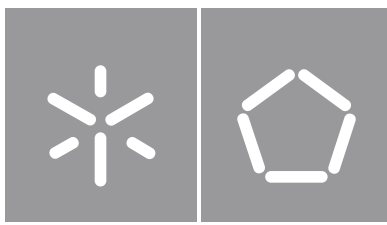




**Universidade do Minho**  
Escola de Engenharia

Joana Ramalho Querido  
**Geração de Lojas Online Orientada Para  
Uma Framework – Funcionalidade de  
Gestão de Produtos**





**Universidade do Minho**

Escola de Engenharia

Joana Ramalho Querido

**Geração de Lojas Online Orientada Para  
Uma Framework – Funcionalidade de  
Gestão de Produtos**

Dissertação de Mestrado

Mestrado Integrado em Engenharia de Telecomunicações e  
Informática

Trabalho efetuado sob a orientação do

**Professor Doutor José Manuel Tavares Vieira Cabral  
e Professor Doutor Sérgio Adriano Fernandes Lopes**



# DIREITOS DE AUTOR E CONDIÇÕES DE UTILIZAÇÃO DO TRABALHO POR TERCEIROS

Este é um trabalho académico que pode ser utilizado por terceiros desde que respeitadas as regras e boas práticas internacionalmente aceites, no que concerne aos direitos de autor e direitos conexos.

Assim, o presente trabalho pode ser utilizado nos termos previstos na licença abaixo indicada.

Caso o utilizador necessite de permissão para poder fazer um uso do trabalho em condições não previstas no licenciamento indicado, deverá contactar o autor, através do RepositóriUM da Universidade do Minho.



**Atribuição**

**CC BY**

*<https://creativecommons.org/licenses/by/4.0/>*

## **AGRADECIMENTOS**

A conclusão desta dissertação contou com o apoio incansável e imprescindível de várias pessoas, a quem desejo deixar um sincero agradecimento.

Antes de mais, aos meus pais, pela educação, motivação e apoio incondicional nesta fase da minha vida.

Aos meus orientadores, o Professor José Manuel Tavares Vieira Cabral e o Professor Sérgio Adriano Fernandes Lopes, pela disponibilidade, paciência e partilha de sugestões ao longo de todo o processo de elaboração desta dissertação.

A todos os meus amigos, pelo constante apoio e presença nos momentos em que mais precisei.

Por último, agradeço a todos os docentes e colegas de curso que, de alguma forma, estiveram presentes neste percurso.

## **DECLARAÇÃO DE INTEGRIDADE**

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

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

## RESUMO

A presente investigação surge num momento pós-pandemia, em que se assistiu a um aumento significativo da utilização de sites de comércio *online*. Neste contexto, é pertinente abordar as metodologias empregues no desenvolvimento destes sites, que têm trazido benefícios tanto para os comerciantes como para os utilizadores. O comércio *online* tem vindo a tornar-se um ponto forte no meio *online*, devido à gestão de recursos ser altamente menor do que no comércio tradicional. Não são necessários funcionários para atender clientes, nem espaço físico com as despesas inerentes ao mesmo. Além disso, a rapidez e fluidez na compra é grande e o comprador não precisa de sair de casa. Outra vantagem é a possibilidade de obter estatísticas concretas sobre o comportamento do utilizador do site e de ajustar o mesmo em conformidade com estratégias de *marketing*, bem como de saber quantas pessoas o visitam e trabalhar em anúncios especificamente para um determinado público alvo.

Neste contexto, o âmbito do presente trabalho está relacionado com a criação de uma ferramenta de geração de uma parte de loja *online*, referente à gestão de produtos e categorias. Esta ferramenta é direcionada a utilizadores com alguma experiência em programação, uma vez que lhes fornece o código total da parte de loja criada. Isto permite que os utilizadores integrem outros componentes e personalizem ainda mais a sua loja. Além disso, a ferramenta fornece ao utilizador um conjunto de customizações a nível de estilização e personalização para a loja ser gerada de acordo com as suas necessidades. O grande benefício desta ferramenta é que o utilizador dispensa a criação de uma loja de raiz e pode obter uma de forma fácil e rápida.

Por fim, o desenvolvimento desta aplicação foi posterior ao desenvolvimento de uma loja *online* fictícia, de forma a serem selecionadas as funcionalidades gerais implementadas em qualquer tipo de loja *online*. Com este trabalho, pretende-se contribuir para a criação de ferramentas mais acessíveis e simples para a criação de lojas *online*.

**Palavras-chave:** comércio online; desenvolvimento web; lojas online



## **ABSTRACT**

This investigation arises in a post-pandemic moment, in which there has been a significant increase in the use of e-commerce websites. In this context, it is pertinent to address the methodologies used in the development of these sites, which have brought benefits to both merchants and users. E-commerce has become a strong point in the online world due to the highly reduced resource management compared to traditional commerce. There is no need for employees to attend to customers, nor physical space with inherent expenses. In addition, the speed and fluidity of the purchase process is high, and the buyer does not need to leave home. Another advantage is the possibility of obtaining concrete statistics about the user's behavior on the site and adjusting it accordingly to marketing strategies, as well as knowing how many people visit it and working on advertisements specifically for a particular target audience.

In this context, the scope of the present work is related to the creation of a tool for generating a part of an online store, regarding product and category management. This tool is aimed at users with some programming experience, as it provides them with the complete code for the created store section. This allows users to integrate other components and further customize their store. Additionally, the tool provides the user with a set of customization options at the styling and personalization level, so the store can be generated according to their needs. The great benefit of this tool is that the user can avoid creating a store from scratch and can obtain one quickly and easily.

Finally, the development of this application was done after the development of a fictional online store, in order to select the general functionalities implemented in any type of online store. With this work, we aim to contribute to the creation of more accessible and simple tools for the creation of online stores.

**Keywords:** e-commerce; web development; online stores

# Índice

<b>1</b>	<b>Introdução</b>	<b>1</b>
1.1	Objetivos . . . . .	1
1.2	Estrutura da dissertação . . . . .	2
<b>2</b>	<b>Estado da Arte</b>	<b>4</b>
2.1	Aplicações relacionadas . . . . .	5
2.1.1	Shopify . . . . .	5
2.1.2	Wix . . . . .	7
2.1.3	Squarespace . . . . .	8
2.1.4	Sellfy . . . . .	9
2.1.5	Shopkit . . . . .	10
2.1.6	Comparação das ferramentas . . . . .	11
2.2	Tecnologias utilizadas . . . . .	12
2.2.1	MongoDB . . . . .	13
2.2.2	Express . . . . .	14
2.2.3	React . . . . .	15
2.2.4	Node . . . . .	15
2.2.5	Redux . . . . .	16
2.2.6	NPM ( <i>Node Package Manager</i> ) . . . . .	17
<b>3</b>	<b>Desenvolvimento da Loja Online</b>	<b>18</b>
3.1	Funcionalidades . . . . .	18
3.2	Concepção . . . . .	20
3.2.1	Arquitetura . . . . .	21
3.2.2	Modelo conceptual . . . . .	22
3.2.3	Mapa de páginas . . . . .	23
3.2.4	API do <i>Backend</i> . . . . .	25
3.3	Implementação . . . . .	30
3.3.1	Base de dados . . . . .	30
3.3.2	Estrutura do código . . . . .	34
3.3.3	Comercialização de produtos . . . . .	35

3.3.4	Backoffice . . . . .	42
3.4	Síntese . . . . .	49
<b>4</b>	<b>Desenvolvimento da Aplicação de Geração de lojas</b>	<b>50</b>
4.1	Funcionalidades . . . . .	50
4.2	<i>Framework</i> . . . . .	51
4.2.1	Arquitetura . . . . .	52
4.2.2	API do <i>Backend</i> . . . . .	53
4.2.3	Base de dados . . . . .	55
4.3	Ferramenta . . . . .	57
4.3.1	Configurações gerais . . . . .	58
4.3.2	Configurações do frontend . . . . .	58
4.3.3	Configurações do <i>backend</i> . . . . .	60
4.3.4	<i>Output</i> da ferramenta . . . . .	61
4.4	Loja parcial gerada . . . . .	64
4.4.1	Integração com outros componentes . . . . .	66
<b>5</b>	<b>Conclusão</b>	<b>69</b>
5.1	Discussão de resultados . . . . .	69
5.2	Conclusões e trabalho futuro . . . . .	70
	<b>Referências</b>	<b>71</b>

## Índice de Figuras

1	Partilha de estado de uma aplicação com e sem Redux [26]	17
2	Diagrama de casos de uso da loja <i>online</i>	20
3	Arquitetura da loja <i>online</i>	21
4	Diagrama de classes da loja <i>online</i>	23
5	Mapa de páginas do <i>backoffice</i> e <i>frontoffice</i>	24
6	Diagrama E-R do modelo de dados	33
7	Estrutura dos ficheiros da loja <i>online</i>	34
8	Página inicial de um utilizador	36
9	Página do perfil de utilizador	36
10	Página de produtos da categoria Ratos	37
11	Secção do catálogo de produtos e favoritos	37
12	Página de um produto	38
13	Diagrama de sequência do processo de adicionar uma <i>review</i>	39
14	Página do carrinho de compras	40
15	Página dos dados de envio de uma encomenda	41
16	Página de uma encomenda	41
17	Diagrama de sequência do pagamento por Paypal	42
18	Página de registar novo utilizador	43
19	Página de <i>login</i>	44
20	Opções do <i>backoffice</i>	44
21	Página de estatísticas	45
22	Página da lista de utilizadores	45
23	Página da lista de produtos	46
24	Página de editar/criar produtos	47
25	Página da lista de encomendas	47
26	Página dos detalhes de uma encomenda	48
27	Página das categorias	49
28	Arquitetura da <i>framework</i>	52
29	Diagrama do modelo de dados da aplicação de criação de lojas <i>online</i>	57
30	Página de customização	58
31	Opções de customização	59

32	Diagrama de atividades de adicionar barra de navegação . . . . .	60
33	Diagrama de sequência de adicionar novo campo . . . . .	61
34	Estrutura dos ficheiros da aplicação de geração de lojas . . . . .	62
35	Página customizada . . . . .	65
36	Página de criar produto . . . . .	65
37	Página de gerir categorias . . . . .	66
38	Edição do ficheiro server.js . . . . .	67
39	Edição do ficheiro store.js . . . . .	67
40	Edição do ficheiro App.js . . . . .	67
41	Página de registar novo utilizador . . . . .	68

## Índice de Tabelas

2.1	Comparação das ferramentas . . . . .	11
3.2	Recursos relativos às categorias . . . . .	26
3.3	Recursos relativos às encomendas . . . . .	27
3.4	Recursos relativos aos utilizadores . . . . .	28
3.5	Recursos relativos aos produtos . . . . .	29
4.6	Recursos relativos à customização . . . . .	54
4.7	Recursos relativos às categorias . . . . .	54
4.8	Recursos relativos aos produtos . . . . .	55
4.9	Ficheiros gerados pela aplicação e respetivas funções . . . . .	63

## **Acrónimos**

**CMS** *Content Management System.*

**CSS** *Cascading Style Sheets.*

**HTML** *Hypertext Markup Language.*

**HTTP** *HyperText Transfer Protocol.*

**IVA** *Imposto sobre Valor Acrescentado.*

**JPEG** *Joint Photographic Experts Group.*

**JSON** *JavaScript Object Notation.*

**JWT** *Json Web Token.*

**NPM** *Node Package Manager.*

**PNG** *Portable Network Graphics.*

**REST** *Representational State Transfer.*

**SDK** *Software Development Kit.*

**SEO** *Search Engine Optimization.*

**SKU** *Stock Keeping Unit.*

**URL** *Uniform Resource Locator.*

# 1 Introdução

Atualmente o mercado *online* encontra-se em crescimento, a par da tecnologia onde, através da utilização de um dispositivo ligado à Internet, qualquer individuo consegue realizar uma compra em qualquer lugar e a qualquer hora, excluindo a necessidade de deslocamento a uma loja física, e recursos como tempo e dinheiro que essas deslocações envolvem.

Desta forma, a compra através de uma loja *online* tornou-se um fator fundamental na vida das pessoas e, conseqüentemente, levou ao aumento da procura em integrar o comércio *online* com lojas físicas, de modo a alargar o público-alvo e não se limitar a um grupo restrito geograficamente. Este comércio também começou a evoluir de forma a que todos os serviços possam ser exercidos somente de forma *online*, eliminando os custos e logísticas indesejados de uma loja física.

Deste modo, tanto para um cliente que pretenda comprar algum serviço ou artigo, como para um empreendedor que queira possuir uma loja para venda, o comércio *online* fornece cada vez mais vantagens e oportunidades face ao comércio tradicional *offline*.

É comum observar que as lojas *online* compartilham uma série de funcionalidades na sua estrutura. Com o intuito de identificar os requisitos essenciais para o funcionamento de qualquer loja *online*, foi desenvolvido previamente um modelo de loja. A partir dessa iniciativa, foi possível extrair elementos generalizáveis e funcionalidades comuns a todas as lojas *online*, permitindo estabelecer um conjunto de requisitos para garantir a operação adequada de uma loja *online* e transportar isso para uma ferramenta de geração de uma parte de loja. A ferramenta em questão é desenvolvida com o objetivo de proporcionar uma experiência eficiente e conveniente para os seus utilizadores ao fornecer soluções simples e fáceis de usar na customização e criação da parte de loja, economizando tempo e recursos aos utilizadores.

O controlo total da ferramenta é entregue aos utilizadores, o que significa que eles podem personalizar a ferramenta de acordo com suas necessidades específicas. Desta forma, os utilizadores podem aproveitar ao máximo as funcionalidades da ferramenta e adaptá-la ao seu fluxo de trabalho.

## 1.1 Objetivos

O objetivo principal desta dissertação é desenvolver uma ferramenta que permita a personalização e automação de funcionalidades generalizadas de uma parte de loja *online*, atendendo à crescente necessidade de integração do comércio *online* e simplificar o processo de criação de lojas, sendo mais fácil e rápido. É importante destacar que muitas das ferramentas existentes no mercado são complexas e exaustivas, o que nem sempre é o objetivo do utilizador. A nova ferramenta proposta visa oferecer



uma solução eficiente, em que utilizadores com algum conhecimento em programação consigam obter uma parte da loja *online* customizada ao seu gosto e a partir daí construir as suas próprias lojas *online* sem ter que as desenvolver de raiz.

A aplicação não abrange todas as funcionalidades de uma loja *online* completa, tendo sido desenvolvida com o propósito de ser integrada com outras partes de loja desenvolvidos por terceiros, de forma a viabilizar a edição de código, uma prática cada vez mais frequente nos dias de hoje. Além disso, disponibilizar todas as funcionalidades de todas as partes de uma loja *online* completa numa ferramenta deste tipo seria uma tarefa exaustiva e extensa.

Por isso, a ferramenta engloba as funcionalidades de gestão de produtos, que inclui a gestão de categorias de uma loja *online*, tendo a possibilidade de estilizar a parte da loja a ser criada e adicionar parâmetros ao componente dos produtos, de modo a atender às especificidades do tipo de produto desejado. Como resultado, a ferramenta deve gerar a parte da loja funcional e personalizada, conforme as especificações do utilizador, com a possibilidade de adicionar manualmente componentes externos para complementar a loja, como por exemplo, a gestão de utilizadores e de encomendas.

Assim, os objetivos específicos são os seguintes:

1. Definir a estrutura e as funcionalidades que uma loja *online* deve seguir.
2. Desenvolver uma loja *online* fictícia com o objetivo de generalizar partes comuns a todas as lojas.
3. Desenvolver uma aplicação que automatize a criação de lojas *online*, permitindo a geração funcional da parte da loja referente à gestão de produtos, de acordo com as especificações definidas pelo utilizador.
4. A ferramenta fornecer o código completo da parte de loja criada.

Com estes objetivos, pretende-se contribuir para a simplificação do processo de criação de lojas *online* e torná-lo mais acessível a utilizadores com algum conhecimento em programação, oferecendo uma solução personalizada e eficiente para as suas necessidades.

## **1.2 Estrutura da dissertação**

Após a introdução inicial, o próximo capítulo apresentado é o Estado da Arte, onde é descrito o processo de aquisição de conhecimentos sobre o funcionamento das lojas *online*. As primeiras abordagens incidem em analisar aplicações já existentes do ramo de criação de lojas *online*. Ainda neste capítulo foram descritas as tecnologias utilizadas no desenvolvimento da ferramenta.

No terceiro capítulo é apresentada a fase de Desenvolvimento da Loja *Online*, que é dividida em três grandes secções: funcionalidades da aplicação, concepção e discussão de resultados. A secção de concepção é dividida em subsecções que descrevem a elaboração do projeto, tais como a arquitetura, o modelo conceptual, o mapa de páginas e a API do *backend*.

A secção de implementação visa abordar aspetos mais relacionados com o código em si, nomeadamente nas secções de base de dados e estrutura do código. Por fim, é apresentada uma síntese das ideias finais obtidas na realização da loja *online* para consolidar as principais conclusões e resultados obtidos.

No capítulo seguinte, é feita a descrição da aplicação de geração de lojas *online*. Esta secção é dividida em subsecções que descrevem as funcionalidades, a *framework* e a ferramenta. Na subsecção da *framework* é detalhada a sua arquitetura, a API do *backend* e a base de dados. Já na secção da ferramenta, são descritas as suas configurações, a loja parcial gerada e integração com outros componentes.

Por fim, no capítulo das conclusões, é feita uma reflexão sobre todo o trabalho desenvolvido. São avaliados os resultados obtidos e são apresentadas sugestões de melhoramentos para o futuro.

## 2 Estado da Arte

A crescente popularidade do comércio eletrônico tem sido um impulso para as empresas investirem cada vez mais na presença *online*, tornando-se praticamente essencial para competir no mercado atual.

Conseqüentemente, o desenvolvimento de ferramentas de construção de lojas *online* tornou-se fundamental para facilitar e agilizar o processo de criação e manutenção dessas lojas pelos utilizadores.

A maioria dessas ferramentas baseiam-se principalmente num sistema de gestão de conteúdos conhecido por CMS (*Content Management System*). O CMS é um conjunto de funções que permite ao utilizador criar, editar e publicar conteúdo na sua loja sem a necessidade de conhecimentos avançados em programação.

Esses sistemas oferecem uma ampla variedade de recursos que permitem aos utilizadores personalizar a sua loja, adicionar e remover produtos, gerir o inventário, processar pagamentos e muito mais. Além disso, oferecem integrações com outras ferramentas de terceiros, como métodos de pagamento, processo de envio e serviços de análise.

Durante a pesquisa, foram consultadas diversas fontes de informação, incluindo livros, artigos científicos e materiais disponíveis na *internet*. Os resultados obtidos indicaram uma variedade de perspectivas sobre o uso dessas ferramentas, permitindo a formação de uma compreensão mais ampla e fundamentada. Esta pesquisa teve uma duração de aproximadamente dois meses que englobou a análise baseada na experiência pessoal destas ferramentas, através dos planos gratuitos. Os motores de pesquisa utilizados foram o Google para pesquisa de informações gerais das ferramentas e para o acesso às mesmas, o Google Scholar para artigos académicos mais aprofundados sobre as ferramentas e o Google Books para aceder aos livros utilizados como referências no estudo das tecnologias. Os termos de pesquisa foram escolhidas para refletir as principais temáticas do estudo como: "Plataformas de *ecommerce*", "CMS para *ecommerce*", "Melhores ferramentas de *ecommerce*", "Tecnologias implementadas" e "Desvantagens do *ecommerce*".

É relevante destacar que, mesmo depois da pesquisa ter sido concluída, novos termos de pesquisa foram adicionados ao estudo com base na necessidade de obter informações adicionais. Todo o processo de pesquisa foi cuidadosamente documentado, o que permite seguir as informações obtidas e garante a credibilidade do estudo.

Nesta secção, serão apresentados os resultados da pesquisa bibliográfica sobre ferramentas CMS e tecnologias utilizadas, visando um maior entendimento sobre o assunto. Além disso, é feita uma comparação das diversas ferramentas de criação de lojas *online* com base em vários parâmetros, incluindo facilidade de uso, personalização, SEO (*Search Engine Optimization*) e *marketing*,

desempenho, suporte ao cliente, escalabilidade e edição de código. O objetivo é ampliar o conhecimento e as práticas tecnológicas necessárias para o desenvolvimento do projeto.

## 2.1 Aplicações relacionadas

A seleção das ferramentas Shopify, Wix, Squarespace e Selffy foi baseada na sua popularidade e reputação, tendo em conta que as mesmas possuem muitos utilizadores e são frequentemente recomendadas em análise e comparações de plataformas de comércio eletrónico.

Além destas ferramentas, foi incluída a Shopkit, uma ferramenta desenvolvida em Portugal e que tem como premissa a criação de um *website* de vendas em 5 minutos. A inclusão desta ferramenta foi importante para avaliar uma opção nacional que promete rapidez e facilidade de criação de lojas *online*.

Após a análise individual de cada ferramenta selecionada e a comparação entre elas, é possível extrair informações e características importantes para a implementação adequada da ferramenta a ser desenvolvida.

### 2.1.1 Shopify

A Shopify é das melhores ferramentas de criação de lojas *online* que conta com cerca de milhões de lojas alojadas por todo o mundo [28].

É uma ferramenta muito completa, sendo que não se foca apenas na criação de lojas *online* mas em todo o comércio envolvente, sendo até possível associar uma loja física usando o Shopify POS.

Quando é iniciado o período de teste gratuito de 14 dias, e depois de ser efetuado o *login*, são feitas algumas perguntas referentes à experiência em comércio do utilizador e se pretende instalar a loja no Facebook, Instagram, Google ou Shopify Button.

De seguida aparece a página inicial, denominada por *Home*, onde se encontram bem visíveis as opções de adicionar produtos à loja, customizar o tema, adicionar um domínio e vídeos de tutoriais de como usar o Shopify.

A seguir ao *Home*, a segunda secção do menu do *backoffice* é a *Orders*, onde é requerido selecionar um plano pago para receber encomendas e poder explorar mais esta funcionalidade. A secção seguinte refere-se aos produtos inseridos previamente e às suas informações, tais como: estado de *stock*, tipo de produto e vendedor. Também é possível editar o inventário na quantidade disponível e criar coleções de produtos. As informações para adicionar um produto são: título, descrição, ativo ou inativo, preço, custo por item, acompanhamento da quantidade ou continuar a vender sem *stock*, quantidade, peso, entre

outros.

De seguida encontram-se secções com a finalidade de adicionar informações sobre clientes e ver o seu histórico de compras, e também contém análises importantes para avaliar as estatísticas da loja. Estas análises são referentes ao total de vendas, número de visitantes, média de verbas por encomenda, entre outros. Contém também a visualização ao vivo dos visitantes *online* no mundo, e em que fase da encomenda se encontram.

Na página acerca do *Marketing* apresentam-se as aplicações de *marketing* disponíveis, os resultados das técnicas de *marketing* utilizadas e a possibilidade de criar campanhas para promover o produtor, por exemplo, via e-mail ou publicar no Facebook. Também existe a possibilidade de se criarem descontos e promoções, códigos promocionais e a página *Apps* para adicionar aplicações extras.

Por último, encontra-se a secção *Online store* onde é feita a customização do site. É apresentado uma biblioteca de temas grátis ou pagos para seleccionar e posteriormente customizar. Após seleccionar o tema, o processo de edição oferece uma ampla gama de opções para adicionar componentes personalizados, incluindo imagens com texto, destaques de produtos, coleções, entre outros. Além disso, é possível personalizar a cor e o estilo do tema, escolher a fonte das letras, adicionar *links* para redes sociais, integrar um *icon* de loja e até mesmo personalizar o *checkout*. Também é disponibilizada uma biblioteca de imagens gratuitas, organizadas por temas, que os utilizadores podem usar se pretenderem aprimorar a sua loja.

Este tipo de customização é bastante simples e não é necessário adquirir qualquer conhecimento em programação. Para o utilizador que quiser alterar o tema de raiz ou criar um novo, existe a opção *code* onde são disponibilizados todos os ficheiros desse tema para o utilizador customizar. Para isso, são necessários conhecimentos de programação nas linguagens Liquid, Javascript e CSS (*Cascading Style Sheets*).

