

# **Universidade da Beira Interior**

## **Departamento de Informática**



**Departamento de  
Informática**

**Nº 82 - 2019: *Plataforma para Gestão de  
Medicamentos Críticos***

Elaborado por:

**Rita Pessoa Correia**

Orientador:

**Professor Doutor Mário M. Freire**

15 de Julho de 2019



# Agradecimentos

A conclusão deste projeto, bem como da maior parte da minha vida académica não seria possível, primeiramente, sem a ajuda e o constante apoio da minha família, nomeadamente dos meus pais. Há tanto e tão pouco para vos dizer, pois aquilo que realmente importa não são as palavras, mas sim as atitudes! Ainda assim, OBRIGADA! Obrigada por estarem sempre comigo, por não me deixarem desistir, por me motivarem a ser todos os dias um pouco melhor. Obrigada por me ensinarem a ser mais exigente comigo própria, a ser determinada e a ter objetivos de vida (ainda mais) reais. Conseguí! Finalmente consegui concretizar aquilo que sempre quis, e por em prática grande parte dos conselhos que me deram ao longo da vida (não só académica). Conseguí tornar-me numa versão melhorada de mim própria e deixar-vos orgulhosos. Não há sentimento melhor. Valeu a pena, como sempre me disseram que valeria! Com vocês aprendi a valorizar aquilo que merece ser valorizado, inclusive o trabalho e o esforço eminentes por de trás de um grande projeto, seja ele qual for. Um momento não é tudo, mas vocês são tudo, em todos os momentos! Vá onde for, estarei sempre presente. Amo-vos muito.

Em segundo lugar, quero também prestar um agradecimento especial a todos os professores do Departamento de Informática, nomeadamente ao meu orientador de projeto Prof. Dr. Mário Freire, e ao presidente do Departamento de Informática Pedro R. M. Inácio, por me darem oportunidades de crescer, de adquirir conhecimentos e de os por em prática. É importante querer e procurar aprender mais, mas com estes magníficos professores, isso torna-se ainda mais fácil!

Um outro agradecimento que não poderia deixar de fazer, é ao meu namorado, Pedro Moreira. O teu apoio foi constante, tanto neste projeto, como em grande parte da minha vida académica, e sem ti não teria certamente sido igual. Ajudaste-me a levantar sempre que ia tropeçando neste percurso, e mostraste-me que não dependo de ninguém a não ser de mim própria, para concretizar tudo aquilo que ambiciono. Obrigada do fundo do meu coração.

Para uma melhor realização deste projeto, a ajuda dos meus afilhados, Rafael Pina e Carlos Robalinho foi crucial! A vocês dedico grande parte desta vitória!. Ajudaram-me a encontrar um rumo, foram o meu ponto de partida, e eu estarei sempre disponível para, futuramente, ser também o vosso.

Outra pessoa que foi sem dúvida uma peça essencial no desenrolar deste projeto (e da minha vida académica em geral), foi o meu grande amigo Rúben Gonçalves, que é o engenheiro informático mais competente e profissional que conheço. Sem a tua ajuda e apoio constante teria sido bastante mais complicado, obrigada!

Por último, mas não menos importante, quero fazer um grande agradecimento a todos os meus amigos que me acompanharam nesta caminhada, nomeadamente à Mariana Dantas e ao Tiago Sena, que observaram de perto todas as etapas da realização deste projeto, e tudo o que cada uma implicou. Obrigado a vocês que não me deixaram desistir, e que me motivaram constantemente a ser ainda melhor.

# Conteúdo

<b>Conteúdo</b>	<b>iii</b>
<b>Lista de Figuras</b>	<b>vii</b>
<b>Lista de Tabelas</b>	<b>ix</b>
<b>1 Introdução</b>	<b>1</b>
1.1 Enquadramento . . . . .	1
1.2 Motivação . . . . .	2
1.3 Objetivos . . . . .	2
1.4 Organização do Documento . . . . .	2
<b>2 Estado da Arte</b>	<b>5</b>
2.1 Introdução . . . . .	5
2.2 <i>Web</i> . . . . .	6
2.3 Servidor . . . . .	7
2.3.1 Servidor <i>Web</i> . . . . .	8
2.4 Arquitetura Cliente-Servidor . . . . .	9
2.5 Base de Dados . . . . .	10
2.5.1 Modelo Relacional . . . . .	10
2.6 <i>Frameworks</i> . . . . .	11
2.6.1 Vantagens do uso de <i>Frameworks</i> . . . . .	11
2.6.2 Desvantagens do uso de <i>Frameworks</i> . . . . .	12
2.6.3 Principais <i>frameworks</i> . . . . .	13
2.6.4 <i>Framework</i> vs Biblioteca de código . . . . .	14
2.7 Conclusões . . . . .	15
<b>3 Tecnologias e Ferramentas Utilizadas</b>	<b>17</b>
3.1 Introdução . . . . .	17
3.2 Servidor . . . . .	18
3.2.1 <i>Apache</i> . . . . .	19

3.2.2	MySQL . . . . .	19
3.3	Linguagem utilizadas . . . . .	20
3.3.1	SQL . . . . .	20
3.3.2	<i>PHP: Hypertext Preprocessor (PHP)</i> . . . . .	21
3.3.3	Hypertext Markup Language (HTML) e Cascading Style Sheets (CSS) . . . . .	22
3.4	<i>Layout</i> . . . . .	22
3.5	Integrated Development Environment (IDE) . . . . .	23
3.6	<i>Framework</i> . . . . .	25
3.7	Conclusões . . . . .	27
<b>4</b>	<b>Engenharia de Software</b>	<b>29</b>
4.1	Introdução . . . . .	29
4.2	Requisitos . . . . .	30
4.2.1	Requisitos funcionais . . . . .	30
4.2.2	Requisitos não funcionais . . . . .	50
4.3	Casos de uso . . . . .	51
4.4	Padrão de desenho arquitectural . . . . .	53
4.4.1	Padrão Model-View-Controller (MVC) . . . . .	53
4.5	Conclusões . . . . .	54
<b>5</b>	<b>Implementação e Testes</b>	<b>55</b>
5.1	Introdução . . . . .	55
5.2	Udemy . . . . .	57
5.3	Base de Dados . . . . .	58
5.3.1	<i>Diagrama Entidade-Associação (DEA)</i> . . . . .	58
5.4	Manual do utilizador . . . . .	60
5.4.1	Login . . . . .	60
5.4.2	<i>Dashboard</i> . . . . .	60
5.4.3	Gestão de contas . . . . .	64
5.4.4	Medicamentos . . . . .	65
5.4.5	Pedidos . . . . .	69
5.4.6	Histórico de medicamentos . . . . .	72
5.4.7	Lista de contactos . . . . .	73
5.4.8	Definições . . . . .	74
5.4.9	Logout . . . . .	75
5.5	Pedidos . . . . .	76
5.5.1	Fazer um pedido . . . . .	76
5.5.2	Pedidos enviados . . . . .	78
5.5.3	Pedidos recebidos . . . . .	81
5.6	Testes . . . . .	85

---

5.6.1	PHPUnit . . . . .	85
5.6.2	<i>Chrome Dev Tools</i> . . . . .	86
5.6.3	Testes manuais . . . . .	88
5.7	Conclusões . . . . .	93
<b>6</b>	<b>Conclusões e Trabalho Futuro</b>	<b>95</b>
6.1	Conclusões Principais . . . . .	95
6.2	Trabalho Futuro . . . . .	96
	<b>Bibliografia</b>	<b>97</b>



# **Lista de Figuras**

2.1	Funcionamento de um servidor <i>Web</i> . . . . .	8
2.2	Esquema arquitetura Cliente-Servidor. . . . .	9
2.3	Tabela de medicamentos referente à Base de Dados (BD) da plataforma. . . . .	10
2.4	Criação de uma <i>Framework</i> . . . . .	11
2.5	Bibliotecas e <i>Frameworks</i> . . . . .	15
3.1	Ambiente Laragon. . . . .	19
3.2	Logótipo MySQL. . . . .	20
3.3	Assistência inteligente no PhpStorm. . . . .	23
3.4	Navegação inteligente no PhpStorm. . . . .	24
3.5	Refatoração rápida e segura no PhpStorm. . . . .	24
3.6	Depuração e testes no PhpStorm. . . . .	25
3.7	Logótipo do Laravel. . . . .	27
4.1	Diagrama de casos de uso do farmacêutico. . . . .	51
4.2	Diagrama de casos de uso do administrador. . . . .	52
4.3	Padrão de desenho arquitectural da plataforma. . . . .	54
5.1	DEA implementado na base de dados. . . . .	59
5.2	Páginas de <i>login</i> . . . . .	61
5.3	Página principal do farmacêutico. . . . .	61
5.4	Página principal do administrador. . . . .	62
5.5	Alterar esquema de cores. . . . .	62
5.6	Alterar cor da barra de navegação e menu lateral. . . . .	63
5.7	Registo de um farmacêutico. . . . .	64
5.8	Lista de contas. . . . .	64
5.9	Alteração de dados de uma conta. . . . .	65
5.10	Inserir um novo medicamento. . . . .	66
5.11	Inserir um medicamento existente. . . . .	66
5.12	Inserir quantidade do medicamento. . . . .	67

5.13	Lista Local de Medicamentos.	67
5.14	Editar quantidade de um medicamento.	68
5.15	Lista Global de medicamentos.	68
5.16	Fazer um pedido - secção 1.	69
5.17	Fazer um pedido - secção 2.	70
5.18	Lista de pedidos enviados.	71
5.19	Detalhes de um pedido.	71
5.20	Lista de pedidos recebidos.	72
5.21	Página de histórico de medicamentos.	73
5.22	Detalhes do histórico de um medicamento específico.	73
5.23	Lista de contactos.	74
5.24	Detalhes de contacto de um centro hospitalar específico.	74
5.25	Definições.	75
5.26	Opção de <i>Logout</i> .	75
5.27	<i>Audit Tests</i> na página principal.	87
5.28	<i>Audit Tests</i> na página de <i>login</i> .	88
5.29	Campos em falta na autenticação.	89
5.30	Credenciais incorretas ou má formatação.	89
5.31	Endereço de <i>email</i> incompleto.	90
5.32	Credenciais incorretas ou má formatação.	90
5.33	Obrigatoriedade de preenchimento de todos os campos no registo de uma conta.	91
5.34	Obrigatoriedade de preenchimento de todos os campos na inserção de medicamentos.	91
5.35	Quantidade do medicamento inválida.	92
5.36	ID de medicamento inválido.	92
5.37	Quantidade inválida.	93

# **Lista de Tabelas**

4.1	Organização de requisitos funcionais da plataforma. . . . .	32
4.2	Requisitos funcionais de <i>login</i> . . . . .	33
4.3	Requisitos funcionais do formulário de <i>login</i> . . . . .	33
4.4	Requisitos funcionais do registo. . . . .	34
4.5	Requisitos funcionais da reposição de palavra-passe. . . . .	35
4.6	Requisitos funcionais da página principal. . . . .	36
4.7	Requisitos funcionais do menu lateral da página principal do farmacêutico. . . . .	37
4.8	Requisitos funcionais da página principal do administrador. . . . .	38
4.9	Requisitos funcionais do menu lateral da página principal do administrador. . . . .	38
4.10	Requisitos funcionais da página de listar contas. . . . .	39
4.11	Requisitos funcionais da página de inserir um medicamento. . . . .	40
4.12	Requisitos funcionais do formulário da página de inserir um medicamento. . . . .	41
4.13	Requisitos funcionais da página de listar os medicamentos locais. .	41
4.14	Requisitos funcionais da página de listar os medicamentos globais. .	42
4.15	Requisitos funcionais da página de inserir um novo pedido. . . . .	42
4.16	Requisitos funcionais do formulário da página de inserir um novo pedido. . . . .	43
4.17	Requisitos funcionais da página de listar os pedidos enviados. . .	44
4.18	Requisitos funcionais de alguns elementos da tabela de pedidos enviados. . . . .	44
4.19	Requisitos funcionais da página de listar os pedidos recebidos. . .	45
4.20	Requisitos funcionais de alguns elementos da tabela de pedidos recebidos. . . . .	45
4.21	Requisitos funcionais da página de histórico de medicamentos. . .	46
4.22	Requisitos funcionais da página de histórico de um medicamento específico. . . . .	47
4.23	Requisitos funcionais da página de contactos. . . . .	48

4.24 Requisitos funcionais da página de contactos de um centro hospitalar específico. . . . .	48
4.25 Requisitos funcionais das definições de conta. . . . .	49
4.26 Requisitos Não Funcionais da plataforma. . . . .	50

# Listas de Excertos de Código

3.1	Exemplo de código em PHP. . . . .	21
3.2	Exemplo de uma <i>master page</i> utilizando o <i>blade</i> . . . . .	26
3.3	Exemplo de uma página que herda o <i>layout</i> da <i>master page</i> com o <i>blade</i> . . . . .	26
5.1	Função <code>fazerPedido()</code> . . . . .	76
5.2	Função <code>store(Request \$request)</code> . . . . .	77
5.3	Função <code>pedidosEnviados()</code> . . . . .	78
5.4	Função <code>pedidosEnviadosID(\$id)</code> . . . . .	79
5.5	Função <code>confirmenvio(\$id)</code> . . . . .	80
5.6	Função <code>rejectenvio(\$id)</code> . . . . .	80
5.7	Função <code>pedidosRecebidos()</code> . . . . .	81
5.8	Função <code>pedidosRecebidosID(\$id)</code> . . . . .	82
5.9	Função <code>confirm(\$id)</code> . . . . .	83
5.10	Testes unitários. . . . .	86



# Acrónimos

<b>ANSI</b>	American National Standards Institute
<b>BD</b>	Base de Dados
<b>CERN</b>	<i>Organização Europeia para a Pesquisa Nuclear</i>
<b>CSS</b>	Cascading Style Sheets
<b>DEA</b>	<i>Diagrama Entidade-Associação</i>
<b>DOM</b>	Modelo de Objeto de Documento
<b>ES6</b>	ECMAScript 6
<b>FCS</b>	Faculdade de Ciências da Saúde
<b>FK</b>	<i>Foreign Key</i>
<b>HTML</b>	Hypertext Markup Language
<b>HTTP</b>	Hypertext Transfer Protocol
<b>HTTPS</b>	Hypertext Transfer Protocol Secure
<b>IBM</b>	International Business Machines
<b>IDE</b>	Integrated Development Environment
<b>ISO</b>	International Organization for Standardization
<b>MVC</b>	Model-View-Controller
<b>NCSA</b>	National Center for Supercomputing Applications
<b>ORM</b>	<i>Object-relational mapping</i>
<b>PHP</b>	<i>PHP: Hypertext Preprocessor</i>

**Sass** Syntactically Awesome Style Sheets

**SEO** *Search Engine Optimization*

**SGBD** Sistema de Gestão de Base de Dados

**SQL** Structured Query Language

**SSL** Secure Sockets Layer

**TI** Tecnologias de Informação

**UBI** Universidade da Beira interior

**UC** Unidade Curricular

**URL** Uniform Resource Locator

**W3C** World Wide Web Consortium

**WWW** *World Wide Web*

# Glossário

**backend** Sistema responsável pelas regras de negócios, *web services* e APIs de uma aplicação. 9

**browsers** É um programa de computador que possibilita aos utilizadores uma interacção com documentos virtuais da Internet, também conhecidos como páginas Web. 22

**cookies** São pequenas etiquetas de *software* que são armazenadas no computador através do navegador, retendo apenas informação relacionada com as preferências do utilizador, não incluindo, como tal, dados pessoais. Estas permitem uma navegação mais rápida e eficiente, eliminando a necessidade do utilizador introduzir repetidamente as mesmas informações. 87

**frontend** Interface de interação direta com o utilizador. 9

**FTP** É um protocolo de transferência de arquivos. É responsável pela transferência de arquivos do servidor para um computador e vice-versa. 7

**hardware** É a parte física do computador, ou seja, o conjunto de peças e equipamentos eletrónicos (como é o exemplo da RAM, do CPU, da *motherboard*,etc). 8

**HTTPS** É o protocolo que um navegador utiliza, para visualizar páginas *web* de forma segura e protegida. 19

**Internet** É um sistema global de redes de computadores interligadas que utilizam um conjunto próprio de protocolos, como é o exemplo do protocolo TCP/IP, com o propósito de servir progressivamente utilizadores no mundo inteiro. 7

**IP** É um rótulo numérico atribuído a cada dispositivo (computador, impressora, smartphone etc.) conectado a uma rede de computadores e que utiliza o Protocolo de Internet para comunicação. 8

**JavaScript** É uma linguagem de programação baseada na linguagem de programação ECMAScript. Actualmente é a linguagem de programação mais utilizada em "*Client-Side*" nos *browsers*. 21

**opensource** É um modelo de desenvolvimento que promove o licenciamento livre de um produto, e a redistribuição universal desse, com a possibilidade de livre consulta, examinação ou modificação do mesmo, sem a necessidade de pagar uma licença comercial, promovendo assim um modelo colaborativo de produção intelectual. 21

**PHP** É uma linguagem de *script*, *opensource* e de uso geral, muito utilizada, e especialmente adequada para o desenvolvimento *Web*, sendo que pode também ser embutida dentro do HTML. 17

**queries** São pedidos de uma informação ou de um dado a uma base de dados. Estes pedidos podem também ser entendidos como uma consulta, uma solicitação ou, ainda, uma requisição. 27

**shell** É uma interface de linha de comando, utilizada para aceder aos serviços de um sistema operativo. 86

**software** É aaaa parte lógica do computador. Esta pode executar processos de manipulação, instruções de execução, redirecionamento e execução das atividades lógicas das máquinas. 7

**span** Refere-se a uma *tag* HTML que funciona como um *container inline* para elementos e conteúdo. É usado para agrupar elementos para propósitos de estilo. 44

**SQL** É uma linguagem padrão de base de dados, utilizada para criar, manter e recuperar a base de dados relacional. 20

**tags** São estruturas de linguagem de marcação que contêm instruções, e possuem uma marca de início e outra de fim para que o navegador possa renderizar uma página *Web*. 22

**TCP/IP** representa um conjunto de protocolos que permitem que diversos equipamentos que constituem uma rede possam comunicar entre si. 6

**webpages** São documentos geralmente no formato HTML, que podem ser mostrados num Web browser, podendo estes também serem denominados de páginas. 17

# Capítulo 1

## Introdução

*“Every once in a while, a new technology, an old problem, and a big idea turn into an innovation.”*

— Dean Kamen

### 1.1 Enquadramento

Este trabalho foi desenvolvido no âmbito da Unidade Curricular (UC) de Projeto, de 3º ano, e é dirigido aos alunos dos cursos de Informática Web e Engenharia Informática da Universidade da Beira interior (UBI).

O presente projeto consiste no desenvolvimento de um protótipo de uma aplicação *web* que permita gerir, analisar informação sobre medicamentos críticos, e efetuar pedidos de trocas destes entre os vários centros hospitalares do país, desde que estes estejam registados na plataforma. Este projeto foi realizado em parceria com a Faculdade de Ciências da Saúde (FCS) da UBI, no laboratório NMCG da mesma, situado no Departamento de Informática e Matemática.

O acesso à plataforma denominada de MedGest é restrito a utilizadores registados e autorizados pelo administrador do centro hospitalar correspondente. Assim, existem dois tipos de utilizadores: os administradores (1 por centro hospitalar) e os farmacêuticos. Cada administrador tem privilégios extra relativamente aos farmacêuticos, nomeadamente a gestão de contas e inserção de um novo medicamento no centro hospitalar em questão. Ambos os utilizadores podem listar todos os medicamentos, fazer um pedido, consultar os pedidos enviados e recebidos, consultar o histórico de movimentos e ver uma lista dos contactos de todos os centros hospitalares. Podem também alterar as suas definições de conta.

## **1.2 Motivação**

Sendo hoje em dia as Tecnologias de Informação (TI) uma das áreas de estudo mais abrangentes, e com mais perspectivas de crescimento, é cada vez mais importante procurar saber cada vez mais. Toda a informação é pouca num mundo que evolui a um ritmo cada vez mais acelerado, nomeadamente através deste tipo de projetos individuais, que envolvem muita pesquisa, muitos testes, e que promovem bastante uma auto-aprendizagem, para que mais tarde seja possível alcançar de forma mais rápida, exigente e eficaz certos objetivos profissionais, multiplicando os resultados até à data existentes.

Desta forma, todas as áreas profissionais estão a evoluir no sentido de uma maior e constante necessidade do uso das novas tecnologias, sendo estas necessárias em praticamente todas as áreas, como é o caso da área da Saúde, uma das áreas mais importantes para nós, Seres Humanos. Cada vez mais, este setor necessita de muita informação a circular de forma rápida, eficaz e segura, onde as aplicações *Web* são extremamente necessárias. Desta forma, a plataforma desenvolvida procura atender à necessidade existente de uma troca de medicamentos críticos entre hospitais, de forma segura e inovadora.

## **1.3 Objetivos**

Os objetivos estabelecidos, após a contextualização com as tecnologias envolventes e a explicação da ideia base da plataforma desenvolvida, são os seguintes:

- Contextualização com as tecnologias e linguagens utilizadas e consequente preparação do ambiente de desenvolvimento;
- Levantamento de requisitos, com colaboração da FCS da UBI e do Centro Hospitalar da Cova da Beira;
- Construção do modelo da base de dados;
- Implementação da aplicação *Web*;
- Aprimoramento e testes;

## **1.4 Organização do Documento**

De modo a refletir o trabalho que foi feito, este documento encontra-se estruturado da seguinte forma:

1. O primeiro capítulo – **Introdução** – apresenta o projeto, a motivação para a sua escolha, o enquadramento para o mesmo, os seus objetivos e a respetiva organização do documento.
2. O segundo capítulo – **Estado da Arte** – analisa diversas tecnologias e conceitos existentes na área das aplicações *Web*, e compara-as, de modo a entender a escolha final que foi feita para utilizar no projeto.
3. O terceiro capítulo – **Tecnologias Utilizadas** – descreve e pormenoriza os conceitos mais importantes no âmbito do desenvolvimento deste projeto, tal como as tecnologias que foram utilizadas durante do desenvolvimento da aplicação *Web*.
4. O quarto capítulo – **Engenharia de Software** – apresenta os requisitos, diagramas e modelos utilizados para a construção da plataforma.
5. O quinto capítulo – **Implementação e Testes** – descreve a forma de implementação e como foram efetuados alguns testes da aplicação, tanto a nível do servidor, como do cliente (incluindo algumas exceções geradas).
6. O sexto capítulo – **Conclusões e Trabalho Futuro** – descreve as conclusões finais sobre o projeto e o trabalho que poderá ser realizado futuramente para complementar, melhorar e estender todo o trabalho até à data existente.



# Capítulo 2

## Estado da Arte

### 2.1 Introdução

Este capítulo irá abordar os principais elementos para a construção de uma aplicação *Web* e as tecnologias existentes nesta área, bem como uma breve comparação entre elas.

Um dos papéis principais das aplicações *Web* passa por interagir com vários utilizadores, simultaneamente. Para garantir o bom funcionamento deste serviço, é necessário um servidor *Web* que forneça e partilhe todas as funcionalidades e dados aos utilizadores da plataforma. Para este efeito de partilha de dados, é necessária a existência de uma base de dados, que permite organizar, relacionar e armazenar todos os dados da aplicação, como por exemplo informação sobre os utilizadores, centros hospitalares, medicamentos, pedidos, etc.

Em Portugal não existe ainda nenhuma plataforma semelhante, o que tornou ainda um maior desafio o desenvolvimento deste projeto[6].

Este capítulo encontra-se organizado pelas seguintes secções:

- secção 2.2 - ***Web*** - define o conceito em questão e refere algumas funcionalidades fundamentais;
- secção 2.3 - **Servidor** - descreve o conceito de servidor, alguns tipos de servidores e a sua importância para qualquer tipo de aplicação;
- secção 2.3.1 - **Servidor Web** - especifica o tipo de servidor em questão e a sua importância;
- secção 2.4 - **Arquitetura Cliente-Servidor** - apresenta a arquitetura Cliente-Servidor e a forma como é feita a comunicação entre ambas as partes;

- secção 2.5 - **Base de Dados** - caracteriza o conceito de base de dados e revela a sua importância no contexto aplicacional;
- secção 2.5.1 - **Modelo Relacional** - descreve o modelo relacional e a sua usabilidade;
- secção 2.6 - **Frameworks** - descreve o conceito de *framework* e a sua importância;
- secção 2.6.1 - **Vantagens do uso de Frameworks** - refere algumas vantagens na utilização de *frameworks*;
- secção 2.6.2 - **Desvantagens do uso de Frameworks** - refere algumas desvantagens na utilização de *frameworks*;
- secção 2.6.3 - **Principais frameworks** - refere as *frameworks* mais utilizadas no desenvolvimento Web, tanto de *backend* como de *frontend*;
- secção 2.6.4 - **Framework vs Biblioteca de código** - reflete nas principais diferenças entre uma *framework* e uma biblioteca de código, descrevendo alguns exemplos;
- secção 2.7 - **Conclusões** - refere as principais conclusões do capítulo.

