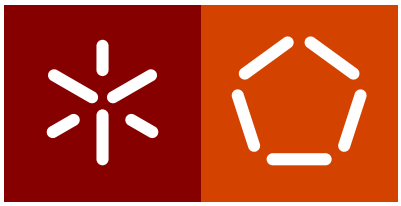
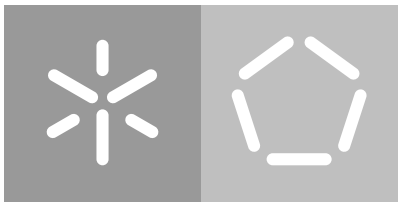


**Universidade do Minho**
Escola de Engenharia
Departamento de Informática

Rúben Correia Cerqueira

# Development of a Web Clinical Management Application

September 2023



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

Rúben Correia Cerqueira

# Development of a Web Clinical Management Application

Master dissertation
Master Degree in Informatics Engineering

Dissertation supervised by
**Professor Pedro Rangel Henriques**

September 2023

## STATEMENT OF INTEGRITY

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

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

Rúben Cerqueira

_____

# ACKNOWLEDGEMENTS

First and foremost I am deeply grateful to my supervisor, Prof. Pedro Rangel Henriques for his invaluable guidance, patience, and support. His wealth of experience, extensive knowledge, and constant availability have encouraged me through the course of this project and led me to complete this thesis.

My gratitude extends to the Informatics Department, University of Minho for the opportunity to take part in this project. Additionally, I would like to thank Wintouch for providing the theme that served as the foundation for this thesis.

Lastly, my appreciation also goes to my family and friends. Their unwavering belief in me has been a constant source of motivation, and I am immensely grateful for their presence in my life.

# ABSTRACT

The time of doing all the work manually is passing by as the influence of developing technology is increasing. Most of the tasks done in a business can be automated by software. Because of that, the growing demand for this kind of technology is making IT companies develop any software that is required.

This report — the dissertation that describes a thesis in Informatics Engineering — covers some of these technologies, focusing on the clinic area. The development of a clinic web application was proposed by Wintouch to help clinic businesses boost their productivity and organization. This Master's work, herein reported, began with the research of the state of the art, studying what the market is like, and analyzing what are the drawbacks of the existing similar applications. The lessons learned at that stage were relevant to design a new web application that can stand out above competitors. The design of the application's architecture is discussed below along with the technologies used to best fit the application to reach the objectives proposed and meet the desired requirements. The report presents a detailed account of the outcomes of the development process, encompassing both backend and frontend implementations. Notable features and functionalities are thoroughly documented, alongside a reflection of the challenges encountered during the development journey.

**Keywords:** software engineering, web application, clinic software, software development

# RESUMO

O tempo de realizar todo o trabalho manualmente está a passar à medida que aumenta a influência do desenvolvimento tecnológico. A maioria das tarefas realizadas num negócio pode ser automatizada por software. Devido a isso, a crescente procura por este tipo de tecnologia está a levar as empresas de TI a desenvolverem qualquer software necessário.

Este relatório - a dissertação que descreve uma tese em Engenharia Informática - aborda algumas dessas tecnologias, com foco na área clínica. O desenvolvimento de uma aplicação web para clínicas foi proposto pela Wintouch para ajudar a impulsionar a produtividade e a organização dos negócios clínicos. Este trabalho de mestrado, aqui relatado, começou com a pesquisa do estado da arte, estudando como está o mercado e analisando as limitações das aplicações similares existentes. As lições aprendidas nessa fase foram relevantes para projetar uma nova aplicação web que se destacasse dos concorrentes. O design da arquitetura da aplicação é discutido abaixo, juntamente com as tecnologias usadas para melhor adequar a aplicação aos objetivos propostos e satisfazer os requisitos desejados.

O relatório apresenta um relato detalhado dos resultados do processo de desenvolvimento, abrangendo tanto as implementações do backend quanto do frontend. Recursos e funcionalidades notáveis são minuciosamente documentados, juntamente com uma discussão dos desafios encontrados durante a jornada de desenvolvimento.

**Palavras-chave:** engenharia de software, aplicação web, software para clínicas, desenvolvimento de software

# CONTENTS

# LIST OF FIGURES

# ACRONYMS

**D**

**DB** Database.

**E**

**EF CORE** Entity Framework Core.

**G**

**GDPR** General Data Protection Regulation.

**O**

**ORM** Object-Relation Mapping.

**OS** Opearative System.

**S**

**SPA** Single Page Applications.

**U**

**UI** User Interface.

**UX** User Experience.

# 1

# INTRODUCTION

Nowadays, modern digital technology has been taking over the lives of so many people as it is continuously being upgraded and developed in order to automatize manual processes and tasks to help people with their needs. This automatization process can be seen and felt everywhere, for instance, our cell phones allow us to call other people, but also they provide us with other services that support in diverse areas such as restaurants (Buergermelster and Loenen van, 1990) or even social security.
Every business can take advantage of the recent computer-supported technology to ease processes of the company that take a lot of effort to organize and manage. Tasks like invoice production and printing, stock control, and salary management can all be automated by a single well-designed and developed application that can save a lot of money for the company and simplify its way of work (Borzekowski, 2009).

## 1.1 MOTIVATION

There are many ways digital technology can simplify and help both patients and professionals in the clinics' area. With the help of a web application, the process of making an appointment can be effortlessly done without being present in the clinic, making patients avoid long waiting queues and the scheduling more straightforward (Almomani and Al-Sarheed, 2016), avoiding conflicts and common human errors. Additionally, the professionals would have access to the client's medical records with a simple click and sorted by date in any place, not only inside the clinic but in pharmacies as well, as patients would not need to take paper prescriptions with them because the pharmacy would have access to that via web application (Sridhar et al., 2009). Those are the clinic processes that would have more advantages using computation technology, but it would also offer the company's common general automatization tools as well, like stock and human resource management, among others.

This context is enough to explain the motivation of Wintouch, an established Portuguese software house leader in retail applications and software for the restaurants market, that is looking forward to developing a new application to support the clinics' business.

## 1.2 OBJECTIVES

The main objective of this Master's Thesis is to develop a Web-based clinic management application that can be further launched in the market. In order to reach that aim, there are some issues that need to be achieved:

- Design and implement a complete, secure, and error-proof web application that can be used by both clinics and patients;
- Ensure a high-level performance of the application and great distinction among other similar applications in the market;
- Assure that all the required functionalities are implemented in the final product.

## 1.3 RESEARCH HYPOTHESIS

By the end of this Master's work, a web clinic management application should be fully operational and running, with the objective of having success in the market and having a strong impact on the clinic software area. The final product, not only should have success in the market but also help with all the necessities a clinic has to grow its business and boost its workflow complexity and economy.

## 1.4 DEVELOPMENT APPROACH

To grant the main objective of this Master's Project, there is a need to adopt an adequate development methodology. The approach that will be followed in this project is composed of the steps below.

- Market study, including the deep identification of the clinics' requirements and the applications provided by competitors, to understand the best approaches to utilize, as well as the best suitable technologies available;
- System architecture design and implementation planning;
- Scrum approach to have a good team communication and synchronization process;
- Development of the web application following the Agile strategy proposed above;

- Results comparison and analysis;
- Incremental unit testing for quality and error-proof assurance.

This is an incremental method, meaning that problems detected in any phase will imply going back and repeating previous tasks.

## 1.5 Document Structure

This Master's dissertation is composed of eight chapters, which are the introduction, state of the art, system architecture, technology selection, frontend and backend development, obstacles and conclusion. In the present chapter, the project context and objectives were presented, along with the expected outcomes of this work.

Chapter 2, the state of the art section will first start to expose the study done in the context market area, describing some suitable potential other clinic management applications, their best and weaker points and finally comparing them, observing what's missing in each one and taking that into account when planning the desired web application.

Chapter 3, system planning begins, with the start of the requirement gathering and further mockup design. This project phase is vital to the success of the application development.

Chapter 4 provides a comprehensive overview of the technologies employed in the application's development.

Chapters 5 and 6 delve into the core objective of this thesis, which is the development of the application. These chapters will elaborate on the development process, present the achieved results, and discuss encountered challenges.

Chapter 7 focuses on the primary obstacles encountered during the development of this Master's project. Additionally, it highlights the learning opportunities derived from these challenges and the strategies employed to overcome them.

Chapter 8 offers a reflective analysis of the completed work, along with a presentation of the project schedule.

Appendix A contains additional information pertaining to a complex component that has been developed specifically for the frontend aspect of the application.

# 2

# SOFTWARE FOR CLOUD CLINIC MANAGEMENT

There is already some work done in the context of cloud clinic software to explore. In this chapter, there will be a presentation of some of the Clinic Web applications already developed, along with some features offered by those software platforms. For each one, the pros and cons will be discussed. This study is crucial to analyze which aspects are missing in the market in the context of clinic management software and what can be explored to have success among the competitors.

## 2.1 DESCRIPTION OF CLINIC WORKFLOW

In a clinic, there are various tasks in its workflow that are clearly candidates to be automatized by an adequate software system.

One of the tasks is the **agenda** or **scheduling**. It is characterized by the reserved time of a day in which the client requested to attend to the services of the business. A good scheduling process can decide the clinic's quality of service. The scheduler must guarantee that a doctor is not being overwhelmed by clients, that is, not having more patients than those he can assist, but also having room for future ones if that is the case. The aspect of human error should be minimal here since a mistake can have a negative impact on a client or even on the doctor, like appointing a physician that is on vacation at that time or scheduling a doctor that is attending to a high number of clients at the same time. At the same time, there is a need to keep a small waiting time queue as clients don't want to be waiting for the booking too long.

Aside from scheduling, there is the booking itself. There is a difference between an **appointment** and a **booking** or **scheduling**. An appointment refers to the act of the doctor listening to the patient's complaints and analyzing the situation so the professional can write a proper prescription for the patient. The latter refers to the block of time reserved for the client to receive the service requested to the clinic, being that an appointment itself, vaccination, or ministrations of the same kind.

In an **appointment**, the first thing the medic needs is the patient history, so his notebook with the details of each previous meetings should be immediately in the hands of the

professional in order to start the present appointment. The notes on the patient's clinical history can come in paper format or digital, but the majority of the time, paperwork is hard to find and time-consuming, so opting for a digital source can reduce the startup time of the service as it can be retrieved through the client's clinical ID.

After obtaining the document, it should be updated as the appointment progresses. That additional information should be appended to the document and consequently, to the **patient's record**. Using paper support, that information can easily be lost, that is why a digital alternative can come in handy to have all the information needed about that patient concise, grouped, and stored in just one place. In a digital support, the patient record can be easily and rapidly retrieved, updated, and saved.

At the end of the **appointment**, there is a need for a prescription. For efficient performance, the doctor has to search for available medicines in order to choose rapidly those that better fit the patient's needs to write the respective prescription. This can be rapidly done by software and properly organized to be further printed so the pharmacies can identify the drugs present in the document and sell them to the client mistake-less.

Also, the state of the clinic can be monitored through the software. There is a possibility to keep track of the appointment's state, that is, if the patient missed it, or if it ended but the payment process has not finished. Additional variables can be also monitored like stock management — which is crucial to check if the service has the conditions to be performed — and human resource management, vital to the operation of every business.

Additionally, there is a need to preserve data confidentiality due to the nature of the business. To assess this issue, the customers have to give permission for the business to store their data by signing a *General Data Protection Regulation* (*GDPR*)[1] document. The software must store the *GDPR* data to avoid data protection-related concerns.

## 2.2 Introduction to clinic management software

There are several clinical applications available from the national or international market. Although all of them provide similar features to support the clinic workflows, the different platforms offer of course distinct interfaces and performances. In the next subsection, some of the platforms found will be explored — namely, *LinkedCare*, *iMed*, and *CliCloud*. The criteria of selection were the availability of information about the application and the competitors' focus — *Alert* developed by *Alert*, *GlobalCare* developed by *Glintt* — as some of them prefer to be more private about the contents and features of their applications and tend to focus on hospitals instead of clinics. As some of them don't offer a trial version, they can only be analyzed through videos or photos of the application made available by them in order to promote the product. There is some other software that was considered

1 https://gdpr-info.eu/

but there were some drawbacks that could not allow its analysis, for instance, *Consultorio Grátis*, which blocks users as they request clinic-only documents to progress its use, and *MedicineOne*, which only have a native desktop version, not adequate to cloud application analysis. Three main features will be analyzed which are:

- Agenda Overview;
- Patient Record Page;
- Appointment Page.

The **agenda overview** represents the view where scheduling takes place. It is one of the most important views of the application as it allows patients to use the clinic's services.

The **patient record page** contains the information of each patient that attended the business. It is crucial to have a smart organization of the information so the querying can be more straightforward and simple.

The **appointment page** permits the doctor to have a place to write everything relevant retaining in an appointment. With a simple view, it can be useful to have less long appointment processes and to provide more accurate prescriptions for the patient.

Those three features will be especially analyzed as they will be the most used hubs by the clinic's professionals being the key parts incorporating the application.

### 2.2.1 *LinkedCare*

LinkedCare[2] is a clinic web application developed by an Indian company that shares the same name as the application. It is very complete as it covers many of the clinic's necessities like scheduling, booking management, and patient record storage. Its main feature is the availability of scheduling online bookings, so patients don't have the need to move to the clinic. It is a worthy way of reducing crowding at the establishment and long waiting queues.

The application provides a free trial version. This allows for deeper experimentation and so a more complete description is possible.

2 www.linkedcare.com

*Agenda*

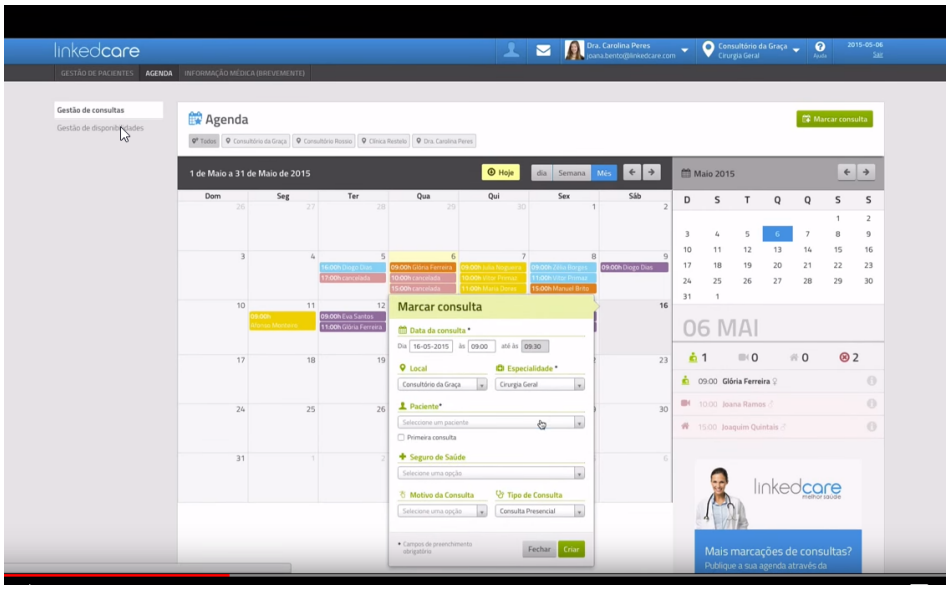



Figure 1: LinkedCare agenda page with an appointment scheduling popup.

It is possible to observe some aspects in Figure 1. This option displays a traditional calendar exhibiting all the appointments settled in each day of the week. For each slot shown in the calendar, some information fields are available: the place, specialty, patient data, scheduling reason, and appointment type. There is also a synthesis dashboard on the right, containing the appointments for the day, the number of schedulings of the day by type, the beginning time, and if it was canceled or not. However, during its demo time, the bug quantity of the agenda was noted, that being one of the disadvantages of the application.

