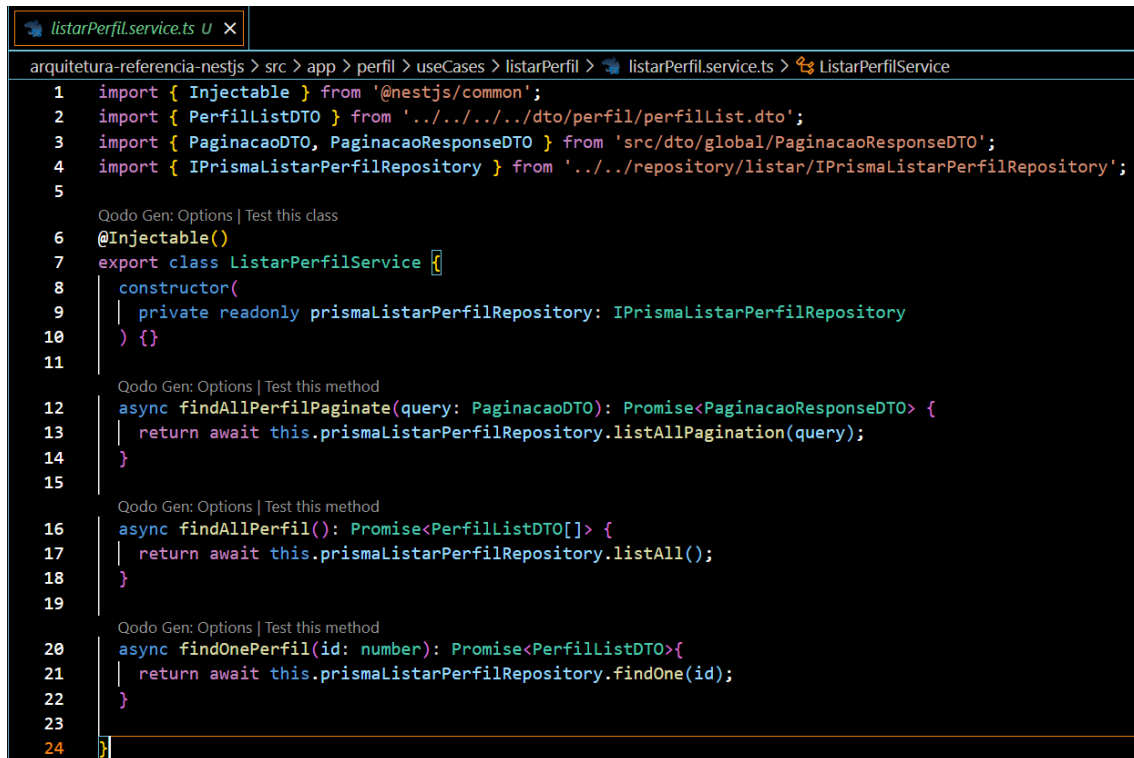


Aplicando SOLID no Nestjs

1. S - Princípio da Responsabilidade Única (Single Responsibility Principle - SRP)

Cada classe deve ter uma única responsabilidade.

No NestJS, esse princípio é facilmente aplicado separando as responsabilidades em **módulos**, **serviços**, **controladores** e **repositórios**.



```
1 import { Injectable } from '@nestjs/common';
2 import { PerfilListDTO } from '...../dto/perfil/perfillist.dto';
3 import { PaginacaoDTO, PaginacaoResponseDTO } from 'src/dto/global/PaginacaoResponseDTO';
4 import { IPrismaListarPerfilRepository } from '...../repository/listar/IPrismaListarPerfilRepository';
5
6 @Injectable()
7 export class ListarPerfilService {
8   constructor(
9     | private readonly prismaListarPerfilRepository: IPrismaListarPerfilRepository
10  ) {}
11
12   async findAllPerfilPaginate(query: PaginacaoDTO): Promise<PaginacaoResponseDTO> {
13     | return await this.prismaListarPerfilRepository.listAllPagination(query);
14   }
15
16   async findAllPerfil(): Promise<PerfilListDTO[]> {
17     | return await this.prismaListarPerfilRepository.listAll();
18   }
19
20   async findOnePerfil(id: number): Promise<PerfilListDTO>{
21     | return await this.prismaListarPerfilRepository.findOne(id);
22   }
23
24 }
```

Aqui, o ListarPerfilService é responsável apenas pela lógica de negócios relacionada a perfil.

