Projeto Final de Desenvolviemto Web

Movie Night Planner

Vasco Neves Machado de Matos João
30005819@students.ual.pt

I. Introdução

A. *Propósito do documento*

Este Documento tem como propósito a realização do relatório do Projeto do turno diurno de Engenharia Informática do 3º ano de 2023/2024 no âmbito da Unidade Curricular Desenvolvimento Web da Universidade Autónoma de Lisboa. Sendo este, um Trabalho de desenvolvimento web Full-Stack com o tema "Movie Night Planner". O mesmo é destinado à avaliação por parte do docente da Unidade Curricular.

B. *Contexto do Projeto*

O Projeto de Desenvolvimento Web consiste na realização de uma aplicação web Full-Stack com integração de uma API externa.

Já o tema deste projeto implica que a aplicação tenha a capacidade de um utilizador escolher filmes e marcar dias e horas para ver esses filmes, seja com amigos ou sozinho.

C. *Visão geral do Documento*

O documento atual tem o objetivo de descrever com claridade o funcionamento deste projeto. Para tal, este documento encontra-se organizado no formato IEEE. Já em relação ao conteúdo este encontra-se, para além desta introdução, dividido nos capítulos Requisitos, Projeto e User Info.

Requisitos, tal como o nome indica, expõe os requisitos necessários ao desenvolvimento e execução da aplicação.

O capítulo Projeto representa a parte técnica do trabalho envolvido com este projeto, ou seja, a arquitetura deste.

Por fim, User Info tem como propósito fornecer os dados que foram usados no projeto de forma a este poder ser replicado nas mesmas condições.

II. Requisitos

A. *Requisitos de Sistema*

Os requisitos do sistema desenvolvido para este projeto podem ser classificados como fazendo parte de 3 aspetos:

1) *Projeto*

- O sistema tem de usar uma base de dados com dicionário de dados.
- Tem de haver CRUD entre o servidor web (Backend) e a base de dados.
- Tem de haver REST entre o browser (Frontend) e o servidor web (Backend).
- Autenticação e autorização são essenciais.
- É necessário o uso de uma API externa.
- O site tem de ser responsivo e ter boa acessibilidade.

2) *Tema*

- Registar utilizadores.
- Ter CRUD no planeamento de movie nights.
- Integração com a API de uma Base de Dados de filmes para se poder escolher o filme.
- Deve ter uma RESTful API para ver os dados dos filmes.

3) *Operacionais*

a) *Front-end:*

Para além do html, JavaScript e CSS já presente na grande maioria das páginas web este também usa:

- Bootstrap 5 incluindo a sua extensão para JavaScript
- Fontawesome

b) *Back-end:*

Já no Back-end o node.js contém as dependências a seguir mencionadas:

- Bcrypt 5.1.1
- Cookie-parser 1.4.6
- Cors 2.8.5
- Dotenv 16.3.1
- Express 4.18.2
- https 1.0.0
- jsonwebtoken 9.0.2
- node-fetch 2.6.1
- pg 8.11.3

c) *Base de Dados:*

Já a base de dados é usa o PostgreSQL 16.

B. Funções do Sistema

1) Autenticação

- Um utilizador de ser capaz de se registar e fazer login com um email e palavra-passe.
- Depois de autenticado um utilizador deve conseguir manter a sessão. Podendo voltar a página mantendo-se autenticado. Isto, mesmo que saia da página desde que não tenha fechado a sessão ou o navegador.

2) Utilizador

- Um utilizador deve ter acesso a uma página de utilizador.
- A página de utilizador deve conter os seus dados e a sua lista de amigos.
- A lista de amigos deverá ter uma opção para adicionar ou remover amigos com base no id desse amigo.

3) Filmes

- Um utilizador deve ter acesso aos filmes mesmo que não tenha feito login.
- Os filmes devem ter o seu nome e uma imagem tipo poster (se disponível).
- Estes devem poder ser pesquisados por popularidade, ranking, novidades e nome.

