ESCOLA
SUPERIOR
DE TECNOLOGIA
E GESTÃO

Programação em Ambiente Web

Licenciatura em Segurança Informática em Redes de Computadores

2020/2021

Christopher Meder – 8170022

Index

Índice de Figura

Introdução

Este trabalho tem como objetivo ensinar a criar aplicações Web com um modelo de Frontend e Backend, ou seja, cliente e servidor. Este projeto foi realizado com o intuito para criar uma plataforma web para ajudar a divulgar locais de interesse numa região. Toda a informação ira ser guardada numa base de dados.

MVC

O Backend foi dividido implementando o seguinte modelo, MVC (Model, View, Controller), ou seja:

- Model, será o formato onde a informação ira ser guardada para enviar para a nossa base de dados, neste caso usamos o MongoDB como base de dados e foi atribuído o nome de PAW.

Na figura seguinte temos um exemplo do formato que foi utilizado:

```
var mongoose = require("mongoose");

var UserSchema = new mongoose.Schema({
    fname: {
        type: String,
        required: true
    },
    lname: {
        type: String,
        required: true
    },
    gender: {
        type: String,
        enum: ["M", "F"],
        default: "M",
        required: true,
    },
    password: {
        type: String,
        required: true
    },
    email: {
        type: String,
        required: true
    },
    permission: {
        type: Number,
        default: 2
    }
});

mongoose.model('User', UserSchema);

module.exports = mongoose.model("users", UserSchema);
```

Figura 1 - Estrutura da BD

A base de dados ira ser dividida em 3 partes: admins, users e locals, para ser mais fácil de armazenar os dados.

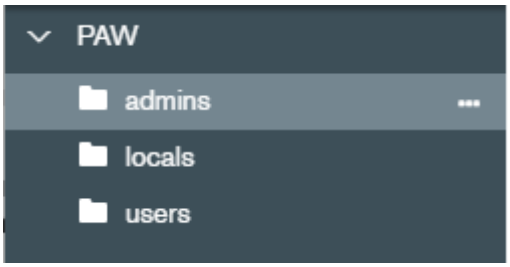

Figura 2 - Base de Dados

- View, tendo em conta que utilizamos o nos HTML no Frontend, não iremos utilizar isto

```
<div class="card" *ngIf="data">
    <h1 mat-dialog-title class="text-center">Local Details</h1>
    <mat-dialog-content>
        <label for="form-name">Name</label>
        <input placeholder="Nome do Local..." [(ngModel)]="data.name" type="text" disabled>
        <td></td>
        <label for="form-local">Local</label>
        <input placeholder="Endreco..." [(ngModel)]="data.local" type="text" disabled>
        <td></td>
        <label for="form-category">Categoria</label>
        <input placeholder="Categoria..." [(ngModel)]="data.category" type="text" disabled>
        <td></td>
        <label for="form-desciption">Descricao</label>
        <input placeholder="Descricao..." [(ngModel)]="data.desciption" type="text" disabled>
        <td></td>
        <label for="form-likes">Likes</label>
        <input  [(ngModel)]="data.likes" type="number">
        <td></td>
        <label for="form-dislikes">Dislikes</label>
        <input [(ngModel)]="data.dislikes" type="number">
        <td></td>
        <mat-dialog-actions>
            <button mat-raised-button [mat-dialog-close]="false"> Fechar</button>
        </mat-dialog-actions>
    </mat-dialog-content>
```

Figura 3 - HTML

- Controller, aqui estão todas as funções utilizadas para os nossos admin, users e locals tal como registar e login.

```
//Criar Admin
adminController.createAdmin = function (req, res) {
    var admin = new Admin(req.body);
    admin.password = bcrypt.hashSync(req.body.password, 8);

    //Find Admin by Email
    Admin.findOne({ email: req.body.email }).exec((err, dbAdmin) => {
        if (err) {
            return res.status(500).send('Error on the server.');
        } else {
            if (dbAdmin === null) {
                if (req.body.permission == 1) {
                    admin.save((err) => {
                        if (err) {
                            return res.status(500).send('Error on the server.');
                        } else {
                            res.json(req.body);
                        }
                    })
                }
            } else {
                console.log("Admin Already Exist");
                res.json({});
            }
        }
    })
}
```

Figura 4 - Controller

A password que será enviada para a nossa base de dados, ira ser encriptada usando bcrypt.hashSync. Para o login é efetuado uma comparação dos hash através dos controladores de autenticação no qual ira verificar se é um utilizador ou um administrador a efetuar o login ou mesmo se existe esse utilizador.

```
email: admin@gmail.com
password: "$2b$08$K/2yxhwZ/oQzLY2sdQviXOGM/NnZHV12Wy7bQhH2C0QHrvMyUdcQO"
  v: 0
```

Figura 5 - Password Hashed

Swagger

Swagger serve para documentar a api criada assim podemos invocar diretamente a partir dele os serviços REST obtendo uma resposta.

