

FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Development of a platform for integrated clinical records of cystic fibrosis patients in a national reference center

Márcia Isabel Reis Teixeira



FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

Mestrado em Engenharia Informática e Computação

Supervisor: Carlos Miguel Ferraz Baquero-Moreno

Second Supervisor: António Fernando Vasconcelos Cunha Castro Coelho

Specialist: Ana Lúcia Cardoso

Specialist: Telma Barbosa

December 12, 2022

Development of a platform for integrated clinical records of cystic fibrosis patients in a national reference center

Márcia Isabel Reis Teixeira

Mestrado em Engenharia Informática e Computação

Approved by . . . :

President: José Manuel de Magalhães Cruz

Referee: Francisco Maria Cruz Nunes

December 12, 2022

Resumo

A medicina personalizada visa fornecer cuidados adequados e individualizados aos pacientes através, por exemplo, da adaptação dos registos de saúde eletrónicos às singularidades de cada doença. Esta personalização é particularmente valiosa no caso de doenças crónicas raras. A fibrose quística é uma doença genética rara que afeta as secreções do paciente. É uma doença altamente complexa com múltiplas terapias em desenvolvimento, que requer um acompanhamento próximo por equipas multidisciplinares de profissionais de saúde, exames médicos regulares, e numerosas terapias farmacológicas e não farmacológicas.

Os sistemas de registos de saúde eletrónicos atualmente utilizados nos hospitais portugueses são os mesmos para diferentes doenças. Esta falta de personalização não satisfaz as necessidades de cuidados de saúde dos doentes com doenças raras, crónicas, complexas e multissistémicas, tais como a fibrose quística. Por conseguinte, é essencial desenvolver um sistema de registos de saúde eletrónicos personalizado que melhor se adapte às particularidades da fibrose quística.

Esta dissertação propõe o desenvolvimento de uma nova plataforma de registos de saúde eletrónicos ajustada às necessidades específicas de cuidados de saúde da fibrose quística. Esta plataforma deverá permitir aos médicos gerir os pacientes e os seus dados pessoais e de saúde, consultas, e exames. Deve ser intuitiva e fácil de utilizar, tornando a entrada e o acesso dos registos de saúde eletrónicos pelos profissionais de saúde com pacientes com fibrose quística mais eficiente, e permitindo uma maior qualidade dos cuidados de saúde.

A plataforma desenvolvida será testada, e pretende-se que seja implementada no Centro Materno-Infantil do Norte (CMIN), um centro de referência nacional para a fibrose quística, com a possibilidade de expansão futura para outros centros de referência nacional fibrose quística.

Keywords: medicina personalizada; fibrose quística; registos de saúde eletrónicos

Abstract

Personalized medicine aims to provide adequate and individualized care for patients through, for example, adapting electronic health records to the singularities of each condition. This personalization is particularly valuable in the case of rare chronic illnesses. Cystic fibrosis is a rare genetic disorder that affects the secretions of the patient. It is a highly complex illness with multiple developing therapies, that requires close monitoring by multidisciplinary teams of physicians, regular medical exams, and numerous pharmacological and non-pharmacological therapies.

The electronic health records systems currently used in Portuguese hospitals are the same for different illnesses. This lack of customization does not meet the care needs of patients with rare, chronic, complex, and multisystem diseases, such as cystic fibrosis. Therefore, it is essential to develop a personalized electronic health records system that is better adapted to the particularities of cystic fibrosis.

This dissertation proposes the development of a new electronic health records platform adjusted to the specific care needs of cystic fibrosis. This platform should enable physicians to manage patients and their personal and health data, appointments, and exams. It should be intuitive and easy to use, streamlining the input and access of electronic health records by healthcare professionals with cystic fibrosis patients, and allowing for a higher quality of care.

The developed platform will be evaluated, and aims to be implemented in Centro Materno-Infantil do Norte (CMIN), a national reference center for cystic fibrosis, with the possibility of future expansion to other cystic fibrosis national reference centers.

Keywords: personalized medicine; cystic fibrosis; electronic health records

Acknowledgements

Na realização desta dissertação contei com o apoio de múltiplas pessoas, às quais quero expressar a mais profunda gratidão.

Começo por expressar os meus agradecimentos ao professor Carlos Baquero-Moreno e ao professor António Coelho, respetivamente orientador e coorientador desta dissertação, por todo o acompanhamento, compreensão, e aconselhamento oferecidos ao longo deste processo. À doutora Ana Lúcia Cardoso e à doutora Telma Barbosa, por toda a disponibilidade, ajuda, e entusiasmo, que tanto ajudou a manter a motivação para a realização do projeto. À Teresa pela companhia e apoio durante todo o projeto que desenvolvemos lado a lado.

Aos meus pais pelo amor e apoio incondicionais que sempre me deram, por sempre acreditarem em mim, e por tudo o que fizeram para criar as condições que me permitiram chegar até aqui. Ao meu irmão por ter estado sempre ao meu lado, por aturar o meu lado de irmã mais velha chata, e por me motivar a ser melhor. Agradeço-vos muito por tudo.

À Debbie e à Xica por todo o apoio e amizade incondicionais, por todas as gargalhadas partilhadas, pela motivação de que precisava nos momentos mais difíceis. À Sara, que mesmo longe esteve sempre perto, por todo o apoio incondicional, por saber sempre o que dizer quando mais precisava, por ser a voz amiga da razão. Ao SCTP por toda a amizade, carinho, e compreensão, por me fazerem rir mesmo nos piores momentos, por me apoiarem durante todo este processo.

A todos o meu mais sincero obrigada. Esta dissertação não teria sido possível sem vocês.

Márcia Teixeira

Contents

1	Introduction	1
1.1	Context	1
1.2	Motivation	2
1.3	Objectives	3
1.4	Document Structure	3
2	State of the Art	5
2.1	Personalized Medicine	5
2.2	Related Works	5
2.2.1	Welldoc	5
2.2.2	SONHO V2	6
3	Methodology	8
3.1	Design Science Methodology	8
3.1.1	Application to the Project	8
3.2	Requirements	9
3.2.1	Functional Requirements	10
3.2.2	Non-functional Requirements	11
3.3	Technologies	11
3.4	Usability Tests	13
4	Implementation	14
4.1	Platform Features	14
4.2	Project Structure	23
4.2.1	Backend	23
4.2.2	Frontend	26
4.3	Addition of Features	28
5	Results	30
6	Future Work	36
7	Conclusions	38
	References	40

List of Figures

3.1	Design cycle in design science research.[22, 13]	9
3.2	Processes in the design cycle.[14, 13]	10
4.1	Patient's profile viewed on a mobile phone screen.	15
4.2	Homepage, with a feedback toast indicating the successful addition of a new patient.	16
4.3	Form to add a patient.	17
4.4	Form to add a professional.	17
4.5	List of adult patients, with one open patient type dropdown and a toast notification indicating the successful transition of a patient.	18
4.6	Patient profile page.	19
4.7	Form to edit patient data on patient profile page.	19
4.8	Patient profile page with complications card expanded.	20
4.9	Modal to add exam on patient profile page.	20
4.10	Patient's appointments list.	21
4.11	Form to add a new appointment.	21
4.12	Patient's therapies page.	22
4.13	Modal to add new medication.	23
5.1	Results of the 1st SUS question ("I think that I would like to use this system frequently").	31
5.2	Results of the 2nd SUS question ("I found the system unnecessarily complex").	32
5.3	Results of the 3rd SUS question ("I thought the system was easy to use").	32
5.4	Results of the 4th SUS question ("I think that I would need the support of a technical person to be able to use this system").	32
5.5	Results of the 5th SUS question ("I found the various functions in this system were well integrated").	33
5.6	Results of the 6th SUS question ("I thought there was too much inconsistency in this system").	33
5.7	Results of the 7th SUS question ("I would imagine that most people would learn to use this system very quickly").	33
5.8	Results of the 8th SUS question ("I found the system very cumbersome to use").	34
5.9	Results of the 9th SUS question ("I felt very confident using the system").	34
5.10	Results of the 10th SUS question ("I needed to learn a lot of things before I could get going with this system").	34
5.11	Guideline for interpretation of SUS score. [21]	35
5.12	Participants feedback on the difficulty of the proposed tasks.	35

Abbreviations

API	Application Programming Interface
CF	Cystic Fibrosis
CFTR	Cystic Fibrosis Transmembrane Conductance Regulator
CMIN	Centro Materno-Infantil do Norte
ECFS	European Cystic Fibrosis Society
EHR	Electronic Health Record
SUS	System Usability Scale
UP	University of Porto
URL	Uniform Resource Locator

Chapter 1

Introduction

This dissertation proposes the development of an electronic health record (EHR) platform adapted to the specific needs of patients with cystic fibrosis (CF). This introductory chapter gives an overview of CF to better understand the needs of these patients and contextualize the dissertation's work, explains the motivation behind the work to be developed, and outlines its main objectives.

1.1 Context

CF is a rare chronic genetic disease of exocrine gland function that involves multiple organ systems. It mainly affects the lungs, with 90% of patients who survive the neonatal period showcasing pulmonary involvement, and with end-stage lung disease being the leading cause of death.[18] Chronic respiratory infections are common, as well as pancreatic enzyme insufficiency, involvement of the liver, kidneys, and intestines, and other associated complications in untreated patients. Physical manifestations of the illness vary widely across patients and with the degree of involvement of the disease and are dependent on several factors such as the patient's age at diagnosis.[18, 15]

CF is caused by a mutation in the cystic fibrosis transmembrane conductance regulator (CFTR) gene.[15] With this protein malfunctioning, secretions such as sweat, digestive fluids, and mucus, that should be thin become thick instead.[24] The diagnosis is usually made by a positive sweat test paired with other risk factors or genetic testing. The median age of diagnosis is 6-8 months, but it varies widely.[18]

There are no known cures for CF, but several treatment methods are used to manage symptoms of the disease and to delay the deterioration in organ function. The main goals of these treatments are to maintain lung function as normal as possible, avoid damage done by infections and thickening of mucus, and manage complications.[18] Some of the possible treatments for different complications are listed ahead.

Mild pulmonary symptoms can be managed at home by increasing the frequency of airway clearance, inhaled bronchodilator treatment, chest physical therapy and postural drainage, increasing the mucolytic agent dornase alfa dose, and using oral antibiotics. Several medications can also be used to treat CF, such as pancreatic enzyme supplements, multivitamins, anti-inflammatory agents, mucolytics, and others. Additionally, certain respiratory or gastrointestinal complications may require surgical therapy, and lung transplantation can be used in the case of end-stage lung disease. Besides these treatments, patients are also encouraged to practice healthy eating, adapted to the specific needs caused by the illness, and an active lifestyle. Due to the variety of symptoms and complex, individual nature of the illness, it is recommended that patients are followed at specialist centers with multidisciplinary care teams, where treatment can be tailored to the individual.[18]

Furthermore, CF patients should be closely monitored by healthcare specialists, with appointments every 2 to 3 months. In these appointments, patients are subject to various screenings and tests (such as pulmonary function testing and oxyhemoglobin saturation), and the healthcare provider may prescribe new therapies and evaluate the results of current ones. Finally, the patient and their parents are educated about the disease and therapies and are provided support for psychosocial issues.[18]

It is clearly shown that CF requires highly specific and personalized care due to its complex, multisystemic characteristics and wide variety of physical manifestations across different patients.

1.2 Motivation

The previous subsection shows that CF requires particular and personalized care due to its complex, multisystemic characteristics and the wide variety of physical manifestations across different patients.

