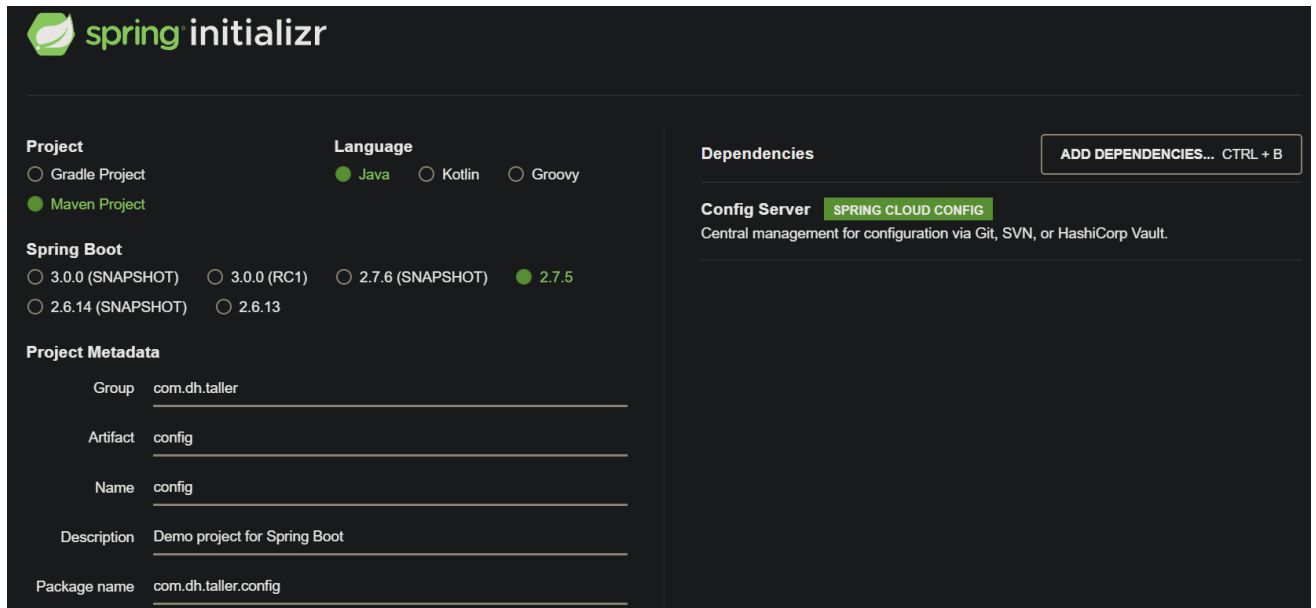


1. Crear MS Config Server

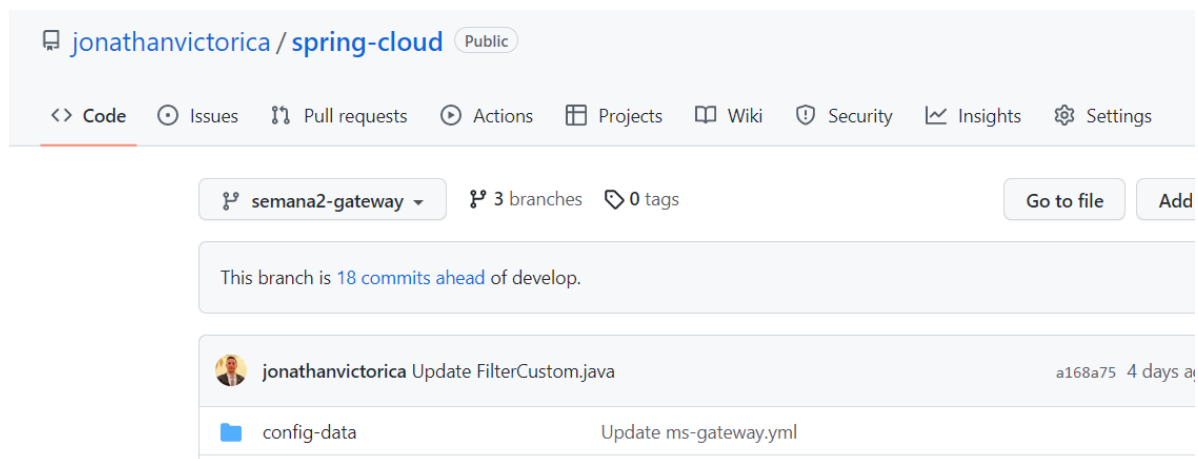
1.1 Ir la página de <https://start.spring.io/> y generar un proyecto base con las siguientes librerías:



The screenshot shows the Spring Initializr web form. The 'Project' section has 'Maven Project' selected. The 'Language' section has 'Java' selected. The 'Spring Boot' section has '2.7.5' selected. The 'Project Metadata' section has the following values: Group: com.dh.taller, Artifact: config, Name: config, Description: Demo project for Spring Boot, Package name: com.dh.taller.config. The 'Dependencies' section has 'Config Server' and 'SPRING CLOUD CONFIG' selected. A button 'ADD DEPENDENCIES... CTRL + B' is visible.

