CRUD BÁSICO

HTML + CSS + JAVASCRIPT + REACT + TYPESCRIPT + NODE + EXPRESS

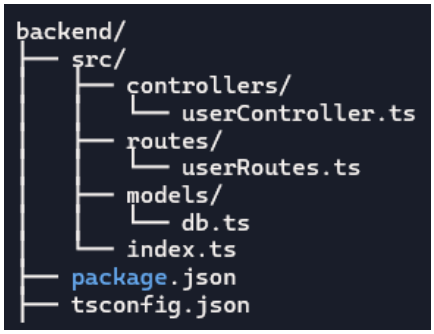
Backend

1) Banco de Dados

Tabela: usuario

Campo	Tipo	Condição	Descrição
id	serial	PRIMARY KEY	Identificator único
nome	VARCHAR(100)	NOT NULL	Nome completo
email	VARCHAR(100)	NOT NULL	e-Mail
telephone	VARCHAR(20)	NOT NULL	Telephone de contato
data_criacao	TIMESTAMP	NOT NULL DEFAULT CURRENT_TIMESTAMP	Data da criação do registro

2) Estrutura do backend (pastas):



3) Comandos estrutura backend:

npm init -y (cria o package.json)
npm i -D ts-node ts-node-dev typescript (instala dependências node e TypeScript)
caso a instalação anterior dê erro rode os comandos a seguir:
npm install -g npm@latest
npm install --save-dev ts-node-dev
tsc --init (cria tsconfig.json com opções e configurações do compilador TypeScript)
npm i dotenv (instalar o pacote dotenv para as variáveis de ambiente e configurações)
npm i pg (instalar a biblioteca que possui ferramentas para acessar o SGBD PostgreSQL)
npm i -D @types/pg (instalar o pacote que contém as definições de tipos do pacote pg)
npm install --save-dev @types/express (para instalar o pacote Express que cria um servidor Node local)
npm install @types/cors (middleware do Express que libera o acesso entre diferentes domínios, permitindo que o frontend se comunique com o backend mesmo em origens distintas)

4) Altere as propriedades da seção scripts no package.json:

```
{
```

```
"name": "exemplo",
"version": "1.0.0",
"description": "",
"main": "index.js",
"scripts": {
 "start": "ts-node ./src/index",
 "initbd": "ts-node ./src/controllers/init"
},
```

5) Crie o arquivo **.**env na raiz do projeto:

Coloque a seguinte variável de ambiente:

PORT=3001

6) Crie o arquivo .gitignore na raiz do projeto:

Coloque a linha **node_modules** para ignorar a pasta node_modules ao enviar o projeto para um repositório no Github.

7) Crie o arquivo para a conexão com o banco de dados: PostgreSQL (models/db.ts)

```
import query from "../models/db";

async function init() {
   return await query(`
      START TRANSACTION;
      DROP TABLE IF EXISTS usuario;
      CREATE TABLE IF NOT EXISTS usuario (
         id serial PRIMARY KEY,
         nome VARCHAR(100) NOT NULL,
         email VARCHAR(100) NOT NULL,
         telefone VARCHAR(20) NOT NULL,
         data_criacao TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP
      );
      COMMIT;
`);
}
init()
   .then((r) => console.log(r))
   .catch((e) => console.log(e));
```

8) Crie o arquivo (src/controllers/init.ts) para criar a tabela no banco de dados. Necessário já ter acessado o SGDB e criado o banco de dados **usuarios**.

```
import { Pool } from 'pg';

export const pool = new Pool({
   user: 'postgres',
   host: 'localhost',
   database: 'usuarios',
   password: '12345',
   port: 5432,
});

async function query(sql: string, params?: any[]) {
   try {
      const res = await pool.query(sql, params);
      if (res.command == 'INSERT') {
         return res.rows[0];
      }
      else if (res.command == 'SELECT') {
         return res.rows;
      }
      else if (res.command == 'DELETE' || res.command == 'UPDATE') {
         return { rowcount: res.rowCount };
      }
      else {
         return { sql };
      }
   }
   catch (e: any) {
      return { message: e.message };
   }
};
export default query;
```

9) Crie o arquivo do controlador (controllers/usuarioControlador.ts)

```
import { Request, Response } from 'express';
import { pool } from '../models/db';

export const getUsuarios = async (_: Request, res: Response) => {
  const result = await pool.query('SELECT * FROM usuarios ORDER BY id');
  res.json(result.rows);
};

export const getUsuarioById = async (req: Request, res: Response) => {
  const { id } = req.params;
  const result = await pool.query('SELECT * FROM usuarios WHERE id = $1', [id]);
  res.json(result.rows[0]);
};

export const createUsuario = async (req: Request, res: Response) => {
  const { nome, email, telefone } = req.body;
  const result = await pool.query(
    'INSERT INTO usuarios (nome, email, telefone, data_criacao) VALUES ($1, $2, $3, NOW())
RETURNING *',
    [nome, email, telefone]
  );
  res.json(result.rows[0]);
};

export const updateUsuario = async (req: Request, res: Response) => {
  const { id } = req.params;
  const { nome, email, telefone } = req.body;
  const result = await pool.query(
    'UPDATE usuarios SET nome = $1, email = $2, telefone = $3 WHERE id = $4 RETURNING
*',
    [nome, email, telefone, id]
  );
  res.json(result.rows[0]);
};

export const deleteUsuario = async (req: Request, res: Response) => {
  const { id } = req.params;
  await pool.query('DELETE FROM usuarios WHERE id = $1', [id]);
  res.sendStatus(204);
};
```

