

Trabalho realizado por Gonçalo Gil 8200335, Luís Costa 8200737 e José Pedro Conde 8200350

Inserir nome do trabalho

**Programação em Ambiente Web**

Índice

[**Introdução** 4](#_Toc105955626)

[Contextualização da temática e a sua pertinência 4](#_Toc105955627)

[Objetivos do trabalho 4](#_Toc105955628)

[**1-** **Autenticação** 5](#_Toc105955629)

[1.1- Login 5](#_Toc105955630)

[**2-** **Controllers** 6](#_Toc105955631)

[2.1- Accounts Controller 6](#_Toc105955632)

[2.2- Books Controller 7](#_Toc105955633)

[2.3 -Client Controller 8](#_Toc105955634)

[2.4 – Employees Controller 9](#_Toc105955635)

[2.5 – Login Controller 9](#_Toc105955636)

[2.6 – Purchase Controller 10](#_Toc105955637)

[**3-** **Rotas** 11](#_Toc105955638)

[3.1- Book Route 11](#_Toc105955639)

[3.2- Client Route 11](#_Toc105955640)

[3.3- Employee Route 11](#_Toc105955641)

[3.4- Purchase Route 11](#_Toc105955642)

[**4-** **Frontend** 12](#_Toc105955643)

[4.1 – Login/Registo 12](#_Toc105955644)

[4.2- Client Dash 13](#_Toc105955645)

[4.3- Market Place 14](#_Toc105955646)

[4.4- Selling 15](#_Toc105955647)

[4.5- Admin Dash 16](#_Toc105955648)

[**Outros aspetos importantes** 17](#_Toc105955649)

[Testes à API 17](#_Toc105955650)

[Documentação 17](#_Toc105955651)

[Organização do trabalho através do GitHub 17](#_Toc105955652)

[Melhorias futuras 18](#_Toc105955653)

[**Conclusão** 19](#_Toc105955654)

[**Bibliografia** 20](#_Toc105955655)

# **Introdução**

# Contextualização da temática e a sua pertinência O presente trabalho foi realizado para a disciplina de Programação em Ambiente Web na licenciatura de Segurança Informática em Redes e Computadores. Esta, de um modo geral, dedica-se ao estudo de modelo cliente-servidor e as principais tecnologias/protocolos associados ao desenvolvimento web, tal como a conceção de aplicações web a partir de *frameworks* como o *NODE.JS* e *ANGULAR.JS*

# Objetivos do trabalho

Conceber uma aplicação para uma pequena empresa no ramo bibliotecário que iniciou projeto de modernização e digitalização do seu negócio. Sendo assim, foi necessário conceber a capacidade de gerir o stock de livros novos e usados na empresa assim como recompensar os clientes pelas suas compras na loja. Foi também necessário disponibilizar os dados do consumo, o tipo de produtos disponíveis bem como as recompensas de fiabilidade.

# **Autenticação**

Através desta componente é possível distinguir a autenticação de um funcionário e de um administrador, sendo enviada uma mensagem de erro caso um funcionário tente aceder a esta página.

Uma imagem com texto

Descrição gerada automaticamente

Figura 1

# 1.1- Login

De forma a fazer login na aplicação, o servidor verifica a existência da informação inserida e, uma vez confirmada essa informação, é retornado um “token” com os dados do utilizador ocultados de modo que estes não circulem de forma visível na rede.

Uma imagem com texto

Descrição gerada automaticamente

Figura 2

# **Controllers**

# 2.1- Accounts Controller

Esta entidade armazena todas as funções relativas às contas dos sistemas de pontos dos clientes, ou seja, as funções “get”, “show”, “create”, “update” e “delete”.

