Universidade de Évora - Escola de Ciências e Tecnologia

Mestrado em Engenharia Informática

Dissertação

Ferramenta de monitorização e gestão de listas de espera no Hospital de Évora

Fernando Jorge Barradas Corvelo

Orientador(es) | Luís Rato

Teresa Gonçalves

Évora 2021

Universidade de Évora - Escola de Ciências e Tecnologia

Mestrado em Engenharia Informática

Dissertação

Ferramenta de monitorização e gestão de listas de espera no Hospital de Évora

Fernando Jorge Barradas Corvelo

Orientador(es) | Luís Rato

Teresa Gonçalves

Évora 2021

A dissertação foi objeto de apreciação e discussão pública pelo seguinte júri nomeado pelo Diretor da Escola de Ciências e Tecnologia:

Presidente	|	Irene Pimenta Rodrigues (Universidade de Évora)
Vogais	|	Teresa Gonçalves (Universidade de Évora) (Orientador) Vitor Beires Nogueira (Universidade de Évora) (Arguente)

Évora 2021

A todos os que me motivaram a não desistir.

Agradecimentos

Em primeiro lugar, gostaria de agradecer aos meus orientadores, Professora Teresa Gonçalves e Professor Luís Rato, por todo o apoio e ajuda que me deram durante a realização deste trabalho.

Em segundo lugar, agradecer também a toda a equipa do Hospital de Évora com que tive a oportunidade de me cruzar durante a fase inicial deste trabalho, e cujos contributos foram essenciais para a sua realização.

Finalmente, um agradecimento muito especial a toda a minha família e amigos que sempre me incentivaram e me deram motivação para completar mais esta etapa do meu percurso académico.

A todos, o meu mais sincero obrigado.

Conteúdo

Lista de Figuras

Lista de Tabelas

Lista de Acrónimos

SONHO Sistema Integrado de Informação Hospitalar

SPMS Serviços Partilhados do Ministério da Saúde

API *Application Programming Interface*

CSV *Comma Separated Values*

ACCDB *Access Database*

SNS Serviço Nacional de Saúde

HTML *HyperText Markup Language*

DOM *Document Object Model*

CSS *Cascading Style Sheets*

W3C *World Wide Web Consortium*

npm *Node Package Manager*

SGBD Sistemas de Gestão de Bases de Dados

SQL *Structured Query Language*

YAML *YAML Ain't Markup Language*

LTS *Long-Term Support*

LAN *Local Area Network*

ERP *Enterprise Resource Planning*

Sumário

O serviço de Consulta Externa do Hospital de Évora conta com uma grande afluência diária de utentes, o que faz com que exista a necessidade de monitorizar e gerir da lista de espera que inevitavelmente acaba por existir. Para isso, é também necessário fazer a gestão do espaço físico da consulta e das agendas médicas, uma vez que existe um número limitado de gabinetes onde as consultas se podem realizar, tal como existe um número limitado de horas que um médico tem disponíveis para realizar consultas.

No âmbito desta dissertação, foi feita uma análise dos processos manuais e digitais atualmente existentes no Hospital para fazer a gestão das atividades referidas, de modo a perceber quais as melhorias que poderiam ser implementadas. Após esta análise, conseguiram-se identificar alguns pontos de melhoria, maioritariamente relacionados com uma aplicação informática que o Hospital utiliza. Assim, chegou-se a um conjunto de requisitos essenciais que foram posteriormente implementados, através do desenvolvimento de uma nova aplicação, com o objetivo de complementar a aplicação informática existente.

No entanto, e apesar de cumprir todos os requisitos identificados, para ser colocada em produção, a ferramenta desenvolvida requer algum trabalho adicional em colaboração com o Hospital, para finalizar o processo através do qual a ferramenta vai ter acesso aos dados do Hospital.

Palavras chave: Monitorização, Gestão, Consulta, Externa, Hospital

Abstract

Monitoring and management tool for waiting lists in Évora's Hospital

The Outpatient Consultation service at the Évora's Hospital has a large daily flow of patients, which means that there is a need to monitor and manage the waiting list that inevitably exists. Thus, it's also necessary to manage the consultation physical space and medical agendas, since there is a limited number of offices where consultations may be conducted, just as there is a limited number of hours that doctors have available to conduct consultations.

Within the scope of this work, an analysis was carried out focusing on the manual and digital processes currently in place at the Hospital, in order to understand what improvements could be achieved. After this analysis, some improvement points were discovered, mostly related to an application that the Hospital currently uses. Thus, a set of key requirements were identified and subsequently applied, in the development of a new application, with the goal of complementing the existing Hospital's application.

However, and despite meeting all the requirements identified, the developed application, in order to be deployed into production, requires some additional work in collaboration with the Hospital, so that the process through which the application will have access to the Hospital's data, may be concluded.

Keywords: Monitoring, Management, Consultation, External, Hospital

Capítulo 1

Introdução

Os sistemas de informação hospitalar são uma parte importante para o bom funcionamento de um hospital. Estes sistemas permitem armazenar e gerir, em formato digital, toda a informação necessária ao funcionamento do mesmo, eliminando assim a necessidade de utilizar registos em formato de papel.

Os primeiros sistemas de informação hospitalar surgiram por volta de 1960, e o seu foco estava principalmente restrito à gestão financeira dos hospitais. Esta limitação devia-se aos elevados custos dos sistemas computacionais dessa altura e à não existência de comunicação em rede entre esses sistemas [52].

Mais tarde, por volta de 1980, com a aparecimento das redes de computadores locais (LAN) e com a disponibilidade de computadores pessoais a um custo mais acessível, os sistemas de informação hospitalar começaram a ser utilizados de forma mais abrangente nas tarefas de gestão hospitalar [52].

Atualmente, existem recursos computacionais e de rede que tornam banal a utilização deste tipo de sistemas. Contudo, os sistemas de informação hospitalar utilizados atualmente por alguns hospitais continuam a apresentar desafios e limitações que servem de motivação para a criação de ferramentas complementares como a que foi desenvolvida no âmbito desta dissertação.

1.1 Motivação

Quando um pedido de consulta entra no sistema do hospital, este pode ter origens distintas. Pode ser proveniente de um Centro de Saúde público, de uma clínica privada, de um pedido por parte do médico para uma consulta

subsequente ou de uma situação de urgência.

Para cada pedido de consulta, dependendo da sua origem, é necessário atribuir um médico, uma data e um gabinete. No Hospital de Évora este processo de atribuição é atualmente executado pelos funcionários administrativos do hospital, de forma manual, verificando a capacidade das agendas médicas, disponibilidade dos gabinetes e alocando os pedidos de consultas aos médicos e gabinetes mais convenientes.

A atribuição é feita numa aplicação informática, mas esta aplicação apenas permite a visualização e a inserção dos dados, sem que exista algum mecanismo que permita agilizar e facilitar o processo.

O único mecanismo parcialmente automático que existe tem por base algumas extrações de dados realizadas regularmente, a partir da referida aplicação, pelos Serviços de Informática do Hospital, e que são posteriormente tratadas na aplicação informática *Microsoft Excel* [29], permitindo produzir um mapa que possibilita uma melhor visualização e cruzamento dos dados relativos às consultas, agendas e gabinetes.

1.2 Objetivos

O processo descrito no ponto anterior permite que haja oportunidade para vários melhoramentos. A partir de algumas reuniões com membros da administração do Hospital, foi possível identificar dois melhoramentos chave. São estes:

- a gestão da lista de espera para primeiras consultas e
- a gestão do espaço físico da consulta, ou seja, dos gabinetes.

Além disso, foi também identificado como um ponto importante a capacidade de gestão das agendas médicas.

Gestão da lista de espera para primeiras consultas. As primeiras consultas representam um maior desafio para o Hospital em relação às consultas subsequentes. Isto porque, ao contrário do que acontece nas consultas subsequentes, nas primeiras consultas o utente ainda não teve contacto com o Hospital, e por isso ainda não está a ser seguido por nenhum médico, o que torna necessário encontrar um médico disponível para atribuir ao utente. Também é necessário ter em conta a prioridade do pedido de consulta, bem como há quanto tempo o utente se encontra em lista de espera.

Assim, existem dois melhoramentos possíveis em relação às primeiras consultas. Em primeiro lugar, a possibilidade de proporcionar aos funcionários administrativos, responsáveis pela gestão das consultas, mecanismos de visualização e ordenação da lista de espera em função da prioridade e antiguidade do pedido. E em segundo lugar, mecanismos para automaticamente encontrar médicos com agenda disponível para cada um dos pedidos de consulta, tendo em conta critérios como a especialidade e a brevidade com que é possível efetivar a consulta.

Gestão do espaço físico da consulta. Tendo em conta o extenso número de gabinetes médicos existentes no Hospital, há a necessidade de saber qual a sua taxa de ocupação, e conseguir rapidamente visualizar a sua disponibilidade.

Para isso, foram identificados melhoramentos que têm como objetivo proporcionar aos funcionários administrativos a possibilidade de visualizar um calendário de ocupação por gabinete, alimentado pelos dados existentes nas agendas médicas.

Gestão das agendas médicas. Apesar de não ser um ponto chave, foi também identificada a necessidade de permitir a visualização das agendas médicas por parte dos funcionários administrativos, de modo a complementar a utilização dos melhoramentos identificados anteriormente.

1.3 Principais Contribuições

A principal contribuição desta dissertação é uma ferramenta para utilização pelo Hospital de Évora para complementar e facilitar o processo manual de gestão da lista de espera para primeiras consultas atualmente existente. A ferramenta também inclui a gestão do espaço físico da consulta e gestão das agendas médicas.

Esta ferramenta pode também ser utilizada por qualquer outro hospital com requisitos semelhantes, requerendo apenas que seja adaptado o processo de sincronização de dados entre a ferramenta e o hospital.

Outra contribuição desta dissertação é um levantamento de sistemas de informação hospitalar existentes no mercado, onde são identificadas as suas funcionalidades e é feita uma análise e comparação das mesmas.

Por fim, também se considera uma contribuição desta dissertação a lista de requisitos apresentada, fruto de várias conversas e reuniões realizadas com

elementos das equipas do Hospital.

1.4 Principais Constrangimentos

Tratando-se de processos hospitalares, existem algumas dificuldades no que diz respeito ao acesso e tratamento dos dados dos utentes. Para além disso, o facto de a aplicação informática utilizada pelo Hospital ter sido desenvolvida há muitos anos, faz com que não permita a utilização de técnicas mais recentes de acesso aos dados como, por exemplo, o acesso através de *APIs*.

Assim, durante a execução do trabalho descrito nesta dissertação, não foi possível aceder diretamente à aplicação para obter os dados necessários à sua realização. O processo encontrado pelos Serviços de Informática do Hospital para a partilha de dados foi a extração dos mesmos diretamente da base de dados da aplicação, excluindo a informação confidencial, e a posterior partilha no formato *CSV* e *ACCDB*.

A partilha neste formato foi suficiente para ter uma visão global sobre as tabelas e colunas da base de dados da aplicação, mas não permitiu identificar inequivocamente as relações entre as várias tabelas, nem o tipo de dados de cada coluna, uma vez que nenhum destes dois formatos contém os metadados necessários para a sua correta identificação.

Assim, e como o Hospital não dispõe de nenhum tipo de documentação sobre a referida aplicação, as relações e os tipos de dados foram inferidos com base na análise dos dados e, para aqueles em que a inferência não foi possível, questionou-se Serviços de Informática do Hospital.

1.5 Organização da Dissertação

Este documento está organizado em seis capítulos que procuram descrever o trabalho realizado no decorrer desta dissertação.

Capítulo 1. O primeiro capítulo, Introdução, tem como objetivo apresentar o tema tratado neste trabalho. Neste capítulo é apresentada a motivação, os objetivos e as principais contribuições do trabalho, bem como os alguns constrangimentos encontrados durante a realização do mesmo.

Capítulo 2. No segundo capítulo, Sistemas de Informação Hospitalar, são apresentadas soluções de sistemas de informação hospitalar existentes no

mercado e é feita uma comparação entre as funcionalidades que cada um disponibiliza.

Capítulo 3. No terceiro capítulo, Tecnologias e Ferramentas, é feita uma análise de tecnologias e ferramentas utilizadas para o desenvolvimento de *software*.

Capítulo 4. O quarto capítulo, Análise e Implementação, tem como objetivo fazer uma análise do problema e apresentar uma proposta de implementação, onde se inclui os requisitos e as ferramentas e tecnologias selecionadas para resolver o problema. É também apresentada a arquitetura do sistema implementado, o processo de integração com o sistema do Hospital, e o processo para futuros desenvolvimentos de novas funcionalidades ou correção de erros.

Capítulo 5. No quinto capítulo, Casos de Uso, são apresentados diferentes cenários de utilização da aplicação desenvolvida.

Capítulo 6. O sexto e último capítulo, Conclusões e Trabalho Futuro, apresenta as conclusões obtidas relativamente ao trabalho realizado no âmbito desta dissertação e uma proposta de trabalho futuro.

Capítulo 2

Sistemas de Informação Hospitalar

Os sistemas de informação hospitalar são sistemas computacionais que permitem a gestão de toda a informação necessária ao funcionamento de um hospital. Atualmente, estes sistemas permitem fazer a gestão de vários departamentos e serviços do hospital. As componentes tradicionais de um sistema de informação hospitalar são: administrativa, clínica e de apoio [48].

A componente administrativa inclui, entre outros, subsistemas de gestão médico-administrativa, como a identificação dos utentes, e de gestão de atividades hospitalares, como a gestão de consultas e gabinetes. A componente clínica inclui, entre outros, a gestão dos registos clínicos dos utentes, como exames, diagnósticos e prognósticos. Finalmente, a componente de apoio inclui, entre outros, as atividades de gestão laboratorial e de exames [48].

