
TP N°5.2 :

Spring Cloud : API Gateway

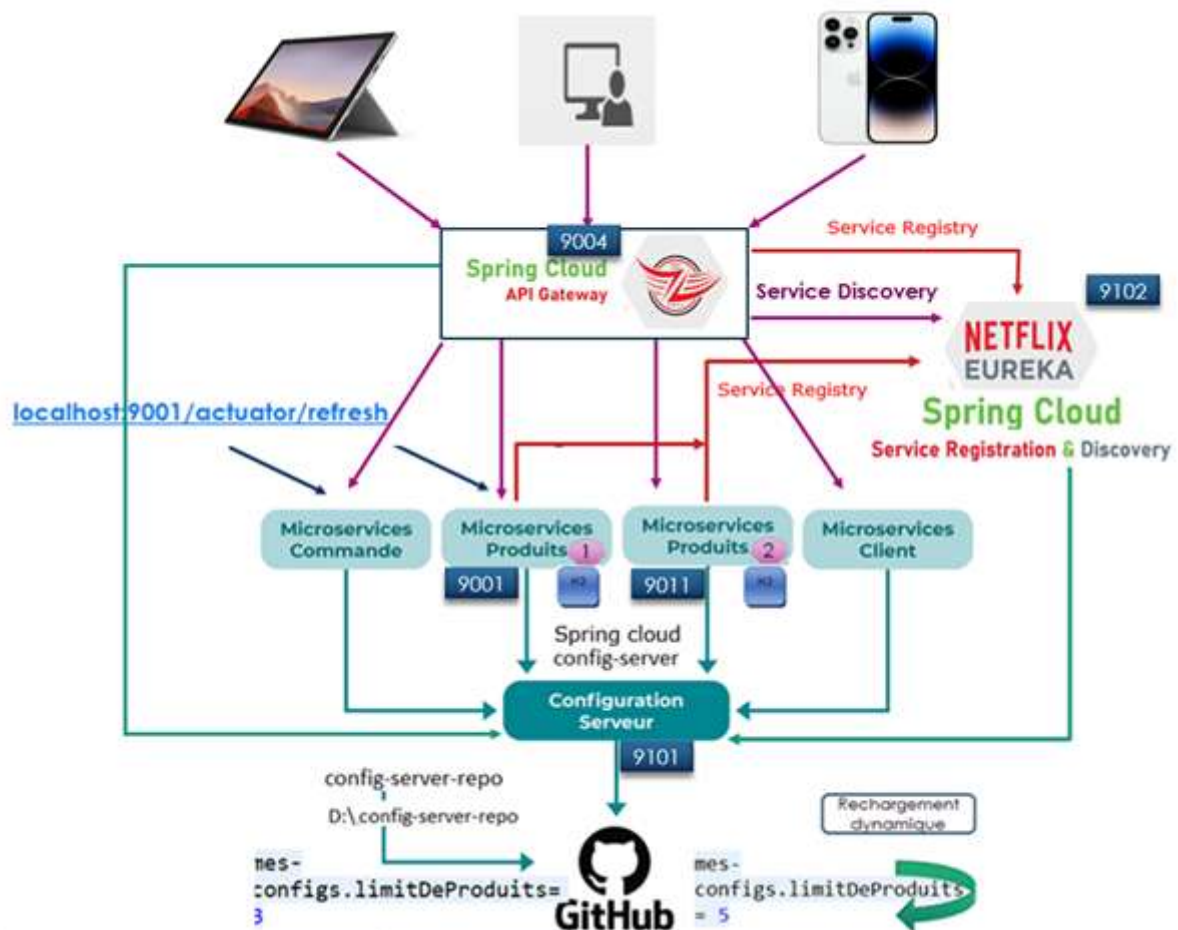
1. Prérequis

- TP3.1 Spring Cloud Config : toute fois ce n'est pas nécessaire d'avoir le Spring Cloud Config Server
- TP4.1 Spring Cloud Eureka Server : toute fois ce n'est pas nécessaire d'avoir le Spring Cloud Eureka Server
- POSTMAN ou un autre outil pour tester les méthodes POST, PUT et DELETE.

2. Objectifs

1. Mise en place d'un Eureka Server : console Eureka
2. Service Registry :
 - a. Annotation `@EnableDiscoveryClient`
 - b. Enregistrement de la gateway sur Eureka
 - c. Clonage d'un microservice et enregistrement auprès de Eureka Server
3. Service Discovery via gateway API
4. Load Balancing

3. Architecture de mise en œuvre



- Un serveur Spring cloud Gateway
- Deux instances du micro-service-produit qui s'enregistrent auprès de Eureka

- Remarquer que Eureka Server et Gateway Server s'enregistrent eux même auprès de Spring cloud config Server à l'instar des autres microservices applicatifs (Produit, commande, client)

4. Mise en place de l'API Gateway

Project <input checked="" type="radio"/> Gradle - Groovy <input type="radio"/> Gradle - Kotlin <input checked="" type="radio"/> Java <input type="radio"/> Kotlin <input type="radio"/> Groovy <input type="radio"/> Maven Spring Boot <input type="radio"/> 3.4.1 (SNAPSHOT) <input checked="" type="radio"/> 3.4.0 <input type="radio"/> 3.3.7 (SNAPSHOT) <input type="radio"/> 3.3.6 Project Metadata Group <input type="text" value="ma. formations"/> Artifact <input type="text" value="gateway-service"/> Name <input type="text" value="gateway-service"/> Description <input type="text" value="Demo project for Spring Boot"/> Package name <input type="text" value="ma. formations. gateway-service"/> Packaging <input checked="" type="radio"/> Jar <input type="radio"/> War Java <input type="radio"/> 23 <input type="radio"/> 21 <input checked="" type="radio"/> 17	Dependencies ADD DEPENDENCIES... CTRL + B Gateway SPRING CLOUD ROUTING Provides a simple, yet effective way to route to APIs in Servlet-based applications. Provides cross-cutting concerns to those APIs such as security, monitoring/metrics, and resiliency. 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. 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. Config Client SPRING CLOUD CONFIG Client that connects to a Spring Cloud Config Server to fetch the application's configuration.
-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