## 2.2 Web

A *World Wide Web* (WWW) ou W3 é um serviço de informação baseada na partilha de documentos, onde a informação está armazenada no formato de hipertexto. A ideia remonta o ano de 1980 e foi desenvolvida por Tim Berners-Lee, um físico e investigador britânico que na altura trabalhava na *Organização Europeia para a Pesquisa Nuclear* (CERN) [1].

A Web acentua em três funcionalidades principais:

- **Hypertext Transfer Protocol (HTTP)**, controla a transferência de dados entre um servidor Web e o cliente, usando para tal os protocolos TCP/IP;
- **Uniform Resource Locator (URL)**, para cada documento ou recurso existe um endereço único;

- **Hypertext Markup Language (HTML)**, estrutura padrão utilizada pelos documentos na *Web* [20].

Após 13 anos, em 30 de Abril de 1993, a *Web* foi aberta ao público, sendo hoje em dia um espaço que pode ser acedido de forma global, através de dispositivos conectados à Internet.

Atualmente, podemos dizer que vivemos na Web 3.0 [2], que é a fase em que os sistemas de procura estão cada vez mais inteligentes e o conteúdo disponível na teia mundial se torna cada vez mais personalizado.

## 2.3 Servidor

O termo "servidor" refere-se a um *software* ou computador, com um sistema centralizado e especializado, que fornece vários serviços a uma rede de computadores (cliente). Este pode conter várias funcionalidades de diferentes tipos e tamanhos, para facilitar também diferentes trabalhos e usos.

Um servidor pode ser criado numa rede local, sendo assim denominado de *localhost*. Neste caso, o próprio computador toma a função de servidor. Existem várias plataformas de ambiente que podem ser utilizadas neste sentido, entre elas o XAMPP, o WAMP, o MAMP, o AMPPS, o Laragon, o Cherokee, o EasyPHP, etc. À sinergia criada entre o servidor e os clientes, chamamos de rede ou arquitetura cliente-servidor, que veremos mais em detalhe na próxima secção.

Os clientes podem-se conectar a um servidor por meio de uma rede local (LAN), ou de uma rede de longa distância (WAN), como a Internet [13].

Existem diversos servidores, tais como:

- Servidor FTP: é responsável pela transferência de arquivos do servidor para um computador e vice-versa;
- Servidor Proxy: é responsável por uma conexão entre um cliente e um servidor externo para atender à solicitação de conexão, aprimoramento de desempenho e acessibilidade;
- Servidor Web: é responsável por hospedar os arquivos de um *site* e servi-lo através de um navegador *Web*;
- Servidor de Base de Dados: é o sistema usado para o armazenamento de bases de dados numa arquitetura Cliente/Servidor, que procura executar tarefas específicas por quem faz uso deste;

- Servidor DNS: oferece uma solução para redes baseadas em TCP/IP, ou seja, possibilita que os clientes utilizem nomes, em vez de endereços IP numéricos, para identificar *hosts* remotos.

### 2.3.1 Servidor Web

Nesta secção referimos diversos tipos de servidores, entre os quais os Servidores *Web*, que iremos aprofundar um pouco mais nesta sub-secção.

Este tipo de servidores é responsável por atender a todas as solicitações feitas para um determinado endereço na Internet. De forma geral, pode-se dizer que um servidor *Web* é um computador que hospeda um ou mais *sites*/aplicações na Internet. No entanto, o mesmo termo pode-se referir tanto ao equipamento físico (*hardware*) como ao programa (*software*) contido neste equipamento. É também possível o termo servidor *Web* se referir a ambos os casos, simultaneamente.

Um servidor *Web*, como equipamento, refere-se ao lugar onde estão armazenados todos os arquivos de um *site*. Este possui componentes internos semelhantes a um computador pessoal comum, como por exemplo um disco, memória RAM, *motherboard*, etc, sendo no entanto a sua arquitetura otimizada para a tarefa de servidor.

Os servidores deste tipo podem ser encontrados em duas formas diferentes: os de torre e os de rack.

Na vertente de servidor *web* como *software*, este ao aceder a um *website*, disputa a comunicação do navegador com o servidor, solicitando e recebendo os dados da página em questão. O servidor físico possui programas específicos para responder ao tipo de solicitação realizada, sendo no caso de um *website* esta solicitação feita através do protocolo HTTP [14].

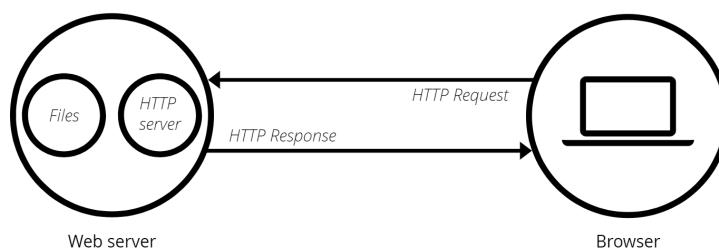


Figura 2.1: Funcionamento de um servidor *Web*.

De seguida, iremos abordar melhor a arquitetura deste modelo, e a forma como este é utilizado no funcionamento da aplicação.

## 2.4 Arquitetura Cliente-Servidor

A arquitetura Cliente-Servidor é uma estrutura de aplicação distribuída, sendo o processamento da informação dividido em módulos ou processos distintos. Um processo é responsável pela manutenção da informação (servidores) e outros responsáveis pela obtenção dos dados (clientes). Os clientes enviam pedidos para o servidor e este, por sua vez, processa-os e envia os resultados dos mesmos [5]. Mais em pormenor, os processos cliente:

- Realizam o envio das mensagens, fazendo algum pedido ao servidor;
- É a parte que interage com o usuário, possui a interface que o utilizador usa para requisitar as tarefas ao servidor, sendo esta chamada de *frontend* da aplicação;
- Gerem as atividades dos utilizadores e realizam as validações dos dados informados por estes.

Enquanto que os processos servidor:

- Respondem a uma mensagem solicitando a realização de alguma tarefa por parte do cliente. Este é chamado de *backend*;
- Podem oferecer serviços a muitos clientes, realizando pesquisas, filtragens e atualizações em bases de dados;
- Os serviços podem ser realizados diretamente pelo processo servidor ou através de processos escravos criados por este para atender cada pedido do cliente, o que libera o processo mestre do servidor para receber outras solicitações [26].

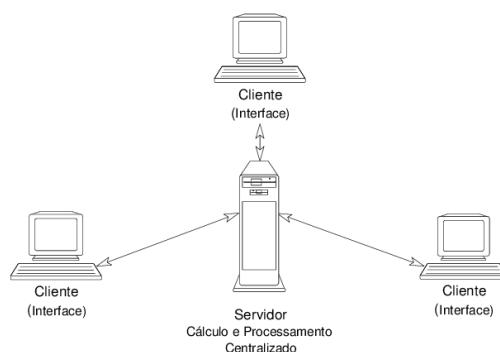


Figura 2.2: Esquema arquitetura Cliente-Servidor.

## 2.5 Base de Dados

Uma base de dados é um local onde pode ser armazenada informação. Essa informação pode ser consultada, alterada, apagada (na totalidade ou parcialmente), através de um Sistema de Gestão de Base de Dados (SGBD). A principal finalidade de um SGBD é assim gerir o acesso e a correta manutenção dos dados armazenados numa ou mais bases de dados. O acesso aos dados é disponibilizado por meio de uma interface que permite a comunicação com a aplicação desenvolvida.

Numa Base de Dados (BD) a informação encontra-se estruturada, facilitando assim a utilidade e longevidade da informação durante um maior período de tempo [24].

Uma BD pode ter diversos modelos que definem como a informação está organizada internamente. Os mais comuns são o modelo de dados e relacional, que definem os atributos de uma tabela e a relação entre as várias tabelas.

### 2.5.1 Modelo Relacional

Este modelo foi inventado por Edgar Frank Codd, como uma nova maneira de representação de dados.

Neste seu trabalho Codd mostrou que uma visão relacional dos dados permite a sua descrição numa maneira natural, sem que sejam necessárias estruturas adicionais para sua representação, promovendo uma maior independência dos dados em relação aos programas. Como complemento, este apresentou também bases para tratar problemas como redundância e consistência dos dados[23].

Neste modelo, a cada tabela está associada uma entidade, e para cada atributo de uma entidade está associada uma coluna na tabela. Cada variante de uma entidade é colocada nas linhas da tabela correspondente, com o valor de cada atributo na coluna respetiva. A figura seguinte representa a tabela Medicamentos da BD da plataforma MedGest, onde as colunas representam os atributos de cada medicamento, e as linhas informações sobre os vários medicamentos até à data existentes.

	<input type="button" value="Edit"/>	<input type="button" value="Copy"/>	<input type="button" value="Delete"/>	<a href="#">id</a>	<a href="#">DCI_id</a>	<a href="#">dosagem_id</a>	<a href="#">forma_id</a>	<a href="#">data_validade</a>	<a href="#">created_at</a>	<a href="#">updated_at</a>
<input type="checkbox"/>	<input type="button" value="Edit"/>	<input type="button" value="Copy"/>	<input type="button" value="Delete"/>	1	1	1	1	2022-06-22	NULL	NULL
<input type="checkbox"/>	<input type="button" value="Edit"/>	<input type="button" value="Copy"/>	<input type="button" value="Delete"/>	2	6	10	5	2022-03-05	2019-06-28 00:52:37	2019-06-28 00:52:37
<input type="checkbox"/>	<input type="button" value="Edit"/>	<input type="button" value="Copy"/>	<input type="button" value="Delete"/>	3	10	15	4	2021-03-06	2019-06-28 00:53:02	2019-06-28 00:53:02
<input type="checkbox"/>	<input type="button" value="Edit"/>	<input type="button" value="Copy"/>	<input type="button" value="Delete"/>	4	7	14	4	2025-03-05	2019-06-28 00:53:42	2019-06-28 00:53:42
<input type="checkbox"/>	<input type="button" value="Edit"/>	<input type="button" value="Copy"/>	<input type="button" value="Delete"/>	5	14	11	1	2026-03-06	2019-06-28 01:33:56	2019-06-28 01:33:56

Figura 2.3: Tabela de medicamentos referente à BD da plataforma.

## 2.6 Frameworks

Uma *framework* é um *template* com diversas funções que podem ser usadas pelo *developer*, e que captura funcionalidades comuns a várias aplicações.

Com uma *framework*, é desnecessário gastar tempo para reproduzir a mesma funcionalidade em diferentes projetos, auxiliando numa gestão ágil de projetos. Esta contém ferramentas, guias, sistemas e componentes que agilizam o processo de desenvolvimento de soluções, ajudando bastante os especialistas de TI na implementação dos seus projetos.

Assim, para uma boa solução de desenvolvimento, deve-se escolher uma boa *framework* de trabalho.

Desta forma, estas plataformas de desenvolvimento fazem com que não seja necessário existir uma preocupação acrescida em reescrever código, podendo o programador focar-se somente na resolução de problemas, ou seja, direcionando os seus esforços para um objetivo final.

Neste sentido, e como parte de uma tendência de procurar reduzir custos e aumentar a produtividade, o uso deste recurso tem se tornado cada vez mais popular. Hoje em dia, já existe uma grande variedade de soluções disponíveis para as mais diversas linguagens, com comunidades que testam e criam diferentes funções [7].

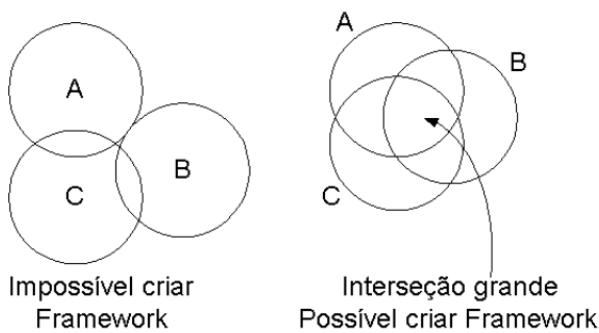


Figura 2.4: Criação de uma *Framework*.

Como mostra a figura 2.4, sendo A, B e C diferentes aplicações, para ser possível a criação de uma *framework* de trabalho comum, estas devem pertencer a um mesmo domínio do problema.

### 2.6.1 Vantagens do uso de *Frameworks*

O principal benefício do uso deste recurso é sua capacidade de economizar tempo no desenvolvimento de *software*. Isto é possível porque existe a reutilização de có-

digos já testados e aparentemente eficazes. Outras vantagens no uso de *frameworks* são:

- Redução de *bugs*: Como já passou por diversos testes, o código de uma *framework* geralmente já está sem *bugs* graves;
- Facilidade de aprendizagem: A maioria das *frameworks* já tem um registo bastante extenso de documentação, o que facilita muito a sua aprendizagem por parte dos programadores;
- Padronização de código: Para que haja compatibilidade, o *developer* deve seguir o mesmo padrão de código usado pela *framework*. Isto contribui para que o código seja mais legível, tornando a manutenção mais fácil;
- Redução de custos: Como todas as bases já são facultadas pela ferramenta, a equipa só precisa de se concentrar na camada de negócio. Isto facilita muito o desenvolvimento de *software* e diminui o tempo de entrega;
- Maior consistência das aplicações: O padrão exigido quando se trabalha com uma *framework*, garante que a aplicação tenha menos falhas do que quando é produzida integralmente desde o início;
- Incentivo ao conhecimento: As *frameworks* são ferramentas e, quanto mais se trabalha com elas, maiores são os conhecimentos adquiridos acerca de seu funcionamento. Em cada projeto, pode-se aproveitar diferentes funcionalidades do recurso para melhorar o resultado final.

### 2.6.2 Desvantagens do uso de *Frameworks*

As vantagens no uso de *frameworks* são muito maiores do que as desvantagens — desde que se saiba escolher a melhor alternativa e se use bem.

Algumas práticas negativas ligadas ao uso deste tipo de ferramenta que devem ser evitadas, podem ser:

- Dependência: É necessário sublinhar que a *framework* é diferente da linguagem de programação usada para escrevê-la. Assim, o programador precisa de conhecer bem a linguagem com a qual está a trabalhar;
- Complexidade de modificação da *framework*: Uma *framework* é uma estrutura complexa com várias funções interligadas. Por este motivo, um *developer* precisa de conhecer muito bem tanto a sua linguagem, como a sua estrutura, se este pretender fazer alterações em qualquer uma das suas funções;

- Códigos desnecessários que podem deixar o programa pesado: Existem *frameworks* de todos os tipos, com as mais variadas funções e diferentes tamanhos. Assim, o programador deve encontrar aquela que tenha apenas as funções que são necessárias para o seu projeto (ou o mínimo possível de componentes extras).

### 2.6.3 Principais *frameworks*

Para uma escolha do modelo de *framework* ideal para as soluções desenvolvidas, é essencial conhecer um pouco sobre cada um dos existentes. Neste sentido, as *frameworks* de *backend* mais utilizadas, são:

- Zend: Alcançou fama rapidamente, principalmente pelo apoio dado por empresas como Google e Microsoft. É uma das *frameworks* mais atualizadas e consistentes do mercado. Não é um modelo simples, sendo que tende a ser mais indicado para os projetos mais robustos;
- Laravel: É considerado uma das *frameworks PHP: Hypertext Preprocessor* (PHP) mais utilizadas no mercado atualmente, funcionando com base no modelo Model-View-Controller (MVC). Isto deve-se principalmente, por ter um carácter robusto e extremamente versátil. O Laravel vem com suporte a API pronto para uso e também possui uma quantidade decente de pacotes que podem estender o seu alcance;
- Spring: A Spring é uma estrutura MVC que utiliza a linguagem Java. O facto de utilizar Java, uma linguagem bastante própria, é um ponto a mais para cada programador Web que a saiba utilizar nos seus projetos. A curva de aprendizagem pode ser bastante acentuada, especialmente para quem não está familiarizado com a linguagem em questão;
- Symfony: Este modelo foi lançado em 2005 e utiliza a arquitetura MVC. O Symfony foi desenvolvido justamente para trabalhar de forma colaborativa com outras metodologias ágeis de desenvolvimento de soluções. Este foca essencialmente em regras de negócio da aplicação. Geralmente é indicado em trabalhos de grande escala e mais robustos.

Existem também diversas *frameworks Web* focadas na parte de *frontend*, sendo estas:

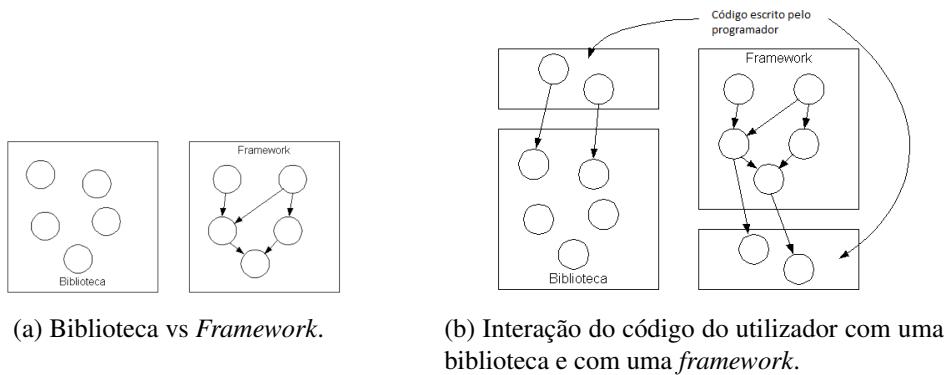
- Angular: é uma *framework de frontend* especializada na criação de aplicações avançadas de uma única página. É uma estrutura muito rica e capaz de criar aplicações completas do lado do cliente;

- React: O React não é uma estrutura/*framework*, mas sim uma biblioteca de *frontend*, no entanto muitos *developers* a consideram uma *framework*, sendo até geralmente comparada nesse contexto. O React foi o primeiro a adotar a arquitetura baseada em componentes que o Angular e muitos outros começaram a adotar posteriormente. O domínio virtual do React torna a manipulação muito mais rápida e fácil de aprender, especialmente graças à sua sintaxe. Este pode ser utilizado tanto no lado do servidor como no lado do cliente;
- Vue: É uma *framework* progressiva, o que significa que se existe já algum projeto, é possível adotar o Vue para uma parte desse projeto, de forma simples. Todo o ecossistema do Vue pode ajudar a construir aplicações *frontend* completas [3];
- Bootstrap: é a alternativa de biblioteca mais conhecida para o desenvolvimento de código de folhas de estilo (CSS), sendo também muitas vezes referido como uma *framework*. Este é responsável pelo estilo visual das páginas e pela criação de um resultado incrível. Uma das principais vantagens de utilizá-lo num projeto está ligada à sua responsividade. É esta ferramenta que dá às telas e aos elementos que as formam, a capacidade de se adequarem ao tamanho do dispositivo do utilizador — seja este um *desktop*, ou um *smartphone*.

#### 2.6.4 *Framework* vs Biblioteca de código

A biblioteca é menos complexa do que a *framework*, sendo cada classe independente das outras. Esta é um dos recursos mais utilizados na área de TI, e cuja principal função é compartilhar soluções já prontas, por meio de funções ou métodos. Por outras palavras, é uma espécie de livraria de implementações de comportamentos, definidas numa linguagem e importadas para o código. Um exemplo de uma biblioteca de códigos bastante utilizada é o jQuery, utilizado para manipulação de código HTML.

A *framework* é geralmente conhecida por ser um conjunto de bibliotecas de códigos abstratos que fazem parte de uma operação maior, e onde as dependências/colaborações estão embutidas, gerando assim um modelo de colaboração que deve ser adaptado ao projeto atual. Esta tem assim um maior grau de complexidade, e está diretamente ligada à arquitetura do *software*. Um exemplo simples são as telas de *login*, pois possuem sempre características semelhantes. Por terem sempre a mesma estrutura, foi criada uma *framework* desta operação, de forma a que essa função seja implementada rapidamente no código, sem que seja preciso reescrevê-la sempre que se quiser acrescentar uma tela de *login* num projeto.

Figura 2.5: Bibliotecas e *Frameworks*.

A figura 2.5 mostra esquematicamente e de forma simplificada a diferença principal entre ambos os conceitos.

## 2.7 Conclusões

Todos os conceitos abordados se englobam, de alguma forma, na plataforma desenvolvida. Foi assim importante referir os conceitos de Cliente e de Servidor, entendendo as diferenças entre ambos e abordar o modo de funcionamento de uma arquitetura Cliente-Servidor, arquitetura utilizada na plataforma, com o uso de um servidor local. Foi implementada uma estrutura bastante pensada de uma BD eficiente, juntamente com uma *framework* de PHP, que auxiliou bastante na criação de uma solução eficaz e segura. Na próxima secção iremos abordar em detalhe todas as ferramentas utilizadas em todas as fases do projeto desenvolvido.



# Capítulo 3

## Tecnologias e Ferramentas Utilizadas

### 3.1 Introdução

Na realização deste projeto foram utilizadas diversas tecnologias no âmbito das aplicações *Web*. Desta forma, irão ser abordadas com detalhe as ferramentas e tecnologias que suportam o servidor local e a base de dados, bem como a *framework* utilizada para a implementação. Este capítulo encontra-se organizado nas seguintes secções:

- secção 3.2 - **Servidor** - apresenta o *software* utilizado como ambiente de desenvolvimento da plataforma MedGest;
- secção 3.2.1 - **Apache** - descreve o servidor *Web* que foi utilizado como suporte à plataforma;
- secção 3.2.2 - **MySQL** - especifica e caracteriza o SGBD utilizado;
- secção 3.3 - **Linguagens utilizadas** - apresenta as linguagens utilizadas na construção da plataforma;
- secção 3.3.1 - **SQL** - descreve a linguagem de pesquisa declarativa utilizada na base de dados;
- secção 3.3.2 - **PHP** - define a linguagem de programação usada PHP e a sua usabilidade.
- secção 3.3.3 - **HTML e CSS** - define a linguagem de interpretação utilizada nas *webpages*;

- secção 3.4 - **Layout** - descreve o *layout* utilizado na aplicação;
- secção 3.5 - **IDE** - refere qual foi o Integrated Development Environment (IDE) utilizado no desenvolvimento da aplicação e descreve as suas principais características;
- secção 3.6 - **Framework** - especifica e caracteriza a *framework* e bibliotecas utilizadas no desenvolvimento da plataforma;
- secção 3.7 - **Conclusões** - apresenta as conclusões sobre a escolha de todas as tecnologias e ferramentas apresentadas anteriormente.

## 3.2 Servidor

Numa fase inicial do projeto, este estava a ser implementado com o auxilio da multiplataforma XAMPP, que tem como base o SGBD *MariaDB*, o servidor *Web Apache* e os interpretadores para linguagens de script *PHP* e *Perl*.

No entanto, este ambiente de desenvolvimento local começou a dar alguns problemas no que toca à construção das páginas *Web* e interação com o *layout*.

Assim, o servidor local que optei por utilizar no desenvolvimento deste projeto foi o Laragon. O Laragon é um ambiente de desenvolvimento gratuito, bastante completo, simplista e estável em ambiente Windows (sistema operativo utilizado no desenvolvimento do projeto). Com o Laragon não existe a necessidade de preocupação com a criação de um *VirtualHost*, sendo este criado de forma automática. É também possível trabalhar com diferentes bases de dados já pré configuradas (por exemplo: MySQL, PostgreSQL e MongoDB), escolher a versão do PHP a usar bem como o servidor *Web* de preferência (Apache ou Nginx). Existe também um *Mail Catcher* e um *Mail Sender* já pré configurados.

Algumas ferramentas que acompanham o Laragon, são:

- Cmder: é um emulador da consola para Windows, que para além de ser esteticamente superior à linha de comandos (Cmd) comum, é portável nas suas configurações e vem com o Git integrado;
- HeidiSQL: é uma ferramenta visual de acesso à base de dados, extremamente leve e com tudo o que é necessário para gerir a base de dados das aplicações;
- *Auto Virtual Hosts*: o Laragon vem com um *virtual host* embutido, ou seja, nada de apontar para o *localhost* dentro dos projetos, pois estes tem uma extensão de URL próprio;

- QuickCreate: é uma ferramenta do Laragon que permite abstrair o processo de criação de uma aplicação.



Figura 3.1: Ambiente Laragon.

### 3.2.1 Apache

O servidor *Web* utilizado e escolhido no ambiente de desenvolvimento Laragon, foi o *Apache*.

O *Apache* é o servidor *Web* mais utilizado no mundo. Este foi criado em 1995 pelo *software developer* Rob McCool, que trabalhava na altura no National Center for Supercomputing Applications (NCSA).

Segundo dados relativos ao ano de 2016, o *Apache* contribuiu em cerca de 46% de todos os *websites* ativos e 43% dos *websites* mais visitados. O servidor *Apache* é também compatível com o protocolo HTTP.

