FCTUC

Information and Coding Technologies for Medical Applications

André Filipe Gonçalves Fonseca

Submitted dissertation to obtain a Master's Degree
in Electrical and Computer Engineering

Supervisor: Rui Pedro Duarte Cortesão, Ph.D

Jury:

Hélder de Jesus Araújo, Ph.D
Jaime Baptista dos Santos, Ph.D
Rui Pedro Duarte Cortesão, Ph.D

Agradecimentos

Cada percurso é feito de altos e baixos, isso é uma certeza que praticamente todos temos. Mas esta noção ganhou ainda mais força ao longo de todo trajeto que me trouxe a este momento. Em introspeção é impossível não lembrar os melhores e piores momentos mas no meio deles todos há algumas constantes, que merecem destaque e os meus maiores agradecimentos.

Começar pelo meu supervisor, Professor Doutor Rui Duarte Cortesão, que me proporcionou um tema que não só era desafiante para mim, como também me dava um "motivo" para trabalhar, uma vez que se aprovado, poderá ser um passo em frente, por mais pequeno que seja, nas melhorias das condições e acesso a saúde neste país.

Também agradecer ao Doutor Ruí Garcia da Medicina Interna dos Hospitais da Universidade de Coimbra, pois foi através do seu feedback que consegui definir metas e um rumo a tomar no desenvolvimento deste projeto.

Um agradecimento especial ao Pepe e à Laura pela ajuda com a escrita desta dissertação, pois com as suas sugestões foi me possível chegar a esta versão final do documento.

Agradecer aos meus amigos e colegas de faculdade, por se terem disponibilizado a testar e dar feedback que se revelou bastante importante para limar arestas e ultimar detalhes deste projeto, e acima de tudo por terem feito destes últimos cinco anos o que eles foram, cheio de momentos que guardarei para a vida bem como toda aprendizagem pessoal associada.

Agradecer ao meu pai por todo o apoio e motivação que me deu, mesmo a quilómetros de distância, permitindo-me manter positivo e empenhado quanto tudo parecia estar a dar errado.

Finalmente, e não menos importante, agradecer à minha mãe, o meu rochedo, que esteve sempre cá e que fez todos os possíveis e impossíveis para eu chegar a este momento sem que me faltasse coisa alguma, além de me ajudar a tornar uma pessoa melhor a cada dia que passa.

Uma última nota, não de agradecimento, mas que acho por bem estar nesta secção, quero dedicar todo este trabalho às minhas avós, pois de uma forma ou de outra, influenciaram-me a escolher e moldar este projeto que espero vir a ser uma ajuda a outras avós, pais e familiares.

A todos, muito obrigado!
André Filipe Gonçalves Fonseca

Abstract

Throughout history, medicine has always been considered as an essential pillar in the development of human society, increasing its longevity through all the work developed in the prevention, follow up and cure of various diseases. For this to happen, a constant work was required by, not only medical entities but also several other scientific areas, that have constantly shown positive results.

A practical example is based on the emergence and evolution of areas such as e-health, electronic medicine, and m-health, mobile medicine, that derives from the former, and has been shown a large-scale growth abroad, largely reflecting the increase in worldwide use of mobile equipment such as smartphones[1].

Following this perspective, and wanting to continue the world trend, this dissertation intends to present the process associated with the development of a medical platform, consisting of a website and an Android application that allow patients, to send information regarding medical measurements such as cardiac function, chronic pain, body temperature and glucose levels, from their smartphones, to a remote database that can be consulted, through the platform's website, by medical staff, thus aiming to increase the quality in medical follow-up.

Therefore, throughout the chapters of this dissertation, a presentation and discussion of the various phases related to the development process of the platform will be done, like the requirements capture, conception, design and implementation of the platform, some functional tests, and finally a set of conclusions about the current state of the platform in terms of readiness and usability as well possible future implementations.

Keywords:

e-Health, m-Health, Medical Platform, Android Development, Web Development.

Resumo

Ao longo da história, a medicina sempre foi considerada como um pilar essencial no desenvolvimento da sociedade humana, aumentando a sua longevidade através de todo o trabalho desenvolvido na prevenção, acompanhamento e cura de doenças dos mais variados tipos. Para tal acontecer, foi necessário um trabalho constante por parte de várias entidades não só médicas como ligadas a outras áreas científicas, apresentando constantemente resultados positivos.

Um exemplo prático baseia-se no aparecimento e evolução de áreas como a e-health, medicina eletrónica, e m-health, medicina móvel, derivante da primeira e que tem apresentado um crescimento em larga escala no estrangeiro, muito em reflexo do aumento da utilização mundial de equipamentos móveis como por exemplo os smartphones [1].

Seguindo essa ótica, e querendo acompanhar a tendência mundial, esta dissertação de mestrado pretende apresentar o processo de desenvolvimento duma plataforma médica, composta por um site e uma applicação Android que permitam aos pacientes, a partir de qualquer lugar, enviar informação referente a medições médicas -função cardíaca, dor crónica, temperatura corporal e glicémia- para uma base de dados remota que pode ser consultada, através do site da plataforma, pelo pessoal médico associado, pretendendo assim aumentar a qualidade do acompanhamento.

Ao longo de vários capítulos serão apresentadas e discutidas as várias fases relacionadas com o processo de desenvolvimento da plataforma como a captura de requisitos, concepção, design e implementação da plataforma e alguns testes de funcionamento, sendo no final tiradas conclusões sobre o estado atual da plataforma em termos de usabilidade atual e possíveis implementações futuras.

Palavras-Chave:

e-Health, m-Health, Plataforma Médica, Desenvolvimento Android, Desenvolvimento Web.

List of Figures

1 Project Stages Diagram 1
2 Existing Solutions Abroad - Interface Snippets 7
3 Gathering Requirements Process 9
4 Diagram of Implemented Solution 13
5 High-Level Design of the Android Application 14
6 High-Level Design of the Web-Application 15
7 Simplified Package Diagram of the Web-Application 16
8 Database Design Model 17
9 Web Interface Design: Login Page 22
10 Web Interface Design: Home Page 24
11 Web Interface Design: Home Page with Header Usage Example 25
12 Web Interface Design: Adding, Editing and Removing Users Pages 25
13 Web Interface Design: Patient's Health Records Page 26
14 Web Interface Design: Patient's Measure Analysis Page 27
15 Android Interface Design Layouts 29
16 Android Application Class Diagram 54

List of Tables

1 Table of Requirements Gathered During the Elicitation Stage 10
2 Table of Final Defined Requirements to be Implemented 12
3 Table with Known Issues and Possible Fixes 44
4 Table of Requirements Met with Justification 47

Contents

List of Figures

List of Tables

1 Introduction **1**
1.1 Objectives . . . 1
1.2 Main Contributions . . . 2
1.3 Dissertation Outline . . . 2

2 Background **3**
2.1 Introduction . . . 3
2.2 Literature Review . . . 3
2.2.1 m-Health . . . 3
2.2.2 Android . . . 4
2.2.3 Web . . . 5
2.2.4 Databases . . . 6
2.3 Existing Solutions . . . 6
2.3.1 Abroad . . . 6
2.3.2 In Portugal . . . 7
2.4 Summary . . . 8

3 Requirements **9**
3.1 Introduction . . . 9
3.2 Gathering Requirements . . . 9
3.2.1 Elicitation . . . 9
3.2.2 Analysis and Specification . . . 10

4 Project Design and Implementation **13**
4.1 Introduction . . . 13
4.1.1 Tools Required . . . 13
4.2 Modeling . . . 14
4.2.1 Android-Application . . . 14
4.2.2 Web-Application . . . 15
4.3 Database . . . 16
4.3.1 Design . . . 17
4.3.2 Implementation . . . 18
4.4 Interface Design . . . 21
4.4.1 Web Interface . . . 22
4.4.2 Android Interface . . . 28
4.5 Background Work . . . 29
4.5.1 Web . . . 29
4.5.2 Android . . . 38

5 Deployment and Testing **42**
5.1 Introduction . . . 42
5.2 Deploying Versions . . . 42
5.2.1 Web . . . 42
5.2.2 Android . . . 42
5.3 Testing . . . 42
5.3.1 Issues and Fixes . . . 43

6 Results and Evaluation **45**
6.1 Requirement Verification . . . 45
6.2 Evaluation . . . 47

7 Conclusion **48**
7.1 Work Review . . . 48
7.2 Future Work . . . 49

8 Bibliography **51**

9 Appendix **54**
9.1 A1 - Class Diagram . . . 54
9.2 A2 - CSS Code Snippet . . . 55

1 Introduction

Mobile Health has been, since its beginning, a very good example of the success that can come out of a fusion between medicine and engineering. The possibilities and growth expectations, since its conception, were immense because it was directly dependent of the evolution in mobile and wireless technologies.

For example, taking the case of computer and smartphone usage, data can be easily acquired regarding the numbers of smartphone usage[1] [2]. Taking into consideration our country, Portugal, in 2017, an average 70% of the general population use a computer with Internet connection and 68% use a smartphone, that by default, will have a wireless connection itself. Taking in consideration the definition of Mobile Health, this means that about 70% of the general population is, without considering any kind of restrictions, a viable candidate to be included into a Mobile Health platform.

That itself, shows a lot of promise, since it has been well established that Mobile Health brings a lot of benefits to medicine[3] [4], for example in follow-ups and self-management of chronic diseases[5] [6] [8]. Although this dissertation doesn't focus on the state of Mobile Health in the world, it's important to see that it has been evolving rapidly[9], becoming a well established and helpful technology[10] [11] that can be ported to Portugal's health-care system.

1.1 Objectives

Considering the state of Electronic and Mobile Health, the concerns about topics like data security, that will be approached along the next chapters of this dissertation, and all the knowledge available regarding the development of applications through coding, the main focus of this dissertation is to describe the process of thought behind the design and implementation of a three-tier medical platform, through its various stages that can be described by the scheme presented in figure 1:

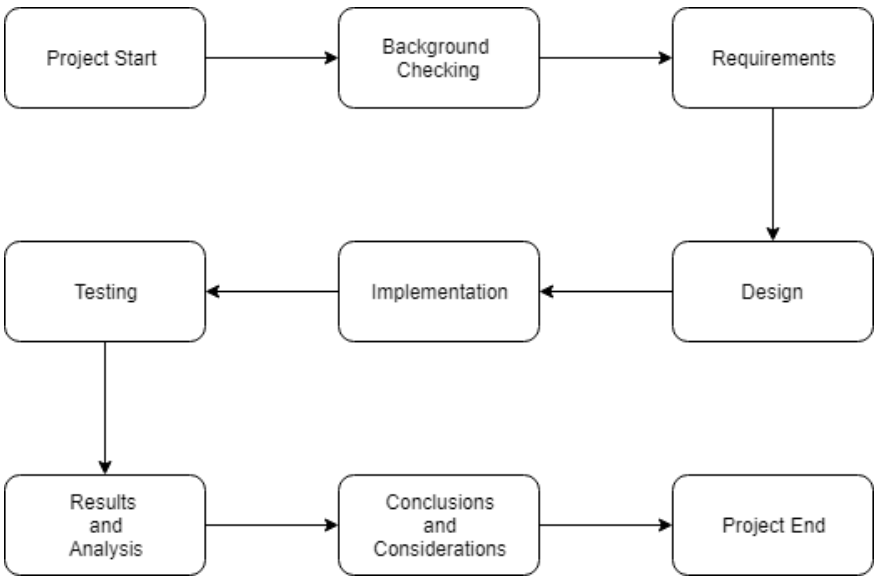


Figure 1: Project Stages Diagram

1.2 Main Contributions

As expected, software design and implementation isn't something new. Nowadays it's difficult not to find a software developed application implemented with success, some with a very high level of scientific complexity. With the various Integrated Development Environment softwares available, multiple algorithm and coding language libraries, documents and manuals that can be looked up online, one can adapt and shape the provided knowledge in order to design and implement software successfully. Therefore, it's a safe assumption to say that the project described in this dissertation has it's basis on various existing open source algorithms, that are well optimized and present good functionality results. The more important algorithms will be referenced, during chapter 4, and given due credit.

1.3 Dissertation Outline

This dissertation is divided in 7 chapters, including this introduction and excluding the bibliographic references and appendix. In the following chapter, the result of background research will be presented, regarding Mobile Health security and Android/Web/Database Development, platforms on which this three-tier project will be implemented. Chapters 3 to 5 will describe the software engineering part of this project, going through the requirements gathering stage, theoretical design and the practical implementation of the project that, in chapter 6, will have its results analyzed, verified and discussed. Finally, in chapter 7, conclusions about the project and future work suggestions will be presented.

2 Background

2.1 Introduction

Prior to the design and implementation of this software project, it's necessary to have a basic understanding about certain key aspects related to software developing, being it Android or Web development, and Mobile Health itself. In order to achieve that knowledge goal and avoid repeating work, since there is already a large number of Mobile Health applications in the world, a literature review of mainly academic papers took place and the results will be presented in the various sections and sub-sections of this chapter. Before starting it's considered a good policy to inform that out of a large number of papers available regarding the various topics discussed in this chapter, only a few were selected and will be referenced, since a great number of them talk about similar things.

2.2 Literature Review

2.2.1 m-Health

In the introductory chapter of this dissertation a reference was made to a paper [10], published in 2014 on the Online Journal of Health Informatics, made by Maged N. Kamel Boulo, Ann C. Brewer, Chante Karimkhani, David B. Buller and Robert P. Dellavalle, which serves as a great starting point to the topic of Mobile Health. In this paper, a review on the state of Mobile Health is made, explaining its various intervention areas such as applications for medical providers, education/teaching health and general public-oriented purposes. From this list, there are two main areas that are directly related to the focus of this project. Firstly, regarding medical providers it's explained that the existing applications are considered more complex because not only they use medical-oriented language but they are mostly seen as helpful tools used for drug-referencing, clinical decision-support and access to medical education materials. In terms of patient-oriented applications, there can be a division of the many existing applications in health and fitness. Again, for the context of this project the main focus goes towards health applications, in which there are many intervention areas like chronic disease or addiction aid, self-diagnosis and self-management.

In both areas, but mostly in the area of patient-oriented applications, there is a large usage, sharing and storage of electronic health-record. That poses a complex case study since these records are seen as sensitive data carriers, which, for the context of this project, is seen as a very important case study, since it intends on exploring and using this tool.

In fact, there are several papers that regard security aspects in medical health applications. For example, in their paper [12], Miloslava Plachkinova, Steven Andrés and Samir Chatterjee talk about the large number of mobile health applications that aren't regulated for example by the United States Food and Drug Administration, FDA, or even regulated at all since it's very easy to launch a smartphone application into the various existing stores, for example Google Play Store, or public websites as an unsigned application which are applications that can be seen as unsafe since it doesn't guarantee that the application hasn't been tampered with. That itself poses a risk to the privacy of their users, and also doesn't guarantee that the application has enough security measures or policies to withstand any kind of hacker attacks. That set of reasons made them propose a classification model for mobile health applications in terms of security and privacy in order to raise awareness towards the lack of

regulation and privacy concerns in those applications.

