Umedicine: Medical Data Visualization Module

Miguel Neves Neto

Dissertation to obtain the Master’s Degree in

Computer Science Engineering

Supervisors
Supervisors: Prof. Helena Isabel de Jesus Galhardas
Prof. João Carlos Serrenho Dias Pereira
Prof. Manuel João Caneira Monteiro da Fonseca

Examination Committee
Chairperson: Paolo Romano
Supervisor: Prof. Helena Isabel de Jesus Galhardas
Member of the Committee: Prof. Daniel Jorge Viegas Gonçalves

May 2017
Resumo

Guardar informação médica dum paciente e apresentar essa informação ou a análise desta, tendo em conta os objectivos do médico, através de gráficos fáceis de compreender durante um diagnóstico, é muito importante de forma a providenciar o melhor serviço de saúde possível. Hoje em dia, existe uma grande variedade de aplicações médicas para apoiar os médicos durante as consultas. No entanto, a maioria destas não apresenta informação médica sob a forma de gráficos, o que leva a que os médicos tenham de procurar pela informação que desejam e em seguida realizar uma análise dessa informação, em vez de simplesmente olhar para um gráfico onde toda essa informação é apresentada de forma clara. Portanto, propusemos e desenvolvemos um módulo de visualização de informação médica para a aplicação Umedicine, que permite aos médicos consultar e monitorizar a evolução de vários parâmetros da saúde do paciente. Este módulo oferece duas funcionalidades: Representação gráfica da informação médica relativa ao histórico dos exames médicos; Dashboard oferecendo uma representação gráfica da evolução de vários parâmetros de saúde, permitindo ao médico criar gráficos e guardá-los no dashboard. Finalmente, validámos a nossa solução através de testes com utilizadores. Os resultados destes testes foram muito satisfatórios pois, apenas três utilizadores cometeram erros, e cada um deles apenas cometeu um erro durante todo o teste. Relativamente ao questionário de usabilidade que os utilizadores preencheram, 99.3% dos utilizadores seleccionou uma das duas melhores classificações possíveis para cada questão apresentada. Este facto demonstra a satisfação dos utilizadores com a nossa solução.

Abstract

Storing medical information about a patient and displaying that information or its analysis according to a physician’s desires through easy to understand graphics during a diagnosis, is very important in order to provide the best possible healthcare service. Nowadays, there are a wide variety of medical software applications to assist physicians during appointments. However, the majority of them do not display medical information through graphics, meaning that the physicians need to search themselves for the information they desire and then perform the analysis as opposing to just look at a graphic where all of that is clear. Therefore, we proposed and developed a medical data visualization module for the Umedicine application, which allows the physicians to consult and monitor the evolution of several patient’s health parameters. This module offers two functionalities: Graphical representation of the medical data regarding the historic of medical exams; Dashboard offering a graphical representation of the evolution of several health parameters allowing the physician to create graphics and save them to the dashboard. Finally, we validated our solution by conducting user tests. The results of these tests were very satisfactory as only three users made mistakes and each user made just a single one. Regarding the usability questionnaire that the users were asked to fill, 99.3% of the users selected one of the two best possible grades for each question presented. This fact shows the satisfaction of the users with our solution.

Keywords: Medical Software Applications, Medical Data Visualization Module, Data Analysis, Medical Graphics, Diagnosis.
Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Resumo</td>
<td>iii</td>
</tr>
<tr>
<td>Abstract</td>
<td>v</td>
</tr>
<tr>
<td>List of Tables</td>
<td>ix</td>
</tr>
<tr>
<td>List of Figures</td>
<td>xii</td>
</tr>
<tr>
<td>Nomenclature</td>
<td>1</td>
</tr>
<tr>
<td>Glossary</td>
<td>1</td>
</tr>
<tr>
<td>1 Introduction</td>
<td>1</td>
</tr>
<tr>
<td>1.1 Umedicine: Tool to Support Urology Appointments</td>
<td>1</td>
</tr>
<tr>
<td>1.2 Problem</td>
<td>2</td>
</tr>
<tr>
<td>1.3 Requirements</td>
<td>3</td>
</tr>
<tr>
<td>1.4 Objectives</td>
<td>4</td>
</tr>
<tr>
<td>1.5 Contributions</td>
<td>4</td>
</tr>
<tr>
<td>1.6 Outline</td>
<td>4</td>
</tr>
<tr>
<td>2 Related Work</td>
<td>7</td>
</tr>
<tr>
<td>2.1 Medical Software applications</td>
<td>7</td>
</tr>
<tr>
<td>2.1.1 My MedicineOne</td>
<td>7</td>
</tr>
<tr>
<td>2.1.2 eClinicalWorks EHR</td>
<td>9</td>
</tr>
<tr>
<td>2.1.3 AllScripts Professional EHR</td>
<td>12</td>
</tr>
<tr>
<td>2.1.4 Kareo</td>
<td>13</td>
</tr>
<tr>
<td>2.1.5 MediTouch EHR</td>
<td>15</td>
</tr>
<tr>
<td>2.1.6 NextGen Healthcare</td>
<td>16</td>
</tr>
<tr>
<td>2.2 Data Visualization tools</td>
<td>18</td>
</tr>
<tr>
<td>2.2.1 Developer tools</td>
<td>18</td>
</tr>
<tr>
<td>2.2.2 Non-Developer tools</td>
<td>21</td>
</tr>
<tr>
<td>2.3 Discussion</td>
<td>24</td>
</tr>
<tr>
<td>2.3.1 Medical software applications</td>
<td>24</td>
</tr>
<tr>
<td>2.3.2 Data visualization tools</td>
<td>25</td>
</tr>
<tr>
<td>3 Solution</td>
<td>26</td>
</tr>
<tr>
<td>3.1 Proposed Solution</td>
<td>26</td>
</tr>
</tbody>
</table>
# Table of Contents

3.1.1 Functionalities ....................................................... 26  
3.1.2 Infrastructure ....................................................... 27  
3.2 Developed Solution ..................................................... 27  
3.2.1 Technologies ......................................................... 27  
3.2.2 Code development ................................................... 31  
3.2.3 Functionalities ....................................................... 34  

4 Validation ................................................................. 39  
4.1 Validation of the user interface ........................................ 39  
4.1.1 Test environment ................................................... 39  
4.1.2 User tasks ........................................................... 40  
4.1.3 Questionnaires ...................................................... 42  
4.1.4 Results ............................................................... 42  

5 Conclusions ................................................................. 53  
5.1 Summary ............................................................... 53  
5.2 Future Work ............................................................ 54  

Bibliography ................................................................. 57
List of Tables

2.1 Requirements fulfilled by the medical software applications. . . . . . . . . . . . . . . . . . . . . 24
2.2 Comparison of the data visualization tools. . . . . . . . . . . . . . . . . . . . . . . . . . . 25
## List of Figures

<table>
<thead>
<tr>
<th></th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>Snapshot of Umedicine’s patient information screen.</td>
<td>2</td>
</tr>
<tr>
<td>2.1</td>
<td>Snapshot of My MedicineOne’s patient information screen.</td>
<td>8</td>
</tr>
<tr>
<td>2.2</td>
<td>Snapshot of eClinicalWorks CCMR dashboard view.</td>
<td>10</td>
</tr>
<tr>
<td>2.3</td>
<td>Snapshot of the eClinicalWorks CCMR patient information screen.</td>
<td>10</td>
</tr>
<tr>
<td>2.4</td>
<td>Snapshot of the eClinicalWorks EMR patient information screen.</td>
<td>11</td>
</tr>
<tr>
<td>2.5</td>
<td>Snapshot of AllScripts Professional EHR report results interface.</td>
<td>13</td>
</tr>
<tr>
<td>2.6</td>
<td>Snapshot of Kareo’s Practice Management dashboard view.</td>
<td>14</td>
</tr>
<tr>
<td>2.7</td>
<td>Snapshot of MediTouch EHR’s user interface.</td>
<td>16</td>
</tr>
<tr>
<td>2.8</td>
<td>Snapshot of NextGen Ambulatory EHR user interface.</td>
<td>17</td>
</tr>
<tr>
<td>2.9</td>
<td>Snapshot of NextGen Dashboard Provider view.</td>
<td>18</td>
</tr>
<tr>
<td>2.10</td>
<td>Snapshot of a graphic created with D3.</td>
<td>19</td>
</tr>
<tr>
<td>2.11</td>
<td>Snapshot of a spline char type regarding the monthly average temperature in Tokyo and London.</td>
<td>21</td>
</tr>
<tr>
<td>2.12</td>
<td>Snapshot of Tableau Public’s graphical interface.</td>
<td>22</td>
</tr>
<tr>
<td>2.13</td>
<td>Snapshot of Chartblocks’ chart creation screen.</td>
<td>23</td>
</tr>
<tr>
<td>3.1</td>
<td>Infrastructure of the medical data visualization module.</td>
<td>27</td>
</tr>
<tr>
<td>3.2</td>
<td>Example of the use of the HTML language for webpage structuring.</td>
<td>28</td>
</tr>
<tr>
<td>3.3</td>
<td>Example of the use of the CSS language for HTML element styling.</td>
<td>28</td>
</tr>
<tr>
<td>3.4</td>
<td>Example of the use of the JavaScript language for the definition of a function.</td>
<td>29</td>
</tr>
<tr>
<td>3.5</td>
<td>Example of the use of the jQuery library for an Ajax get request.</td>
<td>30</td>
</tr>
<tr>
<td>3.6</td>
<td>Example of the use of the D3.js library for importing JSON data.</td>
<td>30</td>
</tr>
<tr>
<td>3.7</td>
<td>Example of the definition of a DAO(Data Access Object) function.</td>
<td>31</td>
</tr>
<tr>
<td>3.8</td>
<td>Questionnaire graphic within the dashboard page.</td>
<td>35</td>
</tr>
<tr>
<td>3.9</td>
<td>Example of the process of the deletion of a created graphic.</td>
<td>36</td>
</tr>
<tr>
<td>3.10</td>
<td>Example of the screen that appears after the selection of a specific exam.</td>
<td>37</td>
</tr>
<tr>
<td>3.11</td>
<td>Example of the screen for the Historic functionality for a specific selected exam.</td>
<td>38</td>
</tr>
<tr>
<td>3.12</td>
<td>Example of the pop-up window displaying the evolution of the historic for all the parameters.</td>
<td>38</td>
</tr>
<tr>
<td>4.1</td>
<td>Table displaying the gathered results for the metric: time to complete the task.</td>
<td>43</td>
</tr>
</tbody>
</table>
4.2 Table displaying the gathered results for the metric: number of clicks needed to complete the task. .......................................................... 43
4.3 Time metric results for each one of the four very simple tasks. ................................. 44
4.4 Ideal clicks needed to completed each one of the four simple tasks and standard deviation of the users’ clicks over the ideal clicks. ........................................ 45
4.5 Time metric results for each one of the four simple tasks. ........................................ 46
4.6 Ideal clicks needed to completed each one of the four simple tasks and standard deviation of the users’ clicks over the ideal clicks. ........................................ 47
4.7 Summary of the results of the questionnaire regarding the usability of the user interface and the user satisfaction. ................................................................. 49
4.8 Results of the questionnaire regarding question 6. .................................................... 50
4.9 Results of the questionnaire regarding question 12. .................................................. 50
4.10 Results of the questionnaire regarding question 13. .................................................. 50
4.11 Results of the questionnaire regarding question 1. .................................................. 51
4.12 Results of the questionnaire regarding question 2. .................................................. 52
4.13 Results of the questionnaire regarding question 8. .................................................. 52
Chapter 1

Introduction

With the ongoing advance of technology that has been occurring in the last years, it was expected that the healthcare providers would not be excluded. Medical institutions in general have been expressing their needs for medical software that would greatly improve the efficiency and quality of the healthcare service provided. With this expected interest of the healthcare providers, the market of medical software applications has been quickly growing, with a wide variety of software applications that are now available. Umedicine, described in Section 1.1, is a software prototype to support urology appointments focused in the needs and requirements of the physicians. It aims to improve the patient oriented service, offering the physicians a usable interface which enables the easy access and management of a patient’s medical information.

