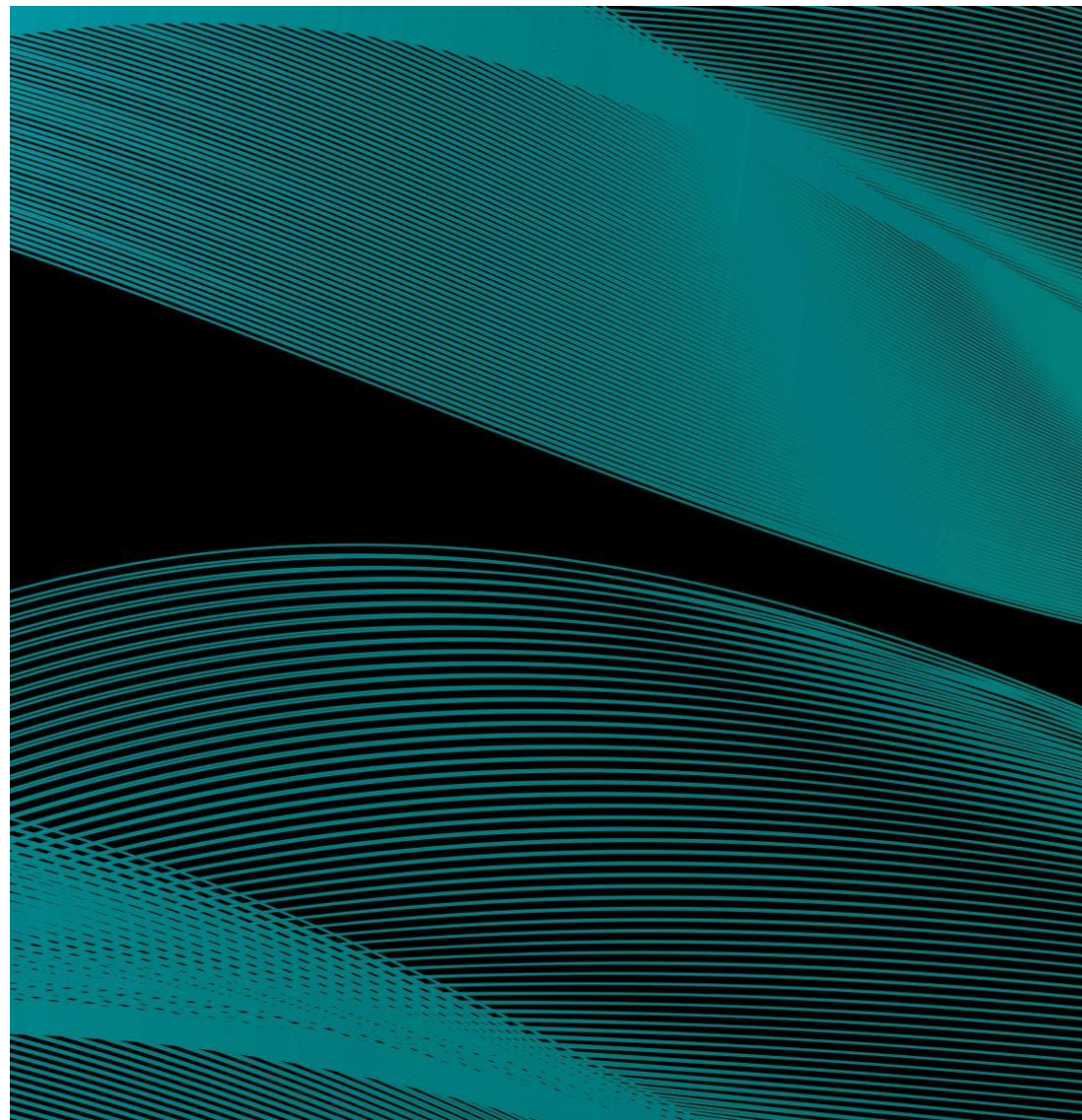

ESTRUTURA BÁSICA DO SPRING BOOT





INJEÇÃO DE DEPENDÊNCIAS



ENTENDENDO A ESTRUTURA DE UM PROJETO SPRING BOOT

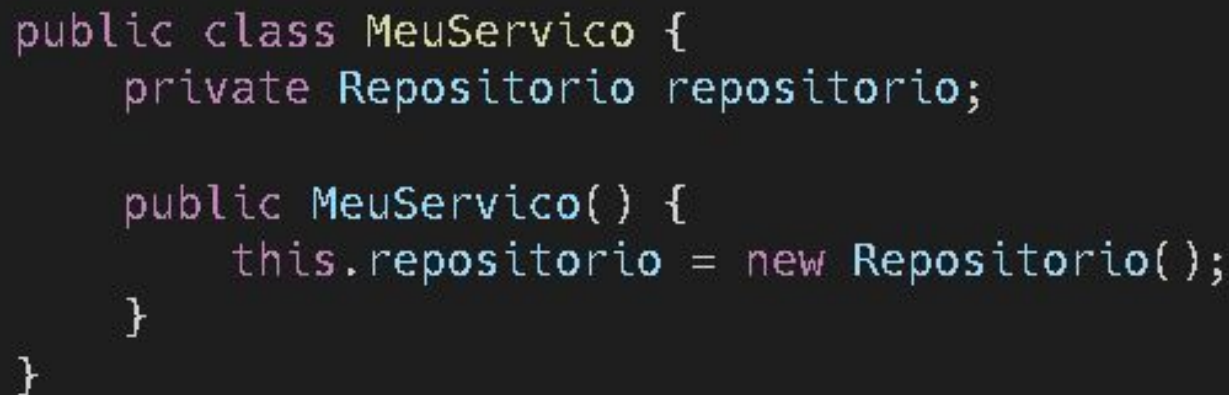
- **src/main/java** - Onde fica o código da aplicação.
 - **src/main/resources** - Arquivos de configuração, como `application.properties`.
 - **src/test/java** - Onde escrevemos nossos testes automatizados.
-

ENTENDENDO A ESTRUTURA DE UM PROJETO SPRING BOOT

- Arquitetura em Camadas
 -  **controller** → Controladores REST
 -  **service** → Regras de negócio
 -  **repository** → Acesso a banco de dados
 -  **model** → Modelos de dados
-

O QUE É INVERSÃO DE CONTROLE (IOC)

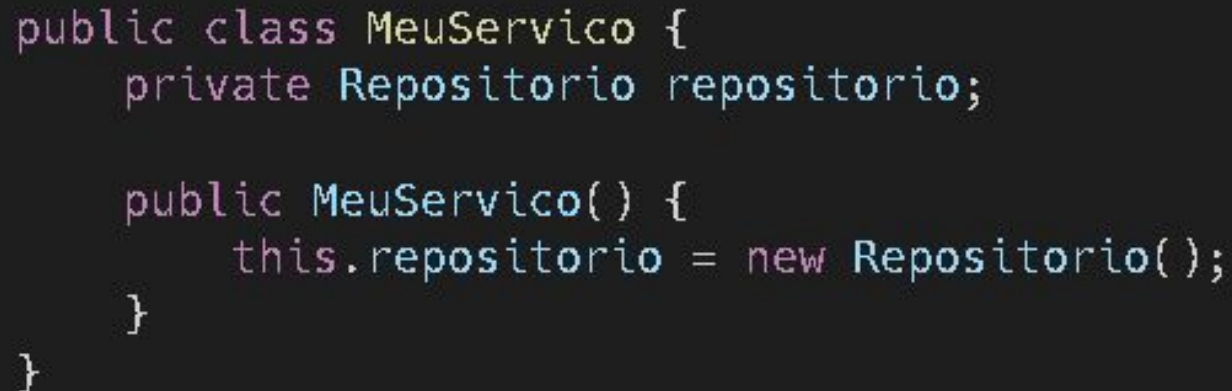
- *Normalmente, quando precisamos de um objeto em Java, fazemos algo assim:*



```
public class MeuServico {  
    private Repositorio repositorio;  
  
    public MeuServico() {  
        this.repositorio = new Repositorio();  
    }  
}
```

O QUE É INVERSÃO DE CONTROLE (IOC)

- Isso significa que a classe *MeuServico* precisa saber como instanciar *Repositorio*. O problema é que isso **acopla fortemente** as classes, dificultando testes e manutenções.



```
public class MeuServico {  
    private Repositorio repositorio;  
  
    public MeuServico() {  
        this.repositorio = new Repositorio();  
    }  
}
```

O QUE É INJEÇÃO DE DEPENDÊNCIAS (DI)

- *A **injeção de dependências** é um mecanismo que permite ao Spring fornecer automaticamente as dependências que um objeto precisa. Com isso, não precisamos mais instanciar os objetos manualmente.*
 - *Vamos ver isso na prática criando um serviço e injetando um repositório nele.*
-

BENEFÍCIOS DA INJEÇÃO DE DEPENDÊNCIAS

- ✓ **Código desacoplado:** As classes não sabem como suas dependências são criadas. Isso facilita mudanças e reuso.
 - ✓ **Facilidade de testes:** Podemos substituir facilmente as dependências reais por mocks em testes.
 - ✓ **Gerenciamento pelo Spring:** O Spring cuida da criação e do ciclo de vida dos objetos.
-