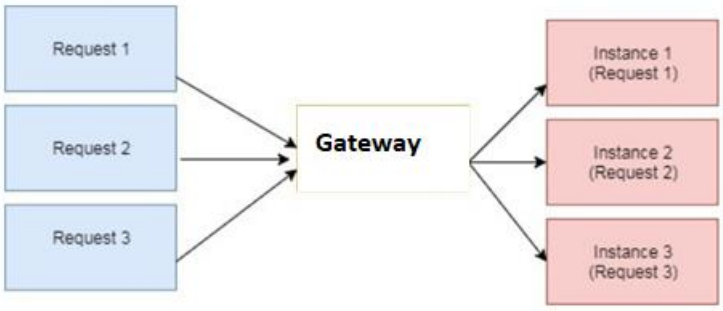
esprit Se former autrement HONORIS UNITED UNIVERSITIES	Année Universitaire : 2023-2024 **Applications Web Distribuées**

Implémentation « API Gateway»

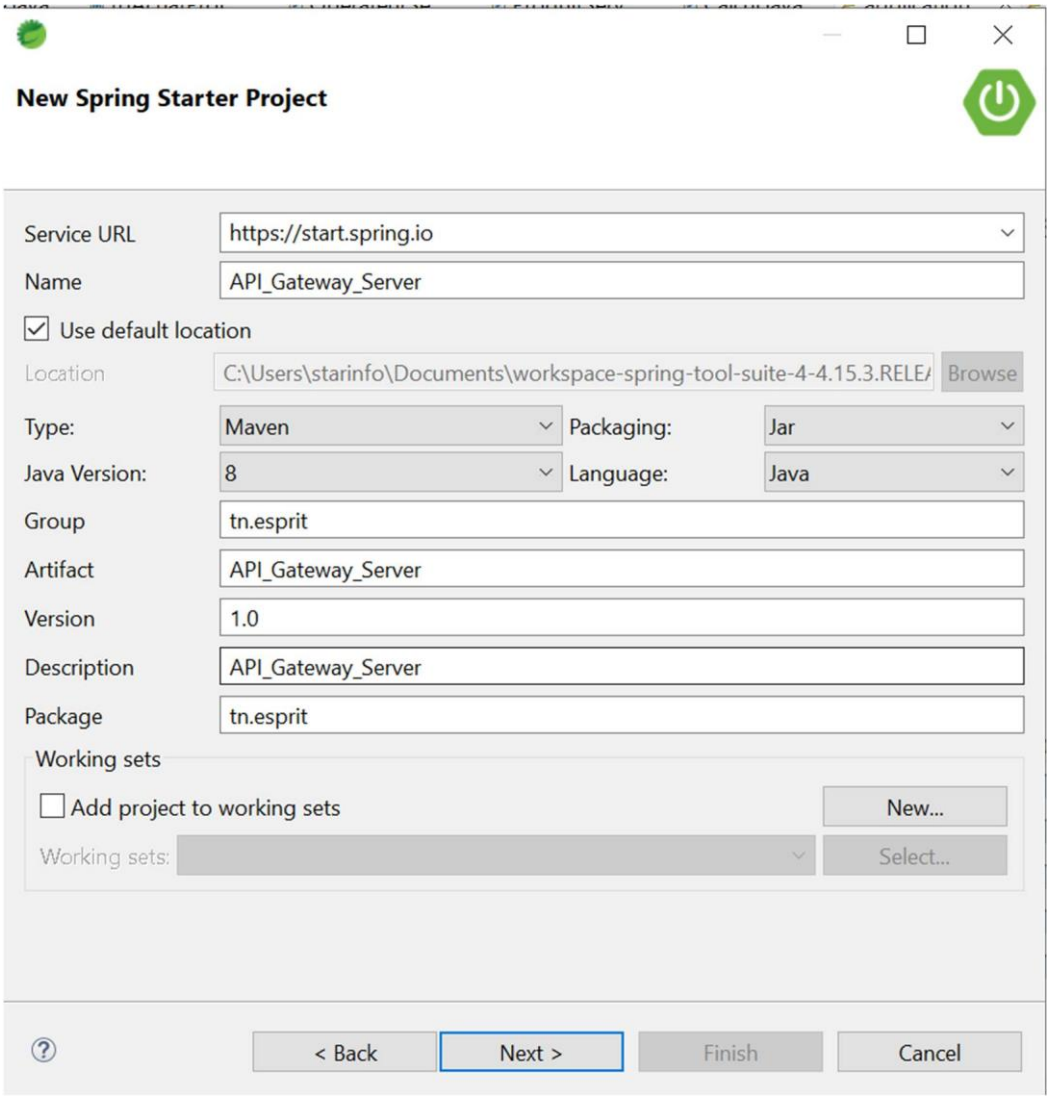
Objectif

Lorsque le Gateway reçoit une demande, il sélectionne l'un des emplacements physiques disponibles et transmet les demandes à l'instance de