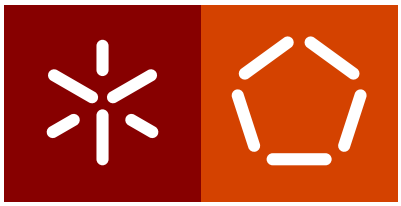
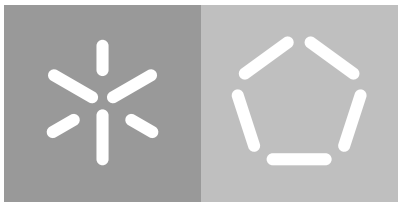


**Universidade do Minho**
Escola de Engenharia
Departamento de Informática

Paulo Alexandre Gonçalves Pacheco

# Self-service Kiosk-based Anamnesis System for Emergency Departments

January 2021



**Universidade do Minho**
Escola de Engenharia
Departamento de Informática

Paulo Alexandre Gonçalves Pacheco

# Self-service Kiosk-based Anamnesis System for Emergency Departments

Master dissertation
Master Degree in Computer Science

Dissertation supervised by
**Pedro Rangel Henriques**
**Nuno Feixa Rodrigues**

January 2021

## AUTHOR COPYRIGHTS AND TERMS OF USAGE BY THIRD PARTIES

## STATEMENT OF INTEGRITY

I hereby declare having conducted this academic work with integrity. I confirm that I have not used plagiarism or any form of undue use of information or falsification of results along the process leading to its elaboration.

I further declare that I have fully acknowledged the Code of Ethical Conduct of the University of Minho.

Paulo Pacheco

_____

# ACKNOWLEDGEMENTS

I am deeply grateful to Professor Nuno Feixa Rodrigues, for the guidance, dedication and time provided, for all the constructive and fundamental criticisms for the achievement of this dissertation project. Your insightful feedback pushed me to sharpen my thinking.
I want to express my deep gratitude to Professor Pedro Rangel Henriques, for his amazing sense of respect when helping me with the editing of the dissertation. I also thank Professor Eva Oliveira for all the support, professionalism and dedication given throughout this dissertation.

I would like to thank Maria Eduarda for her contribution to the project. A warm word for my colleague, flatmate, and great friend Jacinta Santos, who have always been a major source of support when things would get a bit discouraging.

I must express my very profound gratitude to my parents, Bernardino and Maria, for their sacrifices for educating and preparing me for my future, and for providing me with unfailing support and continuous encouragement throughout my years of study. I have to thank my girlfriend, Ana, for all the affection and support and for always showing how proud she is of me. The last word goes for my sister Beatriz, who has been the light of my life and who has given me the extra strength and motivation to get things done.

## ABSTRACT

Emergency departments have a higher number of visits compared to other hospital departments. Technology has played a crucial role in promoting improvements in hospital management and clinical performance. The number of visits to emergency departments has increased considerably, giving rise to crowding situations that cause several adverse effects. This situation negatively affects the provision of emergency services, impairs the quality of health care and increases the time patients wait for medical check-up. One of the leading causes contributing to the crowding is the high number of patients with low severity clinical condition. These are referred to as non-urgent or inappropriate patients, whose clinical situation should be taken care through self-care or primary health care.

It is the responsibility of the institutions to analyse and quantify the possible causes of crowding to find the best solution to mitigate the adverse effects caused. It is believed that non-urgent patients can use the time spent in the waiting room more productively, namely by using a self-service kiosk to which they can provide valuable information to facilitate and accelerate the clinical processing.

This work proposes a solution to be used in the waiting room of emergency departments, which aims to reduce the period of medical check-up. The solution uses a self-service kiosk for the patient to provide relevant clinical data that would otherwise have to be collected by the physician during the clinical observation process. In particular, the kiosk will collect vital signs, past medical history, main complaint and usual medication. This data will be processed and provided to the physician in a structured and uniform way before each medical check-up. The primary purpose of this solution is to reduce the period of patients' medical check-up and thus improve the response capacity of the emergency departments with the same resources.

During the Master's work period, an Android application was implemented for patients to enter the clinical data mentioned above, and a Web application for physicians to access it. Additionally, a data warehouse was implemented to store the data in a consolidated way to discover hidden relationships and patterns in the data. The first moment of evaluation, undertaken in a non-hospital facility, shows positive acceptability by participants, with a large majority considering the system user-friendly. Due to the pandemic, it was impossible to perform the second planned evaluation moment in a real emergency environment.



**Keywords:** Health Kiosk, Self-Service Kiosk, Emergency Department, Crawding, Primary Care

# RESUMO

Os serviços de urgência apresentam um número de visitas superior em comparação com outros serviços presentes nas instituições hospitalares. A afluência aos serviços de urgências tem vindo a aumentar consideravelmente, dando origem a situações de lotação que provocam diversos efeitos negativos nas instituições hospitalares. No geral, este fenômeno afeta negativamente a prestação dos serviços de urgência, prejudica a qualidade dos cuidados de saúde e faz aumentar o tempo que os doentes aguardam pela observação clínica na sala de espera. Uma das principais causas apontadas para o surgimento da lotação é o elevado número de doentes com condição clínica de baixa gravidade. Estes são designados como doentes não-urgentes ou inapropriados, cuja condição clínica poderia ser resolvida, idealmente, com recurso ao auto-cuidado ou a cuidados de saúde primários.

É da responsabilidade das instituições analisar e quantificar as possíveis causas de lotação, de forma a encontrar a melhor solução para atenuar os efeitos negativos provocados. Acredita-se que os doentes não-urgentes tenham a capacidade de utilizar o tempo na sala de espera de forma mais produtiva, através da utilização de um quiosque self-service. Neste sentido e aliada à tecnologia, esta dissertação contextualiza uma solução para ser utilizada na sala de espera dos serviços de urgência, visando reduzir o período de observação clínico. Esta solução vem complementar a realização do procedimento inicial efetuado pelo médico, no consultório, através do uso de um quiosque. Assim, a recolha dos sinais vitais, história médica prévia, queixa principal e medicação habitual será efetuada pelos doentes no quiosque. Estes dados vão ser fornecidos de forma estruturada e organizada ao médico antes da realização da consulta. O objetivo principal desta solução é reduzir o período de observação clínico e assim melhorar a capacidade de resposta dos serviços de urgência com os mesmos recursos hospitalares.

Durante o período da dissertação, foi implementada uma aplicação Android para os pacientes registarem os dados clínicos acima mencionados, e uma aplicação Web para os médicos acederem aos mesmos. Foi implementado também um data warehouse para a descoberta de relações e padrões escondidos nos dados. O primeiro momento de avaliação, realizado num ambiente não hospitalar, mostrou uma aceitabilidade positiva pelos participantes, com grande maioria a considerar o sistema user-friendly. Devido à pandemia, não foi possível realizar o segundo momento de avaliação planeado num serviço de urgências.

**Palavras-Reservadas:** Quiosque de saúde, Self-Service quiosque, Serviço de urgências, Lotação hospitalar, Cuidados primários

# CONTENTS

# LIST OF FIGURES

# LIST OF TABLES

# ACRONYMS

**A**

**API** Application Programming Interface.

**B**

**BP** Blood Pressure.

**D**

**DOM** Document Object Model.

**DW** Data Warehouse.

**E**

**ED** Emergency departments.

**ETL** Extraction-Transformation-Load.

**H**

**HR** Heart Rate.

**HTTP** Hypertext Transfer Protocol.

**I**

**IDE** Integrated Development Environment.

**J**

**JS** JavaScript.

**JSON** JavaScript Object Notation.

**M**

**MVC** Model-View-Controller.

**R**

**REST** Representational State Transfer.

**S**

**SMS** Short Message Service.

**SPO2** Blood Oxygen Saturation.

**U**

**UI** User Interface.

**URI** Uniform Resource Identifier.

**V**

**VDOM** Virtual DOM.

# 1

# INTRODUCTION

This document is a Master's dissertation in Informatics Engineering that reports the project chosen by the author and describes all the work done, as well as discusses the results and the main contributions of the thesis defended. The Master's project consists in a kiosk-based technological solution to mitigate the adverse effects caused by the crowding problem experienced in hospital emergency departments.

## 1.1 CONTEXT

Primary care is delivered in a variety of settings, including small or medium-sized private practices, hospital outpatient departments, community health centres, and integrated care systems. Over the years, there has been a lack of access to primary care [1, 2], leading to increased demand for *Emergency departments (ED)* as a replacement. Although hospital institutions have qualified technical staff and appropriate infrastructure, this demand has created unusual hospital crowding situations [3].

According to the American College of Emergency Physicians, crowding occurs when the identified need for emergency services exceeds available resources for patient care in the ED, hospital, or both [4]. In general, this phenomenon negatively affects the delivery of emergency services, impairing the quality of health care and, consequently, the clinical outcomes of patients. When crowding is high, the time patients wait to be seen by a physician tends to increase, as does the number of patients leaving without medical advice [5].

To prevent these consequences, an effort is needed by the management entities and health care providers of the institutions to identify and quantify the recurring causes of this problem. One of the leading reasons pointed out for the emergence of crowding in ED is the high number of patients with a low severity clinical condition, designated as non-urgent or inappropriate patients [6].

In a study conducted in fifteen countries from Europe, Asia, North America and Africa, it was shown that countries with reportedly strong primary care networks, ED crowding is still a major phenomenon, as is seen in Italy, France, and Saudi Arabia. One of the major factors for ED crowding is "inappropriate use", and the rise in ED visits is often attributed to

use of the ED for problems that could potentially be cared for in a physicians' office [6]. In Portugal, ranked number 12 in overall efficiency by the World Health Organization in 2000, 30% of visits to ED in 2001 could be inadequate [7, 8]. In 2015, a report published by the National Health Service identified a 7% increase in this percentage [9]. It is believed that these patients of low clinical severity, represent internationally between 24% and 40% of all patients [10].

With the improvements in technology, new alternatives have been created to obtain health information over the Internet, television, and telephone, among other options. Moreover, it gave the population the possibility of obtaining data about health parameters and make use of this information, without having to go to a hospital or other type of medical facility [11]. There are already several hospital institutions that resort to technology to provide medical devices that can be used in waiting rooms for assessing the patient's condition [12].

Emerging medical devices open the possibility of creating a system that can help in the collection of vital signs from patients. An instance of such a system for installation in hospitals or clinical centres are self-service kiosks [11].

Self-service kiosks are independent units that contain computer programs and provide services to patients in several domains, such as triage, face-to-face consultation replacement and disease diagnosis. They are widely portable, so they can be moved from one location to another when needed. Kiosks-based programs are primarily interactive and generally have a simplified user interface, such as a touch screen or keyboard [12]. One randomised study from Singapore conducted in 2017, over 12 months, demonstrated that by placing kiosks in waiting rooms, patients were able to use the time between registration and meeting with the physician to engage in productive tasks that aid the medical process [13].

When patients go to an emergency department, they are subjected to a triage process to identify the wristband colour that best represents their medical treatment urgency. After identification, the patient waits in the waiting room for the medical check-up. In medical check-up, the physician performs the initial procedure of collecting the patient's vital signs, usual medication and past medical history, in a process called anamnesis.

This work proposes a technological solution that aims to replace the initial procedure performed in the physician's office, with a kiosk placed in the waiting room of the emergency department. The information associated with the initial procedure, collected through the kiosk, will be made available to the physician before each medical check-up. The expectation is that the physician will no longer need to perform the initial procedure in the office or accelerate it significantly. The primary purpose of this solution is to reduce the period of the patients' medical check-up and thus improve the response capacity of the ED with the same resources. This work is part of a larger project developed by a team in collaboration with Jacinta Santos, who is in charge of human-computer interaction issues.

## 1.2 MOTIVATION

This thesis aims to develop a kiosk that will allow the medical staff to accelerate the clinical observation of the patient and, with this, improve the performance of the ED with the same amount of resources. Several hospital institutions already provide kiosks in their facilities. These have several objectives, such as assisting in triage processes, providing health information, disease prevention, disease diagnosis and face-to-face consultation replacement. The proposed kiosk differs from the previous ones in its purpose to assist patients' medical consultations by replacing the anamnesis procedure.

The implementation of this kiosk as a way to replace the initial procedures performed in the physician's office will enable a reduction in the period of clinical observation. The kiosk will bring advantages primarily to patients who use the kiosk, because they are contributing to a decrease in their waiting time. However, not only those who effectively use the kiosk have advantages. Those who do not use the kiosk will have a possible reduction of time in the waiting room, caused indirectly by those who used it. The possibility of reducing patient waiting times combined with the high turnout in ED points to adherence to the kiosk.

The kiosk's anamnesis process will free up consultation time that physicians can use to focus on other essential aspects of the patient's clinical condition, resulting in better diagnoses.

The different age ranges and digital competence of patients is a significant challenge in the design of the platform. It is, therefore, necessary to focus on the experience of using the kiosk to make it as simple as possible for the patient. One of the challenges is to create the kiosk so that it can be used independently and without prior training. Another challenge is to ensure that the gathering of biometric data from patients and their clinical record is carried out in the fastest and most intuitive way, to make the duration of a full kiosk session as short as possible.

With the continuous use of the kiosk by different patients, it will be possible to obtain a considerable amount of data about the clinical status of patients. In the face of a large amount of data, one can discover consistent patterns and systematic relationships. This will make possible to generate a predictive model which, through the data collected from each patient at the kiosk, will indicate the probable cause of their symptoms. As this platform is embedded in a kiosk, it is possible to easily move it to places where it is more needed. Although the focus is on emergency services, the kiosk can be improved for health care practice in remote areas, where access to these is scarce.

## 1.3 OBJECTIVES

Within the scope of this project, the main goal established was to build a care platform for patients in emergency departments, capable of collecting a set of biometric data, past medical history, usual medication and main complaint, process this information and provide it to the physician before the medical check-up.

This project is divided into two components. The first component consists of a mobile application that communicates with several biometric sensors. The second component will be a web application that will make the information collected in the mobile application available to physicians. The following sub-objectives can be considered:

- Planning a solution that ensures the security and confidentiality of all data involved;
- Development of a prototype of the solution, which should include the following points:
  - A system that allows the patient's identification, and collects the past medical history, usual medication and main complaint;
  - A module to collect the biometric data from sensors;
  - A set of graphical interfaces and video tutorials, whose presentation to the patient depends directly on the state of the application;
  - A storage system to store patients' health data as well as indicators to test the performance of the solution;
  - A back-end system to respond to requests made by the web application and the android application;
  - A system that allows physicians to consult the data from each patient's clinical register;
  - A data warehouse for reporting and data analysis;
- Implementation of the solution's prototype;
- Testing and evaluation of the system.

## 1.4 RESEARCH HYPOTHESIS

From the patients' and doctors' point of view, it is expected that the proposed kiosk-based platform, capable of collecting biometric data from patients and accessing their medical history data, reduces the medical check-up duration in the emergency departments.

## 1.5 DOCUMENT ORGANIZATION

This dissertation is divided into seven chapters. The second chapter, **"Background"**, provides the necessary knowledge that the reader must have to understand the proposed solution. The third chapter, **"Patient Care Kiosks"**, constitutes the state of the art of kiosks for patient care in various domains. The planning of the entire solution and the architecture of the system is carried out in the fourth chapter, **"Patient Care Platform"**. This chapter will cover issues associated with the identification of system requirements and database model.

The fifth chapter, **"System Implementation"**, describes the development of system implementation based on the modelling done in the previous chapter. The sixth chapter, **"Evaluation"**, describes the evaluation and analysis of the system. It is checked whether the solution is suitable for the purpose for which it was built. Finally, in the **"Conclusion"** chapter, the results obtained are discussed, the future work identified, and conclusions are drawn about the whole solution.

# 2

# BACKGROUND

This chapter is devoted to present the background knowledge necessary to understand the problem described in the last chapter and to appraise the solution that will be proposed and implemented. The basilar concepts and technologies that will be referred along the document will be introduced so that the dissertation is as self-contained as possible.

## 2.1 WEB SERVICES

Web Service is a software service used to communicate between two devices on a network, which implements and provides a set of services.

"A Web service is a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-process able format (specifically WSDL). Other systems interact with the Web service in a manner prescribed by its description using SOAP-messages, typically conveyed using HTTP with an XML serialization in conjunction with other Web-related standards. [14] "

Every Web service provides a standard protocol/format for communication, allowing multiple applications to send data to and receive data from the Web service regardless of the programming languages used in those applications. The communication between an application and a Web service happens through an intermediate language, a data transmission and representation format understood by both, such as XML, JSON, CSV, etc. A transport protocol is also used, such as HTTP, SMTP or FTP, but *Hypertext Transfer Protocol (HTTP)* is the most popular because it is simple, stable and widely deployed.
Therefore, for example, when an application needs to send data to a Web service, the data is first serialized into an intermediate language, and then sent to the specific Web service over the Internet according to a transport protocol. When the Web service receives the request, the data is interpreted and deserialized into the Web service's language, the necessary operations are performed, and a reply is sent to the application. This standardized way of communication allows interoperability between different applications.

There are various communication protocols, but the most prominent's are the SOAP protocol and the *Representational State Transfer (REST)* protocol. The first one uses XML as the data representation format and, generally, HTTP to transport the data. The second is a more recent communication protocol that emerged to simplify access to Web services. It also uses the HTTP protocol and allows the use of several data representation formats, such as JSON (one of the most used), XML, or others.

Nowadays, there are several Web services architectures, among them RPC-style Web services architecture and RESTful Web services architecture, the latter being better in scalability, coupling and performance than the first mentioned [15].

### 2.1.1 *RPC Web Services*

This type of Web service sees the server as a set of functions that can be invoked by clients. SOA (Service Oriented Architecture) is the most used architecture in RPC's (Remote Procedure Call), and has as the main goal to abstract business services from the application, defining service contracts, and registering service into services repository so as to be queried and shared by the client. The XML-RPC and SOAP communication protocols are the most frequently used in this type of Web services. These Web services are able to meet the need of small-or-medium-scale interoperation and data sharing, but for larger-scale applications RPC-based Web services lose in scalability, complexity and performance. REST Web Services can solve these problems [15].

### 2.1.2 *REST Web Services*

REST is a hybrid style of Web services architecture that defines a set of architectural constraints to be used for creating Web services. Web services built following the REST architectural style are known as RESTful Web services. Understanding the REST architecture requires an understanding of the definition of resource, representation, and state.
A resource can be anything, a physical object or an abstract concept, as long as it is relevant enough to be referenced as a thing in itself, but it is usually something that can be stored in a computer as a stream of bits. A representation is any useful information about the state of a resource, which can have many different representations. In REST there are two types of states. One corresponds to information about a resource, resource state, and the other to information about the path the client went through the application, application state [15].

As mentioned above, REST presents a set of architectural constraints that restrict how a server can process and respond to client requests. If any service violates these restrictions, it cannot strictly be referred to as RESTful. These restrictions are described below.

- **Uniform Interface**: There must be a uniform way of interacting with a given server, regardless of the device or application type. Everything has to be a resource, and each has to be identified through only one *Uniform Resource Identifier (URI)*.
- **Client-server**: The client application and the application server must be able to evolve separately, without depending on each other. A client must know only the URIs in order to interact with the server.
- **Stateless**: The server should not store anything regarding the last HTTP request a client made. The necessary state to handle the request is contained within the request itself. Each request is treated as if it were a new request. Because the server does not need to maintain, update or communicate the session status, statelessness enables greater availability.
- **Cacheable**: Each server response should include information on whether or not the sent data can be cached, and if so how long it can be stored on the client-side. If a data request is made and it has been cached, then there is no need to send the request back to the server. Cacheability allows to partially or completely eliminate some client-server interactions, thus improving scalability, performance and resource efficiency.
- **Layered system**: The existence of multiple layers between the client and the end server should not affect the request or the response. A client cannot know whether it is connected directly to the server or an intermediary along the way. The existence of servers as intermediaries can improve system availability by allowing load-balancing and shared caches.
- **Code on demand[optional]**: A server response may contain executable code that the client can execute, e.g., JavaScript within an HTML response.

REST uses HTTP methods/verbs to obtain or change data: GET, to obtain one or more resources and can cache the information received; POST, to create a resource that is sent in a request; PUT, to update an already existing resource on the server; DELETE, to delete an already existing resource on the server. The verbs GET, PUT and DELETE entails sending the resource identification in the URI request. The figure 1 illustrates the RESTful Web service architecture. The client interacts with the server through a uniform interface based on HTTP verbs.

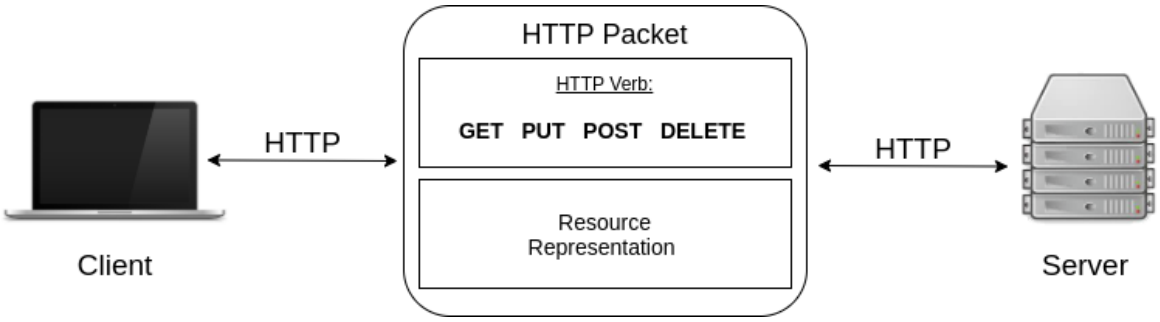



Figure 1: RESTful Web services architecture, adopted from [15].

One of the great advantages of REST is its performance. This performance is mainly due to its simplicity, but also its cacheability. As it is based on already existing web standards it does not require additional implementations, avoiding the dependency on a platform. Thus, by using the HTTP protocol in the data exchange, the client and the server do not need to parse the HTTP messages, reducing the processing load on both. Cacheability has a positive impact on performance, as it avoids some unnecessary client-server interactions. As everything is a resource and each resource is uniquely identified by a URI, it is easier to hide a resource by simply not revealing the URI. The use of HTTP verbs allows defining different access permissions to resources. In case a resource is read-only, it is only necessary to reveal the GET method to the client. Thus, the difficulty of implementing security is reduced [15].

## 2.2 JSON

*JavaScript Object Notation (JSON)* is a data-interchange format that stores information in a structured, organized and logical manner. It is easy for humans to read and write, and easy for machines to parse and generate. It has the advantage of being completely independent of language. These properties of JSON make it an ideal format for communication between applications [16].
A JSON file consists of collections of name/value pairs (realized as an object) and ordered lists of values (realized as an array) which are universal structures used by most languages. Figure 2 presents a JSON object, and Figure 3 exemplifies a JSON object's array.

```
{
    "_id": 1,
    "bandName": "Gojira",
    "origin": "France",
    "genres": ["Progressive metal", "Groove metal"],
    "formedIn": 1996,
    "active": true,
    "studioAlbums": ["Terra Incognita", "The Link",
                     "From Mars to Sirius"]

}
```

Figure 2: JSON object

A JSON object is an unordered set of name/value pairs. This object is enclosed between curly braces, and the name/value pairs are separated by commas. In each name/value pair, each name is followed by ":".

```
"bands": [
    {
         "_id": 1,
         "bandName": "Gojira",
         "origin": "France",
         "genres": ["Progressive metal", "Groove metal"],
         "formedIn": 1996,
         "active": true,
         "studioAlbums": ["Terra Incognita", "The Link",
                          "From Mars to Sirius"]
    },
    {
        "_id": 2,
         "bandName": "Draconian",
         "origin": "Sweden",
         "genres": ["Doom metal", "Gothic metal"],
         "formedIn": 1994,
         "active": true,
         "studioAlbums": ["Arcane Rain Fell", "Sovran" ],
         "labels": "Napalm Records"
    }
]
```

Figure 3: JSON array

A JSON array is an ordered collection of objects, which are separated from each other by commas. The objects in a JSON array are accessed through the index of their position.

## 2.3 ANDROID OS

Android is an open source and Linux-based operating system designed for mobile devices such as smartphones, tablets, smartwatch, and also for televisions. Since 2009, the Android operating system is the most popular in the mobile market [17]. Currently, Android Studio is the official *Integrated Development Environment (IDE)* for the development of Android applications [18]. This IDE provides several tools to assist in the application development process, such as a debugger, emulator, visual layout editor, and code templates. At the time of writing this dissertation, 11.0 is the most recent version of Android OS, and 4.0.1 the most recent version of Android Studio IDE. The essential components for the operation of an Android application are described below.

### 2.3.1 *Platform Architecture*

The Android platform architecture is made up of six main components divided into five layers, as illustrated by the figure 4.

- **System Apps**: Provides a set of applications that developers can use such as e-mail, *Short Message Service (SMS)* messaging, calendars, contacts, internet browsing, and more. For example, if an application needs to send an SMS message, just call the installed SMS application.
- **Hardware Abstraction Layer (HAL)**: Provides standard interfaces that expose the device's hardware resources to the higher-level Java *Application Programming Interface (API)* Framework.
- **Java API Framework**: Provides a set of resources through APIs written in JAVA. These APIs form the building blocks necessary to create apps by simplifying the reuse of core, modular system components and services.
- **Native C/C++ Libraries**: Provides a set of libraries that support the applications. Many core Android system components and services are built from native code that require native libraries written in C and C++.
- **Android Runtime**: Allows to run several virtual machines on devices with low memory by executing DEX (Dalvik Executable) files. DEX is a bytecode format designed specially for Android that's optimized for minimal memory footprint.

- **Linux Kernel**: Kernel is the Android platform core, that manages input and output requests from the software. Linux kernel provides support for memory management, network stack , security management, and others, allowing Android to take advantage of key security features and allows device manufacturers to develop hardware drivers for a well-known kernel.

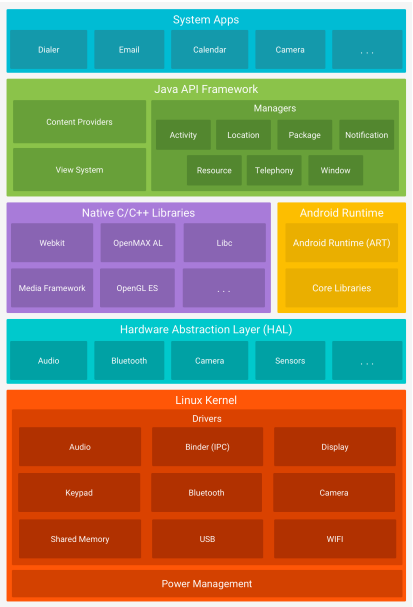



Figure 4: Android software stack, adopted from [19]

### 2.3.2 *Activity's*

*Activities* are an important application component that represents a single screen with a user interface, from which the user can interact with the application. When an application is launched, the Android system starts an activity, and as a user navigates through the application the activities are opened and closed multiple times. Activities have a predefined lifecycle and a set of lifecycle callback's methods linked to them. The figure 5 illustrates these methods. Each of these methods corresponds to a specific stage of the life cycle. Unlike programming paradigms in which apps are launched with a *main()* method, the Android system initiates an activity by invoking specific callback methods. Although it is not necessary to implement all callback methods, it is important to understand each one.

- **onCreate()**: It is called only once during the life cycle of activity and is the only method that needs to be overwritten. In this method, the layout of the activity is defined.
- **onStart()**: Makes the activity visible to the user, and is always performed right after *onCreate()* and whenever the activity returns from background to foreground through

*onRestart()*. It is the ideal method to begin drawing visual elements and running animations, for instance.

- **onResume()**: It is executed before the application is shown to the user, and it is in this state that the activity begins to interact with the user (receive input, p.e). In this state, the activity is on top of an activity stack and visible to the user. The application remains in this state until there is some action that makes it leave.
- **onPause()**: This method is called when the user leaves an activity. The activity is not destroyed, just no longer in the foreground (at the top of the activity stack). This method is generally used to stop things that consume a noticeable amount of CPU.
- **onStop()**: Called when the activity is no longer visible to the user. There are two reasons for this, the activity is destroyed, or another activity takes its place at the top of the stack.
- **onDestroy()**: This method is called before the activity is destroyed to perform any final cleanup. Destroying an activity can happen because it has been called the *finish()* method, or because the system is temporarily destroying the activity instance to save space.

Each activity has an associated file layout, where its visual structure is defined. This file contains the components that will appear on the screen and that will allow interaction with the user. Most applications contain multiple screens for different purposes, which means they comprise multiple activities. Typically, there is one activity that is defined as the main activity, which is the first screen to appear when the user launches the app. Each activity can then start another activity to perform different actions. For the activities of an app to be used it is necessary to register information about them in the app's manifest file.

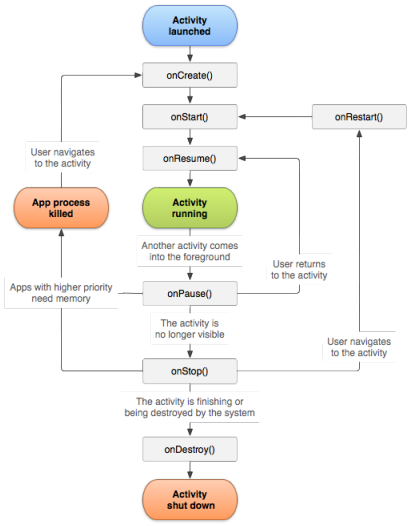



Figure 5: Activity lifecycle, adopted from [20]

### 2.3.3 *Fragments*

A *Fragment* represents a behaviour or a portion of the user interface in an activity. In a single activity, it is possible to have multiple fragments, and each of these can be reused in more than one activity. A Fragment can be seen as a "sub-activity", has its own lifecycle, has its own layout and receives its own input events. Must always be embedded in an activity, and their lifecycle is directly affected by the host activity lifecycle. The Fragment lifecycle, figure 6, contains callbacks methods similar to an activity, such as onCreate, onStart, onResume, and onDestroy.

- **onAttach()**: This method is called when a Fragment is first attached to a host activity.
- **onCreateView()**: Called to draw the Fragment user interface for the first time.
- **onDestroyView()**: It is called when the view previously created by *onCreateView()* has been detached from the fragment. This call can happen if the host activity has stopped, or the activity has removed the fragment.
- **onDetach()**: Called when the fragment is removed from its activity, when the fragment is no longer attached to its activity.

Fragments emerged primarily to support more dynamic and flexible *User Interface (UI)* designs on large screens, such as tablets. As they feature a screen much larger than that of a handset, there's more room to combine and interchange UI components. An activity can send information to the fragments it hosts, and receive information from those fragments.

