
O que Eureka Server ?
E como configurá-lo?
05/04

Agenda:

Monolithic Vs. Microserviços

Vantagens e Desvantagens dos Microservices

Arquitetura de Microserviços

Comunicação Sincrona entre Microserviços

Service Registry / Discovery

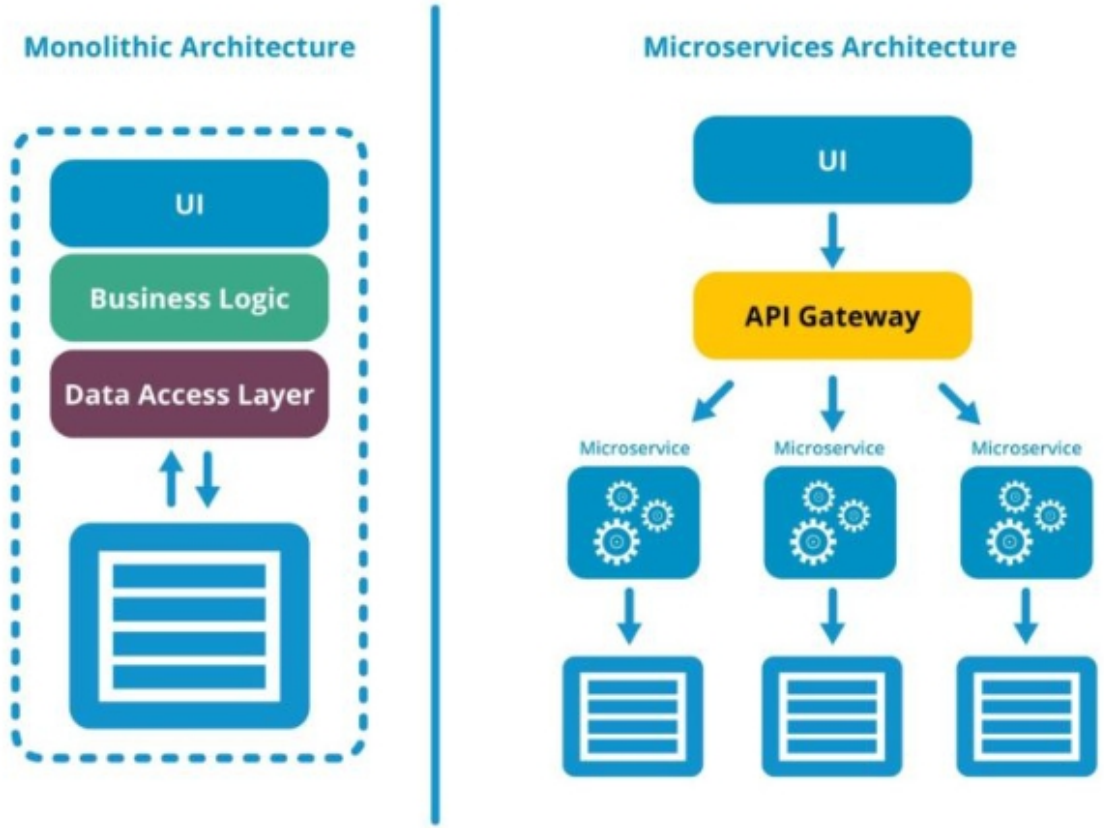
API Composition Pattern

Arquitetura que será criada

Como configurar Eureka Server e Clients

Hands on (Prática)

Monolithic Vs. Microserviços



Características dos microserviços

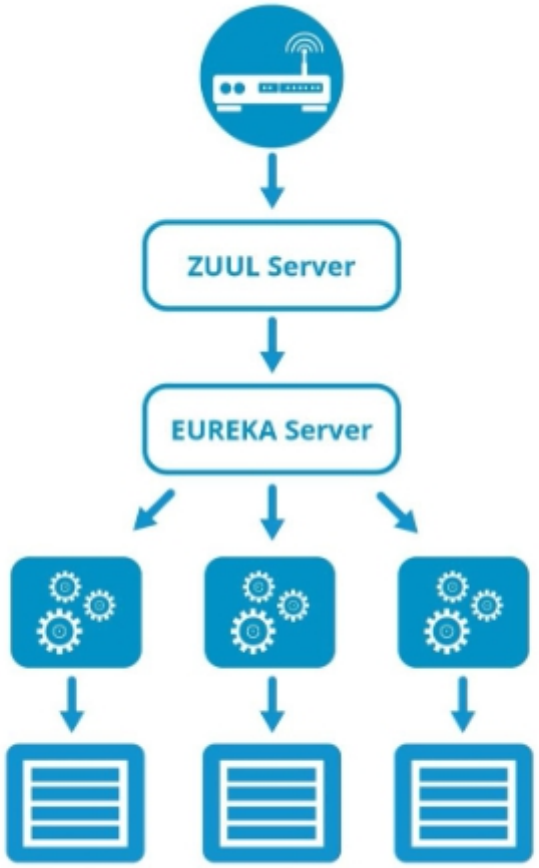
• Autônomo: os microsserviços são independentes e podem ser desenvolvidos e implantados independentemente sem afetar outros serviços.