Para além da opção de testar os serviços gratuitamente, a plataforma oferece três planos padrão pagos, além de mais dois pacotes adicionais, o Lite e o Plus. O plano Lite tem como objetivo ajudar os utilizadores a vender *online* sem a necessidade de construir uma loja completa com o Shopify. O plano Plus é uma solução personalizada direcionada para grandes empresas que requerem recursos mais avançados e suporte especializado.

Na sua arquitetura, são utilizadas tecnologias como Ruby on Rails, React, GraphQL e Liquid. Ruby on Rails é a tecnologia principal utilizada para o desenvolvimento do *backend*, enquanto o React é utilizado para construir a interface do utilizador no *frontend*. GraphQL é uma linguagem de consulta utilizada para obter informações dos servidores. Por fim, Liquid é uma linguagem de *templates* desenvolvida pelo

próprio Shopify. No que diz respeito à base de dados, utiliza a linguagem relacional MySQL. Concluindo, trata-se de uma ferramenta altamente desenvolvida tecnologicamente [29].

## 2.1.2 Wix

O Wix é uma ferramenta especialmente boa para iniciantes, bastante reconhecida pela vasta variedade de *templates*, com vantagens na interação amigável que comporta uma personalização simplificada para o utilizador comum e, tudo isto com um baixo custo [39]. Os utilizadores usam o método "clicar e arrastar" para editar os *templates*, o que é bastante intuitivo para visualizar as alterações efetuadas. O utilizador tem também a opção de responder a algumas perguntas acerca da personalização do site, que o Wix usa para produzir os *templates* de acordo com as preferências do utilizador.

Quando se inicia sessão, são feitas perguntas sobre a finalidade do site, se é para fim pessoal, para um cliente ou empresa, e é abordado também o tema da loja e a experiência em criação de sites. De seguida dá a escolha entre criar o site com o auxílio do editor de *drag-and-drop*, ou então receber um site já personalizado.

Selecionando a segunda opção, são pedidas informações acerca do tipo de negócio, recursos a integrar, por exemplo: *chat*, planos pagos, vídeos, entre outros. Em seguida são apresentados 6 temas, em que cada um tem 3 *layouts* diferentes mas dentro do mesmo tema. A edição dos temas é bastante completa, contendo várias opções de *layouts* e praticamente tudo é personalizável. Também é possível adicionar secções e páginas completas já programadas para tópicos como: suporte, nova coleção, Instagram entre outras. Para adicionar código personalizado ao corpo do site é necessário optar por um dos planos pagos e registar um domínio.

A página inicial do menu principal dispõe de uma secção onde são descritas etapas para configurar o negócio e para o colocar *online*, o que facilita ao utilizador o progresso que tem que realizar para configurar o seu site.

Esta ferramenta disponibiliza vários métodos de pagamento, desde cartões de crédito e débito, paypal, multibanco, Splitit, PaysafeCash entre outros.

O Wix oferece diferentes planos para atender as necessidades de utilizadores pessoais e para empresas que pretendem aceitar pagamentos *online*.

Uma limitação é a ausência de recursos de *upselling* e venda cruzada, que são técnicas comuns utilizadas para aumentar o valor total de compra do utilizador. Essas técnicas consistem em sugerir produtos relacionados ou complementares ao produto que o cliente está interessado em comprar, e também sugerir produtos de valor mais elevado.

O Wix utiliza diversas tecnologias para oferecer uma experiência ao utilizador de alta qualidade e personalizável. Para criar interfaces de utilizador a ferramenta principal utilizada é o React, e para armazenar dados, o MongoDB, porém o MySQL também é utilizado. Para o *backend* utiliza principalmente tecnologias baseadas em Node.js [40].

### 2.1.3 Squarespace

Squarespace é direcionado para ajudar proprietários de empresas a criar a sua própria loja *online*, com destaque na personalização do site ser bastante criativa e completa [31].

O menu principal dispõe de uma secção de ajuda para configurar a loja, com as etapas a cumprir, e sugestões para ampliar o negócio. Essas sugestões incidem em *marketing*, SEO, extensões, entre outros.

A edição das páginas é feita em tempo real e por secções, em que todas elas são personalizáveis. Depois de seleccionar o tema pretendido, é possível editar facilmente o texto e as imagens diretamente na visualização do site, basta clicar no elemento que se pretende alterar. São oferecidos inúmeros exemplos de secções para utilizar, como títulos, listas, gráficos, entre outros. Para além da edição das páginas principais do site também é possível editar a página do carrinho de compras, a página 404 e a página de acesso negado. É possível ainda editar o ícone do *browser* e as redes sociais. O utilizador também tem a possibilidade de escrever o seu próprio código CSS personalizado, caso deseje.

Para adicionar um produto é preciso seleccionar o seu tipo, que define determinadas configurações e recursos e é automaticamente associado um SKU (*Stock Keeping Unit*) ao produto adicionado.

O inventário de produtos conta com a capacidade de enviar notificações de encomenda e fazer reembolsos diretamente do painel. Há também possibilidade de atualizar o *stock*, o preço, adicionar categoria e definir se é produto em destaque.

Este *software* oferece uma funcionalidade que permite a injeção de código personalizado. Essa opção está disponível em áreas específicas, nomeadamente no cabeçalho, no rodapé, na página de bloqueio e na página de confirmação de pedido. Contudo, para aceder a esta funcionalidade, é necessário que o utilizador tenha um plano pago.

A secção de análise da loja é bastante completa. As estatísticas sobre vendas mostram informações sobre o total de receitas, unidades vendidas, pedidos, receita média por pedido e receita média por visita. Todos os campos mostram a evolução em percentagem relativamente ao intervalo de tempo anterior. Também contém estatísticas acerca do tráfego, visitas por país, origens de tráfego, carrinho abandonado, conteúdo do site, entre outros.

O *software* disponibiliza um plano gratuito com duração de 15 dias e outros quatro planos pagos.

As desvantagens incidem na falta de suporte para vários métodos de pagamento, sendo que apenas são suportados o Stripe e o Paypal, e a integração com poucas aplicações de terceiros.

No que diz respeito às tecnologias utilizadas pelo Squarespace, embora a plataforma não forneça muita informação oficialmente, sabe-se que o *frontend* utiliza principalmente as linguagens HTML (*Hypertext Markup Language*), CSS e JavaScript. Em relação à base de dados, faz uso principalmente do MySQL [32].

#### 2.1.4 Sellfy

Este *software* não possui uma base de clientes muito vasto comparando com os principais softwares do mercado, no entanto é bastante satisfatório principalmente para pequenas empresas [27].

A Sellfy oferece aos seus utilizadores quatro opções de planos, incluindo um plano gratuito de uso ilimitado. Com este plano, é possível publicar a loja com até 10 produtos físicos ou produtos por impressão personalizada. Produtos por impressão personalizada são produtos que podem ser personalizados com imagens ou textos especificados pelo cliente antes da sua produção.

Além disso, o plano permite personalizar completamente a loja, escolher um domínio da Sellfy, configurar códigos de desconto e definir taxas.

A adição de produtos é feita por tópicos, em que a informação necessária para adicionar varia em função do escolhido. Os tópicos disponíveis são: produto digital, imprimir a pedido, subscrição, produto físico e brindes. A opção de produtos por impressão personalizada fornece um conjunto de produtos, incluindo roupa, acessórios de roupa, capas de telemóvel e muitos outros, para o utilizador customizar o seu *design*. Depois de criar o produto, a Sellfy trata da produção e envia aos clientes, eliminando a necessidade de contratar uma empresa de produção terceira.

O construtor de loja é bastante simples, permite customizar a loja a partir de um tema básico, adicionar um nome de domínio, inserir um logótipo e escolher as cores de fundo. Existem 5 temas e todos incluem a página principal, produtos, categorias, *checkout* e confirmação de compra. A edição das páginas é feita por módulos, em que é possível modificar o tamanho do módulo, o texto, o alinhamento do texto e imagem, tipografia, o *background* e os botões. Existem apenas 6 módulos diferentes para modificar e adicionar, sendo eles: painel de texto, rede de produtos, formulário de contacto, bloco de notícias, bloco de destaque e código embutido. É também possível gerar botões para incorporar noutra plataforma ou então incorporar a loja Sellfy noutra site.

Todos os utilizadores têm acesso a ferramentas de *marketing*, códigos de desconto, *marketing* por e-mail, carrinho abandonado, e *upselling* que consiste em apresentar outros produtos ou descontos se o



cliente adicionar um produto ao carrinho.

A nível de pagamentos, é bastante restrito mas confiável. Possui apenas 2 métodos, Stripe e Paypal, e não cobra taxas de transação além das embutidas nos métodos de pagamento.

No que diz respeito às tecnologias utilizadas pela Sellfy, a plataforma baseia-se principalmente em HTML, CSS e Javascript.

## 2.1.5 Shopkit

A Shopkit é uma solução inteiramente portuguesa, com mais de oito anos de experiência e mais de trinta e seis mil lojas criadas apenas em Portugal [30].

Esta loja oferece um plano experimental gratuito de 30 dias e 3 planos de subscrição mensal e anual. As principais diferenças entre os planos são a quantidade de produtos que é possível inserir na loja e o número de localizações para moradas de expedição. Todos os planos possuem loja mobile, personalização de *templates*, domínio próprio e suporte técnico. Dentro dos planos existem também extras opcionais, como a aquisição de loja no Facebook e no Instagram.

Para publicar a loja *online* é necessário cumprir uma série de passos que estão descritos logo na página principal, entre eles, adicionar produtos, portes de envio, métodos de pagamento e o domínio.

A secção da personalização do site é bastante simples. É possível personalizar a cor base, adicionar um logótipo, uma imagem principal e se o fundo é uma imagem ou uma cor. De seguida, são apresentados os 8 *templates* disponíveis, em que 3 são grátis, e depois de gravar os dados é possível visualizar o resultado final numa nova página. Existe uma opção de "Avançado" que permite adicionar ou modificar o código dos *templates*. A linguagem usada é baseada em Twig, e sendo esta uma ferramenta *open-source* é disponibilizado um repositório Github com todos os temas grátis e a sua documentação para que seja possível a customização. Com isto, o utilizador tem a opção de utilizar os *templates* disponíveis, customizar os mesmos, criar um tema de raiz, ou pedir um orçamento para a Shopkit customizar o tema.

Na adição de produtos existe a possibilidade de organizar o produto em categorias, adicionar um SKU ou código de barras, definir como produto destaque, novidade ou promoção, e adicionar um vídeo ao produto.

A secção de estatísticas foca-se em duas vertentes, visitantes e encomendas. É possível visualizar nos visitantes que dispositivo (*desktop, mobile, tablet*) os consumidores utilizam, dados demográficos, origem de tráfego (direto, referência, motores de busca) e os respetivos gráficos. Nas encomendas é apresentado um gráfico do valor das vendas ao longo do tempo e os produtos mais populares.

Esta ferramenta apresenta os métodos de pagamento por Paypal, cartão de crédito, referência multibanco e MB WAY. Não é cobrada nenhuma taxa sobre as vendas, no entanto podem existir custos de transação associados aos métodos de pagamento.

## 2.1.6 Comparação das ferramentas

Ao avaliar as ferramentas selecionadas para o projeto, é fundamental considerar as funcionalidades mais importantes para atingir os objetivos desejados. Nesse sentido, foram escolhidos critérios como facilidade de uso, temas e customização, SEO e *marketing*, desempenho, suporte ao cliente e escalabilidade.

A avaliação destas ferramentas foi feita com base em experiência pessoal, bem como análises e comentários de outros utilizadores [6]. Para facilitar a comparação, elaborou-se a tabela 2.1, que apresenta os resultados de cada ferramenta em relação a cada critério selecionado. A escala de comparação varia de fraco a muito bom, ou de alta a baixa, dependendo do critério analisado.

Esta análise ajuda a identificar quais ferramentas são mais adequadas para as necessidades específicas e quais apresentam melhores desempenhos em relação aos critérios estabelecidos.

Tabela 2.1: Comparação das ferramentas

	Shopify	Wix	Squarespace	Sellfy	Shopkit
Fácil de usar	Bom	Muito bom	Bom	Razoável	Bom
Temas e customização	Bom	Muito bom	Bom	Razoável	Fraco
SEO e <i>Marketing</i>	Bom	Bom	Fraco	Razoável	Razoável
Desempenho	Bom	Bom	Bom	Razoável	Bom
Suporte ao cliente	Muito bom	Bom	Razoável	Bom	Bom
Escalabilidade	Alta	Baixa	Alta	Baixa	Baixa
Edição de código	Inclui	Inclui	Inclui	Inclui	Inclui

No que diz respeito à facilidade de uso, esta relaciona-se com a forma como a informação acerca da ferramenta é disponibilizada, com o *design* do site e com a edição e publicação da loja. Todas as ferramentas apresentam um menu com um conjunto de etapas a cumprir até que a loja esteja pronta para ser publicada, o que é bastante intuitivo para o utilizador.

Em todas as ferramentas, o processo de customização da loja é feito numa secção própria, junto às outras secções do *backoffice*. A Shopkit é a ferramenta que possui menos variedade de temas, e a

sua edição é limitada em comparação com as outras. Em relação à customização do *layout* por parte de programadores, todas as ferramentas necessitam do conhecimento de programação da linguagem utilizada.

No quesito SEO e *marketing*, as ferramentas mais bem avaliadas são a Shopify e Wix. Ambas oferecem recursos que ajudam a otimizar a loja para os mecanismos de busca e permitem criar campanhas de *marketing* digital.

Em termos de desempenho, todas as ferramentas apresentam um bom funcionamento em termos de velocidade e estabilidade. Isto significa que estas plataformas são capazes de fornecer tempos de carregamento rápidos, consistentes e com baixa probabilidade de falhas, que garantem uma experiência satisfatória aos utilizadores.

No suporte ao cliente, a Shopify destaca-se com um suporte muito completo. Já a Squarespace deixa a desejar em termos de apoio ao cliente. Esta avaliação é medida por alguns utilizadores sentirem que o tempo de resposta do suporte ao cliente é demorado ou inexistente [33].

Em relação à escalabilidade, as ferramentas que possuem maior classificação apresentam um maior suporte para grandes empresas e de maior escala. Deste modo, a Sellfy e a Shopkit são boas soluções principalmente para pequenas e médias empresas, e as restantes para médias e grandes empresas destacando-se a Shopify.

A edição de código é uma funcionalidade importante que permite aos utilizadores personalizar a aparência e algumas funcionalidades da sua loja. Esta funcionalidade está sempre associada a um componente específico e em separado, por exemplo, edição de cabeçalho ou rodapé. A Shopify e a Wix, oferecem mais opções e permitem que os utilizadores acessem ao código HTML, CSS e JavaScript, enquanto as restantes ferramentas oferecem menos controlo. No entanto, nenhuma das ferramentas disponibiliza o código fonte total das lojas criadas pelos utilizadores.

Em suma, a escolha de uma ferramenta é feita em função das necessidades de cada cliente. Dos parâmetros apresentados observa-se que a preferência tende para as ferramentas que possuam maior facilidade de uso e maior customização, por serem fatores que cativam o utilizador desde os primeiros passos.

## **2.2 Tecnologias utilizadas**

As tecnologias utilizadas no desenvolvimento de um projeto são um fator chave para a sua qualidade, escalabilidade e manutenção. A escolha correta das tecnologias pode tornar o desenvolvimento mais

eficiente e aumentar a satisfação do utilizador final.

Entre as tecnologias utilizadas, destacam-se o Node.js [1], o React [20], o Express [7], e o MongoDB [5], cada uma com as suas particularidades e vantagens específicas.

O Node.js foi escolhido como plataforma de desenvolvimento devido à sua escalabilidade, eficiência e capacidade de lidar com um grande volume de solicitações simultâneas.

O React foi utilizado para desenvolver a interface do utilizador, fornecendo uma experiência de utilizador rica e interativa. Além disso, possui uma grande comunidade de desenvolvedores, o que garante a disponibilidade de recursos e ferramentas para auxiliar o seu uso.

O Express, por sua vez, foi utilizado para criar a estrutura do servidor, permitindo que o código do servidor fosse facilmente organizado e mantido.

Por fim, o MongoDB foi escolhido como base de dados graças à sua escalabilidade, desempenho e capacidade de lidar com grandes volumes de dados. É uma base de dados NoSQL orientado a documentos, que oferece uma abordagem mais flexível e escalável em comparação com as bases de dados relacionais tradicionais.

Além das tecnologias mencionadas anteriormente, foram utilizadas outras tecnologias para complementar a *stack* de tecnologias do projeto, incluindo o NPM [3] e o Redux [23].

Em resumo, a utilização destas tecnologias foi crucial para a construção de um sistema robusto, escalável e eficiente que atendeu plenamente os objetivos propostos.

Nesta secção, serão apresentadas as tecnologias mencionadas seguindo a ordem da sigla MERN (MongoDB, Express, React e Node.js). Além disso, também serão aprofundadas as tecnologias complementares, fornecendo detalhes sobre cada uma das tecnologias e as suas principais funcionalidades no contexto do projeto.

## 2.2.1 MongoDB

MongoDB é uma plataforma de gestão de base de dados que dispõe de uma variedade ampla de usos e é flexível, implementando um sistema através de coleções que permite rapidez e fluidez de trabalho efetivas. Uma coleção é um conjunto de documentos formatados em JSON (*JavaScript Object Notation*) em que a sua estrutura pode variar, o que contribui para a flexibilidade, e sendo uma grande vantagem face a outro tipo de plataforma.

Para entender esta tecnologia, é feita uma enumeração de vantagens e desvantagens relativamente ao uso da mesma.

Vantagens:

- Tem como premissa garantir velocidade e desempenho. Devido à metodologia empregue pelo modelo de coleções, comparativamente a uma base de dados tradicional, é permitida uma velocidade maior.
- Flexibilidade. Possui a capacidade de armazenar dados de diferentes tipos e estruturas, sem depender de uma estrutura fixa. O utilizador pode registar novos dados e manter os existentes sem precisar de alterar a estrutura da base de dados.
- Suporte. Se ocorrer algum problema com o uso do MongoDB, a empresa oferece um suporte técnico através de um agente especializado no mesmo.
- É possível configurar sistemas de automação, o que permite automatizar tarefas repetitivas e otimizar o desempenho do sistema.

Desvantagens:

- Dados duplicados mais frequente. Como cada documento é armazenado com a sua própria estrutura, é comum que as informações sejam duplicadas em vários documentos.
- Uso elevado de memória. Guarda todos os dados em memória, o que pode ser um problema para base de dados que possam exceder a capacidade de memória do servidor.
- Limitação de consulta. Apresenta limitações em consultas complexas que exigem junções de várias coleções.

Como referido anteriormente, nenhum *software* é totalmente eficiente em todos os âmbitos ou propósitos. No desenvolvimento do projeto, o MongoDB assegura a base de dados, encaixando nos propósitos pretendidos.

## 2.2.2 Express

Express é uma *framework* do Node que fornece uma estrutura mínima para a criação de servidores *web* e permite que os desenvolvedores criem aplicações de forma rápida e fácil.

Uma das principais vantagens do Express é a sua capacidade de criar rotas para manipular solicitações HTTP (*HyperText Transfer Protocol*) [38], como GET, POST, PUT e DELETE.

Com o Express, os programadores podem facilmente configurar rotas para manipular essas solicitações e fornecer respostas apropriadas para o cliente. Além disso, o Express fornece uma ampla variedade de *middlewares*, que são funções que podem ser usadas para executar ações específicas

antes ou depois que uma solicitação é tratada. Isso inclui *middlewares* para autenticação, armazenamento em *cache*, compressão e muito mais [15].

No contexto do desenvolvimento, o ficheiro principal de entrada do Express tem a função de carregar os próprios módulos, como controladores e modelos. Além disso, é responsável por definir as rotas da aplicação, aplicar o *middleware* criado, conectar-se à base de dados e iniciar o servidor.

### 2.2.3 React

React é uma biblioteca JavaScript de código aberto que foi desenvolvida pelo Facebook e lançada como *software* livre em 2013. Desde então, tornou-se uma das bibliotecas mais populares e influentes da atualidade, utilizada para criar interfaces de utilizador em aplicações *web* mais recentes.

A abordagem declarativa utilizada pelo React para a criação de interfaces de utilizador tem sido um dos principais motivos para sua popularidade. Essa abordagem permite que os desenvolvedores descrevam como a interface deve ser renderizada em resposta a mudanças de estado. Isso significa que, ao invés de escrever código imperativo para atualizar a interface de utilizador, os desenvolvedores simplesmente descrevem como a interface deve parecer em diferentes estados [36].

Com o React, é possível criar componentes reutilizáveis que permitem uma maior modularidade e reutilização de código. Faz com que o desenvolvimento e a manutenção de aplicações *web* sejam mais fáceis e eficientes.

O React é frequentemente utilizado em conjunto com outras tecnologias, como o Node.js, o Redux e o TypeScript, para criar aplicações mais recentes e escaláveis. Além disso, o React tem uma grande comunidade de desenvolvedores e uma ampla variedade de bibliotecas e ferramentas disponíveis para o seu uso.

Em resumo, o React é uma biblioteca poderosa e flexível que pode ajudar a tornar o desenvolvimento de aplicações *web* mais fácil e eficiente. Sua abordagem declarativa e reativa para a criação de interfaces de utilizador, juntamente com sua capacidade de criar componentes reutilizáveis, tornam-no uma escolha viável para desenvolver componente *frontend*.

### 2.2.4 Node

O Node.js é uma plataforma de desenvolvimento de *software* criado para permitir a execução da linguagem JavaScript no lado do servidor e criar aplicações *web* escaláveis e de elevado desempenho. Desta forma, permite ao programador evitar a troca de linguagem de programação e as convenções de

código associadas.

A arquitetura evita *multi-threads* devido à sua complexidade inerente. É construída em *single-thread*, o que significa que as solicitações são tratadas em paralelo e o servidor não fica bloqueado enquanto espera a resposta de uma solicitação, sendo o consumo de memória mais baixo, a taxa de transferência mais alta, o perfil de latência sob carga é melhor e o modelo de programação é mais simples. Outros sistemas que usam *multi-threads* tendem a ter sobrecarga de memória e serem complexos, o que não é o caso do Node [11].

Concluindo, o Node.js é uma plataforma altamente flexível e escalável que permite o desenvolvimento de aplicações de forma eficiente. Com a sua arquitetura assíncrona, compatibilidade com bibliotecas de código aberto e capacidade de execução em várias plataformas, o Node é a escolha para desenvolver o *backend* da ferramenta.

## 2.2.5 Redux

Redux é uma biblioteca para armazenamento e globalização de estados de aplicações JavaScript, no caso do projeto aplicações React, e implementa uma arquitetura de fluxo [24] constituída por *components*, *actions*, *reducers* e *store*.

O principal uso do Redux é que ao definir os estados da aplicação como globais, qualquer componente pode interagir com o estado de qualquer componente muito facilmente independentemente de ser um componente irmão ou filho. O estado da aplicação é a representação de todos os dados que a aplicação precisa para funcionar corretamente, desde as informações dos utilizadores, informações dos produtos, dados de autenticação e outras informações importantes.

Na figura 1, do lado direito, podemos ver que os estados da aplicação são guardados num ficheiro nomeado por *store*, e os componentes têm acesso aos dados da *store* quando existe alguma alteração de estado nestes componentes.

Por outro lado, no lado esquerdo, vemos que quando um componente faz uma alteração de estado, esta alteração passa por todos os componentes ligados ao componente que fez alteração, sendo mais fácil de perder o controlo do estado.

Concluindo, a arquitetura consiste em permitir que os componentes leiam os dados da *store* e enviem ações para atualizar o seu estado, sem necessidade de passar por outros componentes. Esta técnica implementa muitas otimizações de desempenho interno para que os componentes sejam renderizados apenas quando necessário [2].