Being a chronic illness affecting mainly the respiratory and digestive systems, complications of CF can have a significant impact on the quality of life of both the patient and their family by limiting the individual's ability to complete daily tasks. Even without complications, there is a notable increase in the emotional stress of the patient and their family, and the time-consuming nature of daily treatments that patients are subject to may further negatively impact their quality of life.[17]

However, according to Rosenstein [16], "with appropriate support, most patients can make an age-appropriate adjustment at home and school. Despite myriad problems, the educational, occupational, and marital successes of patients are impressive.". The data shows that appropriate treatment and support are crucial in minimizing the negative impact of CF on patients' quality of life.

Besides the already existing wide array of treatments for CF, there is ongoing research and clinical trials for new therapies, such as gene therapy and CFTR modulator therapies. Medical professionals must be up to date on newly available treatments to provide the best, most adequate care for their patients.

The motivation for the work developed in this dissertation comes from the clear impact that adequate, personalized, specific care has on patients' quality of life and prognosis. With CF being a rare chronic disease that requires extensive clinical tests and has a substantial variety of possible therapies (and new ones under trial), it is crucial for healthcare professionals to have a system that allows them to easily access all of the necessary information and that is adapted to the specific care needs of these patients, allowing them to provide the best possible care and improve patients' quality of life.

The health information systems currently in use in Portugal are the same across all illnesses and fail to meet the needs of patients with CF. It is crucial to develop a better system, that is able to provide the aforementioned benefits.

1.3 Objectives

This dissertation proposes the development of an EHR platform for healthcare professionals working with CF patients. This work is to be developed in collaboration with Centro Materno-Infantil do Norte (CMIN), a national reference center for CF.

The proposed system is adjusted to the specific needs of monitoring CF patients. It should allow physicians to view and update patient data in an easy, organized, and intuitive manner, providing a better user experience than the systems currently in use. Healthcare professionals should be able to easily view and add patients, view and update their personal data, CF diagnosis data, complications, exams, appointments, therapies, and medication.

The developed solution will then be tested and evaluated by its end users, the healthcare professionals.

1.4 Document Structure

After describing the context and the motivation of the dissertation and explaining its primary objectives in Chapter 1, then this document explores the state of the art in Chapter 2. This chapter analyzes the fundamental concepts related to the dissertation and discusses the existing technologies, related work, and other approaches to the same or similar problems.

Then Chapter 3 delves into the methodology used to develop the proposed project. It also goes over the defined requirements, the technologies chosen for the development and reasoning behind these choices, and how usability tests were performed.

Chapter 4 explains the features implemented, the project's structure, and how developers may add new features to the project in the future.

The results of the usability tests are exposed and analyzed in Chapter 5, discussing what these results imply about the functionality of the developed platform.

Chapter 6 discusses what further improvements and additional features could be added in future iterations of the project in order to increase the benefits offered by the platform and its usability.

Finally, Chapter 7 concludes this document, reflecting on the work done throughout this dissertation, how it fulfilled the proposed objectives, and overviewing how it could improve in the future.

Chapter 2

State of the Art

The present chapter explores the main concepts related to the dissertation's work, and analyzes which tools to solve a similar problem are currently in use.

2.1 Personalized Medicine

CF's particular and complex nature calls for a personalized, individualized approach to care. Thus, it is valuable to explore the concept of personalized medicine.

Personalized medicine aims to provide more adequate medical care adapted to each patient's individual needs, instead of using a standardized, "one-size-fits-all" approach. This concept is currently in expansion, both in Portugal and internationally.

One of the ways to apply this approach is through adjusting EHRs to the singularities of an illness, which is especially valuable in the case of rare chronic diseases such as CF, that require the gathering and analysis of a vast amount of clinical data and have a wide selection of possible therapies, including ongoing clinical trials. Medical professionals can benefit from this EHR adaptation since it helps them streamline their work, allowing them to spend less time looking for and filtering through data and medical exams, and more time focused on the problem at hand, therefore providing better care to the patient.

2.2 Related Works

There are existing tools that attempt to solve similar problems as to the one tackled by this dissertation. This section will analyze these tools, study their features, results, and setbacks.

2.2.1 Welldoc

Welldoc is an American privately held company that provides a platform for personalized, "AI-powered" care of patients with chronic illnesses: they offer products that cater to the specific

needs of patients with type 1 or type 2 diabetes, hypertension, heart failure, and patients that want to reduce the risk of developing type 2 diabetes.[2]

The BlueStar® System and the BlueStar® Rx System are the most well-known and awarded Welldoc products. They are intended to be used by the patient on their mobile phones or personal computers, but also in a professional healthcare context.[2] According to Welldoc's website [2], both systems include the following features:

- "secure capture, storage, and transmission of blood glucose data";
- support of medication adherence;
- analysis and report of blood glucose test results;
- "coaching messages (motivational, behavioral, and educational) based on real-time blood glucose values and trends";
- information for the self-management of diabetes;
- "entry of other diabetes-related healthcare information";
- educational information.

Additionally, the BlueStar® Rx also provides an insulin dose calculator that calculates a dose of insulin based on the patient's prescription "for a given amount of carbohydrates and/or blood glucose value". [2]

2.2.2 SONHO V2

SONHO - Sistema Integrado de Informação Hospitalar, is the system in use in 90% of the institutions that are part of the Portuguese national healthcare system (Serviço Nacional de Saúde). It was launched in 1994, with the main goals of managing patients' administrative data, by monitoring their flow in hospitals, and to access current and historical patient clinical data. Since the system was to be used across the national healthcare system, it also allowed for easy information-sharing between the various institutions.[3]

This system is divided into eight different modules [3]:

- Identification Module;
- Emergency Module;
- Hospitalisation Module;
- Outpatient Appointment Module;
- Operating Room Module;
- Day Hospital Module;

- Archive/Statistics Module;
- Billing Module.

These modules are interconnected and allow for the registration, input, update, and access to patient clinical data related to the context of the module. For example, the emergency module registers, among others, the emergency location, specialty, type of accident, priority. It also allows to, for example, admit the patient to the emergency room services and to transfer them to another emergency care unit.[3]

Recently a new version of SONHO, SONHO V2, has been launched, and it is being gradually implemented across Portuguese healthcare units. CMIN, where the solution proposed by this dissertation is to be implemented, already uses this new version of SONHO. SONHO V2 intends to upgrade some aspects of SONHO that have become outdated, solve data storage and security issues, and introduce new features.[4] According to [4], the updates include:

- Improvements in the User Interface (UI);
- Migration to Oracle Database 11g R2;
- Development of service-oriented integration layer;
- Addition of a reporting database, so that management reports can be obtained without overloading the operational database;
- "Other small features, such as the possibility to use the national identity card [(Cartão do Cidadão)] in the patient identification".

Even the updated version of the SONHO system still presents several issues that add delays and obstacles when caring for patients, especially with patients with highly specific needs, such as those with CF. One of the main issues is that clinical notes are a plain text field, not uniformized or parsed. This implementation makes it difficult for doctors to complete crucial tasks such as tracking the evolution of relevant clinical values or information across appointments or knowing which information they have already collected in previous appointments. Clinical test results are also difficult to access.

As mentioned before, this becomes particularly problematic in the case of an illness such as CF, which requires the collection and analysis of extensive amounts of clinical data. The information is not centralized or readily accessible in one place, which represents a setback in the work of healthcare professionals, potentially having negative effects on the quality of care provided.

Chapter 3

Methodology

This chapter explains the methodology used for the project development, requirements definition, and evaluation of the developed platform.

3.1 Design Science Methodology

The methodology used to develop this project was based on the Design Science methodology. This methodology is mainly used in Computer Science and Engineering, and focuses on developing artifacts to attain defined goals and improve their functional performance.[23, 14] Since this dissertation proposes to find and develop a solution for a practical, real-world problem and requires frequent evaluation and validation by professionals in the healthcare field, this methodology was deemed adequate for the project at hand.

The most commonly used design cycle in design science research is described by Takeda et al. in Figure 3.1.[22] Kuechler and Vaishnavi also defined the processes that make up the design cycle, based on Takeda's proposal, as seen in Figure 3.2.[14]

3.1.1 Application to the Project

This project was developed together with physicians and a multimedia graduate student responsible for prototyping, through several iterations of the illustrated design cycle.

The design cycle begins with an enumeration of the problems and deciding which problem should be solved by the project, in this case done through an initial meeting with the physicians participating as specialists at CMIN. In this meeting, it was possible to interact with the EHR systems in use, and discuss with healthcare professionals, the system's end-users. This discussion focused on understanding how physicians use EHR systems, identifying the problems posed by working with the current EHR systems, and what features in an EHR platform would be the most important to maximize performance.

Then, in the "Suggestion" phase, initial requirements were defined after discussion and approval by the specialists. These requirements represented an initial draft of what were to be the

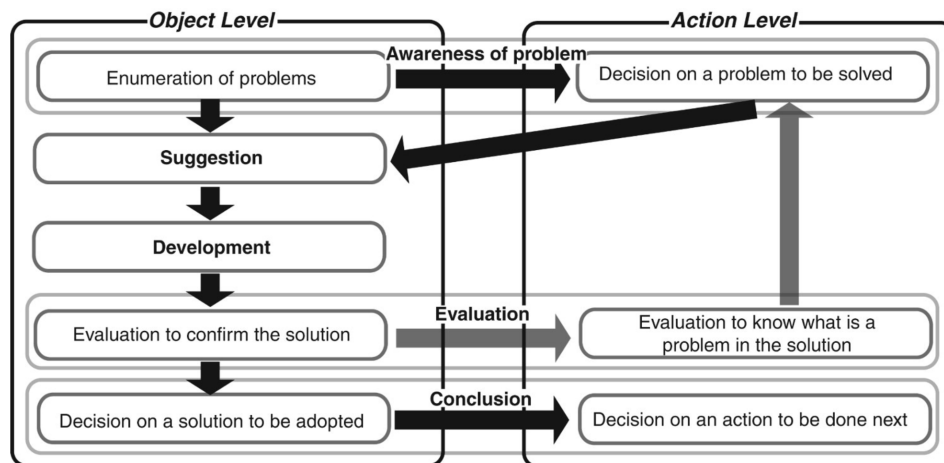


Figure 3.1: Design cycle in design science research.[22, 13]

final requirements of the developed platform. Since this method uses an iterative process, it is possible to refine the requirements and artifacts throughout the project's progress.[14] Refinements were firstly made following the first iteration of development, after which there was a better awareness of technologies, time and technological constraints (such as the inability to integrate with the EHR system used in Portuguese hospitals within the project's time limits). There was a setting of priorities and a more concrete definition of requirements.

The rest of the project's development process was composed of several more iterations of the design cycle, made possible by consistent communication between all parts involved. The physicians would receive updates about the development and feedback about constraints or the need to refine requirements. They would, in turn, collaborate in fine-tuning the requirements, set priorities, and provide feedback about the development thus far, allowing evaluation and approval of the solution. With the prototype being done parallel with the development, there was also constant communication and exchange of ideas and feedback between these parties. The platform was developed following the prototype, and prioritizing physicians' feedback when changes relative to the prototype were necessary due to technological and time constraints.

In the final stage of the project, a final evaluation was performed, as described in Section 3.4.

3.2 Requirements

The requirements for the platform were initially defined after discussion with the physicians involved in the project. They were subject to multiple iterations of changes throughout the project's development, as a result of an ongoing collaboration with the physicians, as described. The final requirements are listed in the following subsections.