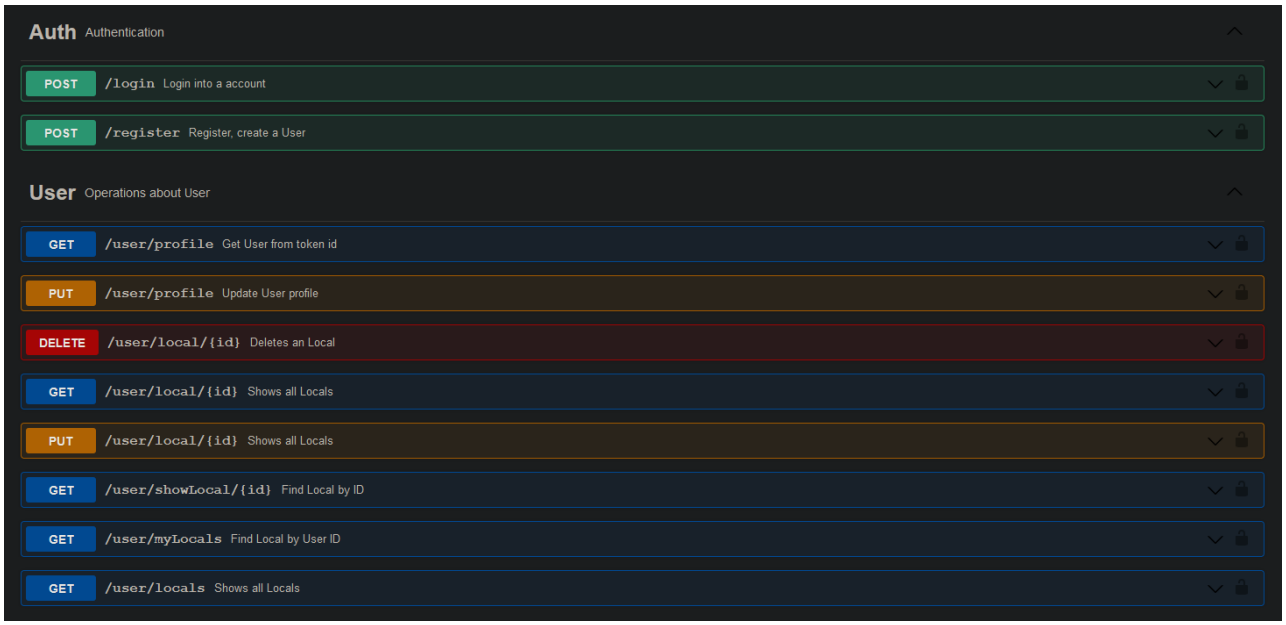

Figura 6 - Swagger 1

Figura 7 - Swagger 2

Angular

O angular apenas ira servir como o site que ira fazer a interação do cliente com o servidor, usando serviços e pedidos para atualizar os dados. Através dos serviços no angular efetuamos a ligação do frontend com o backend

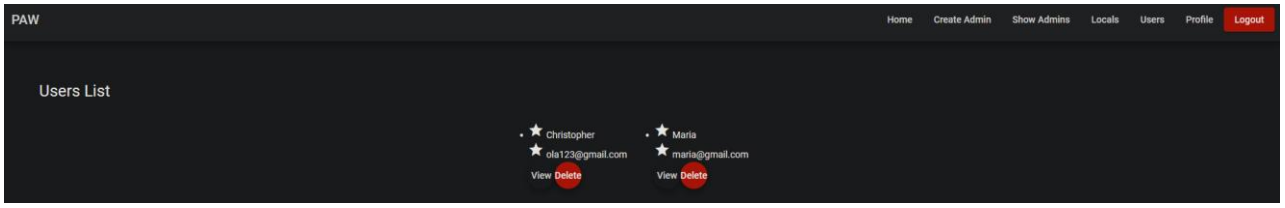

Figura 8 - Angular

Conclusão

Houve alguma dificuldade neste trabalho sendo que o maior problema estava no angular na parte dos “componentes”, pois havia bastante confusão a fazer as ligações com os serviços porque era demasiada informação ao mesmo tempo que não sabia onde é que tinha de fazer ligações e em que sitio a tinha de chamar.

Também não foi possível meter os likes/dislikes a funcionar nem os comentários devido a falta de tempo que tinha para tal.

Tendo em conta isso na minha opinião a parte do BackEnd foi a mais acessível, sendo que fazer as rotas não requer muita complicação.