

Tópicos em Frameworks

Micro Serviços - Cássio Seffrin 2019

<https://github.com/cassioseffrin/projetoAulaSpring>

Micro Serviços

CARACTERÍSTICAS

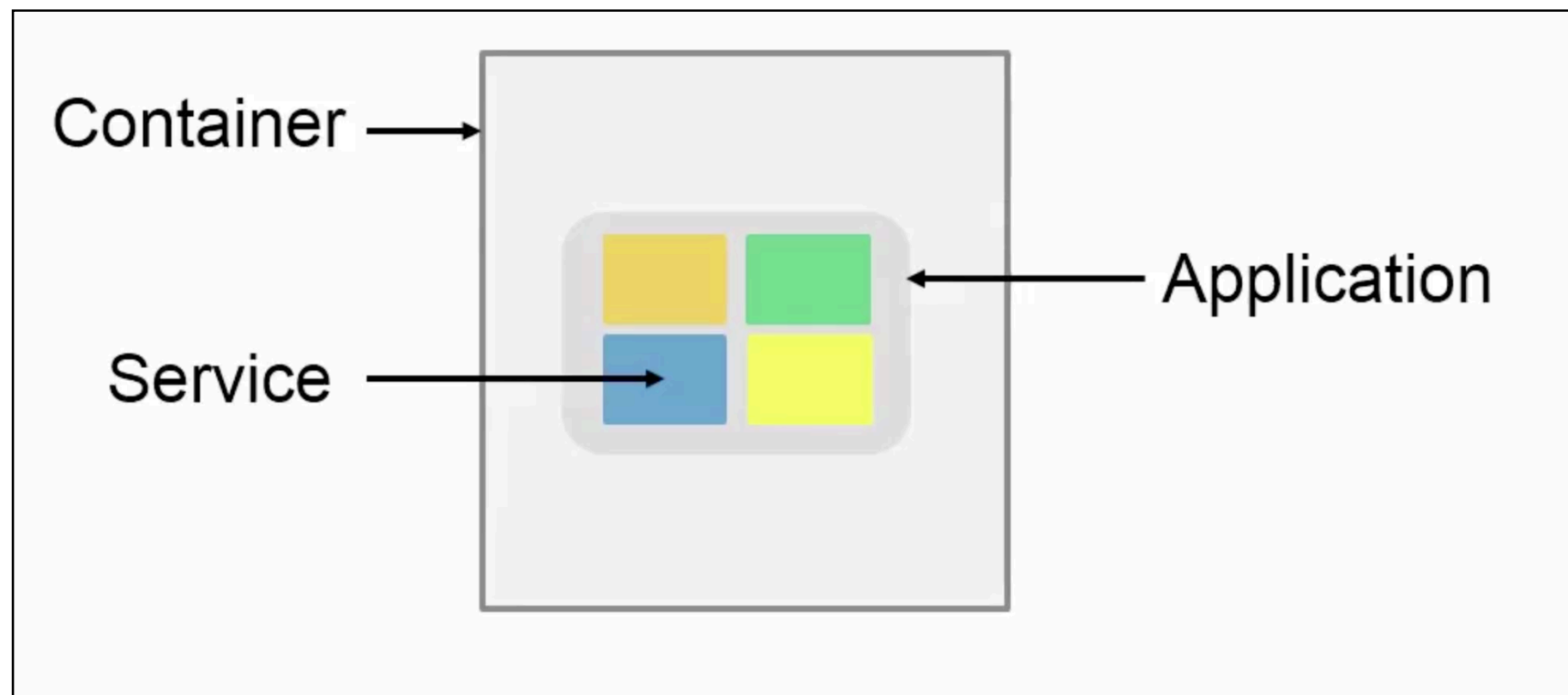
- Pequenos serviços independentes organizados em torno dos recursos de negócios
- O APIS expõe o serviço e seus recursos
- Os serviços interagem em diferentes processos
- Os armazenamentos de dados são focados e específicos para o serviço
- Sem estado
- Tolerantes a Falha