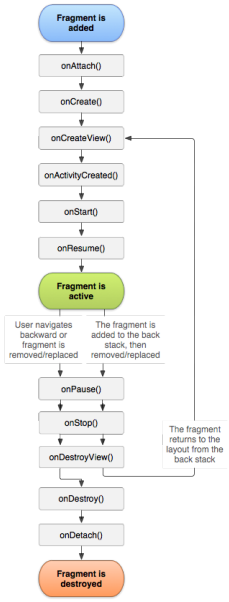


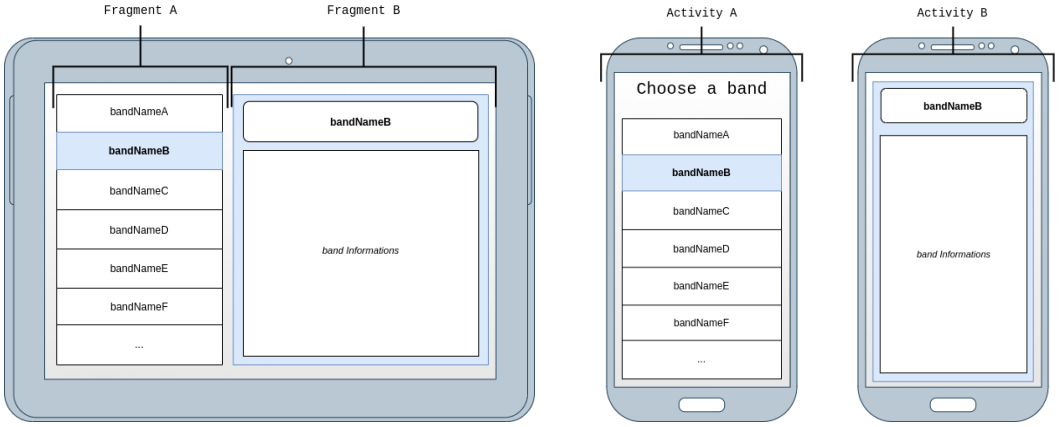

Figure 6: Fragment lifecycle, adopted from [21]

Although it is not necessary to use fragments in an application, it is nevertheless useful to know the advantages that their use can offer. For instance, let's consider an "Encyclopaedia bands" app that provides information about bands, available for tablets and handsets. More specifically, let us consider the following functionality, "A user should be able to select a band and see its information".

Since the screen size of a tablet is larger than that of a handset, it is possible to put a fragment on the left side responsible for showing a band's list and another on the right side that shows the band information. Therefore, in a single activity, users can select a band and view its information, as shown in Figure 7a.

Concerning a handset, it may be necessary to put each of these fragments into different activities when more than one cannot fit within the same activity, as illustrated in figure 7b. As they are the fragments used on the tablet, it is simply to reuse them in each of these activities. So, Activity A contains only the lists of the bands, and when the user selects a band, it starts Activity B, which includes the second fragment to show the band's information.

Therefore, a fragment must be designed as a modular and reusable activity component, so it can be used in multiple activities. This is especially important because a modular fragment allows changing the combinations of fragments for different screen sizes.




(a) Tablet (b) Handset

Figure 7: Example of the use of activities and fragments.

### 2.3.4 *Android manifest*

The Android manifest file essentially presents information needed by the operating system to run the application. In this file are declared the components of the application, such as activities, services, broadcast receivers, and content providers. An Android app must request permission to access sensitive user data (such as contacts, SMS, and location) or certain system features (such as the camera and internet access). This file, as well as the hardware and software features required by the application, declare all permissions required for the application to work properly.

## 2.4 REACT

React is a declarative, efficient, and flexible JavaScript library for building user interfaces or UI components. It allows composing complex UIs from small and isolated pieces of code called *components*. Each component is independent of each other, has its own logic, and can be reused throughout an application, thereby reducing application development time [22]. React's main features are presented below.

### 2.4.1 *Virtual DOM*

*Document Object Model (DOM)* is a programming interface that treats an XML or HTML document as a tree structure where each node is an object representing a part of the document. Web page objects are represented in a DOM document, which is manipulated whenever the state of an object changes. React keeps in memory a lightweight representation of the "real" DOM, called the *Virtual DOM (VDOM)*. When the state of an object changes, the VDOM only changes that object in the real DOM instead of updating all objects [23].

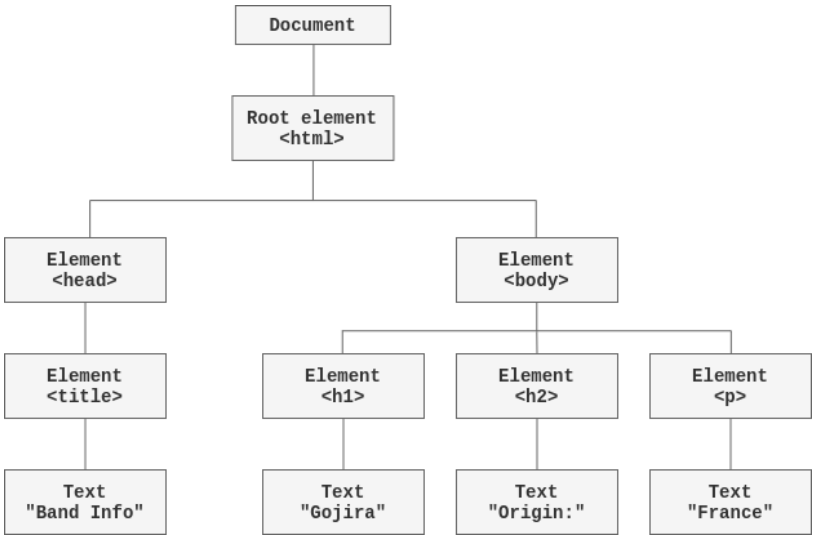



Figure 8: Webpage DOM example.

Figure 8 illustrates an example of a DOM of a webpage. Considering this figure, when the object state "TEXT:Band Info" changes in the React application, only this element will be updated in the VDOM. Then, the VDOM compares its current state with its previous state and update the real DOM with the differences only, thus avoiding updating all the objects. Using VDOM, React allows web applications to run faster.

### 2.4.2 *Components*

There are two types of components in React, stateful components and stateless components. Before diving into the differences between the two, it is necessary to clarify what "state" is in the context of a React component. State represents data that is normally to show the user or information about the component, subject to change. These changes may occur because the database from which they were obtained was updated, the user modified it, system-generated events, among many other reasons. Whenever there is a change in the status of the component, React will efficiently update and re-render the component [22].

A stateful component, defined as a *class*, can maintain internal state data and thus keep track of changing data. It also features a separate render method for returning JSX on the screen. The stateless component, defined as a *function*, does not have a state of its own. It simply prints what is passed to it via "props" (similar to function parameters) or always renders the same thing. Although a stateful component provides more features, it is a good practice to implement simple stateless components so they can be reused.
A Web application implemented in React usually contains components that need to receive data to function properly. In order to pass data from one component to another, "props" (properties) are used, a React built-in object which stores data.

Every component has a predefined set of lifecycle methods, figure 9, through which goes through. A component can be found in one of the following states, mounting, updating or unmounting, which can be thought of as the birth, growth and death of a component, respectively.

- **render()**: It is the only method needed in a class component. Handles the rendering of a component to the UI. The *render()* function should be pure with no side-effects, meaning that it does not modify component state, it will always return the same output when the same inputs are passed.
- **componentDidMount()**: This method is called as soon as the component is mounted and ready, and is commonly used to trigger data loading from a remote endpoint.
- **componentDidUpdate()**: Is is invoked immediately after updating occurs. The most common use case is updating the DOM in response to prop or state changes.
- **componentWillUnmount()**: This method is invoked immediately before a component is unmounted and destroyed. Once a component instance has been unmounted, it will never be mounted again. If any cleanup actions are needed, they should be done here, such as cancel API calls, or clear any caches in storage.

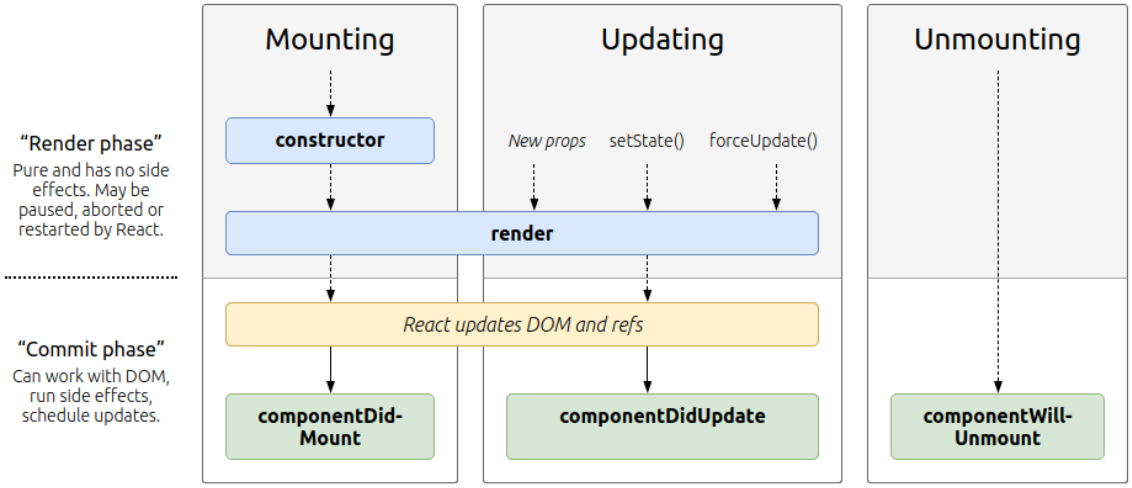
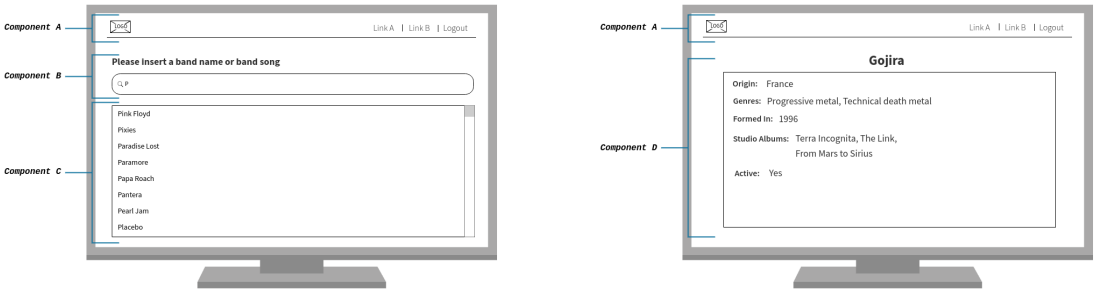



Figure 9: React Lifecycle Methods

To demonstrate the use of components let's consider again the example given previously by the "Encyclopaedia bands" app. Now, let's consider that it is intended to develop a site where users, through a browser, can consult the information of several bands. Figure 10 illustrates a possible layout of two web pages of this application, one to search for a band and the other to see its information.
Web page 10a is divided into three components, one containing the navigation bar (Component A), one for searching (Component B), and one for showing a list of options for selection (Component C). The web page 10b is divided into two fragments, component "Component A" and "Component D", which allows users to see the information of the band they have selected. The ability to reuse components, in this case, "Component A", illustrates one of the main advantages of using React.




(a) Search and select a band.

(b) Band info.

Figure 10: React components usage example

## 2.5 NODE.JS

Node.js is an open-source and cross-platform JavaScript runtime environment that executes outside a web browser. It is normally used to build network programs such as Web servers. Runs the V8 engine, the core of Google Chrome, outside of the browser, makes Node.js perform well [24]. It offers a large number of libraries and frameworks for developers to use. Before presenting the main features of Node.js, it is necessary to understand the difference between synchronous and asynchronous computing. When it is said that something is executed synchronously, it means that a task Y can only be executed when a task X is finished. If each of these two tasks runs in different threads, it means that the Y-thread blocks until the X-thread ends. In turn, when something is executed asynchronously, it means that a Y task can be executed before an X task ends. In a simplistic way, synchronous means "dependency" in some way, and asynchronous means "total independence" between tasks.
A Node.js app runs in a single process, without creating a new thread for every request. Provides a set of asynchronous I/O primitives in its standard library that prevent *JavaScript (JS)* code from blocking. Node.js functions are non-blocking, allowing different commands to run in a non-sequential way. The libraries provided by Node.js are generally written using non-blocking paradigms [24].
This set of features allows Node.js to handle numerous concurrent connections with a single server without introducing the burden of managing thread concurrency, which could be a significant source of bugs [24]. Since Node.js and most front-end technologies (React, Angular, p.e) are written in JavaScript, this allows front-end developers to write the server-side code in addition to the client-side code without the need to learn a different language.

## 2.6 DATA WAREHOUSE

*Data Warehouse (DW)* is subject-oriented, non-volatile, integrated, time variant collection of data which helps in developing strategic decisions. It is important in statistical analysis, for the extraction of relevant information, the discovery of hidden relationships and patterns in the data. A DW can potentially provide numerous benefits to an organization with quality improvement, and decision support by enabling quick and efficient access to information from legacy systems and linkage to multiple operational data sources [25]. Over time, it builds a historical record that can be valuable to data scientists and business analysts. Because of these capabilities, a data warehouse can be considered an organization's "single source of truth" [26].

In conventional DW systems, there is a set of data that is stored in Online Transaction Processing (OLTP) systems, which are periodically exported to Online Analytical Processing (OLAP) systems through an *Extraction-Transformation-Load (ETL)* process. OLTP systems are used to process the data that is generated daily by an organisation's information systems (e.g., Online banking), and OLAP systems to analyse large volumes of information stored in databases. The DW stores data from a set of data sources on which an ETL process is performed. Tools that query the data can be used on DW, from which the data analysis results, such as reports and dashboards. It could be Query tools, reporting tools, Data mining tools, and others. Figure 11 illustrates the architecture of a data warehouse.

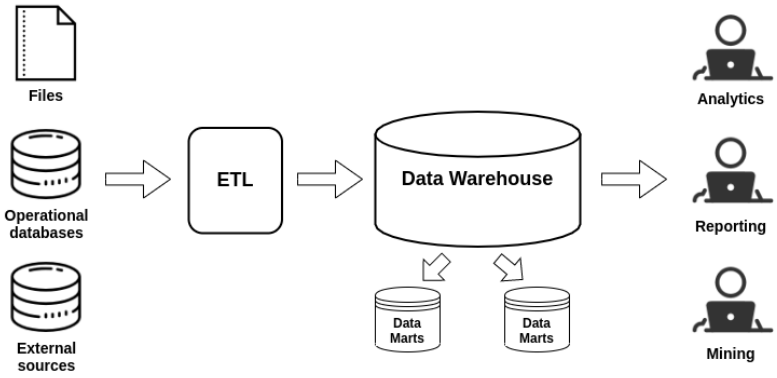



Figure 11: DW architecture, adopted from [27]

The ETL process of data sources for DW has three steps, Extraction, Transformation and Data Loading. *Extraction* means understanding, selecting and copying data from data sources into the staging area, an intermediate storage area. If data transformations occur, they are carried out in the staging area so that the performance of the source system is not degraded. The transformation step consists of the changes that are made to the data, such as cleansing the data, deleting useless fields and combining data from different sources. This step aims to improve data quality to bring more value to the business. The loading step involves loading the transformed data into DW and the physical structuring of the data [27]. The ETL process has to be repeated periodically to keep DW up-to-date.

The DW is based on a multidimensional data model that sees the data in the form of a data cube, which allows the information to be modelled and viewed in multiple dimensions. A DW is formed by dimension tables and fact tables. A *fact table* represents an item, a transaction or a business event and is used to analyze an organization's business process. The term *fact* represents a business measure, and each row in a fact table corresponds to a measurement event [27]. Figure 12 illustrates a fact table that stores the unit quantity and euros sales amount measurements for each product in each sales transaction.

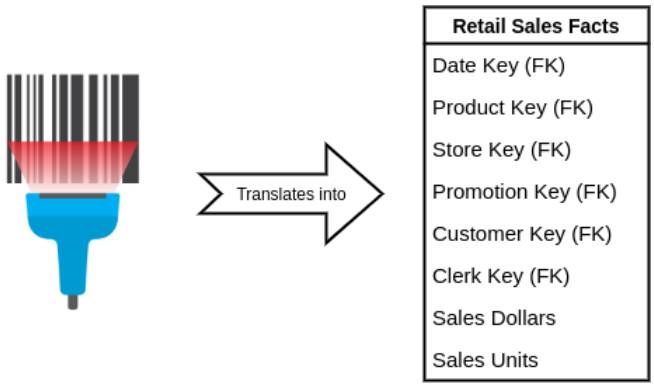



Figure 12: Example of a DW fact table, adopted from [27]

Each fact tables have two or more foreign keys that connect to the dimension tables' primary keys. *Dimensions* are the elements that participate in a fact, i.e. the possible ways of viewing data. A *dimension table* contains the textual context associated with a business process measurement event. They describe the "who, what, where, when, how, and why" associated with the event [27]. The figure 13 illustrates a dimension table that stores the product information.


Figure 13: Example of a DW dimension table, adopted from [27]

The multi-dimensional modelling of a DW can be a *star*, *snowflake* or *constellation* schema [27]. The *star* schema consists of a fact table linked to a set of dimension tables and is similar to a star-like structure. This schema is not normalized, allowing only single joins to create the relationship between the fact table and any dimension tables. Normalization means efficiently organizing the data so that all data dependencies are defined, and each table contains minimal redundancies.

The *snowflake* schema is a refining of the star scheme, in which the dimension tables are normalized into smaller dimension tables similar to a snowflake. Although it provides very low-level data redundancy, the query performance is reduced due to the need to joins multiple dimension tables to fetch the data. The *constellation* schema integrates fact tables sharing different dimension tables, forming a group of stars. Figure 14 illustrates an example of a star scheme to store data on product orders.

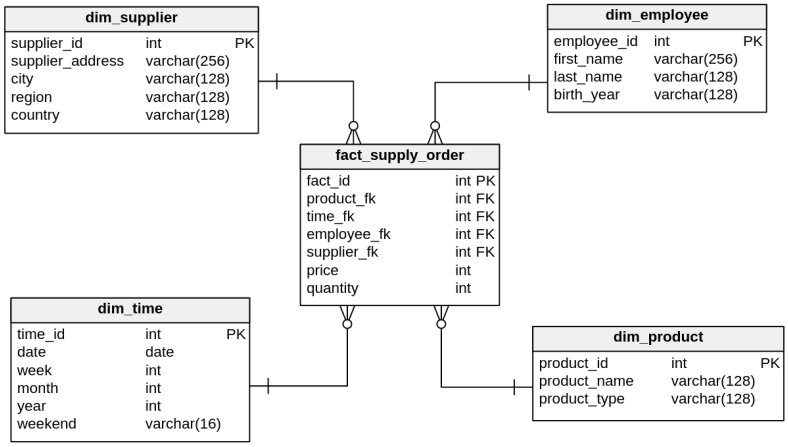



Figure 14: Star schema example

Each fact table record stores the quantity and total price of an order, linking it to a product, supplier, employee, and the instant it was made. This data warehouse permits extracting useful business information, such as which product is most ordered in the summer, which product is least marketed by each supplier and, at which time of the day more orders are placed.

# 3

# PATIENT CARE KIOSKS

In this chapter, we will collect and analyze the several studies relevant to this work, mainly those related to kiosks that allow the collection of vital signs through sensors. The research method used is described and, through the results obtained, a general summary of the most relevant studies for this thesis is made. Finally, the results of these studies will be discussed.

## 3.1 SEARCH METHOD

### 3.1.1 *Search strategy*

A literature search was conducted in PubMed, IEEE Xplore, Web of Science, Cochrane Library, ScienceDirect and Scopus, considering only articles published from January 2009 to October 2019. Search terms included 'hospital kiosks', 'public health kiosk', 'health kiosk', 'healthcare kiosk', 'systematic review kiosk', 'hospital kiosk and medication' and 'urgency kiosk'. The use of Boolean operators among the terms, in addition to the keywords, was a strategy used to obtain more accurate searches. Regarding the choice of the mentioned terms, the keywords were considered, related to kiosks in health area. Publications in newspapers and books were also considered for analysis.

### 3.1.2 *Data items*

The extraction variables taken from each study were: (1) Domain (the process that the kiosk will perform), (2) Area (location of the kiosk), (3) Country (the country where the kiosk was tested), (4) Study period (length of the kiosk test period), (5) System Objective (purpose of the kiosk), (6) Architecture (characteristics of the system architecture), (7) Population (characteristics of the population that intervened in the use of the kiosk), (8) Study parameters (metrics used to evaluate the kiosk) and (9) Results (effects of the application of the kiosk). System architecture refers to the type of system implemented, description of the biosensors and their connectivity, use and presence of touchscreen. The population variable contains

the number of participants who used the kiosk and their average age or range of ages. It also includes the presence of support in the use of the kiosk or patient referral.

## 3.2 SEARCH RESULTS

Assistance in the triage process, health diagnoses and replacement of face-to-face consultations are examples of domains where kiosks are applied [12]. The kiosks in the triage domain work as a pre-consultation tool with the objective of giving healthcare professionals more time to assess a larger number of patients [11]. Concerning diagnostic systems, they were designed to facilitate the diagnosis of common diseases by performing a set of pre-established physiologic and mental tests. Data obtained is then compared with the information available in the kiosk workstation [28, 29]. Other alternatives to the location of this application would be in public places, for population triage or control, as a form of health surveillance of elderly communities [11]. Particularly in these places with difficulties in electricity, transportation, communication and shortage of doctors, medicines and other resources, this technology has been increasingly sought after, replacing the face-to-face consultation.
After analysis and selection of articles, eight studies were obtained. Tables 1 and 2 contain a summary of these studies for later discussion.

### 3.2.1 *Study characteristics*

In all 8 studies, 4 were considered from the triage domain and 4 from face-to-face consultation replacement. Of all the associated countries, 4 belong to America, 2 to Asia, 2 to Europe and 1 to Oceania. Regarding the area, 5 studies were conducted in health areas (Hospital, Polyclinic, Clinic), and the remaining outside this context (Community room, Rest Homes, Private Apartments, University, Villages). The studies period varies between 8 weeks and 12 months. The total number of participants was 2298 and the mean age of these ranged from 12 to 94 years. Most kiosks provided assistance. The main objective of the system in the various studies is to measure vital signs and extract health information from patients according to the domain applied. The system's architecture type varies between Web App and Software. Most studies mentioned the biosensors (height, weight, blood pressure, pulse oximeter, pulse rate, glucometer and heart rate) present and their type of connectivity at kiosks. The use of touchscreen was predominant in most studies. The study parameters identified in the studies meet the evaluation of the effective of the kiosks.
Authors of the studies were contacted to obtain missing information or confirmation of data extracted for the systematic review. In [30], the age range was clarified by contact with the author, since it was not mentioned in the article.

### 3.2.2 *Risk of bias within studies*

To analyse the risk of bias, the limitations of each study were extracted. The use of assistance is a common limitation in most studies, except for studies [11, 31]. The assistance is considered a limitation and may generate unreliable usability results because, as patients receive help during the kiosk usage, they will be influenced by its ease of use. In study [32], the kiosk was tested only at peak hours, which according to the authors, ED stressors would be felt most during these times, and the kiosk's success or failure would be highlighted. The single centre design and modest sample size were the main factors contributing to the increase in the risk of bias in study [13]. In study [11], the usability tests were conducted at a Sciences Faculty, where most participants had some technical background. It was also not the first time they interacted with medical devices or a touchscreen interface, leading to the highest usability rates. Most relevant limitations in [31] are the modest sample size and the accessibility of the kiosks only at certain times. The study period in [30], is only 21 days, which may cause unfeasible results due to the short test period. The lack of information on illiteracy and population's educational level represent other limitations in most studies.

### 3.2.3 *Study results*

This review systematizes and clarifies the results of the various studies related to healthcare kiosks. The various outcomes will enable the creation of relationships between studies, offering a more concrete interpretation of the effectiveness of kiosks in health services. Studies [11, 30, 33, 34] did not take place only in hospital settings, but also in community rooms, villages, universities, rest homes and private apartments. The biosensors that use batteries as an energy source were employed in studies [11, 30, 33]. Bluetooth was used to transmit data in [11, 30, 34]. Studies [13, 11, 30, 31, 35, 33, 34] used kiosk with *Blood Pressure (BP)* sensors and some providers and clinic staff were concerned about the accuracy of measurements. Some patients were concerned to hygiene and about measuring their BP in public, others reported that are worried about abnormal values, particularly those with BPs in the prehypertension [35]. Touchscreen interfaces were used in [13, 11, 30, 33]. Elderly patients showed capacity and willingness to participate in technological interventions [31, 35, 33, 34]. Older adults with specific health issues (e.g., tremor, use of hearing aids) require customized training and assistance [34]. Poor and less educated patients were able to use the kiosks [30].

Table 1 Summary of the included studies and the intervention detail

| Authors | Domain | Area | Country | Study period | System objective | Architecture | | | | Population | | | Study parameters | Results |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | Type | Biosensors | Sensors Connectivity | Touchscreen | Sample size | Age (SD) | Assistant presence | | |
| Coyle, N., et al (2019) [32] | Triage | Hospital | Canada | 10 weeks | Self-identify and capture the arrival times of patients. Alert triage nurses arrival patients and primary complaint before triage | — | — | NA[6] | — | 898 | 53 (21.0) avg.age | Yes | Prove that ED[4]patients can use a self-check-in kiosk upon arrival and to compare time-to-first-identification with current triage system | Time-to-first- identification was 13.6 minutes (time-to-first identification was 4 for intervention patients and 9 for control patients) faster for patients who used the kiosk. Kiosk usability was 97% |
| Ng, G., et al (2018) [13] | FCR[1] | Polyclinic | Malaysia | 12 months | Measure patients physiological parameters and combines of this and their recent laboratory results to classify patients. Furthermore, also produces a result slip for the patient | Web App | BP[2] height, scale | — | Yes | 120 | — | Yes | Evaluate visit duration, patient satisfaction with the management process, health-related quality of life and the occurrence of any adverse | Patients and physicians expressed high levels of acceptance and satisfaction. Kiosk allowed more physician time to be allocated to the management of patients |
| Silva, J., et al (2017) [11] | Triage | University | Portugal | — | Measure vital signs for triage or continued monitoring | Web App | Scale, BP[2] PO[3] | Bluetooth | Yes | 74 | — | No | Assess kiosk usability, the tools developed, the results of the evaluation, the identified problems, and how we are solving those problems | The oximeter is portable. The scale needs calibration. The BP[2] monitor generates difficulties and is the device that takes longer to collect. An average of 26.1 seconds to the process of identification and average of 31.3 seconds for manually entered the data. The average time for a complete kiosk session is 283 seconds. Global acceptance was very positive |
| Soares, E., et al (2016) [30] | Triage | University's and Villages | Portugal and Brazil | 21 days | Measure vital data prior to a consultation, in the scope of a population triage, or for routinely monitoring | Web App | Scale, BP[2] PO[3] | Bluetooth | Yes | 833 | 12-89 y | Yes | Analyse the difficulties of building a simplified Health Kiosk capable of measuring BP[2] weight and PO[3]using PHDs[5] | Incorrect or difficult placement of the BP[2] cuff. Patients the poor and less educated completed the session without help. The kiosk free up human resources. A full session took around 5 min |

Information not provided by the study is represented by a dash;
[1] FCR = Face-to-face consultation replacement;
[2] BP = Blood Pressure;
[3] PO = Pulse Oximeter;
[4] ED = Emergency Department;
[5] PHDs = Personal Health Devices;
[6] NA = Not applicable

Table 2 Summary of the included studies and the intervention detail (continuation)

| Authors | Domain | Area | Country | Study period | System objective | Architecture | | | | Population | | | Study parameters | Results |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | Type | Biosensors | Sensors Connectivity | Touchscreen | Sample size | Age (SD) | Assistant presence | | |
| Bahadin, J., et al (2016) [31] | FCR[1] | Clinic | Malaysia | 2 months | Automates the management of stable patients with chronic conditions to complement face-to-face PCP[2]visits | — | BP | — | — | 95 | 61.4 (6.7) avg.age | — | Show that the kiosk could be a feasible means of delivering care for stable patients with chronic conditions and could generate cost savings for the management of patients with stable chronic disease | Kiosk was easy to use, and 96% agreed that they could use the kiosk instead of a physician. BP[3]reading was higher than that of the nurse. Reduction of 128 physician visits, saved of $5335. Patients need to spend only about 7 min at the kiosk |
| Chung, C. F., et al (2016) [35] | FCR[1] | Clinic | USA[7] | 9 months | Used to measure BP[3] | Web app | BP[3] | — | — | 152 | — | Yes | Evaluate BP[3] kiosk acceptability and usability, as well as its effects on the workflow of patient BP[3]self-measurement | Some older patients seemed to take longer to use the new technology, whereas others felt the self-service technologies were impersonal. 80% of patient thought kiosk BPs[3] were as accurate, than those taken by clinic staff. MA[6]reported that the time saved (1.5 minutes) allowed them to spend more time in clinical stuff |
| Ahn, H. S., et al (2014) [33] | FCR[1] | Hospital Rest Homes Private Apartments | New Zealand | 12 weeks | Gives helpful information to older people. Stores health information of older people for managing their health conditions | Software | BP[3] | Cable | Yes | 99 | — | Yes | Assess feasibility and acceptability in robot System for Older People in Private and Public Places | Kiosk can help older people. The participants in the private apartments were satisfied with the BP[3] measurements service. A kiosk may be more acceptable in rest-homes and hospital lounges than in private homes |
| Demiris, G., et al (2013) [34] | Triage | Community room | USA[7] | 8 weeks | This provides users secure access to their patient profile with the ability to capture relevant vital sign data into their personal health record, and to view pertinent nutritional or educational content | Software | BP[3], HR[5], glucometer, PO[4] scale | Bluetooth | — | 27 | 78-94 y | Yes | Demonstrate how informatics applications can support the assessment and visualization of older adult's wellness. Assess the acceptability and feasibility of the kiosk | Older adults are willing to participate in technology-enhanced interventions. Kiosk is "convenient," "easy and fast," and participants can "repeat the test" and "do it myself." The duration of each session corresponds to 20 minutes (more 5 min, once a week) and is held 3 times a week. The model of a community is cost-effective |

Information not provided by the study is represented by a dash;
[1] FCR = Face-to-face consultation replacement;
[2] PCP = Primary Care Physician;
[3] BP = Blood Pressure;
[4] PO = Pulse Oximeter;
[5] HR = Heart Rate;
[6] MA = Medical Assistant;
[7] USA = United States of America;

## 3.3 DISCUSSION

The studies [11, 33, 31, 13] expressed very positive kiosk acceptability and satisfaction, revealing that people are willing to use this kind of technological interventions to replace standard procedures. In [34, 30], the autonomy of participants during the use of these systems showed that the kiosks can be used practically without assistance. Studies [11, 30] present kiosks for triage or continued monitoring, that allow streamlining procedures in healthcare facilities, for expanding healthcare access to populations that otherwise might not have it, and for performing large scale population triages at very low marginal cost. This evidence shows that kiosks could help more isolated populations with lacking some essential health services.

Almost all studies use biosensors, but only in some of these the data have transmitted the data via Bluetooth [11, 30, 34]. This communication brings benefits in terms of interoperability and accessibility. However, they can cause some problems hard to solve and the only option is to skip the exam and move forward, as mentioned in [30]. The studies [11, 30, 33] present biosensors that use batteries as an energy source, and [11, 30] reported that the use of batteries could create the need for their replacement, which may indirectly interfere with the usability of the system. Biosensors with Bluetooth Low Energy (LE) technology are a possible alternative for battery replacement reduction, since it is intended to provide considerably reduced power consumption.

The quality of the measurements performed by the sensors compared to those performed by health professionals was another evaluation performed in [13, 31, 35]. Most patient respondents in [35] thought kiosk BPs were accurate, even though some reported higher values of BP. The physicians explained that this happened because patients did not have the chance to sit and wait in the reception area before measuring their BP. These instructions regarding the BP sensor should be considered at any kiosk or may be provided by clinical staff in an initial process to reduce unrealistic measurements. In [31], a discrepancy between the values of measurements made by the kiosk and by clinical professionals was described, affecting the confidence in the system. This difference in BP readings may be caused by BP measurements changing from minute to minute and can be affected by various factors. Single measurements are taken by the BP of the kiosk as opposed to repeated measurements by a nurse clinician are another possible factor. The patient's level of anxiety about new measurement protocol in contrast to the familiar face of the nurse clinician which may also account for the observed differences.

