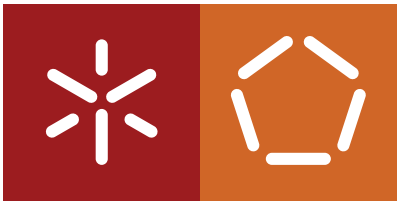
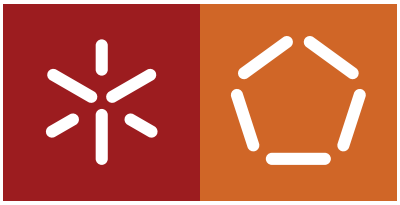


**Universidade do Minho**
Escola de Engenharia
Departamento de Informática

Rui Paulo Couto Brandão Pereira

# eHealth and Patient Relationship Management

Outubro 2022



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

Rui Paulo Couto Brandão Pereira

# eHealth and Patient Relationship Management

Dissertação de Mestrado
Mestrado Engenharia Informática

Trabalho realizado sob orientação do
**Professor Doutor José Manuel Ferreira Machado**

Outubro 2022

## DECLARAÇÃO DE INTEGRIDADE

Declaro ter atuado com integridade na elaboração do presente trabalho académico e confirmo que não recorri à prática de plágio nem a qualquer forma de utilização indevida ou falsificação de informações ou resultados em nenhuma das etapas conducente à sua elaboração.

Mais declaro que conheço e que respeitei o Código de Conduta Ética da Universidade do Minho.

# AGRADECIMENTOS

Completar esta etapa muito importante da minha vida académica não seria possível com o apoio de algumas pessoas, a todos eles os meus sinceros agradecimentos.

Primeiramente agradecer à minha família. Aos meus pais e à minha irmã por toda a paciência, compreensão, apoio, conselhos e ensinamentos e ainda todo o suporte que me têm dado ao longo da minha vida. Agradecer também a todos os meus amigos pela amizade, motivação e a confiança.

Agradeço ainda ao Ailton Moreira pela sua disponibilidade, dedicação, monitorização e acompanhamento durante a realização desta dissertação.

Por fim, um agradecimento ao Professor Doutor José Manuel Ferreira Machado pela oportunidade da realização deste tema de dissertação e pela sua disponibilidade.

# RESUMO

Os dispositivos móveis têm ganho uma enorme relevância ao longo dos últimos anos, especialmente os *smartphones*, que nos dias de hoje, são indispensáveis para a maioria das pessoas, tanto para a sua utilização pessoal como para a sua utilização profissional. Neste contexto e com algumas dificuldades encontradas nas unidades hospitalares, especialmente a falta de comparência dos utentes às respetivas consultas médicas e a falta de uma relação fluída entre o utente e a sua unidade hospitalar, foi proposta uma possível solução, de modo a combater esses problemas e a integrar os dispositivos móveis no acesso do utente aos cuidados de saúde.

No levantamento dos problemas enumerados surgiu a ideia de criar uma aplicação *mobile*, precisamente para que a interação dos utentes com as suas unidades hospitalares sejam facilitadas e fluídas. Para complementar este projeto e de modo a obter um protótipo completamente funcional, foi criada ainda uma aplicação *web*, para os profissionais de saúde, com a capacidade de responder aos pedidos dos utentes, feitos através da aplicação *mobile*.

Para procurar as funcionalidades a implementar nesta aplicação foi feita uma análise de mercado e um levantamento do que os utentes mais necessitam nas deslocações aos hospitais.

Relativamente à estrutura da solução, a aplicação *mobile* foi pensada para ser utilizada em qualquer sistema operativo de um *smartphone*, quer seja *iOS* ou *Android*, por essa razão foi desenvolvida em *React Native*, a aplicação *web* em *React*, o servidor em *Node.JS* e a base de dados relacional em *MySQL*.

**Palavras-Chave:** Dispositivos móveis, *smartphones*, aplicação *mobile*, aplicação *web*, *iOS*, *Android*, *React Native*, *React*, *Node.JS*, *MySQL*.

## ABSTRACT

Mobile devices have gained enormous relevance over the last few years, especially smartphones, which nowadays are indispensable for most people, both for their personal use and for professional use. Within this context and with some difficulties encountered within the healthcare facilities, especially the lack of attendance of the patients to the respective medical appointments and the lack of a fluid relationship between the patient and his healthcare facility, a solution was devised in order to overcome these problems and to integrate mobile devices in the patient's access to health care.

In surveying the listed problems, an idea to creating a mobile application emerged, precisely so that the interaction of patients with their healthcare facilities is facilitated. To make this this project functional, a prototype of the proposed solution was designed and developed, a web application was also created for health professionals, with the ability to respond to user requests made through the mobile application.

To look for the features to be implemented in this application, a market analysis was carried out and a survey of what patients need most when visiting hospitals.

Regarding the structure of the solution, the mobile application was designed to be used in any operating system of a smartphone, whether iOS or Android, for that reason it was developed in React Native, the web application in React, the server in Node.JS and the relational database in MySQL.



**Keywords:** Mobile devices, smartphones, mobile application, web application, *iOS*, *Android*, *React Native*, *React*, *Node.JS*, *MySQL*.

# CONTEÚDO

# LISTA DE FIGURAS

# LISTA DE TABELAS

# LIST OF LISTINGS

**API** - Application Programming Interface

**ART** - Android Runtime

**CORS** - Cross-origin Resource Sharing

**CSS** - Cascading Style Sheets

**DK** - Desgin Knowledge

**DSR** - Design Science Research

**FS** - File System

**HAL** - Hardware Abstracion Layer

**HTML** - HyperText Markup Language

**HTTP** - HyperText Transfer Protocol

**HTTPS** - HyperText Transfer Protocol Secure

**JS** - JavaScript

**JSON** - JavaScript Object Notation

**JSX** - JavaScript Syntax Extension

**JWT** - JSON Web Token

**MVC** - Model View Controller

**OS** - Sistema operativo

**PDF** - Portable Document Format

**POC** - Proof of concept

**REST** - Representational State Transfer

**SDK** - Software Development Kit

**SMS** - Short Message System

**SWOT** - Strengths, Weaknesses, Opportunities, and Threats

**UI** - User Interface

**UX** - User Experience

**XML** - Extensible Markup Language

# 1

# INTRODUÇÃO

## 1.1 PROBLEMAS, DESAFIOS E MOTIVAÇÃO

Esta dissertação tem como o principal objetivo de melhorar a relação de um utente com a sua unidade hospitalar.

Muito dos utentes nas unidades hospitalares têm dificuldades em agendar consultas médicas e quando são agendadas certos utentes têm de esperar muito tempo para que a consulta seja realizada e por outro lado, existem vários problemas associados como por exemplo, a incompatibilidade de horários dos profissionais de saúde com os horários do utente, um desfasamento estrutural entre a procura de cuidados de saúde e a capacidade de resposta (número de profissionais de saúde), entre outros.

A maioria dos utentes que têm de esperar meses por estas consultas, ou acabam por desistir ou procuram as unidades hospitalares privadas, que são bastante mais rápidas na realização destas mesmas consultas. A ocorrer uma destas opções surge um dos maiores problemas nas unidades hospitalares, que é precisamente a falta de comparência dos utentes às respetivas consultas médicas.

O Ministério da Saúde Português estima que desde 2010 não se efetuam por ano cerca de um milhão de consultas por ausência do utente [16]. Este facto prejudica muito os hospitais, principalmente em termos de ocupação dos profissionais. Também relacionado com a ocupação dos profissionais, existe a vertente financeira, porque quando existe uma falta de comparência a uma consulta, os profissionais de saúde que se estavam a preparar para esse atendimento acabam por não o fazer, o que gera o não pagamento da consulta por parte do utente e do trabalho que o pessoal médico se preparou para fazer e não fez.

Tendo isso em conta e no enquadramento da nossa realidade atual, poderemos tentar diminuir o impacto deste problema com o uso das tecnologias. Aproveitando os nossos dispositivos móveis, que hoje em dia são praticamente indispensáveis, irá ser desenvolvida uma aplicação móvel, para os utentes, na tentativa de diminuir estes casos, assim como irá ser criada a aplicação *web* para que os profissionais de saúde possam gerir e satisfazer as necessidades dos utentes.

## 1.2 OBJETIVOS

De forma a minimizar os problemas identificados e com o intuito de haver uma relação mais fluída entre o utente e a sua unidade hospitalar irão ser desenvolvidas aplicações onde seja possível simplificar esta relação e satisfazer as necessidades do utente.

No desenvolvimento destas aplicações, há a intenção de oferecer a possibilidade ao utente:

- Agendamento de consultas;
- Verificação das consultas futuras;
- Efetuar o requerimento para a realização de exames;
- Consultar o histórico de consultas e exames;
- Possibilidade de aceder a informações úteis;
- O desenvolvimento de um protótipo de um *chatbot*, para que os utentes consigam aceder a informações pertinentes mais específicas.

Ao propor estas funcionalidades há o objetivo de tornar a aplicação fácil de ser usada para que os utentes sejam o melhor atendidos e esclarecidos possíveis, além disso é importante também que a organização do ambiente hospitalar possa melhorar.

## 1.3 ESTRUTURA DA DISSERTAÇÃO

Esta dissertação é composta por sete capítulos. No primeiro capítulo está presente uma breve introdução ao tema desta dissertação, assim como, os principais objetivos da mesma.

O segundo capítulo consiste na metodologia adotada na realização desta dissertação, *Design Science Research*.

No terceiro capítulo, denominado de estado de arte, é onde existe uma introdução do tema abordado *e-Health*, assim como uma análise de mercado, com algumas das principais aplicações hospitalares do país. Há ainda a introdução das metodologias de trabalho, dos equipamentos e *softwares* que foram utilizados e das ferramentas que foram usadas para a concretização da solução desenvolvida.

No quarto capítulo, há uma visão geral do trabalho desenvolvido, nomeadamente, as principais funcionalidades da aplicação *mobile* e da aplicação *web*, a arquitetura da solução desenvolvida e ainda uma visão de como foi desenvolvido o servidor e a sua base de dados.

No quinto capítulo, há a demonstração das principais funcionalidades desenvolvidas de cada aplicação e uma breve explicação de que maneira é possível usufruir dessas funcionalidades.

No sexto capítulo está presente uma análise ao trabalho desenvolvido, nomeadamente, a prova de conceito e a análise SWOT.

No sétimo e último capítulo serve para retirar algumas conclusões sobre o trabalho realizado e refletir sobre o que poderá ser feito para melhorar a solução desenvolvida.

# 2

# ABORDAGEM METODOLÓGICA

A metodologia de investigação que vai servir de base de pesquisa para a realização desta dissertação será o *Design Science Research* (DSR).

Na realização desta investigação é importante que se identifique e se defina uma questão de investigação central, que partirá como uma base para este trabalho e que sirva para promover uma fluidez do restante desenvolvimento desta dissertação no futuro.

- Quais as funcionalidades ideias e mais procuradas para satisfazer o paciente na sua relação com a sua unidade hospitalar ?
- Em que medida seria útil um chatbot na interação dos utentes com os serviços de saúde?

Para tentar obter resposta a estas perguntas será adotada a metodologia de investigação anteriormente referida, o *Design Science Research*.

## 2.1 DESIGN SCIENCE RESEARCH

Esta metodologia é, fundamentalmente, um paradigma direcionado para a resolução de problemas. O DSR procura melhorar o conhecimento através da criação de artefactos inovadores e da produção do *design knowledge* (DK) de modo a encontrar soluções para os problemas do mundo real [9].

O DSR é um conjunto de técnicas analíticas que permitem o desenvolvimento de pesquisas nas áreas mais diversas, em particular na engenharia. É um processo rigoroso em que consiste em projetar artefactos para tentar resolver problemas, avaliar o impacto do mesmo e o que está funcional e devolver um *feedback* dos resultados obtidos [36].

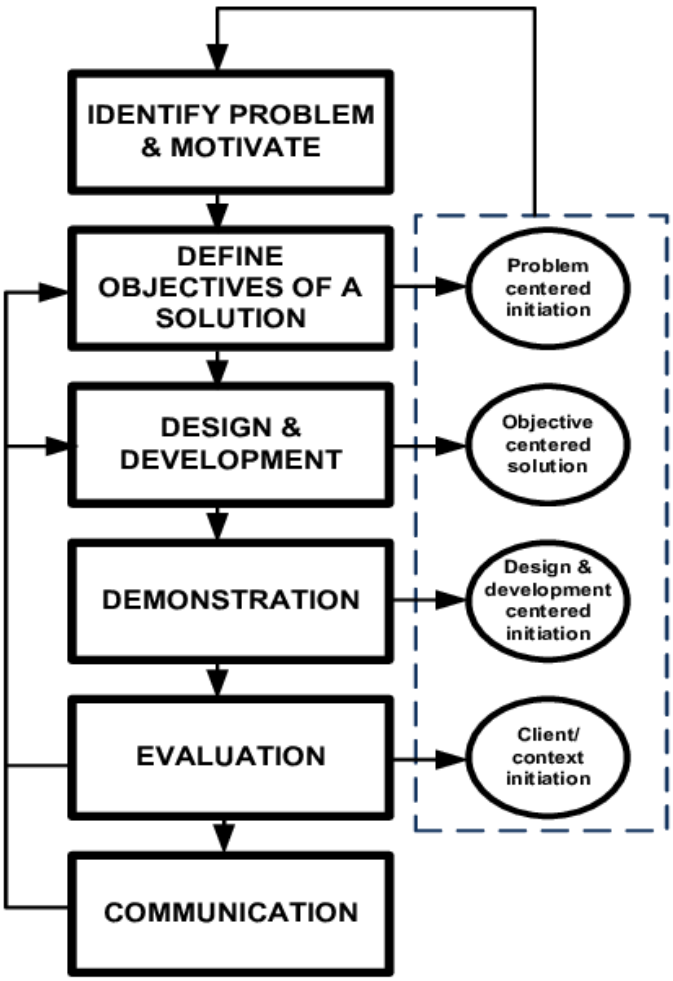



Figura 1: Exemplo de uma abordagem DSR [7]

**Identificação do problema e motivação** - É a primeira fase do método DSR e consiste numa pesquisa específica do problema e na justificação do valor de uma solução. Com a definição clara do problema é possível desenvolver um artefacto capaz de oferecer uma solução. Justificar o valor de uma solução visa a:

- Motivar o investigador e a plateia deste mesmo a seguir uma solução e aceitar os resultados, o que ajuda a perceber o raciocínio associado à interpretação feita pelo investigador sobre o problema.

Tal como tinha sido referido na parte introdutória desta dissertação, os problemas e a motivação associada à realização desta dissertação, encontra-se principalmente no melhoramento da relação paciente/hospital, ou seja, arranjar uma maneira de haver uma relação mais fluida, e dentro das motivações existe a questão financeira dos hospitais e a boa gestão do paciente com os serviços a que pretende aceder de uma unidade hospitalar.

**Definição dos objetivos de uma solução** - Definir os objetivos de uma solução a partir da definição do problema e do conhecimento do que é possível ser feito. Os objetivos podem ser classificados como quantitativos,

termos em que uma solução desejável será melhor do que as atuais, ou qualitativos, que é a descrição de como um novo artefacto terá a capacidade de suportar as soluções para problemas. Os objetivos devem ser deduzidos a partir da especificação do problema. No desenvolvimento desta fase é necessário haver um conhecimento do estado dos problemas e as soluções atuais e a sua eficácia, caso existam.

**Design e desenvolvimento** - Criação do artefacto. Estes artefactos são potencialmente construtores, modelos, métodos ou instâncias. Um *design research artifact* pode ser qualquer objeto desenhado no qual a contribuição da pesquisa esteja incorporada no *design*. Esta fase inclui determinar a funcionalidade do artefacto e a sua arquitetura e só assim posteriormente a criação do artefacto final. Para esta fase é necessário haver o conhecimento do problema e a importância da solução. Os recursos necessários para passar da fase anterior *Definição dos objetivos de uma solução* para esta fase passam pelo conhecimento da teoria que pode ser aplicada numa solução.

**Demonstração** - Demonstração da utilização do artefacto para resolver uma ou mais instâncias do problema. Esta fase pode envolver o seu uso em experiências, simulações, estudo de caso ou outras atividades apropriadas. Para esta fase é importante saber manusear o artefacto para resolver o problema.

**Avaliação** - Observação e manipulação de como um artefacto suporta a solução de um problema. Esta fase envolve a comparação de objetivos de uma solução com os resultados observados com a utilização do artefacto na demonstração, isto requer conhecimentos de métricas relevantes e de técnicas de análise. Dependendo da natureza do problema e do artefacto, a avaliação pode assumir várias formas. Pode incluir a comparação da funcionalidade do artefacto com os objetivos da solução da fase nº2 *Definição dos objetivos de uma solução*, medidas do desempenho dos objetivos quantitativos, como orçamentos ou o material produzido, o resultado de inquéritos, *feedback* do cliente, simulações, etc. Pode incluir também medidas quantificativas do desempenho do sistema, como o tempo de resposta ou a sua disponibilidade. Esta avaliação pode incluir qualquer evidencia empírica apropriada ou prova lógica. No final desta fase, os investigadores podem decidir se voltam atrás para a fase 3, para melhorar a eficácia do artefacto ou se continuam para a próxima fase.

**Comunicação** - Comunicar o problema e a sua importância, o artefacto, a utilidade e a sua inovação, o rigor do *design* e a sua eficácia para investigadores e outros públicos relevantes. Em publicações no âmbito académico, os investigadores usam a estrutura deste processo para estruturar o seu artigo, assim como a estrutura nominal de um processo de pesquisa. Esta última fase requer o conhecimento cultural disciplinar [47] [49], [24].

# 3

# ESTADO DA ARTE

## 3.1 E-HEALTH

Segundo G. Eyesenbach [17], *e-health* é um campo emergente em que se intersetam as informações médicas, a saúde pública e os negócios, referindo-se a serviços de saúde e informações entregues ou trocadas através da Internet e de tecnologias relacionadas.

Este termo não só caracteriza uma ferramenta tecnológica, como também um estado de espírito, uma maneira de pensar, uma atitude e um compromisso com o pensamento global em melhorar os cuidados de saúde local, regional e global usando informação e comunicação tecnológica [27].

O autor também refere a existência de 10 princípios fundamentais no *e-health* [17]:

1. **Efficiency** (Eficiência) - Um dos objetivos do *e-health* é de aumentar a eficiência nos cuidados médicos, diminuindo os custos.

