Documentação Desafio API Bancária

Autor: Oswaldo Beltrani Neto

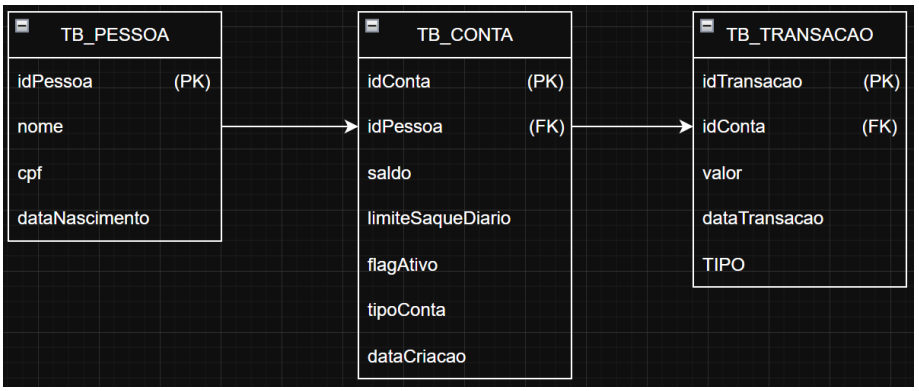
Este projeto contém uma API para gerenciamento de dados bancários.

Sua estrutura está composta da seguinte forma:

Back-end: A API é construída em **Java** utilizando o **framework Spring Boot** e a arquitetura **REST**, com **Maven** para gerenciamento de dependências. A documentação e os testes da API são realizados com **Swagger**, e os dados são armazenados em um banco **SQL Server**.

Front-end: A interface de usuário é desenvolvida com **React**, enquanto a biblioteca **Axios** é utilizada para o envio de requisições HTTP (GET, POST, DELETE e PUT) à API, garantindo a integração entre front-end e back-end.

▼ Documentação do Banco de Dados



▼ 1. Informações Gerais

- Nome do Banco: DESAFIO_API.
- Descrição: Banco de dados para gerenciamento de clientes, contas e transações financeiras.
- Objetivo: Armazenar dados de pessoas, contas vinculadas e suas transações.

▼ 2. Estrutura Geral

O banco é composto por três tabelas principais:

- TB_PESSOA → armazena os dados cadastrais dos clientes.
- TB_CONTA → registra as contas associadas às pessoas.
- TB_TRANSACAO → controla as movimentações financeiras das contas.

▼ 3 Dicionário de Dados

▼ 3.1 Tabela: TB_PESSOA

Descrição: Armazena os dados cadastrair dos clientes do sistema.

Coluna	Tipe	Tamanho	Nulo	AutoIncremento	PK	Referência
idPessoa	BIGINT	8	NÃO	SIM	SIM	-
nome	VARCHAR	100	NÃO	NÃO	NÃO	-
cpf	VARCHAR	14	NÃO	NÃO	NÃO	-
dataNascimento	DATE	3	NÃO	NÃO	NÃO	-

▼ 3.2 Tabela: TB_CONTA

Descrição: Contém os dados das contas bancárias vinculadas às pessoas.

Coluna	Tipo	Tamanho	Nulo	AutoIncremento	PK	FK
idConta	BIGINT	8	NÃO	SIM	SIM	NÃO
idPessoa	BIGINT	8	NÃO	NÃO	NÃO	SIM
saldo	DECIMAL	9	NÃO	NÃO	NÃO	NÃO
limiteSaqueDiario	DECIMAL	9	SIM	NÃO	NÃO	NÃO
flagAtivo	CHAR	1	NÃO	NÃO	NÃO	NÃO
tipoConta	INT	4	NÃO	NÃO	NÃO	NÃO
dataCriacao	DATE	3	NÃO	NÃO	NÃO	NÃO

▼ 3.3 Tabela: TB_TRANSACAO

Descrição: Registra as movimentações financeiras realizadas nas contas.