service réelle. L'ensemble du processus de mise en cache de l'emplacement des instances de service et du transfert de la demande vers l'emplacement réel est fourni sans aucune configuration supplémentaire.

L'objectif de cet atelier est d'organiser l'appel vers les microservices.

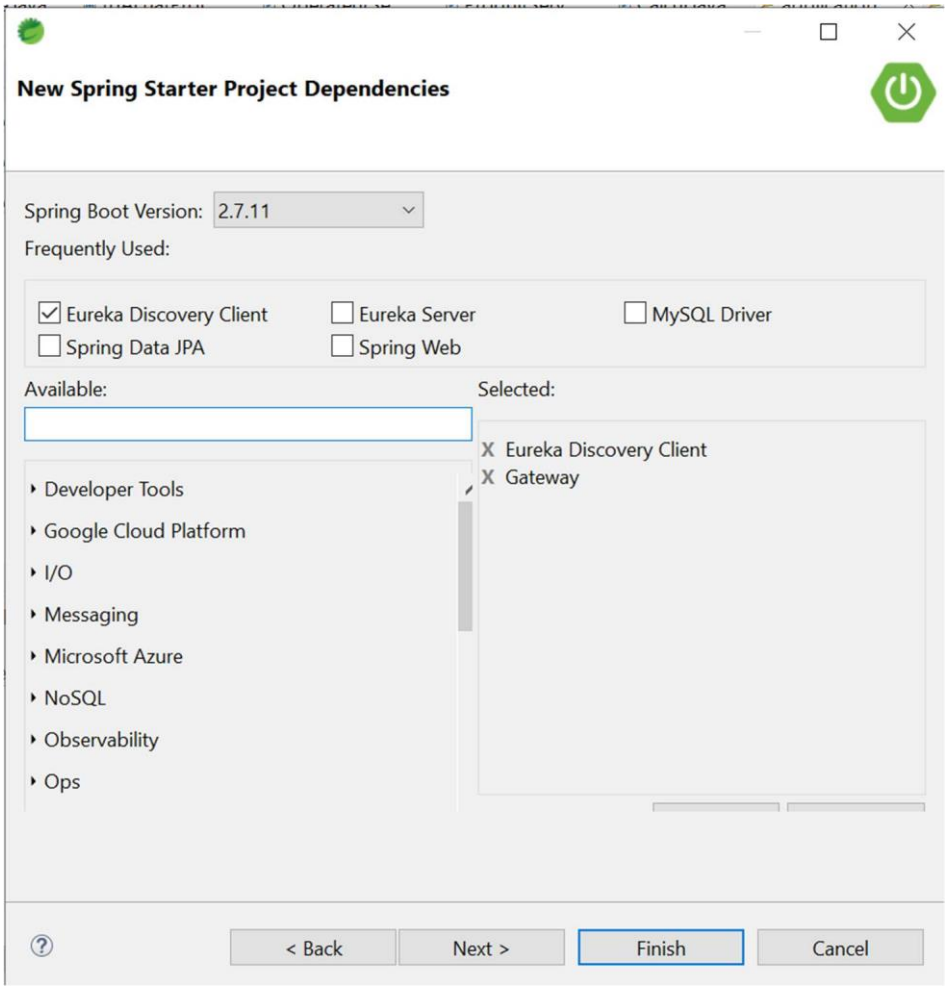


1- Créez un projet Spring Boot en suivant ses étapes :