10) Crie agora o arquivo de rotas (src/routes/usuarioRotas.ts):

```
import express from 'express';
import {
  getUsuarios,
  getUsuarioById,
  createUsuario,
  updateUsuario,
  deleteUsuario,
} from '../controllers/usuarioControlador';

const router = express.Router();

router.get('/usuarios', getUsuarios);
router.get('/usuarios/:id', getUsuarioById);
router.post('/usuarios', createUsuario);
router.put('/usuarios/:id', updateUsuario);
router.delete('/usuarios/:id', deleteUsuario);

export default router;
```

11) E para finalizar a parte do backend, crie o arquivo de inicialização (src/index.ts):

```
import express from 'express';
import cors from 'cors';
import usuarioRotas from './routes/usuarioRotas';
import dotenv from "dotenv";
dotenv.config();

// será usado 3000 se a variável de ambiente não tiver sido definida
const PORT = process.env.PORT || 3000;
const app = express();
app.use(cors());
app.use(express.json());
app.use('/api', usuarioRotas);

app.listen(PORT, () => console.log('Servidor rodando na porta 3001'));
```

12) Os código do backend estão no seguinte repositório no Github:

https://github.com/hdblouro/crudbasicoback

Frontend

13) Estrutura do frontend (pastas):

14) Para começar o projeto do frontend, crie uma pasta qualquer, abra-a no Visual Studio Code, abra um terminal e digite o seguinte comando:

Se for criar o projeto do zero, use o seguinte comando:

npx create-react-app . --template typescript

Se for clonar do Github, use o seguinte comando para instalar todas as dependências necessárias:

npm install

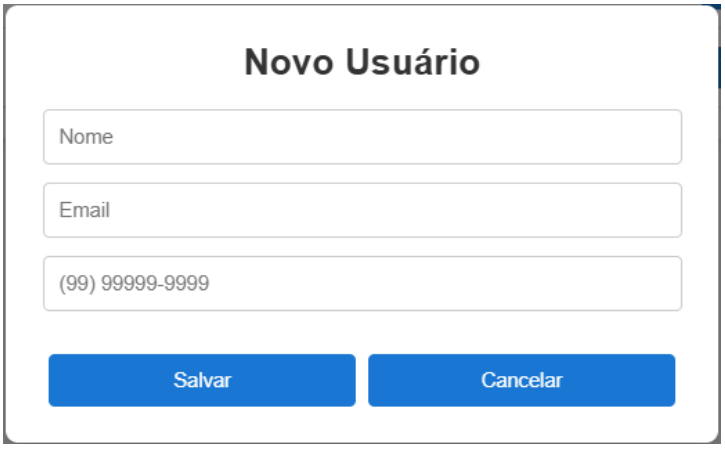
Toda estrutura necessária para o projeto do frontend será criada.

15) Os arquivos criados podem ser clonados do seguinte repositório no Github:

https://github.com/hdblouro/crudbasicofront

16) Essa é a tela base da aplicação web:

17) Essa é a tela de cadastro:



18) Essa é a tela do botão Ver:

19) Essa é a tela do botão Editar:

20) Essa é a tela de confirmação da exclusão:

21) Uma vez clonado o front, instale as dependências necessárias:

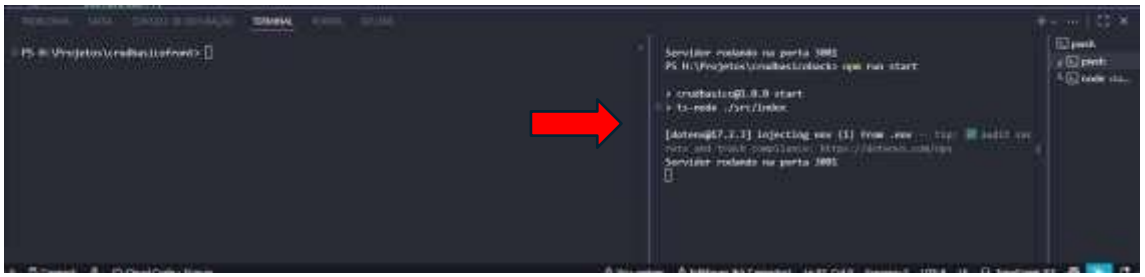
npm install

22) Suba o servidor de banco de dados, no caso o Postgresql.

23) Usando o PGAdmin, crie um banco chamado **usuários**. Altere a senha do usuário **postgres** no arquivo **/src/models/db.ts**. Ou crie um usuário e senha para acesso o banco de dados da sua preferência.

24) Para executar seu projeto, abra um segundo terminal no seu VSCode, navegue pelas pastas até a pasta do seu backend e execute os seguintes comandos:

npm run start



25) No primeiro terminal execute o mesmo comando:

npm run start

26) Nesse momento sua aplicação web irá iniciar automaticamente no seu navegador padrão.