Existem no mercado, nacional e internacional, várias opções de sistemas de informação hospitalar. Estas opções incluem sistemas desenvolvidos por empresas privadas, mas também sistemas desenvolvidos pela comunidade em formato de código aberto.

Nas secções seguintes são analisadas algumas destas opções, e identificadas as suas principais funcionalidades, com especial foco nas capacidades de gestão de utentes, listas de espera, consultas, agendas médicas e gabinetes.

2.1 Soluções Empresariais

2.1.1 Globalcare

O Globalcare [18] é um *software* hospitalar desenvolvido pela Glintt [17]. Este *software* é composto por 4 produtos, entre eles o *Hospital Management System* [19] que é responsável pela gestão hospitalar.

O *Hospital Management System* permite realizar a gestão de utentes, onde se inclui funções de natureza administrativa associada à sua identificação e gestão em unidades hospitalares, bem como o registo e monitorização de todo o seu percurso desde o primeiro contacto até à alta administrativa. Permite também a gestão de listas de espera, consultas, agendas e a consulta de informação operacional, tal como listas de utentes, gestão da sala de espera e atendimentos.

Para além da gestão de utentes, este *software* também permite a gestão da faturação, onde o principal objetivo é o tratamento e emissão de documentos financeiros e de tesouraria.

Por fim, também está disponível uma funcionalidade opcional de gestão de honorários, que permite de uma forma automática fazer o cálculo do valor de pagamento de honorários a atribuir aos profissionais de saúde.

2.1.2 F3M - Gestão de Unidades de Saúde

O *software* Gestão de Unidades de Saúde [14] da F3M [13] permite a gestão integrada de unidades de diferentes dimensões e tipologias, onde se incluem as unidades hospitalares. A oferta da F3M engloba toda a gestão dos processos administrativos de uma unidade de saúde, desde a relação com os utentes, entidades pagadoras, meios complementares de diagnóstico, farmácia, honorários, até à gestão de equipamentos e manutenção [14].

A solução de Gestão Hospitalar incluída neste *software* assegura um controlo global dos recursos disponíveis, otimizando a sua utilização tendo em conta as necessidades dos diferentes serviços. Esta permite a gestão financeira e administrativa, a gestão dos serviços de atendimento em ambulatório, a gestão da capacidade de internamento, o planeamento cirúrgico e a logística hospitalar. Estão também incluídas funcionalidades de gestão das agendas dos profissionais de saúde, gestão de admissões e tempos de espera, gestão da consulta externa, farmácia hospitalar, entre outras.

2.1.3 SIGEHP - Hospitais

O SIGEHP, Sistema Integrado de Gestão Hospitalar, é um de *software* desenvolvido pela SISBIT para a informatização global de hospitais [47]. Este *software* procura responder a necessidades de gestão de recursos, orientadas para a melhoria da qualidade dos serviços prestados à comunidade.

A configuração base do SIGEHP assenta em 6 grupos de aplicações distintas, que permitem a gestão de utentes, processos clínicos, farmácia hospitalar, aprovisionamento hospitalar, gestão de recursos humanos hospitalares e gestão dos serviços financeiros hospitalares.

O conjunto destas aplicações permite a gestão de processos administrativos relativos às admissões e altas dos utentes, gestão de meios complementares de diagnóstico e terapêutica, gestão da consulta externa, gestão de *stocks*, processamento de dados necessários à faturação, gestão de recursos humanos, entre outros.

2.2 Soluções de Código Aberto

2.2.1 *GNU Health*

O *GNU Health* [20] é um projeto de código aberto desenvolvido pela organização sem fins lucrativos *GNU Solidario* [22]. Uma das ferramentas disponibilizadas por este projeto é o *GNU Health Hospital Management (HMIS)*. Esta ferramenta tem como foco a gestão hospitalar, e conta com funcionalidades como a gestão de utentes, a gestão de processos administrativos como consultas, camas, *stocks*, finanças e recursos humanos [21].

2.2.2 *HospitalRun*

O *HospitalRun* [25] é um sistema de informatização hospitalar de código aberto, desenvolvido e mantido por uma comunidade independente. Este *software* foi inicialmente desenvolvido com o objetivo de proporcionar um sistema de informatização hospitalar moderno em hospitais de países em desenvolvimento, onde os recursos são muito limitados. O *HospitalRun* está desenhado com foco no funcionamento *offline*, de forma a poder ser utilizado em centros hospitalares remotos.

As principais funcionalidades do *HospitalRun* são a gestão de utentes, agendamento de consultas, gestão de medicamentos, exames, laboratórios, *stocks* e processamento de faturação.

2.2.3 *Bahmni*

O *Bahmni* [2] é um projeto de código aberto gerido pela *Bahmni Coalition* [3]. Este projeto combina e melhora outros produtos de código aberto existentes, tornando-os numa única solução de gestão médica e hospitalar.

Para a gestão de utentes, o *Bahmni* utiliza o *software* de código aberto *OpenMRS* [37]. As suas funcionalidades mais relevantes são o registo e a identificação dos utentes, a gestão das consultas e o registo dos diagnósticos dos utentes.

Através de outros produtos que inclui, o *Bahmni* consegue ainda realizar a gestão de *stocks*, gestão financeira e de faturação, e a gestão laboratorial.

2.2.4 *ERPNext*

O *ERPNext* [12] é uma *software* desenvolvido pela *Frappe* [16] e surgiu como uma alternativa de código aberto a outras soluções do tipo ERP (*Enterprise Resource Planning*) existentes no mercado. Em cima das funcionalidades padrão de um ERP, foram criados módulos adicionais específicos para diversos setores da industria, como é o caso do setor hospitalar.

O *ERPNext* inclui, assim, funcionalidades como a gestão de utentes e de todo o seu historial clínico, listas de espera, gestão de profissionais de saude e das suas agendas, gestão de consultas, gestão de procedimentos clínicos e prescrições, gestão de gabinetes, gestão de *stocks* e faturação.

2.3 Conclusões

Depois de analisar as soluções apresentadas anteriormente, é possível sumarizar as suas funcionalidades no que diz respeito às capacidades de gestão de utentes, listas de espera, consultas, agendas médicas e gabinetes (tabela 2.1). Desta tabela é possível retirar algumas conclusões.

Pode ver-se que todas as soluções analisadas apresentam as funcionalidades de gestão de utentes e de consultas, o que parece indicar que estas são duas funcionalidades essenciais em sistemas de informação hospitalar.

Por outro lado, apenas duas delas apresentam a funcionalidade de gestão de gabinetes. Sendo que uma delas (o *GNU Health*) não a descreve propriamente como gestão de gabinetes, mas sim como gestão de camas. Ainda assim, a funcionalidade de gestão de camas pode ser adaptada para funcionar como gestão de gabinetes. Isto sugere que a funcionalidade de gestão de gabinetes

Software	**Utentes**	**Espera**	**Consultas**	**Agendas**	**Gabinetes**
Globalcare	X	X	X	X	
F3M	X	X	X	X	
SIGEHP	X		X		
GNU Health	X		X		X
HospitalRun	X		X		
Bahmni	X		X		
ERPNext	X	X	X	X	X

Tabela 2.1: Comparação de *Software* de Informação Hospitalar

é uma das menos relevantes.

Numa posição intermédia podem ver-se as funcionalidades de gestão de listas de espera e de gestão de agendas, em que apenas três das soluções analisadas as apresentam.

Finalmente, é importante realçar o *software ERPNext*, que apresenta todas as funcionalidades destacadas nesta análise.

Capítulo 3

Tecnologias e Ferramentas

No desenvolvimento de *software* podem ser utilizadas diversas tecnologias e ferramentas. Dependendo do tipo de aplicação a desenvolver, podem existir algumas com características que as tornem mais relevantes em relação a outras. No contexto desta dissertação, optou-se por incidir maioritariamente nas tecnologias e ferramentas necessárias para o desenvolvimento de uma aplicação *web*.

De uma forma geral, qualquer aplicação *web* que tenha como finalidade interagir com dados do utilizador, tem como base três componentes principais: o *frontend*, o *backend* e a base de dados.

O *frontend* é a componente responsável pela interação com o utilizador. Esta interação pode ser unidirecional ou bidirecional. Ou seja, a componente de *frontend* pode apenas mostrar informação ao utilizador sem que este tenha que interagir com a mesma, ou, para além disso, a componente de *frontend* pode também ser responsável por registar as interações do utilizador.

Por outro lado, a componente de *backend* não tem qualquer interação com o utilizador. Esta componente é responsável pelo processamento de dados e pela lógica da aplicação. E é também responsável por interagir com a componente de *frontend*.

Finalmente, a componente da base de dados é responsável por guardar e permitir o acesso os dados da aplicação. Esta componente apenas interage com a componente de *backend*, não estado visível diretamente para a componente de *frontend* nem para o utilizador.

Assim, foi realizada uma análise às diversas tecnologias e ferramentas disponíveis para cada uma das componentes referidas anteriormente. A partir dessa análise foram escolhidas aquelas que melhor se adaptam aos objetivos

da aplicação, tendo em conta alguns critérios como, por exemplo, o tipo de licenciamento, completude da documentação, escalabilidade e facilidade de implementação.

3.1 *Backend*

Para a componente de *backend*, foi necessário identificar e analisar duas vertentes: o servidor *web* a utilizar para o desenvolvimento da aplicação e o *software* de gestão de base de dados para a construção da base de dados interna da aplicação.

3.1.1 Servidor *web*

De entre o vasto leque de linguagens de programação que permitem a criação de servidores *web*, foram escolhidas quatro, que não sendo as quatro mais populares, estão entre as linguagens mais populares para desenvolvimento de servidores *web* [57]: *Java*, *JavaScript*, Python e PHP.

Java

O *Java* [27] é uma linguagem de programação bastante popular, criada em 1995 pela Sun Microsystems [39]. O código *Java* é compilado numa linguagem intermédia chamada *bytecode*, que é posteriormente executado pela JVM (*Java Virtual Machine*). Isto permite que o mesmo código seja executado em diversos tipos e arquiteturas de sistemas, desde que exista uma JVM compatível.

Para esta dissertação foi analisada a biblioteca Spring Boot [50]. Esta é uma biblioteca que fornece ao *Java* um conjunto de funcionalidades pré-desenvolvidas que facilitam a criação de aplicações *web*. Foram identificadas as seguintes vantagens:

- Possui grande suporte por parte da comunidade;
- Inclui funcionalidades que facilitam e agilizam o desenvolvimento *web*, como por exemplo a criação de APIs REST;
- Inclui uma camada de acesso de dados, que permite a interação com diversos tipos de base de dados.

Como desvantagem, a biblioteca Spring Boot apresenta uma acentuada curva de aprendizagem.

JavaScript

JavaScript [28] é uma linguagem de programação leve, interpretada ou compilada *just-in-time*, que utiliza funções de primeira classe. Esta linguagem tem como base a linguagem de *scripting* ECMAScript [10]. Embora tenha sido inicialmente desenvolvida como uma linguagem de *scripting* para páginas *web*, hoje em dia é também utilizada em outras aplicações fora da *web*. Uma das mais populares utilizações do *JavaScript* fora do contexto *web* é o *Node.js* [34].

O *Node.js* [34] é um ambiente de execução de *JavaScript* que permite desenvolver aplicações no lado do servidor utilizando exatamente a mesma linguagem que no contexto *web*. Este tem como base o motor de *JavaScript* V8 [51] que é também utilizado no *Google Chrome* [24]. O *Node.js* conta com um gestor de bibliotecas (ou módulos) denominado *npm* (*Node Package Manager*) [36], que permite a partilha e *download* de bibliotecas por parte dos programadores [35].

As suas vantagens incluem:

- Grande comunidade de programação;
- Grande popularidade no desenvolvimento de aplicações *web* devido às bibliotecas dedicadas existentes;
- Grande flexibilidade, uma vez que a mesma linguagem de programação permite desenvolver aplicações no lado do cliente e do servidor.

As principais desvantagens são:

- Tratando-se de uma linguagem interpretada, tem um desempenho reduzido em relação a outras linguagens compiladas;
- A sua natureza *single-threaded* faz com que não seja ideal para aplicações que necessitem executar processos em paralelo.

Python

Python [45] é uma linguagem interpretada, interativa e orientada a objetos. Também suporta outros paradigmas de programação, tais como procedimen-

tal ou funcional. Tal como o *Node.js*, o *Python* também dispõe de um gestor de bibliotecas (ou módulos) denominado *pip* [44].

As principais vantagens que apresenta incluem:

- Grande comunidade de programação;
- Grande popularidade no desenvolvimento de aplicações *web* devido às bibliotecas dedicadas existentes.

Como principal desvantagem, apresenta um desempenho reduzido em relação a outras linguagens, já que se trata de uma linguagem interpretada.

PHP

O PHP [42] é uma linguagem de *scripting* especialmente adequada para o desenvolvimento *web*. O código PHP pode ser introduzido diretamente nos documentos HTML, sendo processado pelo servidor e enviado para o *web browser* apenas o resultado final da execução do código.

Apresenta como vantagens:

- Linguagem de referência no desenvolvimento *web*, dada a sua antiguidade;
- Inclui todas as funcionalidades necessárias para o desenvolvimento *web*;
- Suporte de um vasto leque de bases de dados sem a necessidade de instalar bibliotecas adicionais.

Como principal desvantagem, apresenta uma quebra de popularidade devido ao aparecimento de tecnologias mais modernas, como o *Java*, o *Node.js* e o *Python*.

3.1.2 *Software* de gestão de base de dados

Existem dois tipos principais de bases de dados. Relacionais e não relacionais. Estes diferem na forma como os dados são estruturados e guardados, sendo que para cada um deles existem aplicações especificas onde um é mais vantajoso do que o outro.