Another paper that talks reinforces these concerns regarding the security side of medical health applications[13]. Written by Rajindra Adhikari, Deborah Richards and Karen Scott, this paper followed a similar path of selecting a sample of medical health applications available and classified them in terms of security and privacy. For example, they evaluated if the applications had an authentication service, if it encrypted data, how it was stored and other aspects. From their study they found out that a portion of the applications weren't issue free, results that were also acknowledged by the previous paper[12].

Those security concerns caused a need to review papers that advise on how to handle privacy and security of patient health records, in order to design and implement a both usable and safe project. Starting of with a paper by Rajindra Adhikari, Deborah Richards and Karen Scott[14], a table, number 3, was created with a set of minimal and recommended requirements towards security and privacy that mobile health applications developers should consider. Aspects like access control, i.e., allow and/or block user access to certain parts of the application, authentication measures like having a unique identification number and password that only the user knows, data management and user notification channels that allow users to be updated about breaches, software updates and other kinds of informations.

Another paper[15] by Borja Martínez-Pérez, Isabel de la Torre-Díez and Miguel López-Coronado, describes various types of attacks that mobile applications created for Android's operative systems are subject to, being through the Internet, faulty login systems, third party software, bluetooth and other attack surfaces. From their study, a conclusion is drawn that protection and encryption is a key aspect regarding mobile health applications and that developers should consider creating or investing in a server for storage, in order to disallow breaches of information via third party softwares, despite increasing the cost and time of production.

As a side note, also regarding health records security, specifically in Portugal, there is a document[16] by the ministry of health, that exposes both critical and juridical aspects about privacy in the county's health care system. As expected, there are references to the same kind of key aspects described in the previous mentioned papers like confidentiality, integrity, availability of information and legal conformity regarding privacy.

Despite not having concrete coding tips about how to develop a mobile health application, the case studies about privacy and security revealed themselves as very useful background knowledge towards the developing stages of this project.

2.2.2 Android

Being this mobile health project, the developing of a smartphone application is a certainty. Therefore, a choice between the targeted operative system is required for this project. In order to decide correctly, a small but concrete research regarding the distribution of smartphone operative systems in the world should take place. Analyzing existing data[17], it's clear that Android is the leading operative system in the world with an approximate value of 75% in comparison with iOS's roughly 15%. That can be due to the open-source characteristics that Android has and the fact that it isn't exclusive to a brand like Apple's iOS is. Being open-source, means that the ability to create applications is immense, sentence that can be backed by data[18] from research2guidance web page, which states that Android is, as of 2017, the leading platform for Mobile Health applications with 375000 available. The combination of this data is itself a very strong argument that can help decide which operative system

to choose. Still, taking into account that this project is intended for Portugal's health care system, an analysis of smartphone operative systems distribution is also required. That data can be easily acquired [19] in the same web page where the worldwide distribution was gathered, statcounter. Taking this information into account, the choice of starting with the development of a mobile health application in Android seems like the right choice. Therefore, it's necessary to analyze the current state of Android developing. As it stands, there have already been created 9 distinct versions of Android's operative system. Each one of them bring updates on various aspects, being them interface-wise, functionality-wise, and security-wise, the latter being important for user safety. A small study [20] by SOPHOS, a security software company, states that in 2017 nearly 3.5 million existing applications contained malware of some sort, and a large portion was ransomware which is a malware that captures a system, which usually contains private user records, and demands money in exchange for the systems freedom.

As previously mentioned, Android is open-source, meaning that all its documentation and source code are released to public under a copyright license that allows anyone to study, change and distribute software developed with that same source code. Developing of Android applications can be done with different integrated development environments, required that the application developer uses Android's and JAVA's software development kits, JDK and SDK, that can be easily obtained online. As a quick mention, Android applications can also be developed through Kotlin, that runs on JAVA's virtual machine and has some similarities. This programming language is also open-source, but doesn't have comparable documentation available as JAVA.

In chapter 4 the methods and usage's of JAVA programming language to develop the Android part of this project will be further discussed.

2.2.3 Web

Regarding the second tier of the project, it was also required to make a choice. Either develop a exclusive desktop application, that can be installed on a computer like most programs, or create a online web page that doesn't require installation. The choice to create a web page was promptly taken into consideration because it allowed the existence of a application that was accessible to all kinds of operative systems, Windows, Linux, macOS and other less known ones, via web browser, which solves any kind of installation problems that could occur. Therefore, like on the Android case, a small review was necessary in order to have a basic knowledge of web developing.

A lot can be said about the web in itself but, for the purpose of this project, mainly coding aspects will be mentioned. For example, web page developing is, in its core, done by programing in Hypertext Markup Language, HTML, and Hypertext Preprocessor, PHP, coding languages. Starting with HTML, documentation can be found about both current [21] and previous versions. Currently, HTML is in version 5.2 and is compatible with the main web browsers available like Microsoft Edge, Google Chrome, Mozilla Firefox, Apple's Safari and Opera. As referenced that poses a great advantage in terms of usability in all operative systems. Programming in this language doesn't pose restrictions in compatibility with web browsers, which cannot be said about PHP. Currently on version 7.0, like HTML documentation isn't hard to find [22], but in terms of compatibility, precautions are needed since some functionalities of newer PHP versions aren't yet to be supported by all browsers and that can result in a faulty experience and therefore, a bad implementation of software.

Also, like in Android, security is a key aspect of web applications, especially if is taken into consideration the management and processing of sensible data like electronic health records. An interesting paper[23] by James Walden, Maureen Doyle, Robert Lenhof, and John Murray compares the security levels associated with the choice between JAVA and PHP for web applications. From the start this paper presents an interesting comparison between the two main languages that will be used in this project, despite the fact that, in the context of this paper, JAVA is intended for web application development, which it can. Nonetheless, in their paper they reach the conclusion that both languages have been improving significantly in terms of security and that both languages are head-to-head in that area, with no language having a clear advantage over the other one, which is important considering those languages will be used in this project.

More information regarding the coding aspects will be addressed on chapter 4 of this dissertation like the usage of other web oriented coding languages and its purpose towards this application.

2.2.4 Databases

A pivotal part of this project passes through data storage that can be accessed at any time by multiple users. With that in mind, a remote database is needed to implement. Taking advantage of the already known knowledge regarding MySQL which is an open source system that uses Structured Query Language, SQL, to manage database operations. Like in the previous sections, documentation regarding SQL language and MySQL system is available online.

In chapter 4 the design and implementation this database management language will be discussed and presented with more detail.

2.3 Existing Solutions

Also as part of a background study, it's important to know what applications are already developed and take note of both the functionalities that can be implemented and which ones cannot. As mentioned in section 2.2.2, regarding Android, there are more then 375000 applications which make it impossible to cover. Regarding computer oriented ones there isn't quantifiable information, especially if hospital and/or health facilities based software, that varies with the needs and services that exist in each of this facilities, are taken into consideration.

Therefore, a search was conducted for both abroad and country originated applications, mostly focused in Android developed one's, picking a set of them that were self-classified as most important and similar to the intentions of this project. That means some functionalities of these applications could be ported to the computer side of this platform.

2.3.1 Abroad

1. health2sync by H2 Inc, available for Android and iOS.

   health2sync is a diabetes-center mobile application that allows diabetes carriers to track blood sugar levels, helping them stay one step ahead of their condition. The application allows users to record blood sugar and pressure values as well their current date.

It also allows them to export data in to Excel or PDF format and/or email doctors the data, closing any existing gaps between patient and doctor.

2. SmartBP by evolvemedsys, available for Android and iOS.

   This application is intended to be a blood pressure management tool. That means it will allow users to record and keep track of blood pressure measures.

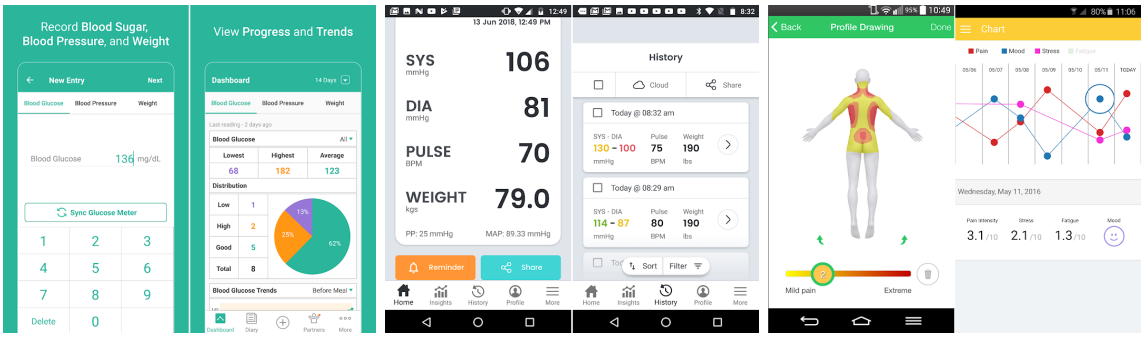
   Like health2sync it also allows users to export the data and therefore sent it to doctors easily, once again closing gaps in the patient-doctor relationship.

3. Pain Companion by Sanovation AG, available for Android and iOS.

   This application allows users to, amongst other uses, keep track of their chronic pain, expressing their intensity levels through a visual pain scale and view it in chart form that allows them to keep track of their pain levels.

   In this application there aren't mentions about exporting data abilities.

Image snippets of these three cases can seen in figure 2, that show interesting aspects of those applications that can be based as inspiration towards this project.



(a) 1 - health2sync (b) 2 - SmartBP (c) 3 - Pain Companion

Figure 2: Existing Solutions Abroad - Interface Snippets

2.3.2 In Portugal

In Portugal, mobile medical applications aren't yet firmly implanted in the health care system. The existing ones, in their majority, were oriented towards scheduling medical appointments or even having them online, accessing health prescriptions and information such as insurance cards, vaccine reports and other types of documents.

Amongst those applications like MySNS Carteira, MySNS, iMED, Vital Health and other applications one stood out in this research. Developed in a partnership composed by Porto's Medicine Faculty, FMUP, CINTESIS - Centro de Investigação em Tecnologias e Serviços de Saúde and Medida, CARAT [24], Control of Allergic Rhinitis and Asthma Test is a medical platform where a user can, through a brief self-administered questionnaire, quantify the degree of control in their Allergic Rhinitis and Asthma. Currently, using the software behind this platform there are three mobile health applications developed, InsperierMundi, Diário da Alergia MACVIA-ARIA and Lung Manager, exclusive to the dutch application store. Both the design and concept behind this platform are very interesting and was better taken into consideration in comparison to the other existing applications.

2.4 Summary

Throughout this chapter, the literature findings exposed in this dissertation, associated with a small but focused research around existing m-health applications, despite not being able to test them properly without creating a user account, basic knowledge and principles regarding security, potential uses/functionalities and other important aspects to this project were acquired, allowing the development stages to be well explained and, most importantly, avoid mistakes that can compromise it.

For example, the need to have authentication and data encryption components has been well established and considered critical. The need to allow the editing and elimination of all the user's data at any time was also considered important. These and other ideas will be discussed in chapter 4 with further detailing and coding knowledge.

3 Requirements

3.1 Introduction

The first step towards a software development process is to gather and set the requirements associated with the project to develop, being functional, non-functional or related to certain restraints regarding design, security, and or other quality aspects. Nowadays, the knowledge behind this process is well established and documented [25], so it's only logical to follow said knowledge. So, for the purpose of a good context towards the next chapters, a description of this process will be done in this chapter, following the scheme presented on figure 3, with some adaptations, since it couldn't be replicated in a full.

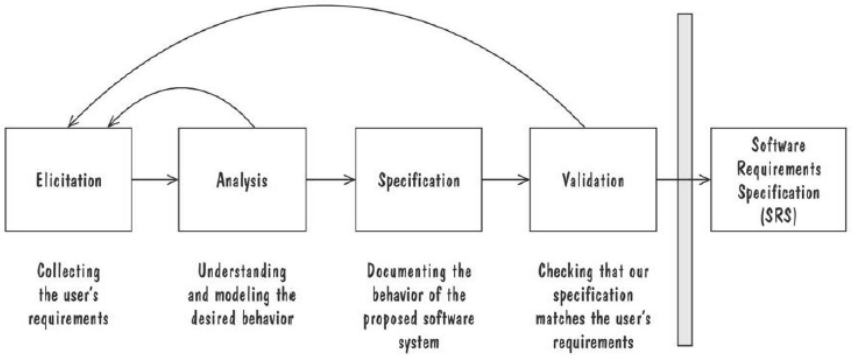


Figure 3: Gathering Requirements Process

3.2 Gathering Requirements

3.2.1 Elicitation

To start this process, it's necessary to have, at least, one person, formally design as a client, that asks for a set of functionalities in one or more meetings. Being this a medical centered platform-application, it makes all sense that the said client is a medical practitioner. For that matter, a gathering was appointed with a internal medicine doctor. From that a set of key requirements was appointed, and are shown now in table 1, without any separation in the multiple possible areas of requirements.

No.	Description
1	Have a window where doctors can only check for patient's basic profile information: - Name. - Age. - Birth Date. - Gender. - Contact Information (Mobile and E-mail). - ID and Health ID numbers.

2	Allow the existence of a manager to take care of the information related to requirement 1 being it adding patients, editing their information or removing it, alongside with the patient himself.
3	Have a window where a doctor can see a medical chart with information about: - Height, cm. - Weight, kg. - Medical History/Background. - Medication (Name, Dosage, Posology and Time Schedule). - Allergies. - Other Information (e.g Diet Guidelines, Blood Type, etc).
4	The manager, referenced in requirement 2 can only access the platform for this tasks being blocked from accessing the information handled by the doctor.
5	In reverse, a doctor can only handle the medical information and have access to it.
6	The platform must be cross-compatible, i.e, have a remote database that can be accessed from multiple and computers, at the same time if required. It can either be a program that is installed on a computer or a web platform.
7	The patients mustn't have access to this platform in any way. Instead, they will have a mobile app that will allow them to send measures related to hearth function, chronic pain, glucose and body temperature.
8	This mobile application is patient-exclusive, therefore it doesn't require any of the functionalities described in requirements 1 to 5. Regarding requirement 6, it's desired to allow the app to run on multiple Android versions, in order to reach a maximum number of patients.
9	Patients are allowed to sent measures that are contained in defined intervals of "what is normal for that kind of measures", preventing them from sending false data (e.g, negative or out-of-scale values).
10	The data sent by the patients to the database can be acessed by the doctor at any time, via platform, displayed in both table and line graph views, in a way that is understandable and easy to read.
11	The patient application could (not mandatory) send a daily reminder.
12	All the sensible data should be encrypted and a login system is required.

Table 1: Table of Requirements Gathered During the Elicitation Stage

3.2.2 Analysis and Specification

Upon review of the requirements presented, a process consisting in the division of said requirements into categories, functional, non-functional, design and process constraints begins. Also, as a some sort of extra, an extension of categories was set in order to prioritize the level of functionality importance for this software project. Those categories are represented by three options: critical, where the requirement must be implemented in order to guarantee a correct execution of software, important, where the requirement should be implemented in order to present itself has a proof of concept that is ready to be used in large scale and finally optional, where the requirement is considered as a pleasant feature to have, but that

doesn't compromise the software if it isn't implemented.

In the following table 2, a revised version of the requirements presented in table 1 will be displayed, with the incorporation of new requirements that were deemed as necessary upon a post and final analysis of the elicitation stage, describing and categorizing them in terms of type and importance.

