

Universidade da Beira Interior

Departamento de Informática



Nº 286 - 2021/2022: *Aplicação Móvel para Strapi*

Elaborado por:

Manuel Ferreira Magalhães

Orientador:

Professor Doutor Nuno Gonçalo Coelho Costa Pombo

24 de junho de 2022

Agradecimentos

Venho aqui a agradecer a um conjunto de pessoas que me motivaram e ajudaram durante estes 3 anos de licenciatura. Um imenso obrigado aos meus pais e avós por um grande apoio que me forneceram nestes anos de esforço e trabalho, por sempre me ajudarem e motivarem quando mais precisava durante longos dias e longas noites de trabalho constante. Por sempre efetuarem sacrifícios em prol da minha educação e da minha vida futura. Um grande obrigado aos meus amigos e colegas de curso por todas as horas de trabalho e diversão bem como todas as ajudas fornecidas para ultrapassar grandes desafios neste curso. Um enorme agradecimento ao Professor Doutor Nuno Gonçalo Coelho Costa Pombo por esta proposta de projeto apelativa e desafiante e por todo o acompanhamento, orientação e aconselhamento constante no desenvolvimento deste projeto final de Licenciatura. Um grande obrigado também ao Centro de Inovação Empresarial da Covilhã, mais concretamente ao colega Gil Marques por toda a ajuda disponibilizada, todas as reuniões de trabalho e pelas novas amizades criadas. Obrigado também ao colega José Bongo de Informática Web pelo fornecimento de uma ferramenta utilizada nos testes deste projeto. Um último agradecimento à Universidade da Beira Interior pela criação da minha vida académica e construção do meu futuro.

Conteúdo

Conteúdo	iii
Lista de Figuras	vii
Lista de Tabelas	ix
1 Introdução	1
1.1 Enquadramento	1
1.2 Motivação	2
1.3 Objetivos	2
1.4 Organização do Documento	3
2 Estado da Arte	5
2.1 Introdução	5
2.2 CMS — <i>Content Management System</i>	5
2.2.1 Introdução	5
2.2.2 Razão de Escolha/Motivação	6
2.3 Strapi - <i>Headless CMS</i>	7
2.3.1 Introdução	7
2.3.2 Razão de Escolha/Motivação:	8
2.4 Flutter Framework	10
2.4.1 Introdução	10
2.4.2 Razão de Escolha/Motivação:	11
2.5 Conclusão	12
3 Tecnologias e Ferramentas Utilizadas	13
3.1 Introdução	13
3.2 Strapi - Headless Content Management System	14
3.2.1 Especificação de um Headless Content Management System	14
3.2.2 Gestão de conteúdo	15
3.2.3 REST API	19
3.2.4 Utilizadores, Papéis e Permissões	20

3.2.5	Envio de <i>Emails</i>	21
3.3	Flutter - Framework de Desenvolvimento de Aplicações Móveis	22
3.3.1	Especificação acerca do Flutter:	22
3.4	Conclusão	24
4	Engenharia de Software	25
4.1	Introdução	25
4.2	Engenharia de Requisitos	26
4.3	Diagramas de Casos de Uso	28
4.4	Diagramas de Atividade	29
4.5	Arquitetura de Sistema	31
4.6	Base de Dados	32
4.7	Wireframes	35
4.8	<i>Mockups</i>	37
4.9	Conclusão	39
5	Implementação, Testes e Manual de Utilização	41
5.1	Introdução	41
5.2	Strapi	42
5.2.1	Instalação e Execução	42
5.2.2	Criação de Conteúdo	43
5.3	Aplicação Móvel	46
5.3.1	Página Inicial	47
5.3.1.1	<i>Layout</i> e Funcionalidades	47
5.3.2	Página Registo	49
5.3.2.1	<i>Layout</i> e Funcionalidades	49
5.3.3	Página Esquecimento <i>Password</i>	50
5.3.3.1	<i>Layout</i> e Funcionalidades	50
5.3.4	Página Reset <i>Password</i>	52
5.3.4.1	<i>Layout</i> e Funcionalidades	52
5.3.5	Página Principal de Utilizador	54
5.3.5.1	<i>Layout</i> e Funcionalidades	54
5.3.6	Página Sobre Nós	55
5.3.6.1	<i>Layout</i> e Funcionalidades	55
5.4	Testes Efetuados	57
5.4.1	Testes para Página Inicial	59
5.4.2	Testes para Página Registo	60
5.4.3	Testes para Página Esquecimento <i>Password</i>	61
5.4.4	Testes para Página Reset <i>Password</i>	62
5.4.5	Testes para Página Principal de Utilizador	63

CONTEÚDO

V

5.4.6	Testes para Página Sobre Nós	64
5.4.7	Exemplos práticos	65
5.4.7.1	Edição de texto e imagem no <i>Single Type About Area</i>	65
5.4.7.2	Edição de texto e imagem no <i>Collection Type Photo Gallery Menu</i>	67
5.5	Manual de Utilização	69
5.5.1	Página Inicial	69
5.5.2	Página Registo	69
5.5.3	Página Esquecimento <i>Password</i>	69
5.5.4	Página <i>Reset Password</i>	70
5.5.5	Página Principal de Utilizador	70
5.5.6	Página Sobre Nós	70
5.6	Conclusão	71
6	Conclusões e Trabalho Futuro	73
6.1	Conclusões Principais	73
6.2	Trabalho Futuro	74
Bibliografia		75

Listas de Figuras

2.1	Tempo total de execução de tarefas : Node . js vs HyperText Preprocessor (PHP) imagem retirada de [1].	9
2.2	Número de pedidos em média que cada tecnologia consegue processar por segundo, imagem retirada de [1].	9
3.1	Vários tipos de arquitetura de Content Management System (Content Management System (CMS)), imagem retirada de [2].	14
3.2	<i>Content-Type Builder</i> na interface da Strapi.	16
3.3	Criar um <i>Single Type</i>	16
3.4	Opções de campos a criar no <i>Single Type</i>	17
3.5	<i>Single Type</i> criado anteriormente com os campos adicionados. . .	17
3.6	Opções de operação num <i>single type</i> no <i>Content Manager</i>	18
3.7	Atribuição de permissões de acesso a utilizadores públicos para a Application Programming Interface (API) do Single Type.	20
3.8	Estrutura da <i>framework</i> Flutter, imagem retirada de [3].	23
4.1	Diagrama de caso de uso relativo ao sistema de autenticação.	28
4.2	Diagrama de caso de uso relativo ao menu principal de utilizador.	29
4.3	Diagrama de atividade relativo ao sistema de autenticação.	30
4.4	Diagrama de atividade relativo ao menu principal de utilizador.	30
4.5	Diagrama de atividade relativo à alteração de credenciais.	31
4.6	Arquitetura do Sistema.	31
4.7	Modelo relacional da base de dados da Strapi.	34
4.8	<i>Wireframe</i> de Página Inicial e de Página Registo.	35
4.9	<i>Wireframe</i> de Página Esquecimento Password e de Página Reset Password.	36
4.10	<i>Wireframe</i> de Página de Utilizador e de Página Sobre Nós.	36
4.11	<i>Mockup</i> de Página Inicial e de Página Registo.	37
4.12	<i>Mockup</i> de Página Esquecimento Password e de Página Reset Password.	38
4.13	<i>Mockup</i> de Página de Utilizador e de Página Sobre Nós.	39
5.1	<i>Collection Type Photo Gallery About Us</i> criado via interface.	44
5.2	<i>Collection Type Photo Gallery Menu</i> criado via interface.	44

5.3	<i>Single Type About Area</i> criado via interface.	45
5.4	<i>Single Type About Area API route</i> dedicado para obtenção de informação via <i>GET Request</i>	46
5.5	<i>Layout</i> da página inicial.	48
5.6	<i>Layout</i> da página de registo.	50
5.7	<i>Layout</i> da página de esquecimento de password.	52
5.8	<i>Layout</i> da página de <i>reset</i> da <i>password</i>	53
5.9	<i>Layout</i> da página principal de utilizador.	55
5.10	<i>Layout</i> da página Sobre Nós.	57
5.11	Conteúdo do <i>Single Type About Area</i> antes da utilização da funcionalidade de edição de texto e imagem da aplicação.	65
5.12	Conteúdo do <i>Single Type About Area</i> depois da utilização da funcionalidade de edição de texto e imagem da aplicação.	65
5.13	Conteúdo da página Menu do <i>website</i> antes da utilização da funcionalidade de edição de texto e imagem da aplicação.	66
5.14	Conteúdo da página Menu do <i>website</i> depois da utilização da funcionalidade de edição de texto e imagem da aplicação.	66
5.15	Conteúdo do <i>Collection Type Photo Gallery Menu</i> antes da utilização da funcionalidade de <i>upload</i> imagem da aplicação.	67
5.16	Conteúdo do <i>Collection Type Photo Gallery Menu</i> depois da utilização da funcionalidade de <i>upload</i> imagem da aplicação.	67
5.17	Conteúdo da página Sobre Nós do <i>website</i> antes da utilização da funcionalidade de <i>upload</i> imagem da aplicação.	68
5.18	Conteúdo da página Menu do <i>website</i> depois da utilização da funcionalidade de <i>upload</i> imagem da aplicação.	68

Listas de Tabelas

3.1	REST API Endpoints - Collection Type.	19
3.2	REST API Endpoints - Single Type.	19
5.1	Testes efetuados para Página Inicial.	59
5.2	Testes efetuados para Página Registro.	60
5.3	Testes efetuados para Página Esquecimento <i>Password</i>	61
5.4	Testes efetuados para Página <i>Reset Password</i>	62
5.5	Testes efetuados para Página Principal de Utilizador.	63
5.6	Testes efetuados para Página Sobre Nós.	64

Listas de Excertos de Código

3.1	Ficheiro <i>plugin</i> para especificação de <i>email provider</i>	21
3.2	Ficheiro de configuração <i>env</i>	22
5.1	Comando necessário para criação de projeto Strapi local.	42
5.2	Comando necessário para execução de projeto Strapi local em modo <i>develop</i>	43
5.3	Comando necessário para execução de projeto Strapi local em modo <i>start</i>	43

Acrónimos

HTML HyperText Markup Language

HTTP HyperText Transfer Protocol

JSON JavaScript Object Notation

LLVM Low Level Virtual Machine

SMTP Simple Mail Transfer Protocol

API Application Programming Interface

APK Android Application Package

ARM Advanced RISC Machines

CLI Command Line Interface

CMS Content Management System

IPA iPhone Application Archive

JWT JSON Web Token

NDK Native Development Kit

PHP HyperText Preprocessor

SDK Software Development Kit

UBI Universidade da Beira Interior

URL Uniform Resource Locator

WEB World Wide Web

UI User Interface

Capítulo

1

Introdução

1.1 Enquadramento

Este trabalho foi desenvolvido, no âmbito da unidade curricular de Projeto na Universidade da Beira Interior (UBI) no ano letivo de 2021/2022 em Licenciatura no curso de Engenharia Informática. Este trabalho comprehende diversas áreas tais como Programação Web, Programação de Dispositivos Móveis e Engenharia de Software.

O tema deste trabalho está inserido principalmente nas áreas de Programação Web e Programação de Dispositivos Móveis mais concretamente na questão da gestão de conteúdo digital de um *website* via um Content Management System (CMS) e do desenvolvimento de uma aplicação móvel que abrange esta gestão. A tecnologia do CMS é a base mais importante para a realização deste projeto.

Visa-se construir uma aplicação móvel que consiga ser executada em sistemas operativos Android e iOS que permita ao seu utilizador conseguir efetuar uma gestão eficiente do conteúdo de um *website* que lhe esteja associado pela utilização da Strapi *Headless CMS* como principal agente de alteração de conteúdo.

1.2 Motivação

Verifica-se atualmente um grande crescimento de mercado no que diz respeito à criação de *websites* em que estes são utilizados para meios de informação ou meios de negócio, por exemplo. Devido a este crescimento exponencial deste tipo de mercado foram desenvolvidas diversas tecnologias para auxiliar todos os interessados em começar o seu próprio *website*.

Em algumas situações é necessário conhecimento na linguagem HyperText Markup Language (HTML) para a criação e gestão do *website* enquanto que noutras são fornecidos os meios necessários sem necessitar de conhecimento adicional. A maior parte das tecnologias que auxiliam na gestão do conteúdo do *website* necessitam que o cliente possua um computador para começar o desenvolvimento.

A motivação por detrás da realização deste projeto é procurar ir além do que está disponível no mercado e poder fornecer ao utilizador um meio móvel que efetue alterações ao *website* sem necessitar de recorrer a um computador. Procura-se também diminuir a dificuldade de gestão de um *website* por parte de um dado utilizador fornecendo-lhe um sistema simples e eficiente neste aspeto.

1.3 Objetivos

O principal objetivo deste trabalho consiste no desenvolvimento de uma aplicação móvel que consiga ser executada em sistemas operativos Android e iOS que permita ao seu utilizador uma gestão eficiente e prática do conteúdo digital de um *website* através de um CMS. No contexto deste projeto procura-se atingir os seguintes objetivos:

- Desenvolver uma aplicação móvel com utilização de uma *framework* de desenvolvimento de aplicações móveis, sendo neste caso a *framework* Flutter;
- Aprendizagem e aplicação da linguagem Dart sendo esta necessária para possibilitar o desenvolvimento na *framework* Flutter;
- Aprendizagem e aplicação do Headless CMS Strapi, para possibilitar uma gestão de conteúdo digital de um dado website;
- Aplicação das Application Programming Interface (API)'s inerentes à Strapi, para posterior *update* do conteúdo digital do website;

- Desenvolver uma aplicação móvel funcional que utiliza a Strapi para gestão de conteúdo.

1.4 Organização do Documento

Nesta secção encontra-se a estrutura de organização deste documento:

1. O primeiro capítulo – **Introdução** – introduz o projeto, a motivação por detrás da sua realização, os objetivos a atingir e a organização do documento;
2. O segundo capítulo – **Estado da Arte** – onde se refere todas as tecnologias utilizadas no projeto com uma breve descrição, todas as razões por detrás das escolhas e a importância de cada tecnologia utilizada no contexto do projeto;
3. O terceiro capítulo – **Tecnologias e Ferramentas Utilizadas** – refere detalhadamente alguns aspetos mais importantes das ferramentas utilizadas bem como aplicações utilizadas no desenvolvimento deste projeto;
4. O quarto capítulo – **Engenharia de Software** – apresenta engenharia de requisitos, diagramas de caso de uso, de atividade, arquitetura do sistema, estrutura da base de dados bem como *wireframes* e *mockups*;
5. O quinto capítulo – **Implementação, Testes e Manual de Utilização** – é representado a implementação de cada tecnologia no trabalho realizado bem como testes a comprovar o desenvolvimento e funcionalidade da aplicação móvel sendo apresentado também um manual de utilização especificando todas as funcionalidades possíveis a utilizar;
6. O sexto capítulo – **Conclusão e Trabalho Futuro** – secção onde é refletido o trabalho desenvolvido no projeto e possíveis funcionalidades e otimizações a implementar no futuro.

Capítulo

2

Estado da Arte

2.1 Introdução

Nesta secção introduz-se o Estado da Arte do projeto referindo todas as tecnologias utilizadas no contexto do tema deste projeto bem como a motivação por cada escolha estabelecendo-se também uma descrição sobre cada ferramenta utilizada e a sua importância na realização deste projeto.

Cada uma das ferramentas utilizadas serão subdivididas nas seguintes secções:

- secção 2.2, referente ao CMS
- secção 2.3, referente à Strapi;
- secção 2.4, referente à *framework* Flutter.

2.2 CMS — *Content Management System*

2.2.1 Introdução

Um ***content management system***, em acrónimo **CMS** , é uma aplicação de software que permite aos utilizadores criar, editar, publicar e armazenar conteúdo digital.

Este tipo de aplicação é maioritariamente usada para gerir conteúdo empresarial e conteúdo World Wide Web (WEB). O conteúdo de um CMS encontra-se armazenado numa base de dados sendo exibido numa camada de apresentação num conjunto de *templates* como um *website* por exemplo. O CMS permite as seguintes funcionalidades:

- Criação de conteúdo, permitindo aos seus utilizadores criar e formatar conteúdo;
- Armazenar conteúdo, permitindo um armazenamento conciso dos dados;
- *Workflows*, permitindo uma boa escalabilidade em questões de papéis desempenhados por diversos responsáveis pelo conteúdo;
- Publicação, permitindo o *upload*, partilha e demonstração de conteúdos;
- Proteção de dados, permitindo um acesso seguro e controlado de conteúdos e informações essenciais que se encontram no *website*.

A descrição referida acima compõe a definição do que é um *content management system*, no entanto, no contexto do projeto é importante referir que existem dois tipos diferentes de CMS, sendo estes:

- **Traditional CMS:** *Content Management System* que conecta o *front-end* e o *back-end* de um website numa aplicação de código base, onde o CMS contém toda a estrutura do respetivo website desde a base de dados até ao conteúdo apresentado;
- **Headless CMS:** *Content Management System* que não conecta ao *front-end* de um *website*, sendo o seu principal propósito gerir o *back-end*. Um *headless CMS* tem como principal objetivo apenas lidar com o conteúdo apresentado num dado *website* não tendo qualquer tipo de influência no *front-end*.

2.2.2 Razão de Escolha/Motivação

O principal objetivo de um uso de um *content management system* é permitir que o conteúdo digital que é publicado seja o mais preciso, coerente e atualizado possível.

Uma das razões mais importantes associadas ao uso de um CMS é a permissão de inclusão e modificação de qualquer tipo de informação para os utilizadores sem estes necessitarem de conhecimento sobre HTML ou de linguagens de programação para alteração do *website*.

Para a realização deste projeto, escolheu-se utilizar o **Headless CMS Strapi**. Como referido na secção 2.2 existem 2 tipos principais de CMS: *Traditional CMS* e *Headless CMS* tendo sendo utilizado o *Headless CMS*. A escolha deste tipo de CMS para o trabalho é explicada pelas seguintes razões:

1. Com este projeto procura-se desenvolver uma aplicação móvel sendo a escolha mais indicada um *Headless CMS* visto que apenas queremos gerir o conteúdo de um dado *website* pois o *Traditional CMS* trata de gerir toda uma estrutura do mesmo.
2. Mais rápido que um *Traditional CMS*.
3. Facilidade de escalabilidade em diversas plataformas ao contrário do *Traditional CMS* que é utilizado para uma plataforma do tipo *website*.
4. Habilidade para gerir conteúdo em diversos canais tecnológicos.
5. Possui serviços de autenticação e protocolos de segurança.
6. Tempo de desenvolvimento reduzido (num *Traditional CMS* pode ser necessário um *rebuild* completo do sistema para alterar algum tipo de estrutura enquanto que um *Headless CMS* consegue alterar a organização sem ser necessário este custo).

O *Headless CMS* Strapi usado para o desenvolvimento deste projeto é abordado na secção 2.3.

2.3 Strapi - *Headless CMS*

2.3.1 Introdução

Strapi é um **Headless Content Management System** sendo usado no contexto deste projeto para gestão do *back-end* de um *website*. Esta tecnologia é relativamente recente, tendo apenas surgido a primeira versão estável para desenvolvedores no ano de 2020, estando ainda em desenvolvimento constante. Alguns casos de uso deste CMS são os seguintes:

- *Websites* estáticos;
- Aplicações Móveis.