Coluna	Tipo	Tamanho	Nulo	AutoIncremento	PK	Referência
idTransacao	BIGINT	8	NÃO	SIM	NÃO	-
idConta	BIGINT	8	NÃO	NÃO	SIM	TB_CONTA.idCo
valor	DECIMAL	9	NÃO	NÃO	NÃO	-
dataTransacao	DATE	3	NÃO	NÃO	NÃO	-
tipo	VARCHAR	10	NÃO	NÃO	NÃO	-

▼ 4. Relacionamentos

1. TB_PESSOA (1) → (N) TB_CONTA
   - Uma pessoa pode ter uma ou mais contas.
   - Relacionamento feito por TB_CONTA.idPessoa → TB_PESSOA.idPessoa.
2. TB_CONTA (1) → (N) TB_TRANSCAO
   - Uma conta pode ter várias transações.
   - Relacionamento feito por TB_TRANSACAO.idConta → TB_CONTA.idConta.

Em resumo:

TB_PESSOA → TB_CONTA → TB_TRANSACAO.

▼ 5. Regras de Negócio

- Cada pessoa deve possuir um CPF único (Outros usuários não podem conter o mesmo CPF).

- Uma conta pertence sempre a uma única pessoa.
- Uma transação deve sempre estar vinculada a uma conta existente.
- O saldo da conta é atualizado de acordo com as transações realizadas.
- O campo flagAtivo determina se uma conta está operacional ou não.
- O campo tipoConta pode definir diferentes categorias de conta (Corrente, Poupança, etc).

▼ 6. Sequências

A fim de gerar ids únicos de forma automática para as tabelas, foi utilizado:

seq_pessoa, seq_conta e seq_transacao.

▼ Documentação da API - Java Spring Boot (REST)

▼ 1. Estrutura Geral do Projeto

Esta API segue a arquitetura em camadas e utiliza Maven para gerenciamento de dependências.

▼ 2. Camadas e Responsabilidades

Camada	Responsabilidade
Controller	Expõe os endpoints REST, recebe requisições HTTP, valida dados básicos e chama o Service correspondente.
Service	Contém a lógica de negócio. Realiza as regras, validações mais complexas e orquestra as chamadas ao repository.
Repository	Interface para acesso ao banco de dados utilizando JpaRepository.
Model	Define as entidades, basicamente, representam as tabelas do banco.
DTO	Objetos de transferência de dados, usados para entrada/saída sem expor diretamente a Model.
Exception	Classes para tratamento de erros, garantindo um retorno padronizado ao cliente em casos de erros.

▼ 3. Arquivos importantes

Arquivo	Função
CorsConfig.java	Configura CORS (Cross-Origin Resource Sharing), permitindo que clientes (Frontend em react) acessem a API sem restringir sua origem.
pom.xml	Arquivo de configuração do Maven. Define dependências, plugins, versão Java e configurações de build do projeto.
application.properties	Contém parâmetros de ambiente

▼ 4. Paths

Paths são os caminhos utilizados para acessar determinadas funcionalidades implementadas na aplicação. Nesta API, temos:

▼ 4.1 PessoaController

- GET ALL→ http://localhost:8080/pessoas

Retorna todas as pessoas cadastradas no banco, retornando um corpo em JSON assim:

```
[
  {
    "idPessoa": 3,
    "nome": "Estefani R",
    "cpf": "098.564.864-00",
    "dataNascimento": "2003-11-11"
  },
```

```
  {
    "idPessoa": 4,
    "nome": "Oswaldo B N",
    "cpf": "293.135.231-44",
    "dataNascimento": "2002-11-25"
  },
  {
    "idPessoa": 5,
    "nome": "Rogério Dallas",
    "cpf": "001.866.845-11",
    "dataNascimento": "1990-01-17"
  }
]
```

- PUT → http://localhost:8080/pessoas/{ID}

Torna possível a atualização de dados préviamente cadastrados, solicitando um corpo JSON, como por exemplo:

```
{
  "nome": "string",
  "cpf": "string",
  "dataNascimento": "2025-08-31"
}
```

- DELETE → http://localhost:8080/pessoas/{ID}

Realiza a deleção de uma conta de acordo com o ID informado, retornando 204 (Sucesso mas sem resposta)

- GET ByID → http://localhost:8080/pessoas/{ID}

Resgata somente a pessoa cadastrada com o ID informado, retornando o seguinte corpo em JSON:

```
{
  "idPessoa": 4,
  "nome": "Oswaldo B N",
  "cpf": "293.135.231-44",
  "dataNascimento": "2002-11-25"
}
```

