1. NestJS Criar aplicação

- **Criando projeto no terminal**

```
npx nest new backend
```

- **Entrar no diretório**

```
cd backend
```

- **No diretório src/ apagar os arquivos**
  - app.controller.spec.ts
  - app.service.ts
  - app.controller.ts

2. NestJS (atualizar)

- **src/app.module.ts**

```
import { Module } from '@nestjs/common';

@Module({
        imports: [],
        controllers: [],
        providers: [],
})
export class AppModule {}
```

3. NestJS Permitir o cors

- **No diretório src/main.ts**

```
import { NestFactory } from '@nestjs/core';
import { AppModule } from './app.module';

async function bootstrap() {
	const app = await NestFactory.create(AppModule, {cors: true});
	await app.listen(process.env.PORT ?? 3000);
}
bootstrap();
```

4. Prisma ORM Instalar

- **No diretório backend**

```
npm install prisma -D
```

- **4.2 Método de conexão banco de Dados**

```
npx prisma init --datasource-provider sqlite
```

- **4.3 Dentro da pasta prisma/shema.prisma**

```
model Produto{
    id              Int             @id @default(autoincrement())
    nome            String          @unique
    descricao       String
    preco           Float
    createdAt       DateTime        @default(now())
    updatedAt       DateTime        @updatedAt
    @@map("produto")
}
```

- **Observação Arquivo .env gerando**
- **Automaticamente configurar de acordo com o banco**

Opcional Caso der Error

```
npm i prisma@6.1.0 -D --silent
```

Rodando nosso primeiro migrate criando a tabela

```
npx prisma migrate dev
```

NestJS criando no nosso modulo de conexão

- **Criando conexão com o banco de dados**

```
npx nest g module db
```

- **Entrar na pasta db**

```
cd src/db
```

- **Criar o service para o prisma**

```
npx nest g service prisma --flat --no-spec
```

- **Entrar na src/db/db.module.ts editar**

```
import { Module } from '@nestjs/common';
import { PrismaService } from './prisma.service';

@Module({
	providers: [PrismaService],
	exports: [PrismaService]
})
export class DbModule {}
```

- **Editar o Service do Prisma**

- **Entrar na src/db/prisma.service.ts**

```
import { Global, Injectable, OnModuleInit } from '@nestjs/common';
import { PrismaClient } from '@prisma/client';

@Global()
@Injectable()
export class PrismaService extends PrismaClient implements OnModuleInit {
  async onModuleInit() {
    await this.$connect();
  }
}
```

6. NestJS Criar e Implentar Resource Produto

- **Voltar para o diretório do projeto**

```
cd .. cd ..
```

- **Agora sim gerar o resource**

```
npx nest g resource produto --no-spec
```

- Agora foi gerado automaticamente os arquivos
  - (Apagar a pasta) Entity
  - (Não mexer) produto.controller.ts
  - produto.module.ts,
  - produto.service.ts
  - Dto

6.1 Vamos começar editando nosso modelo de produto

- create.produto.dto.ts

```
export interface CreateProdutoDto {
	nome: string
	descricao: string
	preco: number
}
```

- **update-produto.dto.ts**

```
import { CreateProdutoDto } from './create-produto.dto';

export interface UpdateProdutoDto extends Partial<CreateProdutoDto> {
	id: number
}
```

- **6.2 Atualizando Modulo**
- **produto.module.ts**

```
import { Module } from '@nestjs/common';
import { ProdutoService } from './produto.service';
import { ProdutoController } from './produto.controller';
import { DbModule } from 'src/db/db.module';

@Module({
	imports: [DbModule],
	controllers: [ProdutoController],
	providers: [ProdutoService],
})
export class ProdutoModule {}
```

6.3 Atualizando Service Produto

- produto.service.ts

```
import { Injectable } from '@nestjs/common';
import { CreateProdutoDto } from './dto/create-produto.dto';
import { UpdateProdutoDto } from './dto/update-produto.dto';
import { PrismaService } from 'src/db/prisma.service';

@Injectable()
export class ProdutoService {
 constructor( private readonly prismaService: PrismaService ){}

 create(createProdutoDto: CreateProdutoDto) {
  return this.prismaService.produto.create({
        data: createProdutoDto,
  });
 }

 findAll() {
        return this.prismaService.produto.findMany();
 }

 findOne(id: number) {
  return this.prismaService.produto.findUnique({
        where: { id },
  });
 }

 update(id: number, updateProdutoDto: UpdateProdutoDto) {
  return this.prismaService.produto.update({
        where: { id },
        data: updateProdutoDto,
  });
 }

 remove(id: number) {
  return this.prismaService.produto.delete({
    where: { id },
  });
 }
}
```

6.4 atualizar src/app.module.ts

```
import { Module } from '@nestjs/common';
import { DbModule } from './db/db.module';
import { ProdutoModule } from './produto/produto.module';

@Module({
	imports: [DbModule, ProdutoModule],
	controllers: [],
	providers: [],
})
export class AppModule {}
```

7. NestJS rodando o projeto

```
npm run start:dev
```

8. Observações Opcionais

- **Conexão com mysql**

```
npx prisma init --datasource-provider mysql
```

- **Nesse exemplo de conexão com mysql**
- **Exemplo 1**

usuario:root,sem senha@localhost:3306/banco__criado__ou__criar

```
DATABASE_URL="mysql://root:@localhost:3306/api__02"
```

- **Exemplo 2**

```
DATABASE_URL="mysql://root:@localhost:3306/appi0101"
```

- **Opcional caso queira dar uma olhadinha usando o prisma studio**

```
npx prisma studio
```