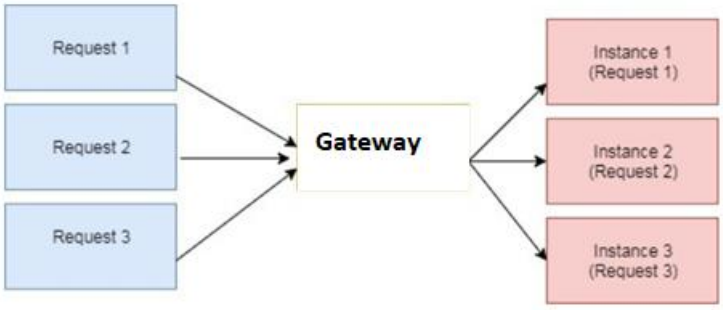
esprit Se former autrement HONORIS UNITED UNIVERSITIES	Année Universitaire : 2023-2024 **Applications Web Distribuées**

Implémentation « API Gateway»

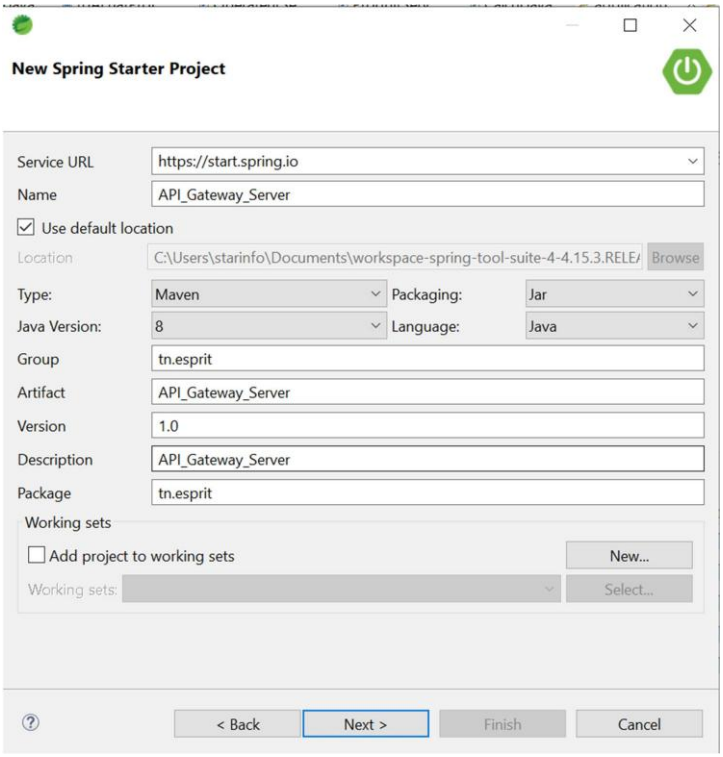
Objectif

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.

