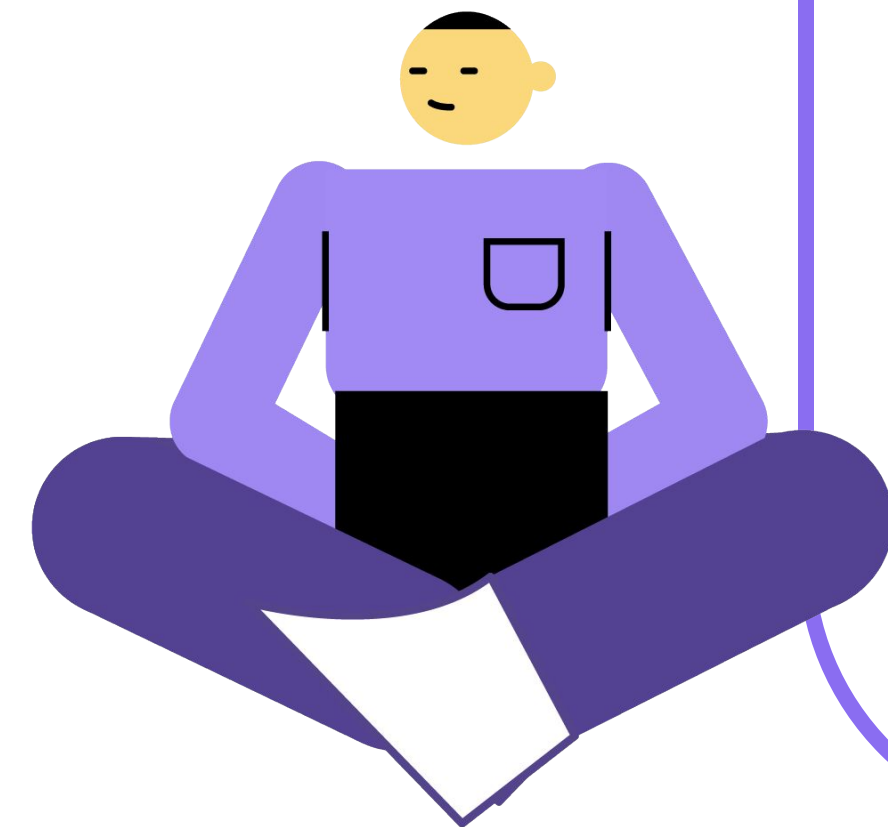


# Configuração do Eureka

# Configuração do Eureka

O Eureka implementa a descoberta de serviço do lado do cliente, o que significa que são os clientes que se comunicam com o servidor de descoberta (**Eureka server**) para obter informações sobre instâncias de microsserviço disponíveis.

A seguir veremos como configurar o Eureka client e o Eureka server.



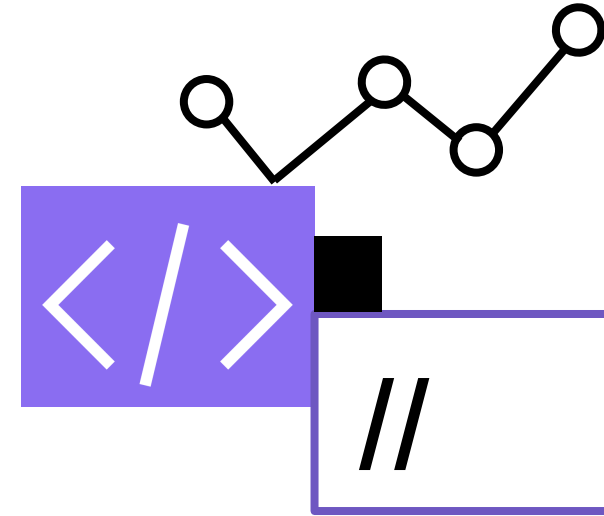
# Índice

- 01 [Eureka client](#)
- 02 [Eureka server](#)