- POST → http://localhost:8080/pessoas

Realiza a criação de uma nova pessoa, solicitando um corpo JSON com o seguinte padrão:

```
{
  "nome": "string",
  "cpf": "string",
  "dataNascimento": "2025-08-31"
}
```

▼ 4.2 ContaController

- GET ByID → http://localhost:8080/contas/{ID}

Busca uma conta através de seu ID, retornando um corpo JSON da seguinte forma:

```
{
  "idConta": 1,
  "idPessoa": 4,
  "saldo": 0,
  "limiteSaqueDiario": 200,
  "flagAtivo": "N",
```

```
  "tipoConta": 1,
  "dataCriacao": "2025-08-28"
}
```

- GET saldo ByID → http://localhost:8080/contas/{ID}/saldo

Busca o saldo de uma conta através do ID, retornando apenas o montante financeiro presente na conta.

- POST → http://localhost:8080/contas

Realiza a criação de uma nova conta, solicitando um corpo JSON com o seguinte padrão:

```
{
  "idPessoa": 0,
  "saldoInicial": 0,
  "limiteSaqueDiario": 0,
  "tipoConta": 0 #Pode ser tanto do tipo 1 ou 0 (corrente ou poupança).
}
```

- PATCH → http://localhost:8080/contas/{ID}/bloquear

Atualiza apenas o atributo “flagAtivo”, bloqueando uma conta, retornando o seguinte corpo em JSON:

```
{
  "idConta": 0,
  "idPessoa": 0,
  "saldo": 0,
  "limiteSaqueDiario": 0,
     "flagAtivo": "N",
  "tipoConta": 0,
  "dataCriacao": "2025-08-31"
}
```

▼ 4.3 TrasacaoController

- POST → http://localhost:8080/transacoes/{ID}/saque

Realiza a criação de um uma nova transação do tipo “SAQUE”, sendo necessário enviar o seguinte JSON na requisição:

```
{
  "valor": 0
}
```

- POST → http://localhost:8080/transacoes/{ID}/deposito

Realiza a criação de um uma nova transação do tipo “DEPOSITO”, sendo necessário enviar o seguinte JSON na requisição:

```
{
  "valor": 0
}
```

- GET → http://localhost:8080/transacoes/{ID}/extrato?start=YYYY-MM-DD&end=YYYY-MM-DD

Implementa a função de extrato por período, retornando o seguinte JSON:

```
[
  {
    "idTransacao": 26,
    "idConta": 9,
    "valor": 150,
```

```
  "dataTransacao": "2025-08-30T00:00:00",
  "tipo": "DEPOSITO"
 },
 {
  "idTransacao": 27,
  "idConta": 9,
  "valor": 150,
  "dataTransacao": "2025-08-30T00:00:00",
  "tipo": "DEPOSITO"
 },
 {
  "idTransacao": 28,
  "idConta": 9,
  "valor": 150,
  "dataTransacao": "2025-08-30T00:00:00",
  "tipo": "SAQUE"
 },
 {
  "idTransacao": 13,
  "idConta": 9,
  "valor": 100,
  "dataTransacao": "2025-08-29T00:00:00",
  "tipo": "SAQUE"
 },
 {
  "idTransacao": 14,
  "idConta": 9,
  "valor": 100,
  "dataTransacao": "2025-08-29T00:00:00",
  "tipo": "DEPOSITO"
 },
 {
  "idTransacao": 15,
  "idConta": 9,
  "valor": 100,
  "dataTransacao": "2025-08-29T00:00:00",
  "tipo": "DEPOSITO"
 },
 {
  "idTransacao": 16,
  "idConta": 9,
  "valor": 100,
  "dataTransacao": "2025-08-29T00:00:00",
  "tipo": "DEPOSITO"
 },
 {
  "idTransacao": 17,
  "idConta": 9,
  "valor": 100,
  "dataTransacao": "2025-08-29T00:00:00",
  "tipo": "DEPOSITO"
 },
 {
  "idTransacao": 18,
  "idConta": 9,
  "valor": 100,
  "dataTransacao": "2025-08-29T00:00:00",
  "tipo": "DEPOSITO"
```

```
  }
]
```