De forma a garantir a segurança na transmissão dos dados através do protocolo HTTP, o *Apache* possui um módulo que permite obter a capacidade do servidor responder a pedidos utilizando o Hypertext Transfer Protocol Secure (HTTPS). O protocolo HTTPS utiliza o protocolo de segurança Secure Sockets Layer (SSL) para encriptar todos os dados transmitidos entre o cliente e o servidor. É ainda possível gerar certificados digitais X.509, pois o protocolo SSL garante esta compatibilidade [19].

Desta forma, a escolha deste servidor garante a compatibilidade em termos de protocolos de comunicação e segurança.

### 3.2.2 MySQL

O SGBD que vem por defeito na versão do Laragon que utilizei é o MySQL. Assim sendo, e visto ser o SGBD *open-source* mais utilizado e popular no desenvolvimento *Web*, optei por mantê-lo e com ele criar e gerir a BD do projeto. O

MySQL é também conhecido pelo seu ótimo desempenho, confiabilidade e facilidade de uso.

O MySQL foi criado na Suécia, por David Axmark, Allan Larsson e o finlandês Michael Widenius, por volta de 1980. Este é um SGBD que utiliza a linguagem SQL, a qual iremos abordar nas próximas secções.

Este SGBD é conhecido pela facilidade, segurança e robustez, sendo este utilizado por diversas grandes corporações como a NASA, HP, Bradesco, Sony, etc. A sua interface simples, e também a sua capacidade de funcionamento em diversos sistemas operativos, são alguns dos motivos para este *software* ser tão usado atualmente, e o seu uso estar a crescer cada vez mais[10].



Figura 3.2: Logótipo MySQL.

### 3.3 Linguagem utilizadas

Para construir e implementar a lógica e funcionalidades da plataforma, foram utilizadas diversas linguagens. A linguagem de pesquisa declarativa e consulta estruturada utilizada na BD foi o SQL, enquanto que a linguagem de programação utilizada para responder à lógica implementada no lado do servidor foi o PHP. Por fim, a linguagem de interpretação das *webpages* foi o HTML, juntamente com folhas de estilos Cascading Style Sheets (CSS).

#### 3.3.1 SQL

A linguagem Structured Query Language (SQL) foi inicialmente desenvolvida no início dos anos 70 nos laboratórios da International Business Machines (IBM) pelos investigadores Raymond F. Boyce e Donald D. Chamberlin, com o objetivo de demonstrar a viabilidade do modelo relacional proposto por E.F.Codd, descrito na

subsecção 2.5.1 [18]. A linguagem SQL diferencia-se das outras linguagens de consulta à BD, na medida em que cada consulta especifica a forma do resultado, e não o caminho até ele.

A expansão de outras linguagens levou à necessidade de ser criado e adaptado um padrão para a linguagem. Deste modo, a American National Standards Institute (ANSI) padronizou o SQL em 1986 e a International Organization for Standardization (ISO) em 1987 [22].

### 3.3.2 PHP

A linguagem de programação utilizada para o desenvolvimento da plataforma foi o PHP. Esta é uma linguagem de *script opensource* bastante utilizada e adequada, principalmente para o desenvolvimento de aplicações *Web*, e que pode ser embutida dentro do código HTML da página.

Um exemplo de código PHP pode ser o seguinte:

```
<!DOCTYPE HTML>
<html>
    <head>
        <title>Exemplo</title>
    </head>
    <body>

        <?php
            echo "Isto é um script em PHP!";
        ?>

    </body>
</html>
```

Excerto de Código 3.1: Exemplo de código em PHP.

Como é visível no exemplo anterior, o código PHP é delimitado pelas instruções de processamento de início (`<?php`) e fim (`?>`) que permitem que seja possível entrar e sair do "modo PHP".

O que distingue o PHP de uma linguagem como o JavaScript no lado do cliente, é que o código é executado no servidor, gerando o HTML que é então enviado para o navegador. Este recebe assim os resultados da execução desse *script* [16].

### 3.3.3 HTML e CSS

O HTML é uma linguagem de marcação e interpretação utilizada na construção de *webpages*, que teve a sua primeira implementação publicada por Tim Berners-Lee no final de 1991.

O HTML em conjunto com o CSS e o JavaScript, formam as três principais tecnologias da WWW. Os *browsers* executados no lado do cliente recebem os documentos HTML vindos de um servidor Web e interpretam estes documentos através de um compilador já implementado no *browser*.

O CSS é um mecanismo usado para dar estilo aos documentos HTML.

Este foi desenvolvido pelo World Wide Web Consortium (W3C) em 1996, e surgiu da necessidade de um mecanismo de formatação da *webpage* com a utilização de *tags*, que o HTML não tinha. *Tags* como `<font>` foram introduzidas apenas na versão 3.2 do HTML e causaram muitos problemas para os *developers*, pois como os *sites* tinham diferentes fontes, cores e estilos, era um processo longo, doloroso e caro reescrever o código. Assim, o CSS foi criado e veio a resolver este problema.

O CSS separa o conteúdo da representação visual do *site* sendo possível, por exemplo, alterar a cor do texto e do fundo, a fonte e o espaçamento entre os parágrafos. Com esta ferramenta é também possível criar tabelas, usar variações de *layouts*, ajustar imagens, e assim por diante [11].

## 3.4 *Layout*

A construção da plataforma na parte do cliente foi conseguida com a ajuda de um modelo de *layout* produzido pelo *pixelcave*, uma plataforma que fornece recursos e *templates* para desenvolvimento Web, denominado de OneUI.

Os produtos comercializados nesta plataforma ajudaram bastante a projetar, codificar e a construir a interface do utilizador da aplicação Web com mais rapidez, eficiência e menos esforço.

O OneUI é assim um modelo de painel de administração e *framework* de interface do utilizador do Bootstrap 4.x, super flexível e com suporte à *framework* Laravel, que permite construir todos os tipos de páginas utilizando o mesmo *layout*. Este *layout* foi construído com as folhas de estilo Syntactically Awesome Style Sheets (Sass) e a linguagem de programação ECMAScript 6 (ES6), vindo embutido com algumas ferramentas de desenvolvimento como o Webpack 4, Babel 7, Gulp 4, Autoprefixer e Browsersync.

O OneUI vem com versões em HTML e PHP. Este vem também no formato Laravel se o projeto em questão for baseado nesta *framework*.

## 3.5 IDE

O IDE utilizado para o desenvolvimento da plataforma MedGest foi o PhpStorm. Este IDE oferece inúmeras vantagens, entre as quais:

- Licença gratuita para estudantes;
- Suporte para várias *frameworks*;
- Inclui todas as ferramentas do PHP: O editor entende bem a estrutura de um projeto, suportando todos os recursos da linguagem PHP. Este fornece também o preenchimento de código, refatoração, prevenção de erros e muito mais;
- Tecnologias *frontend* incluídas: Inclui diversas tecnologias associadas ao *frontend* da página, como o HTML5, CSS, Sass, Stylus, TypeScript e JavaScript, com refatoração e testes de unidade disponíveis. É também possível ver as mudanças de forma instantânea no navegador graças ao *Live Edit*;
- Ferramentas de desenvolvimento internas: Integração dos sistemas de controlo de versão, suporte à implementação remota, bases de dados/SQL, ferramentas de linha de comandos, Docker, Composer, REST Client, etc.

O PhpStorm tem imensos recursos que podem ser bem utilizados e explorados, como a Assistência Inteligente à Codificação. Esta ferramenta permite que ocorram centenas de inspeções que verificam o código enquanto este está a ser escrito, analisando o projeto como um todo. Assim, com o formatador de código e as correções rápidas torna-se mais fácil escrever um código limpo e de fácil manutenção.

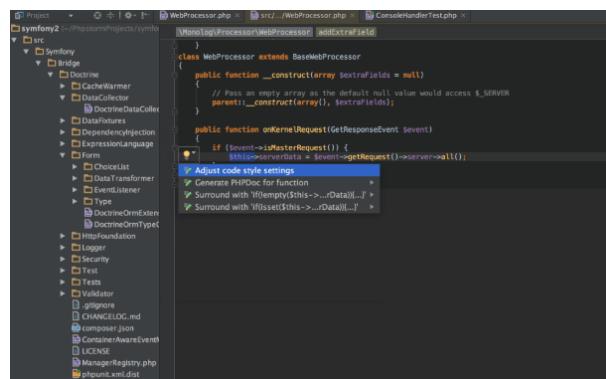


Figura 3.3: Assistência inteligente no PhpStorm.

Existe também um recurso que trata da navegação inteligente por código. Esta funcionalidade é bastante eficiente, rápida e extremamente útil para projetos mais extensos.

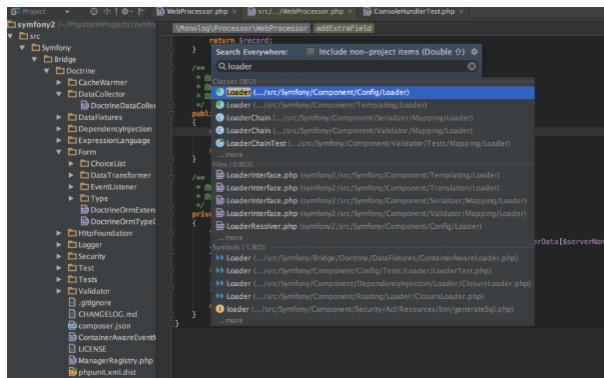


Figura 3.4: Navegação inteligente no PhpStorm.

O mecanismo de refatoração acontece de forma rápida e segura. Esta é uma ferramenta extremamente útil, e que foi bastante utilizada no desenvolvimento do projeto.

Desta forma, processos com o renomear, mover, excluir, extrair métodos, variáveis *inline* etc, realizam automaticamente as alterações em todo o projeto numa questão de segundos e com a opção destas serem desfeitas com segurança.

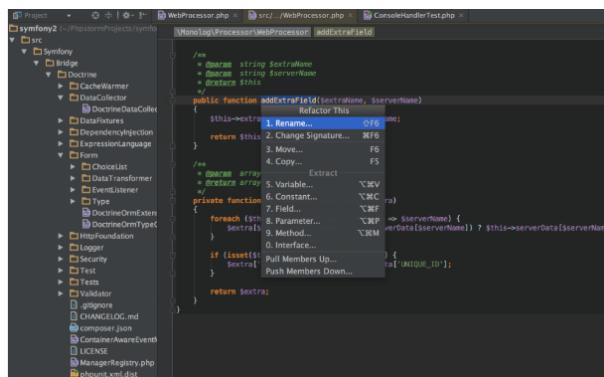


Figura 3.5: Refatoração rápida e segura no PhpStorm.

O PhpStorm tem também um mecanismo de depuração e testes fáceis, sem ser necessária qualquer configuração. Este fornece informações sobre o que acontece na aplicação em cada etapa. Este funciona com o Xdebug e o Zend Debugger, e pode ser usado local e remotamente. Testes Unitários com PHPUnit, BDD com Behat e integração com *profiler* estão também disponíveis [17].

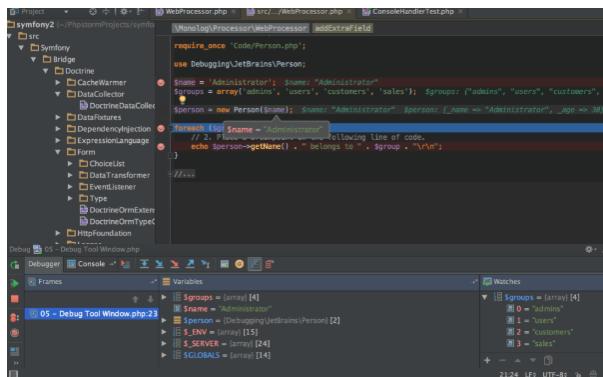


Figura 3.6: Depuração e testes no PhpStorm.

## 3.6 Framework

A *framework* de desenvolvimento utilizada na implementação do projeto foi o Laravel.

Esta é uma *framework backend* robusta e de desenvolvimento rápido para PHP, livre e *opensource*, cujo o principal objetivo é permitir trabalhar de forma estruturada e rápida. Esta oferece um *time-to-market* reduzido e também uma arquitetura de código muito organizada, o que facilita a manutenção de um sistema e também o trabalho em equipa.

O Laravel é uma das *frameworks* mais populares atualmente, e possui uma equipa de *developers* ativa e extremamente competente. Tem ainda uma comunidade enorme e bastante aceitação no mercado. Este *status* de popularidade acontece devido à agilidade de programação de sistemas complexos envolvendo uma grande quantidade de recursos, tais como segurança, acesso a dados e arquitetura da aplicação. Todas estas características, que estão eminentes em qualquer sistema *Web*, são fornecidas nativamente pelo Laravel de um modo simples e intuitivo [27].

O Laravel utiliza o *Composer* para gerir as suas dependências. O *Composer* é assim uma ferramenta que permite gerir, de forma fácil, os pacotes de terceiros da aplicação.

O Laravel destaca-se também bastante na documentação que fornece aos seus utilizadores. Esta é bastante intuitiva, fácil de encontrar e completa.

Esta *framework* utiliza o *Blade*. Este é o compilador de *templates* do Laravel. A diferença dele para outros *templates* é a sua flexibilidade, pois este não restringe o uso de PHP puro misturando a sintaxe do *template*. Os arquivos *blade* devem utilizar a extensão ".blade.php". O grande objetivo do *Blade* é reduzir a quantidade de código PHP inserido no meio do HTML e aumentar o reuso. Os dois principais benefícios do uso do *Blade* são a herança e as secções, permitindo trabalhar facilmente com o conceito de *master page*.

O pedaço de código em baixo representa uma possível estrutura de uma *master page*:

```
<!-- master.blade.php -->
<html>
    <head>
        <title>Exemplo de arquivo Blade</title>
    </head>
    <body>
        <div class="container">
            @yield('conteudo')
        </div>
    </body>
</html>
```

Excerto de Código 3.2: Exemplo de uma *master page* utilizando o *blade*.

A implementação de uma página denominada de "*dashboard.blade.php*" que irá herdar o *layout* da *master page*, é feita da seguinte forma:

```
<!-- dashboard.blade.php -->
@extends('master')
@section('conteudo')
    <p>O conteúdo da nosso dashboard vem aqui!</p>
@endsection
```

Excerto de Código 3.3: Exemplo de uma página que herda o *layout* da *master page* com o *blade*.

Nesta página estamos a herdar toda a estrutura da *master page* utilizando o comando `@extends`, e a definir o conteúdo específico da página através do comando `@section`.

No Laravel existe também o *Eloquent*, que é o *Object-relational mapping* (ORM) padrão desta *framework*. Este aplica o *Design Patter ActiveRecord* onde cada tabela da BD é representada no código através de uma classe “*Model*” que é usada para interagir com essa tabela. Os *Models* permitem a consulta de dados nas tabelas, bem como operações de *insert*, *update* e *delete*. A vantagem do *Eloquent* é a facilidade de implementação, sem ter que escrever SQL puro misturado com o código. Por exemplo, para fazer uma listagem de todos os registo de uma determinada tabela, basta invocar o método `all()` que faz parte do *Eloquent*. No caso de não querermos utilizar o *Eloquent*, existe também o *QueryBuilder*.

Como o próprio nome indica, o *QueryBuilder* é um construtor de *queries*, e pode ser usado para executar a maioria das operações na base de dados.

O Laravel fornece também a consola *Artisan*, que é uma interface da linha de comandos que fornece vários comandos para facilitar o desenvolvimento da aplicação. Para visualizar todos os comandos que esta pode fornecer, basta escrever: `php artisan list` na linha de comandos [12].

O desenvolvimento em Laravel posiciona esta *framework* entre as melhores da atualidade, pois esta consegue satisfazer bastante todas as principais necessidades do desenvolvimento *Web*.



Figura 3.7: Logótipo do Laravel.

O Laravel torna também incrivelmente simples a utilização da biblioteca Bootstrap no *layout* da aplicação.

O Bootstrap é uma ferramenta *opensource*, desenvolvida pelo *designer* Mark Otto e pelo *developer* Jacob Thornton, como uma estrutura que ajuda na consistência do estilo e aspeto das *webpages* [8]. Através desta biblioteca, muitas vezes referida como sendo uma *framework*, foi possível a personalização de diversas componentes da plataforma MedGest, como por exemplo os botões, formulários, barras de navegação, tabelas, entre outros.

## 3.7 Conclusões

Neste capítulo foram caracterizadas e definidas todas as ferramentas utilizadas ao longo do desenvolvimento e implementação da plataforma. Todas elas foram essenciais para cumprir os objetivos propostos para o trabalho.

Foi assim apresentado o *software* utilizado como ambiente de desenvolvimento, e que forneceu ferramentas como o servidor *Web* e a base de dados. Foram também descritas todas as linguagens utilizadas para o desenvolvimento da plataforma, tanto de *frontend* como de *backend*. Por fim, foi caracterizada com algum detalhe a *framework* que tornou todo o processo de desenvolvimento mais simples e seguro.



# Capítulo 4

## Engenharia de *Software*

### 4.1 Introdução

Neste capítulo irão ser apresentadas as ferramentas utilizadas no domínio da Engenharia de *Software*, nomeadamente os requisitos funcionais e não funcionais da plataforma e os diagramas dos casos de uso, referentes a ambos os tipos de utilizadores.

Os diagramas de casos de uso representam uma possível utilização do sistema por um ator, que pode ser uma pessoa, dispositivo físico, mecanismo ou subsistema que interage com o sistema alvo, utilizando algum dos seus serviços, sendo que podem ser identificados com base nos requisitos. Nestes diagramas podem existir dois tipos de atores: o ator principal, que interage diretamente com o sistema, e o ator secundário que interage com outros atores.

Desta forma, o capítulo encontra-se dividido da seguinte maneira:

- secção 4.2 - **Requisitos** - resumo e breve descrição dos conceitos de requisitos funcionais e não funcionais;
- secção 4.2.1 - **Requisitos funcionais** - necessidades subjacentes a uma funcionalidade da plataforma;
- secção 4.2.2 - **Requisitos não funcionais** - necessidades não diretamente subjacentes a uma funcionalidade, mas fundamental a um bom funcionamento;
- secção 4.3 - **Casos de uso** - especificação dos casos de uso do administrador e do farmacêutico, no domínio da aplicação *Web*;

- secção 4.4 - **Padrões de desenho arquitecturais** - especifica o padrão de desenho arquitectural utilizado na plataforma;
- secção 4.4.1 - **Padrão MVC** - descreve o padrão MVC, detalhando-o no contexto da plataforma desenvolvida;
- secção 4.5 - **Conclusões** - reflexão nas principais conclusões do capítulo.

## 4.2 Requisitos

Nesta secção irão ser abordados os requisitos funcionais e não funcionais, que foram considerados durante o desenvolvimento da plataforma.

Os requisitos funcionais descrevem explicitamente as funcionalidades e serviços do sistema, documentando como o sistema deve reagir a entradas específicas, como se deve comportar em determinadas situações e o que não deve acontecer, sendo que os requisitos não devem ter definições contraditórias.

Já os requisitos não funcionais definem propriedades e restrições do sistema como a segurança, desempenho e espaço em disco. Estes podem ser referentes a todo o sistema ou a partes individuais deste. Se os requisitos não funcionais não forem atendidos, o sistema torna-se inútil [25].

Todos estes requisitos foram encontrados tanto na proposta de projeto, como com a ajuda do meu orientador de projeto e com colaboração do Centro Hospitalar Cova da Beira.

### 4.2.1 Requisitos funcionais

Este tipo de requisitos são adjacentes à aplicação *Web*, logo estes contêm as especificações completas, claras e numeradas de todas as necessidades da plataforma para um correto funcionamento da mesma, permitindo que a representação gráfica do *software* seja a mais correta possível e a idealizada pelo utilizador.

A seguinte tabela descreve a ordem pela qual os requisitos foram encontrados e irão ser abordados:

ID Requisito	Nome do Requisito	Descrição do Requisito
RF01	<i>Login</i>	Contém todos os requisitos associados a RF01.X. O utilizador não tem de ter conta para aceder a esta página.

<b>ID Requisito</b>	<b>Nome do Requisito</b>	<b>Descrição do Requisito</b>
RF02	Registo	Contém todos os requisitos associados a RF02.X. O utilizador tem que ser administrador para aceder a esta página.
RF03	Reposição de palavra-passe	Contém todos os requisitos associados a RF03.X. O utilizador não tem de ter conta para aceder a esta página.
RF04	Página principal de farmacêutico	Contém todos os requisitos associados a RF04.X. O utilizador tem que ser um farmacêutico para aceder a esta página.
RF05	Página principal de administrador	Contém todos os requisitos associados a RF05.X. O utilizador tem que ser administrador para aceder a esta página.
RF06	Listar Contas	Contém todos os requisitos associados a RF06.X. O utilizador tem que ser administrador para aceder a estas páginas.
RF07	Inserir medicamentos	Contém todos os requisitos associados a RF07.X. O utilizador tem que ser administrador para aceder a esta página.
RF08	Lista Local de Medicamentos	Contém todos os requisitos associados a RF08.X. O utilizador tem que ter conta para aceder a esta página.
RF09	Lista Global de Medicamentos	Contém todos os requisitos associados a RF09.X. O utilizador tem que ter conta para aceder a esta página.
RF10	Fazer um pedido	Contém todos os requisitos associados a RF10.X. O utilizador tem que ter conta para aceder a esta página.
RF11	Lista de Pedidos Enviados	Contém todos os requisitos associados a RF11.X. O utilizador tem que ter conta para aceder a esta página.
RF12	Lista de Pedidos Recebidos	Contém todos os requisitos associados a RF12.X. O utilizador tem que ter conta para aceder a esta página.
RF13	Histórico de Medicamentos	Contém todos os requisitos associados a RF13.X. O utilizador tem que ter conta para aceder a esta página.

<b>ID Requisito</b>	<b>Nome do Requisito</b>	<b>Descrição do Requisito</b>
RF14	Histórico de um determinado medicamento	Contém todos os requisitos associados a RF14.X. O utilizador tem que ter conta para aceder a esta página.
RF15	Lista de Contactos	Contém todos os requisitos associados a RF15.X. O utilizador tem que ter conta para aceder a esta página.
RF16	Histórico de um determinado centro hospitalar	Contém todos os requisitos associados a RF16.X. O utilizador tem que ter conta para aceder a esta página.
RF17	Definições	Contém todos os requisitos associados a RF17.X. O utilizador tem que ter conta para aceder a esta página.

Tabela 4.1: Organização de requisitos funcionais da plataforma.

<b>ID RF01.X</b>	<b>Nome do Requisito</b>	<b>Descrição do Requisito</b>
RF01.1	<i>Email</i> do utilizador	- O preenchimento deste requisito pelo utilizador é obrigatório. - Caso o utilizador insira um email mal formatado (sem o @), deve ser mostrada uma mensagem de erro a dizer "Inclua um @ no endereço de email", e a caixa de texto fica a vermelho assim que o utilizador começar a editar esse campo. - Caso o email não tenha nenhum caractere depois do "@", é mostrada uma mensagem de erro a dizer "Introduza uma parte a seguir a @". email@teste está incompleto.", e a caixa de texto fica a vermelho assim que o utilizador começar a editar esse campo. - Caso o <i>email</i> não exista deve ser mostrada uma mensagem de falha de autenticação, com a mensagem "Essas credenciais não existem nos nossos registos.", e a caixa de texto fica a vermelho assim que o utilizador começar a editar esse campo.

ID RF01.X	Nome do Requisito	Descrição do Requisito
RF01.2	Palavra-passe do utilizador	O preenchimento deste requisito pelo utilizador é obrigatório. - Abrange codificação alfanumérica, podendo conter letras capitalizadas ou não. - É permitido o uso de caracteres especiais. - Caso a <i>password</i> não exista deve ser mostrada uma mensagem de falha de autenticação, com a mensagem "Essas credenciais não existem nos nossos registos.", e a caixa de texto fica a vermelho assim que o utilizador começar a editar esse campo.
RF01.3	Formulário	Os dados referentes a RF01.1 e RF01.2 devem estar inseridos num formulário branco em bloco, centrado na página.
RF01.4	Botão de <i>login</i>	Deve ser azul, e estar centrado na página, imediatamente depois do formulário referido em RF01.3.

Tabela 4.2: Requisitos funcionais de *login*.

ID RF01.3.X	Nome do Requisito	Descrição do Requisito
RF01.3.1	Cabeçalho	Deve ter a cor azul. - Deve conter a informação sobre em que sessão de <i>login</i> estamos (farmacêutico ou administrador). - Opção de ir para a sessão de <i>login</i> do administrador, opção esta que fica mais escura quando o utilizador passar com o rato em cima. - Opção de hiperligação para a página de esquecimento da palavra-passe, abordada em RF03, hiperligação esta que fica mais escura quando o utilizador passar com o rato em cima.
RF01.3.2	Corpo	Antes dos campos de preenchimento referidos em RF01.1 e RF01.2 deve estar o nome da plataforma, juntamente com o logótipo e uma mensagem a dizer "Por favor insira os seus dados de <i>login</i> ".

Tabela 4.3: Requisitos funcionais do formulário de *login*.