*Patient Record*

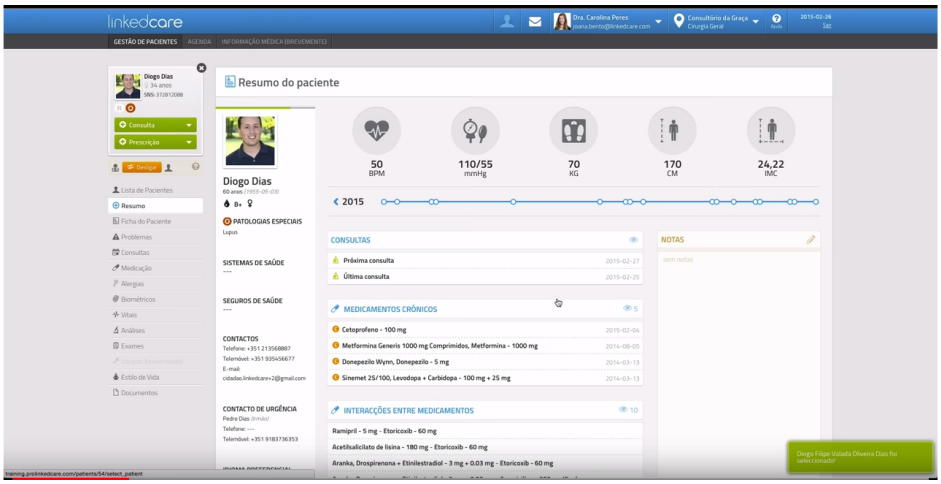



Figure 2: LinkedCare patient record page.

Figure 2 shows that the patient's record is full of information, like its allergies, next and previous appointment, analysis, vitals, current medication, and more. The *User Interface* (*UI*) looks light and has a basic color palette. The simplicity of the dashboard allows the user to retrieve the most important information about the patient first, before exploring more details about the subject.

*Appointment Page*

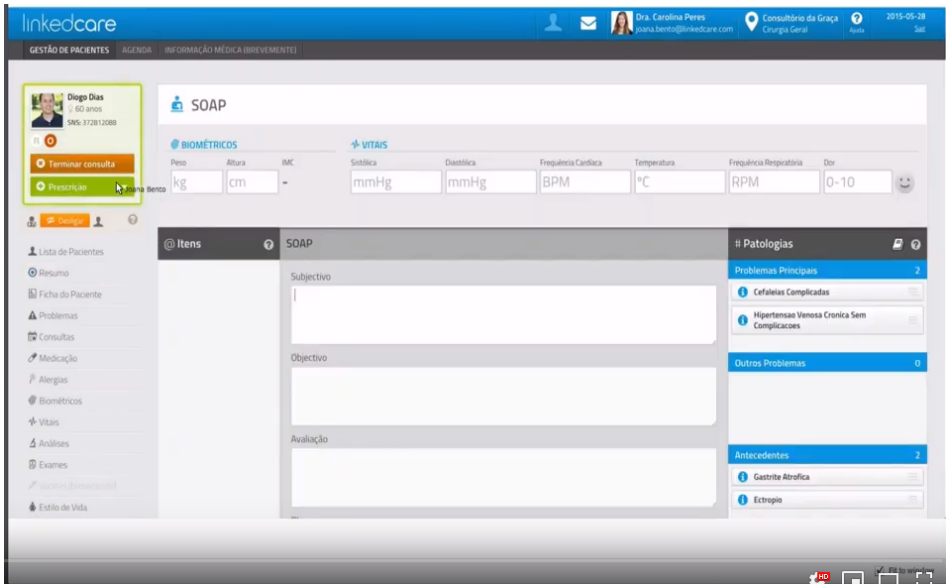



Figure 3: LinkedCare appointment page.

As can be analyzed in Figure 3, there are some characteristics that an appointment needs to have in order to help the doctor register the occurrence and have the right measures to write an appropriate prescription for the client. Basic data should be obtained such as weight and height, as well as the patient's temperature and beat pulse, as they are regular metrics that the doctor needs to track the patient's state. Additionally, it has the SOAP system (subjective, objective, assessment, and plan) to aid with the appointment process. It is a simple *UI* for the user to work on and has decent shortcuts to other types of patient information.

### 2.2.2 *iMed*

Imed[3] is an application developed by ACIN (https://acin.pt/), a Portuguese company with its head office located in Madeira. It's a clinic management software that has a similar solution to LinkedCare, having features like schedule bookings, prescription writing, invoicing, among others. It is one of the most influential applications in the market marked by its simplicity and completeness.

*Agenda*

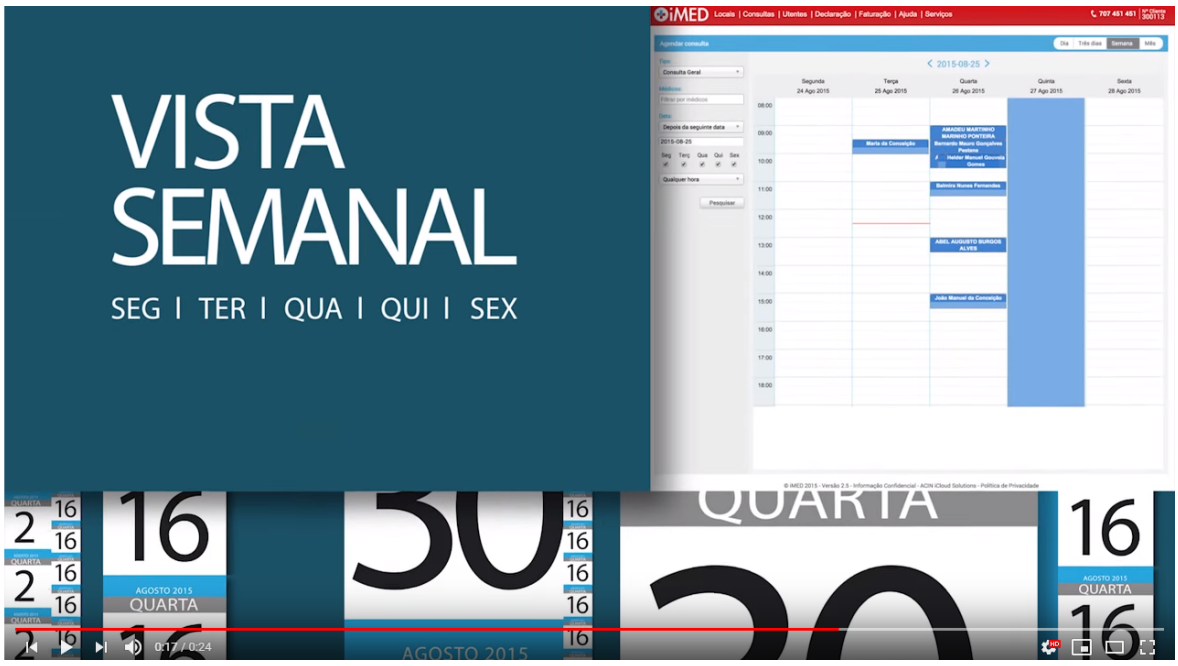



Figure 4: iMed agenda page (weekly view).

It can be observed that the application opted for a simplistic view of the agenda, analyzing Figure 4. There is no additional information besides the scheduler with the days of the week

3 https://imed.pt/

and the appointments already booked. It is also possible to filter the scheduled appointments by day of the week, hour, type, and medic.

They offer the possibility of changing the view of the agenda to day, three days, and monthly. The latter is reasonably different from the presented one, which is going to be shown subsequently.

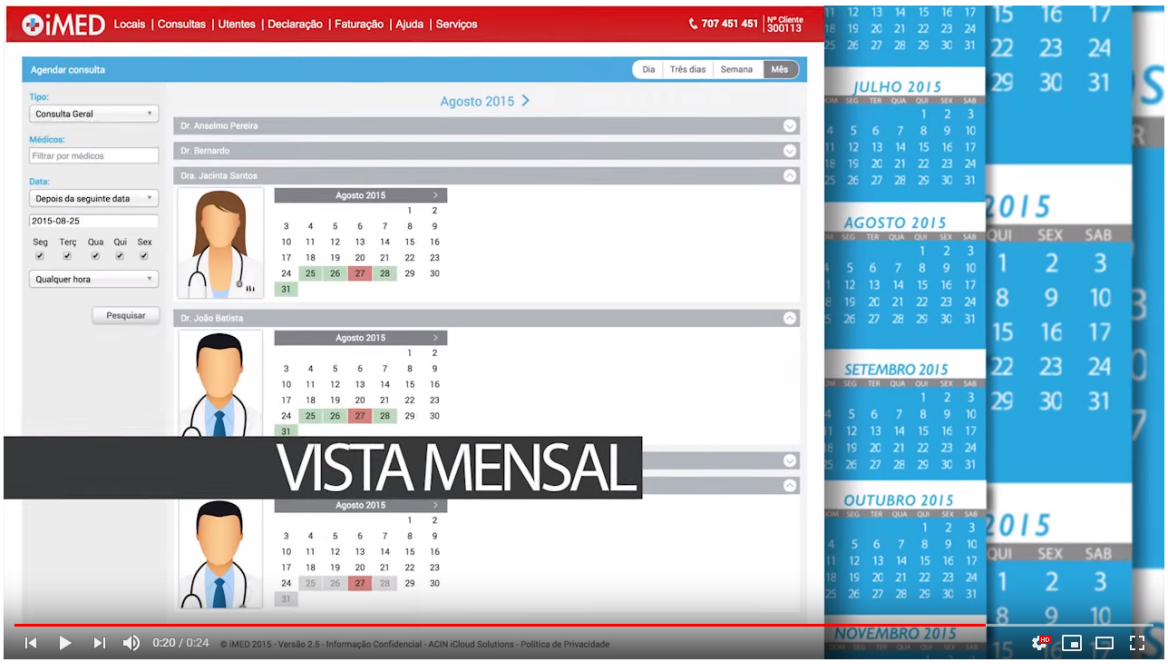



Figure 5: iMed agenda page (monthly view).

It is noted that they took a different approach to the monthly agenda. Instead of showing an entire calendar with all the bookings scheduled, there are tabs for each doctor and each of the tabs contains an agenda with only the bookings corresponding to the respective physician. This is a good solution to reduce the number of bookings that the calendar needs to show, as clinics tend to have lots of schedulings in a day for each doctor. The only con to this approach is that the user is not able to see the general calendar month view even if it has a few bookings registered. So the user needs to scroll through the doctors to find the desired information.

*Patient Record*

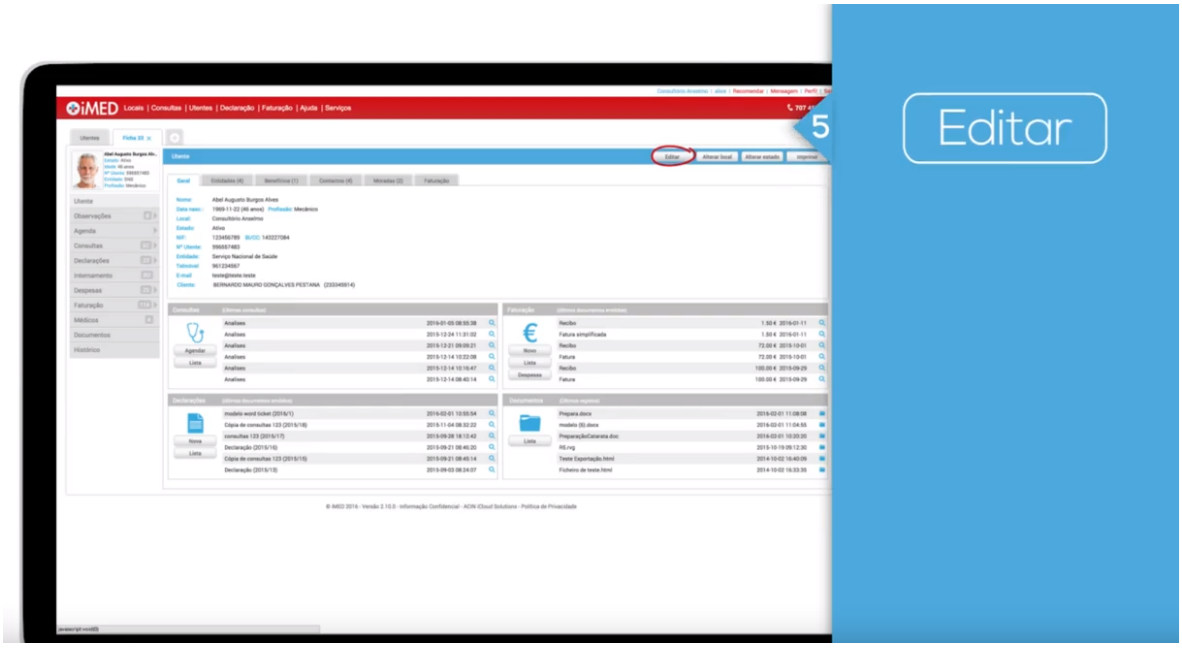



Figure 6: Patient Record Page.

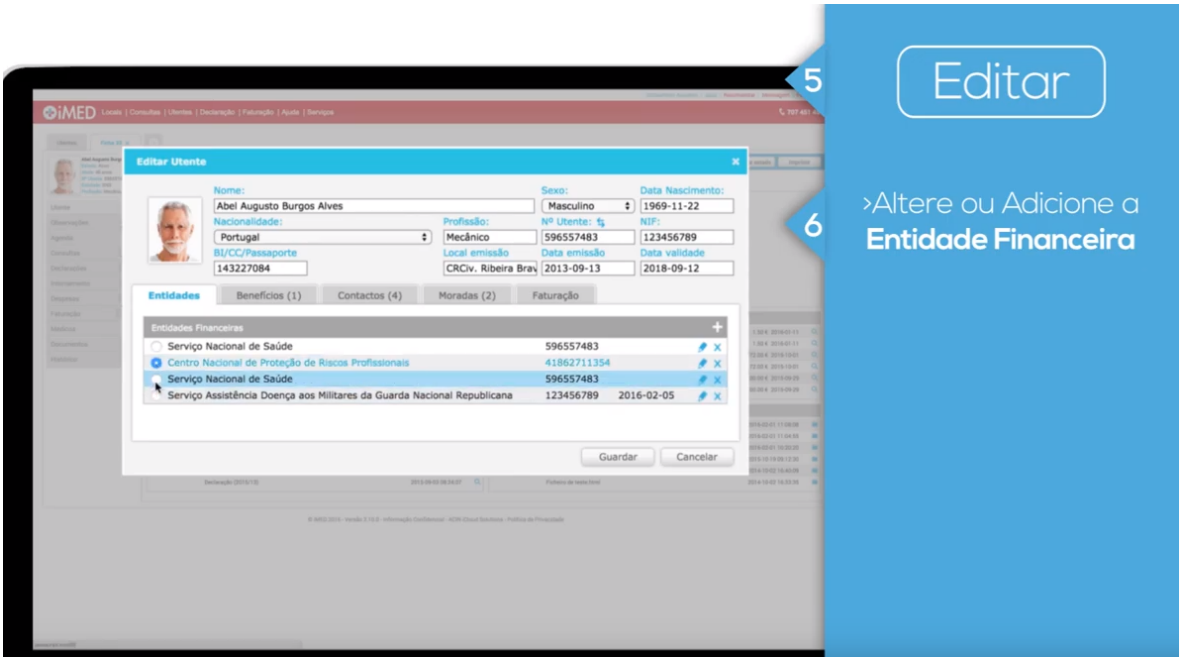



Figure 7: iMed patient record edit page.

Figure 6 shows a detailed page with every information from the patient. The data kind range goes from general and clinic to invoicing, allowing the user to access any type of info

he wants by merely clicking through the categories on the left side of the page. This way of presenting the patient's page has the advantage of having one point of access to all his information, but on the other hand, the excess of tabs and fields makes the experience less user-friendly as users prefer to be guided through the application to accomplish their tasks, not having a high amount of data populating the screen with one click. There is an editing feature that can be observed in figure 7. The main data fields of the patient are permanent on the window and are editable. The additional information is tabbed below the main panel.

*Appointment Page*

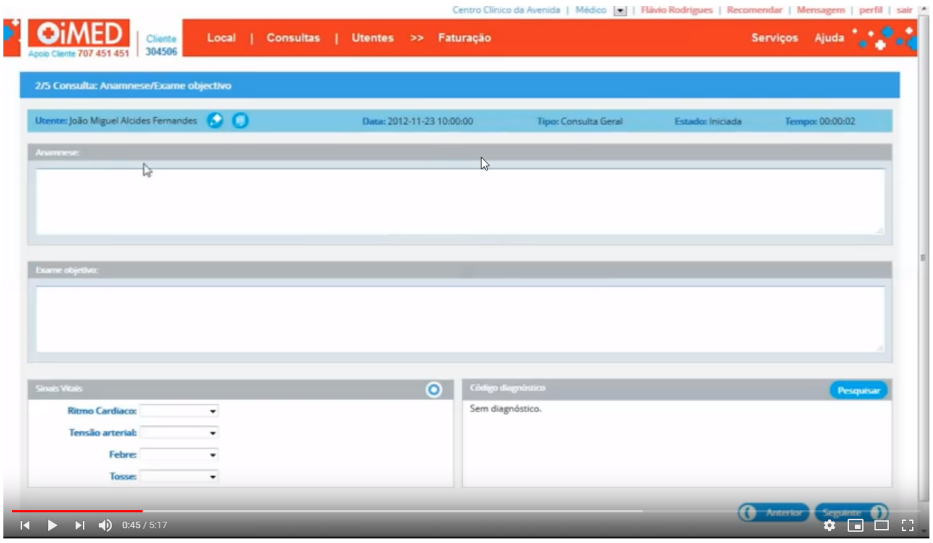



Figure 8: iMed appointment page.