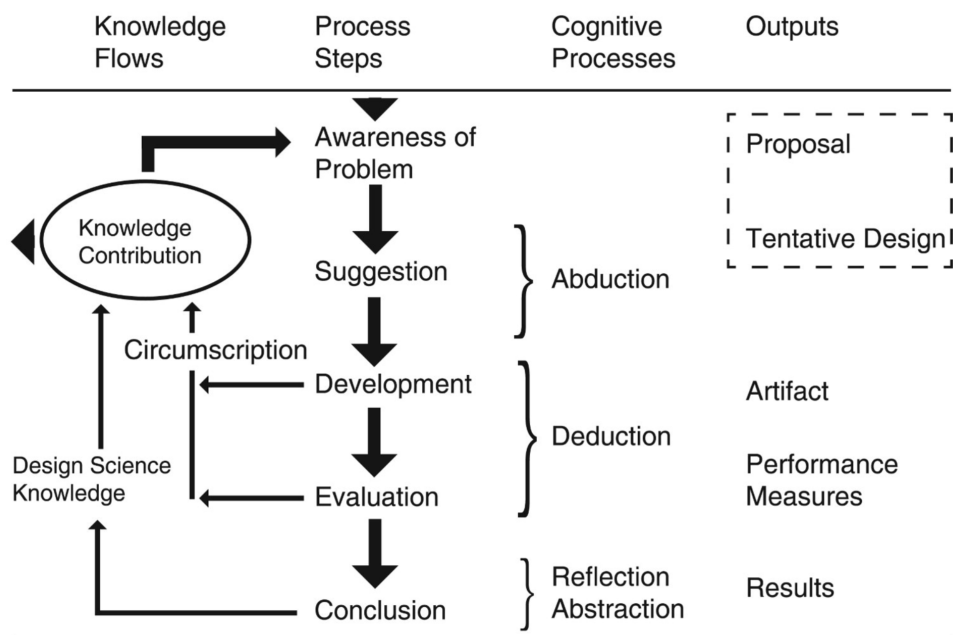


Figure 3.2: Processes in the design cycle.[14, 13]

3.2.1 Functional Requirements

The functional requirements, that specify what the system must do, are listed below. The items in this list represent what the users of the system should be able to do.

- View patients lists with main patient information, with patients separated by type - pediatric, transition, and adult;
- Search patients by name;
- Switch patients' type;
- View patients' profiles, which should include personal and contact information, diagnosis information, complications, and provide access to therapies, exams, and appointments;
- Edit patient personal data, such as contact information and mutation;
- Update patients' complications;
- View and download exams;
- Add exams and upload the correspondent files;
- View appointments' notes;
- Add appointment notes;
- View patients' current non-pharmacological therapies;

- Update patients' current non-pharmacological therapies;
- Add patients' pharmacological therapies;
- View patients' current and historical pharmacological therapies;
- Cease patients' current pharmacological therapies;
- Add a new patient;

3.2.2 Non-functional Requirements

The non-functional requirements, that describe more general properties related to how the system should perform, are listed below.

- System should be designed primarily for desktops, but should be responsive and usable in smaller screens, such as those of phones;
- Storage space should be large and/or expandable, due to increasing number of patients with a large number of exam files for each;
- System should provide easy and intuitive navigation;

3.3 Technologies

After an initial definition and a better understanding of the project's requirements, the most suitable technologies were picked. This section discusses the technologies used to develop the platform and the reasoning behind these choices.

For the several technologies, the following criteria were analyzed: adequacy to the project, cost, and ease of use for the developers (which includes aspects such as the learning curve and how widespread the technology is, since the larger a community around a technology is, the easier it is to find information and help). The priority was picking technologies that were appropriate to develop the proposed system and allowed for better performance, while minimizing costs and development time. It was also taken into account that the project will be subject to further developments and future changes.

Firstly, the choice for the database was MongoDB which is a NoSQL, document-oriented database program.[8] NoSQL uses dynamic schemas for unstructured data, allowing much more flexibility than an SQL database.[19] Since there were going to be several iterations of the design cycle throughout the project's development, during which the requirements would often change, it was essential to have a highly flexible database that allowed to add and remove fields and change relations between elements easily. MongoDB also provides faster processing of large volumes of data compared to SQL databases [19], which is significant considering the growing amount of patients, each with growing amounts of associated information. Complex queries can be more challenging to execute in MongoDB [19], but since most queries in the system will be simple

(such as access by ID, retrieving all documents in a collection, or simple filtering queries), this is not enough of a drawback to justify switching to SQL. Furthermore, MongoDB provides a free tier ideal for the development stage of the project, with other low-priced serverless or dedicated tiers to be used in later stages.[8]

Storing the exam files in MongoDB, however, posed some issues. Due to the growing and possibly large number of patients and exam files per patient, storing all of these files directly in the MongoDB database could create performance problems when scaling the system. This storage would require high memory usage in the database, making queries slower. It would also significantly increase the size of the database, making it more difficult to maintain and increasing costs. Therefore, the decision was to store the exam files in a cloud storage, namely Azure Blob Storage. Azure Blob Storage is a secure and scalable cloud storage, ideal for serving documents to a browser and archiving data.[7] Like MongoDB, it offers cheap, pay-as-you-use pricing plans, allowing for cost-efficient storage, and it is included as part of the "Office 365 A3 for students" subscription offered by the University of Porto (UP), making it an adequate choice for the development of this project.[1]

Secondly, Express.js, a web application framework for Node.js [5], was chosen to develop the backend. Node.js code is written in Javascript [5], a language that is easy to grasp and makes development more seamless when switching between the frontend (which is also developed in Javascript, as explained below) and backend code. It also provides fast performance, a large developer community and support, and portability.[5] Express is a widely used Node.js web framework, which provides support for features that Node.js does not have built-in support for, such as handling different HTTP methods and routing. Both Express.js and Node.js are also open-source and free to use under the MIT license.[5]

Finally, the frontend was developed using React.js, a frontend Javascript library [11], along with several npm packages. Besides being easy to learn and use, React has efficient performance, since it minimizes DOM changes through the use of a Virtual DOM. It does this by monitoring the value of components' states with the Virtual DOM, and when changes occur, it compares the existing DOM state with the new one and updates it in the least costly way.[20] Its DOM implementation and rendering optimizations make it faster than other platforms, such as Angular. Other advantages of React.js include its broad community of users, the large number of existent npm packages for React that significantly reduce development time and improve product quality, and the fact that it is free and open-source.[20]

Three of the main npm packages used were React Bootstrap, React Router DOM, and Axios. React Bootstrap is an adaptation of Bootstrap for React, where each component was rewritten as a React component.[9] Bootstrap is a CSS framework for frontend development, that contains templates for several interface components such as buttons, cards, typography, and layout. Its templates reduce development time and allow for straightforward development of responsive interfaces.[9] React Router DOM provides tools for client-side routing with React in web applications, without the difficulties of manually setting the routes.[10] Axios is a promise-based HTTP client library that offers the possibility of making requests to a Node.js endpoint in a very straight-

forward way: the function names match the HTTP methods, it works well with JSON data, and it has default error handling.[6] Other packages used were React Hot Toast for toast notifications, React Dropzone for the exam file upload interface, and Font Awesome for icons in the application.

3.4 Usability Tests

In the final stage of the project, the prototype was tested. This section explains how these tests were performed.

The evaluation of the platform was done through usability testing with semi-structured interviews, where participants interacted with the application prototype. Since the prototype encompassed the features implemented in the web application and additional ones, the decision was to perform the testing with the prototype to allow for more extensive feedback. The participants were asked to perform specific tasks on the system and then encouraged to explore the prototype independently. Qualitative feedback was given throughout the tests, after which the participants were asked to fill in a form rating their experience with the prototype.

After the first question, where participants are asked their age, the form contains three sections. The first one is based on the system usability scale (SUS) developed by John Brooke, a ten-item Likert scale used to "[give] a global view of subjective assessments of usability".[12] A five-level Likert scale is used, where 1 is "Strongly disagree" and 5 is "Strongly agree". SUS scores range from 0 to 100, with higher scores indicating better usability. In the second section, participants are asked to rate the difficulty of the performed tasks. Finally, in the third section, they are given open-ended questions where they can comment on other aspects of the prototype and give suggestions.

Chapter 4

Implementation

This chapter describes the features effectively implemented in the platform and the project structure, outlining the different components of the system and how they interact with each other.

4.1 Platform Features

After the initial definition of requirements and picking the technologies, the platform started to be developed. During the implementation process, it was factored in which requirements had the highest priority, to ensure that a solid proof of concept was developed. This section outlines the implemented features.

The web application implementation followed the created prototype for the interface design. In the cases where it was not possible to follow the prototype, such as due to features not being implemented at this stage of the project, the overall style, layout, and color scheme were fully respected, maintaining a consistent look across the application. Throughout this process, usability and a satisfying user experience were kept a top priority. Since the platform is to be used firstly by physicians at CMIN, with the possibility of expanding to other Portuguese hospitals, the interface is in Portuguese.

The interface was implemented using a desktop-first approach since this is physicians' primary medium to access EHR applications. However, to increase the application's flexibility and make it usable on smaller screens, split-screen views, and mobile devices, most pages were made responsive to screen size, as can be seen in Figure 4.1.

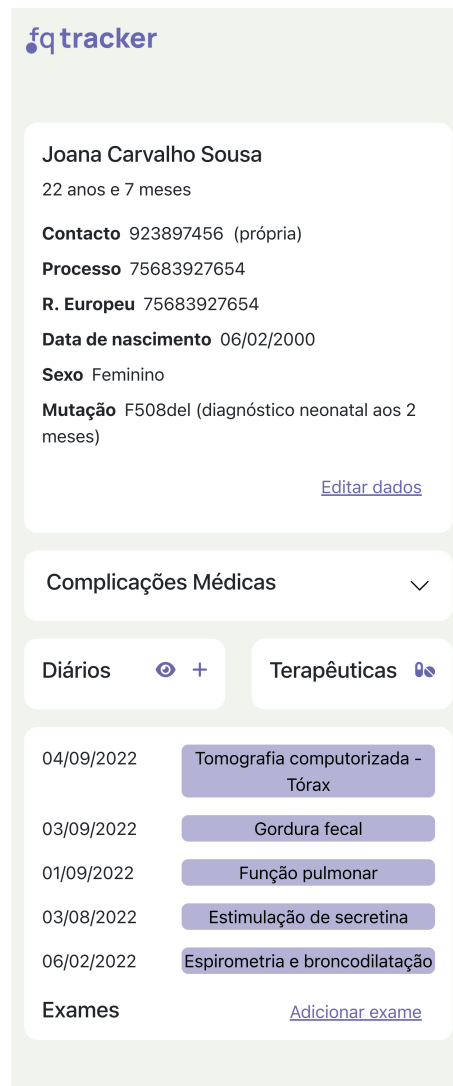


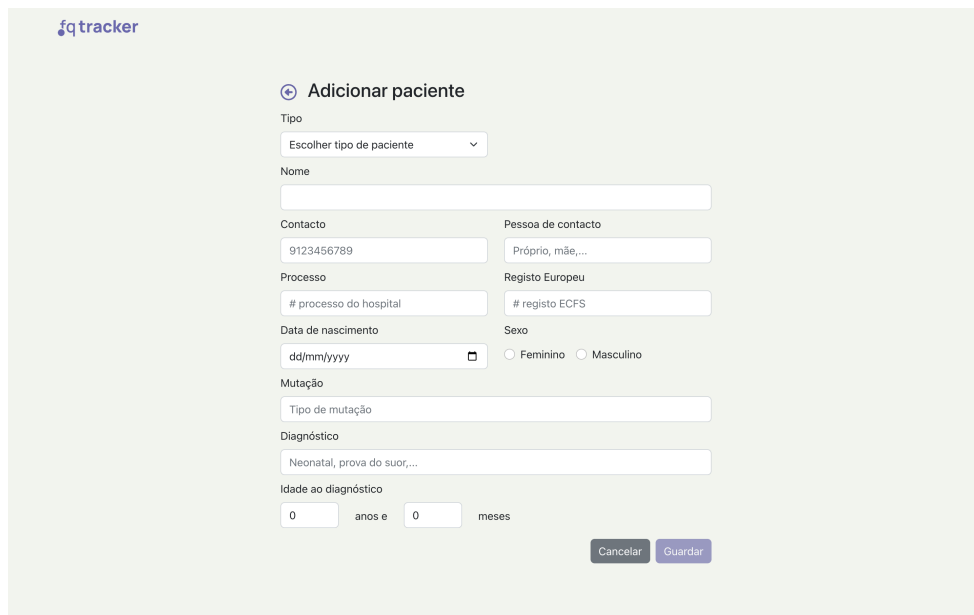
Figure 4.1: Patient's profile viewed on a mobile phone screen.

