Vitor Manoel Vidal Braz

Tutorial de Execução do Projeto –

API CRUD com NestJS, Prisma ORM e SQLite

Governador Valadares

2025

📓 Tutorial de Execução do Projeto

Projeto desenvolvido em sala de aula – Desenvolvimento Web Backend
Frameworks e Tecnologias: Node.js, NestJS, Prisma ORM, SQLite

📌 **Repositório Oficial:** Projeto Desenvolvido em Sala:
https://github.com/vitormanoelvb/projeto-desenvolvido-em-sala-de-aula-dw

---

1. Pré-requisitos

Antes de executar o projeto, certifique-se de ter instalado no computador:

- **Node.js** (versão LTS) → https://nodejs.org
- **NestJS CLI** globalmente:
- `npm i -g @nestjs/cli`
- **Visual Studio Code (VSCode)** com as seguintes extensões:
  - Dracula Official (tema escuro)
  - Material Icon Theme (ícones de pastas/arquivos)
  - DotENV (suporte a `.env`)
  - Prisma (extensão oficial)
  - Prisma Insider (versão de testes)
  - SQLite
  - SQLite Viewer
  - Prettier – Code Formatter (formatação automática de código)

---

2. Clonando o projeto

Acesse o repositório no GitHub e clone para sua máquina:

```
git clone https://github.com/vitormanoelvb/projeto-desenvolvido-em-sala-de-aula-dw
cd projeto-desenvolvido-em-sala-de-aula-dw
```

---

3. Instalando dependências

Dentro da pasta do projeto, execute:

```
npm install
```

---

4. Configuração do Banco de Dados

O projeto utiliza **SQLite** como banco de dados local.
O arquivo `.env` já está configurado:

```
DATABASE_URL="file:./dev.db"
```

Para gerar o banco de dados e aplicar a migration inicial, rode:

```
npx prisma migrate dev --name init
```

Isso criará o arquivo `dev.db` dentro da pasta `prisma/` e aplicará a tabela `usuarios`.

---

5. Estrutura do Banco de Dados

A tabela `usuarios` contém os seguintes campos:

- **id** (String, PK, gerado automaticamente)
- **nome** (String, obrigatório)
- **cpf** (String, único)
- **email** (String, único)

---

6. Executando o servidor

Para rodar o servidor em modo desenvolvimento, utilize:

```
npm run start:dev
```

A API ficará disponível em:

```
http://localhost:3000
```

---

7. Testando as rotas no Insomnia

As rotas disponíveis são:

🔹 Criar usuário (POST)

```
http://localhost:3000/usuario
```

```
{
  "nome": "Ana",
  "cpf": "12345678900",
  "email": "ana@exemplo.com"
}
```

🔹 Listar usuários (GET)

```
http://localhost:3000/usuario
```

🔹 Atualizar usuário (PUT)

```
http://localhost:3000/usuario/{id}
```

```
{
  "nome": "Ana Maria",
  "cpf": "12345678900",
  "email": "ana.maria@exemplo.com"
}
```

🔷 Excluir usuário (DELETE)

`http://localhost:3000/usuario/{id}`

---

8. Verificando no SQLite Explorer

Após executar as rotas, você pode validar diretamente no banco:

1. Abra o **SQLite Explorer** no VSCode (`Ctrl+Shift+P`).
2. Selecione **SQLite: Open Database**.
3. Abra o arquivo `prisma/dev.db`.
4. Clique na tabela **usuarios** para visualizar os registros inseridos, atualizados ou removidos.

---

9. Scripts úteis

```
npm run start         # Executar em produção
npm run start:dev     # Executar em desenvolvimento
npm run build         # Compilar o projeto
npm run test          # Executar testes unitários
npm run test:e2e      # Executar testes end-to-end
npx prisma studio     # Abrir interface gráfica do Prisma
```

---

10. Conclusão

Seguindo este passo a passo, você terá:
✅ O ambiente configurado corretamente.
✅ O servidor rodando com **NestJS + Prisma + SQLite**.
✅ CRUD de usuários funcionando (testado no Insomnia).
✅ Visualização dos dados diretamente no **SQLite Explorer**.