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

Contents

[Introdução 6](#_Toc102730093)

[Enquadramento 7](#_Toc102730094)

[Estrutura 8](#_Toc102730095)

[Explicação da estrutura 9](#_Toc102730096)

[Funcionalidades Propostas 10](#_Toc102730097)

[Milestone1 10](#_Toc102730098)

[Registo de Funcionários 10](#_Toc102730099)

[Funcionário 10](#_Toc102730100)

[Gestão de clientes 11](#_Toc102730101)

[Gestão de livros 12](#_Toc102730102)

[Histórico de operações 12](#_Toc102730103)

[Pontos 12](#_Toc102730104)

[Descontos 12](#_Toc102730105)

[Outras implementações 13](#_Toc102730106)

[Imagens 13](#_Toc102730107)

[Paginação 13](#_Toc102730108)

[Sessões 13](#_Toc102730109)

[HTTPs 14](#_Toc102730110)

[Helmet 14](#_Toc102730111)

[Pesquisa 14](#_Toc102730112)

[Validações 14](#_Toc102730113)

[Conclusões 15](#_Toc102730114)

[Referências 15](#_Toc102730115)

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.

O sistema de gestão de base de dados utilizado é o MongoDB.

# 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/
  + cliente/
  + index/
  + error/

Existem ainda outras diretorias, tais como:

* bin/
* certs/
* node\_modules/

## Explicação da estrutura

Existe uma particularidade da estrutura por nós utilizada que merece esclarecimento, nomeadamente, a existência de replicação de código nos diversos controladores (controllers backoffice, admin e employee), bem como nas vistas.

O processo de estruturação ocorreu durante a divisão de tarefas, ou seja, certo elemento do grupo iria desenvolver o backend/backoffice admin e o outro iria desenvolver o backend/backoffice employee, não existindo assim independência de determinada funcionalidade, comportamento e/ou implementação para o funcionamento geral da aplicação, de certa forma, sendo *quase* possível a comparação a Interfaces Java.

Obviamente esta decisão não é a ideal em termos de produção, ou na otimização e reutilização de código, mas após a avaliação da situação, a restruturação implicaria uma grande quantidade de trabalho, e requeria também alterações no workflow de desenvolvimento, levando-nos a manter a estrutura presente.

A restante estrutura segue os padrões de desenvolvimento da *framework* em uso, fazendo uso das funcionalidades disponíveis 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

## Milestone1

Foram implementadas todas as funcionalidades propostas, existindo pequenos detalhes e comportamentos que posteriormente, no milestone2, irão sofrer alterações, devido aí, já estar presente a componente de negócio. No entanto estas funcionalidades “incompletas” já têm a lógica e respetiva implementação base desenvolvida, restando apenas “interligar” os componentes.

Sendo o grupo pertencente à Licenciatura de Segurança, Redes e Sistemas Informáticos, outras funcionalidades relativas à segurança da aplicação foram também implementadas.

### Registo de Funcionários

Através do portal de *backoffice* e após a autenticação do administrador ou funcionário, o conteúdo e consequente disponibilidade de funcionalidades são apresentadas dinamicamente.

#### Administrador

Sendo o administrador uma conta com total permissão, é permitido a criação, remoção, visualização, atualização e pesquisa de funcionários.

### Funcionário

O funcionário, após a sua autenticação, irá apenas ter acesso ao seu perfil, e capacidades limitadas de atualização das suas informações, não sendo permitida a criação, atualização, visualização e pesquisa de outros funcionários.

### Gestão de clientes

#### Administrador

Nesta funcionalidade, ambos têm total capacidade 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, na ausência de um valor de pontos a atribuir ao novo cliente, 10 pontos são atribuídos automaticamente, independentemente a operação estar a ser realizada pelo administrador ou funcionário.

#### Funcionário

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

### Gestão de livros

#### Administrador

* + Gestao de livros **usados** a aguardar milestone2, implementação de portal de cliente, embora já esteja pensada a arquitetura de backend/backoffice
  + Criar, atualizar, remover em ambos

#### Funcionário

### Histórico de operações

#### Administrador

#### Funcionário

## Pontos

## Descontos

## 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 e presente na 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 certa 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 sessão, realizando o processo de de-autenticar o utilizador.

### 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 tipo de password e/ou informação 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, uma ligação segura usando HTTPs foi implementada, 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, 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

De forma a implementar a funcionalidade de pesquisa, permitindo ao utilizador pesquisar certos objetos de negócio por determinados parâmetros, como por exemplo, *username*, sendo a pesquisa efetuada e executada pelo sistema de gestão de base de dados MongoDB, utilizando expressões regulares e 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 verificações relativas às informações recebidas e as existentes na base de dados, são efetuadas, tais como a verificação de existência de informação replicada (tal como emails e/ou usernames relativos a funcionários e clientes, e ISBNs relativamente aos livros). (*HTML Input Attributes*, n.d.)

# Conclusões

# Referências