As bases de dados relacionais são representadas por tabelas, colunas e linhas. Cada tabela representa uma entidade sobre a qual se pretende guardar informação, em que as colunas correspondem a todos os tipos de informação

a guardar sobre essa entidade, e as linhas são os vários registos que essa entidade contém. Estas tabelas podem ser ligadas entre si através a de chaves primárias e secundárias, e assim representar relações de dados mais complexas.

As bases de dados não relacionais não seguem nenhum modelo padrão, o que significa que podem assumir vários formatos, dependendo da forma como os dados são representados e guardados. Por exemplo, os dados podem ser orientados a colunas ao invés de linhas, fazendo com que o acesso aos dados da mesma coluna seja mais rápido. Ou nem sequer utilizar linhas e colunas, como é o caso das bases de dados orientadas a documentos.

Para efeitos desta dissertação, foram analisados três sistemas de gestão de bases de dados (SGBD), dois relacionais e um não relacional. Estes sistemas foram escolhidos de entre os cinco sistemas SGDB mais populares à data [5].

PostgreSQL

PostgreSQL [43] é um SGBD de código aberto, com mais de 30 anos de desenvolvimento ativo. Tem ganho uma forte reputação devido à sua fiabilidade, funcionalidades e desempenho. O *PostgreSQL* utiliza e estende a linguagem SQL.

As suas vantagens incluem:

- Código aberto e sem custos;
- Comunidade de código aberto que continua a contribuir com melhorias e novas funcionalidades;
- Fiabilidade, funcionalidades e desempenho.

Embora o desempenho seja apontado como um ponto positivo pelos autores, a comunidade tende a não concordar e a apontar a velocidade como um ponto negativo [40].

MySQL

O *MySQL* [31] é um SGBD que, tal como o *PostgreSQL*, utiliza a linguagem SQL. O *MySQL* está disponível em duas versões. Uma de código aberto e outra comercial, licenciada pela *Oracle Corporation* [38].

Apresenta como vantagens [32] [40]:

- Utilização sem custos através da versão de código aberto;
- Grande quantidade de software desenvolvido que permite dar suporte à grande maioria das aplicações;
- Velocidade, fiabilidade, escalabilidade, popularidade e facilidade de utilização.

As suas desvantagens incluem [40]:

- Não apresenta suporte para a totalidade da linguagem SQL, dado o facto de ter sido desenhado com foco na velocidade;
- Algumas funcionalidades apenas estão disponíveis na versão comercial.

MongoDB

O *MongoDB* [30] é uma base de dados não relacional baseada em documentos. Ao contrário dos exemplos apresentados anteriormente, o *MongoDB* não utiliza tabelas, colunas e linhas para representar os seus dados, mas sim coleções e documentos. Ainda assim, pode-se estabelecer uma analogia entre as tabelas e linhas das bases de dados relacionais com as coleções e documentos do *MongoDB*. O equivalente às colunas das bases de dados relacionais está implícito na construção da estrutura dos documentos no *MongoDB*.

Outra diferença é a linguagem utilizada para aceder aos dados. O *MongoDB* utiliza a sua própria linguagem, a *MongoDB Query Language*.

O *MongoDB* está disponível em duas versões. Uma de código aberto e outra comercial.

As suas vantagens incluem:

- Possibilita uma utilização sem custos através da versão de código aberto;
- Desenhado para proporcionar facilidade de desenvolvimento e escalabilidade.

O *MongoDB* apresenta as seguintes desvantagens:

- Curva de aprendizagem devido à forma como os dados devem ser estruturados e representados;
- Curva de aprendizagem na utilização da *MongoDB Query Language*.

3.2 *Frontend*

Para a componente de *Frontend*, foram analisadas duas opções de desenvolvimento de interfaces *web*: desenvolvimento *web* tradicional utilizando HTML [26], CSS [4] e JavaScript [28], e utilização de uma *framework JavaScript*.

3.2.1 Desenvolvimento *web* tradicional

Considera-se como desenvolvimento *web* tradicional à utilização de HTML, CSS e JavaScript em ficheiros estáticos, sem recorrer a utilização de *frameworks* complexas para gestão do DOM (*Document Object Model*), eventos, estado da aplicação e afins.

Esta abordagem apresenta as seguintes vantagens:

- Simplicidade no desenvolvimento;
- Método tradicional de desenvolvimento *web*;
- Curva de aprendizagem reduzida;
- Não requer aprendizagem adicional de uma *framework*;
- Adequado para aplicações de dimensão e complexidade reduzida.

As suas desvantagens são:

- Pouco escalável caso a aplicação aumente de complexidade;
- Não permite a reutilização de componentes *web*, o que implica duplicação de código;
- É necessário desenvolver toda a aplicação de raiz, ou então recorrer à utilização de bibliotecas ou *frameworks* de terceiros.

HTML

HyperText Markup Language (HTML) [26] é o mais básico elemento de construção da *web*, e tem como objetivo definir o significado e a estrutura dos conteúdos *web*.

O termo *Hypertext* refere-se à ligação que as páginas *web* têm entre si, dentro do mesmo *website* ou entre *websites* diferentes.

O termo *Markup* refere-se à forma como o HTML utiliza elementos especiais para marcar o texto, fazendo com que estes sejam mostrados nos *web browsers* de formas diferentes.

CSS

Cascading Style Sheets (CSS) [4] é uma linguagem de estilos utilizada para descrever a forma como os elementos HTML são apresentados num documento *web*.

O CSS está entre as linguagens *core* da *web* e está padronizado entre os *web browsers* de acordo com a especificação do W3C (*World Wide Web Consortium*) [56].

JavaScript

Como visto anteriormente, o *JavaScript* [28] é uma linguagem de programação versátil, que pode também ser utilizada fora do contexto *web*. No contexto *web*, esta é executada pelos *web browsers* e pode ser utilizada para determinar como uma página *web* se comporta em resposta a um determinado evento, de forma dinâmica, e sem que seja necessário recarregar a página.

3.2.2 *Frameworks JavaScript*

As *framework JavaScript* permitem a criação de páginas dinâmicas, recorrendo à criação de um DOM virtual, reutilização de componentes, gestão de eventos e estado da aplicação. À data de escrita desta dissertação, existem três *framework JavaScript* que dominam o mercado [11]: *AngularJS* [1], *React* [46] e *Vue.js* [55].

AngularJS

O *AngularJS* [1] foi criado em 2010 pela *Google* [23]. Define-se como uma *framework* e plataforma de desenvolvimento para a criação de aplicações *single-page*.

Apresenta as seguintes vantagens:

- Desenvolvimento pela *Google*;

- Mantido e atualizado de forma ativa;
- Grande suporte por parte da comunidade;
- Vasto leque de funcionalidades incluídas na *framework*.

As suas desvantagens incluem:

- Elevada curva de aprendizagem;
- Demasiado pesado e complexo para aplicações simples, devido ao elevado número de funcionalidades que inclui.

React

O *React* [46] foi criado em 2013 pelo *Facebook* [15]. Define-se como uma biblioteca *JavaScript* para a construção de *user interfaces*.

As suas vantagens incluem:

- Desenvolvimento pelo *Facebook*;
- Mantido e atualizado de forma ativa;
- Grande suporte por parte da comunidade;
- Curva de aprendizagem menor quando comparado com o *AngularJS*.

Apresenta as seguintes desvantagens:

- Fraca qualidade da documentação;
- Requer a utilização de bibliotecas de terceiros para funcionalidades mais avançadas.

Vue.js

O *Vue.js* [55] foi criado em 2014 por um ex-funcionário da *Google*. Define-se como uma *framework JavaScript* progressiva, capaz de escalar entre uma simples biblioteca até uma complexa framework. Esta *framework* é conhecida por agregar o melhor entre o *React* e *AngularJS*.

Apresenta como vantagens:

- Ativamente mantido e atualizado pela comunidade;
- Grande suporte por parte da comunidade;
- Curva de aprendizagem menor quando comparado com o *React* e *AngularJS*;
- Grande flexibilidade e facilidade no desenvolvimento.

Como desvantagem, apresenta o facto de não ter o suporte de uma grande empresa.

3.3 Ambiente de execução

O ambiente de execução (ou servidor) onde a aplicação vai ser executada é muito importante e pode ser limitativo em relação às tecnologias que estão disponíveis para esse ambiente.

Os ambientes de execução podem ser classificados em dois tipos: dedicados e virtuais. Os ambientes dedicados contam com um sistema operativo principal, que é a única camada entre a máquina e o *software* necessário para a aplicação funcionar. Por outro lado, os ambientes virtuais contam com uma camada de virtualização, que pode ser outro sistema operativo, entre o sistema operativo principal e o *software* necessário para a aplicação.

Em seguida são analisados exemplos destes dois tipos de ambientes de execução.

3.3.1 Ambientes dedicados

Pode-se considerar como ambiente dedicado uma instalação de um sistema operativo *Windows* ou *Unix* numa máquina dedicada. Neste sistema operativo é instalado todo o *software* necessário para toda a aplicação. Desta forma, todo o software partilha os mesmos recursos de *hardware* da máquina onde está instalado.

A principal vantagem deste tipo de ambientes é o desempenho. Uma vez que apenas existe uma camada entre o *software* e o *hardware*, este tipo de implementação tem um desempenho superior ao de um ambiente virtual.

Por outro lado, o facto de os mesmos recursos de *hardware* serem partilhados por todo o *software* pode fazer com que um elemento de *software* possa esgotar todos os recursos e fazer com que os outros deixem de funcionar

como pretendido. Outra desvantagem é o facto de todo o *software* necessário à aplicação ter que ser compatível com o mesmo sistema operativo.

3.3.2 Ambientes virtuais

Num ambiente virtual, existe um sistema operativo principal responsável pelo funcionamento da máquina (*host*), e em cima deste podem existir uma ou mais camadas de virtualização capazes de simular outro sistema operativo (*guest*). É neste segundo sistema operativo que o *software* necessário à aplicação vai ser executado.

Existem duas abordagens principais no que toca a ambientes virtuais. As máquinas virtuais e a *"conteinerização"*.

Máquinas virtuais

Uma máquina virtual é um *software* capaz de simular uma máquina física, e desta forma permite que nela seja instalado um novo sistema operativo, que pode ser diferente do sistema operativo da máquina real. É também possível instalar e executar mais do que um sistema operativo em simultâneo.

Exemplos de *software* que permitem a criação de máquinas virtuais são o *VirtualBox* [53] da *Oracle* [38] e o *VMware* [54] da *DELL Technologies* [6].

Esta abordagem permite, com o mesmo *hardware*, criar várias máquinas independentes, que não têm conhecimento das restantes, e alocar a cada máquina a quantidade de recursos necessários à execução do *software* pretendido.

Como uma máquina virtual não tem conhecimento das restantes máquinas virtuais a serem executadas na mesma máquina física, existe também uma melhoria do ponto de vista da segurança e a garantia que o *software* presente na máquina virtual apenas têm acesso ao que está disponível nessa mesma máquina.

Por outro lado, como as máquinas virtuais simulam por completo uma máquina física, e cada máquina contêm uma instalação completa de um sistema operativo, este processo é bastante exigente do ponto de vista dos recursos de *hardware*. Além disso, como o *software* final é executado sobre um sistema operativo que corre sobre uma camada virtual, este tem um desempenho inferior quando comparado com outro onde a camada virtual não existe.

Conteinerização

Este mecanismo de virtualização distingue-se do anterior na medida em que, em vez de simular uma nova máquina com um novo sistema operativo completo, reutiliza o *core* do sistema operativo da própria máquina, virtualizando apenas a camada de *software*. A cada uma destas instâncias virtuais dá-se o nome de *container*.

Para garantir a segurança, o *core* do sistema operativo da máquina física é partilhado entre os vários *containers* apenas em modo de leitura. Desta forma um *container* não consegue interferir na execução de outro.

O *software* mais popular para a criação e gestão de *containers* é o *Docker* [7]. No *Docker*, um *container* é o resultado do processo de construção de uma imagem. Esta imagem pode ser considerada como o sistema operativo do *container*. O *Docker* fornece uma imagem base a partir da qual podem ser executados *containers*, mas é possível estender esta imagem com *software* adicional, criando assim uma nova imagem, que pode dar origem a novos *containers*.

Como não é necessário simular um sistema operativo completo, os *containers* têm um menor consumo de recursos comparativamente às máquinas virtuais. Tal como nas máquinas virtuais, cada *container* não têm conhecimento da existência dos restantes, garantindo assim um aspeto de segurança importante.

No *Docker*, está disponível um repositório de imagens (*Docker Hub* [9]) que permite aos programadores criar *containers* com uma grande variedade de *software* pré-instalado e pré-configurado, como por exemplo aplicações de bases de dados.

O *Docker* também dispõe de uma ferramenta que permite facilitar a criação e execução de aplicações *multi-container*, o *Docker Compose* [8]. Esta ferramenta permite utilizar um ficheiro *YAML* para definir e configurar todos os *containers* necessários à execução de uma determinada aplicação, e posteriormente, através de um único comando, executar ou parar todos estes *containers*.

Esta abordagem apresenta as seguintes desvantagens:

- Como o *core* do sistema operativo da máquina física é partilhado, caso exista alguma falha de segurança, esta falha também vai afetar todos os *containers*;
- Devido ao *core* do sistema operativo da máquina física ser partilhado com os *containers*, não é possível ter *containers* a executar um sistema

operativo diferente daquele que está a ser executado na máquina física.

Capítulo 4

Análise e Implementação

Neste capítulo é feita uma análise do problema a resolver, e são abordados em maior detalhe os requisitos, as ferramentas e as tecnologias selecionadas para resolver esse problema. É também detalhada a arquitetura do sistema, o processo de integração com o *software* do Hospital, e o processo de *deployment* para o futuro desenvolvimento de novas funcionalidades ou correção de erros.