• Especializado: cada microsserviço é projetado para um recurso específico.

• Stateless: Microsserviços não compartilham o estado do serviço; em alguns casos, se houver a necessidade de manter o estado, ele será mantido em um banco de dados.

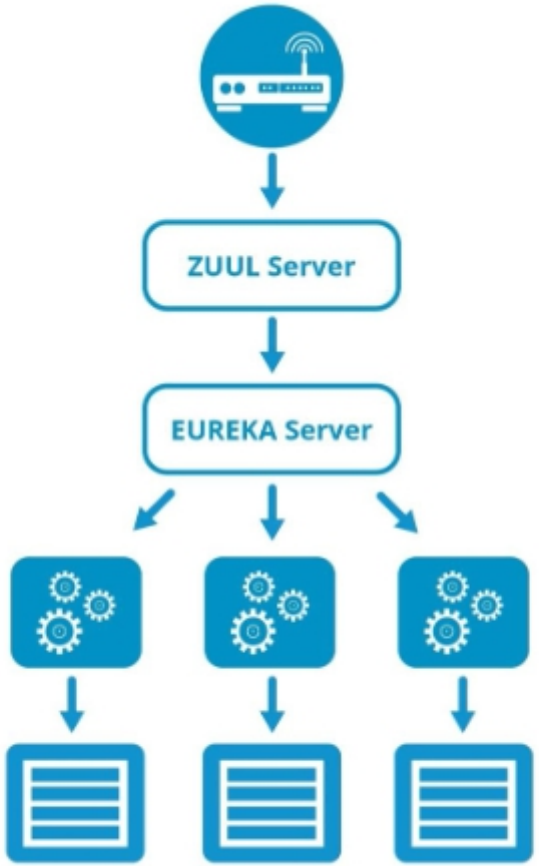
• Interfaces bem definidas (contrato de serviço): Os microsserviços possuem interfaces bem definidas que permitem a comunicação com eles, como um esquema JSON ou WSDL.

Vantagens dos Microservices



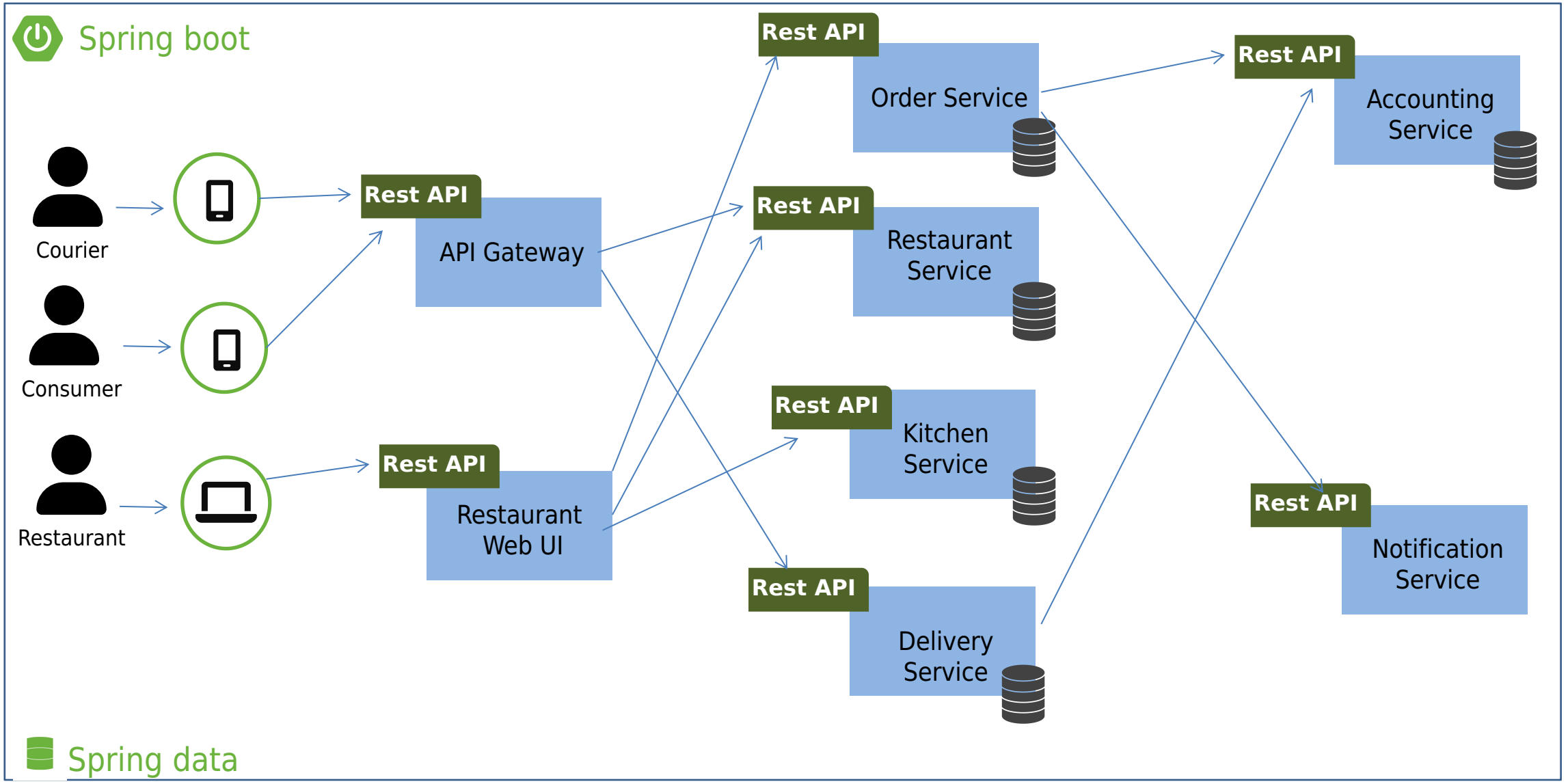
1. Simplicidade para implantar e atualizar
2. Flexibilidade para usar diferentes tecnologias
3. Escalabilidade para modificar ou adicionar recursos
4. Disponibilidade para continuar funcionando
5. Redundância para se manter disponível
6. Agilidade para processar e mudar
7. Independência para ganhar em otimização e produtividade