The BP monitor is the device whose result takes longer to collect, due to the cuff not being rigid. In the choice of this device, one must take into account the type of cuff, to avoid that users have to manually adapt it to their arm [11]. The incorrect or difficult placement of the blood pressure cuff, hint that a cuff-less blood pressure device would be

preferable [30]. The users with several layers of clothing have more difficulty in taking the blood pressure exam, since it involves clearing the left arm of anything that might block the circulation. The choice of a BP wrist monitor may be the solution to this problem since the measurement will be taken at the wrist and there is no need to collect clothing. Blood pressure measurement is a procedure that requires a set of several instructions, making it difficult for the user. Instructions with more compact information to be presented simultaneously with the measurement will make the procedure more supportive and avoid forgetting instructions [11]. George Miller found that people are only able to keep five to nine items in the short-term memory before they forgot or had errors [36]. The concern of the providers, patients and clinic staff and the need to obtain accurate measurements, require that in the choice of BP or another type of sensor, the accuracy of the same must be taken into account. A good accuracy or certification of the sensors provides confidence to the healthcare professionals and patients in the results obtained from the measurements [35].

In [35], patients expressed hygiene concerns, which led them not to use the kiosk. Hospital institutions must provide solutions to ensure the kiosk's hygiene. The kiosk use by different patients requires constant hygienic cleaning of the sensors, which is not always possible, and the choice of sensors that are easy to clean or without body contact with the patient should be taken into account.

The oximeter is a small and extremely portable device, but at the same time, some restrictions have to be made in order to make it safe to be used in a public location [11].

The need for the scale calibration step in the study [11], lead to users spending an excessive amount of time on the weighing, due to not being familiar with scales that have to perform this calibration. The choice of sensors that need calibration should be avoided, so that patients do not suffer this type of problem.

Some patients were confused and concerned by the prehypertension notations on the kiosk paper printout [35]. Results classification can make patients more anxious and worried, affecting their state of health, and should be avoided after each measurement.

Some patients had concerns about measuring their BP in public, in part because of the need to remove an arm from clothing or because of concerns about others seeing their BP readings [35]. The location of the kiosks in a private area, outside the field of view of other patients, makes the user more comfortable to interact with the system. The kiosks should also be placed in places that do not affect the workflow of hospital institutions.

The use of touchscreen offers direct and easy to use interfaces [13, 11, 30, 33]. This feature allows you to save space as it eliminates keyboards and mice. Touch-based kiosks are very easy to operate, even for the elderly, as their intuitive use does not require any technical knowledge about using accessories such as the mouse and keyboard, improving human-computer interaction.

Studies [31, 35, 33, 34] were successfully applied to participants with a more advanced age group, showing that elderly patients demonstrate the capacity and are willing to participate in technological interventions. Still with an initial need for care, the elderly show good learning capacity in the use of kiosks, showing that independent use of kiosks to routinely and unobtrusively assess and identify patterns of elder wellness is possible, without the need to go to hospital environments. The measurement of vital signs in private homes is more satisfying than in hospitals, public spaces or rest homes. This may be due to older adults living independently are not in an environment that monitors their blood pressure every day, so they wanted to check their blood pressure in their homes. For the development of these types of kiosks, it is necessary to take into account their size, because probably smaller kiosks will be more acceptable in terms of portability and comfort in private homes, which essentially contain small partitions [33].

Older adults with specific health issues (e.g. tremor, use of hearing aids) may experience considerable difficulties in using this equipment, which may be overcome by customized training and assistance for this type of users. The kiosk tools should be adapted to the user's preferences in order to allow a comfortable and familiarised use [34].

Although the elderly population begins to participate in technological interventions [31], some still prefer face-to-face visits rather than the use of kiosks [35]. These self-service technologies can be impersonal, causing lack of confidence with the equipment and can be solved by providing assistance or even recommending the use of this type of kiosks by health professionals.

Study [30] provides evidence that even poor and less educated patients were able to use the kiosk. The use of instructions containing text (which is also spoken), images, audio and video and the use of images for iteration with the kiosk can help patients with reading difficulties to understand how to perform each of the measurements on the kiosk. Results showed that the use of colour to discriminate between good and bad results will also help patients to know their state of health without the need for knowledge about standard normal values. A smart-card for authentication also facilitates patient identification process by reducing patient interaction with the system [30].

The implementation of the kiosk resulted in a saving of $5335 concerning to face-to-face consultations, offering a better cost-benefit option to patients [31]. Modularity which allows for the health kiosk to be easily adapted for different use cases [11], the automation of hospital procedures [30], and the application of kiosks in community environments (eliminating the need for monitoring equipment to be installed in every residence) [34] has advantages of being cost-effective, allowing cost reduction in terms of human resources and operating costs, which can be invested in other hospital resources. The cost-effectiveness of a healthcare kiosk in clinical care is a parameter to evaluate in the development of a kiosk since this analysis can also bring cost benefits to patients derived from reduced physician

visits and the flexibility of alternative health service options [13, 33].
The kiosk may be a feasible care option; however, the studies [13, 31] presents a low sample size, which contributes to limit the generalization of the results. In the test phase of a kiosk, one should have a heterogeneous and high sample size to obtain a study that provides well-founded evidence. In the study [30], the Brazilian patients do not need assistance makes the results of the usability of the kiosk more reliable.
The identification time at the self-control kiosk was 13.6 minutes faster, half the time that the standard procedure performed by health professionals would take, possibly avoiding high waiting times caused by ED crowding. Collecting the main complaint in this identification process allows nurses to prioritize triage, being able to select the patients for triage faster and more accurately [32]. The use of kiosks allowed more time for physicians to be allocated to the treatment of other patients since the system allowed performing the procedures, which would normally be performed by health professionals [13, 35]. By utilizing a kiosk for taking blood pressure and performing other activities means the physician can spend more time talking with the patient [37], and the nurses get another 1.5 minutes per patient to spend preparing documents and educational materials as well as handling telephone encounters and voicemails [35]. The participants that used citizen card took less 5.2 seconds to go through the process of identification, than while those who manually entered the data, avoiding minutes-long form-filling [11].
Kiosk usage time was not highly time-consuming, which is beneficial considering the possible use cases [11]. The time of use in [11] was slightly shorter than study [30] since users had a technological background. In relation to study [34], the time of use was the longest, due to the collection of cognitive evaluation data performed at the kiosk. In addition, the study required patients to be enrolled for a period of two months, which made it more conducive to withdrawals.

# 4

# PATIENT CARE PLATFORM

This chapter presents the conceptualised system solution. First, the different kiosk functional and non-functional requirements are presented, followed by the web platform requirements used by physicians to consult the data collected at the kiosk. Based on the identified requirements, the proposed solution architecture is described as well as its components. The conceptual database model, where the data from each clinical register is stored, is explained next.

## 4.1 SYSTEM REQUIREMENTS

In this section, the system requirements will be presented sub-divided into kiosk requirements and web platform requirements.

### 4.1.1 *Kiosk Requirements*

In order to make the scope of the kiosk more perceptible, a use case diagram was drawn. This diagram explains which actors in the system, as well as which functionalities the system should have.

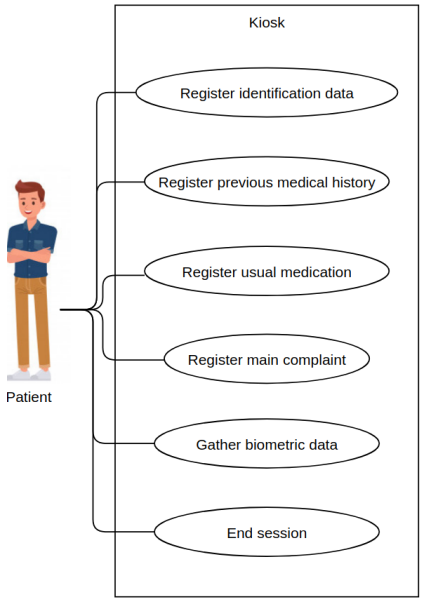



Figure 15: Kiosk use-case diagram

*Use cases*

- **Register identification data.** - A patient must register the personal information. This registration must be carried out via a webcam which allows the data to be scanned and extracted from the citizen card. From the citizen card must be extracted the name, birth date, civil id number, profile photo and nationality.
- **Register past medical history** - A patient manually registers the past medical history.
- **Register usual medication** - A patient must manually register the usual medication. For each medicine, the patient must register the dose, the administration route, the frequency and the times of the day in which it is administered.
- **Register main complaint** - A patient should manually registers the main complaint felt at the moment. The system must explore the main complaint through a set of questions concerning symptoms, relief factors and aggravating factors.
- **Gather biometric data** - A patient should collect the biometric data using sensors.
- **End session** - A patient may end the kiosk session at any time.

Based on these use cases, the several functional and non-functional requirements of the kiosk were identified. These are presented in the Appendix section A.1.

### 4.1.2 *Web Platform Requirements*

A use case diagram was designed to better present the web platform's scope.

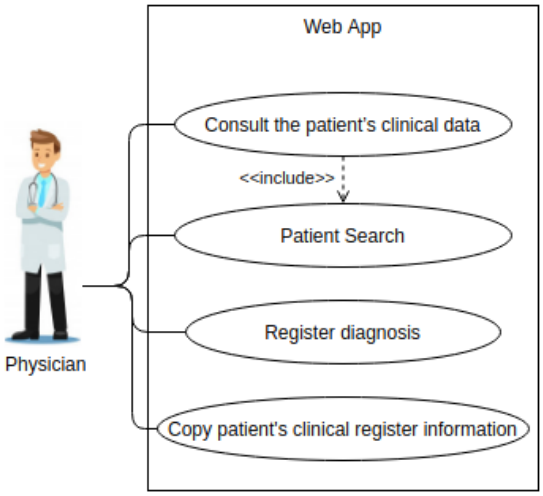



Figure 16: Web Platform use-case diagram

It should be pointed out that in figure 16 the use cases concerning the beginning and end of the session in the application were omitted, these functionalities being available to all users of the web platform.

*Use cases*

- **Consult the patient's clinical data** - A physician can consult the patient's clinical data collected at the kiosk.
- **Register diagnosis** - A physician should register the diagnosis of patients.
- **Copy patient's clinical register information** - A physician should copy all patient register information, or parts of it.
- **Patient Search** - The physician should be able to search for a specific patient.

Based on these use cases, the various functional and non-functional requirements of the web platform were identified. These are presented in the Appendix section B.1.

## 4.2 System architecture

The idealized system architecture for the solution is based on client-server basics and was developed after the system specifications were defined. The architecture consists of an Android client and a Web client, both communicating with the same back-end server system. The back-end features three web services, which perform operations on the same database, and a Firebase server. Figure 17 depicts the different components of the architecture.

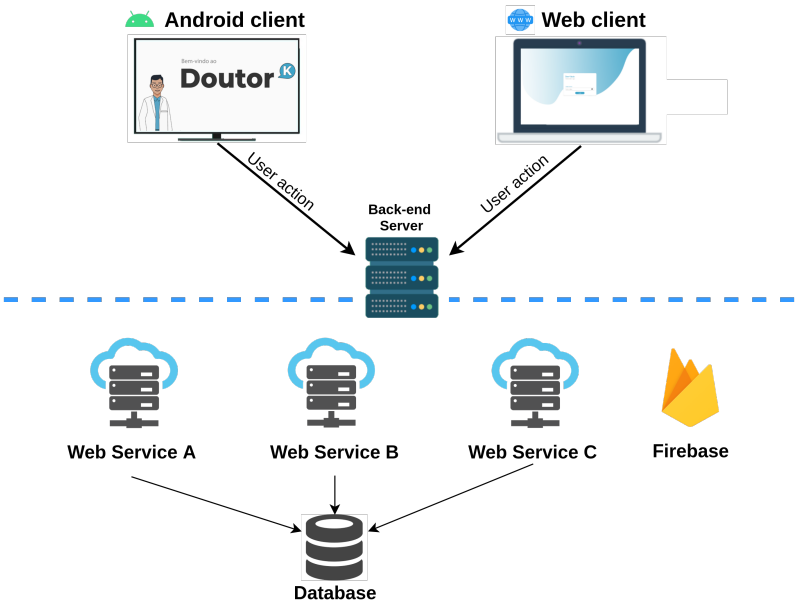



Figure 17: System architecture

Additionally, a data warehouse is set up to store patients' clinical register data in a consolidated format for the discovery of hidden relationships and patterns in the data. This data warehouse should respond to the indicators and queries defined by Jacinta Santos. The data warehouse implementation can be found in Section 5.6.

### 4.2.1 *Android client*

The Android client allows patients to interact with the application and record their clinical condition. Figure 18 illustrates the architecture of the android application through a package diagram.

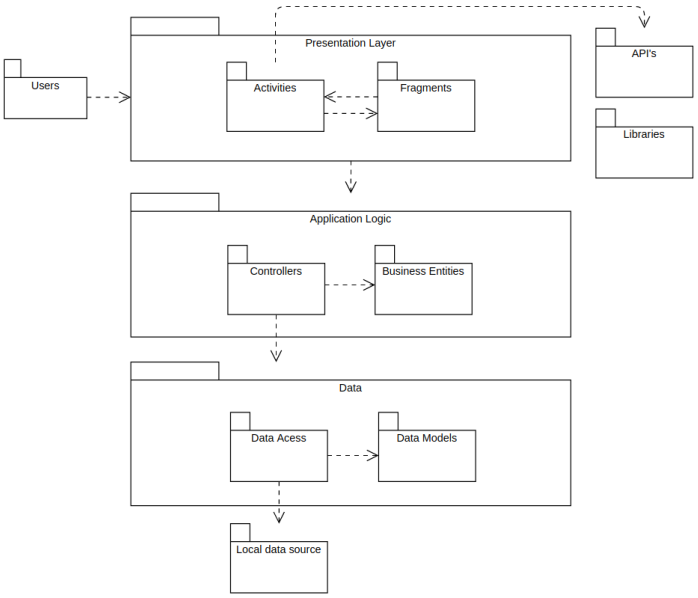



Figure 18: Android architecture

The "Presentation" layer contains all the *activities* and *fragments* of the application. The data collected through the sensors will be received and processed by the application activities and sent to the back-end server. All the logic involving access to the application database data was implemented on *controllers* in the "Application Logic" layer. The "API" component contains the several interfaces of each of the three web services used by the activities to make requests to the web services.
In the "Application Logic" layer, the component "Business entities" stores object that is either created, updated or used during the course of the application, p.e, Medicine and MainComplaint. The "Data" layer is responsible for performing data operations on the application's SQLite database (Local data source). The "Data access" component stores all the *functions* that permit interaction with the database. All data models are in the "Data models" component.
Each of these layers deals with specific aspects of an application, providing a better organization of the code, offering an easier reading or modification of the code.

### 4.2.2 *Web client*

The web application allows physicians to visualise data from patients' clinical registers. The architecture follows the Container-component pattern, where a container does data fetching and then renders it's in the corresponding sub-component.

### 4.2.3 *Web Services*

The correct performance of both clients requires that Web services are connected to the Internet to receive the clients' requests. The fact that the Web services depend on this connectivity can compromise the availability of their services by making Web and Android applications inoperable. Client programming logic will be performed to decrease Web service dependency.

The usual medication collection, past medical history and main complaint are the main features provided by the Android application. The web client will mostly perform data consultation operations. With only one Web service, it is possible to respond to the different requests made by the clients, but this can overload the Web service, causing an increase in the response time to requests.
In order to avoid this problem, three Web services were created, Web Service A and C take requests from the android client and Web Service B takes requests from the web client. Web Service A takes requests for usual medication and past medical history, Web Service C handle requests for main complaints, and Web Service B takes all requests from the web client. This way, each Web service is accountable for responding to specific client requests allowing for better system performance.

The number of users may initially be reduced, but this number may increase significantly in the future, requiring system changes. With three Web services, the system is easier to scale, and only those Web services that are overloaded with requests need to be replicated.

Figure 19 shows a package diagram with the architecture of each web service.

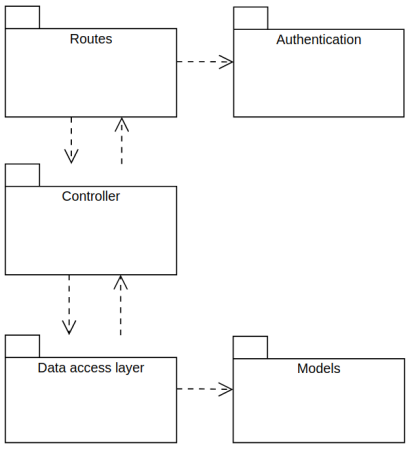



Figure 19: Web service architecture

The "Authentication" component stores the files of the authentication method that allows verifying each request to the web service. The "Routes" component is responsible for defining all the routes of the web service and assigning a controller to each one. Each route has a *middleware* that verifies the authenticity of each request before calling the corresponding controller. The "Controller" component contains controllers responsible for invokes the appropriate action to perform into the database. The "Models" component stores the database models that the "Data access layer" component uses to interact with the database Each individual component is self-contained and unaware of who might be using it, leading to low coupling. Components in a loosely coupled system can be replaced with alternative implementations that provide the same services. Each component of the architecture and its files have distinct responsibilities, enhancing the system's modularity and scalability.

### 4.2.4 *Firebase*

Cloud Storage and Realtime Database are the only Firebase services used in this solution. The Cloud storage service is used to store the patient's citizen card profile images, and the Realtime Database service to store the patient clicks' coordinates.
Cloud Storage for Firebase is a powerful, simple, and cost-effective object storage service, and lets upload and share user generated content, such as images and video. The data is stored in a Google Cloud Storage bucket, an exabyte scale object storage solution with high availability and global redundancy. Cloud Storage lets securely upload these files directly from mobile devices and web browsers, handling spotty networks with ease. If the network connection is poor, the client is able to retry the operation right where it left off, saving users time and bandwidth [38].
Realtime Database service is a database hosted in the cloud. This database allows to store and synchronize data in JSON with the Firebase NoSQL cloud database. They are synchronized in all clients, in real-time, and remain available when the application is offline [39].

### 4.2.5 *Database*

The database stores all the data from patients' clinical registers. The only components that will interact with this database are the three Web services previously presented. The database is a relational database as the data do not change frequently and consistency must be ensured whenever a new clinical register is stored.

## 4.3 DATABASE CONCEPTUAL MODEL

For designing the database, the main steps of a complete database project were followed. The Conceptual model is covered in this section, and the Physical data model in Chapter 5.

The objective of database was defined, bearing in mind the project functional and non-functional requirements: The database shall store information concerning the clinical register and diagnosis of each patient, ensuring total data integrity. Each clinical register must include health problems, usual medication, biometric data, main complaint, and relief factors, aggravation factors and symptoms of it. Indicators to evaluate the system usability and the digital skills of each patient should also be stored in the database.

Looking again at the above-mentioned database goal, the following entities were identified: Clinical Register, Biometric Data, Diagnosis, Medical History, Medicine, Patient, Main Complaint, Relief Factor, Symptom and Aggravation Factor. Considering each entity and what it symbolizes in the context of the problem, the attributes that characterize them were identified. After the entities, relationships and attributes were identified, the Entity-Relationship model was constructed. Figure 20 illustrates the graphic representation of the database conceptual model.

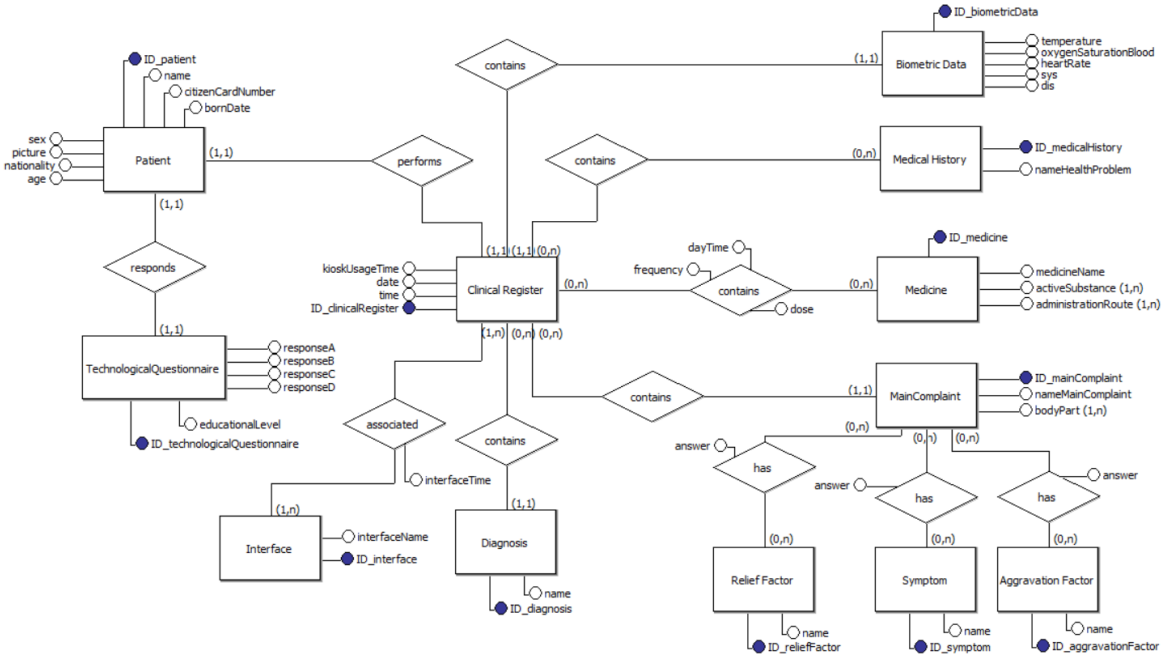



Figure 20: Conceptual model

Some attributes and entities were created to assess the usability of the kiosk. The "kioskUsageTime" and "interfaceTime" attributes represent the duration of a complete kiosk session and the time of the first visit to each interface, respectively. The "Technological Skills" entity was created to store information that permits to evaluate the digital skills of each patient. The "Technological Skills" entity stores the answers given to a set of specific questions whose analysis, performed by Jacinta Santos, will allow the system to adapt the interface to each patient's digital skills. The "picture" attribute, in the "Patient" table, stores the link to the citizen card profile picture of each patient.

# 5

# IMPLEMENTATION

This chapter describes the proposed solution's implementation process. Throughout this chapter, the fundamental aspects of implementing each component of the solution architecture are presented. Two prototypes of the Android application and a web application were implemented during this Master's work period. The implementation of the database and web services is presented first, followed by the implementation of the two prototypes and the web application and, lastly, the data warehouse.

## 5.1 KIOSK PHYSICAL COMPONENTS

Figure 21 shows a high-level kiosk architecture communicating with several biometric sensors and input devices in the physical world.

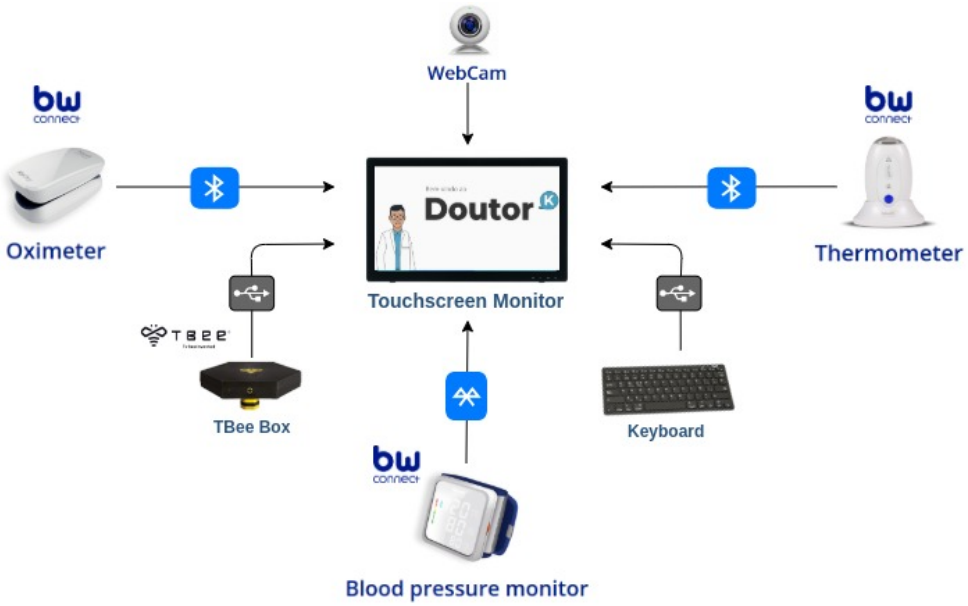



Figure 21: High-level kiosk architecture

By observing the picture, these sensors correspond to the Oximeter (MyOxy BW-OX1 Model), Thermometer (MyThermo BW-CX10 Model) and Blood pressure monitor (MyTensio Wrist BW-BW1 Model), all of which are health devices certified by European standards. These three sensors will communicate with the application via Bluetooth. The Keyboard, WebCam and TBee Box correspond to the input devices and are plugged into the touchscreen monitor via USB. Each component will be described in more detail below.
All the biometric sensor devices are rechargeable and communicate with the application via Bluetooth Low Energy, providing a completely wireless solution that facilitates installation and re-positioning of the kiosk. Bluetooth Low Energy is intended to provide considerably reduced power consumption and is used by applications that do not need to exchange large amounts of data, and can therefore run on battery power for years at a cheaper cost.

*Oximiter*

The Oximeter, measures *Blood Oxygen Saturation (SpO2)* and *Heart Rate (HR)*, and uses a lithium battery (280mAh) that provides autonomy for 400 readings. The device is fairly easy to use: the user just has to place one finger on the device and wait until the measurement is taken.

*Thermometer*

The thermometer, measures a person's temperature in 28 milliseconds, with a precision of +/- 0.2°C. This is a lithium battery (DC3.7V) device with a longevity for approximately 40000 readings. To perform an accurate measurement, the user must position the device at a distance between 1 and 2 inches from the right temporal artery and press the blue button. An important feature of this device is that it does not require body contact with the user, which greatly simplifies the hygienisation between different users.

*Blood pressure monitor*

The BP sensor can measure 3 different parameters: systolic pressure, diastolic pressure and HR. The device has a digital monitor, a wristband cuff and also uses a lithium battery (300mAh), which provides power to perform between 100 and 150 measures.
For an accurate measurement, the user must place the wristband with 1 or 2 fingers below the palm of the left hand, with the monitor button positioned on the right side. The user must press the button twice, once to turn on the sensor and again to start the measurement. Unlike standard BP monitors that require users to roll up their sleeve to insert the cuff above the elbow, this device works on the wrist, making it much easier to use, specially in public environments. Contrary to the other devices, that have a discrete measurement model, the

BP monitor provides continuous measurement, sending a stream of real-time monitoring values.

*WebCam*

The WebCam will be used to extract the information from each patient's citizen card. This device also simplifies the hygienisation, as it does not require contact of the citizen card with other physical components.

*Keyboard*

A physical keyboard was chosen since the digital keyboard overlaps the interface elements. It's a USB Wireless Keyboard and features Intelligent power saving technology, so the keyboard will automatically enter sleep mode if no operation for a period of time. The keyboard will wake up automatically when any key is pressed on the keyboard.

*Touchscreen monitor*

The PROLITE T2735MSC-B2 is a 27-inch touchscreen monitor and is easily adjustable. Interaction with the Android application is done through this monitor.

*TBee Box*

TBee Box is the device that will enable to install and run the android application that will be used on the touchscreen monitor. It's a TV box that allows to give intelligence and extra functionality to a TV, making it possible to turn an old TV into an intelligent TV without having to switch to a newer one. It gives access to Google Play so that new applications can be installed. TBee Box features the Android 7.1.1 (Nougat) operating system and a 720p HD camera. The built-in processor is the Amlogic S905-X capable of decoding H.264/H.265 to 4K video. It has 2 GB of RAM and 16GB of internal storage. It also includes a camera but does not feature autofocus. The absence of autofocus was the reason to get the WebCam previously presented.

## 5.2 DATABASE

Since the type of database is relational, the Database Management System chosen must support this feature. Having this in mind, MySQL was chosen because it is flexible and scalable, provides high performance and availability, supports transactions and offers security features. Figure 22 shows the MySQL database schema.

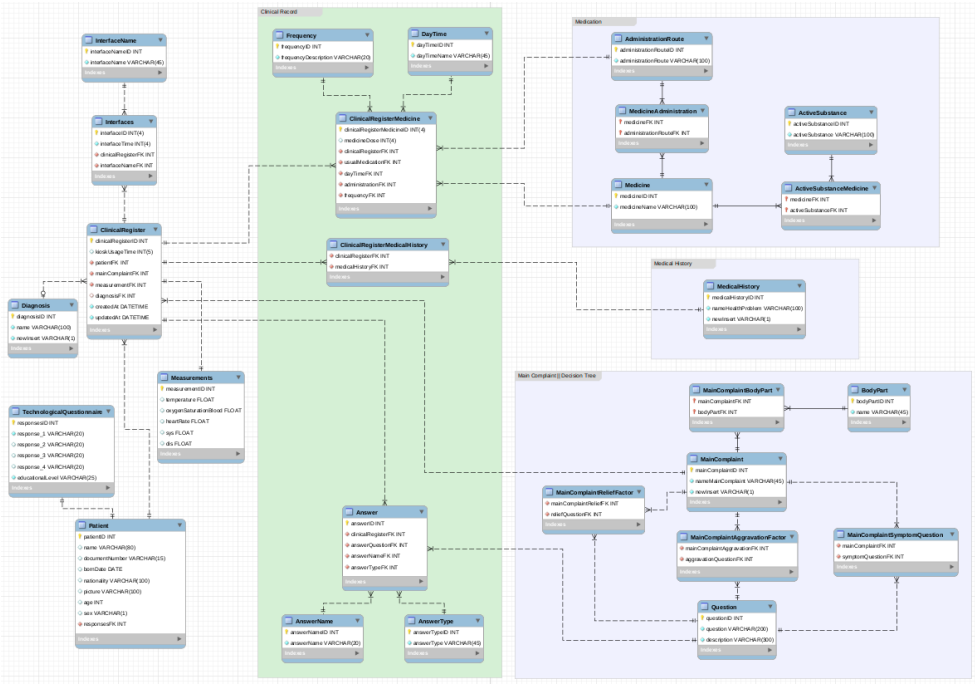


Figure 22: Physical data model

Since the kiosk will be used in Portugal, the "Medication" group tables will store information on all medicines sold in Portugal, and the "MedicalHistory" table the most widespread health problems in Portugal. The tables of the "MainComplaint" group will store information related to the main complaints most frequently cited by patients in this country. The "Diagnosis" table will store the most common diagnoses. These tables will be pre-populated before patients use the system.

It is important to ensure that the database is always in a consistent state whenever a new clinical register is recorded. A transaction was implemented for this purpose, which is illustrated in figure 23.