2. O - Aberto/Fechado (Open/Closed Principle - OCP)

Uma classe deve estar aberta para extensão, mas fechada para modificação.

No NestJS, você pode usar **injeção de dependência** para facilitar a substituição de implementações.

Exemplo: Um repositório pode ser facilmente substituído usando **interfaces**:

```
TS IPismaListarPerfilRepository.ts U X
arquitetura-referencia-nestjs > src > app > perfil > repository > listar > TS IPismaListarPerfilRepository.ts > IPismaListarPerfilRepository
1 import { PaginacaoDTO, PaginacaoResponseDTO } from "src/dto/global/PaginacaoResponseDTO";
2 import { PerfilListDTO } from "src/dto/perfil/perfilList.dto";
3
4 Qodo Gen: Options | Test this class
5 export abstract class IPismaListarPerfilRepository {
6   abstract listAllPagination(data: PaginacaoDTO): Promise<PaginacaoResponseDTO>
7   abstract listAll(): Promise<PerfilListDTO[]>
8   abstract findOne(id: number): Promise<PerfilListDTO>
9 }
```

Implementação para Prisma:

```
TS PrismaListarPerfilRepository.ts U X
arquitetura-referencia-nestjs > src > app > perfil > repository > listar > TS PrismaListarPerfilRepository.ts > ...
8 @Injectable()
9 export class PrismaListarPerfilRepository implements IPismaListarPerfilRepository {
10
11   constructor(private readonly prisma: PrismaService) { }
12
13   Complexity is 3 Everything is cool! | Qodo Gen: Options | Test this method
14   async listAllPagination(data: PaginacaoDTO): Promise<PaginacaoResponseDTO> {
15     const { page, pageSize, search } = data;
16     const where = search
17       ? {
18         OR: [
19           { descricao: { contains: search, mode: 'insensitive' as const } },
20         ].filter(Boolean),
21       }
22       : {};
23     const skip = (parseInt(page) - 1) * parseInt(pageSize);
24     const perfis = await this.prisma.perfil.findMany({
25       skip,
26       take: parseInt(pageSize),
27       where,
28       select: {
29         id: true,
30         descricao: true,
31         createdAt: true,
32         updatedAt: true
33       },
34     });
35     const totalRecords = await this.prisma.perfil.count({ where });
36     return { data: perfis, total: totalRecords };
37   }
```

```

38     async listAll(): Promise<PerfilListDTO[]> {
39         return await this.prisma.perfil.findMany({
40             select: QueryPerfil.select()
41         });
42     }
43
44     Qodo Gen: Options | Test this method
45     async findOne(id: number): Promise<PerfilListDTO> {
46         return await this.prisma.perfil.findFirst({
47             where: { id },
48             select: QueryPerfil.select()
49         });
50     }
51 }

```

Registrando no módulo:

```

listarPerfil.module.ts U x
arquitetura-referencia-nestjs > src > app > perfil > useCases > listarPerfil > module > listarPerfil.module.ts > ...
1 import { Module } from "@nestjs/common";
2 import { ListarPerfilController } from "../listarPerfil.controller";
3 import { ListarPerfilService } from "../listarPerfil.service";
4 import { PrismaModule } from "src/app/prisma/module/prisma.module";
5 import { PrismaListarPerfilRepository } from "src/app/perfil/repository/listar/PrismaListarPerfilRepository";
6 import { IPrismaListarPerfilRepository } from "src/app/perfil/repository/listar/IPrismaListarPerfilRepository";
7
8 Qodo Gen: Options | Test this class
9 @Module({
10     imports: [
11         PrismaModule
12     ],
13     controllers: [ListarPerfilController],
14     providers: [
15         ListarPerfilService,
16         {
17             provide: IPrismaListarPerfilRepository,
18             useClass: PrismaListarPerfilRepository
19         }
20     ],
21     exports: [
22         {
23             provide: IPrismaListarPerfilRepository,
24             useClass: PrismaListarPerfilRepository
25         }
26     ]
27 })
28 export class ListarPerfilModule {}