1.2 Importar proyecto en el IDE

1.3 Crear en el repositorio de github carpeta para centralizar todas las configuraciones:



1.4 Configurar proyecto con las anotaciones correspondientes:

```
@SpringBootApplication
@EnableConfigServer
public class MsConfigServerApplication {
```

1.5 Crear archivo `application.yml` con la siguiente configuración

```
1  server:
2    port: ${PORT:8888}
3
4  spring:
5    application:
6      name: config-server
7    cloud:
8      config:
9        server:
10       git:
11         uri: https://github.com/jonathanvictorica/spring-cloud
12         default-label: semana2-gateway
13         force-pull: true
14         search-paths: config-data
15
16
```

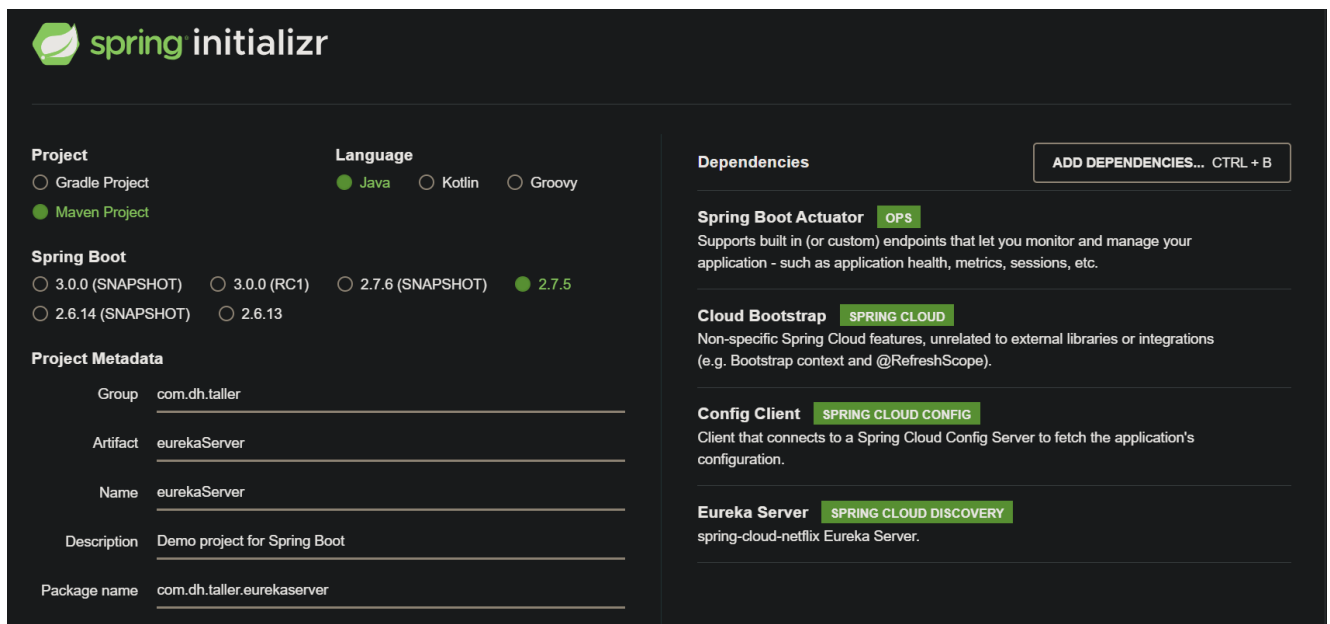
uri: github url donde va estar configuracion

default-label: Branch de github

search-paths: carpeta interna de github

2. EUREKA SERVER

2.1 Configurar Eureka Server



The screenshot shows the Spring Initializr interface with the following configuration:

- Project:** ☒ Maven Project
- Language:** ☒ Java
- Spring Boot:** ☒ 2.7.5
- Project Metadata:**
 - Group: com.dh.taller
 - Artifact: eurekaServer
 - Name: eurekaServer
 - Description: Demo project for Spring Boot
 - Package name: com.dh.taller.eurekaServer
- Dependencies:**
 - Spring Boot Actuator** (OPS): Supports built in (or custom) endpoints that let you monitor and manage your application - such as application health, metrics, sessions, etc.
 - Cloud Bootstrap** (SPRING CLOUD): Non-specific Spring Cloud features, unrelated to external libraries or integrations (e.g. Bootstrap context and @RefreshScope).
 - Config Client** (SPRING CLOUD CONFIG): Client that connects to a Spring Cloud Config Server to fetch the application's configuration.
 - Eureka Server** (SPRING CLOUD DISCOVERY): spring-cloud-netflix Eureka Server.