Micro serviços

BENEFÍCIOS

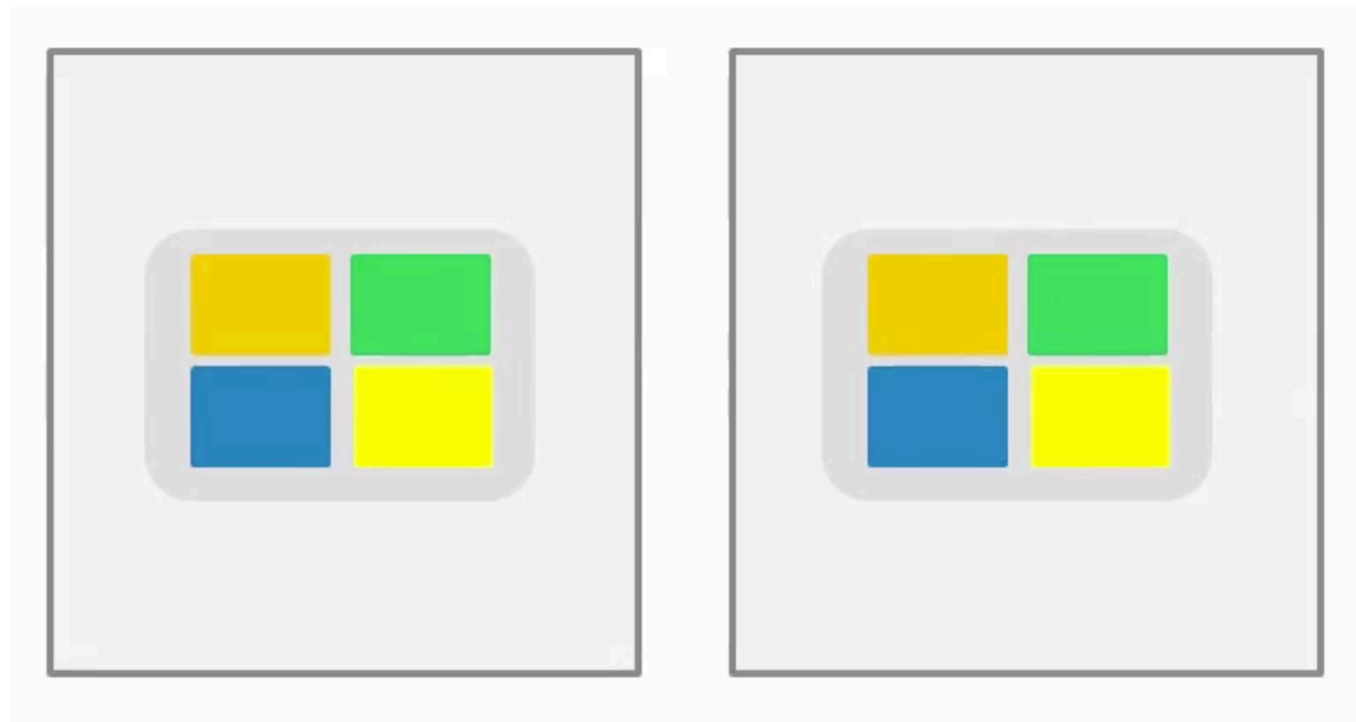
- Separa os recursos de negócios
- Os serviços são fracamente acoplados e modulares
- Altamente escalável e substituível
- Gerenciamento de serviços independente
- Suporta processamento simultâneo
- Permite equipes menores e menor tempo de colocação no mercado