No contexto deste projeto procura-se efetuar alterações no *website* com recurso a uma aplicação móvel que usa o Headless CMS Strapi.

Em suma, efetuaram-se operações de gestão de conteúdo pela Strapi através das suas API realizando operações *back-end*.

2.3.2 Razão de Escolha/Motivação:

Existem diversos CMS presentes no mercado, no entanto, aqui compara-se com um dos mais utilizados no mercado sendo neste caso o CMS WordPress:

Escolheu-se o CMS Strapi uma vez que além de servir o propósito inicial para este trabalho (gestão de *back-end*) possui diversas vantagens relativamente ao competidor mencionado acima.

Relativamente ao WordPress é importante referir que a Strapi foi construída com base em Node.js enquanto WordPress foi construído com base em HyperText Preprocessor (PHP). Ambas são tecnologias bastante importantes para *back-end development* no entanto no que diz respeito a este último tópico a Strapi possui vantagens.

1. Vantagens em comparação a WordPress :

- **Melhor código:** Node.js requer mais linhas de código para executar as mesmas funções que PHP no entanto em Node.js tudo é desenvolvido tendo em conta apenas uma linguagem JavaScript.

No que diz respeito a PHP, apesar de executar certas funções com menos linhas de código, é necessário que o *developer* tenha conhecimento de diversas tecnologias como: Linux, Apache HyperText Transfer Protocol (HTTP) Server, MySQL e o próprio PHP aumentando assim o nível de complexidade de desenvolvimento;

- **Melhor velocidade:** Node.js é um ambiente de desenvolvimento assíncrono, isto significa que consegue executar diversas tarefas de execução num dado intervalo de tempo de forma concorrente enquanto que o PHP é um ambiente de desenvolvimento síncrono o que significa que certas tarefas para serem executadas necessitam de esperar que outras acabem.

Este aspecto de sincronicidade e assincronicidade é muito importante em questões de velocidade de execução de tarefas, podendo concluir que Node.js tem tempos inferiores que PHP contribuindo num desenvolvimento mais rápido.

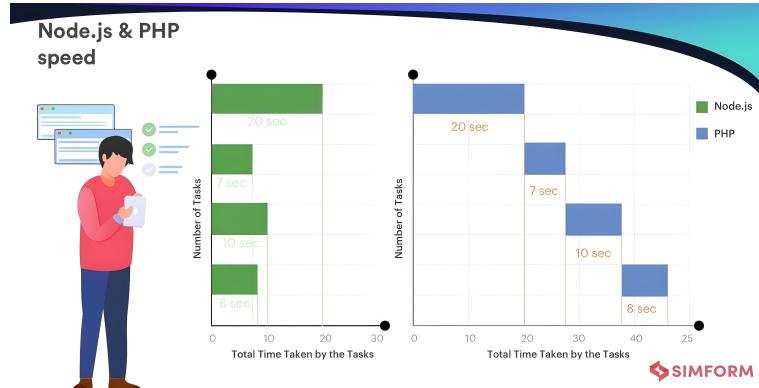


Figura 2.1: Tempo total de execução de tarefas : Node.js vs PHP imagem retirada de [1].

- **Melhor performance:** Node.js devido à sua natureza assíncrona consegue obter execução de módulos de forma concorrente isto significando que múltiplos módulos podem ser executados no mesmo intervalo de tempo (nunca de forma simultânea) e assim evitando que certos pedidos sejam bloqueados em função de outros. PHP bloqueia certos pedidos até um dado pedido estar inteiramente calculado abarcando uma natureza síncrona (certos pedidos a executar necessitam de esperar que um dado pedido acabe de ser processado). Em nível de *performance* pode-se concluir que Node.js é superior ao PHP devido à sua natureza assíncrona.

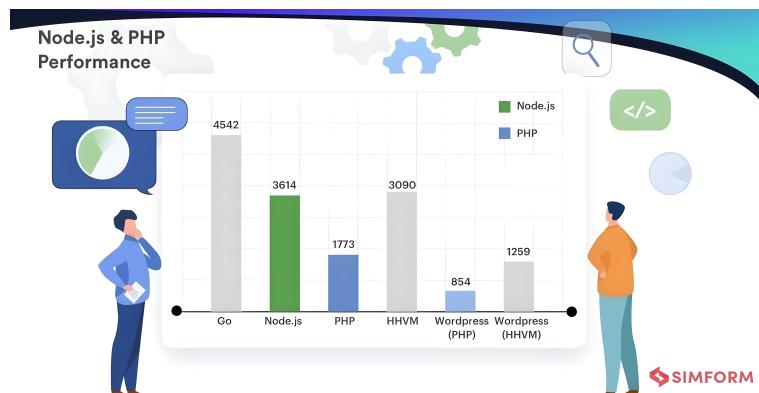


Figura 2.2: Número de pedidos em média que cada tecnologia consegue processar por segundo, imagem retirada de [1].

- **Menor uso de recursos:** Node.js processa vários pedidos num dado intervalo de tempo e devido a isto menos recursos são con-

sumidos ao contrário do PHP que processa um pedido de cada vez e por sua vez exige um maior consumo de recursos do que no caso anterior.

O WordPress foi escolhido de entre um conjunto de competidores do Strapi com o intuito de mostrar algumas diferenças que este CMS possui relativamente aos seus adversários.

2.4 Flutter *Framework*

2.4.1 Introdução

Flutter é um **User Interface (UI) toolkit** portátil, criado pela Google, com o principal objetivo de construir aplicações nativas compiladas para telemóveis, *WEB* e *desktop* a partir de um *codebase*. É uma *framework* híbrida permitindo desenvolver *apps* para Android e iOS e usa a linguagem Dart.

Flutter é totalmente *open-source* e utiliza um *engine* de renderização 2D, conhecido como Skia, para criar visuais nas aplicações móveis e serve um conjunto de propósitos diversificado tendo em conta quem o utiliza:

- **Developers:** construir aplicações móveis;
- **Designers:** fornecimento de *canvas* para atividades de *design*;
- **Engineering managers e Empresas:** permitir a unificação de desenvolvedores de aplicações em equipas de diversas vertentes (telemóvel, *WEB*, *desktop*) para desenvolvimento de aplicações para várias plataformas.

2.4.2 Razão de Escolha/Motivação:

Existem várias *frameworks* que se focam no desenvolvimento de aplicações móveis como, por exemplo:

1. Swiftic: focada em desenvolvimento de aplicações para plataforma iOS;
2. React Native: com base em linguagem JavaScript;
3. Xamarin: com base em linguagem C#.

No contexto deste projeto decidiu-se escolher a *framework* Flutter devido à recente popularidade e atualizações contínuas tanto em documentação como por parte de *developers* e comunidade que utiliza esta *framework*. Possui algumas vantagens notórias em relação aos seus competidores das quais se destacam:

1. **Sistema de renderização 2D único**: o que torna o Flutter único relativamente a outras *frameworks* no mercado é o *engine* de renderização 2D, conhecido como Skia, para visuais das aplicações móveis desenvolvidas nesta *framework*.

O Flutter trata de renderizar, utilizando esta *engine*, os seus componentes *UI* ao contrário dos seus adversários que utilizam componentes de *UI* já existentes do Android e iOS. Os componentes renderizados pelo Skia possuem mais opções de customização do que os referidos anteriormente.

2. **Linguagem de programação Dart**: o Flutter usa Dart como linguagem de programação. Esta linguagem, criada pela Google e usada no Flutter unicamente para *app development*, possui muitas vantagens das quais se destacam:

- **Dart é baseado em programação orientada a objetos** onde componentes da aplicação são convertidos em objetos reutilizáveis, onde cada um dos objetos pode herdar funcionalidades e características de objetos pais permitindo uma reutilização de funções evitando complexidade no código;
- **Compiladores da linguagem são rápidos e fiáveis**: Dart oferece compilações *ahead-of-time*, que compilam automaticamente o código num formato em que a máquina nativa pode facilmente ler e permite que uma aplicação consiga executar ficheiros binários. Ao

compilar o código durante o *build process*, programadores conseguem obter renderização mais rápida ao nível da *UI* em *browsers*. Além do mais para aplicações que executam processos de forma concorrente, Dart reserva certas partes de memória de forma a assegurar múltiplas computações de forma estável e concisa.

- Dart possui uma *syntax* limpa e que previne erros de escrita em nível de código;
- Dart possui uma comunidade bastante aberta com documentação sempre atualizada.

2.5 Conclusão

Nesta secção 2 referiu-se todas as tecnologias utilizadas para a realização deste projeto tendo em conta algumas das suas características e diferenças relativamente a principais competidores de mercado com o intuito de esclarecer as escolhas tomadas no que diz respeito ao desenvolvimento deste projeto.

Capítulo

3

Tecnologias e Ferramentas Utilizadas

3.1 Introdução

Nesta secção estão apresentadas, de uma forma mais detalhada, especificações sobre as tecnologias e plataformas utilizadas para a realização deste projeto. De acordo com as tecnologias referidas na secção 2, neste capítulo é apresentado uma vista mais aprofundada sobre algumas das suas particularidades e características mais relevantes bem como estruturação das mesmas.

3.2 Strapi - Headless Content Management System

3.2.1 Especificação de um Headless Content Management System

Um Headless CMS como a Strapi utiliza uma *Headless Architecture*. Esta arquitetura permite uma publicação em diversos canais tecnológicos.

Atualmente utilizadores procuram uma experiência perfeita em diversas plataformas e equipamentos sendo esta procura respondida pela adoção deste tipo de arquitetura permitindo que diversos sistemas trabalhem em conjunto sem entraves.

■ Como funciona este tipo de arquitetura?

Uma Headless Architecture não tem em conta o *front-end* de um sistema, mas sim o *back-end* do mesmo. O que interessa neste caso é o conteúdo, como criar e modificar o mesmo em que o conteúdo *raw* pode ser publicado por uma *framework* e acomodado num equipamento em qualquer lugar. Para alcançar isto o conteúdo terá que ser redirecionado, por um serviço *web* ou por uma API, para o *front-end* do sistema onde estes mecanismos são os responsáveis por recolher o conteúdo e convertê-lo em material visual.

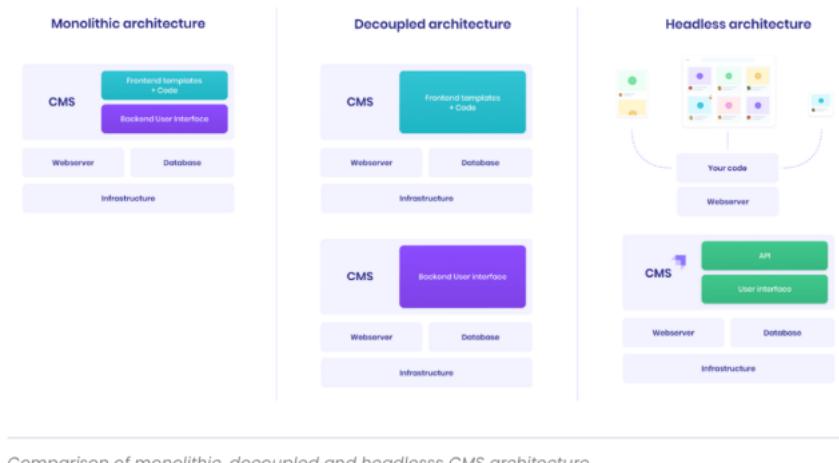


Figura 3.1: Vários tipos de arquitetura de Content Management System (CMS), imagem retirada de [2].

■ Benefícios de utilização de uma Headless Architecture:

- **Liberdade:** a utilização de um CMS que adota esta arquitetura não se encontra limitado em termos de ferramentas dando imensa liberdade aos *developers* na escolha de uma *front-end framework* e serviços;
- **Escabilidade:** uma Headless Architecture é customizável e escalável;
- **Ágil:** permitindo um desenvolvimento conciso e rápido com custo reduzido.

3.2.2 Gestão de conteúdo

Tal como referido na secção 2.3, a Strapi permite uma gestão do *back-end* de um dado *website* tratando da gestão do conteúdo. A *Strapi* utiliza por defeito, o **SQLite** para armazenamento de conteúdo tendo sido este utilizado na realização deste projeto.

O conteúdo é criado pela utilização de um *plugin* que a Strapi possui sendo o **Content-Type Builder** onde este trata de modelar a estrutura de dados da API.

Este *plugin* é uma funcionalidade inerente à Strapi que está sempre ativa por defeito e que não pode ser desativada. Este *plugin* disponibiliza 3 categorias:

- **Collection Types:** Possibilita a gestão de várias *entries*;
- **Single Types:** Possibilita a gestão de apenas uma *entry*;
- **Component:** Estruturas de dados que podem ser usadas em múltiplos *Collection Types* e *Single Types*.

Após a criação de um **Content Type** (*Collection Type* ou *Single Type*), será possível acrescentar diversos campos neste *Content Type* para compor o conteúdo a criar. Seguidamente encontram-se alguns exemplos criados pelo uso da *interface* da Strapi.

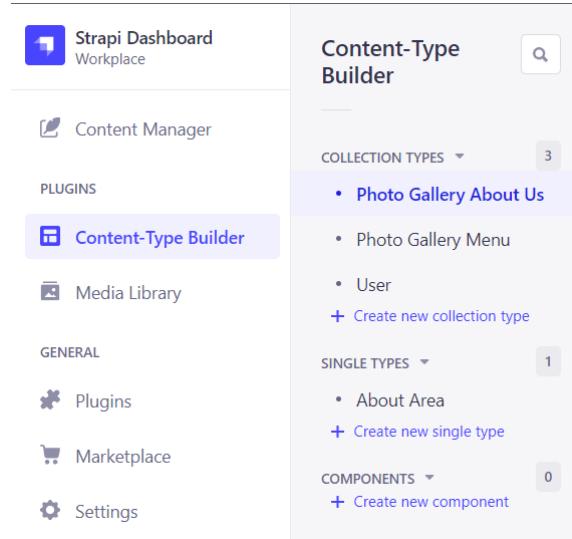
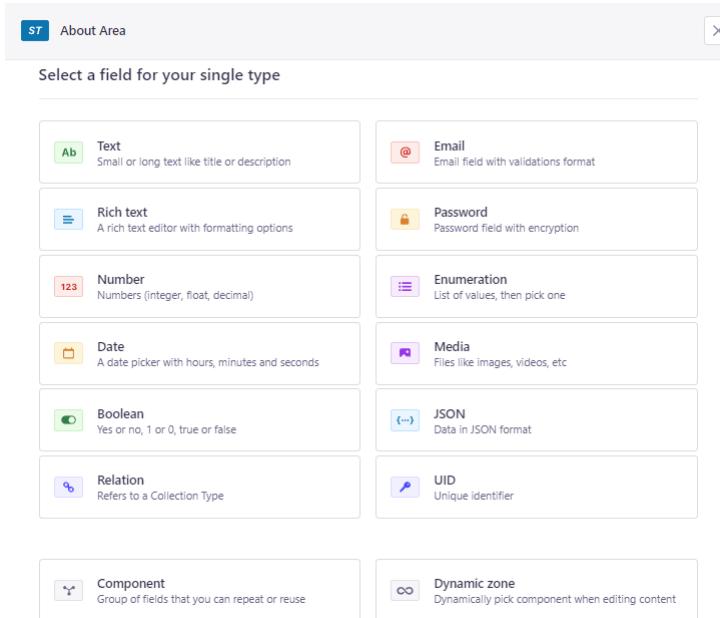


Figura 3.2: *Content-Type Builder* na interface da Strapi.

Figura 3.3: Criar um *Single Type*.

Figura 3.4: Opções de campos a criar no *Single Type*.

NAME	TYPE	
title	Text	
firstParagraph	Text	
aboutImage	Media	
secondParagraph	Text	

Figura 3.5: *Single Type* criado anteriormente com os campos adicionados.

Após a criação do *Collection Type* ou *Single Type*, é necessário gerir conteúdo através do **Content Manager** da Strapi. O **Content Manager** é um *plugin* essencial da Strapi que está ativo por defeito e que não pode ser desativado. Disponibiliza 2 categorias sendo estas: *Collection Types* e *Single Types*.

É importante referir que na Strapi a gestão de conteúdo é apenas permitida a **utilizadores que tenham as permissões necessárias** sendo esta parte referida na secção 3.2.4.

Cada categoria contém um *Collection Type* ou *Single Type* que foram criados no *plugin Content-Type Builder* mencionado anteriormente e exemplificado pelas figuras 3.2, 3.3, 3.4, 3.5. Para um *Single Type* apenas uma *entry* pode ser editada. No *Content Manager* podem ser efetuados diversos tipos de gestão de conteúdo no *single type* criado.

Em 3.6, verifica-se um exemplo de um *single type* com conteúdo já criado.

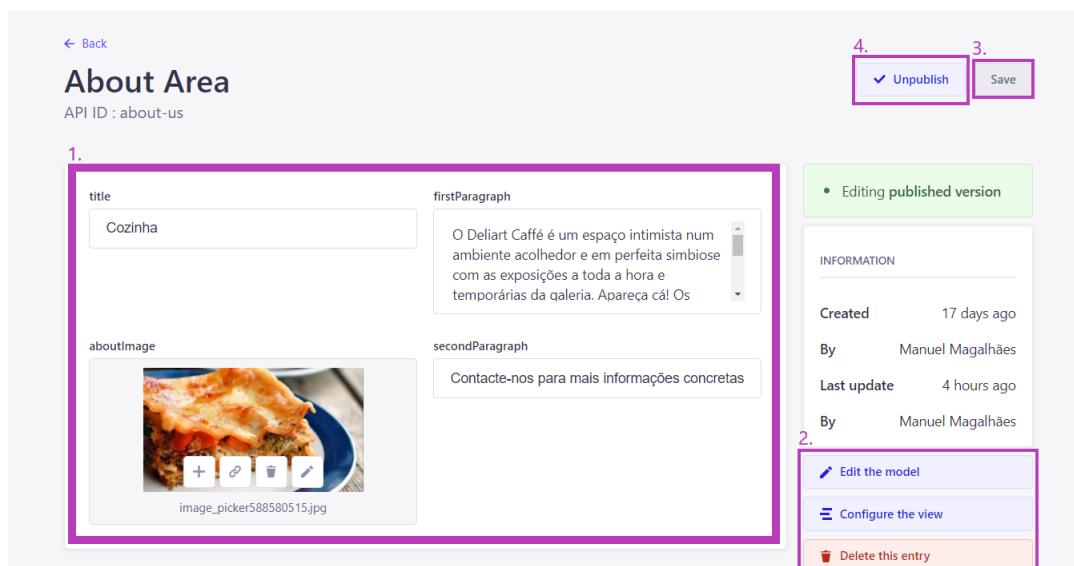


Figura 3.6: Opções de operação num *single type* no *Content Manager*.