2- Ajoutez ces deux starters :

- Eureka Discovery client
- Gateway



Vous obtenez ces deux dépendances dans votre pom.xml :

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

3- Faites un maven update project et clean install

4- Au niveau de la classe main spring Boot de votre projet, ajouter l'annotation :

```
@EnableEurekaClient
```

Pour définir une Gateway, il y'a deux méthodes de travail :

- Ajouter la relation entre gateway et MS à travers le fichier application.properties
- Ajouter la relation entre Gateway et MS à travers une classe de configuration

Méthode 1 :

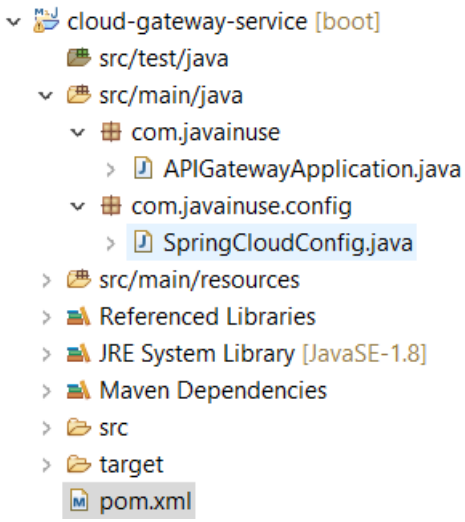
Ajoutez ces lignes dans votre fichier application.properties :

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

eureka.client.serviceUrl.defaultZone = http://localhost:8761/eureka/
eureka.client.register-with-eureka=true

spring.cloud.gateway.discovery.locator.enabled = true

# Candidat Microservice
spring.cloud.gateway.routes[0].id=candidat-service
spring.cloud.gateway.routes[0].uri=http://localhost:8088
spring.cloud.gateway.routes[0].predicates[0]=Path=/candidat/**
```

NB : N'oubliez pas de mettre à jour votre fichier si vous utilisezd'autres ports.

Méthode 2 :

- Ajouter la classe Config comme le montre la figure ci-dessous :



- Le contenu de la classe de configuration **SpringCloudConfig** est le suivant :

```
package com.javainuse.config;

import org.springframework.cloud.gateway.route.RouteLocator;
import org.springframework.cloud.gateway.route.builder.RouteLocatorBuilder;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;

@Configuration
public class SpringCloudConfig {
-
```

```
        @Bean
        public RouteLocator gatewayRoutes(RouteLocatorBuilder builder) {
            return builder.routes()

    //Micro-service 1
                    .route(r -> r.path("/candidat/**")
                            .uri("http://localhost:8088/")
                            .id("candidat-service"))
    .build();
        }