Desvantagens dos Microservices

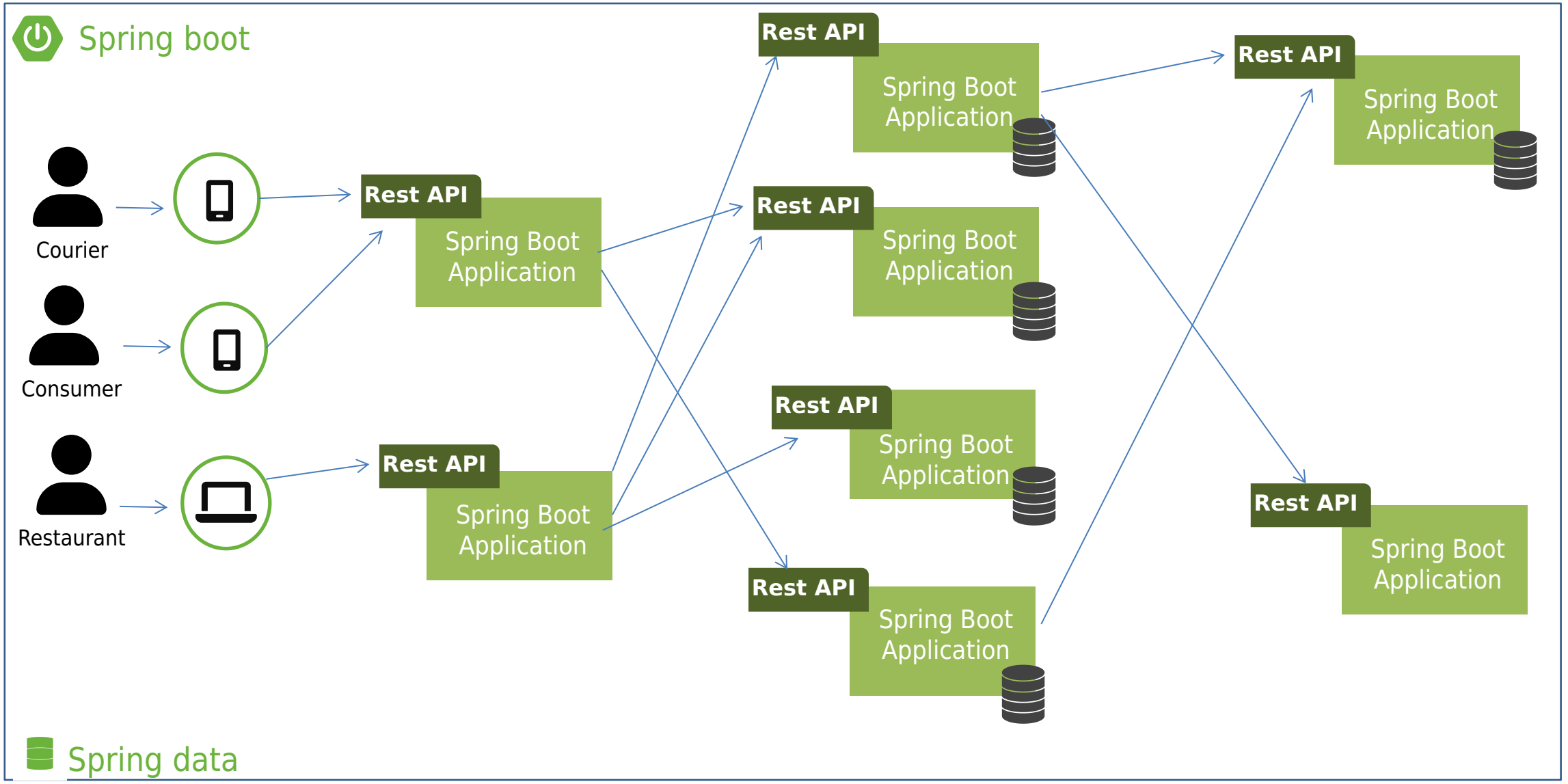


1. Complexidade
2. Potencialmente muito granulares;
3. Latência durante uso intensivo; e
4. Os testes podem ser complexos.
5. Integração com aplicações monolíticas legadas
6. Segurança