2. **Enhancing quality** (Melhoria da qualidade) - Aumentar eficácia não se traduz apenas na redução de custos, mas também na qualidade do serviço prestado. *E-health* pode aumentar a qualidade, por exemplo, ao permitir comparações entre diferentes consumidores e permitir o *feedback* dos mesmos.

3. **Evidence based** (Baseado em evidências) - As intervenções do *e-health* devem ser feitas baseadas em evidências, com o devido rigor de avaliação científica.

4. **Empowerment** (Fortalecimento) - Ao tornar acessível conhecimentos básicos de medicina e a possibilidade de os pacientes acederem a registos eletrónicos é possível que estes tomem decisões baseadas em evidências.

5. **Encouragement** (Encorajamento) - Encorajamento de uma relação paciente/profissional de saúde, em que as decisões podem ser tomadas em conjunto.

6. **Education** (Educação) - Educação de médicos a partir de cursos online e de consumidores.

7. **Enabling information** (Permitir informações) - Permitir a troca de informações e a comunicação da mesma de uma forma global entre as várias unidades hospitalares.

8. **Extending** (Extensão) - Aumentar as fronteiras dos cuidados de saúde. Pode variar de cuidados simples, como conselhos, até cuidados mais complexos, como a prescrição de certos medicamentos.

9. **Ethics** (Ética) - Envolve uma interação inovadora entre paciente/profissional de saúde e apresenta novos desafios e ameaças a questões éticas, como a consulta médica online, consentimentos do paciente, privacidade, entre outros.

10. **Equity** (Igualdade) - O problema da igualdade pois nem toda a gente tem a capacidade financeira de obter equipamentos eletrónicos e muita gente não tem capacidade de manuseamento desses aparelhos, o que pode gerar uma certa desigualdade no acesso ao *e-health*.

## 3.2 ANÁLISE DO MERCADO

Para se encontrar algumas funcionalidades populares e analisar o que existe atualmente no mercado, investigou-se outras aplicações de vários hospitais e unidades hospitalares, tanto a nível nacional como internacional.

### 3.2.1 *App My São João*

Aplicação criada para os utentes do Centro Hospitalar de São João. É um hospital com forte ligação à Faculdade de Medicina da Universidade do Porto. Situa-se no Porto e é um dos maiores hospitais do país. A aplicação que foi criada para os utentes desta unidade hospitalar é denominada de *App My São João* e oferece as seguintes funcionalidades [29]:

| App My São João |
|---|
| Confirmar a chegada para a consulta, sem necessidade de se dirigir ao secretariado ou ao quiosque |
| Solicitar pedido de remarcação de consulta, caso não possa comparecer |
| Realizar vídeoconsulta, de acordo com a indicação do seu médico assistente |
| Registar a admissão na Urgência de Adultos, sem necessidade de se dirigir ao secretariado |

Tabela 1: Informações sobre a App *My São João*

### 3.2.2 *App PROXIMO*

Esta aplicação é direcionada para a gestão de atendimento e filas de espera e permite que seja atendido sem filas, tempos de espera, à distância e com segurança. Esta aplicação está disponível em centenas de serviços públicos e privados, áreas da saúde, justiça, retalho, restauração, educação, cultura e espetáculos, etc

O Hospital da Senhora da Oliveira Guimarães, localizado em Guimarães, e com a intenção de melhorar os serviços prestados ao público, disponibilizou esta aplicação, que disponibiliza as seguintes funcionalidades [50]:

| App PROXIMO |
|---|
| Tirar uma senha para ser atendido na hora |
| Agendar uma senha para quando chegar perto do local |
| Fazer uma marcação ou reserva definindo data e hora |
| Pedir um bilhete individual ou de grupo |
| Confirmar check in no local onde tem marcação ou reserva |

Tabela 2: Informações sobre a App *PROXIMO*

### 3.2.3 *App CHUPorto*

O Centro Hospitalar Universitário do Porto, situado também no Porto, é composto por quatro polos e ofereceu esta aplicação aos seus utentes que tem os objetivos de [51]:

| App CHUPorto |
|---|
| Consultar o histórico da sua relação com o CHUPorto |
| Verificar na agenda as consultas e/ou exames marcados |
| Solicitar remarcação ou cancelamento da consulta e/ou exame agendado |
| Receber notificações de alerta sobre marcações futuras |
| Solicitar declarações de presença para o utente e/ou para o acompanhante |
| Consultar taxas moderadoras em dívida |
| Consultar informação sobre o local da consulta e/ou exame |
| Disponibilizar informações úteis sobre o CHUPorto |

Tabela 3: Informações sobre a App *CHUPorto*

### 3.2.4 *App My Luz*

A rede Hospital da Luz é uma das maiores e mais completas redes de cuidados de saúde privados em Portugal. É constituída por 24 hospitais e clínicas distribuídas pelo país. Estão presentes nas regiões Norte, Centro e Centro-Sul e ainda na ilha da Madeira. Esta rede criou uma aplicação denominada de *My Luz* e oferece ao seu utilizador [10]:

| App My Luz |
|---|
| Associar os seus filhos ou qualquer outro familiar à sua conta, atualizar os seus dados pessoais e fazer a gestão da sua agenda clínica |
| Marcar as suas consultas, videoconsultas e exames, notificações de alerta das próximas marcações e gestão da agenda, onde pode alterar ou cancelar uma marcação |
| Registar dados e medições pessoais que permitirão um conhecimento detalhado e mais completo da evolução do estado de saúde do paciente. |
| Realizar consultas à distância com o seu médico do Hospital da Luz, para discutir resultados de exames, esclarecer dúvidas ou ainda para consulta de rotina |
| Consultar faturas e capacidade de efetuar pagamentos no MY LUZ de forma rápida e segura, com cartão de crédito, MB WAY e PayPal ou referência Multibanco |
| Consultar o histórico dos atos médicos que realizou no Hospital da Luz, reagendar próxima consulta e possibilidade de aceder à declaração de presença |
| Visualizar, guardar ou partilhar com um médico assistente, fora da rede Hospital da Luz, os resultados de análises clínicas, exames de imagiologia e de outras especialidades |

Tabela 4: Informações sobre a App *My Luz*

### 3.2.5 *App Trofa Saúde*

O Trofa Saúde é também uma rede de unidades hospitalares e assume-se como um projeto global de saúde. É um conjunto de hospitais privados, e serve uma população superior a 4 milhões de habitantes, a maioria dos hospitais desta rede situa-se na zona norte do país. Esta rede criou uma aplicação para os seus utentes com o nome de *Trofa Saúde* e oferece a possibilidade de [59]:

| App Trofa Saúde |
|---|
| Aceder a prescrições médicas com a possibilidade de marcação |
| Aceder ao histórico de marcações |
| Consultar a agenda de marcações, incluindo acesso à informação necessária para a preparação de exames |
| Consultar faturas de atos médicos |
| Consultar informação/localização das suas unidades |
| Definir notificações |
| Gerir os elementos do seu agregado familiar |
| Marcar, remarcar ou anular consultas e exames |
| Pagar consultas e exames |
| Realizar a admissão (check-in) a consultas e exames |
| Visualizar relatórios de exames e análises |

Tabela 5: Informações sobre a App *Trofa Saúde*

### 3.2.6 *App MyMediclinic 24x7*

Hospital internacional situado na cidade do Dubai, nos Emirados Árabes Unidos. Oferecem a cobertura de muitas especialidades nos seus serviços e são capazes de oferecer tratamentos e diagnósticos avançados, assegurando que qualquer paciente tem acesso a uma qualidade superior de cuidados de saúde.

Esta unidade hospitalar criou uma aplicação de nome *MyMediclinic 24x7 app*, que tem as seguintes funcionalidades [44]:

| App MyMediclinic 24x7 |
|---|
| Visualização de perfis dos profissionais de saúde |
| Capacidade de procurar médicos através da sua especialidade, localização, línguas faladas, cobertura do seguro e género |
| Consultar as consultas agendadas, tanto presencialmente como online, e escolher o dia e a hora que o paciente preferir |
| Reagendar ou cancelar uma consulta |
| Receber notificações com avisos de uma consulta próxima |
| Ter uma consulta online |
| Gerir as consultas do seu agregado familiar |

Tabela 6: Informações sobre a App *MyMediclinic 24x7*

## 3.3 DISPOSITIVOS MÓVEIS: *smartphones*

Nos dias de hoje, qualquer pessoa possui um *smartphone*, que é o dispositivo móvel mais usado pelo cidadão comum. Na tradução à letra significa telemóveis inteligentes que vai de encontro à sua definição original. Os *smartphones* são dispositivos portáteis que combinam as funcionalidades de um telemóvel com funções computacionais apenas numa unidade. O *hardware* de cada um deles é robusto e é bastante variável e para o funcionamento do mesmo é necessário existir um sistema operativo.

Os *smartphones* oferecem ao utilizador a possibilidade de navegação na internet, funções de multimédia (musicas, vídeos, câmara, entre outros), acesso a várias redes sociais, aplicações e muitas mais funcionalidades computacionais, acrescentando as estas funções, existem também a possibilidade das chamadas de voz e das mensagens de texto que foram herdadas precisamente dos telemóveis.

Os sistemas operativos dos *smartphones* são variados, como por exemplo, *iOS*, *Android*, *Symbian OS*, *Windows Phone*, *Blackberry*, *WebOS*, entre outros [46], [57].

### 3.3.1 *Sistema Operativo dos dispositivos*

*Android*

O *Android* é um sistema operativo desenvolvido pela *Google* para dispositivos móveis. Foi lançado em Novembro de 2007 debaixo da *framework Open Handset Alliance*, com o objetivo de ser *open source* para desenvolvimento de *software* nas plataformas móveis. É baseado no *kernel* do *Linux* e facilita os programadores a escrever código *Java*, ao usar as bibliotecas de *Java* criadas pela *Google*. Este *software* permite um correto funcionamento do dispositivo e é capaz de oferecer ao utilizador uma interface para que seja possível a sua utilização [21], [20].

Na figura seguinte podemos visualizar as camadas presentes no sistema operativo *Android*.

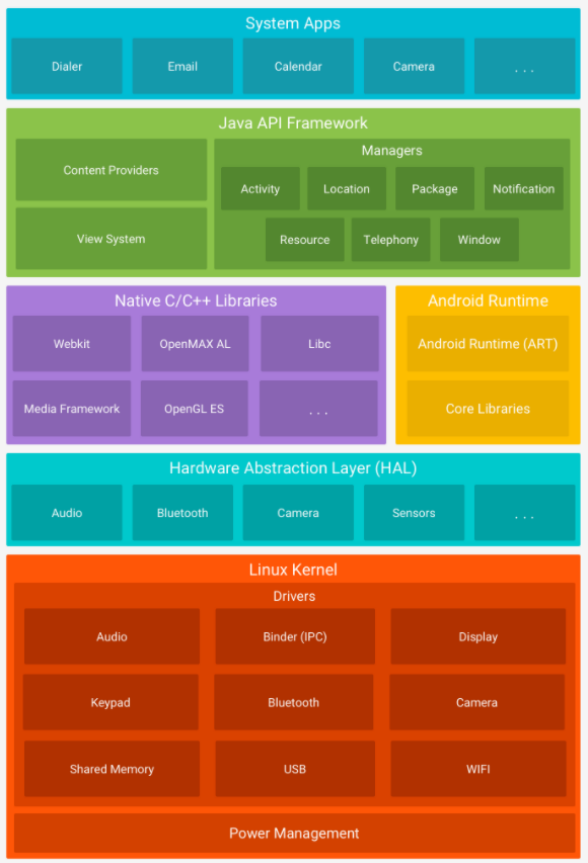



Figura 2: *Stack* do *software* do *Android* [14]

**Linux Kernel** - O *kernel* que serve como base para o sistema operativo *Android* é o *kernel* do *Linux*. O *Android Runtime* confia neste *kernel* para funcionalidades como o *threading* e a gestão de memória de baixo nível.

**Hardware Abstraction Layer (HAL)** - Esta camada oferece interfaces padrão que expõem as capacidades de hardware do dispositivo para uma camada de maior nível que neste caso é a *Java API Framework*. HAL consiste em módulos de bibliotecas, que implementam uma interface para um tipo específico de hardware, como a câmara ou o *Bluetooth*. Quando uma *Framework API* faz uma chamada para aceder ao *hardware* do dispositivo, o sistema *Android* carrega os módulos da biblioteca, presentes nesta camada, para essa componente de hardware.

**Android Runtime (ART)** - Cada aplicação executa o próprio processo com uma instância do ART. Este é destinado a executar várias máquinas virtuais em dispositivos de baixa memória executando arquivos DEX, que é um formato de *bytecode* criado para *Android*, e é otimizado para utilizar o mínimo de memória possível. Os principais recursos de ART são:

- Capacidade de compilação *ahead-of-time* AOT e *just-in-time* JIT;

- Otimização da *Garbage Collection (GC)*, que é uma forma automática de gestão de memória;
- No *Android* 9 ou superior, a conversão de arquivos *DEX* de um pacote de aplicações usa um código de máquina mais compacto;
- Melhor suporte de *debugging*, capacidade de diagnósticos detalhadas e a criação de relatórios de erros, além da capacidade de definir pontos de controlo de forma a vigiar variáveis específicas.

**Native C/C++ Libraries** - Várias componentes e serviços do *Android*, como *ART* e *HAL*, são implementados com código nativo que exigem bibliotecas programadas em C e C++. O *Android* é capaz de fornecer as *Java API Framework* de modo a oferecer a capacidade nativa às aplicações. A título de exemplo, é possível aceder ao *OpenGL* através da *Java OpenGL API* da estrutura do *Android* para que seja possível a criação de gráficos 2D e 3D numa certa aplicação.

Se existir uma aplicação em que seja necessário o uso de código C ou C++, é possível usar o *Android NDK* de forma a utilizar as bibliotecas nativas diretamente no código desenvolvido.

**Java API Framework** - O conjunto de recursos do sistema operativo *Android* está disponível através das API's programadas em Java. Essas API's são usados pelos programadores para criarem aplicações *Android* e oferecem vários recursos, tais como:

- Um IU rico e extensivo com listas, grades, caixas de texto, botões e até um navegador de internet incorporado;
- Recursos, entre eles recursos sem código como *strings* localizadas, gráficos e ficheiros de *layout*;
- Notificações, que permite que as aplicações desenvolvidas exibam alertas ao utilizador;
- Controlador de atividade, que controla o ciclo de vida das aplicações;
- Distribuidores de conteúdo, que permitem que as aplicações acedem aos dados de outras aplicações.

**System Apps** - O *Android* vem com um conjunto de aplicações principais como o e-mail, serviço de SMS, calendários, navegador de internet, contactos, etc. Não existe uma preferência entre as aplicações de origem (as que vem com o *Android*) e as aplicações que são instaladas.

As aplicações que vêm de origem podem ser usadas por outras aplicações criadas pelos programadores, como por exemplo, se uma aplicação quiser enviar uma SMS, é possível chamar o serviço de SMS que já vem instalado no *Android* [14], [13].

### *iOS*

Sistema operativo para muitos dispositivos lançados pela *Apple*, entre os quais o *iPhone* que foi lançado em 2007 e mudou o mercado dos *smartphones*. Aplicações para *iOS* são escritas em *Objetive-C*, que é uma extensão da

linguagem C. Desenvolvimento de aplicações para *iOS* requer um computador que corra o sistema operativo *Mac OS* [21].

A estrutura do sistema operativo *iOS* é *Layered based*, ou seja, a comunicação entre camadas não ocorre diretamente. As camadas de baixo nível oferecem serviços mais básicos que são úteis a praticamente todas as aplicações, enquanto que as camadas de maior nível já fornecem serviços com capacidade gráfica e serviços relacionados com a interface [26].

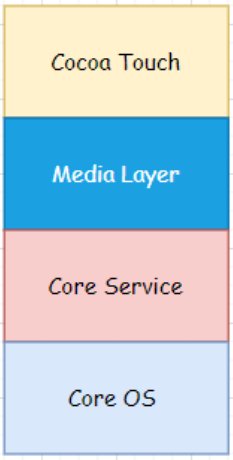



Figura 3: Arquitetura do *iOS*

**Core OS** - Todas as tecnologias do *iOS* são construídas debaixo desta camada, que é a de nível mais baixo da arquitetura, as tecnologias são o *Core Bluetooth Framework*, *External Accessories Framework*, *Accelerate Framework*, *Security Services Framework*, *Local Authorization Framework*, entre outras. Esta camada suporta 64 *bit* o que permite que a aplicação corra com maior velocidade.

**Core Service** - Nesta camada existem algumas *frameworks* importantes que ajudam o sistema operativo a funcionar e a capacidade de fornecer uma melhor funcionalidade. Algumas *frameworks* presentes nesta camada:

- *Address Book Framework* - Fornece acesso aos detalhes de contacto do utilizador;
- *Cloud Kit Framework* - Intermediário que transfere a informação da aplicação para a *iCloud*;
- *Core Data Framework* - Tecnologia usada para gerir o modelo de dados de uma aplicação *Model View Controller* (MVC);
- *Core Foundation Framework* - Fornece a capacidade de gerir dados e serviços para as aplicações *iOS*;
- *Core Location Framework* - *Framework* que ajuda a fornecer a localização e informação para a aplicação;

- *Core Motion Framework* - Todos os dados baseados no movimento são acedidos com a ajuda desta *Framework*;
- *Foundation Framework* - Introduz paradigmas que definem funcionalidades que o *Objetive*-C não consegue cobrir;
- *HealthKit Framework* - Gere as informações relacionadas com a saúde do utilizador;
- *HomeKit Framework* - Usada para trocar informações e controlar dispositivos conectados a partir da casa do utilizador;
- *Social Framework* - Interface para aceder às redes sociais do utilizador;
- *StoreKit Framework* - *Framework* que ajuda na compra de conteúdo e serviços dentro das aplicações *iOS*.

**Media Layer** - Com a ajuda desta camada é possível ativar os gráficos e a tecnologia de áudio presente no sistema, nesta camada existem várias *frameworks*, tais como:

- *ULKit Graphics* - Suporta a visualização de imagens e de animação do conteúdo;
- *Core Graphics Framework* - Suporta vetores 2D e a renderização baseada em imagem é um motor de desenho nativo para o *iOS*;
- *Core Animation* - Ajuda na otimização das animações nas aplicações *iOS*;
- *Media Player Framework* - Ajuda no suporte na reprodução de áudio e permite ao utilizador usar a sua biblioteca de *iTunes*;
- *AV Kit* - Oferece várias interfaces para apresentação de vídeo, gravações e a reprodução de áudio e/ou vídeo;
- *Open AL* - Uma tecnologia padrão da utilização de áudio;
- *Core Images* - Suporte avançado de imagens imóveis;
- *GL Kit* - Gere a renderização 2D e 3D por interfaces aceleradas por hardware.