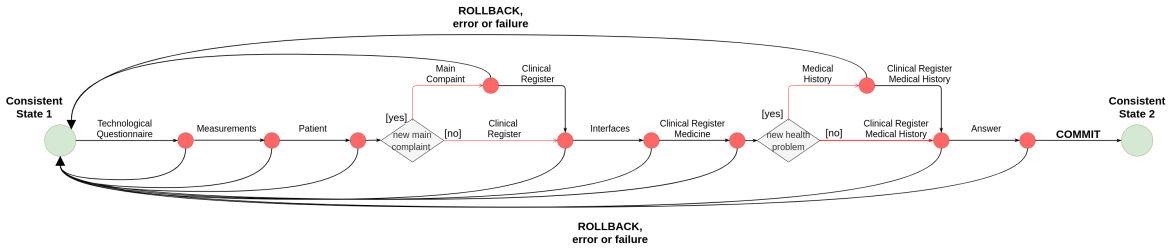



Figure 23: Tasks of the clinical register transaction

This transaction is executed whenever a new clinical register is inserted. Each of the tasks in the illustrated transaction corresponds to an insertion operation in a table, but some of the insertions may not be performed. This only happens in the following cases and tables: when a patient does not present usual medication (ClinicalRegisterMedicine), when a patient has no health problems (ClinicalRegisterMedicalHistory), when the patient presents a main complaint different from the ones the system offers (Answer), when the patient does not register a new health problem (MedicalHistory), and when the patient does not register a new main complaint (MainComplaint).

Looking at the figure 23, the order of insertion in the tables is as follows: TechnologicalQuestionnaire, Measurements, Patient, MainComplaint, ClinicalRegister, Interfaces, ClinicalRegisterMedicine, MedicalHistory, ClinicalRegisterMedicalHistory and Answer. In case any of the entries in these tables fail or cause an error, a ROLLBACK occurs and the database returns to the most recent consistent state. If all tasks are successfully performed, the COMMIT of the new data occurs.

During the kiosk usage, patients will interact with the digital components of each application interface. How this happens is important because it lets determine which interfaces are more difficult to use and why. To make this analysis possible, all the clicks that each patient performs on the different application interfaces will be stored in the Firebase Realtime Database service. The figure 24 shows the structure of the idealized collection where this information is stored.
The set of each patient's clicks coordinates will be associated with their citizen card number. Since several clicks can be performed on each interface, each click must be linked to an interface. For each click, the date and time of the click, visit number to the interface, and x and y values of the coordinate will be stored.

```
users: [
   "CitizenCardNumber": {
        "interfaceName": {
           _id: {
                dateTime: DateTime,
                nVisit: Int,
                posX: Float,
                posY: Float
            },
            ...
        },
        ...
    },
    ...
]
```

Figure 24: Structure of the JSON object where the click coordinates are stored

The front side and the profile photo of the patient's citizen cards will be stored in the Google Cloud Storage. This is a BLOB (Binary Large Object) storage solution that stores unstructured data. The initial layer of structure is called buckets which are like file folders. It's possible to group data like audio files into one bucket and images into another, or put all data for a specific project into one bucket. Each record of the image pair will be associated with the respective citizen card number from which they were extracted so that the respective profile photo of each patient can be obtained later. The figure 25 illustrates the bucket structure.

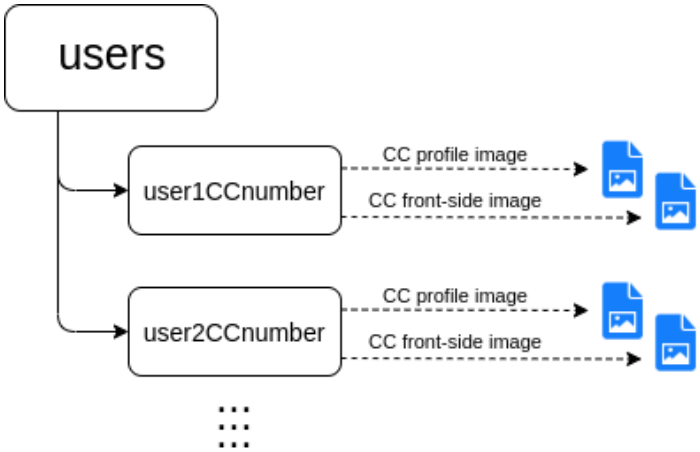



Figure 25: Bucket model structure where the images are stored

## 5.3 WEB SERVICES

The Web services' implementation is presented in this section, the three being similar in how they develop. The Web services follow the RESTful architecture, i.e., everything is considered a resource, and JSON is used as the data transmission format between the client and Web services. Each Web service has its API that gives access to its resources and responds to specific requests from specific clients. Android and Web clients only make HTTP GET and POST requests, as they only need to get resources and create resources. Visual Studio Code was the tool used for the development since it offers features that allow simplifying and accelerate the development process.
To implement each web service was used the technologies stack *Node.js*, *Express.js*, *Sequelize*. Express is a minimal and flexible Node.js framework with utility methods and HTTP middleware, creating a robust API quickly and easily. Sequelize is a promise-based Object-relational mapping (ORM) for Node.js. It supports the dialects PostgreSQL, MySQL, SQLite and MSSQL and features solid transaction support, relations, read replication and more. The used libraries will be mentioned throughout this section.

### 5.3.1 *Authentication*

The three Web services contain endpoints that allow access to personal data or patient clinical register information. Improper access to this information may have serious consequences for the patient and may lead to theft and destruction of private data, or the theft of the patient's identity. It is necessary to ensure that only authorised entities have access to Web services resources in order to avoid security breaches. Blocking unauthorized access plays a crucial role in preventing data breaches. Requests from the Android client and the Web client are the only ones that the Web services must accept, and all others must be ignored.

Before presenting the authentication method implemented, it is important to note that Authentication is different from Authorization, despite being used interchangeably many times. Authentication is the process of verifying who a user is, while authorization is the process of verifying what they have access to. Authentication is usually done before authorization.

#### *Token Based Authentication*

Token based authentication was the authentication method implemented. A token is a piece of encrypted data, which when combined with the correct tokenization system becomes a vital player in securing an application. This authentication method works by ensuring that each request to a server is accompanied by a signed token which the server verifies

for authenticity and only then responds to the request. In Token based authentication, the client sends its credentials (e.g., mail and password) to the server and the server replies with a token. Since then, instead of sending the credentials to the server along with each new request, the client only sends the token for authentication.
Each token is self-contained and contains all the information it needs for authentication, which gives the tokens the stateless feature. This feature is great for scalability as it frees the server from having to store session state. The token has an associated lifetime and expires after a set amount of time, forcing a client to authenticate again.

*JSON Web Token*

There are several types of tokens the one used in the implementation was the JSON Web Token. It is an open standard (RFC 7519) that defines a compact and self-contained way for securely transmitting information between applications as a JSON object. This information can be verified and trusted because it is digitally signed. The JSON Web Token consists of three parts: *Header*, *Payload* and *Signature*. Header is the first part of JWT and consists of two parts: the type of the token (typ), which is JWT, and the signing algorithm being used (alg), usually HMAC SHA256 or RSA. The HMAC SHA256 was used for the implementation.

```
{
    "alg": "HS256",
    "typ": "JWT"
}
```

Figure 26: JWT Header

The payload is the second part of JWT, which contains the claims. Claims are the statements that are sent, typically statements about the authenticated user and additional data. As Web and Android application users do not need to register to access the functionalities, it is not possible to obtain information that uniquely identifies each user, such as username or email, for example. As an alternative, the payload illustrated in figure 27 was implemented. This payload contains two objects, *tokenPass* and *tokenUser*, each one's value is pre-stabilised. Therefore, for a request to be accepted by the server, the attached token must contain the pre-established values of these fields.

```
{
    "tokenUser": "...",
    "tokenPass": "..."
}
```

Figure 27: JWT Payload

The Signature part is the last part of the JWT. To create the signature part it is necessary the encoded header, the encoded payload, a secret, and the algorithm specified in the header. For the HMAC SHA256 algorithm, the signature is generated through the formula shown in figure 28a. The result of this formula will be a JWT token, as shown in figure 28b.

```
HMACSHA256(
    base64UrlEncode(header) + "." +
    base64UrlEncode(payload),
    secret)
```

(a) JWT signature

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.ey
JzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6Ikpva
G4gRG9lIiwiaWF0IjoxNTE2MjM5MDIyfQ.SflKx
wRJSMeKKF2QT4fwpMeJf36POk6yJV_adQssw5c
```

(b) JWT token

Figure 28: JWT signature and JWT token

This signature makes it possible to verify the integrity of the token to ensure that the message has not been modified along the way. This prevents man-in-the-middle attacks, where an attacker could intercept a request and modify its content. The *secret* plays a crucial role because only those in its possession can create, modify and validate a token. Web services will be the only parties to have access to the *secret*. All generated tokens have a maximum expiration time of twelve hours.

The library *jsonwebtoken* was used to implement these features of the JWT token. This library has a set of pre-defined methods and callbacks, which can speed up the JWT token implementation.

The sequence diagrams of the figures 29 and 30 summarise the authentication process carried out for each request performed to each web service. Figure 29 illustrates the sequence of steps that are performed for a client to request and receive a token, and figure 30 the sequence of steps undertaken to authenticate each request.

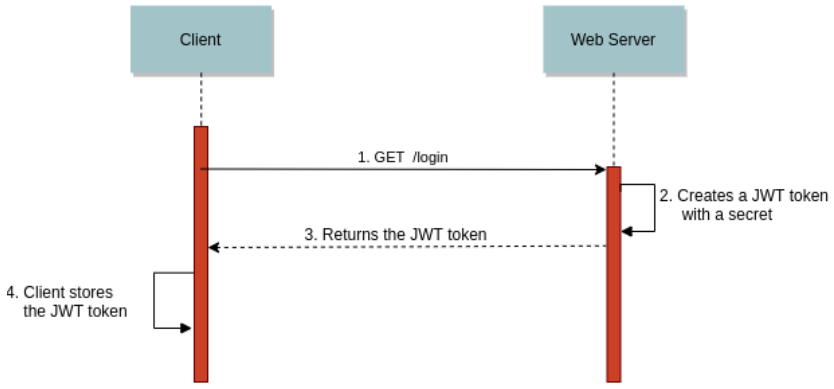



Figure 29: Sequence diagram illustrating the interactions between the client and the Web service in order to obtain a token

The steps in figure 29 are performed whenever a client does not have a token store, or the token is not valid (the 12 hour period has expired, p.e.). When a client receives the token, the client will store it for future use in new requests. In the Web client case, the token is stored in the local browser storage, and in the Android client case, it is stored in memory. When a client already has a valid token stored, the steps shown in the sequence diagram of figure 30 are performed for each request.

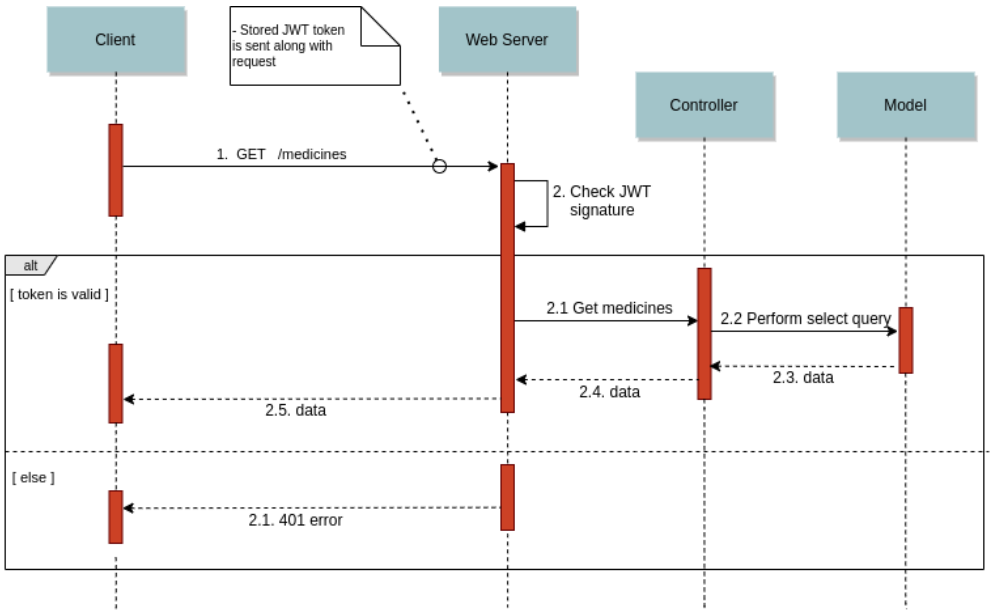



Figure 30: Sequence diagram showing the interactions between the client and the Web service performing a request

Figure 30 shows the sequence of events that are done by each Web service to respond to each request. The diagram is being applied to the request to get the list of all medicines, but it can also be applied to any other request. The token is sent to each Web service API via the HTTP Authorization header of each request. If the token is not valid an error message with response code 401 is sent to the client. The 401 Unauthorized Error is an HTTP response status code that indicates that a request has not been accepted because the authentication credentials are not valid to obtain an intended resource.

### 5.3.2 *Web service A*

The Web service deals with part of the android client's requests. Respond to requests related to the patient's usual medication and medical history is the main responsibility of this Web service. All requests are to consume Web service resources. Tables 3 and 4 shows the API endpoints that allow interaction with the Web service A.

| HTTP method | Resource URI | Response |
|---|---|---|
| GET | /medicine/<br><br>Returns all medicines | [{"medicineID":1,"medicineName":"Zoloft"}, {"medicineID":2,"medicineName":"Brufen"}, ... ] |
| GET | /administrationRoute/<br><br>Returns all administration routes | [{"administrationRouteID":1,"administration Route":"Oral"},{"administrationRouteID":2, "administrationRoute":"Intravenous"}, ...] |
| GET | /medicineAdministration/<br><br>Returns all the associations between each medicine and its administration routes | [{"medicineFK":2,"administrationRouteFK":1}, {"medicineFK":2,"administrationRouteFK":2}, {"medicineFK":3, "administrationRouteFK":}, ...] |
| GET | /healthProblem/<br><br>Returns all health problems | [{"healthProblemID":1,"nameHealthProblem": "Asthma"},{"healthProblemID":2,"nameHealth Problem":"Tuberculosis" }, ...] |

Table 3 Web Service API A description

Table 4 Web Service API A description [continuation]

| HTTP method | Resource URI | Response |
|---|---|---|
| GET | /answerType/<br><br>Returns all questions types name | [{"answerTypeID":1,"answerType":"Relief factor"},{"answerTypeID":2,"answerType":"Aggravation factor"}, ...] |
| GET | /frequency/<br><br>Returns all medicines taking frequencies | [{"frequencyID":1,"frequencyDescription": "Everyday"},{"frequencyID":2, "frequencyDescription": "3 times a week"}, ...] |
| GET | /interfaceName/<br><br>Returns the identification name of all Android application interfaces | [{"interfaceNameID":1,"interfaceName":"Healt problems"},{"interfaceNameID":2, "interfaceName":"MainComplaint"}, ...] |
| GET | /answerName/<br><br>Returns all answer options | [{"answerNameID":1,"answerName":"Yes"}, {"answerNameID":2,"answerName":"No"}, {"answerNameID":3,"answeName":"Less than 7 days"}, ...] |
| GET | /login/<br><br>Allows authentication on the Web service, returns a JWT token | {"token":"eyJhbGciOiJIUzI1NiIsInR5cCI.eyJzdWIiOiIxMjMoNTY3ODkwIwibmFtZSI6IkpvaG4gRG9lIiIyfQ.SflKxwR4fwpMeJf36POk6yJV_adQssw5c"} |

All resources are represented by a single URI. The */login/* URI is the only one that is available without the need for the Android client to be authenticated.

### 5.3.3 *Web service B*

The Web service B only handles requests from the Web client. Almost all operations performed by the Web service are data queries on patients' clinical registers. The only POST operation is triggered when the physician submits the diagnosis of each patient. Most requests involve joining data from different tables in the database, which implies further processing by the database server. Tables 5 and 6 presents the API endpoints that the Web client consumes to provide the patient's clinical register data to the physicians.

| HTTP method | Resource URI | Response |
|---|---|---|
| GET | /patients?patient={patientID}<br><br>Return a specific patient | {"patientID":45,"name":"Manuel Cunha","healthNumber":"00000001","bornDate": "1970-10-10","nationality":"Portugal","picture": "https://firebasestorage.googleapis.com/nnn", "age":50,"sex":'M',responsesFK:45} |
| GET | /measurements?measurement= {measurementID}<br><br>Returns a specific measurement | {"measurementID":45,"oxygenSaturationBlood": 96,"heartRate":86,"sys":108,"dis":69,"sendMail":'N', "sendSMS":'S'} |
| GET | /clinicalRegisters/<br><br>Returns all clinical registers not yet diagnosed | [{"clinicalRegisterID":45,"patientID":45,"name": "Manuel Cunha","healthNumber":"00000001","bornDate": "1970-10-10","nationality":"Portugal","picture": "https://firebasestorage.googleapis.com/nnn", "age":50,"sex":'M',responsesFK:45}, ... ] |
| GET | /clinicalRegisters/register/register= {clinicalRegisterID}<br><br>Returns the data from a specific clinical register | {"patientID":45,"name":"Manuel Cunha","healthNumber":"00000001","bornDate": "1970-10-10","nationality":"Portugal","picture": "https://firebasestorage.googleapis.com/nnn", "age":50,"sex":'M',responsesFK:45, "nameMainComplaint":"Muscular pain","measurementFK":45,"name":"Chest"} |
| GET | /clinicalRegisters/healthProblems? clinicalRegister={clinicalRegisterID}<br><br>Returns health problems associated with a clinical register | [{"nameHealthProblem":"Asthma"}, {"nameHealthProblem":"Tuberculosis"}, {"nameHealthProblem":"Anxiety"}] |
| GET | /clinicalRegisters/usualMedication? clinicalRegister={clinicalRegisterID}<br><br>Returns the usual medication associated with a clinical register | [{"medicineName":"Cloxam","dayTime": "Breakfast","medicineDose":1,"activeSubstance": "Cloxazolam","frequencyDescription":"Every day","administrationRoute":"Oral"}, ...] |

Table 5 Web Service B API Description

All resources are represented by a single URI. The */login/* URI is the only one that is available without the need for the Web client to be authenticated.

Table 6 Web Service B API Description [continuation]

| HTTP method | Resource URI | Response |
|---|---|---|
| GET | /login/<br><br>Allows authentication on the Web service, returns a JWT token | {"token":"eyJhbGciOiJIUzI1NiIsInR5cCI.eyJzdWIiOiIxMjMoNTY3ODkwIwibmFtZSI6IkpvaG4gRG9lIiIyfQ.SflKxwR4fwpMeJf36POk6yJV_adQssw5c"} |
| POST | /diagnosis?diagnosis={diagnosisID}&clinicalRegister={clinicalRegisterID}<br><br>Records a patient's diagnosis. The diagnosis name is already stored in the database | {"Inserted"}<br><br>500, Internal Server Error, Erro ao registar o diagnóstico |
| POST | /diagnosis/newDiagnosis?diagnosisName={name}&clinicalRegister={clinicalRegisterID}<br><br>Records a patient's diagnosis. The diagnosis name is not stored in the database | {"Inserted"}<br><br>500, Internal Server Error, Erro ao registar o diagnóstico |

In the diagnostic register request, if an error occurs during the request processing, the client will receive an error code 500 and description, as shown in the table.

### 5.3.4 *Web service C*

The Web service C handles the remaining requests that the Android client performs. This Web service allows storing the data that each patient enters in the kiosk. The remaining requests are for query main complaint data. Table 7 shows the API endpoints that allow interaction with the Web service C.

| HTTP method | Resource URI | Response |
|---|---|---|
| GET | /mainComplaint/<br><br>Returns all main complaints | [{"mainComplaintID":1,"nameMainComplaint": "Muscular pain","newInsert":'N'},{"mainComplaintID":2, "nameMainComplaint": "Headache","newInsert":'N'}, ...] |
| GET | /mainComplaintAggravationFactor/<br><br>Returns the aggravating factors associated with each main complaint | [{"mainComplaintAggravationFK":1,"aggravation QuestionFK":27},{"mainComplaintAggravation FK":1,"aggravationQuestionFK":28} ...] |
| GET | /mainComplaintBodyPart/<br><br>Returns all body parts associated with each main complaint | [{"mainComplaintFK":1,"bodyPartFK":1}, {"mainComplaintFK":1,"bodyPartFK":2}, ...] |
| GET | /mainComplaintReliefFactor/<br><br>Returns all the relief factors associated with each main complaint | [{"mainComplaintReliefFK":1,"reliefQuestionFK": 20},{"mainComplaintReliefFK":1, "reliefQuestionFK":21}, ...] |
| GET | /mainComplaintSymptomQuestion/<br><br>Returns all symptoms associated with each main complaint | [{"mainComplaintFK":1,"symptomQuestionFK": 4},{"mainComplaintFK":1,"symptomQuestionFK":5}, ...] |
| GET | /question/<br><br>Return all questions | [{"questionID":1,"question":"When did the complaint begin?","description":null}, {"questionID":2,"question":"Do you feel intense pain overnight?","description":"Intense pain is a pain that"}, ...] |
| GET | /login/<br><br>Allows authentication on the Web service, returns a JWT token | {"token":"eyJhbGciOiJIUzI1NiIsInR5cCI.eyJzdWIiOiIxMjMoNTY3ODkwIwibmFtZSI6IkpvaG4gRG9lIiIyfQ.SflKxwR4fwpMeJf36POk6yJV_adQssw5c"} |
| POST | /saveInfos/<br><br>Allows to store the information that the patient registered in the kiosk | {"Transaction has been committed."}<br>{"Transaction has been rolled back."} |

Table 7 Web Service API C Description

All resources are represented by a single URI. The */login/* URI is the only one that is available without the need for the Android client to be authenticated.

The response to the POST */saveInfos/* request indicates whether the data has been committed to the database or rolled back.

## 5.4 ANDROID APPLICATION

The Android application was developed using the Android Studio IDE. The application aims to collect and store a set of specific information on each patient's clinical condition for further access by physicians. This set of information can be broken down into, personal information, medical history, usual medication, main complaint, and biometric data. The Visiomed BluethoodLE framework provided by BewellConnect is used to search, connect and transmit data via BLE. The features of this application are implemented in the Java programming language.
During the Master's work period, two Android application prototypes were implemented. The first prototype aimed to test the first user interface designed and to identify difficulties and problems using the sensors. The tests of this prototype provided a set of relevant findings taken into account for the implementation of the second prototype. The results of these tests and their discussion can be seen in Section 6.1. In this section, the implementation of each prototype is presented.

### 5.4.1 *First prototype*

The main objective of this version is to gather a set of indicators that, after analysis, allow the identification of difficulties experienced by users whilst interacting with the application. This version was implemented bearing in mind that no third party system's needs to access the data from the clinical registers. The data from each clinical register is processed and stored in the application itself, unlike in the second version, where there is a back-end system that processes and stores the data for later access by the web client. The architecture of the application is presented in Fig. 31, depicting a built-in android application that communicate with three biometric devices.

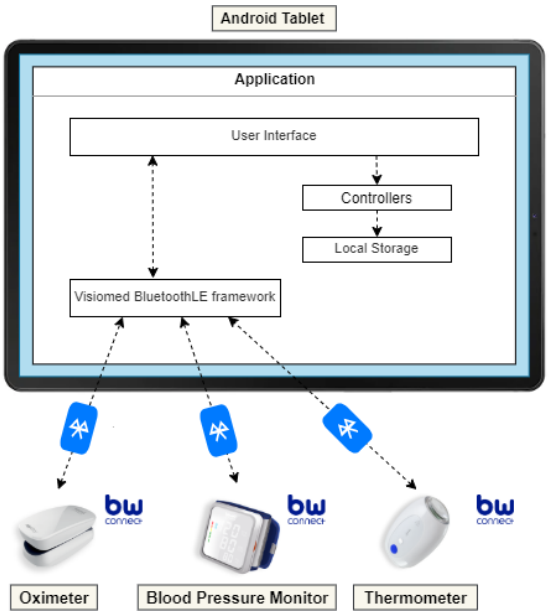



Figure 31: First version architecture

The android application follows the *Model-View-Controller (MVC)* architectural standard, which makes it possible to divide the application into layers with well-defined responsibilities. The framework and the user interface talk to each other directly, and the controllers are responsible for data processing and storage in the application database (SQLite database).

*Data storage*

The data of each clinical register is stored in the database only at the end of each kiosk session. During each session, all data recorded by a patient is stored in memory to allow faster access to them during the entire session. On the other hand, if the data were stored in the database during the session, it was necessary to access the database whenever a change carried out, a change that could rely on several tables. As access to the database is slower than memory access, it was decided to store the data in memory during the session.

*BLE sensors search and connection*

To use the Bluetooth module integrated into the device it was necessary to define some permissions in the Android manifest, figure 32. The access permission to the location of the device was also necessary for the correct functioning of the BLE connection. All these permissions are required to use the Visiomed BluethoodLE framework.

