Estudo de Caso 1: Cadastro Seguro de Usuário (com Criptografia)

Objetivo: Implementar a funcionalidade de cadastro de novos usuários, garantindo que as senhas sejam armazenadas de forma segura.

Regras de Negócio:

- A senha deve ter no mínimo 8 caracteres.
- A senha deve ser criptografada usando um algoritmo seguro (por exemplo, **BCrypt** ou similar) antes de ser salva no banco.
- O e-mail de cada usuário deve ser único no sistema.

Requisição:

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

Request Body (JSON):
JSON
```
{
  "nome": "João Silva",
  "email": "joao.silva@email.com",
  "senha": "SenhaSegura123"
}
```

-

Resposta esperada (JSON):

JSON
```
{
  "mensagem": "Usuário cadastrado com sucesso",
  "usuarioID": 101
}
```

Componentes a implementar:

- DTO (Data Transfer Object) para a entrada (`UsuarioDTO`).
- **Repository** com `@Query` customizada para verificar se o e-mail já existe.
- **Service** com a lógica de negócio (validação e criptografia de senha).
- **Controller** com o endpoint `/api/usuarios/cadastrar`.

Estudo de Caso 2: Autenticação de Usuário (Login)

Objetivo: Permitir que usuários já cadastrados façam login no sistema, validando as credenciais de forma segura.

Regras de Negócio:

- Buscar o usuário pelo e-mail fornecido.
- Comparar a senha enviada na requisição com a senha criptografada salva no banco.
- Caso a validação falhe, retornar um erro de autenticação.
- Se as credenciais estiverem corretas, retornar uma mensagem de sucesso e os dados básicos do usuário (ex: ID e nome).

Requisição:

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

Request Body (JSON):
JSON
```
{
  "email": "joao.silva@email.com",
  "senha": "SenhaSegura123"
}
```

-

Resposta esperada (JSON):

JSON
```
{
  "mensagem": "Login realizado com sucesso",
  "usuario": {
    "id": 101,
    "nome": "João Silva"
  }
}
```

Componentes a implementar:

- DTO (Data Transfer Object) para a entrada e saída.
- **Repository** para buscar o usuário pelo e-mail.
- **Service** com a lógica de autenticação.
- **Controller** com o endpoint `/api/usuarios/login`.

Estudo de Caso 3: Criação de Treino Personalizado (com Blocos e Exercícios)

Objetivo: Implementar a funcionalidade que permite ao usuário criar um novo treino, associando blocos e exercícios do catálogo.

Regras de Negócio:

- O treino deve ser criado com um nome (ex: "Treino de Peito e Tríceps").
- Um treino deve ser composto por pelo menos um bloco.
- Cada bloco deve ter exercícios, que devem ser selecionados do catálogo (`Exercicio`) existente no banco de dados.

Requisição:

- **Endpoint:** `POST /api/treinos/criar`

Request Body (JSON):

```
JSON
{
  "idUsuario": 101,
  "nomeTreino": "Treino de Peito e Tríceps",
  "blocos": [
    {
      "nomeBloco": "Aquecimento",
      "ordem": 1,
      "exercicios": [
        { "idExercicio": 1, "ordem": 1 },
        { "idExercicio": 2, "ordem": 2 }
      ]
    },
    {
      "nomeBloco": "Principal",
      "ordem": 2,
      "exercicios": [
        { "idExercicio": 3, "ordem": 1 },
        { "idExercicio": 4, "ordem": 2 }
      ]
    }
  ]
}
```

-

Resposta esperada (JSON):

```
JSON
{
  "mensagem": "Treino 'Treino de Peito e Tríceps' criado com sucesso!",
  "idTreino": 54
}
```

Componentes a implementar:

- DTO (Data Transfer Object) de entrada com a estrutura aninhada para blocos e exercícios.

- **Repositories** para as entidades `Treino`, `Bloco` e `Bloco_Exercicio`.
- **Service** com a lógica de criação (salvar o treino, os blocos e os exercícios associados em uma única transação).
- **Controller** com o endpoint `/api/treinos/criar`.