Estudo de Caso 11: Cadastro Seguro de Usuários do Sistema (com BCrypt)

Objetivo:

Implementar o **cadastro de usuários** (médicos, recepcionistas ou administradores) no sistema, garantindo que:

- A senha nunca seja armazenada em texto puro.
- O sistema use **BCrypt** para criptografar a senha antes de salvar no banco.
- O login de cada usuário seja único (não pode haver e-mails repetidos).

Regras de Negócio:

- A senha deve ter no mínimo **8 caracteres**.
- Antes de salvar, criptografar com `BCryptPasswordEncoder`.
- Caso já exista usuário com o mesmo e-mail, retornar erro.
- O tipo de perfil (`ADMIN`, `MEDICO`, `RECEPCIONISTA`) deve ser salvo.

Requisição:

- **Endpoint:** `POST /api/usuarios/cadastrar`

Request Body (JSON):

```
{
  "nome": "Maria Ferreira",
  "email": "maria.ferreira@smartmed.com",
  "senha": "SenhaSegura123",
  "perfil": "RECEPCIONISTA"
}
```

Resposta esperada:

```
{
  "mensagem": "Usuário cadastrado com sucesso",
  "usuarioID": 45
}
```

Estudo de Caso 12: Autenticação de Usuário com Verificação de Senha (BCrypt)

Objetivo:

Implementar o **login de usuário**, validando e-mails e senhas de forma segura usando **BCrypt**.

Regras de Negócio:

- Buscar usuário pelo e-mail informado.
- Comparar senha enviada com a senha criptografada no banco usando `BCryptPasswordEncoder.matches()`.
- Caso falhe, retornar erro de autenticação.
- Se válido, retornar informações do usuário e um **token fictício** (simulação de autenticação).

Requisição:

- **Endpoint:** `POST /api/usuarios/login`

Request Body (JSON):

```
{
  "email": "maria.ferreira@smartmed.com",
  "senha": "SenhaSegura123"
}
```

Resposta esperada (simulação de token):

```
{
  "mensagem": "Login realizado com sucesso",
  "usuario": {
    "id": 45,
    "nome": "Maria Ferreira",
    "perfil": "RECEPCIONISTA"
  },
  "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6Ikp..."
}
```