Icon

Description automatically generated with medium confidence

Unidade Curricular

Programação em Ambiente Web

Milestone 1

Docentes:

Fábio André Souto da Silva

Marco Filipe Vieira Gomes

Alunos:

Rui Alexandre Borba Vitorino (8190479)

Gilberto Jorge Da Mota Gomes (8210227)

Licenciatura em Segurança Informática de Redes de Computadores

Índice

[Introdução 6](#_Toc102752871)

[Enquadramento 7](#_Toc102752872)

[Estrutura 8](#_Toc102752873)

[Explicação da estrutura 9](#_Toc102752874)

[Funcionalidades Propostas 10](#_Toc102752875)

[Milestone #1 10](#_Toc102752876)

[Registo de Funcionários 10](#_Toc102752877)

[Funcionário 10](#_Toc102752878)

[Gestão de clientes 11](#_Toc102752879)

[Gestão de livros 11](#_Toc102752880)

[Histórico de Operações 12](#_Toc102752881)

[Pontos 12](#_Toc102752882)

[Descontos 13](#_Toc102752883)

[Outras implementações 14](#_Toc102752884)

[Imagens 14](#_Toc102752885)

[Paginação 14](#_Toc102752886)

[Sessões 14](#_Toc102752887)

[Password Hashing 15](#_Toc102752888)

[HTTPs 16](#_Toc102752889)

[Helmet 16](#_Toc102752890)

[Pesquisa 16](#_Toc102752891)

[Validações 16](#_Toc102752892)

[Brute-Force 17](#_Toc102752893)

[Conclusões 18](#_Toc102752894)

[Referências 19](#_Toc102752895)

Índice Figuras

# Introdução

A “My Library” é uma empresa do ramo bibliotecário que pretende acompanhar as tendências de negócio, e como tal decidiu avançar com um projeto que permita tirar partido do mundo digital.

Para isso, foi decidido implementar uma página web dinâmica, com gestão de back-office.

O objetivo é chegar ao máximo de clientes possível e fidelizá-los, para isso a empresa criou um plano de recompensa contínua através de um sistema de pontos.

O projeto foi dividido em duas partes, front-end e back-end, começando o desenvolvimento do mesmo por esta última com a implementação do portal do back-office.

No back-office será efetuada a gestão dos clientes, do stock de livros, vendas e também dos funcionários da empresa.

A equipa de desenvolvimento optou por dividir o back-office em dois portais, o de administração e o dos funcionários, esta decisão será abordada aquando da explicação da estrutura.

Para assegurar a permanência da informação manipulada pela aplicação foi usado o MongoDB como sistema de gestão de base de dados.

# Enquadramento

No âmbito do desenvolvimento do trabalho prático da unidade curricular de Programação em Ambiente Web, onde o objetivo passa pela criação de uma aplicação web que irá simular uma livraria online.

Esta aplicação irá ter duas componentes, nomeadamente, *front-end* e *back-end*, tendo sido a componente do *back-end* e respetivo *back-office* desenvolvida nesta primeira parte do trabalho ou seja no “Milestone #1”.

A componente *front-end* , portal do cliente, requisito do “Milestone #2” foi parcialmente implementada, apenas para testar questões de administração e gestão de objetos de negócio presentes na base de dados.

# Estrutura

De forma a seguir as boas práticas de desenvolvimento para aplicações que tenham como base a *framework express*, bem como o *template engine* utilizado, EJS, a seguinte estrutura foi concebida.

* app.js
* models/
* routes/
* controllers/
* public/
  + imagens/
  + javascripts/
  + stylesheets/
* views/
  + backoffice/
* /admin
* /employee
* backofficeIndex
* backofficeLogin
  + client/
  + index/
  + error/

Existem ainda outras diretorias, tais como:

* bin/
* certs/
* node\_modules/

## Explicação da estrutura

A estrutura do back-office, como já foi mencionado, foi dividida em dois portais, ou seja, o portal de administração, back-office/admin e o dos funcionários back-office/employee.