Monolitico vs micro servico



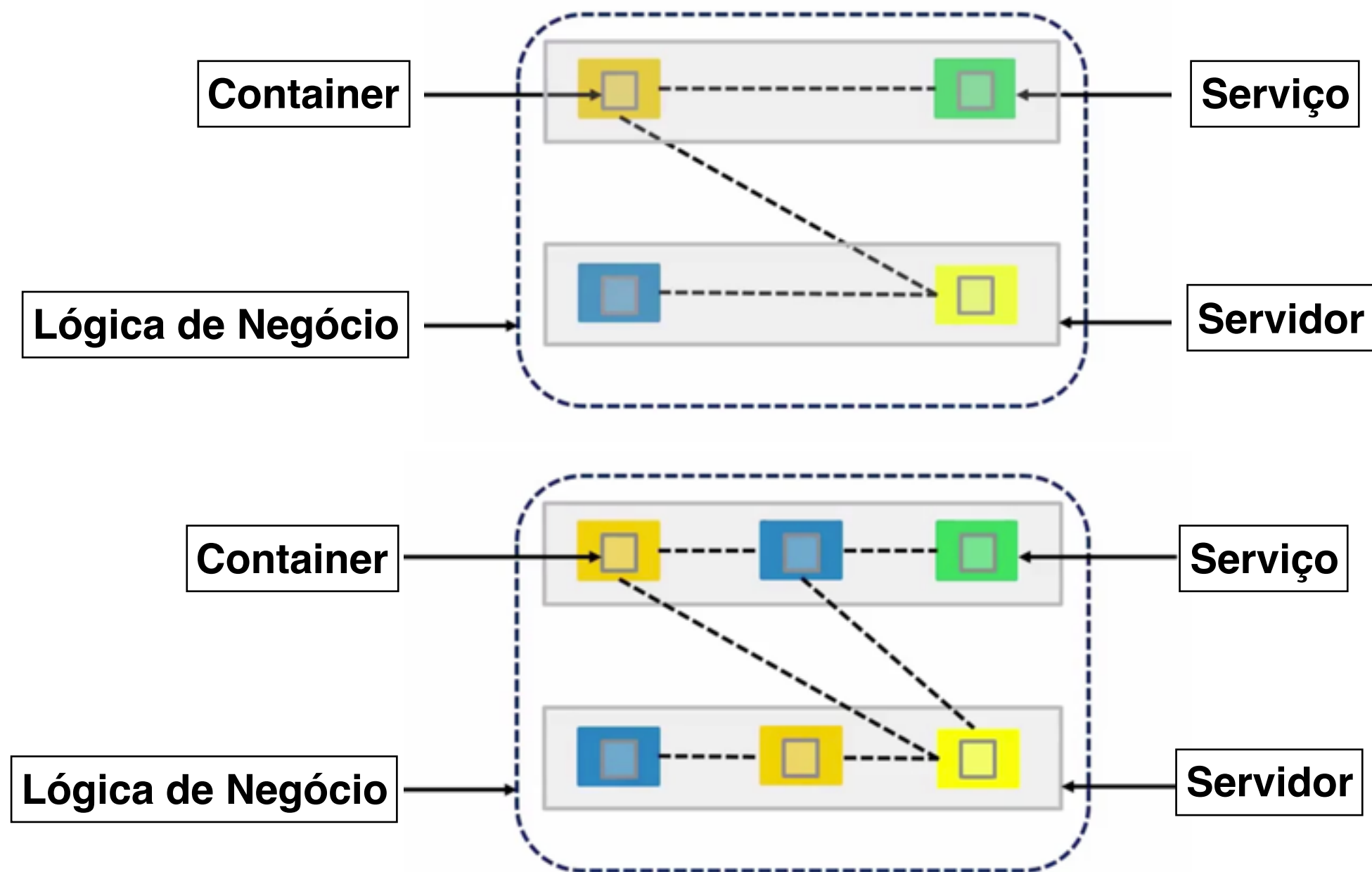
Monolítico como escalar

1. CPU, memória etc..
2. Horizontalmente , dificuldade de manipular sessões



Escalando micro serviços

Definição: Um projeto de software que foca na integração de um grupo de serviços leves e isolados através de comunicação em camadas para formar uma aplicação.



