Universidade da Beira Interior
Departamento de Informática

Nº263 - 2022: InstaNews — Rede Social de Notícias

Elaborado por:

Miguel Matos

Orientador:

Professor João Paulo Cordeiro

11 de julho de 2022

Agradecimentos

Ao concluir o meu projeto final em Engenharia Informática agradeço ao meu orientador Professor João Paulo Cordeiro todo o apoio prestado. Porém não posso deixar de lembrar o apoio de toda a minha família, amigos e colegas por me terem ajudado a alcançar este novo patamar da minha vida. Por isso o meu muito obrigado a todos pela força que sempre me transmitiram.

Conteúdo

Acrónimos

XAMPP	*Cross-platform, Apache, MySQL, PHP and Perl*
HTTP	*Hypertext Transfer Protocol*
API	*Application Programming Interface*
URL	*Uniform resource locator*
UI	*User interface*

Capítulo

1

Introdução

1.1 Enquadramento

Este trabalho foi desenvolvido como projeto final de licenciatura em Engenharia Informática. O mundo das redes sociais tem crescido a um ritmo quase exponencial, durante a última década, existindo atualmente uma taxa de penetração de 53.6% na população mundial. Um em cada dois habitantes do planeta é um utilizador de alguma rede social e estima-se que em 2025 existirão 4.4 mil milhões de utilizadores. Criar uma plataforma onde as pessoas possam ter acesso à informação sobre o quotidiano e poder discutir essa mesma atualidade como se de um fórum se tratasse, parece ser do maior interesse para os utilizadores.

1.2 Motivação UBI

O desenvolvimento de plataformas *Web* é uma das áreas mais cativantes e é indubitavelmente uma das que se pretende aqui, aprofundar mais. Isto porque grande parte da nossa vida é hoje em dia, passada à frente do computador em diferentes tipos de plataformas, por exemplo o Instagram, uma rede social que a maior parte das pessoas usa diáriamente.

1.3 Objetivos

Com este projeto, pretende-se construir uma aplicação que possa servir de rede social centrada na temática das notícias, onde os utilizadores podem realizar os procedimentos habituais de uma rede social. Pretende-se também

que a rede seja auto alimentada a partir de fontes noticiosas existentes, como jornais e televisões com presença online. Logo, um utilizador desta rede social irá ter um espaço onde poderá comentar ou debater com os outros utilizadores sobre as notícias.

1.4 Organização do Documento

De modo a refletir o trabalho que foi feito, este documento encontra-se estruturado da seguinte forma:

1. O primeiro capítulo – **Introdução** – apresenta o projeto, a motivação para a sua escolha, o enquadramento para o mesmo, os seus objetivos e a respetiva organização do documento.

2. O segundo capítulo – **Estado da Arte** – descreve a localização da plataforma em relação ao resto do mercado.

3. O terceiro capítulo – **Tecnologias Utilizadas** – descreve os conceitos mais importantes no âmbito deste projeto, bem como as tecnologias utilizadas durante o desenvolvimento da aplicação.

4. O quarto capítulo – **Implementação e Testes** – detalha como é que o código do projeto está organizado e como é que as funcionalidades base estão implementadas.

5. O quinto capítulo – **Conclusões e Trabalho Futuro** – são discutidas as conclusões que podem ser tiradas após a realização deste projeto.

Capítulo

2

Estado da Arte

2.1 Introdução

Neste capítulo, será inicialmente apresentada a localização desta plataforma em relação ao resto do mercado, e por fim, uma breve conclusão sobre este capítulo.

2.2 Localização da Plataforma no Mercado

Desde sempre, houve necessidade de divulgar notícias. Começou com a divulgação boca a boca, com o surgimento da escrita evoluiu para diversas formas de comunicar. Por volta do século XVII, nasceram os jornais e mais recentemente as plataformas online, onde as pessoas têm acesso à informação de todo o mundo com apenas um clique. Dentro destas, existem plataformas que vão buscar notícias a várias fontes e as publicam, no caso português o Sapo, Notícias ao Minuto e a nível mundial Yahoo! News com 175 milhões de visitantes por mês, Google News com mais de 150 milhões por mês. Existem outras que são independentes, apenas publicam notícias que são da própria autoria como por exemplo, os sites do Público e do Expresso. O problema com estes tipos de plataformas é que apesar disponibilizarem a informação, não permitem a discussão e o debate das mesmas. Apesar de haver algumas que de forma indireta nos permitem fazer isso, como o Facebook e o Instagram, nenhuma tem o foco principal a discussão e debate de notícias. Assim, este projeto visa preencher esse vazio no mercado. A plataforma projetada, prevê ir buscar notícias de várias fontes, disponibilizando as principais manchetes de cada dia e simultaneamente permitir a sua discussão e o seu debate.