4.1 Análise do Problema

A gestão da consulta externa é um tema complexo que comporta vários desafios. É essencial conseguir manter um registo preciso do estado de todos os pedidos de consulta, bem como da disponibilidade dos médicos e gabinetes onde estas consultas se possam vir a realizar.

No caso dos hospitais públicos, é comum utilizar *software* desenvolvido pelo SPMS (Serviços Partilhados do Ministério da Saúde), que nem sempre está em dia com os padrões tecnológicos mais recentes.

Atualmente, no Hospital de Évora, os agendamentos de consultas externas são realizados manualmente pelos funcionários administrativos na aplicação informática SONHO, que é disponibilizada pelo SPMS. Para além disso, esta aplicação também é responsável por registar as agendas médicas, bem como a disponibilidade dos gabinetes para consulta.

4.2 Terminologia e Enquadramento

De forma a melhor compreender os desafios e soluções abordados nesta dissertação, é necessário perceber os conceitos principais em torno da gestão da consulta externa. São estes a aplicação informática SONHO, a designação dos vários tipos de consultas, os processos de criação e alteração das agendas médicas, e as diferentes particularidades dos gabinetes médicos.

4.2.1 SONHO

A aplicação SONHO (Sistema Integrado de Informação Hospitalar) [49] foi desenvolvida na década de 90 com o objetivo de dar suporte ao serviço administrativo dos hospitais, assegurar o controlo da produção e da faturação, e permitir a exportação de informação para indicadores estatísticos.

Existem atualmente duas versões da aplicação. O SONHO V1 e o SONHO V2, sendo que o SONHO V1 se encontra em substituição gradual pelo SONHO V2 que é tecnologicamente e funcionalmente mais adequado às necessidades atuais.

O SONHO V2 visa responder aos problemas técnicos sentidos pelas unidades hospitalares, resultantes da obsolescência da aplicação SONHO V1.

Por forma a assegurar capacidade de evolução para novas funcionalidades e escalabilidade, o SONHO V2 comporta:

- Migração tecnológica para *Oracle Database 11g R2*, *forms* e *reports*, garantindo maior alinhamento com a generalidade de aplicações no mercado;
- Desenvolvimento de uma nova camada de integração (*service oriented*);
- Disponibilização de uma base de dados de *Reporting*, a partir da qual se podem obter relatórios de gestão, retirando carga da base de dados operacional;
- Novas pequenas funcionalidades, como a possibilidade de utilização do cartão de cidadão na identificação do utente.

4.2.2 Consultas

De acordo com a origem das consultas, estas podem ser divididas em dois tipos: *primeiras* ou *segundas* (também designadas subsequentes). As primeiras consultas são aquelas em que o utente vai ser consultado pela primeira

vez para uma determinada especialidade. Quando, no final de uma consulta, o médico pede para voltar a ver o utente, a consulta resultante deste pedido vai ser designada de segunda consulta. As segundas consultas são todas as que não são primeiras, ou seja, todas as consultas depois da primeira são designadas segundas.

Os pedidos de consulta podem ainda ser divididos entre dois tipos: rotina e urgentes. Os pedidos de rotina são todos aqueles que são planeados, enquanto que os urgentes são aqueles que ocorrem a partir de situações de emergência em que é necessária uma consulta não planeada.

Durante o ciclo de vida da consulta, são registadas as datas relativas a cinco momentos. O primeiro momento é a data da receção do pedido da consulta por parte do Hospital. Posteriormente é registada a data em que o pedido é enviado pelos funcionários administrativos para o médico triador, que é responsável por avaliar a prioridade do pedido. Depois desta avaliação, é também registada a data em que o pedido regressa para os funcionários administrativos, antes de ser feito o agendamento. Finalmente, aquando do agendamento, é registada a data da marcação e a data planeada para a consulta.

Dependendo da origem do pedido, a atribuição da prioridade pode ser diferente. Os pedidos provenientes dos cuidados de saúde primários são criados automaticamente na aplicação SONHO a partir de outra aplicação também utilizada no Hospital denominada *Alert P1*, em que já vêm com a prioridade atribuída. Esta prioridade pode ter um de três valores: muito prioritário, prioritário ou normal.

4.2.3 Agendas médicas

As agendas médicas são criadas pelos funcionários administrativos, na aplicação SONHO, a pedido dos médicos. Cada agenda é representada por parâmetros gerais relativos à própria agenda, e uma lista de parâmetros detalhados que representam as várias vagas possíveis nessa agenda.

Os parâmetros gerais de uma agenda são compostos pelo médico à qual pertence, pela sua especialidade, pelo gabinete onde a consulta vai acontecer, pela data de início e fim da agenda, pela hora de início e fim de cada dia de consulta, pelos dias da semana em que o médico vai dar consulta, pela demora média de cada consulta, e finalmente pelo tipo de consultas a que a agenda se refere.

Não existe limite para a data de início e fim da agenda, sendo que esta pode ser criada, por exemplo, anualmente, ou para um único dia. Caso a agenda seja criada para mais do que um dia, os dias em que o médio vai

dar consulta são fixos, tal como o horário para cada dia e o gabinete onde a consulta vai acontecer. A agenda pode ser criada apenas para primeiras consultas, apenas para segundas consultas ou para primeiras e segundas consultas simultaneamente. A demora média pode ser diferente para primeiras e segundas consultas.

Relativamente ao detalhe das agendas, a aplicação SONHO gera automaticamente os vários registos relativos às vagas de cada agenda. Por exemplo, para uma agenda de primeiras consultas, com a duração de uma semana, em que os dias de consulta são segunda-feira e sexta-feira, entre as 9:00h e as 12:00h, com demora média de 30 minutos, vão ser gerados 12 registos de vagas para esta agenda. Neste exemplo, 6 dos 12 registos pertencem às 6 vagas de 30 minutos na segunda-feira, e os restantes 6 pertencem ao período homólogo na sexta-feira. A aplicação SONHO também guarda em cada registo de vaga informação sobre se esta ainda está disponível ou se já foi utilizada.

As agendas médicas também podem ser apagadas a qualquer momento se o médico assim o entender, deixando de estar disponíveis para pedidos de consultas e libertando os gabinetes às quais estavam associadas.

4.2.4 Gabinetes

Os gabinetes médicos estão distribuídos pelas várias zonas do Hospital. Para além dos gabinetes de uso geral, existem ainda gabinetes específicos para um determinado tipo de especialidade quando há necessidade que o gabinete disponha de equipamento médico especializado.

Como referido anteriormente, os gabinetes são atribuídos aos médicos aquando da criação das agendas. Por norma, os médicos mantêm os mesmos gabinetes, mas também lhes podem ser atribuídos gabinetes distintos, por exemplo no caso de consultas urgentes ou por questões de disponibilidade.

À data da escrita desta dissertação, o Hospital de Évora conta com 66 gabinetes, distribuídos por 7 zonas, o que faz com que a tarefa de gestão da ocupação e disponibilidade destes gabinetes seja um dos maiores desafios do Hospital.

4.3 Proposta de Implementação

O processo atual, assente na aplicação SONHO V1, tem algumas limitações que fazem com que a gestão da consulta externa e dos gabinetes envolva bastante trabalho manual por parte dos funcionários administrativos.

Muitas destas limitações resultam do facto de a aplicação SONHO se ter mantido, de uma forma geral, inalterada desde o seu lançamento em 1994. A sua interface de utilizador está desatualizada em função dos padrões atuais.

Outro fator limitador na aplicação SONHO é a falta de normalização do modelo de dados presente na sua base de dados, o que faz com que seja difícil relacionar e cruzar a informação.

Esta limitação torna-se mais grave considerando o elevado volume de dados que a base de dados da aplicação SONHO contém. À data de 15 de Maio de 2020, o conteúdo da base de dados partilhado pelo Hospital tinha a seguinte dimensão: 1911 registos de agendas médicas abertas até à data, apenas para o ano de 2020, o que resulta em 154065 registos de vagas; 20490 registos de consultas marcadas ainda não efetivadas e 4842 registos de desmarcações a partir da data referida; uma média por ano de 44660 registos de pedidos de primeiras consultas, 48504 registos de consultas urgentes e 160103 registos de consultas registadas com agendamento, considerando os dados relativos aos últimos dois anos.

Para superar estas limitações e implementar as melhorias identificadas anteriormente, decidiu-se criar uma nova aplicação. Esta aplicação é composta por duas componentes. Uma componente de *backend* e outra componente de *frontend*.

A componente de *backend* é responsável por gerir a base de dados interna da aplicação, por executar a lógica necessária para a sincronização desta base de dados com a aplicação SONHO, e por processar e agregar todos os dados a serem apresentados ao utilizador.

Por outro lado, a componente de *frontend* é responsável por mostrar os dados recebidos do *backend* de uma forma percetível e intuitiva para o utilizador. É também responsável por recolher as interações do utilizador e transmiti-las ao *backend*.

Para além destas duas componentes, foi também necessário identificar o ambiente onde esta aplicação vai ser executada.

4.4 Definição de Requisitos

Os requisitos da aplicação a desenvolver dividem-se em quatro temas principais: gestão da lista de espera (tabela 4.1), gestão das consultas (tabela 4.2), gestão dos gabinetes (tabela 4.3) e gestão das agendas (tabela 4.4).

#	**Descrição**
R1.1	Visualizar a lista dos utentes em espera para consulta (com ou sem marcação)
R1.2	Visualizar a proveniência do pedido de consulta (publico, privado, etc.)
R1.3	Visualizar detalhe sobre o utente (nome, género, idade, localidade e número de telefone)
R1.4	Ordenação automática da lista dos utentes com base na sua prioridade
R1.5	Possibilidade de ordenar a lista dos utentes com base em outros critérios (data da marcação, proveniência, etc.)
R1.6	Possibilidade de filtrar a lista dos utentes (nome do médico, especialidade, etc.)
R1.7	Possibilidade de gerar de forma automática sugestões de agendamento para um determinado utente
R1.8	Possibilidade de gerar de forma automática sugestões de agendamento para uma lista de utentes

Tabela 4.1: Requisitos para a gestão da lista de espera

#	**Descrição**
R2.1	Visualizar a lista das consultas agendadas
R2.2	Possibilidade de ordenar a lista das consultas com base em outros critérios (data da marcação, data da consulta, etc.)
R2.3	Possibilidade de filtrar a lista das consultas (nome do médico, nome do gabinete, etc.)

Tabela 4.2: Requisitos para a gestão das consultas

#	**Descrição**
R3.1	Visualizar os gabinetes médicos
R3.2	Possibilidade de filtrar os gabinetes (zona, especialidade, etc.)
R3.3	Visualizar as consultas agendadas em cada gabinete num formato de agenda/calendário
R3.4	Visualizar os agendamentos disponíveis em cada gabinete num formato de agenda/calendário
R3.5	Possibilidade de obter uma vista de detalhe de cada intervalo de tempo da agenda/calendário com informação relevante sobre esse intervalo (utente, médico, agenda, etc.)
R3.6	Sinalização automática na agenda/calendário de intervalos de tempo onde existe a possibilidade de agendamento de uma consulta
R3.7	Possibilidade de selecionar o intervalo de tempo da agenda/calendário (diário, semanal, mensal)
R3.8	Possibilidade de gerar de forma automática uma sugestão de agendamentos para um determinado intervalo de tempo, com base nas agendas médicas e na lista de espera priorizada

Tabela 4.3: Requisitos para a gestão dos gabinetes

#	**Descrição**
R4.1	Visualizar as agendas médicas
R4.2	Possibilidade de filtrar as agendas (médico, especialidade, data, etc.)
R4.3	Possibilidade de gerar de forma automática uma sugestão de agendamentos para uma determinada agenda, com base na lista de espera priorizada e nos gabinetes disponíveis

Tabela 4.4: Requisitos para a gestão das agendas

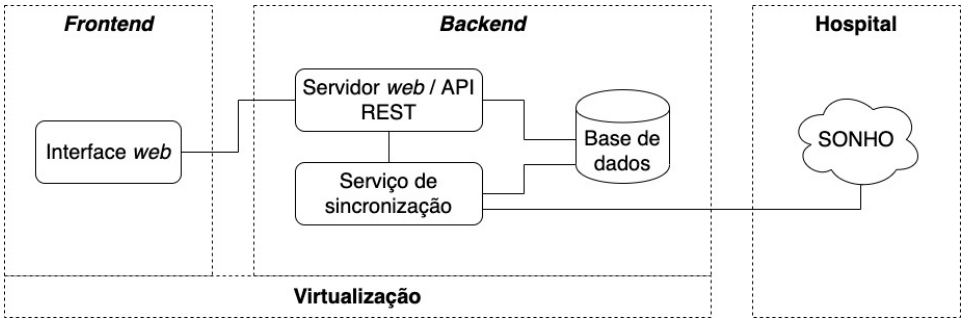

Figura 4.1: Arquitetura da aplicação

4.5 Arquitetura do Sistema

Uma visão geral da arquitetura da aplicação desenvolvida pode ser encontrada na figura 4.1.

Nesta figura pode ver-se que toda a aplicação desenvolvida está assente sobre uma camada de virtualização, que vai ser responsável pela execução da componente de *frontend* e de *backend* da aplicação. Independente desta camada está também a infraestrutura do Hospital onde está a ser executado o SONHO.

A componente de *frontend* vai ser responsável por disponibilizar a interface através da qual o utilizador vai interagir com a aplicação. Esta componente comunica com a componente de *backend* através de uma API REST.

A componente de *backend* engloba três funcionalidades: o servidor *web* que disponibiliza a API REST, a base de dados interna da aplicação, e o serviço de sincronização que é responsável por fazer a sincronização dos dados entre a base de dados interna e o SONHO.

Como se pode ver, o servidor *web* que disponibiliza a API REST não comunica diretamente com o SONHO. Este apenas comunica com a base de dados interna da aplicação e com o serviço de sincronização, que por sua vez vai comunicar com o SONHO.