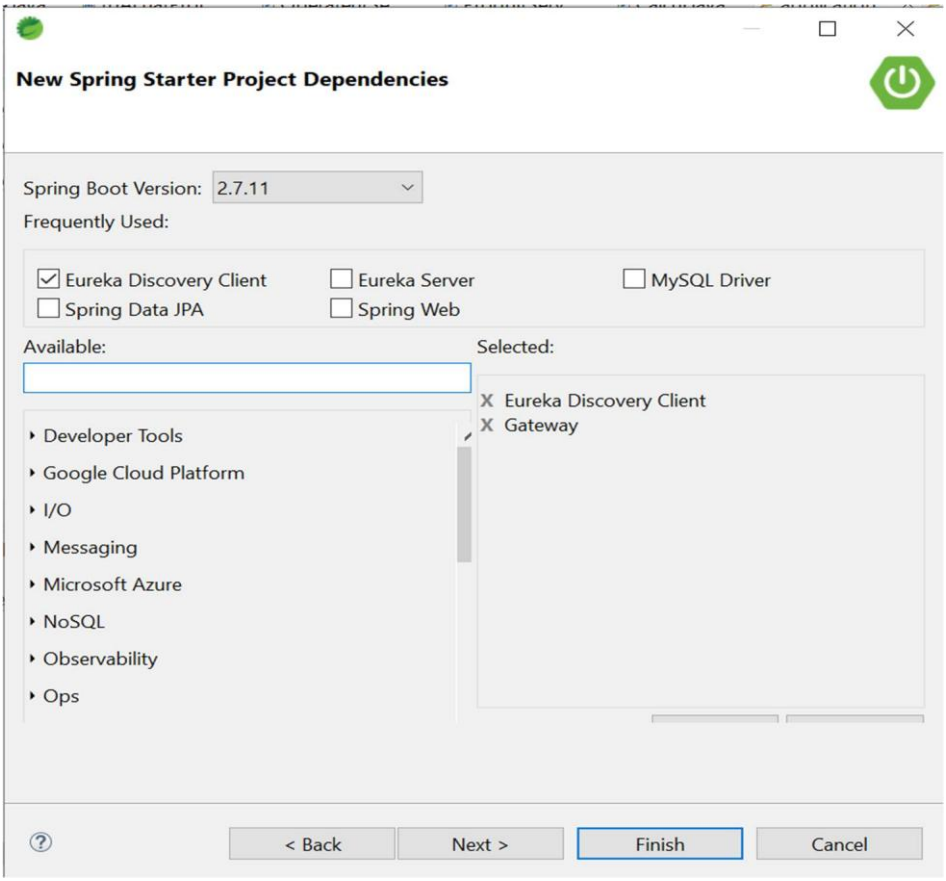
L'objectif de cet atelier est d'organiser l'appel vers les microservices.



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



- Ajoutez ces deux starters : Eureka Discovery client et 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>
```

2- Faites un maven update project et clean install

3- 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 les fichiers application.properties et/ou application.yml
- Ajouter la relation entre Gateway et MS à travers une classe de configuration

❖ Méthode 1 :

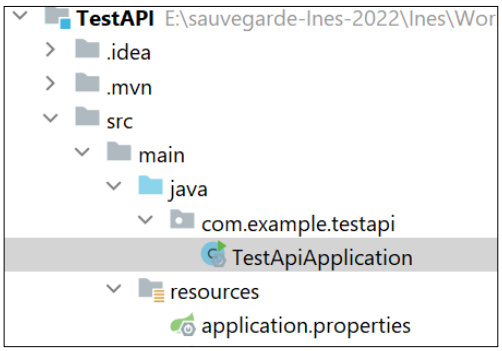
Ajoutez ces lignes dans votre fichier application.properties :

```
server.port= 8090
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:8082
spring.cloud.gateway.routes[0].predicates[0]= Path=/candidats/**
```

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

❖ Méthode 2 :

- Ajouter le code source suivant dans votre classe principale « **TestApiApplication.java** » comme le montre la figure ci-dessous :



-

- Le contenu de la classe **TestApiApplication.java** est le suivant :

```
@SpringBootApplication
@EnableEurekaClient
public class TestApiApplication {

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

```
    @Bean
    public RouteLocator gatewayRoutes(RouteLocatorBuilder builder) {

        return builder.routes()

                .route("candidatMs", r->r.path("/candidats/**",
                                                "/hello",
                                                "/h2")
                                            .uri("http://localhost:8082/"))
                .route("JobMs", r->r.path("/jobs/**")
                                    .uri("http://localhost:8081/"))

                .build();

    }

}
```

Path : c'est le path à utiliser pour accéder au MS-candidat à travers le Gateway (on peut spécifier les différents path d'accès vers les ressources de notre application comme le montre l'exemple du MS Candidat)

