Aula 3.5 – Introdução ao NestJS e Ferramentas de Apoio

> **Tema da Aula:** Conhecendo o framework NestJS e as ferramentas essenciais para criar microserviços escaláveis com Node.js e TypeScript.\ **Duração sugerida:** 3 horas (exposição teórica com demonstrações práticas).\ **Objetivo geral:** Compreender a estrutura e a proposta do NestJS, preparar o ambiente de desenvolvimento e construir a primeira API funcional.

🌟Objetivos da Aula

- Entender o papel do NestJS na construção de microserviços.
- Conhecer os conceitos de **módulos, controladores, serviços** e **injeção de dependência**.
- Aprender a usar o Nest CLI para criar e estruturar projetos.
- Preparar o ambiente com Docker e PostgreSQL.
- Criar uma **rota de status** `/status`.
- Testar a API com Postman ou Insomnia.
- Explorar o Swagger gerado automaticamente.

🌷 Parte 1 – O que é o NestJS?

🌧️Contexto

NestJS é um framework progressivo para construção de aplicações back-end modernas com Node.js e TypeScript. Inspirado em Angular, oferece:

- Modularidade
- Tipagem forte
- Arquitetura em camadas
- Facilidade de testes e escalabilidade

🌧️Vantagens para este projeto

- Baseado em **TypeScript**
- Arquitetura **modular e escalável**
- Integração com PostgreSQL, Docker, Prisma, Swagger, etc.
- Estrutura clara e reutilizável
- CLI poderoso para gerar código com padrões consistentes

🐒 Parte 2 – Conceitos-chave do NestJS

Conceito	Descrição
Módulos (`@Module`)	Agrupam funcionalidade por contexto
Controladores (`@Controller`)	Exposição de rotas HTTP
Serviços (`@Injectable`)	Lógica de negócio
Decorators	`@Get`, `@Post`, `@Param`, `@Body`
Injeção de dependência	Cria e injeta serviços automaticamente
DTOs	Validação e tipagem de dados de entrada
Pipes, Guards	Validação, segurança e regras

Parte 3 – Criando o primeiro microserviço

🌵 Instalação do Nest CLI

```
npm install -g @nestjs/cli
```

✨ Criando o projeto

```
nest new auth-service
```

🌍 Adicionando rota `/status`

```
// src/app.controller.ts
import { Controller, Get } from '@nestjs/common';

@Controller()
export class AppController {
  @Get('status')
  getStatus() {
    return { status: 'ok' };
  }
}
```

🐾 Rodando localmente

```
npm run start:dev
```

Acesse: http://localhost:3000/status

🐳Parte 4 – Dockerização

Dockerfile

```
FROM node:18-alpine

WORKDIR /app

COPY package*.json ./
RUN npm install

COPY . .

RUN npm run build

CMD ["node", "dist/main.js"]
```

.dockerignore

```
node_modules
dist
```

Build e execução

```
docker build -t auth-service .
docker run -p 3000:3000 auth-service
```

🐃Parte 5 – Ferramentas complementares

PostgreSQL com Docker Compose

```
version: '3.9'
services:
  db:
    image: postgres
    environment:
      POSTGRES_USER: postgres
      POSTGRES_PASSWORD: postgres
      POSTGRES_DB: authdb
    ports:
      - "5432:5432"
    volumes:
      - pgdata:/var/lib/postgresql/data
```

```
volumes:
  pgdata:
```

Swagger

```
npm install --save @nestjs/swagger swagger-ui-express
```

Em `main.ts`:

```
import { SwaggerModule, DocumentBuilder } from '@nestjs/swagger';

const config = new DocumentBuilder()
  .setTitle('Auth API')
  .setDescription('API de autenticação')
  .setVersion('1.0')
  .build();

const document = SwaggerModule.createDocument(app, config);
SwaggerModule.setup('api', app, document);
```

Acesse: http://localhost:3000/api

🌞Parte 6 – Atividades da Aula

🙊Atividade 1

- Criar o projeto com NestJS.
- Criar a rota `/status`.
- Dockerizar o serviço.
- Subir no GitHub com README.
- Adicionar Swagger.

🙊Atividade 2

- Responder em equipe:
- O que são controladores, serviços e módulos?
- Como funciona a injeção de dependência no NestJS?
- Quais etapas são necessárias para expor uma rota?

Reflexão Final

- Por que um framework como o NestJS acelera o desenvolvimento de sistemas complexos?
- Como o TypeScript contribui para a segurança do código?
- Como a modularidade ajuda no trabalho em equipe e na manutenção de sistemas?