O Redux também oferece a possibilidade de utilizar o *Redux Devtools* [25], uma extensão do Google

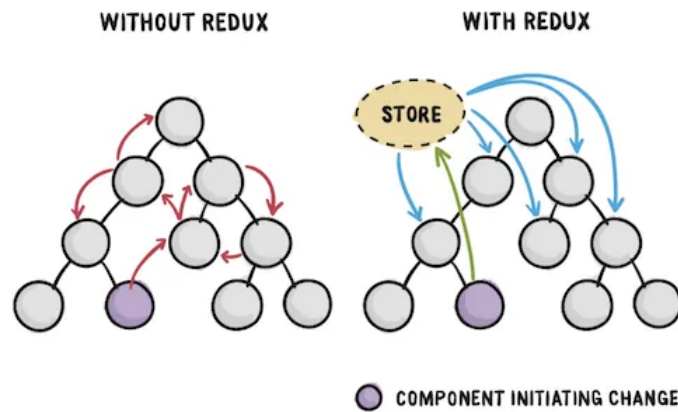


Figura 1: Partilha de estado de uma aplicação com e sem Redux [26]

Chrome que permite fazer o *debug* da aplicação.

## 2.2.6 NPM (Node Package Manager)

NPM é um gestor de pacotes e bibliotecas em *javascript* para projetos, integrado com o Node.js.

Permite que os programadores instalem e gerem as dependências dos seus projetos, bem como partilharem os seus próprios pacotes de *software* com outros programadores. O NPM é uma ferramenta essencial para a comunidade Node.js, e é amplamente utilizada em todo o mundo.

O NPM vem pré-instalado com o Node.js e é acessível a partir do terminal ou *prompt* de comando do sistema operativo. Os desenvolvedores podem usar o NPM para pesquisar pacotes disponíveis, instalar pacotes nos seus projetos e atualizar pacotes para versões mais recentes.

O ficheiro *package.json*, criado a partir do uso do *npm*, ajuda a gerir as dependências do projeto que foram criadas com a instalação de pacotes.

Ao longo do desenvolvimento do projeto, as bibliotecas referidas foram instaladas com o recurso a esta ferramenta.



## 3 Desenvolvimento da Loja Online

Com o propósito de criar uma aplicação voltada para a criação de partes de loja, foi determinado que o primeiro passo seria a criação de uma loja *online* básica que reunisse as características comuns presentes em todas as lojas.

Dessa forma, foi implementada a loja *online* "GamerShop" fictícia, destinada à venda de periféricos para jogos, a qual serviu de base para o desenvolvimento da aplicação.

Nesta secção, é abordado de forma mais detalhada o desenvolvimento da loja *online*, descrevendo as funcionalidades implementadas e os procedimentos de concepção e implementação adotados para a sua criação.

### 3.1 Funcionalidades

A loja *online* possui uma variedade de funcionalidades que são divididas em duas partes distintas: as funcionalidades relacionadas à comercialização dos produtos e as funcionalidades de *backoffice*.

As funcionalidades relacionadas à comercialização dos produtos podem ser acedidas por todos os utilizadores da loja. Já as funcionalidades do *backoffice* são exclusivas para os administradores da loja, que permite a gestão dos produtos, categorias, utilizadores e encomendas e acesso a estatísticas. Estas funcionalidades são importantes para garantir o bom funcionamento da loja e permitir o controlo da mesma.

As funcionalidades estão descritas na seguinte lista:

- Comercialização:
  - Registrar clientes;
  - *Login* e *logout* da sessão;
  - Consultar perfil de utilizador;
  - Editar perfil de utilizador;
  - Pesquisar produtos pelo nome;
  - Filtrar produtos por categoria e subcategoria;
  - Destacar produtos favoritos;
  - Avaliar por classificação os produtos;
  - Comentar produtos;

- Adicionar e remover produtos do carrinho de compras;
  - Alterar quantidade ou eliminar produtos no carrinho de compras;
  - Escolher método de pagamento;
  - Realizar encomenda;
  - Consultar histórico de encomendas;
  - Consultar estado de uma encomenda;
- *Backoffice*:
    - Consultar estatísticas de clientes, encomendas e produtos;
    - Editar e remover utilizadores;
    - Gerir permissões de utilizadores;
    - Consultar detalhes das encomendas;
    - Gerir encomendas;
    - Consultar estatísticas do *site*;
    - Criar, editar e remover produtos;
    - Adicionar e remover categorias e subcategorias;

As principais funcionalidades da loja serão apresentadas de acordo com as permissões de utilizador. Com o recurso ao diagrama de casos de uso da figura 2, são apresentadas as ações que o administrador e o utilizador têm em comum e individualmente. Para além disso, ao executar a ação de adicionar *reviews* é necessário que as ações de *login* e efetuar compra de um produto estejam incluídas. De igual forma, para efetuar a compra de um produto é necessário a ação de *login*, e o produto estar presente no carrinho.

O administrador tem a permissão mais alta na loja e pode executar todas as funcionalidades. Através do acesso exclusivo ao *backoffice*, o administrador pode gerir todos os aspetos da loja, desde a gestão de produtos e encomendas até à gestão de categorias e utilizadores. Além disso, o administrador tem acesso às estatísticas da loja, permitindo-lhe ter uma visão global do seu desempenho.

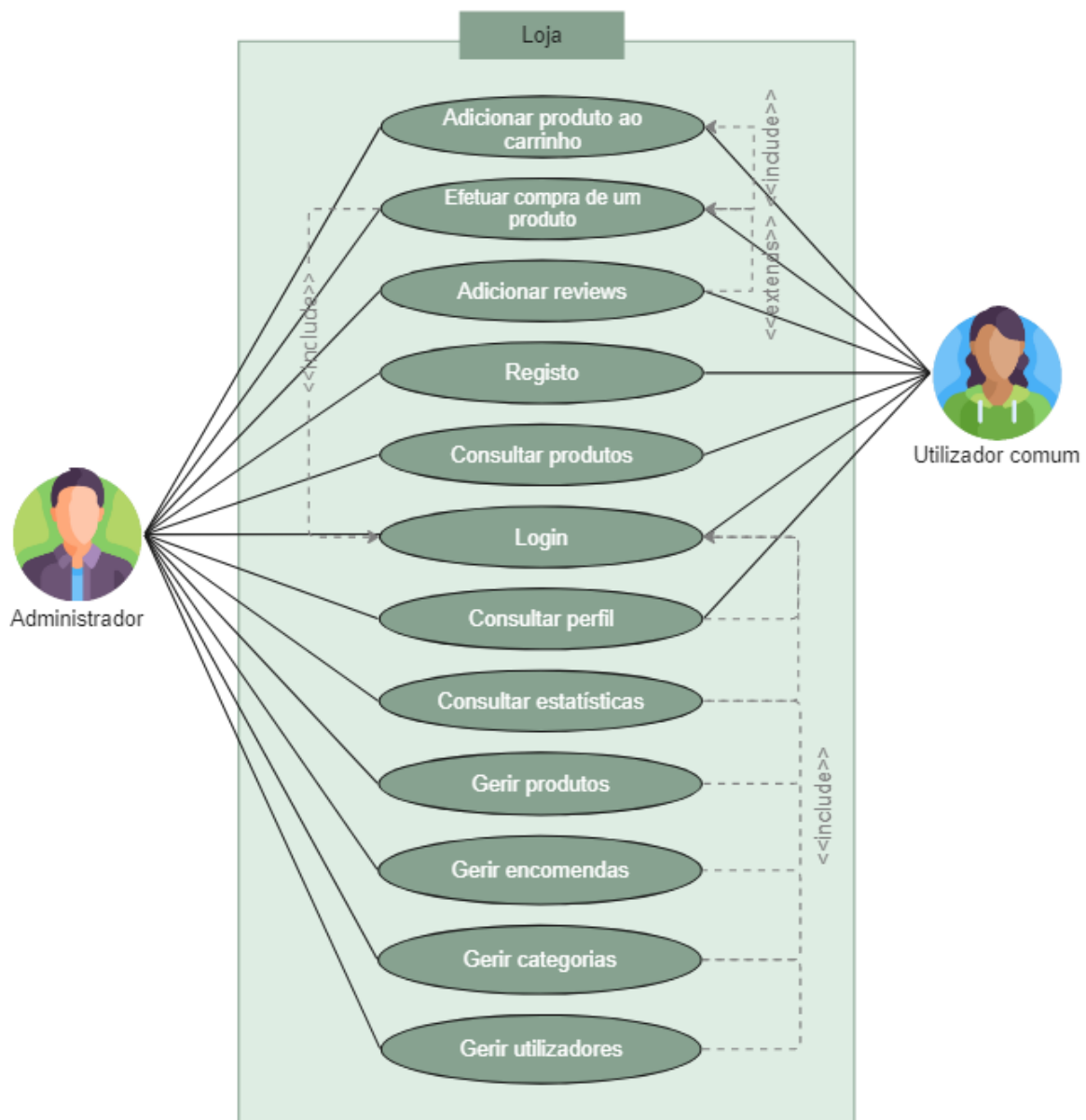


Figura 2: Diagrama de casos de uso da loja *online*

## 3.2 Conceção

Nesta secção, são discutidos elementos fundamentais do projeto, como a arquitetura, o modelo conceptual, o mapa de páginas e a API do *backend*.

A subsecção de arquitetura descreve a estrutura geral do projeto, incluindo a sua organização interna e os principais componentes. Neste contexto, serão apresentadas as tecnologias e ferramentas utilizadas para desenvolver o projeto.

O modelo conceptual, por sua vez, é a representação visual das principais ideias e conceitos do

projeto, que ajudará a garantir que todas as partes envolvidas tenham uma compreensão clara do que está a ser desenvolvido. Nesta subsecção, por meio de um diagrama de classes, serão apresentadas as classes que compõem o sistema, os seus atributos e métodos e como elas interagem entre si para cumprir as funcionalidades da aplicação.

O mapa de páginas descreve a estrutura da aplicação, incluindo as diferentes páginas e funcionalidades que serão disponibilizadas aos utilizadores. Aqui, será possível visualizar como as páginas estão interligadas e como são acessíveis ao utilizador.

Por fim, a subsecção API do *backend* descreve a interface que permitirá a comunicação entre o *frontend* e o *backend* da aplicação. Através da API, o *frontend* poderá enviar solicitações HTTP ao *backend* para aceder recursos e funcionalidades do sistema. Serão apresentados as funcionalidades que cada URL disponibiliza, e como os utilizadores do sistema poderão aceder para executar determinadas ações.

### 3.2.1 Arquitetura

A loja *online* e o respetivo *backoffice* foram desenvolvidos seguindo a arquitetura apresentada na figura 3. Esta arquitetura é baseada em quatro tecnologias: MongoDB [5], Express [15], React [36] e Node [11].

O uso do MongoDB como base de dados não relacional permitiu a criação de um modelo de dados flexível e adaptável às necessidades da aplicação. O Express foi utilizado para construir o servidor da aplicação e tratar das requisições e gestão de rotas. Por sua vez, o React foi escolhido para desenvolver a interface do utilizador da loja *online*. Por fim, o Node foi utilizado para realizar a comunicação entre o servidor e a base de dados. A escolha desta abordagem permite a utilização da linguagem *JavaScript* tanto no *backend* como no *frontend*, além de possuir uma ampla documentação disponível.

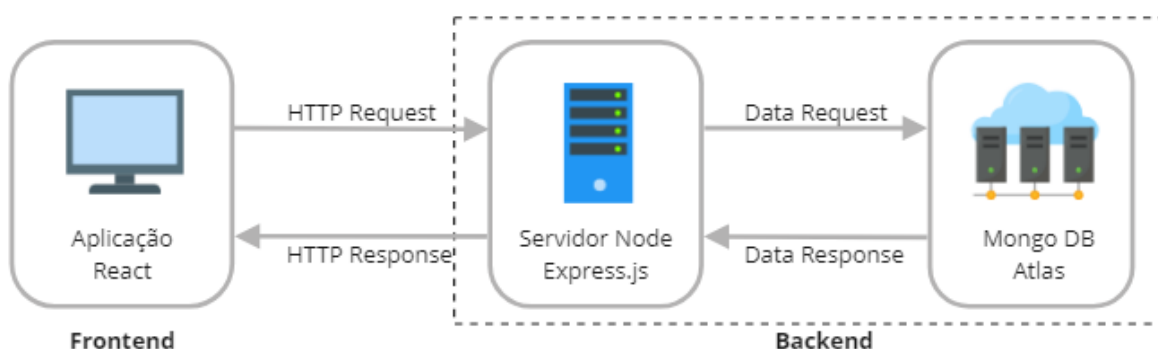


Figura 3: Arquitetura da loja *online*

A comunicação entre todos os elementos da aplicação é fundamentada no protocolo de comunicação HTTP. O cliente envia solicitações *HTTP Requests* ao servidor, que por sua vez responde com as respetivas

respostas HTTP *Response*. Da mesma forma, o servidor estabelece comunicação com a base de dados MongoDB Atlas, onde os dados são armazenados em nuvem. Para estabelecer esta comunicação, é utilizada a biblioteca Mongoose [16].

Para gerir as rotas da aplicação, é utilizada a *framework* Express, que é capaz de criar APIs REST (*Representational State Transfer*) e lidar com diferentes solicitações HTTP num único URL (*Uniform Resource Locator*).

Para a estilização do *frontend*, foram utilizados react-Bootstrap [21] e Bootswatch [10], que são bibliotecas que fornecem um conjunto de componentes predefinidos e estilos CSS personalizados [4], facilitando a criação de uma estrutura gráfica do *site* de forma fácil e com estilos pré-definidos que ajudam a tornar o resultado final mais atraente. Além disso, o CSS nativo também foi utilizado para personalizar ainda mais o estilo dos componentes.

### 3.2.2 Modelo conceptual

Para dar suporte à arquitetura foi desenvolvido um modelo conceptual da loja, representado pelo diagrama de classes da figura 4, que apresenta a estrutura e relações entre as principais entidades do sistema. As classes incluem Utilizador, Carrinho, Produto, Encomenda, Categoria, *Review* e Administrador.

A classe Utilizador é responsável por armazenar informações sobre os utilizadores do sistema e tem uma relação de um para um com a classe Carrinho. Isto significa que cada utilizador possui um carrinho e o carrinho só pertence a um utilizador. De igual forma, o Carrinho corresponde a uma única encomenda e cada encomenda está associada a apenas um carrinho.

Como o Utilizador pode fazer várias encomendas, a relação entre as classes Utilizador e Encomenda é feita por meio de uma associação de um para muitos, indicando que um utilizador pode estar associado a nenhuma ou muitas encomendas mas cada encomenda a apenas um utilizador.

A classe Carrinho e Produto representam uma associação de muitos para muitos em que um carrinho pode conter vários produtos e um produto pode estar em vários carrinhos.

Um produto pode ter zero ou várias *reviews*. Esta relação é representada de um para muitos em que do lado da classe Produto indica que um produto pode ter nenhuma ou várias avaliações, e a da classe *Review* indica que uma avaliação sempre está associada a um produto.

Sendo que um produto tem uma categoria associada, a relação entre as duas classes é de um para muitos, e indica que um produto está sempre associado a uma categoria, enquanto uma categoria pode estar associada a nenhum ou vários produtos.

A classe Administrador é um tipo de Utilizador com permissões especiais dentro do sistema. Desta

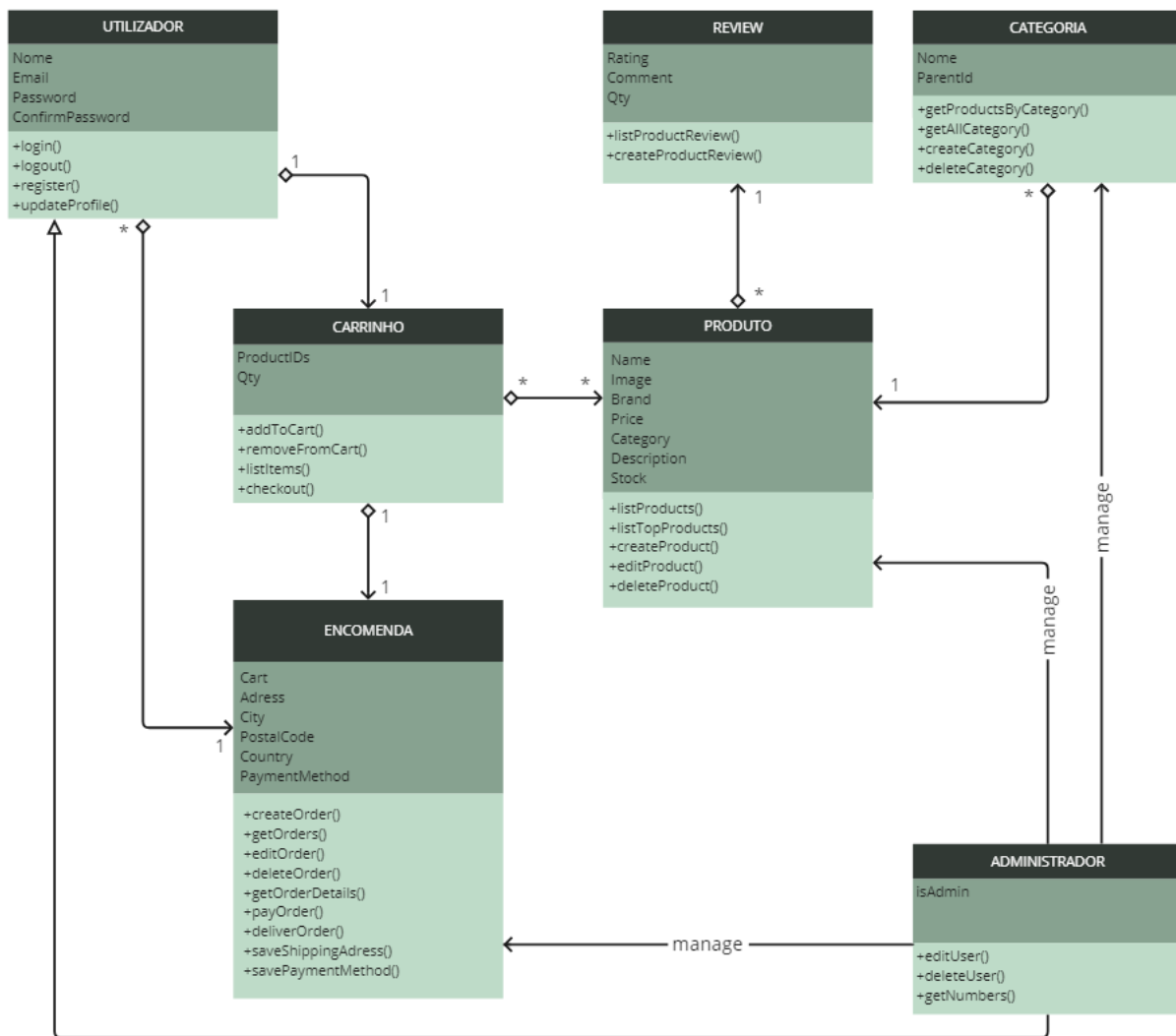


Figura 4: Diagrama de classes da loja *online*

forma, é utilizado o conceito de herança para definir esta relação em que a classe Administrador herda os atributos e métodos da classe Utilizador. A classe Administrador incide na gestão de outras classes, sendo indicado com o conceito de *manage* das classes Produto, Categoria e Encomenda. Além disso, possui métodos na gestão de utilizadores e na visualização das estatísticas.

### 3.2.3 Mapa de páginas

Nesta subsecção é apresentado o mapa de páginas, que demonstra a hierarquia das páginas da loja *online*. A página principal, designada por *Home*, é o topo da hierarquia e a partir dela é possível aceder a outras páginas, que estão descritas na figura 5.

Ao seleccionar uma categoria no menu de categorias da página principal, o utilizador é redirecionado para uma página que apresenta os produtos que pertencem a essa categoria específica.

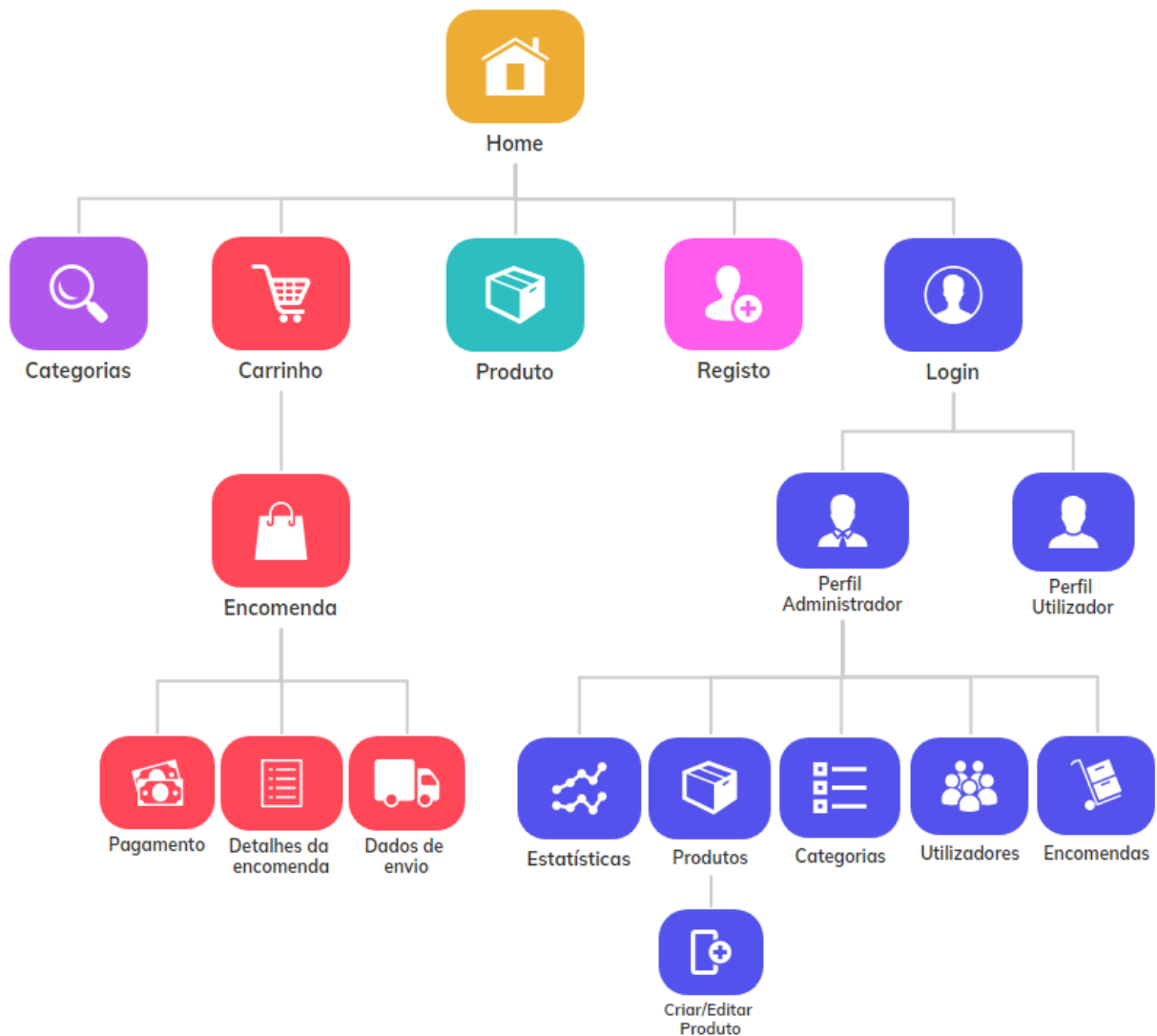


Figura 5: Mapa de páginas do *backoffice* e *frontoffice*

Além disso, na página principal, o utilizador pode aceder o seu carrinho, que corresponde à página do Carrinho. Nessa página, quando o botão de "efetuar encomenda" é acionado, o utilizador é direcionado para a página de Encomenda. Para realizar a encomenda, é necessário seguir uma série de etapas, que incluem visualizar os detalhes da encomenda, preencher os dados de envio e efetuar o pagamento. Cada uma dessas etapas é apresentada numa página separada, seguindo uma hierarquia lógica.

A página do Produto corresponde ao clique do utilizador de um produto apresentado no catálogo dos produtos da pagina principal. Esta página contém os detalhes do produto, como o preço, a descrição, o *stock*, e onde se adicionam as *reviews* e o produto em questão ao carrinho.

A página do Registo apenas apresenta o formulário onde o novo utilizador cria uma conta pela primeira vez e é adicionado à base de dados.