2.3 Conclusão

Neste capítulo, é descrito o estado da arte referente à temática deste projeto, oferecendo assim, uma melhor compreensão dos pressupostos a ter em consideração para se alcançar o objetivo pretendido e disponibilizar às pessoas meios para as poder ajudar a tomar decisões, que de uma forma positiva ou negativa tendem a afetar as suas vidas.

Capítulo

3

Tecnologias e Ferramentas Utilizadas

3.1 Introdução

Neste capítulo serão explicadas todas as tecnologias e ferramentas utilizadas para a realização deste projeto. Elas estão divididas em duas partes, *Front-end* e *Back-end*, sendo que em cada um destes componentes tem varias tecnologias a ser utilizadas.

3.2 *Visual Studio Code*

O Visual Studio Code é um editor de código simplificado com suporte para operações de desenvolvimento, como depuração, execução de tarefas e controlo de versão. O código para este projeto foi desenvolvido com o auxílio desta ferramenta.

3.3 *Cross-platform, Apache, MySQL, PHP and Perl* (XAMPP)

O XAMPP foi utilizado para criar o servidor local.

3.4 *Front-end*

O *front-end* é a parte do programa que lida com os elementos visuais com que o utilizador irá interagir.

3.4.1 *React js* [1]

React é uma biblioteca *open-source* de JavaScript *front-end* para construir interfaces baseadas em componentes. Todas as páginas desta plataforma são componentes, assim como todas as suas secções. Por exemplo A página principal é consituida por dois componentes, *HeaderComponent* e *Feed*. Estes componentes fazem parte de um componente pai, *FeedPage*. Esta lógica aplica-se a todas as páginas da plataforma. O *React js* não só permite criar e manipular componentes, mas também permite ter acesso aos seus *Hooks*. O *useState()* é um *Hook* que permite ter variáveis de estado em componentes funcionais. No fundo dá-nos a capacidade de encapsular o estado local num componente funcional. *useEffect()* é um *Hook* que diz ao *React js* que o componente precisa fazer algo após a sua renderização.

3.4.2 *Axios* [2]

Axios é um cliente *Hypertext Transfer Protocol* (HTTP) baseado em promessas para *JavaScript*. Neste projeto é utilizado para fazer pedidos ao *Application Programming Interface* (API), quer seja para enviar ou para receber dados.

3.4.3 *Material-UI* [3]

Material-UI é uma biblioteca que nos permite importar e usar diferentes componentes para criar uma *interface*. Isto economiza uma quantidade significativa de tempo, pois os *developers* não precisam de escrever os componentes gráficos todos do zero.

3.4.4 *React Router* [4]

React Router é uma biblioteca para roteamento em React. Permite a navegação entre vários componentes numa aplicação *React js*, possibilita alterar o *Uniform resource locator* (URL) do navegador e mantém a *User interface* (UI) sincronizada com a URL. Semelhante ao *React js* temos também *Hooks* do React Router que nos auxiliam. *useNavigate()* devolve uma função que permite navegar entre componentes, por exemplo, após o envio de um formulário. *useParams()* permite a ter acesso aos parâmetros do URL da rota atual, por

exemplo, quando se insere o URL de um determinado utilizador, pretende-se ir buscar esse nome para depois ter acesso às restantes informações.

3.4.5 *React Helmet* [5]

O *React Helmet* é um componente para gerir dinamicamente a seção principal do documento. Neste projeto, foi utilizado com o intuito de definir o título do documento.

3.5 *Back-end*

O *back-end* é a parte do programa que lida com o servidor. É aqui que se processam os dados.

3.5.1 *Node js* [6]

Node js no contexto deste projeto, é utilizado para desenvolver aplicações do lado do servidor, *server-side* da plataforma.