ID RF02.X	Nome do Requisito	Descrição do Requisito
RF02.1	Título e subtítulo	Deve ter o título "Criar conta:", a preto, no canto lateral esquerdo da página. - Deve ter subtítulo Registar farmacêutico:", a cinzento, imediatamente abaixo do título.
RF02.2	Nome de utilizador	O preenchimento deste campo é obrigatório. - Se este campo for deixado em branco aparece uma mensagem a dizer "Preencha este campo". - Abrange codificação alfanumérica, podendo conter letras capitalizadas ou não.
RF02.3	Centro Hospitalar	Menu <i>dropdown</i> com todos os centros hospitalares existentes.
RF02.4	Email	O preenchimento deste campo é obrigatório. - Caso o utilizador insira um <i>email</i> mal formatado (sem o @), deve ser mostrada uma mensagem de erro a dizer "Inclua um @ no endereço de email". - Caso o <i>email</i> não tenha nenhum caractere depois do "@", é mostrada uma mensagem de erro a dizer "Introduza uma parte a seguir a @". <i>email@teste</i> está incompleto..
RF02.5	Palavra-passe	O preenchimento deste campo é obrigatório. - O limite mínimo é de 6 caracteres, não tendo limite máximo. Se o limite mínimo de caracteres não for cumprido, deve mostrar a mensagem "A <i>password</i> deve ter pelo menos 6 caracteres". - Abrange codificação alfanumérica, podendo conter letras capitalizadas ou não. - É permitido o uso de caracteres especiais. - Deve haver um campo de confirmação da palavra-passe. - Ao enviar o formulário, ambos os campos (palavra-passe e confirmação desta) devem ser testados, coincidindo. - Caso as palavras-passes inseridas sejam diferentes, então deverá aparecer uma mensagem de erro com a mensagem "As <i>passwords</i> não coincidem".
RF02.6	Botão	Deve estar imediatamente a baixo do requisito referenciado em RF02.5. - Deve ter a cor azul. - Deve conter a palavra "Registrar".

Tabela 4.4: Requisitos funcionais do registo.

ID RF03.X	Nome do Requisito	Descrição do Requisito
RF03.1	Email	O preenchimento deste campo é obrigatório. - Caso o utilizador insira um <i>email</i> mal formatado (sem o @), deve ser mostrada uma mensagem de erro a dizer "Inclua um @ no endereço de email". - Caso o <i>email</i> não tenha nenhum caractere depois do "@", é mostrada uma mensagem de erro a dizer "Introduza uma parte a seguir a @". <i>email@teste</i> está incompleto.".
RF03.2	Botão de envio	Deverá existir um botão azul imediatamente em baixo da caixa de preenchimento do <i>email</i> referenciado em RF03.1. - Ao clicar no botão, este deve verificar se o <i>email</i> existe na base de dados. Se existir, envia um <i>email</i> com um <i>link</i> de reposição da palavra-passe. Se não existir mostra uma mensagem de erro que diz "Não existe nenhum utilizador com esse endereço de email."

Tabela 4.5: Requisitos funcionais da reposição de palavra-passe.

ID RF04.X	Nome do Requisito	Descrição do Requisito
RF04.1	Carroussel	Deve existir um carrousel a passar imagens de forma automática (com setas de ambos os lados, como forma de passagem manual), com um <i>timer</i> de aproximadamente 5 segundos - Este deve ter bolinhas pequenas em baixo, a representar a imagem atual.
RF04.2	Título	Deve ter um título, imediatamente em cima do carrousel referido em RF04.1, com a frase "Dashboard   Área de Farmacêutico", sendo que a palavra "Dashboard" deve estar a preto, e o restante do título a azul.

ID RF04.X	Nome do Requisito	Descrição do Requisito
RF04.3	Barra superior	Barra na parte superior da página principal. - Opção de "esconder" o menu lateral referido em RF04.4. na parte mais à esquerda da barra. - Menu <i>dropdown</i> na parte mais à direita da barra, com o nome do farmacêutico como título. - Ao ser clicado no <i>dropdown</i> , aparecimento das opções de definições (a ser referidas em RF17.X), e de um botão de <i>logout</i> , onde o utilizador pode sair da sua conta, e voltar à página referida em RF01.X.
RF04.4	Menu lateral	Menu no lado esquerdo da página principal. - Menu cinzento. - Opção de mudança de cor de partes da plataforma no cabeçalho do menu (tudo o que está escrito ou desenhado a azul, passa para a cor escolhida pelo utilizador). - Logótipo clicável que remete à página principal da plataforma também no cabeçalho, sendo que este também é afetado pela mudança de cor.

Tabela 4.6: Requisitos funcionais da página principal.

ID RF04.4.X	Nome do Requisito	Descrição do Requisito
RF04.4.1	<i>Dashboard</i>	Secção inicial clicável que diz " <i>Dashboard</i> " e que remete o farmacêutico para a sua página principal.
RF04.4.2	Medicamentos	Secção de Medicamentos, onde tem 2 opções clicáveis (Lista Local e Lista Global), que remetem às páginas a ser descritas em RF08.X e RF09.X, respetivamente.
RF04.4.3	Pedidos	Secção de Pedidos, onde tem 3 opções clicáveis (Fazer pedido, Pedidos Enviados e Pedidos Recebidos), que remetem às páginas a ser descritas em RF10.X, RF11.X e RF12.X, respetivamente.
RF04.4.4	Histórico	Secção de histórico de medicamentos, onde tem apenas 1 opção clicável (Medicamentos), que remete à página a ser descrita em RF13.X.

<b>ID RF04.4.X</b>	<b>Nome do Re- quisito</b>	<b>Descrição do Requisito</b>
RF04.4.5	Contactos	Secção de contactos, onde tem apenas 1 opção clicável (Lista de contactos), que remete à página a ser descrita em RF15.X.

Tabela 4.7: Requisitos funcionais do menu lateral da página principal do farmacêutico.

<b>ID RF05.X</b>	<b>Nome do Re- quisito</b>	<b>Descrição do Requisito</b>
RF05.1	Carroussel	Deve existir um caurrousel a passar imagens de forma automática (com setas de ambos os lados, como forma de passagem manual), com um <i>timer</i> de aproximadamente 5 segundos - Este deve ter bolinhas pequenas em baixo, a representar a imagem atual.
RF05.2	Título	Deve ter um título, imediatamente em cima do carrousel referido em RF05.1, com a frase "Dashboard   Área de Administrador", sendo que a palavra "Dashboard" deve estar a preto, e o restante do título a azul.
RF05.3	Barra superior	Barra na parte superior da página principal. - Opção de "esconder" o menu lateral referido em RF05.4. na parte mais à esquerda da barra. - Menu <i>dropdown</i> na parte mais à direita da barra, com o nome do administrador como título. - Ao ser clicado no <i>dropdown</i> , aparecimento das opções de definições (a ser referidas em RF15.X), e de um botão de <i>logout</i> , onde o utilizador pode sair da sua conta, e voltar à página referida em RF01.X.

ID RF05.X	Nome do Requisito	Descrição do Requisito
RF05.4	Menu lateral	Menu no lado esquerdo da página principal. - Menu cinzento. - Opção de mudança de cor de partes da plataforma no cabeçalho do menu (tudo o que está escrito ou desenhado a azul, passa para a cor escolhida pelo administrador). - Logótipo clicável que remete à página principal da plataforma também no cabeçalho, sendo que este também é afetado pela mudança de cor.

Tabela 4.8: Requisitos funcionais da página principal do administrador.

ID RF05.4.X	Nome do Requisito	Descrição do Requisito
RF05.4.1	<i>Dashboard</i>	Secção inicial clicável que diz "Dashboard" e que remete o administrador para a sua página principal .
RF05.4.2	Gerir contas	Secção de gestão de contas, onde tem 2 opções clicáveis (Criar Conta e Listar Contas), que remetem às páginas a ser descritas em RF02.X e RF06.X, respetivamente.
RF05.4.3	Medicamentos	Secção de Medicamentos, onde tem 3 opções clicáveis (Inserir, Lista Local e Lista Global), que remetem às páginas a ser descritas em RF07.X, RF08.X e RF09.X, respetivamente.
RF05.4.4	Pedidos	Secção de Pedidos, onde tem 3 opções clicáveis (Fazer pedido, Pedidos Enviados e Pedidos Recebidos), que remetem às páginas a ser descritas em RF10.X, RF11.X e RF12.X, respetivamente.
RF05.4.5	Histórico	Secção de histórico de medicamentos, onde tem apenas 1 opção clicável (Medicamentos), que remete à página a ser descrita em RF13.X.
RF05.4.6	Contactos	Secção de contactos, onde tem apenas 1 opção clicável (Lista de contactos), que remete à página a ser descrita em RF15.X.

Tabela 4.9: Requisitos funcionais do menu lateral da página principal do administrador.

<b>ID RF06.X</b>	<b>Nome do Requisito</b>	<b>Descrição do Requisito</b>
RF06.1	Título e subtítulo	Página com título "Listar Contas", a preto, no canto lateral esquerdo da página. - Página com subtítulo "Lista de Farmacêuticos:", a cinzento, , no canto lateral esquerdo da página, imediatamente em baixo do título.
RF06.2	Tabela	Tabela com paginação. - Tabela com campos "#", referente ao ID de cada farmacêutico, "Nome", referente ao nome de cada farmacêutico, "Email", referente ao <i>email</i> de cada farmacêutico e "Hospital", referente ao centro hospitalar em que cada farmacêutico está inserido. - Cada linha da coluna deve ter 2 botões, um para editar os dados de um utilizador e outro para eliminar um utilizador da base de dados.

Tabela 4.10: Requisitos funcionais da página de listar contas.

<b>ID RF07.X</b>	<b>Nome do Requisito</b>	<b>Descrição do Requisito</b>
RF07.1	Título e subtítulo	Página com título "Inserir medicamento no CH:", a preto, no canto lateral esquerdo da página. - Página com subtítulos "Inserir novo medicamento:" e "Inserir Medicamento existente:", a cinzento.
RF07.2	Formulário relativo ao primeiro subtítulo	Imediatamente em baixo do segundo subtítulo referido em RF07.1. - 5 caixas de preenchimento obrigatório alinhadas horizontalmente.

<b>ID RF07.X</b>	<b>Nome do Requisito</b>	<b>Descrição do Requisito</b>
RF07.3	Tabela relativa ao segundo subtítulo	Imediatamente em baixo do segundo subtítulo referido em RF07.1. - Tabela com paginação. - Tabela com campos "#", referente ao ID de cada medicamento, "DCI", referente ao princípio ativo de cada medicamento, "Dosagem", referente à dosagem que compõe cada medicamento, "Forma", referente à forma em que cada medicamento está composto, e "Data de validade", referente à data de validade de cada medicamento. - Cada linha da coluna deve ter 1 botão no final, para adicionar esse medicamento à base de dados de medicamentos locais, adicionando de seguida a quantidade.

Tabela 4.11: Requisitos funcionais da página de inserir um medicamento.

<b>ID RF07.2.X</b>	<b>Nome do Requisito</b>	<b>Descrição do Requisito</b>
RF07.2.1	DCI	Menu <i>dropdown</i> , com o título "DCI do Medicamento", com uma listagem de todos os DCIs existentes.
RF07.2.2	Dosagem	Menu <i>dropdown</i> , com o título "Dosagem do Medicamento", com uma listagem de todas as dosagens existentes.
RF07.2.3	Forma	Menu <i>dropdown</i> , com o título "Forma do Medicamento", com uma listagem de todos as formas de medicamento existentes.
RF07.2.4	Data de Validade	Campo de preenchimento obrigatório. A data tem que ser válida. Se a data for inválida, aparece uma mensagem de erro a dizer "Introduza um valor válido. O campo está incompleto ou tem uma data inválida.".

<b>ID RF07.2.X</b>	<b>Nome do Re- quisito</b>	<b>Descrição do Requisito</b>
RF07.2.5	Quantidade	Campo de preenchimento obrigatório. Se este campo não for preenchido, aparece uma mensagem de erro, no topo do formulário a vermelho, a dizer "Por favor, insira a quantidade!".
RF07.2.6	Botão	Botão a azul, com o título "Inserir", centrado com o formulário, e imediatamente na linha a baixo do requisito RF07.2.5.

Tabela 4.12: Requisitos funcionais do formulário da página de inserir um medicamento.

<b>ID RF08.X</b>	<b>Nome do Re- quisito</b>	<b>Descrição do Requisito</b>
RF08.1	Título e subtí- tulo	Página com título "Lista Local de Medicamen- tos:", a preto, no canto lateral esquerdo da pá- gina.
RF08.2	Tabela	Tabela com paginação. - Tabela com cam- pos "#", referente ao ID de cada medicamento existente no centro hospitalar do utilizador em questão, "DCI", referente ao DCI de cada me- dicamento, "Dosagem", referente à dosagem de cada medicamento, "Forma", referente à forma em que cada medicamento está com- posto, "Data de validade", referente à data de validade de cada medicamento e "Quantidade", referente à quantidade que existe de um medica- mento no centro hospitalar em questão. - Cada linha da coluna deve ter 1 botão no final, para editar a quantidade desse medicamento no cen- tro hospitalar.

Tabela 4.13: Requisitos funcionais da página de listar os medicamentos locais.

<b>ID RF09.X</b>	<b>Nome do Requisito</b>	<b>Descrição do Requisito</b>
RF09.1	Título	Página com título "Lista Global de Medicamentos:", a preto, no canto lateral esquerdo da página.
RF09.2	Tabela	Tabela com paginação. - Tabela com campos "#", referente ao ID de cada medicamento, "DCI", referente ao DCI de cada medicamento, "Dosagem", referente à dosagem de cada medicamento, "Forma", referente à forma em que cada medicamento está composto, "Data de validade", referente à data de validade de cada medicamento.

Tabela 4.14: Requisitos funcionais da página de listar os medicamentos globais.

<b>ID RF10.X</b>	<b>Nome do Requisito</b>	<b>Descrição do Requisito</b>
RF10.1	Título e subtítulo	Página com título "Fazer um pedido:", a preto, no canto lateral esquerdo da página. - Página com subtítulo Inserir novo pedido:"imediatamente em baixo do título, a cinzento.
RF10.2	Formulário	Formulário em forma de bloco. - Centrado na página. - Formulário com barra de progresso animada, a azul e 2 secções. - Primeira secção denominada de "1. Pedido" e segunda secção denominada de "2. Medicamento".
RF10.3	Botão	Botão centrado horizontalmente na página, no fim do formulário. - Botão de cor azul. - O título do botão deve ser "Submit".

Tabela 4.15: Requisitos funcionais da página de inserir um novo pedido.

<b>ID RF10.2.X</b>	<b>Nome do Re- quisito</b>	<b>Descrição do Requisito</b>
RF10.2.1	Secção de pe- dido	Deve ter 2 menus <i>dropdown</i> alinhados hori- zontalmente. - O primeiro menu, deve ter o título "Centro Hospitalar de destino:" e lista todos os centros hospitalares registados na base de da- dos, à exceção do centro hospitalar do próprio utilizador. - O segundo menu, deve ter o título "Estado de urgência" e lista todos os estados de urgência que um pedido pode ter. - A barra de progresso desta secção deve estar a metade do tamanho (50%).

<b>ID RF10.2.X</b>	<b>Nome do Re- quisito</b>	<b>Descrição do Requisito</b>
RF10.2.2	Secção de me- dicamento	Deve conter uma tabela, com os campos # (ID de cada medicamento), DCI (DCI de cada me- dicamento), Dosagem (Dosagem de cada me- dicamento) e Forma (Forma de cada medica- mento). - Imediatamente em baixo da tabela deve conter 2 campos de preenchimento. - O primeiro campo deve ter o nome "ID do medi- camento:", onde o utilizador insere o ID do me- dicamento que quer pedir. - O segundo campo deve ter o nome "Quantidade:", onde o utiliza- dor insere a quantidade do medicamento que pretende pedir. - A barra de progresso desta secção deve ocupar o máximo do seu tamanho (100%).

Tabela 4.16: Requisitos funcionais do formulário da página de inserir um novo pedido.

<b>ID RF11.X</b>	<b>Nome do Requisito</b>	<b>Descrição do Requisito</b>
RF11.1	Título	Página com título "Lista de Pedidos Enviados:", a preto, no canto lateral esquerdo da página.
RF11.2	Tabela	Tabela com paginação. - Tabela com campos "#", referente ao ID de cada pedido, "Efetuado Por", referente ao utilizador que fez um determinado pedido, "Centro Hospitalar", referente ao centro hospitalar para o qual foi feito o pedido, "Urgência", referente à urgência do pedido (especificado em RF11.2.X), "Estado", referente ao estado de cada pedido (especificado em RF11.2.X), e "Confirmar receção"(especificado em RF11.2.X). - Cada linha da coluna deve ter 1 botão no final com o título "Ver mais...", que mostra mais detalhes do pedido.

Tabela 4.17: Requisitos funcionais da página de listar os pedidos enviados.

<b>ID RF11.2.X</b>	<b>Nome do Requisito</b>	<b>Descrição do Requisito</b>
RF11.2.1	Urgência	Devem existir 3 estados de urgência, sendo estes "Pouco urgente", "Urgente"e "Muito Urgente".
RF11.2.2	Estado	Devem existir 4 estados, sendo estes: "Chegada ao destino", "Em distribuição", "Recebido"e "Rejeitado". - Se o estado for "Chegada ao destino", então este estado está inserido num <i>span</i> verde. - Se o estado for "Em distribuição", então este estado está inserido num <i>span</i> amarelo. - Se o estado for "Recebido", então este estado está inserido num <i>span</i> azul. - Se o estado for "Rejeitado", então este estado está inserido num <i>span</i> vermelho.
RF11.2.3	Confirmar receção	Botão que confirma a chegada do pedido. - Se um pedido já tiver sido recebido ou tiver sido rejeitado, este botão fica inativo.

Tabela 4.18: Requisitos funcionais de alguns elementos da tabela de pedidos enviados.

ID RF12.X	Nome do Requisito	Descrição do Requisito
RF12.1	Título	Página com título "Lista de Pedidos Recebidos:", a preto, no canto lateral esquerdo da página.
RF12.2	Tabela	Tabela com paginação. - Tabela com campos "#", referente ao ID de cada pedido, "Recebido de", referente ao utilizador que fez um determinado pedido, "Centro Hospitalar", referente ao centro hospitalar do qual foi feito o pedido, "Urgência", referente à urgência do pedido (especificado em RF11.2.X), "Estado", referente ao estado de cada pedido (especificado em RF12.2.X), "Confirmar envio"(especificado em RF12.2.X), e "Rejeitar"(especificado em RF12.2.X). - Cada linha da coluna deve ter 1 botão no final com o título "Ver mais...", que mostra mais detalhes do pedido.

Tabela 4.19: Requisitos funcionais da página de listar os pedidos recebidos.

ID RF12.2.X	Nome do Requisito	Descrição do Requisito
RF12.2.1	Confirmar envio	Botão que confirma o envio do pedido. - Se um pedido tiver um estado diferente de "Recebido", este botão fica inativo.
RF12.2.2	Rejeitar	Botão que rejeita um pedido. - Se um pedido tiver um estado diferente de "Recebido", este botão fica inativo.

Tabela 4.20: Requisitos funcionais de alguns elementos da tabela de pedidos recebidos.

<b>ID RF13.X</b>	<b>Nome do Requisito</b>	<b>Descrição do Requisito</b>
RF13.1	Título	Página com título "Histórico de Medicamentos:", a preto, no canto lateral esquerdo da página.
RF13.2	Campo de preenchimento	Deve ter como título "Insira o DCI do medicamento: ". - Se o DCI preenchido pelo utilizado não existir, a tabela referente a RF14.X aparece vazia. - O utilizador pode escrever só parte do DCI (se o DCI for "Ácido Fólico", o utilizador pode apenas escrever "ácido"). - Botão de pesquisa imediatamente depois do campo de preenchimento, com o ícon de uma lupa.
RF13.3	Tabela	Tabela com paginação. - Tabela com campos "#", referente ao ID de cada medicamento no centro hospitalar em questão, "DCI", referente ao DCI de um determinado medicamento, "Dosagem", referente à dosagem de cada medicamento, "Forma", referente à forma em que cada medicamento está composto, "Data de validade", referente à data de validade de cada medicamento e "Quantidade", referente à quantidade que existe de um medicamento no centro hospitalar em questão.

Tabela 4.21: Requisitos funcionais da página de histórico de medicamentos.

<b>ID RF14.X</b>	<b>Nome do Requisito</b>	<b>Descrição do Requisito</b>
RF14.1	Título	Página com título "Histórico de Medicamentos:", a preto, no canto lateral esquerdo da página.
RF14.2	Tabela	Tabela com paginação. - Tabela com campos "#", referente ao ID do medicamento, "DCI", referente ao DCI do medicamento em questão, "Quantidade", referente à quantidade que existe do medicamento selecionado no centro hospitalar correspondente e "Atualização", referente ao dia e hora de uma determinada atualização ao medicamento.
RF14.3	Botão	Botão imediatamente em baixo da tabela referenciada em RF14.2. - O botão deve ser cinzento. - Deve ter um ícon que remete a "voltar para a página anterior". - Deve ter como texto "Voltar".

Tabela 4.22: Requisitos funcionais da página de histórico de um medicamento específico.

<b>ID RF15.X</b>	<b>Nome do Requisito</b>	<b>Descrição do Requisito</b>
RF15.1	Título	Página com título "Lista de contactos:", a preto, no canto lateral esquerdo da página.
RF15.2	Campo de preenchimento	Deve ter como título "Insira o nome do Centro Hospitalar: ". - Se o nome preenchido pelo utilizado não existir, a tabela referente a RF16.X aparece vazia. - O utilizador pode escrever só parte do nome (se o nome for "Pêro da Covilhã", o utilizador pode apenas escrever "pêro"). - Botão de pesquisa imediatamente depois do campo de preenchimento, com o ícone de uma lupa.

<b>ID RF15.X</b>	<b>Nome do Requisito</b>	<b>Descrição do Requisito</b>
RF15.3	Tabela	Tabela com paginação. - Tabela com campos "#", referente ao ID de cada centro hospitalar, "Nome", referente ao nome de um terminado centro hospitalar, "Email", referente ao <i>email</i> associado a cada centro hospitalar e "Telefone", referente ao contacto telefónico de cada centro hospitalar.

Tabela 4.23: Requisitos funcionais da página de contactos.

<b>ID RF16.X</b>	<b>Nome do Requisito</b>	<b>Descrição do Requisito</b>
RF16.1	Título	Página com título "Hospital x", sendo que x se refere ao nome do centro hospitalar escolhido. - As letras devem estar a preto, no canto lateral esquerdo da página.
RF16.2	Tabela	Tabela com paginação. - Tabela com campos "#", referente ao ID do centro hospitalar, "Nome", referente ao nome do centro hospitalar em questão, "Email", referente ao <i>email</i> associado a esse centro hospitalar e "Telefone", referente ao contacto telefónico deo centro hospitalar.
RF16.3	Botão	Botão imediatamente em baixo da tabela referenciada em RF16.2. - O botão deve ser cinzento. - Deve ter um ícone que remete a "voltar para a página anterior". - Deve ter como texto "Voltar".

Tabela 4.24: Requisitos funcionais da página de contactos de um centro hospitalar específico.

ID RF17.X	Nome do Requisito	Descrição do Requisito
RF17.1	Título e subtítulo	Deve ter o título "Definições", a preto, no canto lateral esquerdo da página. - Deve ter subtítulo "Alterar dados:", a cinzento, imediatamente abaixo do título.
RF17.2	Palavra.passe atual	O preenchimento deste campo é obrigatório. A <i>password</i> deve concidir com a <i>password</i> atual do utilizador.
RF17.3	Nova palavra-passe	O preenchimento deste campo é obrigatório. - O limite mínimo é de 6 caracteres, não tendo limite máximo. Se o limite mínimo de caracteres não for cumprido, deve mostrar a mensagem "A <i>nova password</i> deve ter pelo menos 6 caracteres". - Abrange codificação alfanumérica, podendo conter letras capitalizadas ou não. - É permitido o uso de caracteres especiais. - Deve haver um campo de confirmação da palavra-passe. - Ao enviar o formulário, ambos os campos (palavra-passe e confirmação desta) devem ser testados, coincidindo. - Caso as palavras-pases inseridas sejam diferentes, então deverá aparecer uma mensagem de erro com a mensagem "As novas <i>passwords</i> não coincidem".
RF17.4	Nome de utilizador	O preenchimento deste campo é obrigatório. - Se este campo for deixado em branco aparece uma mensagem a dizer "Preencha este campo". - Abrange codificação alfanumérica, podendo conter letras capitalizadas ou não.
RF17.5	Email	O preenchimento deste campo é obrigatório. - Caso o utilizador insira um <i>email</i> inválido, aparece uma mensagem de erro a dizer "O email deve ser válido.".
RF17.6	Botão	Deve estar imediatamente a baixo do requisito referenciado em RF17.5. - Deve ter a cor azul. - Deve conter a palavra "Guardar".

Tabela 4.25: Requisitos funcionais das definições de conta.