```
<uses-permission android:name="android.permission.BLUETOOTH" />
<uses-permission android:name="android.permission.BLUETOOTH_ADMIN" />
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
```

Figure 32: Permissions in Android manifest to use Bluetooth and the location of the device

The *BLUETOOTH* permission allows applications to connect to paired Bluetooth devices and the *BLUETOOTH_ADMIN* permission allows applications to discover and pair Bluetooth devices. The *ACCESS_COARSE_LOCATION* permission allows an app to access approximate location, and the *ACCESS_FINE_LOCATION* permission allows an app to access precise location.

The Visiomed BluethoodLE framework supports simultaneous connections, making it much easier to manage the connection and life-cycle of a Low Energy device. Offers a set of callbacks for each sensor which can be subscribed and adapted to the needs of each application. From this framework the following functions and callbacks were used,

- **onBLEDeviceConnectionStateEvent()**: This callback monitors a sensor connection states and allows actions to be associated to be performed whenever the sensor is in one of these states. This callback is defined for the oximeter and the thermometer.
- **onBPMStateEvent()**: Similar to the previous but for the blood pressure sensor. It monitors both connection and interaction states, for example when the patient presses the button to start the measurement.
- **startBLEScan()**: It receives the type of device to search for and starts its search;
- **stopBLEScan()**: Ends the search for a device;
- **onOximeterEvent()**: Callback that allows to obtain the values of the oximeter measurement;
- **onThermometerEvent()**: Callback that allows to obtain the values of the thermometer measurement;
- **onBPMMeasurementEvent()**: Callback that allows the blood pressure values to be obtained;

Each biometric sensor measurement is carried out in different activities, and each activity searches for and establishes the connection to the respective sensor. The search and connection of the sensors are similar in all activities, changing only the type of device that is searched. The activity diagram in figure 33 shows the search and connection process that occurs in each activity.
Each activity initially checks if the user has already granted permission to access the device's location and to discover and pair Bluetooth devices. In the case permissions have already been granted, the activity starts the process of searching and connecting the sensor, otherwise, the user will be presented with UI for accepting or rejecting them.

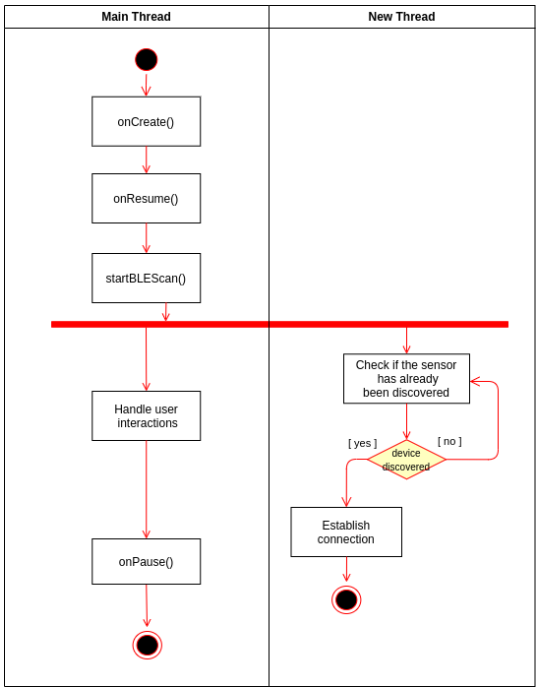



Figure 33: Activity diagram showing the search process and connection of each sensor

The search and connection process is initiated by invoking the *startLEScan()* method. This method first invokes the framework method *startBLEScan()*, which takes as argument the type of device to search for. A thread is then created to establish the connection to the sensor as soon as it is detected. Once the connection is established, the search ends (*stopBLEScan()*) and the thread is interrupted, so it is possible to start the measurement.
The connection is established in a new thread, not in the android main thread, to prevent the main thread from being locked until the connection to the device is established. If it remained locked during the search and connection process, the user could not interact with the UI.

After the device is connected, the measurement can be started. During the measurement, the callbacks will monitor the various measurement states. To the measurement states that allow to identify error situations during the measurement, a message was associated to be shown to the patient. Thus, when an error occurs during a measurement, the user is shown a *dialog* with indications on how to solve it, such as keeping the hand steady during the measurement.

When a measurement is successfully completed, the callbacks that allow the measurement values to be obtained return these values. When the user leaves the activity, the associated sensor is disconnected in life-cycle callback *onPause()*.

*Database*

An SQLite database was developed to store the data of each clinical register in the Android application. This database presents a large part of the database tables illustrated in figure 22, and the data types are different. Before patients use the application, the database is manually pre-populated with medicines, health problems, main complaints, symptoms, and relief and aggravation factors. This pre-population is necessary so that the application can interact with users, and vice versa. Figure 34 shows the SQLite database structure designed.

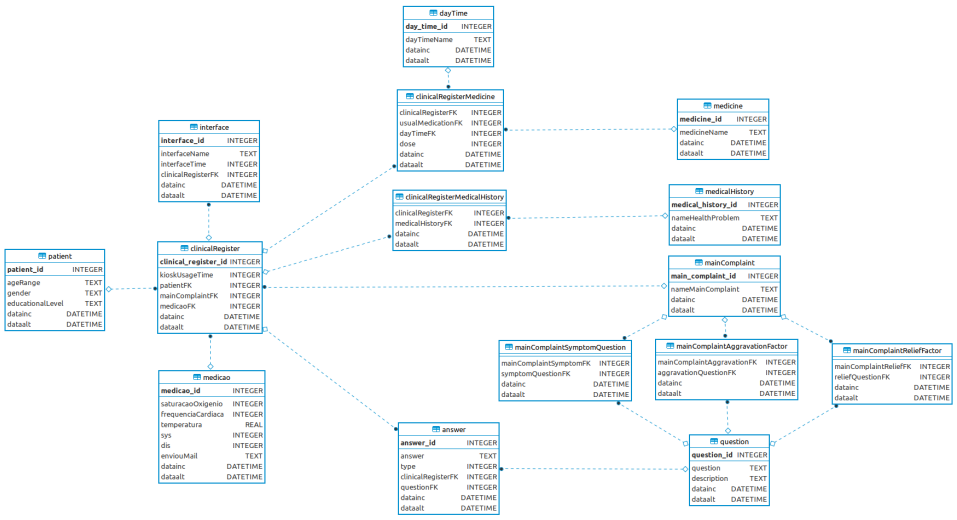


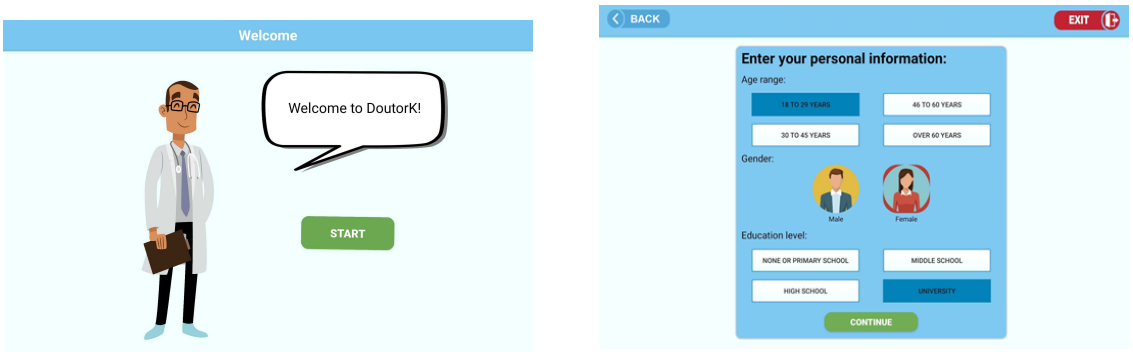

Figure 34: SQLite database of the first Android application prototype

In this application version, the patient is not asked for each medicine's administration route, only the name of each medicine, the periods of the day of taking and the dose. As the data in the clinical registers are not provided to physicians, there is no need to store each medicine's active substance, nor the personal data of each patient such as name, photo and citizen card number. The body parts were not necessary for patients to select the main complaints.
The *interface* and *clinicalRegister* tables store indicators to evaluate the usability and performance of the application. The *interface* table stores the length of time the first visit to each activity was spent, and the *clinicalRegister* the total time of the kiosk session. The *ageRange*, *gender* and *educationalLevel* attributes of the *patient* table correspond to the personal data of patients.

*Application Flowchart*

The application contains several activities with different purposes, and all except the first and the last contain a navigation bar that allows the patient to go back to the previous activity or end the kiosk session. When the session is ended, the data recorded so far by the patient are stored in the database. The patient can leave the kiosk without ending the session, leaving the session open and visible to other patients. A maximum downtime of sixteen minutes was established. Once this limit is met, the data in memory is deleted, and the application launches the *MainActivity* activity again so that another patient can start a new session.

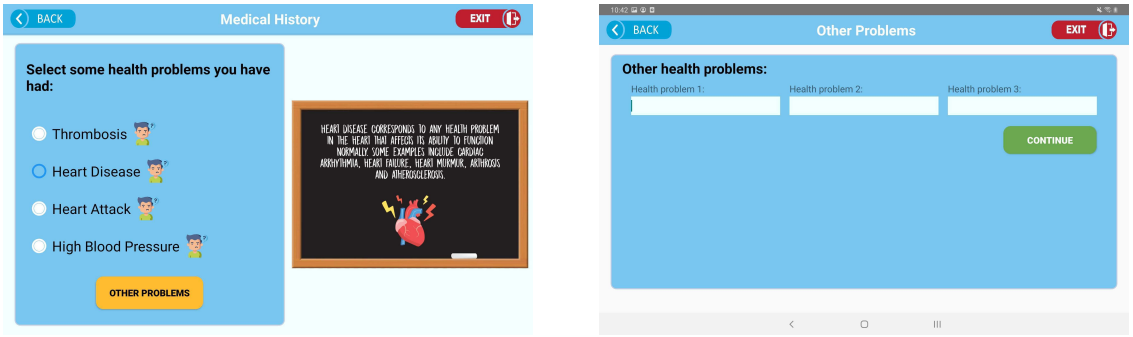
The first activity is the welcome activity, *MainActivity*, which contains a button allowing the patient to start the session (fig. 35a). Once clicked, a *dialog* with the data privacy policy and two buttons to accept or decline it is displayed. When accepted, the activity that allows the collection of patients' data, age range, gender and educational level is launched (fig. 35b).




(a) Main activity (b) Personal information activity

Figure 35: First version, welcome and personal information interfaces

The following five activities are related to health problems. In each of these activities a list of health problems is presented on the left side, and on the right side, a fragment to provide information on each health problem on the list (fig. 36a). When patients have questions about a health problem, they can click on the *info* symbol next to the problem, and the fragment will display a description and an image of the health problem.
In the last activity, when the patient clicks on the *next* button, the UI shows a *dialog* where the patient indicates the existence or not of other health problems and, if so, how many. Once the quantity is indicated, a new activity is launched with a textbox's number equal to the number indicated, in which the patient should write down the name of the health problems (fig. 36b). The maximum number of other problems the patient can write down is five.

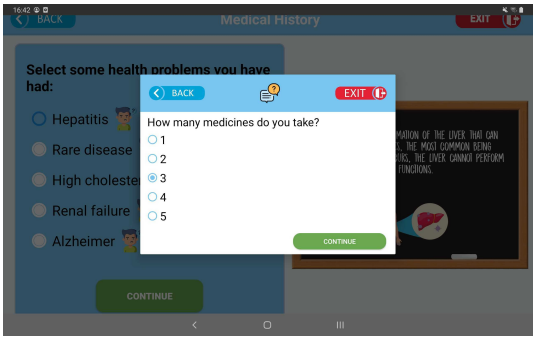



(a) Health problems activity

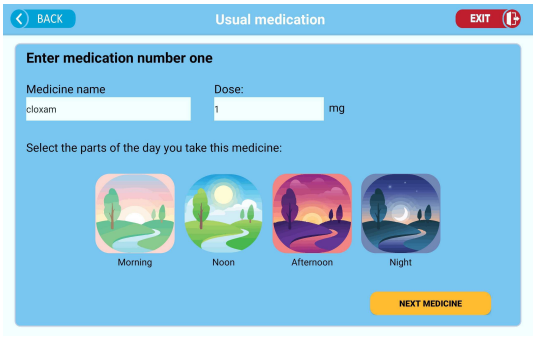
(b) Other health problems activity

Figure 36: First version, health problems interfaces

After the registration of health problems, is questioned whether or not the patient takes medication usually and, if so, how many (fig. 37a). Then as many activities will be launched as the number of medicines indicated. Each activity corresponds to one medicine, and the patient in each one must indicate the medicine name, the dose, and the periods of the day of taking (fig. 37b). To assist the patient in finding and writing the medicine, the *AutoCompleteTextView* element that shows completion suggestions automatically while the patient is typing was used. The maximum number of medicines that the patient can register is five.




(a) Medicines number dialog

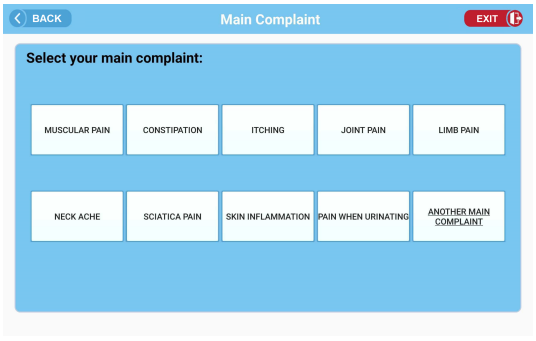



(b) Medicine activity

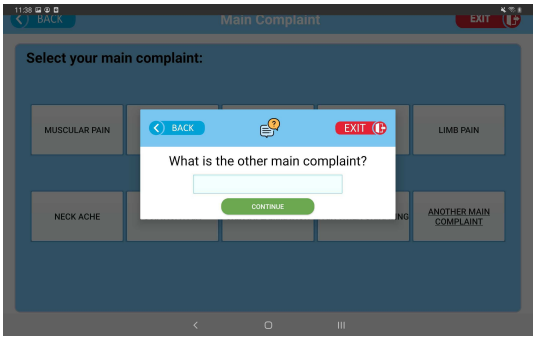
Figure 37: First version, medicine interfaces

The next two activities are related to the main complaint. Each activity contains a set of main complaints that the patient can choose (fig. 38a). If none of these main complaints is the one the patient feels, the patient registers the new main complaint through a *dialog* (fig. 38b).

The choice of one of the main complaints provided by the system or the registration of a new one has an impact on the activities launched next. The system only has symptoms, relief, and aggravation factors associated with the pre-populated main complaints. If a new complaint is entered, the patient goes directly to the activities of the biometric sensors as it is not possible to explore the new main complaint.



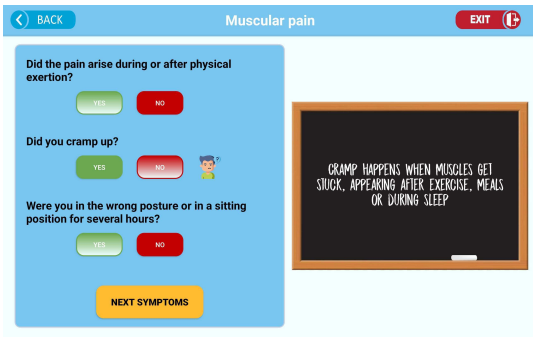

(a) Main complaint activity



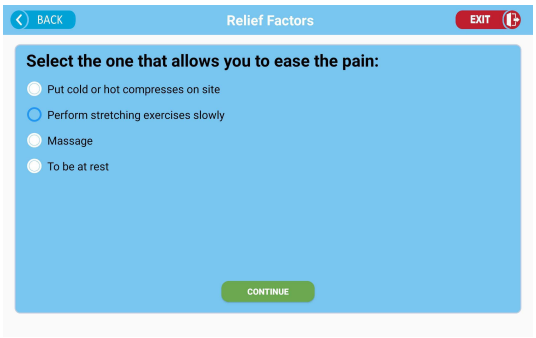

(b) Other main complaint dialog

Figure 38: First version, main complaint interfaces

The exploration of the main complaint takes place in three activities. The first activity is divided into two fragments and performs a set of questions about symptoms that the patient may have felt (fig. 39a). The fragment on the left is responsible for asking the questions and dealing with the answers. The fragment on the right provides descriptions of the questions, similar to what happens in the fragment present in the health problem activities. The next two activities are related to relief and aggravation factors. Both will expose a set of factors that may or may not be selected by the patient (fig. 39b).



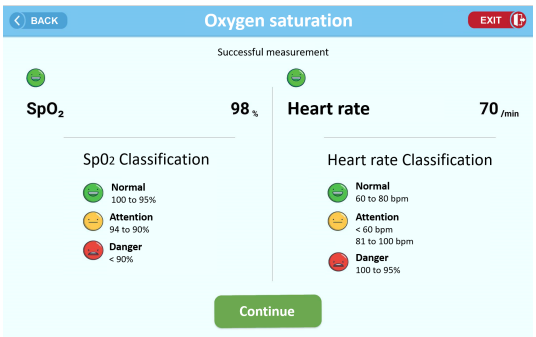

(a) Symptoms activity



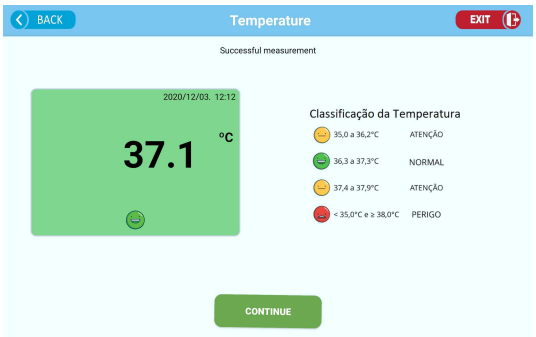

(b) Relief factors activity

Figure 39: First version, symptoms and relief factors interfaces

The next activities allow the measurement of vital signs (fig. 40a,40b,41a). Before each activity, an instructional video is shown illustrating the various steps the patient must follow. All the activities present an image with the default values and conditions of alert so that the patient can analyse the values obtained in each measurement. Whenever an error occurs, the UI displays a *dialog* box with information that the patient must follow to prevent it from happening again.



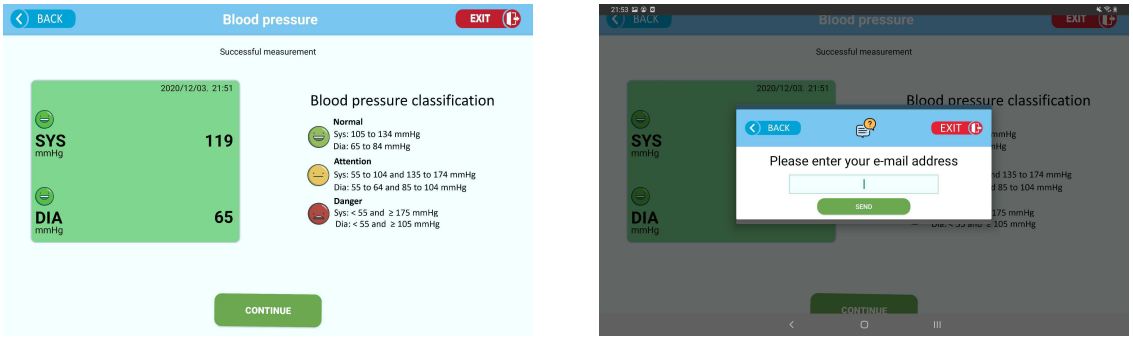

(a) Oximeter activiy




(b) Temperature activity

Figure 40: First version, oximeter and temperature interfaces

The patient was allowed to receive the measurement values by e-mail, although it was not necessary for clinical registration. After the last measurement, the blood pressure measurement, the patient indicates in a *dialog* the target e-mail (fig. 41b). The e-mail is sent asynchronously through Gmail's SMTP (Simple Mail Transfer Protocol) server. The primary purpose of this server is to send, receive, and/or relay outgoing mail between e-mail senders and receivers. An e-mail from Gmail was created for the project and defined as the sender of all e-mails.




(a) Blood pressure activity (b) Send email dialog

Figure 41: First version, blood pressure interface


Figure 42: First version, final interface

The last activity, *FinalActivity*, marks the end of the session and will stay on the screen for 15 seconds (fig. 42). During this time a message is displayed indicating that the patient should go to the waiting room and wait there for their turn. In the background occurs the storage of data that is in memory into the database. When reached the 15 seconds limit, the data in memory is deleted, and the application launches the *MainActivity* activity again so another patient can start a new session.

### 5.4.2 *Second prototype*

Observations obtained during the testing phase of the first prototype prove that the development of a new version of the application is reasonable. The user interface was adapted to a larger screen, the data storage changed, and the process of searching, connecting and transmitting sensor data is the same as in the first prototype. This version was implemented taking into account that the data will be accessed by external entities, in this case, the web clients. The storage of the data will be on a back-end server that is also accessed by the web client, as explained in Section 4.2. All clicks made by the patient on each activity will also be stored to better evaluate the application's usability. The application will communicate with the back-end server through the A and C Web service's APIs.

*Data storage*

During each session, the data entered by patients are stored in memory to allow quick access throughout the session, and their storage is performed only in the last activity, as in the first version. However, in this version, the data are stored in the back-end server database. The sequence diagram in figure 43 illustrates the data storage process.

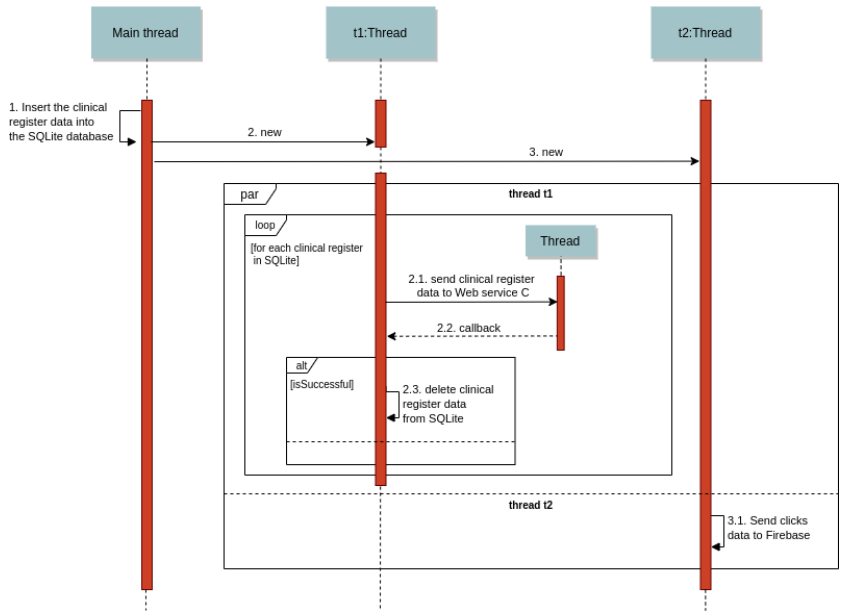



Figure 43: Sequence diagram illustrating the data storage process of the second version of the android application

To prevent data loss due to lack of internet connectivity, the application will store the patient's clinical register data in its database before sending it to the server. The clinical register data will be deleted from this database only after the application receives the acknowledgement from the server. This way, each patient's clinical register is not lost if the application has no internet or has lost its connection during sending. The data is stored by two threads, one of which is responsible for sending the clinical register data to the web service C, and the other is responsible for sending the session click's history to Firebase service.
Whenever a clinical register is sent, possible clinical registrations from other kiosk sessions that have not been registered in the server database are also sent. The clinical registers are sent to the Web service C asynchronously using the Retrofit library. Retrofit is type-safe REST client for Android and Java which aims to make it easier to consume RESTful web services. Retrofit performs and handles a request execution in a separated thread and forces to implement a Callback that is executed once the request finishes, not blocking the main thread. Each request response will dictate whether it is possible to delete the sent clinical register data from the application database.
In parallel, the session click history is sent to the Firebase Realtime Database service. This data is not previously stored in the application database because Firebase itself resends any writes when network connectivity is restored.

*Database*

As in the previous version, an SQLite database was developed to store each clinical register data in the application. This database will also store all medicines, health problems, main complaints, symptoms, and relief and aggravation factors. The application could make requests to the back-end server during the kiosk session to obtain any of these data, but internet connectivity problems or a weak one would compromise the response to requests. Even in the case of a stable connection, the response time to the requests suffers from the data serializations that have to be performed to exchange data between the several back-end server components. Alternatively, the application will request this data from the back-end server the first time the application is launched and insert it into its database. Thus, whenever the application needs any of this data it only needs to query its database. Fig44 illustrates the SQLite database present in the android application.

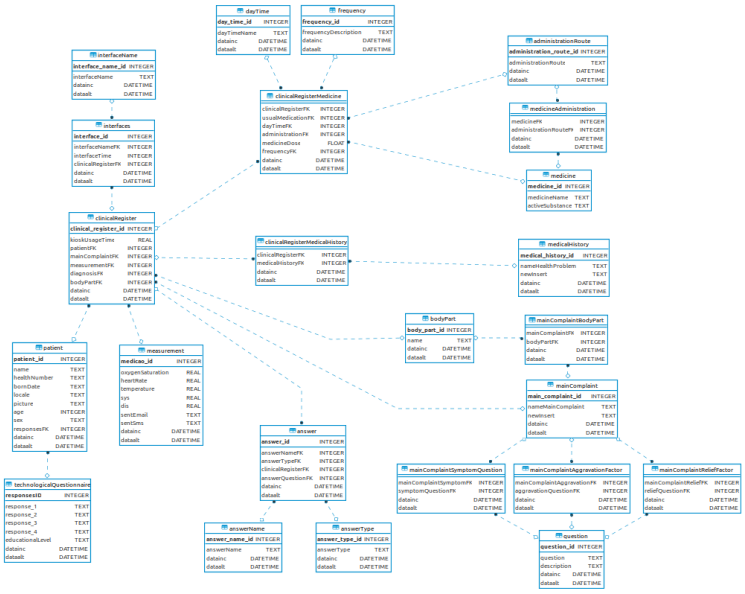


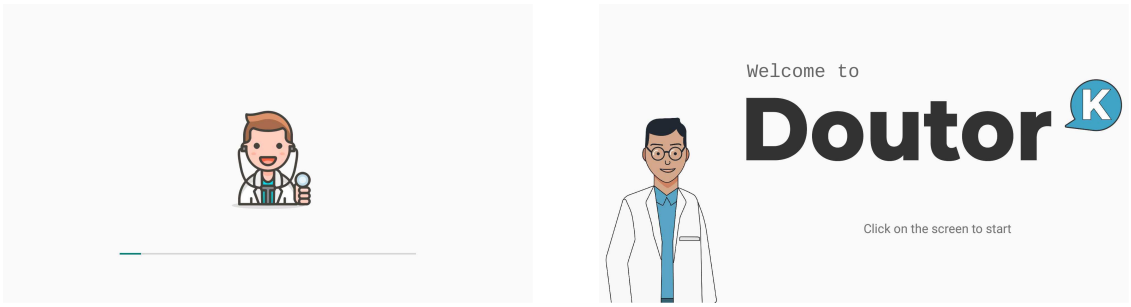
Figure 44: SQLite database of the second Android application prototype

The database contains the same tables and attributes as the back-end server database, except the tables *Diagnosis*, *ActiveSubstance* and *ActiveSubstanceMedicine*. These three tables are only operated by the web client and do not need to be stored in the application database.

*Application Flowchart*

The application contains eleven activities in total, and all except *SplashActivity*, *MainActivity* and *FinalActivity* present a progress bar that provides constant information to patients about their progress in the overall clinical register process, allowing them to envision and estimate the end of the kiosk session. As in the first version, the activities will contain buttons that allow the patient to navigate between activities and end session. When the patient ends the session, the data entered so far will not be sent to the back-end server to be stored, only deleted from memory.

The *SplashActivity* activity is first to be launched (fig. 45a). This activity is responsible for the process of populating the application database. This process starts with the verification of the application's internet connection, followed by the request of a token to the back-end server. Once obtained the token, the application will launch a set of asynchronous GET requests to the endpoints of the Web services A and C APIs that provide the data. When all requests return the requested data, the data is stored in the application database and the *MainActivity* activity is launched. The *MainActivity* activity allows the patient to start the clinical register session and accept or decline the privacy policy (fig. 45b).




(a) SplashActivity (b) MainActivity

Figure 45: Second version, main activity interface

Once the privacy policy consent is granted, the activity *CCdataActivity* is launched, which allows the collection of the patient's personal data (fig. 46). Unlike the first version, where personal data are entered by the patient, the data will be extracted from the patient's citizen card by the application via Webcam. For this, the patient must move the front of the citizen card closer to the webcam. The application will scan and extract the following data from the citizen card, first name, last name, gender, date of birth, age, nationality, citizen card number, profile photo and card front image.

The data extraction process was implemented using the Regula Document Reader *framework* provided by Regula. This framework allows scanning and extracting data from identity documents using machine learning techniques. The extraction process is done entirely on-device, meaning that personal information never touches a third-party server. The Document Reader features a simple yet powerful interface guiding users step-by-step through the scanning process. This form of extraction avoid minutes-long form-filling.

All personal data is stored in memory during session, except the profile photo and the front image of the citizen card, which are sent in a asynchronously request to Firebase to be stored in Google Cloud Storage. The answer to this request contains the URI of the profile photo, which is kept in memory to be sent during the data storage process.

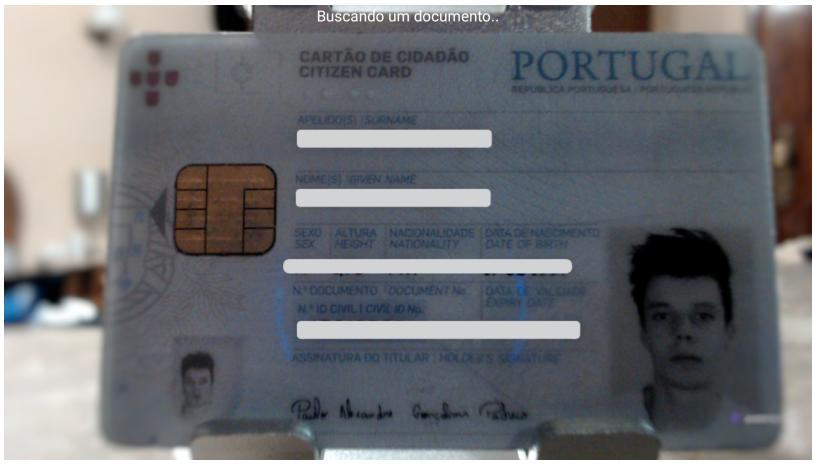



Figure 46: Second version, personal information interface

Next activity, *MedicalHistoryActivity*, is launched as soon as the extracted data is stored in memory. This activity will contain on the left a fragment responsible for presenting a list of health problems. All health problems are shown only in this activity, and not over several activities as in the first version. The insertion of new health problems is done directly in the activity through a text field.
All activities from this one will contain on the right side a fragment that allows the patient to consult and edit the responses given throughout the session. This fragment accesses the data in memory that the patient has entered so far, processes it and shows it to the patient. When the patient clicks on a question's edit button, the application directs the patient to the activity in which the question is performed to make the change.

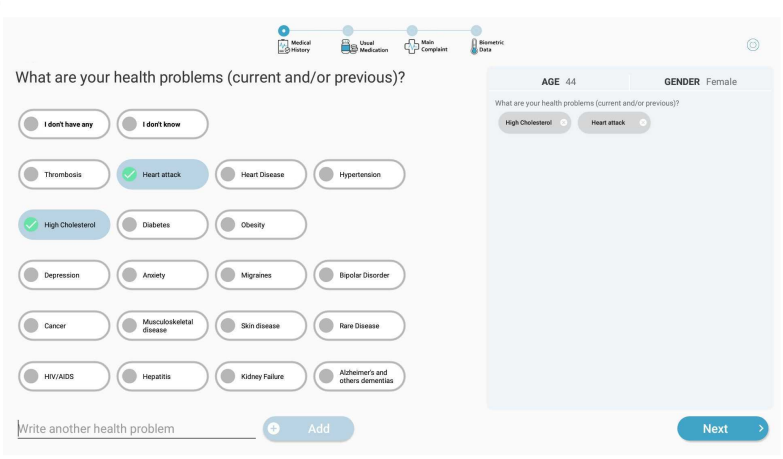
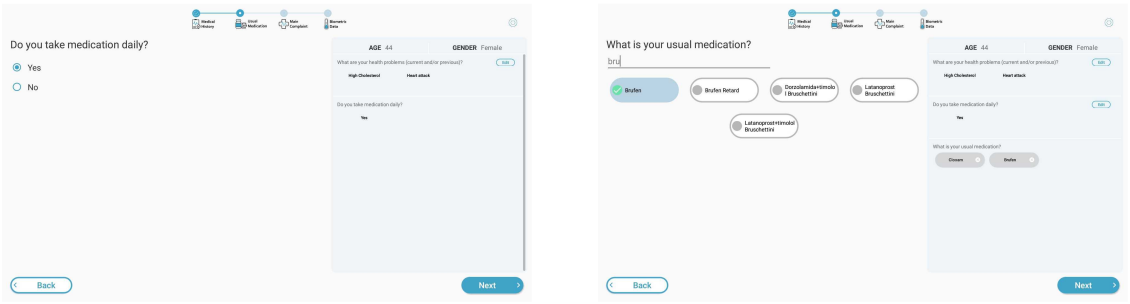



Figure 47: Second version, health problems interface

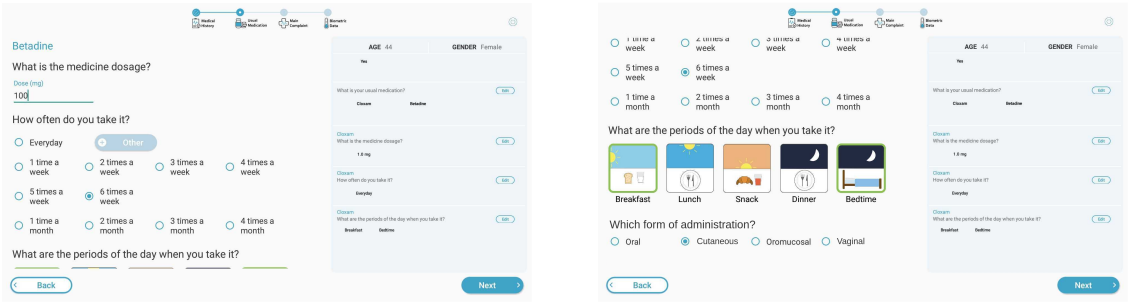
In the next activity, *QuestionMedicineActivity*, is questioned whether the patient takes any usual medication or not (fig. 48a). If not, it goes directly to the main complaint register else the activity *MedicinesActivity* is launched, in which the patient indicates the name of the medicines (fig. 37b). The patient enters as many medicines as desired without being limited to a maximum number.
To assist the patient in writing the medicines' names a fragment was implemented which receives the letters as the patient inserts them in a text filed, and provides a list with all the medicines' names that contain those letters as a substring. The list contains a set of checkboxes with medicine names that could be selected.




(a) QuestionMedicineActivity (b) MedicinesActivity

Figure 48: Second version, medicines names interface

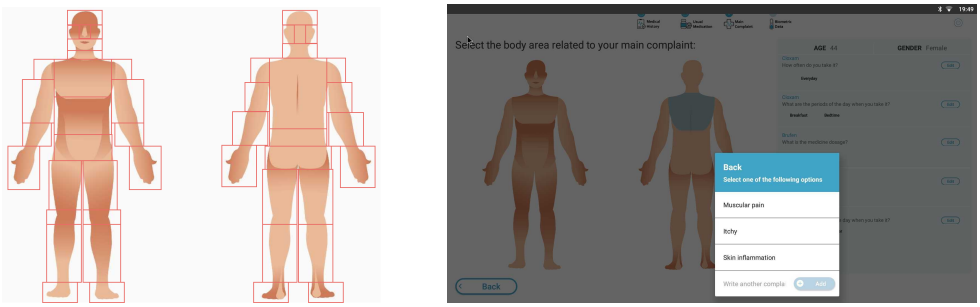
In the activity *MedicineInfosActivity* the patient is asked to insert the dose, frequency, periods of the day of taking, and administration route for each medication indicated in the previous activity (fig. 49a). A fragment was implemented to collect this information for a medicine. The activity will create an instance of this fragment for each registered medicine. The periods of the day images were changed to more representative ones, and there was the addition of the "bedtime" period of the day (fig. 49b). The patient is presented with a set of dose frequencies to select one and is asked for the administration route if the medicine has several.




(a) MedicineInfosActivity (b) MedicineInfosActivity continuation

Figure 49: Second version, medicines infos interface

The *MainComplaintActivity* is the next activity to be launched allowing the patient to register the main complaint (fig 50b). A fragment was implemented to allow patients to select the main complaint from an illustration of the human body to avoid the massive set of buttons with main complaint names from the first version. The illustration is displayed in two images (front and back of the human body), and a male or female illustration body is shown according to the gender of the patient. The patient starts by selecting the human body area associated with discomfort and a *dialog* with the main complaints associated only with the chosen area is presented. The patient selects one of the complaints provided or writes a new main complaint in the text field of the dialog.




(a) Clickable areas (b) MainComplaintActivity

Figure 50: Second version, main complaint interface

Several clickable areas were defined in each of the images to make the association between each area of the body and the main complaints, using the library ClickableAreasImages. This library allows a clickable area of an image to be defined from a point's coordinates and a width and length from the point, and assign an object which is checked when the area is touched.

Each clickable area defined for the two images, figure 50a, was assigned an object containing the name of the body part associated with that area. When the patient clicks on a specific image area, the body part name is got from the object, and the main complaints associated only with that body part are obtained from the application database. Whenever the patient clicks on a body part, the image is replaced for another one with that body part shaded, so that the patient knows if the right body part was selected.

The next three activities are only launched if the patient selects one of the main complaints that the system provides. On the left, the *SymptomsActivity* activity contains a fragment responsible for showing the patient a series of questions (fig. 51). This fragment starts by showing only one question, and the next question will appear only when the patient answers it. The "don't know" answer option was added to each question, and the first version fragment that provides information about the questions was replaced with a *tooltip*.
The *tooltip* is often used to specify extra information about something as an informational text box when the user moves the mouse pointer over an element. Each *tooltip* was implemented with the *SimpleTooltip* library, a library based on PopupWindow to create Tooltips. This way, when the patient clicks on the *info* symbol of a question, a floating container appears showing the description of the question.

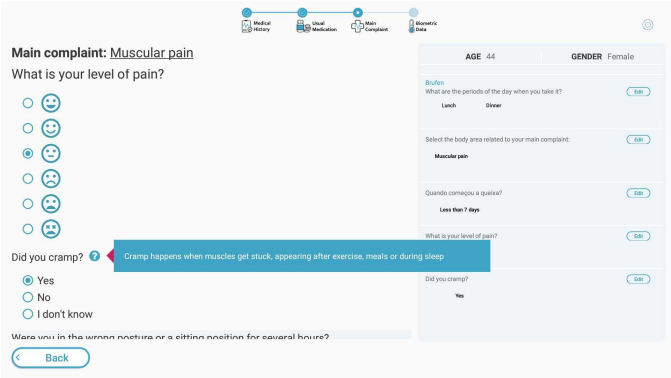
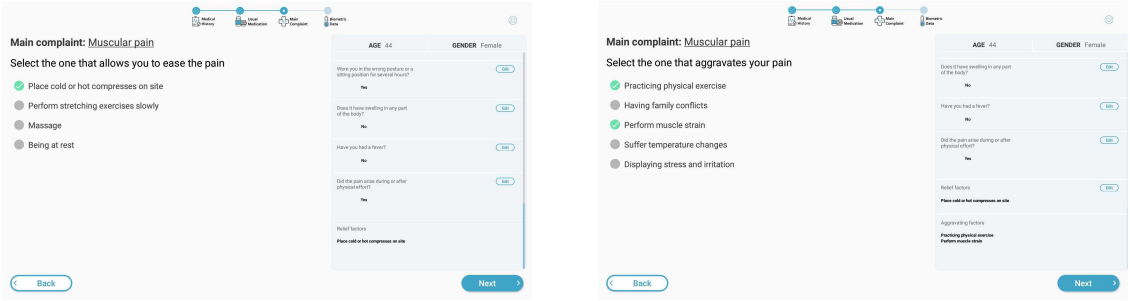



Figure 51: Second version, symptoms interface

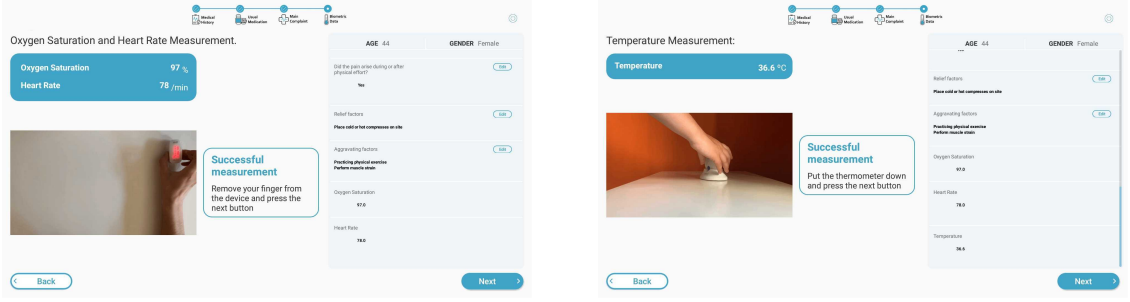
The next two activities have a similar implementation (fig. 52a,52b). Both *ReliefFactorsActivity* and *AggravationFactorsActivity* present a fragment on the left side of the activity that exposes, respectively, a relief factors and aggravating factors that may or may not be selected by the patient.




(a) ReliefFactorsActivity (b) AggravationFactorsActivity

Figure 52: Second version, relief and aggravation factors interfaces

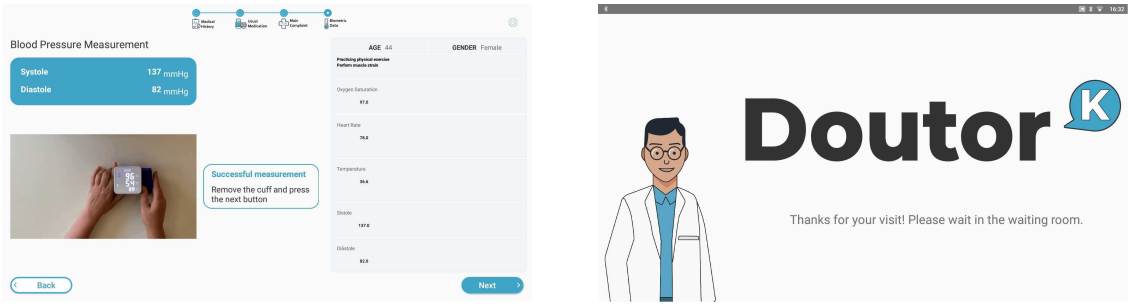
The next three activities measure the vital signs (fig. 53a,53b,54a). During each vital signs measurement, instructions are given on the next step the patient should perform, instead of the instructional video shown before each measurement. The callback, which allows monitoring of connection states for each sensor, was used to provide the instructions. The connection states were associated with the instruction to be shown to the patient.




(a) OximeterActivity (b) TemperatureActivity

Figure 53: Second version, oximeter and temperature interfaces

This version does not feature the sending of the measurement values since it is not an essential step in the anamnesis process, requiring an aditional unnecessary interaction effort. The *FinalActivity* activity, figure 54b, marks the end of the session and exhibits the same behaviour as in the first version, except that the data is stored on the back-end server database.




(a) BloodPressureActivity (b) FinalActivity

Figure 54: Second version, blood pressure and final interface

## 5.5 WEB APPLICATION

The web application was developed using the JavaScript React library. The main functionalities of the application are consultation of patients' clinical register data and diagnosis registration. Before physicians can take advantage of these functionalities, they first need to log in to the application by entering a password. The application has three web pages, *Login*, *SearchPatient* and *PatientData*, and all perform requests to the back-end server, Web service B, to build pages with relevant information, such as the health problems. The implementation and functionalities of each of these pages are described in the following sections.

### 5.5.1 *Login*

The Login page, figure 55, is the page where the web application starts. This page has a simple interface with only one text input for the insertion of the password, and an icon that shows and hides what is written in the input text.

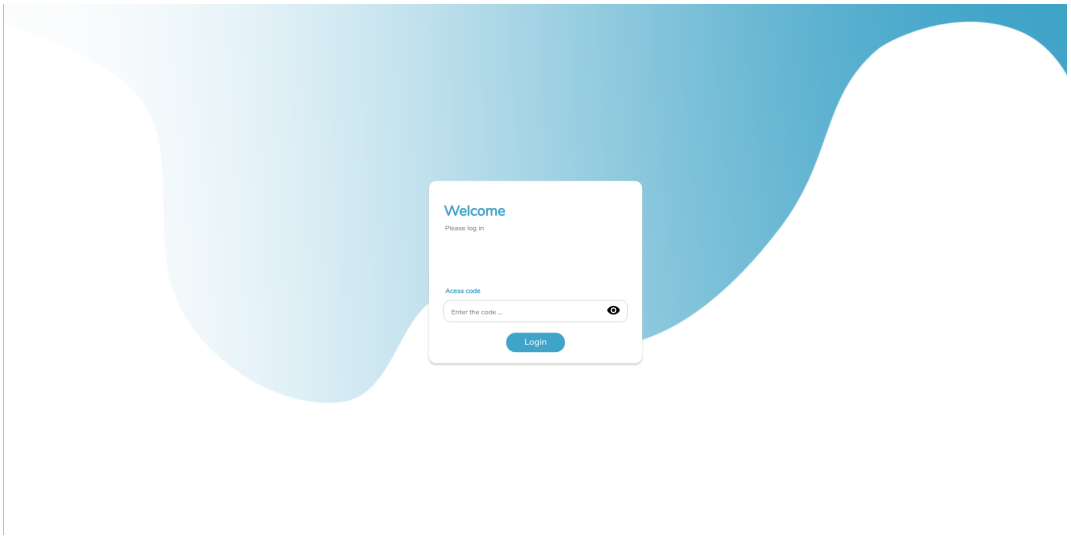



Figure 55: Login web page

This page allows the physician to log in to the application to access the functionalities of the application. The physician enters the password and clicks the Login button. Clicking this button, figure 56, the application will perform a GET request to the server through the */login* route, along with the password that the physician entered in the input text. Before this request is sent, a lookup is made to the IP address where the request comes from, to block requests not coming from Portugal or requests that may constitute a threat to the system.

The IP address lookup was implemented using the IP Address Intelligence API provided by *ipdata*[1], which provides the location of any IP address and performs Proxy, Tor and Threat Detection. If the response provided by this API indicates that there is no threat, the application performs the request to the server and store the token and its expiration date in the browser local storage as soon as the response is received.

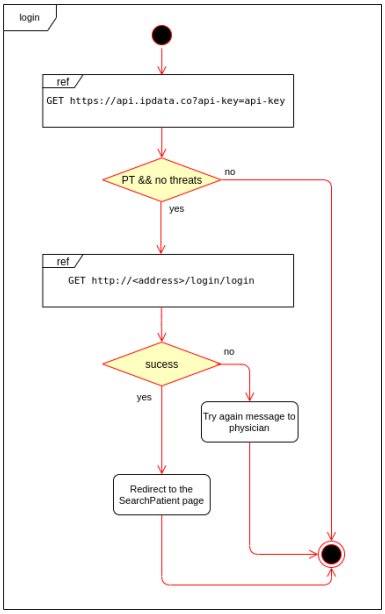



Figure 56: Diagram of activities illustrating the Web application authentication process

Upon successful login, the physician is redirected to the *SearchPatient* page. As long as the token is valid, the physician can use all the platform features. Whenever the application is re-rendered, the expiration date is used to check the validity of the token. If the token is invalid, the physician is redirected to the *Login* page to perform the authentication process again.