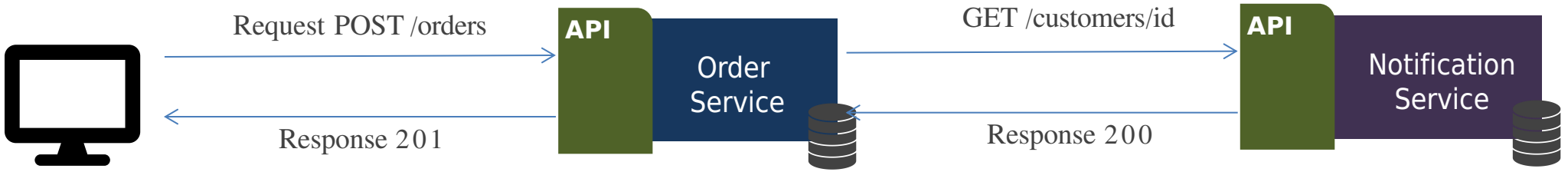
Arquitetura de Microserviços



Arquitetura de Microserviços - Spring Boot



Comunicação Sincrona entre Microserviços



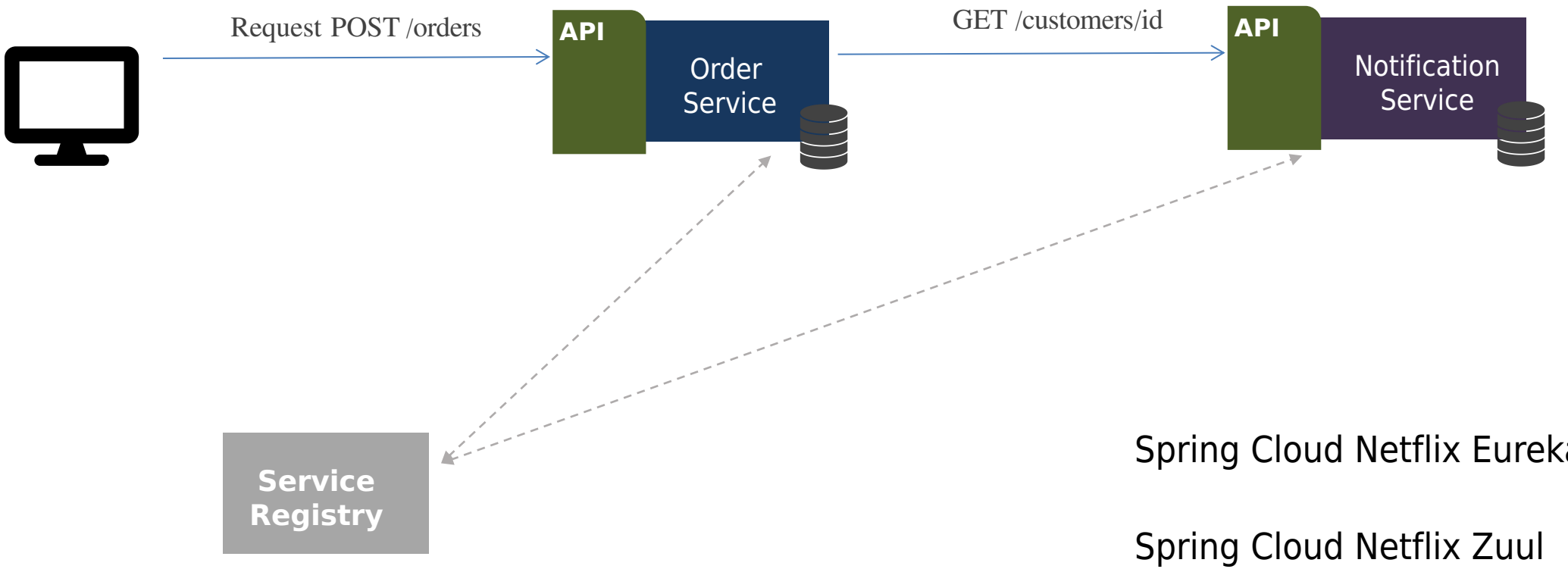
Spring Web: Rest and Spring MVC

Spring Reative Web: Spring Webflux

Spring HATEOAS

Spring Validation

Service Registry / Discovery