### 4.2.2 Requisitos não funcionais

Este tipo de requisitos são responsáveis por informar acerca do desempenho, usabilidade, confiabilidade, segurança, disponibilidade, manutenção e tecnologias envolvidas no sistema.

Tratam ainda dos fatores externos que afetam o sistema e processo de desenvolvimento.

A seguinte tabela descreve alguns requisitos não funcionais encontrados:

ID Requisito	Nome do Requisito	Descrição do Requisito
RNF01	Horário de funcionamento	A plataforma deve estar disponível todos os dias e a todas as horas, exceto quando está em períodos de manutenção, períodos estes que devem ser curtos.
RNF02	Tempo de pesquisa	A aplicação não deve demorar mais que 2 segundos para efetuar uma pesquisa na base de dados.
RNF03	Disponibilidade	Deve poder ser utilizado em computador, <i>tablet</i> ou <i>smartphone</i> .
RNF04	Usabilidade	O sistema é simples e intuitivo. - Não é necessária formação para a sua utilização. - O utilizador deverá conseguir utilizar o <i>software</i> após uma breve explicação do funcionamento do mesmo.
RNF05	Dispositivo	Necessária uma ligação à Internet, aquando o alojamento da plataforma <i>online</i> .
RNF06	Sistema Operativo	O sistema operativo deve ser indiferente.
RNF07	<i>Browser</i>	Para aceder à plataforma é necessário um <i>browser</i> moderno, como: Microsoft Internet Explorer versão 11 ou superior; Apple Safari versão 9 ou superior; Google Chrome versão 46 ou superior; Firefox versão 45 ou superior.

Tabela 4.26: Requisitos Não Funcionais da plataforma.

## 4.3 Casos de uso

Nesta subsecção irão ser apresentados os diagramas de casos de uso considerados durante o desenvolvimento da plataforma.

Os diagramas de casos de uso representam uma possível utilização do sistema por um ator, que pode ser uma pessoa, dispositivo físico, mecanismo ou subsistema que interage com o sistema alvo, utilizando algum dos seus serviços.

Estes diagramas vão demonstrar a interação do sistema com os utilizadores, tanto administradores como farmacêuticos.

No diagrama representado na figura 4.1, podemos observar os casos de uso relativos aos utilizadores do tipo farmacêutico.

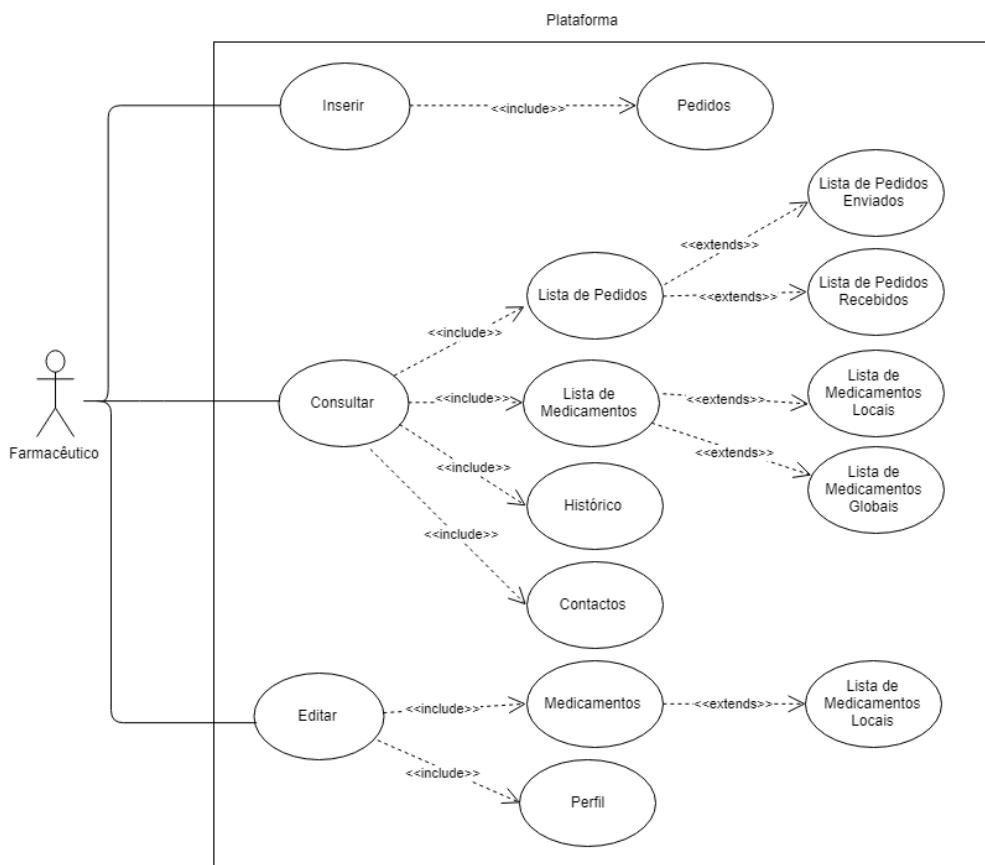


Figura 4.1: Diagrama de casos de uso do farmacêutico.

Neste diagrama, podemos observar a interação que é possível os farmacêuticos terem com a plataforma MedGest. Esta interação passa pela possibilidade de inserir um novo pedido, consultar a lista de todos os pedidos (tanto enviados como recebidos), consultar a lista de medicamentos locais e globais, consultar

o histórico de medicamentos e a lista de contactos. É também possível editar a quantidade de cada medicamento existente no centro hospitalar do farmacêutico e editar os dados do perfil.

Por outro lado, no diagrama representado na figura 4.2, podemos observar os casos de uso relativos aos utilizadores do tipo administrador.

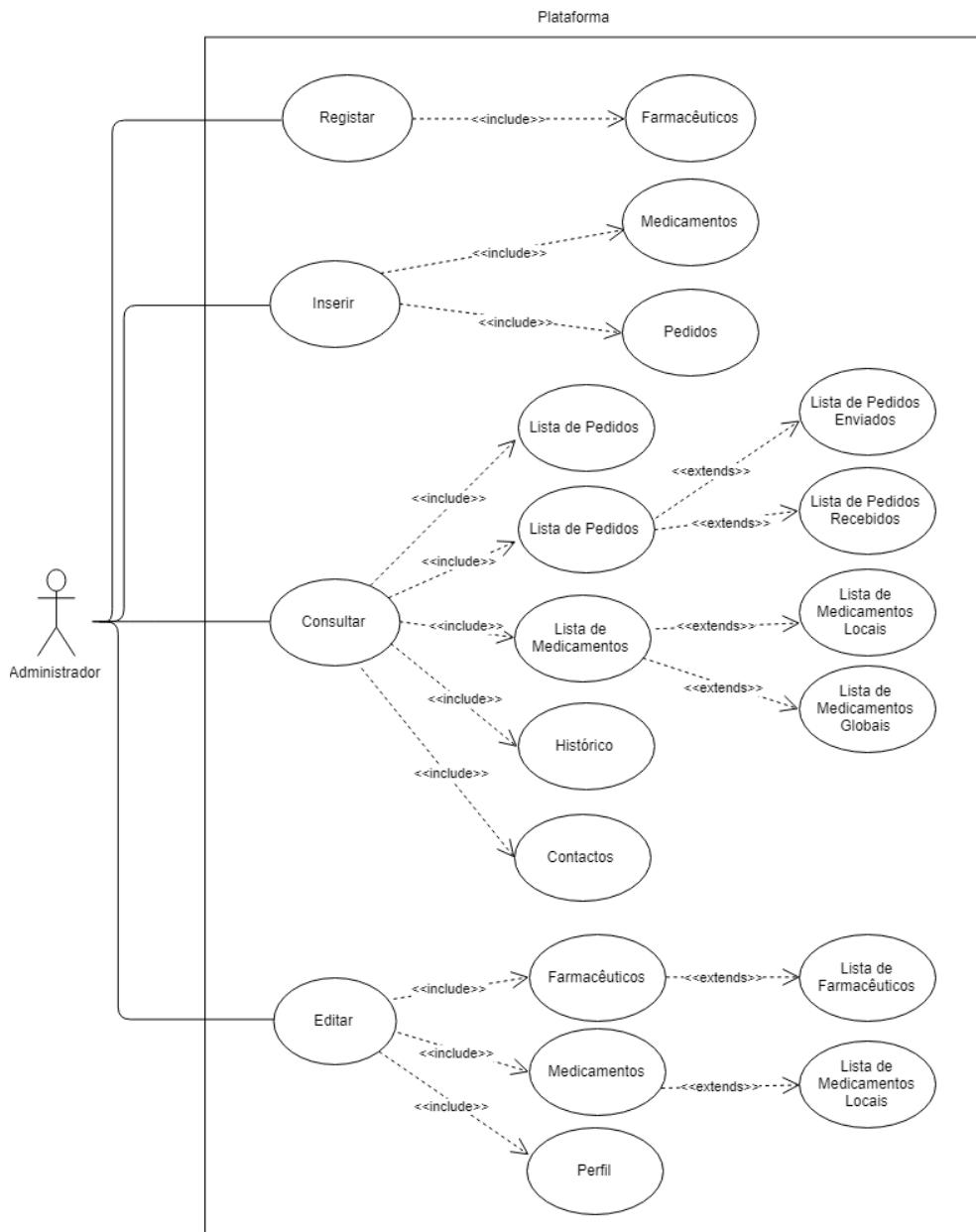


Figura 4.2: Diagrama de casos de uso do administrador.

Neste diagrama, podemos observar a interação que é possível os administradores terem com a plataforma MedGest. Esta interação é idêntica à dos farmacêuticos, tendo apenas algumas funcionalidades extra, como como o registo, edição e remoção de contas de farmacêuticos, inserção de novos medicamentos, consulta da lista de contas de todos os farmacêuticos e edição dos campos "nome", "email" e "centro hospitalar", referente às contas dos farmacêuticos.

## 4.4 Padrão de desenho arquitectural

Existem diversas formas de estruturar o código de um projeto, sendo que este processo pode-se tornar mais ou menos trabalhoso, dependendo da sua arquitetura. Geralmente, é sempre melhor optar por seguir padrões comuns, pois isso será tornar a manutenção do código mais fácil e rápida, facilitando também a compreensão deste por outros eventuais *developers*.

### 4.4.1 Padrão MVC

Este foi o padrão arquitectural utilizado na construção da plataforma, estando este diretamente implícito na *framework* utilizada.

Este padrão permite a separação do código em diferentes objetos lógicos, que servem para tarefas bastantes específicas. Os **Modelos** (*Models*) são utilizados como uma camada de acesso aos dados, onde estes são requisitados e retornados em formatos que possam ser utilizados no decorrer da aplicação. Os **Controladores** (*Controllers*) tratam das requisições, processam os dados retornados dos Modelos e carregam as **Visões** (*Views*) para enviar uma resposta. As Visões são *templates* de saída (construídos através de linguagem de marcação, como HTML, XML, etc.), que são enviados como resposta ao navegador, com o qual o utilizador interage diretamente.

Cada Modelo está assim associado a uma entidade representada por uma tabela na base de dados, sendo que o nome dos modelos implementados na plataforma foram: Admin, DCI, Dosagem, Estado, EstadoPedido, EstadoUser, Forma, Histórico, Hospital, Medicamento, MedicamentoPorCH, Pedido, PedidoLinha, Tipo e User.

Existem também diversos Controladores que trabalham com os Modelos em cima descritos, e que são solicitados a partir de diversas **rotas** existentes. Cada Controlador pode estar associado a uma ou mais rotas, consoante as requisições HTTP que nele são necessárias. Os Controladores referentes aos processos de autenticação, e que são criados por defeito na *framework* aquando a indicação da utilização de mecanismos autenticação são: AdminLoginController, ForgotPasswordCon-

troller, LoginController, RegisterController, ResetPasswordController, VerificationController. Os restantes Controladores que tratam do funcionamento de toda a plataforma internamente, resumem-se em: AdminControlador, ContaControlador, Controller, HomeController, MedicamentoControlador, MovimentoControlador e PedidoControlador.

Existem cerca de 50 Visões diferentes, que podem ser utilizadas e apresentadas ao utilizador. Algumas destas são referentes a processos de autenticação, como a reposição de palavra-passe, os mecanismos de *login* de ambos os tipos de utilizador, bem como o registo. Existe também uma secção exclusivamente dedicada aos *layouts* e *master pages*, como foi explicado no capítulo 3, secção 3.6. As restantes Views são referentes às páginas da plataforma internamente, após os processos de autenticação, como são exemplos as Visões: admin.blade.php, admin\_contactos.blade.php, admin\_settings.blade.php, movimentos.blade.php, pedidosenviados.blade.php, home.blade.php, entre muitos outros.

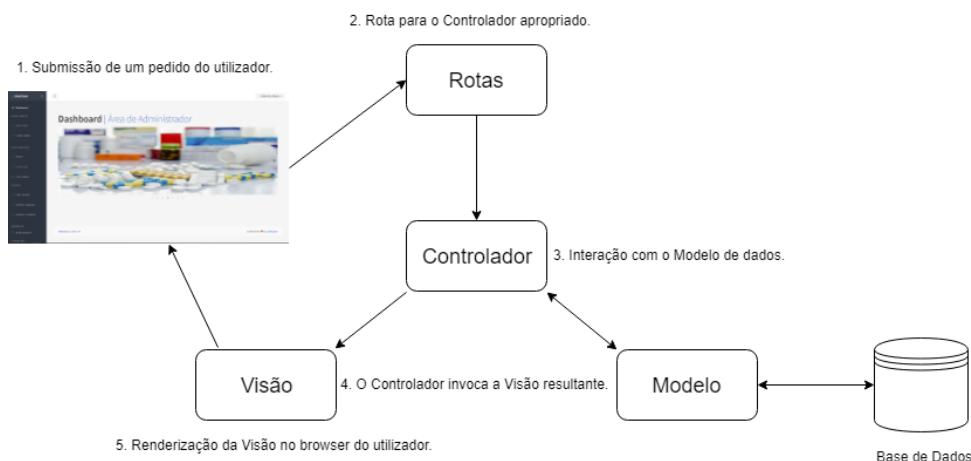


Figura 4.3: Padrão de desenho arquitectural da plataforma.

## 4.5 Conclusões

Tendo em conta todos os requisitos e casos de uso encontrados, é possível idealizar o funcionamento da plataforma de forma eficiente, eficaz e segura.

O padrão de desenho arquitectural utilizado, deu a entender a forma como a informação e o código se encontram tipicamente estruturados numa arquitetura MVC, e como estes componentes interagem entre si, descrevendo o processo por trás de uma requisição do utilizador na interface.

O capítulo seguinte irá abordar a forma de implementação da plataforma, tanto a nível do utilizador, como ao nível do servidor.

# Capítulo 5

## Implementação e Testes

### 5.1 Introdução

Este capítulo vai explicar detalhadamente a forma de implementação que permitiu o desenvolvimento da plataforma. No capítulo 3 foram descritas todas as ferramentas que auxiliaram a implementação da aplicação *Web*.

Inicialmente vai ser apresentada a plataforma que me ensinou todos os conceitos básicos associados à *framework* que utilizei. Sem a ajuda desta plataforma de ensino, o processo de desenvolvimento da aplicação teria sido bastante mais longo e duro.

O funcionamento da plataforma será seguidamente apresentado, juntamente com algumas capturas de ecrã e pedaços de código relevantes.

Por fim, serão abordados alguns testes realizados após a implementação.

Desta forma, o capítulo encontra-se dividido da seguinte maneira:

- secção 5.2 - **Udemy** - apresenta a plataforma de ensino que auxiliou na aprendizagem de conceitos básicos sobre a *framework*;
- secção 5.3 - **Base de dados** - introdução à base de dados implementada na plataforma;
- secção 5.3.1 - **Diagrama Entidade-Associação (DEA)** - demonstração e explicação do modelo de dados utilizado;
- secção 5.4 - **Manual do utilizador** - descrição detalhada do funcionamento da plataforma, ao nível do utilizador;
- secção 5.4.1 -**Login** - forma de utilização do *login* na plataforma;
- secção 5.4.2 -**Dashboard** - forma de utilização da *dashboard* na plataforma;

- secção 5.4.3 -**Gestão de contas** - forma de utilização da secção da gestão de contas na plataforma;
- secção 5.4.4 -**Medicamentos** - forma de utilização da secção de medicamentos na plataforma;
- secção 5.4.5 -**Pedidos** - forma de utilização da secção de pedidos na plataforma;
- secção 5.4.6 -**Histórico de medicamentos** - forma de utilização do histórico de medicamentos na plataforma;
- secção 5.4.7 -**Lista de contactos** - forma de utilização da lista de contactos na plataforma;
- secção 5.4.8 -**Definições** - forma de utilização da secção de definições na plataforma;
- secção 5.4.9 -**Logout** - forma de utilização *logout* na plataforma;
- secção 5.5 - **Pedidos** - descreve o funcionamento dos Pedidos na plataforma ao nível do servidor, com pedaços de código relevantes;
- secção 5.5.1 - **Fazer um pedido** -demonstração e explicação do ato de fazer um pedido na plataforma;
- secção 5.5.2 - **Pedidos enviados** - demonstração e explicação do modo de funcionamento da listagem de pedidos enviados, e operações adjacentes;
- secção 5.5.3 - **Pedidos recebidos** - demonstração e explicação do modo de funcionamento da listagem de pedidos recebidos, e operações adjacentes;
- secção 5.6 - **Testes** - introdução aos testes realizados na plataforma;
- secção 5.6.1 - **PHPUnit** - descrição dos testes utilizados com a ferramenta em questão;
- secção 5.6.2 - **Chrome Dev Tools** - explicação das funcionalidades pertencentes ao conjunto de ferramentas em questão, e testes realizados com estas;
- secção 5.6.3 - **Testes manuais** - apresentação dos testes realizados manualmente, no decorrer da implementação da aplicação Web;
- secção 5.7 - **Conclusão** - apresenta as principais conclusões do capítulo.

## 5.2 Udemy

No incio deste projeto, e depois da escolha da *framework* mais adequada aos objetivos pretendidos, foi necessrio adquirir o mximo de conhecimento possvel acerca desta, para poder usufruir do mximo de recursos e ferramentas que esta disponibiliza. Neste sentido, decidi procurar um curso na Udemy, que me ajudasse a obter o conhecimento que necessitava para implementar a plataforma MedGest. A Udemy, é uma plataforma de ensino, que conecta alunos de qualquer lugar do mundo a vrios instrutores. A plataforma conta j com cerca de 30 milhnes de alunos, 100 mil cursos, 42 mil instrutores, 190 milhnes de inscrições em cursos, 22 milhnes de minutos de vdeos e vdeo-aulas em mais de 50 idiomas diferentes[15]. A Udemy é uma excelente plataforma para todas as pessoas que querem crescer profissionalmente e ganhar mais competências de trabalho.

Desta forma, encontrei na Udemy o curso ideal e que me ajudou a ganhar cada vez mais interesse e competências no domínio do Laravel, curso este denominado de "Laravel 5.6 Completo - O mais poderoso Framework PHP" e que contou com cerca de 28 horas de ensino. Embora este curso não seja gratuito, valeu bastante a pena o preço, pois abordou uma imensidão de assuntos, como por exemplo o Eloquent ORM, relacionamentos, integração com Angular, autenticação, filas de emails, cache com Redis, APIs, etc. Na conclusão do curso em questão, o aluno recebe um certificado de conclusão e fica com uma série competências, entre as quais:

- Criação de aplicações complexas com agilidade;
- Experiência de trabalho com arquiteturas complexas orientadas a objetos;
- Construção rápida de interfaces de acesso à base de dados;
- Entendimento completo da divisão da aplicação em modelos, visões e controladores (arquitetura MVC);
- Domínio dos principais recursos do Laravel.

Todo o curso é muito bom e foi muito bem explicado, tendo apenas como requisitos o conhecimento básico da linguagem PHP e HTML.

O instrutor é um engenheiro de *software* pertencente ao grupo "MPro Consultoria, Desenvolvimento e Treinamento", grupo este que atua em diferentes setores da computação, em especial no desenvolvimento de sistemas para a *Web*, aplicações para dispositivos móveis e sistemas embutidos de alto desempenho. Todos os seus profissionais atuam no mercado de trabalho e também em universidades, possuindo um alto nível de conhecimento sobre as tecnologias utilizadas atualmente [9].

## 5.3 Base de Dados

A base de dados foi o elemento fundamental no desenvolvimento e implementação da plataforma, tendo sido refletida a sua importância em qualquer projeto no capítulo 2 na secção 2.5.

É nesta ferramenta que se encontra toda a informação que permite aos utilizadores desempenhar quase todas as ações na plataforma.

O desenvolvimento desta estrutura de armazenamento foi um processo contínuo e demorado, tendo esta sido alterada algumas vezes aquando a implementação da aplicação *Web*. Foram desenhados, alterados e reinventados diversos modelos de dados, de forma a tornar a base de dados o mais eficiente e realista possível, tendo também em atenção futuros aspectos de manutenção desta.

### 5.3.1 DEA

Um DEA mostra a estrutura lógica de uma BD, incluindo as relações e restrições que determinam como os dados devem ser guardados e acedidos.

Num DEA, pode-se definir o conceito "Entidade" como um objeto do mundo real, sendo esta descrita por um conjunto de atributos. Cada atributo tem associado um domínio de valores possíveis. Sendo assim, um conjunto de entidades pode-se definir como uma coleção de entidades semelhantes, que podem partilhar atributos e associações. Uma chave é o conjunto mínimo de atributos cujos valores identificam univocamente cada entidade do conjunto.

Por outro lado, o conceito de "Associação" define-se como um relacionamento entre duas ou mais entidades, sendo um conjunto de associações, uma coleção de associações semelhantes. A associação pode ter atributos descritivos, onde existe informação sobre essa associação. Podemos chamar de "instância" um conjunto de associações. [21] Algumas vantagens deste modelo são o facto de revelar incoerências atuais, tem uma construção baseada "*Top-Down*" e a facilidade de comunicação que se cria entre informáticos e utilizadores.

O seguinte esquema, descreve o modelo de dados final que foi implementado na plataforma.

As tabelas "Historico", "PasswordResets" e "Migrations" são tabelas auxiliares, daí não estarem ligadas a nenhuma das outras.

A tabela "Historico" armazena o histórico da quantidade de medicamentos em cada centro hospitalar. Esta não possui nenhuma *Foreign Key* (FK), pois guarda simplesmente cópias dos dados do medicamento e hospital com base noutras transações.

A tabela "PasswordResets" armazena o *token* gerado da *password* do utilizador, juntamente com a data em que ocorreu a reposição da palavra-passe. Esta tabela

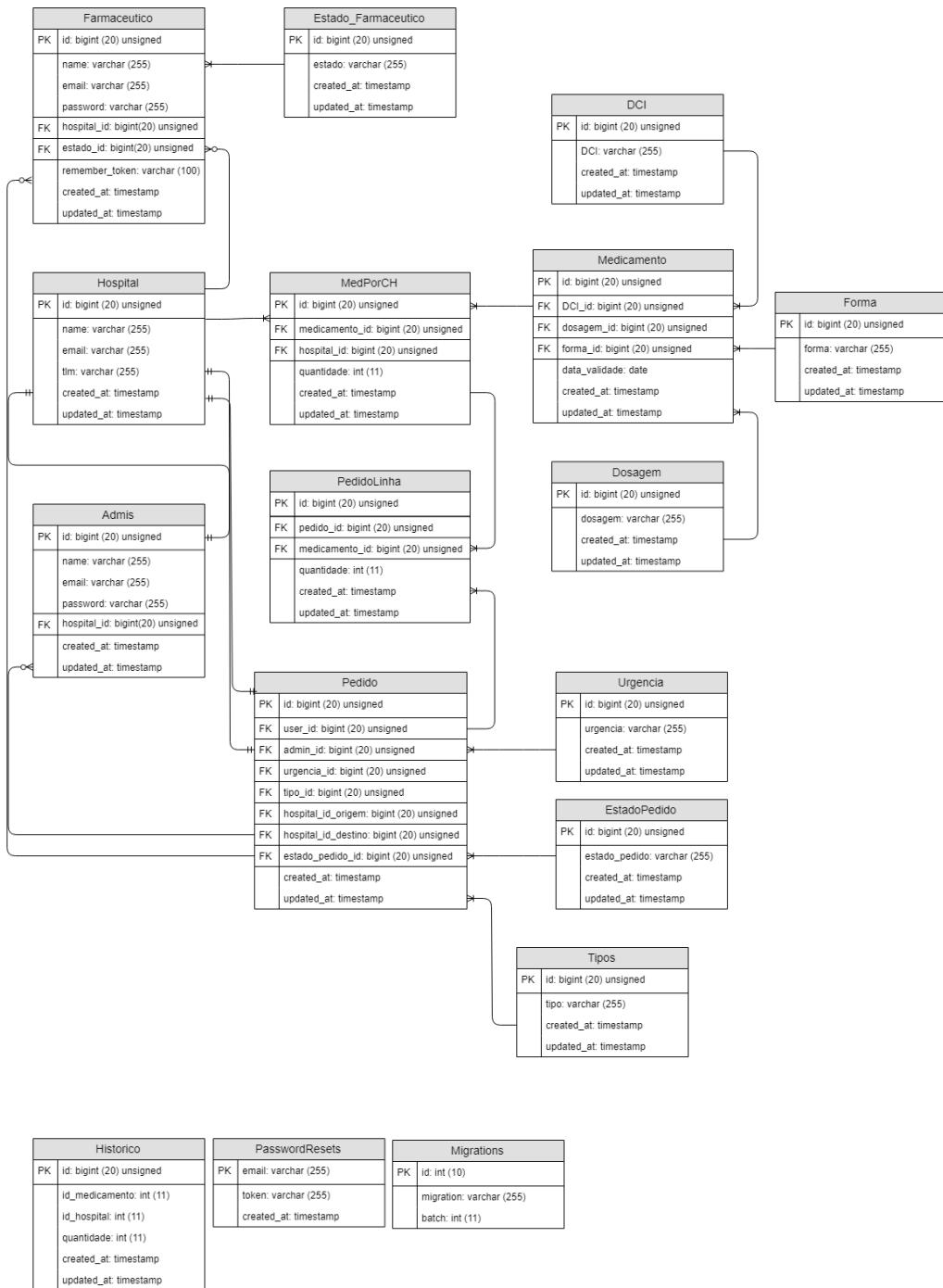


Figura 5.1: DEA implementado na base de dados.

não está a ser utilizada, embora tudo esteja preparado para a sua utilização futura-

mente.