* **Index** – Esta função é cedida através de um “get request” feito à nossa API e é responsável por retornar todas as carteiras presentes na base de dados representando assim todos os clientes que possuem uma assinatura para o sistema de pontos;
* **Show** – Esta função é igualmente cedida através de um “get request” feito à nossa API, recebendo como parâmetro o id do cliente. De seguida, é executado uma pesquisa na base de dados, mais concretamente à tabela “account”, por uma entrada onde o valor do campo “cliente\_id” corresponde ao valor recebido pela rota no parâmetro “cliente\_id” e armazena a resposta numa variável “account”. No passo seguinte, a função verifica o valor da variável “account” analisando se esta contém um valor nulo ou diferente deste. No caso de a variável armazenar um valor nulo, a API retorna uma resposta com o código HTTP 404 informando o utilizador de que o cliente em causa não possui qualquer plano de pontos.
* **Create** – Na função “create”, a nossa API recebe através do corpo do pedido a informação “client\_id”, “points”, “transfer\_description” e regista toda esta informação na base de dados.
* **Update** – Para a função update, é enviado obrigatoriamente o id da conta e opcionalmente a descrição da transferência ou os pontos, informação esta que será utilizada para atualizar a informação guardada anteriormente na base de dados.
* **Delete** – Para a função delete é enviado novamente o id da conta e é executada uma procura na base de dados por uma conta com o id igual ao id enviado para a nossa rota e no caso de se verificar a existência de uma conta que cumpra estes requisitos esta é então removida da base de dados. Caso contrário a API retornará uma mensagem ao utilizador com um código HTTP 404 informando-o da não existência de uma conta com aquele id.

# 2.2- Books Controller

Esta componente contém todas as funções relativas aos livros dos clientes, as funções “search”, “index”, “create”, “update” e “delete”.

* **Search** – Esta função permite efetuar uma pesquisa na base de dados através do autor, título ou isbn. Caso não exista nenhum livro com um desses parâmetros, a API retorna uma mensagem ao utilizador com o código HTTP 404 informando que não foi possível encontrar o livro.
* **Index** - Esta função é cedida através de um “get request” feito à nossa API e tem como objetivo retornar todos os livros presentes na base de dados.
* **Create** – A função “create” recolhe o título do livro, o autor, o código de barras, o preço, o número de unidades em stock, o isbn e o state(?) e regista esta informação na base de dados.
* **Update** – Para esta função, é enviado o id do livro e opcionalmente o título do livro, autor, código de barras, preço e state e de seguida é atualizada a informação previamente guardada.
* **Delete** – Para a função “delete” é enviado novamente o id do livro e é executada uma procura na base de dados por um livro com o id igual ao id enviado para a nossa rota e no caso de se verificar a existência de um livro que cumpra estes requisitos este é então removido da base de dados. Caso contrário a API retornará uma mensagem ao utilizador com um código HTTP 404 informando-o da não existência de um livro com aquele id.

# 2.3 -Client Controller

Esta entidade contém todas as funções direcionadas aos clientes, ou seja, as funções “index”, “search”, “create”, “update” e “delete”.

* **Index** – Esta função é chamada através de um “get request” feito à nossa API e tem a finalidade de retornar todos os clientes presentes na base de dados.
* **Search** – Esta função irá efetuar uma pesquisa na base de dados através do id do cliente. Caso não exista nenhum cliente com um id válido, a API retornará uma menagem com o código de erro HTTP 404.
* **Create** – Na função “create” a nossa API recebe através do corpo do pedido a informação relativa ao “employee\_id”, nome do cliente, email, password e número de telemóvel e guarda essa informação na base de dados.
* **Update** – Esta função recebe o id assim como o nome do cliente. Deste modo, a informação presente na base de dados irá ser atualizada.
* **Delete** - Esta função verifica se existe na base de dados um cliente com o id igual ao id enviado para a nossa rota e no caso de se verificar a existência de um cliente que cumpra estes requisitos este é então removido da base de dados. Caso contrário a API retornará uma mensagem ao utilizador com um código HTTP 404 informando-o da não existência de um cliente com aquele id.

# 2.4 – Employees Controller

Esta componente armazena toda a informação alusiva aos empregados através das seguintes funções: “index”, “search”, “create”, “create\_admin”, “update” e “delete”.

* **Index** -Esta função é cedida através de um “get request” feito à nossa API e é responsável por retornar todos os empregados presentes na base de dados.
* **Search** – Esta função irá efetuar uma pesquisa na base de dados através do id. Caso o id não exista, uma mensagem com o código HTTP 404 irá ser retornada.
* **Create** – Para a função “create” a nossa API recebe através do corpo do pedido a informação relativa ao nome do empregado, email, password e número de telemóvel e regista essa informação na base de dados.
* **Create\_Admin** – À semelhança da função “create”, esta recebe a mesma informação, mas o funcionário fica com o estatuto de administrador.
* **Update** – Esta função recebe o id e o nome do funcionário, esta informação irá ser usada para atualizar os dados anteriormente guardados.
* **Delete** – Para a função “delete” é enviado novamente o id do funcionário e é executada uma procura base de dados por um funcionário com o id igual ao id enviado para a nossa rota e no caso de se verificar a existência de um funcionário que cumpra estes requisitos este é então removido da base de dados. Caso contrário a API retornará uma mensagem ao utilizador com um código HTTP 404 informando-o da não existência de um funcionário com aquele id.