Also, it is important to mention that this table is final in regards to what was the direction followed in the development of this project and was not validated by the client, given that it was granted full freedom, with the only clause being that the asked requirements were met and fulfilled. This meets, as referenced in the introduction of this chapter, the reference to a set of small alterations to the scheme presented, that can be defined specifically by the need to skip the validation stage.

Requirements	Category	Priority
The software must consist of a two-way platform. One side must be focused on the patients, being it a smartphone oriented application. The second one must be focused on the doctors (and managers), being it a desktop oriented application.	Design	Critical
The patient-oriented application will be set to work on Android smartphones, with compatibility to a maximum percentage of usable devices. From data acquired[26] a minimal requirement compatibility with Android API 19+ (i.e, Android 4.4+).	Design	Important
The patient oriented application, being an Android application, will be developed using JAVA language.	Design	Critical
The Android application will have a login system with based on an unique ID, given by the system and a user defined password, that will be encoded and can only be validated using encoding/decoding by the system.	Non Functional	Critical
Regarding the login system, the Android application will have a "remind me" feature that saves the user ID locally.	Design	Optional
The login system will also have a "reset password system" in which the user can generate a new password if and only he validates personal information.	Non Functional	Important
The Android application will allow the users to send data composed by measures regarding heart function, chronic pain, body temperature and glucose levels. The measure will be accompanied by a time-stamp set by the server.	Functional	Critical

The data will be sent to a remote database that can be accessed at all times and by multiple users, if and only they have a established network connection.	Functional	Critical
The medical and manager oriented application will be developed in a web-page model in order to be cross--platform, avoiding being Windows exclusive.	Functional	Important
The web-page platform will have a login system, in all similar to the one existing in the Android app (including a "reset password system", with a different model of action).	Non Functional	Critical
The web-page platform will block access from managers to medical functions and the reverse will happen with medical personnel regarding managerial functions.	Non Functional	Important
The managers will be able to add, edit and remove patients, doctors and other managers to the system.	Functional	Critical
Upon adding a patient/manager/doctor, a user ID is created and sent to them via e-mail, accompanied by instructions to complete the process of registration.	Functional	Important
The doctors will be able to access, create and edit medical records regarding every patient as well as access basic basic information regarding the patients (that is added when a manager inserts a patient into the system).	Functional	Critical
The doctors will also have access to the data sent to the remote database by the patients and will be able to display said data in a filtered-by-patient table and graph view for every category of measures.	Functional	Critical
The web-page platform will be developed using HTML, PHP and CSS coding language.	Design	Important
Both applications of the platform should be user friendly and easy to learn without the need to check for manuals or other tools that aren't direct and in the moment learning.	Design and Process	Important
Both applications of the platform should be in Portuguese.	Design	Important

Table 2: Table of Final Defined Requirements to be Implemented

As a side note, it's important to note that some of these requirements are more complex than others, i.e., a division of requirements into other "sub-requirements" could have been made, but for the purpose of both keeping the integrity of this project and have a complete set of functionalities, they are presented as a whole.

Following this stage, it is possible to start designing and developing this project. For future reference, the three-tiers of this platform-project will be referenced as Android-Application and Web-Application.

4 Project Design and Implementation

4.1 Introduction

Design and implementation of this project has to be done with consideration to phasing, i.e., implementation of the platform piece-by-piece. Taking into consideration that this project requires a remote database, and both a Android and Web applications, those will be considered the three main phases of this project and along this chapter will be approached separately, with the exception of when it's necessary to draw a connection between them.

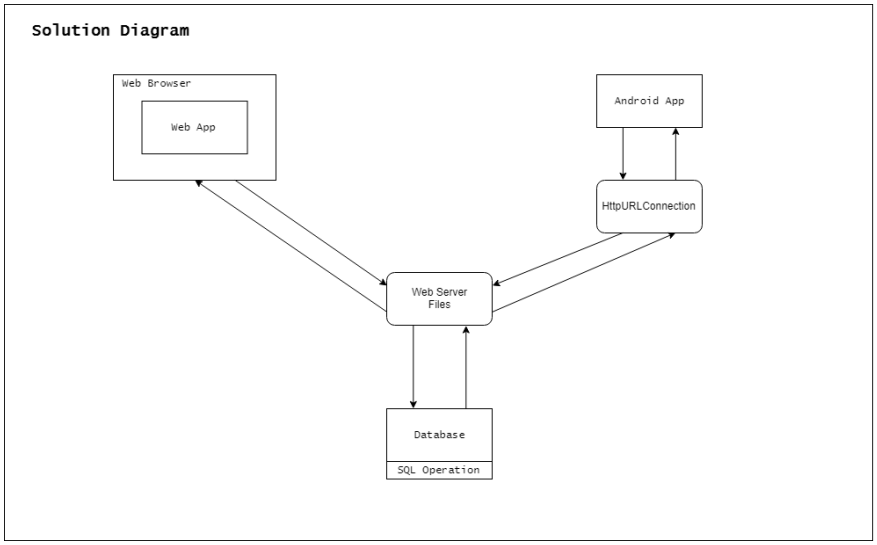


Figure 4: Diagram of Implemented Solution

Figure 4 displays a simple diagram with the solution implemented in this project. The presented structure shows three main components, Web-application, Android-application and the database that communicate differently, either directly communicating with the server or using a intermediate class as a middle-man. The presented structure was totally designed and implemented from scratch as part of this project and will be both exposed and explained during this chapter.

4.1.1 Tools Required

In order to develop this project it's necessary to have a specific set of tools, mostly Integrated Development Environments, IDE's, and Software Development Kits, SDK's, which will be presented briefly as required tools to the developing of this project.

Regarding the database, as mentioned in section 2.2.4, SQL will be language used. In terms of IDE to use it will be the MySQL Workbench, which is open-source. Knowledge about using this language and IDE already existed and for that reason it wasn't required to take time in order to become familiar with them.

The same principle applies to HTML and PHP languages. Knowledge already existed so it was only a matter of choosing which IDE to use. Based on familiarity, the IDE chosen was Microsoft's Visual Studio Code which is also an open-source IDE that allows to develop code on many languages like HTML and PHP. In order to compile, execute and debug the code a server, provided by the faculty to students was used. The only requirement was to send the

files to that server, which was accomplished through a Secure Shell File Transfer Protocol, SFTP, used in a program called WinSCP, free to use as well. To design and implement this project, if a remote server isn't available, a personal one can be created with the aid of an old computer that can be permanently on, or a web development platform like WampServer can be used, having a core difference that it will use the computer's localhost, i.e., the server will be identified by the computer's Internet Protocol address.

Lastly, Android developing, therefore JAVA programming wasn't comparable in terms of knowledge to the previously mentioned languages, so it required to develop some small simple applications and access the existing documentation [27] in order to develop a functional application. In terms of IDE the choice tended towards Android Studio, an IDE that continues the trend of free to use and has a good amount of tutorials and documentation. In order to work properly both JAVA and Android SDK's were required to be downloaded which is easily done online. Android Studio also has the advantage of having an embedded Emulator that allows to run the code in all the existing versions of Android operative systems, provided that the minimal selected Application Programming Interface, API [28], is compatible with that version.

With this set of tools it will be possible to reproduce the design, implementation and results of this project, using its source code.

4.2 Modeling

Before starting the coding side of this project, it's a good and well defended practice to take the gathered requirements and model them into a well founded structure that will be the basis that supports the coding decisions taken. There are many design techniques that can be used, one of the most common is based on the Unified Modeling Language, UML, like use case, class, activity and sequence diagrams, among others.

4.2.1 Android-Application

1. **High-Level Design:**

   First of, a high-level architectural design can be presented by figure 5.

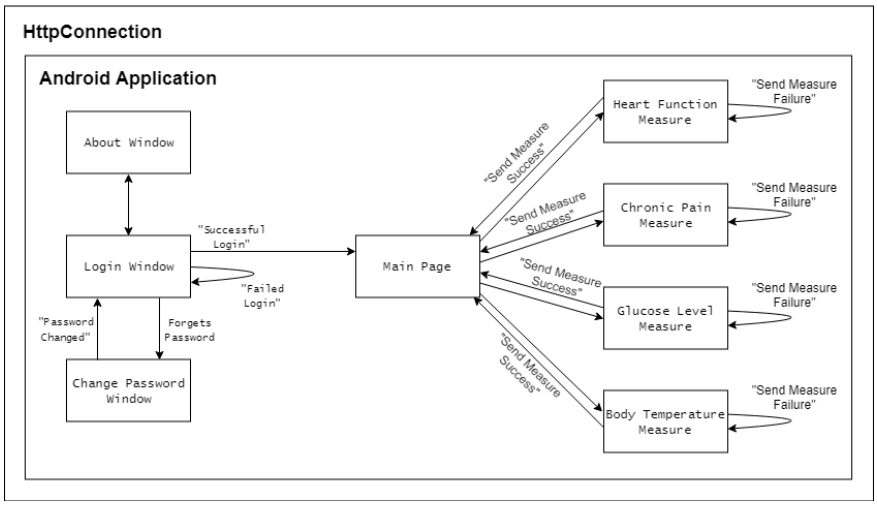


Figure 5: High-Level Design of the Android Application

In it, the basic structure of the application is presented. Despite using a database to login, change password and store the patient's measures, it doesn't "talk" with it directly. Instead, a specific JAVA class[29] is used as a middle-man to reach a server-sided PHP file that will perform those operations. Therefore, it's necessary to create several PHP files, one for each operation, that stay in the server and wait for a call to action. This usage will be further explained in section 4.5.2.

This design model acts like an informal introduction to the architecture presented in detail through the a class diagram.

2. **Class Diagram:**

The class diagram won't be shown in the body part of this document, due to it's size. Instead it will be presented in figure 16 of the appendix's section 9.1.

It's important to note that the presented version of the diagram is final. During the modeling stage of a software engineering project, prior to the coding itself, it's very common to model functionalities that end up not being implemented at all or even ending up not modeling functionalities that are concluded to be necessary for the project, therefore presenting all the diagram's versions wouldn't be relevant.

From a quick analysis it's possible to understand the structure and class relations of this application. Starting with the login page as the "top class", the application will progress towards the home page where the patient will be able to send measures for the remote database, resorting to validating methods and functions, which will further explained in section 4.5.2 of this dissertation

4.2.2 Web-Application

1. **High-Level Design**

Much like the Android-application, a high-level design can be presented as an introduction towards the more formal modeling of the Web-application. That design is presented in figure 6.

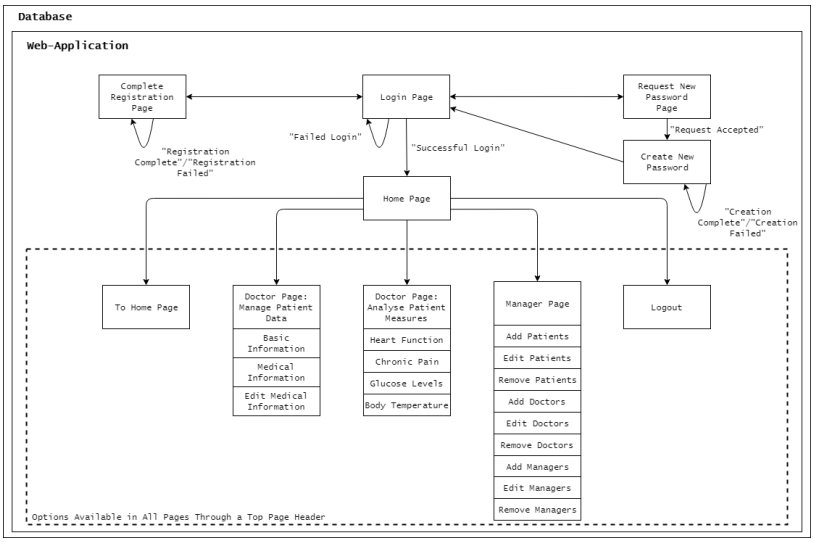


Figure 6: High-Level Design of the Web-Application

The logic behind this diagram in terms of design is very similar to the one in figure 5 where we can see that the starting point of this application is the login page. Like in the previous case as well, the layered structure is considered. Despite not requiring a class method like the Android-application, some preset functions are required in order to establish connection with the database and operate it.

To keep the diagram simple, all the sub-windows associated with three of the five main functionalities of this application aren't expanded. Regardless, the coding aspect of this sub-window functionalities will be approached in section 4.5.1.

2. **Package Diagram (simplified)**

Regarding UML, for a website, it's more common to see package diagrams. Having into account the complexity in operations that the Web-application has, a simplified version of a package diagram is present in figure 7 as a basis for understanding this side of the project.

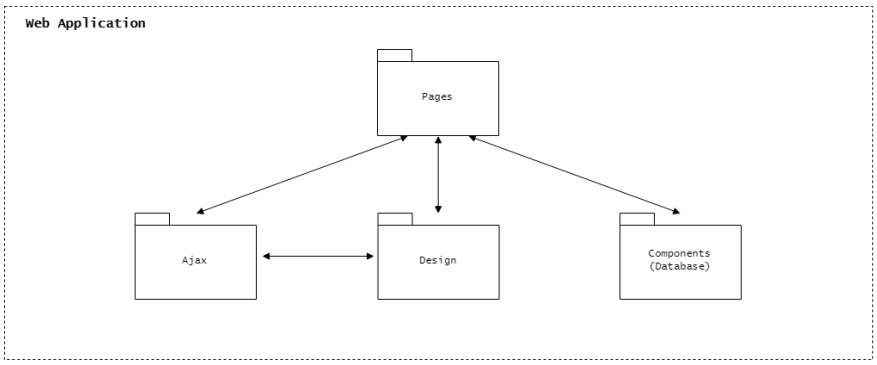


Figure 7: Simplified Package Diagram of the Web-Application

Looking at the diagram, a division between four main packages is presented. At the top there is a package referencing the many pages of the Web-application. It's where all the files with whom the user will interact while using the application are. That package is connected to three distinct packages that handle different aspects of the application. Starting with the Design package, as the name intends, its purpose was to contain all the code files, in CSS, which stand for Cascading Style Sheets, and PHP language, that handle the visual part of the application. Also integrated in a way with the Web-application's appearance there is the Ajax package, that references AJAX, Asynchronous Javascript and XML, methods that will be further discussed in section 4.5.1. Lastly there's a package named components, where all the PHP files that use a specific family of functions to execute operations with the database are located.

4.3 Database

In this section the design and implementation of the database will be presented and discussed. As previously referenced, the language used will be SQL and the IDE will be MySQL Workbench. Before presenting the code involved in this process, a design analysis and discussion will take place in order to understand how the tables, that represent collections of related data, i.e., that is associated with one single entity, are connected.

4.3.1 Design

The design of a database, much like modeling software is very important to the project because it will avoid the creation of unnecessary tables both for the entities itself and the relationships between them, if they are of type many-to-many. Figure 8 presents the final design of the database structure. Like the Android-application's case diagram, the first and final designs tend to be different so it wouldn't be relevant to present them all.

