Engenharia de Software

24/11/2025

# TP3

## Microsserviços e DevOps com Spring Boot e Spring Cloud

Professor(a): Flávio da Silva Neves

Samuel Hermany
INSTITUTO INFNET

# LINK GITHUB

https://github.com/faculdade-infnet/V-2-Microsservicos-e-Spring-Cloud/tree/main/TP3

# Parte 1 - IMPLEMENTAÇÃO DE PERSISTÊNCIA DE DADOS COM SPRING DATA JDBC

## 1. O que são serviços de dados?

Serviços que fornecem armazenamento e gerenciamento de dados, podendo ser bancos de dados SQL/NoSQL, filas ou caches.

## 2. Quais as principais propriedades que devem ser consideradas para garantir a escolha da tecnologia mais adequada para usar algum serviço de dados em um sistema nativo de nuvem?

- Escalabilidade
- Resiliência / tolerância a falhas
- Latência e throughput
- Compatibilidade com programação reativa

## 3. Qual a finalidade de usar o Flyway?

- Garantir que o banco evolua de forma controlada entre ambientes
- Permite versionamento
- Executar as migrações automaticamente no início do Spring Boot

# Parte 2 -

## 4. O que é programação reativa?

Modelo que processa dados de forma assíncrona e orientada a eventos.

## 5. Por que a programação reativa é importante para aplicativos nativos da nuvem e como ela difere da programação imperativa?

- Reativa: não bloqueia threads, suporta alta carga, responde a eventos.
- Imperativa: bloqueante, menos escalável, threads ficam esperando respostas.

## 6. Existem duas opções para definir endpoints RESTful em um aplicativo Spring WebFlux. Quais são elas e como funcionam?

1. **Annotation-based (@RestController + @GetMapping, etc.)**

   É o estilo mais próximo do que estamos acostumados em Spring MVC tradicional, mas funciona de forma reativa quando usamos WebFlux.
   Você define controladores usando anotações como @RestController, @GetMapping, @PostMapping, etc.
   Os métodos retornam tipos reativos, como Mono<T> ou Flux<T>:

   - Mono → zero ou um elemento (ex.: buscar um usuário).

- Flux → zero ou vários elementos (ex.: listar todos os usuários).

2. **Functional endpoints (RouterFunction + HandlerFunction)**
Aqui você define routers e handlers de forma funcional, sem anotações. É mais modular, ideal para APIs muito reativas e com linhas de processamento mais sofisticados.
   - RouterFunction → define a rota HTTP e associa a um handler.
   - HandlerFunction → implementa a lógica do endpoint e retorna Mono ou Flux.

# Parte 3 - Implementação de clientes reativos com WebClient

## 7. O Spring Framework vem com dois clientes que executam requisições HTTP. Quais são eles e como eles funcionam?

1. **RestTemplate → bloqueante, tradicional**

**Funcionamento:** Quando você faz uma requisição com RestTemplate, a thread que fez a chamada fica esperando a resposta do servidor. Nada mais pode ser feito nessa thread enquanto a resposta não chega.

2. **WebClient → não bloqueante, reativo**

**Funcionamento:** WebClient não bloqueia a thread enquanto espera a resposta. Ele retorna um Mono ou Flux (tipos reativos do Project Reactor) que representam a resposta futura.
A thread pode continuar processando outras tarefas e só reage quando a resposta chega.

## 8. Quais as principais diferenças entre o RestTemplate e o WebClient?

- RestTemplate é síncrono, WebClient é assíncrono/reactivo
- WebClient suporta Flux e Mono, melhor integração com WebFlux

# Parte 4 - Testes aplicações reativas com Spring e Testcontainers

## 9. O que é o Testcontainers?

Biblioteca para testes de integração usando containers Docker.

## 10. Quando é recomendado usar o Testcontainers?

Em Testes que precisam de banco real, Kafka, Redis ou outros serviços externos.

## 11. Implemente uma aplicação seguindo os princípios reativos, adicione persistência usando o Spring Data JDBC, use o cliente WebClient para fazer as requisições HTTP e faça os testes usando Testcontainers.

https://github.com/faculdade-infnet/V-2-Microsservicos-e-Spring-Cloud/tree/main/TP3/TP3-projeto

12. Para a aplicação solicitada, vocês devem usar o conceito de microsserviços e seus principais recursos, como Docker e Kubernetes.

https://github.com/faculdade-infnet/V-2-Microsservicos-e-Spring-Cloud/tree/main/TP3/TP3-projeto