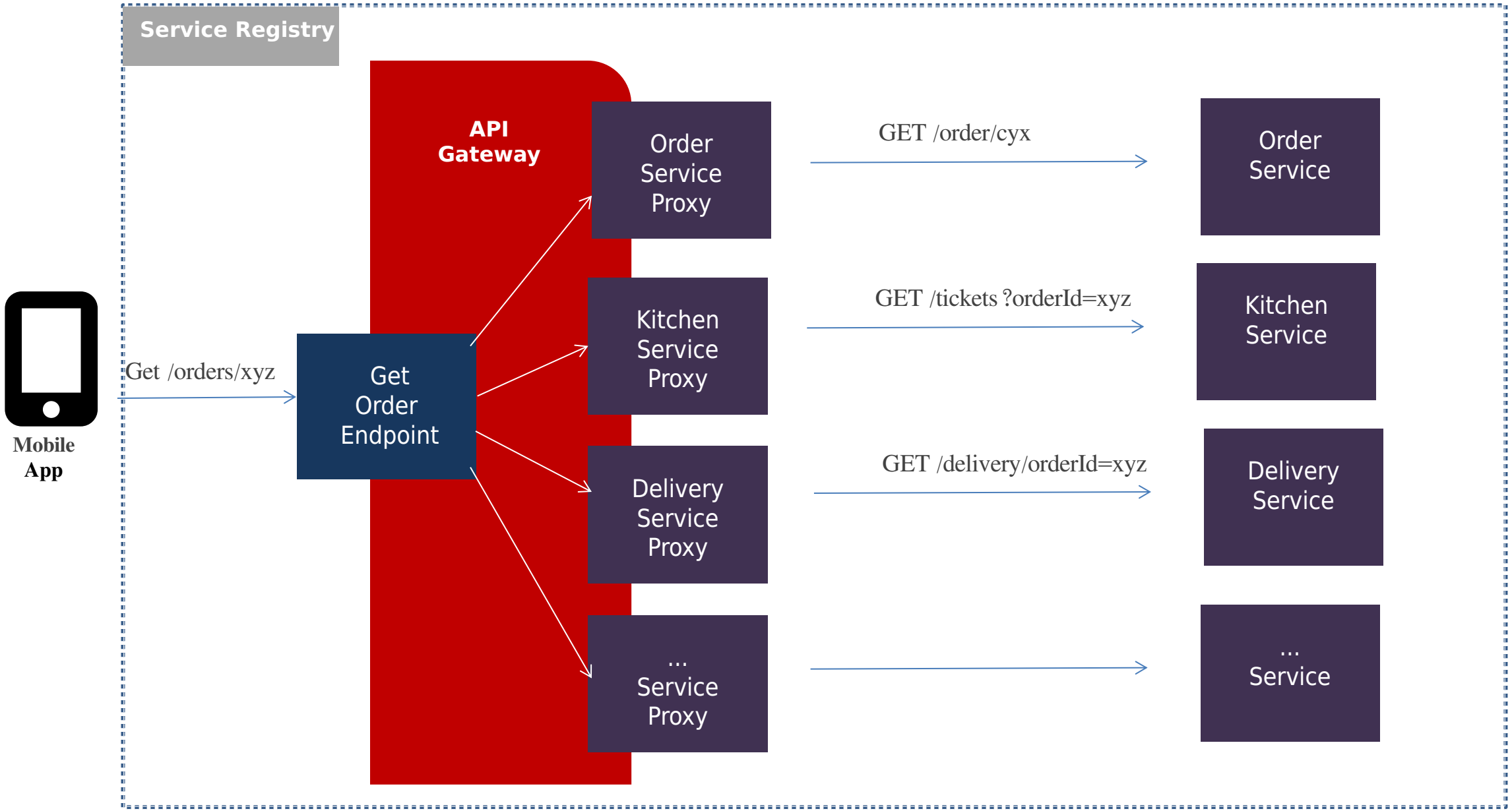


Spring Cloud Netflix Eureka

Spring Cloud Netflix Zuul

Spring Cloud Consul

API Composition Pattern




Como Configurar Eureka
Server com API Gateway

Ferramentas e Bibliotecas usadas no projeto

Java 11 - https://www.oracle.com/br/java/technologies/javase/jdk11-archive-downloads.html