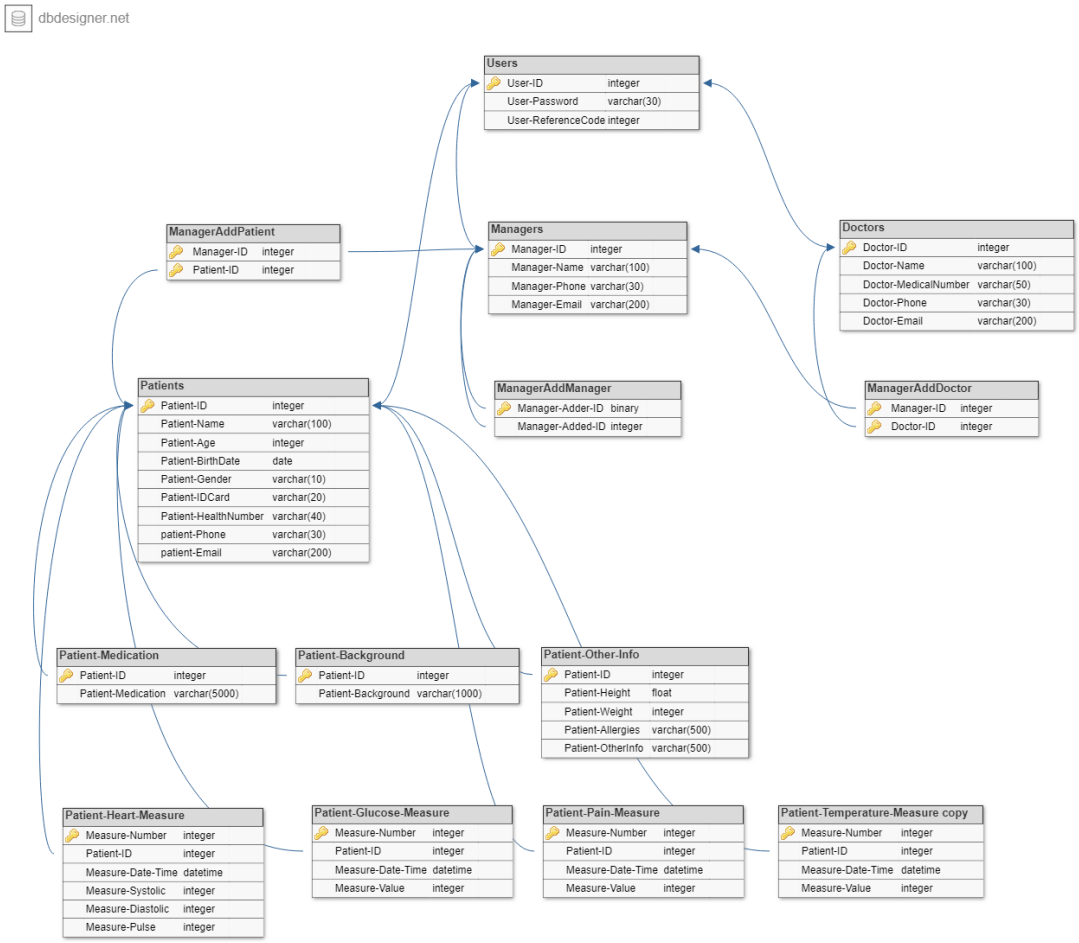


Figure 8: Database Design Model

Before describing the structure presented, one must notice the key symbols that are associated with some attributes of the tables. Those are primary keys, which represent an attribute that is unique in all the table's data, working like both an identifier and a differentiator of data tables, which will help performing targeted queries, which are statements in SQL that allow to fetch, update or modify all the data associated with one or various data attributes that fill up the database tables.

The database's structure starts out with a *Users* table in which the ID, Password and Reference Code, which is a solution to allow or block access to the application's functionalities, will be stored. From there users can be patients, managers or doctors which is set by a relationship of type IS-A, meaning that a user "is-a" patient, doctor or manager. Also, the primary keys of these three tables are equal to the one set on the Users table, which requires

a foreign key to be set. These are described as a link between two tables that share the same attributes but aren't necessarily primary keys. This turns out to be a some sort of security measure in terms of blocking data from being deleted. For example, if someone tries to delete a user's data from the Users table without previously deleting all the link references to other tables, i.e., deleting all the attributes related to that link, an error will occur and won't allow that deletion to proceed. This also means that this is a critical point to take care of while implementing functions that delete data. All this three tables have different attributes in them, with the patients table being the most filled table, containing data that are used for secretary purposes, for example, in order to contact a patient about their condition it's useful to have their e-mail and phone number.

The adding, editing and deletion of user's will only be granted to managers. Therefore, in order to keep track of "who-added-who", which describes a many-to-many database relationship, meaning that different managers can add multiple users, three tables were created that store the data of both the added user, being it a patient, doctor or other manager, and the one who added them.

Since this will be a platform designed mostly to store patient's electronic health records, the remaining tables of this database will be dedicated to storing health records, related to each patient, like their medication, medical background, allergies, basic health data and the measures that patients send via Android-application. These tables are related to the patient in a one-to-one relationship meaning that each set of health records belongs to only one patient. The measure tables will contain, besides the data related to the measures themselves, a date and time stamp that will allow the doctor to analyze the data, knowing the date and time where a particular measure was taken. These measures will have a "order of entrance" attribute as its primary key since it isn't possible to repeat the patient's ID more than once in that table. This issue could be avoided by setting the relationship to many-to-many but that would increase the complexity of this project.

4.3.2 Implementation

Considering the design structure presented in the previous chapter, the only action left, regarding the database's implementation process, is the coding of this structure through the use of SQL language with MySQL Workbench. From the IDE, it's possible to access a personal database that every student, who attended the Databases course, has and perform the operations related with the creation of data tables existent in the project's database. Syntax behind the creation of tables is similar, variating only on the attributes and key links that are set, therefore not all the code will be presented in this section, but it will be referenced as well, if needed. Also, it's important to notice that the names of both attributes and tables are different then the ones present in figure 8. This was due to the fact that, the attribute names given during the implementation phase of the database might not be as perceptible in comparison to the names presented in figure 8. Regardless the attribute values themselves stay the same.

```
1 CREATE TABLE users (
2         userId INT NOT NULL,
3         pWord VARCHAR(30) CHARACTER SET UTF8 COLLATE UTF8_GENERAL_CI NOT
              ↪ NULL,
```

```
        referenceCode INT,
        PRIMARY KEY (userId)
) ENGINE=INNODB CHARSET=UTF8 COLLATE UTF8_GENERAL_CI;
```

Starting out with the users table code, it follows from the table's diagram presented in figure 8. The three main attributes, upon creation must be defined in type, based out of MySQL's Reference Manual[31]. *userId* is set as a INT NOT NULL attribute which means that it will only accept values of type integer that aren't null, *pWord* is set as a VARCHAR(30) which means that it will be an array of characters with a maximum length of 30. Besides that specification, it is also specified the encoding type of the characters that will be added. Out of a various amount of possibilities, UTF-8 encoding was the one selected because it's compatible with all the ASCII characters which is a great advantage in avoiding errors regarding punctuated characters that are displayed has "strange symbols". *referenceCode* is only set as an INT type attribute because one of the reference codes that a user can have will be null, which will be further explained in section 4.5.1. After setting the attributes and their types, it's necessary to set which one of them is the primary key. That is easily achieved using the *PRIMARY KEY* statement. Finally, it's important to set the table's engine and character settings regarding encoding which follows the same logic behind *pWord* attribute with the adding of a *COLLATE* statement that set's rules regarding comparisons between the characters in a character set. The choosing of *INNODB* as an engine brings compatibility with foreign keys which will be used in this database as explained while discussing figure 8.

```
CREATE TABLE patients (
        patientId INT NOT NULL,
        patientName VARCHAR(100) CHARACTER SET UTF8 COLLATE UTF8_GENERAL_CI
            ↪  NOT NULL,
        patientAge INT NOT NULL,
        patientBirthDate DATE NOT NULL,
        patientGender VARCHAR(10) CHARACTER SET UTF8 COLLATE
            ↪ UTF8_GENERAL_CI NOT NULL,
        patientIdCard VARCHAR(20) CHARACTER SET UTF8 COLLATE
            ↪ UTF8_GENERAL_CI NOT NULL,
        patientHealthNumber VARCHAR(40) CHARACTER SET UTF8 COLLATE
            ↪ UTF8_GENERAL_CI NOT NULL,
        patientPhone VARCHAR(30) CHARACTER SET UTF8 COLLATE UTF8_GENERAL_CI
            ↪  NOT NULL,
        patientEMail VARCHAR(200) CHARACTER SET UTF8 COLLATE
            ↪ UTF8_GENERAL_CI NOT NULL,
        PRIMARY KEY (patientId),
        FOREIGN KEY (patientId)
        REFERENCES users (userId)
        ON DELETE CASCADE ON UPDATE CASCADE
) ENGINE=INNODB CHARSET=UTF8 COLLATE UTF8_GENERAL_CI;
```

Next, the code regarding one of the three user possibilities, patients, is presented. Regarding this table there are some key aspects to take notice, that are different from the code of associated with the users table. For example, patient's birth date is stored as a *DATE* attribute. Based out of the reference manual[31], DATE refers, as the name states, to a calendar date. Its format is specified in 'YYYY-MM-DD' requiring attention while adding that information. Another aspect to take notice is related to the fact that attributes regarding patient's *ID Card*, *Health Number* and *Phone* are both set as characters instead of digit type attributes and have a considerate maximum limit of characters. This is due to a security measure based on the encoding of sensitive data. While adding users, and therefore this set of sensitive data, an encoding PHP function will be used, *base64_encode*[30], causing a growth of the encoded string that is, on average, at about $\frac{4}{3}$ the original size of the attribute. Therefore an increase in character space is required and done by excess to avoid errors. Finally, this table shows how the setting up of the link between two tables is done, resorting to the *FOREIGN KEY*. The statement connects this table with the users table through the *REFERENCES* statement and adds the delete/update cascade clause. In order to understand how this works, consider the case of a parent and child. With this set linkage between the two, if a parent is killed, no child can become an orphan, meaning that all children related to that parent will also have to be killed. The same principle applies to the update process. If a parent is updated, all the children must be as well in order to avoid errors.

```
CREATE TABLE managerAddPatient (
       mngID INT,
       patID INT,
       PRIMARY KEY (mngID , patID),
       CONSTRAINT fk_mngID1 FOREIGN KEY (mngID)
       REFERENCES managers (managerName)
       ON DELETE CASCADE ON UPDATE CASCADE,
       CONSTRAINT fk_patID1 FOREIGN KEY (patID)
       REFERENCES patients (patientName)
       ON DELETE CASCADE ON UPDATE CASCADE
) ENGINE=INNODB CHARSET=UTF8 COLLATE UTF8_GENERAL_CI;
```

This table represents the relationship table mentioned in section 4.3.1. If a many-to-many relation exists, it's required to create a table that relates those tables. The logic behind this code syntax is based in the combination of primary and foreign key notions with constraints. These are, rules that can be set during or after the creation of a table that limits the type of data that can be added to that table. This method insures that data in the table is both accurate and reliable in its type and or relation to other tables, aborting operations that don't comply with the rules set by the constraints. In this case, the constraint is used to ensure that the data added to the table, in this case the ID's of both the manager and patient already exist as attributes in the tables which they are related to, in this case the patients and managers table.

```
CREATE TABLE patientOtherInfo (
```

```
        patId INT,
        patientHeight FLOAT,
        patientWeight INT,
        patientAlergies VARCHAR(500) CHARACTER SET UTF8 COLLATE
            ↪ UTF8_GENERAL_CI,
        otherInfo VARCHAR(500) CHARACTER SET UTF8 COLLATE UTF8_GENERAL_CI,
        PRIMARY KEY (patId),
        FOREIGN KEY (patId)
        REFERENCES patients (patientId)
        ON DELETE CASCADE ON UPDATE CASCADE
) ENGINE=INNODB CHARSET=UTF8 COLLATE UTF8_GENERAL_CI;
```

```
CREATE TABLE patientHeart (
        heartMeasureNumber INT UNSIGNED NOT NULL AUTO_INCREMENT,
        patId INT,
        heartDateTime DATETIME,
        systolicValue INT,
        diastolicValue INT,
        pulseValue INT,
        PRIMARY KEY (heartMeasureNumber),
        FOREIGN KEY (patId)
        REFERENCES patients (patientId)
        ON DELETE CASCADE ON UPDATE CASCADE
) ENGINE=INNODB CHARSET=UTF8 COLLATE UTF8_GENERAL_CI;
```

The final two snippets of code regard patient's health data, presenting the "biggest" tables in this database, i.e., *patientOtherInfo* the ones that regard basic health information, which includes both the medication and medical background of the patient and *patientHeart* which represent the data tables that store the measures that patients can send via the Android-application, chronic pain, glucose values and body temperature. As stated in section 4.3.1, all the measures sent will be accompanied by a date and time stamp which is represented by the *DATETIME* type attribute existing visible in the second snippet. From documentation[31], this type attributes are stored in the format 'YYYY-MM-DD HH:MM:SS', which again requires attention from the back-end part of this application.

4.4 Interface Design

Interface designing is considered an important part of software development, categorized partly as front-end developing, stage where the user can interact with the application, see and enter the data that is processed by the background function of the application, commonly known as back-end. Therefore, this section will show some images regarding the interface of both Web and Android interfaces while discussing the process of thought behind the design and how it was implemented. The main goal of the design was to verify one of the requirements presented in table 2 of section 3.2.2, stating that "Both applications of the

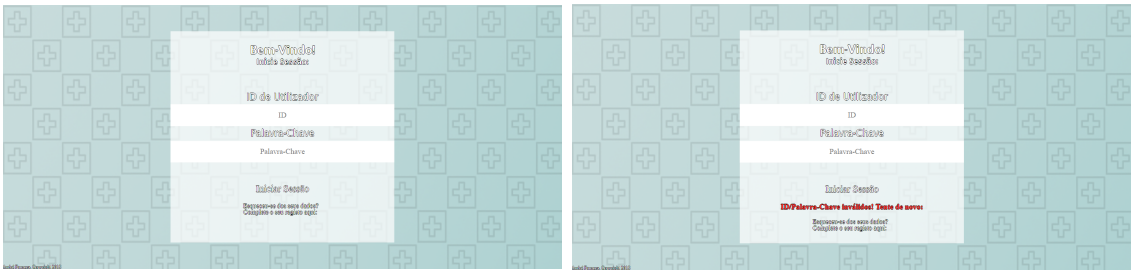
platform should be user friendly and easy to learn without the need to check for manuals or other tools that aren't direct and in the moment".

4.4.1 Web Interface

Starting with the Web-application, designing a Web page was made using the combination of mainly two languages, HTML and CSS. Towards the end part of the design, AJAX, jQuery and Bootstrap related functions were used to simplify the application's interface usage. Considering that the Web-application consists of 22 different pages, some are designed with the same goal or share the same layout. For reference, 3 pages were designed to add a patient, doctor or manager, 6 other pages for editing and removing the same three entities, 4 pages were designed to display patient's measure data and the login, complete registration and password reset pages have the same layout. Therefore the number of pages presented in this section will be much smaller.

1. **Login Page**

   Accessing the Web-application will greet the patient with a simple login page. The design idea intended to use bright colors that don't cause a visual shock to the user.