Apesar de haver replicação de código, optámos por esta estrutura, por acharmos que simplifica a implementação de todo o processo, pois assim passámos a ter vistas, rotas e controladores separados, simplificando todos os processos à posteriori uma vez que a informação disponibilizada nas vistas será sempre adequada ao tipo de utilizador. Apesar de sabermos que podia ser implementado tudo junto, tal como está na página de index do back-office em que a vista apresentada é em função do tipo de utilizador.

A restante estrutura segue os padrões de desenvolvimento da *framework* em uso, fazendo uso das funcionalidades disponibilizadas pela mesma, e pelo *template engine* EJS, tornando algumas páginas estáticas em páginas dinâmicas, consoante a informação recebida e ou o utilizador autenticado.

# Funcionalidades Propostas

## Milestone #1

Todas as funcionalidades propostas para o Milestone #1 foram implementadas.

Neste Milestone já foram implementadas algumas funcionalidades não pedidas, tais como, inserção de imagens, paginação, função de procura, validações de input nos formulários.

Uma vez que os elementos do grupo são alunos da Licenciatura em Segurança Informática em Redes de Computadores, foram implementadas funcionalidades ao nível da segurança da aplicação, tais como, passwords encriptadas, sessões com token’s encriptados, mecanismos de autenticação e autorização via URL e API, comunicações via HTTPs, utilização do midleware Helmet, mecanismo à prova de brute-force com o módulo “rate-limiter-flexible” e o “.env” para ocultar a informação de configuração da ligação à base de dados, algoritmo de hashing das passwords e chave utilizada na assinatura dos token’s.

### Registo de Funcionários

O registo de novos funcionários só pode ser efetuado por um utilizador com privilégios de administração, através do portal de *back-office* .

#### Administrador

Sendo o administrador uma conta com acesso a todas as funcionalidades da aplicação, é permitido a criação, remoção, visualização, atualização e pesquisa de funcionários.

### Funcionário

O funcionário (employee), após a sua autenticação, tem privilégios limitados, tais como, no seu perfil apenas pode alterar os seus dados pessoais e não consegue ver informações dos outros funcionários da empresa.

### Gestão de clientes

#### Administrador

Nesta funcionalidade, ambos têm permissões de criação, remoção, visualização, atualização e pesquisa de clientes, existindo apenas a particularidade de o administrador, conseguir indicar o número de pontos atribuídos durante a criação de um novo cliente e atualização de um já existente.

Durante a criação de um novo cliente, serão atribuídos automaticamente os pontos para este campo do formulário definidos na base de dados, no entanto o administrador tem a possibilidade de alterar este valor, tanto na criação como na gestão de clientes.

#### Funcionário

Como referido, os funcionários têm todas as capacidades de gestão de clientes, exceto a atribuição e atualização dos pontos de determinado cliente, **e**, este também, por motivos de segurança, é impossibilitado a alteração da password de determinado cliente.

Nota: No Milestone #2 ao criar um cliente novo a password será gerada automaticamente e enviada por email, será retirado o acesso a esta informação tanto aos administradores como aos funcionários.

### Gestão de livros

Nesta funcionalidade, ambos têm permissões de criação, remoção, visualização, atualização e pesquisa de livros.

### Histórico de Operações

Neste Milestone #1 está implementada a funcionalidade de venda e tanto os administradores como os empregados poder criar e procurar vendas.

## Pontos

Esta funcionalidade está disponível apenas para os administradores, que pode alterar todos os valores destes campos.