01

# Eureka client

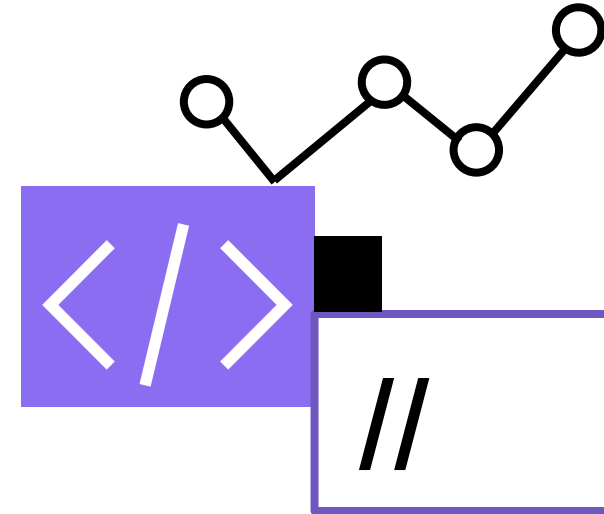


# Eureka client

Para configurar o **Eureka client** em nosso projeto, basta adicionar a dependência `spring-cloud-starter-netflix-eureka-client`.

Por exemplo, usando o **Maven** como gerenciador de dependências:

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



Ao adicionar a dependência, nosso aplicativo é registrado automaticamente no Eureka server, e basta indicar onde o servidor está rodando (por padrão a porta será 8761).

Dentro do arquivo **applicacion.properties**:

```
eureka.client.service-url.defaultZone=  
http://localhost:8761/eureka/
```

Uma propriedade muito importante para configurar é **spring.application.name**. Essa propriedade é usada para fornecer a cada microsserviço o hostname. Mais tarde, os clientes —para comunicar— usarão este nome para formar a URL:

```
spring.application.name=meu-servico
```

Para desabilitar o Eureka client em nosso projeto, podemos adicionar:

```
eureka.client.enabled=false
```

02

# Eureka server

# Eureka server

Para configurar o **Eureka server**, devemos seguir estes passos:

01

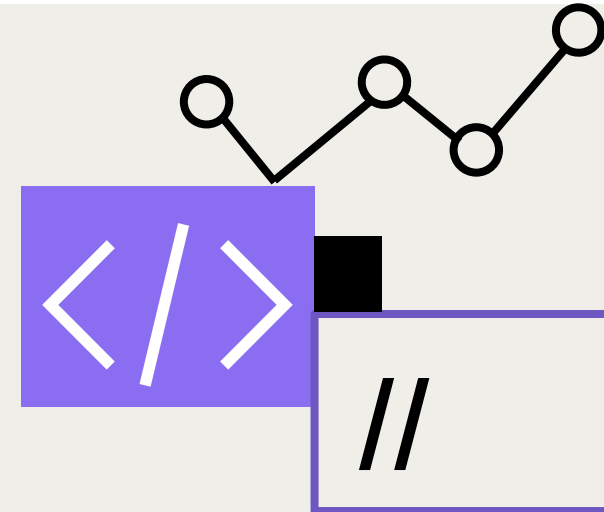
Crie um novo projeto com Spring Boot.

02

Adicione as dependências:

```
spring-cloud-starter-netflix-eureka-server  
spring-boot-starter-web  
spring-boot-starter-actuator
```





Com **Maven**:

{}

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

03

Adicione a anotação **@EnableEurekaServer** na classe principal do nosso projeto. Por exemplo:

```
{}  
  
@SpringBootApplication  
@EnableEurekaServer  
public class Application {  
  
    public static void main(String[] args) {  
        new  
        SpringApplicationBuilder(Application.class).  
        web(true).run(args);  
    }  
}
```

04

Em **application.properties**, o Eureka server pode funcionar como cliente e servidor simultaneamente. Para que ele não se registre como cliente devemos colocar as seguintes propriedades em *false*:

```
eureka.client.register-with-eureka=false  
eureka.client.fetch-registry=false
```

Configuramos a porta:


```
server.port=8761
```

05

Agora podemos executar nosso projeto e ver a página principal do dashboard do Eureka, pois todos os endpoints são mostrados em:

<http://localhost:8761/eureka/>

# Dashboard Eureka Server



[HOME](#) [LAST 1000 SINCE STARTUP](#)

### System Status

Environment	N/A	Current time	2022-02-01T22:08:31 -0300
Data center	N/A	Uptime	00:00
		Lease expiration enabled	false
		Renews threshold	1
		Renews (last min)	0

### DS Replicas

localhost

### Instances currently registered with Eureka

Application	AMIs	Availability Zones	Status
No instances available			

### General Info

Name	Value
total-avail-memory	594mb
num-of-cpus	16
current-memory-usage	166mb (27%)

Muito obrigado!