(a) Web Interface Design: Login Page (b) Login Page with Failure Notification

Figure 9: Web Interface Design: Login Page

   Both the color pallet and background image will be used as the theme for all the pages. As it's possible to analyze, the user has two fields to fill in order to login in the application. Besides that, below the login button, user's have two buttons that will redirect them to a "forgot password" page and a "complete registration" page which have a similar layout as the one presented in the login page.

   If the user fails to login, a patient tries to login or someone tries to access the page without filling the fields or fills them with wrong values, a small text notification will be shown in figure 9b. It will vary between a empty or invalid fields notification and a information that patients can't use the Web-application and must download the Android-application.

   In order to display the page with this setup, as referenced in beginning of this sub-section, a combination of HTML and CSS was used. Snippets of the code used for this setup will be presented next. Since the logic behind HTML and CSS coding syntax will be similar in the multiple pages, the explanation regarding syntax will serve all the snippets of this sub-section.

As a way of distinguishing identifiers and common text, a differentiation between bold and non-bold words will occur.

```
<!DOCTYPE html>
  <html>
   <head>
    <title>Adiutor | Assistente Medico</title>
    <meta charset="utf-8">
    <link rel="stylesheet" type="text/css" href="design/design.css">
   </head>
   <body id="loginBody">
    <div class="background">
     <div class="login-square">
      <h1>Bem-Vindo!</h1>
      <h2>Inicie Sessao:</h2>
       <form action="components/db_login.php" method='POST'>
        <p>ID de Utilizador</p>
        <input type="text" name="userID" placeholder="ID">
        <p>Palavra-Chave</p>
        <input type="password" name="userPW" placeholder="Palavra-Chave
            ↪ ">
        <button type="submit" name="loginSubmit">Iniciar Sessao</button
            ↪ >

        <a href="pages/data_forget.php">Esqueceu-se dos seus dados?</a>
            ↪ <br>
        <a href="pages/signup.php">Complete o seu registo aqui:</a>
       </form>
      </div>
     </div>
    </body>
 </html>

<footer>
 <div class="footer-info">
   <p>Andre Fonseca, Copyright 2018</p>
   </div>
</footer>
```

Starting with the HTML side of the code, there are references to multiple tags. In HTML, tags are tools used to identify code elements, like the *<head>*, *<body>* and *<footer>* tags that define the top, middle and bottom parts of the web page. In total, there are 13 different tags being used in this code snippet, which transfigures onto the CSS file, needing to style all these 13 tags. A snippet of the CSS code that styles this page is shown in figure section 9.2 of the appendix to avoid filling the document with

three pages of just code.

In order to use a CSS file to stylish a page linking is required. First, in HTML, it's necessary to have a *<link>* tag that tells where to get the file that is responsible for the page's styling that is, in this case, on another folder/package of the application's structure. In the CSS file it isn't necessary to create this link. All it takes is the usage of different classes and element names, which must be done in the HTML file, and it will automatically style those elements and classes that compose the web page's structure. With the CSS file, it's possible to style things from shapes to lettering, movement and many other proprieties revealing itself as very useful. In order to reach the snippet presented a lot of trial and error was involved because computer and also smartphones, have different screen sizes and resolutions, therefore it's required to test out the layouts in various screen sizes and adjust the both class and element style proprieties until a final adaptive layout is reached.

2. **Home Page**

After successfully login onto the application, users will be direct towards the application's home page presented in figure 10. The page in itself hasn't much to show as of now, only presenting a greeting message. In the future simple widgets can be considered for this page.

Figure 10: Web Interface Design: Home Page

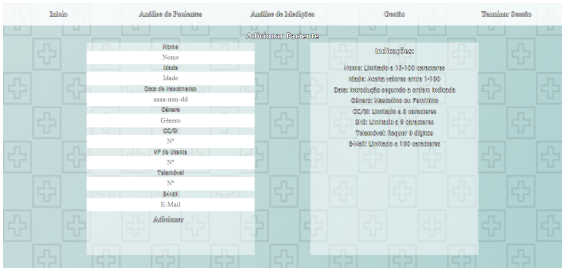
The main focal point of this figure stands in the page's top with the header that is transversal to all the application's pages. With it, from any point the user will be able to drop down a list with all the pages regarding that category without having to go back into the home page. An example of its utilization can be shown in figure 11.

While scrolling through the dropped down options a color hover will appear as an aid for the user to be sure about what option he's choosing. The CSS file doesn't change in its thought structure but, the HTML file's changes slightly. Instead of using forms and other classes it's based out of list tags that are styled in order to be presented at the page's top, using CSS properties. To make this header transversal to all the application's pages a file named *header.php* was created. This file could be called on through the *<link>* HTML tag in other files and drop the page's files in size and avoiding coding repetition rendering the application more efficient in the process which is also an important and common goal while developing software applications.

Figure 11: Web Interface Design: Home Page with Header Usage Example

3. **Add, Edit and Remove Page**

As referenced in the beginning of this section most pages are grouped in terms of layout design. One example is the adding, editing and removing users page. The layout of those pages, regarding only the patients, which are the most complex and filled layouts, is presented in figure 12.



(a) Adding Users Example

(b) Editing Users Example

(c) Removing Users Example

Figure 12: Web Interface Design: Adding, Editing and Removing Users Pages

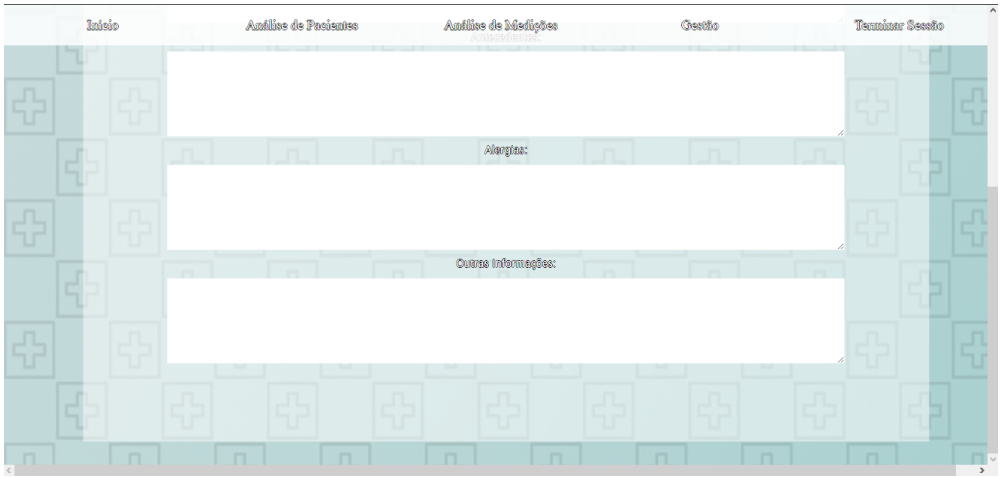
Analyzing the layouts, adding and editing users present pretty much the same type of layout. Only one main difference can be seen. In figure 12b there is a select box on top of the left square where all the user's information is filled out. That select box will contain all the patient's names existing in the database and upon the choice of one, the input fields will fill up automatically, i.e., without refreshing the page, with all the information associated to that patient. The function behind that automatic update will be referenced in section 4.5.1. Regarding the layout for user's removal page, the one associated with patients is the most complex compared to the other

two. That is based with the fact that, upon removal, selecting a user in the select box, like on the editing layout, the manager must tick boxes regarding what kind of measures the patient sent. This measure is implemented in order to avoid the existence of data garbage in the database like measures that are associated to a patient that no longer exists in the system. On both the doctor's and manager's removal page the only needed action is the selection, through the select box, of the doctor/manager's name and pressing the remove button.

4. **Patient's Medical Records Page**

The final layouts to be presented regard only the doctor's side of the application. First off the medical records page. The layout logic is the same as previous presented layouts, differentiating only on the page's and input fields size. To increase these specified sizes only a change of proprieties in the CSS fields are required, making it a simple task to do. The page's full layout is present separated by two figures in figure 13.

(a)



(b)

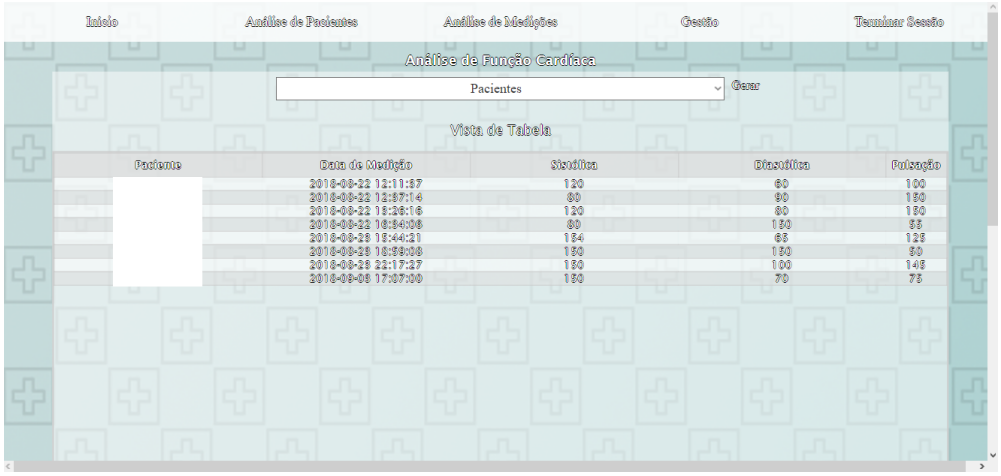
Figure 13: Web Interface Design: Patient's Health Records Page

5. **Patient's Measure Analysis**

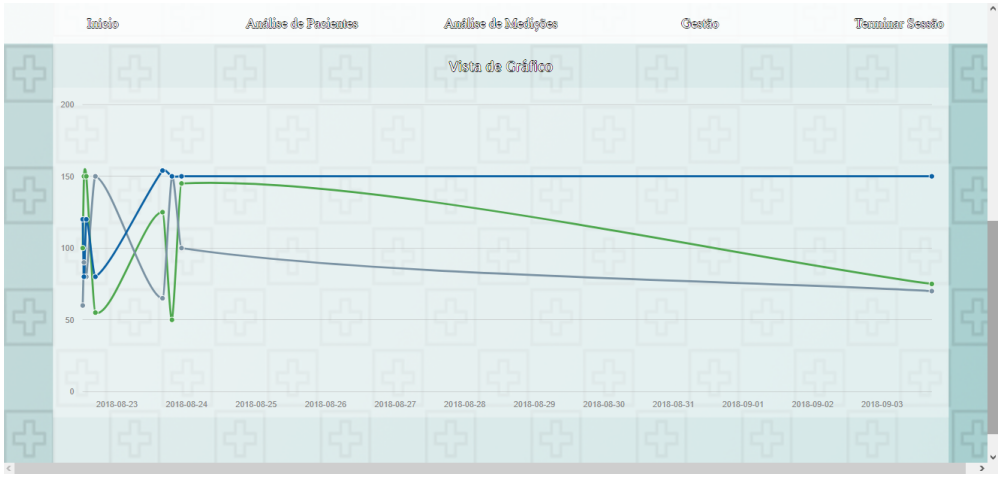
   This final page layout repeats itself four times with the only changes being the data shown. In order to fill all this data, and predicting that in the future data will be much bigger then the one displayed while testing the application, like the previous page layout, the page's size was increased.

   The layout is based out of two separate parts, table and graph view. The functions used to display both the table's data and graph will be discussed in section 4.5.1, but regarding the table, its layout construction was made using the combination of HTML table tags and the CSS file that styles all the pages.

   A quick notice regarding the displayed data presented on figure 14. The "patient" name won't be disclosed in order to maintain the tester's privacy. Also the displayed graphic isn't considered as a expected graph behavior due to the first set of measures being uncontrolled, i.e., validation functions weren't yet implemented to handle bad data inputs. More will be discussed in section 5.3.



(a)



(b)

Figure 14: Web Interface Design: Patient's Measure Analysis Page

4.4.2 Android Interface

Designing the Android-application's interface was much easier than the Web-application's. That was due to the fact that Android Studio, the IDE used to develop its software has an built-in visual layout designer. Therefore its implementation was much easier and is presented in this sub-section on a more simplified manner than the previous interface presentation. One point to take notice before starting is that each layout had to be replicated three times in order to be compatible with different screen sizes and resolutions. Contrary to the Web-application where it was possible to adapt many screens and resolutions setting up proprieties values in the CSS file, that wasn't possible. Therefore, a layout for normal, 2.9"x2" to 4"x3", large, 4"x3" to 6"x4.5" and extra-large, 6"x4.5" up, screen sizes were set up, requiring the adaptation of both icons, in a very similar way to the process used for screen sizes but considering the density of screens, buttons and texts. Also the application was blocked from rotating preventing the need to set up layouts for landscape mode which would increase the complexity of this application.

The layouts are presented together in figure 15 after the textual presentation of the layouts. It will serve as guidance through the sub-figures and a way to explain its setup.

1. **Login Page, 15a**

   Starting out with the login page, which is the first thing that the user, patients to be more specific since the Android-application is exclusive to them, will see. As referenced in the beginning of this sub-section the design was made using an in-built layout designer that fills up the XML, Extensible Markup Language, file with all the proprieties and position values in the layout set by the developer. That is a much more efficient way to design a layout and avoids having the constantly compile in order to view the display.

   The login window uses a very minimalist design with a top two input fields for the patient's ID and password, has a tick box to tell the application to remember the patient's ID and two text buttons, one for login in and another to reset and set a new password within the application whose layout is very much similar varying only in the number of input fields.

2. **About Page, 15b**

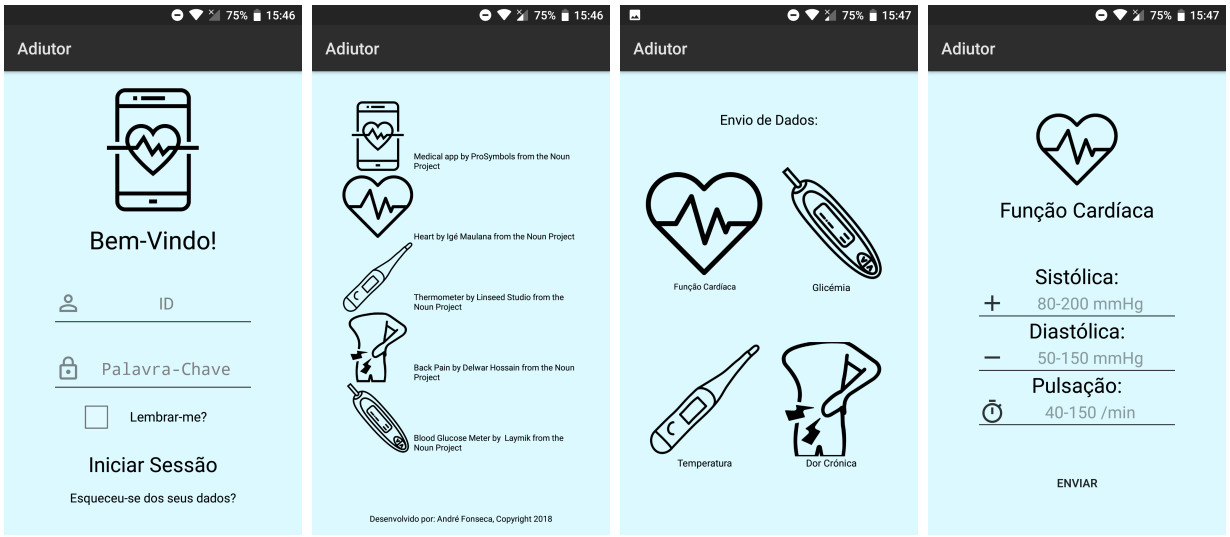
   To access this page the user must press the icon presented in the login window. It works as a "easter egg" which represents a hidden feature of the software, avoiding the user from distractions while using the application. In it credit is given to the designers whose icon's the application uses. These icons were obtained in a website called *The Noun Project* where designers upload their icon designs for developers to use. They can either buy the icons acquiring full rights of the icon or give the designers credit in their applications, which is only fair. The page serves only for this purpose.