- **Operação 1:** os campos criados no *single type* podem ser alterados via interface gráfica após a criação do *single type*;
- **Operação 2:** o modelo com que o *single type* foi criado (campos definidos) pode ser editado. A visualização dos campos pode ser alterada (mudança de *view*). Caso o utilizador deseje pode eliminar a *entry* do *single type* criado;
- **Operação 3:** após alterações efetuadas nos campos do *single type*, esta operação permite o armazenamento das alterações;

- **Operação 4:** após a criação deste *single type* este fica com o estado de ***draft***. Para que este tipo de conteúdo possa ser utilizado, deve ficar com um estado de ***published*** sendo isto possível com esta operação.

Consoante o ponto anterior, mais concretamente na operação 4, um conteúdo em estado ***draft*** não consegue ser obtido no futuro através de um API Request. Para a implementação de gestão de conteúdo deste projeto, todos os *types* criados possuem *entries* com estado de ***published***.

3.2.3 REST API

A REST API é uma API que a Strapi utiliza para permitir o acesso a elementos Content-Types mencionados na secção 3.2.2. Possui parâmetros que são usados para **filtrar, organizar e paginar** resultados e para selecionar **campos**. A criação de um Content-Type cria automaticamente *REST API Endpoints* (pontos de contacto da comunicação entre o sistema e a API).

Estes **API Endpoints** são utilizados para gerir os campos do Content-Type e estão subdivididos nos seguintes métodos para Collection Type e Single Type:

Method	URL	Description
GET	/api/:pluralApiId	Obter uma lista de <i>entries</i>
POST	/api/:pluralApiId	Criar uma <i>entry</i>
GET	/api/:pluralApiId/:documentId	Obter uma <i>entry</i>
PUT	/api/:pluralApiId/:documentId	Atualizar uma <i>entry</i>
DELETE	/api/:pluralApiId/:documentId	Remover uma <i>entry</i>

Tabela 3.1: REST API Endpoints - Collection Type.

Method	URL	Description
GET	/api/:singularApiId	Obter uma <i>entry</i>
PUT	/api/:singularApiId	Atualizar/Criar uma <i>entry</i>
DELETE	/api/:singularApiId	Remover uma <i>entry</i>

Tabela 3.2: REST API Endpoints - Single Type.

A partir da utilização da REST API neste trabalho, será possível gerir o conteúdo que o utilizador deseja **criar, atualizar ou remover** do seu *website*. Será feito uma operação de tratamento dos dados através dos API Endpoints mencionados nas tabelas 3.1 e 3.2 pela operação de *API Requests*.

No contexto das *API Requests*, efetua-se um *request* à API, via Uniform Resource Locator (URL), atribuída ao *Single Type* definido em 3.6. Uma das plataformas utilizadas neste trabalho para testar a **REST API** e consequentes *API Requests* é a plataforma **Postman**[4].

3.2.4 Utilizadores, Papéis e Permissões

Algumas funcionalidades bem como o conteúdo que a Strapi gera obedecem a um sistema de permissões. Estas permissões podem ser atribuídas a *roles*, que estão associados aos utilizadores que utilizam a Strapi para gestão do conteúdo criado. O exemplo mencionado em 3.6 obedece a um sistema de permissões.

Neste exemplo, ao *Single Type* foi atribuído um API ID para que o conteúdo criado pudesse ser acedido via *API Request*. Na Strapi todas as API's criadas estão protegidas por defeito sendo necessário atribuir um certo conjunto de permissões (para acesso, por exemplo) aos dados que a API abrange. O acesso aos dados do *Single Type* só foi possível pela atribuição da *role find*.

Por defeito, a Strapi possui 2 papéis principais para os utilizadores nomeadamente **Public** e **Authenticated**.

Pode-se atribuir certas permissões a utilizadores autenticados e a utilizadores públicos para restringir o acesso às API's e consecutivamente restringir a gestão de conteúdos da Strapi.

The screenshot shows the Strapi 'Settings' interface. On the left sidebar, under 'GLOBAL SETTINGS', 'About-us' is selected. Under 'ADMINISTRATION PANEL', 'Roles' is selected. Under 'EMAIL PLUGIN', 'Configuration' is selected. Under 'USERS & PERMISSIONS PLUGIN', 'Roles' is selected. The main content area is titled 'Public' and describes it as the 'Default role given to unauthenticated user'. It shows the 'Permissions' section with the note 'Only actions bound by a route are listed below'. There are two sections: 'About-us' and 'Photo-gallery-about-us'. The 'About-us' section has a 'Select all' checkbox and three checked action checkboxes: 'delete', 'find', and 'update'. The 'Photo-gallery-about-us' section also has a 'Select all' checkbox and three checked action checkboxes: 'delete', 'find', and 'update'.

Figura 3.7: Atribuição de permissões de acesso a utilizadores públicos para a API do Single Type.

3.2.5 Envio de *Emails*

A *Strapi* consegue utilizar *email providers* para envio de *emails* para os seus utilizadores para questões relativas a acesso e alteração de credenciais de conta. Para uma das funcionalidades deste projeto, mais concretamente para a funcionalidade de *reset password* de um dado utilizador, foi utilizado um *Simple Mail Transfer Protocol (SMTP) provider* sendo o **Sendinblue** a escolha para o envio de *emails*.

Foi necessário alterar algumas definições no projeto local da *Strapi*, mais concretamente na pasta **config** onde se especificou dados essenciais como o tipo de *provider*, *host* e *port* do *SMTP provider* fornecidos pela criação de uma conta externa neste serviço (estas configurações podem ser visualizadas no trecho de código 3.1). A conta atual utilizada para o projeto permite o envio de 300 emails diários a utilizadores da *Strapi*.

```
module.exports = ({ env }) => ({
  // ...
  email: {
    config: {
      provider: 'nodemailer',
      providerOptions: {
        host: env('SMTP_HOST', 'smtp.example.com'),
        port: env('SMTP_PORT', 587),
        auth: {
          user: env('SMTP_USERNAME'),
          pass: env('SMTP_PASSWORD'),
        },
        // ... any custom nodemailer options
      },
      settings: {
        defaultFrom: 'no-reply@strapi.io',
        defaultReplyTo: 'no-reply@strapi.io',
      },
    },
    // ...
  });
});
```

Exerto de Código 3.1: Ficheiro *plugin* para especificação de *email provider*.

Tal como verificado em 3.1, pode-se verificar que a definição de alguns atributos é realizada pela chamada de dado elemento **env**. Este elemento é um ficheiro de configuração que a *Strapi* possui por defeito sendo utilizado para armazenar alguns conteúdos para depois serem utilizados em diversos ficheiros da pasta local do projeto. Este ficheiro de configuração é bastante útil permitindo maior facilidade na alteração de dados críticos ao funcionamento do *Headless CMS*, podendo-se visualizar no trecho de código 3.2 que se utilizou este ficheiro para agrupar dados do **SMTP provider** como o *host*, *port*, *username* e *password* da conta *Sendinblue*.

```
HOST=0.0.0.0
PORT=1337
APP_KEYS=039xTg8ORijArRZsIOj2Kg==,gDJzxoAdEPah+HmMZldsg==,WbKjP15wlKoc7n3S6THa9A==,
         R9hrR70uCm/6dWt6EDFWGg==
JWT_SECRET=f69c64c2-166a-43db-81b7-4f088bf1db7c
API_TOKEN_SALT=077325f9141940b66f0a0106b5190915
SMTP_HOST=smtp-relay.sendinblue.com
SMTP_PORT=587
SMTP_USERNAME=mferreira.magalhaes123@gmail.com
SMTP_PASSWORD=I3GVkmjWH2Sc4CTB
```

Excerto de Código 3.2: Ficheiro de configuração *env*.

3.3 Flutter - Framework de Desenvolvimento de Aplicações Móveis

3.3.1 Especificação acerca do Flutter:

Flutter é um *UI toolkit open source* criado pela Google para desenvolvimento de aplicações móveis. Sendo uma *framework* com objetivo de desenvolver *apps*, o seu *Software Development Kit (SDK)* contém um conjunto de *features* diversificado:

- Framework otimizada, com um sistema de renderização 2D único;
- Quantidade variada e rica de *widgets*;
- *API's* para testes de unidade e integração;
- Interoperabilidade e *plugin API's* que conectam ao sistema e *SDK's* de terceiros;
- Sistema de testagem para correr testes em Windows, Linux e Mac;
- Ferramentas disponibilizadas por *developers* para testagem, *debugging* e análise da aplicação;
- Ferramentas na linha de comando para criação, construção, testagem e compilação das aplicações móveis.

O Flutter é construído com as linguagens C, C++, Dart e com o sistema de renderização 2D Skia.

1. Como o Flutter executa o código em dispositivos Android?

Código C e C++ são compilados com o *Native Development Kit (NDK)*. O código Dart é compilado *ahead-of-time* em bibliotecas x86, Advanced RISC Machines (ARM) e nativas.

Estas bibliotecas são incluídas no *runner* do projeto de Android sendo todo o processo construído numa extensão *Android Application Package (APK)*. Quando a aplicação é lançada, esta carrega a biblioteca Flutter e qualquer renderização, *input* ou tratamento de eventos e por aí diante é delegado para o código compilado Flutter e da *app*.

2. Como o FLutter executa código em dispositivos iOS?

Código C e C++ são compilados com *Low Level Virtual Machine (LLVM)*. O código Dart é compilado *ahead-of-time* em bibliotecas nativas e ARM.

Estas bibliotecas são incluídas no *runner* do projeto iOS sendo todo o processo construído numa extensão *iPhone Application Archive (IPA)*. Quando a aplicação é lançada, esta carrega a biblioteca Flutter e qualquer renderização, *input* ou tratamento de eventos e por aí diante é delegado para o código compilado Flutter e da aplicação.

Relativamente à **utilização de widgets**, o Flutter fornece um conjunto de **widgets** incluindo Material Design e Cupertino (*widgets* estilo iOS) sendo estes geridos e renderizados pela *framework* do Flutter e o sistema de renderização 2D Skia. Os *widgets* são facilmente customizáveis, oferecendo um leque de escolhas de reutilização e de combinação para a criação de novos *widgets*.

Os sistemas operativos onde se consegue construir uma aplicação FLutter são: Linux, macOS, ChromeOS e Windows.



Figura 3.8: Estrutura da *framework* Flutter, imagem retirada de [3].

3.4 Conclusão

Neste capítulo, referiu-se de uma forma mais detalhada e precisa características e aspetos relevantes das principais tecnologias utilizadas para a realização deste trabalho. Para concluir é importante constatar que a gestão de conteúdo, pela Strapi, será gerida na sua completude pela aplicação móvel desenvolvida no contexto deste projeto.

Capítulo

4

Engenharia de Software

4.1 Introdução

Neste capítulo é apresentado uma parte dedicada a Engenharia de *Software* que se subdivide nas seguintes secções:

- **Secção 4.2**, apresentação dos requisitos funcionais e não funcionais do sistema;
- **Secção 4.3**, retrato dos casos de uso respetivos ao sistema desenvolvido;
- **Secção 4.4**, exibição dos diagramas de atividade;
- **Secção 4.5**, designando a arquitetura do sistema desenvolvido;
- **Secção 4.6**, referência à base de dados do sistema;
- **Secção 4.7**, apresentação dos *wireframes* representativos da aplicação;
- **Secção 4.8**, apresentação dos *mockups* representativos da aplicação.

4.2 Engenharia de Requisitos

A engenharia de requisitos é um processo que permite estabelecer os serviços que um dado utilizador requer do sistema bem como as suas restrições e como este é desenvolvido. Os requisitos subdividem-se em requisitos funcionais e requisitos não funcionais.

Os requisitos funcionais são declarações dos serviços que o sistema deve fornecer para um dado utilizador, como o sistema se comporta em situação de uso particulares e como este responde a determinados *inputs* do utilizador. Os requisitos não funcionais são restrições dos serviços que o sistema fornece, estas restrições podem ser restrições em nível de processamento ou em nível temporal, por exemplo. Estes requisitos focam-se na descrição geral do sistema do que em descrições detalhadas de serviços particulares que o sistema deve fornecer ao utilizador.

Para o sistema desenvolvido no contexto deste projeto, foram definidos os seguintes requisitos funcionais de sistema:

1. Interface com sistema de *login* para permitir ao utilizador o *login* pela Strapi;
2. Opção para poder efetuar uma operação de *logout* da conta Strapi;
3. Opção para poder efetuar *upload* de imagens para a Strapi;
4. Opção para poder realizar *upload* texto para a Strapi;
5. Opção de confirmação de upload;
6. Opção de cancelamento de upload;
7. Base de dados que compreenda os dados do conteúdo;
8. Conexão às *API's* da Strapi;
9. Registo de conta da Strapi;
10. Alteração de credenciais Strapi;
11. Estabelecimento de um fornecedor de serviço SMTP para envio de *emails* para utilizadores.

Os respetivos requisitos não funcionais definidos para este sistema são os seguintes:

1. **Eficiência:** o sistema deve ser eficiente em espaço, tempo, em termos computacionais e em termos de utilização de energia podendo efetuar as operações essenciais sem gasto desnecessário de recursos;
2. **Conexão:** o sistema deve conseguir efetuar conexões à Strapi em poucos segundos;
3. **Segurança:** o sistema deve conter protocolos de segurança que garantam conexões seguras à Strapi;
4. **Segurança:** o sistema deve fornecer protocolos de segurança que garantam a segurança das credenciais da conta do utilizador;
5. **Gestão de erros:** o sistema deve informar o utilizador em caso de aparecimento de erros aquando da utilização da aplicação;
6. **Atual:** o sistema deve correr na versão mais atual do sistema operativo a utilizar. A aplicação deve correr tanto em dispositivos Android e iOS;
7. **Fiabilidade:** o sistema deve conter mecanismos que garantam a sua fiabilidade e o *downtime* da aplicação ou dos serviços que esta fornece deve ser o mais baixo possível;
8. **Organização:** o sistema deve estar organizado de maneira simples e ser de fácil utilização, para minimizar erros;
9. **Disponibilidade:** o sistema deve estar disponível durante 24h por dia;
10. **Taxa de Erros:** o sistema deve providenciar métodos para reduzir a taxa de erros do utilizador, fornecendo avisos a qualquer erro causado.

4.3 Diagramas de Casos de Uso

Caso de uso é uma forma de representar as interações entre um dado sistema e o seu utilizador. Os casos de usos são representados neste projeto por diagramas. Estes diagramas representam todas as interações possíveis descritas nos requisitos de sistema. Estes diagramas são compostos por:

- **Atores no processo:** são entidades humanas ou não humanas (sistemas, por exemplo) sendo representados sob a forma de uma pessoa;
- **Interações:** representam todas as possíveis ações dos utilizadores correspondentes ao respetivo diagrama. Estas interações são representadas sob a forma de elipses;
- **Linhas de interação:** as linhas são usadas para relacionar atores e interações. Estas linhas podem ter uma forma de uma seta para representar quem começou a interação.

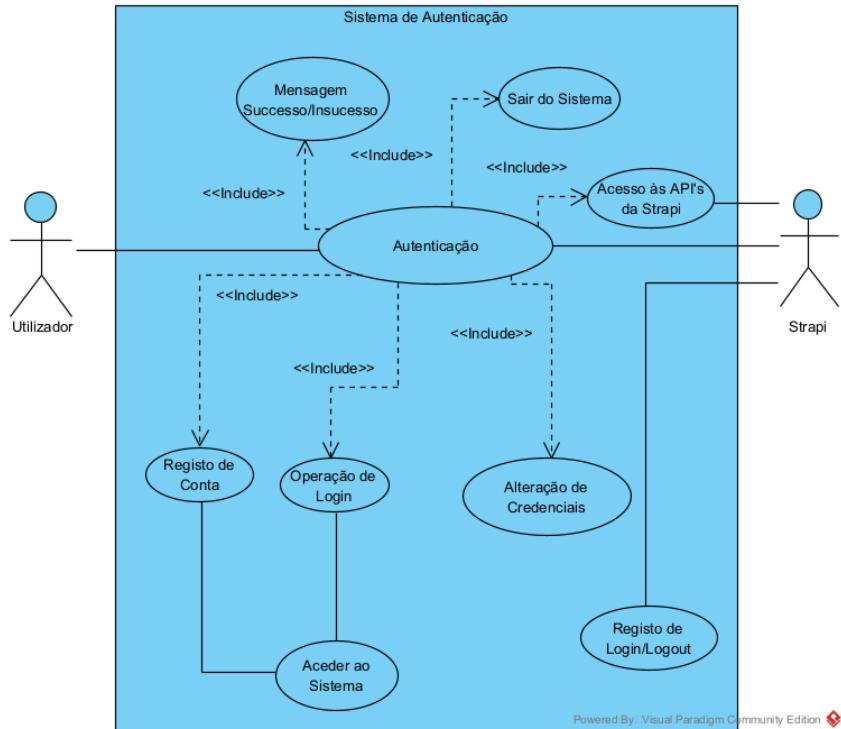


Figura 4.1: Diagrama de caso de uso relativo ao sistema de autenticação.

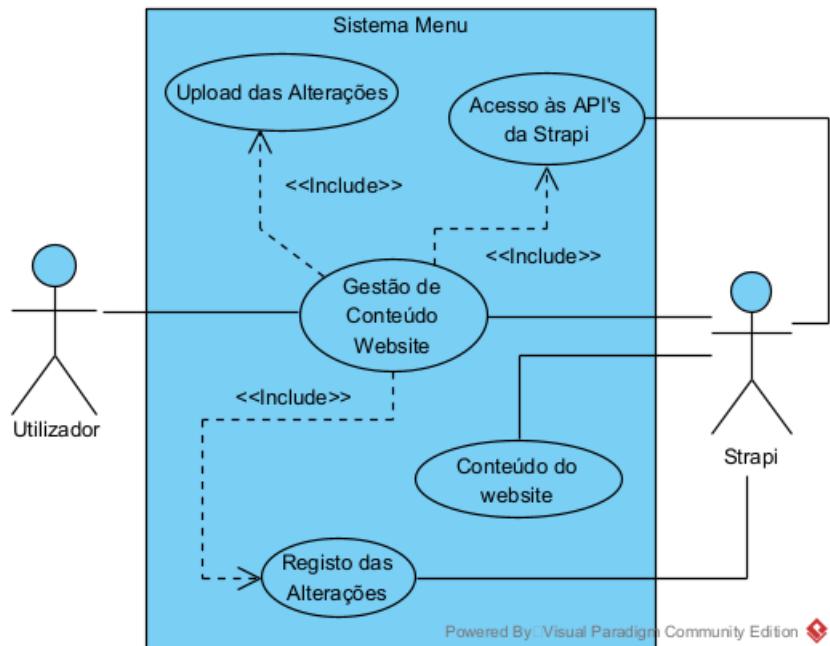


Figura 4.2: Diagrama de caso de uso relativo ao menu principal de utilizador.

4.4 Diagramas de Atividade

Os diagramas de atividade são diagramas utilizados com o principal objetivo de evidenciar as atividades envolvidas no processo entre o utilizador e o sistema. Focam-se sobretudo em demonstrar possíveis ações que o utilizador possa efetuar mediante a utilização do sistema.