As an appointment is an iterative process, the team behind iMed took the same approach to design the application's appointment page. It is composed of a wizard that has five stages.

The first one consists of the main information of the service, like the patient to be attended to, the doctor that is performing the service, and its type, among others. The next stage is illustrated in figure 8. There is a dashboard with the function of collecting the patient's data through his complaints, as well as the client's symptoms analyzed by the physician.

Consequently comes the documents emission. In this phase, the doctor selects the medicines that best suit the patient's state to write his prescription. In this stage, the user has the power of providing medical declarations like absence declarations.

The final steps include document previsualization and printing, and finishing the appointment process.

### 2.2.3 *CliCloud*

CliCloud[4] is a cloud clinic management software developed by GlobalSoft. Like the previous software analyzed, this solution offers the key features that a clinic needs for its workflow like an agenda, patient records, appointments, treatments, and even invoicing. Due to the lack of a trial version and missing footage on their information pages, there is no information that can be obtained regarding the appointment page of the application.

*Agenda*

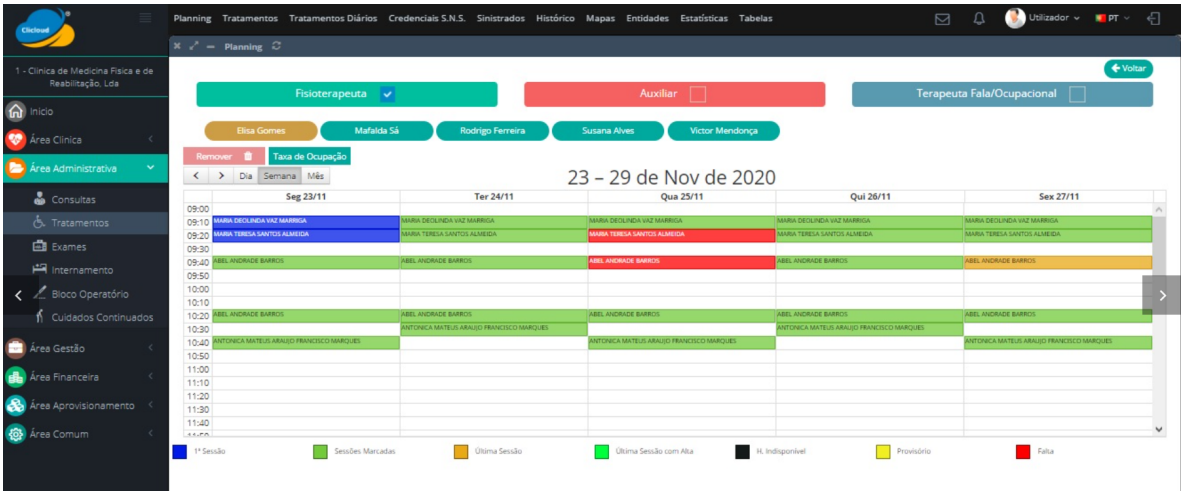



Figure 9: Clicloud agenda overview.

CliCloud agenda overview provides basic utilities to allow the user to realize CRUD (create, read, update, delete) operations in a calendar, as can be observed in Figure 9.

The view consists of the filters section in the upper part of the page, and the bottom part is populated by the calendar view. The user can filter the events appearing in the calendar by doctor specialty and by the medic itself.

There is also a week and day view of the calendar. That flexibility is an advantage as it provides a way for the user to have different perspectives of upcoming events and have an idea of future workload.

Additionally, the calendar provides a way of differentiating the events based on types, for instance, the first session, follow-up session, provisional event, missed events, and a few more.

4 http://clicloud.pt/

*Patient Record*

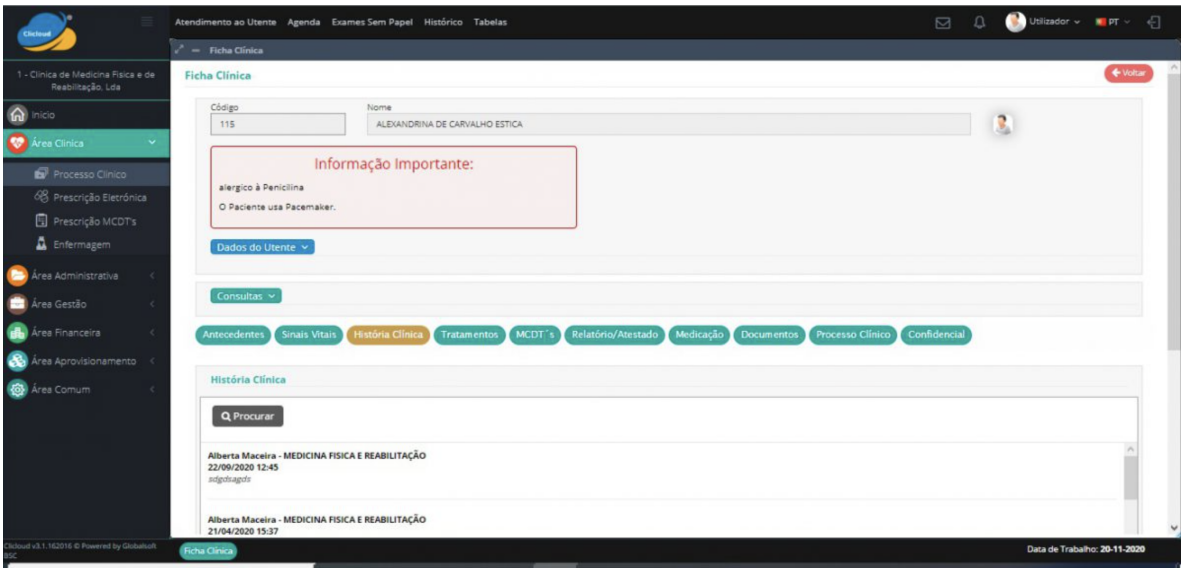



Figure 10: Clicloud patient record page.

The patient record page of CliCloud has a user-friendly look, having as little information as possible, but also offering options to access all types of information of the context's individual (Figure 10). There is a button to access the patient's basic information, below a panel that shows important data about the subject, like its allergies or other matters. The bottom of the page consists of a tabbed panel with additional info about the patient. This additional information includes past medication, treatments, vital signs, and other clinical data that is relevant for a superior clinic quality of service.

## 2.3 SUMMARY

After having a peek at some of the concurrent software, it can be observed that, regarding functionality, the applications have pretty similar workflows. As the applications serve a common purpose, it was expected that their features were similar.

Their main differences are the *UI* perspective, as they have different color palettes and data organization, so it's up to the user to choose the way he wants to operate with the business.

# 3

# HEALTHCARE - PROPOSAL

With the state of the art analyzed, there is a lot of information that can be used to establish a starting point for the current project and delineate the foundation for it. In this chapter, it will be discussed the planning for the working project named Healthcare, bearing in mind what's being successful in the market at the moment.

## 3.1 REQUIREMENTS

For an application to be successful, there is a need to establish some parameters to outline the purpose, the features of the application and to ensure good product performance. Those parameters are called requirements and they serve as a guide for the developer to accomplish the desired final output.

The requirement engineering process generally consists of four steps:

1. Elicitation and analysis;
2. Specification;
3. Validation;
4. Management.

**Elicitation and analysis**, also known as **gathering of requirements**, is the process of identifying and documenting the project's exact requirements from start to finish. It is one of the most critical phases of the project as it has an impact as a whole and can often lead to project failure if incorrectly executed (Lane et al., 2016). This process can be done by interviewing the stakeholders so that the developing team can get the information they need to establish the requirements and progress to the project's next phase.

**Specification** translates to converting the ordinary language, gathered on the previous page, to technical language so the requirements can be better understood and beneficial by the developing team. This technical language may consist of the rephrasing of the requirements, in its categorization, or even the utilization of some tools like data flow

diagrams or entity-relation diagrams. This phase is essential for reducing requirement ambitiousness and delineating the product in agreement with the stakeholders' needs.

**Validation** ensures the final product meets the stakeholder's needs in a stable and reproducible way. It is important to have this validation as it can save potential time-consuming and expensive reworks. This process can be achieved by communication, test case generation, and manual inspection.(Maalem and Zarour, 2016)

**Management** is a process present in the whole system development process. It consists of managing changing requirements as the development progresses as new requirements can emerge as a consequence of business changes (Parsanezhad et al., 2016).

These steps effectively guide the development team as they provide a safe and controlled way of requirement evolution.

### 3.1.1 *Project's Requirements*

For this Master's project, the requirement formulation was executed by interviewing professionals that have knowledge about the clinic's software needs and by examining competitors' products. The latter process is rewarding as the competition had been through the requirement process, so the final application came by the execution of the requirement cycle, which is a safe way of guidance.

*Functional Requirements*

- The application has an agenda view;
- The agenda view has an interactive calendar;
- The user should use the calendar to add, edit, view, and delete bookings;
- The user should use the calendar to add, edit, view, and delete notes;
- The calendar should be able to make recurrent bookings;
- The calendar should be able to filter booking by specialty, type of service, and medic;
- The application allows booking search;
- The user should be able to book hours not defined in the schedule;
- The calendar should allow users to select various cells according to the booking length;
- The application should allow to show available periods;
- The calendar should show each booking's state;

- The application should have a view showing all the patients registered;
- The patient selection page should allow the user to create a new patient record;
- The application should show the patient record after selecting one patient in the patient's selection page;
- The patient record should show all the information about the patient;
- The application should have a configuration page;
- The application should have an appointment page;
- The appointment page should have a section to fill in appointment information;
- The appointment page should allow the user to access the patient clinical history;
- The appointment page should allow to attach files to it;
- The application should have a medic page;
- The medic page should show the list of all the medics working in the clinic;
- The medic page should have all the information of the medic;
- The application should be able to generate invoices from services provided;

*Non-functional Requirements*

- Microsoft technological stack should be used in the development of this project;
- The system should distribute its backend using a microservice approach;
- The system should be able to communicate with multiple microservices asynchronously;
- The system must be able to support multiple visits while maintaining optimal performance;
- The system should be able to provide the requested data to the clients in less than two seconds;
- The application should be running, at least, 95% of the time;
- The application should be able to support English and Portuguese;
- The system should have a basic layout shared between pages composed of navigation options.

## 3.2 System Architecture

One of the early decisions that had to be done is to build an architecture of the system, illustrating the flow of the application, from the clients to the course of their requests to the server. It will also demonstrate in a general way, what layers of the system will communicate with each other in order to address the requests directed to the server.

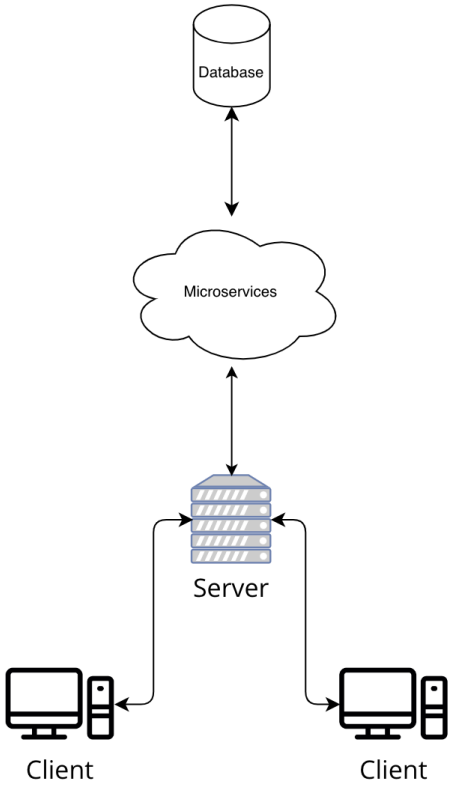



Figure 11: System Architecture.

Figure 11 demonstrates the planned system architecture for this Master's project. It is a simple architecture that involves a microservices approach.

In this system, clients send requests to a server. Based on the nature of the request, the server retrieves the corresponding information by communicating with the appropriate microservice. Each microservice is independent, meaning that it can function even if one or more of the other microservices are not operational. When a microservice receives a request from the server, it retrieves, processes, and returns the requested data from the database.

The communication between the server and each microservice should be implemented asynchronously to ensure low latency in satisfying client requests.

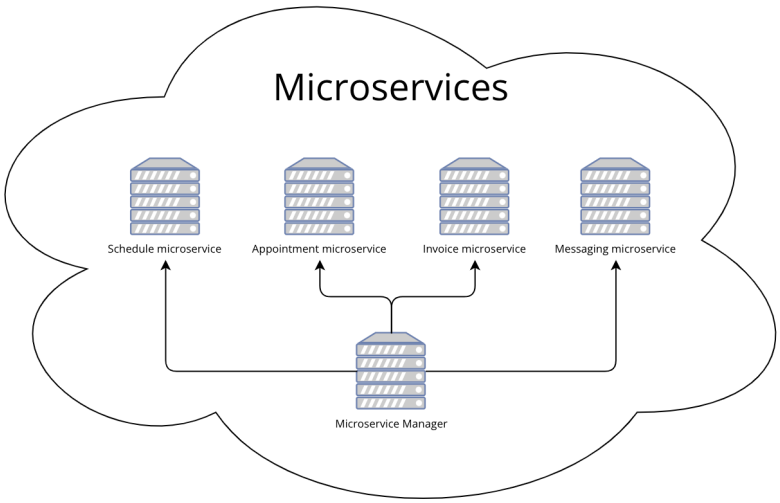



Figure 12: Microservices Architecture.

The microservice architecture, as shown in Figure 12, consists of four microservices:

1. The **schedule microservice** is responsible for managing agenda tasks such as booking and unbooking services and providing useful information for events.
2. The **appointment microservice** handles logic related to doctor appointments, including file handling, patient histories, and appointment data.
3. The **invoice microservice** is responsible for generating invoices for services provided by the clinic and may potentially handle stock management in the future.
4. The **messaging microservice** manages messaging functionality, such as sending reminders to patients regarding their bookings or advertising events happening at the moment.

To ensure scalability, a microservice manager is used to forward requests to the appropriate microservice that can handle the request.

## 3.3 MOCKUPS

To have a better understanding of the application workflow, the conception of mockups takes an important role in that aspect, as they not only show in an illustrated way the view of the expected application, but they also guide the developers on what's the wanted flow of the routine.

This type of prototyping is an effective way to demonstrate the planning of the work to be done to the interested part, allowing the stakeholders to have their opinion and formulate more precise requirements so that the margin of failure of the project lowers considerably.

The software used to create these mockups is named Figma[1], which is a website that offers a free component that enables users to create visually appealing *UI*'s in a practical and straightforward manner. The free component of Figma is not overly restrictive and is a suitable choice for planning website layouts.

For mockup conception, it was decided to present the same views analyzed in state of the art section (2), owing to the fact that they are the essential features that characterize a clinical management application. Since the application is going to be cloud, all types of devices could access the application, so it should adapt to every kind of advice possible to reach the majority of people possible. For that, web and mobile mockups were conceived, as the sizes of the screens will hugely differ, depending if the user is working on a desktop or a smartphone.

In all the mockups designed, there will be a navbar, so the user can freely navigate the page to the services he wants. For the desktop version, the navbar will be vertical, aligned to the left, but that approach would not fit the mobile version. To work around that, the navbar was placed at the bottom of the screen, mobile version only.

### 3.3.1 *Agenda Page*

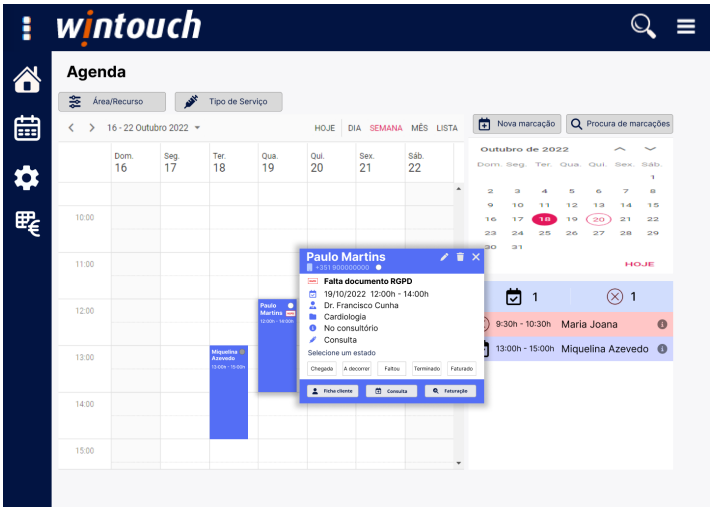



Figure 13: Agenda Page Web Version.

