

## Introdução e Desafios

Para o projeto, foi escolhido utilizar o framework NestJS, que já havia sido apresentado em aula. Ele possui uma clara divisão e organização de módulos e classes que já ajudam a manter um código organizado e limpo.

Para ferramentas relacionadas à persistência de dados, pesquisei e encontrei o Prisma ORM, que é bastante utilizado em conjunto com o NestJS. Por questões de simplicidade, escolhi usar SQLite como banco de dados. A configuração inicial do Prisma com SQLite foi simples, embora tive que pesquisar em sua própria documentação. Entretanto tive alguns problemas ao executar as migrações e o script de seeding. Isso se resolveu ao modificar o modo como as configurações do Prisma estavam definidos (aparentemente algo nas últimas atualizações não está funcionando muito bem, e, como há muito mais material na internet com as configurações de versões anteriores, resolvi adotar o que achei com mais conteúdo na internet).

O projeto contém 4 projetos NestJS, um para cada microserviço e um para o api gateway.

Foi seguida a documentação do NestJS para a criação do api gateway e arquitetura e comunicação entre o api gateway e os outros micro serviços.

Como a comunicação entre os micro serviços é por TCP, foi necessário mudar um pouco a forma como os endpoints dos microserviços estavam definidos (já que antes tudo foi pensado com base Restful para HTTP)

Links de referência:

<https://www.prisma.io/docs/getting-started/quickstart-sqlite>

<https://www.prisma.io/docs/orm/prisma-migrate/workflows/seeding>

<https://docs.nestjs.com/>

<https://docs.nestjs.com/microservices/basics>

Informações de como executar o projeto

(as mesmas também estão disponíveis no README.MD no projeto e no github

<https://github.com/giovannabmarinho/projeto-back-end>):

Abra 4 terminais diferentes para executar os 4 projetos que existem nesse projeto.

No primeiro terminal:

1. Vamos executar o serviço-gestao:

```
cd servico-gestao
```

2. Instale as dependências:

```
npm install
```

3. Execute as migrações do Prisma:

```
npx prisma migrate deploy
```

4. (Opcional) Popule o banco com dados iniciais:

```
npx prisma db seed
```

5. Execute o projeto

```
npm run start:dev
```

No segundo terminal:

1. Vamos executar o serviço-planos-ativo:

```
cd servico-planos-ativos
```

2. Instale as dependências:

```
npm install
```

3. Execute o projeto

```
npm run start:dev
```

No terceiro terminal:

1. Vamos executar o serviço-faturamento:

```
cd servico-faturamento
```

2. Instale as dependências:

```
npm install
```

3. Execute o projeto

```
npm run start:dev
```

No quarto terminal:

1. Vamos executar o api-gateway:

```
cd api-gateway
```

2. Instale as dependências:

```
npm install
```

3. Execute o projeto

```
npm run start:dev
```

O servidor estará disponível em <http://localhost:3000>