To improve the user experience, almost all user actions that involve changes to the data, such as adding new data or editing existing data, are followed by feedback in the form of a toast notification, as seen in the bottom of Figure 4.2. This toast contains information on whether or not the operation was successful and the result of the action. This feedback keeps users informed about their actions and helps them decide their next steps, creating a more reliable experience.



Figure 4.2: Homepage, with a feedback toast indicating the successful addition of a new patient.

When the user first opens the web application, they are presented with the homepage, like presented in Figure 4.2, but in this case without the feedback toast notification. On this homepage, users can select if they want to view the list of patients, which can be separated by patient type or include all patients. The patients can be pediatric (patients under the age of 16), transition patients (patients over the age of 16 but that have not yet completed necessary exams to transition into the adult category), and adults. They also have the possibility to add a new patient or a new professional (physician) using the two purple buttons in the bottom row. To add a patient, the user fills out the patient's type, identifying information (such as name, process number, birthdate), contact information, and CF mutation and diagnosis information in the form seen in Figure 4.3. To add a professional, the user fills out the professional's type, which can be doctor or nurse, name, professional license number, and specialty in the form seen in Figure 4.4. After adding either a patient or a professional, the user is redirected back to the homepage.

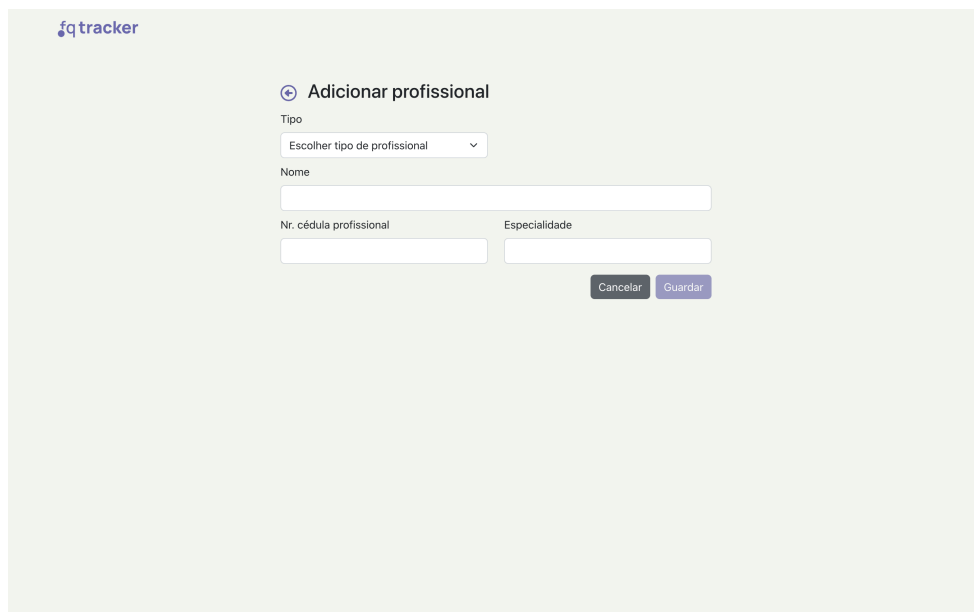


The screenshot shows the 'Adicionar paciente' form in the fqtracker application. The form is titled 'Adicionar paciente' and includes the following fields and options:

- Tipo:** A dropdown menu with the text 'Escolher tipo de paciente'.
- Nome:** A text input field.
- Contacto:** A text input field containing '9123456789'.
- Pessoa de contacto:** A text input field with the placeholder 'Próprio, mãe,...'.
- Processo:** A text input field with the placeholder '# processo do hospital'.
- Registo Europeu:** A text input field with the placeholder '# registo ECFS'.
- Data de nascimento:** A text input field with the placeholder 'dd/mm/yyyy' and a calendar icon.
- Sexo:** Radio buttons for 'Feminino' and 'Masculino'.
- Mutação:** A text input field with the placeholder 'Tipo de mutação'.
- Diagnóstico:** A text input field with the placeholder 'Neonatal, prova do suor,...'.
- Idade ao diagnóstico:** Two text input fields for 'anos' and 'meses', both containing '0'.

At the bottom right of the form are two buttons: 'Cancelar' and 'Guardar'.

Figure 4.3: Form to add a patient.



The screenshot shows the 'Adicionar profissional' form in the fqtracker application. The form is titled 'Adicionar profissional' and includes the following fields and options:

- Tipo:** A dropdown menu with the text 'Escolher tipo de profissional'.
- Nome:** A text input field.
- Nr. cédula profissional:** A text input field.
- Especialidade:** A text input field.

At the bottom right of the form are two buttons: 'Cancelar' and 'Guardar'.

Figure 4.4: Form to add a professional.

The patient listing table, which can be viewed in Figure 4.5, shows basic patient information, such as age and mutation, allowing the user to visit the patient's profile by clicking the "Ver" ("View") button. There is also a dropdown in each row indicating the patient type that, when clicked, can be used to change the patient's type easily. This change is applied immediately, moving the patient to the new corresponding table and removing it from the current one. To avoid forcing the user to navigate to another page to undo their changes in case of a mistake, the

toast displayed on success has the option to "Anular" ("Undo") this change, as seen in Figure 4.5, undoing the action instantly. The user can also search for a patient by their name, and the results on the table are updated as the user types.

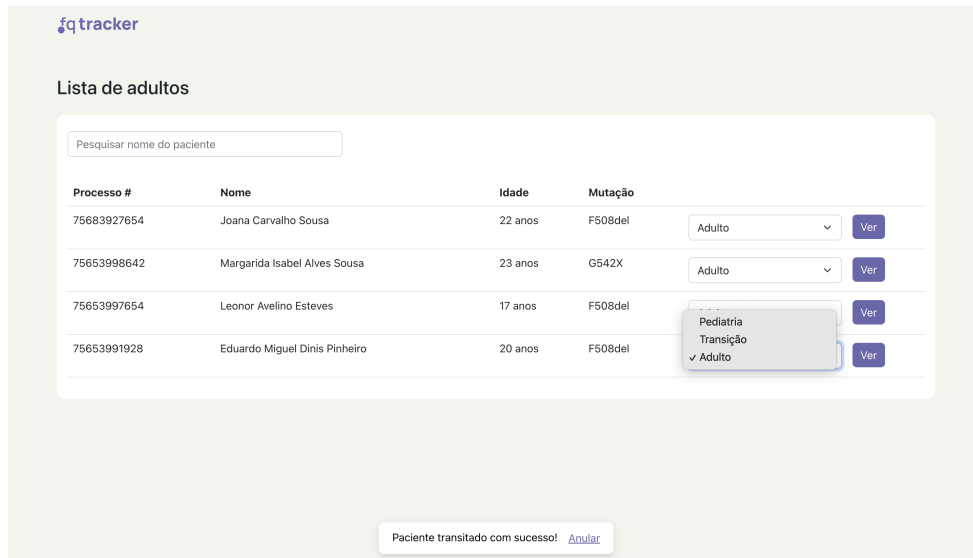


Figure 4.5: List of adult patients, with one open patient type dropdown and a toast notification indicating the successful transition of a patient.

On the patient profile, visible in Figure 4.6, the information related to the patient is categorized in different cards, making for a more intuitive viewing. The top left card contains the patient's personal information, including name, age, birthdate, gender, hospital process number, European Cystic Fibrosis Society (ECFS) registry number, phone contact and contact person (since a lot of the patients are minors, the phone contact can belong to a parent or guardian). It also contains the CF mutation type along with age at diagnosis and diagnosis type. The patient information on this card can be edited by clicking "Editar dados" ("Edit data"), which turns the card into a form with the corresponding fields, as seen in Figure 4.7. The medical complications card can be expanded, as seen in Figure 4.8. It then shows a list of possible complications, each one of them next to a toggle that represents whether or not the patient has that complication (green "Sim" toggle or red "Nao" toggle, respectively). The complications can be updated by clicking on their respective toggles, and these changes are applied immediately without the need to save.

The screenshot shows the patient profile page for Joana Carvalho Sousa. The patient's name is displayed at the top left. Below the name, the age is listed as 22 years and 7 months. The contact information includes a contact number (923897456) and a contact person (própria). The process number is 75683927654, and the patient is identified as R. Europeu. The date of birth is 06/02/2000, and the sex is Feminino. The mutation is F508del, with a note that it was diagnosed neonatally at 2 months. There is an 'Editar dados' link. On the right side, there are tabs for 'Diários' and 'Terapêuticas'. The 'Diários' tab is active, showing a list of dates from 04/09/2022 to 15/01/2022, each with a corresponding exam name: Tomografia computadorizada - Tórax, Gordura fecal, Função pulmonar, Estimulação de secretina, Espirometria e broncodilatação, and Radiolinia Tórax. There is an 'Adicionar exame' link at the bottom right. Below the main profile information, there is a section for 'Complicações Médicas' with a dropdown arrow.

Figure 4.6: Patient profile page.

The screenshot shows the form to edit patient data for Joana Carvalho Sousa. The form fields are as follows: Name (Joana Carvalho Sousa), Contacto (923897456), Pessoa de contacto (própria), Processo (75683927654), R. Europeu (75683927654), Data de nascimento (06/02/2000), Sexo (Feminino), Mutação (F508del), Diagnóstico (neonatal), and Idade ao diagnóstico (0 anos e 2 meses). There are 'Cancelar' and 'Guardar' buttons at the bottom right of the form. On the right side, there are tabs for 'Diários' and 'Terapêuticas'. The 'Diários' tab is active, showing a list of dates from 04/09/2022 to 15/01/2022, each with a corresponding exam name: Tomografia computadorizada - Tórax, Gordura fecal, Função pulmonar, Estimulação de secretina, Espirometria e broncodilatação, and Radiolinia Tórax. There is an 'Adicionar exame' link at the bottom right. Below the main form, there is a section for 'Complicações Médicas' with a dropdown arrow.

Figure 4.7: Form to edit patient data on patient profile page.