* Point To Money: Quando é efetuada a venda, caso o cliente tenha pontos disponíveis e opte por os usar, os pontos do mesmo serão convertidos em €uros e descontados da venda.
* Percentage Per Purchase: Quando é efetuada a venda, o total da mesma é multiplicado pelo valor deste campo, obtendo assim os pontos que serão atribuídos ao cliente.
* Shipping Points: Este é o número de pontos que um cliente tem de acumular para obter um envio gratuito.
* New Client Register: Este é o número de pontos atribuído automaticamente na criação de um novo utilizador.
* Recomendation Client: Este é o número de pontos atribuído automaticamente a um cliente que venha com recomendação.
* Points Per Promotion: Este é o número de pontos atribuído automaticamente numa venda, caso esteja a decorrer ativa uma promoção.

## Descontos

Esta funcionalidade está disponível apenas para os administradores, que pode alterar todos os valores destes campos.

* Discount per sale: Desconto atribuído automaticamente a todas as vendas.
* Discount promotion: Desconto atribuído automaticamente a todas as vendas se houver uma promoção ativa.
* Infantil: Desconto atribuído automaticamente a todas as vendas em função da idade do cliente.
* Juvenil: Desconto atribuído automaticamente a todas as vendas em função da idade do cliente.
* Adulto: Desconto atribuído automaticamente a todas as vendas em função da idade do cliente.
* Sénior: Desconto atribuído automaticamente a todas as vendas em função da idade do cliente.

## Outras implementações

### Imagens

Fazendo uso do *middleware* multer, é possível obter as informações passadas após o envio de um formulário HTML, bem como ficheiros, neste caso imagens, que irão servir de capa de um livro. (*Expressjs/Multer: Node.Js Middleware for Handling `multipart/Form-Data`.*, n.d.)

### Paginação

Foi implementada a funcionalidade de paginação, tornando as páginas mais dinâmicas e visualmente mais simples, realizando a separação da informação recolhida da base de dados.

Esta paginação, continua a ter efeito mesmo durante a pesquisa de determinado objeto. (Yasas Sandeepa, n.d.)

### Sessões

#### Tokens JWT

Fez-se uso de Tokens JWT de modo a implementar mecanismos de autenticação e respetiva autorização aos pontos front-end e back-end. (*Auth0/Node-Jsonwebtoken: JsonWebToken Implementation for Node.Js*, n.d.)

Estando os tokens JWT interligados com o módulo de sessões do Express, foi também importante configurar propriedades de segurança, tais como atribuição do nome da sessão como um conjunto de caracteres aleatório. O mesmo processo de aleatoriedade de geração de chaves de assinatura das cookies é utilizado, efetivamente existindo rotação de chaves de assinatura em cada reinício do serviço.

#### Cookies

Em conjunto com os Tokens JWT, são utilizadas cookies de forma a criar persistência de sessão, existindo um tempo de vida de uma (1) hora. (*Express Cookie-Parser Middleware*, n.d.)

Algumas propriedades de segurança foram configuradas, nomeadamente:

* SameSite: Strict *Restringe o contexto do uso da cookie*
* Secure: true *Restringe o uso da cookie apenas para canais seguros (HTTPs)*
* httpOnly: true *Permite apenas o acesso à cookie via Back-End e não usando JavaScript*

#### Logout

É ainda possível realizar *logout*, limpando as cookies e a “matando” a sessão.

### Password Hashing

Uma característica importante e que infelizmente é muitas vezes descartada por motivos de simplicidade, é a segurança a nível de armazenamento de informações, principalmente no que toca a *passwords*, sendo estas muitas vezes, armazenadas em *clear-text*.

Decidiu-se então realizar o *hashing* da password do administrador, funcionários e clientes, não existindo qualquer informação do tipo sensível em *clear-text* na base de dados.

Durante o processo de *hashing*, boas políticas foram adotadas, tais como, uso de um método de *hashing* seguro (SHA-512, este que se encontra “escondido” no ficheiro de configuração dotenv, estando oculto no código-fonte); quantidade segura de iterações de *hashing* (1000); uso e rotação de um *salt* para cada utilizador. (*Node.Js - Mongoose Password Hashing - Stack Overflow*, n.d.)

