Olá, futur@ colaborador(a)! 🚀

Na Conéctar , estamos sempre em busca de pessoas talentosas que compartilhem nosso compromisso com inovação, qualidade e colaboração. Para avaliar suas habilidades técnicas e sua capacidade de resolver problemas do mundo real, preparamos este desafio prático.

Você será convidado a desenvolver uma aplicação full-stack que simule um sistema de gerenciamento de usuários. Este é um projeto genérico que pode ser adaptado para diferentes cenários no futuro — algo que reflete o dinamismo do nosso dia a dia aqui na Conéctar.

Vamos nessa? 🌟

Objetivo

Desenvolver uma aplicação com NestJS (backend) e ReactJS (frontend) , ambos utilizando TypeScript , para gerenciar informações de usuários. O sistema deve ser robusto, seguro e fácil de usar, proporcionando uma experiência fluida tanto para administradores quanto para usuários regulares.

Entregáveis

1. **Código Fonte** :
    - Link dos repositórios Git contendo os códigos do backend e frontend.
    - Instruções claras no README.md para configurar e executar a aplicação.
2. **Documentação** :
    - Documentação da API (Swagger ou Postman).
    - Explicação das decisões de design e arquitetura.
3. **Demonstração** :
    - Diferencial: Se possível, inclua um link para uma demonstração ao vivo (ex.: deploy no Heroku, Vercel ou Netlify) com usuário e senha de acesso.

Descrição do Sistema

Imagine que você está construindo uma ferramenta interna para a Conéctar . Essa ferramenta permitirá:

1. Gerenciamento de Usuários :
    - Cadastrar novos usuários.
    - Autenticar usuários com login seguro.
    - Listar, editar e excluir informações de usuários (com permissões específicas para administradores).
2. Interface Amigável :
    - Uma tela de login para acessar o sistema.
    - Uma tela de cadastro para novos usuários.
    - Uma tela de listagem de usuários (para administradores) ou perfil do usuário (para usuários regulares).

Nosso foco aqui é criar uma solução que seja escalável, segura e intuitiva. Vamos garantir que a aplicação seja útil tanto para equipes pequenas quanto para grandes grupos de colaboradores!

O Que Esperamos de Você

Backend (NestJS + TypeScript)

1. Autenticação de Usuários :
    - Implemente rotas para registro (`/auth/register`) e login (`/auth/login`).
    - Use JWT para autenticação. Após o login bem-sucedido, retorne um token que será usado nas requisições subsequentes.
    - Diferencial Opcional : Integre com um provedor de login de terceiros, como Google ou Microsoft , para permitir autenticação via OAuth 2.0.
2. Gerenciamento de Usuários :
    - Cada usuário deve ter os seguintes campos:
        - `id`: Identificador único.
        - `name`: Nome completo.
        - `email`: Endereço de email (único).
        - `password`: Senha criptografada (opcional se usar login de terceiros).
        - `role`: Papel do usuário (`admin` ou `user`).
        - `createdAt`: Data de criação.
        - `updatedAt`: Data de última atualização.

3. Operações CRUD :
    - Permita criar, ler, atualizar e excluir usuários.
    - Somente administradores podem listar todos os usuários ou excluir outros usuários.
    - Usuários regulares só podem visualizar e atualizar seus próprios dados.
4. Filtros e Ordenação :
    - Adicione a possibilidade de filtrar usuários por papel (ex.: `?role=admin`).
    - Permita ordenar usuários por nome ou data de criação (ex.: `?sortBy=name&order=asc`).
5. Notificações :
    - Implemente um endpoint que liste usuários inativos (sem login nos últimos 30 dias). Isso ajudará a equipe a identificar contas que precisam de atenção.
6. Testes Automatizados :
    - Inclua testes unitários e de integração para garantir que tudo funcione corretamente.