De entre as tecnologias analisadas no capítulo anterior, foi escolhida uma para cada componente da aplicação. Esta escolha foi feita com base nas características e funcionalidades de cada uma das tecnologias, na forma como essas características se adaptam aos requisitos da aplicação a desenvolver, e também tendo em conta a preferência e experiência pessoal.

As subsecções seguintes detalham cada uma destas componentes.

4.5.1 *Backend*

A componente de *backend* divide-se em duas subcomponentes: o servidor *web* e a base de dados.

Servidor *web*

Para o servidor *web* decidiu-se utilizar o *Node.js*, devido à sua facilidade de utilização, versatilidade e aptidão para o desenvolvimento de aplicações *web*.

O servidor *web* é responsável por fornecer ao *frontend* informação sobre os pedidos de consulta, as consultas agendadas e efetuadas, os gabinetes e as agendas médicas. Esta informação é fornecida sobre a forma de várias API REST.

Para além de fornecer informação, o servidor *web* também disponibiliza duas APIs para simular e sugerir possíveis agendamentos. Uma das APIs sugere possíveis consultas para uma lista de utentes em lista de espera, e a outra sugere utentes em espera para uma lista de agendas médicas disponíveis.

Outra funcionalidade presente no servidor *web* é o módulo de sincronização. Este é responsável por aceder aos dados presentes na aplicação SONHO e processá-los para serem guardados na base de dados interna da aplicação.

Devido à falta de informação por parte do Hospital relativamente ao modelo de dados da aplicação SONHO, bem como dos mecanismos disponíveis para estabelecer uma ligação para obter os dados, o desenvolvimento deste módulo não foi contemplado durante a realização desta dissertação, ficando assim o seu desenvolvimento referenciado como trabalho futuro.

Base de dados

Para a base de dados interna da aplicação foi escolhido o *PostgreSQL*, devido à sua popularidade, a ser de código aberto e a ter todas as suas funcionalidades disponíveis gratuitamente.

De forma a suportar todas as funcionalidades da aplicação, foi implementado o modelo de dados indicado na figura 4.2. Este modelo de dados conta com as seguintes tabelas:

user Tabela utilizada internamente pela aplicação para suportar o processo de *login*.

zone Tabela utilizada para guardar a informação sobre as várias zonas pelas quais os gabinetes se dividem.

office Nesta tabela é guardada a informação sobre todos os gabinetes do hospital. Esta tabela tem uma ligação com a tabela anterior de forma a relacionar os gabinetes com as suas respetivas zonas.

specialty Nesta tabela são guardadas informações sobre todas as especialidades de consulta existentes no hospital.

doctor Nesta tabela são guardadas informações sobre todos os médicos que dão consulta no hospital.

doctor_specialty Esta tabela é utilizada para relacionar um médico a uma ou mais especialidades, uma vez que um único médico pode dar consultas de várias especialidades.

schedule Nesta tabela são guardadas as agendas médicas. Cada registo de agenda está relacionado com um gabinete e um médico, numa determinada especialidade.

scheduletime Nesta tabela são guardadas as vagas relacionadas com cada agenda.

patient Esta tabela é utilizada para guardar a informação sobre os utentes do hospital.

consultation Nesta tabela são guardados todos os pedidos de consulta relacionados com o utente. No caso de a consulta já ter um agendamento, são também guardadas as respetivas relações com o médico, a especialidade e a vaga que corresponde à data da consulta.

4.5.2 *Frontend*

Para o *frontend* da aplicação, decidiu-se utilizar uma *framework JavaScript* em vez de uma abordagem *web* tradicional, uma vez que facilita e acelera o processo de desenvolvimento. De entre as opções analisadas, foi escolhida a

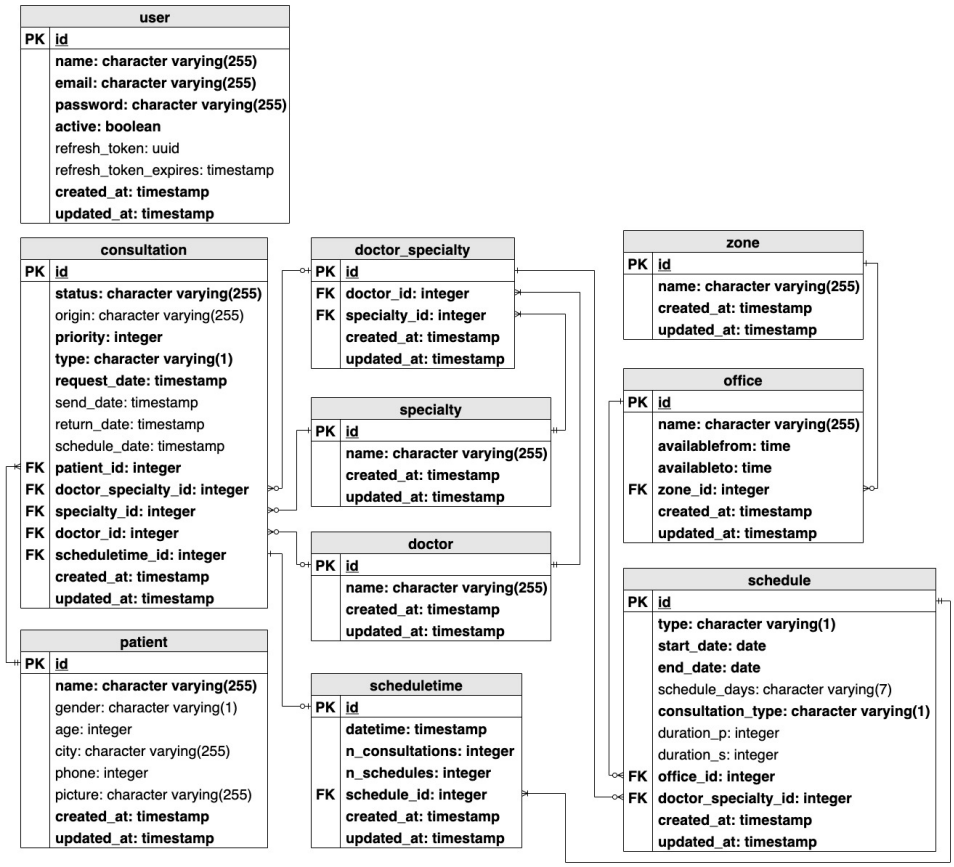

Figura 4.2: Modelo de dados

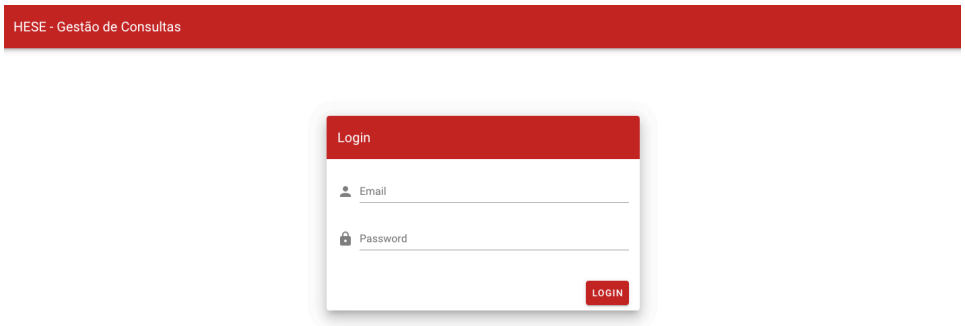

Figura 4.3: *Frontend - Login*

framework Vue.js, devido à sua popularidade crescente e ao facto de combinar as melhores características dos seus antecessores.

Para cumprir os requisitos da aplicação, foram criadas seis páginas: *login*, alteração de *password*, início, consultas, gabinetes e agendas.

Login. A funcionalidade de *login* (figura 4.3) não está propriamente descrita nos requisitos da aplicação, mas foi desenvolvida uma vez que é essencial para impedir acesso não devido à aplicação e aos seus dados. Esta página é composta por um formulário onde é necessário introduzir um email autorizado e a palavra passe correspondente.

Alteração de *password*. Para complementar a funcionalidade de *login*, foi também criada uma página para realizar a alteração da *password* do utilizador (figura 4.4). Esta página é composta por um formulário onde o utilizador deve introduzir a sua *password* atual, a nova *password* pretendida, e repetir esta nova *password* para prevenir um possível erro do utilizador na introdução da nova *password*.

No final do processo, o utilizador recebe uma mensagem de sucesso se a alteração da *password* for sucedida, ou uma mensagem de erro no caso de não ter sido possível alterar a *password*.

Início. A página de início (figura 4.5) também não está descrita nos requisitos da aplicação, mas foi desenvolvida com o objetivo de dar uma pequena introdução e explicação ao utilizador das funcionalidades que estão disponíveis na aplicação.

Figura 4.4: *Frontend* - Alteração da *password*

Figura 4.5: *Frontend* - Início

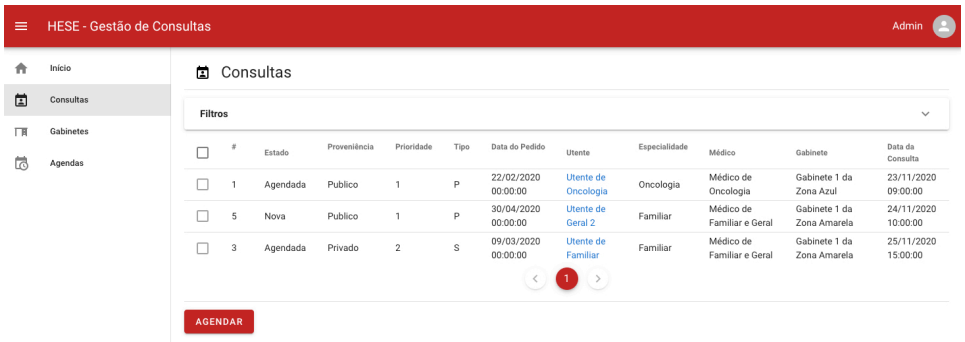

Figura 4.6: *Frontend* - Consultas

Consultas. A página das consultas (figura 4.6) engloba os requisitos de gestão da lista de espera e gestão das consultas descritos nas tabelas 4.1 e 4.2, respetivamente.

Esta página mostra uma tabela com a listagem de todos os pedidos de consultas, com e sem marcação. A tabela permite ordenar a informação por uma ou mais colunas. Existe também uma secção de filtros onde é possível filtrar os registos apresentados na tabela com base nos valores de algumas colunas.

Há ainda a possibilidade de selecionar um ou mais registos de pedidos de consulta (sem marcação) e gerar uma sugestão de agendamentos para esses pedidos. Essa sugestão é mostrada ao utilizador através de um *pop-up*.

Gabinetes. A página dos gabinetes (figura 4.7) contempla os requisitos de gestão de gabinetes descritos na tabela 4.3.

Esta página mostra uma tabela com todos os gabinetes disponíveis no Hospital e pode ser ordenada e filtrada tal como a tabela das consultas.

Cada linha, que corresponde a um gabinete, pode ser expandida para revelar uma agenda/calendário com todos os agendamentos e vagas relacionados. De forma a ser mais fácil identificar os agendamentos e as vagas, os agendamentos que têm uma consulta associada são mostrados a verde, enquanto que as vagas são mostradas a cinzento. É também possível selecionar cada um destes agendamentos e vagas, e visualizar mais informação.

Para as vagas ainda sem consulta associada, é possível gerar uma sugestão de agendamento. Essa sugestão é mostrada ao utilizador através de um *pop-up*.

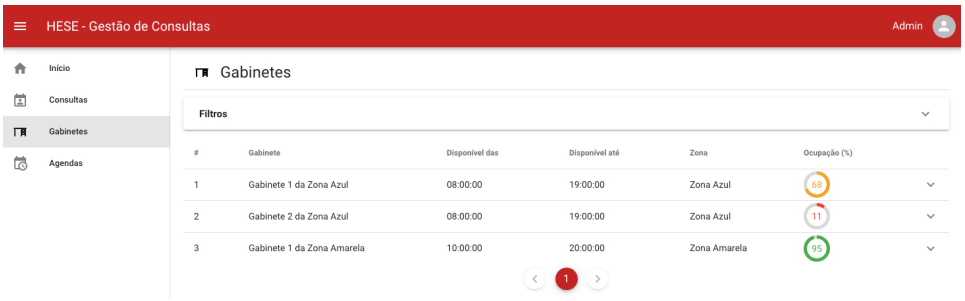

Figura 4.7: *Frontend* - Gabinetes

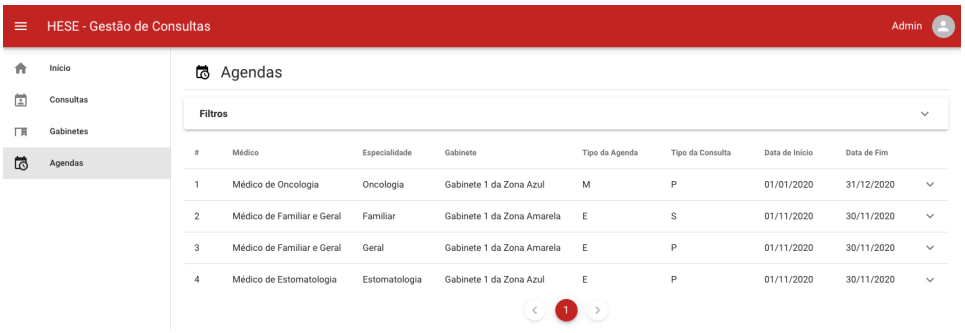

Figura 4.8: *Frontend* - Agendas

Agendas. A página das agendas (figura 4.8) contempla os requisitos de gestão das agendas médicas descritas na tabela 4.4.