De seguida, a página do Login onde o utilizador insere os seus dados previamente registados na

base de dados, permite aceder à página do perfil. Existem dois tipos de perfil, administrador e utilizador comum.

Ao aceder o perfil de administrador, é possível navegar pelas páginas que se referem às funções do *backoffice*. Uma dessas páginas é a de Estatísticas, que permite ao administrador visualizar os dados estatísticos do site. Nesta página, não é possível fazer alterações, apenas obter informações relevantes para a tomada de decisões e análise do desempenho da loja.

A página de Produtos representa a listagem dos produtos disponíveis na loja, bem como a opção de eliminá-los. Para adicionar ou editar um produto, o administrador é direcionado para outra página específica para essa finalidade.

Na página de Categorias, é possível visualizar a lista de categorias existentes na loja, bem como criar novas categorias. Da mesma forma, é possível eliminar categorias já existentes.

A página de Utilizadores corresponde à listagem de todos os utilizadores da loja. O administrador tem a possibilidade de editar o perfil de um utilizador e conceder-lhe permissões de administrador, se necessário. Além disso, é possível excluir utilizadores que não são mais necessários.

Por último, a página de Encomendas apresenta a listagem de todas as encomendas realizadas por todos os utilizadores da loja, incluindo os seus detalhes. Nessa página, o administrador pode visualizar informações como a data da encomenda, o estado atual da entrega e os produtos adquiridos.

### 3.2.4 API do Backend

A API implementada suporta os quatro métodos HTTP: GET, POST, PUT e DELETE. O método GET é utilizado para obter dados que estão armazenados no servidor. O método POST é utilizado para enviar dados para o servidor para que sejam processados e armazenados. O método PUT é utilizado para atualizar recursos já existentes no servidor, enquanto o método DELETE é utilizado para eliminar recursos armazenados no servidor [19].

Primeiramente são apresentados os recursos relativos às categorias, de acordo com a tabela 3.2. Em seguida são apresentadas as tabelas dos recursos relativos às encomendas, na tabela 3.3, aos utilizadores, na tabela 3.4, e aos produtos, na tabela 3.5. As tabelas descrevem a API do *backend* separada por recurso base (categorias, encomendas, utilizadores e produtos) e mostram a identificação de recursos, juntamente com os métodos HTTP suportados. A coluna de *status* contém os possíveis valores para o código de três dígitos que comunica o sucesso ou insucesso de cada pedido. O significado dos códigos é o seguinte:



- *Status* 200: resposta genérica de sucesso
- *Status* 201: resposta de sucesso do método POST
- *Status* 400: solicitação incorreta que o servidor não processa
- *Status* 404: recurso não encontrado

O *middleware* desempenha um papel importante na segurança da aplicação, pois controla o acesso aos recursos. Para aceder a determinados recursos, é necessário autenticar o utilizador, e em outros recursos é necessário ser administrador. A última coluna indica se a autenticação, a autorização ou ambas são necessárias para cada forma de acesso ao recurso.

Tabela 3.2: Recursos relativos às categorias

<b>Funcionalidade</b>	<b>URL</b>	<b>Pedido HTTP</b>	<b>Status</b>	<b>Autenticação/Autorização</b>
Adicionar categoria	/category	POST	201	Autenticação e autorização
			400	
Consultar todas as categorias	/category	GET	200	Autenticação e autorização
			400	
Eliminar categoria	/category/:id	DELETE	200	Autenticação e autorização
			400	

Tabela 3.3: Recursos relativos às encomendas

<b>Funcionalidade</b>	<b>URL</b>	<b>Método HTTP</b>	<b>Status</b>	<b>Autenticação/Autorização</b>
Adicionar encomenda	/order	POST	200	Autenticação
			400	
Consultar todas as encomendas	/order	GET	200	Autenticação e autorização
			400	
Consultar encomendas de utilizador	/order/myorders	GET	200	Autenticação
			400	
Consultar encomenda por id	/order/:id	GET	200	Autenticação
			400	
			404	
Atualizar encomenda como paga	/order/:id/pay	PUT	200	Autenticação
			400	
			404	
Editar encomenda como enviada	/order/:id/deliver	PUT	200	Autenticação e autorização
			400	
			404	

Tabela 3.4: Recursos relativos aos utilizadores

<b>Funcionalidade</b>	<b>URL</b>	<b>Pedido HTTP</b>	<b>Status</b>	<b>Autenticação/Autorização</b>
Registar utilizador	/user	POST	200	Não requerido
			400	
Consultar todos os utilizadores		GET	200	Autenticação e autorização
			400	
Utilizador aceder ao perfil	/user/profile	GET	200	Autenticação
			400	
			404	
Utilizador editar perfil		PUT	200	Autenticação
			400	
			404	
Eliminar utilizador		DELETE	200	Autenticação e autorização
			400	
			404	
Consultar utilizador	/user/:id	GET	200	Autorização e autenticação
			400	
			404	
Editar utilizador		PUT	200	Autorização e autenticação
			400	
			404	
<i>Login</i> do utilizador	/user/login	POST	200	Não requerido
			400	
			401	

Tabela 3.5: Recursos relativos aos produtos

<b>Funcionalidade</b>	<b>URL</b>	<b>Método HTTP</b>	<b>Status</b>	<b>Autenticação/Autorização</b>
Adicionar produto		POST	200	Autenticação e autorização
			400	
Consultar todos os produtos	/product	GET	200	Não requerido
			400	
			404	
Eliminar produto		DELETE	200	Autenticação e autorização
			400	
Editar um produto	/product/:id	PUT	200	Autenticação e autorização
			400	
Consultar um produto		GET	200	Não requerido
		400		
Adicionar <i>review</i>	/product/:id/review	POST	201	Autenticação
			400	
			404	
Consultar produtos favoritos	/product/top	GET	200	Não requerido
			400	

## 3.3 Implementação

Nesta secção, serão descritos detalhadamente os aspetos mais técnicos do projeto, tais como a base de dados, a estrutura do código e a implementação das funcionalidades da aplicação que é dividida em duas partes: a primeira aborda as funcionalidades relacionadas à comercialização de produtos, enquanto a segunda se concentra no *backoffice*, que é a interface administrativa da aplicação..

Para garantir uma implementação eficiente e coerente com os objetivos do projeto, foi seguido um o modelo de dados que serviu como referência para a explicação de algumas funcionalidades.

No final da secção, é apresentada uma síntese dos principais resultados alcançados na implementação das funcionalidades da aplicação. São destacados os aspetos mais relevantes e as conquistas mais significativas alcançadas ao longo do processo de desenvolvimento.

### 3.3.1 Base de dados

A base de dados utilizada é não relacional, o que permite uma maior flexibilidade na estruturação dos dados em comparação a uma base de dados relacional. Como os documentos numa coleção podem ter diferentes estruturas e campos, é possível adaptar o modelo de dados mais facilmente para qualquer tipo de loja. Apesar de haverem algumas relações entre os diferentes dados, ilustrados no diagrama da figura 6, a utilização de uma base de dados não relacional também facilita a inserção e a recuperação dos dados, uma vez que não há necessidade de estabelecer relações complexas. Na base de dados estão implementadas quatro coleções que compõem a loja: Utilizadores, Produtos, Categorias e Encomendas. Cada vez que um novo documento é criado, o campo `_id` é gerado automaticamente pela base de dados, e é associado como chave primária do documento e do tipo *ObjectId*.

A coleção Encomendas é responsável por armazenar todas as informações referentes às encomendas realizadas pelos clientes da aplicação. Como mencionado anteriormente, o campo `_id` é utilizado como identificador único para cada documento presente nesta coleção. Além disso, o campo `utilizador` é utilizado para fazer referência ao *objectId* da coleção Utilizadores. Dessa forma, cada encomenda está associada ao id de um determinado utilizador, permitindo que seja possível saber a origem de cada pedido e gerir as informações de forma mais eficiente.

O campo `orderItems` é um *array* de objetos que armazena as informações de cada produto incluído na encomenda, juntamente com o seu respetivo *objectId* proveniente da coleção de produtos.

De seguida, o campo `ShippingAdress` contém as informações relativas à morada de envio do utilizador. O campo `itemsPrice` representa o preço total dos produtos selecionados pelo utilizador,

enquanto *paymentMethod* indica o método de pagamento utilizado para efetuar a encomenda. O campo *PaymentResult* armazena as informações acerca do pagamento efetuado pelo utilizador através do Paypal, que inclui o id e o estado da transação, bem como informações adicionais, como se o pagamento foi concluído com sucesso ou não.

O campo *taxprice* representa o valor das taxas, correspondente ao IVA (*Imposto sobre Valor Acrescentado*). O campo *Shippingprice* indica o valor cobrado pelo envio do produto, enquanto o campo *totalprice* é a soma dos campos anteriores, correspondendo ao preço total da encomenda.

O campo *isPaid* indica se a encomenda foi paga ou não, enquanto o campo *paidAt* representa a data em que o pagamento foi efetuado. Já o campo *isDelivered* informa se o produto foi enviado para o cliente e, por último, o campo *deliveredAt* indica a data em que o envio foi realizado.

A coleção Produtos é constituída por dois esquemas, que representam todos os produtos disponíveis na loja e as respetivas *reviews* feitas pelos utilizadores. O esquema de produtos inclui o campo *user*, do tipo *objectid*, que identifica o utilizador administrador que o criou. Os campos que caracterizam um produto são: o nome do produto, a imagem que representa o caminho(pasta e nome) em que fica guardada, a marca, a categoria, que é seleccionada a partir da coleção das categorias e a descrição do produto. O campo *rating* representa a classificação média do produto, com base nas *reviews* dos utilizadores. Já o campo *numReviews* indica o número de *reviews* recebidas no total do produto. *Price* é o preço do produto e *countInStock* é a quantidade de *stock* disponível do produto.

Por fim, o esquema das *reviews* é um *array* que contém as informações de todas as *reviews* feitas ao produto. Essas informações incluem o nome do utilizador que fez essa *review*, o *rating* que guarda a informação acerca do número de estrelas atribuído, o texto do comentário feito e o id do utilizador que efetuou a *review*.

A coleção Utilizadores guarda as informações de todos os clientes que se registaram na loja. Os campos desta coleção são: nome, email, *password* e *isAdmin*, que é do tipo *boolean* e indica se o utilizador é administrador ou não.

A coleção Categorias contém apenas os campos correspondentes ao nome, representado pelo campo *category*, e o campo *parentId*. O campo *parentId* é usado para armazenar o ID de outra categoria quando o utilizador pretende criar uma subcategoria. Isto é, se o campo *parentId* estiver vazio, a categoria adicionada é uma categoria principal. Caso o *parentId* guardar o id de uma categoria, significa que é criada uma subcategoria. Ao adicionar o id de uma subcategoria existente ao campo *parentId*, resulta numa sub-subcategoria.

Por exemplo, se existir uma categoria chamada "Periféricos" e o utilizador queira adicionar a

subcategoria "Ratos", o ID da categoria "Periféricos" seria armazenado no campo *parentId* da categoria "Ratos", indicando que "Ratos" é uma subcategoria de "Periféricos".

Se o utilizador desejar adicionar uma sub-subcategoria dentro de "Ratos", como por exemplo "Sem fios", o ID da categoria "Ratos" seria armazenado no campo *parentId* da categoria "Sem fios". Isto indica que "Sem fios" é uma subcategoria de "Ratos", que por sua vez é uma subcategoria de "Periféricos".

Esta abordagem permite criar uma estrutura hierárquica de categorias, onde é possível ter categorias principais, subcategorias e sub-subcategorias, conforme necessário.

Todas as coleções possuem o campo *timestamp*. Esta opção é fornecida pelo Mongoose e, quando declarada como verdadeira, cria dois campos associados a cada documento: o campo de data de criação do documento e o campo de data de atualização, caso o documento seja editado posteriormente.

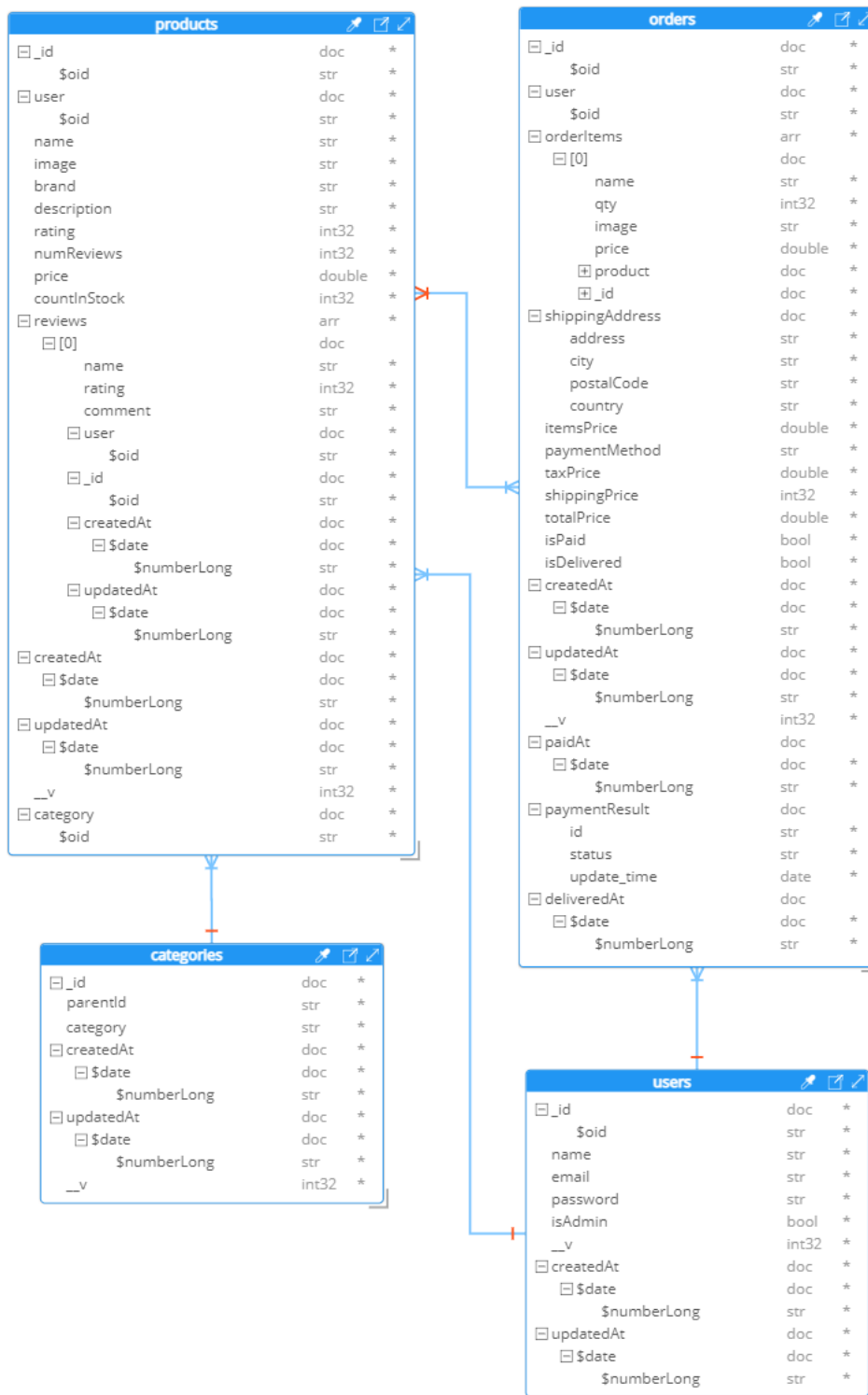


Figura 6: Diagrama E-R do modelo de dados



### 3.3.2 Estrutura do código

Nesta subsecção é apresentada a estrutura dos ficheiros e pastas que constituem o código da aplicação, incluindo o *frontend*, *backend* e base de dados. De seguida é apresentada a API do *backend* implementada.

Na figura 7 é apresentada a organização do *frontend* em vários elementos. O ficheiro `package.json` guarda todos as bibliotecas instaladas por meio do comando `npm`, enquanto a pasta `Public` contém apenas o ficheiro HTML da aplicação React. Por sua vez, a pasta `Src` é a principal detentora do código e se encontra subdividida em pastas e ficheiros.

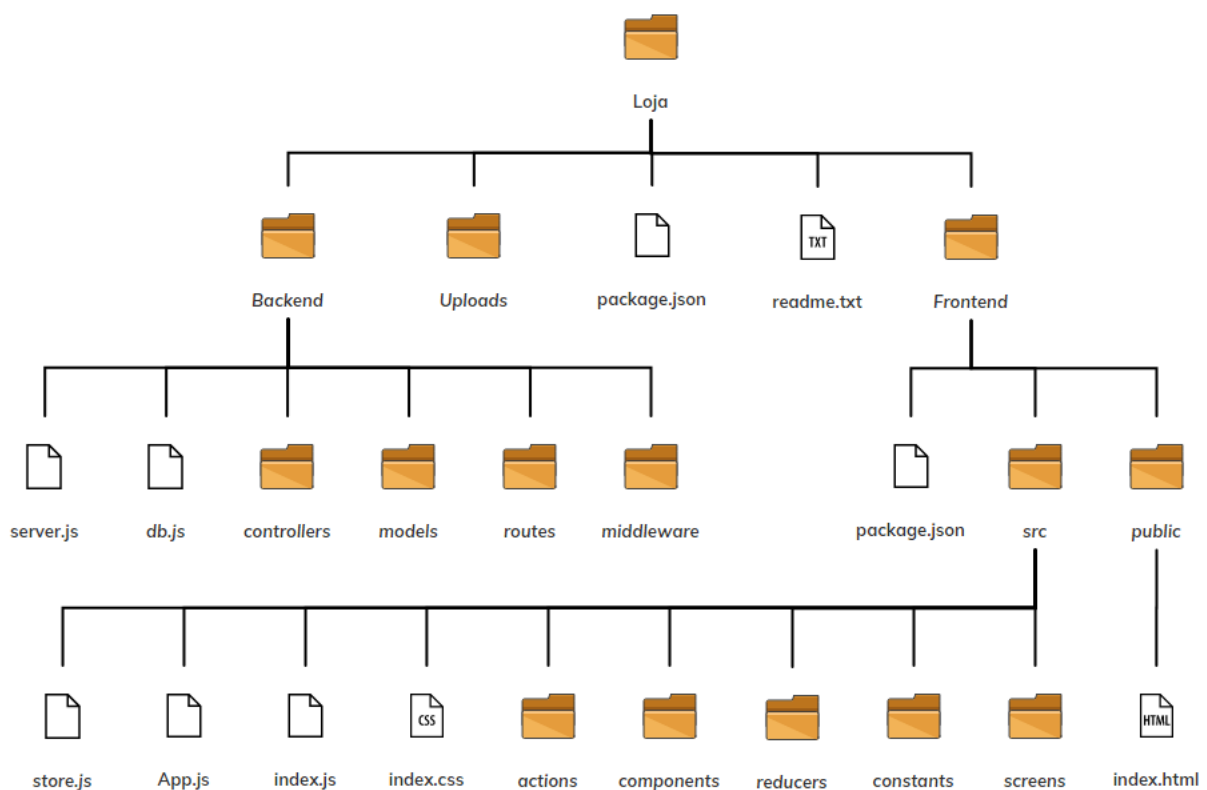


Figura 7: Estrutura dos ficheiros da loja *online*

Dentre as pastas, a *Components* abrange todos os componentes react, a *Actions* guarda os ficheiros compostos por funções que descrevem como o estado da aplicação deve ser atualizado e a pasta *Reducers* abriga os ficheiros que atualizam o estado da aplicação. Na pasta *Screens* encontram-se as páginas mencionadas anteriormente que são constituídas a partir do uso dos *components*.

O ficheiro `store.js` armazena os estados da aplicação. Através do ficheiro `App.js` é definida a estrutura e comportamento principal da aplicação, incluindo a definição das rotas. O ficheiro `index.js` é responsável por iniciar a aplicação e renderizar o conteúdo da página, enquanto o ficheiro `index.css` é um arquivo CSS

que define o estilo principal da aplicação e é importado no ficheiro `index.js`.

No lado do servidor a estrutura é bem simples, em que as *routes* encaminham as solicitações recebidas no URL para os *controllers*, que são funções responsáveis por obter os dados solicitados dos *models*, criam uma página HTML com esses dados e retornam ao utilizador para visualizar no navegador [8].

A pasta *middleware* guarda funções que lidam com a autorização e autenticação de utilizadores, bem como a manipulação de erros. Essas funções são utilizadas quando é necessário realizar uma tarefa antes de enviar uma resposta a uma solicitação, como verificar se o utilizador é administrador antes de eliminar um produto, por exemplo.

Por fim, a pasta *Models* contém os ficheiros que definem a estrutura dos modelos de dados de cada coleção, sendo eles: encomendas, produtos, utilizadores e categorias. Cada elemento na coleção representa um documento na base de dados.

A API desenvolvida segue a arquitetura REST, em que os recursos são identificados por URLs, que especificam o caminho para o recurso solicitado. Esta abordagem facilita a compreensão e a comunicação entre o cliente e o servidor, já que o URL claramente indica ao servidor qual recurso o cliente solicitou.

### 3.3.3 Comercialização de produtos

Na página inicial da aplicação, ilustrada na figura 8, o primeiro componente é a barra de navegação, a qual contém um filtro de pesquisa de produtos por nome. O filtro permite que o utilizador encontre rapidamente um produto ao digitar a palavra associada ao nome do produto. Todos os produtos que possuam a palavra inserida pelo utilizador são apresentados na página.

Além disso, a barra informa que as encomendas com um valor igual ou superior a 50 euros têm direito a transporte gratuito. Este valor deve ser superior ao valor médio que um cliente gasta normalmente numa encomenda, como a loja é fictícia e não há acesso a essa estatística, o valor foi definido de acordo com a média dos preços dos produtos disponíveis.

Outras opções disponíveis na barra de navegação incluem a consulta do perfil do utilizador, bem como a opção de sair da sessão. Por último, o carrinho de compras também é acessível permitindo que o utilizador visualize e manipule os produtos selecionados para compra.

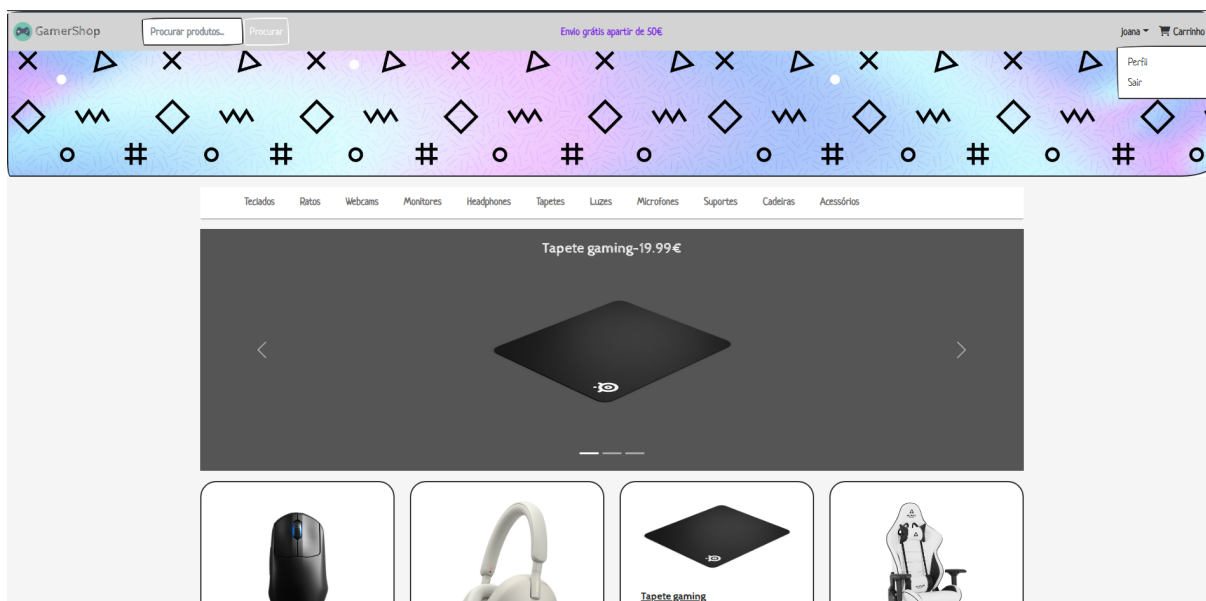


Figura 8: Página inicial de um utilizador

A opção de perfil redireciona o utilizador para a página de Perfil de utilizador, conforme apresentado na figura 9. Nesta página, o utilizador tem a possibilidade de visualizar e editar os dados do seu perfil, tais como nome, email e *password*. Além disso, é apresentada a lista de encomendas que o utilizador realizou, podendo consultar os seus detalhes acionando o botão respetivo.