### 5.5.2 *SearchPatient*

The *SearchPatient* web page, figure 57, is the first page that appears to the physician after logging in and is made up of three elements, a navigation bar, a search input and a set of UI *cards* elements.

1 https://ipdata.co/

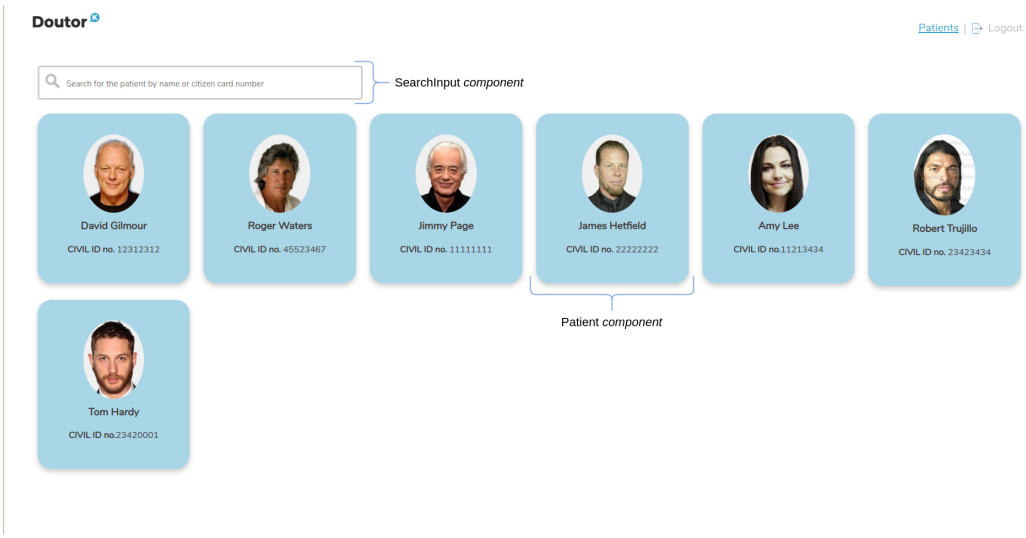



Figure 57: SearchPatient web page

The navigation bar is used for navigating between pages and logging out, the search input to search for a specific patient by name or citizen card number, and each card element shows the profile photo, name and citizen card number of a patient. Each of these elements was implemented as a react component.


Figure 58: NavBar component

The *NavBar* component, figure 58, was implemented as a stateless component and is divided into sub-components *Logo*, *NavigationItem* and *Logout* also stateless. When the application is logged out, the token and expiration date is removed from the localStorage.

As soon as this web page starts, a GET request to the server is performed on the life-cycle method *componentDidMount()* to get the list of patients not diagnosed. Once the answer is received, the list is stored in the component state, and each patient's data is passed to a *Patient* component, which tries to show them to the physician on a *card* element.

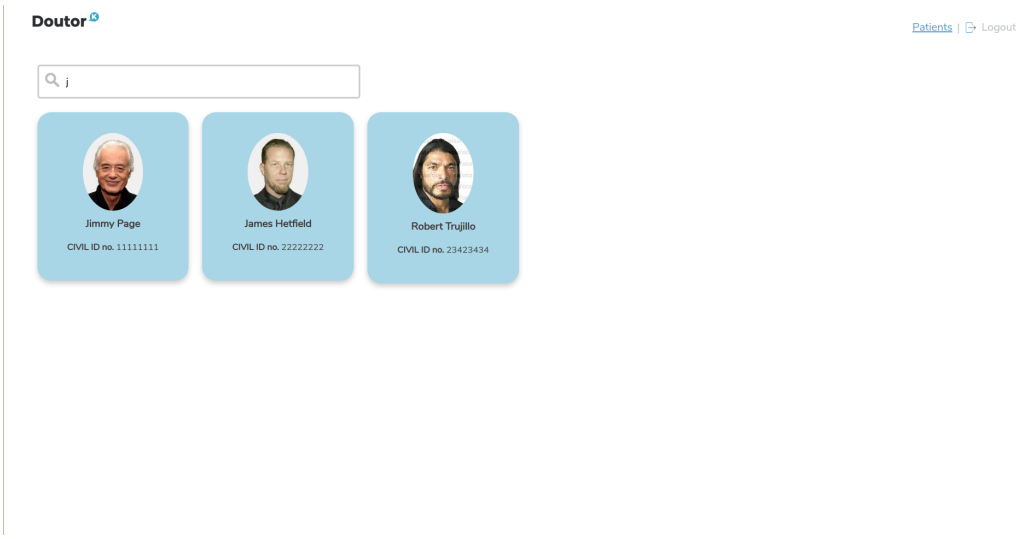



Figure 59: SearchInput component

The list is filtered as the physician enters the name or citizen card number into the *SearchInput* componente, figure 59, and only the patients with the name or citizen card number written as sub-string are displayed. When the physician clicks on a patient, the application sends the patient's id to the *PatientData* page, which shows the patient's clinical register data.

### 5.5.3 *PatientData*

The *PatientData* page, figure 60, allows physicians to visualise in an organised and structured way the clinical condition of a patient. This page comprises eight main elements: a navigation bar, a *card* that provides the patient's personal data and allows to copy all clinical data and register diagnosis, and six *cards* that provide the patient's clinical condition. The health data is presented to the physician divided into the following subjects: Clinical History, Medication, Biometric Data, Main Complaint, Relief Factors and Aggravation Factors. All these elements were implemented as a React component, and the *NavBar* component presented in the previous section was reused.

When this page is launched, the life-cycle method *componentDidMount()* makes a GET request to the server to obtain some data associated with the patient register to be passed to the components via *props*. Once the server response, all components except the *NavBar* component will request the server to obtain their respective data to show on an *card* element. These requests run asynchronously.

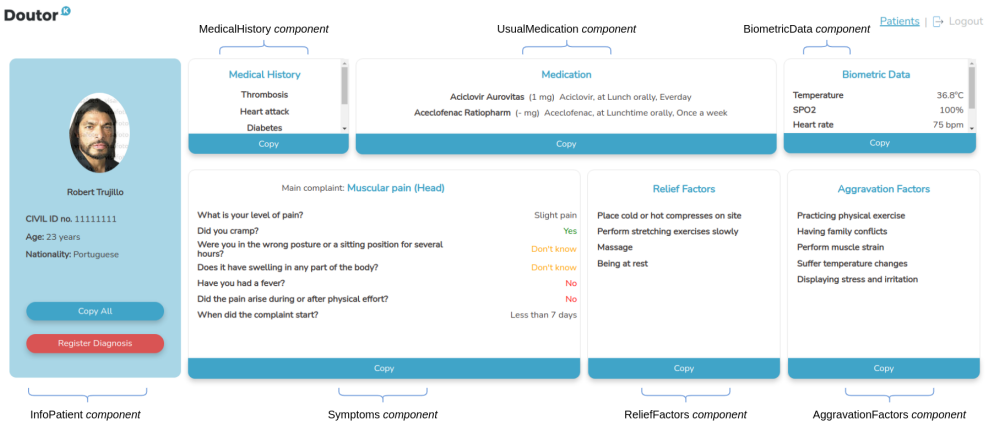



Figure 60: PatientData web page

The *PatientDetails* component has a button, "Copy all", that when clicked copies to the *clipboard* all the health data from the patient's clinical register. If physicians only want to copy a specific set of data, they can use the "Copy" button on the other components, which copies to the *clipboard* only the data present on the *card*.
The *BiometriData* component checks each measured value, and in case it is an abnormal value, a danger icon is presented next to the value. The *UsualMedication* component builds a sentence for each medicine with information about its taking. Each sentence has the following structure: [medicine name] [dose] [active substance] [periods of the day] [administration route] [frequency]. The *Symptoms* component colourizes green, yellow and red to the answers "Yes"," I don't know", and "No", respectively.

The physician records the patient's diagnosis by clicking on the "Record Diagnosis" button, figure 61, which displays a *dialog* with an input text. This *dialog* was implemented as a stateful React component which initially makes a GET request to the server to obtain the list of all diagnoses. The application presents to the physician diagnostic name suggestions as the physician writes a diagnostic name into the input text. If no diagnostic suggestions match the diagnosis entered, the physician can register the diagnosis in the same input text.


Figure 61: Dialog that allows the patient's diagnosis to be registered

When the physician presses the "Submit" button, it is checked whether it is a new diagnosis, and if so, a POST request is performed to the server on which the clinical register id and diagnosis name are sent. If it is one of the listed diagnoses, are sent only the diagnostic id and the clinical register id. The patient is redirected to the *SearchPatient* web page after the diagnosis is successfully registered.

The *dialog* shown in figure 62 was created to prevent the physician from forgetting to register a patient's diagnosis before returning to the *SearchPatient* page. This *dialog* signals the physician that a patient's diagnosis is missing. If the physician clicks on the "yes" button the application redirects the physician to the *SeachPatient* page, otherwise it removes the *dialog* so that the physician can register the patient's diagnosis.


Figure 62: Attention dialog

## 5.6 DATA WAREHOUSE

The purpose of the data warehouse is to store patient clinical register data in a consolidated format for the discovery of hidden relationships and patterns in the data. The back-end server database will periodically undergo an ETL process, in which the data will be stored in a temporary database, staging area, and then transformed and loaded into the data warehouse. This section describes the implementation and population of the staging area and the data warehouse.

Some issues to which the data warehouse is intended to respond was defined by Jacinta Santos, resulting in the indicators presented in Appendix section F. Below are some of these indicators:

- What is the average time of use of the kiosk?
- Which age range and gender have the longest kiosk usage times?
- What are the most common health problems for each age range and gender?
- What is the average time of use of the kiosk by age range and education level?

Indicators of this kind make it possible to identify the kiosk parts where patients experience the most difficulties associated with their digital competence, as well as the health characteristics of all patients. This is important to identify patient clusters with similar data and to find patterns and relationships between these data.

For example, if elderly patients frequently register a particular medicine, the interface can be adapted for these users to display that medicine as soon as it is launched.

### 5.6.1 *Staging area schema*

The relevant information for the data warehouse was selected after the analysis of the back-end server database. The tables allowing the storage of questions, answers, active substances, administration routes, body parts, frequency, and periods of the day were discarded as they did not contain relevant information for the data warehouse. The remaining tables belong to the staging area, which is composed of the following tables:

- **patient**: Table concerning the patients who performed the clinical register;
- **measurement**: Table concerning the measurements taken at the kiosk;
- **technological_questionnaire**: Table concerning the answers to the questionnaire;
- **medicalHistory**: Table on health problems provided by the kiosk;
- **medicine**: Table concerning the medicines provided by the kiosk;
- **interfaceName**: Table concerning the kiosk interfaces;
- **diagnosis**: Table concerning the diagnoses provided to physicians by the web platform;
- **mainComplaint**: Table concerning the main complaints provided by the kiosk;
- **interfaces**: Table concerning the length of stay at each kiosk interface;
- **clinicalRegisterMedicine**: Table concerning the medicines in each clinical register;
- **clinicalregisterMedicalHistory**: Table concerning the health problems in each clinical register;
- **clinicalRegister**: Tabela referente aos registo clinicos realizados pelos pacientes;

After selecting the tables, it was necessary to carry out the same data selection process, but for the attributes. The final logical model of the staging area is shown in figure 63.

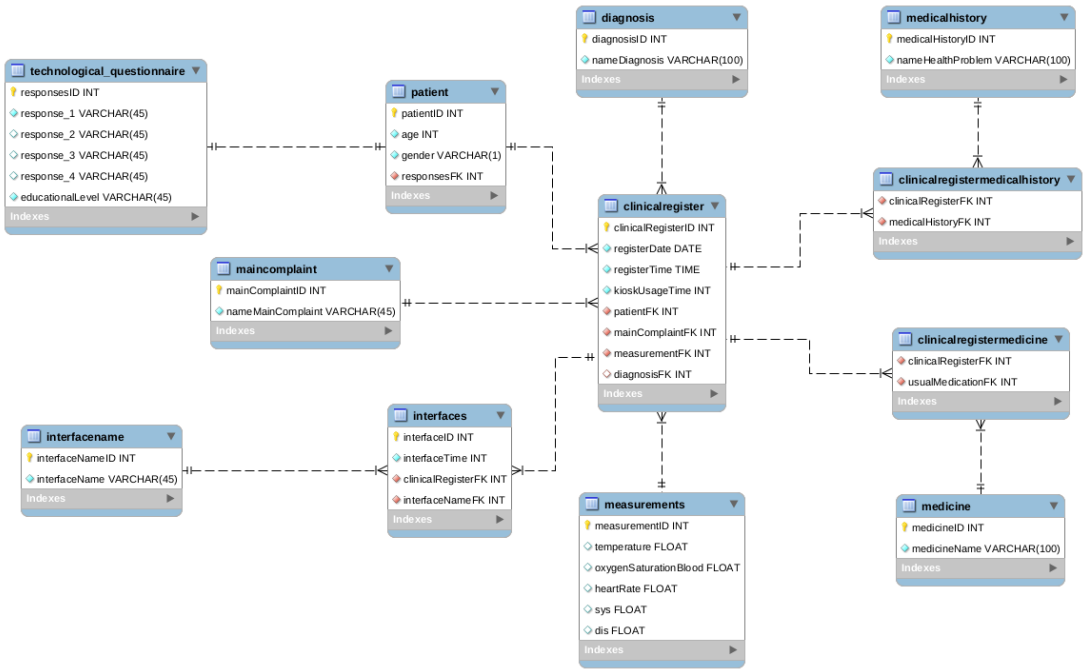



Figure 63: Staging Area

The *clinicalRegisterMedicine* table does not store the dose of the medicine, and the *Diagnosis* table does not store the *newInsert* attribute. In the *Patient* table only the age and gender of the patient is stored. The excluded attributes are due to their irrelevance to the analysis.

### 5.6.2 *Data warehouse schema*

The data warehouse was modelled as a star scheme since it is intended to evaluate only one business event, the clinical registers, and get the best possible query performance. The data warehouse is composed of the following fact and dimension tables, and its logical model is shown in figure 64:

- Facts tables:
  - **facts_clinicalRegister**: Table on clinical registers performed by patients;
- Dimensions tables:
  - **dim_mainComplaint**: Table on main complaints;
  - **dim_medicalHistory**: Table on health problems;
  - **dim_measurements**: Table concerning measurements;
  - **dim_interfaces**: Table concerning the time spent at each kiosk interface;
  - **dim_patient**: Table concerning patients;

- **dim_diagnosis**: Table concerning diagnoses;
- **dim_medicines**: Table concerning medicines;
- **dim_questionnaireResponses**: Table concerning the questionnaire responses;
- **dim_time**: Table with dates used for the clinical registers submission dates;

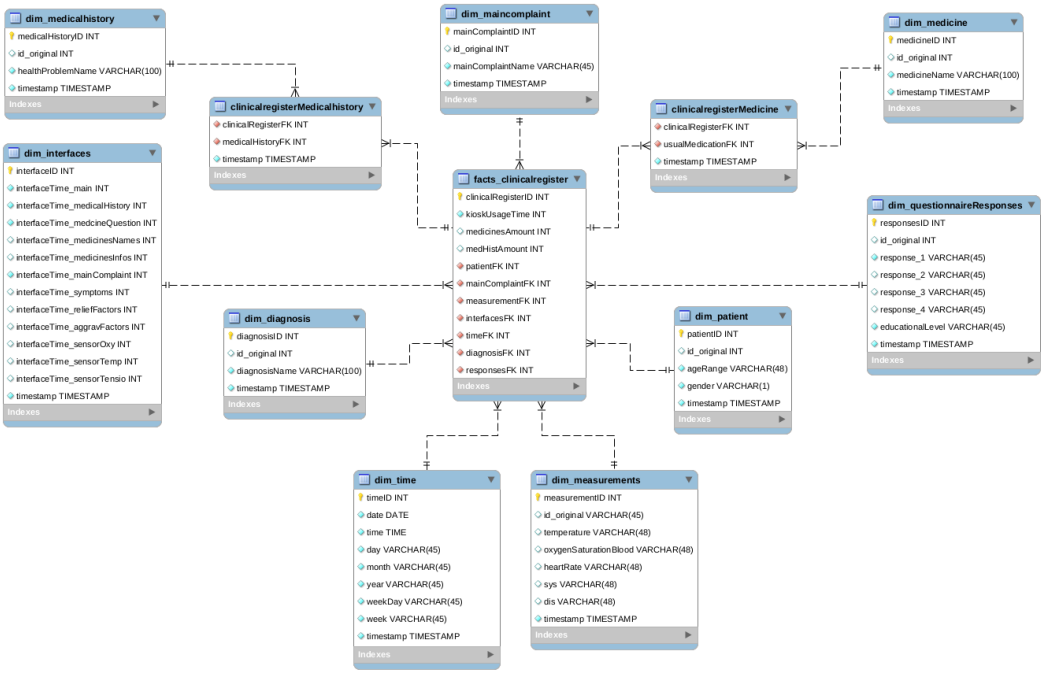



Figure 64: Data warehouse

All dimension tables have a timestamp relative to the moment of each new register insertion. Although no data updates are expected, all dimension tables except the *dim_time* table have the attribute *id_original* for manual data update issues. The dimension time table presents a low level of granularity to obtain answers with greater time detail. The *clinicalregisterMedicalHistory* and *clinicalRegisterMedicine* tables allow associating each clinical register to its health problems and medicines, respectively.

The fact table stores three indicators, the total time of the kiosk session and the number health problems and medication. These last two attributes will have a null value when no health problems or medications are registered. The aggregate functions (MIN, MAX, SUM, COUNT, and AVG) all do the "right thing" with null facts.

### 5.6.3 *Populate staging area*

The population of this temporary database is carried out through triggers SQL, created in the back-end server database, which stipulates that whenever something is entered in the database, it is also entered in the staging area. This is a fast process, which decreases the time the back-end server database is busy. After the data warehouse is populated, the data is removed from all tables in the staging area, using the DROP TABLE statement followed by the creation of all tables again. This ensures that the temporary database always has the "new" data from the back-end server database.

### 5.6.4 *Populate data warehouse*

The data warehouse is populated using *cursors*, which copy the data from the staging area to the corresponding tables into the data warehouse. The tables *dim_mainComplaint*, *dim_medicalHistory*, *dim_medicine* and *dim_diagnosis* are pre-populated with all the main complaints, health problems, medicines and diagnostics registered in the back-end server database. An extra entry was added to the table *dim_diagnosis* to be used when a clinical register does not have a registered diagnosis.
First, all the dimension tables are populated and then the fact table. The value of the attribute *id_original* of each dimension table is set to be the primary key of the associated register.
The fact table is populated by a *procedure* using a cursor. The cursor goes through the *clinicalregister* table of the staging area and for each entry, queries SQL will find the primary keys of the dimension tables associated with the attributes of that entry. These primary keys are stored in auxiliary variables, created at the beginning of the procedure, and then inserted in the fact table as foreign keys. If a clinical register does not have a diagnosis registered, the *diagnosisFK* attribute will store the primary key of the additional register created in the table *dim_diagnosis*, described above.

# 6

# EVALUATION

The evaluation of the self-service kiosk occurs in two moments. First, the first prototype of the Android platform at University of Minho and in a National Digital Competences Conference (Forum INCoDe.2030[1]) is tested. At the end of each kiosk session, users are invited to conduct a system usability questionnaire. In the second moment, the second prototype of the Android application is evaluated at Centro Hospitalar Universitário de São João (CHUSJ[2]) with patients at a real emergency facility. Patients and some nurses are invited to conduct a system usability questionnaire. The prototype of the web platform is evaluated at this moment with physicians from the hospital centre. After using the web platform in consultations, the physicians are invited to conduct a questionnaire about the system. Due to the pandemic, it was not possible to perform the second evaluation moment. The following sections describe the moments of the evaluation in more detail.

## 6.1 FIRST MOMENT OF EVALUATION

The first moment of the evaluation is a pilot with healthy participants. This phase aims to test the designed user interface and all system features to identify problems that may arise and minimize errors in real environment evaluations. To test the Android application a Samsung Galaxy Tab S5e (10.5'')[3] was used.

### 6.1.1 *Testing Methodology*

The evaluation of the kiosk's usability was carried out taking into account the following indicators: (I) average time per screen, (II) average time of a complete kiosk session, (III) application design, (IV) devices interaction, and (V) interaction with the system. These last three indicators will be evaluated through methods (B) and (C), presented below, following

1 https://www.incode2030.gov.pt/.
2 https://portal-chsj.min-saude.pt/.
3 https://www.samsung.com/pt/tablets/galaxy-tab-s/galaxy-tab-s5e-10-5-inch-black-64gb-wi-fi-sm-t720nzkatph/

a methodology similar to the one presented in [11]. To obtain these indicators, we divided the usability test into three parts: (A) kiosk usage time, (B) final questionnaire, and (C) observation report. To facilitate data analysis, PowerBI and Google forms were used to generate statistical graphs.

*Kiosk Usage Time*

The usage time of the kiosk will be analyzed taking into account the usage time per screen and the time of a complete session. The time used per screen allows taking into account which screens the user found more difficulty in terms of interaction, since these will have longer times. The time of a full session will help to reach the overall objective of reducing the session time as much as possible, which can be accomplished both in terms of application design and interaction with the sensor devices.

*Questionnaire*

At the end of each kiosk session, users were invited to conduct a system usability questionnaire. The initial questions consist in determining the user's age range, gender, education level and zip code. These questions are particularly relevant because the system is intended to be used, and was tested, by a very diversified range of users, from which one intends to understand how different factors influence the ability to use the system. To this extent, we are particularly interested in understanding how age and education level influences kiosk usage, and how effective can system design options be in overcoming general usage barriers of digital systems.

The following questions were adapted from the Post-Study System Usability Questionnaire (PSSUQ). It was specifically designed for use at the completion of usability studies [40]. The PSSUQ assesses overall user satisfaction with 16 aspects of the system and interface, such as system utility, information quality and interface quality. The response scale ranges from 7 response options (fully agree, agree, partly agree, neither agree nor disagree, disagree, partly disagree, fully disagree) and not all questions have to be answered, as there may be non-applicable questions. The questionnaire can be found in the Appendix section C.1.

*Observation Reports*

During the test phase of the kiosk, there were two observers watching and creating reports on user interaction with the system. These observations and reports provide valuable information about participants behaviour that cannot be captured by questionnaires nor interface digital monitoring alone. These were particularly insightful to identify and provide a deeper understanding of system interaction problems, difficulties in understanding the

content provided on the screens, problems with the use of the devices, and user suggestions, among other issues.

### 6.1.2 *Usability test results*

The usability evaluation was performed through (A) kiosk usage time, (B) final questionnaire and (C) observation reporting. A total of 32 participants of various ages and educational levels participated in the evaluation.

*Kiosk Usage Time*

Screen usage values, presented in Fig. 65, show that the average time of use was significantly different for each screen.

The screens with the interface to collect biometric data from the user are the ones with the longest time of use, since the measurement devices require a fixed and often significant amount of time. Here, one has to account not only for normal time that each device takes to make a measurement but also on error attempts. The blood pressure monitor clearly shows this problem, since it is the device with the most measurement errors, as explained in subsection C.

The average time of a complete kiosk session from the initial screen to the final screen was 399 seconds. This average time did not include the time of the videos explaining the use of the devices for collecting vital signs.

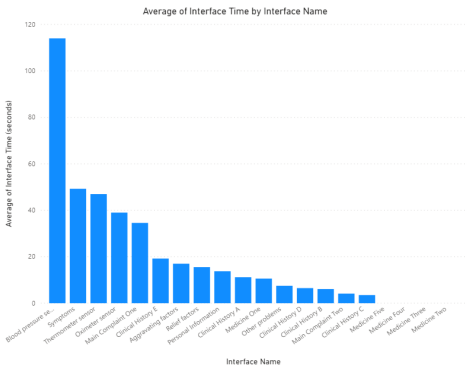



Figure 65: Interface Time Average by Interface Name

*Questionnaire*

When asked to fill-in the final questionnaire, only a small number of participants (19 participants) answered, arguing time unavailability.

The answers given to the questionnaire were essentially from higher education participants (17 participants), with a probably high digital competence.

More than 50% of the participants are between 18 and 29 years old.

About 80% of the participants fully agreed or agreed that the system is simple and easy to use, observing that all tasks could be performed quickly. Only 14 participants responded to have made errors while using the system. All of these participants fully agreed or agreed that the system provided clarity in correcting the problems and it was easy and quick to correct them.

The information presented in the system was clear and easy to find in agreement with 95% of the participants. In addition, about 80% of the participants agreed that the information was well organised and that it actually helped. Everyone found the interface of the system pleasant and were satisfied with the system developed, as the results in Fig. 66 show.


Figure 66: Overall satisfaction responses of the system

*Observation Reports*

Observers during the test phase detected difficulties in gender selection by the users, who justified it by saying that the button seemed unnoticeable as it was different from the buttons of age range and education level.

In contrast, on the clinical history screens, users did not experience difficulty in understanding the health problems presented. Users who did not have health problems mentioned that the “other problems" button, which followed the next health problems, should not be compulsory and raised the possibility of including a button to move on to the next phase, the usual medication.

In the usual medication, it was observed that there was a difficulty on the part of the users to remember their medication and to write it correctly, even with the help of the dropdown, which was little or not used by users with a low level of education. Users with a low educational level also showed doubts about introducing the number of pills they were taking or the dose of each pill.

In the parts of the day, the same users clicked on the textual description of the image button instead of the button itself.

No complaint of difficulty was perceived by the observers in both the main complaint screen and the symptom screens.

Several difficulties were found in the sensor device screens. In the use of the oximeter, questions arose regarding the meaning of the term SpO2 by low education users, which finger to insert and the correct orientation of the device for finger insertion.

On the thermometer, users experienced difficulty in placing the device at the correct distance, as well as when they should press a second button to have the measurement data sent to the application.

The blood pressure monitor was the device that presented the most error messages, due to the difficulty in placing the device correctly. In one case, a user gave up the use of the device after successive unsuccessful attempts. However, this device still showed some advantages during the trial, since the measurement is performed on the wrist and users do have to take any clothes off. This was particularly convenient in the conference trials that were carried during winter season.

The videos explaining the use of each device were not shown because of time unavailability by the participants. Thus, the instructions were given by the observers to the users and in some cases it was necessary to repeat the same explanation, since users forgot the steps to be performed.

Regarding the results of the measurements, all users showed interest in knowing if their vital data was under normal levels and almost everyone wanted to receive this information by e-mail.

On the screens where it was necessary to use the tablet keyboard, it was found that users with a low level of education felt difficulty using it, an example of this was when they wanted to delete a letter without knowing which key to press.

For the final screen, some users suggested that in an ED situation it should inform the patient about the next steps to take.

In general, we received several important general suggestions regarding the interface design. In the exploration of the main complaint screens, users suggested to replace the colours of the “yes" and “no" buttons with the same colour and to replace the “exit" button color to red with less contrast.

The all capitals labels used in the main complaint buttons was reported to affected users’ reading, and it was also suggested to increase the font size when changing to sentence case.

The lack of the progress bar was also an element pointed out by users, referring that when a task contains many steps and necessary actions, it is better to divide these tasks into several subtasks.

In the clinical history and daily medication screens, different buttons were used (e.g. “continue" button) from the buttons used in the rest of the application. This caused some confusion to users, alerting for the need to maintain internal consistency.

### 6.1.3 *Discussion*

The discussion of the results will be presented according to the division used in Section 6.1.2: (A) kiosk usage time, (B) final questionnaire and (C) observation reporting).

*Kiosk Usage Time*

The relation between average time of a complete kiosk session, the age range and education level (Fig. 67 and Fig. 68), shows that participants with age ranges between 30 and 60 have the longest time for a complete session as well as illiterate participants and participants who have completed primary and middle school. This difficult in using the system is probably due to the low level of digital competence of users, which is known to be highly correlated with education level. Surprisingly, we can see that for an age range over 60, the time for a full session of the kiosk was the shortest, contrary to what one would expect. However, these are outlier values, given that the number of participants in this age range is only 2 and their education level is much higher (middle school and university levels) than the mean education level for this age range.

Considering the data available, the age range variable has little statistic significance and can not be used to extract any correlations with the kiosk usability. Contrary to this indicator, the educational level allowed to identify systematic relations with the kiosk’s ease of use. Participants with an university educational level had a lower than average time of complete kiosk session, while all other educational levels where below it.

The devices used during the biometric data collection phase must be fast during measurement so that as little time as possible is lost. Therefore the choice of a thermometer with infrared technology is the best option as it allows measurement in a few seconds with the same accuracy. Another factor to take into account is the simplicity of the devices. A blood pressure monitor with a single key to automatically start the measurement will allow the user to have no doubts about where to press the key, offering a faster measurement. In the oximeter, this simplicity factor should also be present, and it should be a device in which only the finger needs to be inserted in it and it automatically performs the measurement without the need to press any key.

The measurement errors also affect the time of the measurement and an error message should always be displayed for the user with which error, its cause and the way to correct it. This way the frustration caused by the error will be eliminated and the user will be able to solve the error more easily and quickly.

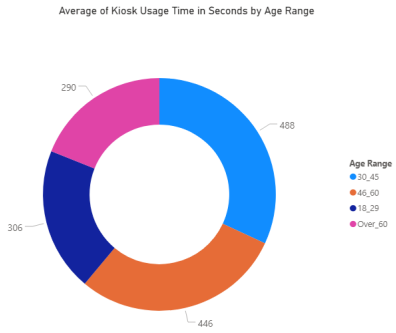



Figure 67: Average of Kiosk Usage Time in Seconds by Age Range

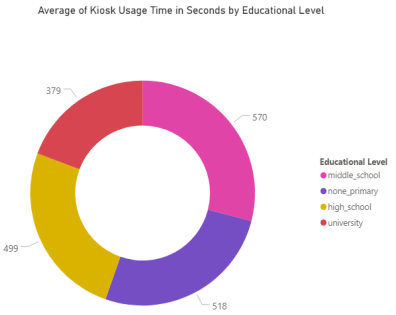



Figure 68: Average of Kiosk Usage Time in Seconds by Educational Level

*Questionnaire*

The positive and encouraging responses obtained may be due to the majority of participants presenting a university level of education and an age range between 18-29. One factor to consider in the future is to add a “not applicable" option to all questions. This would allow to register user doubts and accelerate the questionnaire answering, since users would no long have to look for alternative answers to move to the next question.

*Observation Reports*

Considering that users were forgetting the steps to be taken in using the sensor devices, we may conclude that videos are not an effective single way to explain the use of the devices, since the steps would be easily forgotten. George Miller work supports this[41], where he found that people are only able to keep five to nine items in the short-term memory before they forget or start making mistakes.

The green and red colour associated with each button of the answers to the questions made in the exploration of the main complaint, Fig. 69, can influence user’s responses, because the expansion effect that warmer colours have, when placed next to colder colours,

makes them literally spill out and take dominance [42]. So, users will be more likely to click the “no" button associated with a warm colour than click the “yes" button associated with a cold colour.

The “exit" button used in all screens also showed the colour red, obtaining a highlight that could induce users to click on it unnecessarily. The suggestion was to replace the colours of the “yes" and “no" buttons with the same colour, since similar colours infer a similarity between the objects, and the “exit" button to a red with less contrast, in order to avoid drawing the user’s attention [43].

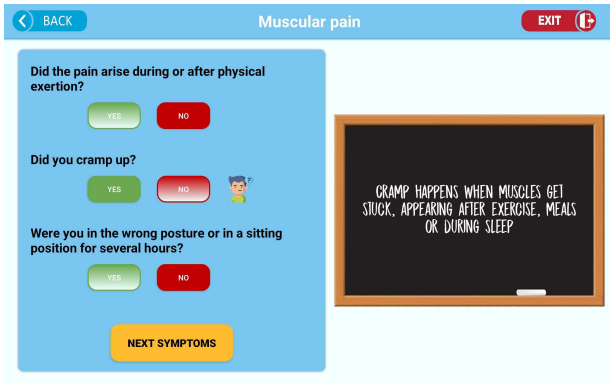



Figure 69: Main complaint exploration screen

Replace the all capitals case to sentence case and increase font size in the main complaint buttons in order to allow for faster reading, given that reading speed is slowed about 13 to 20 per cent when text is set in all capital letters [44].

The progress bar suggested by some users would allow a step-by-step checkout flow, breaking down a complex checkout task into bite-sized chunks, each requiring user small user actions [45]. This subdivision of interface tasks provides constant information to the user about the progress in the overall interface process, allowing him to envision and estimate the end of the kiosk session, an aspect that motivates users and increases the number of sessions carried out to the end.

The internal consistency in the clinical history and daily medication was suggested by users. This is important, to create patterns in language, layout and design throughout the app to help facilitate efficiency. Once a user learns how to do something, they should be able to transfer that skill to other parts of the app [46]. This way keeping the same layout and design on the “continue” buttons on these screens will make it easier for the user to learn.

