 5,000 transactions each month, however, each thousand transactions surpassing the 5.000 transactions cost 4$.

The Microsoft API does not only provide a speech to text conversion, but also a text to speech conversion.

In the same way the Google Cloud Speech API does, the Bing Speech API developed by Microsoft does also recollect data from the requests made to the server to improve its services (Microsoft, n.d.).

2.3.3. CMUSphinx

Among the speech recognition libraries there are a lot open source libraries for the developer to use, however, the most popular one is CMUSphinx.

The main problem with CMUSphinx is that, even though it lets the developer introduce any language model it desires, the CMUSphinx library only provides models for 13 different languages (11 + 2 English dialects).

Even though, it provides a lot of tools for speech recognition related purposes and also provides a BSD-like license that allows commercial distribution of the software (CMUSphinx, n.d.)

Chapter 3

Technologies and Development Environment

In this chapter, the technologies and development environment that has been used to develop the system will be briefly described, and a justification will be provided for the selection of the technology, program or operating system chosen.

3.1. Technologies

In this section, the external technologies used that are packed or used by the final product are listed.

3.1.1. Android Platform

Description

Android is an open source operating system developed by Google Inc. Android, based on the Linux kernel, is aimed to power smartphones and devices with low power usage.

For more information about the Android mobile operating system structure, its components and the SDK go to the Annex A.

Justification

Given the amount of people that already have an Android device as their smartphone (described in section 2.1.1.) and the speech recognition libraries already included in the Android default API, the Android platform has been selected as the client side of the project.

3.1.2 Java

Description

Java is an object oriented programming language developed in 1991 and released in 1995 by Sun Microsystems. The Java programming language is aimed to run in a virtual environment to run in different architectures by simply implementing a different virtual machine in each of the architectures. Java is a high-level programming language designed to provide an easy tool for the developer to develop programs, however, Java lacks of low-level facilities like direct memory access.

Justification

Given the large amount of information that can be found about Java all over the web, and that Android applications use Java as its default programming language, the Java language has been selected as the main programming language of the project, for the server and Android application.

3.1.3. MySQL

Description

Available for a lot of operating systems from the renown Windows to operating systems like IBM AIX, HP-UX or Solaris, MySQL is one of the most popular open source databases in the world. It provides a fast and robust Structured Query Language Database designed for different types of environments, from mission-critical, heavy-load production systems to mass-deployed software (Oracle, 2017).

Justification

MySQL is a free open source SQL database library that can be used under a GNU General Public License. Also, MySQL is well documented and provides a powerful database without increasing the total project costs.

3.1.4. SQLite

Description

SQLite is an in-process library that implements a Structured Query Language database engine self-contained, serverless, transactional and without any previous configuration requirement. SQLite is also one of the most widely used databases in the world, aimed, not to the major enterprises, but to the memory constrained devices such as smartphones and MP3 players. SQLite does not try to replace the databases technologies like Oracle. Instead, SQLite tries to replace the operating system functions to save and store information (SQLite, n.d.).

Justification

SQLite is not only a lightweight SQL database library and of public domain, but also is the default database that the Android platform provides.

3.1.5. Weka

Description

Weka is a java library provided by the Machine Learning Group at the University of Waikato that contains a collection of machine learning algorithms used for data mining. It does also provide tools to pre-process, classify, cluster, associate and visualize the data introduced (Waikato University, n.d.a).

The main objective of the Machine Learning Group involved in the Weka project is to make Machine Learning techniques generally available to the public and to provide those machine learning techniques to the New Zealand industry (Waikato University, n.d.b).

Justification

Weka is free and open source Java library with a GNU General Public License that provides text classification algorithms without increasing the total cost of the project.

3.2 Development Environment

In this section will be listed the technologies, programs and operating systems involved in the development of the server and the Android application.

3.2.1. Android Studio

Description

Android Studio is the official Integrated Development Environment provided by Google to help Android app developers. It is an Integrated Development Environment based in the editor IntelliJ IDEA that provides the developer with an initial project structure, a user interface with lots of tools to help the developer in the software implementation like code completion and style formatting, tools to build and install the system inside an emulator or an android smartphone and a lot of tools to debug and monitor the application (Google Inc, n.d.b) It is available in Windows, Mac and Linux.

Justification

As previously said in the description, Android Studio is the official Integrated Development Environment provided by Google. Android studio is also free, so the final costs of the project are not increased.

3.2.2. Debian

Description

Debian is the operating system developed by the Debian project to create a free operating system. The kernel used by the Debian operating system is Linux, program that holds the entire operating system with its most basic functions. With more than 51.000 packages it lets the users choose how to operate and configure its own system. It does also support a large number of computer architectures in order to fulfill its purpose of being the universal operating system (Debian, 2017). Currently Debian latest stable release is Stretch.

Justification

Debian is free and highly stable operating system that has support for all the technologies that are used inside the project.

3.2.3. MariaDB

Description

MariaDB is one of the most popular Structured Query Language database server. Based on MySQL, it guarantees to stay open source with the support of the MariaDB Fundation. MariaDB due to its fast, scalable and robust system, with a wide variety of plugins and many other tools aims to hold a large amount of applications from website to banking applications (MariaDB Fundation, 2017).

Justification

MariaDB is the default and free package implementing the MySQL database included in the Debian repositories.

3.2.4. OpenJDK

Description

Provided by Oracle, OpenJDK is the open source library that aims to replace the Java Development Kit. With the help of OpenJDK community supported by Oracle, the OpenJDK contains the tools for the developers in order to build Java based applications. As JDK does, OpenJDK provides its own version of the Java Runtime Environment called OpenJRE (Oracle, 2010).

Justification

OpenJDK is the default and free package implementing the Java Development Kit included in the Debian repositories.

3.2.5. Vi Improved (Vim)

Description

Vim is a text editor aiming to create and change any kind of text in an efficient manner. It usually is included inside most UNIX systems and Apple OS X. The main characteristics of Vim are its highly configurable, very stable, consistently being developed and with support of a lot of plugins, programming languages and file formats.

Justification

Vim is a free Integrated Development Environment with a large community that has support for Java. It is also installed by default in Debian.

Chapter 4

Regulatory Framework

In this chapter, the legal aspects involved in the development of the project are described. In the first section of the chapter the Spain regulatory framework related to the storage of medical history is seen. In the second section of the chapter the legal aspects related with licenses of the software included in the project are discussed.

4.1. Data Protection Law in the Spanish Jurisdiction

The data that is stored in the database of the project has is regulated Spanish data protection law "*Ley Orgánica 15/1999, de 13 de diciembre, de Protección de Datos de Carácter Personal.*". The Spanish data protection law states in the first article the following.

"*Artículo 1. Objeto.*

La presente Ley Orgánica tiene por objeto garantizar y proteger, en lo que concierne al tratamiento de los datos personales, las libertades públicas y los derechos fundamentales de las personas físicas, y especialmente de su honor e intimidad personal y familiar."

This first article states that the Spanish data protection law has to guarantee and protect, in everything concerning the treatment of personal data, the public freedom and fundamental rights of the physical persons, and specially their honor, and personal and familiar intimacy.

To summarize, anything stored inside the application has to comply with the Spanish data protection law “*Ley Orgánica 15/1999, de 13 de diciembre, de Protección de Datos de Carácter Personal.*” and with its implementation “*Real Decreto 1720/2007, de 21 de diciembre, por el que se aprueba el Reglamento de desarrollo de la Ley Orgánica 15/1999, de 13 de diciembre, de protección de datos de carácter personal.*”.

It is also remarkable the article 7 section 3 of the *Ley Orgánica 15/1999, de 13 de diciembre, de Protección de Datos de Carácter Personal.*”, that determines that the data of the health of a patient is data specially protected.

The data specially protected, according to the section 3 of the article 81 of the “*Real Decreto 1720/2007, de 21 de diciembre, por el que se aprueba el Reglamento de desarrollo de la Ley Orgánica 15/1999, de 13 de diciembre, de protección de datos de carácter personal.*” should be protected under the basic, medium and high level measures to protect the data. The basic, medium and high level measures to protect user data are described in the chapter 3 and 4 of the same document.

The infringement of the “*Ley Orgánica 15/1999, de 13 de diciembre, de Protección de Datos de Carácter Personal.*” may be punished with fines of up to from 900€ to 600,000€.

The type of infringements is stated in the article 44 of the “*Ley Orgánica 15/1999, de 13 de diciembre, de Protección de Datos de Carácter Personal.*” law, while the type of sanctions are stated in the article 45.

From the article 45 of the “*Ley Orgánica 15/1999, de 13 de diciembre, de Protección de Datos de Carácter Personal.*” the sanctions in Table 4 are shown.

Type of infringement	Tipo de infracción	Sanction Fine
Minor Infringement	Infracción Leve	900€ - 40.000€
Serious Infringement	Infracción Grave	40.001€ - 300.000€
Very Serious Infringement	Infracción Muy Grave	300.001€ - 600.000€

Table 4. Sanction fines by type of infraction

4.2. Medical History Storage in the Spanish Jurisdiction

Within the Spanish jurisdiction, it is stated that, besides complying with the Spanish data protection law "*Ley Orgánica 15/1999, de 13 de diciembre, de Protección de Datos de Carácter Personal.*" and with its implementation "*Real Decreto 1720/2007, de 21 de diciembre, por el que se aprueba el Reglamento de desarrollo de la Ley Orgánica 15/1999, de 13 de diciembre, de protección de datos de carácter personal.*", the storage of the medical history has to complain also with the "Ley 41/2002, de 14 de noviembre, básica reguladora de la autonomía del paciente y de derechos y obligaciones en materia de información y documentación clínica." and with "*Ley 14/1986, de 25 de abril, General de Sanidad.*".

The protection of the data of the user inside the database of the server and the internal storage of the mobile device is out of the scope of this bachelor thesis. However, inside the "Ley 41/2002, de 14 de noviembre, básica reguladora de la autonomía del paciente y de derechos y obligaciones en materia de información y documentación clínica." it is stated the following.

"*Artículo 16. Usos de la historia clínica.*

3. El acceso a la historia clínica con fines judiciales, epidemiológicos, de salud pública, de investigación o de docencia, se rige por lo dispuesto en la Ley Orgánica 15/1999, de 13 de diciembre, de Protección de Datos de Carácter Personal, y en la Ley 14/1986, de 25 de abril, General de Sanidad, y demás normas de aplicación en cada caso. El acceso a la historia clínica con estos fines obliga a preservar los datos de identificación personal del paciente, separados de los de carácter clínicoasistencial, de manera que, como regla general, quede asegurado el anonimato, salvo que el propio paciente haya dado su consentimiento para no separarlos."

The section 3 of the article 16 of the "*Ley 41/2002, de 14 de noviembre, básica reguladora de la autonomía del paciente y de derechos y obligaciones en materia de información y documentación clínica.*" dictates that the access to the medical history with judicial, epidemiological, public health, research, and teaching purposes is governed by the "*Ley Orgánica 15/1999, de 13 de diciembre, de Protección de Datos de Carácter*

Personal" and by the " *Ley 14/1986, de 25 de abril, General de Sanidad"* laws, and by other rules of application in each case. The access to the clinical history with these purposes forces to preserve the personal identification data of the patient, separated from the ones of clinical-assistential character, so that, the anonymity of the patient is assured, unless the patient has given his consent not to separate them, making the the database of the patients and database of the diagnostics separate entities in the system.

Also in the "*Ley 41/2002, de 14 de noviembre, básica reguladora de la autonomía del paciente y de derechos y obligaciones en materia de información y documentación clínica.*" law it is dictated in the article 17 section 1 that the clinical history of the patients should be stored at least 5 years, however, it is possible to store the data in a different medium that the original one.

4.3. Licenses of the Included Software

In the development of software products is important to look after the compliance with the licenses of the technologies of the final product. In this section, each of the licenses of the technologies that are part of the final system are cited. The software licenses are regulated through the "*Real Decreto Legislativo 1/1996, de 12 de abril, por el que se aprueba el texto refundido de la Ley de Propiedad Intelectual, regularizando, aclarando y armonizando las disposiciones legales vigentes sobre la materia.*".

4.3.1. Android SDK

The Android SDK has a custom license agreement that should be accepted at the moment of download of the Android SDK. The agreement dictates that the SDK should be use only to develop applications for compatible implementations of Android. Also, Google holds no right, title or interest on the software applications developed nor intellectual property in those applications.

4.3.2 MariaDB

MariaDB version 10.1.26 is freely distributed under the GNU General Public License V2. The GNU General Public License V2 does not let the user to modify or integrate the code inside the application without making the implemented code open to the public.

However, as MariaDB is not integrated inside the code but used by it as a separate entity, the license is not violated.

4.3.3. MySQL Java Connector

MySQL Java Connector version 5.1.42 is freely distributed under the GNU General Public License V2. The GNU General Public License V2 does not let the user to modify or integrate the code inside the application without making the implemented code open to the public.

In the current project, the MySQL Java Connector is integrated in the code and the source code should be available to the public under a GPL V2 compatible license. However, Oracle has a Free and Open Source Software License Exception that allows developers of Free and Open Source Software applications to include the MySQL Connector inside the application. This exception lets the distribution of the MySQL Java Connector with the compliance of the terms and conditions of the FOSS license.

Also, there are MySQL Commercial licenses to let the developer that do not wish to distribute the source code of the software.

4.3.4. OpenJDK

OpenJDK version 1.8.141 is freely distributed under the GNU General Public License V2. The GNU General Public License V2 does not let the user to modify or integrate the code inside the application without making the implemented code open to the public.

However, the “Classpath” exception provided by Oracle lets the final software be linked to the OpenJDK library with independent modules, and distribute the executable under the terms of the developer choice for each of the independent modules.

4.3.5. SQLite

SQLite is distributed under Public Domain. The developer can modify and distribute the software without any restriction. However, the SQLite software cannot be copyrighted by an external entity.

4.3.6. Weka

Weka version 3.8.1 is freely distributed under the GNU General Public License V3. The GNU General Public License V3 does not let the user to modify or integrate the code inside the application without making the implemented code open to the public.

Given that the Weka libraries are integrated in the code of the system, the source code of the server has to be published under a GPL V3 compatible license.

4.3.7. License Conclusions

If distributed, the source code of the server has to be published under a GPL compatible license, while the source code of the Android application does not have restrictions and can be published under any license.

Chapter 5

System Development

In this chapter, the process of the implementation of the system to evaluate the patient symptoms is detailed.

5.1. Software Engineering Process

Before explaining the process that the final system has followed the different states involved in Software Engineering is explained.

Selection of the Process Model

The first step of Software Engineering is to select the method that will be used in order to develop the final product, from the identification of use cases to the integration and maintenance of the product.

Identification of Use Cases and Definition of Requirements

In this second step all the actors, scenarios and interactions with the system must be identified to later analyze and describe each of the requirements that the final system will provide.

Recognition of the Components and Design of the System

The third step in Software Engineering is to identify all the modules inside of your system and design the interactions between modules that comply with the requirements defined in the previous step.

Definition of the Test Cases

Once the design of the system is completed, the test cases are defined to ensure the correct operation of the system.

In this project, the test cases are replaced by a system evaluation.

Transfer, Documentation and Maintenance of the System

The final step in Software Engineering is to document all the installation, maintenance and creation of an user manual of the system to hand it over to the final user.

The transfer, creation of the user manual and maintenance of the system is out of the scope of this project given that there is not a final client to provide.

5.2. Process Model

During decades, the amount of process' models have been increasing rapidly, letting users choose the model that adapts better the development process of their products. For this specific system, given that the whole project is meant to be the result of a bachelor thesis, and only the student and the tutor are involved in the development of the system, the process model that has been selected has been the prototyping process model.

Prototyping Process Model

The prototyping model allows the users to evaluate the developer proposals before implementation. In this way, the requirements are not closed and can be reconsidered during the development process.

As the Figure 10 shows, the prototyping model follows the subsequent development phases.

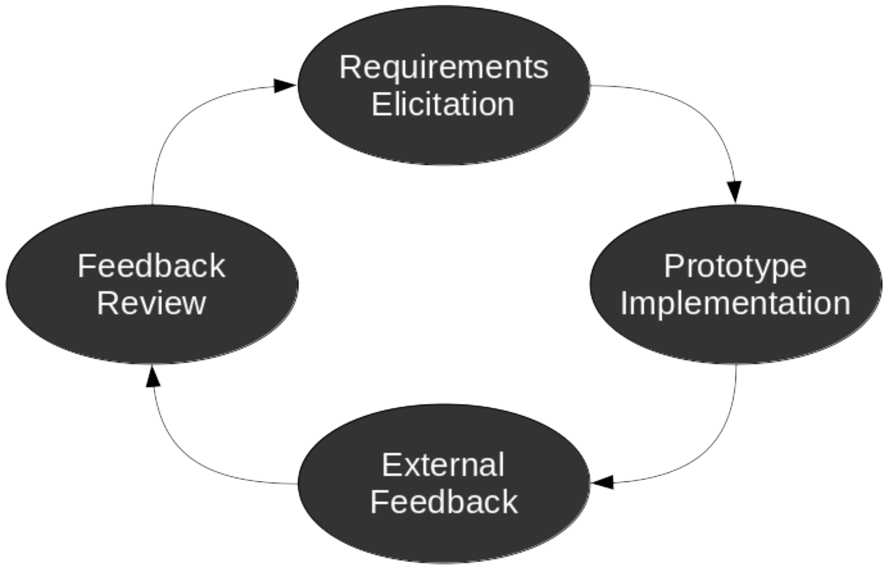


Figure 10. Prototyping process model

Basic Requirement Identification

A basic set of requirements have to be identified before the design of a product in any kind of Software Engineering model. However, in the prototyping model, as the requirements may not be clear enough, only a small set of requirements should be declared.

Implementation of an Initial Prototype

Once a basic set of requirements is declared, the design and implementation of the prototype can start. Even though the prototype does not have all the functionalities the final software will have, it shows an approximation of how the software should be like.

Review of the Prototype

Once the prototype is built, the feedback of external personnel to the project is collected to further develop the system.

Revision of the Prototype and System Requirements

The external personnel feedback is reviewed and the requirements are revised. The cycle continues until the customer expectations are fulfilled.

5.3. Requirements

In the the following section the use cases that define the system are shown. Then the user requirements and system requirements that describe and fulfill each of the use cases are cited below.

5.3.1. User Characteristics

For the design and development of the Android application is expected for the users to have some abilities and characteristics. The list of abilities and characteristics the user should have is listed below.

- The user must have access to a mobile device running an Android based operating system.
- It is required to have full visual capabilities to interact with the Android application.
- The user has to be able to interact physically with the mobile device running the application.
- The user needs to have an email account to use all the functionalities provided by the Android application.

5.3.2. Operational Environment

For a correct user experience it is expected for the device and the location it is used to fulfill some minimum requirements. The requirements expected to fulfill are listed below.

- The device needs to have battery/power enough to be powered on.
- The screen has to be able to display output.
- The touchscreen of the device has to be able to receive input.

- The mobile device needs to be connected to the Internet to use all the functionalities provided by the Android application.
- The device needs to have enough storage to install the application.
- The permissions required by the application must be accepted to use all the functionalities provided by the Android application.

5.3.3. Use Cases

The use cases show the list of actions the different users of the system are allowed to do. The actions are represented through the interaction of actors (users or external systems) and a system.

Each of the use cases is composed by two different components, a table that defines the actors and the purpose of the diagram, and a diagram that shows the different actions conforming an use case.

The tables that identify each of the use cases have the following structure:

Identification	
Actors	
Description	

- **Identification.** Identifier of the use case. The format of the identifier is “UCX_NameOfUseCase”, where X is the number of the use case.
- **Actors.** The actors of the diagram list the external entities that interact with the system for a particular use case.
- **Description.** A brief definition of what the use case is about.

UC01_UserCreation

The Table 5 shows the use case describing the user capacity to create an account, UC01_UserCreation.

UC01_UserCreation	
Actors	Android User
Description	For a user to create an account, first its data is collected and then it is sent to the server for it to store the data.

Table 5. UC01_UserCreation

The Figure 11 shows the use case described in the Table 5.

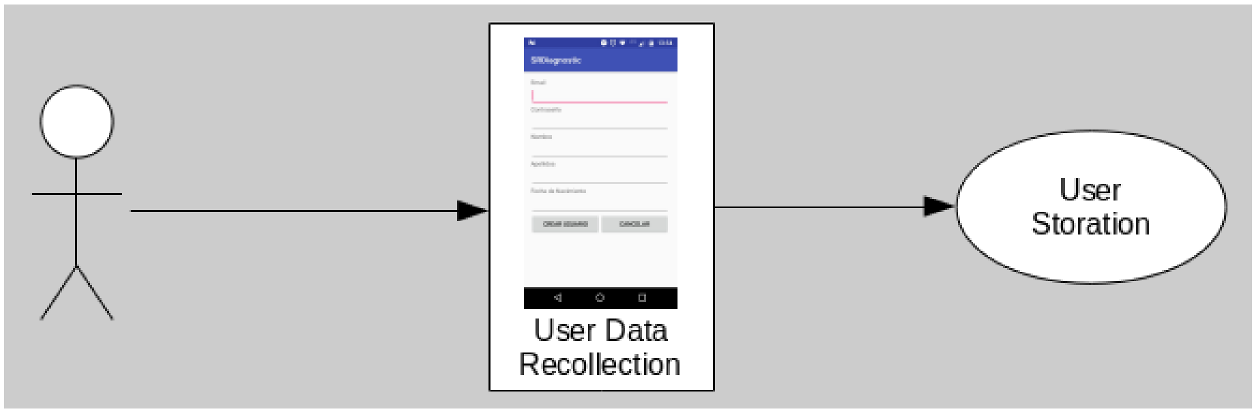


Figure 11. UC01_UserCreation

UC02_UserLogin

The Table 6 shows the use case describing the user capacity to log in inside the application, UC02_UserLogin.