**Cocoa Touch** - Também conhecido como *"Application Layer"*, e esta camada atua como uma interface para o utilizador conseguir trabalhar com o sistema operativo *iOS*. Suporta o *touch* e os eventos de movimentos e muitas outras funcionalidades. Oferece algumas *frameworks* como o *EvenKit Framework*, que mostra uma interface do sistema para haver interação, *GameKit Framework* para os utilizadores serem capazes de partilhar as informações de jogo online usando o *Game Center*, *MapKit Framework* que fornece um mapa navegável com o *scroll* que pode ser incluído na interface do utilizador e o *PushKit Framework* que fornece suporte no registo [26], [31].

## 3.4 ABORDAGENS DE DESENVOLVIMENTO

### 3.4.1 *Abordagem Nativa e Abordagem Multiplataforma*

*Abordagem Nativa*

A abordagem nativa de uma aplicação refere-se ao processo de construir uma aplicação exclusiva para uma determinada plataforma [35].

Aplicações nativas são conhecidas por entregar uma grande qualidade da experiência de utilizador, uma vez que são especificamente aplicações de alta performance. Estas aplicações conseguem aceder a capacidades do *hardware*, tais como sensores e câmaras, ao contrário das aplicações acedidas pelo *browser* [33].

Com o uso de aplicações nativas temos alguns benefícios [54], [56]:

- **Maior performance**, podem ser criadas e otimizadas aplicações nativas para uma certa plataforma, na compilação é ainda usada a linguagem de programação *core* da plataforma usada, isto faz com que seja mais rápido, mais eficiente e responsivo às ações do utilizador;
- **Maior segurança**, uma vez que as aplicações nativas podem aceder a funcionalidades específicas de uma plataforma, incluindo as de segurança;
- **Qualidade da experiência do utilizador**, estas aplicações oferecem uma utilização fluída, pois o *design* da aplicação é feito de acordo com as especificações do sistema operativo, o que faz com que a relação com o utilizador seja fluída;
- **Acesso às funcionalidades do dispositivo**, estas aplicações com este acesso oferecem uma melhor experiência ao utilizador, como por exemplo, as notificações que prendem ainda mais o utilizador à aplicação criada;
- **Número de *bugs* reduzido**, porque os programadores das aplicações nativas têm acesso aos *SDK's* logo que são lançados.

Porém têm também algumas desvantagens [54], [56]:

- **Custo**, uma vez que se o objetivo for a distribuição para várias plataformas, terá, por exemplo, uma equipa para desenvolver a aplicação para *Android* e outra para *iOS*;
- **Tempo de desenvolvimento**, várias equipas a trabalharem na mesma aplicação para diferentes plataformas leva algum tempo, e a sua manutenção e eventuais melhorias irão requerer um esforço adicional;

- **Reutilização de código reduzido**, pois todas as alterações que serão feitas obrigam a criação de código novo.

*Abordagem Multiplataforma*

A abordagem multiplataforma é uma forma que permite que seja criada uma aplicação que seja compatível para diferentes sistemas operativos. Nestas aplicações, algum do código ou até mesmo todo o código poderá ser partilhado [34]. Isto pode ser feito com ferramentas como o *React Native*, *Xamarin* e *Flutter*, entre outras, onde as aplicações podem ser distribuídas em *Android* e em *iOS* [38].

Apesar de esta abordagem permitir poupar tempo e dinheiro, a qualidade do processo é sacrificado, uma vez que é muito difícil criar uma aplicação que corra em vários sistemas operativos nas condições ideais. A aplicação terá também de ter uma camada superior enquanto é utilizada, o que resulta na degradação da sua performance em relação à abordagem nativa [38], [23].

As suas vantagens [54], [56]:

- **Redução de custos**, estas aplicações apenas precisam de uma equipa de programadores, apenas precisam de conhecer a *framework* escolhida para o efeito;
- **Código reutilizável**, uma vez que a abordagem multiplataforma permite o uso de apenas uma fonte de código para todas as plataformas, o que oferece consistência ao projeto;
- **Desenvolvimento rápido**, ao reutilizar código, aumenta a produtividade e é lançado no mercado mais rapidamente;
- **Manutenção simplificada**, ao lidar apenas com um código é mais fácil realizar atualizações e melhorias.

E as desvantagens [54], [56]:

- **Pegada digital maior**, aplicações multiplataforma são normalmente maiores;
- **Dificuldade de integração**, as funcionalidades específicas de uma plataforma poderão não ter compatibilidade;
- **Performance mais baixa**, em comparação com as aplicações nativas, estas têm uma camada superior de computação, o que leva a uma diminuição do seu desempenho;
- **Funcionalidades plataforma atrasadas**, os *SDK's* lançados, inicialmente, são compatíveis apenas com as aplicações nativas, é necessário esperar para haver uma atualização para que estas aplicações multiplataforma tenham acesso às novas funcionalidades.

*Nativa vs Multiplataforma*

Em comparação podemos afirmar que a diferença fundamental da escolha entre estas duas abordagens prende-se, essencialmente, para qual sistema operativo será feita a aplicação *mobile*. A abordagem nativa permite que seja construída uma aplicação para um determinado sistema operativo, enquanto que na abordagem multiplataforma, a aplicação é construída para ser utilizada em vários sistemas operativos [55].

| Vantagens | |
|---|---|
| Nativa | Multiplataforma |
| Melhor performance | Custo reduzido |
| Segurança da informação sobre o utilizador | Código reutilizável |
| Acesso às funcionalidades do dispositivo | Desenvolvimento rápido |
| Número de *bugs* bastante reduzido | Manutenção facilitada |

Tabela 7: Vantagens das abordagens

| Desvantagens | |
|---|---|
| Nativa | Multiplataforma |
| Custo | Dificuldades de integração |
| Tempo de desenvolvimento | Performance reduzida |
| Dificuldade em reutilizar código | Atraso, em relação ao nativo, no acesso às funcionalidades |

Tabela 8: Desvantagens das abordagens

### 3.4.2 *Servidor Web*

O servidor, nesta abordagem *web*, tem a funcionalidade de gerir a informação, ou seja, tratar de fornecer o necessário para o *frontend* da aplicação, que é a interface do utilizador, e tem também a função de ir buscar as informações necessárias à base de dados para responder precisamente aos pedidos feitos pelo *frontend* [6], [43], [42].

Quando um cliente envia um pedido a um servidor, a informação é procurada e enviada de volta.

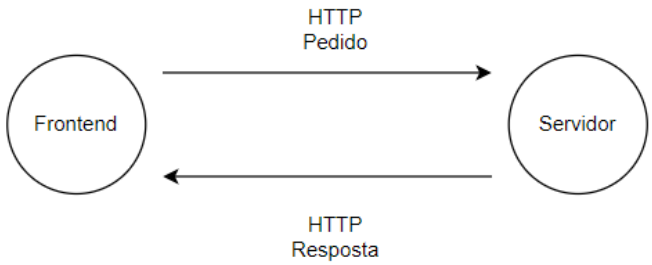



Figura 4: Servidor *Web*

Ao fazer estas trocas de informações é usado o método **HTTP**.

*HTTP*

Este protocolo define um conjunto de métodos de requisição responsáveis por indicar a ação a ser executada para um determinado recurso. Existem diversos métodos existentes no *HTTP*, o mais importantes são:

- GET - Solicita a representação de um recurso específico. Normalmente, uma requisição do método *GET* envolve apenas troca de dados;
- POST - Usado para submeter uma entidade a um recurso específico, frequentemente este método provoca uma adição nos dados guardados;
- PUT - Este método, geralmente é usado para fazer uma atualização dos dados;
- DELETE - Já este método, é usado para remover dados.

### 3.4.3 *Model-View-Controller*

MVC (Model-View-Controller) é um padrão em *design* de *software* usado, normalmente, para implementar a interface, data e o controlo da lógica de uma aplicação. A separação destas componentes oferece uma melhor divisão do trabalho e a sua manutenção é muito mais fácil de ser feita [40], [25].

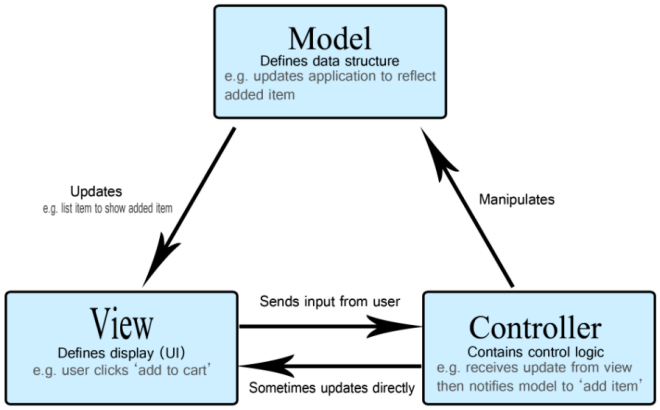



Figura 5: Model View Controller [40]

**Model** - Representa a estrutura de dados, o formato e as restrições com as quais é armazenado. Mantém as informações de uma aplicação. Funciona como a base de dados da aplicação.

**View** - O que é apresentado ao utilizador. Utiliza o *Model* e apresenta o conteúdo da forma que o utilizador queira. Um utilizador também deve ser capaz de alterar a forma como esse mesmo conteúdo é apresentado. Essencialmente, a *View* consiste em páginas dinâmicas e estáticas que são renderizadas à medida que o utilizador faz um pedido das mesmas.

**Controller** - Tem o controlo dos pedidos e tem a função de apresentar as respostas solicitadas ao utilizador. Normalmente, o utilizador interage apenas com a *View*, que irá fazer o pedido ao *Controller* e este mesmo trata da resposta para enviar de novo. O *Controller* renderiza a *View* apropriada com o conteúdo do *Model* como resposta [12].

## 3.5 FERRAMENTAS DE DESENVOLVIMENTO

### 3.5.1 *JavaScript*

O *JavaScript* é uma linguagem de programação de alto nível e é uma das principais tecnologias para o desenvolvimento *Web*, normalmente denominado de *JS*.

O *JS* é o que permite que as páginas *Web* tenham interação com o utilizador, por essa razão, é um elemento crucial do desenvolvimento *Web*, permite implementar itens com uma maior complexidade em páginas *Web*, é a terceira camada das tecnologias padrão *Web*, atrás do *HTML* e do *CSS*. É atualmente a principal linguagem em termos de *client-side*, ou seja, é executada a partir do lado do cliente, o que não é necessário haver uma ligação a um servidor [45], [41], [19].

O porquê do uso desta linguagem:

- Fácil de aprender;
- Correção de erros de código bastante simplificada;
- Tem um alto grau de compatibilidade;
- Linguagem focada na otimização de páginas Web.

### 3.5.2 *React*

*React.JS* é uma biblioteca de *JavaScipt open-source*, que é usada para o desenvolvimento da interface que vai ser usada/vista pelo utilizador e foi criada por Jordan Walke, um engenheiro de software da empresa *Facebook*, nos dias de hoje chamada de *Meta*.

É uma linguagem declarativa, o que faz com que o código seja mais previsível e fácil de fazer *debug*, é baseada em componentes, que permite com a junção de vários componentes criar uma interface complexa do ponto de vista do utilizador.

Na arquitetura MVC – *Model View Controller* – está inserido na categoria *View*, que é precisamente a parte visual da aplicação. A manutenção desta biblioteca é efetuada pela *Meta* e uma comunidade de programadores individuais. Os componentes são bastante fáceis de escrever, uma vez que o *React* usa *JSX (JavaScriptXML)*, que é uma extensão de sintaxe para o *JavaScript* para que a escrita de código seja muita parecida com *HTML*, para facilitar o trabalho do programador [39], [2], [18].

### 3.5.3 *React Native*

*React Native* é uma biblioteca *JavaScript* criada pelo *Facebook*, e é muito usada para a criação de aplicações *Android* e *iOS*, esta permite a criação de aplicações móveis com a linguagem *JavaScript*. O código que o programador desenvolve é convertido para a linguagem nativa do sistema operacional em que o mesmo deseja ver a sua aplicação final. Neste caso, a biblioteca *React Native* usa *APIs* em *Swift*, necessárias para correr em *iOS* e em *Kotlin* que vão ser necessárias em *Android*. Dito isto podemos ver que uma das grandes vantagens do uso desta biblioteca é o facto de o uso de uma codificação poder ser utilizada nas duas plataformas, o que garante uma maior produtividade na elaboração das aplicações móveis, apesar de haver algumas exceções, a maioria do código feito para *Android* vai funcionar no *iOS* e vice-versa [3], [11].

### 3.5.4 *Node.js*

É um ambiente de execução *JavaScript* de código aberto e de multi plataforma.

O *node.js* corre o mecanismo *V8 JavaScript*, que é o núcleo do *google chrome*. Isto permite que o *node.js* seja muito bom em termos de performance.

Uma aplicação *node.js* corre num processo único, sem criar uma *thread* para cada pedido. Esta ferramenta fornece funcionalidades assíncronas e previne que o código seja bloqueado. As bibliotecas em *node.js* são desenvolvidas usando paradigmas *non-blocking*, o que faz com que os bloqueios sejam considerados as exceções e não as situações normais.

Quando o *node.js* corre uma operação de *I/O*, como ler algum conteúdo da rede, acesso a base de dados ou ficheiros de sistema, ao invés de bloquear a *thread*, o *node.js* retorna a sua atividade normal assim que obtém uma resposta. Isto permite que o *node.js* possa gerir muitas ligações paralelas apenas com um servidor, sem a necessidade de gerir a gestão de *threads*, o que poderia gerar uma grande quantidade de *bugs*.

Existe também a vantagem de muitos desenvolvedores de *frontend* que usem *JavaScript* não terem a necessidade de aprender uma nova linguagem para criar o seu *backend*.

O *node.js* oferece muitas bibliotecas, entre as quais o **Express** que vai ser utilizada no nosso servidor para a criação de um servidor *web* que vai fazer a ligação com o *frontend* da aplicação a ser desenvolvida [1], [48], [8].

## 3.6 CONCLUSÃO

Neste capítulo foi dada a introdução ao *e-Health*, foi feita uma revisão geral daquilo que o mercado atual é capaz de oferecer ao utilizador, a análise de várias aplicações hospitalares, quer sejam hospitais públicos, privados, nacionais ou até internacionais e para além disso, uma breve descrição das tecnologias, metodologias e *frameworks* que serão utilizadas para desenvolver uma solução para o problema descrito. Foram ainda analisados os sistemas operativos dos *smarthphones* aos quais a solução irá remeter o seu foco, *Android* e *iOS*.

Em relação à abordagem multiplataforma e a abordagem nativa, apesar de ambas as abordagens terem bastantes aspetos positivos, nesta dissertação a escolha vai recair sobre a abordagem multiplataforma, e para isso, a *framework* utilizada será o *React Native*. A escolha da abordagem prende-se, essencialmente, ao facto de criar uma aplicação que seja funcional, tanto no sistema operativo *iOS*, como no *Android*.

Para o desenvolvimento da solução pensada, será utilizado o *React* para o desenvolvimento da aplicação dos profissionais de saúde, aplicação *web*, o *React Native* para a aplicação *mobile*, que será usada pelos utentes, o *Node.js* irá ser usado para desenvolver o servidor comun às duas aplicações e o *MySQL* que servirá para armazenar os dados enviados pelo servidor.

O próximo capítulo demonstrará uma visão global da solução desenvolvida.

# 4

# VISÃO GLOBAL DA SOLUÇÃO

## 4.1 FUNCIONALIDADES APLICAÇÃO MÓVEL (UTENTE)

| Funcionalidades |
|---|
| Possibilidade de agendar consultas [1] |
| Visualização das consultas próximas [2] |
| Marcação de exames [3] |
| Capacidade de aceder a um relatório médico e/ou exames médicos [4] |
| Possibilidade de aceder a informações úteis [5] |
| Possibilidade de protótipo de *chatbot* |
| Possibilidade de marcar exame/consulta para membros do agregado familiar [6] |

Tabela 9: Funcionalidades da aplicação *mobile* a desenvolver

Funcionalidades presentes em:

1- *App My São João*, *App CHUPorto*, *App My Luz*, *App Trofa Saúde*, *App MyMediclinic 24x7*

2- *App CHUPorto*, *App My Luz*, *App Trofa Saúde*, *App MyMediclinic 24x7*

3- *App CHUPorto*, *App My Luz*, *App Trofa Saúde*

4- *App My Luz*, *App Trofa Saúde*

5- *App CHUPorto*

6- *App My Luz*, *App Trofa Saúde*, *App MyMediclinic 24x7*

## 4.2 FUNCIONALIDADES APLICAÇÃO WEB (PROFISSIONAIS DE SAÚDE)

- Aceder a consultas que os utentes agendaram para o respetivo médico;
- Capacidade de aceitar/rejeitar consultas, conforme a sua disponibilidade;
- Gerar o relatório de uma consulta feita pelo respetivo médico;
- Submeter o resultado de um exame para que possa ser visto pelo utente na aplicação móvel;
- Consultar o perfil de um utente;
- Consultar o perfil de um profissional de saúde;
- Capacidade de alterar alguns campos do respetivo perfil;
- Caso seja administrador da aplicação, terá a capacidade de aceitar utilizadores, remover, tanto na aplicação móvel como na aplicação *web*.

## 4.3 ARQUITETURA DA APLICAÇÃO

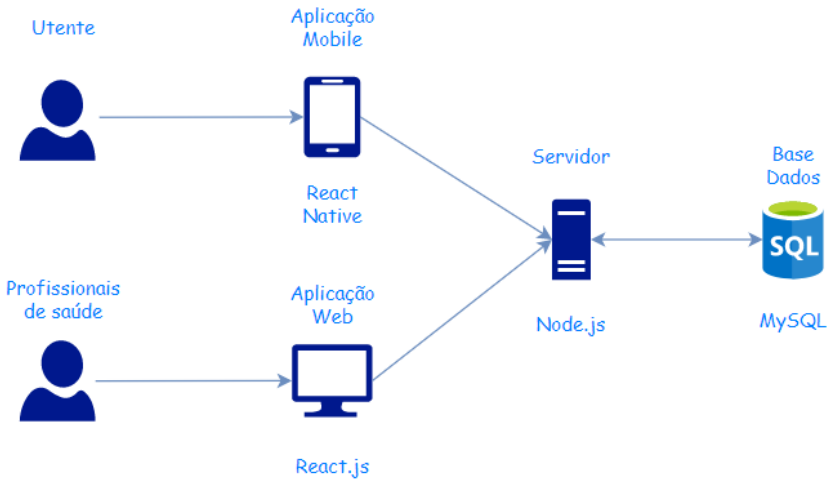



Figura 6: Arquitetura da solução desenvolvida

## 4.4 BASE DE DADOS

A base de dados foi desenvolvida em *MySQL* e tem o objetivo de guardar a informação necessária para o bom funcionamento das aplicações. É uma base de dados relacional e foram criadas várias tabelas com a função de

armazenar os dados. Para uma manipulação mais acessível da base de dados foi usada a aplicação *MySQL Workbench*.