In the medical history, the most common health problems in Portugal are presented for selection, instead of text boxes in which the user would have to typethehealth problems. This choice is because when a patient comes to a doctor’s appointment and the doctor questions him/her about his/her health problems he/she answers that he/she does not present any. This answer from the patient is because they do not remember their health problems or do

not consider that they have a health problem even if they are taking medication associated with the problem. Showing the user possible health problems will remind them of their problems, allowing them not to skip this stage.

In the usual medication, a dropdown has been added to the text box where the user enters the name of their medication to avoid mistakes and help you remember the name of your medication. However, we realize that the dropdown has not been able to help participants with a higher age range and low levels of education. The possible reason for this will be the lack of digital competence and not being used to using this type of elements. Simpler and more common elements such as buttons may be a more feasible option for these cases.

During the exploration of the main complaint, several questions are asked to the user regarding the main complaint. Ascertain medical terms are not common knowledge in society, it is important that the questions asked to the user are rewritten in a simple way with a common vocabulary. When questions, even rewritten, may cause the user to have doubts, the user should be given the opportunity to know more about the terms in doubt, through a simple click and an AlertDialog, for example.

### 6.1.4 *System improvement*

The system improvements presented in this section were directly drawn from the analysis of the obtained results and they amount both to the digital interface as well as to the medical sensor devices. Table. 8 summarizes the differences between the current system and future system improvements. One such improvement involves the use of a citizen card reader to extract personal information quickly, thus replacing the need for the user to enter this data.

The clinical history screens will be reduced to one, allowing the user to view and select all possible health problems they may have, and taking him immediately to the daily medication phase. The implementation of this improvement demands a larger tablet for the necessary visualization and organization of the information.

In the daily medication, the writing of the name of the medication will undergo a process of correction and automatic search without the use of a dropdown. Dose writing will be replaced by a set of buttons associated with the various possible doses of each drug, so that less proficient users understand the information they are asked for. The parts of the day will be associated with the description as a part of the button.

On the device screens, instructions will be displayed at the same time as the measurement, so that the user has always access to the next step to take, thus reducing errors of use due to forgetfulness of steps during measurement. The blood pressure monitor will also be changed to a device that reduces measurement errors.

To allow a larger number of participants to receive their measurement results, the possibility of sending an SMS) will also be added.

The device's keyboard will also be replaced by a simpler and easier to use keyboard so that users with less digital competence will not be in doubt.

Table 8: Current System Versus System Improvement

| Current System | System Improvement |
|---|---|
| Manual entry of the personal information | Extract personal information by Citizen Card |
| Smaller tablet | Larger tablet |
| Four clinical history screens | Only one clinical history screen |
| Manual medication search | Automatic medication search and medication names correction |
| Manual dose entry | Select dose button |
| Device instructions before measurement | Device instructions during measurement |
| MyTensio Wrist BW-BW1Model | A new one with less measurement errors |
| Email measurement results | Email and SMS measurement results |
| Tablet keyboard | Simple and easier to use keyboard |

## 6.2 SECOND MOMENT OF EVALUATION

After obtaining feedback from the evaluations of the first prototype, some features of the Android platform were corrected and improved, the user interface changed and the web platform implemented. This evaluation moment aims to evaluate these components in a real environment, and comprises several phases that can be consulted in the testing protocol established with CHUSJ, which is in Appendix section E. To encourage patients to use the kiosk, nurses will offer a flyer during triage and encourage patients to take advantage of the kiosk during the waiting time. The flyer can be found in Section D of the Appendix.
Physicians who will evaluate the platform are provided with the hyperlink and password before the consultation so that they can use the application on the office computer. They are also provided with a poster explaining the platform's features. The poster is shown in Section D of the Appendix.

### 6.2.1 *Android application testing methodology*

The evaluation of the kiosk's usability was carried out taking into account the same indicators as in the previous version. However, another method, method (D), was added to evaluate user interaction with the device. To obtain the indicators intended, the usability test was divided into four parts: (A) kiosk usage time, (B) final questionnaire, (C) observation report, and (D) coordinates of the user clicks. The methods (A) and (C) follows the same methodology used in the previous version.

*B. Questionnaire*

Similar to the previous version, patients were invited to conduct a system usability questionnaire at the end of kiosk session. This questionnaire is essentially made up of questions to assess the functionalities of the application, error messages quality and help information quality.
The response scale varies between 4 response options (agree, neither agree nor disagree, disagree, not applicable). This scale was reduced from the previous version to obtain more concise and accurate answers. Since all questions are mandatory, the possibility of "not applicable" answers for questions not applicable to the user was added.
Some nurses are asked to use the kiosk and also to complete a questionnaire. The nurses as health professionals could providing us with a professional point of view about the system, which the patient could not transmit. The questionnaire conducted to the nurse differs from the questionnaire conducted to the patient in only one question.
Questionnaires to patients and nurses in the emergency department are presented in the Section C of Appendix.

*D. Coordinates of the User Clicks*

The kiosk application was instrumented to log clicks made by the patient in order to check whether actual user interaction is done according to the design expectations. This allows not only to check if users have any problems with the application but also to group all clicks made by the patients to evaluate whether the flow of the application is simple and if there are interactions that should not exist or were not envisioned. These interactions reflect the frustrations and difficulties felt by the patient regarding some interface elements.
By grouping all the clicks made by the patients on each screen, it's possible to evaluate the usability of that screen. For example, if patients try to interact with unintended parts of the interface, changes could be performed to fix that issue, improving the flow of the application.

### 6.2.2 *Web application testing methodology*

The evaluation of the web platform usability was carried out taking into account the following indicators: (I) effectiveness evaluation of the interface, (II) interface design, and (III) interaction with the system. To obtain these indicators is conducted a questionnaire to emergency department physicians who used the web platform during their consultations.

*Questionnaire*

After using the web platform in consultations, physicians are invited to conduct a questionnaire about the system. This questionnaire is divided into two parts. The first part evaluates the effectiveness of the interface, i.e. whether the application correctly stored the data entered by patients at the kiosk. The second part contains questions to assess web platform usage satisfaction. This questionnaire also contains questions adapted from the PSSUQ.
The response scale varies between 4 response options (agree, neither agree nor disagree, disagree, not applicable ). Since all questions are mandatory, the possibility of "not applicable" answers for questions not applicable to the user was added.
Questionnaire to the physicians in the emergency department is presented in the Section C of Appendix.

# 7

# CONCLUSION

After the scope of the project and the problem it aims to solve have been presented, and after reporting the conceptualization, implementation and evaluation of the proposed solution, it is then possible to summarize the dissertation and draw the main conclusions about the work carried out.

## 7.1 CONCLUSIONS

Crowding in emergency departments has increased over the years due to inappropriate visits by low clinical severity patients, negatively affecting the delivery of emergency services and increasing the waiting time for patients to be seen by a physician. This project proposes a kiosk-based solution which aims to improve both the time and quality of the clinical consultation performed by ED medical staff and thus improve the response capacity of the ED with the same resources.

The state of the art study provides a comprehensive overview of the most relevant accomplishments in the hospital self-service kiosks field to improve clinical care on health services. The study revealed application areas where kiosks, like the one proposed in this dissertation, can provide suitable solutions that haven't been explored. The high levels of kiosks acceptance and satisfaction observed, points to a relevant opportunity for the introduction of self-service kiosks in several healthcare contexts.

The specification of the solution was done considering the defined objectives and all functional and non-functional requirements of the system were identified. Based on the requirements, the solution architecture was proposed, a client-server architecture. The implementation of the proposed solution is divided into five parts: database, web-services, android application, web application, data warehouse.

The database follows the relational model, and a transaction was defined to ensure the consistency of each clinical register insertion into the database. The Web services follow the RESTful architecture and each one is responsible for dealing with part of the requests that the clients perform. Two prototypes of the android application have been developed, the first one aimed at testing the first user interface designed and to identify difficulties and

problems using the sensors.
Observations taken from testing the first prototype led to the development of a new prototype. The functionalities of the web application were implemented using the JavaScript React library. Requests not coming from Portugal or requests that may constitute a threat to the system are blocked. The data warehouse was modelled as a star scheme.
The usability evaluation of the first prototype of the self-service kiosk allowed to understand the aspects that should be improved in this first tested prototype. The heterogeneity of participants allowed to understand in a more global way the difficulties felt during the use of the system. In general, there was positive acceptability by participants, with a large majority considering the system user-friendly. All participants were able to complete a kiosk session from start to finish without any help, with the exception of the screens of the devices for collecting vital signs, for which one it was found a series of improvements based on the carried analysis. The time for a full kiosk session was less than expected, even though it is necessary to take into account that the kiosk was not tested in a real ED.
Unfortunately, it was not possible to carry out the solution evaluation in a hospital setting with patients due to the pandemic situation. The primary purpose of this solution has not been verified and the results of the usability assessment were not obtained. As it was not possible to carry out the data collection, the intelligent system that adapts the interface to each patient's digital skills was not created.

## 7.2 Prospect for future work

Although it was not possible to test the solution in a real environment, there are some suggestions for future work, namely:

- Develop an Android and iOS application for smartphones that allows patients to enter the same data requested by the kiosk.
- Use an API that provides the medicines and their details always updated. In this dissertation, the database has to be populated manually with all medicines, which requires a periodic re-population to refresh them.
- Creating a decision support diagnostic system to help physicians to achieve more accurate diagnoses.
- The ED can be visited by the same patient more than once, so the self-service kiosk should present the patient's history from previous visits, avoiding that the patient has to select again the same health problems or usual medication for example.
- Carry out the second moment of the idealized evaluation;

- Since patients of different ages and digital competence are expected, the creation of a UI/UX system that intelligently adapts to the user is fundamental.
- The web platform could also present a dashboard page of each patient's history of anamnesis information so that the physician could view any past patient information more quickly and intuitively and get support in determining the patient's diagnosis.
- The determination of patterns between health problems, medicines and demographic indicators through data mining techniques could allow health problems to be suggested to the patient for selection through their demographic data and the selected medication.
- A system of main complaint suggestions would also facilitate the patient's interaction with the kiosk application. The determination of patterns between major complaints and past medical history, medication, vital signs, and demographic data through data mining techniques may allow the creation of this system.
- Replacing punctual measurements in the self-service kiosk with continuous monitoring of the patient's vital signs from the time the patient starts the kiosk's session until leaving the ED, providing physicians with the relevant biosignals data of each patient in real-time. This continuous monitoring would inform the physicians of possible worsening or improvement of the health status of the patients during their stay in the ED. The use of a smart all-in-one device to measure vital signs can be a solution for this continuous monitoring.
- The implementation of a solution that allows interoperability between the hospital system and the patient care platform will certainly facilitate the interchange of clinical information of patients between systems. This interoperability will also provide the possibility for the patient care platform to have access to patients' past clinical register.
- The creation of a modular self-service kiosk is fundamental since it can be easily adapted to different use cases, allowing the clinical staff to choose which functionalities the self-service kiosk should have at any given time.

## 7.3 CONTRIBUTION

The Master´s work contributes with:

- A systematic review about health kiosks for patient care;
- The implementation of an Android application capable of collecting vital signs (temperature, blood pressure, blood oxygen saturation), past medical history, usual medication, main complaint as well as reliefs and aggravation factors;
- The implementation of a Web application that provides to physicians the above mention data, in a structured and uniform way, before each medical check-up;
- The implementation of a back-end system to respond to requests made by the applications;
- The implementation of a data warehouse for reporting and data analysis of the patients clinical data;
- A well-defined Test Protocol to test the solution developed in an emergency department environment;

Additionally, this Master´s work also has scientific outcomes, namely a conference publication and the submission of a systematic review to the International Journal of Medical Informatics with the following references, respectively:

- Pacheco, P., Santos, F., Coimbra, J., Oliveira, E., Rodrigues, N. F. (2020, August). Designing Effective User Interface Experiences for a Self-Service Kiosk to Reduce Emergency Department Crowding. In 2020 IEEE 8th International Conference on Serious Games and Applications for Health (SeGAH) (pp. 1-8). IEEE.
- Pacheco, P., Santos, F., Coimbra, J., Oliveira, E., Rodrigues, N. F. The role of kiosks on health services: a systematic review (to be accepted)

# BIBLIOGRAPHY


[1] Thomas Bodenheimer and Hoangmai H Pham. Primary care: current problems and proposed solutions. *Health Affairs*, 29(5):799–805, 2010.

[2] Roger A Rosenblatt and L Gary Hart. Physicians and rural america. *Western Journal of Medicine*, 173(5):348, 2000.

[3] Jesse M Pines and Richard T Griffey. What we have learned from a decade of ed crowding research. *Academic Emergency Medicine*, 22(8):985–987, 2015.

[4] Crowding, american college of emergency physicians. policy statement: crowding [internet] [accessed on 21 september 2019]. 2013. http://www.acep.org/patient-care/policy-statements/crowding/.

[5] Peter McKenna, Samita M Heslin, Peter Viccellio, William K Mallon, Cristina Hernandez, and Eric J Morley. Emergency department and hospital crowding: causes, consequences, and cures. *Clinical and experimental emergency medicine*, 2019.

[6] Jesse M Pines, Joshua A Hilton, Ellen J Weber, Annechien J Alkemade, Hasan Al Shabanah, Philip D Anderson, Michael Bernhard, Alessio Bertini, André Gries, Santiago Ferrandiz, et al. International perspectives on emergency department crowding. *Academic Emergency Medicine*, 18(12):1358–1370, 2011.

[7] Ajay Tandon, Christopher JL Murray, Jeremy A Lauer, and David B Evans. Measuring overall health system performance for 191 countries. *Geneva: World Health Organization*, 2000.

[8] Sofia Pereira, António Oliveira e Silva, Manuel Quintas, Jorge Almeida, Cristina Marujo, Manuel Pizarro, Vítor Angélico, Luisa Fonseca, Eunice Loureiro, Sofia Barroso, et al. Appropriateness of emergency department visits in a portuguese university hospital. *Annals of emergency medicine*, 37(6):580–586, 2001.

[9] Simoes J de Almeida, Gonçalo Figueiredo Augusto, I Fronteira, and C Hernández-Quevedo. Portugal: health system review. *Health systems in transition*, 19(2):1, 2017.

[10] Philip McHale, Sara Wood, Karen Hughes, Mark A Bellis, Ulf Demnitz, and Sacha Wyke. Who uses emergency departments inappropriately and when-a national cross-sectional study using a monitoring data system. *BMC medicine*, 11(1):258, 2013.

[11] João Silva, Pedro Brandão, and Rui Prior. Usability assessment of a health kiosk. pages 260–265, 2017.

[12] Ashish Joshi and Kate Trout. The role of health information kiosks in diverse settings: a systematic review. *Health Information & Libraries Journal*, 31(4):254–273, 2014.

[13] Grace Ng, Sze Wee Tan, and Ngiap Chuan Tan. Health outcomes of patients with chronic disease managed with a healthcare kiosk in primary care: protocol for a pilot randomised controlled trial. *BMJ open*, 8(3):e020265, 2018.

[14] Web services glossary [internet] [accessed on 21 october 2020]. https://www.w3.org/TR/2004/NOTE-ws-gloss-20040211/#webservice.

[15] Xinyang Feng, Jianjing Shen, and Ying Fan. Rest: An alternative to rpc for web services architecture. pages 7–10, 2009.

[16] Introducing json [internet] [accessed on 21 october 2020]. *JSON*. https://www.json.org/json-en.html.

[17] Mobile operating system market share worldwide [internet] [accessed on 21 october 2020]. *StatCounter Global Stats*. https://gs.statcounter.com/os-market-share/mobile/worldwide.

[18] Firebase realtime database [internet] [accessed on 21 october 2020]. *Google*. https://firebase.google.com/docs/database?hl=pt.

[19] Platform architecture : Android developers [internet] [accessed on 21 october 2020]. *Android Developers*. https://developer.android.com/guide/platform.

[20] Understand the activity lifecycle : Android developers [internet] [accessed on 21 october 2020]. *Android Developers*. https://developer.android.com/guide/components/activities/activity-lifecycle.

[21] Fragments : Android developers [internet] [accessed on 21 october 2020]. *Android Developers*. https://developer.android.com/guide/components/fragments.

[22] React – a javascript library for building user interfaces [internet] [accessed on 21 october 2020]. *– A JavaScript library for building user interfaces*. https://reactjs.org/.

[23] Virtual dom and internals [internet] [accessed on 21 october 2020]. *React*. https://reactjs.org/docs/faq-internals.html#what-is-the-virtual-dom.

[24] [internet] [accessed on 21 october 2020]. *Introduction to Node.js*. https://nodejs.dev/learn/introduction-to-nodejs.

[25] Tony R Sahama and Peter R Croll. A data warehouse architecture for clinical data warehousing. 2007.

[26] What is a data warehouse? [internet] [accessed on 21 october 2020]. *What Is a Data Warehouse | Oracle Portugal*. https://www.oracle.com/pt/database/what-is-a-data-warehouse/.

[27] Ralph Kimball and Margy Ross. The data warehouse toolkit: The definitive guide to dimensional modeling. 2013.

[28] Guiqin Sun, Shenyi Tao, Yongqiang Lu, Yu Chen, Yuanchun Shi, Ni Rong, Rui Wang, and Xiaojuan Lu. A low-cost community healthcare kiosk. pages 270–273, 2011.

[29] Robert G Maunder and Jonathan J Hunter. An internet resource for self-assessment of mental health and health behavior: Development and implementation of the self-assessment kiosk. *JMIR mental health*, 5(2):e39, 2018.

[30] Eduardo Soares, Cristina Oliveira, João Maia, Rafael Almeida, Miguel Coimbra, Pedro Brandão, and Rui Prior. Modular health kiosk for health self-assessment. pages 278–280, 2016.

[31] Juliana Bahadin, Eugene Shum, Grace Ng, Nicolette Tan, Pushpavalli Sellayah, and Sze Wee Tan. Follow-up consultation through a healthcare kiosk for patients with stable chronic disease in a primary care setting: a prospective study. *Journal of general internal medicine*, 32(5):534–539, 2017.

[32] Natalie Coyle, Andrew Kennedy, Michael J Schull, Alex Kiss, Darren Hefferon, Paul Sinclair, and Zuhair Alsharafi. The use of a self-check-in kiosk for early patient identification and queuing in the emergency department. *Canadian Journal of Emergency Medicine*, pages 1–4, 2019.

[33] Ho Seok Ahn, I-Han Kuo, Chandan Datta, Rebecca Stafford, Ngaire Kerse, Kathy Peri, Elizabeth Broadbent, and Bruce A MacDonald. Design of a kiosk type healthcare robot system for older people in private and public places. pages 578–589, 2014.

[34] George Demiris, Hilaire J Thompson, Blaine Reeder, Katarzyna Wilamowska, and Oleg Zaslavsky. Using informatics to capture older adults' wellness. *International journal of medical informatics*, 82(11):e232–e241, 2013.

[35] Chia-Fang Chung, Sean A Munson, Matthew J Thompson, Laura-Mae Baldwin, Jeffrey Kaplan, Randall Cline, and Beverly B Green. Implementation of a new kiosk technology for blood pressure management in a family medicine clinic: from the wwami region practice and research network. *The Journal of the American Board of Family Medicine*, 29(5):620–629, 2016.

[36] Interaction design basics [internet] [accessed on 21 october 2020]. *Usability.gov*, 2014. http://www.usability.gov/what-and-why/interaction-design.html.

[37] R Annes Gladia and G Kavya. Design and development of non-invasive kiosk for self-care health management. *Glob. J. Pure Appl. Math*, 13:4787–4794, 2017.

[38] Cloud storage | firebase [internet] [accessed on 21 october 2020]. *Google*. https://firebase.google.com/docs/storage?hl=pt.

[39] Firebase realtime database [internet] [accessed on 21 october 2020]. *Google*. https://firebase.google.com/docs/database?hl=pt.

[40] Annette De Vito Dabbs, Brad A Myers, Kenneth R Mc Curry, Jacqueline Dunbar-Jacob, Robert P Hawkins, Alex Begey, and Mary Amanda Dew. User-centered design and interactive health technologies for patients. *Computers, informatics, nursing: CIN*, 27(3):175, 2009.

[41] Interaction design basics [internet] [accessed on 21 october 2020]. *Usability.gov*, Feb 2014. http://www.usability.gov/what-and-why/interaction-design.html.

[42] [internet] [accessed on 21 october 2020]. *Using Light, Color and Contrast Effectively in UI Design*. https://usabilitypost.com/2008/08/14/using-light-color-and-contrast-effectively-in-ui-design/.

[43] Effective visual communication for graphical user interfaces [internet] [accessed on 21 october 2020]. *Designing Effective User Interfaces*. http://web.cs.wpi.edu/~matt/courses/cs563/talks/smartin/int_design.html.

[44] Saadia Minhas. All caps on ui: Good or bad? [internet] [accessed on 21 october 2020]. Aug 2020. https://blog.prototypr.io/all-caps-on-ui-good-or-bad-2570f14dc457?gi=29b6d8257d5.

[45] A comprehensive guide to mobile app design [internet] [accessed on 21 october 2020]. *Smashing Magazine*, Feb 2018. https://www.smashingmagazine.com/2018/02/comprehensive-guide-to-mobile-app-design/.

[46] User interface design basics [internet] [accessed on 21 october 2020]. *Usability.gov*, May 2014. http://www.usability.gov/what-and-why/user-interface-design.html.

[47] Jacinta Santos. Patient care platform. Master's thesis, Universidade do Minho, a78363@alunos.uminho.pt, 2 2020. to be submitted.

# A

# KIOSK REQUIREMENTS

In this appendix, the functional and non-functional requirements of the kiosk are presented.

## A.1 FUNCTIONAL REQUIREMENTS

**Requirement no. :** 1
**Requirement type:** Functional.
**Use case:** Register identification data.
**Description:** The patient must be able to register the identification data.
**Reason:** As this system collects a set of information on the health of each patient, they must register their identification.

**Requirement no. :** 2
**Requirement type:** Functional.
**Use case:** Register past medical history.
**Description:** The patient should be able to record the past medical history.
**Reason:** As the clinical history can have an impact on clinical diagnosis, it is essential to record the past medical history.

**Requirement no. :** 3
**Requirement type:** Functional.
**Use case:** Register usual medication.
**Description:** The patient should be able to register the usual medication.
**Reason:** The usual medication of patients may affect the prescription of medicines.

**Requirement no. :** 4
**Requirement type:** Functional.
**Use case:** Register main complaint.
**Description:** The patient should be able to register the chief complaint felt at the moment.
**Reason:** The main complaint can help in identifying the health problem.

**Requirement no. :** 5
**Requirement type:** Functional.
**Use case:** Gather biometric data.
**Description:** The patient should be able to collect the biometric data.
**Reason:** As the collection of biometric data is one of the goals of the system, they must be able to do so.

**Requirement no. :** 6
**Requirement type:** Functional.
**Use case:** Gather biometric data.
**Description:** The patient should be able to measure the blood pressure.
**Reason:** Blood pressure is a vital sign, and knowing its value is essential to physicians.

**Requirement no. :** 7
**Requirement type:** Functional.
**Use case:** Gather biometric data.
**Description:** The patient should be able to take the body temperature.
**Reason:** Temperature is a vital sign, and knowing its value is important to physicians.

**Requirement no. :** 8
**Requirement type:** Functional.
**Use case:** Gather biometric data.
**Description:** The patient should be able to measurethe heart rate.
**Reason:** Heart rate is a vital sign, and knowing its value is essential to physicians.

**Requirement no. :** 9
**Requirement type:** Functional.
**Use case:** Gather biometric data.
**Description:** The patient should be able to measure the blood oxygen saturation level.
**Reason:** Since oxygen saturation in the blood is a clinical indicator, knowing its value is essential to physicians.

**Requirement no. :** 10
**Requirement type:** Functional.
**Use case:** End Session.
**Description:** The patient may end the kiosk session at any time.
**Reason:** If the patient withdraws, the data recorded so far must be deleted.

## A.2 NON-FUNCTIONAL REQUIREMENTS

In this section, the non-functional kiosk requirements collected are presented, each describing a restriction imposed on the system.

*Appearance requirements*

**Requirement no. :** 11
**Requirement type:** Appearance requirement.
**Description:** The application should display images and videos referring to the use of the sensors.
**Reason:** Show patients the correct way to use biometric sensors.

**Requirement no. :** 12
**Requirement type:** Appearance Requirement.
**Description:** The system must adapt the different graphical components of the interfaces to each patient's digital competence.
**Reason:** The system, by adapting the different interface components for each patient, allows improving the interaction between the patient and the application.

*Usability requirements*