Intellij Ultimate - https://www.jetbrains.com/pt-br/idea/

Maven 3.8.5 - https://maven.apache.org/download.cgi

Spring Initializr - https://start.spring.io/

Spring Boot - 2.3.3.RELEASE

Spring Cloud Netflix Eureka - https://spring.io/projects/spring-cloud

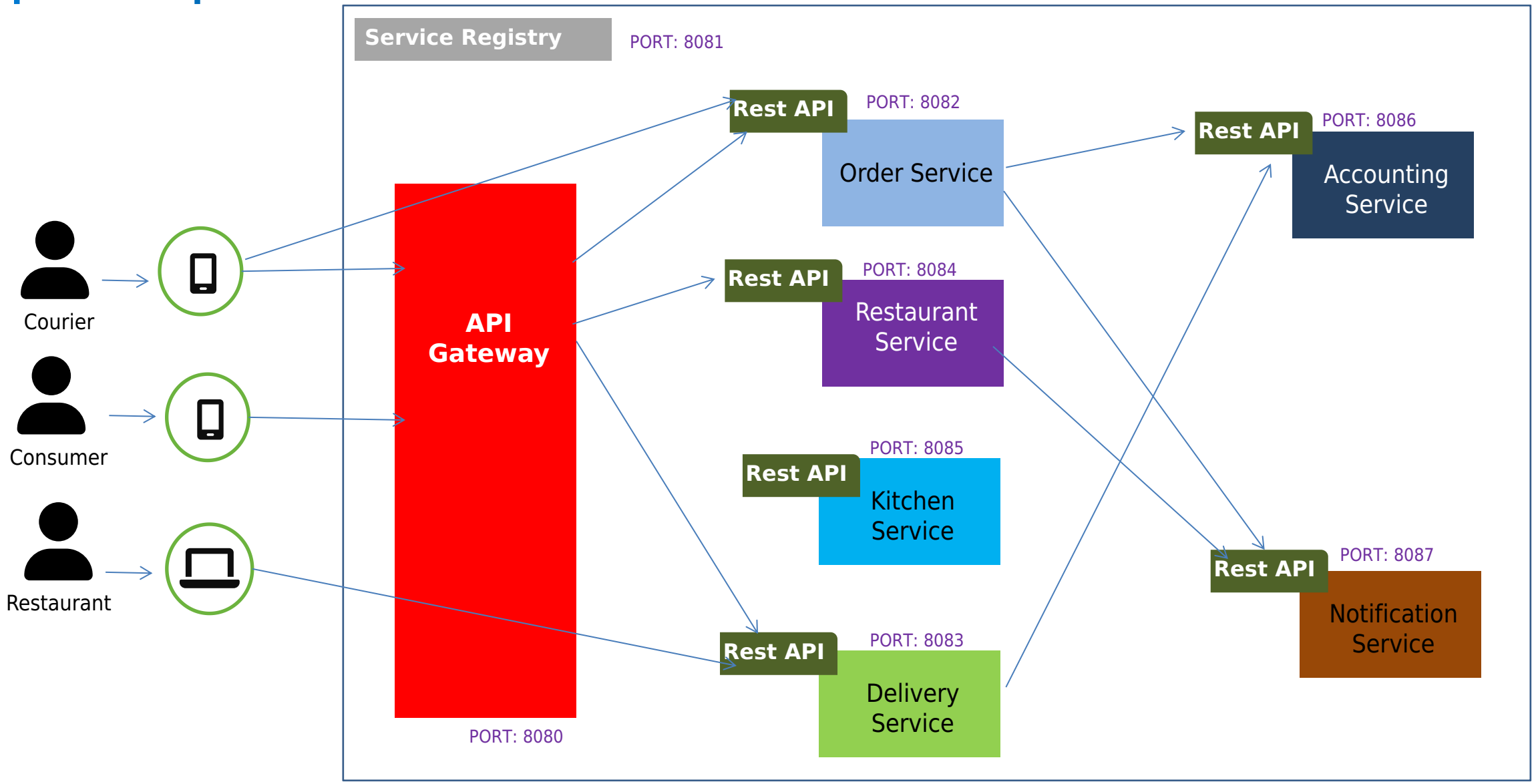
Spring Cloud Netflix Zuul - https://cloud.spring.io/spring-cloud-netflix/reference/html/

Spring Cloud - https://spring.io/blog/2021/07/07/spring-cloud-hoxton-sr12-has-been-released

Spring Cloud OpenFeign - https://spring.io/projects/spring-cloud-openfeign

Spring Boot started Webflux - https://spring.io/guides/gs/reactive-rest-service/

Arquitetura que ser criada:



https://microservices.io/

Visualização do Eureka Server (Server Registry)

http://localhost:8081/

spring Eureka

HOME LAST 1000 SINCE STARTUP

System Status

Environment	test
Data center	default

Current time	2022-05-03T16:39:05 -0300
Uptime	06:17
Lease expiration enabled	true
Renews threshold	13
Renews (last min)	28