Foram criadas 9 tabelas para guardar as informações pretendidas para a correta utilização das aplicações.

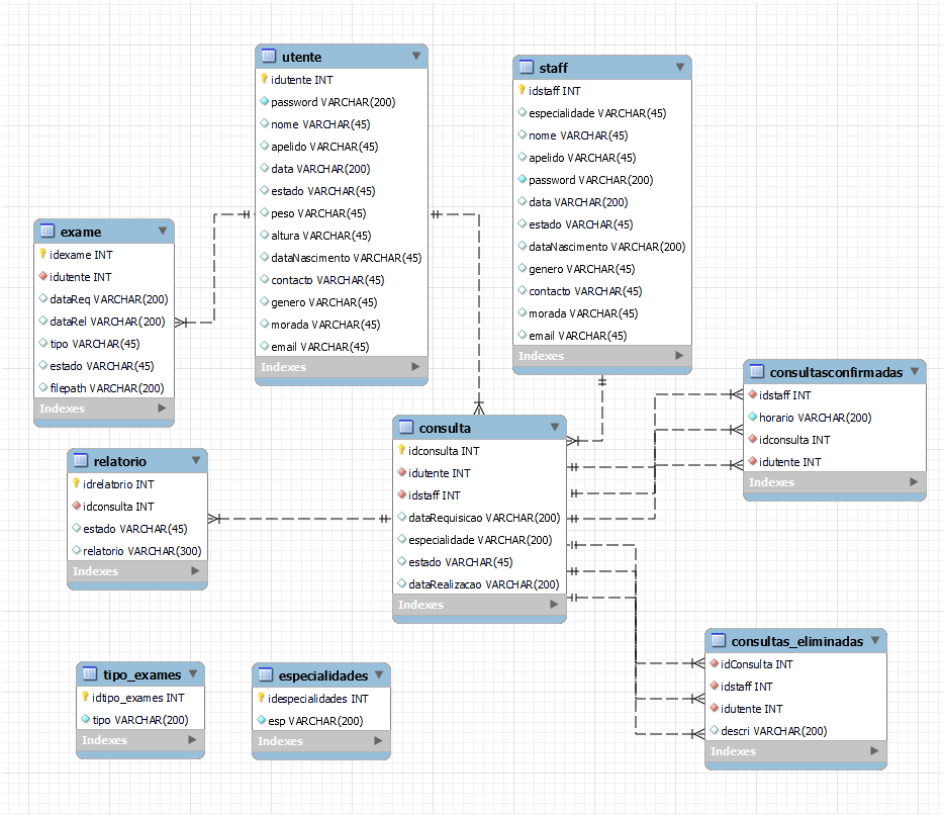



Figura 7: Estrutura base de dados

As tabelas utente e staff são responsáveis por guardar as informações pessoais dos utilizadores das aplicações, a maioria dos campos poderá estar em branco, pois aqueles que são realmente indispensáveis são o *id*, a *password* e o estado, o *id* e a *password* para efeitos de *login* e o estado será responsável por verificar se o respetivo utente/staff tem a devida autorização para aceder à aplicação correspondente. Quando é efetuado um registo os campos **nome** e **apelido** são obrigatórios, e o campo **data** é gerado automaticamente no lado do servidor, uma vez que este serve para armazenar a data em que o utente/staff foi criado. Se for o registo de um membro do *staff* terá também de escolher a **especialidade**. Os restantes campos são de cariz opcional, e apenas poderão ser preenchidos pelo respetivo utilizador no *frontend* que será abordado mais detalhadamente na secção designada para abordar as aplicações.

Nas tabelas exame e consulta servirão para guardar as consultas. Na tabela **consulta** poderemos observar que existem dois campos associados, o **idutente** para saber qual é o utente que marcou esta consulta,

e o **idstaff** para sabermos qual foi o profissional de saúde para o qual o utente marcou a consulta. Para complementar, existe ainda a data em que a consulta foi requerida, a data para o qual o utente pediu para ser efetuada, a especialidade escolhida, e ainda o estado, pois o profissional de saúde ainda terá de aceitar ou não a consulta, se a consulta for aceite, é criada uma instância da mesma com algumas informações na tabela **consultas confirmadas**, se for rejeitada por um médico irá para as **consultas eliminadas**. De referir que sempre que uma consulta é aceite, ao mesmo tempo que entra nas **consultas confirmadas** é gerado um relatório com o **idconsulta** associado.

Já na tabela **exame** há alguns aspetos diferentes, o **idutente** continua presente para que o exame seja associado a um utente, porém não existe o **idstaff**, uma vez que para a realização de um exame não é necessário escolher um médico, as datas funcionam de igual forma, o **tipo** define qual será o tipo de exame pretendido, o **estado** guardará se o exame foi, ou não, realizado. Quando realizado, o ficheiro PDF já estará presente numa das pastas do projeto e o caminho para esse mesmo ficheiro estará guardado no campo **filepath**.

Como referido anteriormente, o relatório é gerado e inserido na respetiva tabela, o **idrelatorio** é igual ao **idconsulta**, o **estado** diz-nos se já foi feito e o campo **relatório** indica as observações feitas pelo médico, depois da consulta. Importa referir que apenas o médico que realizou a consulta poderá aceder ao relatório.

As tabelas de **consultas confirmadas** e de **consultas eliminadas** foram criadas para complementar a tabela de consultas, apresentar ao utente se a consulta foi confirmada ou se foi rejeitada. O principal objetivo delas é disponibilizar algumas informações ao utente, por exemplo, se a consulta for cancelada, é possível ao utente perceber o porquê, se o médico preencher a razão, que é apresentada no *frontend*, e que uma vez preenchida será guardada no campo **descri** das **consultas eliminadas**.

O principal motivo da criação das tabelas **especialidades** e **tipo exames**, foi no pensamento dinâmico da aplicação. Sempre que um utente faz uma requisição de uma consulta ou de um exame, surgem algumas opções para este fazer uma escolha, as opções sugeridas surgem exatamente destas tabelas. Assim como os utentes usufruem deste dinamismo, os profissionais de saúde ao fazer o seu registo também terão de escolher a sua especialidade, que advém desta tabela **especialidades**.

## 4.5 SERVIDOR

O servidor foi desenvolvido em *Node.js* e tem o objetivo de guardar a informação necessária na base de dados e fornecer os dados que são requisitados no *frontend*, tanto na aplicação para os utentes, como na aplicação dos profissionais de saúde.

Para o servidor funcionar de acordo com as especificações enumeradas são necessários alguns *packages* para o seu correto funcionamento. Foram usados:

**mysql** - Tem a função de permitir a interação entre o *Node* e a base de dados *MySQL*.

**jwt** - *Package* com o objetivo de implementar *JSON Web Tokens*.

**express** - É uma *framework* Node.js direcionada para aplicações *web* que fornece várias funcionalidades para aplicações *web* e aplicações *mobile*.

**cors** - É um *package Node.js* que dá a possibilidade ao servidor de permitir ligações CORS (*Cross-origin resource sharing*) com várias configurações.

**multer** - *Middleware* para tratar de *multipart/form-data*, que é usado com o objetivo principal de tratar do *upload* de ficheiros.

**fs** - Módulo que fornece muitas ferramentas para aceder e interagir com o sistema de ficheiros. Não é necessária a sua instalação, uma vez que vem no core do *Node.js*.

**bodyParser** - Objetivo de analisar a integridade de um *body* de um pedido. Tem a função de confirmar se esse mesmo corpo do pedido é confiável.

**path** - Este módulo fornece utensílios para manusear ficheiros e caminhos de diretorias.

**bcrypt** - Uma biblioteca que tem o objetivo de encriptar palavra passes, ou outro tipo de informações que necessitem de encriptação.

### 4.5.1 *Estrutura do servidor*

```
-Servidor
    -Controllers
        controllerAdmin.js
        controllerConsulta.js
        controllerExame.js
        controllerRelatorio.js
        controllerSite.js
        controllerStaff.js
        controllerUtente.js
    -Middleware
        auth.js
    -Routes
        routerAdmin.js
        routerConsulta.js
        routerExame.js
        routerRelatorio.js
        routerSite.js
        routerStaff.js
        routerUtente.js
    config.js
    server.js
```

**Controllers** - Diretoria que guarda os ficheiros que são responsáveis por trocar informações com a base de dados.

**Middleware** - Responsável por garantir a autenticidade dos pedidos efetuados.

**Routes** - Contem as rotas para que os pedidos sejam corretamente feitos.

### 4.5.2 *Mecanismos de segurança*

*Tokens*

Para proteger o servidor e para haver um mecanismo que garanta que os pedidos efetuados ao servidor são protegidos foi usado o *JSON Web Token (JWT)*.

O *JWT* é um *open standard* que define uma maneira compacta e independente de transmitir informação, em segurança, entre duas partes no formato *JSON*. A informação é autêntica e confiável uma vez que é assinada digitalmente. Há algumas formas de assinar estes *tokens*, neste caso específico, foi usada uma palavra chave e com o algoritmo *HMAC* os *JWT* são assinados [30], [28].

Sempre que um utilizador (staff ou utente) efetua o *login* é lhe atribuído um *token* através da função *encode* do *jwt*, para que isso seja possível está escrita uma palavra secreta no ficheiro *config.js* e que é usada para encriptar os *tokens*, e sempre que esse utilizador entrar numa página em que seja necessário efetuar um pedido ao servidor, o *token* é enviado, quer seja como *body* do pedido, como também pode ir no cabeçalho do pedido com o campo *x-access-token*. Cabe ao servidor verificar se o *token* é valido, e assim permitir o acesso às informações requisitadas pelo mesmo.

*Encripatção palavra passe*

Para haver uma maior segurança ao guardar os dados sensíveis dos utilizadores das aplicações foi usada uma função de encriptação de *passwords*, denominada de *bcrypt.js*. Apesar de incorporar um *salt* (pedaço de informação que é usado como um *input* adicional à função para encriptar palavras passes, informação, entre outros) para proteger contra ataques, *bcrypt* é uma função adaptativa, uma vez que ao longo do tempo o número de iterações pode ser aumentado para tornar o processo de encriptação mais lento, o que faz com que esta função continue resistente a ataques *brute-force search* mesmo com o aumento do poder computacional [53], [15].

Existem muitas implementações do *bcrypt* em várias línguas de programação, nomeadamente, *c*, *c++*, *java*, *javascript*, *php*, *python*, entre outras [60].

Neste projeto, a encriptação, serviu essencialmente para guardar as palavra passes encriptadas na base de dados quando há um registo. Quando é efetuado um *login*, é comparada a palavra passe introduzida pelo utilizador com a palavra passe guardada na base de dados, com a ajuda da biblioteca *bcrypt*. O utilizador efetua o seu *login* caso as palavras correspondam.

## 4.6 CONCLUSÃO

Neste capítulo foram apresentadas as funcionalidades que estarão presentes na aplicação destinada aos utentes, aplicação *mobile*, com a referência às aplicações analisadas e estudadas no estado de arte desta dissertação, e as funcionalidades na aplicação *web*, destinada aos profissionais de saúde. Foi apresentada a arquitetura utilizada no desenvolvimento da solução para o problema apresentado inicialmente.

Foi possível visualizar de que forma foi construída a base de dados e uma breve explicação para cada tabela criada, e ainda uma visão geral do servidor das aplicações em que se explicou a sua estrutura, alguns mecanismos de segurança e os *packages* necessários para construir as ligações com o *frontend* e a base de dados.

No capítulo seguinte irá ser abordado, mais especificamente, as funcionalidades mais importantes de cada aplicação criada.

# 5

# FUNCIONALIDADES DAS APLICAÇÕES DESENVOLVIDAS

## 5.1 APLICAÇÃO WEB (STAFF)

Esta aplicação, como foi referido, é destinada aos profissionais de saúde e tem o objetivo de permitir que um médico aceda às consultas que foram marcadas para si, tomar uma ação sobre uma pendente, gerar o relatório da consulta efetuada, submeter resultados de exames, consultar perfis de utentes/médicos, alterar o seu próprio perfil. Esta aplicação serve como um suporte para a aplicação *mobile*.

A aplicação foi desenvolvida em *React* e comunica diretamente com o servidor, a maioria dos pedidos está protegido com a assinatura digital *JSON Web Token*.

Importa referir que esta aplicação não tem qualquer comunicação direta com a aplicação destinada aos utentes, pois poderia pôr em causa a segurança e a autenticidade das mesmas.

### 5.1.1 *Gestão de permissões*

Para haver um maior controlo das aplicações e para que se tenha a certeza de que cada novo utilizador das aplicações é definitivamente um utente/profissional de saúde do hospital, foi criado um perfil em que a sua responsabilidade é de administração da plataforma. Este utilizador terá controlo sobre todos os utilizadores das aplicações, mas para fazer essa gestão apenas o conseguirá na aplicação *web*, destinada aos profissionais de saúde.

O administrador da aplicação terá um papel diferente do resto dos utilizadores das aplicações, uma vez que o seu objetivo apenas vai passar por remover utentes ou profissionais de saúde, ou permitir a estes a utilização da respetiva aplicação.

Na figura seguinte podemos ver dois botões adicionais, que não é possível visualizar num profissional de saúde comum, o botão encarnado serve para a remoção de um utilizador, e o azul para a aprovação desse mesmo utilizador.

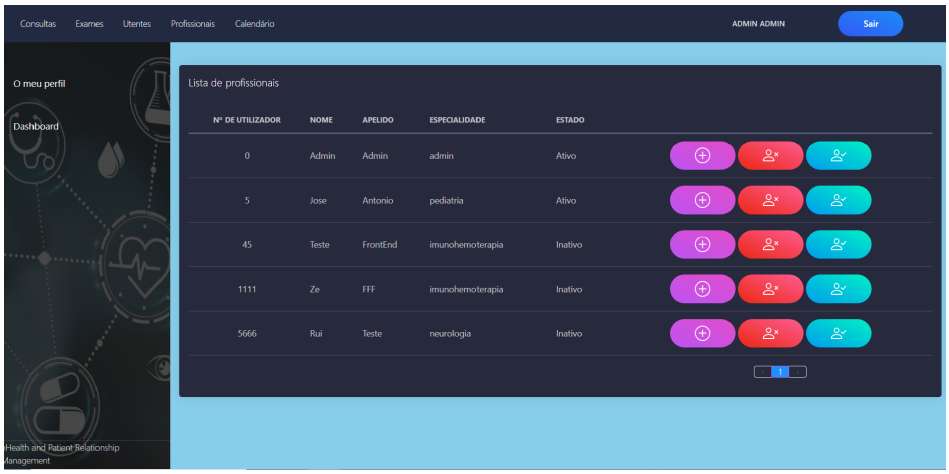



Figura 8: Lista staff prespetiva Admin

O utilizador *Admin* funciona como um *superuser* que tem a possibilidade e a função de permitir o acesso ás aplicações dos utentes e dos profissionais de saúde, também tem a possibilidade de apagar qualquer um deles.

```
app.delete('/utente/:id');
```

A *route* acima é chamada para proceder à eliminação de um utente.

Com o mesmo intuito foi criado um método semelhante, mas para apagar elementos do staff, como podemos ver na *route* abaixo.

```
app.delete('/staff/:id');
```

Estas duas funções, recebem um *id* do pedido que foi efetuado e vão à base de dados, procuram o id que foi inserido, tanto na tabela de utente, como na tabela de staff, e aplicam a *query* de *DELETE* para eliminar o id correspondente da tabela a que corresponde o pedido.

Além da possibilidade da remoção de utentes e staff, o administrador tem também a responsabilidade de decidir se um utente/staff tem permissão para aceder à respetiva aplicação.

```
app.put('/utente/:id');
app.put('/staff/:id');
```

Já na aprovação, é um pouco diferente, tanto a tabela utente, como a tabela staff, têm um campo, denominado de estado, que serve exatamente o propósito de verificar se está aprovado ou não, a usar a aplicação.

Esta função têm o objetivo de aprovar e por isso, atualiza o estado, com a *query* UPDATE do staff/utente para "Ativo". Ao realizar esta operação o *admin* está a permitir que o utilizador tenha acesso à aplicação correspondente.

### 5.1.2 *Consultas e Relatórios*

Nesta página, o médico poderá efetuar ações sobre as consultas que estão marcadas para o mesmo realizar. Uma consulta só poderá ser cancelada, caso ainda não tenha sido validada, e só poderá efetuar o relatório da mesma apenas quando o estado da consulta seja validado.

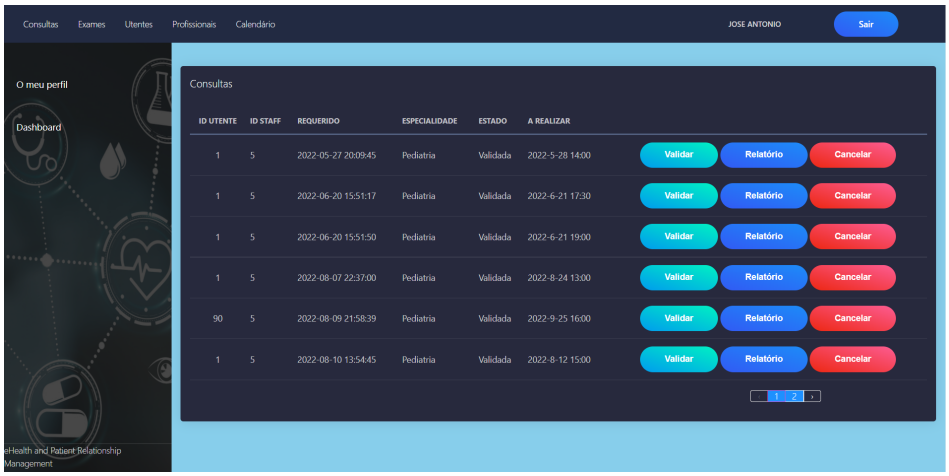



Figura 9: Página Consultas

Foram criadas diversas funções uma vez que as aplicações têm o seu foco direcionado precisamente para as consultas/exames.