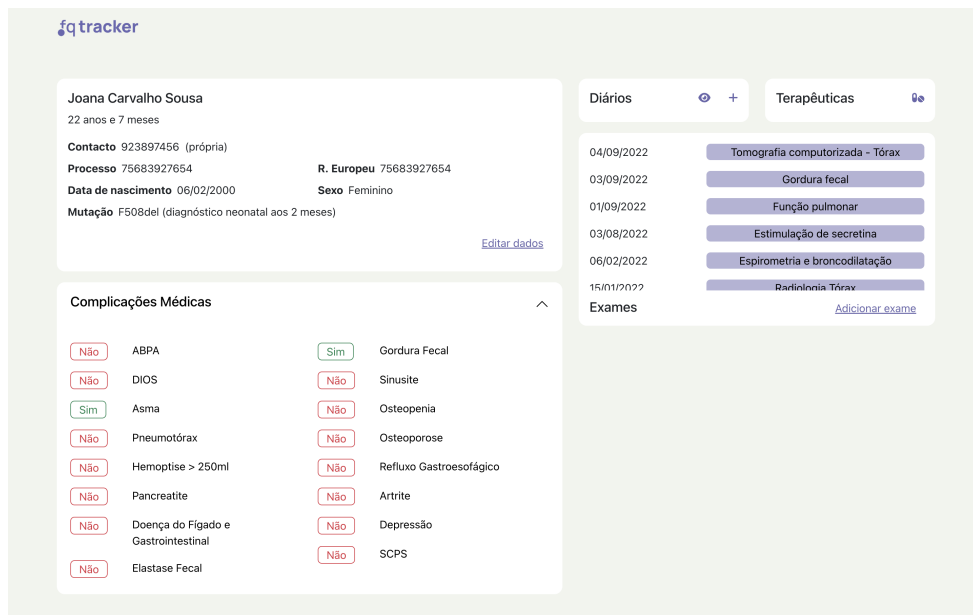


Figure 4.8: Patient profile page with complications card expanded.

The exams card shows a list of all the patient's exams that have been submitted to the platform. It shows the date and name of each one, and the corresponding PDF files can be downloaded by clicking on the name of the exam. When clicking on "Adicionar exame" ("Add exam"), the user opens a modal in which they can add an exam to the current patient, by inputting the name of the exam, date, and drag and dropping the PDF file or selecting it from the file explorer, as seen in Figure 4.9.

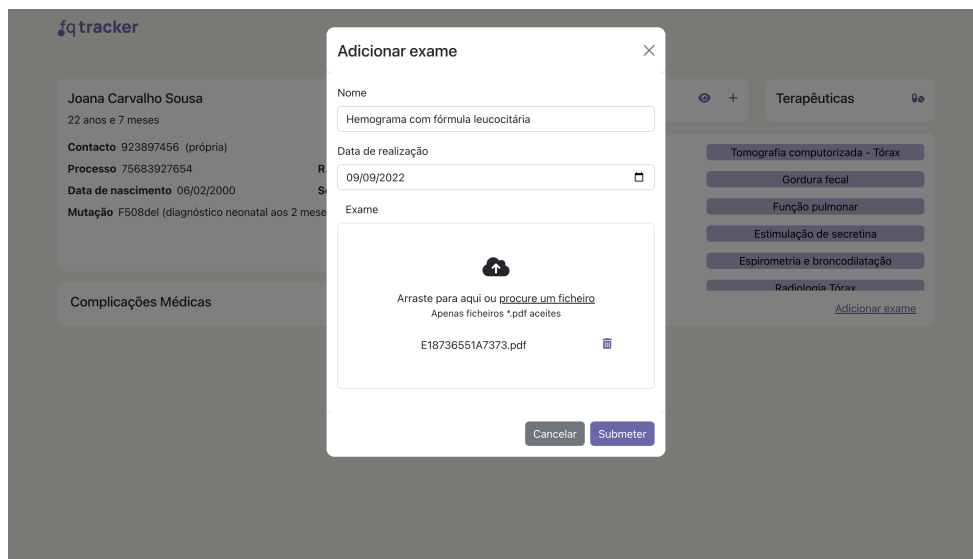


Figure 4.9: Modal to add exam on patient profile page.

Notes from a patient's previous appointments can be viewed by clicking on the eye icon on

the "Diários" card. A list of the patient's appointments is presented, along with the notes of the most recent appointment shown by default, which can be seen on Figure 4.10. To view notes from other appointments, the user should click on the corresponding date and time on the list. New appointments can be added by clicking on "Adicionar entrada" ("Add entry") or by clicking on the plus sign icon on the "Diários" card on the patient's profile. On the page to add an appointment, visible on Figure 4.11, the user selects the physician that is associated with it and writes the appointment notes. The date and time of the appointment do not need to be selected, as they default to the current date and time.

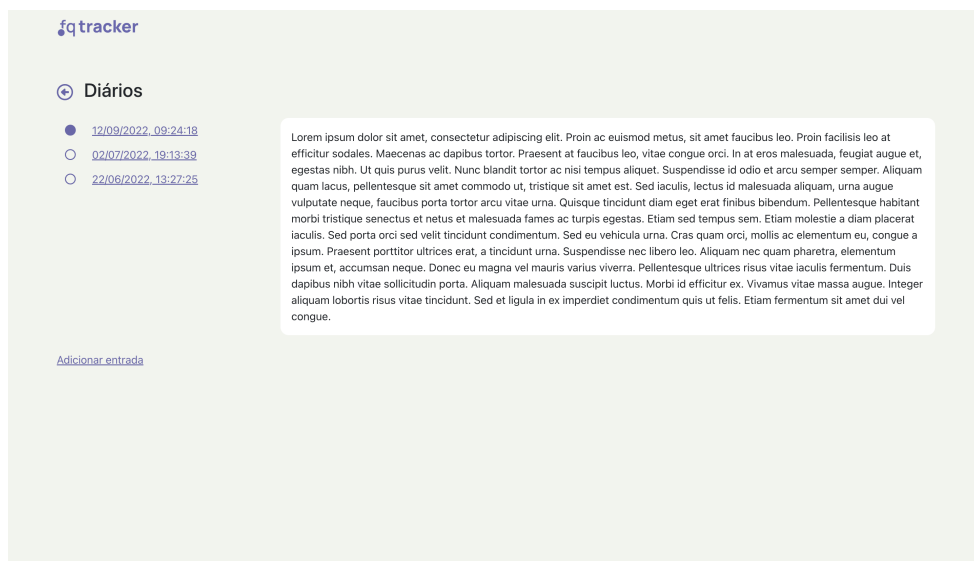


Figure 4.10: Patient's appointments list.

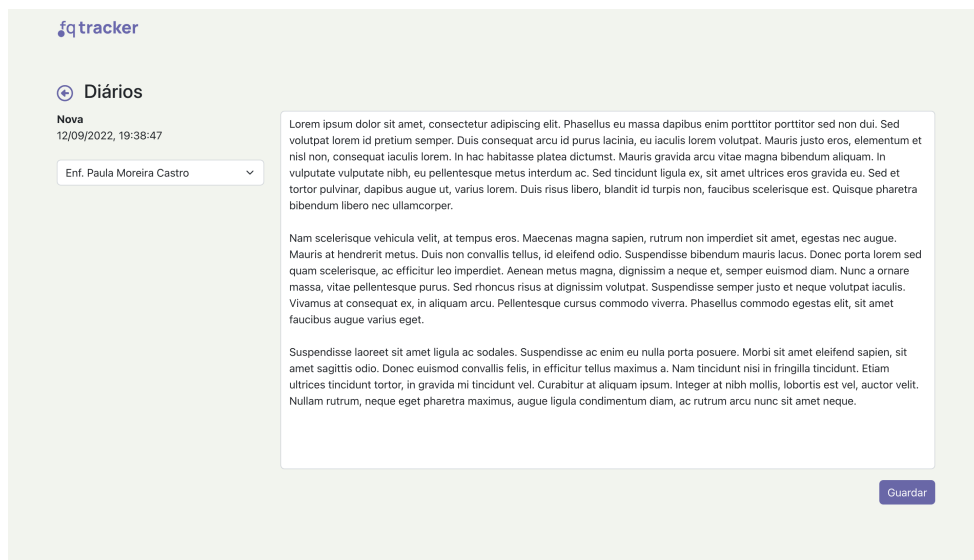


Figure 4.11: Form to add a new appointment.

Patients are subject to both non-pharmacological and pharmacological therapies to manage the illness, which can be viewed on the "Therapies" page by clicking on the corresponding card on the patient profile. On this page, seen in Figure 4.12, therapies are divided between non-pharmacological, on a collapsible card at the top, and pharmacological (medication), on a table at the bottom. The non-pharmacological therapies interface works in the same fashion as the patient's complications: the name of the therapy is preceded by the corresponding toggle that indicates whether or not the patient is currently doing that therapy and can be clicked to change that value, applying the changes immediately. The pharmacological therapies table shows current and historical information about the patient's medications. It shows the date when the patient began that medication, the medication name, administration route (such as oral, inhalation, or intravenous), frequency, dosage, and end date. The end date is an optional field in order to accommodate medications that are taken long-term or do not have a predetermined date for the end of treatment. If the end date of a medication has not yet been reached or if it does not have an end date, the user can cease the medication by clicking on the "X" button under the "Cessar" ("Cease") column. This action updates the end date of the medication to be the same as the current date.

Terapêutica não-farmacológica

Sim Ventilação não invasiva
 Não Dispositivos com PEEP

Não Oxigenoterapia alto fluxo

Não Cinesiterapia respiratória

Terapêutica farmacológica

Início	Medicamento	Via de administração	Frequência	Dose	Fim	Cessar
15/05/2022	Kafrio	Oral	12/12h	75mg/50mg/100mg		<input type="button" value="X"/>
31/08/2022	Tobramicina em pó	Inalatória	12/12h	4*28mg	28/09/2022	<input type="button" value="X"/>

[Adicionar medicação](#)

Figure 4.12: Patient's therapies page.

To add a new medication, the user should press "Adicionar medicação" ("Add medication") at the bottom of the medication table, which will open the modal visible in Figure 4.13. In this modal, the user inputs the medication name, administration route, frequency, dosage, start date (which defaults to the current date but can be changed), and the end date if applicable.

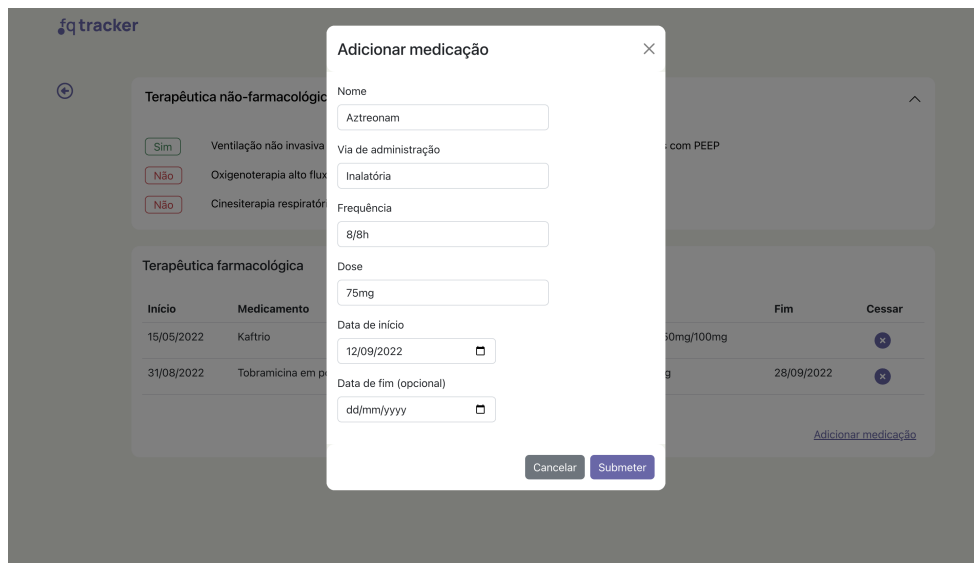


Figure 4.13: Modal to add new medication.

