Atividade: "Do zero ao endpoint — NestJS com IA, Docker, DB e Swagger"

Objetivo

Consolidar, com apoio de IA, os tópicos: **dockerização**, **PostgreSQL**, **TypeORM**, **variáveis de ambiente**, **validação com pipes/DTOs** e **documentação com Swagger**, produzindo artefatos e um microsserviço funcional.

Parte 1 — Organização e Preparação com IA (em grupos)

Guia de estudo + glossário com IA

Peçam à IA:

- Um **guia rápido** dos passos de ambiente: .dockerignore → build/run Docker → **docker-compose** → **PostgreSQL 14** → **TypeORM** → .env seguro → **Swagger** → **ValidationPipe**.
- Um **glossário** com termos e trechos do material: *ValidationPipe (whitelist/forbidNonWhitelisted/transform)*, *DTO*, *ParseUUIDPipe*, *ApiProperty/ApiResponse*, *ConfigModule.forRoot*, *TypeOrmModule.forRootAsync*.

Dica de verificação crítica (já nesta etapa): o guia cita **não versionar** .env e sugiere gerenciadores de segredos? Se não, corrijam.

Mapa conceitual com IA

Estruturem ramos com: **Dockerização** → **Compose** → **DB (PostgreSQL 14)** → **TypeORM + ConfigService/.env** → **Validation global** → **DTOs com class-validator** → **Swagger** (título, tag, Bearer, rota /api). Validem cada ramo com trechos do material.

Geração e avaliação crítica de código com IA

Peçam à IA **códigos prontos** para:

- **Docker**
  - .dockerignore, Dockerfile, docker-compose.yml com **api + db** (PostgreSQL 14, volume, porta 5432; API em 3000 com start:dev). Depois validem com o material.
- **TypeORM + .env**
  - TypeOrmModule.forRootAsync com ConfigService e synchronize **só em dev**, mais .env e env_file no compose. Conferir chaves/valores.
- **Validation/DTO/Controller**
  - main.ts com ValidationPipe (whitelist, forbidNonWhitelisted, transform) e **Swagger** configurado (título, tag, Bearer, /api).
  - DTO com class-validator e @ApiProperty; controller com @Post, @Get(':id') e ParseUUIDPipe.

Checklist de teste:

- docker-compose up -d sobe API e DB?
- Swagger abre em http://localhost:3000/api?
- POST /users valida e retorna 400 quando DTO falha?
- .env está **fora** do versionamento?

Parte 2 — Seminário (≈ 15 min por grupo)

Demonstrem os artefatos (guia/glossário, mapa conceitual e trechos de código), expliquem **o que a IA acertou/errou** e as **correções** que vocês aplicaram. Foquem em:

ValidationPipe

Se a IA configurou **ValidationPipe** corretamente (whitelist/forbid/transform).

Swagger

Se o **Swagger** veio com **Bearer auth** e rota /api.

TypeORM

Se o **TypeORM** ficou **assíncrono** com ConfigService e synchronize só em dev.

Controller e DTOs

Se o **controller** aplica ParseUUIDPipe e os **DTOs** têm validadores/documentação.

Estrutura de apresentação e espírito crítico inspirados no modelo do arquivo 016 (mapas/quiz + análise do que a IA gerou).

Entregas

1 **Mapa conceitual**

(PNG/PDF)

2 **Guia de estudo + glossário**

(PDF/MD)

3 **Repositório**

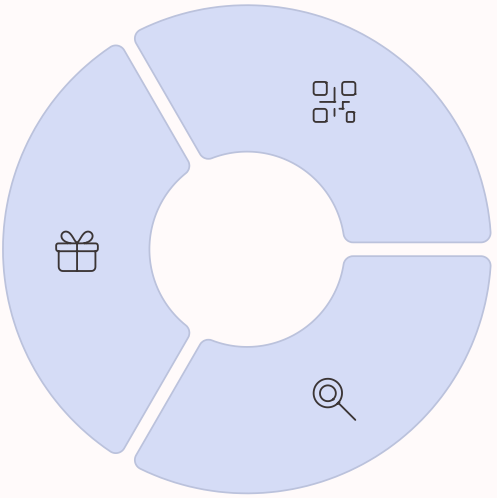
com Dockerfile, docker-compose.yml, .dockerignore, .env.example, main.ts (Validation + Swagger), app.module.ts (TypeORM), DTOs e Controller de users.