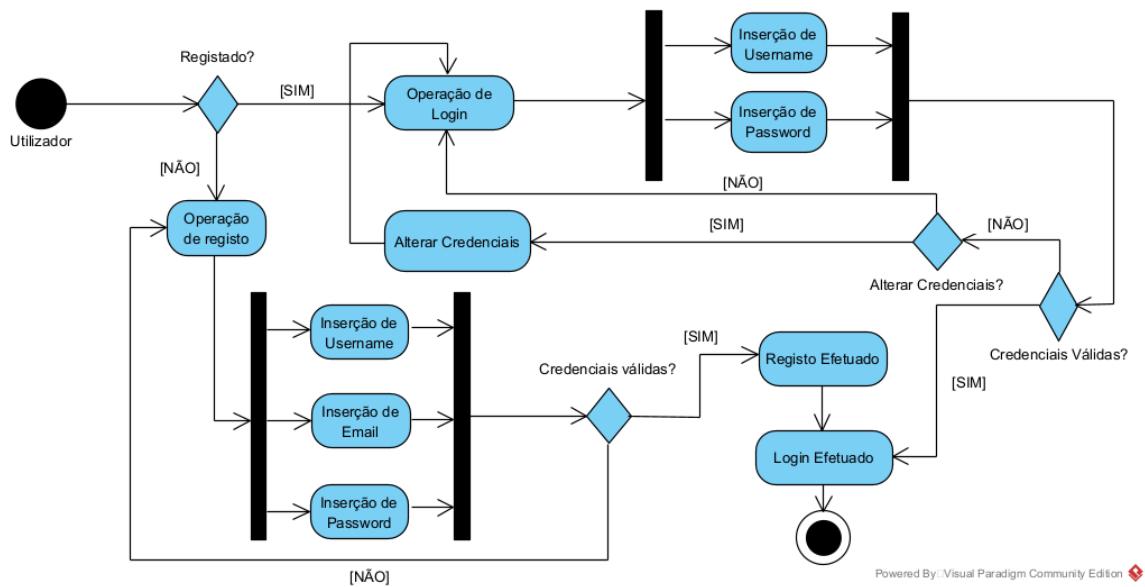


Figura 4.3: Diagrama de atividade relativo ao sistema de autenticação.

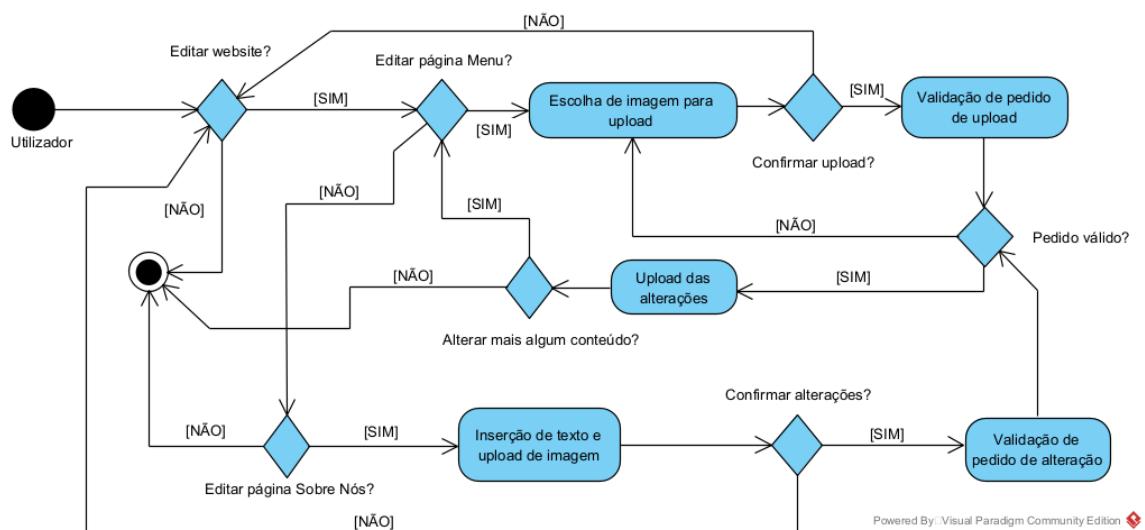


Figura 4.4: Diagrama de atividade relativo ao menu principal de utilizador.

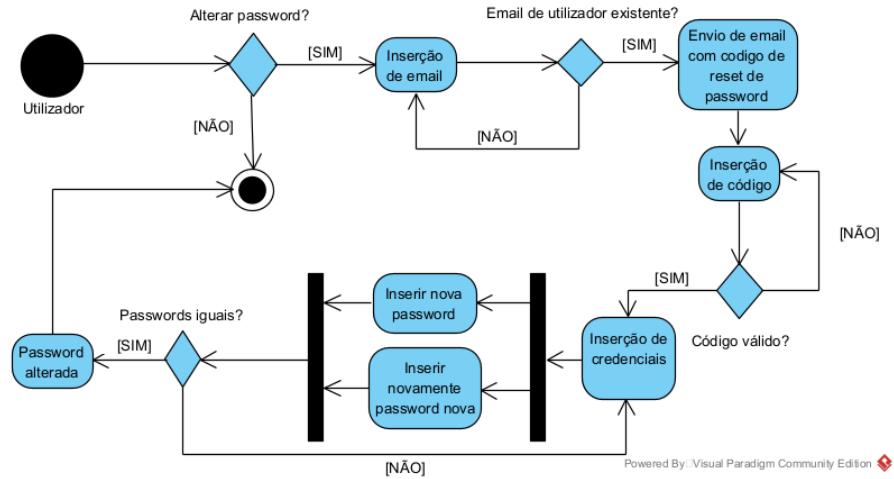


Figura 4.5: Diagrama de atividade relativo à alteração de credenciais.

4.5 Arquitetura de Sistema

A arquitetura do sistema compreende os diversos componentes que o constituem e as suas relações entre eles, permitindo estabelecer um *blueprint* do sistema. Para definir a arquitetura do sistema deste projeto foi elaborado um diagrama de implementação que visa agrupar todos os elementos que o sistema abrange.

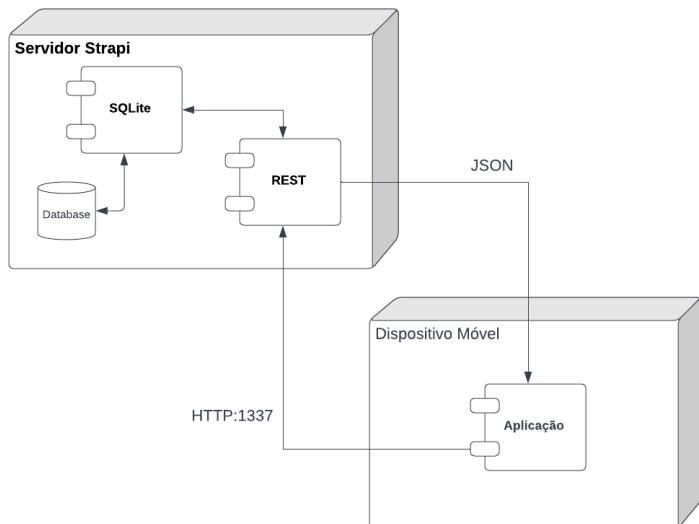


Figura 4.6: Arquitetura do Sistema.

Deste diagrama é importante referir que a aplicação consome a REST API da Strapi de acordo com certas funcionalidades que o utilizador escolha efetuar no contexto da aplicação. Este consumo é efetuado via um HTTP onde o servidor da Strapi se encontra. O servidor está configurado no **localhost** na **porta 1337**.

A partir dos pedidos efetuados pelo utilizador, através da REST API, alguns dados vão ser manipulados conforme a especificação da REST API tendo em conta a funcionalidade que o utilizador efetuou.

Após esta manipulação de dados, o servidor irá retornar informação à aplicação via *JavaScript Object Notation (JSON)* para posteriormente este ser tratado segundo os objetivos das funcionalidades da aplicação.

4.6 Base de Dados

O principal objetivo deste projeto, tal como referido na secção 1.3, consiste no desenvolvimento de uma aplicação móvel para gestão de conteúdo no *Headless CMS* Strapi. Para colocar esta secção 4.6 em contexto, utilizou-se a Strapi como único agente de gestão de conteúdo e para tal é necessária uma base de dados que compreenda o conteúdo a gerir. Por defeito, a Strapi utiliza uma base de dados SQLite para armazenamento do conteúdo que esta gere, tendo sido esta a base de dados utilizada para a realização deste trabalho.

A base de dados comprehende diversas tabelas de sistema elaboradas por defeito pelo *Headless CMS*. Das 21 tabelas que compõem a base de dados, apenas 3 delas foram criadas de raiz para gestão de conteúdo. As tabelas criadas subdividem-se nas seguintes:

- ***photo_gallery_about_uses*** e ***photo_gallery_menus***: estas tabelas armazem e gerem conteúdo em nível de imagens;
- ***about_uses***: esta tabela armazena e gera conteúdo em nível de imagem e texto.

As tabelas referidas em cima foram elaboradas especificamente tendo em conta o conteúdo de um *website* utilizado para posteriores testes usando a aplicação desenvolvida na *framework Flutter*. Esta questão é elaborada com mais detalhe na secção 5.4.

As restantes tabelas, criadas por defeito pela Strapi, são utilizadas para diversos meios inerentes ao sistema. Alguns dos exemplos mais importantes são:

- ***up_users***: responsável pelo armazenamento de utilizadores da Strapi;
- ***up_roles***: responsável pelo armazenamento de diversos papéis que utilizadores podem ter no sistema.
- ***up_users_role_links***: tabela intermediária entre ***up_users*** e ***up_roles*** que associa os utilizadores aos seus respetivos papéis que desempenham;
- ***files***: responsável pela formulação de *media types* que foram utilizados como campos para *Collection Types* e *Single Types* para inserção de imagens. Um exemplo concreto onde esta tabela é usada pode ser visualizado na figura 3.6 onde a imagem do *Single Type* é armazenada nesta mesma tabela.

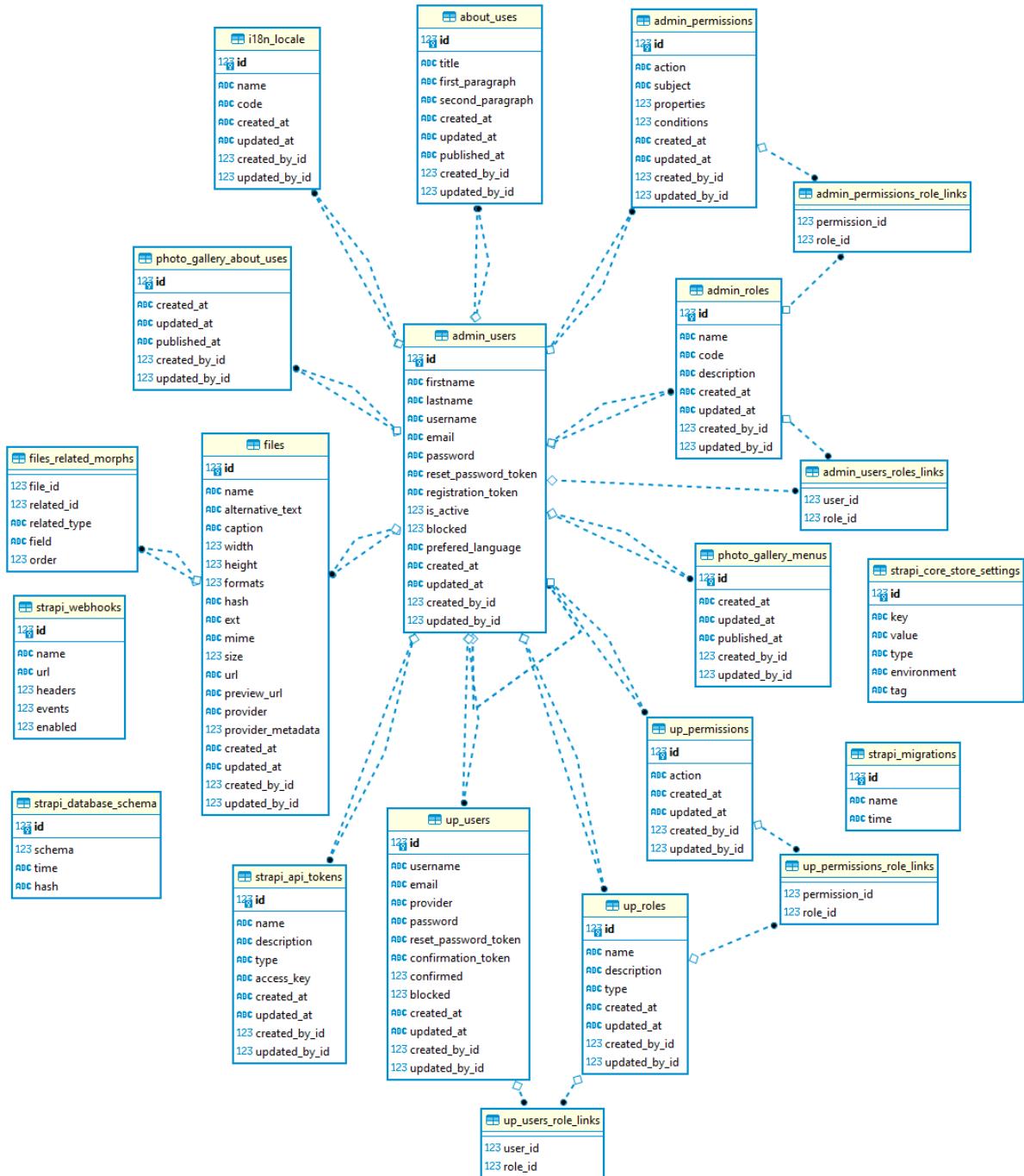


Figura 4.7: Modelo relacional da base de dados da Strapi.

4.7 Wireframes

Wireframes são *blueprints* que ilustram o design básico da interface do utilizador. Estes têm como principal objetivo comunicar facilmente os conteúdos da página, a estrutura da mesma, *layout* e funções associadas. Neste contexto foram elaborados um conjunto de *wireframes* para ilustrar o design básico da aplicação.

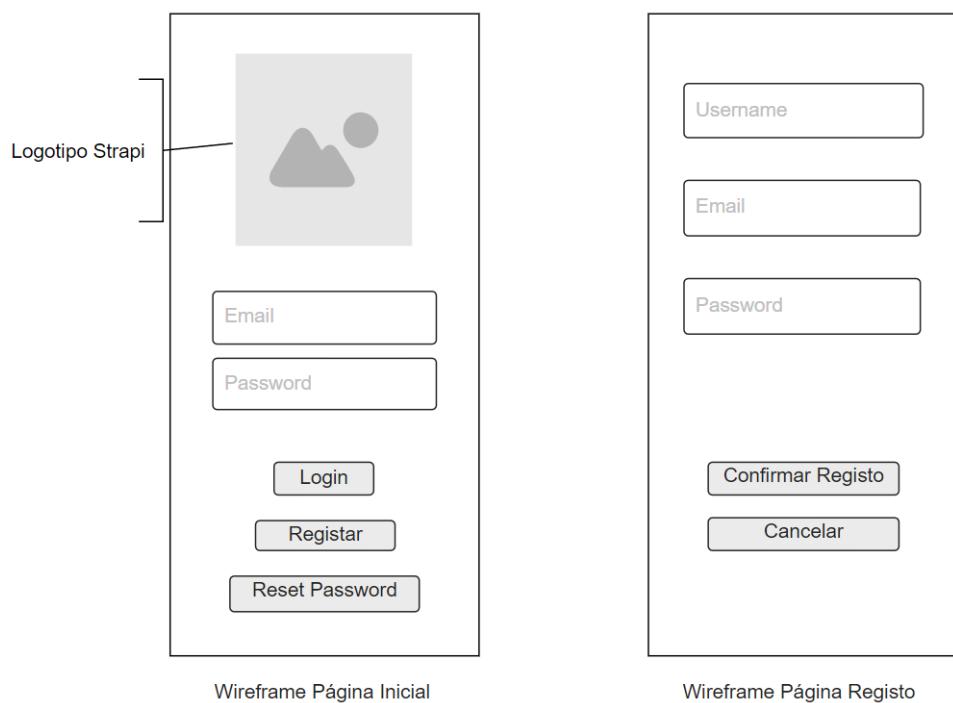


Figura 4.8: *Wireframe* de Página Inicial e de Página Registo.

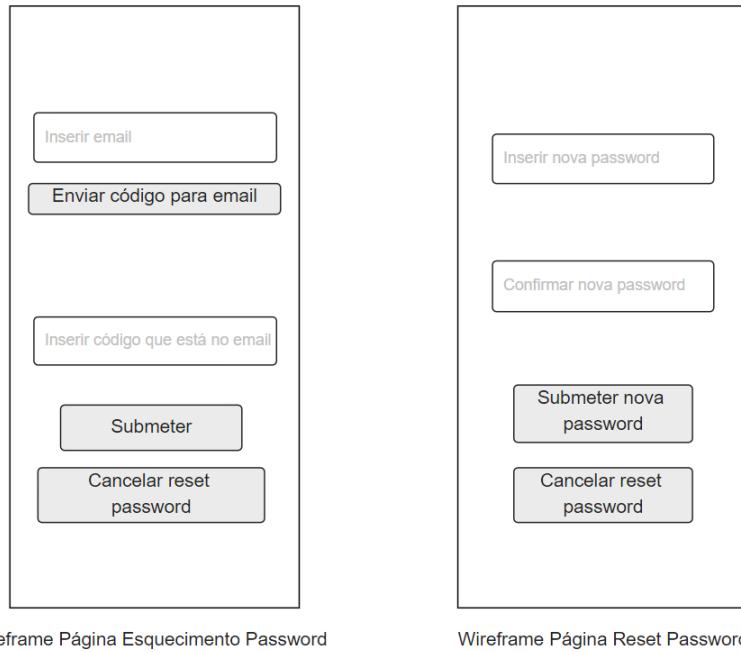


Figura 4.9: *Wireframe* de Página Esquecimento Password e de Página Reset Password.

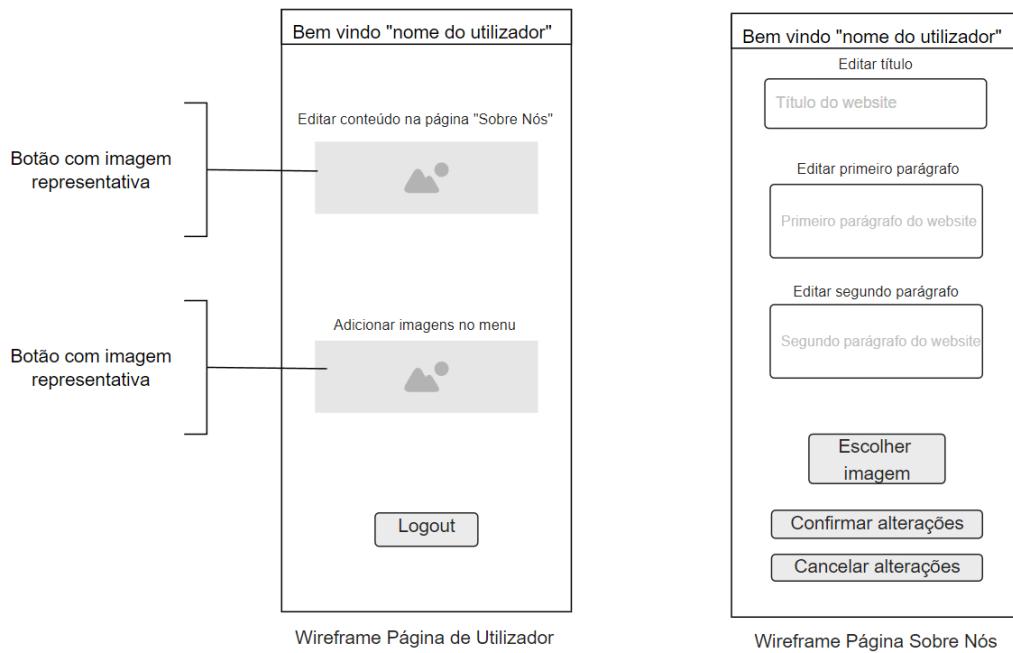


Figura 4.10: *Wireframe* de Página de Utilizador e de Página Sobre Nós.