A tabela "Migrations" armazena todas as migrações que foram efetuadas para a base de dados.

## 5.4 Manual do utilizador

Nesta secção irá ser demonstrado o funcionamento da plataforma do ponto de vista do utilizador. O funcionamento é bastante simples e intuitivo, não existindo grande dificuldade para um utilizador habituado a usar um computador.

Irão ser abordados os procedimentos de *login* de ambos os tipos de utilizador, como é feita a gestão de contas e inserção de novos medicamentos no caso do administrador, a forma de listar todos os medicamentos tanto locais como globais, a forma de fazer, aceitar ou rejeitar um pedido, o modo de funcionamento do histórico de medicamentos e a lista de contactos. Por fim, já também será referido o modo de alteração de dados da conta de ambos os utilizadores.

### 5.4.1 Login

Para um farmacêutico, basta inserir as credenciais de acesso (*email* e *password*) fornecidos pelo administrador do centro hospitalar em questão, nos devidos campos, como é possível observar na figura 5.2

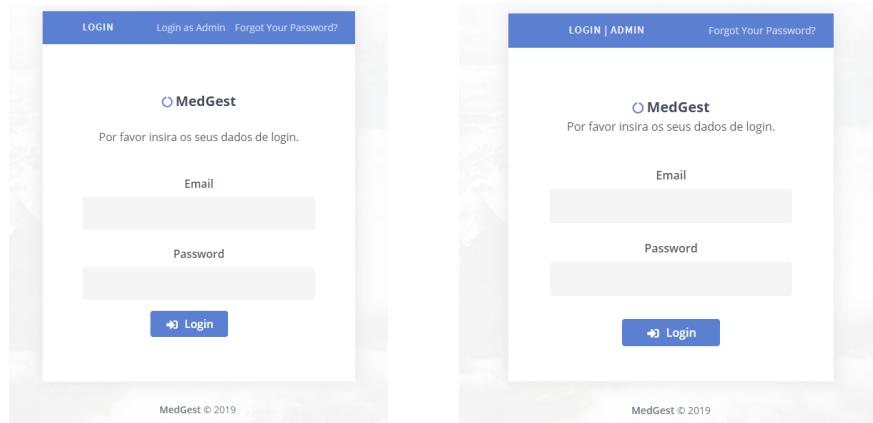
Para o administrador o processo é semelhante, com a exceção de que este necessita de ir para a página de *login* respetiva. Para isto, basta clicar no campo "Login as Admin", que se encontra no cabeçalho da página de *login* genérica. Assim que este tiver acedido a esta página aparecerá no cabeçalho do formulário "Login | Admin", invés de apenas "Login".

### 5.4.2 Dashboard

Após o *login*, o utilizador é redirecionado para a *dashboard* principal da plataforma. No caso do utilizador ser um farmacêutico, esta página principal tem como título "Dashboard | Área de Farmacêutico". Aqui é possível ver um menu lateral cheio de opções bem como um menu *dropdown* no canto superior direito, onde o utilizador pode aceder às suas definições de conta e terminar sessão.

No caso do utilizador o processo é idêntico, havendo no entanto mais opções no menu lateral e dizendo "Dashboard | Área de Administrador" na página principal, como é possível observar na figura 5.4.

Ainda nesta página existe a opção de mudar as cores do *layout* no canto superior esquerdo do menu lateral. Com isto é possível alterar o esquema de cores

Figura 5.2: Páginas de *login*.

MedGest © 2018-19

Rita ▾

Crafted with ❤ by pixelcave

Figura 5.3: Página principal do farmacêutico.

(figura 5.5) e mudar a cor da barra de navegação superior e do próprio lateral, como mostra a figura 5.6.

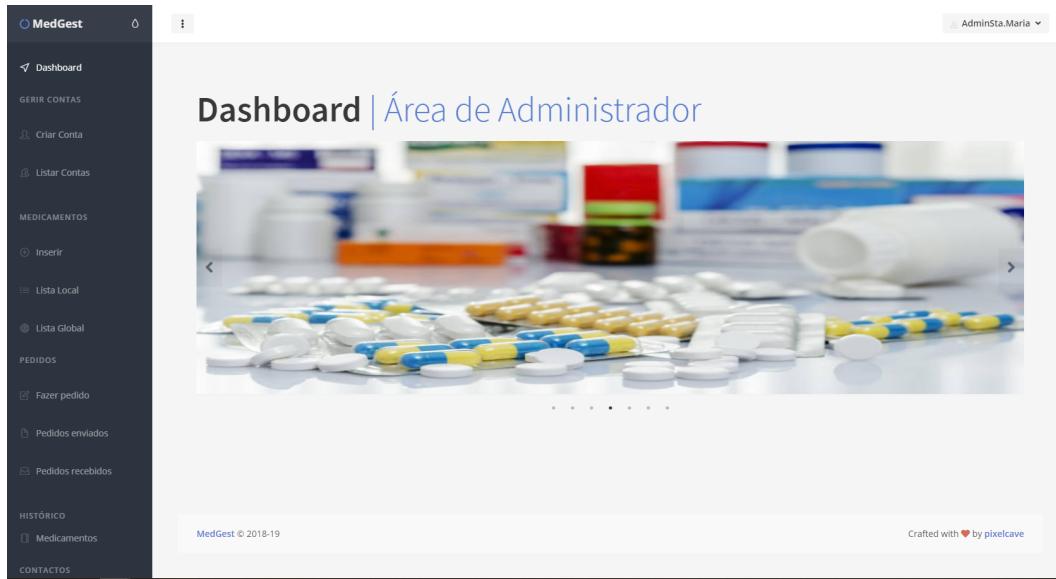


Figura 5.4: Página principal do administrador.



Figura 5.5: Alterar esquema de cores.

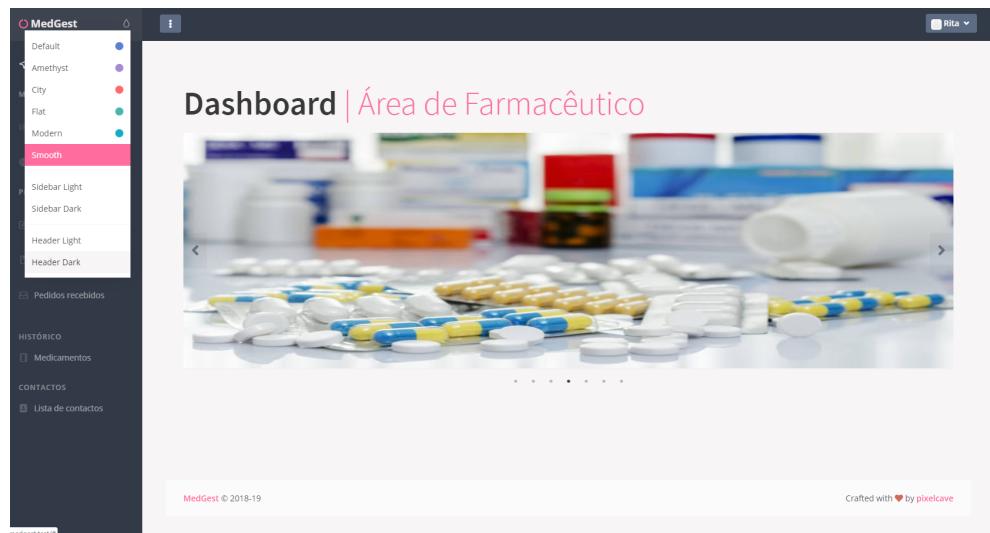


Figura 5.6: Alterar cor da barra de navegação e menu lateral.

### 5.4.3 Gestão de contas

Esta opção existe somente para utilizadores do tipo administrador.

Neste menu é possível registar um novo farmacêutico, preenchendo os campos respetivos ao nome, centro hospitalar, estado (sendo que este atributo pode ser: efetivo, auxiliar, pendente ou inativo), *email*, palavra-passe e confirmação desta. Este procedimento é visível na figura 5.7. A segunda opção deste menu é respetiva à listagem das contas dos utilizadores de um determinado centro hospitalar. É mostrada ao administrador uma tabela com os campos "Nome", "Email", "Estado" e "Data"(referente à data da ultima atualização dos dados de um farmacêutico). No final de cada linha da tabela existe um botão para editar os dados, como mostra a figura 5.8. Ao carregar neste botão, o administrador é diretamente redirecionado para um formulário de alteração de dados, visível na figura 5.9.

Figura 5.7: Registo de um farmacêutico.

	NOME	EMAIL	ESTADO	DATA
Rita	rita.correia7@gmail.com	Efetivo	2019-07-10 23:57:17	
Mariana	marianna@gmail.com	Auxiliar	2019-07-10 23:58:20	
Joaquim	joaquim@gmail.com	Inativo	2019-07-12 02:47:57	
Leonor	leonor@gmail.com	Pendente	2019-07-11 00:26:17	

Figura 5.8: Lista de contas.

The screenshot shows the MedGest application's user interface. On the left is a dark sidebar with white text and icons. It includes links for Dashboard, GERIR CONTAS (Create Account, List Accounts), MEDICAMENTOS (Insert, Local List, Global List), and PEDIDOS (Place Order). The main area is titled 'Editar conta:' (Edit account). It contains four input fields: 'Nome:' with 'Rita' typed in, 'Email:' with 'rita.correia7@gmail.com', 'Estado:' with 'Efetivo' selected from a dropdown, and 'Centro Hospitalar:' with 'Pêro da Covilhã' selected from another dropdown. At the bottom right is a blue 'Guardar' (Save) button.

Figura 5.9: Alteração de dados de uma conta.

#### 5.4.4 Medicamentos

##### Inserir um medicamento

Esta opção aplica-se somente a utilizadores do tipo administrador.

A inserção de um medicamento num medicamento num determinado centro hospitalar, pode-se fazer de duas maneiras.

Caso o administrador pretenda inserir um medicamento novo, que ainda não se encontra registado na base de dados geral, este deve recorrer ao formulário referente ao subtítulo *Inserir novo medicamento:*". Neste formulário terá de preencher todos os campos, sendo estes o DCI, a dosagem e a forma do medicamento, bem como a sua data de validade e quantidade existente no centro hospitalar em questão, como mostra a figura 5.10

Caso o medicamento que é pretendido adicionar ao centro hospitalar já exista na base de dados geral, o administrador deve recorrer à tabela referente ao subtítulo *Inserir Medicamento existente:"*. Aqui é mostrada uma tabela com informação de todos os medicamentos, e com um botão no fim de cada linha. Esse botão permite ao utilizador adicionar esse medicamento ao centro hospitalar em questão (figura 5.11). Ao carregar nesse botão, o utilizador é redirecionado para uma página onde insere assim a quantidade que pretende adicionar, como mostra a figura 5.12.

Não existe a opção de eliminar um medicamento de um centro hospitalar. Neste caso, quando um medicamento não tem stock ou é descontinuado, o administrador deve alterar a sua quantidade para zero, mantendo assim um registo de histórico do medicamento em questão.

Inserir medicamento no CH:

| Inserir novo medicamento:

DCI do Medicamento:  
Ácido Fólico

Dosagem do Medicamento:  
1mg

Forma do Medicamento:  
Comprimido de liberação prolongada

Data de validade:  
dd/mm/aaaa

Quantidade:  
Quantidade

Inserir

| Inserir Medicamento existente:

Exibindo 6 medicamentos de 6 (1 a 6)

Figura 5.10: Inserir um novo medicamento.

Inserir Medicamento existente:

Exibindo 6 medicamentos de 6 (1 a 6)

#	DCI	DOSAGEM	FORMA	DATA DE VALIDADE
1	Ácido Fólico	1mg	Comprimido de liberação prolongada	2022-06-22
2	Hidrocortisona	200mg	Creme	2022-03-06
3	Ibuprofeno	600mg	Comprimido revestido por película	2025-04-24
4	Paracetamol + Cloridrato de difenidramina	10mg	Comprimido para mastigar	2026-03-31
5	Triamcinolona	100mg	Comprimido orodispersível	2028-03-07
6	Dextoprofeno	400mg	Granulado efervescente	2024-02-04

MedGest © 2018-19

Crafted with ❤ by pixelcave

Figura 5.11: Inserir um medicamento existente.

## **Lista Local de Medicamentos**

Esta secção é permitida a ambos os utilizadores, administradores e farmacêuticos. Quando o clique no botão "Lista Local" no menu lateral da aplicação Web, é visível ao utilizador uma lista com todos os medicamentos existentes no centro hospitalar a que o utilizador pertence. Nesta lista é mostrado o ID, DCI, dosagem, forma, data de validade e quantidade do medicamento (figura 5.13). Existe tam-

The screenshot shows a dark-themed application window titled 'MedGest'. On the left, a sidebar menu includes 'Dashboard', 'GERIR CONTAS' (with 'Criar Conta' and 'Listar Contas' options), 'MEDICAMENTOS' (with 'Inserir' and 'Lista Local' options), and 'PEDIDOS' (with 'Fazer pedido' option). The main area displays a modal titled 'Quantidade do Medicamento:' with a text input field labeled 'Quantidade', a blue 'Inserir' button, and a red 'Cancelar' button.

Figura 5.12: Inserir quantidade do medicamento.

bém no final de cada linha da tabela, um botão que remete a uma página de edição manual de quantidade de um medicamento, como mostra a figura 5.14. O valor da quantidade no campo de edição refere-se ao valor atual desta, no medicamento escolhido.

The screenshot shows the 'Lista Local de Medicamentos:' page. The sidebar menu is identical to Figure 5.12. The main content area shows a table titled 'Exibindo 5 medicamentos de 5 (1 a 5)' with columns: #, DCI, DOSAGEM, FORMA, DATA DE VALIDADE, and QUANTIDADE. The data is as follows:

#	DCI	DOSAGEM	FORMA	DATA DE VALIDADE	QUANTIDADE
2	Hidrocortisona	200mg	Creme	2022-03-06	85
3	Ipobrufenol	600mg	Comprimido revestido por película	2025-04-24	200
4	Paracetamol + Cloridrato de difenidramina	10mg	Comprimido para mastigar	2026-03-31	30
5	Triamcinolona	100mg	Comprimido orodispersível	2028-03-07	40
6	Dexcetoprofeno	400mg	Granulado efervescente	2024-02-04	68

Figura 5.13: Lista Local de Medicamentos.



Figura 5.14: Editar quantidade de um medicamento.

### **Lista Global de Medicamentos**

Esta secção é permitida a ambos os utilizadores, administradores e farmacêuticos. Quando o clique no botão "Lista Global" no menu lateral da aplicação Web, é visível ao utilizador uma lista com todos os medicamentos existentes na base de dados geral.

Assim, o utilizador depara-se simplesmente com uma tabela informativa de todos os atributos pertencentes a cada medicamento, como o ID, DCI, dosagem, forma e data de validade, como mostra a figura 5.15.

The screenshot shows the 'MedGest' application with the 'Lista Global' option selected in the sidebar. The main area displays a table titled 'Lista Global de Medicamentos:' with 6 entries. The table columns are '#', 'DCI', 'DOSAGEM', 'FORMA', and 'DATA DE VALIDADE'. The data is as follows:

#	DCI	DOSAGEM	FORMA	DATA DE VALIDADE
1	Ácido Fólico	1mg	Comprimido de liberação prolongada	2022-06-22
2	Hidrocortisona	200mg	Creme	2022-03-06
3	Ipobrufen	600mg	Comprimido revestido por película	2025-04-24
4	Paracetamol + Cloridrato de difenidramina	10mg	Comprimido para mastigar	2026-03-31
5	Triamcinolona	100mg	Comprimido orodispersível	2028-03-07
6	Dexacetoprofeno	400mg	Granulado efervescente	2024-02-04

Figura 5.15: Lista Global de medicamentos.

### 5.4.5 Pedidos

Esta secção é permitida a ambos os utilizadores, administradores e farmacêuticos.

#### Fazer um pedido

Um centro hospitalar pode fazer um pedido de um determinado medicamento a um outro centro hospitalar. Esse pedido pode ser aceite ou rejeitado pelo centro hospitalar a que este se destina.

Quando um utilizador decide fazer um pedido, este depara-se imediatamente com um formulário de duas secções. Na primeira secção, o utilizador tem que selecionar o centro hospitalar a que se destina o pedido, juntamente com o estado de urgência deste (urgente, muito urgente ou pouco urgente), como mostra a figura 5.16. Na segunda secção, o utilizador, perante uma lista de medicamentos, deve escolher o ID do medicamento que pretende pedir, juntamente com a quantidade, como é possível observar na imagem 5.17.

Quando todos os dados do pedido estiverem preenchidos, é apenas necessário clicar em "Submit".

The screenshot shows a user interface for placing a medical request. On the left, a sidebar menu includes 'Listar Contas', 'MEDICAMENTOS' (with 'Inserir', 'Lista Local', 'Lista Global'), 'PEDIDOS' (with 'Fazer pedido' checked), 'Pedidos enviados', 'Pedidos recebidos', 'HISTÓRICO' (with 'Medicamentos'), 'CONTACTOS', and 'Lista de contactos'. The main area is titled 'Fazer um pedido:' with a sub-instruction 'Inserir novo pedido:'. It is divided into two tabs: '1. Pedido' and '2. Medicamento'. Under '1. Pedido', fields for 'Centro Hospitalar de destino:' (set to 'S. João') and 'Estado de urgência:' (set to 'Muito urgente') are shown. Under '2. Medicamento', a table lists several medications with their IDs and names:

ID	Nome
1	Paracetamol 500mg
2	Ibuprofeno 200mg
3	Aspirina 325mg
4	Cetamina 500mg
5	Dextrometorfano 30mg
6	Salbutamol 100mcg
7	Teriflunomida 10mg
8	Fluticasone 100mcg
9	Montelukast 10mg
10	Glucocorticoides 100mg

A blue 'Submit' button is located at the bottom of the form.

Figura 5.16: Fazer um pedido - secção 1.

#### Pedidos enviados

Existe uma lista com todos os pedidos enviados do centro hospitalar do utilizador, ordenados de forma decrescente.

#	DCI	DOSAGEM	FORMA
1	Ácido Fólico	1mg	Comprimido de liberação prolongada
2	Hidrocortisona	200mg	Creme
3	Ibuprofeno	600mg	Comprimido revestido por película
4	Paracetamol + Cloridrato de difenidramina	10mg	Comprimido para mastigar
5	Triamcinolona	100mg	Comprimido orodispersível
6	Dexacetoprofeno	400mg	Granulado efervescente

ID do medicamento:

Quantidade:

Figura 5.17: Fazer um pedido - secção 2.

Nesta tabela, encontra-se o ID do pedido, por quem este foi efetuado, para que centro hospitalar se destina, o estado de urgência e o estado do pedido, como mostra a figura 5.18.

Caso o pedido seja aceite e se encontre em distribuição para o centro hospitalar de origem, o estado dele é "Em distribuição", sendo que quando o hospital de origem confirmar a sua chegada, este estado passa a "Chegada ao destino". Enquanto não houver *feedback* de confirmação, o seu estado mantém-se como "Recebido". No caso de um pedido ser rejeitado pelo centro hospitalar de destino, o seu estado para imediatamente a "Rejeitado".

No final de cada linha existe também um botão de confirmação da receção de um medicamento, e um botão que diz "Ver mais...". É neste último botão que podemos ver os detalhes de um pedido, como o ID, o DCI, a dosagem, a forma, a data de validade e a quantidade do medicamento pedido, bem como a data (dia e hora) a que o pedido foi efetuado, como é possível observar na figura 5.19.

#	EFETUADO POR	CENTRO HOSPITALAR	URGÊNCIA	ESTADO	CONFIRMAR RECEÇÃO	
9	AdminCovilhã	Sta. Maria	Muito urgente	Em distribuição	<input checked="" type="checkbox"/>	<a href="#">Ver mais...</a>
7	AdminCovilhã	S.João	Urgente	Recebido	<input checked="" type="checkbox"/>	<a href="#">Ver mais...</a>
5	AdminCovilhã	Sta. Maria	Pouco urgente	Rejeitado	<input checked="" type="checkbox"/>	<a href="#">Ver mais...</a>
3	AdminCovilhã	S.João	Urgente	Chegada ao destino	<input checked="" type="checkbox"/>	<a href="#">Ver mais...</a>

Figura 5.18: Lista de pedidos enviados.

#	DCI	DOSAGEM	FORMA	DATA DE VALIDADE	QUANTIDADE	DATA/HORA
3	Ibuprofeno	600mg	Comprimido revestido por película	2025-04-24	60	2019-07-12 03:59:41

Figura 5.19: Detalhes de um pedido.

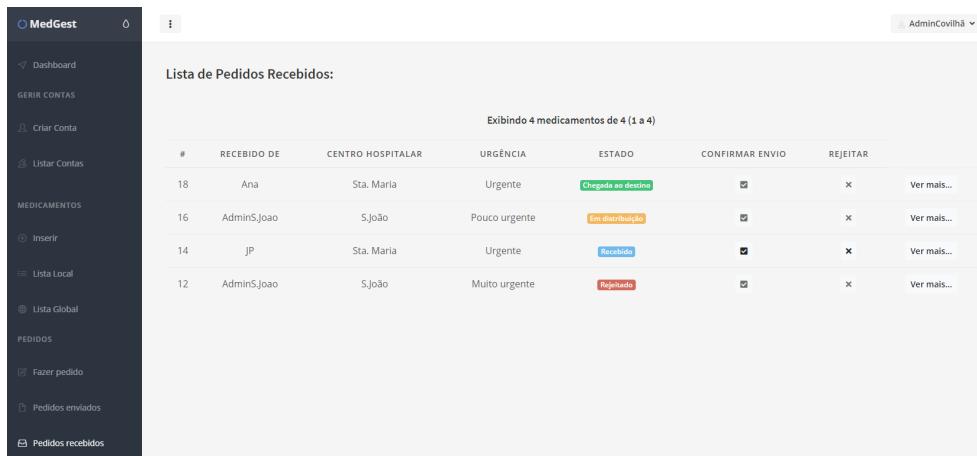
## Pedidos recebidos

Existe uma lista com todos os pedidos recebidos do centro hospitalar do utilizador, ordenados de forma decrescente.

Nesta tabela, encontra-se o ID do pedido, por quem este foi efetuado, o centro hospitalar de origem, o estado de urgência e o estado do pedido, como mostra a figura 5.20.

No final de cada linha existe também um botão de confirmação de envio de um medicamento, um botão de rejeição de um pedido, e um botão que diz "Ver mais...". É neste último botão que podemos ver os detalhes de um pedido, como o ID, o DCI, a dosagem, a forma, a data de validade e a quantidade do medicamento pedido, bem como a data (dia e hora) a que o pedido foi efetuado.

Quando um envio é confirmado, a opção de rejeição e confirmação fica desativada, bem como quando um envio é rejeitado.



#	RECEBIDO DE	CENTRO HOSPITALAR	URGÊNCIA	ESTADO	CONFIRMAR ENVIO	REJEITAR
18	Ana	Sta. Maria	Urgente	Chegada ao destino	<input checked="" type="checkbox"/>	X
16	AdminS.Joao	S.João	Pouco urgente	Em distribuição	<input checked="" type="checkbox"/>	X
14	JP	Sta. Maria	Urgente	Recebido	<input checked="" type="checkbox"/>	X
12	AdminS.Joao	S.João	Muito urgente	Rejeitado	<input checked="" type="checkbox"/>	X

Figura 5.20: Lista de pedidos recebidos.