# 2.5 – Login Controller

Esta componente contém a função “login” que permite ao utilizador efetuar o login na sua conta. O processo de login consiste numa pesquisa na base de dados pelo email e password enviados para o servidor, caso exista um utilizador cujas credenciais correspondam às inseridas a API retornará um “token” e as informações sobre o utilizador

# 2.6 – Purchase Controller

Esta entidade armazena toda a informação relativa às compras efetuadas, por meio das funções: “index”, “search”, “create”, “completePurchase”, “update” e “delete”.

* **Index** – Esta função é cedida através de um “get request” feito à nossa API e tem como objetivo retornar todas as compras presentes na base de dados
* **Search** – Para esta função, é efetuada uma pesquisa na base de dados através do id. Caso não exista nenhuma compra com esse id, é retornada uma mensagem com o código HTTP 404.
* **Create** – Na função “create”, a nossa API recebe através do corpo do pedido a informação “client\_id”, “employee\_id”, “nif” e “books” e regista toda esta informação na base de dados.
* **CompletePurchase** – Esta função efetua uma pesquisa na base de dados através do id, caso não exista nenhuma compra com esse id é retornada uma mensagem com o código HTTP 404. Uma vez encontrada a compra, são adicionados pontos ao cliente.
* **Update** – Para esta função é enviado o id e o preço da compra, esta informação irá ser usada para atualizar os dados previamente guardados.
* **Delete** - Esta função verifica se existe na base de dados uma compra com o id igual ao id enviado para a nossa rota e no caso de se verificar a existência de uma compra que cumpra estes requisitos este é então removida da base de dados. Caso contrário a API retornará uma mensagem ao utilizador com um código HTTP 404 informando-o da não existência de uma compra com aquele id.

# **Rotas**

# 3.1- Book Route

Para todas as rotas que contenham “books” no caminho são disponibilizadas todas as funções de listagem, criação e atualização dos elementos da entidade “books”.

# 3.2- Client Route

Para todas as rotas que contenham “client” no caminho são disponibilizadas todas as funções de listagem, criação e atualização dos elementos da entidade “client”.

# 3.3- Employee Route

Para todas as rotas que contenham “employee” no caminho são disponibilizadas todas as funções de listagem, criação e atualização dos elementos da entidade “employee”.

# 3.4- Purchase Route

Para todas as rotas que contenham “purchase” no caminho são disponibilizadas todas as funções de listagem, criação e atualização dos elementos da entidade “purchase”.