1.1 Umedicine: Tool to Support Urology Appointments

Umedicine is a currently under development medical software application with the purpose of assisting Urology physicians during clinical appointments. It offers functionalities to store appointment data and then to manage and analyse that data in a way that facilitates the work of the physicians as well as brings benefits to their patients, through a better quality of service offered. This software application is being developed according to the requirements specified by a physician of this specialty. Umedicine can be accessed by any device through a web browser, requiring internet connection. The currently implemented functionalities are:

- Data storage of patient personal data, diseases, treatments and questionnaires
- Graphical user interface where it is possible to enter, update and access:
  - Patient personal data
  - Disease (symptom and disease characterization)
  - Questionnaires filled in by patients thus enabling tracking the patient’s health condition before and/or after the treatment.
Figure 1.1: Snapshot of Umedicine’s patient information screen.

Figure 1.1 presents the patient page of the Umedicine application. This is the main page of Umedicine. This page presents the personal information of a patient as well as his/her medical information. As can be seen in the figure, it is segmented in several sections: Disease, Symptoms, Diagnostic Exams, Treatment, Questionnaires. Each of these sections contains specific medical information regarding the patient in analysis.

Umedicine, currently lacks a module that offers a graphical visualization of the data collected and stored. The module that will be developed in this thesis will be integrated in the Umedicine application.

1.2 Problem

Medical software applications are used by different types of users, like administrators, physicians and patients, each one of them having different needs and requirements that need to be satisfied. However, considering the graphical visualization of information, we can say that the majority of medical software applications are more focused in the needs of one type of user: the hospital administrator. The medical software applications support a wide range of administrative functionalities in order to address the needs of administrators: billing modules with graphical analysis tools over predefined parameters (payment per patient for example) and performance indicators such as payment velocity and days to bill, to keep track of the financial situation. The same does not happen concerning the needs of the other types of users: physicians and patients, as the information that each of these two users has access to is presented in the form of tables. Regarding the healthcare service, these applications present generic medical information, offering tools for patient management, appointments management and e-prescribing.

Considering the needs of physicians, we can say that there are three main requirements for a medical software application, in order to enhance his/her performance: i) the application gathers all the appoint-
ment data and stores it in a digital format; ii) must allow a graphical visualization of the evolution of several medical parameters regarding a patient; iii) must allow analysing the data and showing the result of the analysis through graphics. Most of the existing medical software applications are not flexible regarding the graphics availability, limiting the access to graphics to a specific and isolated module or area of the application, which usually is a generic dashboard. The generic dashboard allows the creation of predefined graphics, such as blood pressure, over the patients’ population. However, limiting the access to graphical data to only a specific area of the application is not practical as, for example, if the physician is monitoring a patient’s exam data, and wishes to track the evolution of a certain parameter graphically in that same page, that feature should be allowed, instead of having to go to a different module or area of the application to consult that information. Regarding the analysis of medical information, it is required by the physicians, that an application provides tools to analyse the stored medical information and present those results through easy to understand graphics, so that the physician does not lose a lot of time trying to understand a graphic. Most of the existing medical software applications, such as Meditouch EHR\(^1\) and Kareo\(^2\), only offer tools to graphically analyse administrative data, such as payment velocity, days to bill, refunds and patient assistance. There are software tools that enable the visualization of medical data through graphics, however these do not comply with the needs of the physicians, as it is only possible to create graphics over a population of patients and the graphics available to create are predefined and very limited.

1.3 Requirements

Considering the problem identified in Section 1.2 we identified the following requirements for a medical data visualization module:

- **Visualization of medical information**: It should support visualizing medical information through graphics, in particular:

  *Patient’s health condition evolution*: It should allow tracking the evolution of a patient’s health condition or parameter (PSA, for example) through one or more dimensions.

  *Graphic Availability*: The access to the graphical information should not be limited to a single module or area (dashboard) of the application. It should be also possible the access to graphical information in every page that contains relevant information.

  *Customizable*: It must be possible for the physician to modify or create graphics by choosing which patient data he/she wants to see.

  It should support several types of graphics to show the medical data: bar graphics, linear graphics, etc.

---

\(^1\)https://www.healthfusion.com/ehr-software/

\(^2\)http://www.kareo.com/
1.4 Objectives

The purpose of this thesis is to develop a medical data visualization module capable of assisting Urology physicians during a diagnosis. For this purpose, this module will be focused on requirement ii) of the physicians requirements - must allow a graphical visualization of the evolution of several medical parameters regarding a patient - , described in Section 1.2, however the developed functionalities should also be possible to apply for requirement iii) - must allow analysing the data and showing the result of the analysis through graphics. In order to answer requirement ii), this module must present the patient's data relevant for a diagnosis. Regarding the data presentation method, we chose a digital dashboard for this matter, as it allows presenting several different predefined graphics on the same page while having them organized at the same time by the graphic's parameters, for example, therefore allowing the physicians to quickly consult the graphic within the dashboard they desire to see, or also to create graphics over the parameters they desire to analyse offering the possibility to save these to the dashboard for future access. However, if the physician intends to have a graphical look at the evolution of one or more parameters within a specific patient's exam, this should also be possible. In order for this data to be quickly and efficiently accessed by the physician, this module should present a user-friendly interface.

1.5 Contributions

The main contribution of this thesis, is the following:

1. Graphical representation of the medical data regarding the historic of medical exams.

The functionalities of the solution allow the physician to monitor the evolution of one or more parameters of a specific patient's exam at the same time. While also offering a dashboard, accessible from the patient's page, that provides some predefined graphics for the physician to consult, which were established as important by the physician collaborating in this thesis. Additionally, this dashboard functionality also allows the physician to create his/her own graphics by offering dropdown lists with options for the physician to select. After the graphic is created, the physician may save the graphic, adding it persistently to the graphics that are shown when accessing the dashboard page. The created graphic can be deleted whenever the physician intends to.

1.6 Outline

This document is organized into five chapters. Chapter 2 details the related work, in particular describes other medical software applications analysing them in terms of data visualization and making a comparison with the requirements described in Section 1.3. In this section, we also analyse some data visualization tools concluding with the choice of the one to use for the development of our solution. Chapter 3 starts by presenting the initially proposed solution and its functionalities and then presents
the developed solution, describing briefly the technologies used, the code developed and the functionalities of the solution. Chapter 4 describes the validation that we performed for the new medical data visualization module of Umedicine. Finally, Chapter 5 concludes and summarises future work.
Chapter 2

Related Work

The market of medical software applications is growing at a fast pace, with a large number of applications already available today. To better understand this market, we performed an analysis of these applications, identifying what they provide and what they are missing. For this purpose, we conducted a search on the Web to find the most used applications as it is not possible to cover every application that is available today. In addition, as the purpose of this thesis is to develop a medical data visualization module, we also conducted an analysis on some of the most popular data visualization tools\(^1\), grouping them as developer or non-developer tools, identifying some of the most important features that each of them provides.

In this chapter we describe the chosen applications and data visualization tools. We then characterize the medical software applications according to the requirements identified in Section 1.3 and compare the data visualization tools between each other. For medical software applications, we chose national applications, to better understand the portuguese market, and the most popular applications worldwide\(^2\).

2.1 Medical Software applications

In this section we describe the features of the chosen medical software applications as well as analyse them according to the requirements listed in Section 1.3.

2.1.1 My MedicineOne

My MedicineOne\(^3\) is a cloud-based medical software developed by a Portuguese company named MedicineOne. My MedicineOne covers a variety of areas. Some of these areas, chosen according to the features they offer (patient information screen, appointment management, visualization and analysis of data) are: clinical area, and data treatment. In addition to these areas, the software also offers a free mobile application (I+).

\(^1\)http://thenextweb.com/dd/2015/04/21/the-14-best-data-visualization-tools/

\(^2\)http://www.capterra.com/infographics/top-emr-software

\(^3\)http://www.medicineone.net
In the *clinical area* we can access the patient’s information screen, shown in Figure 2.1, which is divided in four areas:

- **allergies**: shows the patient’s allergies (to a medicine, for example);
- **treatment**: shows the treatment the patient is undergoing;
- **disease**: shows the past and current diseases of the patient;
- **SOAP**: presents the SOAP report of his previous appointments (browsable), which is a method to write notes in a patient’s chart, in which each letter has its own field for filling and where: *S* stands for subjective; *O* stands for objective; *A* stands for assessment; and *P* stands for plan.

This screen also presents a top menu bar, where it is possible to choose one of these areas to access more in-depth information.

Within this module it is also possible to prescribe medicines for the patient. The *data treatment module* makes it possible to analyze all the information stored in My MedicineOne through statistic modules and a performance monitor. This performance monitor enables the access to data treatment options such as graphics and tables. It also enables the calculation of performance indicators to monitor and analyse the current state of the practice, covering areas such as productivity and effectiveness.

The *I+* is a free application available for both iPhone and Android devices which allows the patients to access the most important data included in the clinical file, such as health problems, while also reminding the patients of certain activities like taking the medication.

Regarding My MedicineOne’s user interface, we can say that it is efficient as it only shows the relevant information for each module and its functionalities are straightforward to use. These facts plus the
simplicity of the user interface contribute to a high accuracy which leads to the conclusion that the user interface is user-friendly.

Taking into consideration the visualization of medical information requirement, My MedicineOne does not fulfill this requirement. It offers modules to visually analyse data such as payments and number of patients assisted by each physician. It also computes performance indicators. However, all of this information is focused on the administrative aspects.