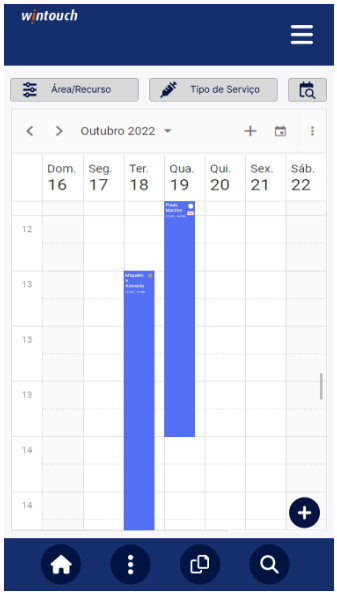



Figure 14: Agenda Page Mobile Version.

1 http://figma.com/

Regarding the agenda page, simplicity and easy access to information were the keywords to conceive this view. It consists of a calendar marking the existing bookings of the business. Each of the bookings exposes basic information, like its title and date, but the user can click on it to have a popup to expose the full information of that event, shown in Figure 13. The same functionality happens to be in the mobile version as well (Figure 14), but instead, takes the user to another view instead of appearing in a popup due to reduced screen sizing.

To easily access all the bookings for the day, there is a monthly calendar in the right position of the page, which allows the user to pick a day. After the day is picked, below the calendar shows the appointments for the day, canceled or not canceled, as well as some basic information about each one. This functionality is not present on the mobile version, as it would force some complexity on the view and that will destroy the simplicity concern. It is assumed that the users would use the mobile version of the application for basic operations.

Aside from the features mentioned, there is a possibility to filter the events viewed and to search for specific events. Those options appear above the calendar.

### 3.3.2 *Patient Record Page*

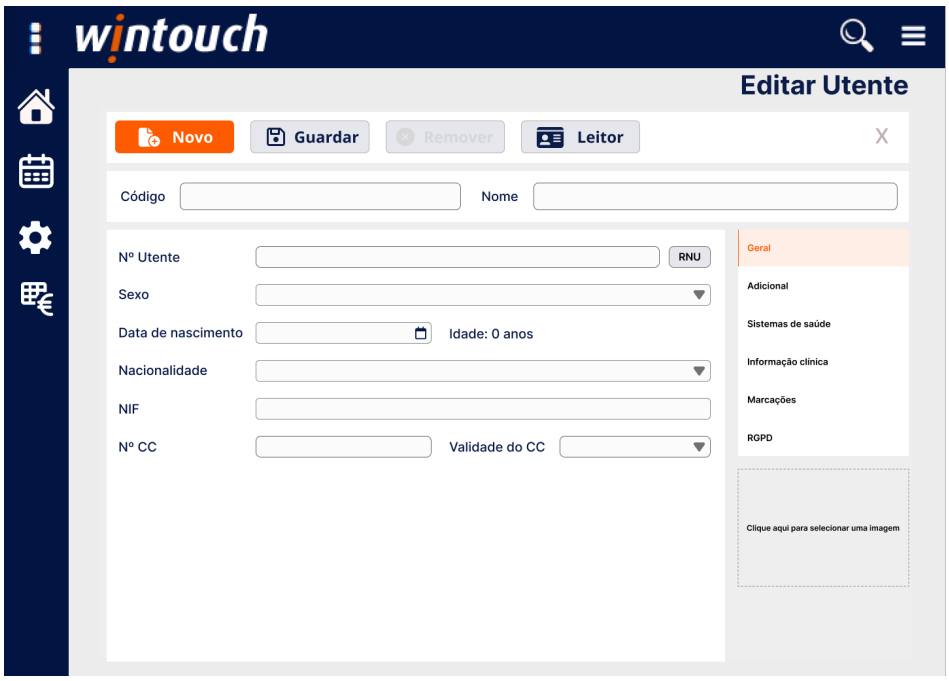



Figure 15: Patient record page web version.

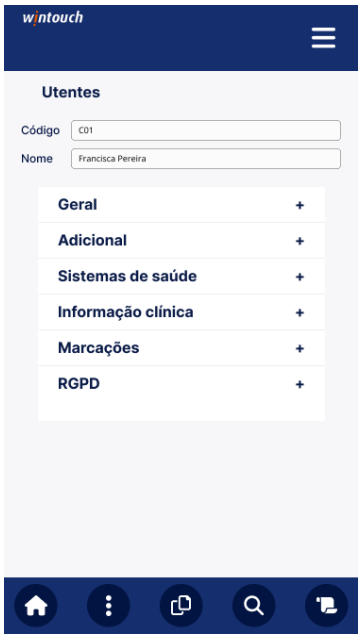



Figure 16: Patient record page mobile version.


Figure 17: Patient record page mobile version with *general* tab open.

For the patient record page, the same keywords were followed, simplicity and easy access to information. Observing the web version (Figure 15), the page consists of four important divisions:

- Actions section
- Identification section
- Data section
- Navigation section

The **actions section** contains the buttons for the interaction of the page. There, the user can create new patient records, save changes to the current one, remove them and even read the information from an identification card.

The **identification section** shows the code and name of the patient that the page refers to. It is a section that stays shown when navigating in the view, as it is crucial to know to whom the record belongs.

The **data section** is where the information will all be. It is the component that occupies most of the screen referring to its importance. It was decided to make the editable fields always editable, that is, there is no need to click an *edit* button to edit them. This approach saves a click and allows the user to see the data and manage them how he wants, making him decide if he wants to keep the changed data by clicking the *save* button.

The **Navigation section** allows the user to navigate different types of information present in the patient record. It is an efficient way to organize the individual's knowledge and display only the information the user wants to check.

Both the action section and navigation section differs depending on the device used to access the page. In the mobile version (Figures 16 and 17), the actions section translates to an additional button in the lower navbar, so the user can click on it and then has a panel to select the interaction he wants to have. regarding the navigation section, the tabs are converted to an accordion style, when each can be expanded to expose the information related to the corresponding context.

### 3.3.3 *Appointment Page*

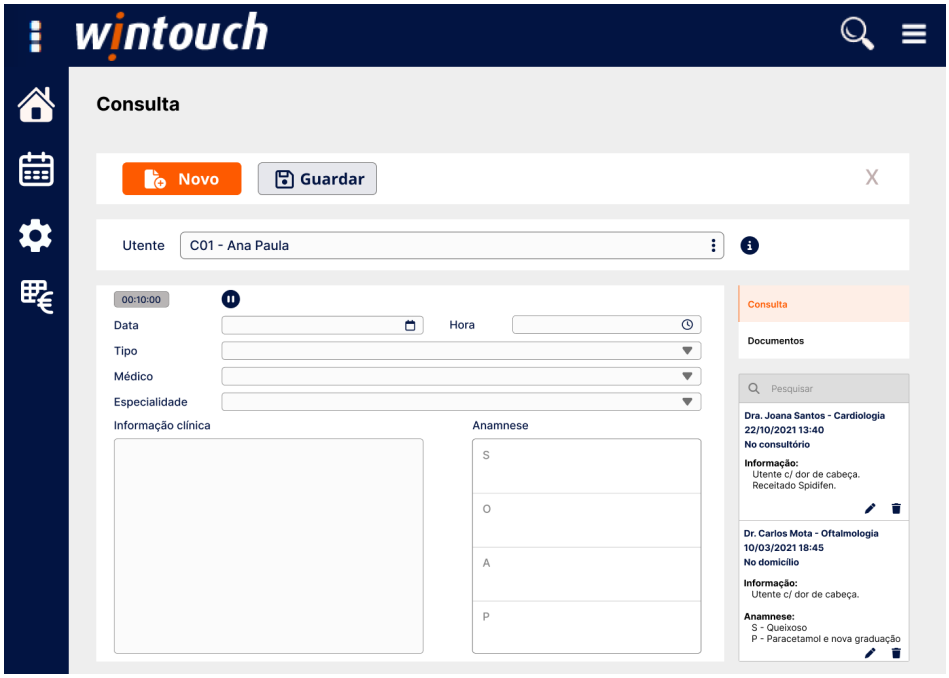



Figure 18: Appointment page web version.

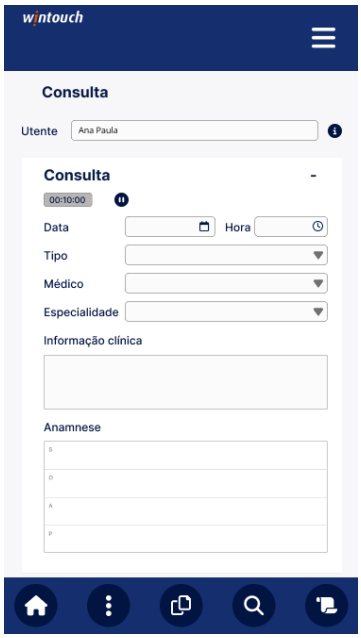



Figure 19: Appointment page mobile general version.

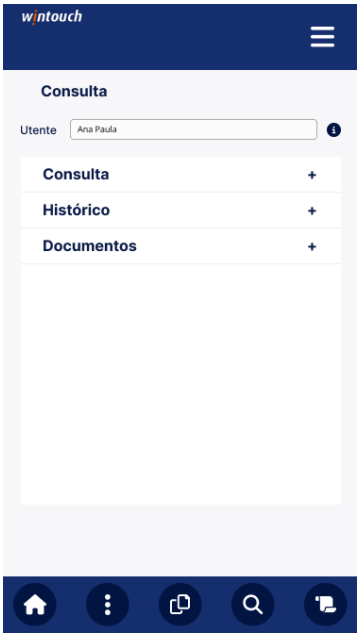



Figure 20: Appointment page mobile version.

The appointment page follows a similar layout to the patient record page, having the same four sections referenced previously. The shared layout can be a benefit as the user might be used to it and so he can instinctively discover the tools and features he wants to use without getting lost with the newer views he experiences in the application.

Analyzing the web version of the view (Figure 18) and the mobile ones (Figure 19 and Figure 20) the differences are the same as the patient record page, with the buttons position and the tab format.

## 3.4 DATA MODEL

To be able to show the data to the user, through the UI component, there is a need to have all the data well structured so it can be precise and adequate to the context.

For that, using the Entity Relationship diagram, the data model could be built, establishing how the data will be organized and stored, as well as defining the relations between each other.

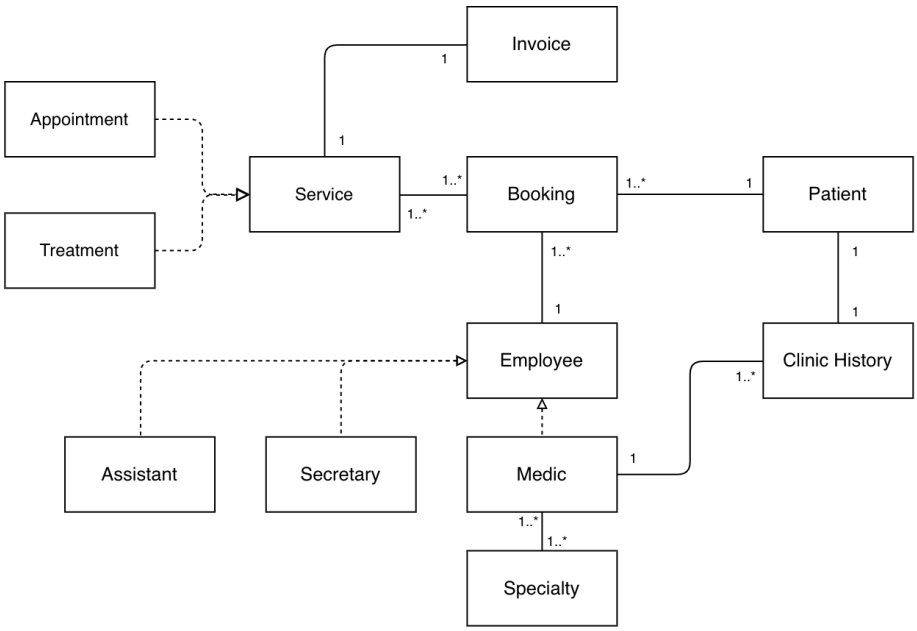



Figure 21: Project's Data Model.

Figure 21 illustrates the resulting data model that can be used as a base sketch for the system. It is a very basic data model, as there is no deep planning like attribute and type definition. The goal of this diagram is to have an overview of the composition.

As the main goal of the service is to sell services, the central entity of the data model will be the booking, as it will permit the business entity to provide the services to the respective clients, or in this case, patients. The booking entity should have information about the patient that will be attended to, the employee that will provide the services, and the services themselves.

A service can be seen as a parent class as it is a general concept that can be specialized into some other entities. An appointment or treatment can be a service, so it will be related to the service entity The core business of the clinic is the services they provide, so there is a relation between the service and the invoice entity to store billing information.

The employee, like the service entity, will be an abstract entity, as it can be specialized by the different types of staff the clinic has, like assistants, secretaries, or doctors. The medic will be a special type of entity as it will be the main resource of the system. So it has one or more specialties and can be associated with the patient's clinical history, through past services done by himself.

# 4

# TECHNOLOGY SELECTION

This chapter introduces the technologies that were used to develop the clinic management web application. An application has two main components — **backend** and **frontend**.

The **backend** is the data access layer, that is, the logic behind the process of gathering, adapting, and storing data.

The **frontend**, on the other hand, represents the *UI* component of the software. Its logic is based on the presentation of the data requested to the backend side of the application to furtherly be shown to the user.

As those two layers have different objectives and clearly distinct behaviors, there are technologies that best suit each. Consequently, the technology selections for each component will be presented in the next sections, followed by a brief description.

## 4.1 BACKEND

As for the backend, the technologies used were based on the Microsoft stack, using C# as the main development language, coupled with the .NET CORE platform, along with related libraries, and using the SQL Server *Database* (*DB*) engine.

### 4.1.1 *C#*

C#[1] is a multi-paradigm high-level development open-source language conceived by Microsoft in 2000. This language is a good option to create software components. Having its roots in C, there is little difference to navigate from C, C++, Java, and JavaScript to it.

1 https://dotnet.microsoft.com/en-us/languages/csharp

### 4.1.2 *.NET CORE*

.NET CORE[2] is a version of .NET that is an open-source computer software framework maintained by Microsoft. It is used to build a wide variety of computer software, from mobile to even gaming areas. The most important aspect that differentiates the .NET CORE version from .NET is the cross *Opearative System* (*OS*) compatibility, which will allow the developed product to adapt to and run in a wider range of environments. There is an almost symbiotic relationship between .NET CORE and C#, as there are some features in the framework only supported by the language and there are some functionalities in the language that require the .NET CORE framework to operate with. Although, this framework supports other languages besides C#, like F# and Visual Basic.

### 4.1.3 *Entity Framework Core*

*Entity Framework Core* (*EF Core*)[3] is a new version of the Entity Framework technology. It is an open-source *Object-Relation Mapping* (*ORM*) developed by Microsoft. *EF Core* allows the developers to access data from the *DB* without focusing on the underlying tables and columns where this data is stored, using objects of domain-specific classes. In the end, this will help developers to write less code and work with a higher level of abstraction when dealing with data.

### 4.1.4 *SQL Server*

SQL Server[4] is a relational database management system developed by Microsoft, so the main functionalities of this technology are storing, maintaining, and retrieving data as it is requested by software applications.

## 4.2 FRONTEND

The main framework of choice to develop the frontend layer is Angular. This framework is dependent on some other technologies that will be described afterward.

2 https://dotnet.microsoft.com/
3 https://learn.microsoft.com/en-us/ef/
4 https://www.microsoft.com/en-us/sql-server

### 4.2.1 *Angular*

Angular[5] is an open-source web development framework developed by Google and by a community of individuals and corporations. It is built in *TypeScript* and consists of a component-based framework to build scalable web applications, having popularity on *Single Page Applications* (*SPA*).

### 4.2.2 *HTML-CSS*

HTML is a markup language used to structure website content. It is used to give meaning to text, images, and other content, making it easier for web browsers to interpret and display them properly.