# **Frontend**

# 4.1 – Login/Registo

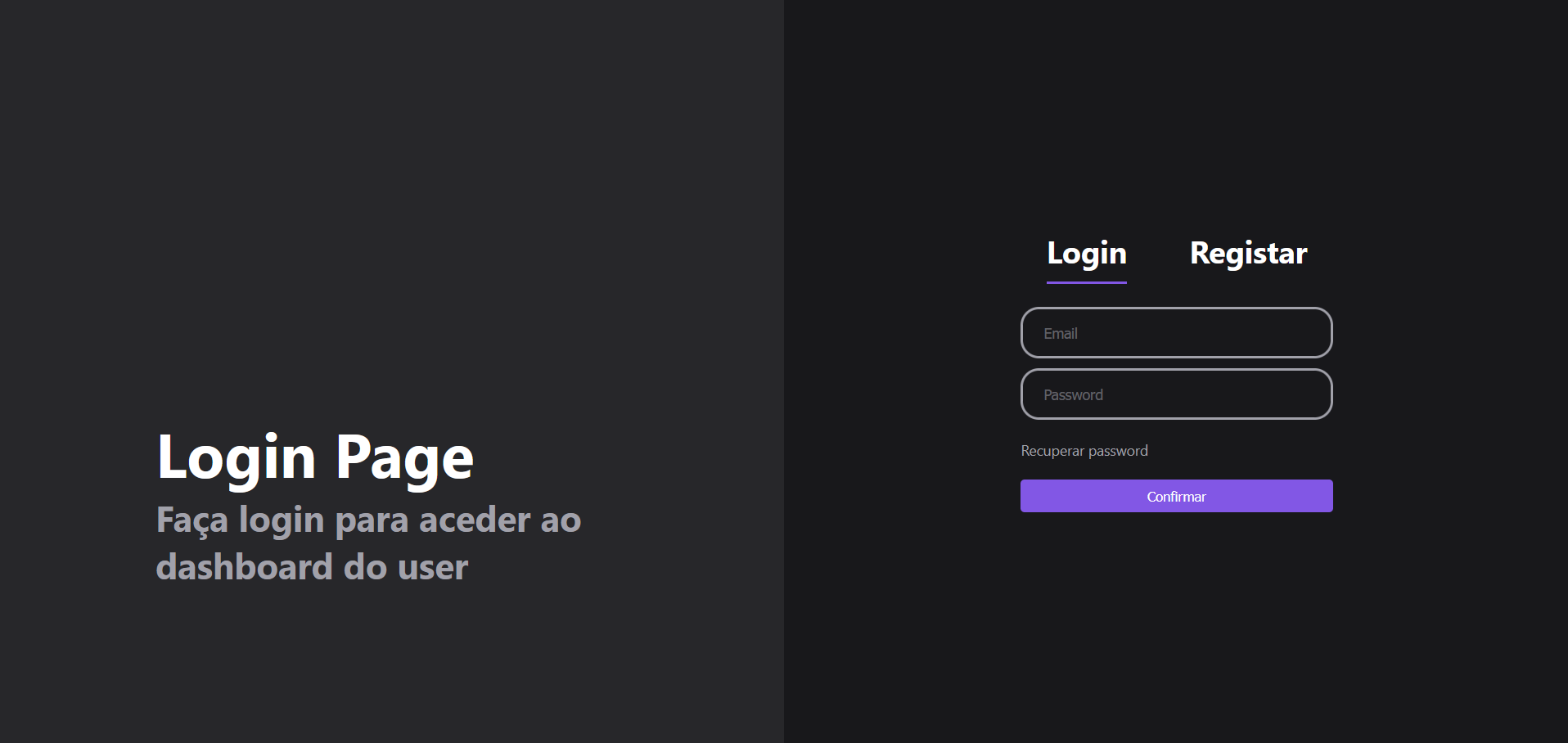
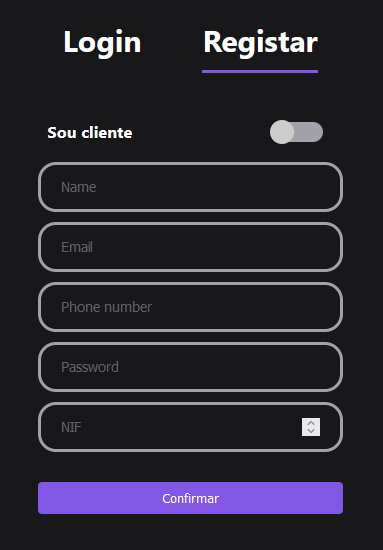


Figura 3



Quando iniciamos o programa é apresentada uma página de login, onde podemos entrar na nossa conta ou então registarmo-nos se não possuirmos uma. Na secção de registo, podemos escolher se somos clientes ou funcionários e após essa escolha é necessário introduzir o nome, email, número de telemóvel, *password* e o NIF para a conta.

Figura 4

# 4.2- Client Dash

Na página *ClientDash* é possível ver toda a informação relativa às fidelizações dos clientes, assim como os livros vendidos e comprados. Através do interface interativo disponibilizado pela biblioteca “Apex Chart”, conseguimos recolher essa informação com precisão uma vez que é possível isolar apenas um dia no gráfico, como é demonstrado na figura 5. É também possível ver o número de pontos adquiridos e gastos pelos clientes (figura 6).