3. **Home Page, 15c**

   If the patient successfully logs into the application, its home page is presented. Again with a very minimalistic design, the patient will be presented with four image buttons that will open the windows that are used to send measure data. Its only purpose, design-wise is to separate all the measure windows. For future work, 7.2, some recommendations will be made in order to expand this window.

4. **Measure Send Page Example, 15d** This layout out of the four send measure windows is the most complex since it is the only one that has more than one measure to send. Regardless, the four windows share the same type of layout. An identifying icon at the top, equal to the one that is shown in the home page window, a set of identifying labels, input fields that contain the interval of values that are accepted by the application, by medical recommendation and a send button.

As a final design note towards the Android-application, while transitioning between windows, login in, creating a new password, sending or failing to send measures messages in the bottom part of the window will be shown informing the patient that their actions were successful or unsuccessful and if they were, why. This design feature was implemented in order to aid the patient to understand and work with the application easily.



(a) Login (b) About (c) Home (d) Send Measure

Figure 15: Android Interface Design Layouts

4.5 Background Work

This section will expose mostly coding snippets, accompanied by explanation regarding back-end processing and some of the front-end functionalities that by working with functions weren't viewed as pure design and front-end functionalities.

4.5.1 Web

Starting once again with the Web-application of this project, the snippets of code displayed will mainly related to the database operations that are made purely with PHP code and, as mentioned in the start of this section some PHP code and a set of AJAX and jQuery functions related with design features will also be shown. In order to avoid repetition, out of all the code, only some snippets regarding the main functions will be shown. Only the first two snippets will show actual pure PHP code to demonstrate the code's structure. Following that a practical and simpler way to explain what the code does can be achieved through

the display of a simplified version of pseudo-code with some references to the functions used to reproduce the pseudo-code action. This is only regarding the files which only serve to operate the database. Code regarding AJAX and jQuery will be shown as a snippet of full on code.

1. **Database Connection**

   In order to be efficient with the multiple database connections that are required within all the back-end processing, a separate file named *db.php* was created with the code presented in this snippet. All that is required on the other files is to include the contents of this file resorting to *include* or *require* commands.

```
<?php
 //Remote database connection params
 $username = "xxxxxxxxxx";
 $password = "xxxxxxxxxx";
 $database = "xx_xxxxxxxxxx";
 $server = "xxxxx.xxxx.xx.xx";

 //Establish the connection
 $connection = mysqli_connect($server,$username, $password, $database);
 mysqli_select_db($connection, $database) or die("Nao foi possivel encontrar a Base de Dados");
 mysqli_set_charset($connection,"utf8");
 ?>
```

   As a side note to preserve the integrity and security of the database, all the variables will be covered up.

2. **Login**

   The login file comprises a great amount of functions and notions used while setting up the background processing of the application. For example, most of the functions related with the database are represented by the *mysqli*[32] group. Starting with the *mysqli_real_escape_string()* which will prevent injection attacks to the database. These can be done for example inputing SQL statements to delete the database on one of the input fields available in the application's layout. Therefore, this function exists to avoid this security threats. Another set of this class functions can be seen in *mysqli_query*, *mysqli_num_rows()* and *mysqli_fetch_assoc()* that, by order execute a SQL statement in the database using its connection values, check if the query was successful by checking if any row alteration took place and fetch data out of the query, if it was successful.

```
<?php
 session_start();
```

```
3
4  if (isset($_POST['loginSubmit'])) {
5  include 'db.php';
6
7  $userID = mysqli_real_escape_string($connection, $_POST['userID']);
8  $userPW = mysqli_real_escape_string($connection, $_POST['userPW']);
9
10 if (empty($userID) || empty($userPW)) {
11  header("Location: ../index.php?login=empty");
12  exit();
13 }
14 else {
15  $sql = "SELECT * FROM users WHERE userId = '$userID'";
16  $result = mysqli_query($connection, $sql);
17  $resultCheck = mysqli_num_rows($result);
18
19  if($resultCheck < 1) {
20   header("Location: ../index.php?login=error");
21   exit();
22  }
23  else {
24   if ($row = mysqli_fetch_assoc($result)) {
25    $decodedPW = base64_decode($row['pWord']);
26    if (strcmp($userPW, $decodedPW) != 0) {
27     header("Location: ../index.php?login=error");
28     exit();
29    }
30    else {
31     $refCode = $row['referenceCode'];
32     $allowedRC1 = 0;
33     $allowrdRC2 = 100;
34     $allowedRC3 = 1;
35
36     if ($refCode != $allowedRC1 && $refCode != $allowrdRC2 && $refCode
          ↪  != $allowedRC3) {
37      header("Location: ../index.php?login=notallowed");
38      exit();
39     }
40     else {
41      $_SESSION['u_id'] = $row['userId'];
42      $_SESSION['u_pw'] = $row['pWord'];
43      $_SESSION['u_rc'] = $row['referenceCode'];
44
45      header("Location: ../pages/mainpage.php?login=sucess");
46      exit();
47     }
```

```
48      }
49     }
50    }
51   }
52  }
53  else {
54   header("Location:␣../index.php?login=error");
55   exit();
56  }
57  ?>
```

Pseudo-Code:

```
Start a session.

Check IF a button request was correctly made.
 IF IT IS, proceed.
  Include file 'db.php'

  Set variable userID = Value sent through POST Method named userID
     ↪ while checking if the variable isn't a SQL statement.
  Set variable userPW = Value sent through POST Method named userPW
     ↪ while checking if the variable isn't a SQL statement.

  IF userID or userPW are empty
   Display error message and exit process.
  IF NOT proceeed
   Prepare the SQL Statement using userID variable.
   Execute the Statement.
   Store the number of affected rows in the database made by that
      ↪ statement in variable resultCheck.
   IF resultCheck is inferior to 1
    Display error message and exit process.
   IF NOT proceed
    Decode the user's password stored in the database.
    Compare IF the decoded password matches the variable userPW.
    IF IT ISN'T
     Display error message and exit process.
    IF IT IS proceed
     Set variables with allowed reference codes that can access the
        ↪ application.
     Check IF the referenceCode associated with the database query
        ↪ result is one of the allowed refrenceCodes.
     IF IT ISN'T
      Display error message and exit process.
```

```
    IF IT IS proceed
     Setup Session variables that will pass on through the pages of
        ↪ the application for further usage and security
        ↪ verifications.
     Redirect the user to the Home Page and exit process.

 IF IT ISN'T
  Display error message and exit process.
```

3. **Adding a User example - Managers**

In order to avoid a very complex and extensive pseudo-code display, the most simple of the adding operations, which regard managers will be shown. While executing adding operations, the start of file's processing process is equal to the one presented in the previous pseudo-code and it will repeat itself in all the files since forms are used in all pages of the Web-application. Therefore and all the following pseudo-codes will disregard this and repeated parts representing them with '...' and a reference to what part is repeated.

```
...
 Store in auxiliary variables the lenght of the variables.
 ... (checking for variable emptyness)
  Check IF the Manager's Name is only made by letters - preg_match()
     ↪ function.
   IF IT ISN'T
    Display error message and exit process.
   IF IT IS proceed
    Check IF the Manager's Name Lenght is between the valid intervals
     IF IT ISN'T
     Display error message and exit process.
     IF IT IS proceed
      ... (repeated process for the Manager's Phone and E-mail with
         ↪ adapations.
          while checking the E-mail a function filter_var() which
             ↪ validates if the string has an email like structure).

          Encode Phone Number and Email.
          CHECK IF there is any manager with the same phone or email
          (using SQL queries).
          IF THERE ARE
           Display error message and exit process.
          IF THERE AREN'T proceed
           Generate random user ID.
           Set the referenceCode with Manager's code.
```

```
            INSERT Manager's attributes in Users Table using SQL Insert
               ↪ Statement: ID and referenceCode used. The user will
               ↪ setup a unique password in the complete registration
               ↪ page which will update this row with the encoded
               ↪ password value
            Check IF query was successful (mysqli_affected_rows(
               ↪ $connection))
             IF IT ISN'T successful
              Display error message and exit process.
             IF IT IS proceed
              ... (repeat process for Managers Table with Name, Phone
                 ↪ and Email values)
              IF IT ISN'T successful
               Display error message and exit process.
              IF IT IS proceed
                Add the ID's of the manager that added and was added to
                   ↪ the ManagerAddManager table through SQL statement.
                Check IF the query was successful
                IF IT ISN'T successful
                 Display error message and exit process.
                IF IT IS proceed
                 Send a welcome email to the Manager with the unique ID
                    ↪ and instructions to complete the registration.
                 Display success message and exit process.
...
```

Editing and removing users won't be shown since the procedure is identical changing only the SQL statements used on both cases to UPDATE or DELETE, and the non existence of a E-Mail send if the user is removed. If information is edited the user will be notified via e-mail that a change was made and if it wasn't authorized or asked by him he should contact the person responsible by the application. This was implemented as a security measure.

As well, the process involved with adding medical information to a patient's health records is mostly similar, starting with validation checks that the data meets restriction intervals and use of SQL statements, not sending E-mails to the patient.

4. **Filling the Select Boxes**

   In order to fill the select boxes referenced in the design section a very simple PHP code using an SQL statement was used. All it took left was to use a foreach loop to fill the select box's options with an echo of the SQL query results. The snippet of this code is displayed below.

```
1 ...
2 <select name="patient" id ="pat" onChange = "getPatient(this.value)"
     ↪ required>
```

```
<option value=""></option>
<?php
 $query = "SELECT␣*␣FROM␣patients␣ORDER␣BY␣patientName";
 $patients = mysqli_query($connection, $query);
 foreach($patients as $patient) {
 echo ("<option␣value=".$patient["patientId"].">".$patient["
    ↪ patientName"]."</option>");
 }
?>
</select>
...
```

5. **Auto-Update with AJAX**

   On some pages, when selecting a patient in the selected boxes all the inputs existent in that page are filled up automatically, i.e., without pressing a button that does a background process and refreshes the page with loaded data on the inputs, that is much more complex than using the approach showed on the snippet bellow. In this case there are two snippets, one that runs in the user web browser and another that runs in the web server. This was done for efficiency purposes since the code of this separate file was needed more than one time.

```
...
<select name="patient" id ="pat" onChange = "getPatient(this.value)"
    ↪ required>
...
<script>
function getPatient(val){
$.ajax({
 type:"POST",
 url:"../ajax/ajax_populate_patients.php",
 data: 'patient='+val,
 success: function(response){
 var result = JSON.parse(response);
 if (result.response == true) {
  var data = result.rows;
  $("#patientName").val(data.patientName);
  $("#patientAge").val(data.patientAge);
  $("#patientBirthDate").val(data.patientBirthDate);
  $("#patientGender").val(data.patientGender);
  $("#patientIdCard").val(atob(data.patientIdCard));
  $("#patientHealthNumber").val(atob(data.patientHealthNumber));
  $("#patientPhone").val(atob(data.patientPhone));
  $("#patientEmail").val(atob(data.patientEMail));
 }
```

```
 else if (result.response == false) {
  var data = result.rows;
  $("#patientName").val(data);
  $("#patientAge").val(data);
  $("#patientBirthDate").val(data);
  $("#patientGender").val(data);
  $("#patientIdCard").val(data);
  $("#patientHealthNumber").val(data);
  $("#patientPhone").val(data.patientPhone);
  $("#patientEmail").val(data.patientEMail);
 }
}
});
}
</script>
```

The first snippet shows a script that is called when a trigger in the select box occurs which is represented by the selection of, in this case's snippet, a patient. The script contains a function that uses AJAX to get a JSON data object which it equals to a variable and loads it to the inputs that are differentiated by ID proprieties. The semi-background process that occurs is described on the next snippet.

ajax_populate_patients.php

```
<?php
 include '../components/db.php';
 $patientId = $_POST['patient'];

 $sql = "SELECT * FROM patients WHERE patientId = '$patientId'";
 $result = mysqli_query($connection, $sql);

 if (mysqli_num_rows($result) > 0) {
  $data = mysqli_fetch_assoc($result);
  echo json_encode(['rows' => $data, 'response' => true]);
 }
 else {
  $data = '';
  echo json_encode(['rows' => $data, 'response' => false]);
 }
 mysqli_close($connection);
 exit();
?>
```

The logic behind this snippet is very similar to the processing done in the database operation files. The only main difference is that the file echoes a JSON object that is

received by the AJAX function which executing has already being explained.

6. **Fill Tables**

   In order to fill the measure tables, AJAX becomes much more complex to use. Therefore, in this and also the graphic case which are presented together, the pressing of a button was required. Once pressed it triggered the page's refreshing and allowed the PHP code to act like the snippet shows.

```
<?php
 include_once '../components/db.php';
 if (isset($_POST['search'])) {
  $patient = $_POST['patient'];

  if ($patient != '') {
   $query = "SELECT␣patients.patientName␣AS␣patient,␣patientHeart.
       ↪ heartDateTime␣AS␣date_time,␣patientHeart.systolicValue␣AS␣
       ↪ systolic_value,␣patientHeart.diastolicValue␣AS␣
       ↪ diastolic_value,␣patientHeart.pulseValue␣AS␣pulse_value␣FROM␣
       ↪ patients␣INNER␣JOIN␣patientHeart␣ON␣patients.patientId␣=␣
       ↪ patientHeart.patId␣WHERE␣patients.patientId␣=␣'$patient'";
   $search_result = mysqli_query($connection, $query);
  }
  else {
   $query = "SELECT␣patients.patientName␣AS␣patient,␣patientHeart.
       ↪ heartDateTime␣AS␣date_time,␣patientHeart.systolicValue␣AS␣
       ↪ systolic_value,␣patientHeart.diastolicValue␣AS␣
       ↪ diastolic_value,␣patientHeart.pulseValue␣AS␣pulse_value␣FROM␣
       ↪ patients␣INNER␣JOIN␣patientHeart␣ON␣patients.patientId␣=␣
       ↪ patientHeart.patId";
   $search_result = mysqli_query($connection, $query);
  }
 }
 else {
  $query = "SELECT␣patients.patientName␣AS␣patient,␣patientHeart.
      ↪ heartDateTime␣AS␣date_time,␣patientHeart.systolicValue␣AS␣
      ↪ systolic_value,␣patientHeart.diastolicValue␣AS␣diastolic_value,
      ↪ ␣patientHeart.pulseValue␣AS␣pulse_value␣FROM␣patients␣INNER␣
      ↪ JOIN␣patientHeart␣ON␣patients.patientId␣=␣patientHeart.patId";
  $search_result = mysqli_query($connection, $query);
 }
?>
...
<?php while($row = mysqli_fetch_array($search_result)):?>
 <tr>
 <td><?php echo $row['patient'];?></td>
 <td><?php echo $row['date_time'];?></td>
```

```
25  <td><?php echo $row['systolic_value'];?></td>
26  <td><?php echo $row['diastolic_value'];?></td>
27  <td><?php echo $row['pulse_value'];?></td>
28  </tr>
29 <?php endwhile;?>
```

In it, it's clear that it works with SQL statements once more, in a similar way to select box's filling. The main differences are the SQL query's complexity since it required to use a INNER JOIN command that allows queries to execute on more then one table, requiring that both tables share a common attribute, normally a foreign key link. After that it was only necessary to fill the table rows with the query's results.