Nesta página pode ser encontrada uma tabela com todas as agendas médicas disponíveis. Tal como nas tabelas anteriores, esta também pode ser ordenada e filtrada.

É também possível expandir cada uma das linhas da tabela (que corresponde a uma agenda) e revelar todas as datas e horas das vagas disponíveis para esta agenda.

Para as vagas ainda sem consulta associada, é possível gerar uma sugestão de agendamento. Essa sugestão é mostrada ao utilizador através de um *pop-up*.

4.5.3 Ambiente de execução

Para o ambiente de execução decidiu-se utilizar um ambiente virtual. De entre as opções analisadas, escolheu-lhe utilizar o *Docker*, uma vez que não é tão exigente do ponto de vista dos recursos de *hardware*, e permite uma

rápida instalação e execução de todo o *software* necessário para o funcionamento da aplicação.

Tirando partido do *Docker Compose*, foram criadas duas configurações diferentes: uma para o ambiente de desenvolvimento e outra para o ambiente de produção. Estas configurações especificam os *containers* que vão ser executados em cada um dos ambientes, e qual o *software* e configurações específicas de cada *software* instalado.

Para a criação destes *containers* foram utilizadas as imagens que se indicam de seguida. Todas estas images estão disponíveis publicamente e gratuitamente no *Docker Hub*.

node:14.15.0-alpine Imagem oficial do *Node.js*. Na versão escolhida, inclui a versão 14.15.0 do *Node.js*, que é a última versão LTS à data de escrita desta dissertação. Esta imagem tem como base a imagem *Alpine*, que é uma distribuição *Linux* bastante compacta e minimalista.

Na aplicação desenvolvida, esta imagem é utilizada como base para dois *containers*: o *container* responsável pelo servidor *web*, e o *container* responsável pelo *frontend*.

postgres:13.0-alpine Imagem oficial do *PostgreSQL*, na versão 13.0, que é a última versão disponível à data de escrita desta dissertação. Esta imagem é também baseada na imagem *Alpine*.

nginx:1.18.0-alpine No ambiente de desenvolvimento, o próprio *Vue.js* executa um servidor *web* que disponibiliza a componente do *frontend* da aplicação, mas no ambiente de produção, é necessário um servidor dedicado para este efeito. Para isso, foi utilizado o *nginx* [33], através da sua imagem oficial, na versão 1.18.0, que é a última versão disponível à data de escrita desta dissertação. Esta imagem é também baseada na imagem *Alpine*.

dpage/pgadmin4 De forma a gerir e visualizar a base de dados interna da aplicação, foi utilizado o *pgAdmin* [41]. Este é disponibilizado através de uma imagem não oficial criada pela comunidade, na versão 4.12.

4.6 Integração com os Sistemas do Hospital

Como já referido, toda a informação necessária para cumprir os requisitos da aplicação encontra-se num único sistema do Hospital: o SONHO. Assim, foi necessário definir de que forma a aplicação desenvolvida vai aceder a estes dados.

Para mitigar o problema de não ter sido possível obter da parte do Hospital uma forma de aceder diretamente ao SONHO, os Serviços de Informática do Hospital apresentaram duas opções.

A primeira opção passaria por tentar encontrar uma forma de fazer este acesso ao SONHO de forma direta. Seria então responsabilidade do Hospital fazer esta investigação e apresentar uma solução, o que poderia demorar tempo indeterminado e, além disso, não se conseguir chegar a um resultado concreto.

Na segunda opção, os dados seriam obtidos através de um processo manual. Este processo já acontece atualmente no Hospital para outras finalidades, sendo possível extrair os dados para diversos formatos. Assim, nesta opção, ficava a cargo do Hospital realizar a extração dos dados do SONHO e importá-los na aplicação, com a periodicidade que entendessem.

A opção ideal seria, obviamente, a primeira, uma vez que se conseguiria realizar o processo de acesso aos dados de forma totalmente automática. Além disso, para permitir realizar a segunda opção seria necessário implementar na aplicação algum tipo de funcionalidade para possibilitar a importação dos dados.

Como na primeira opção não é possível ter garantia de quando, ou se, vai ser possível chegar a um resultado concreto, foi necessário encontrar outra alternativa. Decidiu-se, então, tentar encontrar um meio termo entre as duas opções. Para isso, a solução apresentada passa por implementar uma componente de sincronização de dados na aplicação. Essa componente é responsável por obter os dados de uma localização a acordar (por exemplo, uma pasta especifica onde seriam colocados os resultados da exportação manual do SONHO) e transformá-los no formato utilizado pela base de dados interna da aplicação.

A vantagem de utilizar esta componente de sincronização é que esta pode ser construída de forma a ficar preparada para facilmente alternar entre obter os dados de uma localização específica, ou através de uma eventual ligação direta ao SONHO, quando esta estiver disponível.

Uma vez que esta componente de sincronização depende fortemente do modelo de dados do SONHO, e como não foi possível obter uma visão detalhada

deste modelo, nem acesso a todos os dados necessários, esta componente não foi desenvolvida nesta dissertação.

4.7 *Deploy*

Designa-se *deploy* ao processo de mover alterações desenvolvidas no ambiente de desenvolvimento para o ambiente de produção.

Como explicado anteriormente, os ambientes criados para a aplicação desenvolvida têm como base *containers Docker*. Os *containers* necessários a cada ambiente estão especificados nos ficheiros de configuração do *Docker Compose*, tal como a forma como alguns deste *containers* são construídos.

Existem dois *containers* que têm processos de construção particularmente distintos nos dois ambientes. São estes os *containers* de *Node.js* que servem de suporte ao servidor *web* e à componente de *frontend*.

No ambiente de desenvolvimento pretende-se ter uma experiência de desenvolvimento contínua, em que a aplicação muda em tempo real em função das alterações que são feitas no código. Esta seria a experiência normal de desenvolvimento caso não fosse utilizado o *Docker*. Para manter a mesma experiência no *Docker*, foi utilizada uma funcionalidade que permite que o sistema de ficheiros da máquina principal seja partilhado em tempo real com os *containers*. Desta forma, os *containers Docker* vão detetar as alterações feitas na máquina principal e refleti-las imediatamente nas aplicações que estes estão a executar.

Por outro lado, no ambiente de produção pretende-se uma experiência menos volátil, ou seja, pretende-se que as alterações que este sofre sejam precisas e previamente planeadas. Assim, não existe a partilha em tempo real do sistema de ficheiros da máquina principal com os *containers*. Para mover o código desenvolvido para dentro dos *containers* utiliza-se outra funcionalidade do *Docker* que permite a cópia de ficheiros da máquina principal para os *containers* aquando da criação dos mesmos. Desta forma, o conteúdo dos *containers* mantêm-se inalterado até que estes sejam recriados.

Existe também uma particularidade relativamente ao *container* que executa o *PostgreSQL* que é importante realçar. No *PostgreSQL*, existe uma pasta predefinida onde é guardado todo o conteúdo da base de dados. De forma a garantir que este conteúdo não se perde quando o *container* é reiniciado, é utilizada a mesma funcionalidade do *Docker* de partilha do sistema de ficheiros da máquina principal descrita anteriormente. Assim, uma pasta do sistema de ficheiros da máquina principal é partilhada com o *container* precisamente na localização da pasta predefinida pelo *PostgreSQL*, fazendo

com que quando o *PostgreSQL* atualize o conteúdo desta pasta predefinida, esta atualização ocorra na realidade na pasta da máquina principal. Desta forma, o conteúdo da base de dados fica sempre seguro na máquina principal. Além disso, quando o *container* for novamente iniciado, este conteúdo volta a ser partilhado com o *container* exatamente no mesmo estado em que estava quando o *container* foi parado pela última vez.

Posto isto, o processo de *deploy* pode passar pela simples recriação do ambiente de produção, depois das alterações estarem devidamente testadas no ambiente de desenvolvimento. Isto vai fazer com que as novas alterações sejam copiadas e executadas pelos *containers* que suportam o ambiente de produção.

Este processo de recriação é simples, rápido e seguro. E como visto anteriormente, não afeta o conteúdo da base de dados.

Capítulo 5

Casos de Uso

Neste capítulo são abordadas as diversas funcionalidades da aplicação. Estas funcionalidades têm como base os requisitos da aplicação propostos, e também algumas funcionalidades que foram consideradas importantes incluir.

5.1 Registo de utilizadores

Uma vez que a aplicação desenvolvida disponibiliza acesso a dados confidenciais, o acesso a estes dados necessita ser controlado. Desta forma, os utilizadores devem ser registados antes de poderem ter acesso à aplicação.

Por estas razões, não faz sentido que os utilizadores possam fazer o seu próprio registo. Assim, de forma a manter a segurança, os utilizadores são registados manualmente no *backend* da aplicação.

Existe, para isso, um processo manual que é necessário executar pelo administrador da aplicação diretamente no servidor *web* da aplicação de forma a criar os utilizadores na base de dados.

Cada registo de utilizador é composto por um nome de utilizador no formato de email, por uma password, por um atributo que define se o utilizador está ativo ou não, e por um *token* e a sua data de expiração que são utilizados na verificação da validade do último *login* do utilizador.

Neste registo inicial, o administrador da aplicação vai também atribuir uma *password* temporária aos utilizadores, que pode depois ser alterada por estes depois do primeiro *login*.

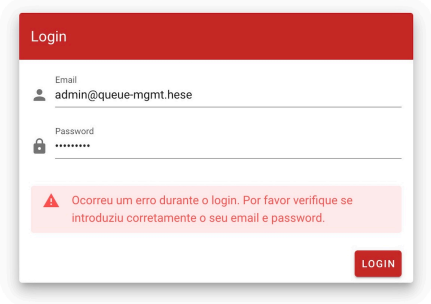

Figura 5.1: Caso de uso - Página de *login* com mensagem de erro

5.2 *Login*

A funcionalidade de login permite que apenas utilizadores autorizados consigam aceder à aplicação. Para que um utilizador esteja autorizado a aceder à aplicação, este tem que existir e estar ativo na base de dados da aplicação.

A página de *login* (figura 4.3) é a página inicial da aplicação. Qualquer utilizador que aceda à aplicação pela primeira vez vai-lhe ser apresentada esta página. E esta é a única página que o utilizador vai conseguir aceder antes de estar autorizado, mesmo que tente navegar para outra página, vai ser sempre redirecionado para a página de *login*.

Para realizar o *login* é necessário um nome de utilizador no formato de email, e a respetiva *password*. Se o email e a *password* fornecidos corresponderem com algum utilizador existente na base de dados, este vai ser autorizado a aceder à aplicação e vai ser redirecionado para a página inicial da mesma. Caso contrário, o utilizador vai permanecer na página de *login* e vai-lhe ser apresentada uma mensagem de erro (figura 5.1).

5.3 Alteração da *password*

Como a registo inicial do utilizador não é festo pelo mesmo, este pode decidir alterar a *password* que foi inicialmente criada. Para isso, existe no menu do utilizador a opção ***Alterar password***, como se pode ver na figura 5.2.

Depois de clicar nessa opção, o utilizador vai ser redirecionado para a página de alteração da password (figura 5.3). Nesta página o utilizador deve fornecer 3 informações: a *password* atual, a nova *password* pretendida, e a confirmação desta nova *password* de forma a minimizar possíveis erros.

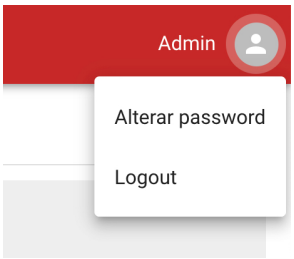

Figura 5.2: Caso de uso - Botão para alterar a *password*

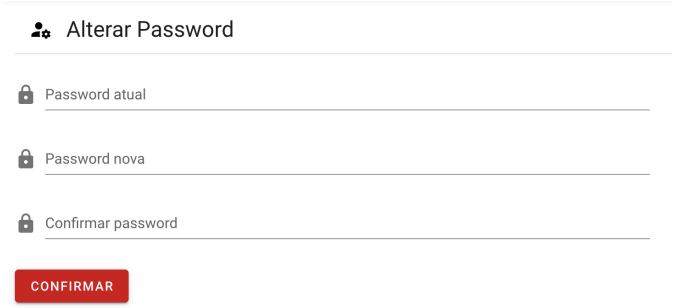

Figura 5.3: Caso de uso - Alterar *password*

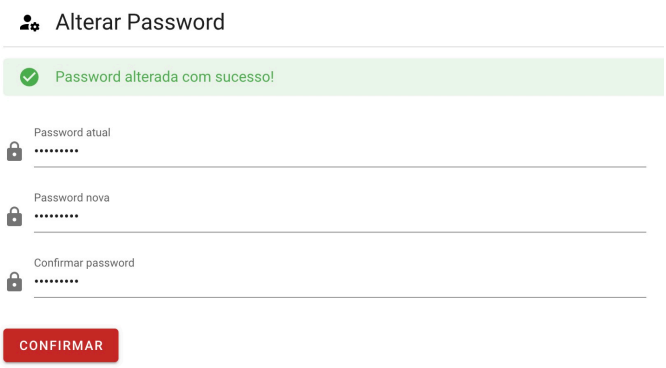

Figura 5.4: Caso de uso - Alterar *password* com mensagem de sucesso

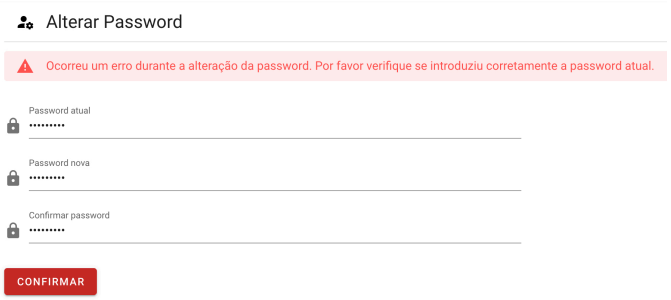

Figura 5.5: Caso de uso - Alterar *password* com mensagem de erro

Caso o utilizador introduza toda a informação corretamente e a password tenha sido alterada com sucesso, vai-lhe ser apresentada uma mensagem de sucesso (figura 5.4). Caso contrário, vai-lhe ser apresentada uma mensagem de erro (figura 5.5).