URI : c'est l'endpoint vers le Micro Service adéquat

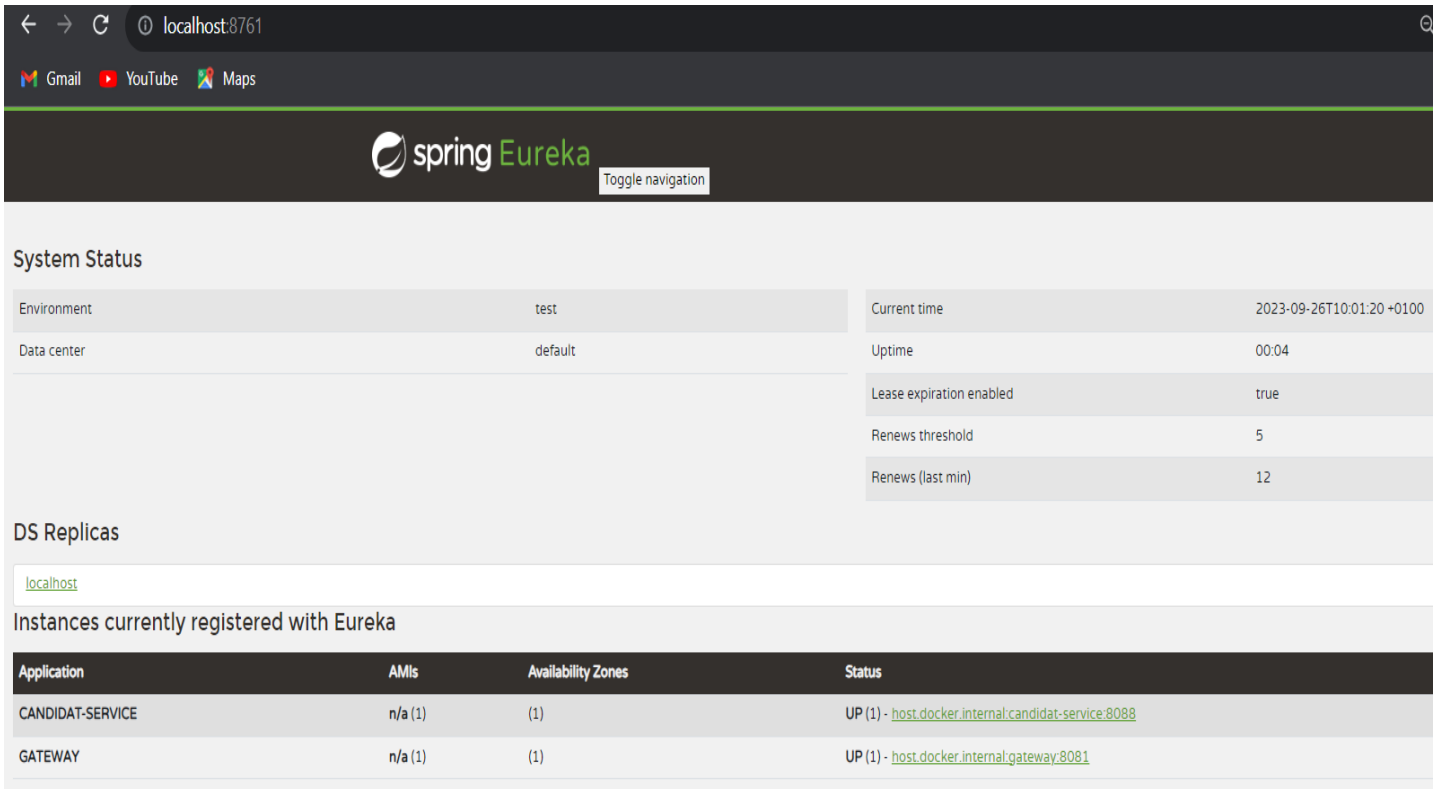
Id : c'est l'id du MS

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

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

http://localhost:8761/

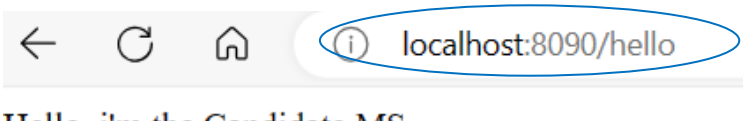
Vous obtenez le résultat suivant :



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

Exemple1 : l'accès vers la liste des candidats :

Exemple2 : l'accès vers la ressource hello :



Hello, i'm the Candidate MS