Foi criada uma função para eliminar uma consulta, o procedimento é bastante simples, primeiramente começa por inserir na tabela das consultas eliminadas o **idconsulta**, **idstaff** (médico que iria realizar a consulta) e o **idutente**(utente que pediu a consulta), atualiza a **descri**, que como referido anteriormente, serve como a razão pela qual a consulta foi rejeitada pelo médico, e depois de inserirmos a consulta na tabela de consultas eliminadas, a consulta é eliminada da tabela de consultas principal. O médico ao eliminar uma consulta é chamado o seguinte roteamento:

```
app.post('/elimina-consulta');
```

Já na validação de uma consulta, há uma atualização do **estado** na tabela principal, ao mesmo tempo é criado um relatório, na tabela de relatórios para esta mesma consulta e ainda há a adição dos parâmetros desta consulta à tabela das consultas confirmadas. Pode ser chamada assim:

```
1 app.post('/consulta/valida');
```

Depois da consulta ser validada, o utilizador *logado* poderá aceder ao relatório e escrever qualquer observação que ache pertinente, como é possível observar na figura seguinte. O relatório é guardado na base de dados e ficará disponível na aplicação destinada aos utentes para poder ser visualizado.

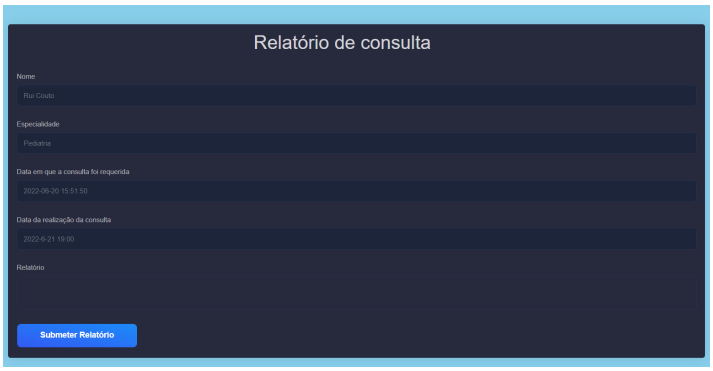



Figura 10: Relatório da consulta

```
1 app.put('/relatorio');
```

Ao fazer este pedido é necessário que o *body* do pedido esteja preenchido para que o seu conteúdo seja atualizado para a base de dados e seja possível, mais tarde, ser visualizada na aplicação destinada aos utentes.

### 5.1.3 *Exames*

Quanto aos exames, qualquer utilizador da aplicação *web* tem autorização para fazer *upload* de um exame, a aplicação apenas aceita ficheiros em formato *PDF*, e uma vez submetido, não é possível alterá-lo.

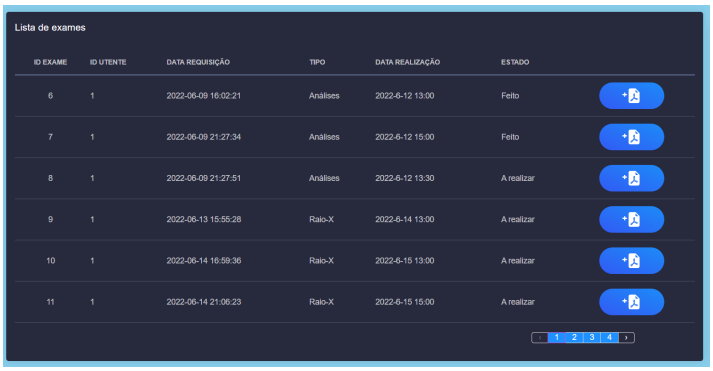



Figura 11: Lista exames

Na figura acima podemos ver a lista de exames marcados na unidade hospitalar, para submeter um ficheiro terá de pressionar o botão azul em frente ao exame pretendido.

Ao dar *upload*, o ficheiro segue o roteamento:

```
app.post('/image/:id');
```

Em que o *id*, é o número de utente. Depois do *upload* os ficheiros são guardados numa pasta destinada para o efeito, além de que o *filepath* também é guardado, com a intenção de quando o utente quiser aceder ao seu exame, caso já tenha sido realizado, lhe seja possível. O nome atribuído ao exame é o número do exame (*id exame* da tabela da base de dados)

### 5.1.4 *Dashboard*

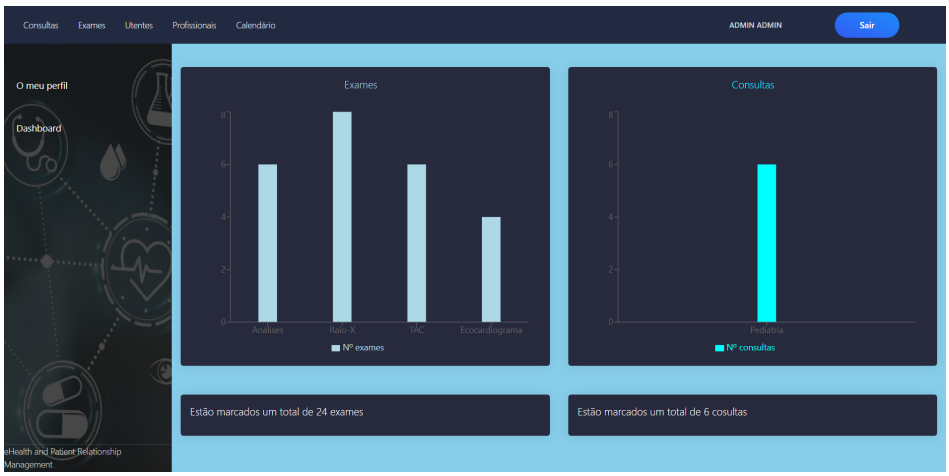



Figura 12: *Dashboard*

Na *dashboard* temos algumas informações tais como o número total de exames, o número total de consultas, gráficos, um deles com os tipos de exame marcados e o outro com o tipo de consultas marcadas.

Para ir buscar estes números é possível através das chamadas seguintes:

Gráfico dos exames

```
app.get('/nexames');
```

Gráfico das consultas

```
app.get('/nconsulta');
```

Número de exames

```
app.get('/numero-exames');
```

Número de consultas

```
app.get('/numero-consultas');
```

## 5.2 APLICAÇÃO MOBILE (UTENTES)

A aplicação mobile é a aplicação central desta dissertação. As principais funcionalidades encontram-se nesta aplicação e tudo o que foi feito anteriormente serve como suporte precisamente para esta aplicação.

Foi feita com o intuito de ser utilizada pelos utentes, para facilitar a relação dos mesmos com a sua unidade hospitalar, melhorar o atendimento e tentar esclarecer o utente da melhor forma possível.

As funcionalidades principais passam por:

- Marcação de consultas;
- Marcação de exames;
- Verificar calendário de exames e consultas;

- Acesso a relatório de consulta;
- Acesso a resultado de exame;
- *Chatbot*.

### 5.2.1 *Marcação de consultas*

Nesta secção irá ser explicado e exemplificado como, através da aplicação, é possível marcar uma consulta.

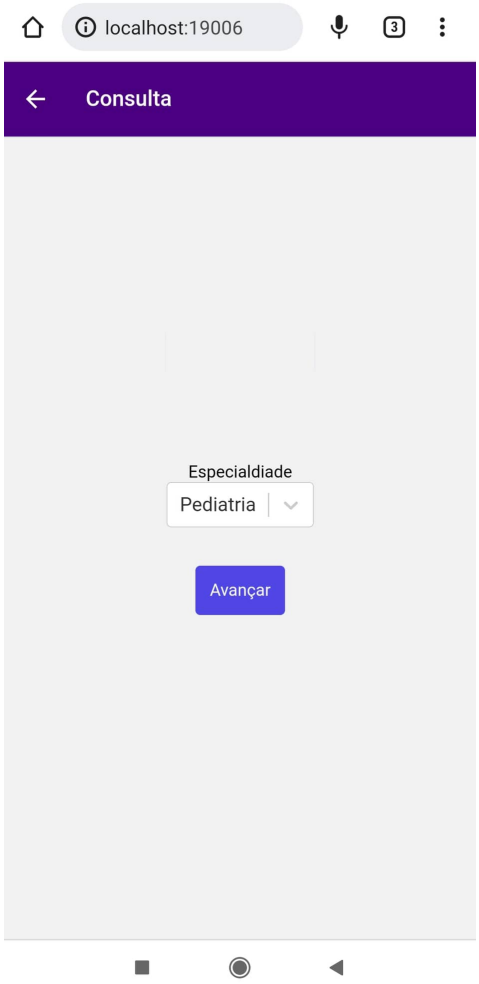



Figura 13: Seleção da especialidade

Na figura acima vemos a primeira página, após o utilizador pressionar o botão de marcação de consulta. Aqui terá de escolher uma das especialidades oferecidas pela unidade hospitalar.

Como visto anteriormente, os tipos de especialidade estão guardados na base de dados, para que sempre que seja necessário retirar ou adicionar uma nova especialidade, esta seleção seja feita de forma dinâmica. O pedido ao servidor é feito da seguinte forma:

```
1 app.get('/esp');
```

Depois de escolhida a especialidade desejada, avançará para a página mostrada na figura abaixo, para escolher a data e o horário pretendido.

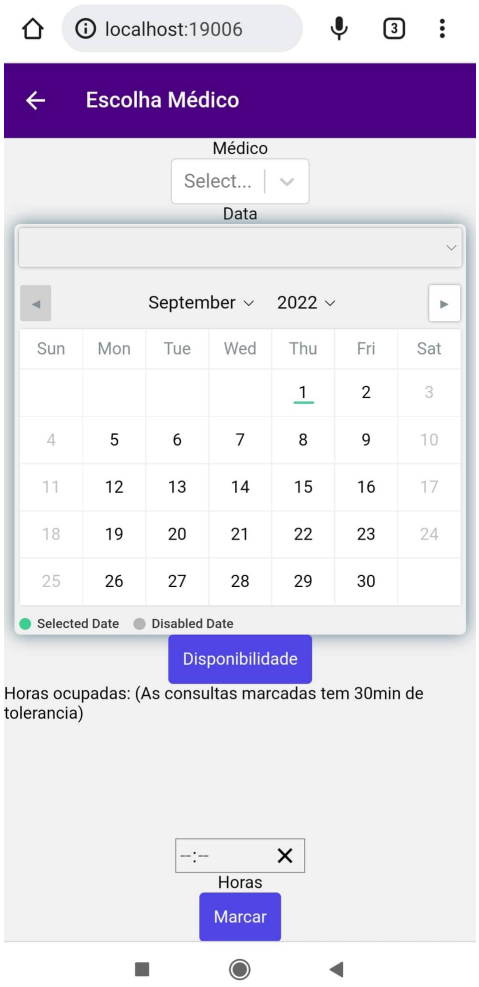



Figura 14: Seleção do horário de consulta

Na seleção da data e do horário , já existem algumas restrições, terá de haver um médico, um dia e uma hora selecionada. Algumas especialidades poderão ainda não ter médicos associados o que faz com que seja impossível marcar uma consulta para essa especialidade.

Nas datas não é permitido marcar para dias passados, nem para os fins de semana e só é permitido marcar consultas no espaço periódico de um ano.

Quanto às horas, só poderá haver marcações entre as 09:00 e as 18:00, que é normalmente, o horário de trabalho dos médicos.

Depois de marcar as consultas ainda terá de obter a aprovação do respetivo médico.

Importa também referir que cada consulta marcada terá um espaço de 1 hora reservado, ou seja, quando uma consulta é marcada para as 15:30, é impossível a outro utente marcar uma consulta entre as 15:01 e as 15:59 para que as consultas não se sobreponham.

Há a funcionalidade de um utente poder consultar a disponibilidade de um médico para um certo dia, ao selecionar o médico desejado, a data para quando quer a consulta e após isso, pressionar o botão de disponibilidade.

Quando não existe nenhuma marcação há uma mensagem de que não existe nenhuma consulta marcada. O utilizador é livre de escolher qualquer hora do dia para efetuar a consulta desejada.

Ao efetuar uma marcação de uma consulta a sequência é a seguinte:

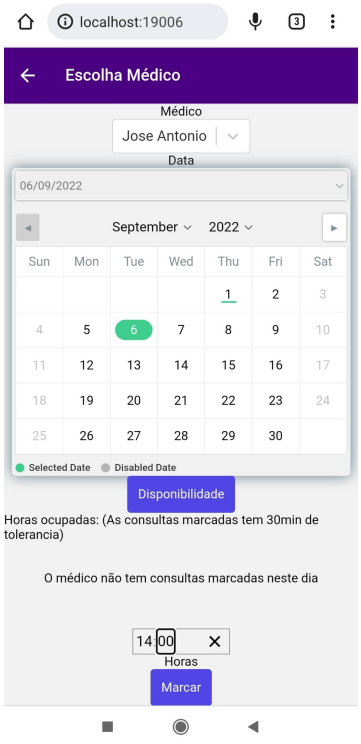



Figura 15: Marcação de consulta

Ao pressionar o botão de marcação de consulta, é enviado um pedido ao servidor:

```
1 app.post('/consulta');
```

Neste pedido, no cabeçalho encontra-se o *header x-access-token* para assegurar que o pedido está digitalmente assinado, e no *body* encontra-se o número de utente, o número do profissional de saúde, a especialidade e a data.

No lado do servidor há uma verificação na base de dados se os campos primários não têm nenhuma sobreposição, depois há uma verificação das datas para assegurar o intervalo de 30 minutos antes e depois da hora marcada.

Caso esteja tudo correto é devolvida a mensagem **Consulta inserida mas à espera de validação**. Quando há uma tentativa de marcar uma consulta com um intervalo inferior a 30 minutos de uma já marcada há a devolução da mensagem **A data escolhida já se encontra ocupada**.

### 5.2.2 *Marcação de exames*

Outra funcionalidade da aplicação mobile passava pela marcação de exames. Nesta secção irá ser demonstrado como o fazer.

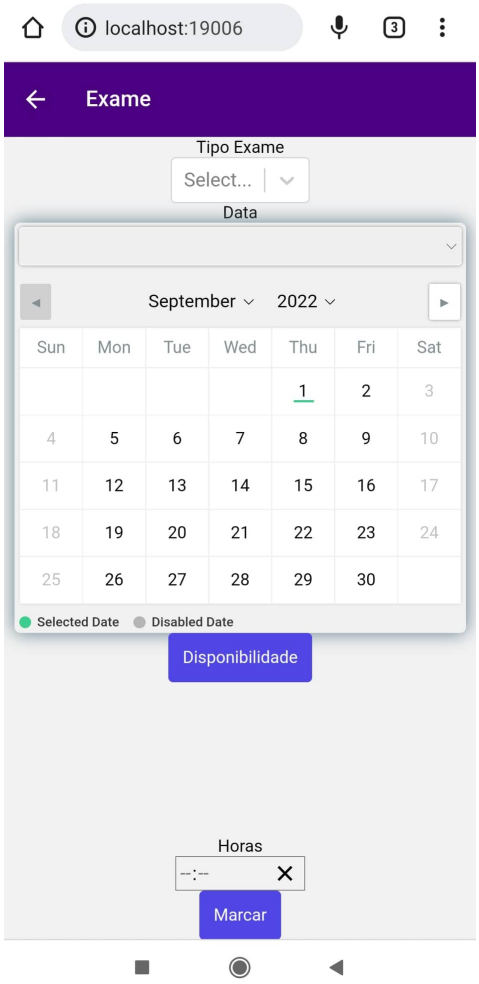



Figura 16: Página de marcação de exame

Na figura acima vemos a página da marcação de exames, é muito parecida com a de marcação de consultas, há a diferença de que a seleção do tipo de exame é escolhida na mesma página, uma vez que não é necessário a escolha de nenhum médico. Quanto à escolha de datas para a marcação do exame funciona do mesmo modo do que o de marcação de consultas.

Não é possível marcar para os fins de semana e o intervalo horário é das 9:00 às 18:00.

Só pode ser marcado um tipo de exame para a mesma hora, por exemplo, se for marcado um TAC para as 15:00 do dia 03-09-2022, durante as 14:31 e 15:29 não poderá ser marcado outro TAC por outro utente qualquer.

De realçar que, tal como nas consultas, sempre que um exame é marcado é reservado um espaço temporal de 30 minutos, após e antes do exame.

Para escolher o tipo de exames disponíveis para os utentes, é feito um pedido ao servidor através da rota:

```
app.get('/tipo-exame');
```

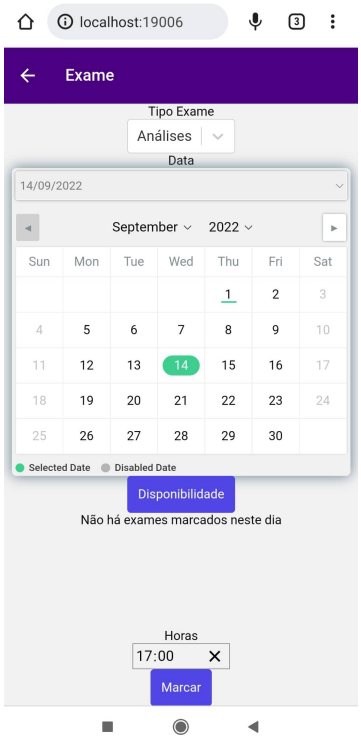



Figura 17: Marcação de exame

Quanto à finalização da marcação de exame, após todos os campos estarem preenchidos há a chamada da rota, quando o botão marcar é clicado

```
app.post('/exame');
```

Ao enviar o pedido ao servidor, o *token* vai no cabeçalho do pedido e no seu corpo encontra-se o número de utente, o tipo de exame e a data.

Se não houver nenhum erro, as datas escolhidas estejam disponíveis o servidor envia como resposta a mensagem **Exame marcado**.

### 5.2.3 *Tabelas de exames e consultas*

Nesta secção irão ser abordadas as tabelas construídas para listar todos os exames/consultas requeridos pelo utente.