5.4 Gestão da lista de espera e das consultas

A gestão da lista de espera e das consultas são dois requisitos separados, mas que foram unificados numa única página, uma vez que a lista de espera é composta por pedidos de consulta sem marcação, e as consultas são pedidos de consulta com marcação.

Assim, foi criada uma página ***Consultas*** onde é possível visualizar os pedidos de consulta com e sem marcação (figura 4.6).

Nesta página é possível filtrar facilmente entre as consultas com e sem marcação através da coluna ***Estado***. Estes filtros podem ser aplicados expandindo o menu ***Filtros*** (figura 5.6).

Outra funcionalidade desta página é a sugestão de agendamentos. É possível selecionar pedidos de consulta ainda não agendadas (figura 5.7) e obter uma sugestão de agendamentos possíveis (figura 5.8), através do botão ***Agendar***.

5.5 Gestão dos gabinetes

Para realizar a gestão dos gabinetes foi criada a página ***Gabinetes*** (figura 4.7). Nesta página é possível visualizar uma lista com todos os gabinetes do hospital.

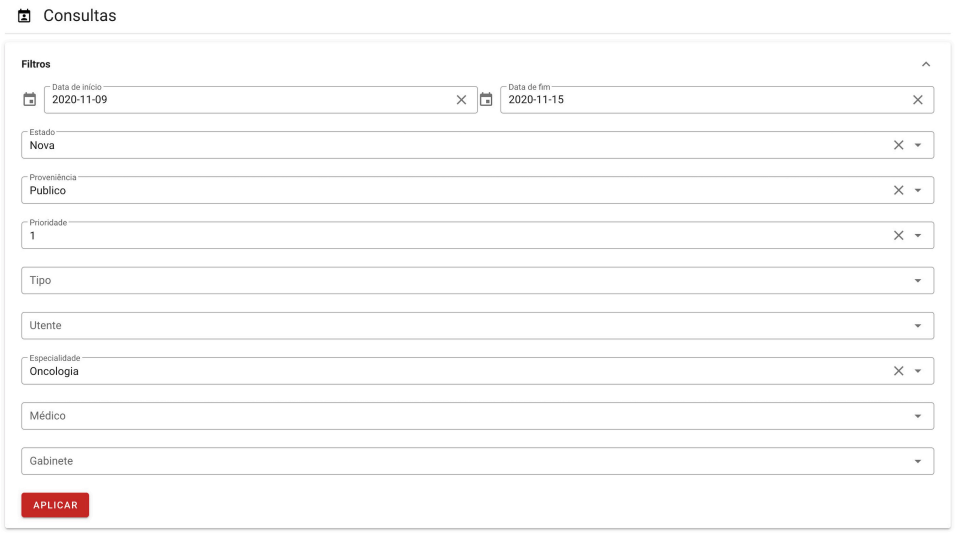

Figura 5.6: Caso de uso - Consultas - Filtros

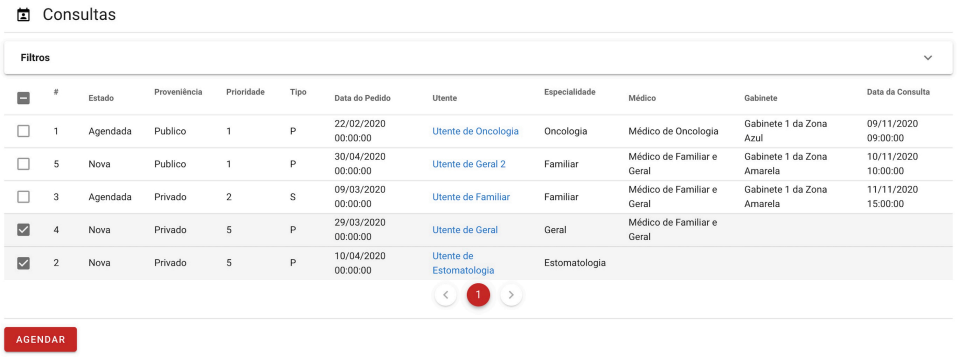

Figura 5.7: Caso de uso - Consultas - Seleção

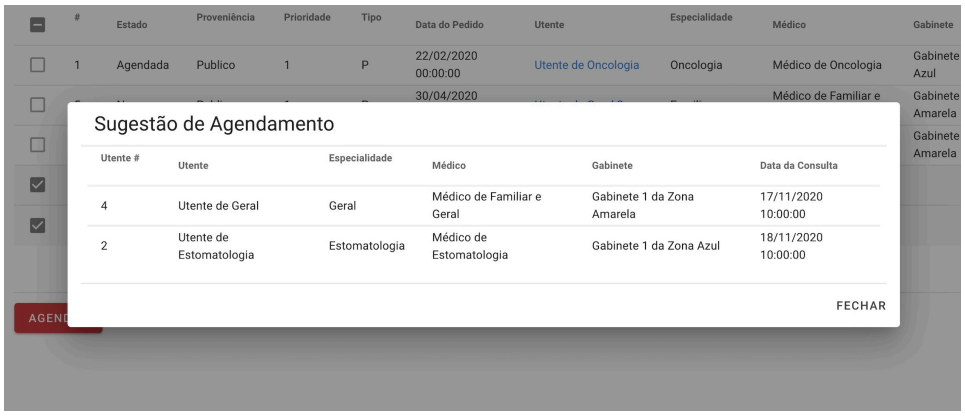

Figura 5.8: Caso de uso - Consultas - Sugestão de agendamento

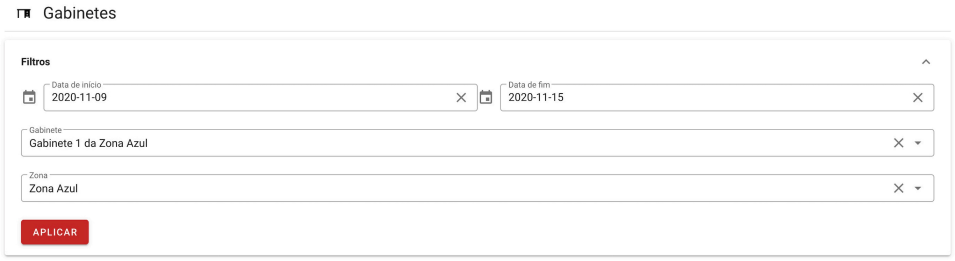

Figura 5.9: Caso de uso - Gabinetes - Filtros

Tal como na página das consultas, é também possível aplicar filtros na lista dos gabinetes com base em alguns parâmetros, como se pode ver na figura 5.9.

Para cada gabinete, é possível expandir o seu registo na tabela de forma a obter uma vista de detalhe de todas as consultas e vagas do mesmo (figura 5.10). Esta vista tem o formato de agenda/calendário, e para cada consulta ou vaga é possível obter mais informação clicando sobre as mesmas (figura 5.11).

No caso de uma vaga, é também possível obter uma sugestão de agendamento através do botão ***Agendar*** (figura 5.12).

5.6 Gestão das agendas

Para realizar a gestão das agendas médicas foi criada a página ***Agendas*** (figura 4.8). Nesta página é possível visualizar uma lista com todas as agendas médicas abertas no sistema.

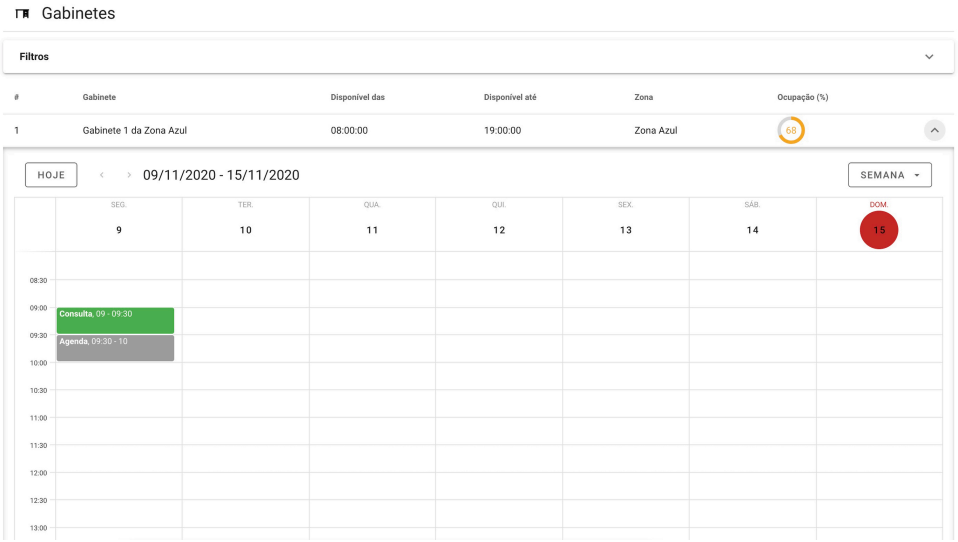

Figura 5.10: Caso de uso - Gabinetes - Agenda

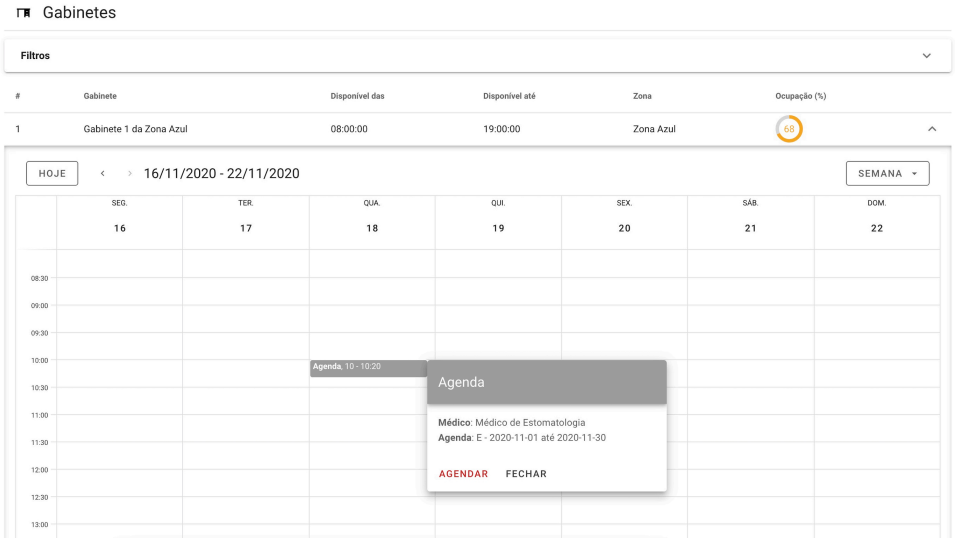

Figura 5.11: Caso de uso - Gabinetes - Detalhe da vaga

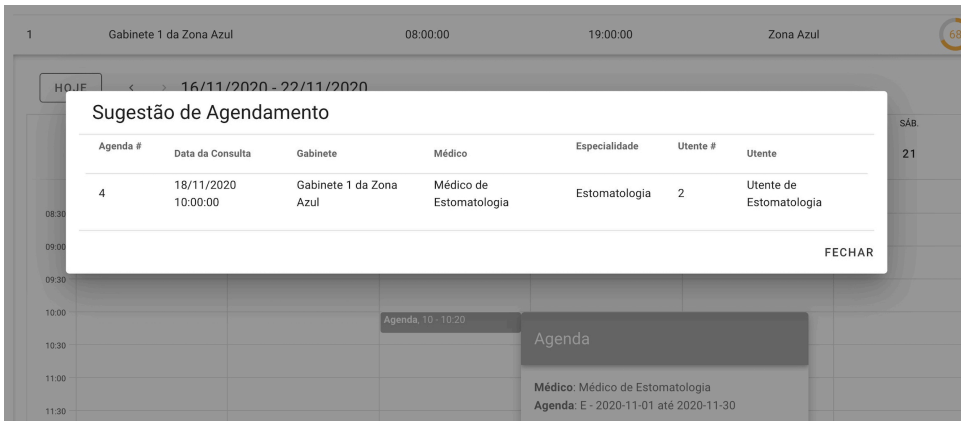

Figura 5.12: Caso de uso - Gabinetes - Sugestão de agendamento

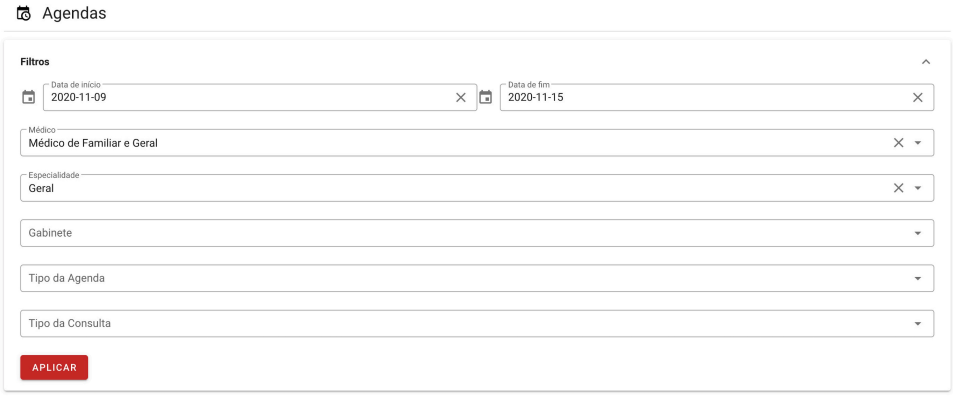

Figura 5.13: Caso de uso - Agendas - Filtros

Tal como nas páginas anteriores, é também possível aplicar filtros na lista das agendas com base em alguns parâmetros, como se pode ver na figura 5.13.