UC02_UserLogin	
Actors	Android User
Description	For a user to login into its account, first its email and password its collected and then it is sent to the server for it to authenticate and retrieve the user.

Table 6. UC02_UserLogin

The Figure 12 shows the use case described in the Table 6.

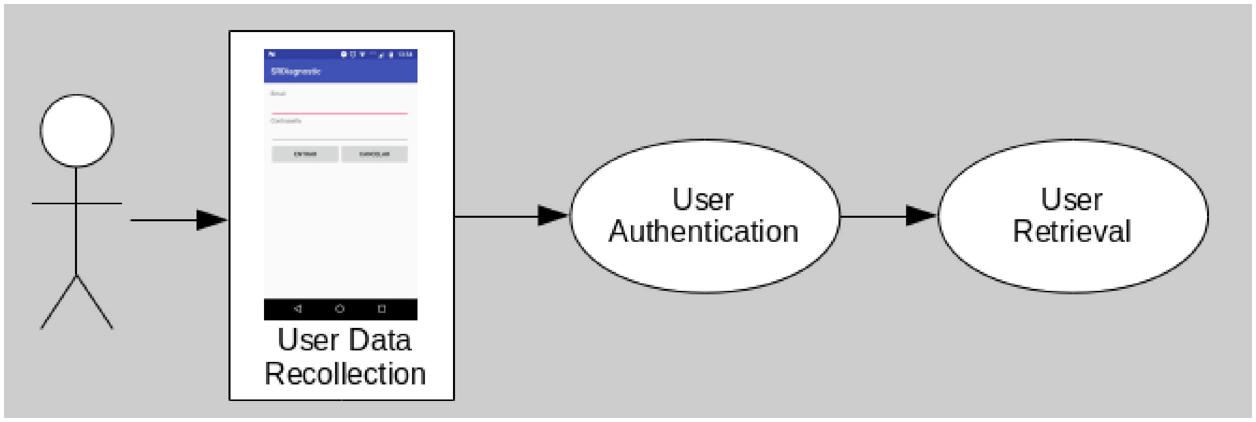


Figure 12. UC02_UserLogin

UC03_UserLogOut

The Table 7 shows the use case describing the user capacity to log out from the application, UC03_UserLogOut.

UC03_UserLogOut	
Actors	Android User
Description	For a user to log out, the user must press the log out button from the android application and the application must delete all data stored.

Table 7. UC03_UserLogOut

The Figure 13 shows the use case described in the Table 7.

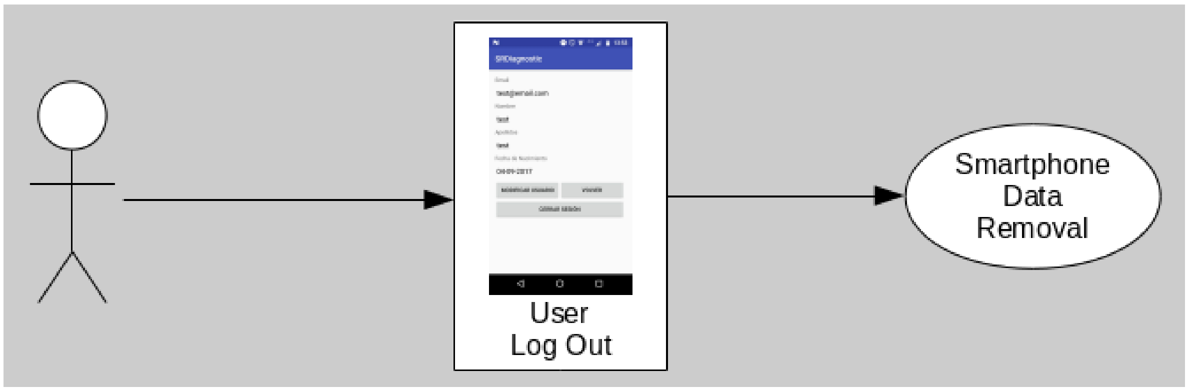


Figure 13. UC03_UserLogOut

UC04_UserModification

The Table 8 shows the use case describing the user capacity to modify its account, UC04_UserModification.

UC04_UserModification	
Actors	Android User
Description	For a user to modify its account, the password must be introduced in the Android application, then the new user data must be collected, the data is sent to the server where the user account is authenticated with the old email and password and finally modified.

Table 8. UC04_UserModification

The Figure 14 shows the use case described in the Table 8.

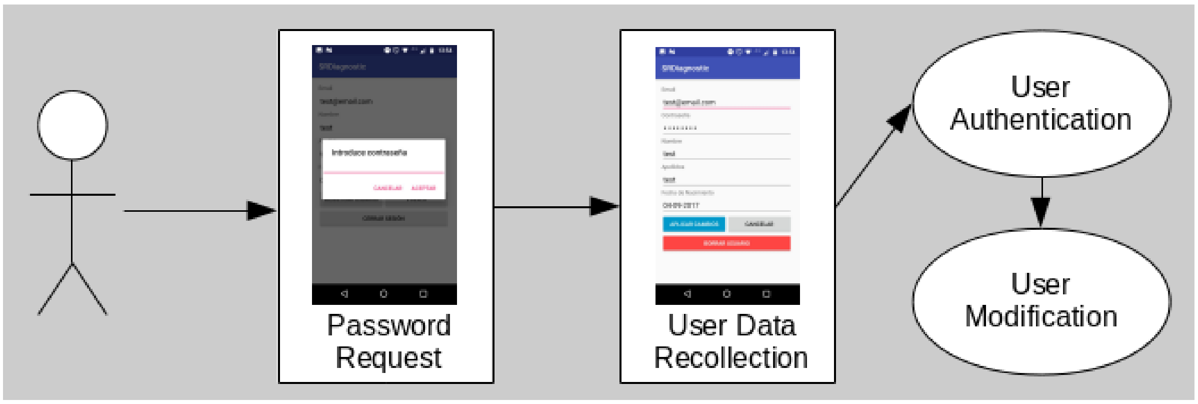


Figure 14. UC04_UserModification

UC05_UserDeletion

The Table 9 shows the use case describing the user capacity to delete its account, UC05_UserDeletion.

UC05_UserDeletion	
Actors	Android User
Description	For a user to modify its account, the password must be introduced in the Android application, then the user should confirm to delete its account, the server authenticates the account of the user, deletes the account from the server and all the data is finally removed from the mobile device.

Table 9. UC05_UserDeletion

The Figure 15 shows the use case described in the Table 9.

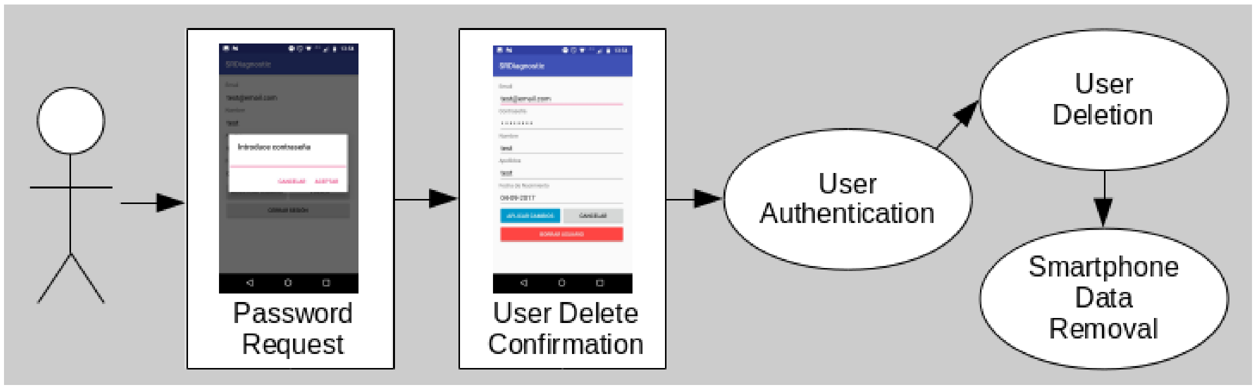


Figure 15. UC05_UserDeletion

UC06_ProfileViewer

The Table 10 shows the use case describing the user capacity to view its profile, UC06_ProfileViewer.

UC06_ProfileViewer	
Actors	Android User
Description	For a user to view its account it must get inside the Android activity that allows the user to see its profile.

Table 10. UC06_ProfileViewer

The Figure 16 shows the use case described in the Table 10.

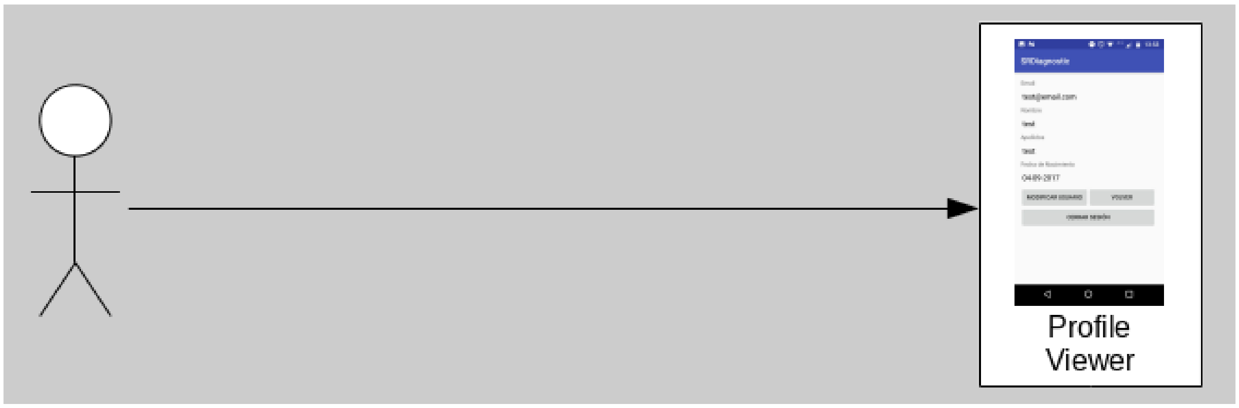


Figure 16. UC06_ProfileViewer

UC07_DiagnosticRetrieval

The Table 11 shows the use case describing the user capacity to get a diagnostic from a symptom being introduced, UC07_DiagnosticRetrieval.

UC07_DiagnosticRetrieval	
Actors	Android User
Description	For a user to get a new diagnostic from a symptom it must introduce the it, trough the speech recognizer or keyboard. Then the symptoms is sent to the server classifier for it to retrieve the diseases and recovery plans associated to it. To continue, the user has to select one of the diseases showed for the disease to be displayed. With the disease displayed, the user can select to store it. To store the diagnostic, the server authenticates the user, stores the diagnostic with the user identification, and sends back the diagnostic to the mobile device.

Table 11. UC07_DiagnosticRetrieval

The Figure 17 shows the use case described in the Table 11.

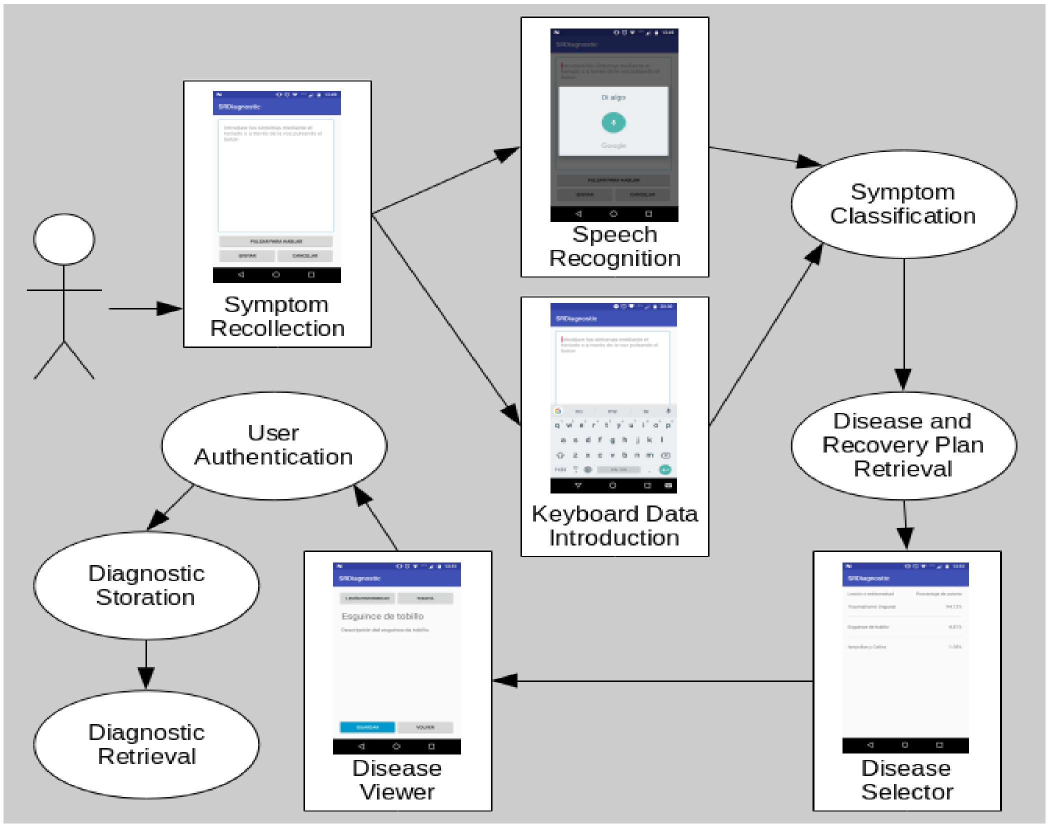


Figure 17. UC07_DiagnosticRetrieval

UC08_DiagnosticViewer

The Table 12 shows the use case describing the user capacity to view the diagnostics associated to it, UC08_DiagnosticViewer.

UC08_DiagnosticViewer	
Actors	Android User
Description	For a user to view its diagnostics it must get inside the Android activity that allows the user to see its diagnostics.

Table 12. UC08_DiagnosticViewer

The Figure 18 shows the use case described in the Table 12.

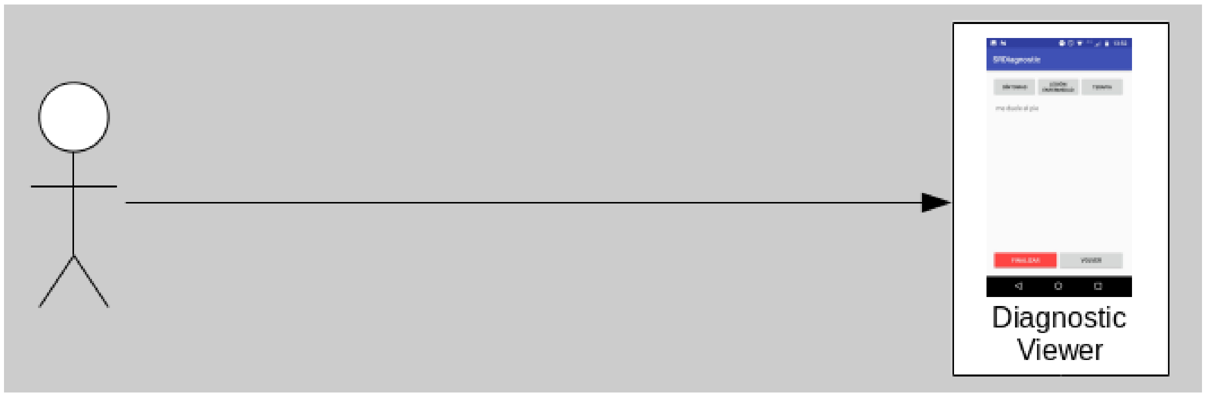


Figure 18. UC08_DiagnosticViewer

UC09_DiagnosticEvaluation

The Table 13 shows the use case describing the user capacity to evaluate a diagnostic associated to it, UC09_DiagnosticEvaluation.

UC09_DiagnosticEvaluation	
Actors	Android User
Description	For a user to evaluate the diagnostic it must have finished the recovery plan, then the user has to select if the diagnostic has been successful or not, and then the result has to be saved into the server by modifying the diagnostic.

Table 13. UC09_DiagnosticEvaluation

The Figure 19 shows the use case described in the Table 13.

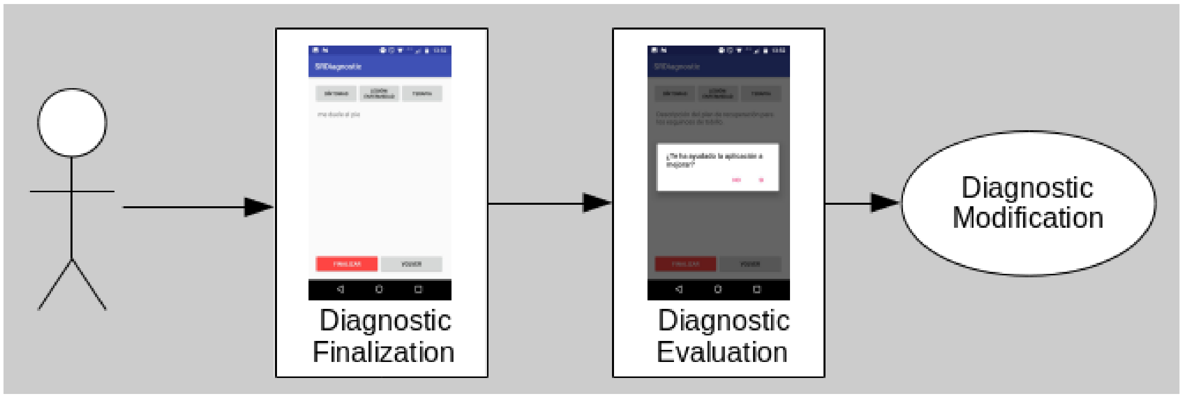


Figure 19. UC09_DiagnosticEvaluation

UC10_SettingsSelection

The Table 14 shows the use case describing the user capacity to change the settings stored inside the application, UC10_SettingsSelection.

UC10_SettingsSelection	
Actors	Android User
Description	For a user to select the settings, first must select the preferred settings and then the settings must be stored inside the mobile device.

Table 14. UC10_SettingsSelection

The Figure 20 shows the use case described in the Table 14.

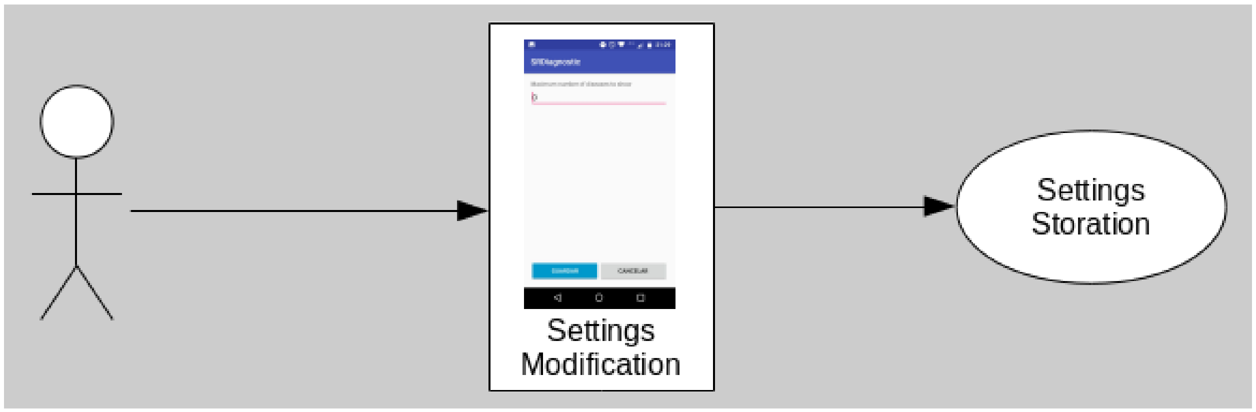


Figure 20. UC10_SettingsSelection

5.3.4. User Requirements

5.3.4.1. User Requirements Specification

Before introducing the requirements that define the system, the format of the requirements must be defined.

The tables that identify each of the User Requirements have the following structure:

Identification		
Priority:	Requirement:	Stability:
Description:		

Below a list with each of the fields in the table with a brief definition and its values is shown.

- **Identification.** Identifier of the User Requirement. The format of the identifier is “URX_NameOfRequirement”, where the X is the number identifier of the requirement.
- **Priority.** The priority indicates the preference in which the requirement must be fulfilled. The levels of priority are High, Medium and Low.
- **Requirement.** The requirement field indicates the importance of the requirement in the system. The levels of requirement are Essential, Desirable and Optional.
- **Stability.** The stability field shows how solid a requirement will be along the whole Software Engineer process. The levels of verifiability are High, Medium and Low.
- **Description.** A brief definition of the what the requirement is about.

5.3.4.2. Capability Requirements

UR01_UserCreation

The Table 15 describes the user requirement UR01_UserCreation.

UR01_UserCreation		
Priority: High	Requirement: Essential	Stability: High
Description: The user shall be able to create an user account.		

Table 15. UR01_UserCreation

UR02_UserLogin

The Table 16 describes the user requirement UR02_UserLogin.

UR02_UserLogin		
Priority: High	Requirement: Essential	Stability: High
Description: The user shall be able to log in into its account.		

Table 16. UR02_UserLogin

UR03_UserLogOut

The Table 17 describes the user requirement UR03_UserLogOut.

UR03_UserLogOut		
Priority: High	Requirement: Essential	Stability: High
Description: If the user is already logged, it shall be able to log out from the Android application		

Table 17. UR03_UserLogOut

UR04_UserModification

The Table 18 describes the user requirement UR04_UserModification.

UR04_UserModification		
Priority: High	Requirement: Essential	Stability: High
Description: The user shall be able to modify its user account.		

Table 18. UR04_UserModification

UR05_UserDeletion

The Table 19 describes the user requirement UR05_UserDeletion.