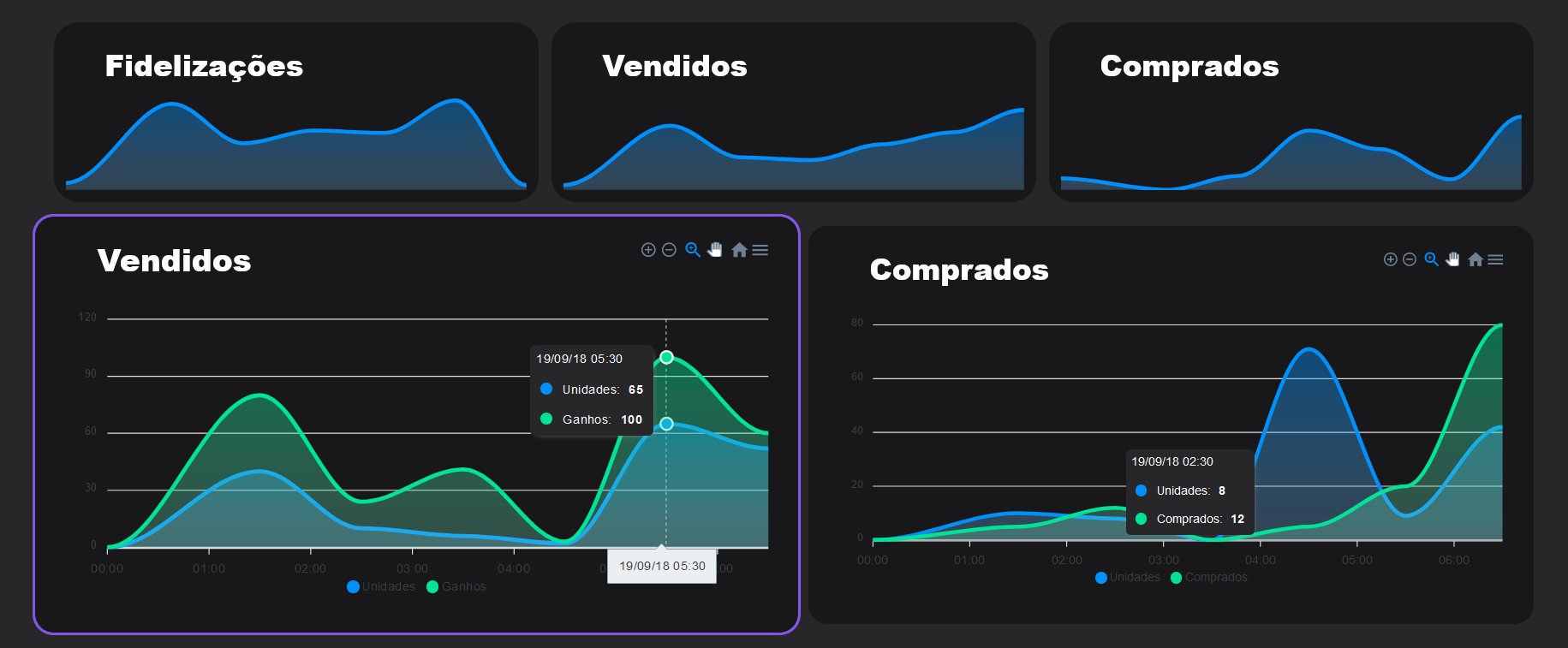


Figura 5

Uma imagem com texto, apresentação, eletrónica, captura de ecrã

Descrição gerada automaticamente

Figura 6

Através da biblioteca “Apex Chart”, o utilizador consegue aplicar zoom ao gráfico, assim como selecionar a área pretendida. É também possível exportar o gráfico em SVG, PNG e CSV.

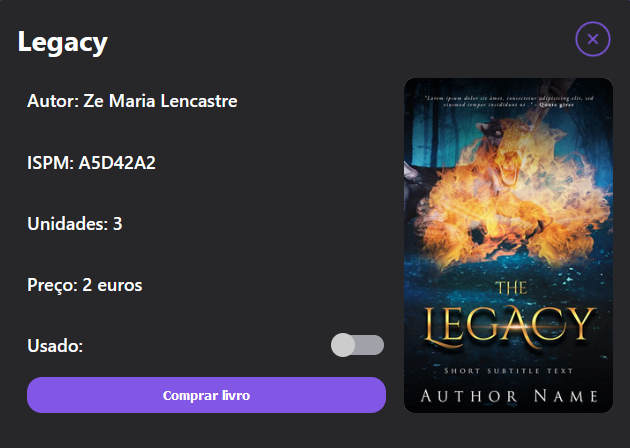
# 4.3- Market Place

Uma imagem com texto, cacho, vários

Descrição gerada automaticamente

Figura 7

Esta secção apresenta os livros disponíveis na loja, se o utilizador pretender efetuar uma certa compra apenas tem de carregar no livro pretendido e de seguida aparece a seguinte janela (figura 8) com os detalhes do livro.

 Uma imagem com texto, captura de ecrã, monitor

Descrição gerada automaticamente

Figura 8 Figura 9

Caso não consiga encontrar o livro pretendido, o utilizador pode auxiliar-se da secção de filtros para procurar um livro por determinados critérios (figura 9).