CSS is a stylesheet language used to style and format HTML content. Can be used to customize the appearance of website pages and make them more appealing.

These two technologies are commonly used together to make websites as HTML provides the content and structure, while CSS determines how that content is presented to the user.

### 4.2.3 *TypeScript*

TypeScript is a programming language that is very similar to JavaScript. It was developed by Microsoft and is characterized as a strongly typed language. This typing is beneficial in error prevention and discovery so that can save you time and effort by helping you find and fix problems early on in the development process.

As it is fully compatible with JavaScript, it can even be used in JavaScript projects.

5 https://angular.io/

# 5

# DEVELOPMENT - BACKEND

To commence the application's development, the most optimal approach towards resolving this issue involves dividing it into smaller sub-problems. In this particular scenario, it would be beneficial to start with the development of microservices as they not only act as dependencies for the Healthcare application but also comprise more manageable projects that facilitate a favorable starting point for development stages.

As per the preconceived plan, the application is intended to have four microservices, namely Schedule, Appointment, Invoice, and Messaging. Out of the four, the Schedule microservice holds paramount significance as it enables the business to operate and cater to its clientele in an organized manner, thereby establishing an optimal starting point for development. Subsequently, the Invoice microservice can be developed, followed by the Appointment microservice, and ultimately concluding with the Messaging microservice. However, under the time given, only the Schedule microservice could be developed, which will be described nextly.

## 5.1 DATABASE

The development of the Schedule microservice commenced with the backend aspect. Since the work was initiated from scratch, it became imperative to establish preliminary settings to construct a suitable development environment. These settings encompassed tasks such as creating database string connections, generating ports, populating variables, and executing other essential configurations.

During the initial phase, the database will be confined to a local setting. Thus, the configurations were being made while keeping this condition in mind. These configurations enable the application to function across multiple machines without requiring additional setup adjustments. The said configurations were preserved in a dedicated JSON file, which would be read by the application's backend section, and accordingly apply the data contained in the file.

Upon completion of the configuration stage, the subsequent task entails preparing the database. Utilizing the *EF Core ORM*, it becomes feasible to generate script files that facilitate

the creation of specified tables, relationships, and constraints. These script files are known as *Migrations*, and they are preserved in a designated directory.

Migrations serve as an advantageous tool for database version control. Essentially, they enable the database to adapt to modifications in requirements while preserving stored data, thereby promoting its progression. Moreover, if the need arises to revert changes, such action is also possible due to the sequential storage of script files. This feature facilitates rollback while minimizing data loss.

To initiate the preparation of the database, it is essential to define the models required for the system to adequately store the necessary data. With the assistance of *EF Core* technology, these models can be used to map the tables that will be utilized in the database and specify their configurations based on the presented conditions. The functioning of this Object-Relational Mapping (ORM) can be observed in Figure 30. The ORM maps the models created as objects in memory to their corresponding relational database tables using its internal mapping logic.

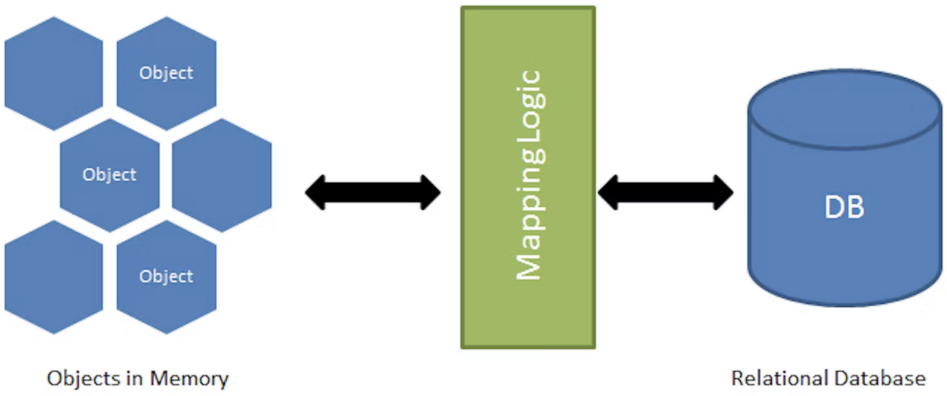



Figure 22: ORM workflow

## 5.2 SCHEDULE

### 5.2.1 *Database models*

The schedule database consists of x models designed to store information pertaining to the microservice. Each table, except for those created to facilitate N-to-N relationships, has an identifier enabling the selection of specific rows. These bridge tables can be identified using the IDs of the tables to be linked, and their combination forms a composite key.

To establish 1-to-N and 1-to-1 relationships, a foreign key is allocated to the dependent entity in the former case, and to either entity in the latter.

Regarding the creation of the model, a class is instantiated to define the architecture for each table that requires mapping. This particular class contains properties that represent all the attributes of the table, which are essentially its columns.

Each of these properties is associated with a type, which is subsequently processed by *EF Core* in order to map the corresponding database type based on the chosen technology. For instance, if a property has a class of string and the database technology being utilized is SQL Server, then the corresponding attribute will be of type VARCHAR. The default mappings can be found in the official documentation[1].

In order to illustrate the concept of these models comprehensively, three models shall be described, as they encompass the majority of the particular aspects needed to address intricate cases. Specifically, the models to be examined are the appointment model, resource model, and resource_speciality model. The remaining ones shall adhere to the same principles utilized by the exemplars mentioned above.

*Appointment*

The appointment model is the most important and influential model of the schedule microservice given its nature. It is the model that unites almost all the other models and connects them together. With that in mind, it will be constituted by many relationships. Note that this appointment model is a model that constitutes the Schedule microservice, not related to the Appointment microservice.

| Parameter | Parameter Function | .NET Data Type | SQL Server Mapped Type |
|---|---|---|---|
| Id | Primary Key | GUID | uniqueidentifier |
| StartDate | Data Field | DateTimeOffset | datetime |
| EndDate | Data Field | DateTimeOffset | datetime |
| Subject | Data Field | string | nvarchar(max) |
| Notes | Data Field | string | nvarchar(max) |
| DeletedDate | Data Field | DateTimeOffset | datetime |
| Type | Data Field | Enum | int |
| Assets | Navigation Field | ICollection | - |
| EntityId | Foreign Key | GUID | uniqueidentifier |

Table 1: Appointment model composition with its respective data type mappings

The appointment model is composed of several fields, some of which are described in Table 1. These fields can be categorized into four types based on their function:

- **Primary Key**: This field is used for row identification queries.

1 Entity Framework Parameter Data Type Mapping

- **Data Field**: This field is used to store data related to the appointment model itself.
- **Navigation Field**: This field is specific to the *EF Core* and does not have any effect on the database. It is used to access related data with more ease and simplicity.
- **Foreign Key**: This field is used to establish a relationship between rows.

In addition to the four types of fields mentioned above, navigation fields can also have a type that corresponds to the related model. If the navigation field represents the singular side of the relationship, then its type will be the related model type. If it represents the plural side of the relationship, then its type will be a list containing the related rows.

Overall, understanding the different types of fields in the appointment model and how they are used is important for developing and maintaining the model effectively.

*Resource*

As the resource model, it is one of the models that have an N-to-N relationship. This is a special relationship as it needs to have an additional model to support the relationship, behaving as a bridge table. With the *EF Core* technology it is possible to simplify this logic and make the relationship more direct. However, it is possible to also access the bridge table in case there are data fields related to the relationships, having two separate navigation fields for the same relationship. These fields that characterize the table can be observed in Table 2.

| Parameter | Parameter Function | .NET Data Type | SQL Map Type |
|---|---|---|---|
| Id | Primary Key | GUID | uniqueidentifier |
| Name | Data Field | string | nvarchar(max) |
| Specialities | Navigation Field | ICollection | - |
| Resources_Specialities | Navigation Field | Resource_Speciality | - |

Table 2: Resource model composition with its respective data type mappings

*Resource_Speciality*

The "resource_speciality" table serves as a bridge table to facilitate the N-to-N relationship between the "Resource" table and the "Specialities" table, as previously mentioned. A distinctive feature of this table is its composite key, which comprises two foreign keys from the table and serves as a primary key for identification purposes. This composite key ensures that each entry in the table is unique and can be accessed efficiently. The properties of the table are illustrated on Table 3.

| Parameter | Parameter Function | .NET Data Type | SQL Server Mapped Type |
|---|---|---|---|
| ResourceId | Foreign Key/Primary Key | GUID | uniqueidentifier |
| SpecialityId | Foreign Key/Primary Key | GUID | uniqueidentifier |

Table 3: Resource_Speciality model composition with its respective data type mappings

### 5.2.2 *Endpoints*

The primary role of the backend side of an application is to handle and store data. This logic is crucial since it provides the frontend side of the application with the necessary data to present to the user. In order for the frontend to obtain the data it requires, it must establish communication with the backend using a suitable method. The most commonly used method for establishing this communication, which was chosen to implement, is through REST API. This well-known architectural style employs HTTP requests to access and utilize data (Surwase, 2016). Different types of requests are employed by this style to request various data-handling functions, which are denoted in the request header. The four most widely used requests are *POST*, *GET*, *PUT*, and *DELETE*, which correspond to the primary data operations (CRUD), namely, **Create**, **Read**, **Update**, and **Delete**, respectively.

The request types mentioned in the context of web development are typically made against a target resource, which is identified by the input URL provided in the request. The URL corresponds to a specific backend endpoint, with each endpoint mapping to a backend controller that is responsible for handling the request and generating an appropriate response.

An endpoint is a specific URL that maps to a particular resource or functionality within an application. It provides a way to access the application's API, and it is associated with a backend controller, which is a logical unit of code responsible for processing requests and responses and providing a level of abstraction between the routing mechanism and the application logic.

Once a request is received by the controller, it redirects the request to the corresponding service that will handle the data received with the proper context needed. These services are designed to process the request, execute the required context validations, and execute the necessary operations to generate a response that meets the requestor's needs. The service layer typically acts as an intermediary between the controller and the data access layer, ensuring that the necessary data is retrieved or updated and that any required business logic is executed before generating the final response.

*Controllers*

It was decided to create a controller for each model created for the database to simplify data management and collection. For instance, the corresponding URL for the Resource model would be "https://(SERVER_ADDRESS)/resource". Therefore, if the frontend wanted to retrieve all the resources, it would create a GET request for the URL. The other request types would be made against the same URL by providing the respective parameters in the request body or appending them to the URL, depending on the request type.

The proposed approach enables the frontend to establish communication with the backend and fulfill its data requirements. However, when it comes to data querying, there is a need to retrieve information that satisfies specific criteria. The basic REST API architecture provides only one criterion to query data, which is the resource identifier (id), specified in the URL section as shown: "https://(SERVER_ADDRESS)/resource/(id)". To address this limitation, OData[2] was incorporated into the communication layer to facilitate data querying. OData is a protocol that defines the best practices for constructing and consuming RESTful APIs.

By utilizing OData, the frontend can retrieve all data for a specific resource and also query data by fields, sort the response data, and limit the amount of data retrieved, providing additional functionality regarding data collection. This can be achieved through sending a GET request to a specific URL that incorporates the OData query parameters.

For instance, if the frontend wanted to fetch all resources that have the name "John", a GET request should be sent to the following URL:

"https://(SERVER_ADDRESS)/resource?filter=Name eq 'John'".

In this case, the "filter" parameter is used to specify the condition of the query, which is to retrieve all resources that have a name equal to "John". Other OData query parameters, such as "select", "orderby", and "top", can also be used to further customize the query and retrieve the desired data.

2 https://www.odata.org/

# 6

# DEVELOPMENT - FRONTEND

Upon completing a rudimentary functioning backend, the decision was to commence the development of the frontend to level the advancement of both layers of the application.

The term "rudimentary functioning backend" refers to the accomplishment of a fundamental API, featuring various endpoints that map to their respective controllers responsible for processing incoming data through their own logic. These endpoints embody the entities that form the application. Utilizing the data furnished by the backend's API makes it feasible to construct the frontend using the same data instead of resorting to simulated static data to populate the product.

The development of the frontend began along with the backend, with a focus on the microservices, explicitly beginning with the schedule microservice, and proceeding in alignment with its progression.

## 6.1 SCHEDULE MICROSERVICE

The frontend development began by constructing the views for the configurations of the application, as they are the most independent components of the system. Three pages were required to fulfill the configurable necessities of the schedule, namely the general schedule settings, notification settings, and booking settings.

With the development of the settings done, the next step was to define a view for resource management, that is, the client page, resource page, service type page, among others. Those pages allow the user to configure which entities will be available to be operated in the schedule.

Upon the conclusion of the preceding task, the implementation of the main schedule view was commenced. This view encompasses an agenda display containing all the created bookings, based on the selected view option. Furthermore, a dedicated section will be incorporated into the platform, enabling users to access bookings made on a specific date. It is essential to acknowledge that this particular feature will exclusively be accessible on the desktop version, owing to screen dimension constraints.

Subsequent development endeavors predominantly concentrated on enhancing the functionality of the aforementioned main view. These improvements encompass areas such as booking management, view customization, and the introduction of additional features.

### 6.1.1 *General schedule settings*

The system provides the user with the capability to configure general settings pertaining to the scheduler through a dedicated view. This view, as illustrated in Figure 23, enables the user to select the business's working hours and the working hours of each resource associated with the business, observed in Figure 24, which can be accessed clicking the button on the designated section. Furthermore, it offers a setting to specify the timescale employed in the scheduler.

Ensuring that the scheduler is highly user-friendly is of utmost importance. To attain this goal, the scheduler should not be excessively rigid, but instead, allow users to make their desired bookings. Rather than restricting time intervals for reservations, the scheduler should provide proactive notifications to users if their chosen time interval falls outside of working hours or if the requested resource is unavailable during that time.

To facilitate this, the scheduler can be configured with options that provide users with clear indications of the working hours applicable to their business. This will enable users to choose appropriate booking times and avoid resource unavailability.

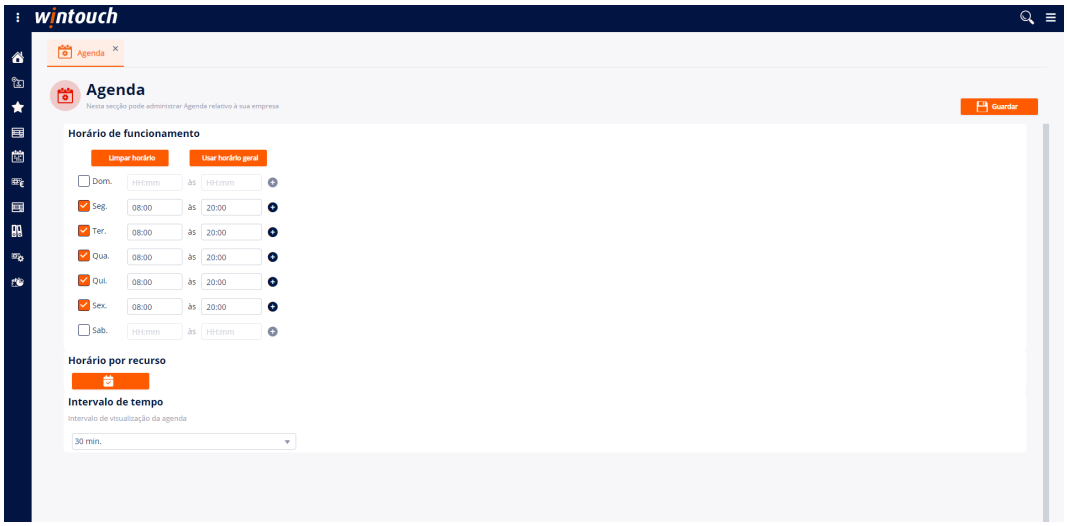



Figure 23: General schedule configuration view

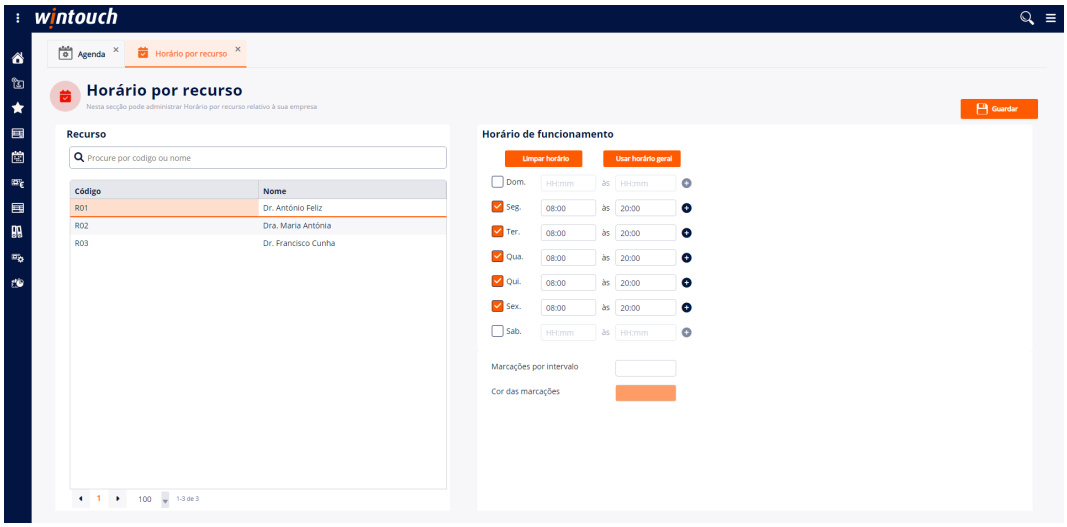



Figure 24: General schedule by resource configuration view

### 6.1.2 *Notification settings*

The view dedicated to notification settings, illustrated in Figure 25, offers users an array of customizable options regarding notifications related to bookings. The view comprises three message templates that can be personalized for booking creation, booking cancellation, and booking reminders. These templates can be tailored to reflect the organization's branding and tone, as well as the nature of the booking.

