Relatório DAW

Miguel Rosa a76936
Tiago Antunes a76920
Afonso Bagagem a80041

13 December 2024

Teachers:
Prof. Noélia Susana Costa Correia
Prof. Rúben David de Sousa Gomes

Description:
Relatório apresentado em cumprimento dos requisitos da cadeira de Desenvolvimento de Aplicações Web

Índice

1 Introdução

Este projeto foi desenvolvido com base nos conteúdos lecionados na disciplina de Desenvolvimento de Aplicações Web, utilizando os princípios e ferramentas ensinadas no decorrer da mesma, com outros frutos de pesquisa, tendo em vista a criação de uma loja virtual.

2 REST API

A REST API foi desenvolvida utilizando Express e está configurada no ficheiro server.ts.

A API disponibiliza endpoints para produtos, utilizadores e encomendas, que são geridos pelos respetivos controladores: productController.ts, userController.ts e orderController.ts.

Cada controlador lida com as operações CRUD (Create, Read, Update, Delete) para os recursos relevantes e interage com as bases de dados NeDB (pode consultar a configuração e manipulação da base de dados em server/src/utils/database.ts).

A lógica de arranque do servidor pode ser visualizada no ficheiro main.ts, onde a aplicação Express é inicializada, o middleware é configurado e as rotas para cada controlador são definidas.

- **Endpoints de Produto**: Geridos pelo `productController.ts`, estes endpoints permitem consultar todos os produtos, consultar pedidos pelo ID ou adicionar produtos:
  - `POST /api/products`: Adiciona um novo produto.
  - `GET /api/products`: Obtém todos os produtos.
  - `GET /api/products/:id`: Obtém um produto específico pelo ID.
  - `DELETE /api/products/:id`: Remove um produto pelo ID.
- **Endpoints de Utilizador**: Geridos pelo `userController.ts`, estes endpoints lidam com funcionalidades relacionadas com utilizadores, como registo, login e gestão de perfis:
  - `POST /api/users`: Regista um novo utilizador.
  - `GET /api/users`: Obtém todos os utilizadores.
  - `GET /api/users/:id`: Obtém informações de um utilizador específico.
  - `DELETE /api/users/:id`: Remove um utilizador pelo ID.
- **Endpoints de Encomenda**: Geridos pelo `orderController.ts`, estes endpoints tratam dos dados das encomendas, incluindo a criação, atualização ou consulta de informações de encomendas:
  - `POST /api/orders`: Cria uma nova encomenda.
  - `GET /api/orders`: Obtém todas as encomendas.
  - `GET /api/orders/:id`: Obtém informações de uma encomenda específica.
  - `PUT /api/orders/:id`: Atualiza uma encomenda específica.

3 Funcionalidade e Requisitos

3.1 Funcionalidades do Projeto

Servidor (`server/src/server.ts`)

- **REST API Endpoints:**
  - **Products:** Geridos pelo `productController.ts`.
  - **Users:** Geridos pelo `userController.ts`.
  - **Orders:** Geridos pelo `orderController.ts`.
- **Middleware:**
  - **CORS:** Configurado para permitir pedidos a partir de `http://localhost:9000`.
  - **Ficheiros Estáticos:** Serve ficheiros estáticos da diretoria `images`.
- **Database:** Utiliza NeDB para gerir as bases de dados de produtos, utilizadores e encomendas.

Cliente

- **Aplicação React:**
  - **Components:** Inclui as componentes `app.tsx`, `Login.tsx`, `Message.tsx`, `Navbar.tsx`, `Orders.tsx` `Product.tsx`, `Register.tsx`, e `ShoppingCart.tsx`.
  - **State Management:** Gerido através de Redux slices encontrados em `cartSlice.ts`, `ordersSlice.ts`, `productsSlice.ts`, e `userSlice.ts`.
  - **Routing:** Gerido pelo ficheiro `router.tsx`.
- **Styling:** Gerido através do ficheiro `main.css`.

3.2 Requisitos do Projeto

Stack:

- **Server:** Node.js, Express, TypeScript, NeDB.
- **Client:** React, TypeScript, Redux Toolkit, Webpack.

Segurança:

- Hashing de palavras-passe implementado com `bcrypt` no `userController.ts`.

Development Tools:

- **Testes:** Configurado com Jest (`jest.config.js`).
- **Processo de Build:** Gerido via Webpack (`webpack.config.js`) e TypeScript (`tsconfig.json`).

Comunicação com a API:

- O cliente comunica com as APIs do servidor utilizando Axios, configurado no `productsSlice.ts`.

Operações na Base de Dados:

- Scripts para popular as bases de dados localizados em `insertProducts.js`, `insertOrders.js` e `insertUsers.js`.

Configuração CORS:

- Configurado para garantir pedidos seguros de origens diferentes a partir da aplicação cliente.