UR05_UserDeletion		
Priority: High	Requirement: Essential	Stability: High
Description: The user shall be able to delete its user account.		

Table 19. UR05_UserDeletion

UR06_UserProfile

The Table 20 describes the user requirement UR06_UserProfile.

UR06_UserProfile		
Priority: High	Requirement: Essential	Stability: High
Description: The user shall be able view its account.		

Table 20. UR06_UserProfile

UR07_SymptomRetrievalKeyboard

The Table 21 describes the user requirement UR07_SymptomRetrievalKeyboard.

UR07_SymptomRetrievalKeyboard		
Priority: High	Requirement: Essential	Stability: High
Description: The user shall be able to introduce its symptoms with the keyboard.		

Table 21. UR07_SymptomRetrievalKeyboard

UR08_SymptomRetrievalSpeechRecognition

The Table 22 describes the user requirement UR08_SymptomRetrievalSpeechRecognition.

UR08_SymptomRetrievalSpeechRecognition		
Priority: High	Requirement: Essential	Stability: High
Description: The user shall be able to introduce its symptoms with its voice.		

Table 22. UR08_SymptomRetrievalSpeechRecognition

UR09_SymptomClassification

The Table 23 describes the user requirement UR09_SymptomClassification.

UR09_SymptomClassification		
Priority: High	Requirement: Essential	Stability: High
Description: The user shall be able to get a set of diseases and recovery plans associated to a symptom introduced from the Android application through the help of a classifier up to date.		

Table 23. UR09_SymptomClassification

UR10_DiseasesSetMaxNumber

The Table 24 describes the user requirement UR10_DiseasesSetMaxNumber.

UR10_DiseasesSetMaxNumber		
Priority: Low	Requirement: Optional	Stability: High
Description: The user shall be able to set the number of diseases associated to the symptom it wants to be retrieved.		

Table 24. UR10_DiseasesSetMaxNumber

UR11_DiseaseViewer

The Table 25 describes the user requirement UR11_DiseaseViewer.

UR11_DiseaseViewer		
Priority: High	Requirement: Essential	Stability: High
Description: The user shall be able to see the disease and the recovery plan associated before saving it as a diagnostic.		

Table 25. UR11_DiseaseViewer

UR12_DiagnosticSaver

The Table 26 describes the user requirement UR12_DiagnosticSaver.

UR12_DiagnosticSaver		
Priority: High	Requirement: Essential	Stability: High
Description: The user shall be able to save the selected disease and recovery plan with the symptoms in the form of diagnostic inside its account and its smartphone.		

Table 26. UR12_DiagnosticSaver

UR13_MissingDiagnosticsRetrieval

The Table 27 describes the user requirement UR13_MissingDiagnosticsRetrieval.

UR13_MissingDiagnosticsRetrieval		
Priority: High	Requirement: Essential	Stability: High
Description: The user shall be able to retrieve all its active diagnostics from the server and store them inside the smartphone.		

Table 27. UR13_MissingDiagnosticsRetrieval

UR14_DiagnosticViewer

The Table 28 describes the user requirement UR14_DiagnosticViewer.

UR14_DiagnosticViewer		
Priority: High	Requirement: Essential	Stability: High
Description: The user shall be able to view the complete diagnostic saved in the smartphone, with the disease and recovery plan associated to it.		

Table 28. UR14_DiagnosticViewer

UR15_DiagnosticRemoval

The Table 29 describes the user requirement UR15_DiagnosticRemoval.

UR15_DiagnosticRemoval		
Priority: High	Requirement: Essential	Stability: High
Description: The user shall be able to finish a recovery plan and remove the diagnostic from its smartphone.		

Table 29. UR15_DiagnosticRemoval

UR16_DiagnosticEvaluation

The Table 30 describes the user requirement UR16_DiagnosticEvaluation.

UR16_DiagnosticEvaluation		
Priority: Medium	Requirement: Desirable	Stability: High
Description: If a recovery plan has finished, before the diagnostic is removed, the user shall evaluate the diagnostic.		

Table 30. UR16_DiagnosticEvaluation

UR17_AndroidApplicationUsage

The Table 31 describes the user requirement UR17_AndroidApplicationUsage.

UR17_AndroidApplicationUsage		
Priority: High	Requirement: Essential	Stability: High
Description: The user shall be able to add, delete, get or modify an user, to classify a symptom, to save a diagnostic, to retrieve all the diagnostics of the patient or to rate the diagnostic through the use of an Android application.		

Table 31. UR17_AndroidApplicationUsage

5.3.4.3 Constraint Requirements

UR18_InternetConnection

The Table 32 describes the user requirement UR18_InternetConnection.

UR18_InternetConnection		
Priority: High	Requirement: Essential	Stability: High
Description: The user shall be able to access the symptom classification service, its diagnostics, and the user creation, modification and deletion service whenever the user has Internet connection.		

Table 32. UR18_InternetConnection

UR19_EmailAccount

The Table 33 describes the user requirement UR19_EmailAccount.

UR19_EmailAccount		
Priority: High	Requirement: Essential	Stability: High
Description: The user shall create an account using its email address.		

Table 33. UR19_EmailAccount

5.3.5. System Requirements

5.3.5.1. System Requirements Specification

In the same way the User Requirements where defined in section 5.3.4, before introducing the system requirements, the format the requirements have shall be described.

The tables that identify each of the User Requirements have the following structure:

Identification		
Priority:	Requirement:	Stability:
Description:		

Below a list with each of the fields in the table with a brief definition and its values is shown.

- **Identification.** Identifier of the System Requirement. The format of the identifier is "SRX_NameOfRequirement", where the X is the number identifier of the requirement.
- **Priority.** The priority indicates the preference in which the requirement must be fulfilled. The levels of priority are High, Medium and Low.
- **Requirement.** The requirement field indicates the importance of the requirement in the system. The levels of requirement are Essential, Desirable and Optional.
- **Stability.** The stability field shows how solid a requirement will be along the whole Software Engineer process. The levels of verifiability are High, Medium and Low.
- **Description.** A brief definition of the what the requirement is about.

5.3.5.2. Functional Requirements Statement

SR01_UserAccount

The Table 34 describes the system requirement SR01_UserAccount.

SR01_UserAccount		
Priority: High	Requirement: Essential	Stability: High
Description: The system shall store the patients/users identification number, birth date, email, name password and surname.		

Table 34. SR01_UserAccount

SR02_RecoveryPlan

The Table 35 describes the system requirement SR02_RecoveryPlan.

SR02_RecoveryPlan		
Priority: High	Requirement: Essential	Stability: High
Description: The system shall store the recovery plans identification number and description.		

Table 35. SR02_RecoveryPlan

SR03_Disease

The Table 36 describes the system requirement SR03_Disease.

SR03_Disease		
Priority: High	Requirement: Essential	Stability: High
Description: The system shall store the diseases identification number, name, description and the identification number of the recovery plan associated to the disease.		

Table 36. SR03_Disease

SR04_Diagnostic

The Table 37 describes the system requirement SR04_Diagnostic.

SR04_Diagnostic		
Priority: High	Requirement: Essential	Stability: High
Description: The system shall store the diagnostics identification number, the identification numbers of the disease, patient, and recovery plan associated to the diagnostic, the result of the diagnostic, the date and time of the diagnostic, and the symptoms associated to the patient inside the diagnostic.		

Table 37. SR04_Diagnostic

SR05_RequestCommandNumber

The Table 38 describes the system requirement SR05_RequestCommandNumber.

SR05_RequestCommandNumber		
Priority: High	Requirement: Essential	Stability: High
Description: The system shall use a specific number to execute each of the request the Android application makes to the server.		

Table 38. SR05_RequestCommandNumber

SR06_ServerThreads

The Table 39 describes the system requirement SR06_ServerThreads.

SR06_ServerThreads		
Priority: High	Requirement: Essential	Stability: High
Description: The server shall create a different thread for each client request to add, delete, get or modify an user, to classify a symptom, to save a diagnostic, to retrieve all the diagnostics of the patient or to rate the diagnostic.		

Table 39. SR06_ServerThreads

SR07_ServerUserCreation

The Table 40 describes the system requirement SR07_ServerUserCreation.

SR07_ServerUserCreation		
Priority: High	Requirement: Essential	Stability: High
Description: After the Android application has sent the user creation request with the birth date, email, name, password and surname of the patient/user, the server shall create a new patient/user.		

Table 40. SR07_ServerUserCreation

SR08_ServerUserRetrieval

The Table 41 describes the system requirement SR07_ServerUserRetrieval.

SR08_ServerUserRetrieval		
Priority: High	Requirement: Essential	Stability: High
Description: After the Android application has sent the get patient/user request with the email and password of the patient/user, the server shall retrieve the patient/user to the client.		

Table 41. SR08_ServerUserRetrieval

SR09_ServerUserModification

The Table 42 describes the system requirement SR09_ServerUserModification.

SR09_ServerUserModification		
Priority: High	Requirement: Essential	Stability: High
Description: After the Android application has sent the modify patient/user account request with the email and password of the patient/user, and the new data of the patient, the server shall modify the patient/user.		

Table 42. SR09_ServerUserModification

SR10_ServerUserDeletion

The Table 43 describes the system requirement SR10_ServerUserDeletion.

SR10_ServerUserDeletion		
Priority: High	Requirement: Essential	Stability: High
Description: After the Android application has sent the delete request with the email and password of the patient/user, the server shall create a delete the patient/user.		

Table 43. SR10_ServerUserDeletion

SR11_ServerNaiveBayesClassificator

The Table 44 describes the system requirement SR11_ServerNaiveBayesClassificator.

SR11_ServerNaiveBayesClassificator		
Priority: High	Requirement: Essential	Stability: High
Description: The server shall implement a Naive Bayes classifier in order to classify the different symptoms a client has.		

Table 44. SR11_ServerNaiveBayesClassificator

SR12_ServerClassifierUpdater

The Table 45 describes the system requirement SR12_ServerClassifierUpdater.

SR12_ServerClassifierUpdater		
Priority: Low	Requirement: Essential	Stability: High
Description: The server shall update the classifier with all the diagnostics inside the database.		

Table 45. SR12_ServerClassifierUpdater

SR13_ServerDiagnosticClassification

The Table 46 describes the system requirement SR13_ServerDiagnosticClassification.

SR13_ServerDiagnosticClassification		
Priority: High	Requirement: Essential	Stability: High
Description: The server shall classify the symptoms sent by the Android application through the Naive Bayes classifier.		

Table 46. SR13_ServerDiagnosticClassification

SR14_ServerDiagnosticClassificationOutput

The Table 47 describes the system requirement SR14_ServerDiagnosticClassificationOutput.

SR14_ServerDiagnosticClassificationOutput		
Priority: High	Requirement: Essential	Stability: High
Description: After the Android application has sent the classify symptoms request with the symptoms, the server shall send a set of diseases, recovery plans and success rates of the diseases being the ones reflected by the symptoms classification to the client.		

Table 47. SR14_ServerDiagnosticClassificationOutput

SR15_ServerDiagnosticStorage

The Table 48 describes the system requirement SR15_ServerDiagnosticStorage.

SR15_ServerDiagnosticStorage		
Priority: High	Requirement: Essential	Stability: High
Description: After the Android application has sent the save diagnostic request with the symptoms, email and password of the user, and disease and plan identifier, the server shall send the diagnostic back to the user to confirm that the diagnostic has been correctly saved in the user account.		

Table 48. SR15_ServerDiagnosticStorage

SR16_ServerActiveDiagnoticRetrieval

The Table 49 describes the system requirement SR16_ServerActiveDiagnosticRetrieval.

SR16_ServerActiveDiagnosticRetrieval		
Priority: High	Requirement: Essential	Stability: High
Description: After receiving all the diagnostics the Android application has from a user with the email and the password of the user, the server shall send to the user the active diagnostics that are not in the smartphone with the diseases and recovery plans associated to those active diagnostics.		

Table 49. SR16_ServerActiveDiagnosticRetrieval

SR17_ServerDiagnosticEvaluation

The Table 50 describes the system requirement SR17_ServerDiagnosticEvaluation.

SR17_ServerDiagnosticEvaluation		
Priority: Medium	Requirement: Desirable	Stability: High
Description: After receiving the evaluate diagnostic request with the diagnostic identifier and email and password of the user, the server shall modify the diagnostic result according to the patient/user satisfaction (2 if satisfied and 1 if not).		

Table 50. SR17_ServerDiagnosticEvaluation

SR18_ServerConfirmationCode

The Table 51 describes the system requirement SR18_ServerConfirmationCode.

SR18_ServerConfirmationCode		
Priority: High	Requirement: Essential	Stability: High
Description: The server shall send a confirmation code to the Android application if the request sent has been successfully executed.		

Table 51. SR18_ServerConfirmationCode

SR19_ServerUnexpectedErrorCode

The Table 52 describes the system requirement SR19_ServerUnexpectedErrorCode.

SR19_ServerUnexpectedErrorCode		
Priority: High	Requirement: Essential	Stability: High
Description: The server shall send an error message if the request sent by the Android application has suffered an unexpected error.		

Table 52. SR19_ServerUnexpectedErrorCode

SR20_ServerWrongEmailPasswordCode

The Table 53 describes the system requirement SR20_ServerWrongEmailPasswordCode.

SR20_ServerWrongEmailPasswordCode		
Priority: Medium	Requirement: Desirable	Stability: High
Description: The server shall send an error message if the email or password sent by the Android application is not the correct one or has been taken.		

Table 53. SR20_ServerWrongEmailPasswordCode

SR21_ServerBlankFieldCode

The Table 54 describes the system requirement SR21_ServerBlankFieldCode.

SR21_ServerBlankFieldCode		
Priority: Medium	Requirement: Desirable	Stability: High
Description: The server shall send an error message if a field necessary for the server to complete the request sent from the Android application has been left in blank.		

Table 54. SR21_ServerBlankFieldCode

SR22_ApplicationSettingsInitialization

The Table 55 describes the system requirement SR22_ApplicationSettingsInitialization.

SR22_ApplicationSettingsInitialization		
Priority: Medium	Requirement: Desirable	Stability: High
Description: The Android application shall initialize the system settings if no user is stored inside smartphone.		

Table 55. SR22_ApplicationSettingsInitialization

SR23_ApplicationNoUserRedirection

The Table 56 describes the system requirement SR23_ApplicationNoUserRedirection.

SR23_ApplicationNoUserRedirection		
Priority: High	Requirement: Essential	Stability: High
Description: The Android application shall redirect the user to an activity that lets the user create a new account or login into an existing one if no user is stored inside the smartphone.		

Table 56. SR23_ApplicationNoUserRedirection

SR24_ApplicationUserRedirection

The Table 57 describes the system requirement SR24_ApplicationUserRedirection.

SR24_ApplicationUserRedirection		
Priority: High	Requirement: Essential	Stability: High
Description: The Android application shall redirect the user to an activity that lets the user view, modify and delete the user, classify symptoms, and save and store diagnostics if an account is stored inside the application.		

Table 57. SR24_ApplicationUserRedirection

SR25_ApplicationUserCreation

The Table 58 describes the system requirement SR25_ApplicationUserCreation.

SR25_ApplicationIserCreation		
Priority: High	Requirement: Essential	Stability: High
Description: The Android application shall provide the user with the ability of creating an account by introducing the birth date, email, name, password and surname in the user creation activity.		

Table 58. SR25_ApplicationUserCreation

SR26_ApplicationUserCreationStorage

The Table 59 describes the system requirement SR26_ApplicationUserCreationStorage.

SR26_ApplicationUserCreationStorage		
Priority: High	Requirement: Essential	Stability: High
Description: If an user has been correctly created in the server the Android application shall store the user in the smartphone.		

Table 59. SR26_ApplicationUserCreationStorage

SR27_ApplicationUserLogin

The Table 60 describes the system requirement SR27_ApplicationUserLogin.

SR27_ApplicationUserLogin		
Priority: High	Requirement: Essential	Stability: High
Description: The user login activity shall let the user introduce the email and password.		

Table 60. SR27_ApplicationUserLogin

SR28_ApplicationUserLoginStorage

The Table 61 describes the system requirement SR28_ApplicationUserLoginStorage.

SR28_ApplicationUserLoginStorage		
Priority: High	Requirement: Essential	Stability: High
Description: If an user has been correctly sent from the server the Android application shall store the user in the smartphone.		

Table 61. SR28_ApplicationUserLoginStorage

SR29_ApplicationMissingActiveDiagnosticsRetrieval

The Table 62 describes the system requirement SR29_ApplicationMissingActiveDiagnosticsRetrieval.

SR29_ApplicationMissingActiveDiagnosticsRetrieval		
Priority: High	Requirement: Essential	Stability: High
Description: The Android application shall retrieve all active diagnostics the user does not have in its smartphone when the user is logged in.		

Table 62. SR29_ApplicationMissingActiveDiagnosticsRetrieval

SR30_ApplicationMissingActiveDiagnosticsStorage

The Table 63 describes the system requirement SR30_ApplicationMissingActiveDiagnosticsStorage.

SR30_ApplicationMissingActiveDiagnosticsStorage		
Priority: High	Requirement: Essential	Stability: High
Description: If the server has sent all active diagnostics the user does not have in its smartphone, the Android application shall store them.		

Table 63. SR30_ApplicationMissingDiagnosticsStorage

SR31_ApplicationSymptomsKeyboardInput

The Table 64 describes the system requirement SR31_ApplicationSymptomsKeyboardInput.

SR31_ApplicationSymptomsKeyboardInput		
Priority: High	Requirement: Essential	Stability: High
Description: The Android application shall let the user introduce the symptoms through keyboard offered by the Android platform.		

Table 64. SR31_ApplicationSymptomsKeyboardInput

SR32_ApplicationSymptomsSpeechRecognitionInput

The Table 65 describes the system requirement SR32_ApplicationSymptomsSpeechRecognitionInput.

SR32_ApplicationSymptomsSpeechRecognitionInput		
Priority: High	Requirement: Essential	Stability: High
Description: The Android application shall let the user introduce the symptoms through a speech recognition service offered by the Android platform.		

Table 65. SR32_ApplicationSymptomsSpeechRecognitionInput

SR33_ApplicationSymptomsClassification

The Table 66 describes the system requirement SR33_ApplicationSymptomsClassification.

SR33_ApplicationSymptomsClassification		
Priority: High	Requirement: Essential	Stability: High
Description: The Android application shall let the user send the symptoms to the server for the server to classify them.		

Table 66. SR33_ApplicationSymptomsClassification

SR34_ApplicationSymptomsClassificationOutput

SR34_ApplicationSymptomsClassificationOutput		
Priority: High	Requirement: Essential	Stability: High
Description: The Android application shall receive a set of diseases, recovery plans and doubles as a result of the server applying the classifier to the symptoms introduced by the user.		

Table 67. SR34_ApplicationSymptomsClassificationOutput

SR35_ApplicationSymptomsClassificationOutputViewer

The Table 68 describes the system requirement SR35_ApplicationSymptomsClassificationOutputViewer.

SR35_ApplicationSymptomsClassificationOutputViewer		
Priority: High	Requirement: Essential	Stability: High
Description: The Android application shall show the diseases associated to the symptom introduced along with the percentage of success of the user having the disease.		

Table 68. SR35_ApplicationSymptomsClassificationOutputViewer

SR36_ApplicationSymptomsClassificationOutputSelection

The Table 69 describes the system requirement SR36_ApplicationSymptomsClassificationOutputSelection.

SR36_ApplicationSymptomsClassificationOutputSelection		
Priority: High	Requirement: Essential	Stability: High
Description: The Android application shall let user choose a disease from a set of diseases sent from the server.		

Table 69. SR36_ApplicationSymptomsClassificationOutputSelection

SR37_ApplicationDiseaseViewer

The Table 70 describes the system requirement SR37_ApplicationDiseaseViewer.

SR37_ApplicationDiseaseViewer		
Priority: High	Requirement: Essential	Stability: High
Description: If a user has selected a disease, the Android application shall let the user view the name of the disease and its description along with a description of a recovery plan associated to the disease.		

Table 70. SR37_ApplicationDiseaseViewer

SR38_ApplicationDiagnosticSaver

The Table 71 describes the system requirement SR38_ApplicationDiagnosticSaver.

SR38_ApplicationDiagnosticSaver		
Priority: High	Requirement: Essential	Stability: High
Description: The Android application shall let the user save the diagnostic of the disease selected inside the Android application along with the recovery plan and the disease.		

Table 71. SR38_ApplicationDiagnosticSaver

SR39_ApplicationProfileViewer

The Table 72 describes the system requirement SR39_ApplicationProfileViewer.

SR39_ApplicationProfileViewer		
Priority: High	Requirement: Essential	Stability: High
Description: The Android application shall provide an activity that shows the user all the data in the account stored in the smartphone except the password.		

Table 72. SR39_ApplicationProfileViewer

SR40_ApplicationUserLogOut

The Table 73 describes the system requirement SR40_ApplicationUserLogOut.

SR40_ApplicationUserLogOut		
Priority: High	Requirement: Essential	Stability: High
Description: The Android application shall let the user logout from the application deleting all the patients, diagnostics, diseases and recovery plans stored inside the application.		

Table 73. SR40_ApplicationUserLogOut

SR41_ApplicationProfileModificationPasswordRequest

The Table 74 describes the system requirement SR41_ApplicationProfileModificationPasswordRequest.

SR41_ApplicationProfileModificationPasswordRequest		
Priority: Medium	Requirement: Desirable	Stability: High
Description: The Android application shall require the password of the user to let the user modify the data in its account or to delete it.		

Table 74. SR41_ApplicationProfileModificationPasswordRequest

SR42_ApplicationUserModificationConfirmation

The Table 75 describes the system requirement SR42_ApplicationUserModificationConfirmation.

SR42_ApplicationUserModificationConfirmation		
Priority: High	Requirement: Essential	Stability: High
Description: If the server has correctly saved the modified user account, the Android application shall save the new data of the user account inside the smartphone.		