4) Noites

- Cada noite de filmes deve ter e um e só um filme associado.
- Estas também deveram ter uma data com horas e minutos e uma breve descrição.
- Para além disso também devem poder ser editadas e removidas.

C. Restrições gerais

Para este projeto foram detetadas as seguintes restrições:

- A base de dados não deverá guardar os dados dos filmes contendo apenas uma referência ao id dos mesmos para poderem ser referenciados mais tarde.
- A página web não pode ser contacto direto com a base de dados.
- Os dados como chaves de criptografia, detalhes de autenticação de base dos dados e a chave de API da base de dados de filmes devem estar todos num ficheiro dotenv.

D. Considerações de design

Em relação as considerações de design que foram feitas ao longo do projeto, estas encontram-se principalmente na divisão dos seus componentes. Seja esta tanto física como logica.

Isto é, as diferentes funcionalidades do sistema estão divididos tanto em ficheiros diferentes como usam rotas que levam a routers diferentes.

Para além disso, também foi considerada a segurança da informação no sentido em que os servidores estão a correr em protocolo https em vez de http.

III. PROJETO

A. Arquitetura do sistema

1) Front-end

O Front-end deste projeto é representado por uma única página web que faz uso de AJAX para atualizar a página com ajuda de JavaScript e CSS.

Este é composto por 1 ficheiro html “index.js”, 1 ficheiro CSS “styles.css” e 5 ficheiros js:

- script.js

Este ficheiro é responsável pelas mudanças na página que não dependem do servidor como fazer aparecer o modal de login e de registo e também a mudança da barrade navegação de transparente para escuro.

- user.js

Já este tem como função tudo o que é gerir informação do utilizador. Seja popular a página de utilizador, ir buscar as suas informações ou registar um novo utilizador.

- auth.js

Por sua vez auth.js define como um utilizador deve se autenticar. Seja por login com credenciais ou com um token.

- movies.js

O ficheiro movie.js tem por sua vez a responsabilidade de pedir ao servidor os filmes necessários para preencher a página principal e a página de filmes.

- nights.js

Por fim, nights.js trata exclusivamente das noites de filmes. Seja para criar modificar ou até removê-las.

2) Back-end

O Back-end é composto por 3 servidores server.js, authServer.js e redirect.js.

- server.js

Este é o servidor principal. O mesmo é o qual que se conecta a grande maioria da base de dados. Como tal, para dividir as suas rotas mais facilmente este faz uso de 3 routers, “movies”, “nights” e “users”.

Em relação as rotas cada uma destas é composta por 3 elementos o router.js, o controller.js e queries.js. Isto, com exceção do movies que não possui querie pois não se conecta a base de dados, mas sim a API externa.

O router.js tem como única função fornecer as rotas para o servidor.

Já o controller.js é o coração de cada rota. Este conecta-se a base de dados, executa os pedidos e devolve ao cliente.

Por fim queries.js tem o simples objetivo de guardar as queries necessárias ao uso da base de dados. Estes são guardadas aqui pois são usadas mais que uma vez regularmente.

- authServer.js

Tal como o nome indica este servidor tem como propósito tratar da autenticação. Tal como o servidor anterior este também tem acesso a rotas. Mas neste caso é só uma "auth".

Para além dos ficheiros já mencionados no servidor anterior esta rota tem mais 1 middleware.js

Este middleware.js proporciona ao resto das rotas autenticação automática para que não se tenha de estar a pedir ao utilizador os seus credenciais constantemente.

- redirect.js

Por fim este é o servidor mais simples tem como único propósito redirecionar os utilizadores de http para https.