Apart from message personalization, the notification settings view enables users to specify the time interval between booking reminders. This feature ensures that clients receive timely notifications before their booking without being inundated with excessive notifications. Users have the flexibility to select from a range of time intervals, such as 48 hours, 24 hours, or 12 hours before the scheduled booking.

The customization of notification settings empowers the application to improve client engagement and diminish missed bookings. The capability to personalize message templates ensures that notifications are consistent with the organization's brand and communication style, while the reminder interval feature facilitates effective and timely communication with clients.

To facilitate the process of customizing templates, a dedicated section has been implemented within the application. This section provides users with a set of buttons that enable the insertion of tags corresponding to the relevant context information into the focused input section. For instance, if the text contains the "«d»" tag, the notification message sent to the client will replace this tag with the date of the relevant booking, enhancing user experience and functionality.

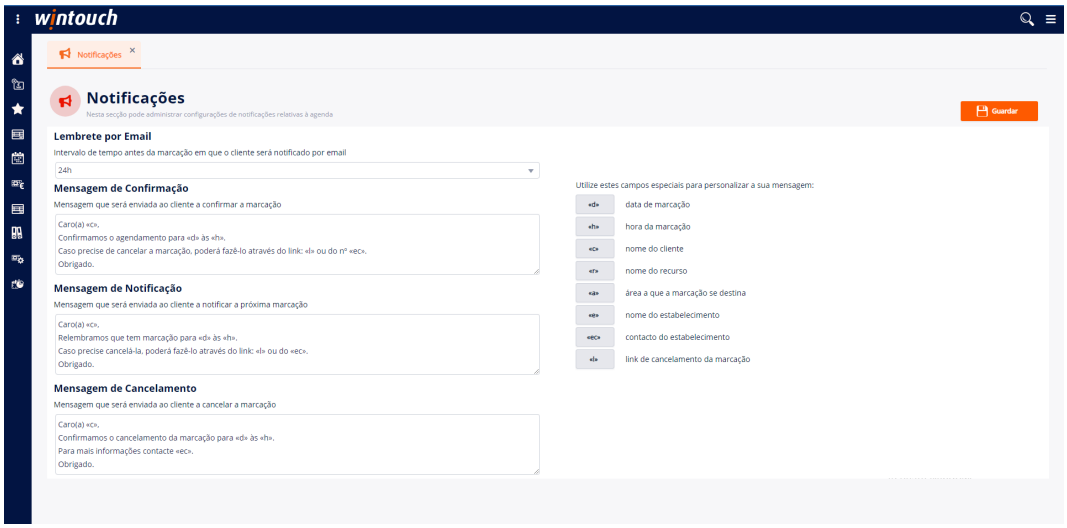



Figure 25: Notification configuration view

### 6.1.3 *Booking settings*

The booking settings view is a crucial component of the application, offering users a range of options to customize booking-related settings. The view comprises three distinct sections, each providing users with a unique set of features and tools to optimize booking management.

The first section is dedicated to synchronization with Google Calendar, allowing users to seamlessly integrate booking scheduling with their existing Google Calendar. This feature enables users to avoid scheduling conflicts and maintain consistency across their scheduling platforms.

The second section of the booking settings view focuses on booking states. In this section, users can configure the label name and color corresponding to each booking state. This feature offers users the flexibility to create a customized set of booking states that align with their unique business requirements. Additionally, this feature helps users to quickly and easily identify bookings in different states, enabling them to manage bookings more efficiently.

The final section of the booking settings view is dedicated to user authentication. In this section, users can choose whether to validate the authenticity of the user before creating the booking. This feature adds an additional layer of security to the booking scheduling process, reducing the likelihood of fraudulent bookings.

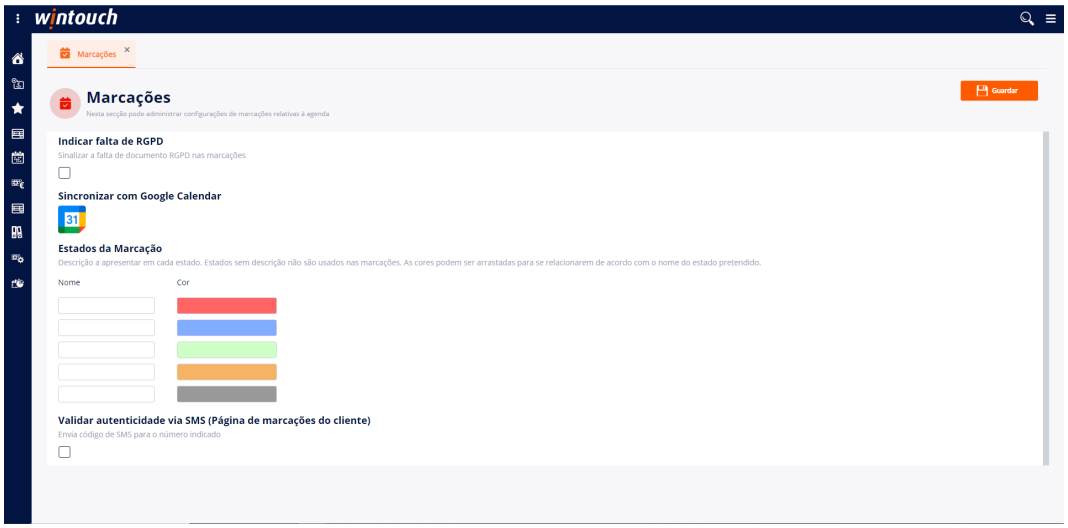



Figure 26: Booking configuration view

### 6.1.4 *Schedule, Version 1*

The schedule main view, illustrated in Figure 27, is a user interface designed to manage bookings and organize daily schedules efficiently. It consists of two main components: the Schedule Component and the Day Bookings Component.

On the left side of the view, the Schedule Component allows users to manage their bookings effectively. It provides a user-friendly interface to create, edit, and delete bookings, as observed in Figure 28. Users can input various details for each booking, such as title, time, contacts, and additional notes. The Schedule Component offers a seamless experience for scheduling and organizing bookings, ensuring users can easily navigate and make changes as needed.

On the right side of the view, the Day Bookings Component provides a visual representation of the days in a month. Users can easily navigate through different months and select a specific day by clicking on the corresponding date. Below the calendar, the bookings for the selected day are displayed, labeled by their respective types.

However, due to the increasing price of the existing Schedule Component, a thorough search for suitable third-party components was conducted. Despite the effort, none of the available options aligned with the specific interests and requirements of the project. The decision was made not to purchase any of the existing components.

As a result, the project team decided to develop a custom Schedule Component from scratch. The development process involved in-depth research, planning, and design to create a solution tailored to the project's needs. Detailed information about the new custom component and its development process can be found in the appendix section of the documentation. This approach ensures that the schedule management feature will

be specifically crafted to meet the project's interests while maintaining a high level of functionality and usability.

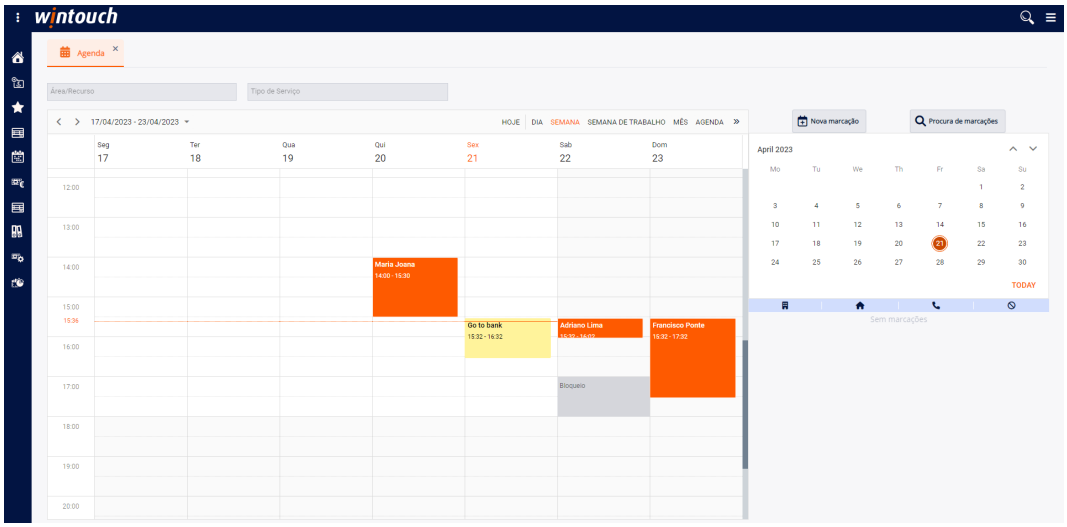



Figure 27: Schedule view

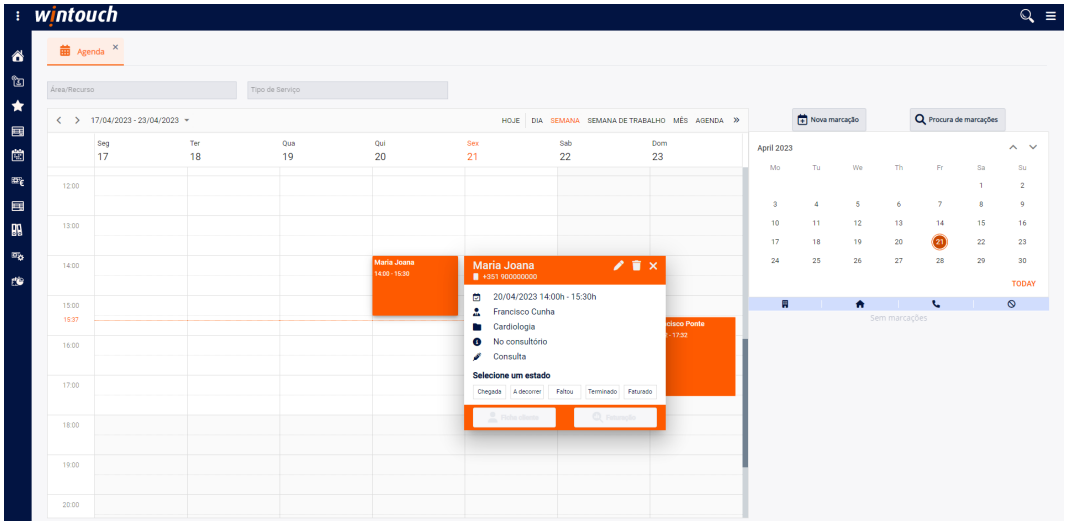



Figure 28: Booking quick information popup view

### 6.1.5 *Schedule, Version 2*

The development of a brand new, from-scratch native schedule component offers a promising solution to the difficulties previously mentioned. This comprehensive tool addresses the challenges faced in managing bookings and organizing schedules effectively. With its user-friendly interface and a wide range of features, it aims to streamline the scheduling process and enhance productivity. The design of this component draws heavily from its predecessor,

exhibiting noticeable similarities. Figure 29 showcases the updated version of the main page, illustrating its enhancements.

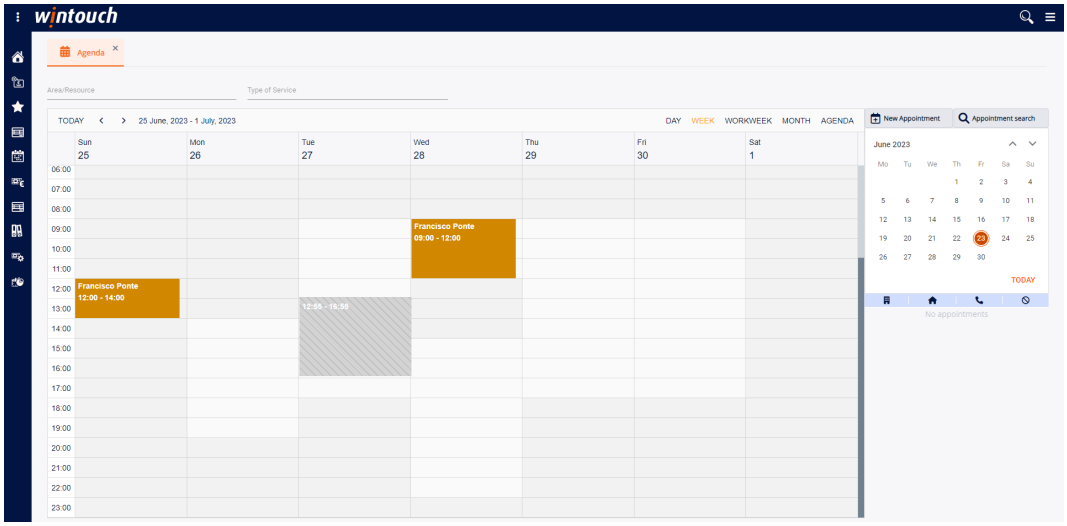



Figure 29: Schedule view version 2

The native schedule component includes essential features for efficient booking handling. Users can easily create, edit, and delete bookings with just a few clicks. It provides multiple views, such as day, multi-day, and month, enabling seamless navigation through different timeframes. This flexibility empowers users to gain a clear understanding of their upcoming commitments.

An intuitive drag-and-drop functionality sets the native schedule component apart. Users can effortlessly move bookings to different time slots or days by simply clicking and dragging them on the calendar. Additionally, the component supports booking resizing, allowing users to adjust their duration according to changing circumstances. This feature proves particularly useful when bookings need to be extended or shortened without recreating them entirely.

The native schedule component also caters to complex scheduling scenarios with advanced features for viewing multiple bookings. It intelligently displays overlapping bookings, enabling users to easily discern the details of each booking and plan accordingly.

For detailed insights into the implementation of the component, please consult Appendix A. It provides a comprehensive breakdown of the development processes involved in creating this robust scheduling solution.

### 6.1.6 *Booking editor*

The booking editor is a highly functional interface designed to efficiently schedule bookings, blocks, and notes. The editor is composed of a form that is divided into three sections:

booking information, resource information, and client information. This organized approach ensures that all relevant information is captured in a user-friendly manner.

The first section of the form is dedicated to booking information and includes fields for date, services, and recurrence. This section enables users to specify the details of the booking, such as the date and time, the type of service required, and the frequency of the booking if it is to be recurring.

The second section of the form is dedicated to resource information and includes fields for the name and specialty of the resource being scheduled. This section enables users to specify the specific resource required for the booking, such as a particular doctor or technician, and their area of expertise.

The third and final section of the form is dedicated to client information and includes fields for the client's name, contact information, and other relevant details. This section enables users to capture important client information, such as contact information and any special requirements or requests.

In addition to these functional sections, the booking editor also includes action buttons in the header for saving, creating a new booking, duplicating an existing booking, and removing a booking. These buttons streamline the scheduling process and provide users with the ability to manage bookings with ease.

The booking editor also provides users with the ability to create bookings, blocks, or notes by choosing from a radio button. This feature enables users to quickly and easily schedule bookings or create blocks of time for resources that are not available for bookings.

In summary, the booking editor is a highly functional and user-friendly interface that enables users to efficiently schedule bookings, blocks, and notes. Its organized form and action buttons, coupled with the ability to create bookings, blocks, or notes, make it an invaluable tool for any scheduling needs.