2.1.2 eClinicalWorks EHR

The eClinicalWorks EHR solution is widely used all around the world. This EHR is composed by modules such as: The eClinicalWorks EMR; eClinicalWorks Practice Management; Patient Portal; Population Health (CCMR); and Chronic Care Management (CCM). From all of these modules we consider the EMR, Practice Management, Patient Portal and Population Health (CCMR) to be the most relevant to describe when taking into account the purpose of this thesis.

Within the eClinicalWorks EMR module it is possible to:

- **Customize the tabs** that appear on the screen with the Interactive Clinical Wizard (ICW).

- **Access clinical content** such as patient’s medication, personal details, problem history, labs and test results.

- **Communicate internally** with the internal messaging system which allows communication between physicians and/or nurses through messages, ensuring that everyone can access up-to-date information.

- **Add ICD-10 assessment codes** to the assessments. These codes can then be used to search for a specific ICD-10 code using the eClinicalWorks ICD-10 Search Capabilities.

Now, regarding the eClinicalWorks Practice Management module, which is focused on the performance of the practice, it offers functionalities such as:

- **Schedule management**: allowing to manage physicians’ schedules, so that the physicians know what they have scheduled for the day and can rearrange their schedules if something urgent comes up.

- **Bill management**: supports electronic claims submission and allows the payment of services.

- **eBO Reporting**: provides physicians or administrators reports of key elements such as immunizations, allergies and procedure codes. It is possible to create a graphical report to monitor blood pressure for all patients between a certain selected date according to that patient’s history, for example.

The eClinicalWorks Patient Portal is a module developed to enable patients to access their clinical data, such as personal details, appointments, test results communicated by their physician or prescription refill

requests. With this module, patients can access their health information directly from wherever they are. The eClinicalWorks Population Health (CCMR) module has features such as community analytics and patient satisfaction surveys. Regarding the community analytics, show in Figure 2.2, the module provides a generic dashboard where it is possible to check a lot of different information presented through graphics, whether it is patient-related or practice-related information. This provided information can be about, for example, a patient’s heart, a patient’s satisfaction survey or disease related information through scorecards, such as diabetes or cardiovascular diseases. It also helps monitoring if a patient’s care plan has been followed, displaying a graphic that shows the completion of the selected steps over time.

Figure 2.2: Snapshot of eClinicalWorks CCMR dashboard view.

Figure 2.3: Snapshot of the eClinicalWorks CCMR patient information screen.
The CCMR patient screen, shown in Figure 2.3, presents a risk score for each patient that helps identifying the patients who have the greatest needs and also displays a table with that patient’s visits information. Within this screen it is also possible to change from a patient view to a community view, to check the initial assessment performed by the physician and to have a more in-depth view of that patient’s risks.

**eClinicalWorks EMR/PM**

Analysing this module according to the requirements, we conclude that the user interface, shown in Figure 2.4, is not user-friendly as it is difficult to learn and use for new users. It presents icons and data at several different areas in the same screen which makes it confusing, complex and therefore non-efficient.

![Snapshot of the eClinicalWorks EMR patient information screen.](image)

Regarding the visualization of medical information requirement, this module allows visualizing medical information such as vitals and allergies. It also allows tracking a patient’s health condition evolution, using blood pressure and a specified date range as dimensions, for example. However, these graphics are only accessible within the generic dashboard of the eBO reporting tool. In addition, this eBO reporting tool is not customizable as the reports and metadata elements are all predefined.

**eClinicalWorks CCMR**

Analysing this module according to the requirements, we conclude that the user interface of the CCMR module is user-friendly, as it simple, every action is labeled and does not contain any complex functionalities, making it efficient and accurate.

Regarding the visualization of medical information requirement, this module allows monitoring aspects
like diabetes, blood pressure and cardiovascular disease through graphics. This module also allows tracking a patient's health condition evolution, using blood pressure and a specified date range as dimensions, for example. However, these graphics are only accessible within the generic dashboard and are not customizable as they are based on already predefined parameters.

2.1.3 AllScripts Professional EHR

AllScripts Professional EHR is one of the most popular ambulatory care systems for medical practices worldwide. This EHR is one of a huge variety of modules developed by AllScripts for healthcare solutions including also, for example: AllScripts Practice Management; AllScripts Patient Flow. We consider the AllScripts Professional EHR to be the most relevant for what we intend to develop, as it was developed for providing physicians a complete system that meets their needs.

This system presents features such as: Appointment management; Patient details; E-prescribing; and Patient Portal. With the appointment management feature it is possible to view all the appointments for a specific physician, displaying the patient name, patient number, the corresponding check-in time, the start time, and the check-out time, and a comment field for the purpose of adding any additional detail to the appointment making it easier for the doctor to know what the purpose of each appointment is.

The patient detail feature has two main areas: the patient sheet area; the patient manager area. Within the patient sheet area it is possible to check and update the medical history for a patient, the diagnostic studies, the patient's personal and family information, current medication, surgeries. In the patient manager area it is possible to access lab results and send them to the patient portal, access the patient appointment schedule and care management for the chronic diseases.

With the e-prescribing feature it is possible to create a prescription, digitally sign it and send it to the patient's preferred pharmacy immediately after or prior signing off the patient's visit.

Regarding the patient portal, Allscripts FollowMyHealth is the patient portal provided by AllScripts with the purpose of allowing patients to participate and have a more active role in managing their health. This portal is accessible via Web or with any mobile device by using the available mobile application. It is also possible for patients to get in touch with their physicians by sending messages with any doubts or questions they have regarding their health (view appointments, test and labs results, or renew medications).

Besides these most important features, we found some others that we considered interesting and worth referring:

- Medication management: alerts physicians to allergies and possible interactions that may occur between a patient and a given medicine.

- AllScripts EAuth: allows receiving authorizations in the prescription of certain medicines.

- NoteSwift: facilitates completing patient documentation through speech recognition, drastically reducing mouse clicks to about only five per note.
Analysing this system according to the requirements listed in Section 1.3, the user-interface is well organised, presenting a menu on the left with all the possible actions. On the top side of the screen, once a patient is selected, it is displayed that patient's information. The right side of the screen displays the available actions such as checking for result notifications or messages. In the middle area it is presented all the information. However, this user interface is not user-friendly as it requires time to get used and to understand how all the functionalities work. Regarding user comments\(^7\), the user interface presents too many descriptors that have nearly the same meaning which makes it hard for charting. Regarding the insertion of data, there are a lot of columns which need to be filled and if, for example, the data is inserted in the wrong column there is no way to delete it, not being therefore efficient and accurate.

Taking into consideration the visualization of medical information requirement, this requirement is also not fulfilled, as we only found evidences that it supports monitoring administrative aspects through graphics, as seen in Figure 2.5 which presents the number of patients in each practice.

![Figure 2.5: Snapshot of AllScripts Professional EHR report results interface.](image)

### 2.1.4 Kareo

Kareo software\(^8\) is one of the most used medical softwares. Its latest version currently consists of four main modules such as: EHR; DoctorBase; Practice Management; Medical Billing.

The EHR module was designed to work as "plug and play", being up and running as fast as possible. Within this module it is possible to:

- Create customizable clinical notes

\(^7\)http://www.softwareadvice.com/medical/allscripts-ehr-profile/

\(^8\)http://www.kareo.com/
- Add and search for patients
- Create and edit appointments
- E-prescribe directly to the patient’s preferred pharmacy
- Send secure messages to the network, for example, between physicians

This module also offers an area where it is possible to see the scheduled patients and their appointments, check for labs and studies results and see open notes or flagged messages. There is also an integrated patient portal where it is possible for patients to access medical and billing information allowing them to have a more active role in managing their health.

Within the Practice Management module it is possible to check bills and analyse reports as well as check for patient alerts according to their health. It is also possible to manage schedules and see practice performance according to metrics and graphics through a generic dashboard, shown in figure 2.6.

![Figure 2.6: Snapshot of Kareo’s Practice Management dashboard view.](image)

The DoctorBase module is a marketing tool to help attracting new patients offering features that make it easy for a patient to find the brand through Google. This module allows ranking and rating doctors online and share on social media.

The medical billing module has the purpose of ensuring that payments are always up-to-date, having a team available to take care of these matters.

Taking into consideration the determined requirements and analysing the user interface, we conclude that Kareo’s user interface is user-friendly. Kareo offers a visually appealing graphical interface, it is easy to learn for new users as the screens are simple and only contain the essential information and its functionalities are straightforward to use.
Regarding the visualization of medical information requirement, Kareo does not fulfill this requirement. The only information analysed is administrative information such as payment velocity and days to bill, not allowing to monitor, for example, the evolution of a patient’s health condition according to parameters.

2.1.5 MediTouch EHR

Meditouch software was developed by Healthfusion and is composed by two modules: EHR; Billing. According to the purpose of these thesis, we will describe the MediTouch EHR. MediTouch EHR is a cloud-based medical software which is exclusively available in the United States. Being MediTouch EHR a cloud-based solution, it enables the access to data from anywhere with only the need of internet connection, however the access is limited on the mobile domain as it only presents an iPad application. MediTouch EHR software presently supports 19 specialties and offers a wide variety of features such as: Appointment management; e-Prescribing; charting; and patient portal.

For appointment management the software allows the physician to access all the available and relevant information about the patient namely: allergies and vital signs such as measured temperatures, test results, patient complaints, medications and family history. For reporting the examination findings, it provides the physician a SOAP format report. It also allows taking a snapshot of a patient’s finding, send it directly to the patient’s chart and then draw on it using fingertips or a mouse. Within the charting feature, MediTouch EHR provides some pre-loaded templates, being these templates customizable and being possible to save them on favourites after. It also suggests a diagnosis based on the findings reported and allows the physician to choose one of his predefined care plans for that diagnosis or it can capture the speech using the native iPad dictation tool.

For e-prescribing, MediTouch went through testing process to ensure that the prescription produced is compliant with the Medicare e-prescribing directive and other requirements. It allows prescribing directly to the patient’s favourite pharmacy.

The patient portal enables the patient to access their physician’s schedule, and allows patients to schedule appointments online. It is also possible for the patient ask for medication renewal and, for example, to check lab results previously shared by their physician.

Analysing MediTouch according to the user interface requirement, we can say, based on user comments also, that MediTouch EHR has a user-friendly interface, as its functionalities are straightforward to use. It also does not let you send invalid information, which makes it accurate, as it prevents user errors. In addition, each screen is simple and not overfilled with information as can be seen in Figure 2.7, which presents an example of the user-interface.

Regarding the visualization of medical information requirement, this system does not fulfill this requirement. The only graphics available are inside a generic dashboard which belongs to the billing module and thus only covers billing aspects, such as payments. It is therefore not possible to monitor the evolution of a patient’s health condition through graphics, for example.