Micro serviços

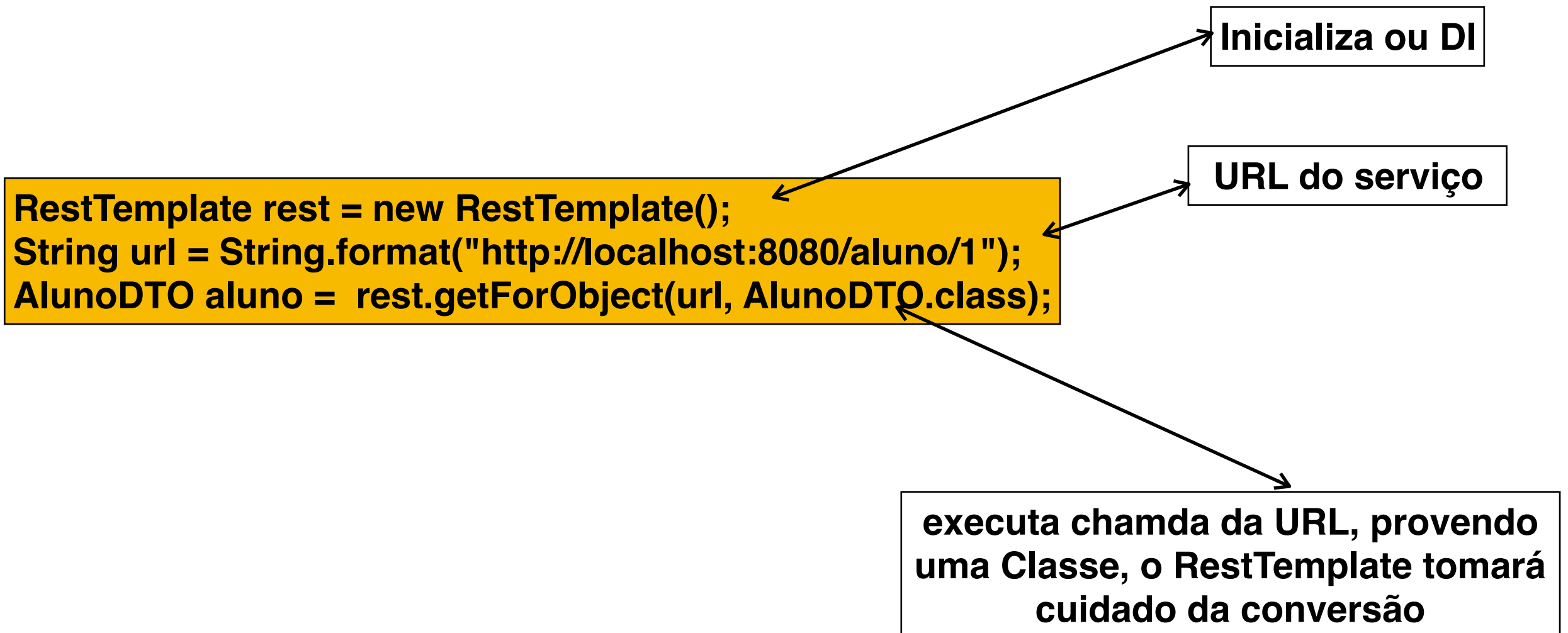
- A arquitetura de micro serviços sugere que cada serviço deve manipular seus próprios dados. Portanto, qualquer serviço (Serviço A) dependente de dados pertencentes a outro serviço (serviço B) deve acessar esses dados não fazendo chamadas diretas ao banco de dados, mas através da API fornecida pelo segundo serviço (serviço B).
- Então, Como lidar com relacionamentos de chave estrangeira?

Micro serviços

- É possível usar um banco de dados compartilhado para vários micro serviços.
Os padrões para gerenciamento de dados de microserviços :
<http://microservices.io/patterns/data/database-per-service.html>.
- Micro serviços mais autônomos. Nessa situação, você deve duplicar alguns dos seus dados entre vários micro serviços. Você pode compartilhar os dados com chamadas da API entre micro serviços ou com mensagens assíncronas. Depende da sua infraestrutura e frequência de alteração dos dados. Se não estiver mudando com frequência, você deve duplicar os dados com eventos assíncronos.

Micro serviços

O Spring Rest Template provê uma forma bem fácil de chamar APIs REST



Micro serviços

REST Template

Método tradicional usando RestTemplate

```
@GetMapping("/pegarDadosFinanceiro/{id}")
public @ResponseBody AlunoDTO pegarDados(@PathVariable Long id) {
    RestTemplate rest = new RestTemplate();
    String url = "http://localhost:8080/aluno/" + id;
    AlunoDTO aluno = rest.getForObject(url, AlunoDTO.class);
    return aluno;
}
```

Micro serviços

Spring Cloud Feign

- Cliente REST Declarativo da Netflix
- Chamando serviços REST usando as bibliotecas Feign
- O que é o Feign ?
- Alternativa para o REST Template

Micro serviços

Spring Cloud Feign

Como funciona?