Table 75. SR42_ApplicationUserModificationConfirmation

SR43_ApplicationUserDeletionConfirmation

The Table 76 describes the system requirement SR43_ApplicationUserDeletionConfirmation.

SR43_ApplicationUserDeletionConfirmation		
Priority: High	Requirement: Essential	Stability: High
Description: If the server has correctly deleted the user account, the Android application shall remove all the patients, diagnostics, diseases and recovery plans stored inside the application.		

Table 76. SR43_ApplicationUserDeletionConfirmation

SR44_ApplicationMaximumDiseasesSetRetrieval

The Table 77 describes the system requirement SR44_ApplicationMaximumDiseasesSetRetrieval.

SR44_ApplicationMaximumDiseasesSetRetrieval		
Priority: Low	Requirement: Optional	Stability: High
Description: The Android application shall let the user store the maximum number of diseases retrieved by the server after the symptoms of the user are sent.		

Table 77. SR44_ApplicationMaximumDiseasesSetRetrieval

SR45_ApplicationDiagnosticViewer

The Table 78 describes the system requirement SR45_ApplicationDiagnosticViewer.

SR45_ApplicationDiagnosticViewer		
Priority: High	Requirement: Essential	Stability: High
Description: The Android application shall let the user view the diagnostics stored inside the application with the diseases and recovery plans associated to it in an Android activity.		

Table 78. SR45_ApplicationDiagnosticViewer

SR46_ApplicationDiagnosticRemoval

SR46_ApplicationDiagnosticRemoval		
Priority: High	Requirement: Essential	Stability: High
Description: The Android application shall let the user remove the diagnostic stored inside the smartphone.		

Table 79. SR46_ApplicationDiagnosticRemoval

SR47_ApplicationDiagnosticEvaluation

The Table 80 describes the system requirement SR47_ApplicationDiagnosticEvaluation.

SR47_ApplicationDiagnosticEvaluation		
Priority: Medium	Requirement: Desirable	Stability: High
Description: The Android application shall request the evaluation of a diagnostic once the diagnostic has been removed from the smartphone.		

Table 80. SR47_ApplicationDiagnosticEvaluation

SR48_ApplicationInternet

The Table 81 describes the system requirement SR48_ApplicationInternet.

SR48_ApplicationInternet		
Priority: High	Requirement: Essential	Stability: High
Description: The Android application shall not communicate with the server if no Internet connection is provided.		

Table 81. SR48_ApplicationInternet

SR49_PrivateDataRetrievalProtection

The Table 82 describes the system requirement SR49_PrivateDataRetrievalProtection.

SR49_PrivateDataRetrievalProtection		
Priority: High	Requirement: Essential	Stability: High
Description: The Android application shall send at least the email and password of a user to the server each time the application request a diagnostic or user account.		

Table 82. SR49_PrivateDataRetrievalProtection

SR50_SystemCommunication

The Table 83 describes the system requirement SR50_SystemCommunication.

SR50_SystemCommunication		
Priority: High	Requirement: Essential	Stability: High
Description: The communication between the Android application and the server shall be made through Java sockets.		

Table 83. SR50_SystemCommunication

5.3.5.3. Non-Functional Requirements Statement

SR51_MinimumSDKVersion

The Table 84 describes the system requirement SR51_MinimumSDKVersion.

SR51_MinimumSDKVersion		
Priority: High	Requirement: Essential	Stability: High
Description: The Android application shall work, at least, with the SDK version 15.		

Table 84. SR51_MinimumSDKVersion

SR52_UserDataDatabase

The Table 85 describes the system requirement SR52_UserDataDatabase.

SR52_UserDataDatabase		
Priority: High	Requirement: Essential	Stability: High
Description: The server shall have a database only to store the data of patients/users.		

Table 85. SR52_UserDataDatabase

SR53_NonUserDataDatabase

The Table 86 describes the system requirement SR53_NonUserDataDatabase.

SR53_NonUserDataDatabase		
Priority: High	Requirement: Essential	Stability: High
Description: The server shall have a database only to store the data of diagnostics, diseases and recovery plans.		

Table 86. SR53_NonUserDataDatabase

SR54_UserDataDatabaseLocation

The Table 87 describes the system requirement SR54_UserDataDatabaseLocation.

SR54_UserDataDatabaseLocation		
Priority: High	Requirement: Desirable	Stability: Medium
Description: The database where the server stores the data of the all patients/users should be located at the server localhost with port 3306.		

Table 87. SR54_UserDataDatabaseLocation

SR55_NonUserDataDatabaseLocation

The Table 88 describes the system requirement SR55_NonUserDataDatabaseLocation.

SR55_NonUserDataDatabaseLocation		
Priority: High	Requirement: Desirable	Stability: Medium
Description: The database where the server stores the data of the diagnostics, diseases and recovery plans should be located at the server localhost with port 3306.		

Table 88. SR55_NonUserDataDatabaseLocation

SR56_ServerIP

The Table 89 describes the system requirement SR56_ServerIP.

SR56_ServerIP		
Priority: High	Requirement: Desirable	Stability: Low
Description: The IP address used for the Android application to communicate with the server should be the 192.168.1.41.		

Table 89. SR56_ServerIP

SR57_ServerPort

The Table 90 describes the system requirement SR57_ServerPort.

SR57_ServerPort		
Priority: High	Requirement: Desirable	Stability: Low
Description: The port number used for the Android application to communicate with the server should be the 60102.		

Table 90. SR57_ServerPort

5.3.5.3. Traceability Matrix

The Table 91 has the traceability matrix that joins the user and system requirements.

	UR																		
SR	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19
01	x	x																	
02									x		x		x						
03									x		x		x						
04												x	x			x			
05	x	x							x			x	x			x			
06	x	x							x			x	x			x			
07	x																		
08		x																	
09				x															
10					x														
11									x										
12									x										
13									x										
14									x										
15												x							
16													x						
17																x			
18	x	x		x	x							x	x			x			
19	x	x		x	x							x	x			x			
20	x	x		x	x							x	x			x			
21	x	x		x	x							x	x			x			
22																			
23	x	x																	
24			x	x	x	x	x	x	x	x	x	x	x	x	x	x			
25	x																		
26	x																		
27		x																	
28		x																	
29													x						
30													x						
31							x												
32								x											
33									x										
34									x										
35											x								
36											x								
37											x								
38												x							
39						x													
40			x																
41				x	x														
42				x															
43					x														
44										x									
45														x					
46															x				
47																x			
48																		x	
49	x	x		x	x							x	x			x			x
50	x	x		x	x				x			x	x			x			
51																	x		
52	x	x		x	x														
53									x			x	x			x			
54	x	x		x	x														
55									x			x	x			x			
56	x	x		x	x				x			x	x			x			
57	x	x		x	x				x			x	x			x			

Table 91. System requirements traceability matrix

5.4. System Design

In the following section a general overview of the system architecture is introduced to understand the major components that build the system. Then the class diagrams of the classes implemented that form the server and the Android application are shown along with the traceability matrix that connects the requirements with the design. To finish, each of the methods that build the classes are explained.

5.4.1. General System Architecture

The main purpose of the system is to diagnose a disease depending on the symptoms introduced through mobile device. The Figure 21 shows the two components implemented with each of the modules each component has. A description of the components is provided below and the modules inside each of the components are detailed in section 5.2.

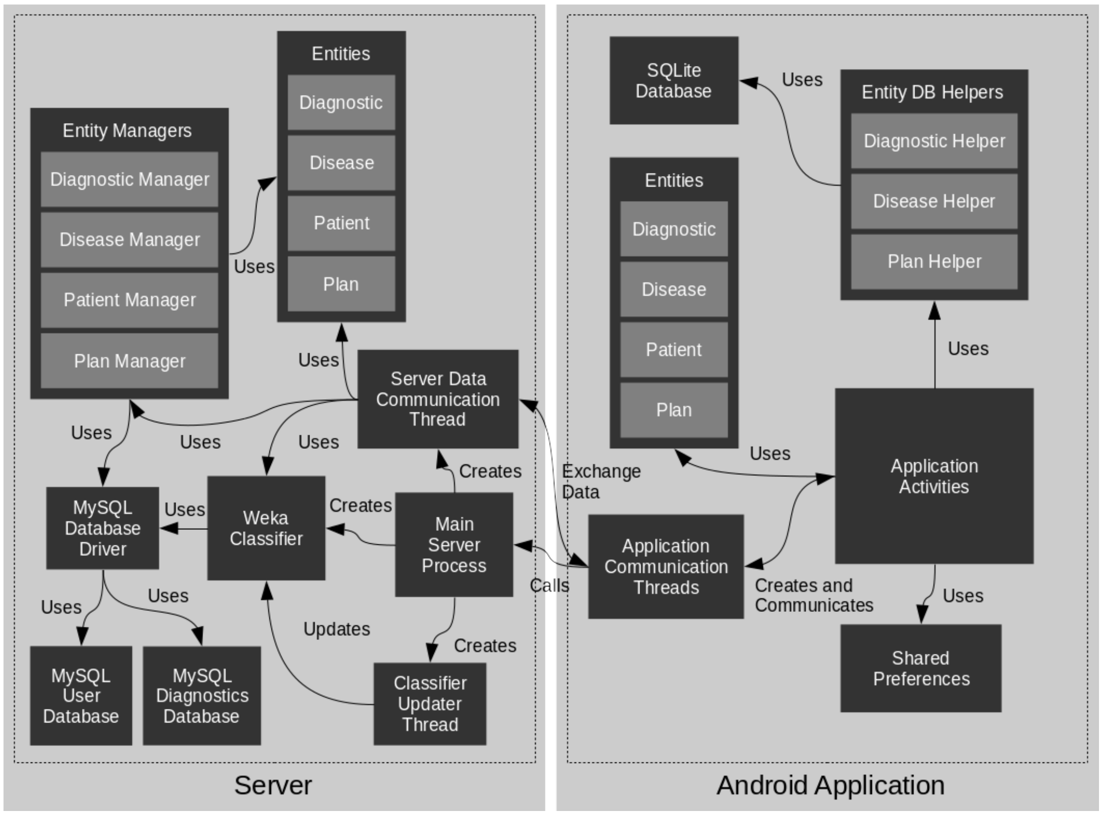


Figure 21. General System Architecture

Server

The server has as purpose the management of user accounts, the classification of the symptoms of the users to retrieve a set of possible injuries or diseases that the patient has and a possible set of plans to make the user feel better, and to store the diagnostics with the user opinions to improve future diagnostics.

Android Application

The Android application provides the server with the users and symptoms. The Android application has as main purpose of providing an interface to the user to diagnose the injury or disease associated to the symptoms that are acquired, or well through the mobile device microphone with help of the speech recognizer provided by Android, or through the keyboard provided also by the Android platform. Once the user has selected the possible injury or disease, it is saved along with its therapy to improve the condition of the user and to later be evaluated to improve the quality of the Weka Classifier.

5.4.2. System Modules

The following sections 5.2.1, Server Modules, and 5.2.2, Android Application Modules, will provide a brief explanation of the modules that conform each component of the system. In each of the sections first a graphical representation is shown and a brief explanation of each module is given.

5.4.2.1. Server Modules

The modules that are inside the Server of the system is shown in the Figure 22.

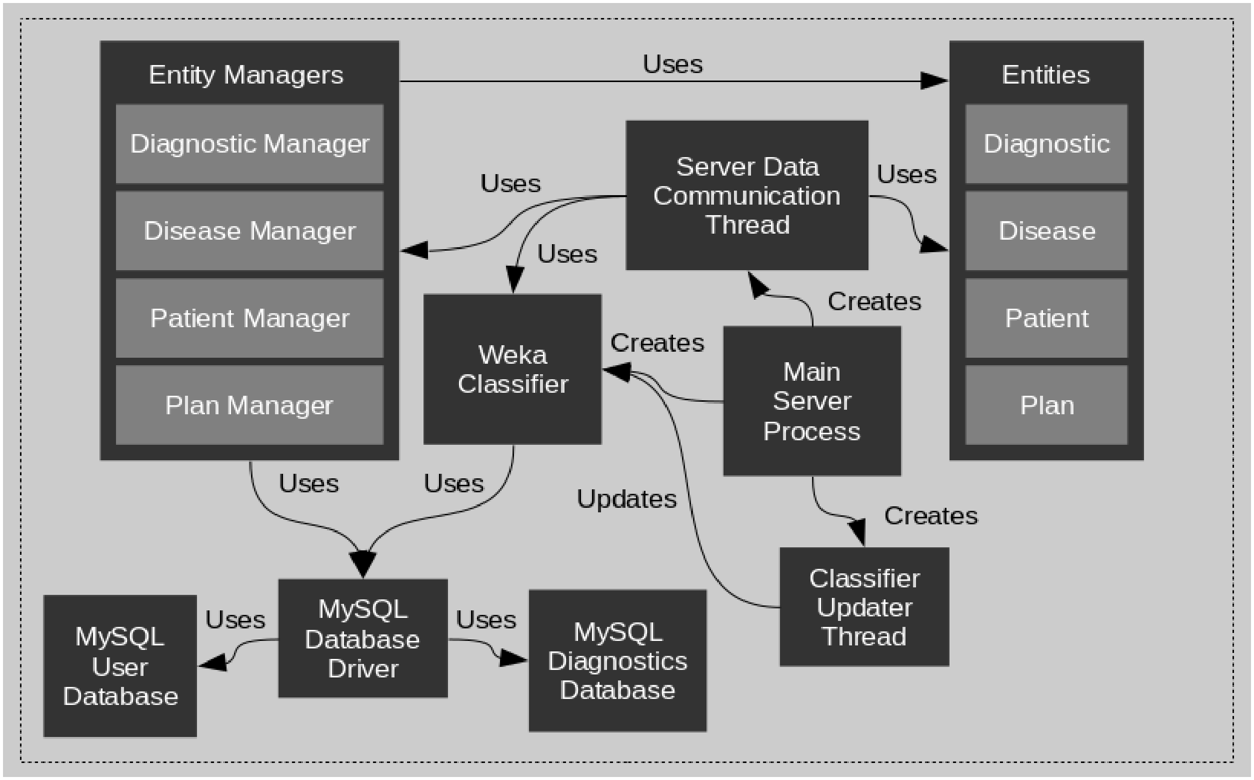


Figure 22. Server General Architecture

Classifier Updater Thread

The Classifier Update Thread module , as its own name indicates, maintains the Classifier up to date, each time a certain time interval is surpassed, the Classifier Updater Thread calls the Classifier in order to update it.

Entities

The Entities module contains the classes that represent the entities or tables in the MySQL Database module. The existing entities inside the Entities module are the patients or users that use the application, the diseases, the plans containing the therapies to counter the disease, and the diagnostics that provide a relationship between the patient, the disease, the plan, the symptoms that the patient has and the result of the diagnostic.

Entity Managers

The entity manager module provides an interface for better communication between the MySQL Database Driver module and the Entities module in order to provide a clear code and a better understanding of the design.

Main Server Process

The Main Server Process module initializes and gets the system resources and listen the incoming connections indefinitely. The modules that are initialized first are the Weka Classifier and the Classifier Updater Thread, then each time a new connection is created a new Server Data Communication Thread is created in order to manage the incoming connection.

MySQL Database Driver

The MySQL Database Driver module is an interface to communicate the database with the rest of the modules through the utilization of the Java Database Connectivity library inside Java, also known as JDBC.

MySQL Diagnostics Database

The MySQL Diagnostic Database module stores and retrieves all the entities inside the Entities module except the patient entity.

For better understanding of the MySQL Diagnostics Database module go to section 3.1.3.

MySQL User Database

The MySQL User Database module stores and retrieves the patients entity inside the Entities module.

For better understanding of the MySQL User Database module go to section 3.1.3.

Server Data Communication Thread

The Server Data Communication Thread module receives the connection that the Main Process module got in order to process the command that has arrived through the socket. The commands the Server Data Communication Thread process are the following:

- **Add user.** Command that creates a new user and stores it inside the database through the Entity Manager module.
- **Get user.** Command that gets an existing user from the database through the use of the Entity Manager module.
- **Edit user.** Command that edits the data of an existing user from the database with the new data received from the socket through the Entity Manager module.
- **Delete user.** Command that removes an existing user from the database through the Entity Manager module.
- **Check symptoms treatments.** Command that receives a symptom, and through the use of the Weka Classifier module, retrieves to the user a set of injuries or diseases associated with the symptom and the plans associated with the treatment of the injury or disease.
- **Save diagnostic.** Command that receives the selected disease and plan, along with the symptom of a patient to create a new diagnostic, associate it with a patient already in the database, and saves it in the database through the Entity Manager module.
- **Get user diagnostics.** Command that retrieves all the active diagnostics of a patient that are not currently in the patient possession from the database through the Entity Manager module.

- **Evaluate diagnostic.** Command that receives an user rating to the diagnostic of the symptom evaluated and stores it in the database through the Entity Manager module to improve incoming symptom classifications trough the Weka Classifier module.

Weka Classifier

The Weka Classifier module communicates with the database through the MySQL Database Driver module, get all the symptoms stored in the database along with the diagnostics and through the functions provided by the Weka library to prepare data, the data inside the database is modified to later create a Naive Bayes classifier.

Also, if a symptom is received for the Weka Classifier module to classify, the symptom is prepared to be introduced inside the classifier and get the set of diseases associated with the symptoms of the patient along with its percentages.

For better understanding of the Weka Classifier module go to section 3.1.5.

5.4.2.2. Android Application Modules

The modules that are inside the Android application of the system is shown in the Figure 23.

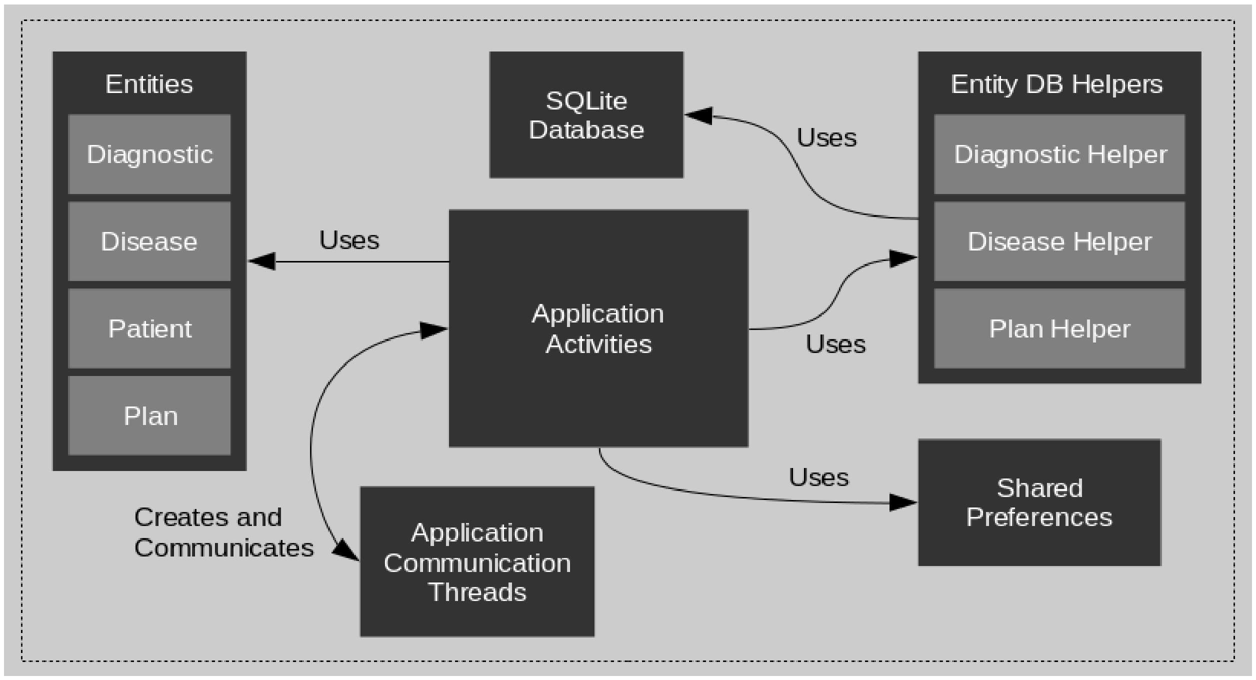


Figure 23. Android Application General Architecture

Application Activities

The Application Activities module contains all the layouts and functionalities of the interface of the Android Application.

Application Communication Thread

The Application Communication Thread module is launched each time an activity inside the Application Activities module needs to communicate with the server. For each command the application wants to send, a new thread must be launched.

Entities

The Entities module contains the classes that represents the entities or tables in the SQLite Database module and in the Shared Preferences file of users. The existing entities inside the Entities module are the patients or users that use the application, the diseases, the plans containing the therapies to counter the disease, and the diagnostics that provide a relationship among the patient, the disease, the plan and the symptoms that the patient has and the result of the diagnostic.

Entity Manager

The entity manager module provides an interface for better communication between the SQLite Database module and the Entities module in order to provide a clear code and a better understanding of the design. Even though there is an entity called patients inside the Entities module, there is no patient manager inside the Entities Manager module because only one patient is stored in the application inside the shared preferences file of users.

Speech Recognizer

The speech recognizer module can be launched each time a patient wants to introduce a new symptom. The speech recognizer receives an audio stream and sends it to a server from the Google Inc company for them to process (Google Inc, n.d.c). Once the stream is processed, a string of characters is received with a phrase (in text form) said by the patient.

SQLite Database

The SQLite database module stores and retrieves all the entities inside the Entities module.

For better understanding of the MySQL Database module go to section 3.1.4.

5.4.3. Class Diagram

In this section the different classes forming the system are described.

Due to the server and the Android application are two different components two different subsections are provided below for each of the components. However, because of the size of the classes, a different table is provided for each of the classes inside each of the class diagrams.