---

9https://www.healthfusion.com/ehr-software/
10http://www.softwareadvice.com/medical/healthfusion-meditouch-profile/
2.1.6 NextGen Healthcare

NextGen Healthcare software \(^{11}\) is composed by different modules: EHR; Practice management; Analytics; Patient Portal; and Productivity tools. For the purpose of this project we will only focus on the NextGen Ambulatory EHR and Analytics.

Within the NextGen Ambulatory EHR it is possible to manage appointments, providing a SOAP template for the provider to register the symptoms found in the patient check up along with the suggested care plan. It is also possible to check for patient information such as allergies, current or past medication, medical history, problems and diagnoses, being all of this information available directly in the patient's page.

Within the EHR it is also possible to run some predefined growth graphics for patients, such as BMI, height, weight and weight for length, allowing to analyse, for example, the patients height comparing to their age.

With the PatientSync it is possible for a provider to access patient data without having connectivity.

NextGen Healthcare also developed a tool called NextGen Dashboard which belongs to the analytics module, that allows the analysis and interpretation of the patient data directly from NextGen Ambulatory EHR and NextGen Practice Management. With the NextGen Dashboard it is possible to share key data across the organization, being possible to have a dashboard view over several different organizational roles, such as: Business office; Community View; Executive; Practice Administrator; Provider; and

\(^{11}\)https://www.nextgen.com/
Figure 2.8: Snapshot of NextGen Ambulatory EHR user interface.

Each of these roles has already some predefined graphics but it is also possible to customize your own graphic reports according to the metrics you intend to use. You can also choose which reports you want to appear in your dashboard view whether they are the predefined ones or some others that you already have created.

Analysing NextGen's user interface, shown in Figure 2.8, we can say that it is a user-friendly interface, as it is well organized and each icon has its own label leading to an efficient and accurate to use interface. The functionalities offered are not complex and are straightforward to use. The dashboard's user-interface, which can be seen in Figure 2.9 is also simple, easy to understand and work with.

Regarding the visualization of medical information requirement, with the NextGen Dashboard it is possible to graphically analyse medical information such as diabetes, based on a group of patients. These graphics are customizable, being possible to choose the type of graphic, the data source, and the dimensions to measure. However, these graphics are only accessible through the NextGen Dashboard, which is a generic dashboard.
2.2 **Data Visualization tools**

In this section we will talk about some of the existing data visualization tools. These tools are usually grouped in two different groups: developer and non-developer tools. The non-developer oriented tools use a graphical interface for the construction of graphics and, through predefined scripts it is possible to visualize the created graphics on a specific application. These tools also make it easier for the user to customize his graphics as they offer a wide variety of customizability options which are selectable. However, the developer oriented tools are tools that give more flexibility to the user regarding the creation of graphics as they require programming skills for their creation.

2.2.1 **Developer tools**

This section describes the features of the chosen developer data visualization tools. We chose D3.js and Highcharts as both of them belong to the most used data visualization tools and allow for external data loading from servers, which is essential for the purpose of this thesis.
D3.js

D3.js\(^{12}\) stands for Data-Driven Documents. It is a free and open-source JavaScript library that allows the manipulation of documents based on data, having the purpose of displaying this information through various types of graphics. D3, being a developer tool, combines a variety of languages like HTML, CSS and SVG which all together allow for a very flexible creation of graphics, where basically any kind of graphic and analysis can be made possible. D3’s capabilities are designed to work at its fullest in the most modern browsers, for example, Internet Explorer 9+, and the fact that it does not limit the developer to a single framework is what contributes the most for its flexibility. D3 supports large dataset and dynamic behaviours for interaction and animation, while also offering a wide variety of components and plugins. D3 does not offer any pre-built charts, however, it offers a gallery\(^ {13}\) where it is possible to access some of the graphics created which contain also the code used for their creation. An example of this, can be seen in Figure 2.10.

---

**Figure 2.10:** Snapshot of a graphic created with D3.

Chord diagrams show directed relationships among a group of entities. This example also demonstrates simple interactivity by using mouseover filtering. Layout inspired by Martin Krzywinski’s beautiful work on Circos.

D3 offers many functions grouped by different types, being some of them the following:

- **Selections (Core):** these functions allow the modification of a document by, for example, choosing

\(^{12}\)https://d3js.org/  
\(^{13}\)https://github.com/mbostock/d3/wiki/Gallery
a specific tag in the HTML code and setting or changing its properties.

- **Transitions (Core):** these functions are related to animated transitions and allow setting a transition and defining its parameters like delay, duration and style, for example.

- **Colors (Core):** these functions allow specifying a color in RGB space, changing an already defined color and converting an RGB color to HSL or String.

- **Quantitative (Scales):** these functions allow, for example, constructing a linear quantitative scale, specifying the scales input domain and output range or creating a new scale from an existing one.

- **Shapes (SVG):** these functions allow creating different types of shapes like lines (linear curves, radial lines) as in a line chart, piecewise linear areas as in an area chart, or other shapes.

- **Pie (Layout):** these functions allow the creation of a pie layout, being possible to set the overall start and end angle of the pie as well as its pad angle. It is also possible to control the clockwise order of the pie slices.

- **Polygon (Geometry):** these functions allow the creation of a polygon from a specified array of points and to compute its area.

D3 also offers dynamic properties as, for example, it allows the specification of styles, attributes and other properties as function of data, instead of simple constants. Although D3 has a steep learning curve, its flexibility make it one of the most used javascript libraries for data visualization softwares.

**Highcharts**

Highcharts is a library written in JavaScript, developed by a company named HighSoft AS, with the purpose of data visualization and charting.

Highcharts is free for non-commercial and personal use, however if it is intended for commercial use it requires buying a licence which has a price range that varies from 150 USD to 5.020 USD, regarding the objectives of the company with highcharts. Highcharts is highly compatible with the web browsers, working in all modern mobile and desktop browsers such as Internet Explorer from version 6 and the iPhone or iPad.

The fact that Highcharts is fully developed in JavaScript makes it very easy to use since it does not require client side plugins like Flash or Java and it also does not require installing anything into the web server like PHP or ASP.NET. For Highcharts to work it is only required two JavaScript files: the highcharts.js core and either jQuery, Mootools or Prototype framework. Highcharts also allows significant flexibility as it is possible for the developer to download the source code and make his own edits. Through a full API it is possible to add, remove and modify points or axes after a chart is created. This, in combination with other APIs like jQuery, allow for dynamic charts which are constantly updating with data from servers, files and so on.

\[\text{14} \text{http://www.highcharts.com/}\]
Regarding the chart types, highcharts supports a significant variety of charts such as: line; spline; areaspline; column; bar; pie; scatter; angular gauges; arearange; areasplinerange; columnrange and polar charts. It also allows the combination of some of these chart types into a single chart. Still regarding the charts, it is possible to compare several different variables that are not the same scale by assigning them to the x or y axis and it is also possible to add labels to the chart points with some important text information and to rotate them in any angle. Figure 2.11 presents an example of a spline chart type.

![Chart Example](https://example.com/chart.png)

**Figure 2.11**: Snapshot of a spline chart type regarding the monthly average temperature in Tokyo and London.

Regarding the external data loading, with highcharts it is possible to load data from local objects, files or servers.

After the chart is created, it is possible to export it to different format types such as SVG, PNG, JPG or PDF or even just to print the chart directly from a web page.

### 2.2.2 Non-Developer tools

This section describes the features of the chosen non-developer data visualization tools. We chose Tableau Public as it is a highly known and also, as well as Chartblocks, is one of the most popular data visualization tools and both of them allow for external data loading from servers, which is essential regarding the purpose of this thesis.

**Tableau Public**

Tableau Public\(^{15}\) is a free software solution of Tableau and it is one of the most popular softwares for data visualization and analysis. With this software it is possible to open data from many different data sources...
sources, whether it is from files (Excel, Access, Text and Statistical) or from databases, and to work and analyse that data in order to obtain several different types of insights.

Once Tableau Public accesses the data contained in the file or database, it presents a drag and drop graphical interface, seen in Figure 2.12, where it is possible to choose the parameters to analyse and the type of graphic that you want the analysis to be displayed in. Once the graphic is created, it is also possible to detail, color, resize or label the graphical information according to any dimension, allowing a more in-depth analysis of the information. Finally, when the user is satisfied with the graphic, it is possible to save that graphic on the user’s profile, which contains 1GB of space, to be used in the future.

With Tableau Public, it is also possible to create a dashboard where the user can insert all of the graphics he wants to access, and to rearrange those graphics according to his likes. After the dashboard is done, Tableau Public offers the ability to save and publish it, obtaining a website view of the desired dashboard where it is then possible to share it, for example, on social media or to integrate it on a specific application by copying and pasting the presented javascript code. The data of the dashboard is saved in the Tableau Public Servers.

Tableau Public for a non-developer tool, offers great variety and flexibility and it is easy to use. However, if the user is struggling with Tableau Public, Tableau Public’s website displays some video tutorials which cover the basic aspects of working with their solution.

---

16 https://public.tableau.com/s/resources
Chartblocks

Chartblocks\textsuperscript{17} in an online chart builder that has the purpose of allowing the creation of quick charts over data. The data over which the chart is to be created can be imported from some different data sources such as: spreadsheets; databases; and live feeds. An interesting feature of Chartblocks is that it allows the definition of scheduled imports, being this way possible for the user to define periodical data imports releasing him of some worries he might had regarding this matter.

Chartblocks also offers a wizard which analyses all the information imported and that suggests, from all that data, the data he considers important for the type of chart that the user wants to create, highlighting the columns which contain that data. This selection can be changed if for example, the wizard misdetects some column with important data, or if the user intends to add another column with more information to the essential data columns.

Regarding the chart types available in Chartblocks, these are limited, however they cover the most common chart types and probably are more than enough of a variety to cover different user needs. The chart creation, shown in Figure 2.13, is customizable, allowing the user to take control and manage several chart aspects so that he can adjust it to his charting needs. The aspects which the user can manage are: chart type; chart colors; sizes; fonts; and the number of ticks in each axis.

Chartblocks combines the power of HTML5 with D3.js which together allow the creation of charts that work on any browser and device, self-adjusting to the different screen sizes and also to render the charts as Scalable Vector Graphics, so that they can be used on retina screens and for high-quality

\begin{figure}
\centering
\includegraphics[width=\textwidth]{chartblocks.png}
\caption{Snapshot of Chartblocks' chart creation screen.}
\end{figure}

\textsuperscript{17}http://www.chartblocks.com/en/
printed documents.
Once the user is satisfied with the chart he has created he can then save it and use the sharing tools to publish it on different places, such as social media or a web site, using the embed code that the sharing tool presents for that chart.