7. **Morris Graphs**

The last main snippet presented shows the usage of a Morris Graph[34]. The algorithm that is available through GitHub and it's free to use, provided it is given credit. Since the graphics produced by this algorithm are very simple to implement and are good to the eye they were used in this application. To display the graph all that was needed is based on the snippet of code presented bellow.

```
1  ...
2  <p>Vista de Grafico</p>
3  <div id="chart"></div>
4  ...
5  <script>
6  Morris.Line({
7   element : 'chart',
8   data:[<?php echo $chart_data; ?>],
9   xkey:'date_time',
10  ykeys:['systolic_value', 'diastolic_value', 'pulse_value'],
11  labels:['Sistolica', 'Diastolica', 'Pulsacao'],
12  hideHover:'auto',
13  stacked:true
14 });
15 </script>
```

The success and performance of the implementations presented will be discussed in chapter 6.

4.5.2 Android

Android's background work is much more simple, reflection of the great amount of documentation available being it an open-source software and the different levels of complexity that exist between the Web and Android applications. The main focal points regarding the background processing repeat themselves. The most repeated one is the establishment of

connection between application and the remove server, which includes the database. This process was implemented based out of an existing algorithm[29] that uses the *HttpURLConnection* making alterations in order to adapt the code to meet the project's context. Since API 22, which came with Android 5.1, codenamed Kit-Kat, this algorithm started to be used in substitution of the Apache module which used either a POST or GET methods.

In order to setup this connection class, some steps are required. Firstly, the URL to which application is going to connect must be given. This URL will be the address to a PHP file that is placed on the server. Then a *URL.openConnection()* function is needed in order to obtain a new connection. This function will cast a result, either positive or negative, regarding the connection. Secondly a request must be prepared which includes the information to be sent towards the server, in this case the data to validate a login or regarding a patient measure, via POST method, also set up as a propriety of the request. After sending the information, a output given by the PHP code must be received signaling success or failure of the operation.

The following code snippet shows one of the PHP files that this class connects to, showing a similar code flow presented in the snippets shown in section 4.5.1, receiving the data, validating it and performing the database operation using the *mysqli* class functions.

```
1  <?php
2   include "db.php";
3   $result = '';
4
5   if (isset($_POST['userId']) && isset($_POST['password'])) {
6    $uID = $_POST["userId"];
7    $uPW = $_POST["password"];
8
9    $uID_int = intval($uID);
10
11   $loginQuery = "SELECT␣*␣FROM␣users␣WHERE␣userId␣=␣'$uID_int'";
12   $resultQuery = mysqli_query($connection, $loginQuery);
13   $resultCheck = mysqli_num_rows($resultQuery);
14
15   if ($resultCheck < 1) {
16    $result = "false";
17   }
18   else {
19    if ($values = mysqli_fetch_assoc($resultQuery)) {
20     $decoded_pw = base64_decode($values['pWord']);
21     if (strcmp($uPW, $decoded_pw) != 0) {
22      $result = "false";
23     }
24     else {
25      if ($values['referenceCode'] != 10 && $values['referenceCode'] != 0) {
26       $result = "no-patient";
27      }
28      else {
```

```
29      $result = "true";
30     }
31    }
32   }
33  }
34 }
35 else {
36 $result = "empty";
37 }
38
39 echo $result;
40 ?>
```

The PHP code echoes three possible messages that trigger different responses in the Android-application accompanied with TOAST notifications which are window elements used to display brief messages during a few seconds. If the echoed message is positive the application will display a TOAST notification and transition to a different activity. Besides the echoed messages, the JAVA code in itself has different triggers related to a errors in connection, formally designed as exceptions. The following snippet presents the function that receives either the echoed messages or the exceptions accompanied by a TOAST notification.

```
1  ...
2  @Override
3  protected void onPostExecute(String result) {
4   progressDialog.dismiss();
5
6   if (result.equalsIgnoreCase("true")) {
7    Toast.makeText(LoginWindowActivity.this,"Bem-Vindo!", Toast.LENGTH_LONG).
       ↪ show();
8
9    Intent intent = new Intent(LoginWindowActivity.this, HomePage.class);
10   intent.putExtra("sessionIDhome", userID.getText().toString());
11   startActivity(intent);
12   LoginWindowActivity.this.finish();
13  }
14  else if (result.equalsIgnoreCase("false")) {
15  Toast.makeText(LoginWindowActivity.this,"ID/Palavra-Chave␣Invalidos␣ou␣
       ↪ Vazios.\nTente␣de␣Novo!", Toast.LENGTH_LONG).show();
16  }
17  else if (result.equalsIgnoreCase("Exception") || result.equalsIgnoreCase("
       ↪ Error")) {
18  Toast.makeText(LoginWindowActivity.this,"Problema␣de␣Ligacao.", Toast.
       ↪ LENGTH_LONG).show();
19  }
20  else if (result.equalsIgnoreCase("no-patient")) {
```

```
21  Toast.makeText(LoginWindowActivity.this,"Aplicacao␣Apenas␣para␣Pacientes.\
        ↪ nUtilize␣a␣plataforma␣Web.", Toast.LENGTH_LONG).show();
22  }
23 }
24 ...
```

If the operation is successful the application will transition to another window, which in code terms are defined as activities. These transitions or other kinds of operations, use *Intents*[36]. In the context of this application two kinds of intents are used. As already referenced, to transition between windows, upon the validation of certain conditions like the pressing of a button, shown in the code snippet bellow, or to deploy timed notifications. Both this cases were implemented using the available documentation[36] [37] modifying the code in order to work with the application's context.

5 Deployment and Testing

5.1 Introduction

The focus of this chapter will be to present and make a very brief discussion about the deployment and testing process of this project that will serve as a basis to the next two chapters 6 and 7. Both the deployment of versions and testing must ensure, at minimum, that this is a functional and valid software application. Before deploying and testing it a name was needed. The proposed name for the Application both Web and Android was *Adiutor* which is Latin for *Assistant*. It seemed suited since the main goal of this software application is to provide assistant for both health practitioners and recipients.

5.2 Deploying Versions

5.2.1 Web

The notion of deployment regarding the Web-Application is mostly based on compatibility with all the browsers since there is no installation needed per se. The storing of files in the server and its access serve the purpose of an installation so the only thing that can be done is checking the compatibility of all the functionalities with all the main existing browsers. Out of them, the Web-application is indeed compatible and presents no performance issues that aren't related to the Internet connection.

5.2.2 Android

The deployment of a Android application can be done in two main ways. The release of an .apk file that is unsigned and unverified by Google's Play Store that can only exist in websites, clouds, etc., or uploading the .apk file to the Google Play Store. At this current moment in time and to allow a small testing phase, the first option was the one taken. Besides the testing motive, since this project will be property of the faculty, deciding if the application is launched or not to Google's Play Store must be done under their approval.

5.3 Testing

Testing software allows the developer to catch the commonly known bugs and solve them before sending the application to operate in the real world. For that purpose there are three possible phases of testing that can occur, Alpha, Beta and Unit.

Alpha testing is a phase where tests are done in a small controlled environment in order to identify early defects that the software might have. The aim is to carry out the tasks that a typical user might perform. On the other hand beta testing consists of allowing a limited number of end-users to try out the application in order to get constructive feedback related to the software. It will reduce the risk of failure and is considered to be the final test before sending the software to customers so its important to have as much feedback possible. Lastly unit testing is a kind of test in which individual units of the source code are tested, i.e., all the functions and methods that make up a file are tested separately to verify if they fit their purpose in the file and therefore in the project.

Both Web and Android applications have gone through Alpha and Beta phases, with the Android application as the main target of the tests. Unit testing in it's definition didn't occur, but one can argue that during both Alpha and Beta phases, unit testing happened.

During Beta testing, a small group, of less than ten people, were able to test the application in full being either the Web or Android applications, meaning that people of this group acted as either a patient or as a doctor/manager. This phase was when more bugs and recommendations were gathered regarding mostly details like action confirmation pop-up messages, notification display and other small details in the Web application and window transitioning, layout specification for compatibility with different screen sizes and once again, other small details in terms of the Android application. During an Alpha testing phase it's much more difficult to catch this bugs and think about minor details. Despite that, isolated tests were performed regarding actions like adding, editing and removing users, verifying if all the links worked properly, and if the outcome of those actions met the expected behavior. In a way, this could resemble unit testing since the applications were tested piece-by-piece in order to avoid fatal errors that could compromise Beta testing.

Despite both being important, the fact that with Beta testing a person unknown to the project, that doesn't know workaround and what the application is supposed to do is much more keen on finding issues than the developer itself.

5.3.1 Issues and Fixes

A small number of issues that aren't necessarily software related were founded and will be presented in table 3, with possible fixes to that issue which can easily be implemented as future work, being discussed in section 7.2. The Android application was tested mostly with Android's version 7 and 8 like the Xiaomi Mi5, One Plus 1, 2, X, 3T and 6, Asus Zenfone 3 and 4 in terms of real devices. Versions 5 and 6 were tested using Android Studio's built-in emulator. Both tests were successful in terms of performance and usability with the exception of some issues, presented in the table.

Issue Nº	Target	Description	Fix
1	Android	Notifications don't work with Android Oreo and further (Version 8.0+) because it uses different notification algorithm that doesn't coexist with the implemented one, crashing the Application.	Develop a version of the Android Application compatible with only Android Oreo+ (e.g Android Pie, version 9) and therefore uses only the new notification algorithm.
2	Android	Some of the newer smartphones, especially with a 18:9 screen display, present a out of shape layout, meaning that it isn't possibly recognized as either a large or extra large screen device.	Possibly implement layouts for Extra Extra Large Screen Sizes or layouts targeting the various existing screen resolutions and densities.

3	Web	While using a smartphone browser to access the application, some of the layout parts appear mostly deformed.	Implement a Website version that is compatible with smartphone versions and enable the Website to automatically detect if it must present computer or smartphone version.

Table 3: Table with Known Issues and Possible Fixes

This table serves also as suggestions for future work presented in section 7.2, but since they are considered as fixes, they aren't classified as a pure future work possibility.

6 Results and Evaluation

This chapter documents the key results that have emerged as a result of the software's development stage. Naturally, having reaching the project's conclusion, the main result is the creation of a functional and ready to use mobile health application, which was the main goal behind the practical part of this dissertation, with this document being the theoretical part of it.

6.1 Requirement Verification

The requirement verification process aims to ensure that an application conforms to its original specification, presented by table 2. Luckily all the requirements that were set up in that table were also met, meaning that the application fulfilled, at least in theory, the goals it was set out to accomplish. Table 4 will present the requirement's description from table 2 and a brief justification explaining why or how it was met by the presented version of this project.

Requirement Description	Justification
The software must consist of a two-way platform. One side must be focused on the patients, being it a smartphone oriented application. The second one must be focused on the doctors (and managers), being it a desktop oriented application.	Both applications were successfully implemented and tested successfully.
The patient-oriented application will be set to work on Android smartphones, with compatibility to a maximum percentage of usable devices. From data acquired, a minimal requirement compatibility with Android API 19+ (i.e, Android 4.4+)	The Android application was created using Android Studio, where a minimal API compatibility can be set. Testing, both on real devices and emulated confirmed that compatibility.
The patient oriented application, being an Android application, will be developed using JAVA language.	Confirmed through the Android project files.
The Android application will have a login system with based on an unique ID, given by the system and a user defined password, that will be encoded and can only be validated using encoding/decoding by the system.	Confirmed through project files. Base64 was the method used to encode and decode the password.
Regarding the login system, the Android application will have a "remind me" feature that saves the user ID locally.	Feature implemented using Shared Preferences method. Usage tested during Alpha and Beta testing successfully.

The login system will also have a "reset password system" in which the user can generate a new password if and only he validates personal information.	Confirmed through the Android project files. Usage tested during Alpha and Beta testing successfully.
The Android application will allow the users to send data composed by measures regarding heart function, chronic pain, body temperature and glucose levels. The measure will be accompanied by a time-stamp set by the server.	Confirmed through the Android project files. Usage tested during Alpha and Beta testing successfully while checking the database periodically.
The data will be sent to a remote database that can be accessed at all times and by multiple users, if and only they have a established network connection.	With Beta testing it was verified that multiple users could access the server via application and perform those operations at different times or at the same time.
The medical and manager oriented application will be developed in a web-page model in order to be cross-platform, avoiding being Windows exclusive.	Through Browser compatibility testing during both Alpha and Beta testing, it was possible to access the Web application with Opera, Microsoft Edge, Chrome and Apple's Safari.
The web-page platform will have a login system, in all similar to the one existing in the Android app (including a "reset password system", with a different model of action).	Confirmed through the Web-application project files and during Alpha and Beta testing.
The web-page platform will block access from managers to medical functions and the reverse will happen with medical personnel regarding managerial functions	Confirmed through the project files and during Alpha and Beta testing.
The managers will be able to add, edit and remove patients, doctors and other managers to the system.	Confirmed through the project files and during Alpha and Beta testing.
Upon adding a patient/manager/doctor, a user ID is created and sent to them via e-mail, accompanied by instructions to complete the process of registration.	Confirmed through the project files and during Alpha and Beta testing. The received e-mail is sent by the server, in most cases sent to the SPAM inbox.
The doctors will be able to access, create and edit medical records regarding every patient as well as access basic basic information regarding the patients (that is added when a manager inserts a patient into the system)	Confirmed through the project files and during Alpha and Beta testing. The received e-mail is sent by the server.

The doctors will also have access to the data sent to the remote database by the patients and will be able to display said data in a filtered-by-patient table and graph view for every category of measures.	Confirmed through the Web-app project files. Usage tested during Alpha and Beta testing successfully.
The web-page platform will be developed using HTML, PHP and CSS coding language.	Confirmed through the project files.
Both applications of the platform should be user friendly and easy to learn without the need to check for manuals or other tools that aren't direct and in the moment learning.	Feedback from Beta testing didn't report problems regarding the usage of both applications. It was considered simple and easy to use.
Both applications of the platform should be in Portuguese.	Confirmed through the project files.

Table 4: Table of Requirements Met with Justification

6.2 Evaluation

With the information gathered from the testing phase of this projected and the requirements verification table presented before it's possible to evaluate the current state of this software application.