**Requirement no. :** 13
**Requirement type:** Usability requirement.
**Description:** The system shall use terms appropriate to the context in which the system operates.
**Reason:** The system must use terms and vocabulary appropriate to the context in which it is inserted to facilitate its understanding by patients.

**Requirement no. :** 14
**Requirement type:** Usability requirement.
**Description:** The system should be easy to use.
**Reason:** The system should be user-friendly for all patients.

*Performance requirements*

**Requirement no. :** 15
**Requirement type:** Performance requirement.
**Description:** The process of collecting biometric data should be quick.
**Reason:** The collection process should be fast to reduce the effort exerted on the patient.

**Requirement no. :** 16
**Requirement type:** Performance requirement.
**Description:** The system should provide answers to patient requests in a fast way.
**Reason:** The interaction of the patient with the system should be smooth so that the use of the system is pleasant.

**Requirement no. :** 17
**Requirement type:** Performance requirement.
**Description:** The system should always be available.
**Reason:** The system should always be available so as not to impede its use by patients.

*Security requirements*

**Requirement no. :** 18
**Requirement type:** Security requirement.
**Description:** The application should protect information about patients.
**Reason:** It is essential that patients feel secure about the privacy of their data.

*Legal requirements*

**Requirement no. :** 19
**Requirement type:** Legal requirement.
**Description:** The patient must agree to the data privacy policy.
**Reason:** It is crucial that the patient, as well as being aware of the measures taken to protect the data, agrees to them.

# B

# WEB PLATFORM REQUIREMENTS

In this appendix, the functional and non-functional requirements of the web platform are presented.

## B.1 FUNCTIONAL REQUIREMENTS

**Requirement no. :** 1
**Requirement type:** Functional.
**Use case:** Consult the patient's clinical data.
**Description:** The physician should be able to consult a patient's clinical data.
**Reason:** Since a patient's diagnosis is dependent on their clinical data, the physician must be able to consult them.

**Requirement no. :** 2
**Requirement type:** Functional.
**Use case:** Consult the patient's clinical data.
**Description:** The physician should be able to consult the patient's medical history.
**Reason:** A patient's medical history can have an impact on their diagnosis.

**Requirement no. :** 3
**Requirement type:** Functional.
**Use case:** Consult the patient's clinical data.
**Description:** The physician should be able to consult the patient's usual medication.
**Reason:** A patient's usual medication can have an impact on the prescription.

**Requirement no. :** 4
**Requirement type:** Functional.
**Use case:** Consult the patient's clinical data.
**Description:** The physician should be able to consult the patient's main complaint.
**Reason:** The patient's main complaint may have an impact onthediagnosis.

**Requirement no. :** 5
**Requirement type:** Functional.
**Use case:** Consult the patient's clinical data.
**Description:** The physician should be able to consult the patient's blood pressure.
**Reason:** Knowledge of the patient's blood pressure can influence their diagnosis.

**Requirement no. :** 6
**Requirement type:** Functional.
**Use case:** Consult the patient's clinical data.
**Description:** The physician should be able to consult the patient's temperature.
**Reason:** Knowledge of the patient's temperature can influence their diagnosis.

**Requirement no. :** 7
**Requirement type:** Functional.
**Use case:** Consult the patient's clinical data.
**Description:** The physician should be able to consult the patient's heart rate.
**Reason:** Knowledge of the patient's heart rate can influence their diagnosis.

**Requirement no. :** 8
**Requirement type:** Functional.
**Use case:** Consult the patient's clinical data.
**Description:** The physician should be able to consult the patient's blood oxygen saturation level.
**Reason:** Knowledge of the patient's blood oxygen saturation level may influence their diagnosis.

**Requirement no. :** 9
**Requirement type:** Functional.
**Use case:** Register diagnosis.
**Description:** The physician should be able to register the patient's diagnosis.
**Reason:** Diagnosis is recorded to identify patterns between patient data, collected at the kiosk, and their clinical diagnosis.

**Requirement no. :** 10
**Requirement type:** Functional.
**Use case:** Copy patient's register information.
**Description:** The physician should be able to copy patient's register information.
**Reason:** Copying the information from patients' clinical records speeds up the medical check-up.

**Requirement no. :** 11
**Requirement type:** Functional.
**Use case:** Login.
**Description:** The system should check whether the credentials entered by the user are valid.
**Reason:** Validation of the data entered by the user is necessary to avoid undue access to the system.

**Requirement no. :** 12
**Requirement type:** Functional.
**Use case:** Logout.
**Description:** A user logs out of the system.
**Reason:** When the user has no more tasks to do in the system, it may be the user's wish to log out.

**Requirement no. :** 13
**Requirement type:** Functional.
**Use case:** Patient Search.
**Description:** The physician may search the patient.
**Reason:** This action is essential so that the doctor can consult the correct patient's clinical register.

## B.2 NON-FUNCTIONAL REQUIREMENTS

In this section, the non-functional web platform requirements collected are presented, each describing a restriction imposed on the system.

*Usability requirement*

**Requirement no. :** 14
**Requirement type:** Usability requirement.
**Description:** The system shall use terms appropriate to the context in which the system operates.
**Reason:** The system must use terms and vocabulary appropriate to the context in which it is inserted to facilitate its understanding by physicians.

**Requirement no. :** 15
**Requirement type:** Usability requirement.
**Description:** The system should be easy to use.
**Reason:** The system should be user-friendly for all physicians.

*Performance requirements*

**Requirement no. :** 16
**Requirement type:** Performance requirement.
**Description:** The system should provide answers to physician requests in a fast way.
**Reason:** The interaction of the physician with the system should be smooth so that the use of the system is pleasant.

**Requirement no. :** 17
**Requirement type:** Performance requirement.
**Description:** The system should always be available.
**Reason:** The system should always be available so as not to impede its use by physicians.

*Security requirements*

**Requirement no. :** 18
**Requirement type:** Security requirement.
**Description:** The application should protect information about patients.
**Reason:** It is essential that patients feel secure about the privacy of their data.

*Legal requirements*

**Requirement no. :** 19
**Requirement type:** Legal requirement.
**Description:** The system must comply with the General Data Protection Regulations, in force since 25 May 2018.
**Reason:** The system must comply with current laws.

# C

# QUESTIONNAIRES

In this appendix, all the questionnaires created for the self-service kiosk evaluation are shown.

## C.1 QUESTIONNAIRE OF THE FIRST SELF-SERVICE KIOSK PROTOTYPE

- Overall, I am satisfied with how easy it is to use this system?
- It was simple to use this system?
- I was able to complete the tasks and scenarios quickly using this system?
- I felt comfortable using this system?
- It was easy to learn to use this system?
- I believe I could become productive quickly using this system?
- The system gave error messages that clearly told me how to fix problems?
- Whenever I made a mistake using the system, I could recover easily and quickly?
- The information (such as help, on-screen messages, and other documentation) provided with this system was clear?
- It was easy to find the information I needed?
- The information was effective in helping me complete the tasks and scenarios?
- The organization of information on the system screens was clear?
- The interface of this system was pleasant?
- I liked using the interface of this system?
- This system has all the functions and capabilities I expect it to have?
- Overall, I am satisfied with this system?

## C.2 QUESTIONNAIRES OF THE SECOND SELF-SERVICE KIOSK PROTOTYPE

In this section two questionnaires will be described for two different types of users.

### C.2.1 *Patient Questionnaire*

- Do I believe that the use of the citizen card was a good way to collect personal information?
- Did the response history help me to remember and edit the answers given?
- Did the selection of a body part help me identify my main complaint more quickly?

- The measurement through step-by-step videos, helped me to make the measurement?
- I believe that having assistance is indispensable?
- Overall, am I satisfied with the ease of use of this system?
- I was able to complete the tasks quickly?
- It was easy to learn to use this system?
- The system gave error messages that clearly told me how to fix problems?
- Whenever I made a mistake using the system, I could recover easily and quickly?
- The information (such as help, on-screen messages, and other documentation) provided with this system was clear?
- Overall, I am satisfied with this system?

### C.2.2 *Nurse Questionnaire*

- Do I believe that the use of the citizen card was a good way to collect personal information?
- Did the response history help me to remember and edit the answers given?
- Did the selection of a body part help me identify my main complaint more quickly?
- The measurement through step-by-step videos, helped me to make the measurement?
- I believe that having assistance is indispensable?
- Do I believe that the information collected was sufficient?
- Overall, I am satisfied with how easy it is to use this system?
- I was able to complete the tasks quickly?
- It was easy to learn to use this system?
- The system gave error messages that clearly told me how to fix problems?
- Whenever I made a mistake using the system, I could recover easily and quickly?
- The information (such as help, on-screen messages, and other documentation) provided with this system was clear?
- Overall, I am satisfied with this system?

## C.3 Questionnaire of the Web Platform Prototype

In this section a two-part questionnaire will be described, which will be answered by physicians.

### C.3.1 *Evaluation Effectiveness - First Part*

- Have all health problems been registered by the patient (If the patient only registered A and did not register B)?
- Have all health problems been registered correctly (If the patient registered A and should have registered B )?
- Has all the usual medication been registered by the patient (If the patient only registered A and did not register B)?

- Has all the usual medication been registered correctly (If the patient registered A and should have registered B )?
- Were all measurement values reliable (No major discrepancies between measurements at the kiosk and measurements in the query)?
- Were the responses from the exploration of the main complaint similar to the responses given in the consultation?

### C.3.2 *Satisfaction - Second Part*

- Was it easy to research the patients?
- Has the patient's usual medication been clearly presented?
- Did the danger signs present in the biometric data of the patients help me to have an immediate perception of the abnormal values of the patients?
- Did the colors in the answers to the questions of the exploration of the main complaint help me to have an immediate perception of the positive and negative responses?
- Has the need to scroll become uncomfortable?
- Was the functionality of the "Copy" and "Copy all" buttons useful?
- Did the registers collected help me determine the diagnosis more quickly?
- The presence of a dropdown with several diagnoses allowed me to insert the diagnosis faster?
- Was the information collected from the patient sufficient?
- Overall, I am satisfied with how easy it is to use this system?
- It was easy to learn to use this system?
- The information provided with this system was clear?
- Overall, I am satisfied with this system?

# D

## FLYER AND POSTER

This appendix shows the flyer designed for patients, Figures 70 and 71, and the poster created for physicians, Figure 72.

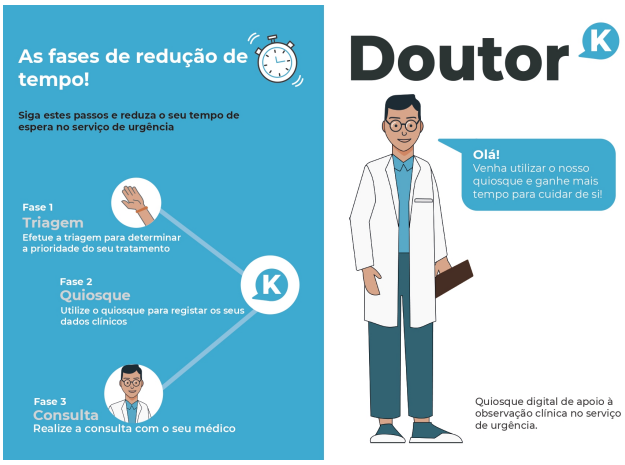



Figure 70: Flyer to help patients (front)

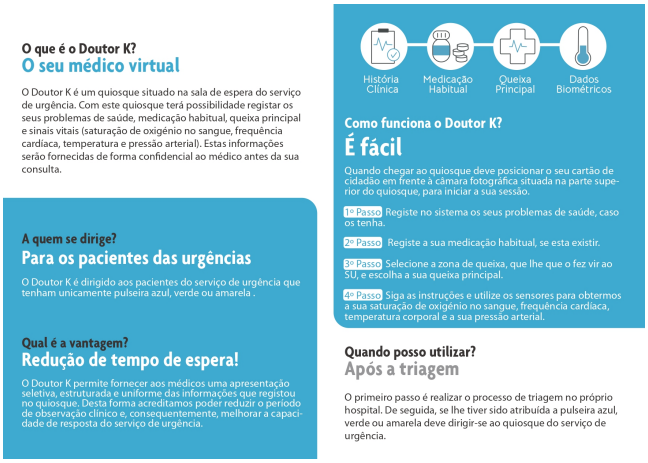



Figure 71: Flyer to help patients (back)

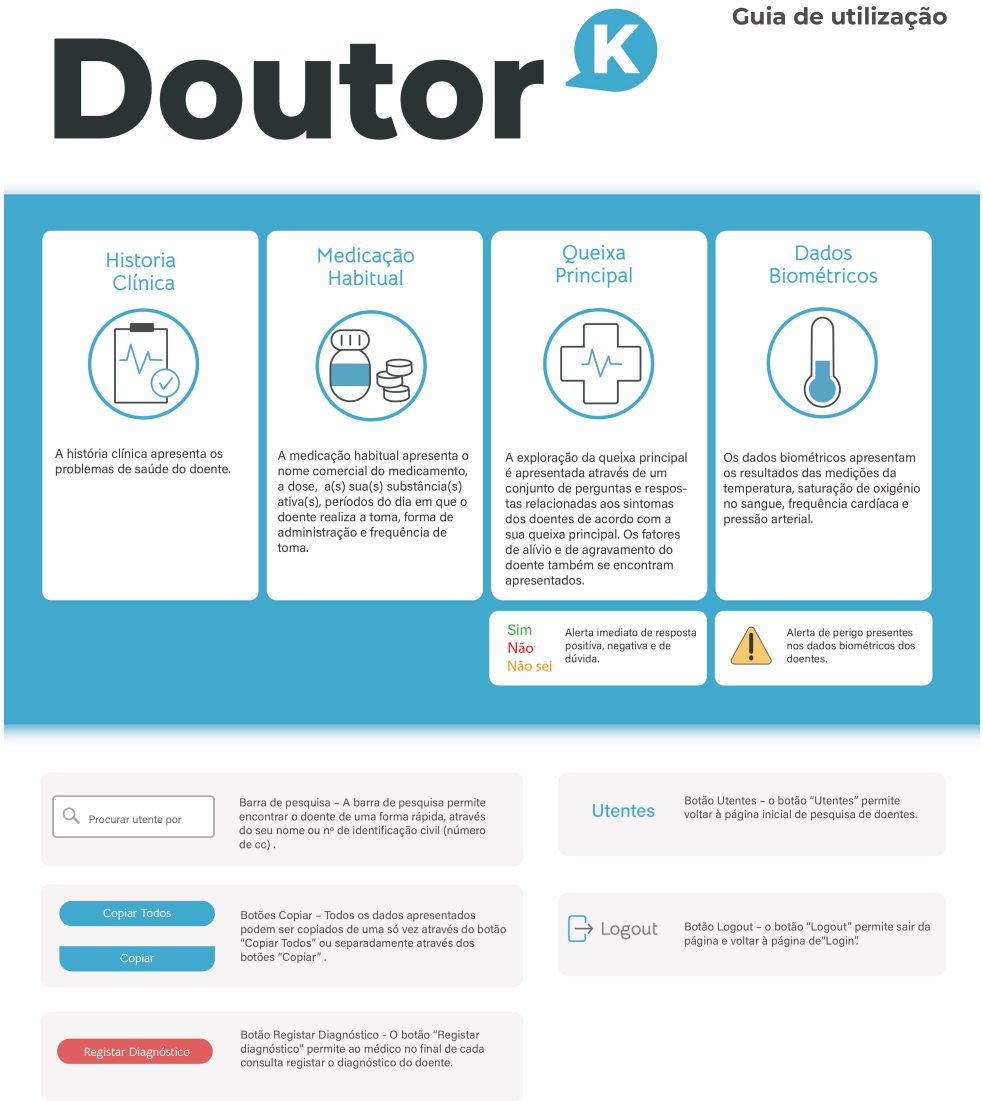



Figure 72: Poster to support physicians

# E

# TEST PROTOCOL

In this appendix, the test protocol is presented.

## E.1 INTRODUCTION

According to the American College of Emergency Physicians, crowding in the emergency department (ED) occurs when the need for emergency services exceeds the resources available for the care of patients in the ED, hospital or both. In general, this phenomenon negatively affects the provision of emergency services, impairing the quality of health care and, consequently, the clinical outcomes of patients. When capacity is high, the time patients wait to see a physician tends to increase, as does the number of patients leaving without medical advice. One of the main reasons for crowding in the ED is the high number of non-urgent or inappropriate patient presentations.

Emerging medical devices open up the possibility of setting up systems that can help collect health information from patients in hospitals or health centres. Examples of such systems are self-service kiosks, independent units that contain computer programs and provide services to patients. In general, these systems are largely portable, interactive and have simple user interfaces such as a touch screen or keyboard. A study by Grace Ng showed that by placing kiosks in waiting rooms, patients were able to use the time between registration and consultation with the physician to engage in productive tasks that help the medical process. This technological solution aims to improve both the time and the quality of the anamnesis procedure performed by healthcare professionals in the ED by introducing a self-service kiosk placed in the ED waiting room. Information on medical history, medication, main complaint, symptoms and vital signs collected by patients through the kiosk will be made available to the physician before each clinical observation. The hypothesis considered is that by providing a selective, structured and uniform presentation of the anamnesis information of each patient, the physician's observation can proceed much faster and more accurately, focusing on confirmação of the most relevant aspects. The main objective of this solution is to reduce the period of clinical observation of patients and thus improve the response capacity of the ED with the same resources.

## E.2 PROJECT DESCRIPTION

This platform, will be provided by the Universidade do Minho, consisting of a set of sensors (thermometer, finger oximeter and pulse blood pressure monitor) from Bewell Connect, a keyboard, a TBee Box, a webcam and a touchscreen monitor.

The patient will access the kiosk via an Android application, where various questions will be asked in accordance with the initial clinical observation procedure carried out by the physicians (anamnesis). Thus, the system integrated in the kiosk should allow the quick registration of this clinical data in an intuitive way, so that the patient is able to use the application independently and without prior training.

It is envisaged that the entire test phase will be carried out within a period of two months.

## E.3 TEST PHASE I

The test aims to identify:

- The effectiveness of the interface, to record correct and complete patient information. In emergency consultations, the physician will carry out the anamnesis process and compare the patient's responses with those recorded in the system. The result of this evaluation will be obtained through a questionnaire to the physician;
- The way how the records collected by the kiosk are presented to physicians. This will be identified through a questionnaire to physicians;
- The impact of the use of the platform in the clinical observation process in obtaining the patient's diagnosis. This impact will be determined through a questionnaire to the physicians, where it is intended to understand whether the system has helped in obtaining the patient's diagnosis;
- A set of attributes:
  - Age range;
  - Gender;
  - Educational level;
  - Nationality
  - Digital competence;
  - Time of a full kiosk session;
  - Time per interface;
  - Number of clicks in unexpected locations per interface;
  - Number of times the patient needed to use the doubt icon on the main complaint exploration interface;
  - Values in blood oxygen saturation, temperature, heart rate and blood pressure measurements;
  - Using the history for navigation or back button;
  - Give-ups in the use of each sensor;
  - Patient questionnaire questions;
  - Need of assistance for each screen;
  - General difficulties, taken from the observation.

**How the test will work:**
During this phase, there will be only one test group, the intervention group corresponding to the patients who used the kiosk before the consultation. Patients in the intervention group will be referred by flyers, offered at triage, and by assistants to the kiosk, where they will be provided with assistance.

**Who can participate?**
All patients who, after triage in Manchester, have blue, green or yellow wristbands are eligible to participate.

**Where are trials carried out?**
This study will be carried out in the ED of the Centro Hospitalar Universitário de São João. The approval of the project was obtained by the ethics committee of the hospital.

**How long does the clinical trial last?**
Phase I trial aims to obtain 50 user records. As soon as the desired goals are identified, the platform will be later tested in Phase II.

The patient questionnaire will be taken at the end of each session at the kiosk, where a tablet with a questionnaire to fill in will be provided. The physician's questionnaire will be carried out at the end of the test phase to a set of randomly chosen physicians. In the possibility to conduct interviews, these will replace the patient and physician questionnaires, as interviews provide more detailed and reliable answers.

If some nurses use the kiosk, a specific questionnaire will be carried out for these healthcare professionals to get feedback on their use. This use by these healthcare professionals will have to take place at the beginning or end of this phase so as not to interfere with the records collected from patients. As with patient and physician questionnaires, if an interview is possible it will replace the questionnaire.

## E.4 Test Phase II

The test aims to identify:

- The same set of attributes of Phase I.
- The types of users of the kiosk. These types of users will be identified using semi-supervised learning;
- The impact of platform use on the clinical observation process. That is, if allowed:
  - Reduction of the period of clinical observation;
  - Helped in obtaining the patient's diagnosis.

  This impact will be determined through time comparisons of emergency consultations, between control and intervention groups and a questionnaire to physicians.
- The way how the records collected by the kiosk are presented to physicians to understand if they are shown in a selective, structured and uniform manner. This will be captured through a questionnaire to physicians.

The patient questionnaire will be carried out at the end of each session at the kiosk, where a tablet with its respective questionnaire will be provided for completion. The physician's questionnaire will be carried out at the end of the test phase to a set of randomly chosen physicians. This questionnaire will present only the second part, the questionnaire of satisfaction. The possibility of interviews will replace the patient and physician questionnaires, as more detailed and reliable answers will be obtained.

**How the trial will work:**
During this phase there will be two groups of tests. The control group, which corresponds to patients who will perform the emergency consultation without using the kiosk, and the intervention group, which corresponds to patients who used the kiosk before the consultation. Patients in the intervention group will be referred by flyers, offered at triage, and by assistants to the kiosk, where they will be provided with assistance.

**How long will the clinical trial last?**
Phase II aims to obtain 100 user records. As soon as the desired goals are identified, the platform will be tested later in Phase III or Phase IV.

## E.5 TEST PHASE III

This test will only be carried out if a considerable number of modifications are made.

The test aims to identify:

- The effectiveness of the interface, to record correct and complete patient information. In emergency consultations, the physician will carry out the anamnesis process and compare the patient's responses with those recorded in the system;
- The result of the effectiveness above mentioned will be obtained by means of a questionnaire to the physician;
- The way how the records collected by the kiosk are presented to physicians. This will be identified through a questionnaire to physicians;
- The evaluation of the self-adaptive interfaces, through the process of observation, questionnaire and other variables taken from the interaction of the patient with the kiosk.

The trial will operate in the same way as Phase I trial, with the aim of obtaining 50 records.

## E.6 TEST PHASE IV

Phase IV test seeks to assess:

- The self-adaptive interfaces;
- Whether the records collected by the kiosk are presented to physicians in a selective, structured and uniform manner (this evaluation will only occur if the interface of the physician's web platform is changed, between Phase I and Phase IV);
- The period of clinical observation has been reduced.

Determination of evaluations:

- Through the observation process, questionnaire and other variables taken from patient interaction with the kiosk;
- Through a questionnaire to the physicians;
- Performing a comparison of times of use of each interface and complete session of the system in relation to the previous phase;
- Through the times of each emergency consultation, it is intended to determine whether a reduction in the period of clinical observation has occurred. A time comparison of the emergency consultations between the control group and the intervention group will be carried out.

The trial will operate in the same way as Phase II trial, with the aim of obtaining 50 records.

# F

## DATA WAREHOUSE INDICATORS

This appendix describes all the indicators or queries to which the Data Warehouse should be able to respond. These indicators were provided by Jacinta Santos as part of her Master's thesis [47].

**What is the average time of use of the kiosk?**

This indicator will allow you to know whether the kiosk has been used longer or shorter than the first version of the self-service kiosk.

**Which age range and gender have the longest kiosk usage times?**

This indicator can provide us with evidence on which age ranges and gender have the greatest difficulty when using the kiosk.

**What is the age range and gender that presents the results of dangerous measurements of blood oxygen saturation, temperature, heart rate and blood pressure?**

This indicator can be used to determine which age ranges and gender should be given more attention since they present results of dangerous measurements. Monitoring vital signs in these age ranges and gender outside the hospital environment may be a possibility, preventing future health problems.

**What are the most common health problems for each age range and gender?**

This indicator becomes essential for the identification of patterns between health problems with age and gender.

**How many health problems do each age range and gender have on average?**

Determining which age range and gender has the most health problems may allow us to see which patients probably visited the ED most due to their weaknesses.

**What are the most commonly taken medications for each age range and gender?**

This indicator can be useful for hospital pharmacies, informing them of the medicines they should have in stock for each age range and gender. The identification of patterns between medicines with age and gender can also be provided through this indicator.

**How many medicines does patient take on average for each age range and gender intheusual medication?**

A high number of medicines in the usual medication can make the patient's taking of each medicine confusing and lead to forgetting to take some medicine. Determining the age ranges and gender that take a high number of medicines is essential for increased vigilance of correct compliance with the usual medication.

**What is the most common main complaint and diagnosis by age range and gender?**

This indicator will allow the identification of patterns between main complaints and diagnoses with age ranges and gender. It will also provide support to the physician in determining the diagnosis of a patient with a certain age range and gender.

**What health problems are associated with the results of dangerous measurements of blood oxygen saturation, temperature, heart rate and blood pressure?**

The association of health problems with this category of measurement results highlights which patients with certain health problems should be monitored for their vital signs to prevent aggravation of these same health problems or the appearance of others. This indicator also allows the identification of patterns between health problems with dangerous measurement results.

**What are the main complaints and diagnoses that are most associated with the results of dangerous measurements of blood oxygen saturation, temperature, heart rate and blood pressure ?**

Results of dangerous measurements can be associated with certain main complaints and diagnoses. When this association is determined, support can be offered to the physician's decision making in determining the patient's diagnosis.

**What are the health problems and the main complaints most associated with each diagnosis?**

The support for the physician's decision making in determining the patient's diagnosis is essential. The identification of patterns between health problems and the main complaint with the diagnosis can provide this support.

**What are the most common health problems, medicines, main complaints and diagnoses?**

This indicator will provide the most common health problems, medicines, main complaints and diagnoses of patients visiting the ED.

**What is the day of the week with more patients in the ED?**

This indicator will allow the hospital to know which day of the week is normally the most crowded. This contribution will help the hospital to manage the number of visits to the ED.

**Which month has the highest number of visits to the ED?**

Knowing which month is the most crowded will allow hospitals to manage the number of visits to the ED in an appropriate and timely manner.

**What is the most common main complaint and diagnosis per month?**

Determining the most common main complaint and diagnosis per month will help hospitals and physicians. Hospitals, for example, can prepare their pharmacies in those months with the stocks of medicines prescribed by those diagnoses. This indicator will also provide decision making support for the physician in determining the diagnosis.

**What is the most common level of education per age range?**

This indicator will provide information for the age ranges with the lowest level of education.

**What is the average time of use of the kiosk by age range and education level?**

This indicator makes it possible to check whether the age range and level of educational attainment influence the average time of use of the kiosk.

**What is the average time of use of the kiosk for each answer given to questions about the digital competence of the patient?**

This indicator makes it possible to check whether the digital competence of the patient influences the average time of use of the kiosk.

**Which age range and level of education has the least digital competence?**

This indicator makes it possible to check whether the age range and level of education influence the digital competence of each patient.

**Which interfaces have the most interaction time?**

This indicator allows us to see which interfaces the patient found most difficult to interact with.

**Which interfaces have the longest interaction time per age range and education level?**

Determining whether the age range and level of education are associated with the time of use of each interface is essential for the construction of interfaces adapted to each type of patient.