Figura 9: Página do perfil de utilizador

Ainda na página principal, encontra-se o menu de categorias e subcategorias existentes. Ao passar o *cursor* pela categoria principal, o menu expande e apresenta as subcategorias por baixo da categoria principal. Desta forma o utilizador visualiza imediatamente quais os tipos de produtos que existem e se tem o que se procura. Ao clicar numa categoria do menu, é direcionado para a página responsável por apenas listar os produtos correspondentes a essa categoria, como mostra a figura 10.

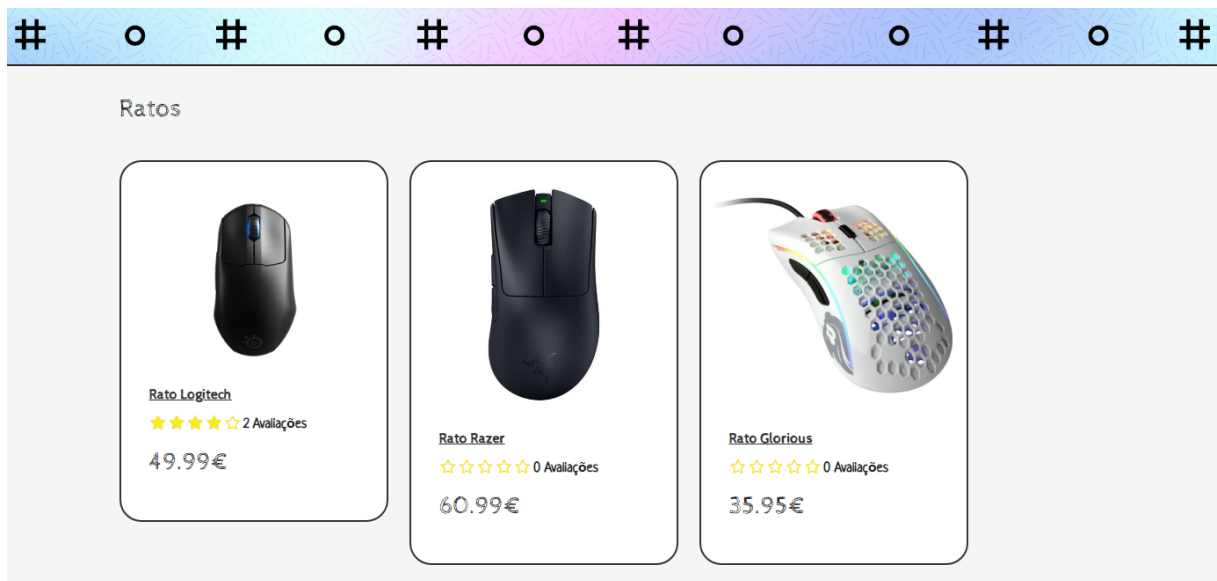


Figura 10: Página de produtos da categoria Ratos

De seguida, na página apresentada na figura 11, encontra-se um carrossel dos produtos favoritos dos utilizadores. A seleção dos produtos apresentados é feita com base no campo *rating* das *reviews*, em que os que possuem o *rating* mais alto são os escolhidos como favoritos.

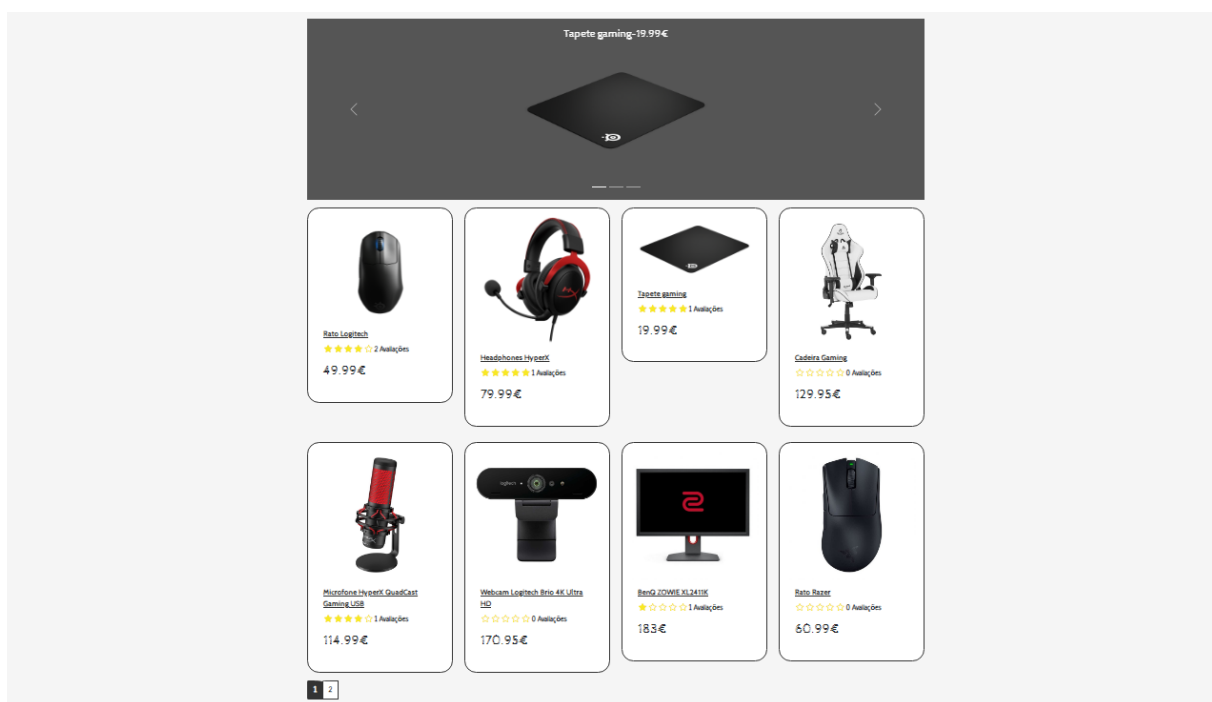


Figura 11: Secção do catálogo de produtos e favoritos

O próximo componente é o catálogo dos produtos, onde é possível visualizar a imagem do produto, juntamente com o seu nome, preço e classificação numa escala de 0 a 5 estrelas. Esta classificação é uma

forma dos utilizadores expressarem a sua opinião sobre a qualidade do produto, em que 0 representa uma avaliação muito fraca e 5 uma avaliação excelente. Ainda nesta página está implementada uma estratégia de paginação para visualizar um certo número de produtos por página definido no *backend*, para evitar que a página fique demasiado longa na existência de muitos produtos.

Ao clicar num produto, o utilizador é direcionado para a página do produto apresentado na figura 12, onde aparecem mais detalhes do mesmo como a descrição e o estado de *stock*. É também permitido seleccionar a quantidade do produto de acordo com o *stock* disponível e adicionar ao carrinho.

No lado direito da página, é exibida a média das classificações juntamente com o número total de avaliações, preço e descrição do produto.

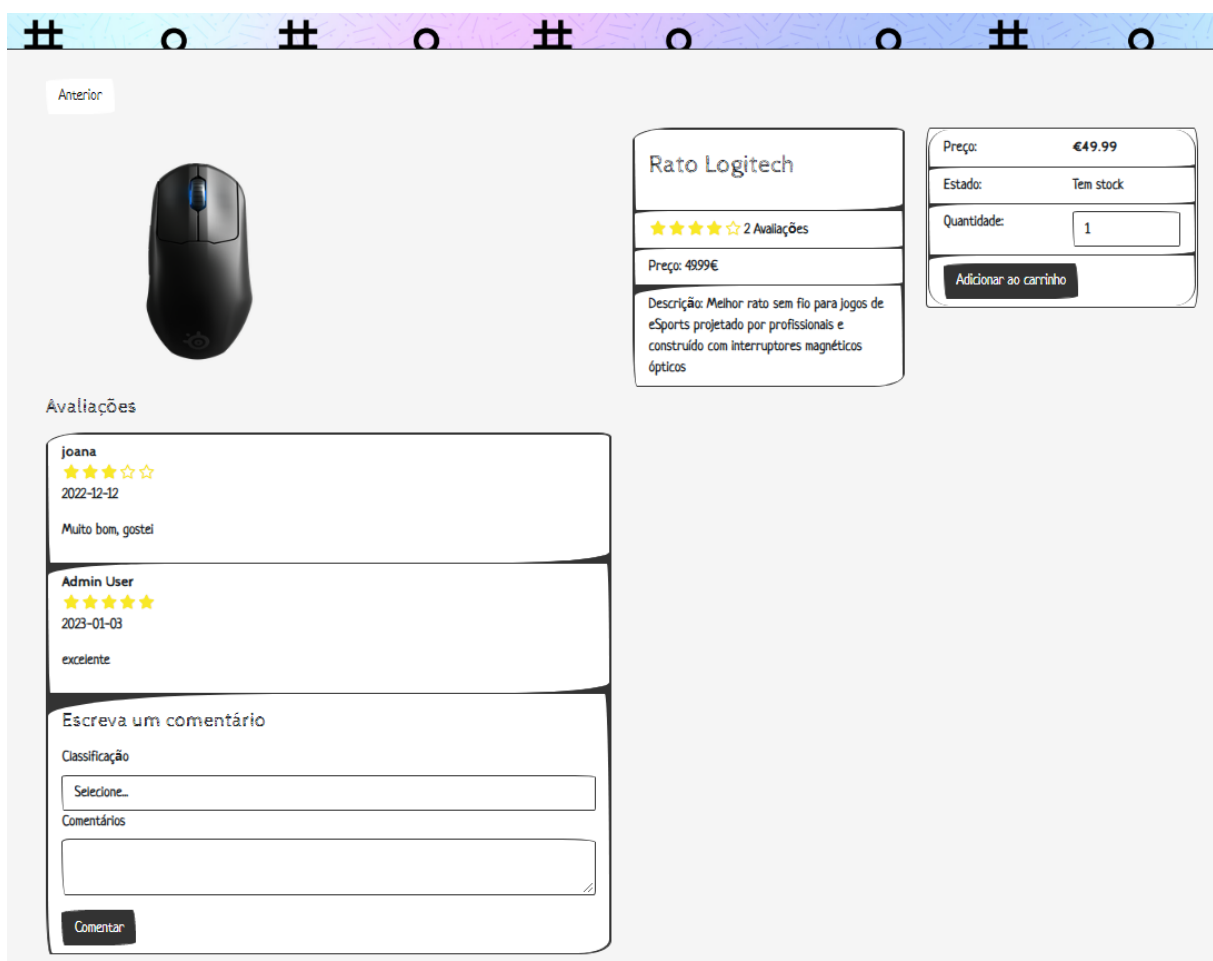


Figura 12: Página de um produto

Na página de detalhes do produto, é possível para o utilizador deixar uma avaliação do produto, fornecendo uma classificação numa escala de 0 a 5 estrelas e adicionando um comentário, se assim desejar. Para iniciar esta ação, o utilizador deve preencher os campos disponíveis para a classificação em estrelas e o comentário em texto, e por fim clicar no botão "Comentar", que está localizado na parte

inferior da página.

O processo para a realização desta ação é ilustrado em um diagrama de sequência, que pode ser visto na figura 13. O diagrama mostra o procedimento após clicar no botão "Comentar" até a *review* ser adicionada.

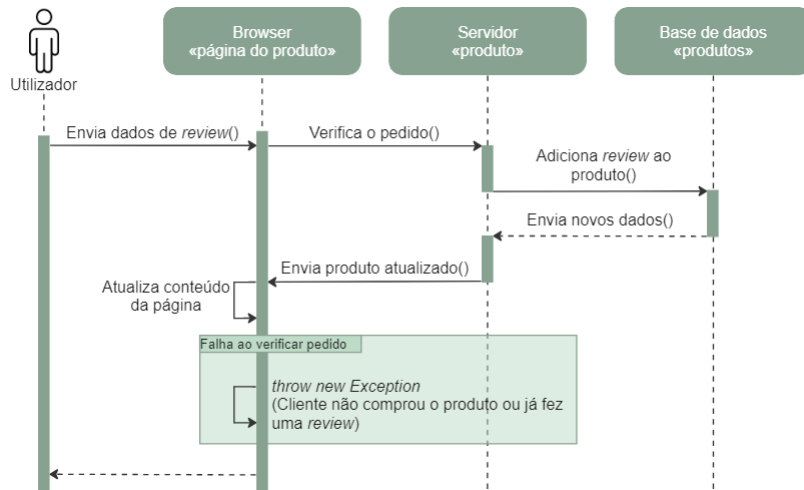


Figura 13: Diagrama de sequência do processo de adicionar uma *review*

Após o utilizador submeter os campos preenchidos na página de detalhes do produto, o servidor recebe o pedido HTTP POST e verifica se este respeita dois requisitos importantes. Primeiro, o servidor verifica se o utilizador que está a submeter a *review* comprou o produto em questão, verificando se o id do utilizador que submeteu o pedido corresponde ao id do utilizador que comprou o produto. Em segundo lugar, o servidor verifica se o utilizador ainda não submeteu um comentário anteriormente a este produto, verificando se no campo das *reviews* do documento do produto tem o id do utilizador correspondente.

Se ambos os requisitos forem cumpridos, a *review* é adicionada à base de dados e o servidor retorna uma resposta de sucesso ao cliente. Em seguida, a página de detalhes do produto é atualizada para exibir a nova avaliação junto com as avaliações anteriores. Caso contrário, o utilizador é informado de que não cumpre os requisitos e a *review* não é adicionada.

Este processo é fundamental para garantir a integridade das informações na base de dados e prevenir a submissão de *reviews* falsas ou duplicadas.

Na página do carrinho, que é exibida após a adição de um produto, conforme ilustrado na figura 14, o produto selecionado é automaticamente adicionado com uma quantidade padrão de 1. O utilizador tem a possibilidade de alterar essa quantidade de acordo com o *stock* disponível. Além disso, é possível eliminar totalmente o produto do carrinho clicando no botão com o ícone de caixote do lixo. A página também exibe o número total de produtos no carrinho, o preço total dos produtos e uma opção para prosseguir

para o *checkout* e finalizar a compra.

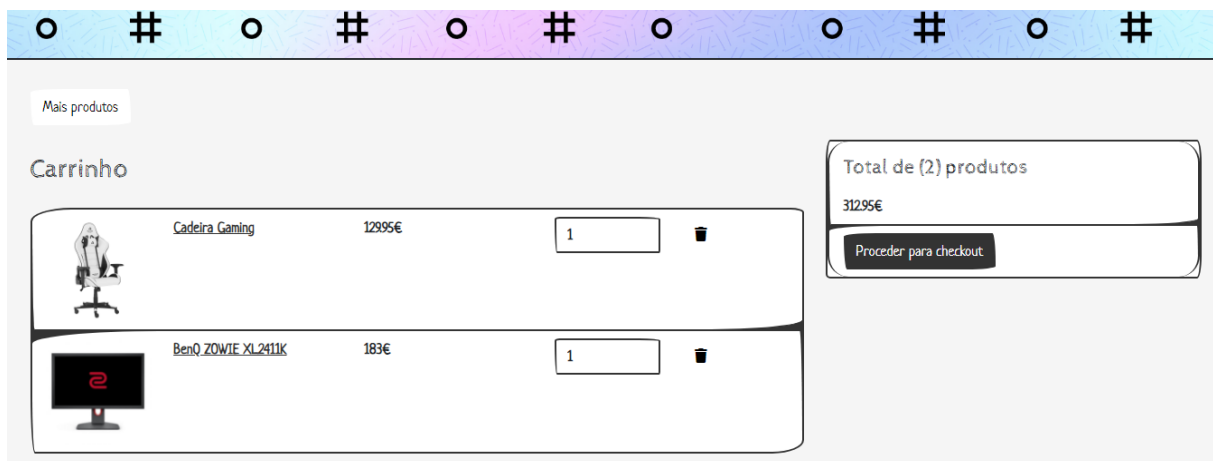


Figura 14: Página do carrinho de compras

A realização de uma encomenda ocorre em etapas, sendo que a primeira exige que o utilizador esteja com a sua sessão iniciada no *site* e seja um utilizador registado.

Caso o utilizador não esteja com sessão iniciada, é redirecionado para a página de *login*, e assim que fizer o *login* com sucesso, é redirecionado de volta para a página da encomenda, já no passo seguinte. Na etapa dos dados de envio, apresentada na figura 15, o utilizador preenche as informações como morada, cidade, código postal e país. Em seguida, é necessário seleccionar o método de pagamento que, neste caso, é exclusivamente o Paypal.

A última etapa do processo consiste em apresentar ao utilizador os dados da encomenda para que possa confirmá-la, incluindo o valor total dos produtos, o valor das taxas correspondentes ao IVA de 23% sobre o valor dos produtos e o custo de envio. O custo de envio é de 0 caso o valor total da encomenda atinja os 50 euros, caso contrário é cobrado um valor adicional fixo.

Entrar na conta   Dados de envio   Pagamento   Encomenda

## Encomenda

Morada

Cidade

Código Postal

País

**Continuar**

Figura 15: Página dos dados de envio de uma encomenda

Além disso, são apresentados os botões fornecidos pela biblioteca *react-paypal-button-v2* [22] para proceder ao pagamento, conforme mostrado na figura 16. Este é o único método de pagamento disponível e, para fazer a sua integração com a aplicação, utilizou-se o JS SDK (*Software Development Kit*) fornecido pelo Paypal [12]. É possível testar o funcionamento do sistema sem custos, utilizando contas *sandbox* criadas na secção de *developers* do Paypal.

ID da encomenda: 64123d435d9e67e8aac4b2dd

**Encomenda**

Nome: joana  
 Email: joana\_querido7@hotmail.com  
 Morada: Rua Central n°923, Fiães, Santa Maria da Feira,4505-254, Portugal

Não enviada ainda

**Método de pagamento**

Método: Paypal

Não pago

**Artigos encomendados**

	Cadeira_Gaming	1 x 12995€ = 12995€
	BenQ ZOWIE XL2411K	1 x 183€ = 183€

**Detalhes da encomenda**

Produtos	31295€
Envio	0€
Taxas	71.98€
<b>Total</b>	<b>38493€</b>

**PayPal**

Cartão de débito ou crédito

Powered by PayPal

Figura 16: Página de uma encomenda



Durante o processo de pagamento numa loja online que integra o Paypal como forma de pagamento, ao clicar no botão do Paypal, um *modal* de pagamento é exibido ao utilizador na própria página e o processo que decorre para ser efetuado está descrito no diagrama de seqüência da figura 17. Este *modal* contém um formulário de *login* do Paypal, onde o utilizador insere os seus dados para aceder à sua conta. Uma vez autenticado, o utilizador pode concluir a compra por meio do *modal*. Para garantir o correto funcionamento dessa integração e simular transações reais, foram criadas contas de *sandbox* para os utilizadores e para a loja. Por fim, se a transação for bem sucedida, o estado do pedido será atualizado para "pago" e refletido no *frontend*, confirmando a conclusão da compra.

Após o utilizador efetuar o pagamento de uma encomenda com sucesso, o servidor envia para o email do cliente a confirmação da encomenda. Para enviar os emails foi usada a biblioteca *mailgun-js* do Mailgun [14]. Mailgun é um serviço que disponibiliza um plano gratuito, com algumas limitações, incluindo a restrição de enviar emails apenas para endereços de email autorizados previamente e a um limite de cinco emails diferentes. É importante destacar que, caso o serviço fosse projetado para uma plataforma de maior escala, era recomendável optar por um plano pago para evitar limitações e garantir uma melhor qualidade de serviço.

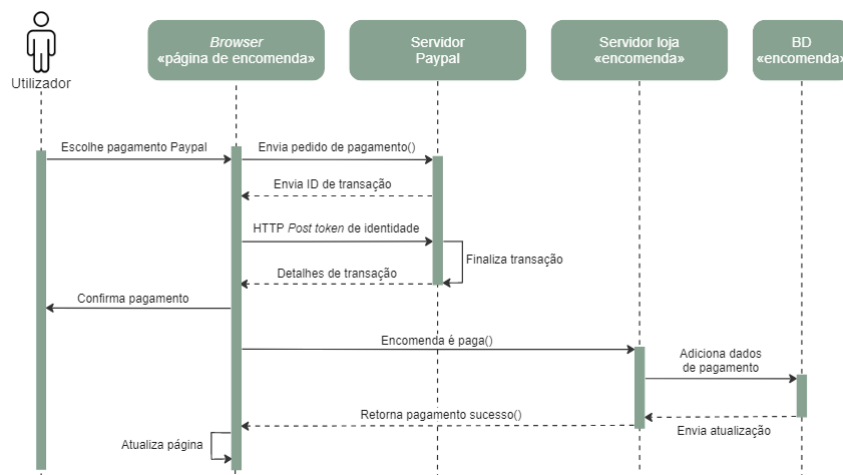


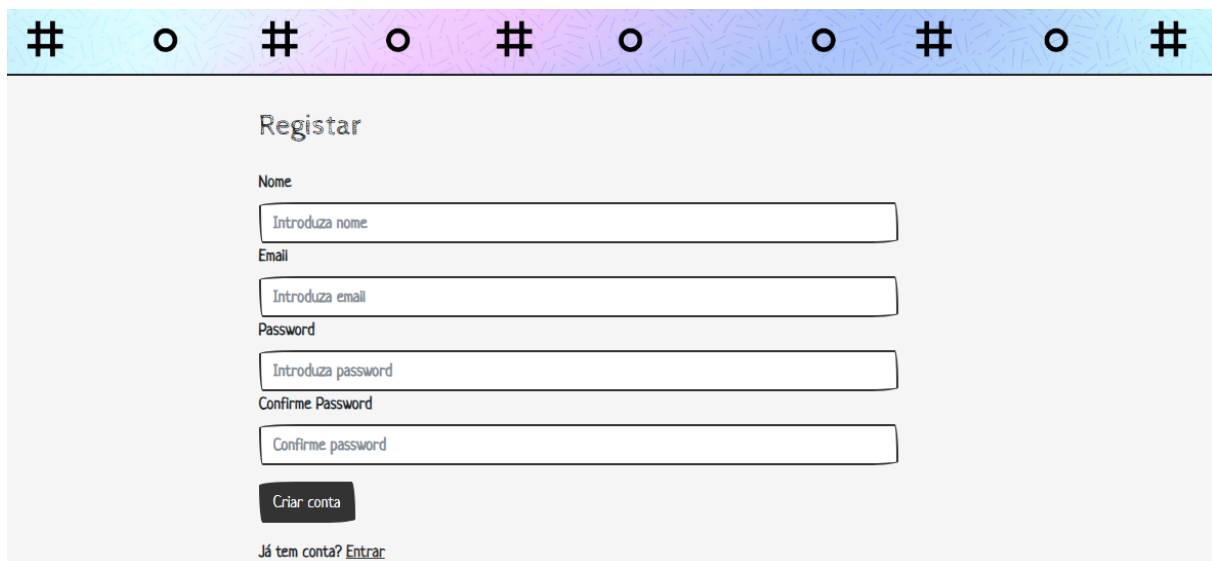
Figura 17: Diagrama de seqüência do pagamento por Paypal

### 3.3.4 Backoffice

Para aceder às funcionalidades do *backoffice*, é necessário efetuar o *login* com uma conta de administrador registada previamente na base de dados. Para isso, o utilizador deve inserir o email e *password* correspondente. Caso o email ou *password* estejam errados ou não existam na base de dados, o utilizador receberá um aviso do mesmo.

A página do Registo apresenta um formulário onde o novo utilizador cria uma conta pela primeira vez e é adicionado à base de dados. O formulário é composto pelos campos nome, email, *password* e confirmação de *password*. O email tem que possuir o formato correto, contendo o caractere "@" na sua composição para que possa ser aprovado. Além disso, o conteúdo do campo *password* tem que coincidir com o campo confirmar *password* para ser aceite.