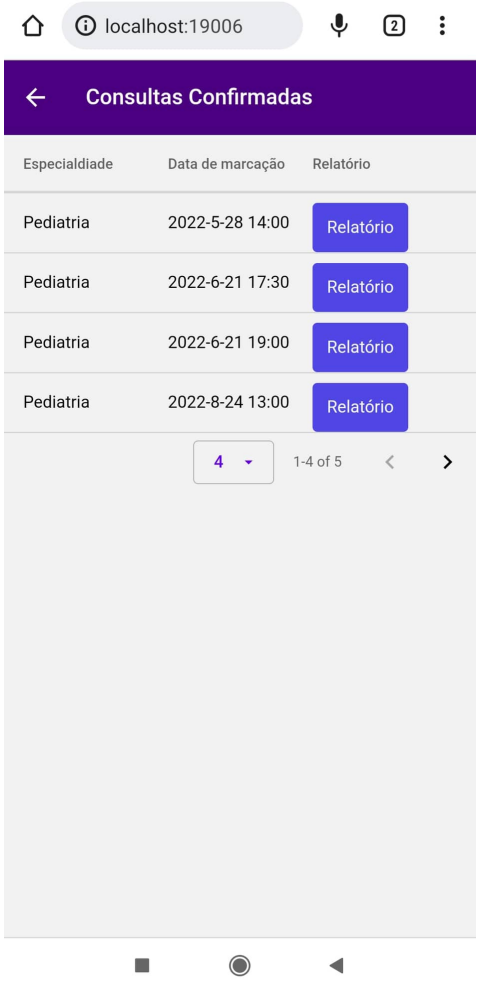



Figura 18: Tabela de consultas confirmadas

Aqui podemos ver a tabela das consultas que foram confirmadas, sempre que existe a confirmação de uma consulta há a criação do espaço para o seu respetivo relatório. O relatório é sempre possível de ser aberto, poderá é ainda estar em branco, caso o respetivo médico ainda não tenha escrito qualquer informação. Para recolher a informação necessária para preencher a tabela é necessário seguir o caminho:

```
1 app.get('/consultaConfirmada/:idUtente');
```

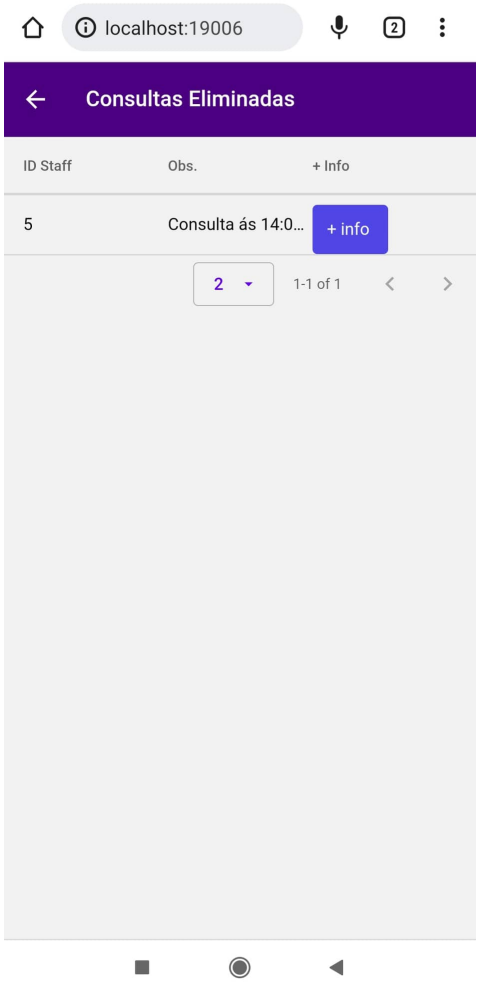



Figura 19: Tabela de consultas eliminadas

Quanto às consultas eliminadas, são inseridas nesta tabela depois de haver uma rejeição da consulta por parte do respetivo médico. Apenas é possível observar o ID do médico e a observação feita pelo mesmo, que serve como a razão para o cancelamento da consulta. Uma vez que normalmente a observação será um pouco extensa foi decidido criar uma página para que o utente pudesse ler o texto completo.

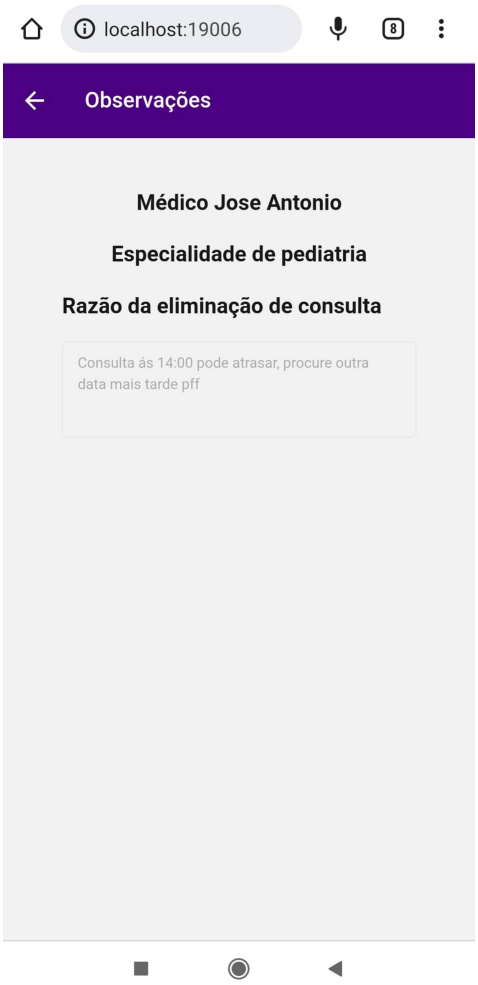



Figura 20: Página de observações

Tanto como no preenchimento da tabela das consultas eliminadas, como na informação adicional acerca da razão pela qual uma consulta foi eliminada, ao servidor é feito o seguinte pedido:

```
1 app.get('/consultaEliminada/:idUtente');
```

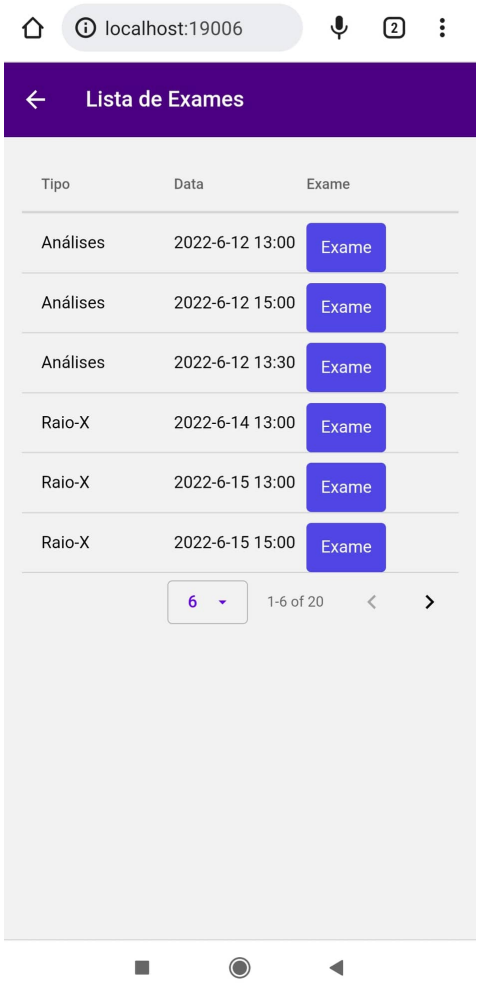



Figura 21: Tabela de exames

Quanto à lista de exames é possível ao utente ver o tipo de exame e a data em que foi realizado, apenas poderá acessar ao exame caso o mesmo tenha sido submetido na base de dados por um profissional de saúde. Se ainda não foi submetido nenhum exame, aparece um aviso ao utilizador que o exame ainda não está disponível.

Para obter informações sobre o exame, o pedido é:

```
app.get('/exame/:idUtente');
```

### 5.2.4 *Relatório de consulta*


Figura 22: Página do relatório

No relatório da consulta, como já tinha sido visto anteriormente, está apenas disponível depois de uma consulta ter sido validada. O médico ao confirmar uma consulta, no lado do servidor é automaticamente criado um relatório e é esse mesmo relatório que o utente poderá aceder.

Ao abrir a página do relatório o utente poderá ver quem foi o médico que o atendeu, a especialidade e a data em que a consulta se realizou, assim como o que o médico escreveu no seu relatório na aplicação destinada aos profissionais de saúde (aplicação *web*).

Para aceder ao relatório o pedido ao servidor é feito da seguinte forma:

```
1 app.get('/consulta/info/:idConsulta');
```

### 5.2.5 *Resultado de exame*

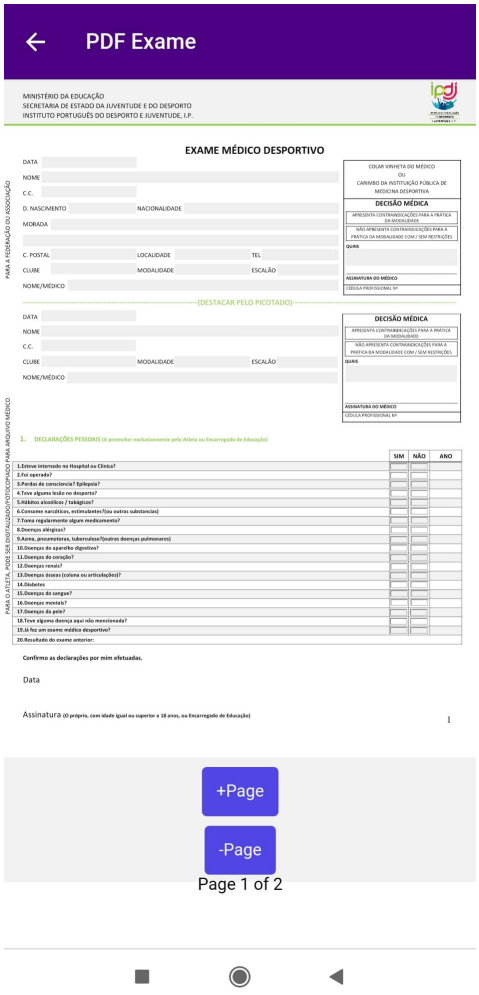



Figura 23: Ficheiro PDF do exame

A partir da tabela de exames é possível chegar a esta página, mas para que esta página esteja disponível é necessário que o exame já tenha sido realizado, caso contrário, não é possível aceder à página do resultado do exame.

Na exibição do resultado de exame, é um pouco diferente, uma vez que se trata de um ficheiro *PDF*. Para controlar a passagem de página e o seu retrocesso foram criados dois botões com essa funcionalidade.

O acesso ao ficheiro está guardado na base de dados, na tabela destinada ao exame. Quando um exame é realizado o seu estado é alterado. Há um campo denominado *filepath* para que o caminho para o ficheiro seja guardado. É a este campo que a página vai buscar o caminho para renderizar o ficheiro pretendido.

### 5.2.6 *Chatbot*

A criação do *chatbot* foi a funcionalidade implementada diferenciada das aplicações analisadas no estado de arte desta dissertação. O *package* descarregado para a utilização do *chatbot* foi o *react-native-chatbot-expo* [4], que, por sua vez, foi inspirado no *react-simple-chatbot* [5], mas com as condições necessárias para funcionar corretamente no *react native*.

Através da documentação oferecida foi possível perceber que o *chatbot* oferece a capacidade de criar passos, foi por esse meio que este *chatbot* foi implementado.

De salientar também, que a maioria dos passos gerados, uma vez que era necessário ir buscar informação ao servidor, foi gerado por renderização de componentes para que fosse possível haver uma maior dinâmica no *chatbot*.

*Menu Inicial*

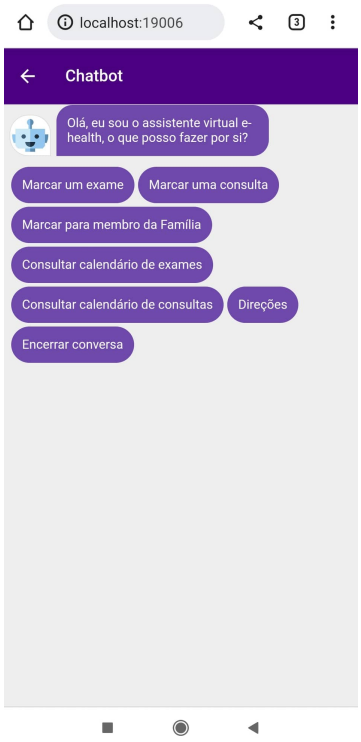



Figura 24: Opçoes do *chatbot*

No menu inicial existem várias opções, cabe ao utilizador escolher qual das funcionalidades irá querer utilizar.

*Marcação de exame*

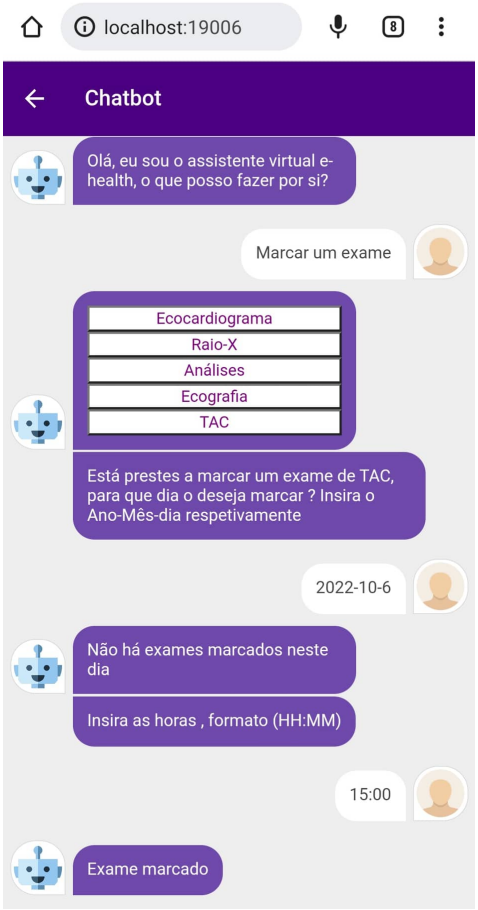



Figura 25: Marcação de exame através do *chatbot*

Na marcação de um exame através do *chatbot*, há uma renderização de componentes em que vai buscar os tipos de exame disponíveis através do pedido

```
1 app.get('/tipo-exame');
```

Depois de fazer este pedido, há a criação de alguns botões para que o utente escolha o tipo de exame pretendido. Após a sua escolha, é necessário inserir a data pretendida e com esta data é feito outro pedido ao servidor

```
1 app.get('/agenda/exames/:tipo/:dia');
```

Este pedido, caso haja algum exame desse tipo nesse dia, irá retornar a data e a hora do exame marcado para que não haja sobreposição de exames. Neste caso não há nenhum exame TAC marcado nesse dia, portanto a mensagem retornada foi **Não há exames marcados neste dia**. Após isso é pedido a hora no formato correto e depois de a inserir é enviado ao servidor o pedido de marcação de exame

```
app.post('/exame');
```

Se tudo estiver inserido corretamente a mensagem é enviada pelo servidor e escrita pelo *chatbot* **Exame Marcado**.

*Marcação de consulta*

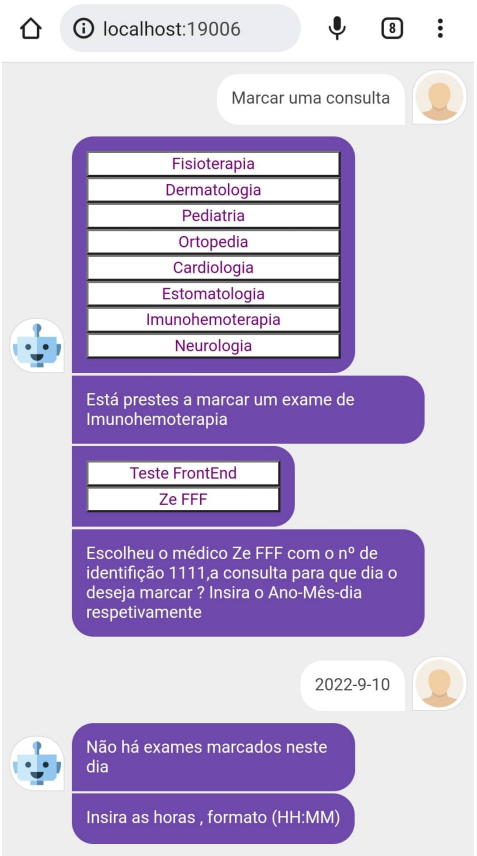



Figura 26: Marcação de consulta através do *chatbot*

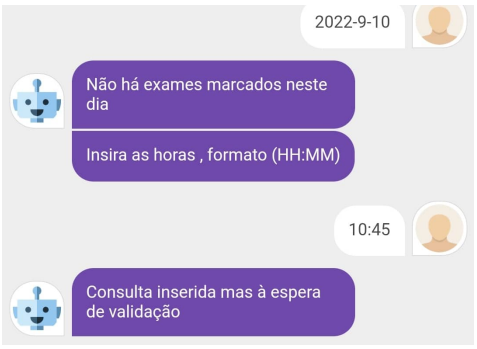



Figura 27: Continuação da marcação de consulta através do *chatbot*

A marcação de consultas segue a mesma lógica da marcação de exames, há um pedido ao servidor das especialidades disponíveis

```
app.get('/esp');
```

Depois de haver este pedido há outro para saber quais são os médicos da especialidade escolhida, caso não haja médicos dessa especialidade é impossível continuar.

```
app.get('/staff');
```

O pedido é feito como está descrito acima, mas depois de executar o pedido, há uma verificação se a especialidade do médico coincide com a especialidade escolhida. Ao escolher o médico, é pedido para inserir uma data, o pedido ao servidor é feito para verificação da disponibilidade do médico nesse dia

```
app.get('/agenda/:id/:dia');
```

Após esta sequência de pedidos, o utilizador terá de inserir as horas para o qual quer marcar a consulta e irá ser feito um pedido ao servidor com todas as informações necessárias.

```
app.post('/consulta');
```

Se a consulta for marcada com sucesso, surgirá a mensagem **Consulta inserida mas à espera de validação**, significa que foi inserida na base de dados, só está pendente da aprovação do respetivo médico.

*Marcação para membro do agregado familiar*

Na marcação de exames/consultas para membros do agregado familiar, os primeiros dados que o utilizador terá de inserir é o número de utente do familiar para o qual pretende marcar a consulta. Quando isto acontece é enviado o seguinte pedido ao servidor

```
app.put('/utente/id');
```

Este pedido retorna a informação de um certo utente. No caso do pedido gerar uma resposta em branco significa que não existe nenhum utente com esse número e surge a mensagem de que não existe nenhum utente com esse número registado, no caso oposto, em que há reconhecimento do número do utente o utente pode seguir em frente e marcar uma consulta ou um exame para o seu familiar. A marcação tanto da consulta como do exame, seguem exatamente os mesmos passos como na marcação para si mesmo. O único aspeto que vai mudar é apenas quando há o pedido

```
app.post('/consulta');
OU
app.post('/exame');
```

Ao executar os pedidos acima, o número de utente enviado, ao invés de ser o número do utilizador da aplicação, será o número que o utilizador inseriu no inicio da conversa com o *chatbot*.