Para cada agenda, é possível obter uma vista de detalhe de todas as suas vagas através da expansão do seu registo na tabela (figura 5.14). Nesta vista, e caso as vagas ainda não estejam preenchidas, é também possível obter uma sugestão de agendamento, para uma ou mais vagas, através do botão ***Agendar*** (figura 5.15).

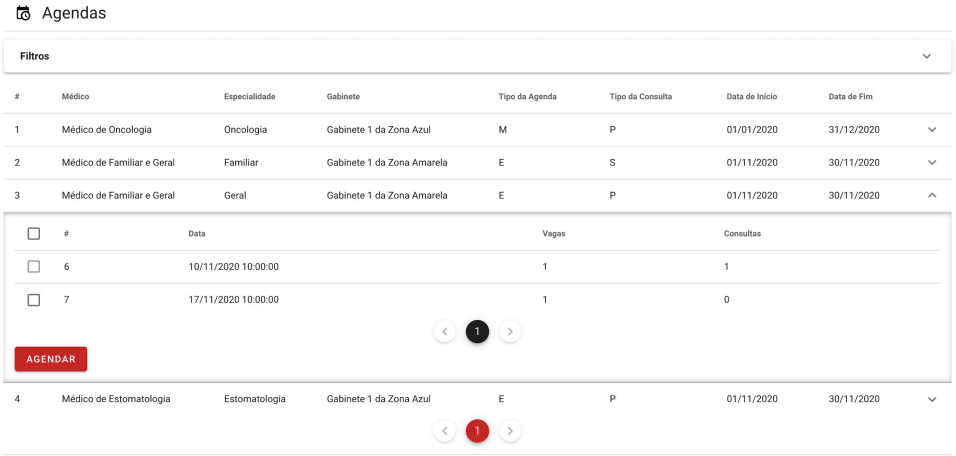

Figura 5.14: Caso de uso - Agendas - Vagas

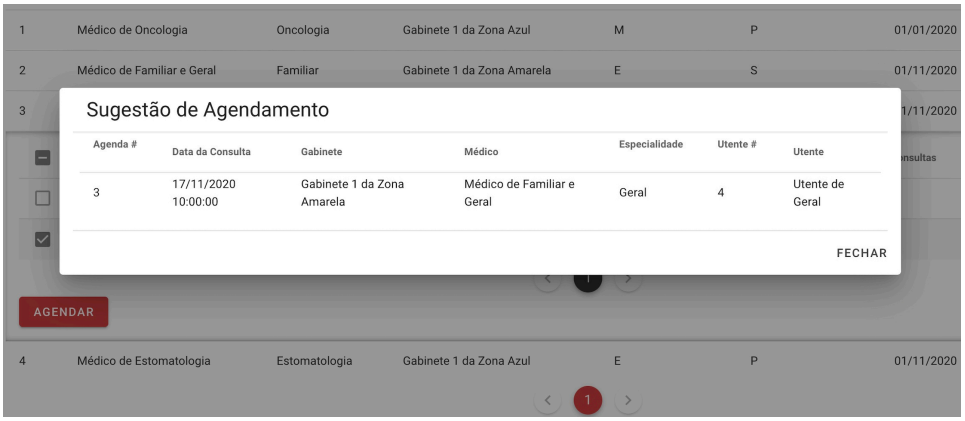

Figura 5.15: Caso de uso - Agendas - Sugestão de agendamento

Capítulo 6

Conclusões e Trabalho Futuro

Neste capítulo são apresentadas as conclusões obtidas relativamente ao trabalho realizado, e são apresentadas propostas de possível trabalho futuro.

6.1 Conclusões

O objetivo principal desta tese foi desenvolver uma ferramenta capaz de complementar e facilitar o trabalho manual atualmente existente no Hospital de Évora, no que diz respeito à gestão da lista de espera para primeiras consultas, incluindo a gestão do espaço físico da consulta e gestão das agendas médicas.

De forma a perceber quais os principais desafios e dificuldades existentes no Hospital, foram realizadas várias reuniões com elementos das equipas do Hospital, de forma a obter vários pontos de vista para o problema. Dessas reuniões resultou um conjunto de requisitos que foram novamente discutidos e aprovados pelo Hospital.

Posteriormente foi necessário manter um contacto próximo com os Serviços de Informática do Hospital, de modo a obter os detalhes técnicos sobre quais dos sistemas existentes no Hospital seriam necessários para cumprir os requisitos identificados, e sobre como aceder aos dados desses mesmos sistemas. No entanto, como explicado anteriormente, este processo de acesso aos dados não foi o ideal. Como resultado, foi necessário assumir um conjunto de pressupostos em relação aos dados de modo a conseguir avançar com o trabalho.

Assim, para ser colocada em produção, a ferramenta desenvolvida requer algum trabalho adicional, em colaboração com o Hospital, de forma a apro-

fundar o detalhe sobre o modelo de dados existente, e sobre qual a melhor forma de acesso aos dados.

Isto não invalida o facto de todos os objetivos terem sido cumpridos, umas vez que a ferramenta desenvolvida implementa todos os requisitos propostos, no que diz respeito à gestão da lista de espera para primeiras consultas, à gestão do espaço físico da consulta e à gestão das agendas médicas.

Além disso, devido à forma com a ferramenta foi desenvolvida, este trabalho adicional relacionado com os dados pode ser também desenvolvido com outros hospitais, permitindo assim que a ferramenta seja utilizada noutros hospitais com requisitos semelhantes.

6.2 Trabalho Futuro

Como já referido, o principal trabalho futuro consiste em aprofundar a colaboração com o Hospital para conseguir ter mais detalhe relativamente aos dados e à forma de como aceder aos mesmos. Posteriormente, seria também necessário um trabalho de adaptação da ferramenta para aceder e utilizar estes dados. Isto iria possibilitar o funcionamento da ferramenta com os dados reais do Hospital.

Posto isto, outra possibilidade de trabalho futuro seria a realização de testes de aceitação com os utilizadores finais da ferramenta, com o objetivo de identificar eventuais falhas ou possíveis melhoramentos para a mesma.

Finalmente, o último passo seria a instalação da ferramenta na infraestrutura informática do Hospital.

Bibliografia

[1] AngularJS Superheroic JavaScript MVW Framework. https://angularjs.org. Acedido em: 2020-11-18.

[2] Bahmni Open Source EMR | ThoughtWorks. https://www.bahmni.org. Acedido em: 2020-11-22.

[3] Bahmni Coalition Bahmni. https://www.bahmni.org/bahmni-coalition. Acedido em: 2020-11-22.

[4] CSS: Cascading Style Sheets | MDN. https://developer.mozilla.org/en-US/docs/Web/CSS. Acedido em: 2020-11-18.

[5] DB-Engines Ranking - popularity ranking of database management systems. https://db-engines.com/en/ranking. Acedido em: 2020-11-18.

[6] Digital Transformation - IT Workforce Solutions | Dell Technologies. https://www.delltechnologies.com. Acedido em: 2020-11-18.

[7] Empowering App Development for Developers | Docker. https://www.docker.com. Acedido em: 2020-11-18.

[8] Overview of Docker Compose | Docker Documentation. https://docs.docker.com/compose. Acedido em: 2020-11-18.

[9] Docker Hub. https://hub.docker.com. Acedido em: 2020-11-18.

[10] TC39 - ECMAScript. https://www.ecma-international.org/memento/tc39.htm. Acedido em: 2020-11-18.

[11] E. Elliott. Top JavaScript Frameworks and Topics to Learn in 2020 and the New Decade | by Eric Elliott | JavaScript Scene | Medium. https://medium.com/javascript-scene/top-javascript-frameworks-and-topics-to-learn-in-2020-and-the-new-decade-ced6e9d812f9. Acedido em: 2020-11-18.

[12] Open Source Cloud ERP Software - ERPNext. https://erpnext.com. Acedido em: 2020-11-22.

[13] F3M Information Systems, S.A. https://www.f3m.pt. Acedido em: 2020-11-22.

[14] F3M - GESTÃO DE UNIDADES DE SAÚDE. https://www.f3m.pt/pt/software/saude/gestao-de-unidades-de-saude. Acedido em: 2020-11-22.

[15] About Facebook. https://about.fb.com. Acedido em: 2020-11-18.

[16] Frappe: Framework + Apps. https://frappe.io. Acedido em: 2020-11-22.

[17] Glintt Homepage. https://www.glintt.com. Acedido em: 2020-11-21.

[18] GlobalCare. https://www.glintt.com/en/WHAT-WE-DO/Product-Services/SoftwareSolutions/Pages/GlobalCare.aspx. Acedido em: 2020-11-21.

[19] Hospital Management System - Globalcare. https://globalcare.glintt.com/hospital-management-system/. Acedido em: 2020-11-21.

[20] GNU Health | Freedom and Equity in Healthcare. https://www.gnuhealth.org. Acedido em: 2020-11-22.

[21] GNU Health/Introduction - Wikibooks, open books for an open world. https://en.wikibooks.org/wiki/GNU_Health/Introduction. Acedido em: 2020-11-22.

[22] GNU Solidario - Advancing Social Medicine. https://www.gnusolidario.org. Acedido em: 2020-11-22.

[23] Google - About Google, Our Culture Company News. https://about.google. Acedido em: 2020-11-18.

[24] Google Chrome. https://www.google.com/chrome/. Acedido em: 2020-11-18.

[25] HospitalRun - Open source software for developing world hospitals. https://hospitalrun.io. Acedido em: 2020-11-22.

[26] HTML: HyperText Markup Language | MDN. https://developer.mozilla.org/en-US/docs/Web/HTML. Acedido em: 2020-11-18.

[27] Java | Oracle. https://www.java.com. Acedido em: 2020-11-18.

[28] JavaScript | MDN. https://developer.mozilla.org/en-US/docs/Web/JavaScript. Acedido em: 2020-11-18.

[29] Microsoft Excel, Software de Folha de Cálculo. https://www.microsoft.com/pt-pt/microsoft-365/excel. Acedido em: 2020-11-18.

[30] The most popular database for modern apps | MongoDB. https://www.mongodb.com. Acedido em: 2020-11-18.

[31] MySQL. https://www.mysql.com. Acedido em: 2020-11-18.

[32] MySQL :: MySQL 8.0 Reference Manual :: 1.2.1 What is MySQL? https://dev.mysql.com/doc/refman/8.0/en/what-is-mysql.html. Acedido em: 2020-11-18.

[33] NGINX | High Performance Load Balancer, Web Server, Reverse Proxy. https://www.nginx.com. Acedido em: 2020-11-18.

[34] Node.js. https://nodejs.org/en/. Acedido em: 2020-11-18.

[35] Node.js - MDN Web Docs Glossary: Definitions of Web-related terms | MDN. https://developer.mozilla.org/en-US/docs/Glossary/Node.js. Acedido em: 2020-11-18.

[36] npm | build amazing things. https://www.npmjs.com. Acedido em: 2020-11-18.

[37] OpenMRS. https://openmrs.org. Acedido em: 2020-11-22.

[38] Oracle | Integrated Cloud Applications and Platform Services. https://www.oracle.com. Acedido em: 2020-11-18.

[39] Oracle and Sun Microsystems | Strategic Acquisitions | Oracle. https://www.oracle.com/sun/. Acedido em: 2020-11-18.

[40] ostezer and M. Drake. SQLite vs MySQL vs PostgreSQL: A Comparison Of Relational Database Management Systems | DigitalOcean. https://www.digitalocean.com/community/tutorials/sqlite-vs-mysql-vs-postgresql-a-comparison-of-relational-database-management-systems. Acedido em: 2020-11-18.

[41] pgAdmin - PostgreSQL Tools. https://www.pgadmin.org. Acedido em: 2020-11-18.

[42] PHP: Hypertext Preprocessor. https://www.php.net. Acedido em: 2020-11-18.

[43] PostgreSQL: The world's most advanced open source database. https://www.postgresql.org. Acedido em: 2020-11-18.

[44] PyPI - The Python Package Index. https://pypi.org. Acedido em: 2020-11-18.

[45] Welcome to Python.org. https://www.python.org. Acedido em: 2020-11-18.

[46] React A JavaScript library for building user interfaces. https://reactjs.org. Acedido em: 2020-11-18.

[47] Página Inicial - SISBIT. https://www.sisbit.pt. Acedido em: 2020-11-22.

[48] Sistemas de Informação em Saúde. http://im.med.up.pt/si_saude/si_saude.html. Acedido em: 2020-11-21.

[49] SONHO Hospitalar | Sistema Integrado de Informação Hospitalar SPMS. https://www.spms.min-saude.pt/2019/01/product-sclinicohospitalar. Acedido em: 2020-11-20.

[50] Spring Boot. https://spring.io/projects/spring-boot. Acedido em: 2020-11-18.

[51] V8 JavaScript engine. https://v8.dev. Acedido em: 2020-11-27.

[52] D. Venter. The evolution of Hospital Information Systems - OH Blog. https://blog.orionhealth.com/the-evolution-of-hospital-information-systems. Acedido em: 2020-11-21.

[53] Oracle VM VirtualBox. https://www.virtualbox.org. Acedido em: 2020-11-18.

[54] VMware - Delivering a Digital Foundation For Businesses. https://www.vmware.com. Acedido em: 2020-11-18.

[55] Vue.js. https://vuejs.org. Acedido em: 2020-11-18.

[56] World Wide Web Consortium (W3C). https://www.w3.org. Acedido em: 2020-11-18.

[57] Usage statistics of server-side programming languages for websites. https://w3techs.com/technologies/overview/programming_language. Acedido em: 2020-11-18.