3.5.2 *ExpressJs* [7]

O *ExpressJS* é uma *framework* do *Node js* que ajuda a criar *server-side applications*. No contexto deste programa, vai ser utilizado para ajudar a gerir o servidor e as rotas.

3.5.3 *JSON Web Tokens* [8]

JSON Web Token, é usado para compartilhar informações de segurança entre duas partes, um cliente e um servidor. Neste projeto, serve para verificar se a pessoa que acedeu à página está autenticada.

3.5.4 *MySQL* [9]

MySQL é um sistema de gerenciamento de bases de dados relacional baseado em *SQL*. É utilizado para guardar qualquer tipo de informações, sobre utilizadores, posts, etc.

3.5.5 *Bcrypt* [10]

Bcrypt é utilizado para fazer hash das passwords introduzidas pelos utilizadores no registo e para comparar *hashes* para o *login*.

3.5.6 *Node-cron* [11]

Node-cron é utilizado para executar certas funções a uma determinada altura do dia. Isto porque, se pretende que todos os dias à meia noite, sejam recolhidas as principais noticias do dia.

3.6 Conclusões

Resumidamente, a maior parte das tecnologias e ferramentas utilizadas serviram para auxiliar as duas principais, *Node js - Back-end* e *React js - Frond-end.*

Capítulo

4

Implementação e Testes

4.1 Introdução

Neste capítulo, irá ser descrito como funciona a base do projeto e como está estruturado.

4.2 Estrutura do projeto

O projeto está dividido em duas grandes partes, *front-end* e *back-end*. Enquanto que a parte de *front-end* está focada nos elementos visuais com que o utilizador irá interagir, o *back-end* centra-se no lado do servidor

4.3 Estrutura

4.3.1 *Front-end*

O *front-end* está estruturado com base em duas pastas, ***components*** e ***page***. Na pasta ***pages*** é onde se situa componente principal de uma página que depois é utilizado para conter os componentes das *features*, como o *header*, *posts*, *modals*, etc. Esses componentes estão localizados na pasta ***components***.

4.3.2 *Back-end*

O *back-end* está dividido entre o ficheiro principal **index.js** e uma pasta ***routes*** que tem guardada todos os *endpoints*. Foi feita esta divisão, de modo a que fosse assegurada a organização do ficheiro **index.js**, caso não fosse assim, o

ficheiro teria mais de 1000 linhas o que tornaria tudo mais desorganizado e difícil de gerir. Os *endpoints* estão todos guardados dentro da pasta ***routes***. Dentro dela, ainda existem mais pastas para garantir melhor organização dos *endpoints*. Por exemplo, agrupar todos os *endpoints* relacionados com utilizadores dentro de uma pasta, todos os relacionados com *posts* dentro de outra e assim sucessivamente.

4.4 Pedidos à API

Podemos fazer dois tipos de pedidos. Pedidos protegidos e pedidos desprotegidos. A diferença entre eles é que para o utilizador fazer um pedido protegido ele precisa de estar validado, enquanto que, nos desprotegidos toda a gente os podem realizar. Neste projeto, os pedidos são todos protegidos exceto aqueles que o utilizador utiliza para efetuar o ***Log In*** e o **Registo**.

4.5 Autenticação e Autorização

A **autenticação** verifica a identidade de um utilizador ou serviço e a **autorização** determina os seus direitos de acesso. Embora os dois termos pareçam semelhantes, eles desempenham papéis diferentes, mas igualmente essenciais na proteção de uma aplicação. Combinados, eles determinam a segurança de um sistema.