2.3 Discussion

In this section we will start by presenting, in Section 2.3.1, the summary of the analysis of the medical software applications regarding the requirements listed in Section 1.3. We will also present an analysis of the data visualization tools based on comparisons between each of them.

2.3.1 Medical software applications

Table 2.1 summarizes the software applications described in this chapter using, as comparison criteria, the requirements identified in Section 1.3.

<table>
<thead>
<tr>
<th>Software Application</th>
<th>Visualization of Medical Information</th>
<th>Patient health condition evolution</th>
<th>Graphic Availability</th>
<th>Customizable</th>
</tr>
</thead>
<tbody>
<tr>
<td>My MedicineOne</td>
<td></td>
<td>N/D</td>
<td>N/D</td>
<td>N/D</td>
</tr>
<tr>
<td>eClinicalWorks</td>
<td>EMR/PM</td>
<td>✅</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td></td>
<td>CCMR</td>
<td>✅</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>AllScripts Professional EHR</td>
<td></td>
<td>N/D</td>
<td>N/D</td>
<td>N/D</td>
</tr>
<tr>
<td>Kareo</td>
<td></td>
<td>N/D</td>
<td>N/D</td>
<td>N/D</td>
</tr>
<tr>
<td>MediTouch EHR</td>
<td></td>
<td>N/D</td>
<td>N/D</td>
<td>N/D</td>
</tr>
<tr>
<td>NextGen Healthcare</td>
<td></td>
<td>✅</td>
<td>✗</td>
<td>✓</td>
</tr>
</tbody>
</table>

Table 2.1: Requirements fulfilled by the medical software applications.

As we can observe in Table 2.1, the majority of the analysed medical software applications do not show medical information through graphics, being therefore not defined (N/D) for this requirement. Regarding the medical software applications that present medical information through graphics, we can see that all of them enable monitoring a patient’s health condition evolution, for example, the evolution of a patient’s blood pressure based on data collected from previous appointments. Considering the graphic availability, in all of the applications, these graphics are only accessible through a single area, the generic dashboard. In addition, only NextGen Healthcare enables the physician to create customizable graphics, being possible to choose the type of graphic and the data source that contains information, for example. Therefore, we conclude that none of the analysed medical software applications fulfills all of the requirements.
2.3.2 Data visualization tools

Table 2.2 presents a comparison of the several data visualization tools according to some important parameters for this thesis.

<table>
<thead>
<tr>
<th>Type</th>
<th>User</th>
<th>Chart Types</th>
<th>Customization</th>
<th>Data source</th>
</tr>
</thead>
<tbody>
<tr>
<td>D3.js</td>
<td>JavaScript Lib</td>
<td>Developer</td>
<td>Custom</td>
<td>Yes - through programming</td>
</tr>
<tr>
<td>Highcharts</td>
<td>JavaScript Lib</td>
<td>Developer</td>
<td>Fixed</td>
<td>Yes - through programming</td>
</tr>
<tr>
<td>Tableau Public</td>
<td>free software</td>
<td>Non-developer</td>
<td>Fixed</td>
<td>Yes - Chart type, color, size</td>
</tr>
<tr>
<td>Chartblocks</td>
<td>online chart builder</td>
<td>Non-developer</td>
<td>Fixed</td>
<td>Yes - Chart type, color, size</td>
</tr>
</tbody>
</table>

Table 2.2: Comparison of the data visualization tools.

As we can observe in Table 2.2, all of the data visualization tools enable chart customization when creating a chart, being possible to customize several parameters like, for example, the size, background color, chart color and text labels. All of these tools allow data extraction from databases which is crucial for this thesis, however, Chartblocks only allows it on a paid version. So, now considering D3.js, Highcharts and Tableau Public, from these three only D3.js offers non-limited chart types. Highcharts and Tableau Public only offer some predefined chart types to choose from, being therefore more limited than D3.js.

Concluding, D3.js is the best choice of a data visualization tool to use in this thesis.
Chapter 3

Solution

The purpose of this thesis is to develop a medical data visualization module for *Umedicine* that graphically presents to the physician all the required patient data to aid the physician when performing the diagnose of a patient.

3.1 Proposed Solution

This section describes the initially proposed features for this module, in Section 3.1.1, as well as presents its corresponding infrastructure, in Section 3.1.2.

3.1.1 Functionalities

To assure that the requirements, described in Section 1.3, are met, the medical data visualization module should have the following main functionalities:

- *Generic Dashboard*: This dashboard will contain graphics over some parameters regarding the whole population for the physician to analyse. It will be possible for the physician to search and filter the shown graphics, for example, if instead of analysing the whole population, the physician wants to analyse a specific subset of it. When accessing this dashboard, it will display some initial graphics, allowing the physician to add saved graphics (previously created by the physician) that he/she considers important to the ones initially shown or remove graphics that were being shown, so that when the physician accesses this dashboard again, the initial page will present the graphics he/she considers important. It will also allow the creation of new graphics as well as the deletion of graphics that are not needed anymore from the database.

- *Dashboard of a patient*: When accessing this dashboard, physicians will be shown that patient's medical data through graphics. This dashboard will be accessible from the patient page. This patient dashboard will display some initial patient graphics which will be customizable, allowing the addition of saved graphics to the dashboard or removal of current ones that the physician intends that are not important, keeping this way only the important graphics in its initial page. It will
also allow the physician to search for more graphics of that patient as well as the creation of new graphics as well and the deletion of graphics that are not needed anymore.

3.1.2 Infrastructure

This medical data visualization module complies to the following infrastructure, presented in Figure 3.1.

![Diagram of infrastructure](image)

Figure 3.1: Infrastructure of the medical data visualization module.

There will be applications or web browsers running the medical data visualization module and accessing the dashboards. These dashboards will need to access specific information and therefore will send requests over the Internet to the web server, which contains the code of the medical data visualization module. The web server receives the requests and in order to answer them, it will execute SQL queries over Umédicine’s relational database, retrieving data from it. Once the web server obtains the requested data, it will send a response with this data over the Internet to the application or web browser that requested it.

3.2 Developed Solution

This section contains all the information regarding the development of our solution, presenting the final solution with all the functionalities that are currently available to use.

3.2.1 Technologies

This section presents all the programming languages, libraries and frameworks applied during the development of our solution: HTML, CSS, JavaScript, jQuery, D3.js and the Spring framework. I will describe shortly each one of them.
**HTML**

HTML was one of the many languages used to develop the solution. It was used with the purpose of structuring the page where the information to be presented would appear and how it would appear.

```html
<div class="ui-field-container">
  <label for="select-native-4">X-Axis</label>
  <select name="select-native-4" id="select-native-4" onclick="namegen();">
    <option>Escolha...</option>
    <option id="T4">Tempo</option>
    <option id="VP4">Volume Préstata</option>
    <option id="PSAT4">PSA total</option>
    <option id="Qmax">Qmax</option>
    <option id="Omax">Omean</option>
    <option id="F4">Flow time</option>
    <option id="IPSS">IPSS</option>
  </select>
</div>
```

Figure 3.2: Example of the use of the HTML language for webpage structuring.

Figure 3.2 shows an example of the use of the HTML language for webpage structuring, where in this case it is shown the creation of a dropdown menu with some predefined options that can be chosen for the x-axis of a graphic.

**CSS**

CSS language defines the style of the elements and of the HTML document. This language describes how these elements should be displayed.

CSS is very important as it enables the modification (and factorization) of the way how similar elements are shown in different web pages just by changing the style of each element within the file, instead of having to change the element in each web page where it appears.

```css
.axis line {
  fill: none;
  stroke: #000;
  shape-rendering: crispEdges;
}

.line {
  fill: none;
  stroke: red;
  stroke-width: 1.5px;
}
```

Figure 3.3: Example of the use of the CSS language for HTML element styling.
Figure 3.3 shows an example of the use of CSS to style the way the axis of the graphs would appear and also the way that the lines of values within the graphic would be displayed, in this case it was defined as with the color red and with the width of 1.5 pixels.

JavaScript

JavaScript language is, very shortly, the programming language of the Web. JavaScript language was essential for the development of our solution as it was the language used to manage the events within a web page and add the desired functionalities.

```javascript
function apagarSepador(name) {
    var criados = JSON.parse(localStorage.getItem("nomestabs"));
    var separador = name;
    // var sepensor = JSON.parse(localStorage.getItem("nomestabs"));
    if (confirm("¿Tienes seguro que quieres eliminar?")) == true {
        if (criados !== null) {
            $.get('/pacientes/dashboard/removeSep', {nameSep: separador}).done(function() {
                location.reload();
                // $(separador).remove();
            }).fail(function() {
                alert("error erasing!");
            });
        }
    }
}
```

Figure 3.4: Example of the use of the JavaScript language for the definition of a function.

Figure 3.4 presents an example of a Javascript function developed in our solution. This function corresponds to the delete tab function which enables the user to erase a previously created graphic. It is also possible to see some additional verifications within the code of the function to ensure it works as it is supposed to.

jQuery

jQuery is a very fast and full of features JavaScript library. It serves as a great help, with benefits in manipulating HTML documents, event handling, animation and Ajax requests for example.

In the development of this solution, jQuery was used essentially in two ways:

- **HTML elements handling:** with the id of the corresponding HTML element it is possible to get the corresponding element using jQuery and then do the corresponding modifications or even remove it.

- **Ajax requests:** this is one of the most essential uses of jQuery as it makes Ajax requests such as posts or gets very simple. Without, for example, the Ajax get requests it would have not been possible to develop this solution as it is mandatory in order to communicate with Spring framework, in which Umecidicine is being developed.
Figure 3.5: Example of the use of the jQuery library for an Ajax get request.

Figure 3.5, shows an example of an Ajax request using jQuery, which in this case is calling a specific Spring controller which corresponds to the one with the defined URL and then, after the request is complete, the code within the done function is executed.

D3.js

D3.js as stated in Section 2.2.1, is a free and open-source javascript library that allows the manipulation of documents based on data, having the purpose of displaying this information through various types of graphics.

D3.js library is the body of all the solution and what made it possible to turn numeric information into graphical information. With D3.js was created everything that is graphical within the solution, more concretely, from the definition of the graphic’s axis to lines of values and its legend.

There are no limits to what is possible to create with this very powerful library. It provides a huge variety of functions which allows the programmer a high level of customization when deciding on how he wants to present the data.

```javascript
$.get("/pacientes/dashboard/removeSep", {nomeSep: sepaapagar}).done(function(){
    location.reload();
    // $(sepaapagar).remove();
}).fail(function(){
    alert("error erasing!");
});
```

Figure 3.6: Example of the use of the D3.js library for importing JSON data.