## 5.4.6 Histórico de medicamentos

Esta secção é permitida a ambos os utilizadores, administradores e farmacêuticos. Nesta secção é apresentada ao utilizador uma lista de medicamentos locais, com informação do ID, DCI, dosagem, forma, data de validade e quantidade destes (figura 5.21). Com isto, o utilizador pode inserir o nome completo ou parcial de um DCI no campo de pesquisa, de forma a ver o historial do medicamento a ele associado, como mostra a figura 5.22.

The screenshot shows a user interface for managing medicine history. On the left is a sidebar with links for 'Lista Global', 'PEDIDOS' (Fazer pedido, Pedidos enviados, Pedidos recebidos), 'HISTÓRICO' (Medicamentos), and 'CONTACTOS' (Lista de contactos). The main area has a search bar ('Insira o DCI do Medicamento') and a table titled 'Exibindo 5 medicamentos de 5 (1 a 5)'. The table columns are '#', 'DCI', 'DOSAGEM', 'FORMA', 'DATA DE VALIDADE', and 'QUANTIDADE'. The data rows are:

#	DCI	DOSAGEM	FORMA	DATA DE VALIDADE	QUANTIDADE
2	Hidrocortisona	200mg	Creme	2022-03-06	100
3	Iopobrufen	600mg	Comprimido revestido por película	2025-04-24	200
4	Paracetamol + Cloridrato de difenidramina	10mg	Comprimido para mastigar	2026-03-31	25
5	Triamcinolona	100mg	Comprimido orodispersível	2028-03-07	65
6	Dexcetoprofeno	400mg	Granulado efervescente	2024-02-04	68

Figura 5.21: Página de histórico de medicamentos.

The screenshot shows a detailed view of a specific medicine's history. The title is 'Histórico de Medicamentos:' and it lists two entries for 'Paracetamol + Cloridrato de difenidramina'. The columns are '#', 'DCI', 'QUANTIDADE', and 'ATUALIZAÇÃO'. The data rows are:

#	DCI	QUANTIDADE	ATUALIZAÇÃO
4	Paracetamol + Cloridrato de difenidramina	25	2019-07-12 04:09:08
4	Paracetamol + Cloridrato de difenidramina	30	2019-07-11 00:48:25

A 'Voltar' (Back) button is at the bottom left.

Figura 5.22: Detalhes do histórico de um medicamento específico.

#### 5.4.7 Lista de contactos

Esta secção é permitida a ambos os utilizadores, administradores e farmacêuticos. Esta opção permite aos utilizadores consultarem informação sobre outros centros hospitalares registados na plataforma.

Deste modo, é visível para o utilizador uma tabela informativa, que contém o ID, o nome, o *email* e o contacto telefónico dos centros hospitalares, como mostra a figura 5.23. Existe também uma opção de pesquisa, onde o utilizador insere no campo respetivo o nome do hospital que pretende consultar. Quando o botão de pesquisa for clicado, o utilizador é redirecionado para uma nova página onde estão os dados do hospital pretendido, como está representado na figura 5.24.

#	NOME	EMAIL	TELEFONE
1	Pêro da Covilhã	info@chcbeira.min-saude.pt	275 330 000
2	S.João	geral@chsj.min-saude.pt	225 512 100
3	Sta. Maria	hsm@hsimporto.pt	225 082 000

Figura 5.23: Lista de contactos.

#	NOME	EMAIL	TELEFONE
2	S.João	geral@chsj.min-saude.pt	225 512 100

Figura 5.24: Detalhes de contacto de um centro hospitalar específico.

#### 5.4.8 Definições

Esta secção é permitida a ambos os utilizadores, administradores e farmacêuticos. Se um utilizador pretender alterar os dados da sua conta, deve-se dirigir à opção "Definições", que se encontra no menu *dropdown* na barra de navegação superior da plataforma, como mostra a alínea a) da figura 5.25.

Na página para a qual o utilizador é redirecionado após o clique nesta opção, este pode alterar as definições da sua conta, incluindo a palavra-passe, o nome de utilizador e o *email*, como é visível na alínea b) da figura 5.25.

Para a alteração da palavra-passe da conta, o utilizador tem que escrever corretamente a sua palavra-passe atual no campo com título "Palavra-passe atual". Após o preenchimento do campo referido, este deve prosseguir para a alteração da *password*. Para este efeito, é necessário colocar a nova palavra-passe no campo "Nova palavra-passe", repetindo-a de seguida no campo referente ao título "Confirmar nova palavra-passe".

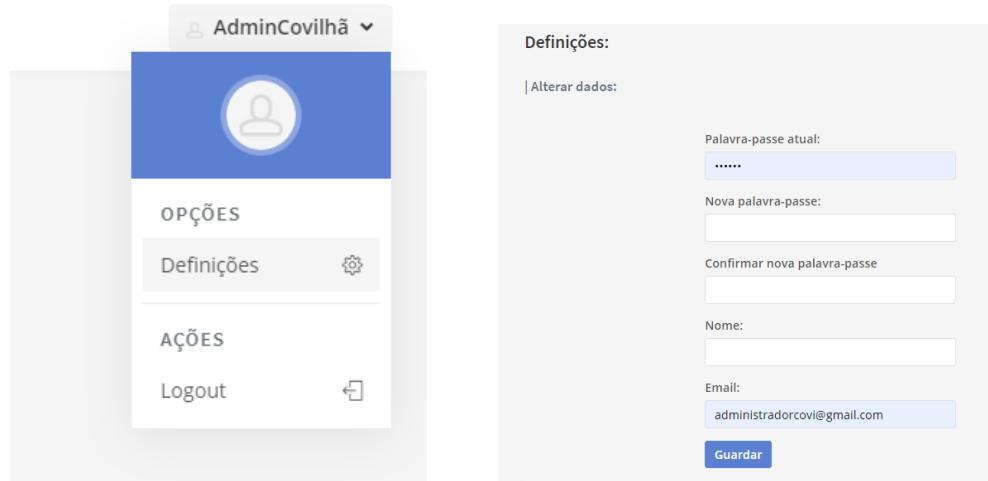
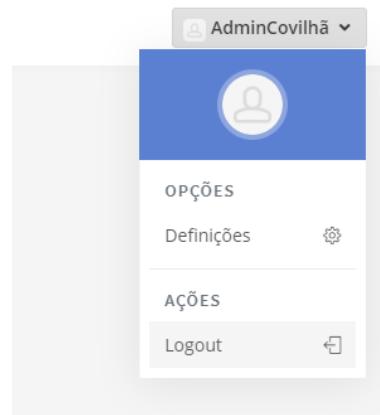


Figura 5.25: Definições.

Se todos os campos estiverem válidos, aparecerá uma notificação a dizer "Definições alteradas com sucesso!".

#### 5.4.9 Logout

Esta secção é permitida a ambos os utilizadores, administradores e farmacêuticos. Para um utilizador terminar sessão da sua conta, basta este se dirigir ao menu *dropdown* da barra de navegação superior, e clicar na opção "Logout", como mostra a figura 5.26.

Figura 5.26: Opção de *Logout*.

## 5.5 Pedidos

Esta secção vai detalhar o modo de funcionamento interno do processo de um pedido feito por um utilizador na interface da aplicação *Web*.

### 5.5.1 Fazer um pedido

Quando um utilizador acede à página de inserção de um novo pedido, este acede diretamente à função `fazerPedido()`, que se encontra no Controlador denominado de "PedidoControlador", caso o utilizador seja um farmacêutico (caso contrário, esta função encontra-se no AdminControlador). Esta função retorna assim a *View* que o utilizador imediatamente vê na interface gráfica, e que é possível observar nas figuras 5.16 e 5.17.

A função descrita, está programada da seguinte forma:

```
public function fazerPedido()
{
    $hosp = Hospital::all();
    $estado = Estado::all();
    $med= Medicamento::paginate(10);
    $dci= DCI::all();
    $forma= Forma::all();
    $dosagem = Dosagem::all();
    $selectedMed = Medicamento::first()->id;

    return view('fazerpedido', compact('hosp', 'estado', 'med',
        'dci', 'forma', 'dosagem', 'selectedMed'));
}
```

Exerto de Código 5.1: Função `fazerPedido()`.

Como é possível observar, a variável `$hosp` acede à base de dados, a partir do modelo "Hospital", e faz uma requisição de obtenção de todas as instâncias do modelo "Hospital" criadas. Por outras palavras, o método `::all()` retorna, neste caso, todos os hospitais até à data armazenados na base de dados.

O método `::paginate()` é o que permite a paginação da tabela existente na Visão correspondente, neste caso a cada 10 elementos.

O método `::first()` retorna apenas o primeiro elemento do *array* de resultados encontrados, devolvendo neste caso, apenas o ID desse elemento.

Por fim, o utilizador é retornado para a *View* "fazerpedido.blade.php" passando para esta, através do método `compact()`, todos as instâncias em cima requisitadas.

Aquando o correto preenchimento de todos os campos requeridos no formulário de inserção de um novo pedido, como é mostrado na secção 5.4.5, o utilizador recorre imediatamente à função `store(Request $request)`, como é demonstrado no excerto de código 5.2.

```
public function store(Request $request){
    $idmed = $request->get('idMed');
    $hospital = $request->get('wizard-progress2-hospital');
    $estado = $request->get('wizard-progress2-estado');
    $quantidade = $request->get('qtdMed');
    $find = Medicamento::find($idmed);
    if(isset($find)) {
        $pedido1 = new Pedido();
        $pedidolinha = new PedidoLinha();
        $pedido2 = new Pedido();

        $pedido1->user_id = Auth::user()->id;
        $pedido1->estado_id = $estado;
        $pedido1->tipo_id = 1;
        $pedido1->hospital_id_origem = Auth::user()->
            hospital_id;
        $pedido1->hospital_id_destino = $hospital;
        $pedido1->estado_pedido_id = 1;
        $pedido1->save();

        $pedidolinha->pedido_id = $pedido1->id;
        $pedidolinha->medicamento_id = $idmed;
        $pedidolinha->quantidade = $quantidade;
        $pedidolinha->save();

        $pedido2->user_id = Auth::user()->id;
        $pedido2->estado_id = $estado;
        $pedido2->tipo_id = 2;
        $pedido2->hospital_id_origem = Auth::user()->
            hospital_id;
        $pedido2->hospital_id_destino = $hospital;
        $pedido2->estado_pedido_id = 1;
        $pedido2->save();
    }
} else{
    return redirect('/fazerpedido')->with("error", "Esse
        ID nao existe. Por favor escolha um ID valido!");
}
return redirect('/fazerpedido')->with("success", "Pedido
    enviado com sucesso!");
}
```

Excerto de Código 5.2: Função `store(Request $request)`.

No excerto de código 5.2, o método `$request->get()` é o que permite obter os dados inseridos pelo utilizador no formulário. Desta forma, é possível trabalhar com esta informação, no Controlador respetivo. Antes de realizar operações com esses dados, o sistema verifica se o ID do medicamento que o utilizador inseriu no formulário de facto existe na base de dados, através do método `::find($id)`. Caso não exista, é retornado ao utilizador uma mensagem de erro. Caso a condição se verifique, acontecem uma série de transações, passando estas pela criação de dois novos pedidos - visto que a tabela de pedidos enviado e recebidos é a mesma, esta contém um atributo designado de "tipo\_id" que armazena "1" caso o pedido seja enviado e "2" caso o pedido seja recebido - sendo que o `$pedido1` se refere ao pedido enviado e o `$pedido2`, criando também uma instância do Modelo "PedidoLinha", instância esta que se refere aos detalhes de cada linha de um pedido enviado.

De seguida, é armazenado no `$pedido1` todos os campos necessários, sendo estes o utilizador que efetuou o pedido, o estado de urgência deste, o tipo, o hospital de origem e destino e o estado da encomenda do pedido. O método `Auth::user()` permite aceder aos dados do utilizador atual, e o método `"save()"` armazena os novos dados na base de dados.

Para as restantes variáveis (`$pedidolinha` e `$pedido2`) o procedimento é semelhante. Caso não ocorra nenhum erro, aparece uma mensagem de sucesso ao utilizador.

### 5.5.2 Pedidos enviados

```
public function pedidosEnviados()
{
    $hospu = Auth::user()->hospital_id;
    $med = Medicamento::all();
    $estado = Estado::all();
    $hosp = Hospital::all();
    $estadopdd = EstadoPedido::all();
    $pdd = Pedido::where('tipo_id', 1)->where(
        'hospital_id_origem', '=', $hospu)->orderBy('id', 'desc')->paginate(10);
    $pddlinha = PedidoLinha::all();
    $admin = Admin::where('hospital_id', $hospu)->get();
    $user = User::where('hospital_id', $hospu)->get();
    return view('pedidosenviados', compact('table', 'pdd', 'pddlinha', 'med', 'estado', 'hosp', 'hospu', 'admin', 'user', 'estadopdd'));
}
```

Exceto de Código 5.3: Função `pedidosEnviados()`.

O excerto de código 5.3 é referente à página referida na figura 5.18.

Aqui enviamos para a *View* o ID do centro hospitalar do utilizador atual, e todas as instâncias dos Modelos "PedidoLinha", "Medicamento", "Estado", "Hospital" e "EstadoPedido".

A variável \$pdd é referente aos pedidos que têm o atributo "tipo\_id" com o valor de "1" (ou seja, pedidos enviados) e onde o hospital de origem tem o valor do centro hospitalar atual do utilizador. Esta informação será organizada na *View* por ordem decrescente e será paginada a cada 10 elementos.

De seguida procuramos na tabela de administradores aquele que está responsável pelo centro hospitalar do utilizador atual (visto que apenas existe 1 administrador por centro hospitalar), bem como todos os farmacêuticos inseridos nesse mesmo centro hospitalar. Através do método `get()`, é possível passar para a Visão o *array* que contém todos os elementos correspondentes a uma determinada condição.

Por fim, passamos todos os dados obtidos para a *View* correspondente, onde estes vão ser tratados e mostrados de forma correta ao utilizador, na interface gráfica da plataforma.

```
public function pedidosEnviadosID($id)
{
    $med = Medicamento::all();
    $estado = Estado::all();
    $estadopdd = EstadoPedido::all();
    $dci = DCI::all();
    $forma = Forma::all();
    $dosagem = Dosagem::all();
    $pddlinha = PedidoLinha::where('pedido_id', $id)->get();

    return view('pedidosenviados_id', compact('pddlinha',
        'id', 'med', 'estado', 'estadopdd', 'dci', 'dosagem', 'forma',
    ));
}
```

Excerto de Código 5.4: Função `pedidosEnviadosID($id)`.

É com o código apresentado no excerto 5.4, que o utilizador tem a possibilidade de aceder à *View* apresentada na figura 5.19, e assim ver os detalhes de um determinado pedido.

Paralelamente aos excertos de código apresentados anteriormente nesta secção, também este vai aceder aos dados armazenados nas tabelas da base de dados com a ajuda do método `::all()`. Assim, é possível mostrar na *View* correspondente, todos os atributos específicos de cada medicamento.

A variável `$pddlinha`, através do Modelo "PedidoLinha", vai buscar à tabela em questão todas as instâncias onde o atributo `pedido_id` tenha o valor do pedido

selecionado pelo utilizador.

Por fim, todos os elementos são enviados para a *View*, onde são estruturados visualmente para o utilizador.

```
public function confirmenvio($id)
{
    $pedido = Pedido :: findOrFail($id);
    $pedido->estado_pedido_id = 2;
    $pedido->save();

    $pdd2 = Pedido :: findOrFail($id - 1);
    $pdd2->estado_pedido_id = 2;
    $pdd2->save();

    return redirect('/pedidosrecebidos');
}
```

Exerto de Código 5.5: Função confirmenvio(\$id).

O código apresentado no excerto 5.5, é referente à confirmação do envio de um pedido do centro hospitalar de destino para o centro hospitalar que está na origem deste.

Nesta etapa, foi apenas necessário mudar o estado do pedido com o ID recebido como parâmetro na função (caso este exista), de "1"(Recebido) para "2"("Em distribuição"), em ambos as instâncias do pedido (enviado e recebido), guardando estas alterações na base de dados.

```
public function rejectenvio($id)
{
    $pedido = Pedido :: findOrFail($id);
    $pedido->estado_pedido_id = 4;
    $pedido->save();

    $pdd2 = Pedido :: findOrFail($id - 1);
    $pdd2->estado_pedido_id = 4;
    $pdd2->save();

    return redirect('/pedidosrecebidos');
}
```

Exerto de Código 5.6: Função rejectenvio(\$id).

O código apresentado no excerto 5.6, é referente à rejeição do envio de um pedido do centro hospitalar de destino para o centro hospitalar que está na origem deste.

Paralelamente ao excerto apresentado em 5.5, foi apenas necessário mudar o estado do pedido com o ID recebido como parâmetro na função (caso este exista), de "1"(Recebido) para "4"("Rejeitado"), em ambos as instâncias do pedido (enviado e recebido), guardando estas alterações na base de dados.

### 5.5.3 Pedidos recebidos

```
public function pedidosRecebidos()
{
    $hospu = Auth::user()->>hospital_id;
    $pddlinha = PedidoLinha::all();
    $med = Medicamento::all();
    $estado = Estado::all();
    $hosp = Hospital::all();
    $estadopdd = EstadoPedido::all();
    $admin = Admin::all();
    $user = User::all();
    $pdd = Pedido::where('tipo_id', 2)>where(
        'hospital_id_destino', '=', $hospu )>orderBy('id', 'desc')>paginate(10);

    return view('pedidosrecebidos', compact('pdd', 'pddlinha',
        'med', 'estado', 'hosp', 'admin', 'user', 'estadopdd'));
}
```

Excerto de Código 5.7: Função pedidosRecebidos().

O excerto de código 5.7 é relativo à página referida na figura 5.20.

Aqui enviamos para a *View* todas as instâncias dos Modelos "PedidoLinha", "Medicamento", "Estado", "Hospital", "EstadoPedido", "Admin"e "User".

A variável \$pdd é referente aos pedidos que têm o atributo "tipo\_id" com o valor de "2"(ou seja, pedidos recebidos) e onde o hospital de destino tem o valor do centro hospitalar atual do utilizador. Esta informação será organizada na *View* por ordem decrescente e será paginada a cada 10 elementos.

Por fim, passamos todos os dados obtidos para a *View* correspondente, onde estes vão ser tratados e mostrados de forma correta ao utilizador, na interface gráfica da aplicação Web.

```

public function pedidosRecebidosID($id)
{
    $pddenv = Pedido::findOrFail($id);
    $createdpdd = $pddenv->created_at;

    $pddrec = Pedido::where('created_at', $createdpdd)->
        value('id');
    $pddlinha = PedidoLinha::where('pedido_id', $pddrec)->
        get();

    $med = Medicamento::all();
    $estado = Estado::all();
    $estadopdd = EstadoPedido::all();
    $dci = DCI::all();
    $forma = Forma::all();
    $dosagem = Dosagem::all();

    return view('pedidosrecebidos_id', compact('pddlinha',
        'id', 'med', 'estado', 'estadopdd', 'dci', 'dosagem', 'forma
        '));
}

```

Exerto de Código 5.8: Função `pedidosRecebidosID($id)`.

É com o código apresentado no exerto 5.8, que o utilizador tem a possibilidade de aceder aos detalhes de um determinado pedido recebido.

Paralelamente aos excertos de código apresentados anteriormente nesta secção, também este vai aceder aos dados armazenados nas tabelas da base de dados com a ajuda do método `::all()`. Assim, é possível mostrar na *View* correspondente, todos os atributos específicos de cada medicamento pertencente ao pedido selecionado. Primeiramente, foi necessário encontrar o pedido enviado correspondente ao ID que a função recebe como parâmetro, através do método `findOrFail()`.

Quando um utilizador faz um pedido, e como foi visto em excertos de código anteriores, é gerado na base de dados duas instâncias relativas aquele pedido, uma onde o atributo "tipo" tem o valor de "Enviado" e outra onde o mesmo atributo tem o valor de "Recebido". Assim, como ambas as linhas da coluna são criadas ao mesmo tempo, estas possuem a mesma data de criação inserida no campo "created\_at". Nesta ordem de ideias, foi criada uma variável `createpdd` que toma o valor da *timestamp* do pedido que outrora foi enviado.

Desta forma, através de uma cláusula `where`, foi identificado o ID do pedido recebido correspondente.

A variável `$pddlinha`, através do Modelo "PedidoLinha", vai buscar à tabela em questão a instância onde o atributo `pedido_id` tem o valor da variável `$pddrec`, que contém o ID do pedido em questão. Por fim, todos os elementos são enviados

para a *View*, onde são estruturados visualmente para o utilizador.

```
public function confirm($id)
{
    $pedido = Pedido::findOrFail($id);
    $pedido->estado_pedido_id = 3;
    $pedido->save();

    $pddl = PedidoLinha::where('pedido_id', $id)->value(
        'medicamento_id');
    $pdd12 = PedidoLinha::where('pedido_id', $id)->value(
        'quantidade');

    $medch = MedicamentoPorCH::where('medicamento_id', $pddl)
        ->where('hospital_id', $pedido->hospital_id_origem)
        ->first();

    $medch->quantidade = $medch->quantidade + $pdd12;
    $medch->save();

    $hist = new Historico();
    $hist->medicamento_id = $pddl;
    $hist->hospital_id = $pedido->hospital_id_origem;
    $hist->quantidade = $medch->quantidade;
    $hist->save();

    $pedido2 = Pedido::find($id+1);
    $pedido2->estado_pedido_id = 3;
    $pedido2->save();

    $medch2 = MedicamentoPorCH::where('medicamento_id',
        $pddl)
        ->where('hospital_id', $pedido->hospital_id_destino)
        ->first();

    $medch2->quantidade = $medch2->quantidade - $pdd12;
    $medch2->save();

    $hist2 = new Historico();
    $hist2->medicamento_id = $pddl;
    $hist2->hospital_id = $pedido->hospital_id_destino;
    $hist2->quantidade = $medch2->quantidade;
    $hist2->save();

    return redirect('/pedidosenviados');
}
```

Excerto de Código 5.9: Função `confirm($id)`.

O excerto de código 5.9, foi a solução encontrada para a confirmação de chegada de um medicamento derivado da requisição de um pedido.

Em primeiro lugar, o sistema vai tentar encontrar na base de dados o pedido com o valor de ID recebido pelo método *POST*, proveniente da requisição HTTP feita na *View*, aquando o clique no botão de confirmação de chegada de um pedido. De seguida, vai ocorrer uma alteração do estado do pedido que se encontrava "Em distribuição", para "Chegada ao destino".

Recorrendo ao Modelo "PedidoLinha", através de cláusulas **Where**, foram armazenados os valores dos atributos "medicamento\_id" numa variável denominada de \$pddl e "quantidade" numa variável denominada de \$pddl2.

O próximo passo, recorrendo ao Modelo "MedicamentoPorCH", foi encontrar uma instância deste onde o atributo "medicamento\_id" tivesse o valor armazenado em \$pddl e o atributo "hospital\_id" tomasse o valor do atributo "hospital\_id\_origem" referente ao pedido em questão. O primeiro (e único) resultado desta *query*, ficou armazenado na variável \$medch. O atributo "quantidade" referente à instância encontrada com as cláusulas impostas em \$medch, foi assim atualizado para a soma da quantidade previamente existente e da quantidade correspondente à variável \$pddl2. Com o auxílio do método `::save()`, a nova quantidade existente do medicamento no centro hospitalar em questão, foi assim guardada.

Foi também necessário criar uma nova instância do modelo "Historico", utilizando o comando `new Historico()`, instância esta que foi guardada numa variável denominada de \$hist. Todos os atributos relativos a esta instância foram guardados. O atributo "medicamento\_id" tomou o valor guardado em \$pddl, o "hospital\_id" tomou o valor do atributo "hospital\_id\_origem" armazenado no pedido, e a "quantidade" tomou o valor do mesmo atributo referente a \$medch.

Similarmente à primeira transação realizada neste excerto de código, também o pedido recebido correspondente ao pedido que foi enviado, teve o seu estado alterado para "Chegada ao destino".

Visto que o centro hospitalar que recebeu o pedido aumentou o seu *stock* de quantidade, em contrapartida o centro hospitalar que o enviou desceu o *stock*, na mesma quantidade. Para realizar esta operação, recorreu-se à criação de uma nova variável denominada de \$medch2. O valor atribuído a esta variável resultou de uma cláusula **Where**, onde para cada instância do Modelo "MedicamentoPorCH", se encontrou aquela onde o atributo "medicamento\_id" tomasse o valor de \$pddl e o atributo "hospital\_id" tomasse o valor do atributo "hospital\_id\_destino", relativo ao pedido. Depois de encontrado o resultado da restrição, atualizou-se o atributo "quantidade" para o valor que previamente existia menos o valor de \$pddl2, guardando o novo valor.

Por último, ocorreu também uma nova criação de uma instância do Modelo Historico, denominada de \$hist2 referente aos dados do centro hospitalar de destino. Assim, o atributo "medicamento\_id" tomou o valor da variável \$pddl1, o atributo

"hospital\_id" tomou o valor do atributo "hospital\_id\_destino" referente ao pedido, e o atributo "quantidade" tomou o valor do mesmo atributo referente à instância armazenada na variável \$medch2.