Nesta página é apresentada ao utilizador a opção de se redirecionar para a página de *login*, caso o utilizador já possua uma conta registada na aplicação.



The image shows a registration form titled "Registar". It features four input fields: "Nome" (with placeholder "Introduza nome"), "Email" (with placeholder "Introduza email"), "Password" (with placeholder "Introduza password"), and "Confirme Password" (with placeholder "Confirme password"). Below the fields is a dark button labeled "Criar conta". At the bottom, there is a link "Já tem conta? Entrar". The form is set against a light gray background with a decorative header bar at the top containing a repeating pattern of hash symbols and circles.

Figura 18: Página de registar novo utilizador

Algumas rotas precisam de autenticação e autorização para se poder navegar nelas. A autorização é feita através do JWT (*Json Web Token*) de identificação que é criado cada vez que o utilizador é autenticado ao fazer *login*, o token é criado com o respetivo id de utilizador. Quando um utilizador quer fazer alguma ação que envolva uma rota protegida, como aceder ao perfil, o *token* é enviado nesse pedido.

No lado do servidor, para proteger rotas e verificar se esse pedido é válido foi criado um *middleware* que procura o *token* no cabeçalho de autorização. Se o encontra, extrai o *token* e, se tudo estiver correto, o utilizador extraído do mesmo é autorizado.

Também pode ser necessário verificar se o utilizador é administrador, porque existem rotas que são apenas acessíveis se isto for verificado. Para isso, o *middleware* implementado consiste em verificar se o campo *isAdmin* correspondente ao id do utilizador, é afirmativo. Se for afirmativo, tem acesso às rotas de administrador; se não é enviado um erro a informar que não é administrador.

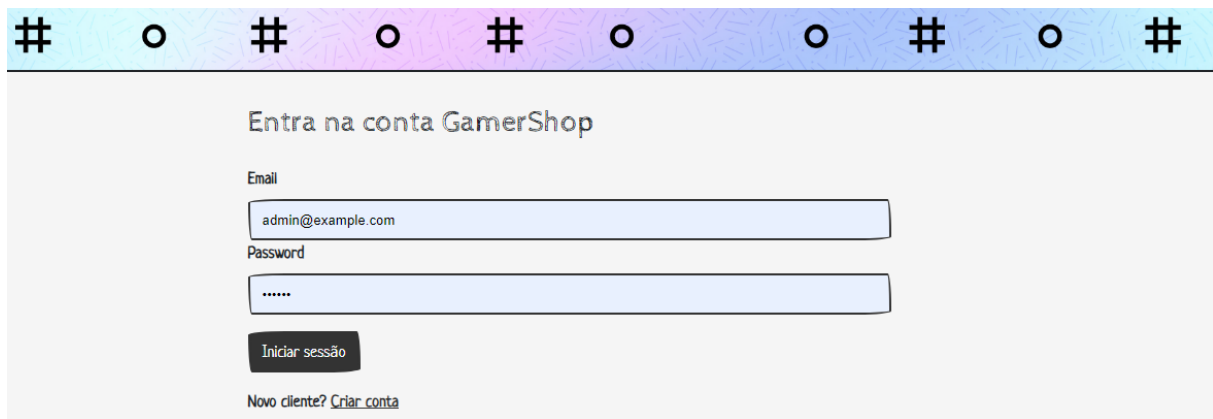


Figura 19: Página de *login*

Quando o *login* como administrador é feito com sucesso, é-se redirecionado para a página inicial da loja, onde se tem acesso ao conteúdo correspondente ao *backoffice*, aparecendo um menu na barra de navegação junto ao carrinho e ao perfil de utilizador como mostra a figura 20 .



Figura 20: Opções do *backoffice*

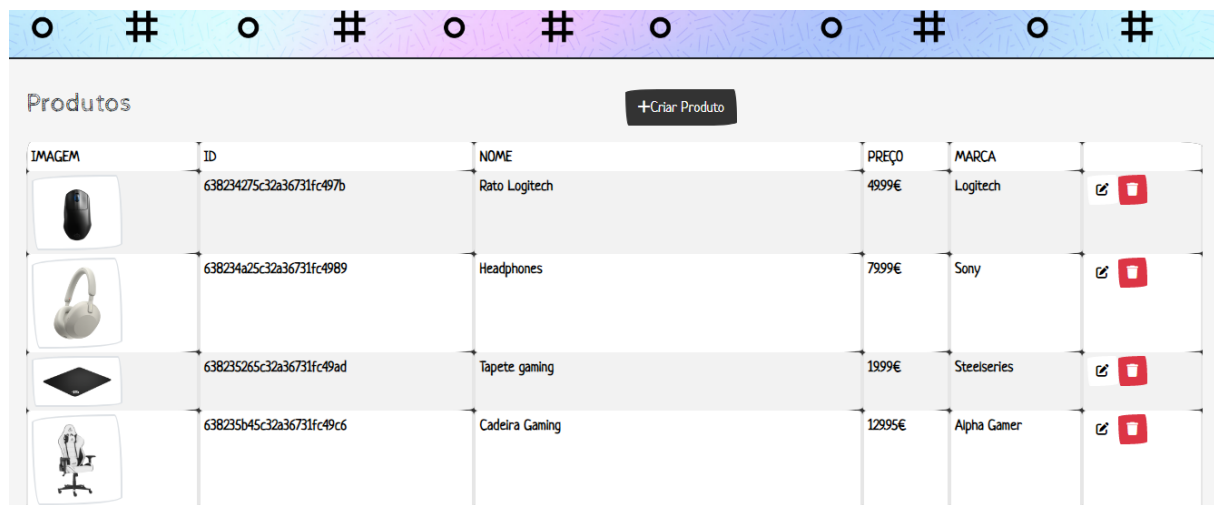
A primeira opção do menu é acerca das estatísticas da loja. É possível observar o número de encomendas, utilizadores, produtos e categorias existentes na loja. O gráfico acerca dos utilizadores mostra os dados relativos à percentagem de utilizadores criados nos diversos meses do ano. O gráfico de valor das encomendas por mês representa o valor das encomendas registadas, em que a barra vermelha corresponde às encomendas não pagas e a barra verde corresponde efetivamente às encomendas que foram pagas por mês. Esta informação é importante para avaliar o desempenho financeiro da loja.

O gráfico abaixo corresponde ao *stock* dos produtos da loja.

Para construir estes gráficos, foi utilizada a ferramenta Charts do MongoDB Atlas [17], que permite a criação de gráficos personalizados de forma fácil e intuitiva. A utilização destes gráficos e estatísticas é



igualmente eliminar ou editar um produto, pressionando os botões apresentados por um caixote do lixo e um caderno com lápis, respetivamente. É nesta secção que o administrador também pode criar um novo produto.



The screenshot shows a web interface for a product list. At the top, there is a decorative header with a repeating pattern of circles and hash symbols. Below the header, the word "Produtos" is displayed on the left, and a "+Criar Produto" button is on the right. The main content is a table with the following columns: IMAGEM, ID, NOME, PREÇO, MARCA, and two action icons (edit and delete). The table contains four rows of product data.













IMAGEM	ID	NOME	PREÇO	MARCA		
	638234275c32a36731fc497b	Rato Logitech	4999€	Logitech		
	638234a25c32a36731fc4989	Headphones	7999€	Sony		
	638235265c32a36731fc49ad	Tapete gaming	1999€	Steelseries		
	638235b45c32a36731fc49c6	Cadeira Gaming	12995€	Alpha Gamer		

Figura 23: Página da lista de produtos

O formulário para criar um novo produto ou editar um existente, apresentado na figura 24, é constituído pelos seguintes campos: o campo de preço, não sendo possível acrescentar números negativos, o campo da imagem que é uma opção para escolher um ficheiro, nos formatos JPEG (*Joint Photographic Experts Group*) ou PNG (*Portable Network Graphics*), o campo da marca, o *stock* inicial do produto, o campo da categoria onde é permitido escolher uma categoria que já esteja guardada na base de dados para associar ao produto e, por último, a descrição do produto.

Para a funcionalidade de escolher uma imagem é usada a biblioteca Multer [18], que permite guardar os ficheiros que o utilizador fez *upload* localmente num diretório. Se o ficheiro selecionado não corresponder aos formatos permitidos, o ficheiro não é guardado.

Ao pressionar o botão Criar Produto, um novo produto é inserido na base de dados. Caso o utilizador deseje editar um produto, é direcionado para o mesmo formulário de adição de produto, mas com os campos já preenchidos com os dados do produto a ser editado. Ao pressionar o botão de editar, os campos modificados são atualizados na base de dados. Ao pressionar o botão de eliminar, o produto é eliminado da base de dados.

Figura 24: Página de editar/criar produtos

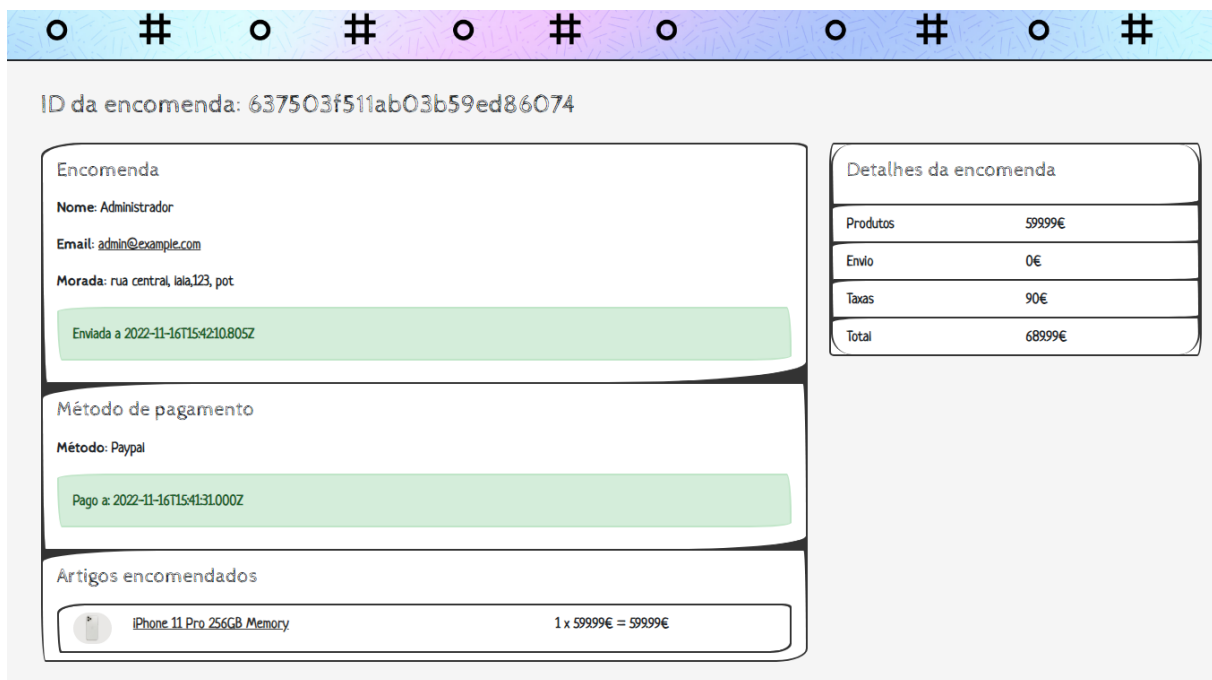
Na opção de encomendas é apresentada a lista de encomendas com as suas informações relacionadas, como mostra a figura 25. Essas informações são: o id da encomenda, o nome de utilizador que efetuou a encomenda, respetiva data em que foi feita a encomenda, quantidade, e se a encomenda foi paga e entregue.

ID	UTILIZADOR	DATA	TOTAL	PAGO	ENTREGUE	DETALHES
632dd8acF35ab5ec6a78ba58	john querido	2022-09-23	175948€	×	×	Detalhes
633d8611567794e93a5d92e0	john querido	2022-10-05	68999€	×	×	Detalhes
637503f511ab03b59ed86074	Administrador	2022-11-16	68999€	2022-11-16	2022-11-16	Detalhes
6383efedd68edced543de757	Joana	2022-11-27	14948€	×	×	Detalhes
6383f0482430c923b5afdee0	Joana	2022-11-27	19659€	×	×	Detalhes
639748023a6c8a47a2319842	Joana	2022-12-12	15749€	2022-12-12	2022-12-16	Detalhes
63974a2e3a6c8a47a23198f3	Joana	2022-12-12	18398€	2022-12-12	2022-12-16	Detalhes

Figura 25: Página da lista de encomendas

Por último, na coluna dos detalhes é possível consultar mais informações acerca do email do

utilizador, morada, método de pagamento e os artigos que seleccionou. Nesta secção, representada na figura 26, é também permitido ao administrador marcar uma encomenda como entregue, se esta já foi paga previamente.



ID da encomenda: 637503f511ab03b59ed86074

#### Encomenda

Nome: Administrador  
Email: admin@example.com  
Morada: rua central, lala,123, pot


Enviada a 2022-11-16T15:42:10.805Z

#### Método de pagamento

Método: Paypal

Pago a: 2022-11-16T15:41:31.000Z

#### Artigos encomendados

 iPhone 11 Pro 256GB Memory	1 x 59999€ = 59999€
---	---------------------

#### Detalhes da encomenda

Produtos	59999€
Envio	0€
Taxas	90€
Total	68999€

Figura 26: Página dos detalhes de uma encomenda

A ultima opção do menu de administrador permite a gestão de categorias. Como é apresentado na figura 27, o administrador tem a possibilidade de adicionar uma nova categoria e associá-la a uma já existente, criando uma subcategoria. Se for criada uma nova categoria associada a uma subcategoria, forma-se uma sub-subcategoria e assim sucessivamente.

No caso de apenas preencher o campo do nome da categoria sem associar a outra categoria, é criada a categoria principal. Nesta secção é também possível visualizar a lista de categorias organizadas em árvore, para compreender a relação entre categorias e subcategorias. Para eliminar uma categoria ou subcategoria, o utilizador deve seleccionar a pretendida da lista e pressionar o botão de apagar. Este botão ativa uma janela que solicita a confirmação do utilizador para eliminar as categorias seleccionadas. Ao eliminar uma categoria, as subcategorias associadas são automaticamente eliminadas.

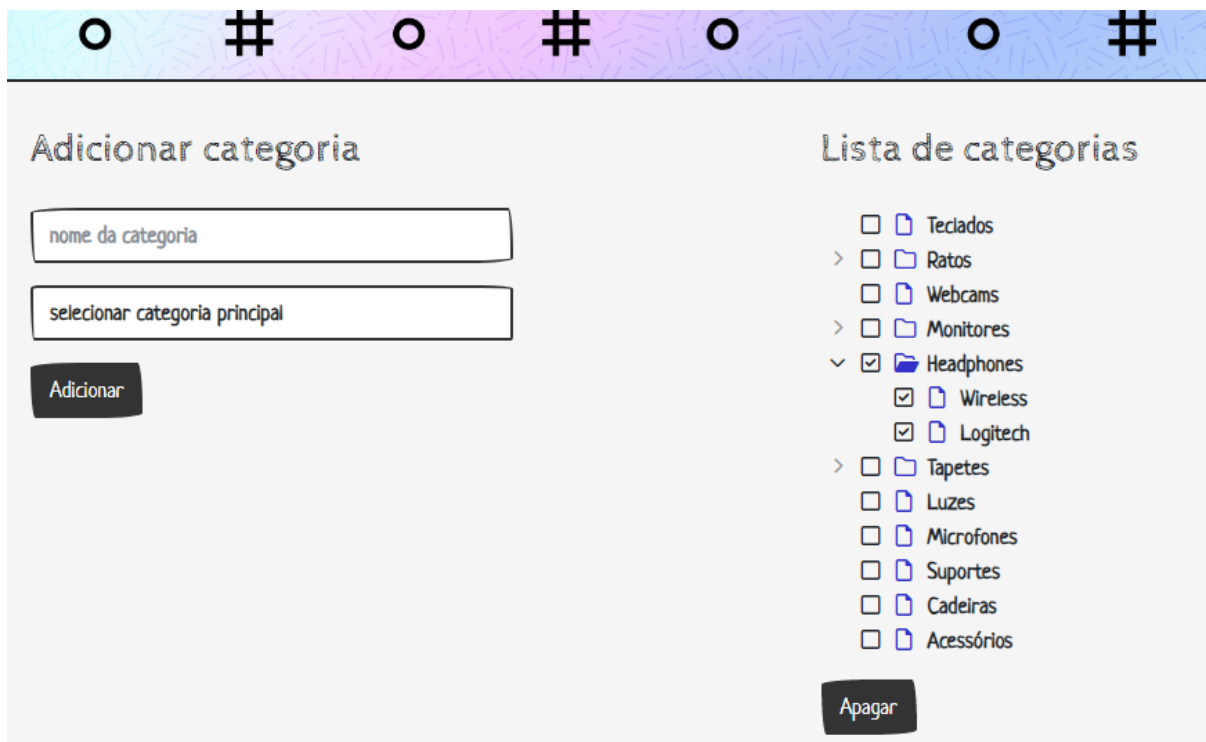


Figura 27: Página das categorias

### 3.4 Síntese

A implementação da loja *online* foi uma etapa crucial no desenvolvimento deste projeto. A construção da loja fictícia permitiu levantar as funcionalidades gerais necessárias para o funcionamento de qualquer tipo de loja *online*.

A partir da construção da loja fictícia, foram estabelecidos os elementos que servem de base para a ferramenta de criação de lojas *online* e geração de parte de loja *online*, especificamente os produtos. Esses elementos englobam as funcionalidades essenciais que os utilizadores desejam para gerir e personalizar as suas lojas, incluindo a gestão de produtos e categorias.

Mais especificamente, foram identificados os componentes responsáveis pela listagem e gestão de produtos, permitindo que os utilizadores possam facilmente adicionar, editar e remover produtos das suas lojas. Também foram criadas as funcionalidades de gestão de categorias, que permitem aos utilizadores categorizar os produtos em diferentes secções.

Em resumo, a construção desta loja permitiu identificar e desenvolver as funcionalidades para uma loja *online*. Esses elementos fundamentais vão ser aproveitados e usados como base na ferramenta de criação de lojas *online*, permitindo que os utilizadores personalizem a parte da loja gerada, e adicionem outros componentes necessários.



## 4 Desenvolvimento da Aplicação de Geração de Lojas

Nesta secção é descrito a aplicação para criação de uma parte de loja *online*, que inclui as funcionalidades, configurações da *framework*, e a utilização da ferramenta. Neste seguimento, foi estabelecido como objetivo esta aplicação abordar apenas a gestão e listagem de produtos e categorias, incluindo a personalização da página de listagem de produtos.

A ferramenta de geração de lojas utilizou as mesmas tecnologias que a loja *online* desenvolvida anteriormente e consiste em disponibilizar um formulário com parâmetros de customização, que são preenchidos pelo utilizador. Uma vez concluído o preenchimento do formulário, a parte da loja é obtida de acordo com os parâmetros definidos e o utilizador poderá fazer o *download* dos ficheiros correspondentes, usar a parte da loja criada a partir do seu computador e integrar com mais partes de loja.

### 4.1 Funcionalidades

As principais funções da ferramenta de geração de lojas incidem na customização da página de listagem e gestão de produtos, que inclui as funcionalidades de *backoffice* de produtos e de categorias. As funcionalidades da ferramenta estão divididas em duas categorias: customização, que inclui os parâmetros do formulário oferecido ao utilizador para customizar a página, e funções independentes do formulário, como a gestão do *backoffice* e outras funções relacionadas com a própria aplicação.

Desta forma, o formulário apresenta as seguintes funcionalidades para customização:

- Inserir nome e email do utilizador;
- Inserir nome da loja;
- Adicionar barra de navegação;
  - Adicionar barra de pesquisa;
  - Alterar cor da barra de navegação;
  - Alterar cor da letra;
  - Adicionar logótipo;
  - Alterar tamanho;
- Adicionar rodapé;
- Alterar cor dos botões;
- Alterar tamanho das imagens dos produtos;

- Ativar *darkmode*;
- Configurar campos do produto;
  - Alterar nome do campo;
  - Adicionar um novo campo;

As funcionalidades relacionadas com o *backoffice* e a própria aplicação incluem:

- Consultar lista de produtos;
- Consultar lista de categorias;
- Criar novo produto;
- Editar produto;
- Eliminar produto;
- Criar categorias e subcategorias;
- Eliminar categorias;
- Pesquisar produtos na lista;
- Pré-visualização da página customizada;
- Limpar o formulário;
- Guardar alterações;
- *Download* dos ficheiros;

## 4.2 Framework

A *framework* é composta por uma estrutura de código que serve de base à aplicação e é personalizada pela ferramenta para gerar uma parte específica da loja *online*. É importante notar que a ferramenta gera apenas uma parte da loja *online* baseada em certos componentes da *framework* que se concentram na gestão de produtos e categorias.

Nesta secção, serão apresentados os aspetos técnicos fundamentais da *framework*, incluindo a arquitetura, a API do *backend* e a base de dados. A arquitetura apresenta a estrutura de código dos componentes que foram implementados. A API do *backend* é responsável por fornecer os recursos de customização, gestão de produtos e categorias. Por fim, a base de dados inclui os modelos nos quais as informações relativas à loja e aos parâmetros de customização foram definidos.

## 4.2.1 Arquitetura

A arquitetura da *framework* é composta por quatro componentes do *backend*, que são: produtos, categorias, utilizadores e encomendas. Cada um destes componentes possui a sua própria API e base de dados correspondente, sendo responsáveis por gerir as informações específicas que compõem o sistema.

No âmbito deste projeto, foram desenvolvidos os componentes da gestão de produtos e categorias, que têm como principal função gerir os dados referentes a esses elementos, como demonstra a figura 28. O componente de produtos é responsável por listar, criar, apagar e editar produtos, bem como permitir a adição de um novo campo na personalização dos produtos. O componente de categorias inclui funções para criar, listar e apagar categorias. Estes componentes estão intimamente ligados, uma vez que um produto precisa de estar associado a uma categoria.

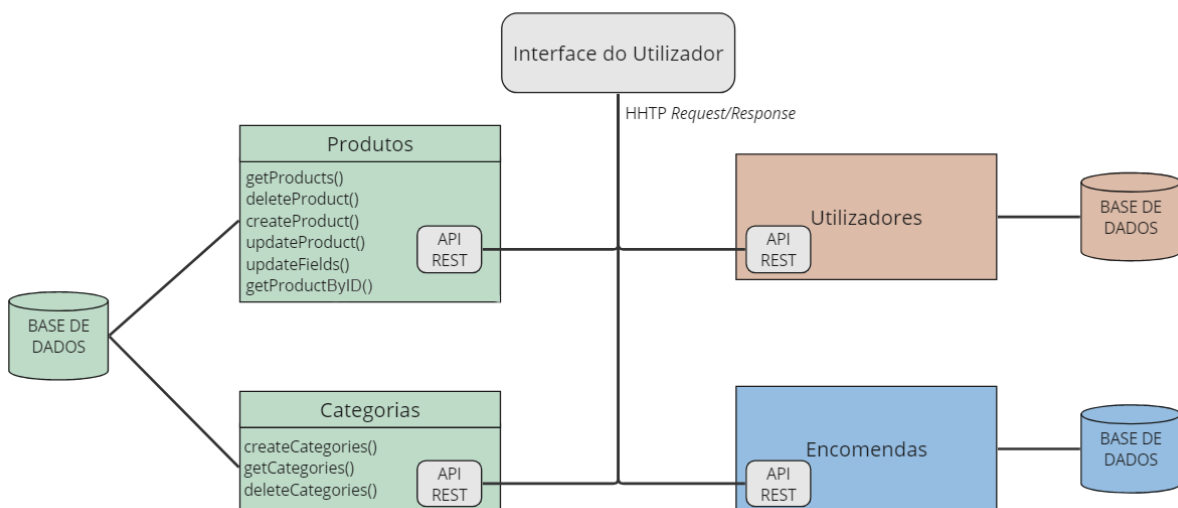


Figura 28: Arquitetura da *framework*

A interação da interface do utilizador com os componentes é feita através da API REST de cada um. A interface do utilizador representa a estrutura do *frontend* específica para o contexto de gestão de produtos e categorias, semelhante à interface da loja *online* desenvolvida.