4 Layout

O layout da aplicação foi concebido para ser o mais simples e interativo possível. Os componentes do layout incluem:

- **Navbar:** Contém links para as seguintes áreas:
  - **Home:** Página inicial.
  - **Produtos:** Secção para explorar os produtos disponíveis.
  - **Login:** Disponível caso o utilizador não tenha sessão iniciada.
  - **Sessão iniciada:** Quando o utilizador tem sessão iniciada, os links incluem *Carrinho*, *Encomendas*, o nome do utilizador, e a opção de *Logout*.
- **Ponto de entrada:** Apresenta ao utilizador os produtos de destaque.
- **Área de Produtos:** Onde o utilizador pode visualizar e selecionar os produtos que deseja adquirir.
- **Carrinho:** Secção onde o utilizador pode visualizar os produtos selecionados e proceder à compra.
- **Área de Encomendas:** Permite ao utilizador visualizar as suas encomendas.
- **Secção de Login:** Permite ao utilizador iniciar sessão. Caso não esteja registado, apresenta a opção para criar uma conta.

5 Configuração do Projeto

A configuração do projeto foi realizada para garantir um ambiente de desenvolvimento funcional, organizado e eficiente. Esta secção detalha os passos seguidos para configurar o servidor, cliente e ferramentas de desenvolvimento.

5.1 Configuração do Servidor

- **Dependências:** Foram instaladas as seguintes dependências no servidor:
  - `express`: Para criar a API REST.
  - `cors`: Para permitir pedidos de diferentes origens.
  - `nedb`: Para gestão da base de dados.
  - `bcrypt`: Para hashing de palavras-passe.
  - `typescript`: Para desenvolvimento em TypeScript.
  - `ts-node`: Para execução de ficheiros TypeScript sem necessidade de compilação manual.
- **Scripts npm:**
  - `start`: Inicializa o servidor utilizando `ts-node`.
  - `build`: Compila os ficheiros TypeScript para JavaScript.

5.2 Configuração do Cliente

- **Dependências:** Foram instaladas as seguintes dependências no cliente:
  - `react` e `react-dom`: Para a interface do utilizador.
  - `redux-toolkit`: Para gestão de estado.
  - `axios`: Para comunicação com a API.
  - `typescript`: Para desenvolvimento em TypeScript.
  - `webpack` e `webpack-dev-server`: Para empacotar e executar no ambiente de desenvolvimento.
- **Scripts npm:**
  - `start`: Inicializa o cliente em modo de desenvolvimento através do `webpack-dev-server`.
  - `build`: Empacota o código para produção utilizando `webpack`.

5.3 Ferramentas de Desenvolvimento

- **Testes:** O projeto utiliza o `Jest` para testes unitários. A configuração encontra-se no ficheiro `jest.config.js`.
- **Compilação e Build:**
  - O cliente e servidor utilizam ficheiros de configuração separados para `TypeScript` (`tsconfig.json`).
  - O processo de build do cliente é gerido pelo `webpack.config.js`.

- **Scripts de Base de Dados:** Foram criados scripts para popular as bases de dados iniciais:
  - `insertProducts.js`: Adiciona produtos à base de dados.
  - `insertUsers.js`: Adiciona utilizadores à base de dados.
  - `insertOrders.js`: Adiciona encomendas de teste.

5.4 Ambiente de Desenvolvimento

- **Configuração CORS:** O servidor permite pedidos de `http://localhost:9000`, garantindo o funcionamento do cliente em modo de desenvolvimento.
- **Configuração Local:** O cliente é servido em `http://localhost:9000` e comunica com o servidor em `http://localhost:3000`.
- **Debugging:** Ferramentas de desenvolvimento de React e Redux foram utilizadas para depuração de estado e interface.

6 Extras/Mudanças

Durante o desenvolvimento do projeto, introduzidas várias melhorias e funcionalidades em relação ao projeto base, destacando-se as seguintes:

- **Área de Testes:** Foi criada uma área dedicada a testes unitários utilizando `Jest`, que inclui:
  - Verificação da integridade das bases de dados (`products`, `users` e `orders`).
  - Testes para as operações de inserção, atualização, leitura e remoção de dados.
- **Gestão de Bases de Dados:** Foi implementada uma área específica denominada `api`, que permite:
  - Gestão direta das bases de dados (produtos, utilizadores e encomendas).
  - Monitorização e manipulação dos registos através de interfaces específicas ou scripts utilitários.
- **Funcionalidades de Autenticação:** A estrutura do site foi expandida para incluir funcionalidades de login e registo de utilizadores, incluindo:
  - Criação de contas de utilizador através do endpoint `POST /api/users`.
  - Implementação de áreas reservadas acessíveis apenas após autenticação bem-sucedida.