4.2 Project Structure

This section describes the system's architecture and the project file structure. It goes over the different elements that make up the system and their interactions.

The project files are separated into two directories. The frontend directory contains the different views, client-side routing, and Axios requests, and the backend directory contains the database models and the Express application programming interface (API).

4.2.1 Backend

The file `server.js`, located at the root of the backend directory, imports the necessary libraries, connects to Mongoose, defines the routers that should be used for the API requests depending on the uniform resource locator (URL), and designates the port on which the server should run.

The backend contains two subdirectories: `models` and `routes`. The `models` directory has files that define the Mongoose models used to interact with the documents in the MongoDB database. There are three files defining three main schemas: `professional`, `appointment`, and `patient`.

The `professional` collection, used to store the physicians' data, has a `type`, which can be `doctor` or `nurse`, `name`, `professional license number`, and `specialty`, with all the fields being mandatory. The `appointment` collection, used for data related to appointments, stores the IDs of the patient and the professional involved in the appointment as a reference, the `date`, which defaults to the current date, and `notes`, with all fields except the notes being mandatory.

The `patient` collection stores data related to the patients, with the mandatory fields being the `patient type` (`pediatric`, `transition`, or `adult`), `name`, `hospital process number`, `ECFS registry number`, `gender`, `birth date`, `contact`, `contact person`, `age at diagnosis`, `diagnosis type`, and `CF mutation`.

The patient also has arrays of therapies (referring to the patient's current non-pharmacological therapies names), medications (including current and historical medications), complications (patient's current complications names), and secretion results. The medications, exams, and secretion results each have their own subschema since, unlike the other fields that were represented by a simple string or date, these have various parameters that define them. The exam schema stores the exam name, date (defaulting to the current date), and file name, which is the name of the Azure blob containing the exam PDF file. The medication schema contains the name, administration, frequency, dosage, start date, and end date. The secretion results relate to a requirement that was initially discussed but later dropped in favor of other higher priority requirements. CF patients have to do secretions analyses regularly to monitor their illness, and this schema, which has the date of secretion delivery, date of results, and the results as a string, would be used to store the results of those analyses. Although this feature was not implemented, the schema was left in the database to facilitate future implementation.

The other directory in the backend, routes, defines routes for handling HTTP requests using Express Router middleware. These routes either respond to the request or handle the error, using the default Express error handling by passing the error to the built-in next() function. The routes are organized into three different files, each with their router, in accordance with the database model files: professional, appointment, and patient.

The routes relating to professionals are presented ahead, with the corresponding HTTP verb and URL path:

- `GET professionals/` - returns all existing professionals.
- `GET professionals/:id` - returns the professional with the specified ID. It also returns the appointments associated with the professional, but to avoid sending too much unnecessary information in the response, it only sends the IDs and dates of the appointments. The notes of the appointments can then be retrieved using the appointments IDs.
- `POST professionals/` - creates a new professional, with the type, name, license number, and specialty specified.

The routes relating to appointments are listed below:

- `GET appointments/:id` - returns the appointment with the given ID.
- `POST appointments/` - creates a new appointment, with the specified professional (ID), patient (ID), date, and notes.

Besides the routes, the patients file also deals with the exam file upload to Azure Blob Storage. The frontend sends the file on the request with the encoding method of "multipart/form-data", which supports file upload. Usually, the Multer middleware is used to handle this type of encoding type with Node.js and Express.js; however, since Multer uploads the files to a directory in the server, and the goal is to store the files on Azure, it was adequate for this project. Therefore, the

npm package Multer Azure Blob Storage, which is a Multer engine for storage in Azure, was used. To download exam files, the connection with Azure is made by creating a BlobServiceClient object from the Azure Storage connection string. The container that stores the exam files is obtained from the BlobServiceClient, and the containerClient is created. To store the files in Azure, a Multer storage is created using the Azure container data, such as connection string, access key, and container name. The routes defined in the patient file are as follows:

- *GET* `patients/` - returns all patients, with the option to make a query by the patient name, returning only patients whose name contains the searched name.
- *GET* `patients/pediatric` - returns all pediatric patients, with the option to make a query by the patient name.
- *GET* `patients/transition` - returns all transition patients, with the option to make a query by the patient name.
- *GET* `patients/adults` - returns all adult patients, with the option to make a query by the patient name.
- *GET* `patients/:id` - returns the patient with the specified ID. It also returns the appointments associated with the patient, but to avoid sending too much unnecessary information in the response, it only sends the IDs and dates of the appointments.
- *PUT* `patients/:id/edit` - updates the patient with the given ID so that its data is the same as the patient data sent in the request body.
- *POST* `patients/` - creates a new patient, with the specified type, hospital process number, ECFS registry number, name, contact, contact person, birth date, gender, age at diagnosis, type of diagnosis, and mutation.
- *GET* `patients/complications` - returns a pre-defined array with all possible medical complications (not just the ones the patient currently has). This array is defined at the beginning of the file.
- *GET* `patients/therapies` - returns a pre-defined array with all the possible non-pharmacological therapies (not just the ones the patient is currently doing). This array is defined at the beginning of the file.
- *GET* `patients/:id/complications` - returns the complications that the patient with the specified ID currently has.
- *PUT* `patients/:id/complications` - receives an array of complications and updates the complications of the patient with the specified ID to be the same as the given array.
- *GET* `patients/:id/therapies` - returns the therapies that the patient with the specified ID is currently doing.

- `PUT patients/:id/therapies` - receives an array of non-pharmacological therapies and updates the non-pharmacological therapies of the patient with the specified ID to be the same as the given array.
- `GET patients/:id/medication` - returns the current and historical medications of the patient with the specified ID.
- `PUT patients/:id/medication` - updates the medications of the patient with the specified ID by adding a given medication.
- `PUT patients/:id/medication/:med_id/cease` - ceases the medication with the given med ID belonging to the patient with the given ID, by updating its end date to be the current date.
- `PUT patients/:id/exam` - adds an exam to the patient, with the given file, exam name, and date. The file is uploaded using the Multer Storage as explained above. After uploading the file to Azure, Multer makes file data available, such as the name of the blob it was uploaded to. This blob name is stored into the database as the file name.
- `GET patients/:id/exam/:filename` - downloads the exam with that is stored in the blob with the given name (filename) belonging to the patient with the specified ID. It obtains the blob from the containerClient created initially, the downloads it.
- `GET patients/:id/age` - calculates and returns the age of the patient with the given ID. The date is divided into two fields: months and years.

4.2.2 Frontend

The frontend directory has the package.json file at its root, which defines the project's metadata and npm dependencies, a public directory, which includes the web application's favicon and the index.html, and the src directory, with the project's code.

At its root, the src directory has several files necessary for the overall function of the application. The App.js file includes the header component and the toaster component from React Hot Toast (needed to use toast notifications in the application) and will be a part of all the application screens. The index.js file defines the application's routes with React Router, defining which views should be rendered for different URL paths. The App.css defines custom styling for the different components, and there is also an images directory containing the application's logo.

The rest of the frontend files are organized into three directories according to their function: services, views, and components.

The services directory holds the Axios functions that send the HTTP requests to the API endpoints. The functions are organized in the same way as the Express routes, divided between appointments, professionals, and patients, and each function is responsible for one HTTP request. The api.service.js file defines the base URL for the HTTP requests, which was localhost on the port 5000 throughout this phase of development but may be changed in the future.

The views directory contains the different screens of the application. These are the React components rendered by the routes, and each of them is a Bootstrap container that includes the components that make up the screen organized in a grid layout. The views, or screens, are the following:

- *ProfessionalDashboard* - The homepage, with the buttons to access different lists of patients and to add a patient or physician.
- *NewPatient* - Contains the new patient form. The "submit" button is disabled until the data entered in the form is valid, and after successful submission the user is redirected to the homepage.
- *NewProfessional* - Contains the new professional form, which works in the same way as the new patient form.
- *PatientsList* - Screen that shows a table with the patients, with a search field for the name. The patients table is a separate component included here. The type of patients ("pediatric", "transition", "adults", or "all") is passed as a prop to this component, and the request to get patients from the API is made accordingly.
- *PatientProfile* - Shows the patient's profile, with the ID of the patient passed as a URL parameter. The identification, complications, and exams cards are their own components. The patient profile has a state, "editPatientMode", that is true when the user has pressed the button to edit the patient and false by default; when true, instead of rendering the regular patient identification card, the edit patient form will be rendered.
- *PatientAppointments* - Shows the patient's appointments, with the ID of the patient passed as a URL parameter. The active appointment, that is the appointment whose notes are being displayed, is managed by a state.
- *NewAppointment* - Contains the new appointment form. The ID of the patient that the appointment will be created for is passed as a URL parameter. The form works in a similar manner to the new patient and professional forms: the "submit" button is only disabled until all the data is valid and successful submission redirects to appointments page.
- *PatientTherapies* - Screen with the patients' non-pharmacological and pharmacological therapies. These two types of therapies are shown in different components, and the patient ID, passed as a parameter to *PatientTherapies*, is passed as a prop to these components.

The components directory includes React components that are used in the views. The decision to separate these components from the views aims to improve code readability, minimize code repetition, and facilitate development. The components are listed below:

- *Header* - Application header, with the logo that, when clicked, redirects the user to the homepage.

- *PatientsListTable* - Table with the patients, that receives the patients as a prop. When a patient is updated due to the user transitioning them, the list of patients is immediately updated using a callback function.
- *EditPatientCard* - Card with the form to edit patient information, with a similar layout to the regular patient identification card. Like the other forms, the submit button is only active when the data entered is valid.
- *PatientComplicationsCard* - Accordion component, with styling similar to the cards for a uniform look, that contains the patient's complications and the toggles to edit them. The patient's complications are managed by a state, updated every time a toggle is clicked. When the complications state changes, a useEffect hook sends the Axios request to update the patient's complications on the database.
- *PatientExamsCard* - Card that lists the patient's exams, which are received as a prop. When the button containing the exam name is clicked, a new tab is opened to download the exam file. The add exam modal can be opened by pressing the "Add exam" button.
- *AddExamModal* - Modal with the form to add an exam to the patient. A state on the patient profile controls the visibility of the modal. The submit button is active when the name and date have been entered and a file has been added to the Dropzone, either through dragging and dropping it or the system file explorer.
- *PatientsTherapiesCard* - Accordion component open by default, styled to achieve the same look as the cards, open by default, that contains the patients' non-pharmacological therapies and the toggles to edit them, in the same way as the complications.
- *PatientMedicationsCard* - Card that contains a table with the patient's medications. When the "cease" button is clicked, which is disabled unless the end date is non-existent or ahead of the current date, the Axios request to cease the medication is sent.
- *AddMedicationModal* - Modal with the form to add a new medication to the patient. A state on the medications card controls the visibility of the modal. The submit button is active when all the data, except the optional end date, has been entered.

4.3 Addition of Features

The developed platform can be improved in future iterations through the addition of new features, and this section explains how to do so.

Adding new models to the database or changing the fields of existing models is straightforward. New models can be added by adding new schemas, exported as Mongoose models, in new files, to the models directory. Adding new fields to existing models, or removing or modifying existing fields should be done directly in the model's file. It is important to remember that when

the models are updated, the API routes that interact with them and the respective Axios requests should be updated as well. New routes should be added to the file corresponding to the model that it relates to. When adding a new router, the `app.use()` function should define the base path to which the new router middleware will be mounted, in the `server.js` file.

