## 🧪 Teste Técnico - NestJS

### 📋 Descrição

O objetivo deste teste é desenvolver uma API RESTful utilizando **NestJS** para gerenciamento de usuários. A aplicação deve permitir a **criação**, **busca**, **atualização** e **remoção** de usuários, seguindo a arquitetura padrão do NestJS (Modules, Controllers, Services, DTOs).

Além disso, a aplicação deve realizar o envio de **e-mail de boas-vindas** após a criação de um novo usuário.

---

### 📦 Estrutura da Entidade User

**Campos obrigatórios:**

- nome: string
- telefone: string (único)
- email: string (único)
- senha: string (criptografada com bcrypt)

---

### 🔧 Funcionalidades e Endpoints

### ✅ POST /users – Criar usuário

**Descrição:**
Cria um novo usuário no sistema.

**Regras:**

- E-mail e telefone devem ser únicos.
- A senha deve ser armazenada de forma criptografada.
- Após a criação bem-sucedida, enviar um **e-mail de boas-vindas** para o usuário.

**Validações:**

- Retornar **409 Conflict** se e-mail ou telefone já existirem.
- Retornar **500 Internal Server Error** em caso de erro no banco.
- Retornar **200 OK** com uma mensagem de sucesso ao criar o usuário e enviar o e-mail.

### 📧 Envio de E-mail de Boas-vindas

**Descrição:**
Após o usuário ser criado com sucesso, a aplicação deve enviar um e-mail de boas-vindas com as seguintes características:

- Assunto: Bem-vindo ao sistema!
- Corpo: Deve conter o nome do usuário e uma mensagem amigável.
- Ferramenta: Utilizar @nestjs-modules/mailer com nodemailer.

O envio do e-mail deve ser tratado com segurança: caso o e-mail falhe, o usuário ainda deve ser criado, mas deve ser logado o erro e retornado um aviso de que o e-mail não foi enviado.

---

### 🔍 GET /users/:id – Buscar usuário por ID

**Descrição:**
Retorna os dados de um usuário com base no seu ID.

**Validações:**

- Retornar **404 Not Found** se o usuário não for encontrado.
- Retornar **200 OK** com os dados do usuário caso exista.

---

### 🖍️ PUT /users/:id – Atualizar usuário

**Descrição:**
Atualiza os dados de um usuário existente.

**Regras:**

- Pode atualizar nome, telefone, email e senha.
- Senha deve continuar criptografada.
- Telefone e email devem continuar únicos.
- Retornar **409 Conflict** se e-mail/telefone já estiverem em uso por outro usuário.
- Retornar **404 Not Found** se o usuário não existir.
- Retornar **500 Internal Server Error** em caso de erro.
- Retornar **200 OK** com os dados atualizados.

### ❌ DELETE /users/:id – Remover usuário

**Descrição:**
Remove um usuário do sistema com base no ID.

**Validações:**

- Retornar **404 Not Found** se o usuário não for encontrado.
- Retornar **500 Internal Server Error** em caso de erro.
- Retornar **200 OK** com uma mensagem de confirmação.

---

### 🛠 Requisitos Técnicos

- Projeto deve seguir estrutura padrão do NestJS.
- Usar **TypeORM** ou **Prisma**.
- Usar class-validator para validação.
- Usar bcrypt para criptografia de senha.
- Usar @nestjs-modules/mailer + nodemailer para envio de e-mails.
- Tratar exceções com status HTTP apropriados e mensagens descritivas.
- Utilizar DTOs para validação de entrada e saída.