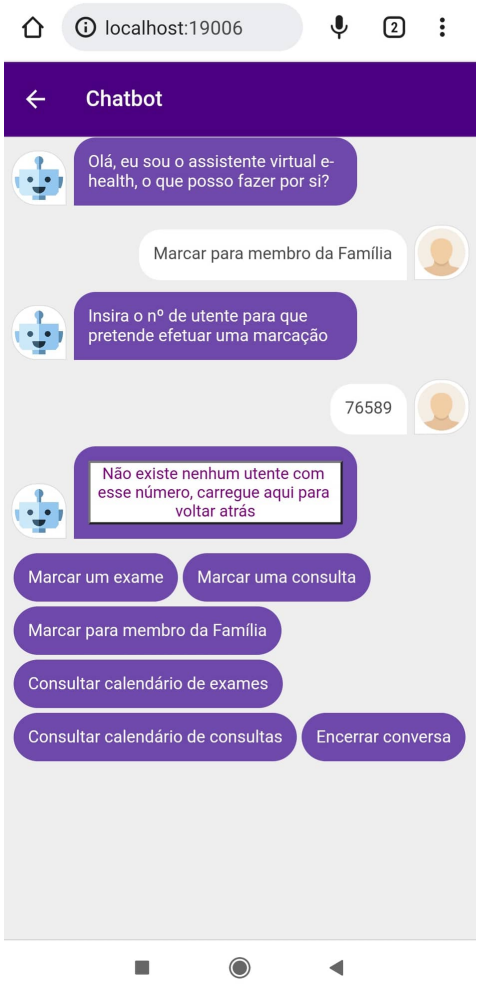



Figura 28: Marcação para membro do agregado familiar inválida

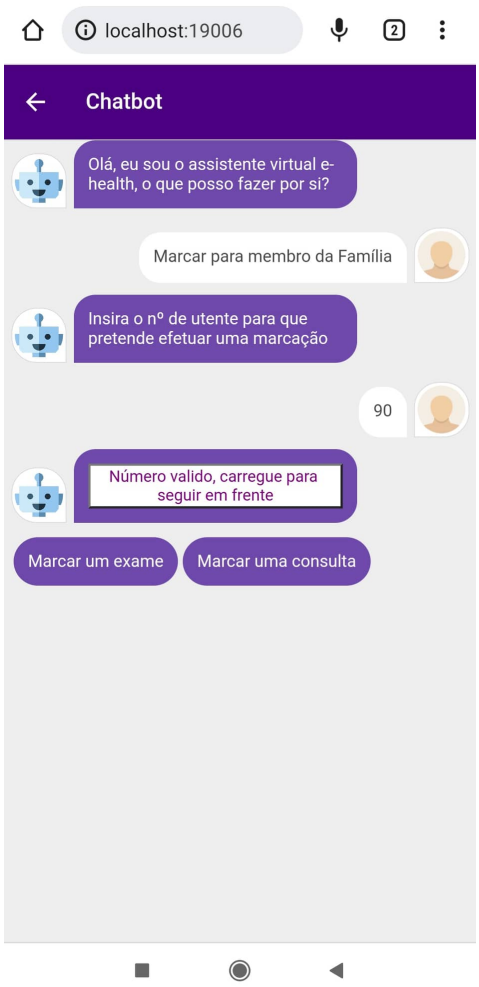



Figura 29: Marcação para membro do agregado familiar

*Calendário exame/consulta*

O objetivo principal desta funcionalidade era mostrar quantos exames/consultas tem um utente num determinado dia. Para isso foi requerido ao servidor as consultas/exames do respetivo utente e foi feito um algoritmo para inserir essas datas no calendário.

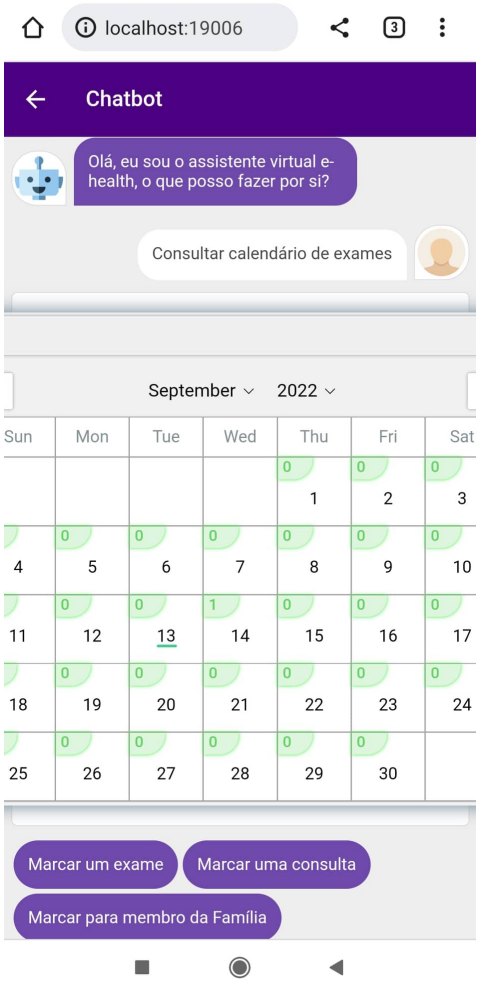



Figura 30: Calendário de exames

*Direções*

Ao pedir as direções, ao utilizador é lhe mostrado um botão. Ao pressionar este botão irá ser aberto o *Google Maps* em que o destino (neste caso, foram inseridas as coordenadas para a universidade do Minho) irá aparecer centrado.

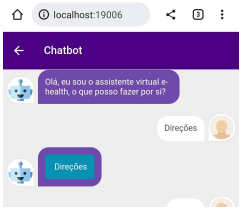



Figura 31: Obter direções

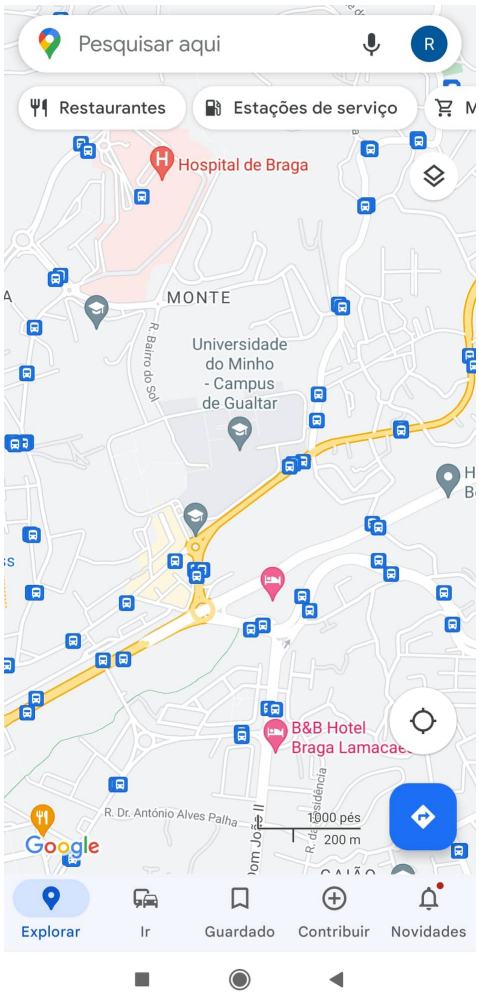



Figura 32: Google Maps com as coordenadas centrais

# 6

# PROVA DE CONCEITO E ANÁLISE SWOT

## 6.1 PROVA DE CONCEITO

A prova de conceito ( *Proof of concept* (POC) ) é um exercício no qual o foco é determinar se uma ideia pode ser tornada realidade. Esta prova tem o objetivo de testar a viabilidade de uma ideia, ou se essa ideia vai funcionar da maneira para o qual a mesma foi criada.

A prova de conceito não tem a intenção de explorar as exigências de mercado nem de determinar processos para a melhor produção possível, apenas se foca na oportunidade de explorar o potencial da ideia a ser desenvolvida [52].

Alguns dos benefícios do POC são [37]:

- Oferecer informações a possíveis investidores se um projeto, ou a ideia de um produto é viável e quão valioso será para o público alvo;
- Informar as equipas de desenvolvimento *feedback* do utilizador, público alvo e algumas falhas que poderão ser críticas que tenha escapado na fase de desenvolvimento;
- Apesar de não ser o seu foco, o POC poderá oferecer informações acerca da exigência do mercado e acaba por funcionar como uma base para o projeto final.

De modo a colocar o *POC* à prova foi feita uma demonstração do que sucede nas aplicações quando são executadas as principais funcionalidades implementadas.

### 6.1.1 *Marcação consulta*

Após a marcação da consulta, na tabela destinada às consultas pendentes, podemos observar as consultas marcadas. Enquanto que o médico não valida uma consulta ou a rejeita, esse registo de pedido de marcação de consulta irá permanecer na tabela de consultas pendentes da aplicação *mobile*.

Nas figuras abaixo, à esquerda está visível a consulta na aplicação *web* que é possível ser observada pelo médico para o qual a consulta foi marcada, neste caso, José António. O médico terá os botões Validar, Relatório e Cancelar para tomar a ação desejada.

À direita vemos a inserção do registo de consulta na tabela de consultas pendentes. Isto acontece quando a consulta é marcada, o utente terá de esperar pela validação do médico para que a consulta seja realizada.


Figura 33: Tabela consultas médico aplicação web

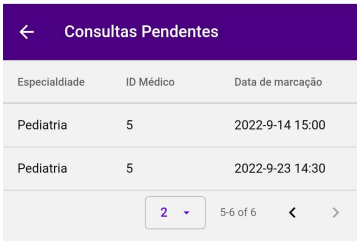



Figura 34: Tabela consultas pendentes aplicação mobile

### 6.1.2 *Marcação exame*

Ao marcar um exame o utente terá de escolher, o tipo de exame pretendido, o dia e a hora, tal como foi demonstrado na secção da marcação de exame no capítulo anterior.

Ao inserir os dados necessários e ao pressionar o botão de marcação de exame, o pedido é enviado ao servidor e inserido na base de dados. Após isso é possível observar o exame na tabela de exames do utente e na tabela de exames da aplicação *web*.

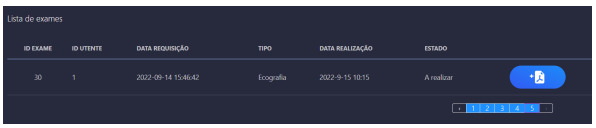



Figura 35: Tabela exames aplicação web

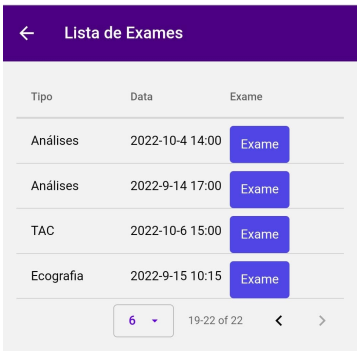



Figura 36: Tabela exames aplicação mobile

### 6.1.3 *Relatório*

A escrita do relatório faz parte das funcionalidades permitidas aos médicos. Para que isso aconteça é necessário que o médico para o qual a consulta foi marcada, primeiramente valide a consulta e posteriormente poderá escrever as informações que achar pertinentes no relatório da consulta.

Após a escrita do relatório é possível ao utente ver o que o médico escreveu na figura à direita. Na figura à esquerda podemos ver como o médico poderá preencher o relatório.

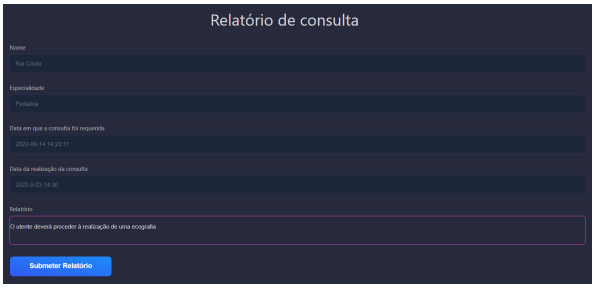



Figura 37: Espaço reservado para a escrita do relatório da consulta


Figura 38: Relatório aplicação mobile

### 6.1.4 *Exame*

No *upload* de um resultado de um exame, qualquer profissional de saúde com acesso à aplicação *web* poderá fazê-lo. Terá de submeter um ficheiro PDF e o servidor terá a função de guardar esse ficheiro e guardar o caminho do ficheiro na base de dados para que posteriormente a aplicação *mobile* lhe consiga aceder.

Após a sequência de passos descritos acima, podemos ver na figura seguinte, a visualização do ficheiro submetido.

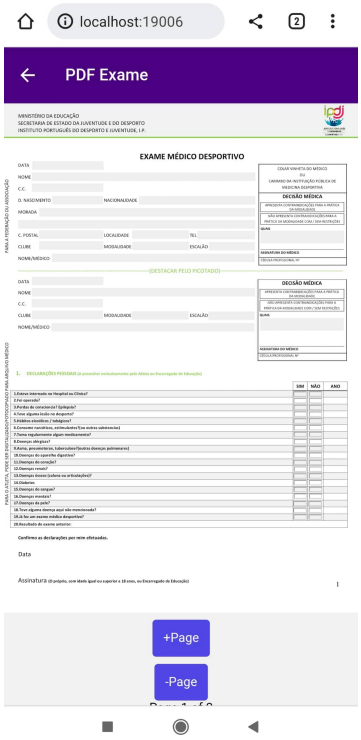



Figura 39: Resultado do exame na app mobile

### 6.1.5 *Marcação consulta chatbot*

Uma marcação de consulta através do *chatbot* requer exatamente as mesmas informações do que a marcação normal, pede a especialidade, o médico pretendido, a data e a hora, estas informações terão de ser preenchidas pelo utente para se poder efetuar a sua marcação.

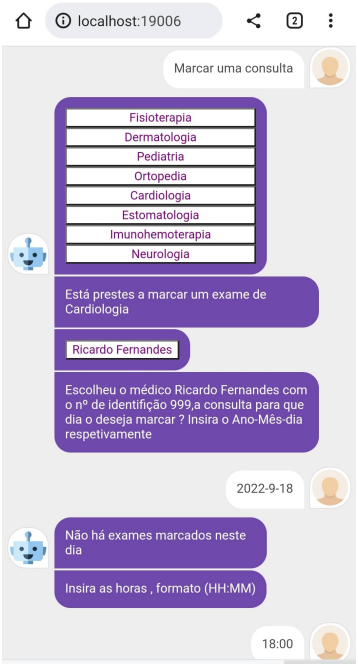



Figura 40: Marcação de consulta através do *chatbot*


Figura 41: Tabela consultas após marcação consulta pelo *chatbot*

Após a marcação da consulta pelo *chatbot* vemos nas figuras acima, à direita na aplicação *web* a tabela de consultas e à esquerda o modo de como é efetuada a marcação.

### 6.1.6 *Marcação exame chatbot*

O método de marcação é o mesmo, em que o utente terá de escolher o tipo de exame, a data e a hora, por esta ordem.

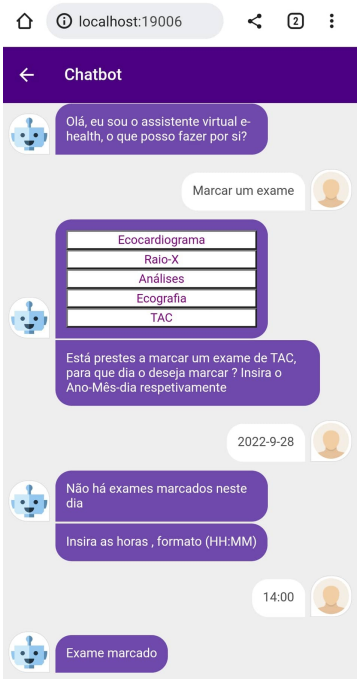



Figura 42: Marcação de exame através do *chatbot*

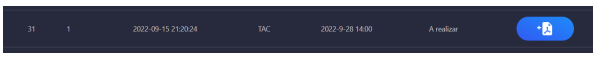



Figura 43: Tabela exames após marcação *chatbot*

Depois de efetuada a marcação, o exame é inserido na tabela de exames quer da aplicação *web*, quer na base de dados na tabela correspondente.

## 6.2 ANÁLISE SWOT

SWOT (*strengths, weaknesses, opportunities* e *threats*) é um tipo de análise usada para avaliar a posição competitiva de uma empresa e é uma maneira de desenvolver um plano estratégico. Esta análise avalia factores externos e internos, assim como o potencial atual e uma projeção de futuro [22].

É desenhado para facilitar e para demonstrar uma visão realista, baseada em factos e em informações recolhidas das forças e das fraquezas de uma organização, neste caso concreto, será a solução apresentada para resolver os problemas levantadas no 1º capítulo desta dissertação [32].

Cada análise SWOT inclui quatro categorias [58]:

**Forças (Strenghts)** - Forças descrevem em quais aspectos é que a solução se destaca e que o diferencia da competição. Só poderá ser considerada uma força se essa mesma se destacar dos demais senão acabará por ser um requisito necessário.

**Fraquezas (Weaknesses)** - As fraquezas são o que impedem um projeto de atingir o seu potencial máximo. Existem aspectos que precisam de ser melhorados e algumas práticas que poderão ser evitadas. Ao analisar as fraquezas é importante ser realista e honesto para que essas fraquezas possam ser ultrapassadas.

**Oportunidades (Opportunities)** - As oportunidades são fatores externos que podem dar uma vantagem competitiva. Explorar estas oportunidades poderão aumentar a competitividade e até ter uma liderança de mercado. Mesmo que sejam pequenas oportunidades, tudo o que seja útil para enriquecer a solução é bem-vindo.

**Ameaças (Threats)** - Ameaças incluí qualquer coisa que poderá afetar a solução apresentada. É importante antecipar ameaças e tomar uma ação, antes do projeto se tornar vítima destas mesmas ameaças.

