API de Usuários e Posts — Trabalho Acadêmico

Este projeto é uma **API RESTful** desenvolvida em **Node.js** com **TypeScript** e **Express**. Foi estruturado em camadas para garantir a **separação de responsabilidades**, tipagem robusta e o uso de Status Codes HTTP apropriados para cada operação.

1. Funcionalidades Principais

A API oferece endpoints para gerenciar usuários e posts, implementando validações e regras de negócio:

Usuários

1. Listar todos os usuários.
2. Buscar usuários por **ID**, **nome** (busca por *query param*) ou **faixa etária** (busca por *route params*).
3. Adicionar e atualizar (PUT) todos os dados de usuários.
4. Remover usuários inativos que **não são administradores** e **não possuem posts**.

Posts

1. Listar todos os posts.
2. Criar um novo post com validações de conteúdo e existência do autor.
3. Atualizar parcialmente (PATCH) um post, permitindo apenas a modificação do **título**, **conteúdo** ou status de **publicação**.
4. Remover um post com **regras de autorização** (apenas o autor ou um administrador pode excluir).

2. Tecnologias Utilizadas

- **Node.js**: Ambiente de execução assíncrono.
- **TypeScript**: Linguagem tipada, fundamental para a robustez do código.
- **Express**: Framework web leve para Node.js.
- **Sequelize** : ORM utilizado para as operações de banco de dados.
- **Bcrypt**: Para criptografia segura de senhas no cadastro e atualização de usuários.
- **Custom Errors**: Classes de erro personalizadas para garantir que as respostas HTTP (400, 403, 404, 409, 500) sejam consistentes.

3. Arquitetura do Projeto

O projeto segue um padrão baseado em camadas para isolar lógica de negócio, acesso a dados e controle de requisições:

Pasta	Descrição
controllers	**Camada de Controle** (HTTP). Lida com a requisição, validações básicas de entrada e formatação da resposta.
services	**Camada de Serviço** (Lógica de Negócio). Implementa as regras de negócio, coordena a manipulação de dados e executa validações complexas.
repositories	**Camada de Repositório** (Acesso a Dados). Lida com as consultas e manipulação direta do banco de dados (queries SQL com Sequelize).
data	Arquivos de configuração e conexão com o banco de dados.
models	Definição das estruturas de dados do banco (Schemas ou Models).
routes	Definição das rotas da API e seus respectivos *controllers*.
errors	Classes para tratamento de erros personalizados.
types	Definição das interfaces e tipos de dados utilizados por toda a aplicação.
index.ts	Arquivo principal que inicializa a aplicação Express.

4. Endpoints da API

Os endpoints podem ser testados usando a coleção Trabalho1-API.postman_collection.json na raiz do projeto.

4.1. Endpoints de Usuários (/users)

Método	URI	Descrição	Exemplo de URI
GET	/users	Lista todos os usuários.	http://localhost:3000/users
GET	/users/id/:id	Retorna um usuário pelo ID.	http://localhost:3000/users/id/1
GET	/users/search?search=:nome	Retorna usuários cujo nome contenha o termo de busca.	http://localhost:3000/users/search?search=Joao
GET	/users/ageBetween/:min/:max	Retorna usuários com idade entre :min e :max.	http://localhost:3000/users/ageBetween/20/30
POST	/users/newUser	Cria um novo usuário. **Corpo JSON** esperado.	http://localhost:3000/users/newUser
PUT	/users/id/:id	Atualiza **todos** os dados do usuário.	http://localhost:3000/users/id/1
DELETE	/users/removeInatives	Remove usuários que não são admin e não têm posts.	http://localhost:3000/users/removeInatives

4.2. Endpoints de Posts (/posts)

Método	URI	Descrição	Regra de Autorização
GET	/posts/	Lista todos os posts.	N/A
POST	/posts/newPost	Cria um novo post.	authorId deve ser válido.
PATCH	/posts/patch/idPost/:idPost/idUser/:idUser	Atualização parcial do post.	Apenas o autor (:idUser) ou admin pode editar.
DELETE	/posts/delete/idPost/:idPost/idUser/:idUser	Deleta um post.	Apenas o autor (:idUser) ou admin pode deletar.

5. Como Executar o Projeto

Pré-requisitos: Certifique-se de ter o Node.js (versão 18+) e o npm instalados.

1. **Instale as dependências**:
   npm install

2. **Inicie o servidor**:
   npm run dev

   O servidor será iniciado em http://localhost:3000.