4.8 Mockups

Mockups são uma versão mais detalhada dos *Wireframes* onde normalmente apresentam, ao nível da *UI*, visuais mais representativos de uma possível versão final da aplicação. Foram desenvolvidos diversos *Mockups* tendo em conta a elaboração dos *Wireframes*.

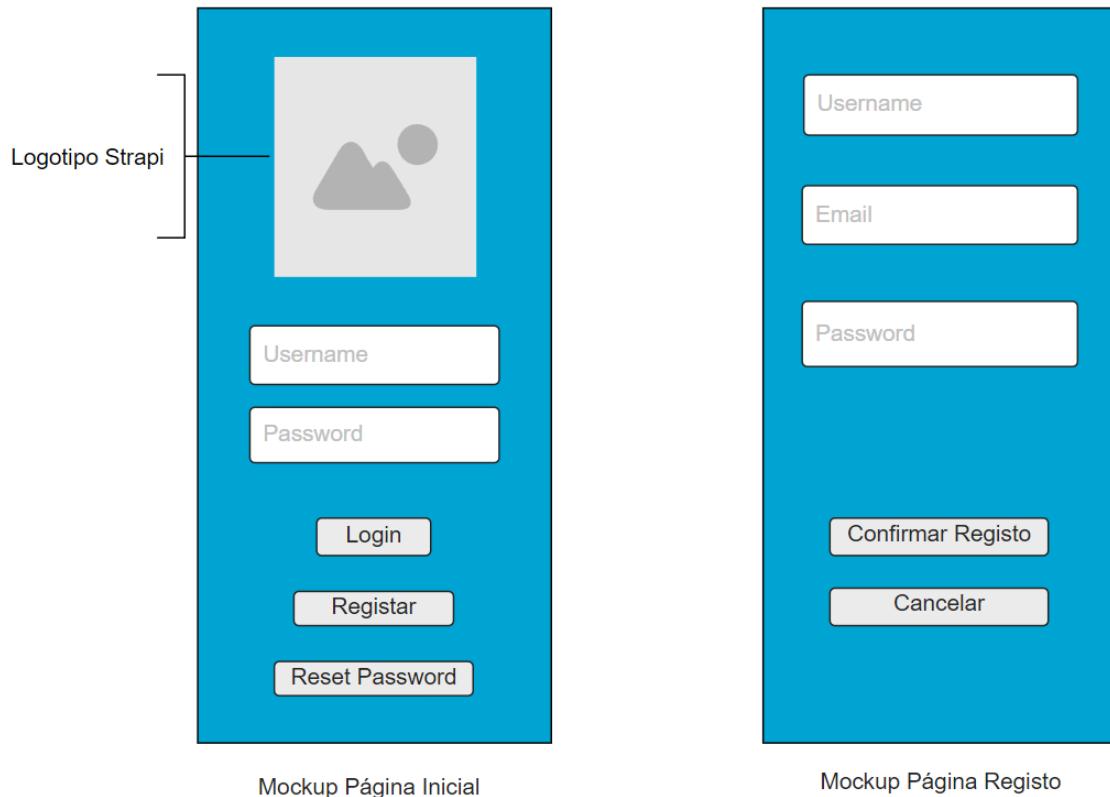
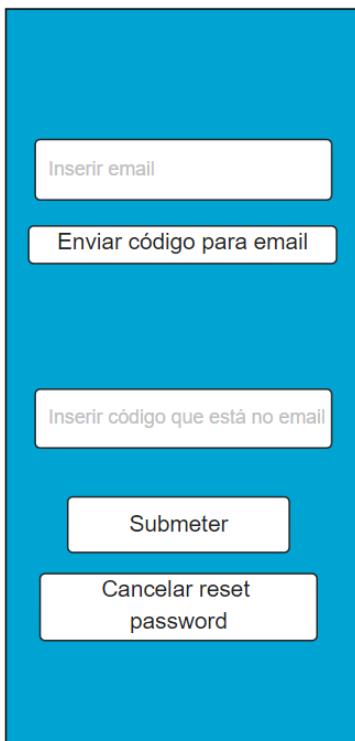
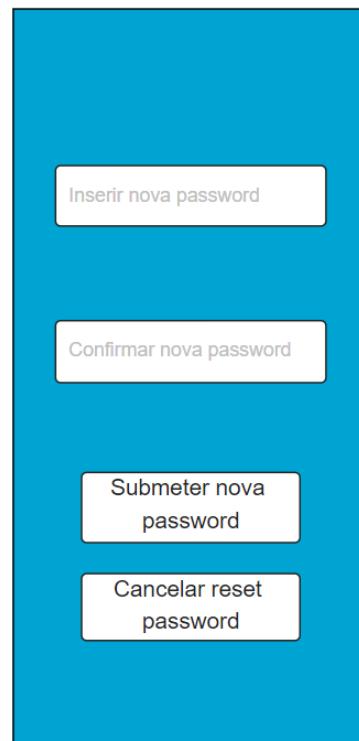


Figura 4.11: *Mockup* de Página Inicial e de Página Registo.



Mockup Página Esquecimento Password



Mockup Página Reset Password

Figura 4.12: *Mockup* de Página Esquecimento Password e de Página Reset Password.

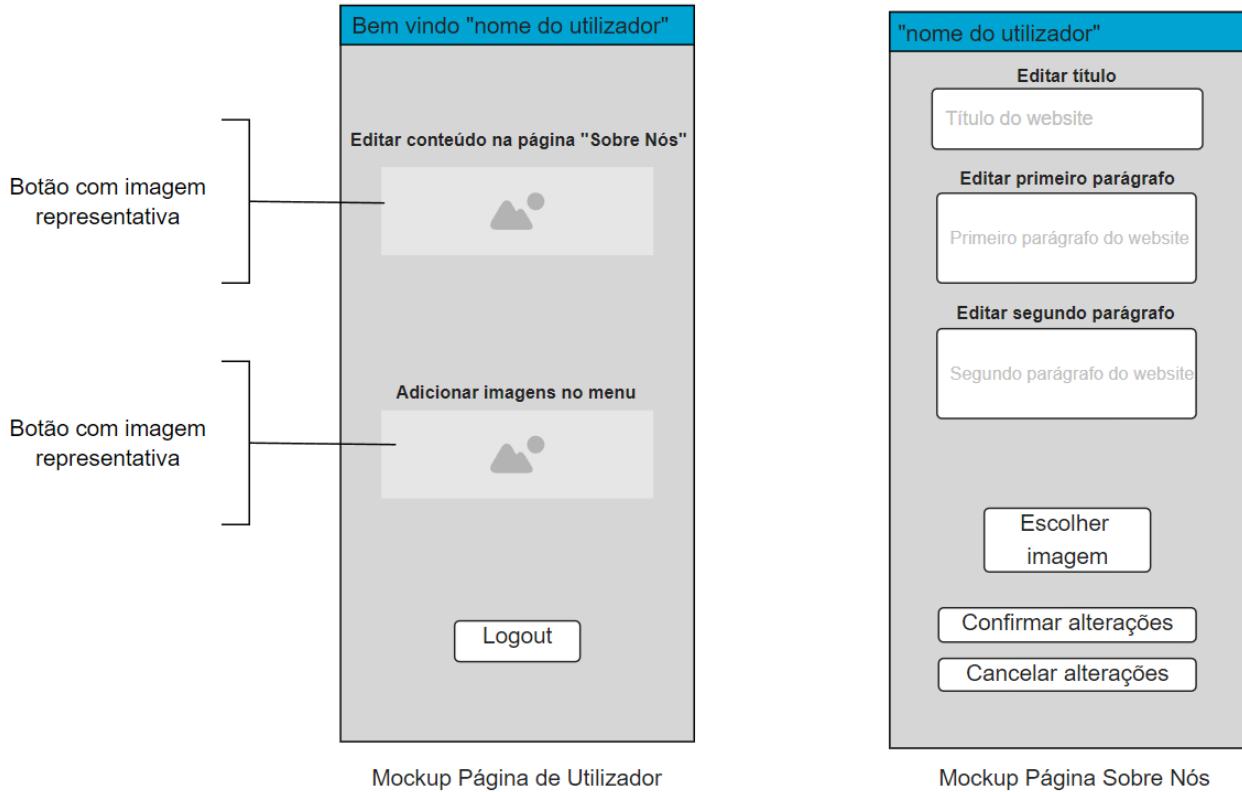


Figura 4.13: *Mockup* de Página de Utilizador e de Página Sobre Nós.

4.9 Conclusão

Com este capítulo pode-se concluir que para um desenvolvimento adequado e correto de um sistema é necessário efetuar certas operações de planeamento e especificação. Estas operações envolvem especificação de requisitos de sistema, elaboração de diagramas que representam o sistema, especificações sobre base de dados e arquitetura do sistema bem como *design's* possíveis a desenvolver. A conjugação destes elementos contribui para um desenvolvimento conciso e consistente de um sistema.

Capítulo

5

Implementação, Testes e Manual de Utilização

5.1 Introdução

Este capítulo comprehende a implementação do sistema agrupando as funcionalidades de extrema importância, detalhes relativos a Strapi e respetiva gestão do conteúdo, *layout* final da aplicação móvel e respetivos testes efetuados às funcionalidades da aplicação bem como exemplos práticos aplicados pela aplicação móvel. O capítulo encontra-se estruturado nas seguintes secções:

- Secção 5.2: referência a aspetos importantes que permitem a funcionalidade do *Headless CMS* Strapi;
- Secção 5.3: exibição e consequente explicação de funcionalidades importantes na aplicação móvel bem como *layout* final desenvolvido;
- Secção 5.4: conjunto de testes realizados às funcionalidades da aplicação e respetivos exemplos.
- Secção 5.5: manual de utilização da aplicação.

5.2 Strapi

5.2.1 Instalação e Execução

A Strapi, tal como referido em 2.3.1, é um CMS que tem como principal objetivo a gestão de conteúdo de um dado *website* através do uso das suas API's. A versão da Strapi utilizada para este projeto é 4.1.5 *Community Edition*. Para uma configuração inicial, é necessário executar um comando no terminal do *Command Prompt* do *Windows* para ser criada uma pasta local da Strapi na máquina. Previamente a este comando é necessário instalar na máquina *host* do projeto Strapi um pré-requisito de *software* para que o comando previamente referido seja executado corretamente.

O pré-requisito referido trata-se da instalação do *Node.js* na máquina. Este *software* é principalmente usado para servidores *event-driven*, *websites* e fundamentalmente para serviços *back-end* que utilizam *API's*. A Strapi necessita deste *software* para poder construir o projeto local na máquina bem como de um *package* inerente ao *Node.js* sendo este o *NPM*. Este *package* é necessário para correr *scripts* da Strapi *Command Line Interface* (*CLI*) que possibilitam a criação e execução do projeto local na máquina.

A versão do *Node.js* e do *package NPM* instalados são 16.14.1 e 8.5.0 respetivamente. A instalação deste *software* e respetivo *package* foi realizado via *download* de um executável de instalação do *website* oficial do *software*. Após a instalação bem sucedida deste software, o comando 5.1 pode ser executado via *Command Prompt* do *Windows* para criação do projeto Strapi local na máquina.

```
> npx create-strapi-app@latest my-project --quickstart
```

Exerto de Código 5.1: Comando necessário para criação de projeto Strapi local.

Seguidamente à criação do projeto, um *URL* específico será acedido via *browser* sendo depois necessário criar uma conta de administrador para o projeto local. Este administrador terá total acesso a todo o conteúdo do projeto Strapi via uma interface de utilizador num *browser*. Com base neste utilizador é possível efetuar qualquer tipo de operação pela interface e foi a partir desta que se efetuou a criação de conteúdos que serão referidos a seguir.

Importante referir também que este projeto Strapi é criado por defeito na **porta 1337** do **localhost**, tendo sido esta definição utilizada para o projeto.

Após a criação do projeto, é possível executá-lo por comandos pelo *Command Prompt* do *Windows* no caminho onde a raiz do projeto Strapi se encontra. Existem 2 tipos diferentes de execuções, a execução em modo *develop*

e em modo *start*. O comando 5.2 permite executar o projeto localmente proporcionando um URL para o utilizador se dirigir num *browser* e colocar as suas credenciais de administrador. O comando 5.3 efetua a mesma funcionalidade sendo a diferença mais importante a permissão de alterações a estruturas de dados no *plugin Content-Type Builder* referido na subsecção 3.2.2.

Estruturas de dados neste *plugin* não permitem alterações no modo *start*. No modo *develop* qualquer tipo de operação realizado no conteúdo do *plugin* efetua um comando *autoReload* sendo este necessário para efetuar as devidas alterações, no entanto no modo *start* este *autoReload* não existe e consequentemente não é possível efetuar qualquer tipo de alteração a estruturas de dados no *plugin Content-Type Builder*.

O modo *develop* é mais adequado para o desenvolvimento contínuo do projeto enquanto o modo *start* é adequado para apresentação da versão final do mesmo.

```
> npm run develop
```

Excerto de Código 5.2: Comando necessário para execução de projeto Strapi local em modo *develop*.

```
> npm run start
```

Excerto de Código 5.3: Comando necessário para execução de projeto Strapi local em modo *start*.

5.2.2 Criação de Conteúdo

A Strapi possibilita a criação de conteúdo através da utilização do *plugin Content-Type Builder* sendo através deste que se elaboraram 3 estruturas de dados diferentes. Tal como referido no 3.^º parágrafo da secção 4.6, foram elaborados 3 *Content Types* de acordo com conteúdos específicos de um website utilizado para testes (este detalhe será posteriormente referido na secção 5.4).

Estes 3 *Content Types* subdividem-se em:

- ***Collection Type Photo Gallery About Us***: Estrutura de dados criada com diversas *entries* que têm como único campo uma imagem. Este *Content Type* é apenas utilizado para imagens;
- ***Collection Type Photo Gallery Menu***: Possui a mesma estrutura do *Collection Type* referindo anteriormente sendo apenas utilizado para imagens;

- **Single Type About Area:** Estrutura de dados com apenas uma *entry* criada com 3 campos para inserção de texto e 1 campo para inserção de imagem.

The screenshot shows the Content Management System interface for the "Photo Gallery About Us" collection type. On the left, the navigation sidebar lists "COLLECTION TYPES" (Photo Gallery About Us, Photo Gallery Menu, User) and "SINGLE TYPES" (About Area). The main area displays a table titled "Photo Gallery About Us" with 5 entries found. The table columns are ID, PHOTO, CREATEDAT, UPDATEDAT, and STATE. All entries have an ID from 1 to 7, were created and updated on Friday, June 17, 2022, at various times between 6:25 PM and 6:26 PM, and are in a "Published" state. Action buttons for edit, delete, and preview are visible for each entry.

ID	PHOTO	CREATEDAT	UPDATEDAT	STATE
1		Friday, June 17, 2022 at 6:25 PM	Friday, June 17, 2022 at 6:26 PM	Published
2		Friday, June 17, 2022 at 6:25 PM	Friday, June 17, 2022 at 6:26 PM	Published
5		Friday, June 17, 2022 at 6:25 PM	Friday, June 17, 2022 at 6:26 PM	Published
6		Friday, June 17, 2022 at 6:26 PM	Friday, June 17, 2022 at 6:26 PM	Published
7		Friday, June 17, 2022 at 6:26 PM	Friday, June 17, 2022 at 6:26 PM	Published

Figura 5.1: Collection Type Photo Gallery About Us criado via interface.

The screenshot shows the Content Management System interface for the "Photo Gallery Menu" collection type. The navigation sidebar is identical to Figure 5.1. The main area displays a table titled "Photo Gallery Menu" with 4 entries found. The table columns are ID, GALLERY, CREATEDAT, UPDATEDAT, and STATE. Entries 1, 43, 89, and 91 were created and updated on June 10, 11, 17, and 19, 2022, respectively, at various times. All are in a "Published" state. Action buttons for edit, delete, and preview are visible.

ID	GALLERY	CREATEDAT	UPDATEDAT	STATE
1		Friday, June 10, 2022 at 4:02 PM	Friday, June 10, 2022 at 4:03 PM	Published
43		Saturday, June 11, 2022 at 9:33 PM	Saturday, June 11, 2022 at 9:33 PM	Published
89		Friday, June 17, 2022 at 12:24 PM	Friday, June 17, 2022 at 12:24 PM	Published
91		Sunday, June 19, 2022 at 11:54 AM	Sunday, June 19, 2022 at 11:54 AM	Published

Figura 5.2: Collection Type Photo Gallery Menu criado via interface.

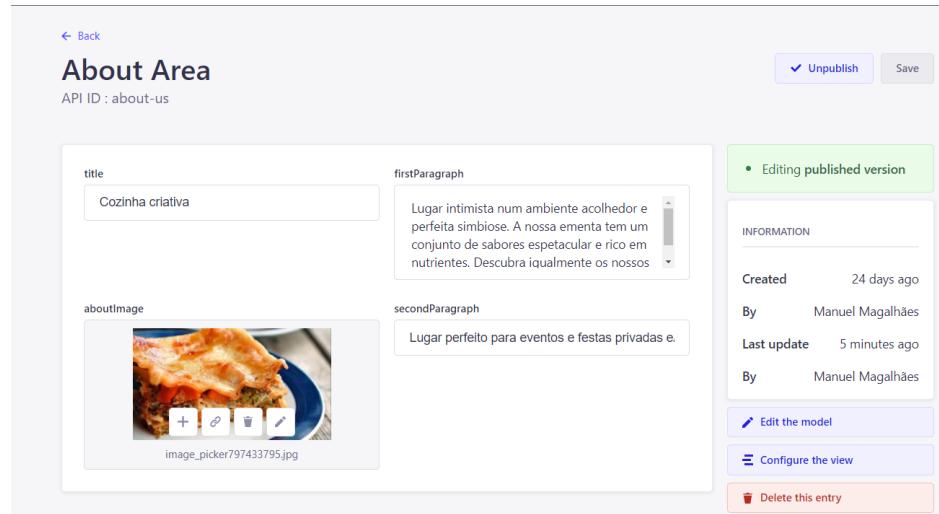


Figura 5.3: *Single Type About Area* criado via interface.