# 4.4- Selling

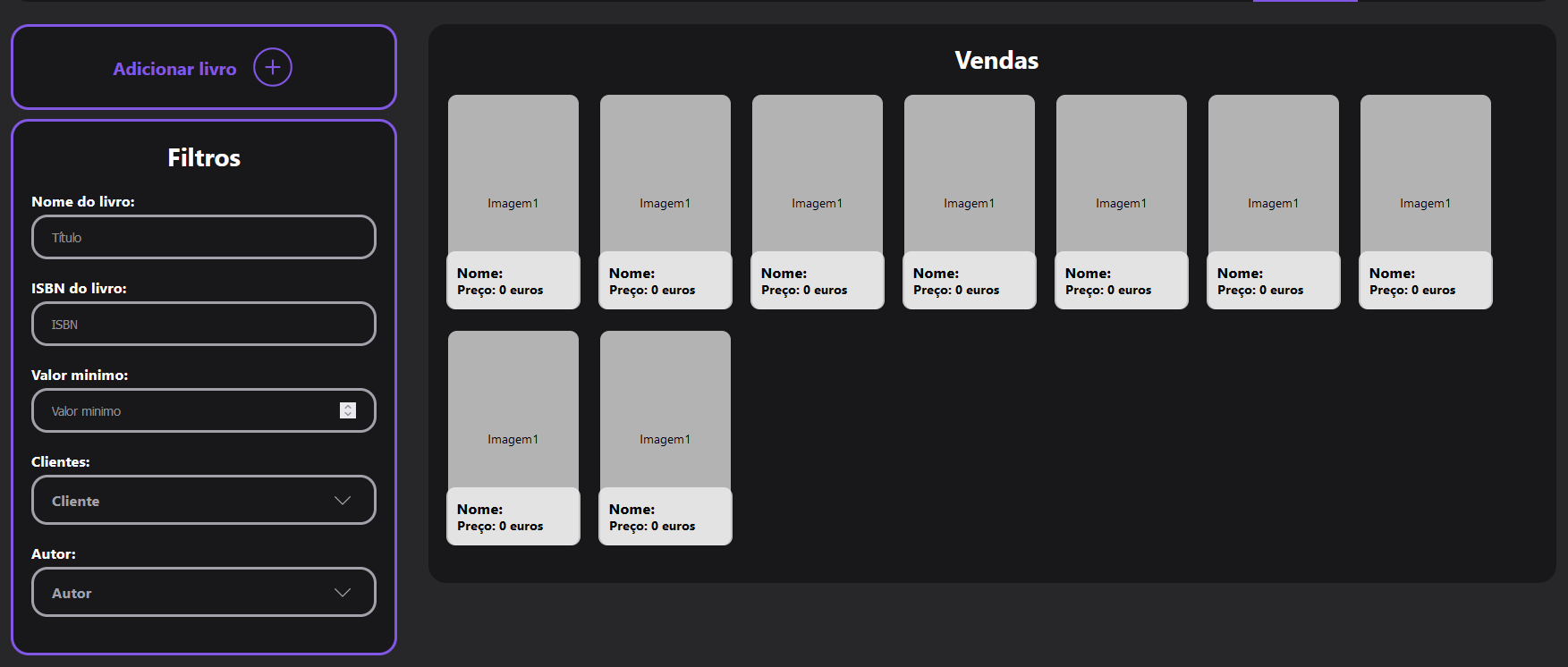
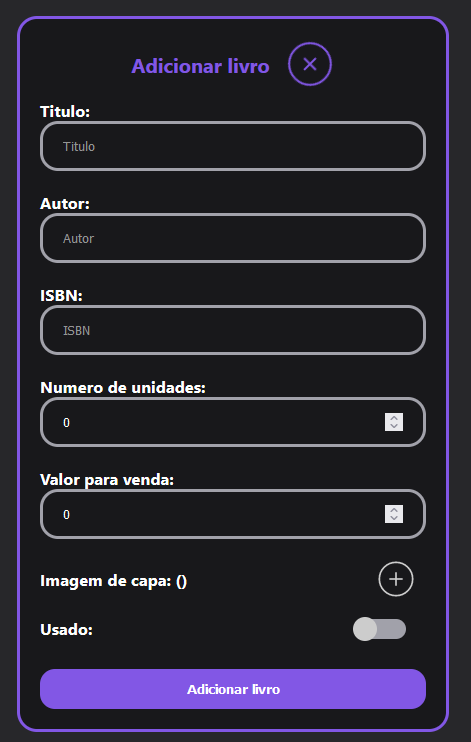


Figura 10

Esta página permite ao consultar as vendas efetuadas na loja. À semelhança do *Market Place,* é possível ver os livros individualmente. Deste modo, o utilizador pode verificar as características do livro. É também possível procurar um livro por um determinado critério através da secção de “filtros”, caso não encontre o livro na página “vendas”.