Figure 3.6 shows an example of data import from the database in JSON. The `d3.json` function receives as arguments an url and also a callback function as an optional argument, and it returns a request to get the JSON file at the desired URL. The JSON data is then stored within the `params` array.

```javascript
d3.json(datafile, function(error, params) {
    if(error){
        console.log("error");
        return;
    }
    params.forEach(function(d) {
        d.data = parseDate(d.data);
        d.valor = +d.valor;
    });
});
```
The Spring Framework makes it very easy to develop Java-based applications and to deploy them on any platform. This framework is the foundation of the Umedicine application and for that reason, in order to integrate the graphical information which was being developed with D3.js into Umedicine, it was also needed to make some modifications and adaptations to the original code in order for it to work with the framework, such as the creation of controllers.

```java
@Overwrite
public List<Parametros> getVolProxt(String nProcesso)
{
    SELECT ("*").FROM("exam_realizouexamevalor NATURAL JOIN exam_realizouexame").WHERE("nprocesso = :nProcesso and
    putParameter("nProcesso", nProcesso);
    return execQuery((ResultSet rs) -> {
        List<Parametros> resultado = new ArrayList<>();
        while(rs.next())
        {
            String nprocesso = rs.getString("nprocesso");
            int idExe = rs.getInt("idRealizouExame");
            Date date = rs.getDate("data");
            String val = rs.getString("valor");
            String nomePar = rs.getString("nomeParametro");
            resultado.add(new Parametros(nprocesso, idExe, val, data, nomePar));
        }
        return resultado;
    });
}
```

Figure 3.7: Example of the definition of a DAO(Data Access Object) function.

Figure 3.7, shows the definition of a DAO(Data Access Object) function where we can see it executes an SQL query over the database to extract the data needed for the graphic and then in the end it creates an object, for each line of data, with the obtained values.

### 3.2.2 Code development

This section explains the essential steps which were taken when developing the code for the solution.

**Entity creation**

In order to develop our solution, first of all it was needed to create an entity that would store the data gathered from the database when extracting it through SQL queries. We developed an entity called *Parametros* that contains as attributes the most important parameters we needed to retrieve from the database, which were the following:

- *nprocesso*: patient's process number;
- *idExame*: ID of the exam;
• **data**: date of the gathered value for the parameter;

• **valor**: value of the parameter;

• **nomeP**: name of the parameter to be retrieved.

This entity also contains the essential methods in order to be able to use it correctly such as get and set functions for each of the parameters and two constructors: one for creating an instance with only the name of the parameter, and the other one to create an entity with values assigned to all its attributes.

**DAO functions**

We needed to create DAO functions in order to actually retrieve the data from the database. Therefore we created *ParametrosPacienteDAO* and *ParametrosPacienteDAOImpl* classes. The *ParametrosPacienteDAO* class is an interface containing all the methods required. The *ParametrosPacienteDAOImpl* class implements the previously mentioned interface.

There were several functions created in order to fetch the data for each parameter and also to fetch data for the graphics’ legends.

Essentially these functions execute SQL queries over the database and then that information is stored in a *ResultSet*. This *ResultSet* is a list in which each element contains one record of the data retrieved by the query. We then search this list and create a list of *Parametros* which contains in each element the data needed to construct the graphics. Finally, we return this list of *Parametros*.

**Services**

We created a service class, named *ParametrosPacienteService*, which essentially has several methods, each of which calls a specific method within the already developed DAO class. For this purpose, this service class has an instance of the *ParametrosPacienteDAO* interface.

**Web Service**

We created a Web Service class named *DashboardPacienteWebServices*.

This class is the one responsible for triggering all the process of data gathering from the database. For this purpose, it contains an instance of the *ParametrosPacienteService*. Then, in each of its functions, it calls a specific service (depending on the data we need to obtain), stores the result in a list of *Parametros* and returns it.

Spring is essential in this web service class, as with it, it is possible to make every function produce JSON results. This is essential for the creation of graphics, as with the D3.js library, we need the data to be structured in JSON. Spring also allows us to define a value (url) for each function, which is also very important, as it is needed in order to correctly call D3.js methods.
Using D3.js for the creation of graphics

We created and modified two javascript pages, one for the dashboard and the other one, which was already created, for historic. Within each of these pages we then started the process of creating graphics with the D3.js library. The process of creating a graphic with D3.js, essentially complies three steps:

- SVG creation
- Using D3.json function
- Adding a legend

Regarding the first step - SVG creation - this is the step that defines the area which the graphic will occupy. SVG is the rectangle where the graphic with all its components will fit. For this step, we used the d3.select method, in which we specify the element where we wish to append the svg, and then we specify the attributes for that svg, such as the id, width and style. This method basically allows us to define an html element with the attributes we desire.

The second step - Using D3.json function - is the most essential one for creating a graphic. The D3.json function requires as arguments an URL and an option callback function.

With the D3.json function we define all the characteristics of a graphic. The URL that the function requires as argument is the value we specified for the each function of the Web Service and in the callback function we put a function with the array we wish to store in the data from the database. Next, we search the array and parse the date to the desired format and transform the values which come in string format to integer. Then we define the range of values for the X and Y axis. This is based, of course, in the values we obtain from the data extracted from the database.

Next, we define the axis themselves with the method nameofthesvg.append(‘nameoftheelement’).attr(...). In order for these axis to be drawn and position correctly, we need to have previously defined variables with the area of the axis, the type of scale, the orientation and the tick format, for example YYYY-MM-DD.

After all these processes, we add the line of values to the graphic, where we specify the data we want to use to draw the line, with svg.append(“path”).datum(array), and format it by adding specific attributes.

Then, to add the points to the line of values, it required some good extra hours to understand how it should be implemented and what needed to be defined such as, the radius for the points and the scale that affects them.

For the third step - Adding a legend - we needed to have previously defined some colors, stored in variables, so that we could use them when adding each rectangle of the legend to the graphic. These colors were specified in RGB. With the method legend.append(‘rect’).attr(...), we defined the format of the legend: width, height and fill color. Additionally, in the case of clickable legends, we also added some extra verifications in order to see which rectangle of the legend was being clicked, in order to show the correct line of values.

Finally we also had to add some functions to resize the graphics correctly when, for example, the user...
resizes the window. These functions, very briefly, gather the new values of the window’s dimensions and apply them to the graphics that are being displayed.

Refactoring

The methods for creating an svg, creating the corresponding graphic and resizing the graphics were refactored, so that we would not have to use the same code various times. For this process, it was needed to conduct a deep analysis on what was common from graphic to graphic, in order to join that into a function that could be called in the future. Of course this, as expected, brought more difficulties such as for example, graphics with more parameters would need more data from the database and therefore we needed to add some extra verifications.

Although, from all the refactored methods, the one considered most challenging, was refactoring the creation of legends. The method of creating legends seemed very strict and it required a long analysis process in order to define the best way to refactor this process. The fact that graphics can have different number of parameters, is of course, a challenge when refactoring this function. We came to the conclusion that the parameters should be stored in an array and also the colors to be applied in each rectangle of the legend. The most complex step was applying the arrays correctly within the process of creating the legend. We then added a for each loop that would search all the parameters of the graphic and for each one would have to create a rectangle for the legend with the same color as the line of values for that parameter within the graphic. This is where the array of colors was used. There were some more steps taken in order for this to work correctly and after all of the analysis and testing was done, the function ended up by working as intended.

3.2.3 Functionalities

This section presents all the working functionalities of our solution. This section is divided in two subsections: the Dashboard and the Historic.

Dashboard

The Dashboard is the core functionality of the solution. It is available in the patient’s page of Um medicine, which presents that patient’s medical information.

By default, the Dashboard page presents five predefined tabs. The first four tabs correspond to predefined graphics that were confirmed relevant by the physician collaborating on this project. The fifth tab allows the user to create another graphic showing the variation of one or two parameters of the patient’s data. The user-specified graphics can be persistently added to the dashboard or they can be seen only once.
The four main predefined, which can be seen in Figure 3.8, graphics are: Qmax, PSAs, Prostate Volume and Questionnaire. Figure 3.8 presents the questionnaire graphic, which possesses a clickable legend. Each of these graphics is presented over time, which means that the x-axis in all four tabs corresponds to the time dimension (date).

When accessing the fifth tab, named *Criacao de grafico*, the physician is presented some dropdown menus with predefined options for the x and y axis. It is also possible to monitor the variation of two parameters over time in the same graphic by adding an option from the dropdown menu to both y-axis. Once the parameters are selected, a name for the graph is automatically generated which the physician can modify if he/she thinks it's not suitable.

In order for the graphic to be created, it is then needed to click on the button which says "Criar", which will display the graphic according to the selected parameters and, if the physician wants to save that graphic for posterior uses/consults all that needs to be done is to click the "Guardar" button. After a graphic is saved, a new tab appears with the name of the graphic which was created and it is possible to delete that tab if the physician considers that the graphic is no longer important or relevant to keep.

Figure 3.9 presents a graphic that was created by the physician to monitor the evolution of the Flow-time against Time. As can be seen, after clicking on the "x" button to delete the tab, a dialog box appears in order to confirm the physician desire to erase that graphic from the dashboard.

For as long as the physician does not erase the tab, it will appear in the posterior accesses to the patient's dashboard.
Historic functionality consisted on allowing the physician to consult the variation of values for the parameters of a specific exam. Each parameter is associated to an urologic subcategory and their values were only shown by means of a table.

To this functionality that already existed, it was added the possibility of checking the variation of one or more parameters through graphics. This graphical functionality is divided in two main features. The first feature allows monitoring the variation of the parameters belonging to a specific urologic subcategory and the second feature allows consulting the variation of one or more selected parameters.

In order to develop this functionality it was needed to gather the values presented in the table. For this, we had to create several arrays where, when creating the table, we would store those values in the arrays. Then, for the first feature, we had to create buttons, which generate a pop-up, for each urologic subcategory with HTML and CSS, and verify, with JavaScript, if any of those buttons was being clicked and which button it was. Finally, considering the chosen subcategory, it is called a specific function that then gathers the desired parameters values (based on the subcategory) from the arrays and creates the graphics, taking into consideration the range of values for the determination of the x-axis and y-axis, lines of values, points and legends.

For the second feature, we had to associate checkboxes to every parameter and a general one that would select all of the others. Additionally, we also had to create a create button that would generate a pop-up which would show the variation of the selected parameters. This was done with HTML and CSS languages. Then, it was created an array that would contain the names of the selected parameters. Finally, it was associated a function to the create button that, essentially, verifies which parameters are in the array and then gathers the values for those parameters. After that, the function creates the axis, points, lines of values and legends that are part of the graphics, displaying them within the pop-up.
The access to this functionality is done through the patient page, under the “Exames Auxiliares de Diagnostico” section, by clicking on the desired exam over which the physician needs to monitor the evolution of one or more parameters. Following this step, a new page will appear presenting a table with the last parameters’ values and the physician needs to click the “Historico” button in order to go for the feature’s page.