Todos os *Content Types* anteriores foram criados previamente com o processo utilizado na figura 3.3 e 3.4 apenas diferindo no tipo de *Content Type* escolhido. Importante referir que apenas o *Collection Type Photo Gallery Menu* e *Single Type About Area* são geridos via aplicação móvel, enquanto o *Collection Type Photo Gallery About Us* apenas é utilizado para demonstrar conteúdo via interface da Strapi.

No exemplo do *Single Type* referido na secção 3.2.4 foi necessário a atribuição de um API ID aquando a criação do mesmo. Importante constatar que em qualquer criação de estruturas de dados, mais explicitamente em qualquer criação de *Content Types*, é sempre necessário estabelecer um API ID singular e plural (demonstrado na figura 3.3).

Esta questão de estabelecimento de API's traz uma questão muito importante para a discussão de acesso das mesmas tendo já sido referido na secção 3.2.4. As permissões de utilizadores definem o acesso às API's dos *Content Types* criados. Para todos os *Content Types* criados foram anteriormente definidas permissões para utilizadores **Authenticated**. Foram definidas as seguintes permissões:

- **delete, find e update** para remoção, procura e atualização de conteúdo respetivamente. Associado ao *Single Type About Area*;
- **create, delete, find, findOne e update** para criação, remoção, procura total, procura específica e atualização de conteúdo respetivamente. Associado ao *Collection Type Photo Gallery About Us* e ao *Collection Type Photo Gallery Menu*.

Além destas estruturas de dados é importante referir que existe um *Collection Type* elaborado por defeito pela *Strapi* designado por *Users*. Este *Content Type* agrupa todos os utilizadores do sistema que efetuaram um registo de conta pela aplicação móvel. A esta estrutura de dados está associada uma API responsável pela autenticação, registo e alterações de credenciais chamada em múltiplas ocasiões.

As posteriores chamadas a estas API's serão tratadas via aplicação móvel na secção 5.3 para posterior obtenção e gestão remota de conteúdo.

Na figura 5.4 exemplifica-se uma rota de API via *GET Request* utilizada na implementação de uma funcionalidade referida na secção 5.3.6.1. Para diversas implementações aplicadas na aplicação móvel, muitas rotas de API's são chamadas em algumas funcionalidades presentes na secção 5.3. Em cada uma das funcionalidades explicadas na secção 5.3 serão especificadas as rotas utilizadas.

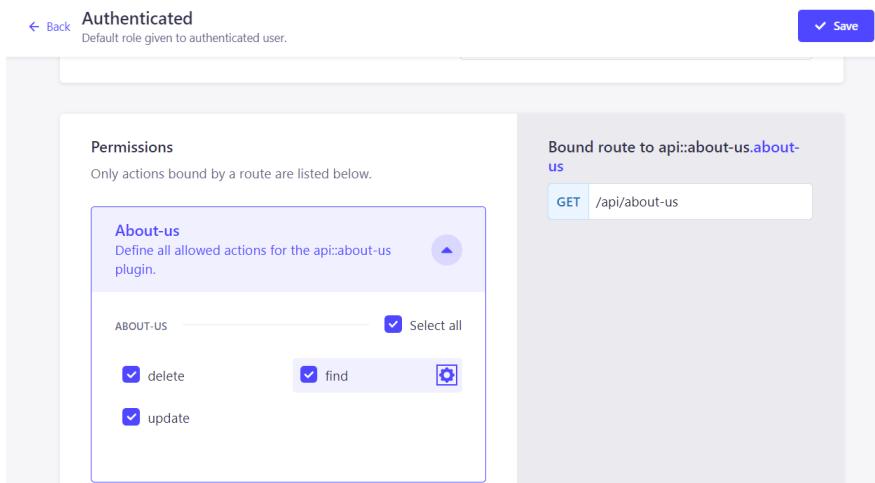


Figura 5.4: *Single Type About Area API route* dedicado para obtenção de informação via *GET Request*.

5.3 Aplicação Móvel

Para o desenvolvimento da aplicação móvel foi utilizada a linguagem de programação Dart na versão 2.16.1 juntamente com uma ferramenta de *performance* e de *debugging* DevTools na versão 2.9.2. Tudo isto foi agrupado na *framework* Flutter na versão 2.10.3.

Nas secções 4.7 e 4.8 elaborou-se um conjunto de *design's* possíveis para a interface da aplicação móvel. Nesta secção é apresentado o *design final da apresentação* bem como as funcionalidades inerentes à aplicação móvel.

Nas subsecções 5.3.5 e 5.3.6 o design e posteriores implementações da aplicação tiveram em conta o conteúdo de um *website* já existente. O *website* utilizado encontra-se referenciado em [5]. Importante referir que testes efetuados ao conteúdo do *website* são realizados em contexto local (ficheiros *HTML* locais na máquina).

5.3.1 Página Inicial

5.3.1.1 *Layout* e Funcionalidades

O *layout* da página inicial da aplicação pode ser visualizado pela figura 5.5. Esta página é composta por diversos elementos visuais apresentados ao utilizador. Estes elementos subdividem-se nos seguintes:

1. **Imagen do logotipo da Strapi;**
2. **TextField para inserção de email:** Este *widget* de inserção de texto via *input* do utilizador é designado por *TextField*. Isto é um *widget* do Flutter que permite a inserção de texto via *input*. Este elemento armazena o valor de *email* inserido pelo utilizador;
3. **TextField para inserção de password:** Este *widget* também é um *TextField* (tal como no elemento referido acima). Armazena o valor de *password* inserido pelo utilizador;
4. **TextButton “Login“ para submissão das credenciais introduzidas nos TextFields:** O *TextButton* é um *widget* do Flutter que define a funcionalidade de um botão na interface. Este botão é responsável pela **funcionalidade de autenticação dos utilizadores**.

A rota da API utilizada para esta funcionalidade é *api/auth/local*.

Esta funcionalidade irá tratar de processar os dados introduzidos pelo utilizador nos *TextFields* e efetuar uma chamada à API que trata da autenticação de pedidos de *login* efetuados. Efetua-se um *POST Request* para a API especificada e é através de uma resposta JSON que se verifica se o pedido foi ou não bem efetuado através do *status code* retornado. Em caso afirmativo, o utilizador será redirecionado para outra página para começar a gerir conteúdo da Strapi. Em caso negativo serão apresentadas mensagens de erro a indicar o insucesso da operação.

Importante referir no momento que se efetua a chamada à API através do *POST Request*, se esta for bem sucedida, a resposta irá retornar um ***JSON Web Token (JWT) Token***. Este *Token* possibilitará o uso de certas operações na aplicação a utilizadores **autenticados**.

5. ***TextButton* “Registrar” para redirecionamento para página de registo:**

Tal como referido, no ponto 4, este elemento também é um ***TextButton*** e apenas trata de redirecionar o utilizador para uma página dedicada a registo de utilizadores.

6. ***TextButton* “Reset Password” para redirecionamento para página de *esquecimento password*:** Utilizado para redirecionar utilizador para uma página intermediária afeta à funcionalidade de alteração de *password*.

Caso o pedido efetuado pelo utilizador, em qualquer uma das situações, demore 10 segundos a ser processado (*timeout* de 10 segundos) isto significa que o servidor encontra-se atualmente *offline* sendo este estado apresentado ao utilizador via *dialog*.

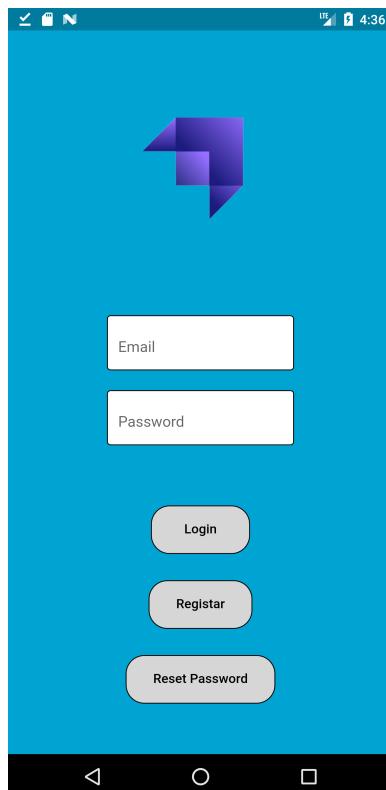


Figura 5.5: *Layout* da página inicial.

5.3.2 Página Registo

5.3.2.1 Layout e Funcionalidades

O *layout* desta página pode ser visualizado pela figura 5.6. Esta página agrupa diversos *widgets* com que o utilizador consegue interagir. Estes *widgets* são os seguintes:

1. **TextField para inserção de *username*:** Widget utilizado para recolher conteúdo introduzido pelo utilizador no que diz respeito ao campo de *username*;
2. **TextField para inserção de *email*:** Widget utilizado para recolher conteúdo introduzido pelo utilizador no que diz respeito ao campo de *email*;
3. **TextField para inserção de *password*:** Widget utilizado para obter conteúdo introduzido pelo utilizador para o campo de *password*;
4. **TextButton “Confirmar Registo”:** Operação dedicada ao registo de um novo utilizador na Strapi.

Esta funcionalidade tem como principal objetivo registar um utilizador na Strapi. Para tal acontecer é necessário que os valores introduzidos sejam válidos e pode-se verificar pela figura 5.6 que existe validação de *inputs* para os *TextFields* de *email* e de *password*. O *email* deverá conter um caractere ‘@’ e um caractere ‘.’ para ser válido e a *password* deverá conter no mínimo 8 caracteres com pelo menos 1 letra e 1 número. Após a introdução de todos os valores, o utilizador pode efetuar registo na aplicação pelo uso do botão.

Posteriormente existe uma verificação ao nível do *username* (se o *username* introduzido já existe) e ao nível do *email* (se o *email* introduzido já existe). Se nenhum destes valores já existir na Strapi o registo do novo utilizador é efetuado. Caso contrário é apresentado ao utilizador um *dialog* com mensagens de erro a indicar o que correu mal.

Caso o pedido efetuado pelo utilizador demore 10 segundos (*timeout* de 10 segundos) a ser processado isto significa que o servidor encontra-se atualmente *offline* sendo este estado apresentado ao utilizador via *dialog*.

Esta operação de registo foi efetuada utilizando uma chamada à API da autenticação para registo com a rota ***api/auth/local/register*** via *POST Request*.

5. **TextButton “Cancelar“ para redirecionamento para página inicial:** Caso o utilizador decida cancelar a sua operação de registo é apresentado um *dialog* para confirmação de cancelamento. Se este decidir cancelar será redirecionado para a página inicial caso contrário mantém-se na mesma página.

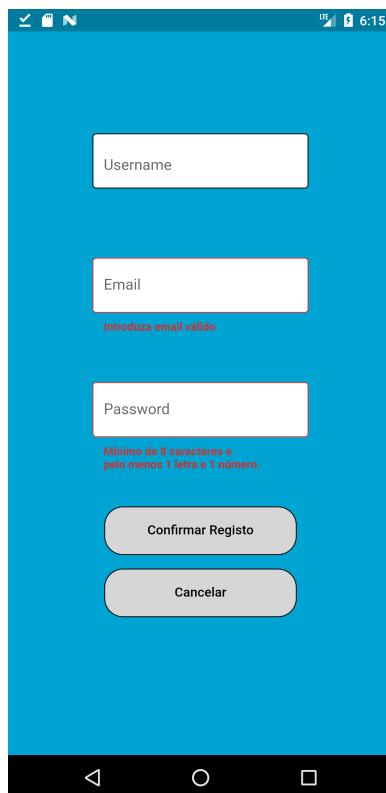


Figura 5.6: *Layout* da página de registo.

5.3.3 Página Esquecimento *Password*

5.3.3.1 *Layout* e Funcionalidades

O *layout* desta página está demonstrado na figura 5.7. Esta página atua como uma página intermediária para a funcionalidade de *reset password* de um dado utilizador. Os *widgets* utilizados para o *layout* são os seguintes:

- **Textfield para inserção de *email*:** este *widget* trata de recolher o conteúdo introduzido pelo utilizador no que diz respeito ao *email* do mesmo;

- **TextButton “Enviar código para email”:** este *widget* tem como funcionalidade permitir um envio de email para os utilizadores com um código para ser usado no *TextField* seguinte. A API da Strapi que trata do *reset* da *password* do utilizador necessita de um campo adicional. Este campo é um código gerado pela Strapi sendo este obtido por outra chamada a outra API.

Esta última necessita do *email* do utilizador que requer a alteração da *password*. Neste caso quando o utilizador efetua a operação é enviado um email via um *SMTP provider*, referido na secção 3.2.5. Importante constatar que a Strapi verifica se o *email* introduzido existe no sistema. No caso onde não existe é emitido um *dialog* ao utilizador a informá-lo desse facto. A rota da API utilizada para esta funcionalidade é ***api/auth/forgot-password*** via *POST Request*.

- **Textfield para inserção de código:** este *widget* trata de recolher o conteúdo introduzido pelo utilizador no que diz respeito ao código enviado previamente por *email*;
- **TextButton “Submeter”:** Este botão tem como finalidade validar o código introduzido pelo utilizador no segundo *Textfield*. O código inserido é comparado com o código enviado no *email* e se ambos forem iguais o utilizador será redirecionado para a página *Reset Password* que implementa a funcionalidade de *reset* da palavra-passe do utilizador. Caso contrário é apresentado ao utilizador um *dialog* a indicar que o código não é o mesmo;
- **TextButton “Cancelar reset password”:** Este *widget* remete o utilizador para a página inicial da aplicação, pois este escolheu cancelar a operação de *reset* da sua palavra-passe.

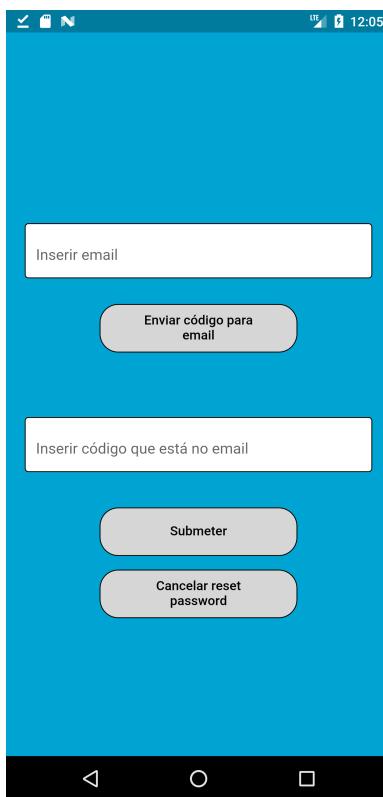


Figura 5.7: *Layout* da página de esquecimento de password.

5.3.4 Página Reset Password

5.3.4.1 Layout e Funcionalidades

O *layout* desta página está demonstrado na figura 5.8. Esta página implementa a funcionalidade de *reset* da palavra passe de utilizador. Utiliza o código obtido na página intermediária de esquecimento de password referida em 5.3.3 para posterior reset da *password* numa chamada à API da Strapi. Os *widgets* que compõem esta página são os seguintes:

- **TextField para inserção da nova *password*:** este *widget* recolhe o conteúdo introduzido pelo utilizador no que diz respeito à nova palavra-passe;
- **TextField para confirmação da *password*:** este *widget* recolhe o conteúdo introduzido pelo utilizador para confirmar a nova palavra-passe;
- **TextButton “Submeter nova password”:** neste caso após o utilizador submeter o pedido de alteração de *password* a funcionalidade associ-

ada é aplicada. Importante constatar que se o conteúdo introduzido pelo utilizador em ambos os *TextFields* não for o mesmo é apresentado uma mensagem de erro a informar que a nova *password* não está confirmada. Apenas quando o conteúdo de ambos for o mesmo é que é submetida a nova palavra-passe do utilizador. A submissão do pedido de alteração irá conter os valores inseridos nos campos desta página bem como o código obtido na página esquecimento de password referida em 5.3.3. A junção do código com os valores inseridos efetua uma mudança da palavra-passe da conta do utilizador.

Pela visualização da figura 5.8 pode-se verificar que existe validação de *input* para as palavras passes inseridas onde cada uma delas deve ter no mínimo 8 caracteres com pelo menos 1 letra e 1 número.

A rota da API utilizada para esta funcionalidade é ***api/auth/reset-password*** via *POST Request*.

- ***TextButton* “Cancelar reset password”:** Este *widget* remete o utilizador para a página inicial da aplicação, pois este escolheu cancelar a operação de *reset* da sua palavra-passe.

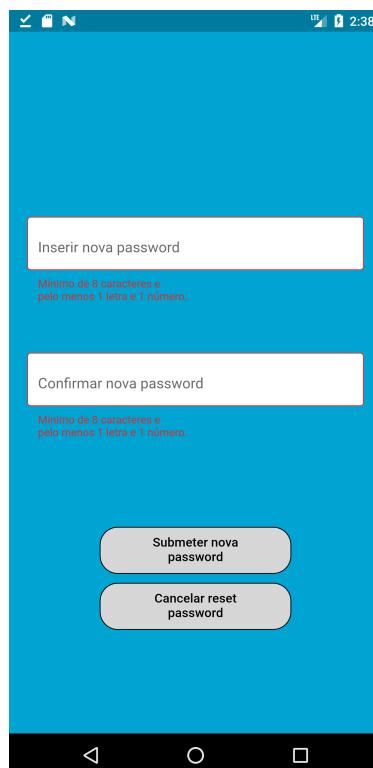


Figura 5.8: Layout da página de *reset* da *password*.

5.3.5 Página Principal de Utilizador

5.3.5.1 *Layout* e Funcionalidades

O *layout* desta página está demonstrado na figura 5.9. Esta página é responsável pela funcionalidade de *upload* de imagens para o *Collection Type Photo Gallery Menu* referido em 5.2.2. A página é composta pelos seguintes *widgets*:

- **Text** para indicar a operação de edição de conteúdo ao utilizador;
- **IconButton que remete o utilizador para a página Sobre Nós**, referida na secção 5.3.6 para alteração de conteúdo no *Single Type About Area*;
- **Text** para indicar a operação de upload de imagens ao utilizador;
- **IconButton que implementa o upload de imagens no Collection Type Photo Gallery Menu**: a funcionalidade associada a este botão possibilita a obtenção de uma imagem da galeria do dispositivo e posterior chamada à API associada ao *Collection Type* referido para efetuar o *upload* da imagem.

Neste tipo de situação, o utilizador após ter escolhido efetuar a operação será remetido para a galeria do dispositivo onde terá de selecionar uma imagem para fazer *upload*. Após ter escolhido a imagem poderá confirmar a submissão através de um *dialog*. Efetua-se uma chamada à API do *Collection Type Photo Gallery Menu* via um *POST Request*. Após esta chamada é apresentado um *dialog* a indicar o sucesso do *upload*. Um *dialog* apresenta uma mensagem de erro caso o utilizador não submeta corretamente a imagem.

A rota da API utilizada para esta funcionalidade é ***api/photo-gallery-menus***.

- **TextButton “Logout”**: o utilizador efetua uma operação de *logout* sendo redirecionado para a página inicial.



Figura 5.9: *Layout* da página principal de utilizador.