O projeto desta dissertação foi posto à prova e submetido a uma análise SWOT com o objetivo de descobrir e identificar as suas forças, fraquezas, oportunidades e as suas ameaças.

| Forças / Fraquezas | |
|---|---|
| Melhor organização do horário de cada profissional de saúde | Layout das aplicações poderá ser melhorado |
| Facilidade de marcação de consultas/exames para os utentes | Bibliotecas usadas com algumas limitações em termos estéticos principalmente na aplicação mobile |
| Fácil acesso a resultados de exames/relatórios de consultas | Poucas funcionalidades |
| Apenas os caminhos dos ficheiros (exames) são guardados, o que torna tanto o servidor como a base de dados menos sobrecarregados | *Chatbot* difícil de alterar o estilo das mensagens, restrições na largura das mensagens renderizadas |
| Acessibilidade ao chatbot que poderá simplificar ações dos utentes | Organização de código, especialmente nas aplicações *frontend* |
| Autenticidade dos dados com a assinatura digital dos pedidos efetuados JSON Web Token | Dificuldade em criar uma conversa com o *chatbot* mais natural |
| Segurança dos dados sensíveis dos utilizadores da aplicação com a encriptação das palavra passes com bcrypt | ———————————- |

Tabela 10: Forças e Fraquezas (análise SWOT)

| Oportunidades / Ameaças | |
|---|---|
| Implementação de novas funcionalidades para tornar a aplicação mais rica | Apesar de haver uma certa segurança no sistema, haverá sempre fraquezas ao nível de segurança, como por exemplo SQL injection, que poderá destruir a base de dados |
| Aumentar as funcionalidades oferecidas pelo chatbot | Como existem duas aplicações frontend qualquer alteração profunda ao sistema irá provocar alguma complexidade |
| Possível colaboração com aplicações já existentes | Competição de aplicações concorrentes. |

Tabela 11: Oportunidades e Ameaças (análise SWOT)

## 6.3 CONCLUSÃO

Durante este capítulo foi abordada e realizada a prova de conceito, em que foi realizada uma pequena demonstração das principais funcionalidades das aplicações e foi realizada a análise *Strengths Weaknesses Opportunities Threats* (SWOT) da solução apresentada. Ao realizar esta análise foi possível descobrir as suas principais forças, as principais fraquezas, as oportunidades que poderão melhorar a qualidade da solução e as falhas que podem ameaçar a integridade do projeto.

Durante as análises realizadas foi possível explorar da melhor maneira possível o potencial da solução apresentada, mas também aquilo que poderá desvaforecer essa mesma solução.

Num olhar global a implementação deste projeto com alguns ajustes poderá ser útil para as questões levantadas na introdução desta dissertação na qual pretendia responder à satisfação do utente e da facilidade da relação com a sua unidade hospitalar e à melhor organização hospitalar.

No próximo e último capítulo servirá para retirar algumas conclusões sobre o trabalho desenvolvido e refletir sobre alterações futuras que possam garantir a evolução deste projeto.

# 7

# CONCLUSÃO E TRABALHO FUTURO

Neste capítulo irá ser feito o levantamento das principais conclusões do desenvolvimento desta dissertação e ainda uma reflexão sobre aquilo que poderá ser melhorado ou até introduzido para enriquecer o projeto desenvolvido.

## 7.1 SÍNTESE DO TRABALHO

Tendo em conta os problemas abordados na introdução desta dissertação, é possível constatar que de certa medida foi possível satisfazer os principais objetivos que foram propostos, como a fluidez da relação entre um paciente e a sua unidade hospitalar, melhor organização do horário de cada profissional de saúde e com a concretização destes dois objetivos, por consequente poderá aumentar a faturação de um hospital, por exemplo, ao melhorar a organização da marcação de consultas/exames evitar que os profissionais de saúde preparam consultas/exames que não se vão realizar o que poderá significar uma diminuição de custos. Com as aplicações desenvolvidas procurou-se de certa maneira organizar melhor o ambiente hospitalar.

### 7.1.1 *Identificação dos Objetivos*

O principal objetivo no desenvolvimento desta dissertação passava pela realização de uma solução que respondesse às dificuldades de um ambiente hospitalar. Para esse fim, foi utilizada uma metodologia de investigação denominada *Design Science Research*, na qual se levantaram duas questões, **Quais as funcionalidades ideias e mais procuradas para satisfazer o paciente na sua relação com a sua unidade hospitalar ?** e **Em que medida seria útil um *chatbot* na interação dos utentes com os serviços de saúde ?**. De modo a responder a estas questões foi construída a solução apresentada ao longo da dissertação.

| Objetivos |
|---|
| Facilitar o agendamento de consultas |
| Facilitar o agendamento de exames |
| Melhorar a organização hospitalar |
| Evitar falta de comparência dos utentes às consultas |
| Gerir membro do agregado familiar |

Tabela 12: Objetivos

### 7.1.2 *Identificação dos Problemas*

Na introdução desta dissertação foram abordados os problemas que surgiam na maioria das unidades hospitalares. De modo a que esses problemas tenham um maior destaque, foi feito um levantamento destes problemas, como podemos ver na tabela abaixo.

| Problemas |
|---|
| Falta de comparência a eventos médicos por parte dos utentes |
| Faturação do ambiente hospitalar |
| Organização hospitalar |
| Relação do utente com a sua unidade hospitalar |

Tabela 13: Problemas

### 7.1.3 *Identificação das Funcionalidades*

No mapeamento das funcionalidades, com o intuito de atingir os objetivos propostos e de responder aos problemas que foram levantados, foram enumeradas as funcionalidades que foram implementadas no desenvolvimento da solução. Na escolha das funcionalidades foi analisado o mercado nomeadamente as aplicações *App My São João*, *App PROXIMO*, *App CHUPorto*, *App My Luz*, *App Trofa Saúde* e *App MyMediclinic 24x7* e o respetivo *feedback*. Dessa forma foi possível perceber o que satisfazia os utentes e o que faltava, com essa análise, foram criadas as funcionalidades da solução apresentada.

| Funcionalidades |
|---|
| Marcação de consulta |
| Marcação de exame |
| Acesso a relatórios de consultas |
| Acesso a resultados de exame |
| Visualização dos exames e consultas futuras |
| Chatbot |
| Marcação de consulta/exame para membro do agregado familiar |

Tabela 14: Funcionalidades

### 7.1.4 *Conclusões finais*

Na abordagem metodológica, *Design Science Research*, foram abordadas duas questões centrais. Uma delas era **Quais as funcionalidades ideias e mais procuradas para satisfazer o paciente na sua relação com a sua unidade hospitalar**. De forma a obter uma resposta a esta questão, no estado de arte foi feito um levantamento das funcionalidades que se encontravam na maioria das aplicações hospitalares e procurou-se algum *feedback* em relação às mesmas, de modo a que fosse possível implementar as funcionalidades de acordo com os requisitos da maioria dos utentes. Após uma análise profunda do que o mercado oferecia, foi implementado, com sucesso, as funcionalidades seguintes:

- Marcação de exames/consultas;
- Visualização de relatórios médicos;
- Visualização de exames médicos;
- Visualização das marcações de exames/consultas;

Como resposta à pergunta levantada, podemos então afirmar que as funcionalidades indicadas acima eram as funcionalidades ideais e mais procuradas pelos utentes, quando procuram uma aplicação hospitalar.

Foi ainda levantada uma segunda questão, **Em que medida seria útil um *chatbot* na interação dos utentes com os serviços de saúde**. A resposta concreta a esta pergunta é difícil de ser respondida, uma vez que não foram realizados testes à aplicação de potenciais utilizadores da mesma, mas tendo em conta o *chatbot* desenvolvido podemos concluir que acabaria por ter proveito, já que é capaz de executar as funcionalidades que foram respondidas na primeira pergunta. Além dessas funcionalidades, foi introduzido no *chatbot* um método capaz de marcar consultas/exames para um membro do agregado familiar.

Com o levantamento das funcionalidades, dos objetivos e dos problemas nas secções anteriores foi possível realizar uma pequena matriz.

| Funcionalidades / Objetivos / Problemas | | |
|---|---|---|
| Marcação de consulta/exame | Organização hospitalar | Dificuldade em marcar eventos médicos |
| Acesso a relatórios de consultas e a resultados de exame | Facilitar a relação utente/hospital | Dificuldade de acesso a relatórios e resultado de exames |
| Visualização dos exames e consultas futuras | Aumentar faturação | Falta de comparência dos utentes |
| *Chatbot* | Facilidade do uso das funcionalidades | Eventuais dificuldades no uso da aplicação |
| Marcação de consulta/exame para membro do agregado familiar | Capacidade de marcar eventos para membros do agregado familiar sem ter de trocar de conta | *Feedback* dos utentes com falta desta funcionalidade |

Tabela 15: Trabalho desenvolvido

Na realização da matriz foram levantados também outros problemas associados que ainda não tinham sido referidos, como a eventual dificuldade do uso da aplicação, que foi respondido com a criação do *chatbot* e o *feedback* de muitos utentes com a falta de uma funcionalidade que permitisse marcar consultas/exames para membros do agregado familiar na qual foi criada uma funcionalidade para esse fim, com o uso do *chatbot*.

## 7.2 TRABALHO FUTURO

Neste subcapítulo é destinado a refletir sobre o trabalho futuro que poderá ser feito para melhorar o projeto desta dissertação.

Através da análise *SWOT* realizada no capítulo anterior, já houve um levantamento de umas pequenas alterações que pudessem ser úteis para o projeto.

Antes de abordar as alterações que poderiam ser feitas, era importante que este trabalho fosse alvo de testes de pessoas que fossem potenciais utilizadoras destas aplicações, desse modo, com o *feedback* obtido já era possível realizar alterações para satisfazer esses mesmos utilizadores.

Quanto a trabalho que já poderá ser implementado:

- Implementação de mais funcionalidades;
- Sistema de senhas, existente na aplicação PROXIMO, abordado no estado de arte;

- Melhoramento da interface gráfica, especialmente da aplicação mobile;
- Maior organização do código, principalmente nas aplicações *frontend*;
- Exploração de bibliotecas que sejam mais coerentes com a aplicação *mobile*;
- Aumentar a segurança, em especial no lado do servidor ao evitar escrever *querys* em *MySQL*;
- Alterar a renderização de componentes do *chatbot* e permitir a manipulação da largura e altura das mensagens mostradas;
- Tornar o *chatbot* mais inteligente, uma vez que o mesmo é bastante limitado em termos de troca de mensagens com o utilizador.

# BIBLIOGRAFIA


[1] Introduction to node.js. URL https://nodejs.dev/learn.

[2] React, a javascript library for building user interfaces, . URL https://reactjs.org/.

[3] React native learn once, write anywhere, . URL https://reactnative.dev/.

[4] React native chatbot expo, . URL https://github.com/jzarca01/react-native-chatbot-expo.

[5] React simple chatbot, . URL https://lucasbassetti.com.br/react-simple-chatbot/#/.

[6] Tutorialspoint web server. URL https://www.tutorialspoint.com/internet_technologies/web_servers.htm.

[7] Julius Azasoo and Kwame Boateng. A retrofit design science methodology for smart metering design in developing countries. 06 2015. doi: 10.1109/ICCSA.2015.23.

[8] Sunil Bangare, S Gupta, M Dalal, and A Inamdar. Using node.js to build high speed and scalable backend database server. *nternational Journal of Research in Advent Technology (E-ISSN: 2321-9637)*, 4:19, 04 2016.

[9] Jan vom Brocke, Alan Hevner, and Alexander Maedche. *Introduction to Design Science Research*, pages 1–13. 09 2020. ISBN 978-3-030-46780-7. doi: 10.1007/978-3-030-46781-4_1.

[10] Hospital da Luz. My luz. URL https://www.hospitaldaluz.pt/pt/para-clientes/my-luz.

[11] William Danielsson. React native application development. *Linköpings universitet, Swedia*, 10(4):10, 2016.

[12] John Deacon. Model-view-controller (mvc) architecture. *https://www.academia.edu/30077059/Model-View-Controller_MVC_Architecture*, 28, 2009.

[13] Android Developers. What is android? *https://www.academia.edu/2537177/What_is_Android*, 2011.

[14] Android Developers. Android platfrom architecture, 2020. URL https://developer.android.com/guide/platform.

[15] Levent Ertaul, Manpreet Kaur, and Venkata Arun Kumar R Gudise. Implementation and performance analysis of pbkdf2, bcrypt, scrypt algorithms. In *Proceedings of the International Conference on Wireless Networks (ICWN)*, page 66. The Steering Committee of The World Congress in Computer Science, Computer . . . , 2016.

[16] Expresso. Faltas a consultas médicas passam a ser pagas, 2013. URL https://expresso.pt/sociedade/faltas-a-consultas-medicas-passam-a-ser-pagas=f797366.

[17] G. Eysenbach. What is e-health? *J Med Internet Res*, 3(2):e20, Jun 2001. ISSN 1438-8871. doi: 10.2196/jmir.3.2.e20.

[18] Artemij Fedosejev. *React. js essentials*. Packt Publishing Ltd, 2015.

[19] David Flanagan. *JavaScript: o guia definitivo*. Bookman Editora, 2004.

[20] Przemyslaw Gilski and Jacek Stefanski. Android os: A review. *Tem Journal*, 4(1):116, 2015.

[21] Tor-Morten Grønli, Jarle Hansen, Gheorghita Ghinea, and Muhammad Younas. Mobile application platform heterogeneity: Android vs windows phone vs ios vs firefox os. *Proceedings - International Conference on Advanced Information Networking and Applications, AINA*, pages 635–641, 05 2014. doi: 10.1109/AINA.2014.78.

[22] Emet GURL. Swot analysis: A theoretical review. 2017.

[23] Henning Heitkötter, Sebastian Hanschke, and Tim A Majchrzak. Evaluating cross-platform development approaches for mobile applications. In *International Conference on Web Information Systems and Technologies*, pages 120–138. Springer, 2012.

[24] Alan R Hevner. A three cycle view of design science research. *Scandinavian journal of information systems*, 19(2):4, 2007.

[25] Parikshit Hooda. Model-view-controller(mvc) architecture for node applications, 2018. URL https://www.geeksforgeeks.org/model-view-controllermvc-architecture-for-node-applications/.

[26] jagroopofficial. Architecture of ios operating system, 2021. URL https://www.geeksforgeeks.org/architecture-of-ios-operating-system/.

[27] Karl Jähn and Eckhard Nagel. *e-Health*. Springer, 2004.

[28] Michael Jones, John Bradley, and Nat Sakimura. Json web token (jwt). Technical report, 2015.

[29] Hospital São João. Instale a app my são joão. URL https://portal-chsj.min-saude.pt/frontoffice/pages/16?news_id=1144.

[30] JWT. Introduction to json web tokens. URL https://jwt.io/introduction.

[31] M Kanoi and Yavatmal Jdiet. Internal structure of ios and building tools for ios apps. *International Journal Of Computer Science And Applications*, 6(2):220–225, 2013.

[32] Will Kenton. Swot analysis: How to with table and example. URL https://www.investopedia.com/terms/s/swot.asp.

[33] Peter Koffer. What is a native mobile app development? URL https://mdevelopers.com/blog/what-is-a-native-mobile-app-development-.

[34] Kotlin. What is cross-platform mobile development? URL https://kotlinlang.org/docs/cross-platform-mobile-development.html#cross-platform-mobile-development-definition-and-solutions.

[35] Paweł Kozielecki. Cross-platform vs native app development: What's the difference? URL https://www.netguru.com/blog/cross-platform-vs-native-app-development.

[36] Daniel Lacerda, Aline Dresch, Adriano Proença, and José Antonio Valle Antunes Júnior. Design science research: A research method to production engineering. *Gestão Produção*, 20:741–761, 12 2012. doi: 10.1590/S0104-530X2013005000014.

[37] William Malsam. What is proof of concept (poc)? definition, steps best practices. URL https://www.projectmanager.com/blog/proof-of-concept-definition.

[38] Anastasiya Marchuk. Native vs cross-platform development: Pros cons revealed. URL https://www.uptech.team/blog/native-vs-cross-platform-app-development.

[39] MDN. React getting started. URL https://developer.mozilla.org/pt-BR/docs/Learn/Tools_and_testing/Client-side_JavaScript_frameworks/React_getting_started.

[40] MDN. Mvc, 2021. URL https://developer.mozilla.org/en-US/docs/Glossary/MVC.

[41] MDN. Javascript, 2022. URL https://developer.mozilla.org/pt-BR/docs/Glossary/JavaScript.

[42] MDN. Métodos de requisição http, 2022. URL https://developer.mozilla.org/pt-BR/docs/Web/HTTP/Methods.

[43] MDN. O que é um servidor web (web server)?, 2022. URL https://developer.mozilla.org/pt-BR/docs/Learn/Common_questions/What_is_a_web_server.

[44] Mediclinic. Mymediclinic 24x7. URL https://www.mediclinic.ae/en/corporate/mymediclinic-24x7.html.

[45] Microsoft. Javascript basics, 2021. URL https://docs.microsoft.com/pt-pt/learn/modules/build-simple-website/5-javascript-basics.

[46] Oladotun Okediran, Oladiran Arulogun, Ganiyu Adesina, and Oyeleye Akinwale. Mobile operating systems and application development platforms: A survey. *International Journal of Advanced Networking and Applications*, 6:2195–2201, 08 2014.

[47] Ken Peffers, Tuure Tuunanen, Marcus A. Rothenberger, and Samir Chatterjee. A design science research methodology for information systems research. *Journal of Management Information Systems*, 24(3):45–77, 2007. doi: 10.2753/MIS0742-1222240302.

[48] Caio Ribeiro Pereira. *Aplicações web real-time com Node. js.* Editora Casa do Código, 2014.

[49] Mariano Pimentel, Denise Filippo, and Thiago Marcondes Santos. Design science research: pesquisa científica atrelada ao design de artefatos. *RE@ D-Revista de Educação a Distância e eLearning*, 3(1):37–61, 2020.

[50] Google Play. Proximo®. URL https://play.google.com/store/apps/details?id=proside.gamobile&hl=pt_PT&gl=US.

[51] CHU Porto. Chuporto app. URL https://www.chporto.pt/v0L/app.

[52] Mary K. Pratt. proof of concept (poc). URL https://www.techtarget.com/searchcio/definition/proof-of-concept-POC.

[53] Niels Provos and David Mazieres. Bcrypt algorithm. In *USENIX*, 1999.

[54] Jacob Schmitt. Native vs cross-platform mobile app development. URL https://circleci.com/blog/native-vs-cross-platform-mobile-dev/.

[55] Nikita Shevtsiv and Andrii Striuk. Cross platform development vs native development. CEUR Workshop Proceedings, 2021.

[56] Jignesh Solanki. Native vs. cross platform: Decoding best choice for your apps. URL https://www.simform.com/blog/native-vs-cross-platform-development/.

[57] Paul A Soukup. Smartphones. 2015.

[58] Mind Tools Content Team. Swot analysis understanding your business, informing your strategy. URL https://www.mindtools.com/pages/article/newTMC_05.htm.

[59] TrofaSaúde. App trofa saúde. URL https://www.trofasaude.pt/app-trofa-saude/.

[60] Wikipedia. bcrypt. URL https://en.wikipedia.org/wiki/Bcrypt.