### HTTPs

Seguindo o exemplo do mundo real e o padrão de desenvolvimento de páginas web, foi implementada a funcionalidade de HTTPS, para que as ligações sejam efetuadas com segurança utilizando certificados *self-signed*. (*Node.Js - Enabling HTTPS on Express.Js - Stack Overflow*, n.d.)

### Helmet

Assegurando algumas configurações de segurança, nomeadamente a introdução de determinados parâmetros nos *headers* no protocolo HTTP/HTTPS, o Helmet permite facilmente melhorar a segurança da nossa aplicação web contra diversos ataques. (*Helmetjs/Helmet: Help Secure Express Apps with Various HTTP Headers*, n.d.)

### Pesquisa

Foi implementada a funcionalidade de pesquisa, que permite ao utilizador pesquisar certos objetos de negócio em função de determinados parâmetros, como por exemplo, *username*, email, ISBN.

As pesquisas são efetuadas pelo sistema de gestão de base de dados MongoDB, utilizando expressões regulares e object\_ids filtrando os documentos encontrados. (*Query Documents — MongoDB Manual*, n.d.)

### Validações

Presente nos ficheiros EJS, estão simples, mas eficazes validações perante os dados a serem introduzidos, tais como limitações do tamanho de caracteres, padrões e ou tipo de dados.

Na parte de back-end são efetuadas verificações relativas às informações recebidas e as existentes na base de dados, tais como a verificação de informação já existente, tal como emails, usernames de funcionários e de clientes ou ainda ISBNs relativamente aos livros). (*HTML Input Attributes*, n.d.)

### Brute-Force

Para proteger a aplicação de ataques de brute-force foi implementada esta funcionalidade recorrendo ao módulo “rate-limiter-flexible”. Ou seja, definimos que sempre que sejam efetuados dez pedidos em menos de dez segundos, sejam eles de GET ou POST, o IP de onde estiver a ser lançado esse “ataque” será automaticamente bloqueado durante cinco minutos. (*Animir/Node-Rate-Limiter-Flexible: Node.Js Rate Limit Requests by Key with Atomic Increments in Single Process or Distributed Environment.*, n.d.)

# Conclusões

# Referências

*animir/node-rate-limiter-flexible: Node.js rate limit requests by key with atomic increments in single process or distributed environment.* (n.d.). Retrieved May 6, 2022, from https://github.com/animir/node-rate-limiter-flexible

*auth0/node-jsonwebtoken: JsonWebToken implementation for node.js*. (n.d.). Retrieved May 6, 2022, from https://github.com/auth0/node-jsonwebtoken

*Express cookie-parser middleware*. (n.d.). Retrieved May 6, 2022, from http://expressjs.com/en/resources/middleware/cookie-parser.html

*expressjs/multer: Node.js middleware for handling `multipart/form-data`.* (n.d.). Retrieved May 6, 2022, from https://github.com/expressjs/multer

*helmetjs/helmet: Help secure Express apps with various HTTP headers*. (n.d.). Retrieved May 6, 2022, from https://github.com/helmetjs/helmet

*HTML Input Attributes*. (n.d.). Retrieved May 6, 2022, from https://www.w3schools.com/html/html\_form\_attributes.asp

*node.js - Enabling HTTPS on express.js - Stack Overflow*. (n.d.). Retrieved May 6, 2022, from https://stackoverflow.com/questions/11744975/enabling-https-on-express-js

*node.js - Mongoose password hashing - Stack Overflow*. (n.d.). Retrieved May 6, 2022, from https://stackoverflow.com/questions/14588032/mongoose-password-hashing

*Query Documents — MongoDB Manual*. (n.d.). Retrieved May 6, 2022, from https://www.mongodb.com/docs/manual/tutorial/query-documents/

Yasas Sandeepa. (n.d.). *Build a Paginated API Using Node.js, Express, and MongoD*. Retrieved May 6, 2022, from https://betterprogramming.pub/build-a-paginated-api-using-node-js-express-and-mongodb-227ed5dc2b4b