Ecori Task Manager

Claudio Henrique Gramulha

Contents

1 Task

1.1 Funcionalidades implementadas

- Uma rota de listagem de tasks, aceitando parâmetros como filtros (title ou description) e paginação (page e pageSize);
- Uma rota de Criação de uma task, recebendo os campos, title, description. O objeto completo, na base de dados (PostgreSQL), tem mais 3 campos: completed_at (para controlar quando uma task foi marcada como completa), created_at (para controlar quando a task foi criada) e updated_at (para controlar quando a task foi modificada);
- Uma rota de atualizar uma task, passando na rota o id da task, e aceitando parâmetros como title, description;
- Uma rota de Remover uma task, passando na rota o id da task;
- Uma rota de Marcar uma task como concluída, passando na rota o id da task;
- Controle de login;
- Todas as funcionalidades do Back-end foram implementadas no Front-end.

1.2 Tecnologias utilizadas

- Javascript
- Node.js
- React
- PostgreSQL

1.3 Dependências auxiliares

- bcrypt: para fazer hash de senhas para armazenar-las de forma segura;
- cors: para permitir que o servidor especifique quais origens têm permissão para acessar seus recursos;
- dotenv: para carregar as variáveis de ambiente do arquivo ".env" para a aplicação em desenvolvimento;
- express: framework web para Node.js para facilitar a criação de aplicativos web e APIs. Fornecendo uma série de recursos que simplificam o processo de desenvolvimento, como roteamento de URLs, middleware para manipulação de requisições e respostas, etc;
- jsonwebtoken: para criar e verificar tokens de acesso que são usados para autenticar usuários em sistemas web e APIs;
- pg: que fornece uma interface para trabalhar com o banco de dados PostgreSQL;
- uuid: para identificar de maneira única e global as tasks da aplicação.

2 Como instalar

1. Clone o repositório *https://github.com/claudgoeswild/ecori-tech-test* no Github
2. Navegue até a pasta client e instale todos os requerimentos utilizando o comando: npm i
3. Navegue até a pasta server e instale todos os requerimentos utilizando o comando: npm i
4. Baixe, instale e crie um usuário no PostgreSQL
5. Faça o set-up da database do projeto da mesma maneira que está feito em db.sql, dentro da pasta server

6. Faça o set-up dos arquivos .env tanto na raiz da pasta server como na raiz da pasta client
    - Na pasta client, pode-se colocar: REACT_APP_SERVERURL=http://localhost:8000 para testes locais
    - Na pasta server, o set-up deve conter as informações para acessar o banco de dados:
        - USRNAME=seuUsername
        - PASSWORD=suaSenha
        - HOST=localhost
        - DBPORT=5432 (ou outro port que estiver usando)
7. Tanto no diretório server como no diretório client, rode o comando : npm run start

3 Documentação da API

- GET /tasks:
    - Faz fetch de tasks baseado nos parametros:
        - userEmail (String): E-mail do usuário dono das tasks;
        - title (String, opcional): Título da tarefa;
        - description (String, opcional): Descrição da task;
        - page (Número, opcional): Número da página para paginação;
        - pageSize (Número, opcional): Número de tarefas por página.
    - Retorna: Array de tarefas que correspondem aos parâmetros especificados.
- POST /tasks:
    - Cria uma nova task;
    - Body:
        - user_email (String): E-mail do usuário que está criando a task;
        - title (String): Título da task;
        - description (String): Descrição da task;
        - created_at (Data): Data e hora em que a task foi criada.
    - Retorna: Detalhes da nova tarefa criada.
- PUT /tasks/:id:
    - Edita uma task existente;
    - Parâmetros:
        - id (String): ID da task a ser editada.
    - Body:
        - user_email (String): E-mail atualizado do usuário associado à task;
        - title (String): Título atualizado da task;
        - description (String): Descrição atualizada da task;
        - updated_at (Data): Data e hora atualizadas da task;
        - completed_at (Data): Data e hora em que a task foi concluída.
    - Retorna: Detalhes da tarefa editada.
- DELETE /tasks/:id:
    - Exclui uma task;
    - Parâmetros:
        - id (String): ID da task a ser excluída.

  - Retorna: Detalhes da tarefa excluída.

- POST /signup:
  - Registra um novo usuário;
  - Body:
    * email (String): E-mail do usuário;
    * password (String): Senha do usuário.
  - Retorna: E-mail do usuário registrado juntamente com um token JWT.

- POST /login:
  - Faz login de um usuário existente;
  - Body:
    * email (String): E-mail do usuário;
    * password (String): Senha do usuário.
  - Retorna: E-mail do usuário logado juntamente com um token JWT.

4 Como melhorar a aplicação no futuro

Acredito que com a criação dessa aplicação, tive a oportunidade de testar meus conhecimentos e aprender mais sobre node e react. Existem certos pontos que podem ser melhorados ou feitos de outra maneira, como, por exemplo:

- Adicionar verificação/confirmação de e-mail;
- Melhorar as mensagens de erro no código;
- Adicionar modo noturno;
- Refatorar algumas partes do código para torna-lo mais legível e eficiente, como por exemplo, o arquivo css (em casos de aplicações maiores), partes do código do modal e paginação;
- Criar configurações do usuário para que o mesmo possa alterar e-mail, senhas, deletar sua conta, etc;
- Introduzir tipagem ao código com TypeScript;
- Incluir testes;
- Entre outros...