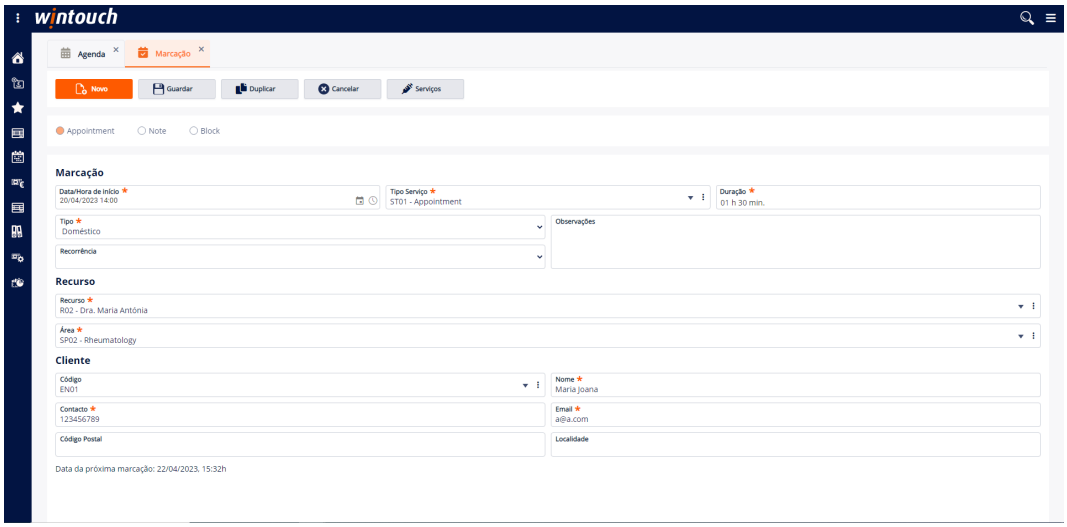



Figure 30: Booking editor view

# 7

# PROJECT DRAWBACKS AND OBSTACLES

This chapter focuses on the obstacles that significantly delayed the development of a web clinic management application. The project aimed to create an efficient system for managing clinics through a web-based application. Several challenges impeded progress, including evolving requirements, undetected base application problems, team collaboration and meetings, and unexpected obstacles such as procurement delays. This chapter provides an analysis of these obstacles and discusses the measures taken to overcome them.

1. **Requirement Changing**: One of the primary challenges that affected the project timeline was the continuous changes in requirements. As the project progressed, stakeholders provided new insights and revised their expectations, necessitating adjustments to the application's functionalities. These changes demanded additional time for analysis, design modifications, and code implementation, causing delays in the development process.

2. **Undetected Base Application Problems**: During the testing phase, unforeseen issues within the base application surfaced, which had not been identified earlier. These problems included software bugs and compatibility issues. Addressing these issues required extensive investigation, debugging, and modifications to the existing codebase. Consequently, valuable time was spent rectifying these problems, impacting the overall progress of the project.

3. **Team Collaboration and Meetings**: Effective team collaboration and regular meetings were vital for the success of the web clinic management application. However, these collaborative processes presented their own set of challenges. Coordinating schedules, ensuring efficient communication among team members, and aligning efforts towards a common goal proved to be time-consuming. Frequent discussions and decision-making further added to the project timeline.

4. **Procurement Delays**: An unplanned obstacle that resulted in delays was the procurement of a specific frontend component required for the application. Unfortunately, when the decision to purchase the component was made, the market conditions led to

a significantly higher price compared to the time of the decision. Consequently, the high price of the desired component led to the alternative approach of developing it internally. This unforeseen delay had a direct impact on the overall timeline and completion of the application.

The development of the web clinic management application encountered several obstacles that significantly delayed its progress. The challenges, including evolving requirements, undetected base application problems, team collaboration and meetings, and unexpected procurement delays, had a cumulative effect on the project timeline. However, by addressing these obstacles proactively and implementing appropriate mitigation strategies, such as thorough requirement analysis, rigorous testing, efficient team communication, and contingency planning, it was possible to navigate these hurdles and successfully complete part of the application.

Despite the setbacks, these obstacles provided valuable learning experiences. The ability to adapt to evolving requirements, troubleshoot complex issues, collaborate effectively, and manage unexpected challenges are essential skills developed through this project. By reflecting on these obstacles and the strategies used to overcome them, valuable insights can be gained to enhance future project management and development endeavors.

# 8

# CONCLUSION

The development of a web clinic application has been an ambitious project that aimed to address the complex challenges in the healthcare industry. Although the full realization of the application within the timeframe of this thesis was not achieved, significant progress has been made, particularly in the completion of the scheduling microservice.

The scheduling microservice, as one of the key components of the web clinic application, has been successfully designed and implemented. This microservice aids healthcare professionals with an automated system to manage their schedules, offering flexibility and robustness regarding booking management. It serves as a crucial foundation for the overall functionality of the web clinic application.

The completion of the scheduling microservice stands as a noteworthy accomplishment within the ambit of this Master's work. It marks the successful application of contemporary development methodologies and technologies to address a specific facet of the web clinic application. Furthermore, meticulous planning has been undertaken to ensure this developmental endeavor's seamless continuation, encompassing the core application's remaining components.

Despite natural limitations such as team collaboration, requirement changing, and development issues, this Master's work lays a solid foundation for future research and development in web clinic applications. Completing the scheduling microservice provides valuable insights and lessons learned, contributing to the body of knowledge in healthcare technology.

Following the successful implementation of the schedule microservice, it is imperative to persevere with the advancement of the application beyond the scope of this thesis. The subsequent stages of development should encompass the remaining microservices that constitute the principal application, namely the Appointment, Invoice, and Messaging microservices. This concerted effort is crucial in obtaining a complete and user-friendly healthcare application that facilitates operations within the healthcare industry.

In conclusion, the development of a web clinic application presented an ambitious project within the timeframe of this thesis. Although the project's full completion was not realized, the successful development of the scheduling microservice marks a significant achievement. The learning and the groundwork laid through this research provide a strong basis for future

advancements in the field of web clinic applications, with the ultimate goal of improving patient care and efficiency in the healthcare industry.

# BIBLIOGRAPHY


Iman Almomani and Ahlam AlSarheed. Enhancing outpatient clinics management software by reducing patients' waiting time. *Journal of Infection and Public Health*, 9(6):734–743, 2016. ISSN 1876-0341. doi: https://doi.org/10.1016/j.jiph.2016.09.005. URL https://www.sciencedirect.com/science/article/pii/S1876034116301447. Emerging Trends and Technologies in Healthcare in conjunction to the 1st International Saudi Health Informatics conference held in Riyadh, Kingdom of Saudi Arabia 12-14 April 2016.

Ron Borzekowski. Measuring the cost impact of hospital information systems: 1987–1994. *Journal of Health Economics*, 28(5):938–949, 2009. ISSN 0167-6296. doi: https://doi.org/10.1016/j.jhealeco.2009.06.004. URL https://www.sciencedirect.com/science/article/pii/S0167629609000617.

James Buergermelster and Darrell Loenen van. Computer hardware and software for controlling restaurant operations. *Hospitality Research Journal*, 14(2):35–46, 1990.

Stephen Lane, Paidi O'Raghallaigh, and David Sammon. Requirements gathering: the journey. *Journal of Decision Systems*, 25(sup1):302–312, 2016. doi: 10.1080/12460125.2016.1187390. URL https://doi.org/10.1080/12460125.2016.1187390.

Sourour Maalem and Nacereddine Zarour. Challenge of validation in requirements engineering. *Journal of Innovation in Digital Ecosystems*, 3(1):15–21, 2016. ISSN 2352-6645. doi: https://doi.org/10.1016/j.jides.2016.05.001. URL https://www.sciencedirect.com/science/article/pii/S2352664516300025. Special issue on Pattern Analysis and Intelligent Systems – With revised selected papers of the PAIS conference.

Pouriya Parsanezhad, Väino Tarandi, and Ragnar Lund. Formalized requirements management in the briefing and design phase, a pivotal review of literature. *Journal of Information Technology in Construction (ITcon)*, 21:272–291, 2016.

G.R. Sridhar, Allam Appa Rao, M.V. Muraleedharan, R.V. Jaya Kumar, and Venkat Yarabati. Electronic medical records and hospital management systems for management of diabetes. *Diabetes Metabolic Syndrome: Clinical Research Reviews*, 3(1):55–59, 2009. ISSN 1871-4021. doi: https://doi.org/10.1016/j.dsx.2008.10.008. URL https://www.sciencedirect.com/science/article/pii/S1871402108001124.

Vijay Surwase. Rest api modeling languages-a developer's perspective. *Int. J. Sci. Technol. Eng*, 2(10):634–637, 2016.

# A

# SCHEDULE COMPONENT

This appendix provides comprehensive details regarding the implementation of the native schedule component utilized in the scheduling microservice frontend layer. It aims to elucidate the rendering methods employed to achieve the desired features that ensure a superior user experience and fluidity. By providing these implementation details in the appendix, readers will understand the strategies employed to develop the native schedule component and its associated frontend layer.

## A.1 SCHEDULE VIEWS

The objective of the schedule component is to streamline the process of creating, editing, and removing bookings. Its primary purpose is to assist users in determining the level of busyness for a specific period of time, enabling them to make informed decisions to enhance the quality of service and productivity within their business.

To achieve this objective, the schedule component provides various options for displaying the existing bookings, facilitating the effective communication of this information to the user. Several views have been developed to address this challenge:

- **Day View**: Presents the schedule for a single day, allowing users to focus on the bookings and activities occurring within that specific day.
- **Week View**: Provides a comprehensive overview of the schedule for an entire week, enabling users to plan and manage bookings across multiple days.
- **WorkWeek View**: Similar to the week view, but excludes non-working days defined by the business, allowing users to concentrate on workdays and allocate bookings accordingly.
- **Month View**: Offers a monthly calendar representation of the schedule, providing an overview of bookings and their distribution throughout the month.

- **Agenda View**: Presents a list-based view of bookings, giving users a chronological order of scheduled events, regardless of the specific date.

By offering these diverse views, the schedule component aims to ensure that users can easily access the desired information in a simplified and efficient manner.

### A.1.1 *Single and Multi-Day Views*

The day, week, and workweek views are essentially identical, with the only variation being the number of days displayed. These views provide the same range of functionalities and share a common layout.

The view comprises a top header bar that displays the days being viewed in columns. Additionally, there is a left column consisting of blocks representing sequential hours. The main component is a cell grid, which is composed of cells representing specific positions in terms of both day and hour. A visual representation of this layout can be seen in Figures 31, 32, 33.

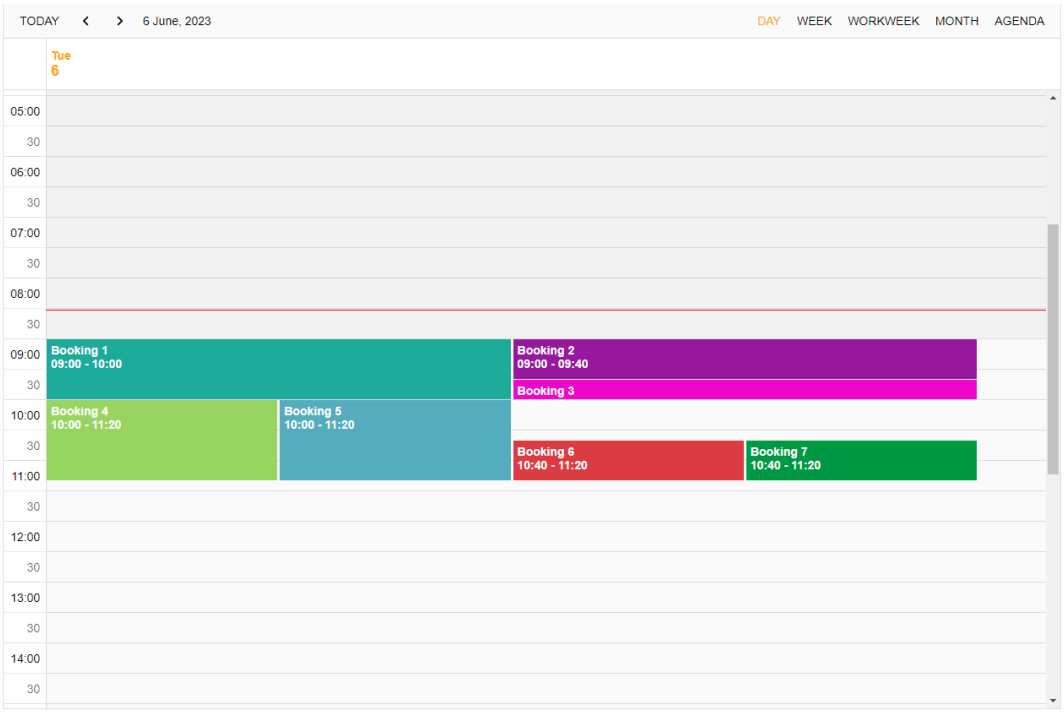



Figure 31: Day View

An indicator now mark was also added to the component. It consists on a red marker that delineates what's the current day and the current time. If the first column does not represent the current day, a dashed line is created in the previous columns to help the user check the current time.

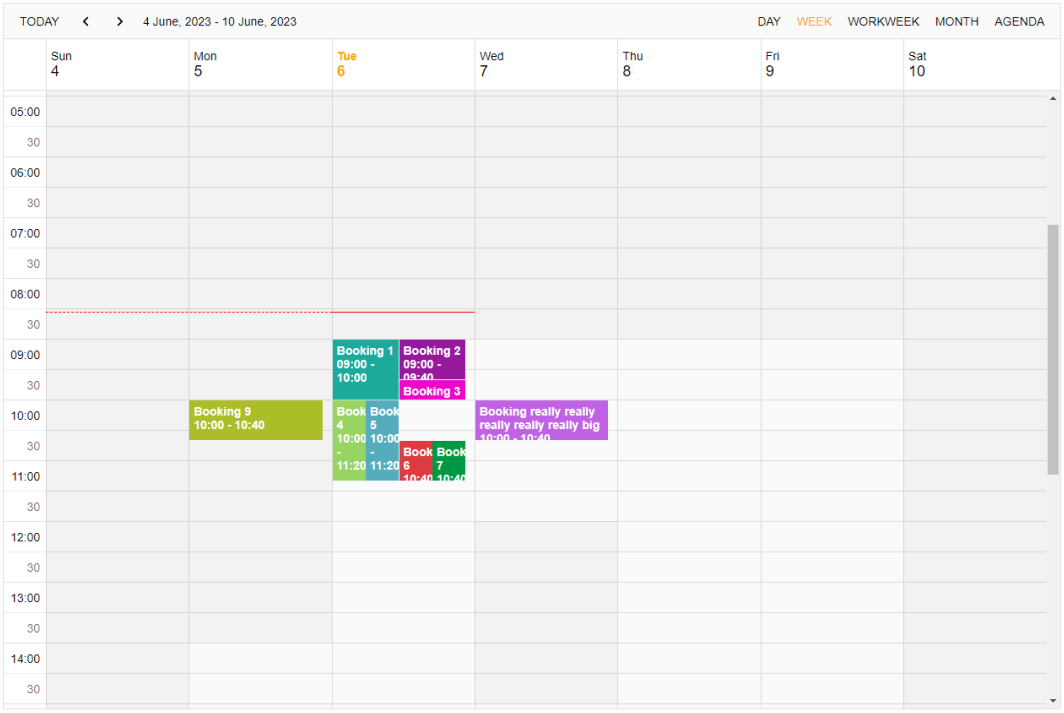



Figure 32: Week View

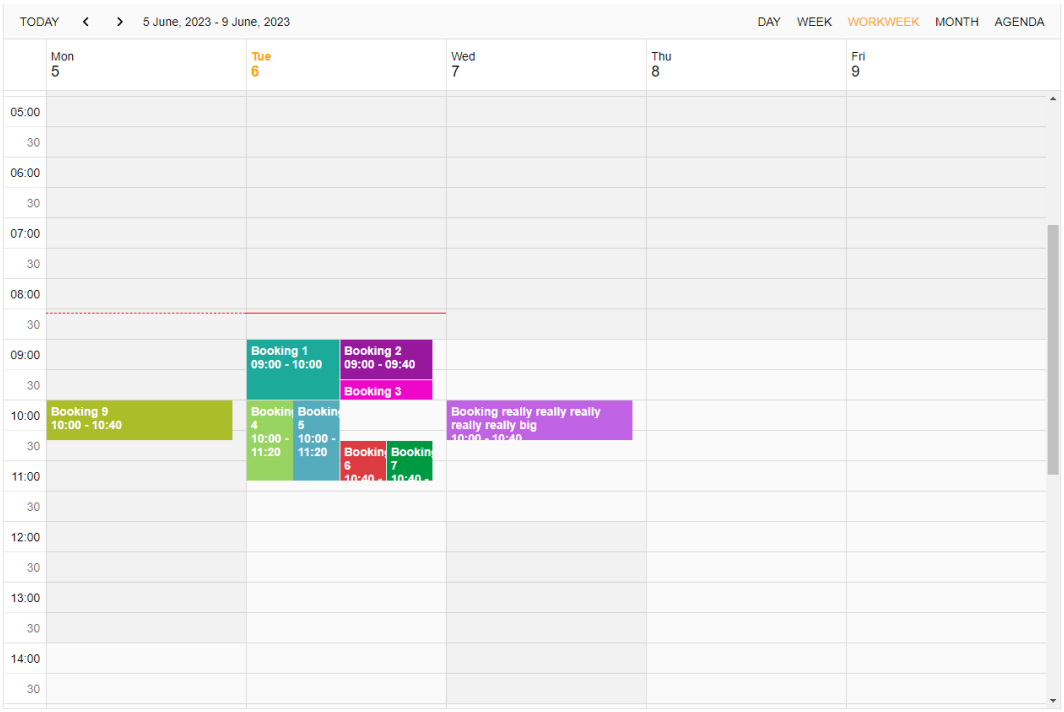



Figure 33: Workweek View

The component includes a feature called the current time indicator mark. This mark, represented by a red marker, serves to highlight the current day and time. If the first column does not represent the current day, a dashed line is displayed in the preceding columns to assist the user in identifying the current time.

