Manual de Execução

1. Requisitos

- Node.js >= 18
- Banco de dados MySQL
- Yarn ou npm

2. Instalação

```
git clone <REPO_URL>
cd desafioAPI/bancotest
npm install
```

3. Configuração do Ambiente

A. **Script de inicialização (`app/scripts/init-db.js`)**

```
import { connection, createDatabase } from '../database/db.js';

async function initDB() {
  try {
    await createDatabase();

    await connection.query(`
      CREATE TABLE IF NOT EXISTS Pessoas (
      idPessoa INT AUTO_INCREMENT PRIMARY KEY,
      nome VARCHAR(100) NOT NULL,
      cpf VARCHAR(14) UNIQUE NOT NULL,
      dataNascimento DATE NOT NULL,
      senha VARCHAR(255) NOT NULL
      );
    `);
```

```
    await connection.query(`
      CREATE TABLE IF NOT EXISTS Contas (
        idConta INT AUTO_INCREMENT PRIMARY KEY,
        idPessoa INT NOT NULL,
        saldo DECIMAL(15,2) NOT NULL DEFAULT 0.00,
        limiteSaqueDiario DECIMAL(15,2) NOT NULL DEFAULT 1000.00,
        flagAtivo BOOLEAN NOT NULL DEFAULT TRUE,
        tipoConta INT NOT NULL,
        dataCriacao DATETIME NOT NULL DEFAULT CURRENT_TIMESTAMP,
        FOREIGN KEY (idPessoa) REFERENCES Pessoas(idPessoa)
      );
    `);

    await connection.query(`
      CREATE TABLE IF NOT EXISTS Transacoes (
        idTransacao INT AUTO_INCREMENT PRIMARY KEY,
        idConta INT NOT NULL,
        valor DECIMAL(15,2) NOT NULL,
        dataTransacao DATETIME NOT NULL DEFAULT CURRENT_TIMESTAMP,
        FOREIGN KEY (idConta) REFERENCES Contas(idConta)
      );
    `);

    console.log('Banco inicializado com sucesso!');
    process.exit(0);
  } catch (err) {
    console.error('Erro ao inicializar o banco:', err);
    process.exit(1);
  }
}

initDB();
```

B. **Conexão com o banco (app/database/db.js)**

```
import mysql from "mysql2/promise";


const dbName = 'bancoteste2';


export async function createDatabase() {
  try {
    const connection = await mysql.createConnection({
      host: 'localhost',
      user: 'root',
      port: 3306,
      password: 'nicoleto'
    });


    await connection.query(`CREATE DATABASE IF NOT EXISTS
\`${dbName}\``);
    console.log(`Banco de dados '${dbName}' criado ou já existente.`);
    await connection.end();
  } catch (err) {
    console.error('Erro ao criar o banco de dados:', err);
  }
}


createDatabase();


export const connection = mysql.createPool({
  host: 'localhost',
  user: 'root',
  port: 3306,
  password: 'nicoleto',
  database: dbName,
  waitForConnections: true,
  connectionLimit: 10,
});
```

C. Scripts no `package.json` e `package.json` dentro da pasta `/app`

```
"scripts": {
    {...}
    "init-db": "ts-node app/scripts/init-db.js",
    "dev": "npm run init-db && next dev --turbopack",
  },
```

4. Execução

Para a execução do projeto, use o comando:

```
npm run dev
```

Documentação Técnica

2. Rotas Principais

Caminho	Descrição
`/cadastro`	Registro de novo usuário
`/login`	Autenticação
`/home`	Painel principal
`/deposito`	Realizar depósito
`/saque`	Realizar saque
`/extrato`	Histórico de transações
`/config`	Configurações da conta

2. Arquitetura

- **Frontend**: React (Next.js), TypeScript, Tailwind CSS
- **Backend**: API REST em rotas do Next.js
- **Banco**: MySQL
- **Autenticação**: Cookie HTTP + API `/login`

2. Fluxo de Funcionalidades

- **Cadastro:** `POST /api/pessoas`
- **Login:** `POST /api/login` → Cria cookie e navega para a página principal
- **Home:** `GET /api/pessoas` + `GET /api/transacoes/extrato`
- **Depósito/Saque:**
  - `POST /api/deposito` ou `POST /api/saque`
  - `GET /api/deposito/extrato` ou `/api/saque/extrato`
- **Extrato por período:**
  - `GET /api/transacoes/extrato?inicio=YYYY-MM-DD&fim=YYYY-MM-DD` (Como Default, o valor é os últimos **30 dias**)
- **Configuração de conta:**
  - `POST /api/usuario/desbloquear`
  - `POST /api/usuario/bloquear`
  - `POST /api/usuario/limite`
  - `GET /api/usuario`
  - `POST /api/logout`

3. Segurança

- Validação do CPF e formato de entrada
- Popups de senha para operações sensíveis
- Campos protegidos por autenticação
- Conta bloqueável manualmente

Casos para testes

1. Cadastro

- **Sucesso:** Nome, CPF válido, data de nascimento e senha
- **Erro:** CPF duplicado → erro esperado

2. Login

- **Sucesso:** CPF e senha corretos → redireciona
- **Erro:** CPF/senha incorretos → mensagem de erro

3. Depósito

- **Sucesso:** Valor positivo com senha correta
- **Erro:** Senha errada → operação falha
- **Erro:** Valor inválido → impedido

4. Saque

- **Sucesso:** Valor dentro do limite diário
- **Erro:** Valor excedente → bloqueado
- **Erro:** Conta inativa → erro esperado

5. Extrato

- **Sucesso:** Últimos 30 dias e por período
- **Sucesso:** Diferencia cor/valor entre depósitos e saques

6. Configurações

- **Sucesso:** Alterar limite de saque
- **Sucesso:** Bloquear e desbloquear conta
- **Sucesso:** Ver dados detalhados da conta