Definir interfaces para o código do cliente REST

- Anotar interface como `@FeignClient`
- Anotar assinaturas de métodos com anotações do Spring MVC
- Outras implementações como JAX/RS plugavel

O Spring Cloud o implementará em tempo de execução

- Escaneia as interfaces
- Implementa automaticamente o código para chamar o serviço REST e processar resposta

Micro serviços

Spring Cloud Feign

Anotar a aplicação com @EnableFeignClients

```
@SpringBootApplication
@EnableFeignClients("edu.cassio.escolafinanceiro")
public class EscolaFinanceiroApplication {

    public static void main(String[] args) {
        SpringApplication.run(EscolaFinanceiroApplication.class, args);
    }

}
```

Micro serviços

Feign Interface

Criar uma interface (nao classe)

```
@FeignClient(name = "aula", url = "localhost:8080")
public interface EscolaMatriculaProxy {

    @GetMapping("/aluno/{id}")
    public AlunoDTO pegarDados(@PathVariable("id") Long id);

}
```

Criar uma classe Controller

```
@Autowired
private EscolaMatriculaProxy escolaMatriculaProxy;

@GetMapping("/pegarDadosFinanceiro-feign/{id}")
public @ResponseBody AlunoDTO pegarDadosFeign(@PathVariable Long id) {
    AlunoDTO aluno = escolaMatriculaProxy.pegarDados(id);
    return aluno;
}
```

Micro serviços

Spring Cloud Feign

Dependências Spring Initializr

- Spring Web
- Cloud Bootstrap
- Lombok
- MySQL Driver
- Spring Data JPA
- Config Client

Micro serviços

Spring Cloud Feign

Ou adicionar as seguintes dependências no pom.xml

```
<dependency>
  <groupId>org.springframework.cloud</groupId>
  <artifactId>spring-cloud-starter</artifactId>
</dependency>
<dependency>
  <groupId>org.springframework.cloud</groupId>
  <artifactId>spring-cloud-starter-config</artifactId>
</dependency>
<dependency>
  <groupId>org.springframework.cloud</groupId>
  <artifactId>spring-cloud-starter-openfeign</artifactId>
</dependency>
<dependencyManagement>
  <dependencies>
    <dependency>
      <groupId>org.springframework.cloud</groupId>
      <artifactId>spring-cloud-dependencies</artifactId>
      <version>${spring-cloud.version}</version>
      <type>pom</type>
      <scope>import</scope>
    </dependency>
  </dependencies>
</dependencyManagement>
```


Micro serviços

Eureka

- Serviço Descoberta
- Eureka Server
- Descoberta do cliente

Micro serviços

Eureka

- Serviço de Descoberta : As arquiteturas de micro serviço resultam em um grande número de chamadas entre serviços
- Muito desafiador para configurar
- Como um aplicativo pode encontrar facilmente todas as outras dependências de tempo de execução?
- Configuração manual
- A descoberta de serviços prove um single lookup service (único serviço de 'pesquisa')
- Os clientes se registram, descobrem outros registrantes.
- Soluções: Eureka, PaceMaker, Etcd, Zookeeper, SmartStack, etc.

Micro serviços

Eureka descoberta de serviço cliente/servidor

- Parte do Spring Cloud Netflix testada pelo Netflix
- Eureka fornece um servidor de 'pesquisa'.
Geralmente tornado altamente disponível executando várias cópias
As cópias replicam o estado dos serviços registrados.
- Os Serviços "Cliente" são registrados no Eureka
Fornece metadados no host, porta, URL do indicador de integridade, etc.
- Os Serviços de Cliente enviam pulsações (heartbeats) para o Eureka
Eureka remove os serviços sem heartbeats.

Micro serviços

Servidor Eureka

É uma simples aplicação spring boot web com uma dependência e uma anotação

```
<dependency>  
    <groupId>org.springframework.cloud</groupId>  
    <artifactId>spring-cloud-starter-netflix-eureka-server</artifactId>  
</dependency>
```

```
@SpringBootApplication  
@EnableEurekaServer  
public class EurekaServerApplication {  
  
    public static void main(String[] args) {  
        SpringApplication.run(EurekaServerApplication.class, args);  
    }  
  
}
```

Micro serviços

Servidor Eureka - múltiplos servidores

- Tipicamente vários servidores Eureka devem ser executados simultaneamente.
Caso contrário, você receberá avisos nos logs
Os servidores Eureka se comunicam entre si para compartilhar o estado
Fornece alta disponibilidade
- Cada o servidor deve saber a URL dos outros
Pode ser fornecido pelo server Config
Server One (JAR), vários perfis (profiles)



Micro serviços

Servidor Eureka - múltiplos servidores

Opções de configuração comuns para o Eureka Server:

<https://github.com/Netflix/eureka/wiki/Configuring-Eureka>

application.yml

```
server:
  port: 8090

eureka:
  client:
    register-with-eureka: false
    fetch-registry: false
    service-url: http://localhost:8091/eureka, http://localhost:8092/eureka

  instance:
    status-page-url-path: ${management.context Path}/info
    health-check-url-path: ${management.contextPath/health
    hostname: localhost
```

bootstrap.yml

```
spring:
  application:
    name: eureka-server
```

Micro serviços

Spring Boot application registrado com Eureka

Dependencia:

```
<dependency>  
  <groupId>org.springframework.cloud</groupId>  
  <artifactId>spring-cloud-starter-eureka-server</artifactId>  
</dependency>
```

```
Somente a localizacao do Eureka requer configuração explicita  
@SpringBootApplication  
@EnableDiscoveryClient  
public class Application {  
  
}
```

Default fallback.
Qualquer instância do Eureka (geralmente
algumas URLs separadas por vírgula)

```
#application.properties  
eureka.client.serviceUrl.defaultZone: http://server: 9000/eureka/
```



Micro serviços

@EnableDiscoveryClient

Registra automaticamente o cliente no servidor Eureka
Registra o nome, host e porta do aplicativo


Usando valores do Spring Environment.
Mas pode ser substituído.
Atribua a seu aplicativo `spring.application.name`

Torna este aplicativo uma "instância" e um "cliente"

Ele pode localizar outros serviços do cliente

Service ID (Eureka VIP)
Corresponde a
`spring.application.name`

```
@Autowired
DiscoveryClient clientes;
@GetMapping("/servicos")
public @ResponseBody String getURI() {
    List<ServiceInstance> lstRecursosDaInstancia = clientes.getInstances("AULA");
    if (lstRecursosDaInstancia != null && lstRecursosDaInstancia.size() > 0) {
        ServiceInstance serviceInstance = lstRecursosDaInstancia.get(0);
        return serviceInstance.getUri().toString();
    }
    return "servico nao encontrado";
}
```



Micro serviços

Resumo

- **Passive Service Discovery**
 - Ter serviços se registrar e encontrar outros automaticamente
- **Spring Cloud Eureka Server**
 - Mantém registros, compartilha informações sobre outros registrantes
 - Sincroniza-se com outros servidores
- **Spring Cloud Eureka Client**
 - Conecta-se ao servidor para registrar-se e obter informações sobre outros clientes.

Micro serviços - hands on

Novo Projeto com Spring Initializr

- Spring Boot Actuator
- Eureka Discovery Client
- Spring Web

Modificar a classe Application class. Adicionar @EnableDiscoveryClient.

bootstrap.yml

```
spring:  
  application:  
    name: sujeito
```

application.yml

```
eureka:  
  client:  
    serviceUrl:  
      defaultZone: http://localhost:8090/eureka/  
  
server:  
  port: 8091  
  
adjetivos: linda, feia, fabulosa, feiosa, deslumbrante, perfumada
```

Micro serviços - hands on

Criar um controller para adjetivo, artigo, substantivo, sujeito e verbo

```
@RestController
public class Controller {
    @Value("${adjetivos}")
    public String adjetivos;

    @GetMapping("/")
    public @ResponseBody String getWord() {
        String[] arr = adjetivos.split(",");
        int i = (int) Math.round(Math.random() * (arr.length - 1));
        return arr[i];
    }
}
```

Micro serviços - hands on

Criar um controller para montar a sentença toda

```
@RestController
public class Controller {

    @Autowired
    DiscoveryClient clientes;

    @GetMapping("/frase")
    public @ResponseBody String getFrase() {
        return getFracaoFrase("SUJEITO") + " " + getFracaoFrase("VERBO") + " " +
            getFracaoFrase("ARTIGO") + " " + getFracaoFrase("SUBSTANTIVO") + " "
            + getFracaoFrase("ADJETIVO") + ".";
    }

    public String getFracaoFrase(String microServico) {
        List<ServiceInstance> listaServicos = clientes.getInstances(microServico);
        if (listaServicos != null && listaServicos.size() > 0) {
            URI uri = listaServicos.get(0).getUri();
            if (uri != null) {
                return (new RestTemplate()).getForObject(uri, String.class);
            }
        }
        return null;
    }
}
```