Frontend (ReactJS + TypeScript)

1. Tela de Login :
    - Crie um formulário simples com campos para email e senha.
    - Diferencial Opcional : Adicione botões para login com Google ou Microsoft .
    - Após o login bem-sucedido, redirecione o usuário para a tela de listagem de usuários (se for admin) ou para o perfil do usuário (se for usuário regular).
2. Tela de Cadastro :
    - Crie um formulário com campos para nome, email e senha.
    - Após o cadastro bem-sucedido, redirecione o usuário para a tela de login.
3. Tela de Listagem de Usuários (Somente Admins) :
    - Exiba uma lista de todos os usuários, mostrando nome, email, papel e status (ativo/inativo).
    - Inclua botões para editar ou excluir usuários.(Opcional)
    - Adicione filtros para buscar usuários por papel e ordenar por nome ou data de criação.(Opcional)
4. Tela de Perfil do Usuário (Usuários Regulares) :
    - Mostre as informações do usuário logado: nome, email e data de criação.

- Permita que o usuário atualize seu nome ou senha.

5. Interface Responsiva :
    - Garanta que a interface funcione bem em dispositivos móveis e desktops.

Requisitos Não Funcionais

1. Escalabilidade :
    - Projete o sistema para suportar muitos usuários sem perder desempenho.
2. Segurança :
    - Proteja contra vulnerabilidades comuns, como SQL Injection e XSS.
    - Use criptografia para senhas e tokens seguros para autenticação.
3. Documentação :
    - Forneça documentação clara da API usando Swagger.
    - Inclua instruções detalhadas no README.md para configurar e executar a aplicação.
4. Arquitetura Limpa :
    - Organize o código de forma modular e fácil de manter.

Tecnologias Recomendadas

Backend

- Framework : NestJS com TypeScript.
- Banco de Dados : PostgreSQL ou MySQL.
- ORM : TypeORM ou Sequelize.
- Autenticação : `jsonwebtoken`, `bcrypt`, e bibliotecas como `passport-google-oauth20` ou `passport-microsoft` para integração com provedores externos.
- Testes : Jest ou Mocha/Chai.
- Documentação : Swagger.

Frontend

- Framework : ReactJS com TypeScript.
- Estado Global : Context API ou Redux.
- Rotas : React Router.
- Estilização : CSS Modules, Styled Components ou TailwindCSS.
- HTTP Client : Axios ou Fetch API.
- Login Social : Bibliotecas como `react-google-login` ou `msal-react`.

Critérios de Avaliação

1. Backend :
    - Código limpo, organizado e bem estruturado.
    - Funcionalidades implementadas conforme especificado.
    - Cobertura de testes unitários e de integração.
    - Segurança e proteção contra vulnerabilidades.
2. Frontend :
    - Interface intuitiva, responsiva e fácil de usar.
    - Integração adequada com o backend.
    - Uso eficiente de estado global e rotas.
    - Estilização limpa e consistente.
3. Integração com Provedores Externos (Diferencial) :
    - Implementação funcional e segura de login com Google ou Microsoft.
    - Documentação clara sobre como configurar as credenciais dos provedores.
4. Documentação :
    - Documentação clara da API (Swagger).
    - Explicação das decisões de design e arquitetura no README.md.

Dicas da Conéctar

1. Planeje Bem :
    - Antes de começar a codificar, reserve um tempo para entender os requisitos e planejar sua abordagem.
2. Priorize o Essencial :
    - Foque nas funcionalidades básicas primeiro. Se sobrar tempo, adicione melhorias opcionais, como a integração com login de terceiros.
3. Seja Criativo :

- Sinta-se à vontade para propor melhorias ou funcionalidades adicionais que você acredita que poderiam agregar valor ao sistema.

4. Pergunte se Precisar :
    - Caso tenha dúvidas sobre o enunciado ou os requisitos, não hesite em entrar em contato conosco.

Esperamos que este desafio seja uma oportunidade para você mostrar suas habilidades e criatividade! Estamos an