5.4.3.1. Class Diagram Specification

The tables that describe each of the classes of the diagram has the following structure.

Name of the table	
Dependencies	
Purpose	
Variables	
Functions	
Functionality	

- **Name of the table:** Name of the table indicates the name of the component. This name has the following format "CX_NameOfClass", where C means Component and X is the name of the class.
- **Dependencies.** Dependencies describes all the classes needed for the construction of the class described.
- **Purpose.** Purpose lists all the requirements that are fulfilled through the creation of the class.
- **Variables.** Variables lists all the global variables inside the class.
- **Functions.** Functions lists all the functions inside the class.
- **Functionality.** Functionality describes briefly which is the objective of the class and what is it used for.

5.4.3.2. Server Class Diagram

The Figure 24 represents the class diagram of the server. After the figure 24 with the server class diagram, the tables with the information of each of the components in the diagram are shown.

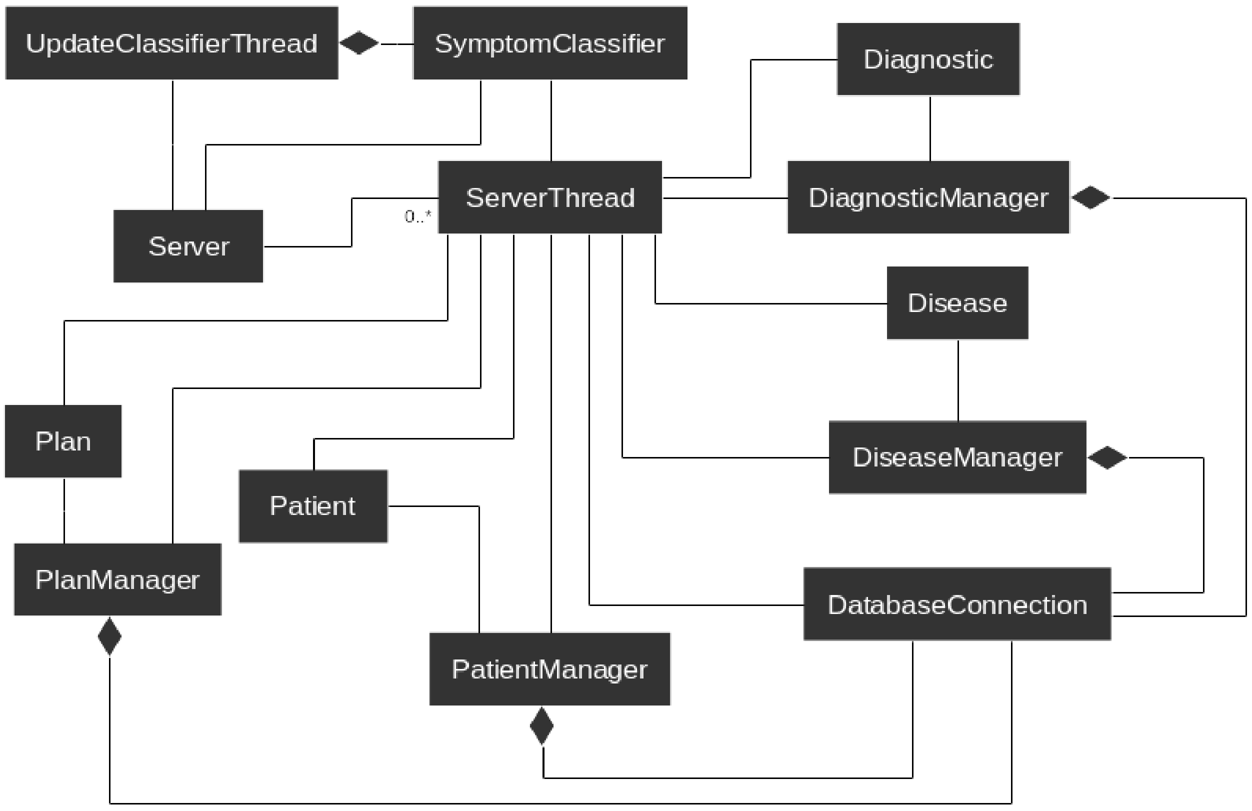


Figure 24. Server Class Diagram

Component 01: DatabaseConnection

The Table 92 describes the server class DatabaseConnection.

C01_DatabaseConnection	
Dependencies	- none -
Purpose	SR01_UserAccount, SR02_RecoveryPlan, SR03_Disease, SR04_Diagnostic

Variables	`- connection: Connection`
	`- db_driver: String`
	`- db_url: String`
	`- db_username: String`
	`- db_password: String`

Functions	`<<constructor>> DatabaseConnection(String driver, String url, String username, String password)`
	`+ getConnection(): Connection`
	`+ connect(): Connection`
	`+ disconnect(): void`

Functionality	The DatabaseConnection server provides an easy interface to connect an disconnect from the server. To establish the connection to the database with the DatabaseConnection class, the developer only needs to introduce the driver location, the url of the database, and the username and password associated to the database. DatabaseConnection uses JDBC to connect to the database.

Table 92. C01_DatabaseConnection

Component 02: Diagnostic

The Table 93 describes the server class Diagnostic.

C02_Diagnostic	
Dependencies	- none -
Purpose	SR04_Diagnostic

Variables	`- diagnostic_id: int`
	`- disease_id: int`
	`- patient_id: int`
	`- plan_id: int`
	`- result: int`
	`- diagnostic_date: String`
	`- symptoms: String`

Functions	`<<constructor>> Diagnostic()`
	`+ getID(): int`
	`+ getDiseaseID(): int`
	`+ getPatientID(): int`
	`+ getPlanID(): int`
	`+ getResult(): int`
	`+ getDiagnosticDate(): String`
	`+ getSymptoms(): String`
	`+ setID(int diagnostic_id): void`
	`+ setDiseaseID(int disease_id): void`
	`+ setPatientID(int patient_id): void`
	`+ setPlanID(int plan_id): void`
	`+ setResult(int result): void`
	`+ setDiagnosticDate(String diagnostic_date): void`
	`+ setSymptoms(String symptoms): void`

Functionality	The Diagnostic class is associated to the “diagnostic” table inside the database. This table stores the id of the diagnostic, the symptoms of the patient, the disease associated to the symptoms, the plan associated to the disease, the result of the diagnostic, its date and the patient that required the service to diagnose the symptoms.

Table 93. C02_Diagnostic

Component 03: DiagnosticManager

The Table 94 describes the server class DiagnosticManager.

C03_DiagnosticManager	
Dependencies	DatabaseConnection, Diagnostic
Purpose	SR04_Diagnostic
Variables	`- databaseConnection: DatabaseConnection`

Functions	`<<constructor>> DiagnosticManager(DatabaseConnection databaseConnection)`
	`+ addDiagnostic(String symptoms, int patient_id, int disease_id, int plan_id): int`
	`+ deleteDiagnostic(int diagnostic_id): int`
	`+ getDiagnostic(int diagnostic_id): Diagnostic`
	`+ getActiveDiagnosticsFromPatient(int patient_id): ArrayList<Diagnostic>`
	`+ updateDiagnosticResult(int diagnostic_id, int result): int`
	`- executeGetDiagnosticsQuery(String query): ArrayList<Diagnostic>`

Functionality	The DiagnosticManager class is an interface to ease the deletion, insertion, modification and recollection of diagnostics from the database through the DatabaseConnection class.

Table 94. C03_DiagnosticManager

Component 04: Disease

The Table 95 describes the server class Disease.

C04_Disease	
Dependencies	- none -
Purpose	SR03_Disease

Variables	`- disease_id: int`
	`- name: String`
	`- description: String`
	`- plan_id: int`

Functions	`<<constructor>> Disease()`
	`+ getID(): int`
	`+ getName(): String`
	`+ getDescription(): String`
	`+ getPlanID(): int`
	`+ setID(int disease_id): void`
	`+ setName(String name): void`
	`+ setDescription(String description): void`
	`+ setPlanID(int plan_id): void`

Functionality	The Disease class is associated to the “disease” table inside the database. This table stores the id of the disease, its name, its description and the id of the plan associated.

Table 95. C04_Disease

Component 05: DiseaseManager

The Table 96 describes the server class DiseaseManager.

C05_DiseaseManager	
Dependencies	DatabaseConnection, Disease
Purpose	SR03_Disease
Variables	`- databaseConnection: DatabaseConnection`

Functions	`<<constructor>> DiseaseManager(DatabaseConnection databaseConnection)`
	`+ getDiseaseByID(int disease_id): Disease`

Functionality	The DiseaseManager class is an interface to ease the recollection of diseases from the database through the DatabaseConnection class.

Table 96. C05_DiseaseManager

Component 06: Patient

The Table 97 describes the server class Patient.

C06_Patient	
Dependencies	- none -
Purpose	SR01_UserAccount

Variables	`- patient_id: int`
	`- birthdate: String`
	`- email: String`
	`- password: String`
	`- surname: String`

Functions	`<<constructor>> Patient()`
	`+ getID(): int`
	`+ getBirthDate(): String`
	`+ getEmail(): String`
	`+ getName(): String`
	`+ getPassword(): String`
	`+ getSurname(): String`
	`+ setID(int patient_id): void`
	`+ setBirthDate(String birthdate): void`
	`+ setEmail(String email): void`
	`+ setPassword(String password): void`
	`+ setSurname(String surname): void`

Functionality	The Patient class is associated to the “patient” table inside the database. This table stores the id of the patient, its email and password, its name and surname, and its birthdate.

Table 97. C06_Patient

Component 07: PatientManager

The Table 98 describes the server class PatientManager.

C07_PatientManager	
Dependencies	DatabaseConnection, Patient
Purpose	SR01_UserAccount
Variables	`- databaseConnection: DatabaseConnection`

Functions	`<<constructor>> PatientManager(DatabaseConnection databaseConnection)`
	`+ addPatient(Patient patient): int`
	`+ addPatient(String birthdate, String email, String name, String password, String surname): int`
	`+ deletePatient(String email): int`
	`+ getPatientByID(int patient_id): Patient`
	`+ getPatientByEmail(String email): Patient`
	`+ updateDiagnosticResult(Patient oldPatient, Patient newPatient): int`
	`- executeGetPatientsQuery(String query): ArrayList<Patient>`

Functionality	The PatientManager class is an interface to ease the deletion, insertion, modification and recollection of patients from the database through the DatabaseConnection class.

Table 98. C07_PatientManager

Component 08: Plan

The Table 99 describes the server class Plan.

C08_Plan	
Dependencies	- none -
Purpose	SR02_RecoveryPlan

Variables	`- plan_id: int`
	`- description: String`

Functions	`<<constructor>> Plan()`
	`+ getID(): int`
	`+ getDescription(): String`
	`+ setID(int plan_id): void`
	`+ setDescription(String description): void`

Functionality	The Plan class is associated to the "plan" table inside the database. This table stores the id of the plan and its description.

Table 99. C08_Plan

Component 09: PlanManager

The Table 100 describes the server class PlanManager.

C09_PlanManager	
Dependencies	DatabaseConnection, Plan
Purpose	SR02_RecoveryPlan
Variables	`- databaseConnection: DatabaseConnection`

Functions	`<<constructor>> PlanManager(DatabaseConnection databaseConnection)`
	`+ getPlanByID(int plan_id): Plan`

Functionality	The PlanManager class is an interface to ease the recollection of plans from the database through the DatabaseConnection class.

Table 100. C09_PlanManager

Component 10: Server

The Table 101 describes the server class Server.

C10_Server	
Dependencies	ServerThread, SymptomClassifier, UpdateClassifierThread
Purpose	SR06_ServerThreads, SR56_ServerIP, SR57_ServerPort
Variables	`- K_CONNECTION_SOCKET: int`
Functions	`+ main(String[] args): void`
Functionality	The Server class initializes the classifier that is used by the ServerThread class to classify symptoms, launches the UpdateClassifierThread, that, as its own name indicates, is in charge of updating the classifier, and listen to the incoming connections inside an infinite loop. Each time a new connection is received, a new ServerThread is created that is in charge of the communication with the client.

Table 101. C10_Server

Component 11: ServerThread

The Table 102 describes the server class ServerThread.

C11_ServerThread	
Dependencies	DatabaseConnetion, Diagnostic, DiagnosticManager, Disease, DiseaseManager, Patient, PatientManager, Plan, PlanManager, SymptomClassifier
Purpose	SR05_RequestCommandNumber, SR07_ServerUserCreation, SR08_UserRetrieval, SR09_ServerUserModification, SR10_ServerUserDeletion, SR13_ServerDiagnosticClassification, SR14_ServerDiagnosticClassificationOutput, SR15_ServerDiagnosticStorage SR16_ServerActiveDiagnosticRetrieval, SR17_ServerDiagnosticEvaluation, SR18_ServerConfirmationCode, SR19_ServerUnexpectedErrorCode, SR20_ServerWrongEmailPasswordCode, SR21_ServerBlankFieldCode, SR52_UserDataDatabase, SR53_NonUserDataDatabase, SR54_UserDataDatabaseLocation, SR55_UserDataDatabaseLocation

Variables	`- K_DB_DIAGNOSTIC_DRIVER: String`
	`- K_DB_DIAGNOSTIC_URL: String`
	`- K_DB_DIAGNOSTIC_USERNAME: String`
	`- K_DB_DIAGNOSTIC_PASSWORD: String`
	`- K_DB_PATIENT_DRIVER: String`
	`- K_DB_PATIENT_URL: String`
	`- K_DB_PATIENT_USERNAME: String`
	`- K_DB_PATIENT_PASSWORD: String`
	`- objectInputStream: ObjectInputStream`
	`- objectOutputStream: ObjectOutputStream`
	`- socket: Socket`
	`- symptomClassifier: SymptomClassifier`

Functions	`<<constructor>> ServerThread(Socket socket,`
	`      SymptomClassifier symptomClassifier)`
	`+ run(): void`
	`- addUser(): void`
	`- getUser(): void`
	`- editUser(): void`
	`- deleteUser(): void`
	`- checkSymptomTreatment(): void`
	`- saveDiagnostic(): void`
	`- getUserDiagnostics(): void`
	`- evaluateDiagnostics(): void`

Functionality	The SeverThread class is where all the communication with the client and all the commands are executed. Each time the user of the

	application wants to create, delete, get or modify a patient, and each time the patient wants to evaluate a new symptom, store it, retrieve the active diagnostics and evaluate them, a connection has to be made with the Server for later the ServerThread execute the command.

Table 102. C11_ServerThread

Component 12: SymptomClassifier

The Table 103 describes the server class SymptomClassifier.

C12_SymptomClassifier	
Dependencies	- none -
Purpose	SR11_ServerNaiveBayesClassificator, SR13_ServerDiagnosticClassification

Variables	`- data: Instances`
	`- classifier: NaiveBayes`
	`- filter: StringToWordVector`

Functions	`<<constructor>> SymptomClassifier()`
	`+ update(): void`
	`+ classify(String symptoms, int numDiseases,`
	`      ArrayList<Integer>, ArrayList<Double>): int`
	`- makeInstance(String string, Instances instances):`
	`      Instance`

Functionality	The SymptomClassifier class main purpose is to classify the symptoms introduce by the user, returning as a result the id of the diseases associated to that symptom and the percentage of the patient having each of the diseases.

Table 103. C12_SymptomClassifier

Component 13: UpdateClassifierThread

The Table 104 describes the server class UpdateClassifierThread.

C13_UpdateClassifierThread	
Dependencies	SymptomClassifier
Purpose	SR12_ServerClassifierUpdater

Variables	`- K_MILLISECONDS_HOUR: int`
	`- symptomClassifier: SymptomClassifier`

Functions	`<<constructor>> UpdateClassifierThread(SymptomClassifier symptomClassifier)`
	`+ run()`

Functionality	The UpdateClassifierThread class is a thread that runs indefinitely with the purpose of maintaining the SymptomClassifier that the Server class has passed to the UpdateClassifierThread class. It updates the classifier at least one time each hour.

Table 104. C13_UpdateClassifierThread

5.4.3.3. Android Application Class Diagram

For better comprehension of the Android class diagram, the class diagram in the Figure 25 shows the relation among activities. Once the class diagram showing the activities, one class diagram for each activity is displayed. After all the class diagrams have been introduced, the tables with the information of each of the components in the diagrams are shown.

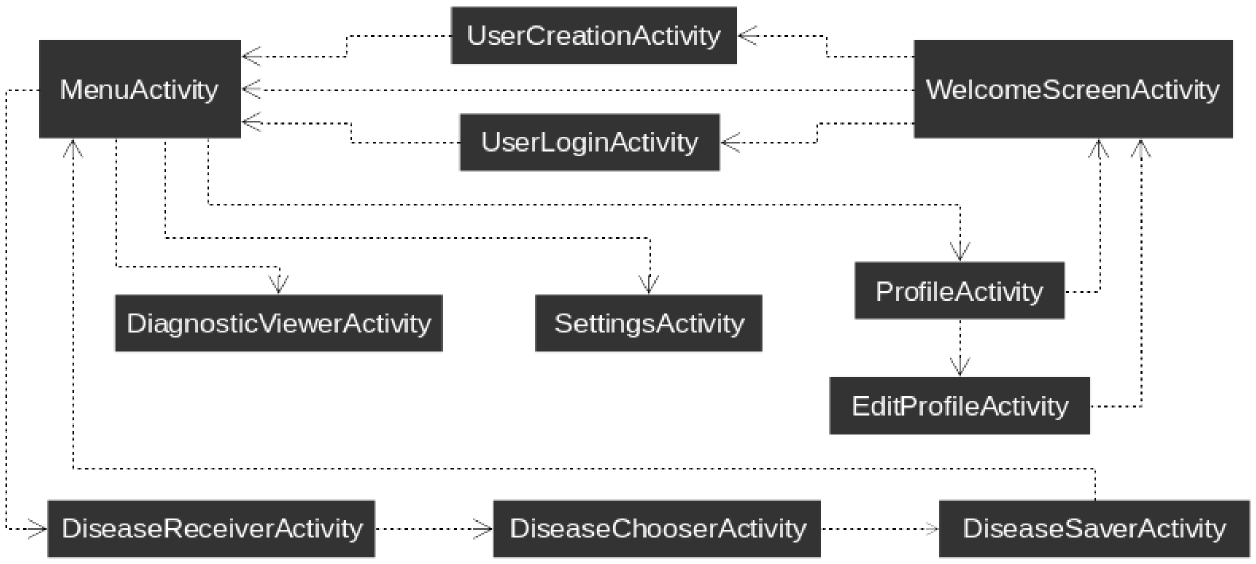


Figure 25. Android Application Activities Class Diagram

DiagnosticViewerActivity Class Diagram

The Figure 26 shows the relations the class DiagnosticViewerActivity has.

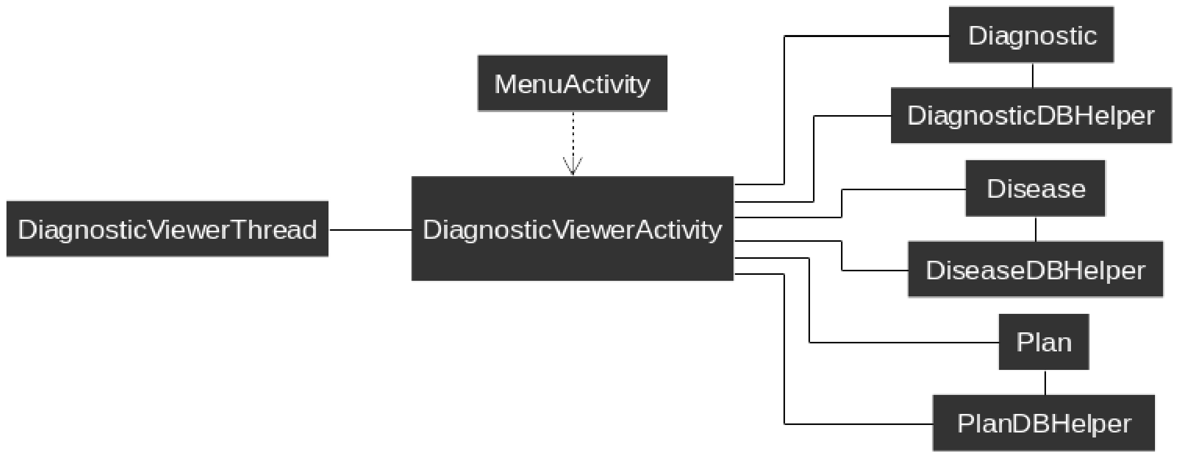


Figure 26. DiagnosticViewerActivity Class Diagram

DiseaseChooserActivity Class Diagram

The Figure 27 shows the relations the class DiseaseChooserActivity has.

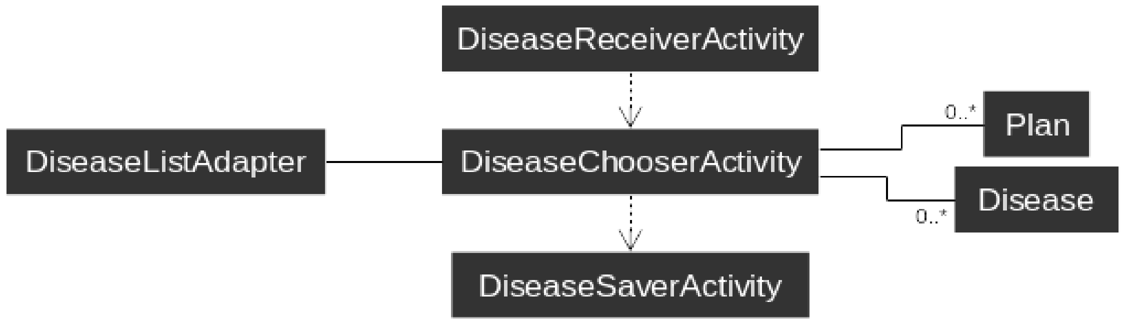


Figure 27. DiseaseChooserActivity Class Diagram

DiseaseReceiverActivity Class Diagram

The Figure 28 shows the relations the class DiseaseReceiverActivity has.