DS Replicas

localhost

Instances currently registered with Eureka

Application	AMIs	Availability Zones	Status
ACCOUNTING	n/a (1)	(1)	UP (1) - weder:accounting:8086
DELIVERY	n/a (1)	(1)	UP (1) - weder:delivery:8083
GATEWAY	n/a (1)	(1)	UP (1) - weder:gateway:8080
KITCHEN	n/a (1)	(1)	UP (1) - weder:kitchen:8085
NOTIFICATION	n/a (1)	(1)	UP (1) - weder:notification:8087
ORDER	n/a (1)	(1)	UP (1) - weder:order:8082
RESTAURANT	n/a (1)	(1)	UP (1) - weder:restaurant:8084

General Info

Name	Value
total-avail-memory	546mb
environment	test
num-of-cpus	8

Visualização Application.properties

application.properties

```
spring.application.name=gateway
server.servlet.context-path=/gateway
server.port=8080

eureka.instance.prefer-ip-address=true
eureka.client.fetch-registry=true
eureka.client.register-with-eureka=true
eureka.client.service-url.defaultZone=http://localhost:8081/eureka

zuul.sensitive-headers=Cookie
```

Figura 01 - Gateway

application.properties

```
spring.application.name=registry
server.port= 8081

eureka.client.fetch-registry=false
eureka.client.register-with-eureka=false
eureka.client.service-url.defaultZone=http://localhost:${server.port}/eureka/
```

Figura 02 - Registry - Eureka Server

https://cloud.spring.io/spring-cloud-netflix/multi/multi_spring-cloud-eureka-server.html

Visualização Application.properties

application.properties

```
spring.application.name=order
server.port=8082

eureka.instance.prefer-ip-address=true
eureka.client.service-url.defaultZone=http://localhost:8081/eureka/
eureka.client.register-with-eureka=true

api.url.accounting = http://localhost:8086/v1/accounting

```

application.properties

```
spring.application.name=accounting
server.port=8086

eureka.instance.prefer-ip-address=true
eureka.client.service-url.defaultZone=http://localhost:8081/eureka/
eureka.client.register-with-eureka=true


```

application.properties

```
spring.application.name=delivery
server.port=8083

eureka.instance.prefer-ip-address=true
eureka.client.service-url.defaultZone=http://localhost:8081/eureka/
eureka.client.register-with-eureka=true

api.url.accounting = http://localhost:8086/v1/accounting

```

application.properties

```
spring.application.name=notification
server.port=8087

eureka.instance.prefer-ip-address=true
eureka.client.service-url.defaultZone=http://localhost:8081/eureka/
eureka.client.register-with-eureka=true