2.2 Importar proyecto en el IDE

2.3 Configurar anotaciones de la libreria de spring-eureka-server

```
5  import org.springframework.cloud.netflix.eureka.server.EnableEurekaServer;
6
7  @SpringBootApplication
8  @EnableEurekaServer
9  public class MsEurekaServerApplication {
10
```

2.4 Crear archivo bootstrap.yml para eureka server puede obtener su configuración

7 lines (7 sloc) | 122 Bytes

7 lines (7 sloc) | 122 Bytes

```
1  spring:
2    application:
3      name: ms-eurekaServer
4    cloud:
5      config:
6        enabled: true
7        uri: http://localhost:8888
```

2.5 Crear en el repositorio de github dedicado a la configuración centralizada un archivo con la configuración de eureka server

```
1  server:
2    port: ${PORT:8761}
3
4  spring:
5    application:
6      name: ms-eurekaServer
7  eureka:
8    instance:
9      hostname: localhost
10   client:
11     registerWithEureka: false
12     fetchRegistry: false
13     serviceUrl:
14       defaultZone: http://${eureka.instance.hostname}:${server.port}/eureka/
```

3. API GATEWAY

3.1 crear Api gateway

Project <input type="radio"/> Gradle Project <input checked="" type="radio"/> Maven Project	Language <input checked="" type="radio"/> Java <input type="radio"/> Kotlin <input type="radio"/> Groovy	Dependencies ADD DEPENDENCIES... CTRL + B
Spring Boot <input type="radio"/> 3.0.0 (SNAPSHOT) <input type="radio"/> 3.0.0 (RC1) <input type="radio"/> 2.7.6 (SNAPSHOT) <input checked="" type="radio"/> 2.7.5 <input type="radio"/> 2.6.14 (SNAPSHOT) <input type="radio"/> 2.6.13		Spring Boot Actuator OPS Supports built in (or custom) endpoints that let you monitor and manage your application - such as application health, metrics, sessions, etc.
Project Metadata		Cloud Bootstrap SPRING CLOUD Non-specific Spring Cloud features, unrelated to external libraries or integrations (e.g. Bootstrap context and @RefreshScope).
Group	com.dh.taller	Config Client SPRING CLOUD CONFIG Client that connects to a Spring Cloud Config Server to fetch the application's configuration.
Artifact	eurekaServer	Eureka Discovery Client SPRING CLOUD DISCOVERY A REST based service for locating services for the purpose of load balancing and failover of middle-tier servers.
Name	eurekaServer	Cloud LoadBalancer SPRING CLOUD ROUTING Client-side load-balancing with Spring Cloud LoadBalancer.
Description	Demo project for Spring Boot	Gateway SPRING CLOUD ROUTING Provides a simple, yet effective way to route to APIs and provide cross cutting concerns to them such as security, monitoring/metrics, and resiliency.
Package name	com.dh.taller.eurekaServer	
Packaging	<input checked="" type="radio"/> Jar <input type="radio"/> War	
Java	<input type="radio"/> 19 <input checked="" type="radio"/> 17 <input type="radio"/> 11 <input type="radio"/> 8	

3.2 Importar proyecto en IDE

3.3 Crear archivo bootstrap.yml para api gateway puede obtener su configuración

```
7 lines (7 sloc) | 117 Bytes

1  spring:
2    application:
3      name: ms-gateway
4    cloud:
5      config:
6        enabled: true
7        uri: http://localhost:8888
```

3.4 Crear en el repositorio de github dedicado a la configuración centralizada un archivo con la configuración de api gateway

```
info:
  app:
    name: ${spring.application.name}

eureka:
  client:
    registerWithEureka: false
    fetchRegistry: true
    serviceURL:
      defaultZone: http://localhost:8761/eureka
```

```

server:
  port: ${PORT:${SERVER_PORT:8080}}

spring:
  application:
    name: ms-gateway
  cloud:
    gateway:
      default-filters:
        - name: FilterCustom
      discovery:
        locator:
          enabled: true
      routes:
        - id: mscourse
          uri: lb://ms-course
          predicates:
            - Path=/course/**
        - id: mssubscription
          uri: lb://ms-subscription
          predicates:
            - Path=/subscription/**

```

Por cada grupo de endpoint (RestController("/pathController")) se configura un grupo de routes.

id: id único de ese grupo de endpoints

uri: Url relativa o específica(localhost:8085) del servicio que expone ese endpoint

predicate: pathController (puede ser uno o más)

default-filters: es opcional, puede eliminarse

4. Configurar cada microservicio de nuestro ecosistema.

4.1 Descomentar de cada proyecto las librerías que son clientes de eureka, config, y loadbalance

4.2 Extraer el archivo de configuración y centralizarlo en el github config

4.3 Configurar el archivo bootstraps.yml correspondiente

```

7 lines (7 sloc) | 122 Bytes
1  spring:
2    application:
3      name: ms-eurekaServer
4    cloud:
5      config:
6        enabled: true
7        uri: http://localhost:8888

```

(spring.application.name): nombre de nuestro microservicio

5. Ejecutar test:

Levantar instancia de configserver, eurekaserver, apigateway.

Levantar dos instancias de cada microservicio para comprobar que se registran y levantan de manera dinámica.

Lugar donde estan los test por api-gateway

<https://github.com/jonathanvictorica/spotify-spring-cloud/blob/develop/api-playlist/test/testApiGateway.http>