5.3.6 Página Sobre Nós

5.3.6.1 *Layout e Funcionalidades*

O *layout* desta página está demonstrado em 5.10. Esta página é responsável pela alteração de conteúdo no *Single Type About Area* referido em 5.2.2. Os *widgets* que compõem esta página são os seguintes:

- **TextField para inserção de título:** este *widget* é utilizado para gerir conteúdo relativo ao título da página “Sobre Nós“ do *website*;
- **TextField para inserção de primeiro parágrafo:** este *widget* é utilizado para gerir conteúdo do primeiro parágrafo da página “Sobre Nós“ do *website*;
- **TextField para inserção de segundo parágrafo:** este *widget* é utilizado para gerir conteúdo do segundo parágrafo da página “Sobre Nós“ do *website*;

- **TextButton “Escolher imagem”:** este *widget* possibilita a escolha de uma imagem para inserir na página. Um *dialog* é apresentado para mensagens de erro quando a imagem não foi obtida corretamente.
- **TextButton “Confirmar alterações”:** este *widget* é responsável pela implementação das alterações efetuadas no conteúdo. Neste caso o utilizador efetua a operação de *upload* das alterações sendo chamada, para este efeito, uma API associada ao *Single Type About Area* via *PUT Request*.

Importante referir que esta funcionalidade necessitou de 2 *PUT Requests* para ser aplicada corretamente. Atualmente a Strapi não consegue atualizar imagens em *Single Types*. Tendo isto em conta foi necessário efetuar um **primeiro *PUT Request*** para eliminar o conteúdo associado ao campo *media* (que é o campo da imagem no *Type*) e foi seguidamente efetuado um **segundo *PUT Request*** para atualizar o conteúdo associado aos campos do *Single Type*.

A rota da API utilizada para esta funcionalidade é ***api/about-us***.

- **TextButton “Cancelar alterações”:** este *widget* apresenta um *dialog* ao utilizador para confirmar se este deseja cancelar as alterações efetuadas. Se desejar é remetido para a página principal de utilizador caso contrário permanece na mesma página.

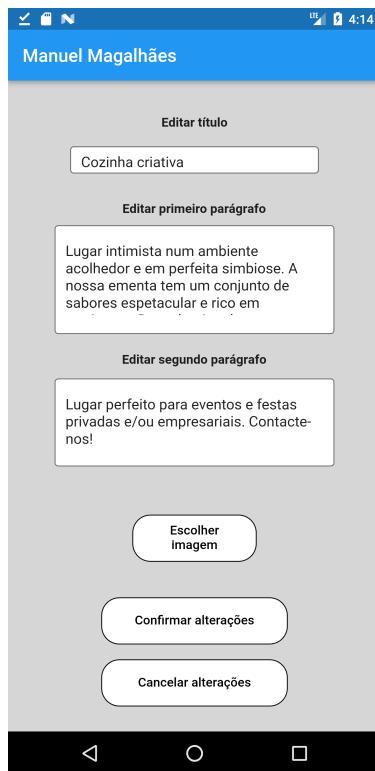


Figura 5.10: *Layout* da página Sobre Nós.

5.4 Testes Efetuados

Diversos testes foram realizados ao longo do desenvolvimento da aplicação de forma a testar todas as funcionalidades que iam sendo implementadas. O ambiente de testes utilizado para a aplicação desenvolvida foi o **Android Studio**, um ambiente dedicado para emulação de aplicações com sistema operativo *Android*. Neste ambiente de emulação foram utilizados 3 dispositivos diferentes para testes sendo estes:

- *Pixel 2 XL* com o sistema operativo *Android 7.1.1*, um ecrã de 6 polegadas e resolução de 1440 x 2880 pixels;
- *Pixel 3 XL* com o sistema operativo *Android 7.0*, um ecrã de 6.3 polegadas e resolução de 1440 x 2960 pixels;
- *Nexus 5X* com o sistema operativo *Android 7.0*, um ecrã de 5.2 polegadas e resolução de 1080 x 1920 pixels;

Importante referir que os testes efetuados pela aplicação além de compreenderem interações com o projeto local da *Strapi* também envolveram a utilização de uma ferramenta fornecida por um colega do curso de Informática *Web*. Esta ferramenta agrupa um conjunto de ficheiros escritos em linguagem **JavaScript** que efetuam uma operação *fetch* da **fetch API** que possibilita a recolha de informação da *Strapi*. Posteriormente esta ferramenta também inclui a **aplicação de HTML no website**, com os dados obtidos via *fetch*.

Esta ferramenta foi utilizada sobretudo para testes a um nível mais realista, demonstrando a principal aplicação do *HeadLess CMS Strapi* na alteração de conteúdo de *sites*. Exemplos práticos tanto a nível da *Strapi* bem como a nível da aplicação de conteúdo no *website* são demonstrados na subsecção 5.4.7.

Esta secção encontra-se subdividida nas seguintes subsecções:

- Subsecção 5.4.1 que comprehende todos os testes possíveis referidos na tabela 5.1 para a página inicial da aplicação;
- Subsecção 5.4.2 que agrupa todos os testes que podem ser efetuados na página de registo estando evidenciado na tabela 5.2;
- Subsecção 5.4.3 que comprehende todos os testes possíveis referidos na tabela 5.3 para a página de esquecimento da *password*;
- Subsecção 5.4.4 que evidencia todos os testes que podem ser efetuados para a página de *reset* da *password* exibidos pela tabela 5.4;
- Subsecção 5.4.5 que agrupa todos os testes possíveis que podem ser realizados na página principal de utilizador, evidenciados pela tabela 5.5;
- Subsecção 5.4.6 que comprehende todos os testes que podem ser efetuados na página Sobre Nós evidenciados pela tabela 5.6.

5.4.1 Testes para Página Inicial

ID	Descrição de Teste	Resultado Esperado	Resultado Obtido
1	Pressionar botão “Login” com credenciais corretas.	Efetuar <i>Login</i> e redirecionar utilizador para página principal de utilizador.	<i>Login</i> efetuado e redirecionamento para página principal de utilizador.
2	Pressionar botão “Login” com credenciais incorretas.	Apresentar <i>Dialog</i> com mensagem de erro.	Apresentação do <i>Dialog</i> com a mensagem de erro.
3	Pressionar botão “Registar”.	Redirecionar utilizador para página de registo.	Redirecionamento do utilizador para página de registo.
4	Pressionar botão “Reset Password” e confirmar <i>reset</i> .	Apresentar <i>Dialog</i> para confirmação e redirecionar utilizador para página de esquecimento de <i>password</i> em caso afirmativo.	Apresentação do <i>Dialog</i> , escolha da opção “Sim” e redirecionamento do utilizador para página de esquecimento de <i>password</i> .
5	Pressionar botão “Reset Password” e cancelar <i>reset</i> .	Apresentar <i>Dialog</i> para confirmação e manter utilizador na página inicial em caso negativo.	Apresentação do <i>Dialog</i> , escolha da opção “Não” e utilizador manteve-se na página inicial.
6	Pressionar botão “Login” ou “Registrar” ou “Reset Password” com servidor offline.	Apresentar <i>Dialog</i> com mensagem de erro.	Apresentação do <i>Dialog</i> com a mensagem de erro.

Tabela 5.1: Testes efetuados para Página Inicial.

5.4.2 Testes para Página Registo

ID	Descrição de Teste	Resultado Esperado	Resultado Obtido
7	Pressionar botão “Confirmar Registo” com valores válidos.	Apresentar <i>Dialog</i> com mensagem de sucesso e redirecionar utilizador para página principal de utilizador.	Apresentação do <i>Dialog</i> com mensagem de sucesso e redirecionamento do utilizador para página principal de utilizador.
8	Pressionar botão “Confirmar Registo” com valores inválidos nos campos <i>username</i> e/ou <i>email</i> .	Apresentar <i>Dialog</i> com mensagem de insucesso.	Apresentação do <i>Dialog</i> com mensagem de insucesso.
9	Pressionar botão “Confirmar Registo” com valores inválidos ou inexistentes nos campos.	Apresentar <i>Dialog</i> com mensagem de insucesso.	Apresentação do <i>Dialog</i> com mensagem de insucesso.
10	Pressionar botão “Confirmar Registo” com valores já existentes na Strapi nos campos <i>username</i> e/ou <i>email</i> .	Apresentar <i>Dialog</i> com mensagem de insucesso.	Apresentação do <i>Dialog</i> com mensagem de insucesso.
11	Pressionar botão “Confirmar Registo” com servidor <i>offline</i> .	Apresentar <i>Dialog</i> com mensagem de erro.	Apresentação do <i>Dialog</i> com mensagem de erro.
12	Pressionar botão “Cancelar” e confirmar operação.	Apresentar <i>Dialog</i> para confirmação e redirecionar utilizador para página inicial em caso afirmativo.	Apresentação do <i>Dialog</i> , escolha da opção “Sim” e redirecionamento do utilizador para página inicial.
13	Pressionar botão “Cancelar” e cancelar operação.	Apresentar <i>Dialog</i> para confirmação e manter utilizador na página.	Apresentação do <i>Dialog</i> , escolha da opção “Não” e utilizador manteve-se na página atual.

Tabela 5.2: Testes efetuados para Página Registo.

5.4.3 Testes para Página Esquecimento Password

ID	Descrição de Teste	Resultado Esperado	Resultado Obtido
14	Pressionar botão “Enviar código para email” com email existente na Strapi.	Enviar email com código para <i>reset</i> de password.	Email enviado com código para <i>reset</i> de password.
15	Pressionar botão “Enviar código para email” com email não existente na Strapi.	Apresentar <i>Dialog</i> com mensagem de erro.	Apresentação de <i>Dialog</i> com mensagem de erro.
16	Pressionar botão “Enviar código para email” com servidor offline.	Apresentar <i>Dialog</i> com mensagem de erro.	Apresentação do <i>Dialog</i> com mensagem de erro.
17	Pressionar botão “Submeter” com código enviado por email pela Strapi.	Redirecionar utilizador para página <i>Reset Password</i> .	Redirecionamento de utilizador para página <i>Reset Password</i> .
18	Pressionar botão “Submeter” com código incorreto ou inválido.	Apresentar <i>Dialog</i> com mensagem de erro.	Apresentação de <i>Dialog</i> com mensagem de erro.
19	Pressionar botão “Cancelar reset password”.	Redirecionar utilizador para página inicial.	Redirecionamento do utilizador para página inicial.

Tabela 5.3: Testes efetuados para Página Esquecimento Password.

5.4.4 Testes para Página *Reset Password*

ID	Descrição de Teste	Resultado Esperado	Resultado Obtido
20	Pressionar botão “Submeter nova <i>password</i> ” com valores de <i>password</i> iguais.	Apresentar <i>Dialog</i> com mensagem de sucesso e redirecionar utilizador para página inicial.	Apresentação de <i>Dialog</i> com mensagem de sucesso e redirecionamento do utilizador para página inicial.
21	Pressionar botão “Submeter nova <i>password</i> ” com valores de <i>password</i> diferentes.	Apresentar <i>Dialog</i> com mensagem de erro.	Apresentação de <i>Dialog</i> com mensagem de erro.
22	Pressionar botão “Submeter nova <i>password</i> ” com valores de <i>password</i> inválidos.	Apresentar <i>Dialog</i> com mensagem de erro.	Apresentação de <i>Dialog</i> com mensagem de erro.
23	Pressionar botão “Submeter nova <i>password</i> ” com servidor offline.	Apresentar <i>Dialog</i> com mensagem de erro.	Apresentação do <i>Dialog</i> com mensagem de erro.
24	Pressionar botão “Cancelar <i>reset password</i> ”.	Redirecionar utilizador para página inicial.	Redirecionamento do utilizador para página inicial.

Tabela 5.4: Testes efetuados para Página *Reset Password*.

5.4.5 Testes para Página Principal de Utilizador

ID	Descrição de Teste	Resultado Esperado	Resultado Obtido
25	Pressionar botão com legenda “Editar conteúdo na página Sobre Nós”.	Apresentar <i>Dialog</i> com mensagem de sucesso e redirecionar utilizador para página Sobre Nós.	Apresentação de <i>Dialog</i> com mensagem de sucesso e redirecionamento do utilizador para página Sobre Nós.
26	Pressionar botão com legenda “Editar conteúdo na página Sobre Nós” com servidor offline.	Apresentar <i>Dialog</i> com mensagem de erro.	Apresentação do <i>Dialog</i> com mensagem de erro.
27	Pressionar botão com legenda “Adicionar imagens no menu” e escolher imagem.	Redirecionar utilizador para galeria do dispositivo, escolher imagem e confirmar <i>upload</i> .	Redirecionamento do utilizador para galeria do dispositivo, escolha da imagem e confirmação do <i>upload</i> .
28	Pressionar botão com legenda “Adicionar imagens no menu” e não escolher imagem.	Redirecionar utilizador para galeria do dispositivo, não escolher imagem e apresentar <i>Dialog</i> de erro.	Redirecionamento do utilizador para galeria do dispositivo, não escolheu imagem e apresentação do <i>Dialog</i> de erro.
29	Pressionar botão com legenda “Adicionar imagens no menu” com servidor offline.	Apresentar <i>Dialog</i> com mensagem de erro.	Apresentação do <i>Dialog</i> com mensagem de erro.
30	Pressionar botão “Logout” e confirmar operação.	Apresentar <i>Dialog</i> , confirmar operação e redirecionar utilizador para página inicial.	Apresentação do <i>Dialog</i> , escolha da opção “Sim” e redirecionamento para a página inicial.
31	Pressionar botão “Logout” e cancelar operação.	Apresentar <i>Dialog</i> , cancelar operação e manter utilizador na página atual.	Apresentação do <i>Dialog</i> , escolha da opção “Não” e utilizador manteve-se na página atual.

Tabela 5.5: Testes efetuados para Página Principal de Utilizador.

5.4.6 Testes para Página Sobre Nós

ID	Descrição de Teste	Resultado Esperado	Resultado Obtido
32	Pressionar botão “Escolher imagem”.	Redirecionar utilizador para galeria do dispositivo, escolher imagem.	Redirecionamento do utilizador para galeria do dispositivo e escolha da imagem.
33	Pressionar botão “Confirmar alterações” e confirmar operação.	Apresentar <i>Dialog</i> para confirmar operação e efetuar <i>upload</i> das alterações em caso afirmativo.	Apresentação de <i>Dialog</i> , escolha da opção “Sim” e <i>upload</i> das alterações realizado.
34	Pressionar botão “Confirmar alterações” e cancelar operação.	Apresentar <i>Dialog</i> para cancelar operação e manter utilizador na página atual.	Apresentação de <i>Dialog</i> , escolha da opção “Não” e utilizador manteve-se na página atual.
35	Pressionar botão “Confirmar alterações” com servidor offline.	Apresentar <i>Dialog</i> com mensagem de erro.	Apresentação do <i>Dialog</i> com mensagem de erro.
35	Pressionar botão “Confirmar alterações” sem ter submetido imagem.	Apresentar <i>Dialog</i> com mensagem de erro.	Apresentação do <i>Dialog</i> com mensagem de erro.
36	Pressionar botão “Cancelar alterações” e confirmar operação.	Apresentar <i>Dialog</i> , confirmar operação e redirecionar utilizador para página principal de utilizador.	Apresentação do <i>Dialog</i> , escolha da opção “Sim” e redirecionamento para a página principal de utilizador.
37	Pressionar botão “Cancelar alterações” e cancelar operação.	Apresentar <i>Dialog</i> , cancelar operação e manter utilizador na página atual.	Apresentação do <i>Dialog</i> , escolha da opção “Não” e utilizador manteve-se na página atual.

Tabela 5.6: Testes efetuados para Página Sobre Nós.

5.4.7 Exemplos práticos

5.4.7.1 Edição de texto e imagem no *Single Type About Area*

The screenshot shows the 'About Area' content editor interface. At the top, there are buttons for 'Back', 'Unpublish', and 'Save'. The main area contains fields for 'title' (set to 'Cozinha criativa'), 'firstParagraph' (containing text about a restaurant's menu), 'aboutImage' (a thumbnail of a pizza slice), and 'secondParagraph' (containing text about the perfect place for events). On the right side, a sidebar shows the 'INFORMATION' section with details like 'Created' (24 days ago), 'By' (Manuel Magalhães), and 'Last update' (5 minutes ago). Below the information are buttons for 'Edit the model', 'Configure the view', and 'Delete this entry'.

Figura 5.11: Conteúdo do *Single Type About Area* antes da utilização da funcionalidade de edição de texto e imagem da aplicação.

This screenshot shows the same 'About Area' editor after changes have been made. The 'title' field now reads 'Cozinha mediterrânea criativa'. The 'aboutImage' field now displays a thumbnail of a meal featuring various vegetables and a bowl. The rest of the interface remains largely the same, with the 'INFORMATION' sidebar showing the updated timestamp for the last update.

Figura 5.12: Conteúdo do *Single Type About Area* depois da utilização da funcionalidade de edição de texto e imagem da aplicação.



[Conheça nosso restaurante](#)

Cozinha criativa

Lugar intimista num ambiente acolhedor e perfeita simbiose. A nossa ementa tem um conjunto de sabores espetacular e rico em nutrientes. Descubra igualmente os nossos sumos naturais e a nossa mercearia fina.

Lugar perfeito para eventos e festas privadas e/ou empresariais. Contate-nos!

Figura 5.13: Conteúdo da página Menu do *website* antes da utilização da funcionalidade de edição de texto e imagem da aplicação.



[Conheça nosso restaurante](#)

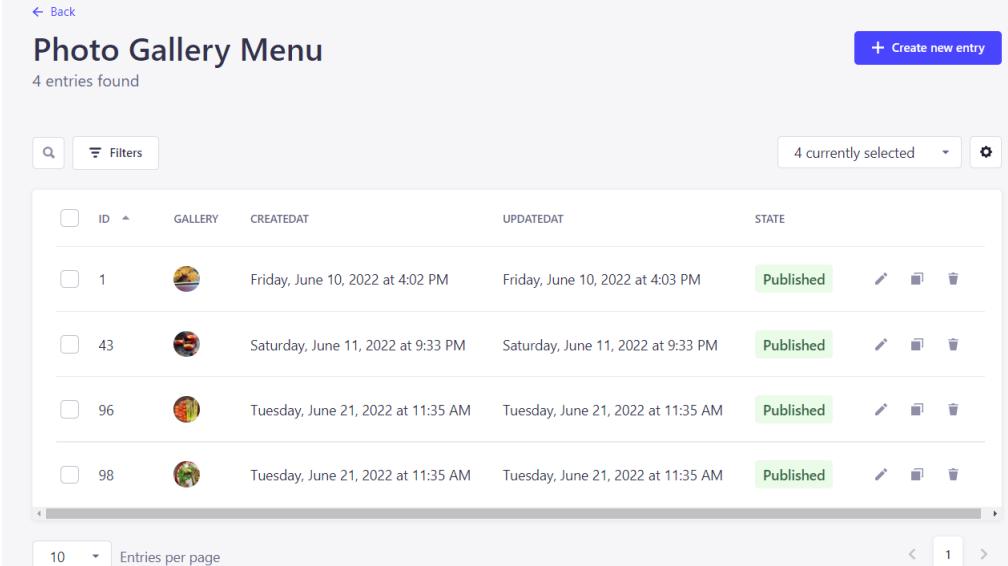
Cozinha mediterrânea criativa

Lugar intimista num ambiente acolhedor. A nossa ementa tem um conjunto de sabores rico em nutrientes. Descubra igualmente os nossos sumos naturais e a nossa mercearia fina.