3) Base de Dados

A base de dados para este projeto é relativamente simples esta é composta 5 tabelas que se conformam da seguinte maneira.

a) Users:

User_id	**Email**	**Passoword_hash**	**Active**	**Role**
Integer	Varchar (255)	Varchar(255)	Boolean	Varchar (255)
Pkey	Unique Not null	Not null	Not null	Not null

Tabela 1 - Users

b) Movie_nights:

Movie_night_id	**Movie_id**	**Movie_night _date**	**Description**
Integer	Integer	Integer	Text
Pkey	Not null	Timestamp Not null	

Tabela 2 - Movie_nights

c) User_nights

Movie_night_id	**User_id**	**Confirmed**	**Night_host**
Integer	Integer	Boolean	Boolean
Fkey		Not null	Not null
Pkey			

Tabela 3 - User_nights

d) Friendlist:

User_id	**Friend_id**
Integer	Integer
Fkey	Fkey
Pkey	

Tabela 4 - Friendlist

e) Tokens:

Token_id	**User_id**	**Token**
Integer	Integer	Text
Pkey	Fkey	Not null

Tabela 5 - Tokens

4) API Externa

Já a API Externa tem um propósito simples, fornecer os quais filmes existem de acordo com os dados de pesquisa que são dados.

A API escolhida para este propósito foi a "The Movie Database (TMDB) API".

IV. User Info

A. GitHub

https://github.com/VascoJ2000/DW_Projeto

B. Variáveis de ambiente

1) Base de dados

- DB_USER
- DB_HOST
- DB_DB
- DB_PASS
- DB_PORT

2) Autenticação

- ACCESS_TOKEN_SECRET
- REFRESH_TOKEN_SECRET

3) Servidor

- SERVER_PORT
- AUTH_PORT

4) API da base de dados de filmes

- TMBD_API_KEY

C. Software utilizado no desenvolvimento

- Visual Studio Code
  - Live Server
  - REST Client
- PM2
- pgAdmin 4

- Google Chrome

D. Correr o software

1) Preparar o software

Para o software correr é necessário instalar as suas dependências mais importantes. Ou seja, nodejs, PostgreSQL e qualquer browser com as últimas atualizações.

De seguida é necessário criar a base de dados para este sistema. Esta teve como nome em desenvolvimento movieNights, mas, poderá ter qualquer outro nome desde que DB_DB seja alterado. Aqui é só correr o ficheiro database.sql e a base de dados estará pronta.

Também é preciso fazer download das dependências do servidor. Para isso é só abrir um terminal na pasta onde se encontra package.json e escrever o comando "npm i".

Ainda é necessário criar certificados para o servidor. Se já tiver os certificados é só criar uma pasta cert na pasta principal e colocar os ficheiros key.pem e certificate.pem nessa pasta. Senão, basta correr o comando "& 'C:\Program Files\Git\usr\bin\openssl.exe' req -newkey rsa:2048 -nodes -keyout key.pem -x509 -days 365 -out certificate.pem" e colocar os ficheiros já mencionados na pasta cert.

Por fim é necessário criar um ficheiro dotenv. Este deverá ter variáveis correspondentes as variáveis de ambiente.

2) Iniciar o software

a) Opção 1 – usando pm2

Esta opção torna a execução do sistema mais simples pois é só necessário correr o comando:

- pm2 start server.js authServer.js redirect.js

b) Opção 2 – usando nodejs normal

Para esta opção é necessário ter 3 terminais onde cada corre o seguinte comando:

- node server.js
- node authServer.js
- node redirect.js

3) Encontrar o site

Agora para se encontrar o site que corresponde ao sistema basta ir ao seu navegador e escrever "localhost" ou "127.0.0.1".

Aqui pode encontrar-se um problema se os certificados não forem reconhecidos por nenhuma entidade. Isto fará que o site seja considerado inseguro. Para prosseguir basta clicar em "Avançado" e "Seguir de qualquer modo".

Para se certificar que o site não tem problemas com a autenticação deverá confirmar se o mesmo não acontece com o servidor de autenticação. Ou seja, colocar no browser "https://localhost:4000" ou "https://127.0.0.1:4000" e confirmar se o mesmo não acontece.