Figure 3.10: Example of the screen that appears after the selection of a specific exam.

Figure 3.10 presents the first screen that appears when accessing the historic functionality, which displays a table with the parameters of that exam and the values for each one. The values that appear correspond to the values gathered in the last exam.

As can be seen in the figure, the page provides a button named “Historico” which after clicking will lead to the page where it is possible to track the evolution of the parameters.

Figure 3.11 presents the Historic screen. Within this figure it is possible to verify the two, previously mentioned, new features: monitorization of the variation of the parameters belonging to a specific urologic subcategory; display of the variation of one or more selected parameters.

The first feature is accessed by clicking on any of the buttons with the image of a graphic. The second feature is accessed by checking the desired parameters and clicking on the "Criar grafico" button in the bottom left of the page.

For both features the graphics shown in the generated pop-up are grouped by unity of measurement. This means that, for example, all the parameters that have the unity “ng/mL” will appear in the same graphic. Additionally, each page of the pop-up may contain up to four graphics and, in case there are more than four graphics, it is generated another page. The physician may navigate through the pop-up pages freely.
Figure 3.11: Example of the screen for the Historic functionality for a specific selected exam.

Figure 3.12: Example of the pop-up window displaying the evolution of the historic for all the parameters.

Figure 3.12, presents the pop-up window which is generated after selecting all the parameters and clicking the button to create the graphic. It is also possible to verify what was already mentioned before, which is the grouping of the parameters by their unity. In this case, the pop-up window contains more than one page as each page only presents four graphics.
Chapter 4

Validation

This chapter refers to the evaluation and validation of the developed solution. It starts by describing the type of evaluation conducted and also what was evaluated. Finally, it describes the results of the evaluation, concluding on these results.

4.1 Validation of the user interface

To validate and evaluate this module’s user interface, we performed tests with users. The user tests evaluation was based in the methodology presented in “Task Centered User-Interface Design: A practical introduction” by Clayton Lewis and John Rieman. This evaluation was performed by ten users with ages between 20-50 years old, without any medical background. Initially, it was provided to the users a brief explanation of the purpose of the evaluation and also the purpose of the application and of what was developed. Then, these users were given tasks to execute within the Medical Data Visualization Module, where, for each task, we used as performance metrics, the following: the time needed to complete it; number of help requests; number of mouse clicks; and the number of errors committed. Before executing the tasks, none of the users had had any previous contact with application.

After each task was completed the users were asked to fulfill a questionnaire where for that task they would evaluate the difficulty on understanding the purpose of the task and the difficulty on completing it. In the end, the users filled a satisfaction questionnaire where they manifested the utility degree of the user interface.

4.1.1 Test environment

This section describes the environment conditions on which the user tests were performed. The characteristic of the test environment was the following: Well lighthened space; Microsoft Surface pro 3 to execute the tasks; Wireless connection to the internet with 200Mb/s download and 200mb/s upload; Provided a sheet with the tasks to execute; Each user performed the tasks in an individual and isolated way and in the same order.
4.1.2  User tasks

This section presents the tasks that were performed by the users during the evaluation of the user interface of our solution. These tasks are representative of the tasks that the end users of the system will need to perform in order to visualize the patient data graphically.

Every user started each task from the patient page. After the conclusion of each task, the user went back to the patient page.

Task 1

- **Definition:** Check the evolution of the patient's Qmax.

- **Solution:** The user must access the "Dashboard" tab in the patient page and then access the "Qmax" graphic tab.

- **Purpose:** The purpose of this task is to evaluate the good performance of the already predefined graphics within the Dashboard function, in this case, the "Qmax" graphic.

Task 2

- **Definition:** Check the evolution of "volume da prostata" and Questionario IIEF and IPSS

- **Solution:** The user must access the "Dashboard" tab in the paciente page and click in the "Volume Prostata" tab and then access the "Questionarios" tab and select in the graphic's legend the checkboxes corresponding to "IIEF" and "IPSS".

- **Purpose:** The purpose of this task is to evaluate the performance of the user according to the metrics, described in 4.1, when visualizing already predefined graphics within the Dashboard function, in this case, the "Volume Prostata" graphic and the "IIEF" and "IPSS" questionnaires' graphic.

Task 3

- **Definition:** Create and save two graphics. The first one to analyse the variation of Flow time across time and the second one to analyse the relation between volume da prostata and PSA total.

- **Solution:** The user must access the "Dashboard" tab in the pacient page and then click on the "Criacao de grafico" tab and select the dimension "Tempo" for the X-Axis and the parameter "Flow Time" for the Y-Axis. Then, the user must click on the "Criar" button and after the graphic appears the user must click on the "Guardar" button in order to save the graphic into a tab. For the second graphic it is the same procedure but the user must select the "Volume da prostata" parameter for the X/Y Axis and the "PSA total" parameter for the Y/X Axis.
- **Purpose:** The purpose of this task is to evaluate the performance of the user according to the metrics, described in 4.1, when creating and saving a graphic by selecting the parameters he wishes to analyse from the dropdown lists.

**Task 4**

- **Definition:** Check the graphic containing the evolution of the flow time across time.

- **Solution:** The user must access the "Dashboard" tab in the patient page and click on the tab corresponding to the previously created "Tempo-Flow Time" graphic.

- **Purpose:** The purpose of this task is to evaluate the performance of the user according to the metrics, described in 4.1, when accessing a previously user created graphic.

**Task 5**

- **Definition:** Erase the tab corresponding to the graphic containing the variation of flow time across time.

- **Solution:** The user must access the "Dashboard" tab in the patient page and click on "x" button within the "Tempo-Flow time" tab in order to erase it and then confirm its deletion in the pop-up.

- **Purpose:** The purpose of this task is to evaluate the performance of the user according to the metrics, described in 4.1, when erasing a previously created graphic.

**Task 6**

- **Definition:** Check the graphic of Perfil urologico’s historic from the patient's analysis.

- **Solution:** The user must access the "Analises" record within the "Exames Auxiliares de Diagnostico" section of the patient page and then click on the "Historico" button. Finally, the user must click on the "Ver em grafico" button next to the "Perfil Urologico" entry within the table.

- **Purpose:** The purpose of this task is to evaluate the performance of the user according to the metrics, described in 4.1, when he wants to visualize the variation of the parameters within a specific urologic subcategory within the historic of an exam.

**Task 7**

- **Definition:** Check the graphic of PSA total, PSA livre, Sodio, Creatinina, AST, LDH and Colesterol LDL’s historic from the patient's analysis.

- **Solution:** The user must access the "Analises" record within the "Exames Auxiliares de Diagnostico" section of the patient page and then click on the "Historico" button. After this is done, the user must select the indicated parameters and click on the "Criar grafico" button, verifying that the corresponding parameters appear within the generated graphics.
• **Purpose:** The purpose of this task is to evaluate the performance of the user according to the metrics, described in 4.1, when creating a graphic with the desired parameters within the Historic page.

**Task 8**

• **Definition:** Check the graphic of all parameters' historic from the patient's analysis.

• **Solution:** The user must access the "Analises" record within the "Exames Auxiliares de Diagnóstico" section of the patient page and then click on the "Historico" button. Finally the user must click on the "Seleccionar todos" checkbox in the top of the table and then click on the "Criar grafico" button and verify that all the corresponding parameters appear within the graphics.

• **Purpose:** The purpose of this task is to evaluate the performance of the user according to the metrics, described in 4.1, when creating a graphic with all the parameters of the exam and navigating through the different presented pages in the pop-up.

### 4.1.3 Questionnaires

This section explains the purposes of the questionnaires, already mentioned in Section 4.1, which were used during the user tests to validate the usability of the user interface and to evaluate the difficulty on completing the tasks mentioned in Section 4.1.2 with the Medical Data Visualization Module.

The first questionnaire, is a very simple one and was created with the purpose of evaluating the difficulty on understanding the purpose of each task and the difficulty on completing it. To measure the difficulty on completing the tasks it was used the SEQ (Simple Ease Question) which defines a 7-point rating scale to better measure this matter. This questionnaire has a question per task. Whenever the user finishes a task, he has to answer the question concerning the task.

The second questionnaire was created with purpose of evaluating the usability of the user interface and satisfaction of the user with the system, and was based on the already defined CSUQ (Computer System Usability Questionnaire), with some minor changes in order to better adapt to the context of this thesis.

### 4.1.4 Results

This section contains the result of the user tests.

This section is organised into two parts. The first one analyses the metric results collected during the execution of each task. The second part presents an analysis over the results of the questionnaire regarding the usability of the user interface.

---

1. [https://docs.google.com/forms/d/1PuyPVUL4_-7wZ7yk6jWyNA3aSfL3Jt2bTktV0ZUBEA/viewform?edit_requested=true](https://docs.google.com/forms/d/1PuyPVUL4_-7wZ7yk6jWyNA3aSfL3Jt2bTktV0ZUBEA/viewform?edit_requested=true)
2. [https://measuringu.com/seq10/](https://measuringu.com/seq10/)
3. [https://docs.google.com/forms/d/1Wz3wHuoDjUpdeosSTtdwqM8QBDGRsIBcmkJkysyQc/viewform?edit_requested=true](https://docs.google.com/forms/d/1Wz3wHuoDjUpdeosSTtdwqM8QBDGRsIBcmkJkysyQc/viewform?edit_requested=true)
Metrics results analysis

**Time results by task**

Figure 4.1: Table displaying the gathered results for the metric: time to complete the task.

Figure 4.1 presents the average time needed for the users to complete each task as well as the standard deviation of the users’ times over the average time.

**Clicks result by task**

Figure 4.2: Table displaying the gathered results for the metric: number of clicks needed to complete the task.

Figure 4.2 presents the ideal clicks needed for the users to complete each task as well as the standard deviation of the users’ number of clicks over the ideal number.
These results will now be used to conduct an analysis of the metrics for each group of tasks: grouped according to the difficulty of the tasks. There are two groups of tasks: very simple, and simple tasks.

**Very simple tasks**

The tasks that are part of the very simple tasks group are:

- **Task 1** - Check the evolution of a patient’s Qmax.
- **Task 4** - Check the graphic containing the evolution of the flow time across time.
- **Task 5** - Erase the tab corresponding to the graphic containing the variation of flow time across time.
- **Task 7** - Check the graphic of PSA total, PSA livre, Sodio, Creatinina, AST, LDH and Colesterol LDL’s historic from the patient’s analysis.

During the realization of any of these four tasks, no user committed any error or asked for any help.

**Time results - Very simple tasks**

![Time metric results for each one of the four very simple tasks](image)

**Figure 4.3**: Time metric results for each one of the four very simple tasks.

*Figure 4.3* presents the average time needed for the users to completed the previously named tasks as well as the standard deviation over that average time.
Figure 4.4 presents the ideal clicks needed for the users to complete the previously named tasks as well as the standard deviation of the users’ clicks over those ideal clicks.