Testing it enable the possibility to fix minor bugs that are bound to happen in any software development project. Having fixed those errors, both the Web and Android applications presented good results in terms of performance. Starting with the Android application it presents, in the current version, an average size of 4,42Mb after installation. This is due to the fact that not many functionalities are implemented, which isn't a bad thing, and those who are, regard mostly to interface usage and communication with the server. In terms of size, the Web-application is much bigger, but that's also a reflection of the difference in complexity between both applications. With testing, mostly Alpha, it was possible to use the application without reporting performance errors, but one can argue that it can be due to a good Internet connection. Regardless, only design suggestions where made during Beta testing, proving that the application works correctly.

In short, a positive evaluation of this software can be made also affirming that it is ready to be tested with a bigger group sample, in a real world context, provided it is accompanied carefully.

7 Conclusion

In this conclusion chapter a review of all the developed work and reflect about the entire project, both practical and theoretical. The main goal of this dissertation in terms of a project was to develop a mobile health application that could be used by hospitals and other facilities that execute medical follow up of patients and require to be updated about the patient's hearth function, chronic pain, glucose levels and body temperature, at least with this software's version. After reviewing both the project and this document some future work suggestions will be exposed and discussed.

In order to access the project's files, a link to for a cloud storage service is provided here. The project has also been uploaded to Github but at current time is set as private, condition that can be changed in the future.

7.1 Work Review

Calling back to figure 1, this project had many stages that allowed this dissertation's software application to be developed. This project started with a general research where it was possible to review academic papers and other documents regarding Mobile Health and Software Development which contemplated an analysis of existing solutions. From that research, several considerations regarding mostly security issues and ideas that could be implemented were taken and proved valuable to the project.

After this research a process of requirements capture took place and from it, a good amount of functionalities where defined and set as objectives that this project would have to meet in order to be successful. Luckily those requirements where met completely and some other minor functionalities where implemented, mostly coming from the feedback received during the Beta testing phase of this project.

Coming back to the projects work-flow, after setting the requirements of this project, a Interface Design took place in order to establish the foundations and interactions that this project would have in both the Android and Web applications. This was a very important stage since one of the requirements explicitly stated that the interface should be user friendly and simple. Therefore, a minimalist design was implemented in order to close gaps between the user and background work that both the applications did. Being satisfied with the overall look of the project, it then proceeded to the background work implementation.

In this stage, all the background components of this project were implemented, from the database's structure and existing relationships between the data tables, the Android-server connection on the patient-oriented application and all the server-browser operations that the Web-application made. This three main components were the main focus of this project since the success and failure depended of their implementation. Luckily, extensive documentation regarding standardized functions and methods is available online, which were modified to fit the context of this project with success, being able to present a fully working platform that could be tested.

Testing this project, although using a small group of people, proved to be effective in order to catch and solve software bugs related to both design and work features, giving good indications overall that this project is able to be tested by a bigger group of people expecting good results.

The overall feeling towards this project is that it was successful in it's implementation and that it could will possibly be a good first step towards the usage of m-health applications

in our country's health care system.

7.2 Future Work

Upon a final analysis to the software project some future work implementations can be suggested in order to expand the dimension and usability of this application. Therefore a number of suggestions will be given in this sub-section that aren't related to the fix suggestions that are shown in table 3, which are considered as priorities.

1. **More Security Measures**

   This is a very important subject to note. Despite taking upon consideration the literature's review findings, there is no such thing like "too many security measures". In fact, one key measure that must be implemented is the use of a secure website, i.e., the server where this application is currently projected doesn't offer a Secure Socket Layer, SSL [39] [40], which is considered as a standard for sensitive data security, increasing the Web-application's trustfulness levels.

   This layer is easily attained while setting a domain for the Website which this application MUST DO in order to be a viable and secure application.

2. **Port the Android Application to iPhones**

   If this platform goes out to public usage, it would be a good idea to port the Android application to be used by iPhone users, extending it's reach usage to all the smartphone users in the country.

3. **More Functionalities 1: Data Exporting**

   While reviewing existing solutions, a common functionality was the ability to export data in formats that allow them to be sent via E-mail or be printed. This could turn out to be a useful functionality to be implemented, allowing doctors to print the patient's health records to the patients file or giving it to the patient. It can be argued that it goes against the purpose of electronic health records, but some patients might be more "old-school" than others and require a piece of paper with the information.

4. **More Functionalities 2: Doctor and Patient Chat**

   A promising functionality that can be implemented regards the ability to create a communication window between doctors and patients that don't require an appointment. This could help reduce the size of waiting lists drastically but requires caution while setting up. To avoid having doctors inboxes filled with spam, patients could only be allowed to send one message, with a limited set of words and could only send another after the doctor responds to that patient.

   This way a some sort of blockage would be set avoiding an overflow that doctors couldn't handle.

5. **More Functionalities 3: Patient Inter-Connection Chat**

   Similar to the previous discussed point, this functionality would work as a forum where, for example, patients with chronic pain would be able to share experiences, thoughts and stories regarding their condition. It would be interesting to create this bond between patients while helping them in the process, even if not directly.

6. **More Functionalities 4: Checking Measure History**

   A functionality that was considered for this version of the software but due to time constraints wasn't able to be implemented, will give patient's the ability to, like the doctors, view their measure's history both in table and graphic views, allowing them to better accompany their progress. This functionality could also be merged with the exporting functionalities allowing patients to show their data to other doctors that aren't in the system.

8 Bibliography

[1] Top 50 Countries/Markets by Smartphone Users and Penetration: https://newzoo.com/insights/rankings/top-50-countries-by-smartphone-penetration-and-users/

[2] PORDATA, "Agregados domésticos privados com computador, com ligação à Internet e com ligação à Internet através de banda larga(%):" https://www.pordata.pt/Municipios/Agregados+domésticos+privados+com+computador++com+ligação+à+Internet+e+com+ligação+à+Internet+através+de+banda+larga+(percentagem)-797

[3] C. Lee Ventola, MS, "Mobile Devices and Apps for Health Care Professionals: Uses and Benefits".

[4] Caroline Free, Gemma Phillips, Lambert Felix, Leandro Galli, Vikram Patel, Philip Edwards, "The effectiveness of M-health technologies for improving health and health services: a systematic review protocol".

[5] Omar El-Gayar, Ph.D., Prem Timsina, B.E., Nevine Nawar, M.D., M.P.H., Ph.D., and Wael Eid, M.D., FACP, C.D.E., "Mobile Applications for Diabetes Self-Management: Status and Potential".

[6] John Wiley & Sons A/S. Published by John Wiley & Sons Ltd., 2013, "The Promise and Peril of Mobile Health Applications for Diabetes and Endocrinology".

[7] David D. Luxton, Russell A. McCann, Nigel E. Bush, Matthew C. Mishkind, and Greg M. Reger, "mHealth for Mental Health: Integrating Smartphone Technology in Behavioral Healthcare".

[8] Sparsh Agarwal, B.Eng. (Hons) and Chiew Tong Lau, B.Eng. (Hons), M.A.Sc., Ph.D., "Remote Health Monitoring Using Mobile Phones and Web Services".

[9] Dr. Chandrashan Perera MBBS, "The Evolution of E-Health – Mobile Technology and mHealth".

[10] Maged N. Kamel Boulo, Ann C. Brewer, Chante Karimkhani, David B. Buller, Robert P. Dellavalle, "Mobile medical and health apps: state of the art, concerns, regulatory control and certification".

[11] World Health Organization, Global Observatory for e-Health series - Volume 3: "mHealth: New horizons for health through mobile technologies".

[12] Miloslava Plachkinova, Steven Andrés, Samir Chatterjee, "A Taxonomy of mHealth Apps – Security and Privacy Concerns".

[13] Rajindra Adhikari, Deborah Richards, Karen Scott, "Security and Privacy Issues Related to the Use of Mobile Health Apps".

[14] Milan Markovia, Zoran Savia, and Branko Kovacevia, "Secure Mobile Health Systems: Principles and Solutions", presented in "M-Health Emerging Mobile Health Systems", Edited by Robert S.H. Istepanian, Swamy Laxminarayan and Constantinos S. Pattichis.

[15] Borja Martínez-Pérez, Isabel de la Torre-Díez, Miguel López-Coronado, "Privacy and Security in Mobile Health Apps: A Review and Recommendations".

[16] Serviços Partilhados do Ministério da Saúde: "Privacidade da Informação no Setor da Saúde".

[17] statcounter, "Mobile Operating System Market Share Worldwide. Aug 2009-August 2018":
http://gs.statcounter.com/os-market-share/mobile/worldwide/#monthly-200908-201808

[18] 325,000 mobile health apps available in 2017 – Android now the leading mHealth platform:
https://research2guidance.com/325000-mobile-health-apps-available-in-2017/

[19] statcounter: "Mobile Operating System Market Share Portugal":
http://gs.statcounter.com/os-market-share/mobile/portugal

[20] Naked Security by SOPHOS, "2018 Malware Forecast: the onward march of Android malware":
https://nakedsecurity.sophos.com/2017/11/07/2018-malware-forecast-the-onward-march-of-android-malware/

[21] HTML 5.2 - W3C Recommendation: https://www.w3.org/TR/html52/

[22] PHP Manual: http://php.net/manual/en/

[23] James Walden, Maureen Doyle, Robert Lenhof, John Murray, "Idea: Java vs. PHP: Security Implications of Language Choice for Web Applications".

[24] CARAT - Controlo da Asma e Rinite Alérgica Teste:
https://www.new.caratnetwork.org/pt-pt/?option=com_content&view=article&id=67

[25] Software Engineering: Theory and Practice, 4th edition, Chapter 4 - "Capturing the Requirements". Shari Lawrence Pfleeger and Joanne M. Atlee. Pearson International Edition, Pearson Prentice Hall, 2010.

[26] Distribution, in percentage, of active Android OS versions around the world: https://developer.android.com/about/dashboards/

[27] Introduction to Android, Android Developers: https://developer.android.com/guide/

[28] Defining Android OS compatibility, Android Developers:
https://developer.android.com/guide/topics/manifest/uses-sdk-element

[29] Android Devs, HttpURLConnection:
https://developer.android.com/reference/java/net/HttpURLConnection

[30] S. Josefsson, "The Base16, Base32, and Base64 Data Encodings".

[31] MySQL Reference Manual, Chapter 11 Data Types:
https://dev.mysql.com/doc/refman/8.0/en/data-types.html

[32] MYSQL Improved Extension - PHP Online Manual: https://secure.php.net/manual/en/book.mysqli.php

[33] jQuery.ajax() API Documentation: http://api.jquery.com/jquery.ajax/

[34] Morris.js Graphs on Github, 2013, Olly Smith: http://morrisjs.github.io/morris.js/

[35] Android PHP Mysql Login Tutorial Using HttpURLConnection, Gururaj P Kharvi: http://androidcss.com/android/android-php-mysql-login-tutorial/

[36] Android Intents and Intents Filter, Android Developers: https://developer.android.com/guide/components/intents-filters

[37] Android Notifications, Android Developers: https://developer.android.com/guide/topics/ui/notifiers/notifications

[38] "Create and Manage Notification Channels", Android Developers: https://developer.android.com/training/notify-user/channels

[39] Kipp E.B. Hickman, "The SSL Protocol": http://www.webstart.com/jed/papers/HRM/references/ssl.html

[40] A. Freier, P. Karlton, P. Kocher, "The Secure Sockets Layer (SSL) Protocol Version 3.0".

9 Appendix

9.1 A1 - Class Diagram

Figure 16: Android Application Class Diagram

9.2 A2 - CSS Code Snippet

```
#loginBody .background {
 margin: 0 auto;
 padding: 0;
 background: url("images/background.png");
 height: 100vh;
 width: 100vw;
 background-size: auto;
 -webkit-background-size: auto;
 -moz-background-size: auto;
 -o-background-size: auto;
 background-size: auto;
}

#loginBody .login-square {
 width: 40vw;
 height: 80vh;
 background: rgba(255,255,255,0.7);
 top: 50%;
 left: 50%;
 position: absolute;
 transform: translate(-50%, -50%);
 box-sizing: border-box;
}

#loginBody .login-square h1 {
 color: #fff;
 text-shadow: -1px 0 black, 0 1px black, 1px 0 black, 0 -1px black;
 margin: 0;
 padding: 5vh 0 1vh 0;
 text-align: center;
 font-size: 30px;
 font-family: 'Lucida Sans';
}

#loginBody .login-square h2 {
 color: #fff;
 text-shadow: -1px 0 black, 0 1px black, 1px 0 black, 0 -1px black;
 margin: 0;
 padding: 0 0 10vh 0;
 text-align: center;
 font-size: 20px;
 font-family: 'Lucida Sans';
}

```

```
45 #loginBody .login-square p {
46  color: #fff;
47  text-shadow: -1px 0 black, 0 1px black, 1px 0 black, 0 -1px black;
48  margin: 0;
49  padding: 0 0 1vh 0;
50  font-size: 25px;
51  font-family: 'Lucida Sans';
52  text-align: center;
53 }
54
55 #loginBody .login-square input {
56  border: none;
57  float: left;
58  width: 100%;
59  height: 8vh;
60  text-align: center;
61  margin-bottom: 5vh;
62 }
63
64 #loginBody .login-square input[type="text"], input[type="password"] {
65  border-bottom: 1px;
66  outline: none;
67  height: 8vh;
68  font-family: 'Lucinda Sans';
69  font-size: 20px;
70  margin-bottom: 1vh;
71 }
72
73 #loginBody .login-square button[type="submit"] {
74  border: none;
75  outline: none;
76  background: none;
77  width: 100%;
78  height: 8vh;
79  font-size: 25px;
80  font-family: 'Lucinda Sans';
81  color: #fff;
82  text-shadow: -1px 0 black, 0 1px black, 1px 0 black, 0 -1px black;
83  margin-top: 1vh;
84  margin-bottom: 1vh;
85 }
86
87 #loginBody .login-square button[type="submit"]:hover {
88  cursor: pointer;
89  border: none;
90  outline: none;
```

```
 background: none;
 width: 100%;
 font-size: 25px;
 font-family: 'Lucinda Sans';
 color: #ccc;
 text-shadow: -1px 0 black, 0 1px black, 1px 0 black, 0 -1px black;
 margin-top: 1vh;
 margin-bottom: 1vh;
}

#loginBody .login-square h3 {
 width: 100%;
 padding: 0 0 1vh;
 text-decoration: none;
 text-align: center;
 font-size: 20px;
 font-family: 'Lucinda Sans';
 color: #ff3333;
 text-shadow: -1px 0 black, 0 1px black, 1px 0 black, 0 -1px black;
 margin-bottom: 2vh;
}

#loginBody .login-square a {
 width: 100%;
 text-decoration: none;
 text-align: center;
 font-size: 17px;
 font-family: 'Lucinda Sans';
 color: #fff;
 text-shadow: -1px 0 black, 0 1px black, 1px 0 black, 0 -1px black;
 margin-left: 32.5%;
}

#loginBody .login-square a:hover {
 width: 100%;
 text-decoration: none;
 text-align: center;
 font-size: 17px;
 font-family: 'Lucinda Sans';
 color: #ccc;
 text-shadow: -1px 0 black, 0 1px black, 1px 0 black, 0 -1px black;
 margin-left: 32.5%;
}
```