Embora seja possível utilizar uma base de dados partilhada para gerir todos os componentes, adotou-se uma estrutura semelhante aos microserviços, em que cada componente tem a sua base de dados. Isto torna a integração de componentes mais simples uma vez que não é necessário fundir as suas base de dados. No entanto, ao contrário da arquitetura de microserviços só há um serviço que atende os pedidos de todos os componentes. As funções de cada componente são os respetivos

*controllers* que implementam a API *web*. Os métodos aplicados para o *controller* do componente Produtos foram: o método de *getProducts()* responsável por obter os produtos, *deleteProduct()* para remover produtos, o método *createProduct()* para criar produtos, o *updateProduct()* para editar os produtos, o método *getProductByID()* que obtém o produto identificado pelo id, e por fim, o método *updateFields()* responsável por atualizar o campo *novocampo* do modelo de produto. Este método é suportado na coleção de *Options*, em que é guardada a informação do utilizador acerca do nome do novo campo, e atualiza o campo *name* correspondente na coleção dos *Produtosapp*. Os restantes métodos estão associados à coleção *Produtosapp* da base de dados desenvolvida, apresentada na figura 29. No componente *Categorias* o *controller* contém os métodos: *createCategories()* responsável por criar categorias, o *getCategories()* para obter a lista de categorias e o *deleteCategories()* para eliminar categorias. Estes métodos estão associados à coleção *Categorias*. Todos estes métodos são independentes dos outros componentes da *framework*, que não foram desenvolvidos neste trabalho.

Os componentes *Utilizadores* e *Encomendas* não apresentam funções porque não fornecem qualquer API local aos componentes generalizados neste trabalho. Os componentes ficam assim isolados por dois tipos de API: uma *web* que é responsável por receber os pedidos vindos do *frontend*, e uma local para responder a eventuais necessidades dos outros componentes.

A estrutura física da *framework* é diferente da estrutura adotada para a loja, uma vez que, na *framework* os componentes implementados (produtos e categorias) são separados por pastas, permitindo uma separação clara das funções de cada um. Na secção 4.3.4, é apresentada a estrutura que demonstra como os componentes gerados foram organizados, de modo a facilitar a sua identificação e permitir uma melhor integração com os outros componentes.

Desta forma, a arquitetura adotada permite que os programadores possam trabalhar em paralelo em cada um dos componentes, sem afetar o funcionamento do sistema como um todo. Isso possibilita que o desenvolvimento seja mais independente e eficiente, uma vez que cada um pode se concentrar no seu componente sem interferir no trabalho dos outros.

## 4.2.2 API do Backend

Cada campo presente no formulário corresponde a um parâmetro do componente a ser customizado. Após o preenchimento do formulário, os valores são armazenados na base de dados e em seguida há uma substituição desses valores com os valores anteriores de cada campo, sendo assim uma customização dinâmica. Com base nessa atualização, é gerada a nova parte da loja que corresponde ao componente customizado.

As tabelas seguintes contêm informações detalhadas sobre os recursos definidos para cada funcionalidade da *framework*. Nestas tabelas, é possível verificar os métodos HTTP utilizados, os códigos de *status* retornados pelo servidor e as respectivas respostas.

Na tabela 4.6 são apresentados os recursos relativos ao formulário que permitem o acesso e customização do componente em questão. Nas tabelas 4.7 e 4.8, são apresentados os recursos relativos às funcionalidades de gestão associadas às categorias e aos produtos.

Tabela 4.6: Recursos relativos à customização

<b>Funcionalidade</b>	<b>URL</b>	<b>Método HTTP</b>	<b>Status</b>	<b>Resposta</b>
Criar página customizada	/cms/page	POST	200	Página é criada
			400	Erro ao criar página
Obter página customizada	/cms/page/:id	GET	200	Retorna página
			400	Erro ao retornar página

Tabela 4.7: Recursos relativos às categorias

<b>Funcionalidade</b>	<b>URL</b>	<b>Pedido HTTP</b>	<b>Status</b>	<b>Resposta</b>
Adicionar categoria	/category	POST	201	Categoria criada com sucesso
			400	Erro ao criar categoria
Consultar todas as categorias		GET	200	Retorna todas as categorias
			400	Erro ao retornar categorias
Eliminar categoria	/category/:id	DELETE	200	Categoria eliminada com sucesso
			400	Erro ao eliminar categoria

Tabela 4.8: Recursos relativos aos produtos

Funcionalidade	URL	Método HTTP	Status	Resposta	
Adicionar produto	/product	POST	200	Produto é adicionado	
			400	Erro ao criar produto	
Consultar todos os produtos		GET	200	Retorna todos os produtos	
			400	Erro ao retornar produto	
			404	Produto não encontrado	
Eliminar produto		DELETE	200	Produto é eliminado	
			400	Erro ao eliminar produto	
Editar um produto		/product/:id	PUT	200	Produto é editado
				400	Erro ao editar produto
Consultar um produto			GET	200	Retorna o produto
	400			Erro ao obter produto	
Adicionar <i>review</i>	/product/:id/review		POST	201	<i>Review</i> adicionada
				400	Erro ao adicionar <i>review</i>
Consultar produtos favoritos	/product/top	GET	200	Retorna produtos favoritos	
			400	Erro ao retornar favoritos	
Adicionar novo campo	/product/campo	PUT	200	Novo campo adicionado	
			400	Erro ao adicionar campo	

### 4.2.3 Base de dados

O modelo de dados da aplicação consiste em três coleções: *Options*, *Produtosapp*, e *Categorias*. A coleção *Options* armazena o conteúdo de todos os campos preenchidos no formulário de customização.

A coleção *Produtosapp* tem a estrutura idêntica à coleção de produtos da loja *online* com exceção do novo campo que o utilizador pode adicionar para acrescentar informação acerca do produto. Este novo campo é composto por um nome, que representa o nome do campo, e pelo *content*, que representa o conteúdo do campo. Esta opção de adicionar um campo é suportada na coleção *Options*. O campo *novocampo* dentro dessa coleção é do tipo *boolean* e armazena a posição positiva ou negativa do utilizador em querer adicionar um novo campo. Além disso, o campo "novocamponome" guarda o

nome do campo que o utilizador atribuir. Ainda na coleção *Produtosapp*, o campo do logótipo armazena a imagem selecionada pelo utilizador no formulário de customização e é convertido para o formato Base64 antes de ser armazenado na base de dados. Diferentemente das imagens dos produtos, que são armazenadas localmente e apenas o caminho para a imagem é armazenado na base de dados, a imagem do logótipo é armazenada como uma *string* na base de dados, permitindo que a parte da loja gerada possa descodificar a *string* e ter acesso à imagem do logótipo. Sendo que o tamanho máximo de um documento suportado pelo MongoDB é 16 megabytes, a imagem do logótipo deve ter um tamanho inferior a esse limite.

A coleção das *Categorias* não foi sujeita a nenhuma alteração, por isso tem exatamente a mesma estrutura do que a coleção de categorias da loja *online*. Esta coleção é responsável por categorizar os produtos.

O diagrama apresentado na figura 29 permite uma visualização detalhada das coleções e campos presentes no modelo de dados.



Figura 29: Diagrama do modelo de dados da aplicação de criação de lojas *online*

### 4.3 Ferramenta

A ferramenta de geração de lojas é responsável por personalizar o código da *framework* e gerar a parte loja *online* customizada. Nesta secção, são apresentadas as configurações gerais da loja, do *frontend* e do *backend*, além das configurações de saída. É apresentada a loja parcial gerada e personalizada de acordo com as opções do utilizador. Além disso, é discutida a integração da parte da loja com outros componentes.



### 4.3.1 Configurações gerais

A primeira secção do formulário é referente aos dados gerais do utilizador, em que deve introduzir informações básicas, como o seu nome, endereço de email e o nome da loja que deseja criar. Esses dados são importantes para o controlo de autenticação com um componente de Utilizadores a ser adicionado, em que para aceder às funcionalidades da *framework* o utilizador precisa de ser administrador.

Na figura 30, é possível observar a vista inicial do formulário, que apresenta diversas opções e campos a serem preenchidos pelo utilizador. É importante destacar que todas as informações solicitadas são fundamentais para a criação e funcionamento adequado da loja. Além disso, o formulário evidencia facilidade de uso para o utilizador.



Figura 30: Página de customização

### 4.3.2 Configurações do frontend

A próxima secção, apresentada na figura 31, inclui todas as configurações de customização do *frontend* e *backend* da página de listagem e gestão de produtos. Foram utilizados botões *toggle*, que apresentam uma interface simples e intuitiva, em que o utilizador pode facilmente ativar ou desativar uma funcionalidade apenas com um clique.

Quando o botão é pressionado, a funcionalidade associada é ativada e o botão reflete visualmente

essa mudança de estado. Quando pressionado novamente, a funcionalidade é desativada e o botão volta ao seu estado original. Esta abordagem é muito útil nas situações em que o utilizador precisa controlar uma determinada função de forma rápida e fácil, e se houver mais configurações associadas a essa função serão mostradas abaixo do botão correspondente.

The image shows a configuration interface for a product listing page. It features several sections with toggle switches and input fields:

- Incluir barra de navegação:** A toggle switch is turned on. Below it, a panel contains:
  - Incluir barra de procura:** A toggle switch is turned off.
  - Tamanho barra de navegação:** An input field with the placeholder text "Introduza tamanho".
  - Cor da barra de navegação:** A color picker set to "#4502ec".
  - Cor da letra:** A color picker set to "#ffffff".
  - Logotipo:** A file upload area with buttons for "Escolher ficheiro", "Nenhum ficheiro selecionado", and "Carregar".
- Adicionar rodapé:** A toggle switch is turned on. Below it, a text input field labeled "Texto do rodapé".
- Alterar campos do produto:** A toggle switch is turned on. Below it, a panel contains:
  - Alterar campo "Marca":** An input field with the placeholder text "Introduza campo".
  - Adicionar novo campo:** A toggle switch is turned off.
- Darkmode:** A toggle switch is turned on.
- Cor do botão:** A color picker set to "#4502ec".
- Tamanho das imagens:** An input field with the value "100".

Figura 31: Opções de customização

A primeira opção é de adicionar barra de navegação, em que o botão aparece por *default* como desativado. Se o utilizador pressionar o botão, é disparada a ação de que pretende adicionar a barra de navegação e novos campos relativos à customização da barra aparecem. As opções de customização da barra de navegação são: adicionar uma barra de pesquisa, adicionar um logótipo, alterar a cor da barra, alterar o tamanho e alterar a cor da letra.

Na figura 32 é possível visualizar o processo de adicionar o componente da barra de navegação à página de listagem de produtos que é iniciado na leitura da coleção respetiva à customização. O valor do campo *navbar* determina a decisão de adicionar a barra de navegação ou não. Se a condição for falsa, não é adicionada, se for verdadeira o processo de adição começa. Os parâmetros respetivos à estilização da barra de navegação referidos anteriormente são enviados para o componente da barra, incluindo o valor do campo respetivo à adição do componente da barra de pesquisa. O componente da barra de navegação é assim preenchido com os valores recebidos e é adicionado à página.

### Adicionar Barra de Navegação

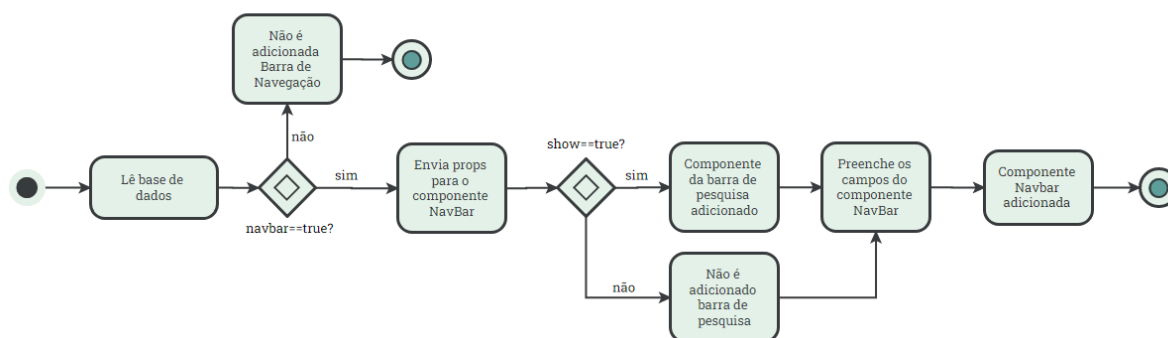


Figura 32: Diagrama de atividades de adicionar barra de navegação

A próxima opção, que já não faz parte da customização da barra de navegação, é a da adição de um rodapé. No caso do utilizador acionar o botão, a adição é feita e uma caixa de texto para preencher o respetivo conteúdo é apresentada.

A próxima opção é referente ao aspeto da tabela em que é possível estilizar em *darkmode*. Inicialmente a tabela encontra-se com o estilo base que é um fundo branco e cinzento claro, se o botão for acionado o estilo da tabela de listagem dos produtos fica com o fundo cinzento escuro.

A seguir, é possível escolher uma cor para o botão da página e, por fim, definir o tamanho das imagens.

Para personalizar elementos da interface gráfica que utilizam CSS, como cor e tamanho, foi utilizado a biblioteca *styled-components* [34]. Essa biblioteca permite escrever CSS no mesmo ambiente em que está o código *javascript*, facilitando a manipulação desses elementos diretamente no código fonte do componente a personalizar. Os pedaços de código em CSS são formatados em *template literals* [35] do JavaScript. Esta funcionalidade permite inserir as variáveis de customização dentro da string de CSS.

### 4.3.3 Configurações do backend

A opção seguinte diz respeito a uma customização que envolve o *backend* da loja que surgiu de forma a que o modelo dos produtos fosse o mais indicado possível para o tipo de produto que o utilizador pretender. Primeiramente, é permitido ao utilizador alterar o campo *marca* para outro nome, e é possível adicionar um novo campo. Este processo está descrito no diagrama da figura 33.

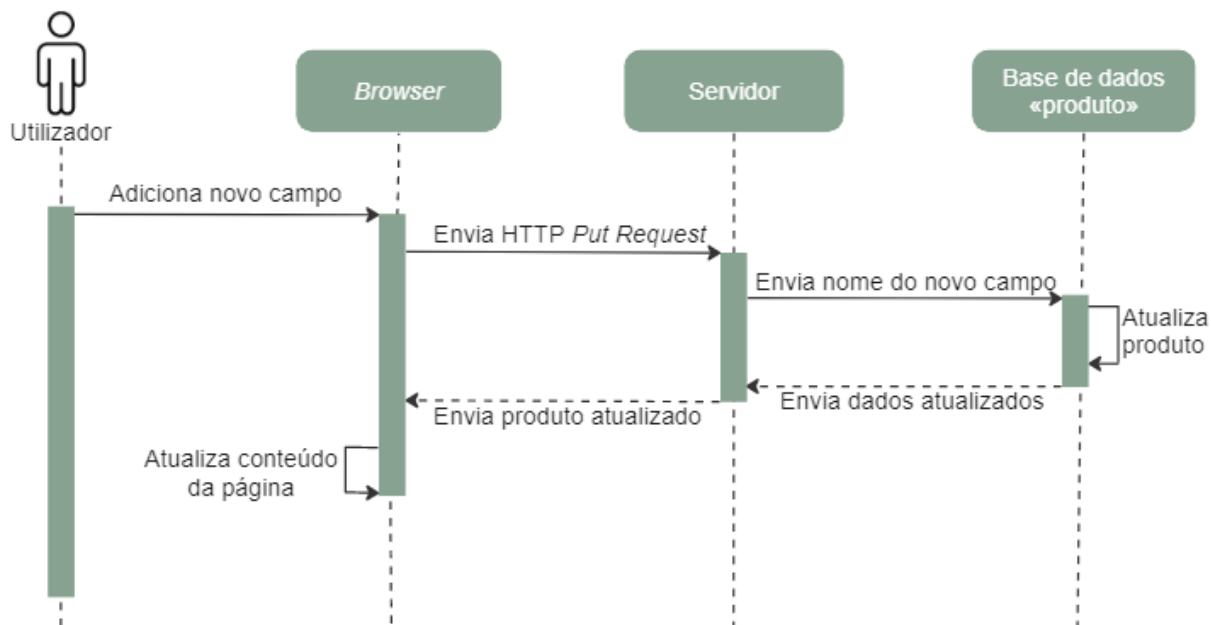


Figura 33: Diagrama de sequência de adicionar novo campo

Inicialmente o campo *novocampo*, no modelo do produto encontra-se vazio e quando acionada esta opção por parte do utilizador, o nome dado para o novo campo é adicionado à base dados. Desta forma, é possível a customização do modelo do produto de acordo com as necessidades do utilizador. Para além de aparecer este novo dado na listagem dos produtos, ao adicionar um produto este campo é incluído na customização do mesmo.

#### 4.3.4 Output da ferramenta

A ultima secção da interface da ferramenta de geração apresenta os botões de pré-visualizar, limpar, guardar e *download*. A funcionalidade de pré-visualizar consiste em exibir ao utilizador a página formatada de acordo com as opções que selecionou na secção anterior e em tempo real, ou seja, à medida que o utilizador vai selecionando as opções de customização, esta secção apresenta a página de acordo com essas escolhas.

O botão limpar serve para colocar o valor de todos os campos para o valor inicial apresentado, que são os campos limpos.

A opção de guardar é para o utilizador guardar as alterações, que se desejar, pode voltar à aplicação mais tarde e continuar a customização. Estas alterações são guardadas na base de dados.

O ultimo botão é para efetuar o *download* dos ficheiros. Para os ficheiros do *download* corresponderem à customização feita é preciso primeiramente guardar as alterações, se não será

gerada a *framework* com a customização *default* baseada na loja *online* anteriormente desenvolvida. Após clicar no botão e iniciado o *download*, é mostrado ao utilizador uma mensagem de sucesso acerca do *download* e os campos do formulário são limpos.

O *download* consiste na transferência de um ficheiro *zip*, que é construído recorrendo à biblioteca JSZip[13], com todos os ficheiros que são precisos para o funcionamento da *framework*, incluindo a organização por pastas organizadas de acordo com a estrutura da figura 34. Para guardar os ficheiros diretamente no computador do utilizador, e ser feito o *download* com sucesso, a solução foi implementada com o recurso ao pacote *file-saver* [9].

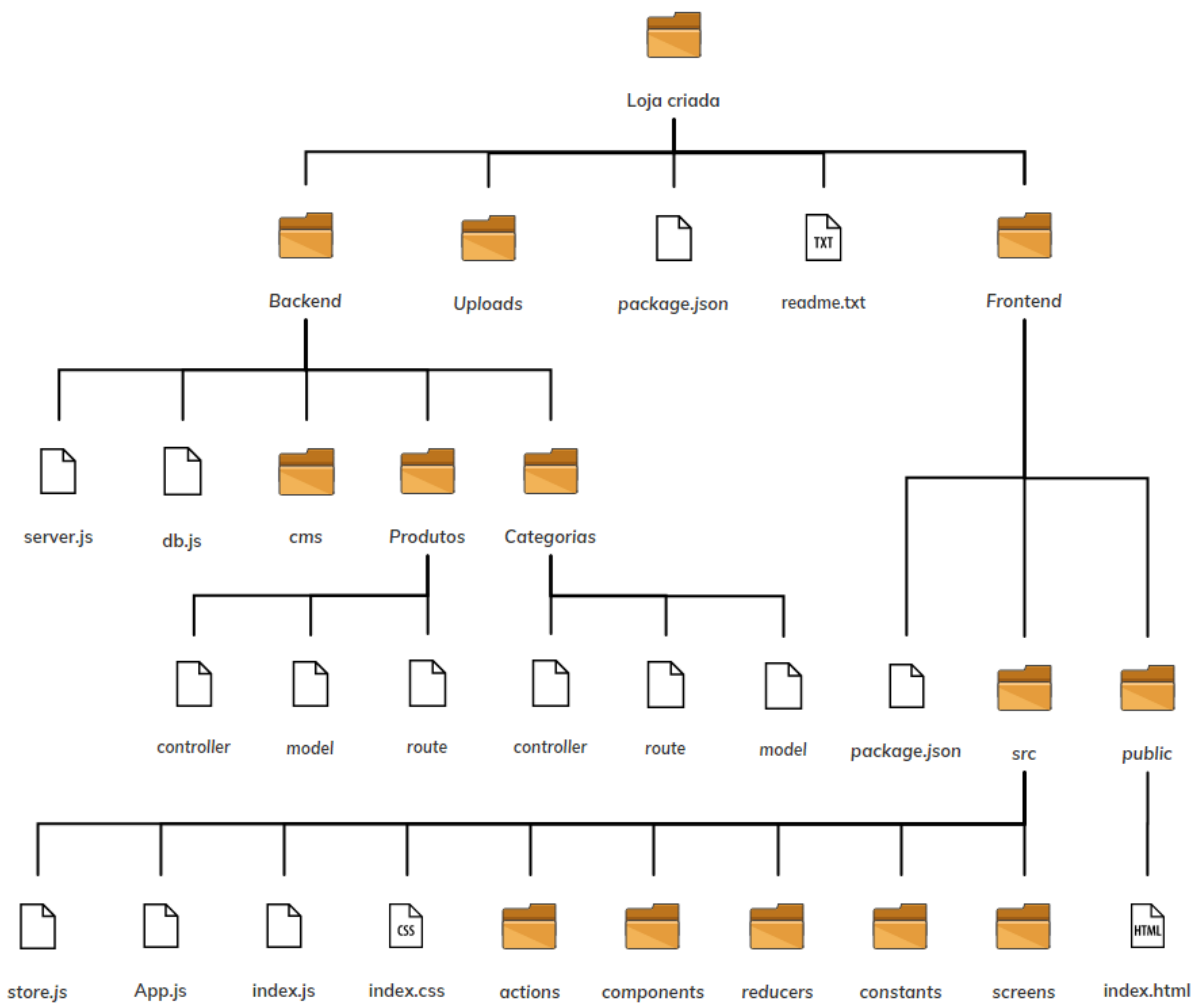


Figura 34: Estrutura dos ficheiros da aplicação de geração de lojas

A tabela 4.9 apresenta uma lista de todos os ficheiros que são armazenados pelo utilizador da aplicação, juntamente com as suas respetivas funções. Cada ficheiro é identificado pelo nome e o tipo de ficheiro.

Tabela 4.9: Ficheiros gerados pela aplicação e respetivas funções

<b>Ficheiro</b>	<b>Função</b>
package.json	Lista de dependências via npm para instalação no <i>backend</i> e <i>frontend</i> .
index.html	Conteúdo html que é exibido no navegador.
App.js	Responsável por definir as rotas do React.
index.css	Responsável pela estilização dos componentes.
index.js	Componente principal que implementa os outros componentes.
store.js	Responsável por guardar os estados globais da aplicação( <i>state</i> ).
productActions.js	Funções que definem alterações do estado do produto como: listar, criar, eliminar e editar produtos.
categoryActions.js	Funções que descrevem alterações do estado das categorias como: listar, criar e eliminar categorias.
productReducer.s	Funções que guardam estado atual das alterações dos produtos.
categoryReducer.js	Funções que guardam estado atual das alterações das categorias.
Footer.js	Componente do rodapé.
Header.js	Componente do cabeçalho.
SearchBox.js	Componente da barra de procura.
Message.js	Componente responsável pelas mensagens de erro e sucesso.
useHome.js	Responsável por obter os dados da customização.
ProductEditScreen.js	Página de edição e criação de um produto.
ProductListScreen.js	Página de listagem e gestão de produtos.
CategoriesList.js	Página de listagem e gestão das categorias.
server.js	Responsável pela atribuição das rotas e iniciação do servidor.
productModel.js	Responsável pela definição do modelo de dados dos produtos.
categoryModel.js	Responsável pela definição do modelo de dados das categorias.
cmsModel.js	Responsável pela definição do modelo de dados da customização.
productRoutes.js	Define as rotas associadas aos produtos.
categoryRoutes.js	Define as rotas associadas às categorias.
cmsRoutes.js	Define as rotas associadas á customização.
uploadRoutes.js	Define as rotas associadas ao upload de imagens.

productController.js	Funções responsáveis pela criação, listagem, eliminação e edição de produtos.
categoryController.js	Funções responsáveis por listar, criar e eliminar categorias.
db.js	Responsável pela criação e conexão com a base de dados.
README.txt	Instruções de uso da <i>framework</i> .