```

3. L - Princípio da Substituição de Liskov (Liskov Substitution Principle - LSP)

Subtipos devem ser substituíveis por seus tipos base sem alterar o funcionamento do sistema.

Isso é alcançado garantindo que as implementações respeitem as interfaces ou contratos.

- Use **interfaces** para garantir que implementações possam ser trocadas sem quebrar o código.

Exemplo: Se você tem uma interface UserRepository, qualquer implementação, como UserRepositoryImpl ou MockUserRepository, deve funcionar sem alterações no serviço que a utiliza.

4. Interface Segregation Principle (ISP)

As classes não devem ser forçadas a depender de métodos que não utilizam.

Defina interfaces específicas para cada necessidade, evitando interfaces "inchadas".

- **Exemplo:**
Em vez de uma interface genérica para repositórios:

```
13 export abstract class IPrismaPerfilRepository {  
14     abstract listAllPagination(data: PaginacaoDTO): Promise<PaginacaoResponseDTO>  
15     abstract listAll(): Promise<PerfillistDTO[]>  
16     abstract findOne(id: number): Promise<PerfillistDTO>  
17     abstract create(descricao: string): Promise<PerfilDTO>  
18     abstract update(dados: PerfilDTO): Promise<void>  
19 }
```

- **Exemplo:**
Crie interfaces menores e específicas:

```
4 export abstract class IPrismaListarPerfilRepository {  
5     abstract listAllPagination(data: PaginacaoDTO): Promise<PaginacaoResponseDTO>  
6     abstract listAll(): Promise<PerfillistDTO[]>  
7     abstract findOne(id: number): Promise<PerfillistDTO>  
8 }  
9
```

5. Dependency Inversion Principle (DIP)

Dependa de abstrações, não de implementações.

No NestJS, isso é facilitado com Inversão de dependência.

```
1 import { Injectable } from '@nestjs/common';
2 import { PerfillistDTO } from '../../dto/perfil/perfillist.dto';
3 import { PaginacaoDTO, PaginacaoResponseDTO } from 'src/dto/global/PaginacaoResponseDTO';
4 import { IPrismaListarPerfilRepository } from '../../repository/listar/IPrismaListarPerfilRepository';
5
6 @Injectable()
7 export class ListarPerfilService {
8   constructor(
9     | private readonly prismaListarPerfilRepository: IPrismaListarPerfilRepository
10   ) {}
11
12   Qodo Gen: Options | Test this method
13   async findAllPerfilPaginate(query: PaginacaoDTO): Promise<PaginacaoResponseDTO> {
14     | return await this.prismaListarPerfilRepository.listAllPagination(query);
15   }
16
17   Qodo Gen: Options | Test this method
18   async findAllPerfil(): Promise<PerfillistDTO[]> {
19     | return await this.prismaListarPerfilRepository.listAll();
20   }
21
22   Qodo Gen: Options | Test this method
23   async findOnePerfil(id: number): Promise<PerfillistDTO>{
24     | return await this.prismaListarPerfilRepository.findOne(id);
25   }
26 }
```

Aqui, o **ListarPerfilService** depende da interface **IPrismaListarPerfilRepository**, não da implementação concreta da classe **PrismaListarPerfilRepository**. Isso facilita a troca de implementações, por exemplo, de **PrismaListarPerfilRepository** para **PrismaListarPerfilRepositoryInMemory**.

Acesso a Aplicação

Login: tech.lead

Senha: Tech@2025