A new React component with a Bootstrap container is added to the views directory to add a new screen to the interface. Separate components can be created to, for instance, break down long blocks of code into smaller, more readable components, define components used multiple times, or isolate blocks of code with complex functionality from the rest of the code. The route that defines the path for the new view component should then be added to the `index.js` file.

Chapter 5

Results

In the final stage of the project's development, usability tests were used to evaluate the fulfillment of the project's objectives, as described in Chapter 3. The tests were performed and their results and feedback was then analyzed, as presented in this chapter.

There were three test participants, ages 46, 33, and 24. Two of them are physicians that work with CF patients, and one of them is a university student. Testing the prototype with someone outside of the health field allows to understand how it would be for a user without knowledge about medical terms and EHR systems to use the platform, further showing its usability and ease of use.

The results of the SUS-based section of the form are presented in Figures 5.1 to 5.10. The SUS score of this section was 100. Taking into account that a result above 68 is considered above average, as seen in Figure 5.11, this result is an extremely encouraging indicator.

The participants' evaluation of the difficulty of the tasks they were requested to perform is shown in Figure 5.12. A scale ranging from "Very difficult" (blue) to "Very easy" (purple) was used. As can be seen, the tasks were mostly rated "Easy" or "Very easy", with two tasks being evaluated by one participant as "Neutral" in difficulty. The performed tasks were the following (listed in the order they are presented in Figure 5.12):

- View a patient's profile.
- View a patient's medical complications.
- Add exam results.
- Add a patient's medication.
- Add appointment notes.
- Transition patient.

The last section of the form, with open-ended questions, received responses from one of the participants. The questions and responses were as listed below:

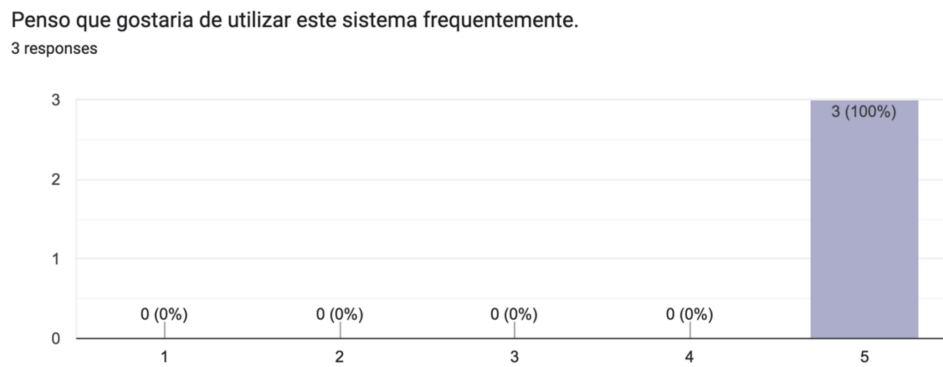


Figure 5.1: Results of the 1st SUS question ("I think that I would like to use this system frequently").

- "Which aspects of FQ Tracker do you find the most positive?" - "The perfect symbiosis between aesthetics and functionality."
- "Which aspects of FQ Tracker do you find negative." - "Nothing to comment."
- "Other suggestions/comments?" - "Possibly expanding the click area on small buttons and removing the double click on the patient category."

Some feedback was also given during the interviews as the participants interacted with the prototype. These comments were always positive and focused mainly on the aesthetic appearance of the prototype and the ease of use, with participants saying statements like "It is very intuitive" and "I have used a lot of health platforms and none of them were this easy [to use]".

Achei o sistema desnecessariamente complexo.

3 responses

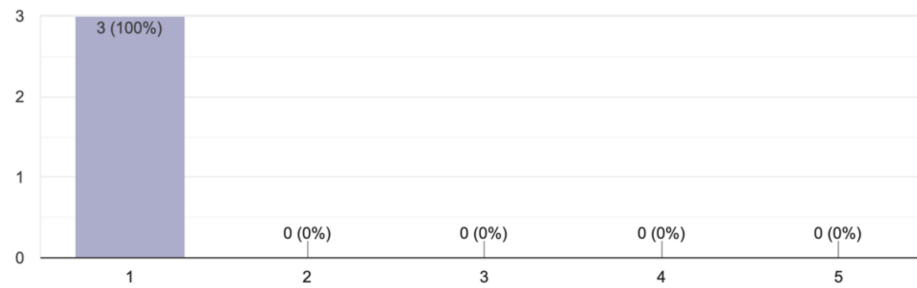


Figure 5.2: Results of the 2nd SUS question ("I found the system unnecessarily complex").

Achei o sistema fácil de usar.

3 responses

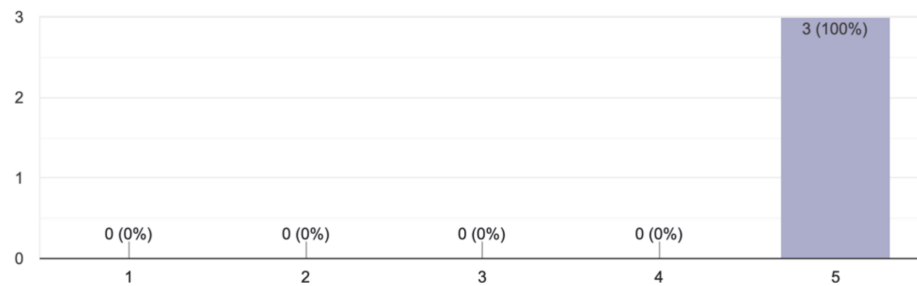


Figure 5.3: Results of the 3rd SUS question ("I thought the system was easy to use").

Penso que precisaria de ajuda de uma pessoa com conhecimentos técnicos para usar o sistema.

3 responses

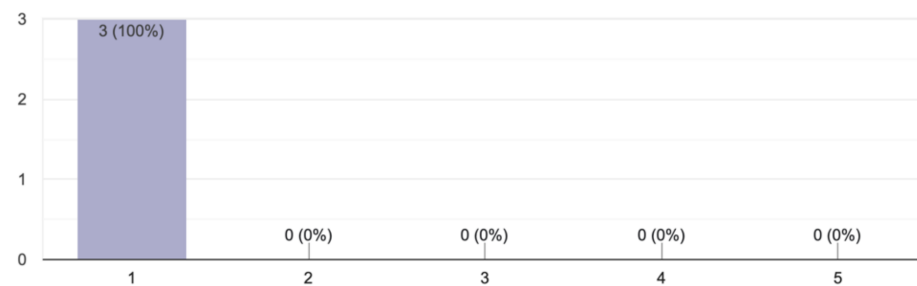


Figure 5.4: Results of the 4th SUS question ("I think that I would need the support of a technical person to be able to use this system").

Acho que as várias funções do sistema estão bem integradas.

3 responses

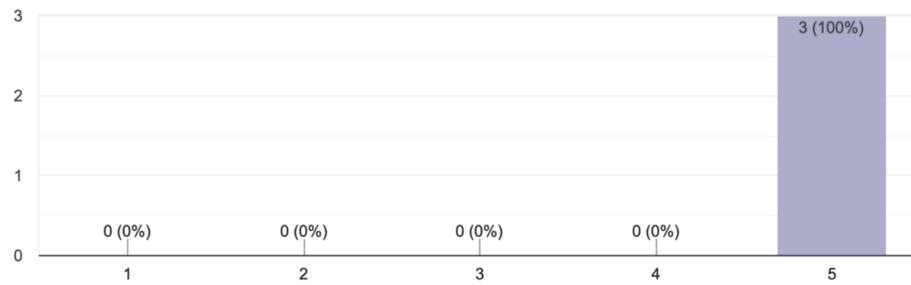


Figure 5.5: Results of the 5th SUS question ("I found the various functions in this system were well integrated").

Penso que o sistema apresenta muita inconsistência.

3 responses

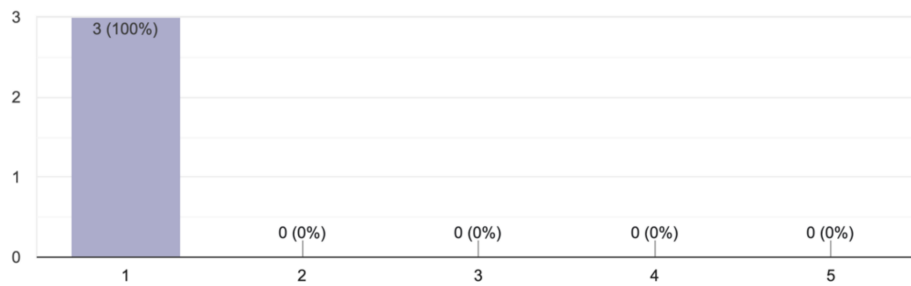


Figure 5.6: Results of the 6th SUS question ("I thought there was too much inconsistency in this system").

Penso que a maioria das pessoas aprenderão a usar este sistema rapidamente.

3 responses

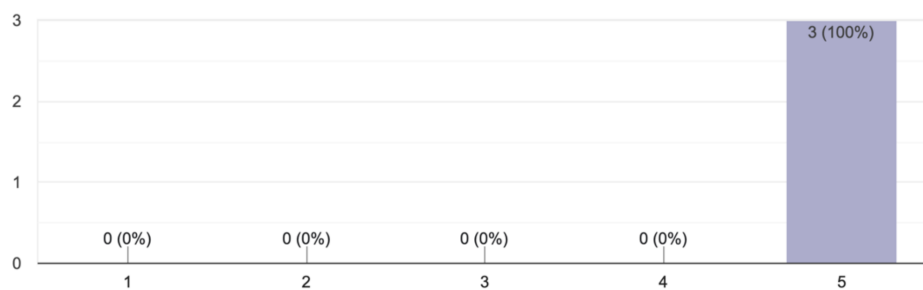


Figure 5.7: Results of the 7th SUS question ("I would imagine that most people would learn to use this system very quickly").

Achei a utilização do sistema incomoda.
3 responses

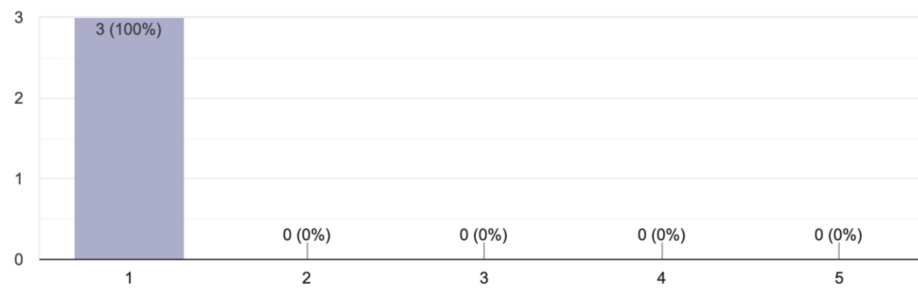


Figure 5.8: Results of the 8th SUS question ("I found the system very cumbersome to use").

Senti-me confiante ao navegar no sistema.
3 responses

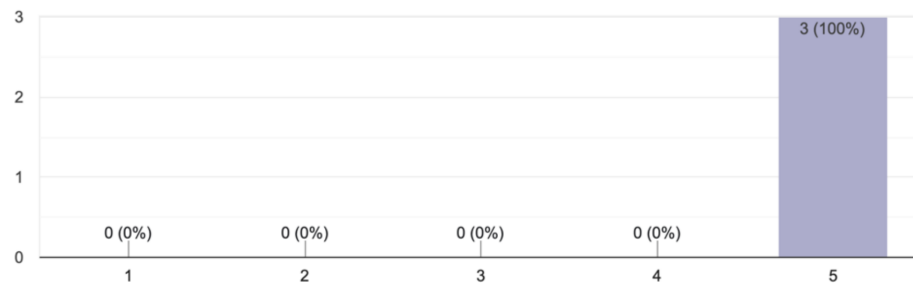


Figure 5.9: Results of the 9th SUS question ("I felt very confident using the system").