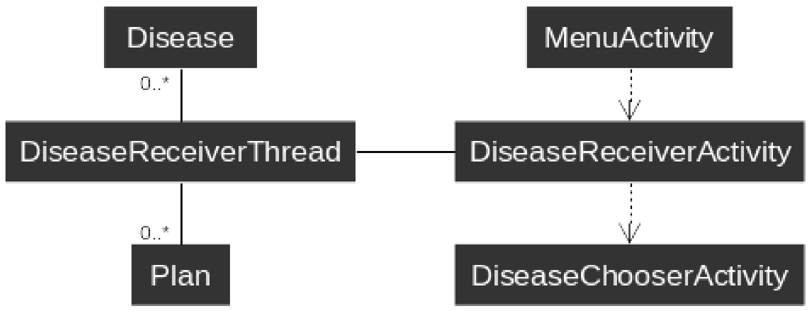


Figure 28. DiseaseReceiverActivity Class Diagram

DiseaseSaverActivity Class Diagram

The Figure 29 shows the relations the class DiseaseSaverActivity has.

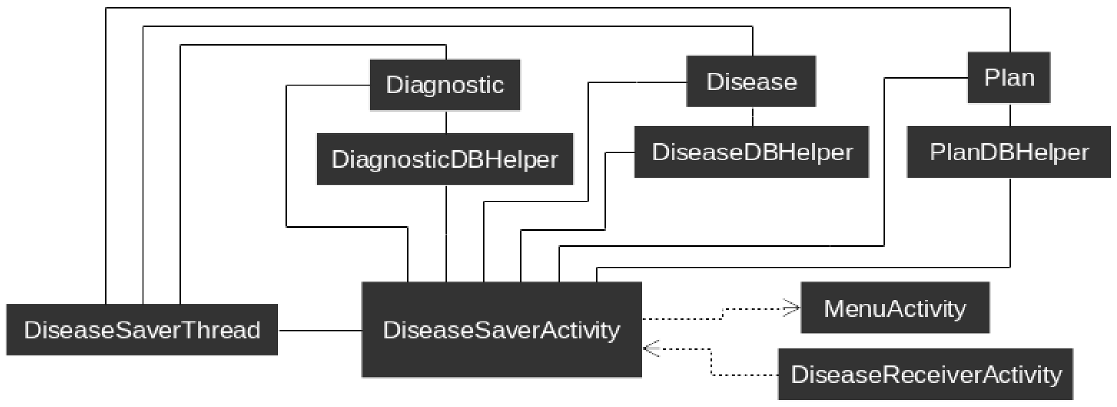


Figure 29. DiseaseSaverActivity Class Diagram

EditProfileActivity Class Diagram

The Figure 30 shows the relations the class EditProfileActivity has.

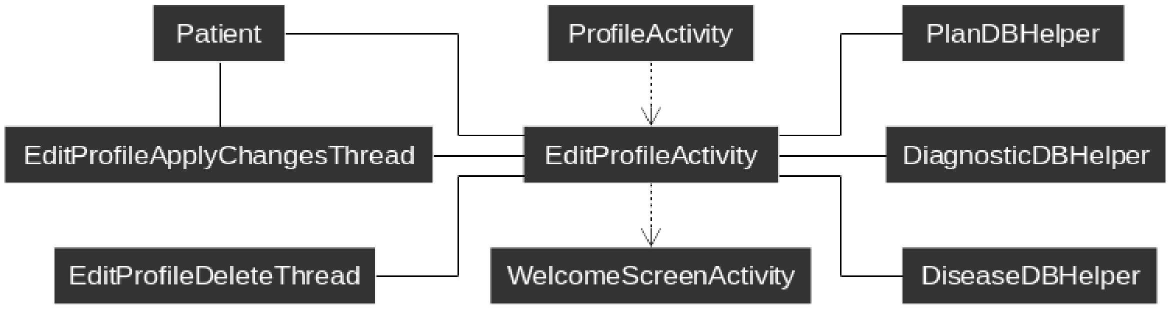


Figure 30. EditProfileActivity Class Diagram

MenuActivity Class Diagram

The Figure 31 shows the relations the class MenuActivity has.

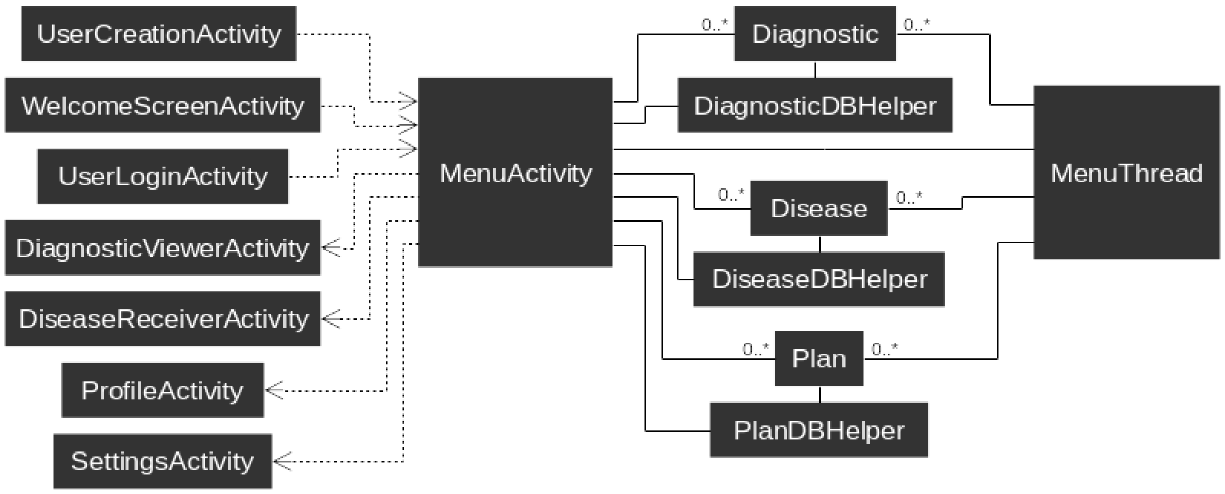


Figure 31. MenuActivity Class Diagram

ProfileActivity Class Diagram

The Figure 32 shows the relations the class ProfileActivity has.

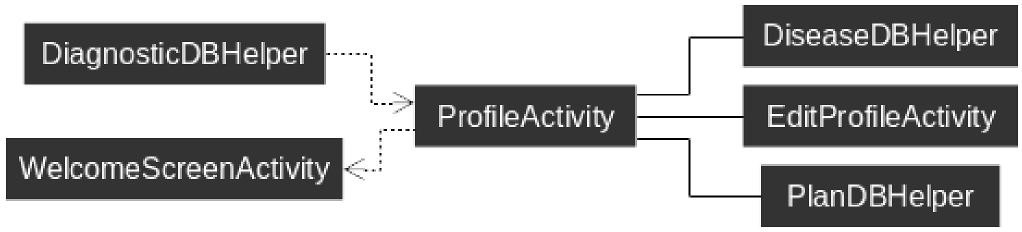


Figure 32. ProfileActivity Class Diagram

SettingsActivity Class Diagram

The Figure 33 shows the relations the class SettingsActivity has.

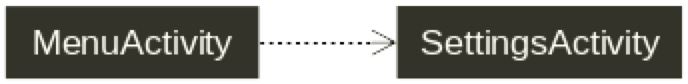


Figure 33. SettingsActivity Class Diagram

UserCreationActivity Class Diagram

The Figure 34 shows the relations the class UserCreationActivity has.

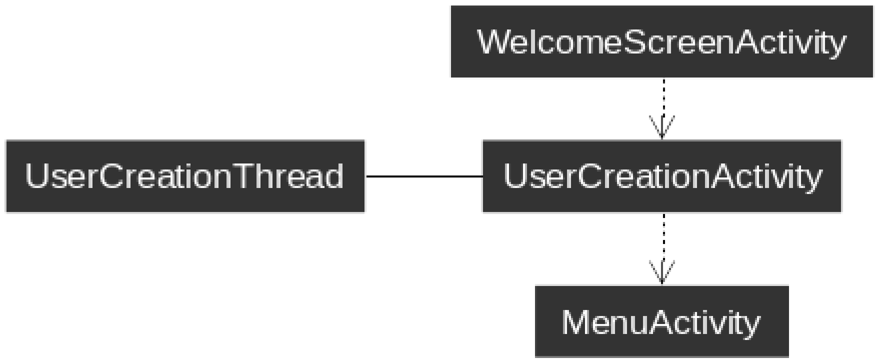


Figure 34. UserCreationActivity Class Diagram

UserLoginActivity Class Diagram

The Figure 35 shows the relations the class UserLoginActivity has.

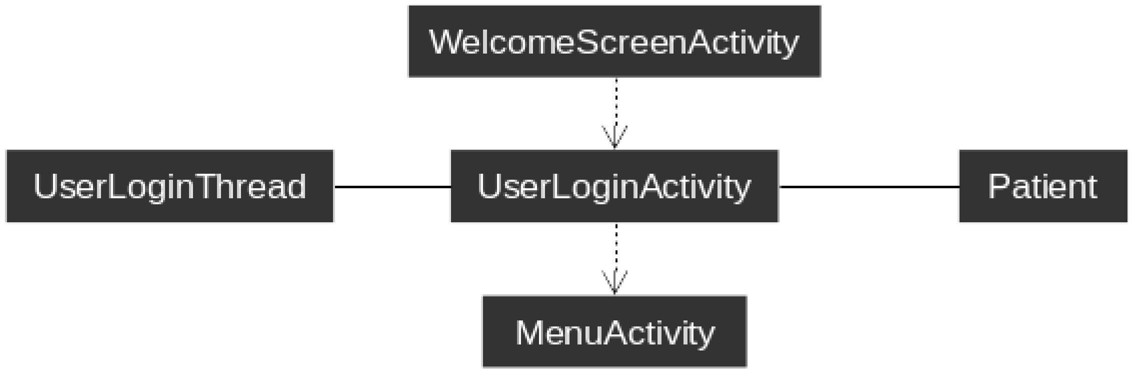


Figure 35. UserLoginActivity Class Diagram

WelcomeScreenActivity Class Diagram

The Figure 36 shows the relations the class WelcomeScreenActivity has.

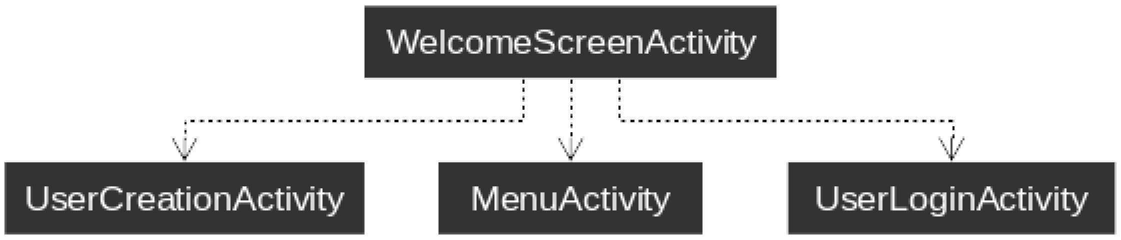


Figure 36. WelcomeScreenActivity Class Diagram

Component 14: Diagnostic

The Table 105 describes the application class Diagnostic.

C14_Diagnostic	
Dependencies	- none -
Purpose	SR04_Diagnostic

Variables	`- diagnostic_id: int`
	`- disease_id: int`
	`- patient_id: int`
	`- plan_id: int`
	`- result: int`
	`- diagnostic_date: String`
	`- symptoms: String`

Functions	`<<constructor>> Diagnostic()`
	`+ getID(): int`
	`+ getDiseaseID(): int`
	`+ getPatientID(): int`
	`+ getPlanID(): int`
	`+ getResult(): int`
	`+ getDiagnosticDate(): String`
	`+ getSymptoms(): String`
	`+ setID(int diagnostic_id): void`
	`+ setDiseaseID(int disease_id): void`
	`+ setPatientID(int patient_id): void`
	`+ setPlanID(int plan_id): void`
	`+ setResult(int result): void`
	`+ setDiagnosticDate(String diagnostic_date): void`
	`+ setSymptoms(String symptoms): void`

Functionality	The Diagnostic class is associated to the “diagnostic” table inside the database of the server. This table stores the id of the diagnostic, the symptoms of the patient, the disease associated to the symptoms, the plan associated to the disease, the result of the diagnostic, its date and the patient that required the service to diagnose the symptoms.

Table 105. C14_Diagnostic

Component 15: DiagnosticDBHelper

The Table 106 describes the application class DiagnosticDBHelper.

C15_DiagnosticDBHelper	
Dependencies	Diagnostic
Purpose	SR04_Diagnostic, SR51_MinimumSDKVersion

Variables	`- K_DATABASE_NAME: String`
	`- K_DATABASE_VERSION: int`
	`- K_DIAGNOSTIC_TABLE_NAME: String`
	`- K_DIAGNOSTIC_COLUMN_ID: String`
	`- K_DIAGNOSTIC_COLUMN_DISEASE_ID: String`
	`- K_DIAGNOSTIC_COLUMN_PATIENT_ID: String`
	`- K_DIAGNOSTIC_COLUMN_PLAN_ID: String`
	`- K_DIAGNOSTIC_COLUMN_RESULT: String`
	`- K_DIAGNOSTIC_COLUMN_DATE: String`
	`- K_DIAGNOSTIC_COLUMN_SYMPTOMS: String`

Functions	`<<constructor>> DiagnosticDBHelper(Context context)`
	`+ onCreate(SQLiteDatabase db): void`
	`+ onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion): void`
	`+ onDowngrade(SQLiteDatabase db, int oldVersion, int newVersion): void diagnostic_id): void`
	`+ deleteDiagnostic(int diagnostic_id): void`
	`+ getDiagnostic(int diagnostic_id): Diagnostic`
	`+ getDiagnostics(): ArrayList<Diagnostic>`
	`+ getDiagnosticsByDiseaseID(int disease_id): ArrayList<Diagnostic>`
	`+ getDiagnosticsByPlanID(int plan_id): ArrayList<Diagnostic>`
	`+ insertDiagnostic(Diagnostic diagnostic): void`

Functionality	The DiagnosticDBHelper class provides an interface to delete, insert and retrieve the diagnostics from the SQLite database inside the smartphone.

Table 106. C15_DiagnosticDBHelper

Component 16: DiagnosticViewerActivity

The Table 107 describes the application class DiagnosticViewerActivity.

C16_DiagnosticViewerActivity	
Dependencies	Diagnostic, DiagnosticDBHelper, DiagnosticViewerThread, Disease, DiseaseDBHelper, Plan, PlanDBHelper
Purpose	SR45_ApplicationDiagnosticViewer, SR46_ApplicationDiagnosticRemoval, SR47_ApplicationDiagnosticEvaluation, SR48_ApplicationInternet, SR51_MinimumSDKVersion

Variables	`- K_SP_USER: String`
	`- K_SP_USER_EMAIL: String`
	`- K_SP_USER_PASSWORD: String`
	`- buttonBack: Button`
	`- buttonDisease: Button`
	`- buttonFinish: Button`
	`- buttonPlan: Button`
	`- buttonSymptoms: Button`
	`- contentWindow: LinearLayout`
	`- diagnostic: Diagnostic`
	`- disease: Disease`
	`- plan: Plan`
	`- email: String`
	`- password: String`
	`- thread: Thread`

Functions	`+ onCreate(Bundle savedInstance): void`
	`- onClickListenerFunction_ButtonDisease(): void`
	`- onClickListenerFunction_ButtonFinish(): void`
	`- onClickListenerFunction_ButtonPlan(): void`
	`- onClickListenerFunction_ButtonSymptoms(): void`
	`- removeDiagnostic(): void`
	`- isNetworkAvailable(): boolean`

Functionality	The DiagnosticDBHelper class provides an user interface to show to the patient the complete diagnostic saved in the patient account.

Table 107. C16_DiagnosticViewerActivity

Component 17: DiagnosticViewerThread

The Table 108 describes the application class DiagnosticViewerThread.

C17_DiagnosticViewerThread	
Dependencies	- none -
Purpose	SR05_RequestCommandNumber, SR47_ApplicationDiagnosticEvaluation, SR49_PrivateDataRetrievalProtection, SR50_SystemCommunication, SR56_ServerIP, SR57_ServerPort

Variables	`- diagnotic_id: int`
	`- diagnostic_result: int`
	`- result: int`
	`- email: String`
	`- password: String`

Functions	`<<constructor>> DiagnosticViewerThread(String email, String password, int diagnostic_id, int diagnostic_result)`
	`+ getResult(): int`
	`+ run(): void`

Functionality	The DiagnosticViewerThread class is used from the DiagnosticViewerActivity class to communicate with the server and evaluate the diagnostic previously saved in the smartphone once the plan has finished.

Table 108. C17_DiagnosticViewerThread

Component 18: Disease

The Table 109 describes the application class Disease.

C18_Disease	
Dependencies	- none -
Purpose	SR03_Disease

Variables	`- disease_id: int`
	`- name: String`
	`- description: String`
	`- plan_id: int`

Functions	`<<constructor>> Disease()`
	`+ getID(): int`
	`+ getName(): String`
	`+ getDescription(): String`
	`+ getPlanID(): int`
	`+ setID(int disease_id): void`
	`+ setName(String name): void`
	`+ setDescription(String description): void`
	`+ setPlanID(int plan_id): void`

Functionality	The Disease class is associated to the “disease” table inside the database of the server. This table stores the id of the disease, its name, its description and the id of the plan associated.

Table 109. C18_Disease

Component 19: DiseaseChooserActivity

The Table 110 describes the application class DiseaseChooserActivity.

C19_DiseaseChooserActivity	
Dependencies	Disease, DiseaseListAdapter, Plan, DiseaseSaverActivity
Purpose	SR35_ApplicationSymptomsClassificationOutputViewer, SR36_ApplicationSymptomsClassificationOutputSelection, SR51_MinimumSDKVersion

Variables	`- successPercentageArray: double[]`
	`- diseaseArray: Disease[]`
	`- planArray: Plan[]`
	`- symptoms: String`

Functions	`+ onCreate(Bundle savedInstance): void`
	`- getExtras(): void`

Functionality	The DiseaseChooserActivity class provides an user interface to show to the patient the diseases that relate the most with the symptoms introduced through the DiseaseReceiverActivity class.

Table 110. C19_DiseaseChooserActivity

Component 20: DiseaseDBHelper

The Table 111 describes the application class DiseaseDBHelper.

C20_DiseaseDBHelper	
Dependencies	Disease
Purpose	SR03_Disease, SR51_MinimumSDKVersion

Variables	`- K_DATABASE_NAME: String`
	`- K_DATABASE_VERSION: int`
	`- K_DISEASE_TABLE_NAME: String`
	`- K_DISEASE_COLUMN_ID: String`
	`- K_DISEASE_COLUMN_NAME: String`
	`- K_DISEASE_COLUMN_DESCRIPTION: String`
	`- K_DIAGNOSTIC_COLUMN_PLAN_ID: String`

Functions	`<<constructor>> DiseaseDBHelper(Context context)`
	`+ onCreate(SQLiteDatabase db): void`
	`+ onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion): void`
	`+ onDowngrade(SQLiteDatabase db, int oldVersion, int newVersion): void diagnostic_id): void`
	`+ deleteDisease(int disease_id): void`
	`+ getDisease(int disease_id): Disease`
	`+ getDiseases(): ArrayList<Disease>`
	`+ insertDisease(Disease disease): void`

Functionality	The DiseaseDBHelper class provides an interface to delete, insert and retrieve the diseases from the SQLite database inside the smartphone.

Table 111. C20_DiseaseDBHelper

Component 21: DiseaseListAdapter

The Table 112 describes the application class DiseaseListAdapter.

C21_DiseaseListAdapter	
Dependencies	Disease
Purpose	SR35_ApplicationSymptomsClassificationOutputViewer, SR36_ApplicationSymptomsClassificationOutputSelection, SR51_MinimumSDKVersion

Variables	`- listData: Disease[]`
	`- layoutInflater: LayoutInflater`
	`- successPercentageArray: double[]`

Functions	`<<constructor>> DiseasListAdapter(Context context, Disease[]   listData, double[] successPercentageArray)`
	`+ getCount(): int`
	`+ getItem(int position): Object`
	`+ getItemId(int position): long`
	`+ getView(int position, View convertView, ViewGroup parent): View`

Functionality	The DiseaseListAdapter class is an interface to provide the ListView class from the Android API with a custom layout for each of the elements in the list.

Table 112. C21_DiseaseListAdapter

Component 22: DiseaseReceiverActivity

The Table 113 describes the application class DiseaseReceiverActivity.

C22_DiseaseReceiverActivity	
Dependencies	DiseaseReceiverThread, DiseaseChooserActivity
Purpose	SR31_ApplicationSymptomsKeyboardInput, SR32_ApplicationSymptomsSpeechRecognitionInput, SR33_ApplicationSymptomsClassification, SR34_ApplicationSymptomsClassificationOutput, SR48_ApplicationInternet, SR51_MinimumSDKVersion

Variables	`- K_REQ_CODE_SPEECH_INPUT: int`
	`- K_SP_SETTINGS: String`
	`- K_SP_SETTINGS_MAX_DISEASES: String`
	`- buttonCancel: Button`
	`- buttonSpeechRecognizer: Button`
	`- buttonSubmit: Button`
	`- editTextSpeechRecognizer: EditText`

Functions	`+ onCreate(Bundle savedInstance): void`
	`- onCLickListenerFunction_ButtonSpeechRecognizer: void`
	`- onClickListenerFunction_ButtonSubmit(): void`
	`- onActivityResult(int requestCode, int resultCode, Intent data): void`
	`- isNetworkAvailable(): boolean`

Functionality	The DiseaseReceiverActivity class provides an user interface to introduce through the keyboard or through the speech recognizer provided by the Android platform the symptoms of the patient. Once the symptoms have been collected, they are sent to the server through the DiseaseReceiverThread class.