## 5.6 Testes

"Qualidade de software é um processo sistemático que focaliza todas as etapas e artefatos produzidos, com o objetivo de garantir a conformidade de processos e produtos, prevenindo e eliminando defeitos"

— Alexandre Bartié

Foram realizados diversos tipos de testes, tanto ao nível do utilizador como ao nível do servidor. Para este efeito, foram realizados testes unitários, com a ferramenta PHPUnit, testes ao nível do utilizador com a utilização da ferramenta *Chrome Dev Tools*, e testes manuais no decorrer de toda a implementação do projeto.

### 5.6.1 PHPUnit

O PHPUnit é uma ferramenta de teste para *frameworks* PHP, orientada aos programadores.

Esta ferramenta tem como principal objetivo testar partes isoladas do código, de modo a evitar falhas e erros quando estas comunicam entre si.

Esta ferramenta tem diversas vantagens, entre as quais:

- Os testes são automatizados, ou seja, não precisam de intervenção manual durante sua execução;
- São executados continuamente durante o ciclo de desenvolvimento;
- Detectam falhas tanto de digitação e lógica, como também, comportamentos inesperados.

Com a pouca experiência no uso desta ferramenta, os testes unitários foram bastante simples incidindo, na sua maioria, no teste de rotas de redirecccionamento, verificando o *status* retornado.

No excerto de código 5.10, encontra-se um exemplo de alguns destes testes efetuados:

```
<?php

namespace Tests\Unit;
use Tests\TestCase;

class UserTest extends TestCase
{
    public function testelogin()
    {
        $response = $this->get('/login');
        $response->assertStatus(200);
    }

    public function testedashboard()
    {
        $response = $this->get('/');
        $response->assertStatus(302);
    }

    public function testemedlocal()
    {
        $response = $this->get('/listarmedlocal');
        $response->assertStatus(302);
    }
}
```

Exerto de Código 5.10: Testes unitários.

### 5.6.2 *Chrome Dev Tools*

O *Chrome Dev Tools* é um conjunto de ferramentas de inspeção, depuração e otimização de páginas Web, que está incorporado no browser Google Chrome. Este vem com imensos painéis de ferramentas, entre os quais:

- *Device Mode*: indicado para criar experiências Web totalmente responsivas e voltadas para dispositivos móveis;
- *Elements*: utilizado para iterar no *layout* e criar um *site* manipulando livremente o Modelo de Objeto de Documento (DOM) e o CSS;
- *Console*: usado para registar informações de diagnóstico durante o desenvolvimento ou como *shell* para interagir com o JavaScript da página;
- *Sources*: ferramenta de depuração do JavaScript usando pontos de interrupção neste painel. É também possível conectar arquivos locais por meio de Espaços de trabalho, de modo a usar o editor do DevTools em tempo real;

- *Network*: utilizado para obtenção de *insights* sobre recursos solicitados e para otimizar o desempenho do carregamento da página;
- *Timeline*: ferramenta utilizada para melhorar o desempenho da página em tempo de execução, gravando e explorando os diversos eventos que acontecem durante o ciclo de vida de um *site*;
- *Profiles*: este é o painel indicado para obter mais informações do que aquelas fornecidas pela *Timeline* como, por exemplo, para rastrear vazamentos de memória;
- *Application*: inspeciona todos os recursos carregados, incluindo bases de dados SQL ou IndexedDB, armazenamento local e da sessão, *cookies*, cache da aplicação, imagens, fontes e folhas de estilo;
- *Security*: este painel tem como finalidade a depuração de problemas de conteúdo misto e com os certificados, entre outros. [4]

Os testes realizados com esta ferramenta, para a página principal do administrador, deram os seguintes resultados:

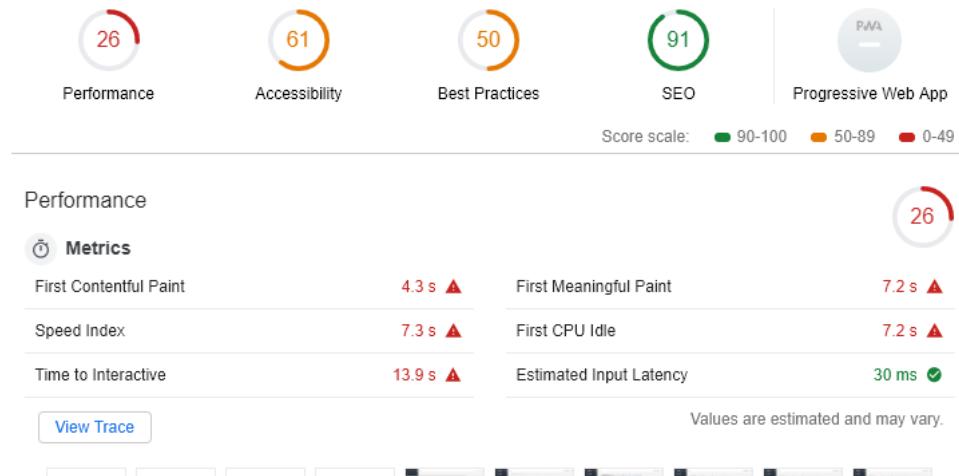


Figura 5.27: *Audit Tests* na página principal.

Com a realização dos testes demonstrados na figura 5.27, é possível concluir que na *dashboard* do utilizador, a parte da *performance* é claramente alvo de melhorias futuras. Em contrapartida a análise de *Search Engine Optimization* (SEO) foi bastante bem conseguida.

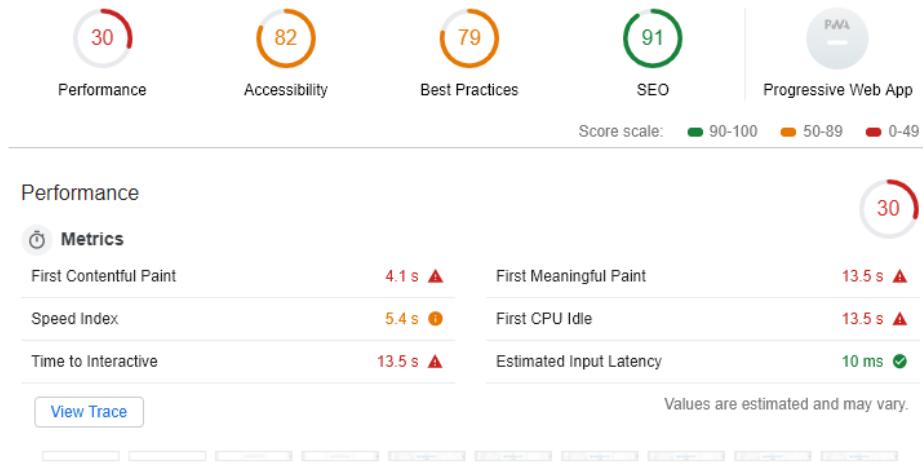


Figura 5.28: Audit Tests na página de *login*.

Para a página principal de *login*, os testes realizados deram os seguintes resultados:

Com os gráficos representados na figura 5.29, é possível concluir que houve uma notória melhoria em todos os campos, à exceção do campo de SEO que se manteve igual. Ainda assim, a parte da performance mantém-se baixa, devendo esta ser analisada pormenorizadamente para trabalhos futuros, ou melhoria do projeto.

### 5.6.3 Testes manuais

Os testes manuais foram realizados ao longo de todo o processo de desenvolvimento, tendo existido diversas melhorias constantes e tendo-se criado melhores soluções de implementação para algumas funcionalidades.

Ao longo do desenvolvimento da aplicação *Web*, foram implementadas diversas mensagens de erro, caso algo corra mal.

#### Mensagens de erro na autenticação

Ambas as imagens reproduzidas na figura 5.29, demonstram mensagens de erros, quando um dos campos de autenticação não é preenchido.

As imagem a) da figura 5.30, demonstra a mensagem de erro que é reproduzida, quando as credenciais de acesso do utilizador se encontram incorretas. A imagem b) referente à mesma figura, representa um erro de má formatação no *email*, onde naquele caso falta um "@".

Na figura 5.31, a mensagem de erro é devido ao endereço de *email* se encontrar

incompleto.

The figure consists of two side-by-side screenshots of a login form for 'MedGest'. Both screenshots show a blue header bar with 'LOGIN', 'Login as Admin', and 'Forgot Your Password?'. Below the header is the MedGest logo and a message: 'Por favor insira os seus dados de login.' (Please enter your login details.)

**(a) Campo password por preencher:** The 'Email' field contains 'rita.correia7@gmail.com'. The 'Password' field is empty. A red error box with an exclamation mark is positioned above the empty password field, containing the text 'Preencha este campo.' (Fill this field.).

**(b) Campo email por preencher:** The 'Email' field is empty. A red error box with an exclamation mark is positioned above the empty email field, containing the text 'Preencha este campo.' (Fill this field.). The 'Password' field contains '\*\*\*\*\*' and the 'Login' button is visible at the bottom.

Figura 5.29: Campos em falta na autenticação.

The figure consists of two side-by-side screenshots of a login form for 'MedGest'. Both screenshots show a blue header bar with 'LOGIN', 'Login as Admin', and 'Forgot Your Password?'. Below the header is the MedGest logo and a message: 'Por favor insira os seus dados de login.' (Please enter your login details.)

**(a) Credenciais de acesso incorretas:** The 'Email' field contains 'rita.correia7@gmail.com'. A red error box with an exclamation mark is positioned below the field, containing the text 'Essas credenciais não existem nos nossos registos.' (These credentials do not exist in our records.). The 'Password' field contains '\*\*\*\*\*' and the 'Login' button is visible at the bottom.

**(b) Formatação de email incorreta:** The 'Email' field contains 'rita.correia7'. A red error box with an exclamation mark is positioned above the field, containing the text 'Inclua um "@" no endereço de email. Falta um "@" em "rita.correia7".' (Include an "@" in the email address. There is no "@" in "rita.correia7"). The 'Password' field contains '\*\*\*\*\*' and the 'Login' button is visible at the bottom.

Figura 5.30: Credenciais incorretas ou má formatação.

The screenshot shows the MedGest login interface. At the top, there is a blue header bar with the word "LOGIN" and links for "Login as Admin" and "Forgot Your Password?". Below the header, the MedGest logo is displayed with the text "Por favor insira os seus dados de login." (Please enter your login details.). There are two input fields: "Email" containing "rita.correia7@" and "Password" containing several dots. A tooltip message "Introduza uma parte a seguir a '@'. 'rita.correia7@' está incompleto." (Please enter a part after '@'. 'rita.correia7@' is incomplete.) is shown above the password field. At the bottom is a blue "Login" button with a right-pointing arrow.

Figura 5.31: Endereço de *email* incompleto.

### Mensagens de erro no registo de farmacêuticos

Para além das mensagens de erro apresentadas para a autenticação, no ato de registo de um farmacêutico existem ainda erros de palavra-passe, incluindo o tamanho mínimo de 6 caracteres que esta deve ter e a não coincidência de ambas. Existe também uma mensagem de erro devido ao esquecimento de preenchimento de alguns campos.

The image contains two side-by-side screenshots of a pharmaceutical registration form. Both screenshots show the following fields: Nome (Nome: Teste), Centro Hospitalar (Centro Hospitalar: Pêro da Covilhã), Estado (Estado: Efetivo), Email (Email: teste@teste.com), and Palavra-passe (Palavra-passe: .....). Below the password field, a red error message "As passwords não coincidem." (The passwords do not match.) is visible. In the second screenshot, the Email field has been modified to "tteste" and the Palavra-passe field has been modified to ".....", with corresponding red error messages "A password deve ter pelo menos 6 caracteres." (The password must have at least 6 characters.) appearing below each field.

(a) Credenciais de acesso incorretas.

(b) Formatação de *email* incorreta.

Figura 5.32: Credenciais incorretas ou má formatação.

(a) O campo "Nome" não foi preenchido.

(b) O campo "Confirmar palavra-passe" não foi preenchido.

Figura 5.33: Obrigatoriedade de preenchimento de todos os campos no registo de uma conta.

### Mensagens de erro na inserção de medicamentos

Quando um administrador pretende inserir um novo medicamento no seu centro hospitalar, é obrigatório este preencher todos os campos. Caso contrário, aparecerão as mensagens de erro demonstradas na figura 5.34.

(a) O campo "Data de validade" não tem um resultado válido.

(b) O campo "Quantidade" não foi preenchido.

Figura 5.34: Obrigatoriedade de preenchimento de todos os campos na inserção de medicamentos.

Quando o administrador pretende adicionar um medicamento já existente na base de dados ao seu centro hospitalar, este quem que inserir a quantidade do mesmo. O valor da quantidade tem que ser válido, não podendo ser um valor inferior a 1, como mostra a figura 5.35.

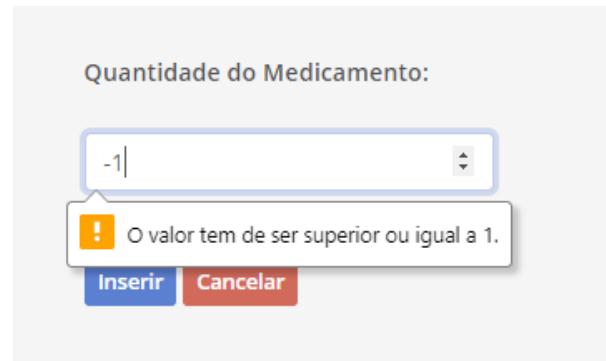


Figura 5.35: Quantidade do medicamento inválida.

### Mensagens de erro nos pedidos

Quando um utilizador está a preencher o formulário para fazer um pedido, este deve inserir um ID de medicamento e quantidade válidos, caso contrário aparecem mensagens de erro para ambos os casos, como é visível na figura 5.36 e 5.37, respetivamente.

#	DCI	DOSAGEM	FORMA
1	Ácido Fólico	1mg	Comprimido de liberação prolongada
2	Hidrocortisona	200mg	Creme
3	Ipobrufen	600mg	Comprimido revestido por película
4	Paracetamol + Cloridrato de difenidramina	10mg	Comprimido para mastigar
5	Triamcinolona	100mg	Comprimido orodispersível
6	Dexcetoprofeno	400mg	Granulado efervescente
7	Ácido Fólico	1mg	Comprimido de liberação prolongada

ID do medicamento:  
10

Quantidade:

Figura 5.36: ID de medicamento inválido.

1. Pedido		2. Medicamento	
#	DCI	DOSAGEM	FORMA
1	Ácido Fólico	1mg	Comprimido de liberação prolongada
2	Hidrocortisona	200mg	Creme
3	Ipobrufenol	600mg	Comprimido revestido por película
4	Paracetamol + Cloridrato de difenidramina	10mg	Comprimido para mastigar
5	Triamcinolona	100mg	Comprimido orodispersível
6	Dexacetoprofeno	400mg	Granulado efervescente
7	Ácido Fólico	1mg	Comprimido de liberação prolongada

ID do medicamento:  
5

Quantidade:  
q ! O valor tem de ser superior ou igual a 1.

Figura 5.37: Quantidade inválida.

## 5.7 Conclusões

Neste capítulo foram abordadas diversas temáticas, que seguiram a ordem de implementação da plataforma ao longo do ultimo semestre.

Inicialmente comecei este percurso desafiante com um curso *online* que acentuou de forma bastante rápida e eficaz a minha curva de aprendizagem no que toca à *framework* utilizada.

A criação da base de dados, foi um processo contínuo durante todo o desenvolvimento do projeto, de modo a esta ter uma estrutura sólida, consistente e de fácil manutenção. Este foi talvez um dos processos mais demorados e sujeito a alterações à medida do decorrer da implementação.

Foi descrito assim também todo o manual do utilizador, que com a ajuda de capturas de vários ecrãs, correspondentes a várias *Views* da plataforma, tornam o processo de adaptação muito mais simples.

Visto que o core do projeto incide principalmente na troca de pedidos entre centros hospitalares, decidi colocar e explicar o código que me levou a determinadas soluções de implementação.

Por fim, e ainda com alguma falta de experiência na área de testes, foram descritos todos aqueles que foram feitos, em diferentes plataformas, e da melhor maneira possível.



# Capítulo 6

## Conclusões e Trabalho Futuro

Este capítulo tem como objetivo apresentar as conclusões principais do projeto, face aos objetivos inicialmente propostos. Serão também referidas possíveis melhorias da plataforma e o trabalho futuro que poderá ser realizado.

### 6.1 Conclusões Principais

O projeto foi realizado durante o segundo semestre do terceiro ano no curso de Informática Web, o que condicionou um pouco em termos de tempo a minha curva de aprendizagem, visto que não possuia quaisquer conhecimentos sobre a *framework* utilizada, nem sobre a linguagem PHP.

O espaço laboratorial que me foi disponibilizado para o desenrolar deste projeto, facilitou todo o processo de desenvolvimento e conclusão deste trabalho.

Na minha opinião, o projeto atendeu a quase todos os objetivos indicados na proposta inicial, incluindo o foco na segurança que me foi solicitado, e ao qual dediquei uma boa parte do tempo, assim como ao nível da interface gráfica para o utilizador.

Foi um trabalho que me permitiu crescer imenso, tanto a nível académico, como a nível profissional e, até diria, pessoal. Conseguí atingir as metas que me impus a mim própria nos *timings* possíveis, e por isso não poderia estar mais orgulhosa do meu desempenho e do desenvolvimento desta plataforma a que dei o nome de MedGest.

É de salientar que existe sem dúvida ainda muita coisa a melhorar, e acredito que se houvesse mais tempo, o resultado seria ainda melhor.

Este projeto permitiu também perceber a importância e crescimento que as TI estão a ter em todas as áreas, nomeadamente nas áreas da Saúde. A criação deste tipo de plataformas permite agilizar todas as operações nos centros hospitalares que envolvem controlo, gestão de medicamentos e *stock*, facilitando ainda o pro-

cesso de comunicação entre os vários hospitais.

## 6.2 Trabalho Futuro

Devido ao imenso foco que tive em tentar arranjar boas soluções de segurança, no meadamente nos processos de autenticação que achei indispensáveis, não atingi apenas um dos objetivos que constavam na proposta de projeto, sendo este um sistema de notificações por *email* e/ou telemóvel aquando a proximidade do término da data de validade de um medicamento. Por este motivo, o primeiro passo do trabalho futuro no desenvolvimento da plataforma, passaria exatamente por este ponto.

Como trabalho futuro, é também necessário melhorar a qualidade da *performance* da aplicação *Web*, melhorar o perfil do utilizador, adicionando uma foto deste e mais detalhes pessoais e colocar a plataforma *online*.

Seria interessante desenvolver uma sala de *chat* entre todos os colaboradores de um centro hospitalar, e um *feed* de notícias relevantes.

Seria também inovador o desenvolvimento de uma versão da plataforma para sistemas *mobile* iOS e Android.

Uma boa solução de *hosting* da plataforma, poderia ainda passar pela implementação desta numa plataforma de *cloud*...

...quem sabe numa versão posterior do projeto.

# Bibliografia

- [1] O que é World Wide Web?, 2008. [Online] <https://www.tecmundo.com.br/web/759-o-que-e-world-wide-web-.htm>. Último acesso a 15 de Julho de 2019.
- [2] O que é a Web 3.0?, 2009. [Online] <https://www.publico.pt/2009/06/29/tecnologia/noticia/o-que-e-a-web-30-1389325>. Último acesso a 15 de Julho de 2019.
- [3] 10 frameworks de desenvolvimento web essenciais para o seu código, 2019. [Online] <https://blog.mastertech.com.br/tecnologia/frameworks-de-desenvolvimento/>. Último acesso a 5 de Julho de 2019.
- [4] Chrome DevTools | Tools for Web Developers | Google Developers, 2019. [Online] <https://developers.google.com/web/tools/chrome-devtools/>. Último acesso a 13 de Julho de 2019.
- [5] Como funciona a arquitetura cliente servidor, 2019. [Online] <https://arqserv.wordpress.com/2012/03/17/como-funciona-a-arquitetura-cliente-servidor/>. Último acesso a 4 de Julho de 2019.
- [6] Entenda o que é Framework - Gaea Consulting, 2019. [Online] <https://gaea.com.br/entenda-o-que-e-framework/>. Último acesso a 5 de Julho de 2019.
- [7] Entenda o que é Framework - Gaea Consulting, 2019. [Online] <https://gaea.com.br/entenda-o-que-e-framework/>. Último acesso a 5 de Julho de 2019.
- [8] History, 2019. [Online] <https://v4-alpha.getbootstrap.com/about/history/>. Último acesso a 7 de Julho de 2019.
- [9] Laravel 5.6 Completo - O mais poderoso Framework PHP, 2019. [Online] <https://www.udemy.com/laravelcompleto/>. Último acesso a 7 de Julho de 2019.

- [10] MySQL - Banco de Dados, 2019. [Online] <https://www.infoescola.com/informatica/mysql/>. Último acesso a 6 de Julho de 2019.
- [11] O que é CSS? Guia Básico para Iniciantes, 2019. [Online] <https://www.hostinger.com.br/tutoriais/o-que-e-css-guia-basico-de-css/>. Último acesso a 6 de Julho de 2019.
- [12] O que é Laravel? Porque usá-lo?, 2019. [Online] <https://medium.com/joaorobertopb/o-que-%C3%A9-laravel-porque-us%C3%A1-lo-955c95d2453d>. Último acesso a 7 de Julho de 2019.
- [13] O que é um servidor? Quais são os tipos de servidores?, 2019. [Online] <https://trogadas.com/blog/tipos-de-servidores/>. Último acesso a 2 de Julho de 2019.
- [14] O que é um servidor web (web server) - Tudo Sobre Hospedagem de Sites, 2019. [Online] <https://tudosobrehospedagemdesites.com.br/servidor-web/>. Último acesso a 2 de Julho de 2019.
- [15] Online Courses - Learn Anything, On Your Schedule, 2019. [Online] <https://www.udemy.com/>. Último acesso a 7 de Julho de 2019.
- [16] PHP: O que é o PHP? - Manual, 2019. [Online] [https://www.php.net/manual/pt\\_BR/intro-whatis.php](https://www.php.net/manual/pt_BR/intro-whatis.php). Último acesso a 4 de Julho de 2019.
- [17] PhpStorm: The Lightning-Smart IDE for PHP Programming by JetBrains, 2019. [Online] <https://www.jetbrains.com/phpstorm/>. Último acesso a 7 de Julho de 2019.
- [18] SQL | Definition, Programming, History, 2019. [Online] <https://www.cleverism.com/skills-and-tools/sql/>. Último acesso a 6 de Julho de 2019.
- [19] Welcome to The Apache Software Foundation!, 2019. [Online] <https://www.apache.org/>. Último acesso a 6 de Julho de 2019.
- [20] World Wide Web, 2019. [Online] [https://developer.mozilla.org/pt-PT/docs/Gloss%C3%A1rio/World\\_Wide\\_Web](https://developer.mozilla.org/pt-PT/docs/Gloss%C3%A1rio/World_Wide_Web). Último acesso a 15 de Julho de 2019.
- [21] Alberto Sardinha. Modelo Entidade-Associação (EA), 2012. [Online] <https://fenix.tecnico.ulisboa.pt/downloadFile/3779579572812/mod02-1-Modelo-EA.pdf>. Último acesso a 11 de Julho de 2019.

- 
- [22] Allyn Grey de Almeida Lima. Padrão SQL e sua Evolução, 2019. [Online] <http://www.ic.unicamp.br/~geovane/mo410-091/Ch05-PadraoSQL-art.pdf>. Último acesso a 6 de Julho de 2019.
  - [23] Carla Geovana do N. Macário e Stefano Monteiro Baldo. O Modelo Relacional, 2019. [Online] <http://www.ic.unicamp.br/~geovane/mo410-091/Ch03-RM-Resumo.pdf>. Último acesso a 6 de Julho de 2019.
  - [24] José Coelho. Introdução à base de dados, 2011. [Online] <https://repositorioaberto.uab.pt/bitstream/10400.2/3462/1/Introdu%C3%A7%C3%A3o%20%C3%A0%20Base%20de%20Dados.pdf>. Último acesso a 4 de Julho de 2019.
  - [25] Eduardo Figueiredo. Requisitos Funcionais e Requisitos Não Funcionais, 2019. [Online] [https://homepages.dcc.ufmg.br/~figueiredo/disciplinas/aulas/req-funcional-rnf\\_v01.pdf](https://homepages.dcc.ufmg.br/~figueiredo/disciplinas/aulas/req-funcional-rnf_v01.pdf). Último acesso a 7 de Julho de 2019.
  - [26] Prof. Renato Fileto. Sistemas Cliente-Servidor, 2019. [Online] <http://www.inf.ufsc.br/~r.fileto/Disciplinas/BD-Avancado/Aulas/03-ClienteServidor.pdf>. Último acesso a 4 de Julho de 2019.
  - [27] Taylor Otwell. Laravel - The PHP Framework For Web Artisans, 2019. [Online] <https://laravel.com/>. Último acesso a 7 de Julho de 2019.