Regarding the rendering of bookings, the positioning of each booking within the rectangular grid is determined using geometric calculations. The position and dimensions of each booking are calculated based on its length and the days to which it belongs.

To improve *User Experience* (*UX*), the component allows the user to click in the events to show a quick information popup (Figure 34), shared by all the available views. This element provides more information about the event, along with some interactivity, like booking editing and deletion.

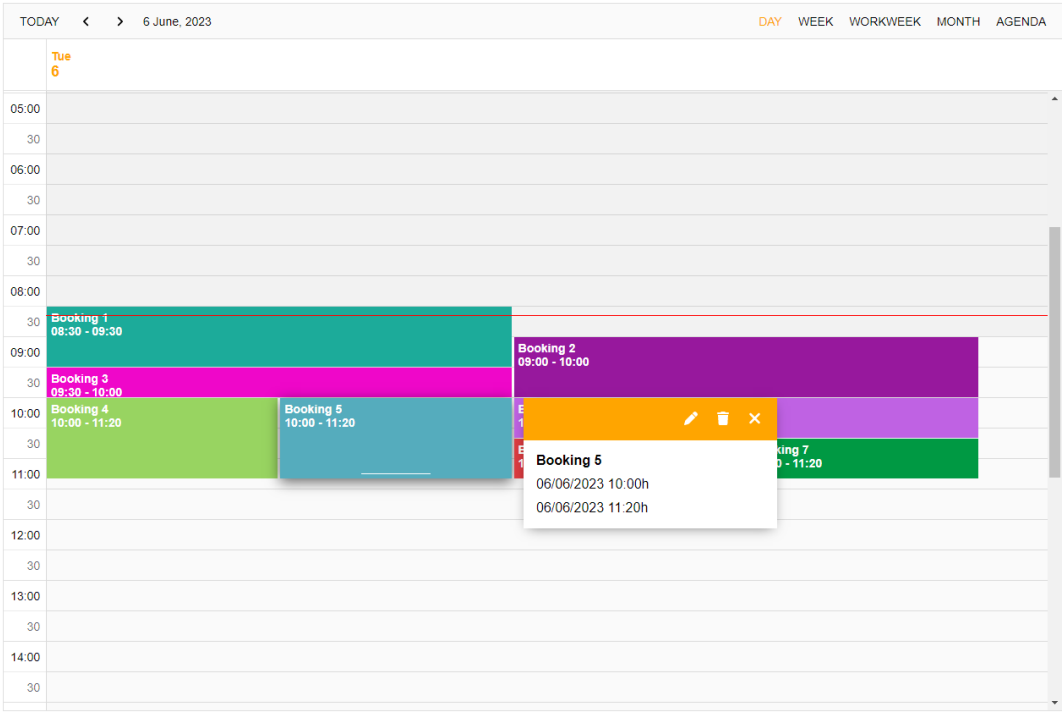
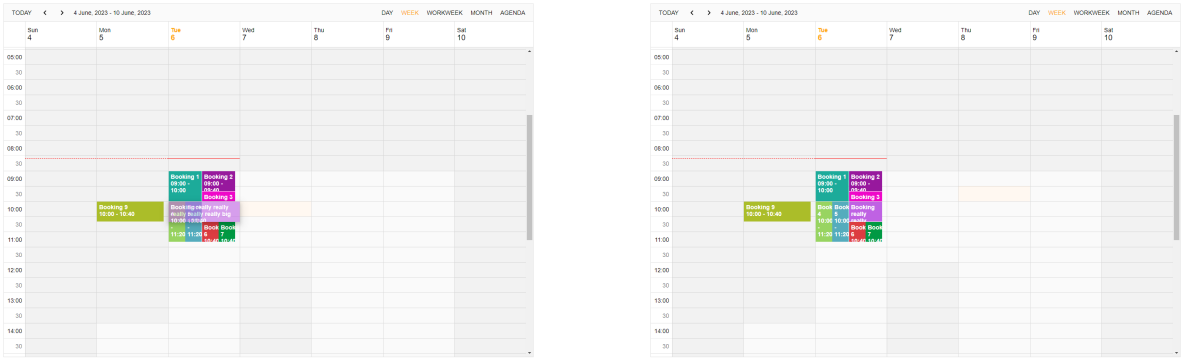



Figure 34: Quick information popup

Additionally, clicking on cells allows booking creation, by opening an editor where the user can fill in the needed information to create a new event. The system also allows two types of event dragging, which are drag and drop and resizing. The user can drag and drop events to reschedule events and resize them to change their duration. This functionality adds ease of use and simplifies booking management as well.

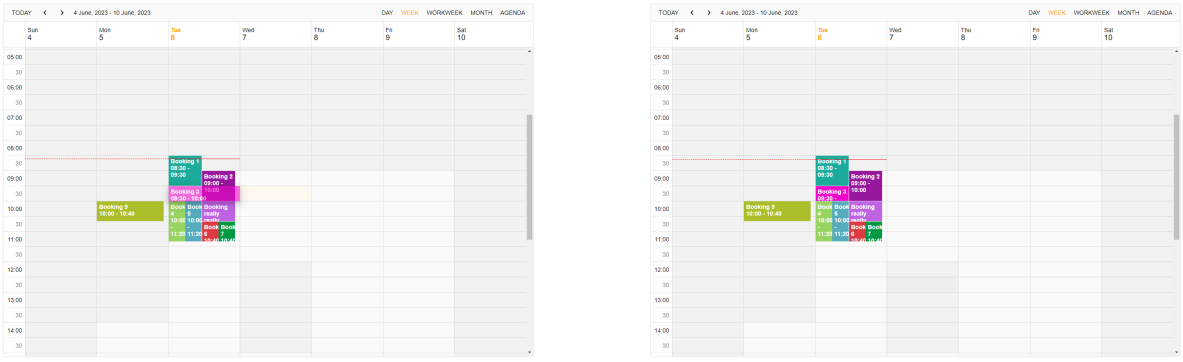
There is an issue that needs to be addressed regarding bookings that occur simultaneously. When multiple bookings share the same time interval, their overlapping causes certain bookings to become hidden. To resolve this problem, an algorithm has been developed to prevent booking collisions within the same column.

Upon creating a booking, the system determines its position on the grid. It then examines the number of overlapping bookings and calculates their widths and positions based on the available space within the column and the start times of the bookings. Bookings that start earlier than overlapping ones are positioned on the left side, and in cases where the start times are the same, the duration of the bookings determines their placement, as can be observed with a booking resize in Figure 37. It is important to note that the main objective is to make appointments occupy the most space possible so their information is more exposed to the user, so if there are gaps to fill by appointments overlapping, instead of always appending new ones to the right, the system tries to fit them into available space, if it exists. Examples of this idea are illustrated in Figures 35 and 36.



(a) Dragging booking (b) Dropping booking

Figure 35: Drag and Drop example. Booking fits on available space at the right

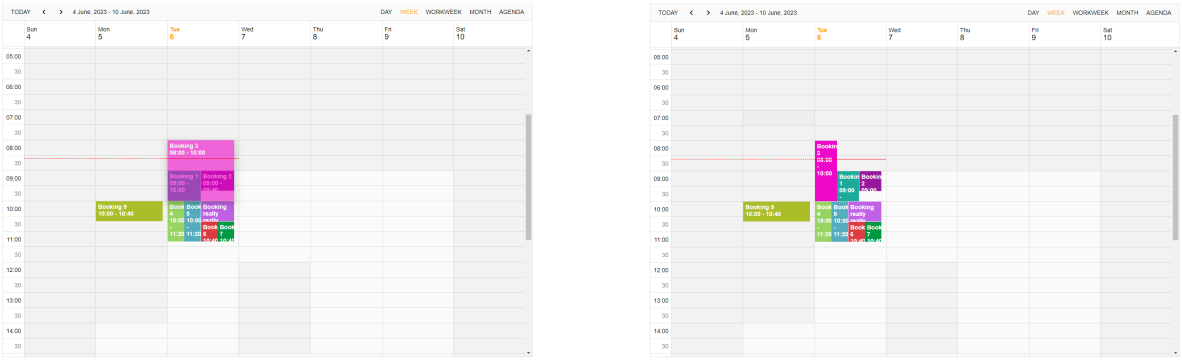


(a) Dragging booking (b) Dropping booking

Figure 36: Drag and Drop example. Booking fits on available space at the left

### A.1.2 *Month View*

The month view of the application presents a layout similar to physical calendars, facilitating user familiarity. It showcases the days of the month as cells arranged in a grid format, with



(a) Resizing booking (b) After resize

Figure 37: Resize example

each row representing a week. To aid comprehension, the header bar of the view displays the days of the week, providing clear associations between specific month days and their corresponding weekdays. An illustration can be observed in Figure 38.

In terms of booking rendering, geometry was not required for this view. Each booking is assigned to one or multiple cells, depending on its duration. Consequently, each booking is represented as a child element within the corresponding cell. In cases where the bookings exceed the cell's capacity, a label indicating the number of additional bookings for that particular day is displayed below the existing bookings, hiding the *overflowing* ones.

Functionality-wise, users can interact with the month view in several ways. They can click on the cells to navigate to the day view for the selected day. Furthermore, by clicking on bookings, a quick information popup is shown, providing specific details about the booking. Additionally, users have the option to intuitively manage bookings by dragging and dropping them to different cells, facilitating easy booking management and enhancing the overall user experience.

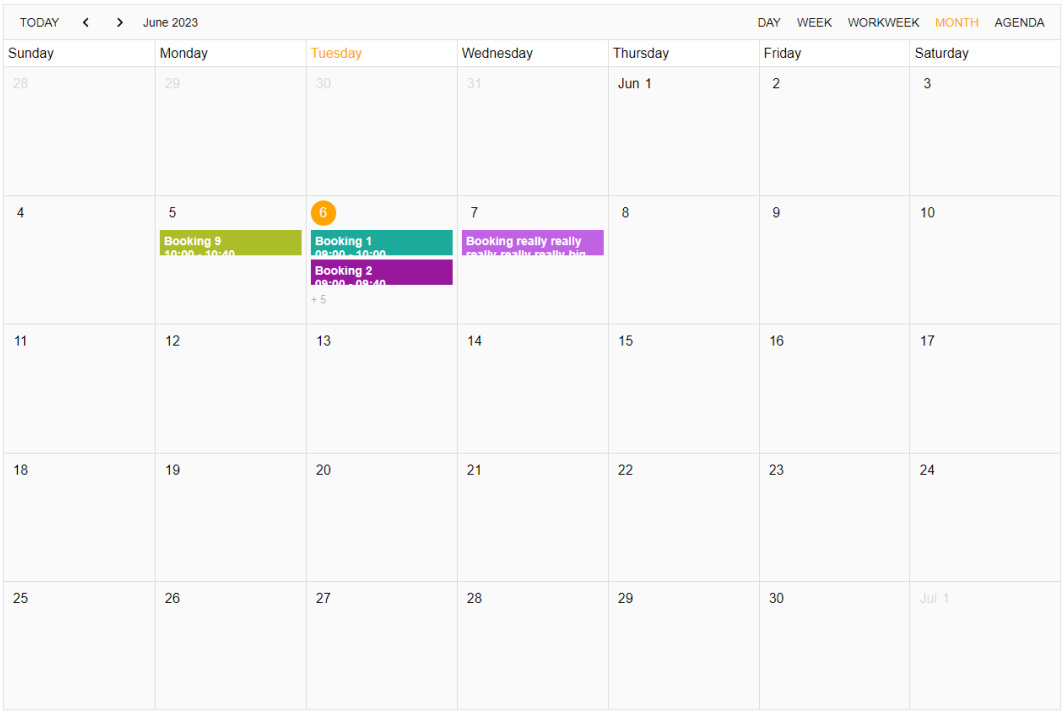



Figure 38: Month View

### A.1.3 *Agenda View*

The Agenda View is an advantageous display option for users who desire a concise overview of their bookings within a specific time frame. It consists of two sections: a left section showcasing the corresponding day and a right section presenting the bookings. Each day is represented as a block, encompassing the bookings associated with it. The bookings are arranged in ascending order based on their start times, and if multiple bookings have the same start time, their duration is taken into consideration for the ordering, as can be observed in Figure 39.

Additionally to the daily blocks, these blocks are further grouped into month blocks. Each month block begins with a header that displays the month associated with the days contained within it. This design choice aims to enhance user comprehension and facilitate a clear display of information.

In terms of interactivity, users have the option to click on a specific day within the Agenda View to navigate to the corresponding day view, providing a more detailed overview of the selected day. Furthermore, users can click on individual bookings within the view, triggering the display of the quick information popup.

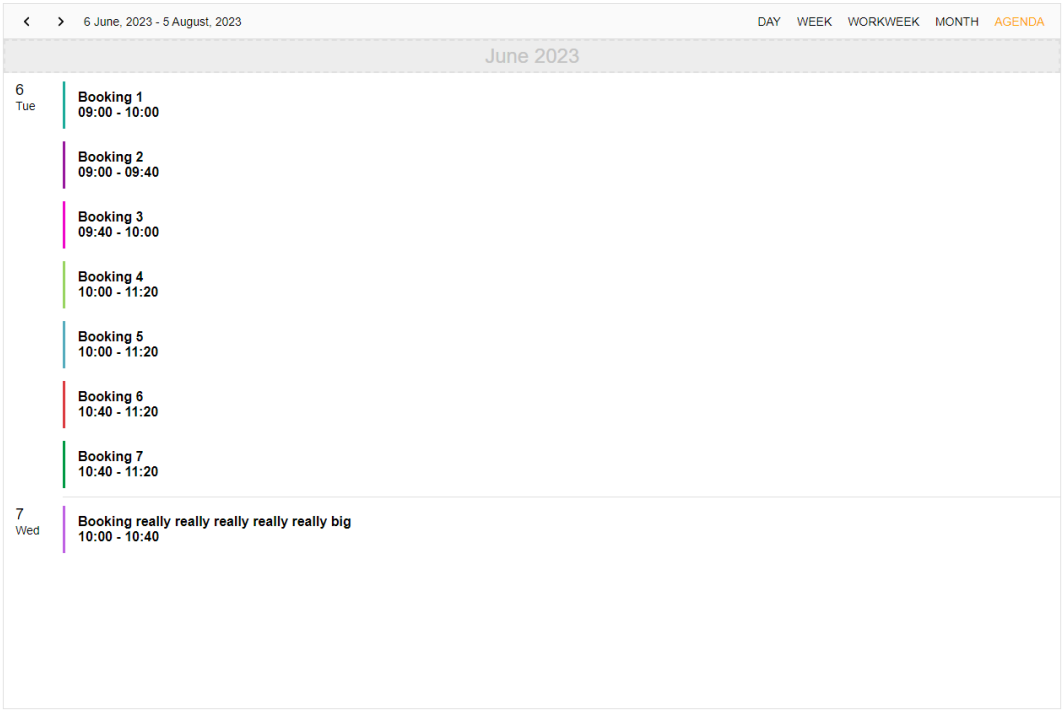



Figure 39: Agenda View