Através da página representada na figura 11, é possível adicionar livros à secção de vendas. Podemos verificar que é necessário adicionar as características de um determinado livro, assim como o número de unidades, o valor, a condição do livro e ainda importar uma imagem de capa.

Figura 11

# 4.5- Admin Dash

Esta página é semelhante à *ClientDash,* no entanto, possui funcionalidades extra que apenas um administrador pode usufruir. Como é demonstrado na figura 12, o administrador pode consultar o número de novos utilizadores bem como gerir os funcionários e os clientes. É possível apagar, submeter e, no caso de ser cliente, fidelizar utilizadores.

Uma imagem com texto, monitor, exibido, captura de ecrã

Descrição gerada automaticamente

Figura 12

# **Outros aspetos importantes**

# Testes à API

Adicionalmente, foram realizados teste à API utilizando as ferramentas *POSTMAN* e *NEWMAN* com o objetivo de validar todo o tipo de pedidos, tempos de resposta, códigos de erro e a presença de *content-types* nos *headers*.

Estes, podem ser executados automaticamente a todas as rotas utilizando um único comando, yarn test-api na route do projeto.

# Documentação

A fim de capacitar o avaliador, de uma mais fácil e ágil interação com os sistemas desenvolvidos foram criados ficheiros explicativos dos comandos a executar e considerações a ter aquando à avaliação do projeto (estes ficheiros denominam-se por README.md e encontram-se em todas as pastas principais do projeto sendo elas *backend, frontend* e a *root)*.

Complementarmente, foi desenvolvido com auxílio da biblioteca *swagger,* uma documentação referente à API desenvolvida na qual é possível visualizar todas as rotas existentes, os parâmetros constituintes das mesmas, os diferentes resultados emitidos como resposta no caso de situações de sucesso ou de entidades não encontradas e ainda testar o funcionamento destas rotas a partir de uma interface gráfica.

# Organização do trabalho através do GitHub

No teor da organização e codificação do presente trabalho, foi utilizado a plataforma GitHub. Esta proporcionou ao grupo a divisão de tarefas através da abertura de *issues* e *branches* para desenvolvimento. Consequentemente foi utilizada a capacidade de *pull requests* de maneira a fundir o código feito em *branches* secundárias para a *branch* principal.

O uso desta ferramenta possibilitou um método de trabalho mais organizado e consistente.

# Melhorias futuras

* Impossibilidade de remover administradores, funcionalidade a ser implementada no componente “admin tools”
* Processo de recuperação de password
* Autenticação de dois fatores

# **Conclusão**

O presente trabalho enriqueceu os nossos conhecimentos em HTML, JAVASCRIPT, NODE.JS e CSS, tendo criado bases para mais tarde aplicarmos estas linguagens e arquiteturas na área profissional.

Na nossa ótica, um dos pontos que mais enriquece o nosso trabalho foi o desenvolvimento do *frontend* utilizando *frameworks* como ANGULAR.JS, auxiliados por bibliotecas terceiras (*Apex Chart*) deram um aspeto visual bastante apelativo ao nosso programa. Por outro lado, a conceção deste trabalho usando novas tecnologias e abordagens revelou-se desafiante no sentido em que houve pouco tempo para se dedicar ao estudo aprofundado dos temas em questão.

Para concluir, este projeto revelou-se ímprobo porém, através dos conhecimentos adquiridos em aula e através de estudos autónomos conseguimos superar este desafio e entregar um trabalho que respeita todos os pontos pedidos no enunciado.

# **Bibliografia**

* Documentação disponibilizada no *Moodle* da disciplina
* <https://angular.io/docs>
* <https://nodejs.org/en/docs/>
* <https://learning.postman.com/docs/getting-started/introduction/>
* <https://learning.postman.com/docs/running-collections/using-newman-cli/command-line-integration-with-newman/>