4.5.1 Autenticação

A autenticação utilizador é feita sempre através do componente do *header*, isto porque é um componente comum a todas as páginas.
Para manter o código mais organizado a função que permite validar o utilizador foi posta num ficheiro separado, **Uservalidity.js**. Esta função verifica se o utlizador tem algum *token* guardado no *local storage*, se tiver verifica a validade do *access token*, se não tiver o utilizador será redirecionado para o *login*. Esta parte é feita através do *endpoint* **validar_user**.
Caso o *access token* esteja válido significa que o utilizador está autenticado, se não estiver a função irá tentar gerar outro *access token* através do *refresh token* guardado na base de dados, o que é feito através da chamada do *endpoint* **novo_token**. No entanto, se esse também se encontrar inválido o utilizador irá ser redirecionado para *login*.

```
import Axios from "axios";

export const Uservalidity = async () => {
  let resposta = 0;
  // Se a resposta for 1 significa que o utilizador esta validado
  // Se a resposta for -1 significa que o utilizador nao est  validado
  async function check() {
    // Obter o token
    const token = localStorage.getItem("token");

    // Se existir token
    if (token) {
      // Declarar o header
      const headers = {
        Authorization: `Bearer ${token}`,
      };
      // Chamar endpoint
      await Axios.post("http://localhost:3001/validar_user", null, {
        headers,
      }).then(async (response) => {
        // Significa que o utilizador nao tem nenhum token, logo tem que
            ser redirecionado para o login
        if (response.data === -2) {
          resposta = -1;
        } // Se existir token mas for inv lido
        else if (response.data === -1) {
          // Chamar o endpoint para tentar criar outro access token a
              partir do refresh token que esta guardado na base de dados
          await Axios.post("http://localhost:3001/novo_token", {
            token: token,
          }).then((response_new) => {
            // Refresh token esta invalido portanto redirecionar para o
                login
            if (response_new.data === -1) {
              resposta = -1;
            } // Foi criado novo token
            else {
              localStorage.setItem("token", response_new.data);
              resposta = 1;
            }
          });

          // Se o token for valido
        } else if (response.data === 1) {
          resposta = 1;
        }
      });
    }
```

```
    // Se nao existir token nenhum
    else {
      resposta = -1;
    }
  }

  //Chamar funcao
  await check();
  return resposta;
};
```

Excerto de Código 4.1: Função Uservalidity.

```
const express = require("express");
const router = express.Router();
const jwt = require("jsonwebtoken");
var db = require("../../database");

// Verificar se o access token e valido
router.use("/", function authenticate(req, res) {
  //Dar Request ao header
  const authHeader = req.headers.authorization;
  //Check bearer/token e undefined
  if (typeof authHeader !== "undefined") {
    //Obter token
    const token = authHeader && authHeader.split(" ")[1];
    //Verificar a validade do token
    jwt.verify(
      token,
      process.env.ACCESS_TOKEN_SECRET,
      async (expired, UserData) => {
        // Se houver algum erro significa que o token esta invalido
        if (expired) {
          res.send("-1");
        }
        // Se nao ocorrer erros significa que esta v lido
        else {
          res.send("1");
        }
      }
    );
  } // Se o token nao estiver definido e porque o utilizador nao deu
    login
  // Redirecionar para a pagina login
  else {
    res.send("-2");
  }
```

```
});

module.exports = router;
```

Excerto de Código 4.2: *Endpoint* validar_user

```
const express = require("express");
const router = express.Router();
const jwt = require("jsonwebtoken");
var db = require("../../database");

// Temos duas opcoes
// Temos o Refresh token valido e podemos criar outro access token a
    partir dele
// Nao temos o Refresh token valido e temos de criar um novo refresh
    token e access token caso seja isso redirecionar para o login

router.use("/", async (req, res) => {
  let RefreshToken;
  let UserData;
  let token = req.body.token;
  let UserId;
  try {
    //Dar decode ao access token para obter o id do utilizador
    UserId = jwt.decode(token).id;

    // Ir buscar o Refresh Token do utilizador com base no id obtido
    let sqlToken = "SELECT RefreshToken FROM users WHERE  id = ?";

    await new Promise((resolve, reject) =>
      db.query(sqlToken, UserId, (err, result) => {
        if (err) {
          reject(err);
        } else {
          resolve(result);
          RefreshToken = result[0].RefreshToken;
        }
      })
    );
    // Verificar se o Refresh Token e valido
    jwt.verify(
      RefreshToken,
      process.env.REFRESH_TOKEN_SECRET,
      async (expired, TokenData) => {
        // Se o refresh token for invalido
        if (expired) {
```

```
          resposta = -1;
          res.send("-1");
        }
        // Se o refresh token for valido
        else {
          console.log("Access token renovado");
          // Como o refresh token e valido vamos retirar as suas
              informacoes
          UserData = {
            email: TokenData.email,
            id: TokenData.id,
          };
          // Assinar um novo access token
          const access_token = jwt.sign(
            UserData,
            process.env.ACCESS_TOKEN_SECRET,
            { expiresIn: "1y" }
          );

          // Mandar o novo token
          resposta = 1;
          res.json(access_token);
        }
      }
    );
  } catch {
    res.send("-1");
  }
});
module.exports = router;
```