Analysing the results from both of the figures, figure 4.3 and figure 4.4, we can see that these tasks were concluded very quickly, being the maximum average time 29 seconds corresponding to task 7 and the standard deviation around 3 seconds. Additionally, there was no standard deviation when comparing the number of clicks needed and the ideal clicks for any of these tasks, which affirms that all users completed the tasks using the ideal number of clicks.

Therefore, having analyzed these results we can conclude that the users considered these tasks very simple to execute.

Simple tasks

The tasks that are part of the simple tasks group are:

- **Task 2** - Check the evolution of volume da prostata and Questionario IIEF and IPSS.
- **Task 3** - Create and save two graphics. The first one to analyse the variation of Flow time across time and the second one to analyse the relation between volume da prostata and PSA total.
- **Task 6** - Check the graphic of Perfil urologico’s historic from the patient’s analysis.
- **Task 8** - Check the graphic of all parameters’ historic from the patient’s analysis.
During the realization of any of these four tasks, no user asked for any help.

However, when analysing the number of errors committed when performing these four tasks, we obtained 3 users which committed errors.

One of the users committed one error during the realization of task 2. This user’s error was the fact that he clicked on another section in the patient page (the diagnostic exams section) as it showed a “volume prostata” parameter in one of the exams, however the user quickly understood, due to the information presented, that that page was not the correct one, and remembered that the Dashboard contained in one tab that specific graphic.

The other two users committed errors when performing task 6. One of these two errors was due to lack of concentration of the user, as the user clicked by mistake in the wrong exam within the diagnostic exams section and once, he accessed the wrong page, he understood that he had clicked on the wrong exam and then clicked on the correct one. The other error was the fact that the user clicked on the Dashboard tab instead of the analysis exam in the diagnostic exams section, the user then re-read the task and understood quickly what he had to do.

Figure 4.5: Time metric results for each one of the four simple tasks.

Figure 4.5 presents the average time needed for the users to completed the previously named tasks as well as the standard deviation over that average time.
Figure 4.6 presents the ideal clicks needed for the users to completed the previously named tasks as well as the standard deviation of the users' needed clicks to conclude the tasks over the ideal clicks.

Analysing the results from both of the figures, figure 4.5 and figure 4.6, we can see that these tasks were concluded very quickly, being the maximum average time 37 seconds corresponding to task 3 and the standard deviation around 5 seconds. These tasks, however required, in average, more time to complete when comparing to the very simple tasks, as for example, task 3 required significant number of clicks to complete, and also the tasks were slightly more complex, which lead to the fact that users spent an additional time to think of what to do, or to errors when performing these tasks, increasing the time taken to complete the task.

Additionally, there was only a slight standard deviation when comparing the number of clicks needed and the ideal clicks for any of these tasks. These cases correspond to the users that committed errors when performing the tasks which caused them to need extra clicks in order to complete the task. Therefore, having analyzed these results we can conclude that the users considered theses tasks simple to execute.

**Usability of the user interface and user satisfaction questionnaire**

This section contains a summary of the results obtained with the Usability of the user interface and user satisfaction questionnaire.

This section starts by presenting all the questions within the questionnaire and after, it conducts an analysis on the average results for all the questions.
Questions

This section describes all the questions within the questionnaire which was based on the already defined and known CSUQ (Computer System Usability Questionnaire) with some adaptations. This questionnaire is composed by 13 questions which are the following:

- 1. Overall, I am satisfied with how easy it is to use this system.
- 2. It was simple to use this system.
- 3. I can efficiently complete my work using this system.
- 4. I am able to complete my work quickly.
- 5. I feel comfortable using this system.
- 6. It was easy to learn to use this system.
- 7. The information provided with this system is clear.
- 8. It is easy to find the information I needed.
- 9. The information is effective in helping me complete the tasks.
- 10. The organization of information on the system screens is clear.
- 11. The interface of this system is pleasant.
- 12. I like using the interface of this system.
- 13. Overall, I am satisfied with this system.

Analysis of the results

This section starts by providing a summary of the results obtained during the user tests evaluation on which will be conducted a brief analysis. It then presents some of the questions within the questionnaire that greatly contributed to the results of the questionnaire, grouped according to their classification: Excellent and Very good.

Overall analysis

Figure 4.7, presents a summary of the results of the questionnaire. This figure was generated by calculating the average percentage (%) of each classification grade for all the 13 questions, presenting therefore the most frequent classifications throughout the entire questionnaire.
Through the analysis of the results displayed in figure 4.7 it can be concluded that the user tests were very satisfying as the two best possible grades, which correspond to grade 6 and 7, are the ones with the most percentage throughout all the questions. Grade 7, which is the best grade, clearly dominates the results of the questionnaire with 61.9%, which is almost the double of the average percentage of grade 6 (37.4%).

**Excellent classification questions**

Some of the questions that contributed to the large percentage assigned to grade 7 (61.9%), when analysing the results of the questionnaire were the following:

- 6. *It was easy to learn to use this system.*
- 12. *I like using the interface of this system.*
- 13. *Overall, I am satisfied with this system.*

Following are the results of questionnaire for these three questions. These results were obtained from google drive, which offers a functionality that generates graphics considering the answers of the users for each question presented.
6. It was easy to learn to use this system
10 respostas

Figure 4.8: Results of the questionnaire regarding question 6.

12. I like using the interface of this system
10 respostas

Figure 4.9: Results of the questionnaire regarding question 12.

13. Overall, I am satisfied with this system
10 respostas

Figure 4.10: Results of the questionnaire regarding question 13.
By analysing the images that present the results for each of these three questions, we can affirm that:

- Our solution is easy to learn: supported by figure 4.8;
- Our solution offers a user-friendly interface: supported by figure 4.9;
- Overall, the users were satisfied with the solution: supported by figure 4.10.

**Very good classification questions**

Some of the questions that contributed to the percentages assigned to grade 6 (37.4%) and grade 5 (0.7%), when analysing the results of the questionnaire were the following:

- 1. *Overall, I am satisfied with how easy it is to use this system.*
- 2. *It was simple to use this system*
- 8. *It is easy to find the information I needed.*

Following are the results of questionnaire for these three questions. These results were obtained from google drive, which offers a functionality that generates graphics considering the answers of the users for each question presented.

**1. Overall, I am satisfied with how easy it is to use this system**

10 respostas

![Figure 4.11: Results of the questionnaire regarding question 1.](image)
2. It was simple to use this system

By analysing the images that present the results for each of these three questions, we can affirm that:

- Our solution is easy and simple to use: supported by figure 4.11 and figure 4.12;

8. It is easy to find the information I needed

- Our solution presents clear and intuitive user-interface: supported by figure 4.13.
Chapter 5

Conclusions

This chapter describes, in Section 5.1, the main conclusions from the work developed in this thesis, presenting the final solution for the medical data visualization module, incorporated in the Umedicine application.

Section 5.2, describes some possible functionalities to be added to our solution in its future versions.

5.1 Summary

In this thesis we proposed a Medical data visualization module for the Umedicine application. Our solution has as purpose allowing the physicians to access a patient’s medical data through graphics. By presenting the information gathered through graphics, physicians can quickly analyse the evolution of the desired parameters during a patient’s appointment, supporting them in the decision of the best treatment to be applied. The information gathered in order to create the graphics of the medical data was extracted from the Umedicine’s database.

The main objective of this thesis was to address the conclusions of the analysis from the State of the art which stated that: the majority of the medical software applications provide graphics focused on the administrative aspects and when considering the medical data, they are very limited. In order for our solution to deliver a good graphical analysis over the medical data, we used as data visualization tool, the D3.js library, which is a powerful library and complies to the requirements described in Section 1.3, being highly customizable.

The solution developed in this thesis was the following:

- Medical data visualization module with following features:

  *Dashboard*: the Dashboard page presents five predefined tabs. The first four tabs correspond to predefined graphics that were confirmed relevant by the physician collaborating on this project. The fifth tab allows the user to create a graphic showing the variation of one or two parameters of the patient's data. The user-specified graphics can be persistently added to the dashboard of the user or they can be seen only once.
Historic: To the historic functionality that already existed, which consisted in allowing the physician to consult the variation of values for the parameters of a specific exam through a table, we added the possibility of checking the variation of one or more parameters through graphics. This graphical functionality is divided in two main features. The first feature allows monitoring the variation of the parameters belonging to a specific urologic subcategory and the second feature allows consulting the variation of one or more selected parameters.

Finally, in order to validate the solution developed and its functionalities, we conducted a user tests evaluation covering each functionality within this new module, measuring the usability of the user-interface developed through each user’s performance during these tests and the results of the questionnaires. This user tests evaluation had also the purpose of identifying possible issues within the functionalities developed as well as gather feedback on some possible improvements to the module.

5.2 Future Work

This section describes some of the possible improvements to be considered in the following versions of this medical data visualization module.

The current version of our solution has limitations of customization, when it comes for example, to the graphic types available when creating a new graphic within the Dashboard. The module in its actual state does not provide the user the possibility of choosing what type of graphic he desires to create. It should be possible for the user to customize the following aspects of the graphics regarding the creation of a new graphic:

- **Type of graphic to be created**: it should be possible to the user to select the type of graphic he wants to create from a list with options;

- **Background color**: The user should be able to select the background color for the graphic from a list with options;

- **Color for each parameter**: The user should be able to select which color to assign to each parameter from a list with options;

To allow the customization named in the previous items, it should be added to the creation of graphics’ page some fields, each one corresponding to one of the items listed, and then for each field it should be added a list with the types of graphics available to choose or the colors available. Then, the chosen options from the lists for each field should be passed as an argument to the function that creates the graphic with the chosen parameters, and within that function it should be verified which type of graphic was chosen. If the selected type of graphic was, for example, a bars graphic then the function to create a bars graphic would be called, having as arguments also the chosen background color and parameter’s color. Finally, when creating the graphic, the chosen colors for the background and parameters should
be used in the corresponding functions provided by the D3.js library.

One other current limitation of the Medical data visualization module is that it does not allow the user to export the graphics to JPG or PNG, for example. In the future, it should be possible to export the already created graphics or the user-created graphics to one of this formats. This functionality would allow the user to view the graphics much more quickly and comfortable by just opening the corresponding file, as opposed to having to login in the application to consult the graphic again. For this purpose, it should be added a button that, when clicked, would pop-up a window with some fields to fill, such as, for example:

- *Name for the file*
- *Location to save within the computer*

The values for these fields would then be passed to a javascript that would create a file with the chosen name in the selected directory. This function would also select the HTML element corresponding to graphic and would capture it, incorporating the captured image of the graphic within the file.