Precisaria de aprender muitas coisas novas antes de poder utilizar este sistema.
3 responses

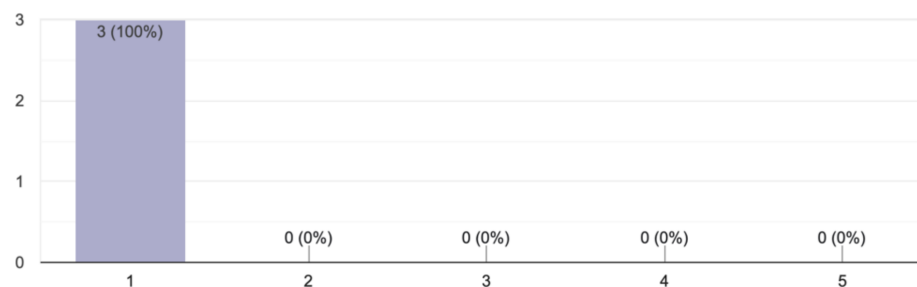


Figure 5.10: Results of the 10th SUS question ("I needed to learn a lot of things before I could get going with this system").

SUS Score	Grade	Adjective Rating
> 80.3	A	Excellent
68 – 80.3	B	Good
68	C	Okay
51 – 68	D	Poor
< 51	F	Awful

Figure 5.11: Guideline for interpretation of SUS score. [21]

Como classificas a dificuldades das tarefas?

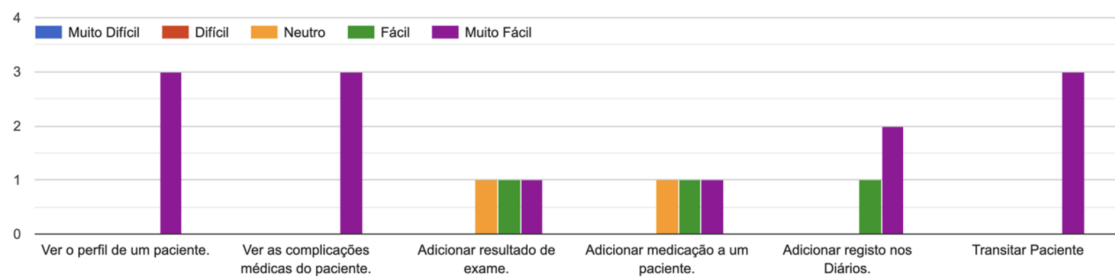


Figure 5.12: Participants feedback on the difficulty of the proposed tasks.

Chapter 6

Future Work

The developed system and the feedback obtained from usability tests show that the proposed platform may constitute a helpful tool in solving the problems outlined in Chapters 1 and 2. This chapter explores which features and improvements could be implemented as future work to best benefit the platform users. Some of the features outlined below have already been prototyped, which should facilitate the development.

One of the top priorities for the project's next iteration should be authentication as a physician. Physicians should have personal, protected accounts on the platform, be able to log into their accounts, and view and edit their information. The current form to add a physician should be a form to register on the platform instead, with added authentication parameters. Physician authentication would also simplify some operations, such as creating an appointment: it would not be necessary to select the physician associated with the appointment since it would automatically be the one that is logged in at the moment.

Listing secretions analysis results and signaling patients for transplants were two other unimplemented discussed with the project specialists. Patients regularly send their secretions for analysis, and their physicians should be able to note when secretions are delivered and when analysis results are available. This feature should, at least, allow physicians to record a secretion delivery of a patient and then, when results are available, log these results on the platform. The transplant feature is necessary since a small but significant percentage of CF patients have to undergo transplants, usually of the lungs. This feature should allow physicians to mark which stage of the transplant process a patient is in, if the patient is going through one. It should also provide a list of transplant patients, with the option to view only patients on a specific stage, which allows physicians to see, for instance, the waiting list for transplants. Though further discussion is needed to specify this feature better, the stages of the transplant process should, in principle, include at least referral, waiting list, and post-transplant.

CF patients have various medical exams that must be performed periodically. To facilitate this process, a feature that allows physicians to register these exams on the patient's profile, and view a list of how much time is left until they need to be performed again, should be implemented.

Appointment notes input should also be more personalized, which would allow physicians to

more easily input data in an appointment and reduce human error. An input form with various personalized fields also allows the creation of graphs, global or specific to the patient, that can track several data, such as iron levels. This feature requires detailed and extensive discussion with physicians to define the fields, and it is necessary to consider that these fields will vary between pediatric, transition, and adult patients. When submitting this appointment data, a plain text report should also be generated, so that it could be duplicated into the hospital's system.

A notification system can further improve the value of the features mentioned above. Physicians can be notified of when patients deliver secretions, when analysis results are available, or when a patient has an upcoming periodic exam.

The platform's fundamental features should be available offline to improve the system's usability and availability. One of the ways this can be done is by implementing a client-side database that periodically syncs with the server. With this implementation, information about the patients should be available offline, and physicians should be allowed to view and edit this information and add therapies and appointment notes without a connection. This way, if the connection fails during or before an appointment, the physician can still register its notes and any relevant patient updates, which will be synced with the server when a connection is available again.

For a more efficient experience, the platform should integrate with the hospital's current system. The data of the platform and the system in use should sync so that appointment notes, therapies, and other information does not need to be inputted separately into the two systems. This implementation would also allow physicians to view their appointment calendar on the platform and schedule appointments that would then be officially scheduled on the hospital's system. This integration poses several bureaucratic and technological challenges and is expected to be a lengthy process, due to which it was not implemented in this project iteration. Although ambitious and challenging, this integration is expected to bring significant benefits to its users and create the potential for many more features. Thus, it should be treated as a priority.

A web or mobile application for patients can be implemented in the future to complement the functionality of the web application implemented for physicians. Patients could authenticate into their account and view and update some of their data, report when they deliver secretions, and, if the authentication with the hospital's system is implemented, see a calendar with their future appointments. They could also chat with physicians between appointments, making communication more efficient since it is centralized on one platform.

Chapter 7

Conclusions

This dissertation focused on creating a solution for the problem of non-personalized EHRs in CF. Due to the complex and specific nature of the illness, a customized EHR system would significantly streamline the data input and access work done by physicians, also allowing them to provide better quality care.

To achieve this goal, the requirements were thoroughly discussed and defined with physicians, a prototype for the application was designed, and a web application that followed the defined requirements and prototype was implemented. The project's initial scope was reduced in order to guarantee an implemented proof of concept within the given time frame.

Consistent communication between all parties involved in the project was one of the critical elements of its development. The knowledge of health professionals, especially those that work with patients with CF and are aware of its specificities, was essential for the successful development of the prototype and the web application.

The web application was implemented keeping in mind that there should be future iterations of development. This way, the project's code was structured in a way that aims to simplify future development by being easily readable, understandable, and carefully organized.

After the development of the prototype and implementation of the web application, the prototype was subject to usability testing. The feedback from these tests was highly positive and encouraging.

The understanding of the context of the problem, discussion of requirements, prototyping, implementation, and positive feedback received, shows that this dissertation can constitute a solid basis and proof of concept for an EHR system that solves the current problems. Although future work (including overcoming bureaucratic and technological challenges) is necessary for the application to be entirely usable and integrated with the physicians' day-to-day work, the usability results showed great potential and enthusiasm for the project.

This dissertation succeeded in developing a proof of concept for an EHR system adapted to the specific needs of patients with CF, validated by physicians. It is expected that, with the continuation of the development of the project, the EHR system could be expanded to other CF

centers, integrated with their systems, and even complemented by an application made for patients, improving the quality of provided care.

References

- [1] Azure blob storage pricing. Available at <https://azure.microsoft.com/en-us/pricing/details/storage/blobs/>, Accessed last time in September 2022.
- [2] Digital health platform for chronic care | welldoc. Available at <https://www.opennotes.org/>, Accessed last time in February 2022.
- [3] Sonho - aprendis. Available at <http://aprendis.gim.med.up.pt/index.php/SONHO>, Accessed last time in September 2022, 6 2017.
- [4] Sonho hospitalar | sistema integrado de informação hospitalar – spms. Available at <https://www.spms.min-saude.pt/2019/01/product-sclinicohospitalar/>, Accessed last time in September 2022, 1 2019.
- [5] Express/node introduction. Available at <https://developer.mozilla.org/en-US/docs/Learn/Server-side/Exp/Introduction>, Accessed last time in September 2022, 9 2022.
- [6] Getting started | axios docs. Available at <https://axios-http.com/docs/intro/>, Accessed last time in September 2022, 2022.
- [7] Introduction to azure storage. Available at <https://docs.microsoft.com/en-us/azure/storage/common/storage-introduction/>, Accessed last time in September 2022, 3 2022.
- [8] MongoDB: The developer data platform. Available at <https://www.mongodb.com/>, Accessed last time in September 2022, 2022.
- [9] React-bootstrap · react-bootstrap documentation. Available at <https://react-bootstrap.github.io/>, Accessed last time in September 2022, 2022.
- [10] React router. Available at <https://reactrouter.com/en/main/>, Accessed last time in September 2022, 2022.
- [11] React – a javascript library for building user interfaces. Available at <https://reactjs.org/>, Accessed last time in September 2022, 2022.
- [12] John Brooke. Sus: A quick and dirty usability scale. *Usability Eval. Ind.*, 189, 11 1995.
- [13] Anna-Karin Carstensen and Jonte Bernhard. Design science research – a powerful tool for improving methods in engineering education research. *European Journal of Engineering Education*, 44:85–102, 3 2019.

- [14] Bill Kuechler and Vijay Vaishnavi. On theory development in design science research: anatomy of a research project. *European Journal of Information Systems*, 17:489–504, 10 2008.
- [15] Brian P O’Sullivan and Steven D Freedman. Cystic fibrosis. *The Lancet*, 373:1891–1904, 5 2009.
- [16] Beryl J. Rosenstein. Cystic fibrosis - pediatrics. Merck Manuals Professional Edition. Available at <https://www.merckmanuals.com/professional/pediatrics/cystic-fibrosis-cf/cystic-fibrosis#>, Accessed last time in February 2022, 8 2021.
- [17] Tim G Schmitz and Lutz Goldbeck. The effect of inpatient rehabilitation programmes on quality of life in patients with cystic fibrosis: A multi-center study. *Health and Quality of Life Outcomes*, 4:8, 12 2006.
- [18] Girish D. Sharma and Kenan Haver. Cystic fibrosis. *Medscape*, 2020.
- [19] Mark Smallcombe. Introduction to azure storage. Available at <https://www.integrate.io/blog/the-sql-vs-nosql-difference/>, Accessed last time in September 2022, 7 2021.
- [20] Suraj Surve. Why you should use react.js for web development. Available at <https://www.freecodecamp.org/news/why-use-react-for-web-development/>, Accessed last time in September 2022, 2 2021.
- [21] Will T. Measuring and interpreting system usability scale (sus), 2 2021.
- [22] Hideaki Takeda, Paul Veerkamp, and Hiroyuki Yoshikawa. Modeling design process. *AI Magazine*, 12 1990.
- [23] Vijay Vaishnavi and Bill Kuechler. Design science research in information systems. 1 2004.
- [24] James R. Yankaskas, Bruce C. Marshall, Beth Sufian, Richard H. Simon, and David Rodman. Cystic fibrosis adult care. *Chest*, 125:1S–39S, 1 2004.