Critérios de Avaliação (rubrica)

Clareza da apresentação

(estrutura, tempo, didática)

30%



Qualidade dos artefatos

(mapa coerente, guia útil, código executando em Docker)

40%

Análise crítica

do que a IA gerou (ajustes em .env, Swagger, DTOs, pipes, compose)

30%

Sugestão de pesos: 30% apresentação | 40% código+deploy | 30% análise crítica.

Prompts prontos (copie e cole para a IA)

Compose + DB

"Crie um docker-compose.yml para um serviço NestJS (api) e um **PostgreSQL 14** (db) com volume persistente, portas **3000** e **5432**, variável DATABASE_URL e command: npm run start:dev para a API."

TypeORM + .env

"Implemente TypeOrmModule.forRootAsync usando ConfigService, busque DB_HOST/DB_PORT/DB_USER/DB_PASSWORD/DB_NAME do .env, e deixe synchronize **ativo apenas em desenvolvimento**."

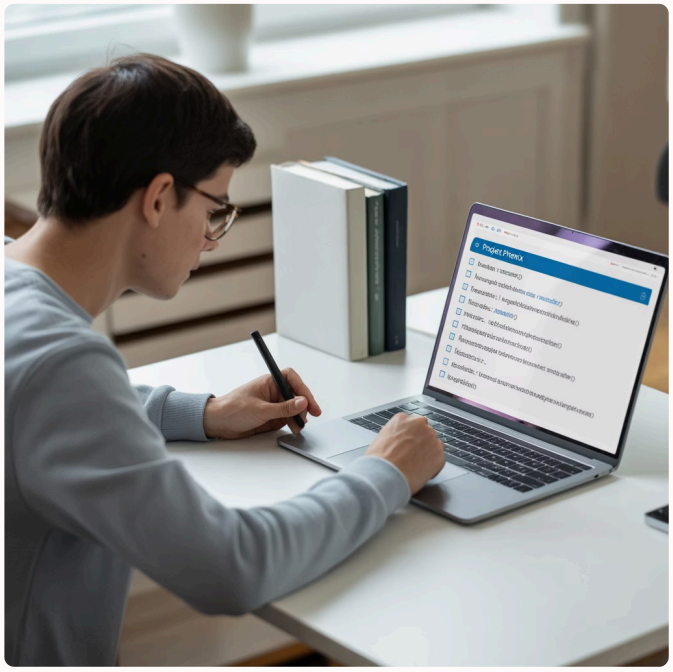
Validation + Swagger (main.ts)

"Configure ValidationPipe global com whitelist, forbidNonWhitelisted e transform; adicione Swagger com **título**, **tag** e **Bearer auth** publicado em /api."

DTO + Controller

"Crie CreateUserDto com @ApiProperty, @IsNotEmpty, @IsEmail, @MinLength e @Matches; desenvolva UsersController com @Post() (usa DTO) e @Get(':id') com ParseUUIDPipe."

Checklist final (rápido)



docker-compose up -d inicializa **API + DB**?

.env **não** versionado; use .env.example.

TypeORM **assíncrono** com ConfigService (valores vindos do .env).

ValidationPipe aplicado globalmente.

Swagger acessível em /api com **Bearer auth**.

DTOs e Controllers **documentados e validados** (inclui ParseUUIDPipe).

Tempo sugerido

25 min

guia/glossário com IA

50–60 min

geração/ajustes de código + testes

1 2 3 4

20 min

mapa conceitual

15 min

seminário por grupo