Table 113. C22_DiseaseReceiverActivity

Component 23: DiseaseReceiverThread

The Table 114 describes the application class DiseaseReceiverThread.

C23_DiseaseReceiverThread	
Dependencies	- none -
Purpose	SR05_RequestCommandNumber, SR33_ApplicationSymptomsClassification, SR34_ApplicationSymptomsClassificationOutput, SR50_SystemCommunication, SR56_ServerIP, SR57_ServerPort

Variables	`- max_diseases: int`
	`- diseaseIDArray: int[]`
	`- diseaseNameArray: String[]`
	`- diseaseDescriptionArray: String[]`
	`- planIDArray: int[]`
	`- planDescriptionArray: String[]`
	`- successPercentageArray: double[]`
	`- result: int`
	`- symptoms: String`

Functions	`<<constructor>> DiseaseReceiverThread(String symptoms, int max_diseases)`
	`+ getDiseaseIDArray(): int[]`
	`+ getDiseaseNameArray(): String[]`
	`+ getDiseaseDescriptionArray(): String[]`
	`+ getPlanIDArray(): int[]`
	`+ getPlanDescriptionArray: String[]`
	`+ getSuccessPercentageArray: double[]`
	`+ getResult(): int`
	`+ run: void`

Functionality	The DiseaseReceiverThread class is used from the DiseaseReceiverActivity class to communicate with the server and send the symptoms the patient has to get a set of diseases that match with the symptoms introduced. The size of the diseases set has been previously stored in the "SP_Settings" shared preferences file, with the name "SP_Settings_Max_Diseases".

Table 114. C23_DiseaseReceiverActivity

Component 24: DiseaseSaverActivity

The Table 115 describes the application class DiseaseSaverActivity.

C24_DiseaseSaverActivity	
Dependencies	Diagnostic, DiagnosticDBHelper, Disease, DiseaseDBHelper, DiseaseSaverThread, MenuActivity, Plan, PlanDBHelper
Purpose	SR37_ApplicationDiseaseViewer, SR38_ApplicationDiagnosticSaver, SR48_ApplicationInternet, SR51_MinimumSDKVersion

Variables	`- K_SP_USER: String`
	`- K_SP_USER_EMAIL: Strng`
	`- K_SP_USER_PASSWORD: String`
	`- buttonBack: Button`
	`- buttonDisease: Button`
	`- buttonPlan: Button`
	`- buttonSave: Button`
	`- disease: Disease`
	`- contentWindow: LinearLayout`
	`- plan: Plan`
	`- symptoms: String`

Functions	`+ onCreate(Bundle savedInstance): void`
	`- getExtras: void`
	`- onCLickListenerFunction_ButtonDisease(): void`
	`- onClickListenerFunction_ButtonPlan(): void`
	`- onClickListenerFunction_ButtonSave(): void`

Functionality	The DiseaseSaverActivity class provides an user interface to show to the patient the complete diagnostic before being saved inside the account of the patient.

Table 115. C24_DiseaseSaverActivity

Component 25: DiseaseSaverThread

The Table 116 describes the application class DiseaseSaverThread.

C25_DiseaseSaverThread	
Dependencies	Diagnostic, Disease, Plan
Purpose	SR05_RequestCommandNumber, SR37_ApplicationDiseaseViewer, SR38_ApplicationDiagnosticSaver, SR49_PrivateDataRetrievalProtection, SR50_SystemCommunication, SR56_ServerIP, SR57_ServerPort

Variables	`- diagnostic: Diagnostic`
	`- disease: Disease`
	`- result: int`
	`- plan: Plan`
	`- email: String`
	`- password: String`
	`- symptoms: String`

Functions	`<<constructor>> DiseaseSaverThread(Disease disease, Plan plan, String email, String password, String symptoms)`
	`+ getResult(): int`
	`+ getDiagnostic(): Diagnostic`
	`+ run: void`

Functionality	The DiseaseSaverThread class is used from the DiseaseSaverActivity class to communicate with the server and send the diagnostic that the patient has chosen for the server to store. Once the diagnostic has been sent, the server sends the smartphone back the diagnose to confirm that the diagnostic has been correctly saved.

Table 116. C25_DiseaseSaverThread

Component 26: EditProfileActivity

The Table 117 describes the application class EditProfileActivity.

C26_EditProfileActivity	
Dependencies	Patient, EditProfileApplyChangesThread, EditProfileDeletThread, WelcomeScreenActivity
Purpose	SR23_ApplicationNoUserRedirection, SR42_ApplicationUserModificationConfirmation, SR43_ApplicationUserDeletionConfirmation, SR48_ApplicationInternet, SR51_MinimumSDKVersion

Variables	`- K_DATABASE_NAME_DIAGNOSTIC: String`
	`- K_DATABASE_NAME_DISEASE: String`
	`- K_DATABASE_NAME_PLAN: String`
	`- K_SP_USER: String`
	`- K_SP_USER_BIRTHDATE: String`
	`- K_SP_USER_EMAIL: String`
	`- K_SP_USER_PASSWORD: String`
	`- K_SP_USER_SURNAME: String`
	`- buttonApplyChanges: Button`
	`- buttonCancel: Button`
	`- buttonDelete: Button`
	`- calendar: Calendar`
	`- date: DatePickerDialog.OnDateSetListener`
	`- editTextBirthDate: EditText`
	`- editTextEmail: EditText`
	`- editTextName: EditText`
	`- editTextPassword: EditText`
	`- editTextSurname: EditText`
	`- userEmail: String`
	`- userPassword: String`

Functions	`+ onCreate(Bundle savedInstance): void`
	`- initializeTextViews(): void`
	`- onCLickListenerFunction_ButtonApplyChanges(): void`
	`- onClickListenerFunction_ButtonDelete(): void`
	`- updateLabel(): void`
	`- isNetworkAvailable(): boolean`

Functionality	The EditProfileActivity class provides an user interface to let the patient modify or delete its account through the EditProfileApplyChangesThread and EditProfileDeleteThread classes.

Table 117. C26_EditProfileActivity

Component 27: EditProfileApplyChangesThread

The Table 118 describes the application class EditProfileApplyChangesThread.

C27_EditProfileApplyChangesThread	
Dependencies	Patient
Purpose	SR05_RequestCommandNumber, SR42_ApplicationUserModificationConfirmation, SR49_PrivateDataRetrievalProtection, SR50_SystemCommunication, SR56_ServerIP, SR57_ServerPort

Variables	`- result: int`
	`- patient: Patient`
	`- email: String`
	`- password: String`

Functions	`<<constructor>> EditProfileApplyChangesThread(String email, String password, Patient patient)`
	`+ getResult(): int`
	`+ run: void`

Functionality	The EditProfileApplyChangesThread class is used from the EditProfileActivity class to communicate with the server and send the new data from the user account. The server retrieves the number 0 if the data is correctly stored, -1 if there is an unexpected error, -2 if the email and/or password is incorrect, and -3 if any of the fields is left in blank.

Table 118. C27_EditProfileApplyChangesThread

Component 28: EditProfileDeleteThread

The Table 119 describes the application class EditProfileDeleteThread.

C28_EditProfileDeleteThread	
Dependencies	- none -
Purpose	SR05_RequestCommandNumber, SR43_ApplicationUserDeletionConfirmation, SR49_PrivateDataRetrievalProtection, SR50_SystemCommunication, SR56_ServerIP, SR57_ServerPort

Variables	`- result: int`
	`- email: String`
	`- password: String`

Functions	`<<constructor>> EditProfileDeleteThread(String email, String password)`
	`+ getResult(): int`
	`+ run: void`

Functionality	The EditProfileDeleteThread class is used from the EditProfileActivity class to communicate with the server and send a petition to delete an account from a patient. The server retrieves the number 0 if the patient is correctly deleted and -1 if there is an unexpected error.

Table 119. C28EditProfileDeleteThread

Component 29: MenuActivity

The Table 120 describes the application class MenuActivity.

C29_MenuActivity	
Dependencies	Diagnostic, DiagnosticDBHelper, DiagnosticViewerActivity, Disease, DiseaseDBHelper, DiseaseReceiverActivity, MenuThread, Plan, PlanDBHelper, ProfileActivity, SettingsActivity
Purpose	SR29_ApplicationMissingActiveDiagnosticsRetrieval, SR30_ApplicationMissingActiveDiagnosticsStorage, SR48_ApplicationInternet, SR51_MinimumSDKVersion

Variables	`- K_SP_USER: String`
	`- K_SP_USER_EMAIL: String`
	`- K_SP_USER_PASSWORD: String`
	`- buttonArrayList: ArrayList<Button>`
	`- diagnosticArrayList: ArrayList<Diagnostic>`
	`- buttonAux: Button`
	`- buttonNewInjury: Button`
	`- ButtonProfile: Button`
	`- ButtonSettings: Button`
	`- layoutDiagnostics: LinearLayout`

Functions	`+ onCreate(Bundle savedInstance): void`
	`+ onStart(): void`
	`- getMissingDiagnostics(): void`
	`- onCLickListenerFunction_ButtonNewInjury(): void`
	`- onClickListenerFunction_ButtonProfile(): void`
	`- onClickListenerFunction_ButtonSettings(): void`
	`- updateDiagnostics(): void`
	`- isNetworkAvailable(): boolean`

Functionality	The MenuActivity class provides an user interface to let the patient choose to introduce a new disease or injury to be diagnosed, to view and modify or delete its account, to change the settings, and to see its diagnostics.

Table 120. C29_MenuActivity

Component 30: MenuThread

The Table 121 describes the application class MenuThread.

C30_MenuThread	
Dependencies	Diagnostic, Disease, Plan
Purpose	SR05_RequestCommandNumber, SR29_ApplicationMissingActiveDiagnosticsRetrieval, SR30_ApplicationMissingActiveDiagnosticsStorage, SR49_PrivateDataRetrievalProtection, SR50_SystemCommunication, SR56_ServerIP, SR57_ServerPort

Variables	`- diagnosticArrayList: ArrayList<Diagnotic>`
	`- diseaseArrayList: ArrayList<Disease>`
	`- planArrayList: ArrayList<Plan>`
	`- result: int`
	`- diagnosticIDArray: int[]`
	`- email: String`
	`- password: String`

Functions	`<<constructor>> MenuThread(String email, String password, int[] diagnosticIDArray)`
	`+ getDiagnosticArrayList(): ArrayList<Diagnostic>`
	`+ getDiseaseArrayList(): ArrayList<Disease>`
	`+ getPlanArrayList(): ArrayList<Plan>`
	`+ getResult(): int`
	`+ run: void`

Functionality	The MenuThread class is used from the MenuActivity class to communicate with the server and retrieve all the diagnostics that are not currently stored inside the smartphone.

Table 121. C30_MenuThread

Component 31: Patient

The Table 122 describes the application class Patient.

C31_Patient	
Dependencies	- none -
Purpose	SR01_UserAccount

Variables	`- patient_id: int`
	`- birthdate: String`
	`- email: String`
	`- password: String`
	`- surname: String`

Functions	`<<constructor>> Patient()`
	`+ getID(): int`
	`+ getBirthDate(): String`
	`+ getEmail(): String`
	`+ getName(): String`
	`+ getPassword(): String`
	`+ getSurname(): String`
	`+ setID(int patient_id): void`
	`+ setBirthDate(String birthdate): void`
	`+ setEmail(String email): void`
	`+ setPassword(String password): void`
	`+ setSurname(String surname): void`

Functionality	The Patient class is associated to the “patient” table inside the server database. This table stores the id of the patient, its email and password, its name and surname, and its birthdate.

Table 122. C31_Patient

Component 32: Plan

The Table 123 describes the application class Plan.

C32_Plan	
Dependencies	- none -
Purpose	SR02_RecoveryPlan

Variables	`- plan_id: int`
	`- description: String`

Functions	`<<constructor>> Plan()`
	`+ getID(): int`
	`+ getDescription(): String`
	`+ setID(int plan_id): void`
	`+ setDescription(String description): void`

Functionality	The Plan class is associated to the "plan" table inside the server database. This table stores the id of the plan and its description.

Table 123. C32_Plan

Component 33: PlanDBHelper

The Table 124 describes the application class PlanDBHelper.

C33_PlanDBHelper	
Dependencies	Plan
Purpose	SR02_RecoveryPlan, SR51_MinimumSDKVersion

Variables	`- K_DATABASE_NAME: String`
	`- K_DATABASE_VERSION: int`
	`- K_PLAN_TABLE_NAME: String`
	`- K_PLAN_COLUMN_ID: String`
	`- K_PLAN_COLUMN_DESCRIPTION: String`

Functions	`<<constructor>> PlanDBHelper(Context context)`
	`+ onCreate(SQLiteDatabase db): void`
	`+ onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion): void`
	`+ onDowngrade(SQLiteDatabase db, int oldVersion, int newVersion): void diagnostic_id): void`
	`+ deletePlan(int plan_id): void`
	`+ getPlan(int plan_id): Plan`
	`+ getPlans(): ArrayList<Plan>`
	`+ insertPlan(Plan plan): void`

Functionality	The PlanDBHelper class provides an interface to delete, insert and retrieve the recovery plans from the SQLite database inside the smartphone.

Table 124. C33_PlanDBHelper

Component 34: ProfileActivity

The Table 125 describes the application class ProfileActivity.

C34_ProfileActivity	
Dependencies	EditProfileActivity, WelcomeScreenActivity
Purpose	SR23_ApplicationNoUserRedirection, SR39_ApplicationProfileViewer, SR40_ApplicationUserLogOut, SR41_ApplicationProfileModificationPasswordRequest, SR51_MinimumSDKVersion

Variables	`- K_DATABASE_NAME_DIAGNOSTIC: String`
	`- K_DATABASE_NAME_DISEASE: String`
	`- K_DATABASE_NAME_PLAN: String`
	`- K_SP_USER: String`
	`- K_SP_USER_BIRTHDATE: String`
	`- K_SP_USER_EMAIL: String`
	`- K_SP_USER_NAME: String`
	`- K_SP_USER_PASSWORD: String`
	`- K_SP_USER_SURNAME: String`
	`- buttonBack: Button`
	`- buttonLogOut: Button`
	`- buttonModify: Button`
	`- sharedPreferencesPassword: String`
	`- userPassword: String`
	`- textViewBirthdate: TextView`
	`- textViewEmail: TextView`
	`- textViewName: TextView`
	`- textViewSurname: TextView`

Functions	`+ onCreate(Budle savedInstanceState): void`
	`+ onStart(): void`
	`- updateTextViews(): void`
	`- onClickListenerFunction_ButtonLogOut(): void`
	`- onClickListenerFunction_ButtonModify(): void`

Functionality	The PlanDBHelper class provides an interface to delete, insert and retrieve the recovery plans from the SQLite database inside the smartphone.

Table 125. C34_ProfileActivity

Component 35: SettingsActivity

The Table 126 describes the application class SettingsActivity.

C35_SettingsActivity	
Dependencies	- none -
Purpose	SR44_ApplicationMaximumDiseasesSetRetrieval, SR51_MinimumSDKVersion

Variables	`- K_SP_SETTINGS: String`
	`- K_SP_SETTINGS_MAX_DISEASES: String`
	`- buttonCancel: Button`
	`- buttonSave: Button`
	`- editTextMaxSettings`

Functions	`+ onCreate(Budle savedInstanceState): void`
	`- onClickListenerFunction_ButtonSave(): void`

Functionality	The SettingsActivity class provides an user interface to select the different parameters that the system uses. These parameters are stored in the “SP_Settings” shared preferences file. The parameter that can be modified inside the SettingsActivity class is the maximum number of diseases, that has the name “SP_Settings_MaxDiseases” inside the “SP_Settings” shared preferences file.

Table 126. C35_SettingsActivity

Component 36: UserCreationActivity

The Table 127 describes the application class UserCreationActivity.

C36_UserCreationActivity	
Dependencies	MenuActivity, UserCreationThread
Purpose	SR24_ApplicationUserRedirection, SR25_ApplicationUserCreation, SR26_ApplicationUserCreationStorage, SR48_ApplicationInternet, SR51_MinimumSDKVersion

Variables	`- K_SP_USER: String`
	`- K_SP_USER_BIRTHDATE: String`
	`- K_SP_USER_EMAIL: String`
	`- K_SP_USER_NAME: String`
	`- K_SP_USER_PASSWORD: String`
	`- K_SP_USER_SURNAME: String`
	`- buttonCancel: Button`
	`- buttonSignUp: Button`
	`- calendar: Calendar`
	`- date: DatePickerDialog.OnDateSetListener`
	`- editTextBirthDate: EditText`
	`- editTextEmail: EditText`

Functions	`+ onCreate(Budle savedInstanceState): void`
	`- onClickListenerFunction_ButtonSignUp(): void`
	`- updateLabel(): void`
	`- isNetworkAvailable(): boolean`

Functionality	The UserCreationActivity class provides an user interface to create a new patient account inside the server through the use of the UserCreationThread.

Table 127. C36_UserCreationActivity

Component 37: UserCreationThread

The Table 128 describes the application class UserCreationThread.

C37_UserCreationThread	
Dependencies	- none -
Purpose	SR05_RequestCommandNumber, SR25_ApplicationUserCreation, SR49_PrivateDataRetrievalProtection, SR50_SystemCommunication, SR56_ServerIP, SR57_ServerPort

Variables	`- result: byte`
	`- birthdate: String`
	`- email: String`
	`- name: String`
	`- password: String`
	`- surname: String`

Functions	`<<constructor>> UserCreationThread(String birthdate, String email, String name, String password, String surname)`
	`+ getResult(): byte`
	`+ run(): void`

Functionality	The UserCreationThread class is used from the UserCreationActivity class to communicate with the server and create the new user inside the server. The server retrieves the number 0 if the user has been correctly stored, -1 if there was an unexpected error, -2 if the user is already in the database, and -3 if there is any field left in blank.

Table 128. C37_UserCreationThread

Component 38: UserLoginActivity

The Table 129 describes the application class UserLoginActivity.

C38_UserLoginActivity	
Dependencies	MenuActivity, UserLoginThread
Purpose	SR24_ApplicationUserRedirection, SR27_ApplicationUserLogin, SR28_ApplicationUserLoginStorage, SR48_ApplicationInternet, SR51_MinimumSDKVersion

Variables	`- K_SP_USER: String`
	`- K_SP_USER_BIRTHDATE: String`
	`- K_SP_USER_EMAIL: String`
	`- K_SP_USER_NAME: String`
	`- K_SP_USER_PASSWORD: String`
	`- K_SP_USER_SURNAME: String`
	`- buttonCancel: Button`
	`- buttonLogin: Button`
	`- editTextEmail: EditText`
	`- editTextPassword: EditText`

Functions	`+ onCreate(Budle savedInstanceState): void`
	`- onClickListenerFunction_ButtonLogin(): void`
	`- isNetworkAvailable(): boolean`

Functionality	The UserLoginActivity class provides an user interface to retrieve a patient account already inside the server through the use of the UserLoginThread.

Table 129. C38_UserLoginActivity

Component 39: UserLoginThread

The Table 130 describes the application class UserLoginThread.

C39_UserLoginThread	
Dependencies	Patient
Purpose	SR05_RequestCommandNumber, SR27_ApplicationUserLogin, SR49_PrivateDataRetrievalProtection, SR50_SystemCommunication, SR56_ServerIP, SR57_ServerPort

Variables	`- result: byte`
	`- patient: Patient`
	`- password: String`
	`- surname: String`

Functions	`<<constructor>> UserLoginThread(String email, String password)`
	`+ getPatient(): Patient`
	`+ getResult(): byte`
	`+ run(): void`

Functionality	The UserLoginThread class is used from the UserLoginActivity class to communicate with the server to retrieve an user already inside the server. The server retrieves the number 0 when the user is correctly retreived, -1 if there was an unexpected error, -2 if the user is already in the database, and -3 if there is any field left in blank.

Table 130. C39_UserLoginThread

Component 40: WelcomeScreenActivity

The Table 131 describes the application class WelcomeScreenActivity.

C40_WelcomeScreenActivity	
Dependencies	MenuActivity, UserCreationActivity, UserLoginActivity
Purpose	SR22_ApplicationSettingsInitialization, SR24_ApplicationUserRedirection, SR51_MinimumSDKVersion

Variables	`- K_SP_SETTINGS: String`
	`- K_SP_SETTINGS_MAX_DISEASES: String`
	`- K_SP_USER: String`
	`- K_SP_USER_EMAIL: String`

Functions	`+ onCreate(Budle savedInstanceState): void`
Functionality	The WelcomeActivity class initializes the default parameters of the application and provides an user interface to go to the UserCreationActivity for the patient to create a new account, or to the UserLoginActivity for the patient to sign up with an already created account. If there is a user already signed up inside the application the WelcomeActivity calls to the MenuActivity class.

Table 131. C40_WelcomeScreenActivity

5.4.3.4. Traceability Matrix

The Table 91 has the traceability matrix that joins the system requirements and classes.