```

Visualização Organizacional do Projeto no Intellij

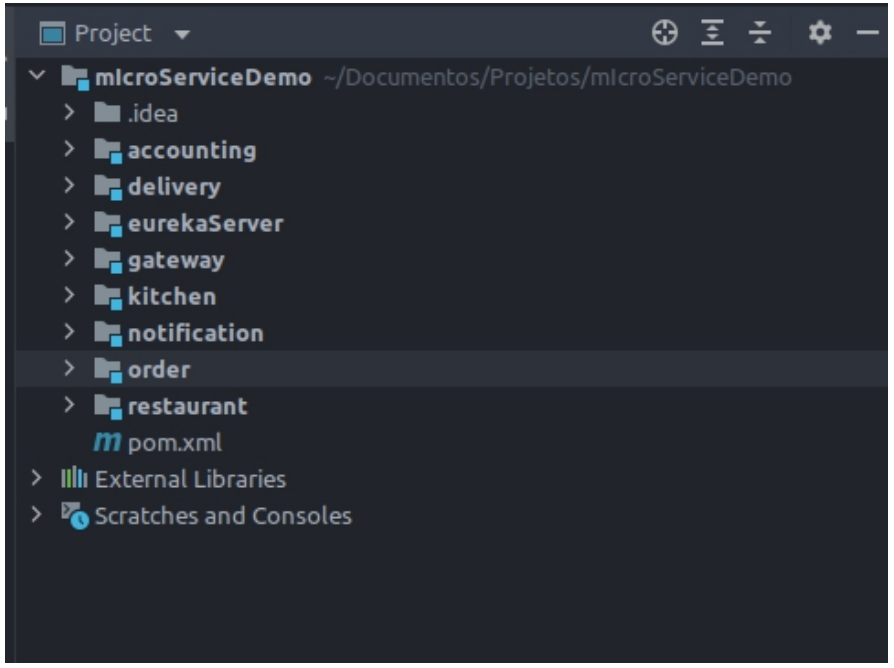


Figura 01 - Project

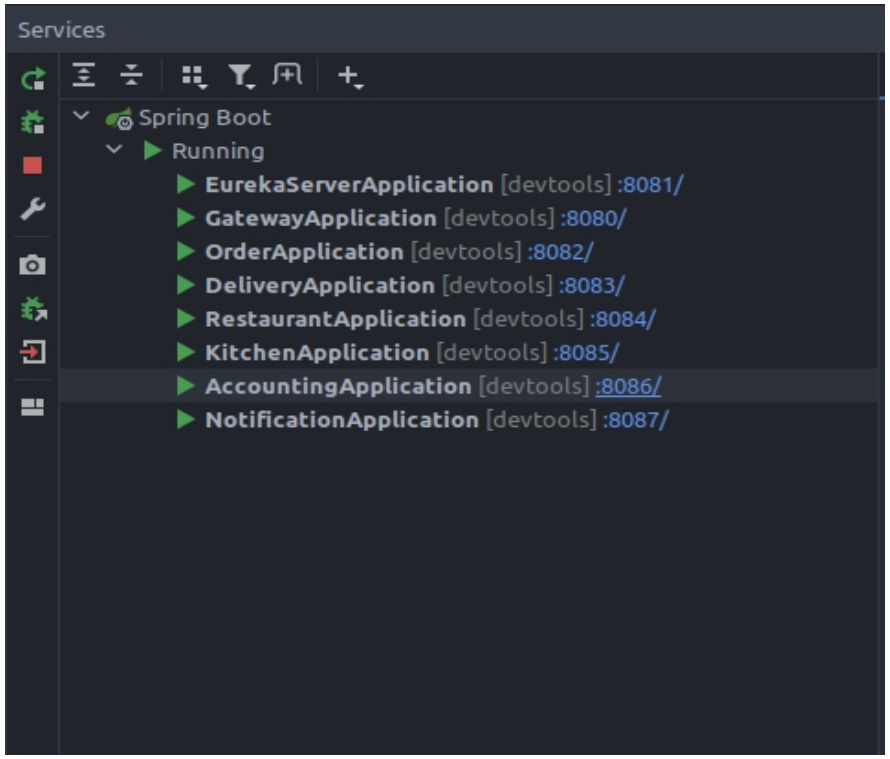


Figura 02 - Services

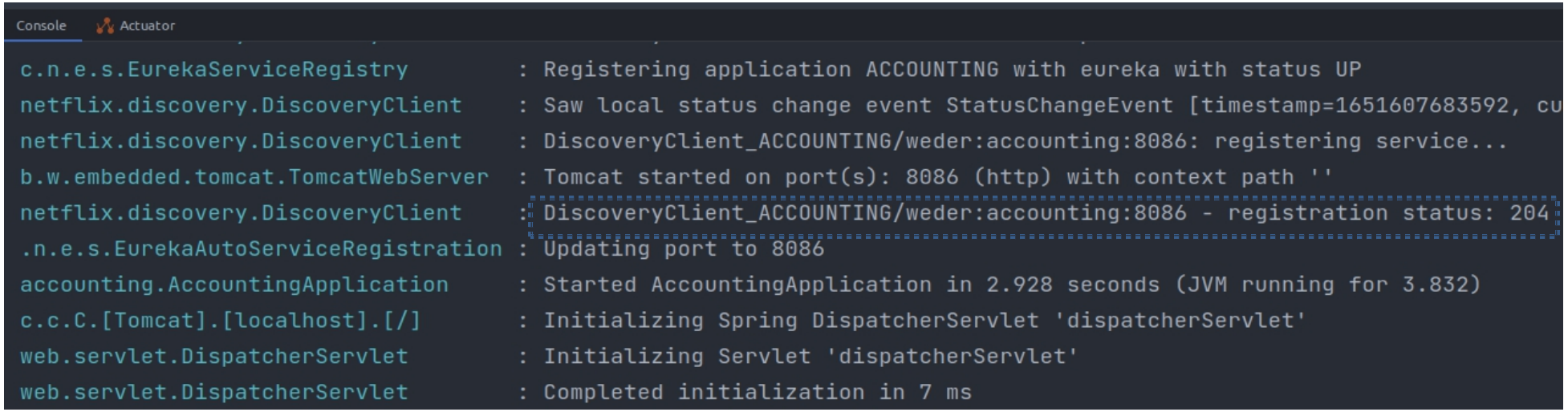


Figura 03 -Console

Visualização Application (main) Projeto no IntelliJ

EurekaServerApplication.java

```java
package com.eureka.eurekaserver;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.cloud.netflix.eureka.server.EnableEurekaServer;

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

Figura 01 - Eureka main

GatewayApplication.java

```java
package com.wsousa.gateway;
import ...
@SpringBootApplication
@EnableZuulProxy
@EnableEurekaClient
public class GatewayApplication {
    public static void main(String[] args) { SpringApplication.run(GatewayApplication.class, args); }
}
```

Figura 02 - Gateway main

OrderApplication.java

```java
package com.wsousa.order;

import ...

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

Figura 03 - Order main


Hands on

Obrigado!!