    }
```

Path : c'est le path à utiliser pour accéder au MS-candidat à travers le Gateway

URI : c'est l'emplacement de MS-candidat

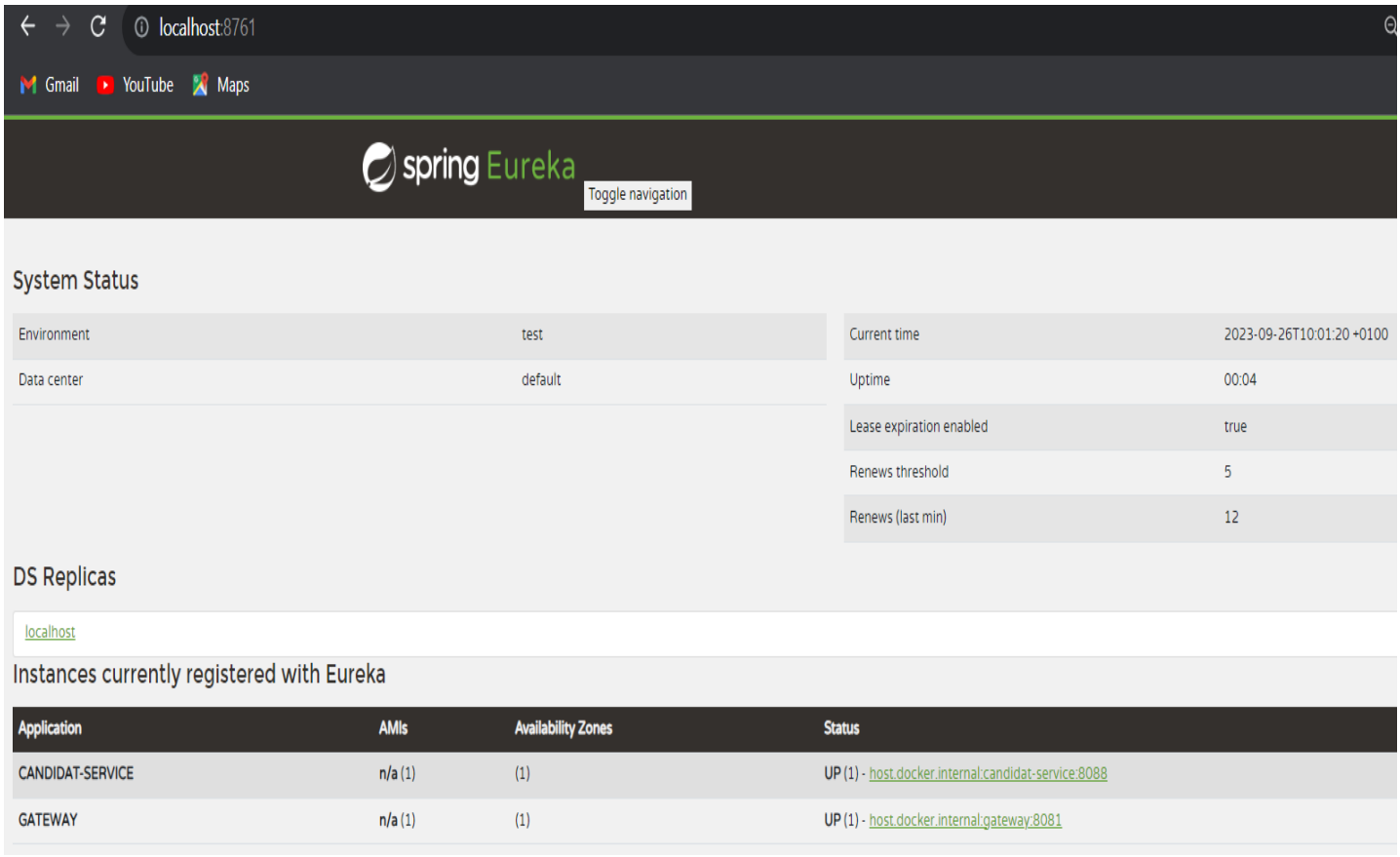
Id : c'est l'id de MS-candidat

NB : dans le fichier **application.properties**, vous ajouter juste le port et le nom de l'application

5- Accédez à l'interface de votre Eureka server :

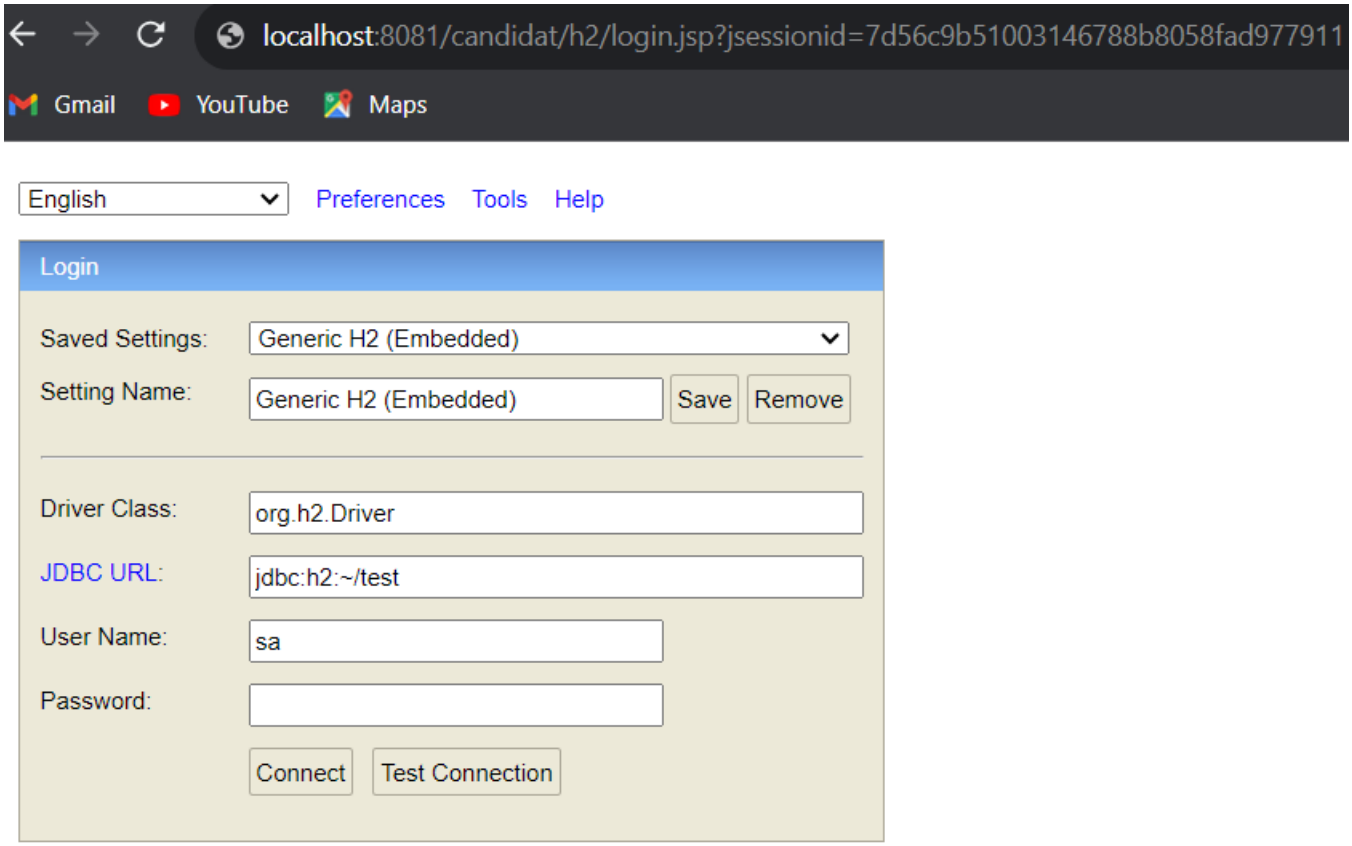
http://localhost:8761/

Vous obtenez le résultat suivant :



6- Testez votre application en utilisant l'url (le numéro de port) de votre API-Gateway service :

Exemple :


localhost:8081/candidat/h2/login.jsp?jsessionid=7d56c9b51003146788b8058fad977911
Gmail
YouTube
Maps
English
Preferences
Tools
Help
Login
Saved Settings:
Generic H2 (Embedded)
Setting Name:
Generic H2 (Embedded)
Save
Remove
Driver Class:
org.h2.Driver
JDBC URL:
jdbc:h2:~/test
User Name:
sa
Password:
Connect
Test Connection