	Component																																							
SR	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40
01	X					X	X																								X									
02	X							X	X																							X	X							
03	X			X	X													X		X																				
04	X	X	X											X	X																									
05											X						X						X		X		X	X		X									X	
06										X																														
07											X																													
08											X																													
09											X																													
10											X																													
11												X																												
12													X																											
13											X	X																												
14											X																													
15											X																													
16											X																													
17											X																													
18											X																													
19											X																													
20											X																													
21											X																													
22																																								X
23																										X								X						
24																																				X		X		X
25																																				X	X			
26																																				X				
27																																						X	X	
28																																						X		
29																													X	X										
30																													X	X										
31																						X																		
32																						X																		
33																						X	X																	
34																						X	X																	
35																			X		X																			
36																			X		X																			
37																								X	X															
38																								X	X															
39																																		X						
40																																		X						
41																																		X						
42																										X	X													
43																										X		X												
44																																			X					
45																X																								
46																X																								
47																X	X																							
48																X						X		X		X			X							X		X		
49																	X								X		X	X		X							X		X	
50																	X						X		X		X	X		X							X		X	
51															X	X			X	X	X	X		X		X			X				X	X	X	X		X		X
52											X																													
53											X																													
54											X																													
55											X																													
56										X							X						X		X		X	X		X							X		X	
57										X							X						X		X		X	X		X							X		X	

Table 132. System Architecture Traceability Matrix

Chapter 6

Evaluation of the System

In this chapter the system implemented from the Chapter 5 is evaluated with help of personnel external to the project.

In order to give a look at the results of the system 7 people have participated in the evaluation of it. The age of the participants was between 19 and 59 years old, where 6 of the participants were men and 1 a woman, and none of them has any type of medical studies.

In order to evaluate the system each of the participants has been asked to simulate having three different types of diseases, a ankle sprain, a blister and a toe nail trauma.

Each of the diseases introduced in the database of the application has been taken from the book “Running Injuries, Treatment and Prevention” published by Jeff Galloway (Galloway, J., 2009). 5 different symptoms based on this book have been introduced for the classifier of the server to work.

The symptoms are introduced in the Android application through the interface shown in the Figure 37.

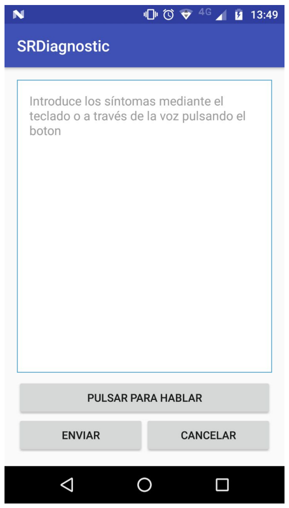
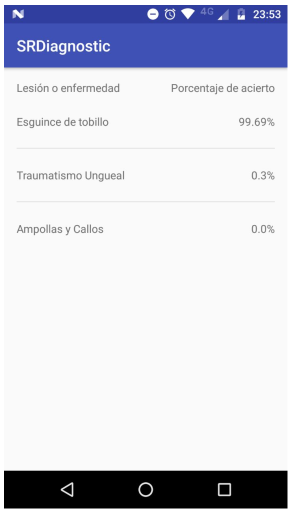

13:49
SRDiagnostic
Introduce los síntomas mediante el teclado o a través de la voz pulsando el boton
PULSAR PARA HABLAR
ENVIAR
CANCELAR

Figure 37. Android application symptom introduction interface

Each of the results is presented to the user of the application through the interface shown in the Figure 38, where the disease is shown in the left most part of the screen and the success percentage of the disease been the actual one on the left most part of the screen.



Lesión o enfermedad	Porcentaje de acierto
Esguince de tobillo	99.69%
Traumatismo Ungueal	0.3%
Ampollas y Callos	0.0%

Figure 38. Android application disease selector

Each of the participants has introduced the symptoms of an ankle sprain, of a blister and of a toe nail trauma separately.

The average success rates of the symptoms introduced by disease are the following.

Ankle Sprains Simulation

Disease	Success Percentage
Ankle Sprains	97.62%
Blisters and Calluses	2.33%
Toe Nail Traumas	0.03%

Table 133. Ankle sprains simulation average results

As shown in the Table 133, for the ankle sprains simulation, the classifier seems to provide good answers. The maximum error value provided has been 15.6% for blisters and calluses.

Blisters Simulation

Disease	Success Percentage
Ankle Sprains	3.75%
Blisters and Calluses	96.06%
Toe Nail Traumas	0.18%

Table 134. Blisters simulation average results

As shown in the table 134, for the blisters simulation, it seems that the classifier does provide the good answer due to the short and similar symptoms the participants have simulated. The maximum error value has been 26.22% for an ankle sprain.

Toe Nail Trauma Simulation

Disease	Success Percentage
Ankle Sprains	27.54%
Blisters and Calluses	17.62%
Toe Nail Traumas	54.80%

Table 135. Toe nail trauma simulation average results

As shown in the table 135, for the toe nail trauma simulation, even though the average success rate of the toe nail trauma is 54.80 percent, ankle sprains got a maximum value of 99.4% and blisters has got a maximum value of 97.91%.

Conclusion

Due to the lack of data stored in the database, the results seem to give results as 99.99% correct or 0.0% for some other cases. Even though, for the three cases represented (ankle sprains, blisters and toe nail traumas), the application seems to guess the disease with a percentage rate higher than

Chapter 7

Conclusions and Future work

In this final chapter a brief recapitulation of the project is discussed along with some final conclusions. Also, some future additions that have to be made to the system along with some improvements are left for future projects and research.

7.1. Conclusions

In the first chapter of the project has been the problems that are occurring nowadays, and a final objective for this project have been introduced.

To follow up, in the second chapter of the project, the main mobile operating systems that are currently available are seen along with medical applications that are nowadays popular among users of the different mobile operating systems platforms. Also, some of the most known speech recognition APIs used by developers have been shown.

In the third chapter of the project, the technologies used in the project have been described and it has been justified reason behind the choice of the technologies selected.

In the fourth chapter, the Spanish regulation is shown. In this chapter is described the different laws that restricts the design of the system along with reason behind the impossibility of releasing the final product of the project. Also, the software licenses of the technology used is described.

In the fifth chapter is where the solution to the problem described in the first chapter has been treated. This fifth chapter goes from the use cases of the system, to the design of it.

To finalize, an evaluation of the system is shown in the sixth chapter, where it is shown the accuracy of the system. Also, it is described the economic and social impact, where it is described the lack of impact due to the Spanish data protection law.

With the evaluation of the system has been shown that through text classification techniques the symptoms of the patient are classified correctly in most of the cases (with an average success rate higher than 50%). However, it has also been shown that the database has not currently as much data as it should to evaluate the symptoms of a user in a correct manner.

Even though, the text classification combined with speech recognition techniques is an area not yet exploited in medical applications and could be further developed.

7.2 Future Work

This system is not yet finished for release and can be used for future projects. Below are described some additions that have and could be made in order to release and improve the system.

- Add a security layer to the system to encrypt all the data related to the user in the database. Given the Spanish regulation, without the data of the user securely stored, the system is not yet capable of being released. However, the addition of the security layer is out of the scope of this project.
- The addition of JSON files to better describe the diseases and recovery plans for the user to select the disease most appropriate to its symptoms. Given that now only a string of text is describing the disease and recovery plans of the patient, the user is not as capable of selecting the right disease as it should. With the addition of JSON developer could introduce videos, images and other media to describe better, not only the disease, but the recovery plan.

- Improve the classifier of the system. Currently the classifier of the system is not yet as capable of classify symptoms as it should. It would be necessary to treat more the data of the database to build a better classifier.
- Introduce a module to keep track of the user data and, through observation of patterns, inform of the user more accurately of its condition.
- Introduce a module to keep track of all the users and, through observation of more general patterns, inform of the user more accurately of its condition (for example, if its winter, and everybody has a cold, it is more likely for the user to have a cold).
- Optimization of the application for different sizes of phones or tablets.

Also, out of the Android application, as the symptoms are being introduced directly by the user and not by responding a battery of questions, data analysis techniques could be applied to the database of the server where the diagnostics are stored to further study the needs of the population and to know where to invest manpower in research projects.

Annex A

Android Platform

A.1. Android Structure

The diagram shown in the figure shows the components of the Android operating system. Each of the components will be described below.

A.1.1. The Linux Kernel

The foundation of the Android operating system relies in the Linux kernel to provide Android with an operating system basic functionalities (Google Inc, n.d.d).

Linux, derived from the Unix operating system, is an open source kernel that includes all the features expected in a modern operating system as multitasking, virtual memory, shared libraries and much more functionalities (Linux Kernel Organization Inc, 2017).

A.1.2. Hardware Abstraction Layer

To communicate with the higher level Java API Framework the Hardware Abstraction Layer provides standard interfaces for the Java API Framework to access hardware components through the multiple library modules of the Hardware Abstraction Layer (Google Inc, n.d.d).

A.1.3. Android Runtime

The Android Runtime is system execution model that follows the running Android Applications. It is written to be executed in multiple virtual machines and aimed to run in run in low-memory devices. The Android Runtime reads and executes DEX files with DEX bytecode, files that were built to run in its predecessor, the Dalvik Virtual Machine (Google Inc, n.d.d).

A.1.4. Native C/C++ Libraries

Native libraries that are written in C and C++ are needed to tune some core components and services from the Android system. These libraries can be used from the Java API Framework already implemented in Android or through the Android Native Development Kit. Some of the libraries that can be found inside the Native C/C++ Libraries components are the renown libc library or the OpenGL library (Google Inc, n.d.d).

A.1.5. Java API Framework

Most of the Android Operating System is written in the Java language through the Java API Framework implemented by Google Inc. It provides the building blocks for the Android Applications found inside (Google Inc, n.d.d). Some of the most important components inside the Java API Framework are the following:

- **View System.** The Android View System provides the developer with modules for them to build the user interface of their application. The user interface is built through Views and ViewGroups, where each View is a drawn object inside the application and a ViewGroup an object holding Views in order to provide a layout (Google Inc, n.d.e).

- **Resource Manager.** The Resource Manager provides tools to the developer to externalize the strings, graphics and layouts from the code inside the application through XML files. Making the resources external to the application through the Resource Manager allows the developer to provide different resources for different types of devices in a simple manner (Google Inc, n.d.f).

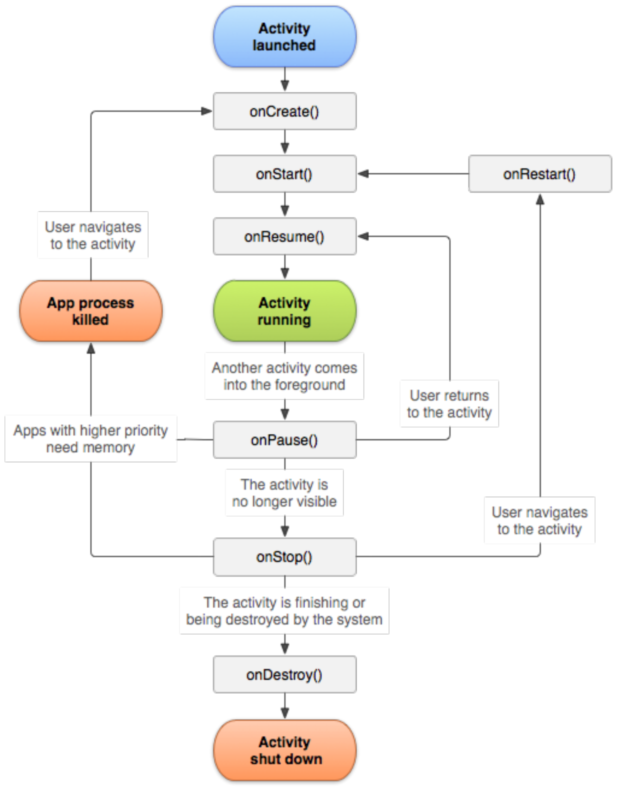
- **Notification Manager.** The Notification Manager lets the developer access to the alerts inside the status bar and from Android 8.0 (API level 26), the Notification Manager can also display notifications on application icons (Google Inc, n.d.g).
- **Content Provider.** The Content Provider lets the applications inside the Android platform access data from the application itself or from other applications, and to exchange data between Android applications. All the data sent through a Content Provider is encapsulated in order to provide the security mechanisms executed in the Android operating system (Google Inc, n.d.h).
- **Activity Manager.** The Activity Manager holds the activity stack of an application along with the lifecycle of the application. For more information about Android activities go to section A.2.1.

A.2. Components of an Android Application

A.2.1. Activity

An activity is the fundamental building block of the Android applications. An activity is the process that holds the user interface for the user to interact and process information (Google Inc, n.d.i).

An activity in Android goes through the different states, shown in the Figure 39, as the user navigates through. In total there are 6 different states, onCreate(), onStart(), onResume(), onPause(), onStop() and onDestroy().


Activity launched
onCreate()
onStart()
onRestart()
User navigates to the activity
onResume()
App process killed
Activity running
Another activity comes into the foreground
User returns to the activity
Apps with higher priority need memory
onPause()
The activity is no longer visible
User navigates to the activity
onStop()
The activity is finishing or being destroyed by the system
onDestroy()
Activity shut down
Figure 39. Android activity life cycle

Those states go from when the user initializes or creates the activity, making the activity go to the activity stack (to memory), to the moment when the user abandons or destroy the activity, making the activity be removed from the activity stack (from memory). Passing through different states depending if the user goes to another different activity or just abandon the activity momentarily (Google Inc, n.d.j).

A.2.2. Broadcast Receiver

The broadcast receiver enables the system to deliver events outside the application developed and allows it to respond signals from the system. This component does not have an interface, however, through the arrival of a signal a notification can be displayed (Google Inc, n.d.i).

A.2.3. Intent

An intent lets the Android system activate Android components asynchronously. However, for activities and services, the intent defines the action to perform and specify the data for a component to be started, meanwhile for broadcast receivers the intent only defines the announcement that will be broadcast (Google Inc, n.d.i).

A.2.4. Manifest file

The android manifest file is an XML file that holds the components inside an application, identifies the permissions the application requires, the minimum API level the application needs to run, the hardware and software used by the application, and aplication libraries that do not form part of the default Android framework (Google Inc, n.d.i).

A.2.5. Service

An Android Service is a long-running process that is executed in the background as a daemon and does not interact with the user. (Google Inc, n.d.i).

A.3. SDK

The Software Development kit is the environment developed by Google to provide the developers with libraries, tools and documentation to simplify the process of implementing an Android application.

Glossary

- **Algorithm.** Set of finite and predefined instructions that allows a system or an user to solve a problem.
- **API (Application Programming Interface).** An application programming interface is the name of the interface to a function or procedure that a developer uses from a software already made.
- **Application.**
- **BSD (Berkeley Software Distribution).** The BSD is a permissive license that lets the user modify a program or include parts of it in another system without having to make the source code open source. However, the developer of the original system needs to be mentioned in the source code and/or binary.
- **DB (Database).** A database is the part of a system that is in charge of storing information.
- **DEX (Dalvik Executable).** A Dalvik executable is the application of Android already prepared to run in a Dalvik virtual machine, the virtual machine where most Android applications run.
- **FOSS (Free and Open Source Software).** FOSS is any piece of software that is distributed in a free and open source way in order to be used, copied or studied.
- **GNU (GNU is Not Unix).** GNU is an operating that falls under the category of free software (being not permissive). It aims to make all software open sourced.
- **GPL (General Public License).** Software license that aims to make all code open source by not being permissive. GPL licensed software cannot be integrated in another software without making the software adopt a GPL compatible license or by using an exception of the license.
- **IDE (Integrated Development Environment).** User programs that provides the developer of programs and applications with tools for faster implementation of them.

- **IP (Internet Protocol).** The Internet protocol is used to manage the information send from one device to another in the form of packets through the internet.
- **JDBC (Java Database Connectivity).** Interface that the Java library provides to the developer to ease the access to the databases of the companies.
- **JDK (Java Development Kit).** The Java development kit provides the developer with the tools and libraries necessary to build Java programs.
- **JSON (JavaScript Object Notation).** It is a standard file format aim to the transmit data in the form of objects.
- **Kernel.** The Kernel of an Operating System is the low level software in charge of communicating the hardware components of the operating system with the applications the user can use.
- **Library.** In this project, a library is a collection of software with specified interfaces that perform a determined function.
- **Machine Learning.** Computer Science field aimed to develop algorithms able to find patterns in a provide dataset.
- **MariaDB.** Free and open source database that was born from the MySQL database to remain as free and open source forever.
- **MySQL.** One of the most popular databases in the world aiming to work with large amounts of data.
- **OS (Operating System).** Is the piece of software that provides an interface between user applications and hardware components of a device.
- **SDK (Software Development Kit).** A software development kit is a piece of libraries and compilers aimed to build a program. In this document the SDK is always referred to be the Android SDK, aimed to build Android applications.
- **Smartphone.** A smartphone is a device with an operating system that has a touchscreen as an input, that lets the user navigate through the Internet, store and retrieve files and make phone calls among other functionalities.
- **SQLite.** Public domain database aimed to run in devices with low power usage and to replace the system functions to access files.

- **Speech Recognition.** Methodology and technologies that allows to the user to translate the voice into text.
- **SR (Software Requirement).** Element in the software engineering process aimed to specify the User Requirements.
- **UC (Use Case).** Element in the software engineering process that defines the user interactions with the system.
- **UR (User Requirement).** Element in the software engineering process aimed to present the system components and interactions in an abstract form.

Bibliography

Ahuja, N., Ozdalga, A. and Ozdalga, E. (2012): "The Smartphone in Medicine: A Review of Current and Potential Use Among Physicians and Students", *Journal of Medical Internet Research*, Sep-Oct, vol. 14, issue 5, e128.

Apple Inc (2017a): "iOS 11, A giant step for iPhone. A monumental leap for iPad.", available at https://www.apple.com/ios/ios-11/, last accessed in September 2017.

Apple Inc (2017b): "App Store", available at https://developer.apple.com/support/app-store/, last accessed in September 2017.

Carey, B. (2016): "Smartphone speech recognition is faster and more accurate than typing", *Standford Engineering*, August 25, available at https://engineering.stanford.edu/news/smartphone-speech-recognition-faster-and-more-accurate-typing, last accessed in August 2017.

CMUSphinx (n.d.): "About CMUSphinx", available at https://cmusphinx.github.io/wiki/about/, last accessed in September 2017.

Debian (2017): "About Debian", available at https://www.debian.org/intro/about, last accessed in August 2017.

Google Inc (2017): "Dashboards", available at https://developer.android.com/about/dashboards/index.html, last accessed in September 2017.

Google Inc (n.d.a): "Cloud Speech API", available at https://cloud.google.com/speech/, last accessed in September 2017.

Google Inc (n.d.b): "Meet Android Studio", available at https://developer.android.com/studio/intro/index.html, last accessed in August 2017.

Google Inc (n.d.c): "RecognizerIntent", available at https://developer.android.com/reference/android/speech/RecognizerIntent.html, last accessed in August 2017.

Google Inc (n.d.d): "Platform Architecture", available at https://developer.android.com/guide/platform/index.html, last accessed in August 2017.

Google Inc (n.d.e): “UI Overview”, available at https://developer.android.com/guide/topics/ui/overview.html, last accessed in August 2017.

Google Inc (n.d.f): “Resources Overview”, available at https://developer.android.com/guide/topics/resources/overview.html, last accessed in August 2017.

Google Inc (n.d.g): “Notifications”, available at https://developer.android.com/guide/topics/ui/notifiers/notifications.html, last accessed in August 2017.

Google Inc (n.d.h): “Content Providers”, available at https://developer.android.com/guide/topics/providers/content-providers.html, last accessed in August 2017.

Google Inc (n.d.i): “Application Fundamentals”, available at https://developer.android.com/guide/components/fundamentals.html, last accessed in August 2017.

Google Inc (n.d.j): “The Activity Lifecycle”, available at https://developer.android.com/guide/components/activities/activity-lifecycle.html, last accessed in August 2017.

WebMD Health Corp. (2014): “What We Do For Our Users”, available at http://www.webmd.com/about-webmd-policies/about-what-we-do-for-our-users, last accessed in September 2017.

Linux Kernel Organization Inc (2017): “About Linux Kernel”, available at https://www.kernel.org/linux.html, last accessed in August 2017.

MariaDB Fundation (2017): “About MariaDB”, available at https://mariadb.org/about/, last accessed in August 2017.

Microsoft (n.d.): “Bing Speech API”, available at https://azure.microsoft.com/en-us/services/cognitive-services/speech/, last accessed in September 2017.

Oracle (2017): “Chapter 1 General Information”, available at https://dev.mysql.com/doc/refman/5.7/en/introduction.html, last accessed in August 2017.

Smith, B. et al. (2008): "Challenges of self-reported medical conditions and electronic medical records among members of a large military cohort", *BMC Medical Research Methodology*, BioMed Centre, 5th June 2008, vol. 8, issue 37.

SQLite (n.d.): “About SQLite”, available at https://www.sqlite.org/about.html, last accessed in August 2017.

Statista (2016): “Share of Windows Phone operating system versions worldwide in March 2016*”, available at https://www.statista.com/statistics/544313/windows-phone-os-versions-worldwide/, last accessed in September 2017.

Statista (2017a): “Number of smartphone users worldwide from 2014 to 2020 (in billions)”, available at https://www.statista.com/statistics/330695/number-of-smartphone-users-worldwide/, last accessed in August 2017.

Statista (2017b): “Smartphone user penetration as percentage of total global population from 2014 to 2020”, available at https://www.statista.com/statistics/203734/global-smartphone-penetration-per-capita-since-2005/, last accessed in August 2017.

Vim the editor (n.d.): “Vim - the ubiquitous text editor”, available at https://vim.sourceforge.io/, last accessed in August 2017.

Vincent, J (2017): “99.6 percent of new smartphones run Android or iOS”, *TheVerge.com*, February 16, available at https://www.theverge.com/2017/2/16/14634656/android-ios-market-share-blackberry-2016, last accessed in September 2017.

Waikato University (n.d.a): “Weka 3: Data Mining Software in Java”, available at http://www.cs.waikato.ac.nz/~ml/weka/, last accessed in August 2017.

Waikato University (n.d.b): “Machine Learning Group”, available at http://www.cs.waikato.ac.nz/~ml/index.html, last accessed in August 2017.