Excerto de Código 4.3: *Endpoint* novo_token

4.5.2 Autorização

A forma de ver se um utilizador está autorizado e autenticado é muito semelhante. A função utilizada para as pedidos protegidos ***Authfunctions.js*** é igual à ***Uservalidity.js***, só difere no facto de chamar o *endpoint* ***auth*** em vez do **validar_user**. O ***auth*** aje como *middleware*, significa isto que quando uma pessoa faz um *request* do género ***http://localhost:3001/auth/info*** o pedido passa primeiro pelo *endpoint* ***auth*** e só depois é que vai para o ***info***. Se no caso de um utilizador fizer um pedido destes e não estiver validado, o pedido pára no ***auth*** e redireciona-o para o *login*.

```
const express = require("express");
```

```
const router = express.Router();
const jwt = require("jsonwebtoken");
var db = require("../../database");

// Verificar se o access token e valido
// Se for vai para a proxima funcao visto que isto e middleware

router.use("/", function authenticate(req, res, next) {
  //Obter o header
  const authHeader = req.headers.authorization;
  //Verificar se o token esta indefinido
  if (typeof authHeader !== "undefined") {
    //Obter o token
    const token = authHeader && authHeader.split(" ")[1];
    //Verificar a validade do access token
    jwt.verify(
      token,
      process.env.ACCESS_TOKEN_SECRET,
      async (expired, UserData) => {
        // Se ocorrer um erro
        if (expired) {
          res.send("-1");
        }
        // Se nao ocorrer erro nenhum passamos para a proxima funcao
        else {
          req.user = UserData;
          next();
        }
      }
    );
  } else {
    // Se o token nao estiver definido e porque o utilizador nao deu
        login
    // Redirecionar para a pagina login
    console.log("redirect para login...");
    res.send("-2");
  }
});

module.exports = router;
```

Excerto de Código 4.4: *Endpoint* auth

4.6 Conclusões

Neste capítulo, foi visto como é que o projeto foi estruturado assim como as funcionalidades base, que permitem o funcionamento básico da aplicação.

Capítulo

5

Conclusões e Trabalho Futuro

5.1 Conclusões Principais

Realizaraaa este projeto, envolveu muito estudo sobre como estruturar um projeto desta dimensão assim como assegurar a sua segurança. Apesar da existência de alguns impasses na realização do mesmo, o autor sente que adquiriu experiência, não só ao nível das ferramentas utilizadas, a destacar o *React js* e *Node js*, mas também a linguagem *Javascript*.

5.1.1 Objetivos Alcançados

Apreciação dos *posts*/comentários	Objetivo não alcançado
Responsividade do *website*	Objetivo não alcançado
Auto alimentação de noticias	Objetivo alcançado
Partilhar *posts*	Objetivo alcançado
Perfil	Objetivo alcançado
Editar perfil	Objetivo alcançado
Pesquisar utilizadores	Objetivo alcançado
Sistema de amizade	Objetivo alcançado
Comentar *posts*	Objetivo alcançado

A plataforma cumpre os objetivos específicos que foram propostos, ou seja, a leitura, partilha e discussão dos principais acontecimentos do dia.

5.2 Trabalho Futuro

Como trabalho futuro, teria como proposta desenvolver os objetivos que não foram alcançados neste trabalho.

Bibliografia

[1] React js. https://reactjs.org/.

[2] Axios. https://axios-http.com/docs/intro.

[3] Material-UI. https://mui.com/.

[4] React Router. https://reactrouter.com/.

[5] React Helmet. https://www.npmjs.com/package/react-helmet.

[6] Node js. https://nodejs.org/en/.

[7] Express js. https://expressjs.com/.

[8] JSON Web Token. https://jwt.io/.

[9] MySQL. https://www.mysql.com/.

[10] Bcrypt. https://www.npmjs.com/package/bcrypt.

[11] Node-cron. https://www.npmjs.com/package/node-cron.