Lugar perfeito para eventos e festas privadas e/ou empresariais. Contate-nos para mais informações!

Figura 5.14: Conteúdo da página Menu do *website* depois da utilização da funcionalidade de edição de texto e imagem da aplicação.

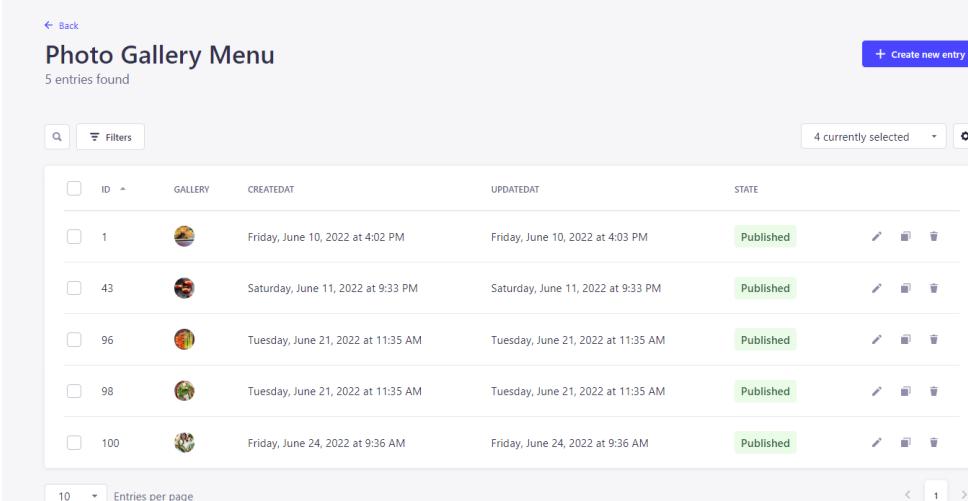
5.4.7.2 Edição de texto e imagem no *Collection Type Photo Gallery Menu*



<input type="checkbox"/>	ID	GALLERY	CREATEDAT	UPDATEDAT	STATE	<input type="button" value="Edit"/>	<input type="button" value="Delete"/>
<input type="checkbox"/>	1		Friday, June 10, 2022 at 4:02 PM	Friday, June 10, 2022 at 4:03 PM	Published	<input type="button" value="Edit"/>	<input type="button" value="Delete"/>
<input type="checkbox"/>	43		Saturday, June 11, 2022 at 9:33 PM	Saturday, June 11, 2022 at 9:33 PM	Published	<input type="button" value="Edit"/>	<input type="button" value="Delete"/>
<input type="checkbox"/>	96		Tuesday, June 21, 2022 at 11:35 AM	Tuesday, June 21, 2022 at 11:35 AM	Published	<input type="button" value="Edit"/>	<input type="button" value="Delete"/>
<input type="checkbox"/>	98		Tuesday, June 21, 2022 at 11:35 AM	Tuesday, June 21, 2022 at 11:35 AM	Published	<input type="button" value="Edit"/>	<input type="button" value="Delete"/>

10 Entries per page

Figura 5.15: Conteúdo do *Collection Type Photo Gallery Menu* antes da utilização da funcionalidade de *upload* imagem da aplicação.



<input type="checkbox"/>	ID	GALLERY	CREATEDAT	UPDATEDAT	STATE	<input type="button" value="Edit"/>	<input type="button" value="Delete"/>
<input type="checkbox"/>	1		Friday, June 10, 2022 at 4:02 PM	Friday, June 10, 2022 at 4:03 PM	Published	<input type="button" value="Edit"/>	<input type="button" value="Delete"/>
<input type="checkbox"/>	43		Saturday, June 11, 2022 at 9:33 PM	Saturday, June 11, 2022 at 9:33 PM	Published	<input type="button" value="Edit"/>	<input type="button" value="Delete"/>
<input type="checkbox"/>	96		Tuesday, June 21, 2022 at 11:35 AM	Tuesday, June 21, 2022 at 11:35 AM	Published	<input type="button" value="Edit"/>	<input type="button" value="Delete"/>
<input type="checkbox"/>	98		Tuesday, June 21, 2022 at 11:35 AM	Tuesday, June 21, 2022 at 11:35 AM	Published	<input type="button" value="Edit"/>	<input type="button" value="Delete"/>
<input type="checkbox"/>	100		Friday, June 24, 2022 at 9:36 AM	Friday, June 24, 2022 at 9:36 AM	Published	<input type="button" value="Edit"/>	<input type="button" value="Delete"/>

10 Entries per page

Figura 5.16: Conteúdo do *Collection Type Photo Gallery Menu* depois da utilização da funcionalidade de *upload* imagem da aplicação.

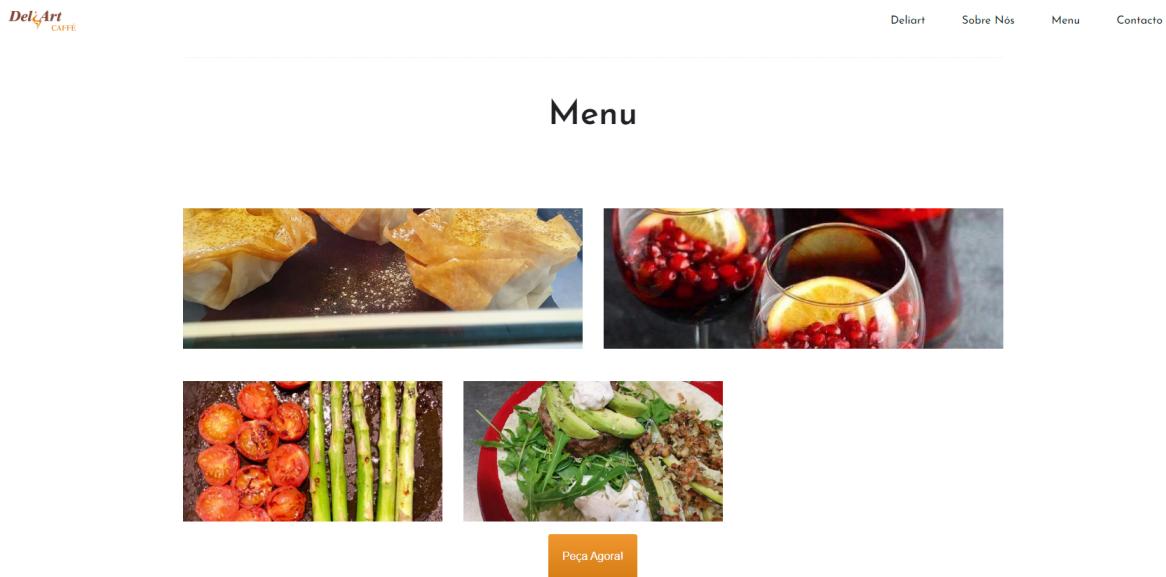


Figura 5.17: Conteúdo da página Sobre Nós do *website* antes da utilização da funcionalidade de *upload* imagem da aplicação.

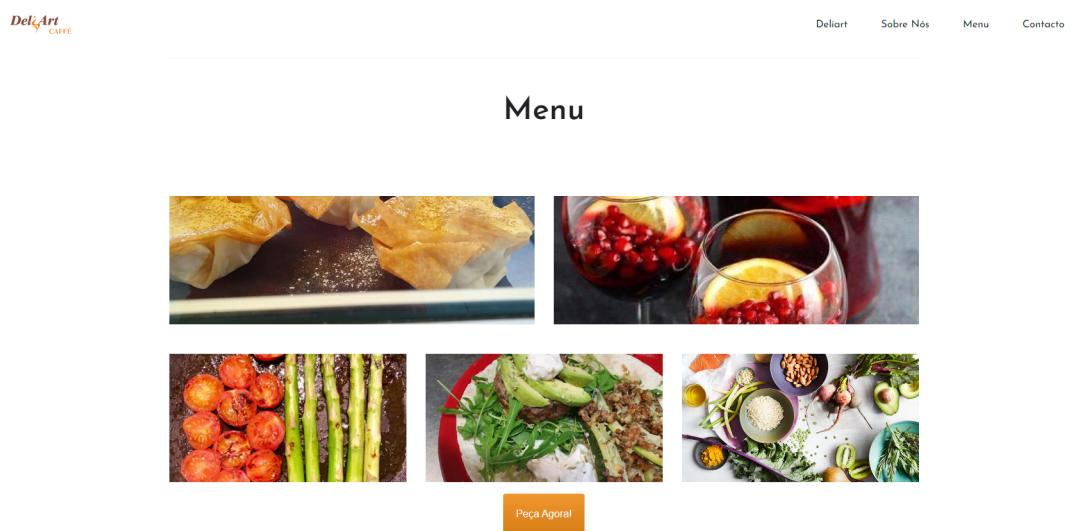


Figura 5.18: Conteúdo da página Menu do *website* depois da utilização da funcionalidade de *upload* imagem da aplicação.

5.5 Manual de Utilização

Nesta secção é apresentado um manual de utilização da aplicação móvel desenvolvida neste projeto. Como a aplicação móvel encontra-se subdividida em várias páginas de utilizador é apresentado uma lista de funcionalidades que o utilizador pode desempenhar em cada uma delas.

5.5.1 Página Inicial

Na página inicial da aplicação, referenciada pela figura 5.5, o utilizador pode efetuar o seguinte conjunto de operações:

- Inserir credenciais de *email* e *password*;
- Operação de *login* na conta da Strapi;
- Iniciar operação de registo de conta da Strapi;
- Iniciar operação de *reset* da *password* de conta da Strapi.

5.5.2 Página Registo

Na página de registo da aplicação, referenciada pela figura 5.6, o utilizador pode efetuar o seguinte conjunto de operações:

- Inserir credenciais de *username*, *email* e *password*;
- Efetuar registo de conta Strapi;
- Cancelar operação de registo.

5.5.3 Página Esquecimento Password

Na página de esquecimento da *password*, referenciada pela figura 5.7, o utilizador pode efetuar o seguinte conjunto de operações:

- Inserir *email*;
- Obter código para o *reset* da *password* via *email*;
- Inserir código obtido via *email*;
- Iniciar operação de *reset* da *password* da conta;
- Cancelar operação de *reset* da *password* da conta.

5.5.4 Página *Reset Password*

Na página de *reset da password*, referenciada pela figura 5.8, o utilizador pode efetuar o seguinte conjunto de operações:

- Inserir nova *password* de conta;
- Inserir novamente a nova *password* de conta para confirmação;
- Submeter pedido de alteração de *password*;
- Cancelar pedido de *reset da password* da conta.

5.5.5 Página Principal de Utilizador

Na página principal do utilizador, referenciada pela figura 5.9, o utilizador pode efetuar o seguinte conjunto de operações:

- Escolher opção de alteração de conteúdo na página “Sobre Nós”;
- Upload de imagem para página “Menu”;
- Operação de *logout* da conta Strapi.

5.5.6 Página Sobre Nós

Na página Sobre Nós, referenciada pela figura 5.10, o utilizador pode efetuar o seguinte conjunto de operações:

- Inserção de texto para o título da página “Sobre Nós”;
- Inserção de texto para o primeiro parágrafo da página “Sobre Nós”;
- Inserção de texto para o segundo parágrafo da página “Sobre Nós”;
- Upload de imagem para a página “Sobre Nós”;
- Confirmar upload das alterações efetuadas;
- Cancelar upload das alterações efetuadas.

5.6 Conclusão

Neste capítulo foram abordados aspectos muito relevantes relativamente à implementação do Headless CMS Strapi tanto em contexto da própria *interface* bem como na aplicação móvel, implementação da aplicação móvel desenvolvida e respetivo *design* escolhido para a mesma, referência ao tipo de testes possíveis e efetuados no sistema e exemplos práticos de como as funcionalidades implementadas impactam a gestão de conteúdo na Strapi.

Capítulo

6

Conclusões e Trabalho Futuro

6.1 Conclusões Principais

Este projeto teve como objetivo principal o desenvolvimento de uma aplicação móvel que possibilitasse a gestão de conteúdo de um *website* através da utilização de um *content management system*. A aplicação do *Headless CMS* Strapi apresentou uma perspetiva diferente no que diz respeito à gestão de conteúdo digital onde se efetuou uma implementação de um sistema que visa permitir ao seu utilizador efetuar operações simples, rápidas e eficientes no que diz respeito ao conteúdo do *back-end* do seu *website*. Além disto, foram aplicadas novas tecnologias, como a *framework* Flutter, para permitir um desenvolvimento mais consistente e mais abrangente, elaborando uma aplicação móvel que permitisse ser usada em diferentes sistemas operativos como *Android* e *iOS*. Com a aprendizagem e utilização do CMS foi-me destacada a importância e consequente eficiência deste tipo de *software* para gestão de conteúdo digital que é atualmente utilizado por muitos *websites*. A aprendizagem do CMS Strapi e da *framework* Flutter permitiu-me desenvolver novas capacidades técnicas e contribuiu para melhorar o meu leque de conhecimento no que diz respeito a tecnologias associadas à área de Programação Web e de Programação de Dispositivos Móveis. Ao longo deste projeto, foram surgindo alguns problemas e restrições relativamente à utilização da Strapi ,visto ser um *software* relativamente recente, algumas das funcionalidades implementadas na aplicação móvel exigiram um maior tempo de desenvolvimento do que o previsto. As fases de Engenharia de *Software* contribuíram imenso para um desenvolvimento mais organizado, eficiente e realista do trabalho desenvolvido. Numa vista global, a realização deste projeto possibilitou-me aumentar o meu conhecimento em diversas áreas e

fortalecer as minhas capacidades técnicas no desenvolvimento de aplicações móveis.

6.2 Trabalho Futuro

Durante o desenvolvimento da aplicação móvel, existiram alguns problemas relativos às funcionalidades da Strapi. Algumas destas estavam mal desenhadas acabando por criar resultados inesperados durante o desenvolvimento da aplicação. Uma possível implementação futura neste trabalho seria adaptar as futuras correções a algumas funcionalidades da Strapi à aplicação móvel (atualização de funcionalidades).

O projeto Strapi desenvolvido foi realizado no *localhost* da máquina. Uma futura implementação seria migrar este projeto para um servidor para ser posteriormente utilizado fora do contexto local.

Tal como referido em 5.2.2 os *Content Type* foram criados pelo uso da *interface* da Strapi. Uma futura implementação seria criar, através do uso da aplicação móvel, *Content Type* dinâmicos para cada utilizador.

Bibliografia

- [1] Tejas Kaneriya. Node.js vs PHP: A Honest Comparative Study With All The Answers, 2022. [Online] <https://www.simform.com/blog/nodejs-vs-php/>, Último acesso a 11 de Março 2022.
- [2] Victor Coisne. Understanding Headless Architecture: What Is It?, 2021. [Online] <https://strapi.io/blog/understanding-headless-architecture-what-is-it>, Último acesso a 3 de Março 2022.
- [3] Flutter FAQ. [Online] <https://docs.flutter.dev/resources/faq>, Último acesso a 4 de Março 2022.
- [4] [Online] <https://www.postman.com>, Último acesso a 15 de Junho 2022.
- [5] [Online] <https://deliartcaffe.com/index.html>, Último acesso a 21 de Junho 2022.
- [6] Sarah Amsler and Fred Churchville. content management system (CMS), 2021. [Online] <https://www.techtarget.com/searchcontentmanagement/definition/content-management-system-CMS>, Último acesso a 3 de Março 2022.
- [7] What is a content management system (CMS)?, 2015. [Online] <https://www.optimizely.com/optimization-glossary/content-management-system/>, Último acesso a 3 de Março 2022.
- [8] Chris Osterhout. Why Choose a CMS?, 2013. [Online] <https://www.wearediagram.com/blog/why-choose-a-cms>, Último acesso a 3 de Março 2022.
- [9] Traditional CMS vs Headless CMS?, 2018. [Online] <https://www.udig.com/digging-in/traditional-cms-vs-headless-cms/>, Último acesso a 3 de Março 2022.
- [10] Rafał Łuksza. Should I use Headless CMS? Use Cases, Pros and Cons, 2020. [Online] <https://www.softkraft.co/headless-cms-pros-cons/>, Último acesso a 3 de Março 2022.

- [11] Yves Do. 10 reasons to use a headless CMS, 2019. [Online] <https://strapi.io/blog/10-reasons-headless-cms>, Último acesso a 3 de Março 2022.
- [12] Strapi - Open source Node.js Headless CMS, 2017. [Online] <https://strapi.io/>, Último acesso a 3 de Março 2022.
- [13] Frequently Asked Questions, 2021. [Online] <https://strapi.io/faq>, Último acesso a 3 de Março 2022.
- [14] Sam Lihou. A review of Strapi, the Node.js headless CMS, 2021. [Online] <https://www.gravitywell.co.uk/insights/a-review-of-strapi-the-nodejs-headless-cms/>, Último acesso a 4 de Março 2022.
- [15] What Is The Difference Between Front-End And Back-End Development?, 2019. [Online] <https://www.conceptatech.com/blog/difference-front-end-back-end-development>, Último acesso a 4 de Março 2022.
- [16] Nutan Nichat. Build your app with Headless CMS(Strapi), Node.js, and Next.js (Part -1), 2021. [Online] <https://medium.com/globant/build-your-app-with-headless-cms-strapi-node-js-and-next-js-part-2-74bc952fc7f1>, Último acesso a 1 de Abril 2022.
- [17] Lucid Software Inc. Deployment Diagram Tutorial. [Online] <https://www.lucidchart.com/pages/uml-deployment-diagram>, Último acesso a 19 de Junho 2022.
- [18] Sendinblue. [Online] <https://pt.sendinblue.com/>, Último acesso a 20 de Junho 2022.
- [19] [Online] <https://online.visual-paradigm.com/app/diagrams/>, Último acesso a 19 de Junho 2022.
- [20] Dileep Gupta. Top 15 Mobile App Development Frameworks in 2021 and Beyond, 2021, 2020. [Online] <https://appinventiv.com/blog/mobile-app-development-frameworks/>, Último acesso a 4 de Março 2022.
- [21] User Interface Design. [Online] <https://www.interaction-design.org/literature/topics/ui-design>, Último acesso a 4 de Março 2022.
- [22] Thuan Nguyen Duc. How Flutter Could Be a Game-Changer in Cross-Platform Development, 2020. [Online] <https://www.pentalog.com/blog/it-development-technology/flutter-game-changer-in-cross-platform-development>, Último acesso a 4 de Março 2022.

- [23] Kerry Doyle. 4 reasons Dart is still a language worth learning, 2021. [Online] <https://www.techtarget.com/searchapparchitecture/tip/4-reasons-Dart-is-still-a-language-worth-learning>, Último acesso a 5 de Março 2022.
- [24] Ian Sommerville. *Software Engineering, 10th Edition*. ISBN: 978-1-292-09613-1. Pearson Education, 2016. Último acesso a 16 de Junho de 2022.