## 4.4 Loja parcial gerada

Após a transferência do ficheiro *zip*, o utilizador tem total acesso ao conteúdo da parte de loja formada. Para aceder às suas funcionalidades, primeiramente o programador tem de extrair a pasta para o seu computador e de seguida inserir o comando *npm install* na linha de comandos para instalação dos pacotes presentes no ficheiro *package.json*, tanto no *backend* como no *frontend*.

Depois da instalação dos pacotes nos dois lados, o comando para iniciar tanto o servidor como a aplicação é o *npm run dev*. A aplicação abre no *browser* e a página inicial é a página de listagem de produtos. Para aceder à página de criação de produtos aciona-se o botão de criar produto ou pela pesquisa no *browser* pelo URL correspondente, de igual forma para a página de categorias.

A página da gestão e listagem de produtos passou por alterações significativas em termos de estilização e de gestão, como podemos observar na figura 35.

Uma vez que o utilizador seleciona a opção de adicionar um novo campo no formulário de customização, o novo campo é adicionado à listagem dos produtos, o que difere da loja *online* que não oferece essa funcionalidade. Com essa possibilidade, o utilizador fica apto a vender qualquer tipo de produto.

Além disso, foi adicionada a funcionalidade de pesquisa por produtos na tabela, tornando mais fácil e rápido encontrar um produto específico na lista. Quando o utilizador insere o nome de um produto na caixa de pesquisa, apenas os produtos correspondentes ao termo de pesquisa são exibidos na tabela, facilitando a gestão de produtos em lojas com um grande número de produtos.

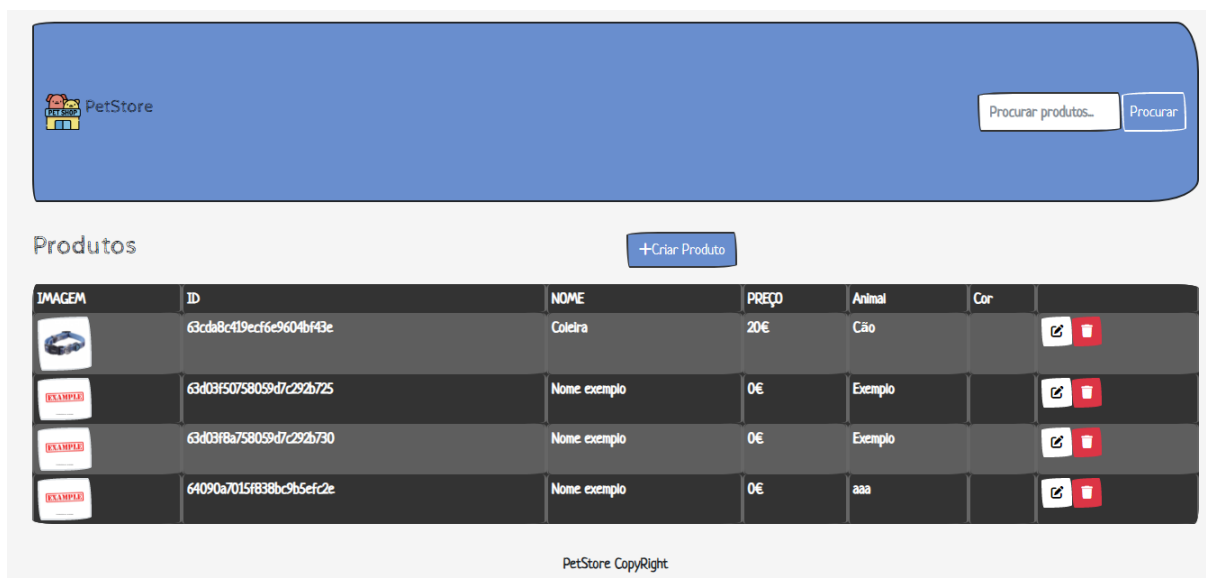


Figura 35: Página customizada

O utilizador tem acesso às funções de gestão de produtos nesta página. Tal como acontecia na loja *online*, pressionando o botão com o ícone do lixo, o produto da mesma linha é eliminado. Ao clicar no botão de edição, o utilizador é redirecionado para a página de editar produto onde pode alterar o conteúdo dos campos do produto, incluindo o novo campo adicionado, como podemos observar na figura 36.

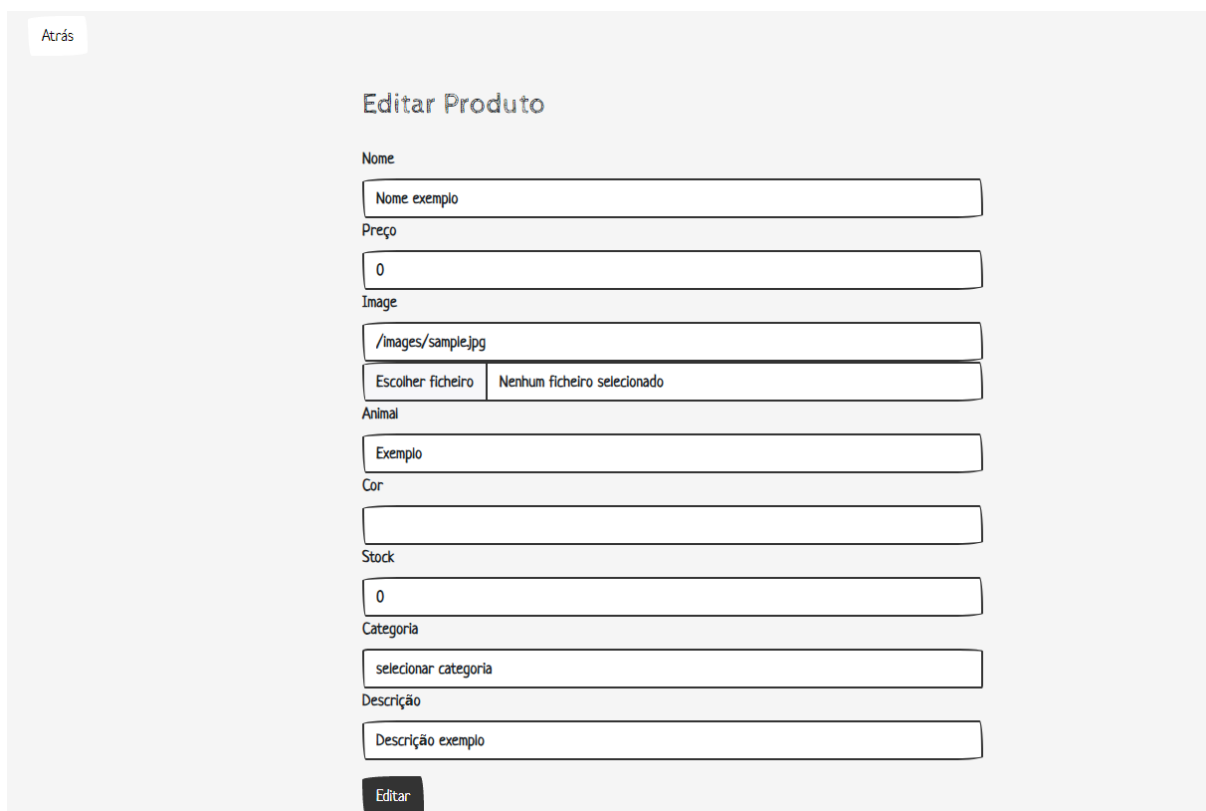


Figura 36: Página de criar produto



A criação de um novo produto apenas difere, comparando com a loja *online*, na existência do novo campo que pode ser preenchido com o formato *String* e *Integer*.

Visto que para criar um novo produto, é necessário atribuir uma categoria, também é incluída essa possibilidade como demonstra a figura 37. Para isso o utilizador quando acede à página das categorias, pode fazer a sua gestão de igual forma à loja *online*, criando categorias, subcategorias, sub subcategorias e eliminar as que desejar. Desta forma, ao criar um produto pode associar a categoria pretendida e o componente de listagem e gestão dos produtos fica completo.

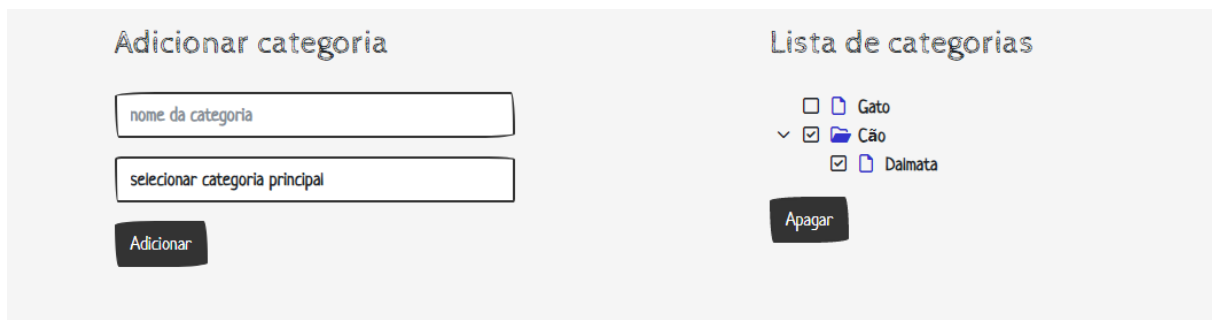


Figura 37: Página de gerir categorias

#### 4.4.1 Integração com outros componentes

É necessário adicionar outros componentes para se obter uma loja totalmente funcional e com os requisitos mínimos para o seu funcionamento.

Esta integração pode ser feita manualmente, em que os novos ficheiros sigam a mesma estrutura dos componentes implementados correspondentes à gestão de produtos e categorias apresentado na figura 34. Para adicionar um novo componente, o *backend* gerado tem que receber um conjunto de ficheiros que formam um componente funcional:

- Ficheiro *controller*: onde estariam as funcionalidades específicas para as rotas do novo componente.
- Ficheiro *model*: responsável por guardar o esquema de dados do componente.
- Ficheiro *routes*: com as rotas definidas de acordo com as novas funcionalidades do componente.

Depois de adicionar estes novos ficheiros que formam um componente no *backend*, apenas é necessário acrescentar no ficheiro *server.js* a rota da API do novo componente representado na figura 38. Assim o *backend* do componente adicionado está pronto a ser utilizado.

```
app.use('/api/category', categoryRoutes)
app.use('/api/products', productRoutes)
app.use('/api/users', userRoutes)
```

Figura 38: Edição do ficheiro server.js

Para adicionar o *frontend*, o processo é semelhante ao do *backend* mas mais extenso, sendo preciso adicionar mais ficheiros e configurar a comunicação com o *backend*.

Para demonstrar o funcionamento da integração de componentes, foi utilizado o componente *Utilizadores* como exemplo para efetuar o Registo de utilizadores.

Primeiramente, é necessário adicionar o código da página que é exibida ao utilizador na pasta *screens*, que será a mesma que a loja *online*.

Em seguida, foram acrescentados os ficheiros correspondentes ao *Reducer*, *Constants* e *Actions* que fazem interligação com o *backend* do componente *Utilizadores* e gerem os estados da aplicação. Para configurar a comunicação entre o *frontend* e o *backend* do sistema, é utilizado a biblioteca *Axios* que permite fazer as requisições HTTP do lado do cliente [37].

Com estes novos ficheiros adicionados, o próximo passo é editar o ficheiro *store.js* e declarar os estados do novo componente, que estão guardados no ficheiro *Reducer* adicionado previamente, conforme a figura 39.

```
export const store = configureStore({
  reducer: {
    productList: productListReducer,
    productDetails: productDetailsReducer,
    productDelete: productDeleteReducer,
    productCreate: productCreateReducer,
    productUpdate: productUpdateReducer,
    userLogin: userLoginReducer,
    userRegister: userRegisterReducer,
    userDetails: userDetailsReducer,
    userUpdateProfile: userUpdateProfileReducer,
    userList: userListReducer,
    userDelete: userDeleteReducer,
    userUpdate: userUpdateReducer,
```


Figura 39: Edição do ficheiro store.js

No ficheiro *App.js* é preciso inserir a rota correspondente à página do Registo de utilizadores, como demonstra a figura 40. Ao inserir um novo utilizador, a sua informação é guardada na coleção da base de dados dos *Utilizadores* como é esperado.

```
<Router>
  <main className="py-3">
    <Container>
      <Routes>
        <Route path="/productlist" element={}<ProductListScreen /> exact />
        <Route path="/product/:id/edit" element={}<ProductEditScreen /> />
        <Route path="/register" element={}<RegisterScreen /> />
        <Route path="/categories" element={}<CategoriesList /> exact />
        <Route path="/download" element={}<Download /> exact />
```

Figura 40: Edição do ficheiro App.js

Assim, a página do Registo de Utilizadores ficou funcional e integrada com os componentes de *backend* e *frontend* correspondentes aos *Utilizadores* e mostrada na figura 41.



The image shows a registration form with the following elements:

- Registar** (Title)
- Nome** (Label) with an input field containing the placeholder text "Introduza nome".
- Email** (Label) with an input field containing the placeholder text "Introduza email".
- Password** (Label) with an input field containing the placeholder text "Introduza password".
- Confirme Password** (Label) with an input field containing the placeholder text "Confirme password".
- Criar conta** (Submit button).
- Já tem conta? [Entrar](#)** (Link for existing users).

Figura 41: Página de registar novo utilizador

Este processo de integração de componentes é importante para garantir a funcionalidade do sistema como um todo e facilitar o desenvolvimento de novas funcionalidades sendo que seguindo esta lógica, o utilizador pode adicionar outros componentes e completar assim a loja gerada.

## 5 Conclusão

Inicialmente, realizou-se uma pesquisa do estado da arte em aplicações semelhantes, com o objetivo de reunir requisitos e funcionalidades relevantes para a construção das ferramentas em questão. Devido à fase de aprendizagem inicial das tecnologias, a construção da loja *online* acabou por ser um processo mais demorado e complexo do que o esperado, mas permitiu implementar a loja com funcionalidades importantes. A loja final permitiu identificar os requisitos necessários para a construção de qualquer tipo de loja, e servir como base para a criação da ferramenta de geração de lojas.

Após a loja *online* estar completamente funcional, foi iniciado o desenvolvimento da ferramenta de geração de lojas que abrange as funcionalidades respetivas aos componentes de produtos e categorias. Primeiramente, foram definidas quais as funcionalidades que a ferramenta deveria implementar e que estrutura deveria adotar para que a integração com outros componentes fosse possível. Isso envolveu a análise da estrutura física da loja e a identificação de como cada componente estava organizado. Com a definição dessas funcionalidades e estrutura, foi criado um formulário para que o programador pudesse preencher com as personalizações a serem incluídas na parte de loja gerada. De seguida, foram tratadas as questões do *download* dos ficheiros para que o programador tivesse total acesso ao código da parte de loja que criou. Por fim, foi realizada a fase de testes de funcionalidade da parte de loja gerada e a integração dessa parte com outros componentes.

### 5.1 Discussão de resultados

A aplicação de geração de lojas desenvolvida permite a criação rápida e simples de uma parte da loja *online* responsável pela listagem e gestão de produtos e categorias, personalizada pelo utilizador. O componente responsável pelas categorias também é criado, uma vez que a associação de um produto a uma categoria é necessária. O componente das categorias não sofre alterações em termos de funcionalidades e estilização em relação ao da loja *online*, sendo apenas inserido para completar o componente de listagem e gestão de produtos.

A parte da loja criada é disponibilizada ao utilizador através do *download* de um arquivo *zip* contendo todos os ficheiros necessários. Após a instalação dos pacotes requeridos, o utilizador pode aceder ao conteúdo no *browser*, podendo ainda adicionar outras partes de uma loja manualmente, se necessário.

A funcionalidade de customização de produtos é um dos recursos mais importantes. Ao permitir que o utilizador personalize o modelo de produtos de acordo com as suas necessidades, a loja torna-se capaz de oferecer qualquer tipo de produto ou serviço. Além disso, a customização oferece várias opções de

estilização que têm realmente impacto na mudança do aspeto da página, tornando-a mais atraente para o utilizador.

A possibilidade de integração da parte da loja com outras partes necessárias para o funcionamento de uma loja *online*, também é uma funcionalidade importante. Esta funcionalidade é bastante pertinente porque permite ao utilizador integrar as partes que achar necessárias para o funcionamento de uma loja completa, de acordo com as suas necessidades específicas.

A ferramenta de geração de lojas é uma solução que atinge os principais objetivos de poupar tempo e recursos ao programador. Com ela, não é necessário desenvolver todo o *backend* e *frontend* do zero para obter uma loja *online* funcional e customizada. Além disso, com o acesso que o programador tem aos ficheiros da parte de loja, permite a inclusão de outros componentes criados por terceiros, o que significa que o programador pode expandir as funcionalidades da sua loja sem ter que passar pelo trabalho de criar tudo de raiz.

## 5.2 Conclusões e trabalho futuro

A dissertação teve como objetivo principal desenvolver uma ferramenta para a geração de lojas *online*, juntamente com uma aplicação básica de loja *online*. Neste sentido, pode-se afirmar que a meta foi atingida com sucesso.

A solução desenvolvida é bastante abrangente em termos de funcionalidades e, portanto, pode ser utilizada como base para o desenvolvimento de ferramentas mais avançadas no futuro. Isso significa que a dissertação é apenas o ponto de partida para futuras melhorias e inovações. Apesar disso, a ferramenta ainda possui algumas limitações e possibilidades de melhoria. Por exemplo, seria desejável incluir uma maior diversidade de customização para as lojas criadas, para que cada loja possa ter uma aparência quase única. Isto poderá incluir a criação de *templates* com diferentes estilos que alterem a aparência de vários elementos de uma só vez, e também a adição de um *template* de criação de produtos ao *backend* gerado, de forma ao utilizador não ter que inserir todos os campos um a um, mas serem preenchidos automaticamente com base em produtos inseridos anteriormente, mantendo o processo de criação de loja rápido e simples. Além disso, a funcionalidade de integração de componentes poderia ser aprimorada, tornando-a mais automática e menos manual. Outra área de melhoria seria a implementação de suporte ao cliente, para os utilizadores esclarecerem possíveis dúvidas na geração e utilização da loja.

No entanto, é importante ressaltar que a implementação de tais melhorias requer um alto nível de complexidade e abrangência de competências, o que exigiria mais tempo e recursos para serem

realizadas. Portanto, apesar das limitações e áreas de melhoria, a dissertação foi capaz de oferecer uma solução sólida e funcional para a criação de lojas *online*.

## Referências

- [1] *About Node.js*. URL: <https://nodejs.org/en/about>. (acedido em 12.05.2023).
- [2] Daniel Bugl. *Learning Redux*. Packt Publishing, 2017. ISBN: 978-178-646-239-8.
- [3] *Build amazing things*. URL: <https://www.npmjs.com/>. (acedido em 23.05.2022).
- [4] *CSS Introduction*. URL: [https://www.w3schools.com/css/css\\_intro.asp](https://www.w3schools.com/css/css_intro.asp). (acedido em 11.01.2023).
- [5] Eelco Plugge David Hows Peter Membrey. *Introdução ao MongoDB*. Novatec Editora, 2019. ISBN: 978-857-522-808-1.
- [6] Darren DeMatas. *11 Best Ecommerce Platforms Compared and Rated For 2022*. URL: <https://www.ecommerceceo.com/ecommerce-platforms/>. (acedido em 12.05.2022).
- [7] *Express*. URL: <https://expressjs.com/>. (acedido em 12.05.2023).
- [8] *Express Tutorial*. URL: [https://developer.mozilla.org/en-US/docs/Learn/Server-side/Express\\_Nodejs/routes](https://developer.mozilla.org/en-US/docs/Learn/Server-side/Express_Nodejs/routes). (acedido em 24.01.2023).
- [9] *FileSaver.js*. URL: <https://www.npmjs.com/package/file-saver>. (acedido em 12.01.2023).
- [10] *Free themes for Bootstrap*. URL: <https://bootswatch.com/>. (acedido em 11.01.2023).
- [11] David Herron. *Node.js Web Development*. 5th Edition. Packt Publishing, 2020. ISBN: 978-183-898-757-2.
- [12] *JavaScript SDK script configuration*. URL: <https://developer.paypal.com/sdk/js/configuration/>. (acedido em 23.07.2022).
- [13] *JSZip*. URL: <https://www.npmjs.com/package/jszip>. (acedido em 12.01.2023).
- [14] *mailgun-js*. URL: <https://www.npmjs.com/package/@types/mailgun-js>. (acedido em 17.07.2022).
- [15] Azat Mardan. *Express.js Guide: The Comprehensive Book on Express.js*. Createspace Independent Pub, 2013. ISBN: 978-149-426-927-2.
- [16] *MongoDB and Mongoose: Compatibility and Comparison*. URL: <https://www.mongodb.com/developer/languages/javascript/mongoose-versus-nodejs-driver/>. (acedido em 22.01.2023).

- [17] *MongoDB Charts*. URL: <https://www.mongodb.com/docs/charts/>. (acedido em 27.10.2022).
- [18] *Multer*. URL: <http://expressjs.com/en/resources/middleware/multer.html>. (acedido em 23.10.2022).
- [19] *O que é a API RESTful?* URL: <https://aws.amazon.com/pt/what-is/restful-api/>. (acedido em 23.01.2023).
- [20] *React*. URL: <https://react.dev/>. (acedido em 12.05.2023).
- [21] *React Bootstrap*. URL: <https://react-bootstrap.github.io/getting-started/introduction>. (acedido em 12.11.2022).
- [22] *react-paypal-button-v2*. URL: <https://www.npmjs.com/package/react-paypal-button-v2>. (acedido em 17.07.2022).
- [23] *Redux*. URL: <https://redux.js.org/>. (acedido em 23.07.2022).
- [24] *Redux and The Flux Architecture*. URL: <https://www.geeksforgeeks.org/redux-and-the-flux-architecture/>. (acedido em 23.01.2023).
- [25] *Redux DevTools*. URL: <https://chrome.google.com/webstore/detail/redux-devtools/lmhkpmbekcpmknklioebfkpmmfibljd>. (acedido em 23.01.2023).
- [26] Wellington Oliveira dos Santos. URL: <https://blog.tecsinapse.com.br/utilizando-react-redux-firebase-2bf93ea9f422>. (acedido em 23.01.2023).
- [27] *Sellfy*. URL: <https://sellfy.com/>. (acedido em 15.05.2022).
- [28] *Shopify*. URL: <https://www.shopify.com/>. (acedido em 12.05.2022).
- [29] *Shopify Engineering*. URL: <https://shopify.engineering/e-commerce-at-scale-inside-shopifys-tech-stack>. (acedido em 17.05.2022).
- [30] *Shopkit*. URL: <https://shopk.it/>. (acedido em 15.05.2022).
- [31] *Squarespace*. URL: <https://pt.squarespace.com/>. (acedido em 15.05.2022).
- [32] *Squarespace Engineering*. URL: <https://engineering.squarespace.com/>. (acedido em 18.05.2022).
- [33] *Squarespace Reviews, Product Details*. URL: <https://www.g2.com/products/squarespace/reviews>. (acedido em 07.01.2023).



- [34] *styled-components*. URL: <https://www.npmjs.com/package/styled-components>. (acedido em 11.12.2023).
- [35] *Template literals*. URL: [https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Template\\_literals](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Template_literals). (acedido em 11.12.2023).
- [36] *Try React*. URL: <https://legacy.reactjs.org/docs/getting-started.html>. (acedido em 21.01.2023).
- [37] *What is Axios?* URL: <https://axios-http.com/docs/intro>. (acedido em 21.01.2023).
- [38] *What is HTTP?* URL: [https://www.w3schools.com/whatis/whatis\\_http.asp](https://www.w3schools.com/whatis/whatis_http.asp). (acedido em 22.01.2023).
- [39] *Wix*. URL: <https://wix.com/>. (acedido em 14.05.2022).
- [40] *Wix Engineering*. URL: <https://www.wix.engineering/>. (acedido em 18.05.2022).