- Le fichier « pom.xml » généré comme suit :

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
    https://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
    <version>3.1.5</version>
    <relativePath/>
  </parent>
  <groupId>ma. formations</groupId>
  <artifactId>gateway-service</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <name>gateway-service</name>
  <description>gateway-service</description>
  <properties>
    <java.version>17</java.version>
    <spring-cloud.version>2022.0.4</spring-cloud.version>
  </properties>
  <dependencies>
    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-actuator</artifactId>
```

```

    </dependency>
    <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>
    <dependency>
      <groupId>org.springframework.cloud</groupId>
      <artifactId>spring-cloud-starter-config</artifactId>
    </dependency>

    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-test</artifactId>
      <scope>test</scope>
    </dependency>
  </dependencies>
  <dependencyManagement>
    <dependencies>
      <dependency>
        <groupId>org.springframework.cloud</groupId>
        <artifactId>spring-cloud-dependencies</artifactId>
        <version>${spring-cloud.version}</version>
        <type>pom</type>
        <scope>import</scope>
      </dependency>
    </dependencies>
  </dependencyManagement>

  <build>
    <plugins>
      <plugin>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-maven-plugin</artifactId>
      </plugin>
    </plugins>
  </build>
</project>

```

Explication :

- ✓ Le starter **spring-cloud-starter-netflix-eureka-client** permet au MS de s'enregistrer au niveau de l'Eureka server.
- ✓ Le starter **spring-cloud-starter-config** permet au MS de d'externaliser sa configuration en utilisant **Spring Cloud Config**.

- a. Le fichier « `bootstrap.properties` » du projet du « gateway-service »:

```
spring.application.name=gateway-service
server.port=9999
spring.cloud.config.uri=http://localhost:9101
```

Explication :

- ✓ La propriété `spring.config.import` permet au MS d'importer sa configuration centralisée au niveau du MS *config-service*.

- b. Le fichier « `gateway-service.properties` » qui sera synchronisé avec Github :

```
server.port=9999
#L'API Gateway s'enregistre auprès de Eureka
eureka.client.serviceUrl.defaultZone=http://localhost:9102/eureka/
```

- Au niveau de la classe de démarrage ajouter la méthode suivante :

```
package ma.formations.msa.gatewayservice;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.cloud.client.discovery.ReactiveDiscoveryClient;
import org.springframework.cloud.gateway.discovery.DiscoveryClientRouteDefinitionLocator;
import org.springframework.cloud.gateway.discovery.DiscoveryLocatorProperties;
import org.springframework.context.annotation.Bean;

@SpringBootApplication
public class GatewayServiceApplication {

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

    @Bean
    DiscoveryClientRouteDefinitionLocator locator(ReactiveDiscoveryClient rdc, DiscoveryLocatorProperties dlp){
        return new DiscoveryClientRouteDefinitionLocator(rdc, dlp);
    }
}
```

Explication :

- ✓ Le Bean `DiscoveryClientRouteDefinitionLocator` permet de configurer automatiquement les routes. Par exemple, lorsque vous demandez l'URI <http://localhost:9999/ACCOUNT-SERVICE/accounts>, la Gateway (gateway-service) contacte le MS *discovery-service* pour

demander l'URL du MS dont le nom est **account-service**, ensuite redirige vers l'Endpoint fourni pour ce MS, dans ce cas <http://localhost:8083/accounts>.

- ✓ vous pouvez également configurer les routes manuellement dans le fichier `application.properties` ou bien `application.yml`.

5. Test de l'API Gateway

Test direct du microservice

- Accéder au lien localhost:8083/api/accounts pour consulter la liste des comptes :



```
1  [
2    {
3      "id": "a1dce51e-9f85-4142-a0a0-7294976b7953",
4      "balance": 2320.9340199317908,
5      "createdAt": "2024-06-10",
6      "type": "CURRENT_ACCOUNT",
7      "currency": "MAD",
8      "customerId": 1,
9      "customer": null
10   },
11   {
12     "id": "cbb521d3-63c3-43f3-9642-21a6773e41a4",
13     "balance": 68574.67921308964,
14     "createdAt": "2024-06-10",
15     "type": "SAVING_ACCOUNT",
16     "currency": "MAD",
17     "customerId": 1,
18     "customer": null
19   },
20   {
21     "id": "794f3bbf-2841-4ed9-b548-3814dddbacd",
22     "balance": 52580.48923272165,
23     "createdAt": "2024-06-10",
24     "type": "CURRENT_ACCOUNT",
25     "currency": "MAD",
26     "customerId": 2,
27     "customer": null
28   },
29   {
30     "id": "da559a10-46e2-452c-8bd2-1d75c71e5cc0",
31     "balance": 49898.05516759577,
32     "createdAt": "2024-06-10",
33     "type": "SAVING_ACCOUNT",
34     "currency": "MAD",
35     "customerId": 2,
36     "customer": null
37   }
38 ]
```

Test du micro-service via GATWAY

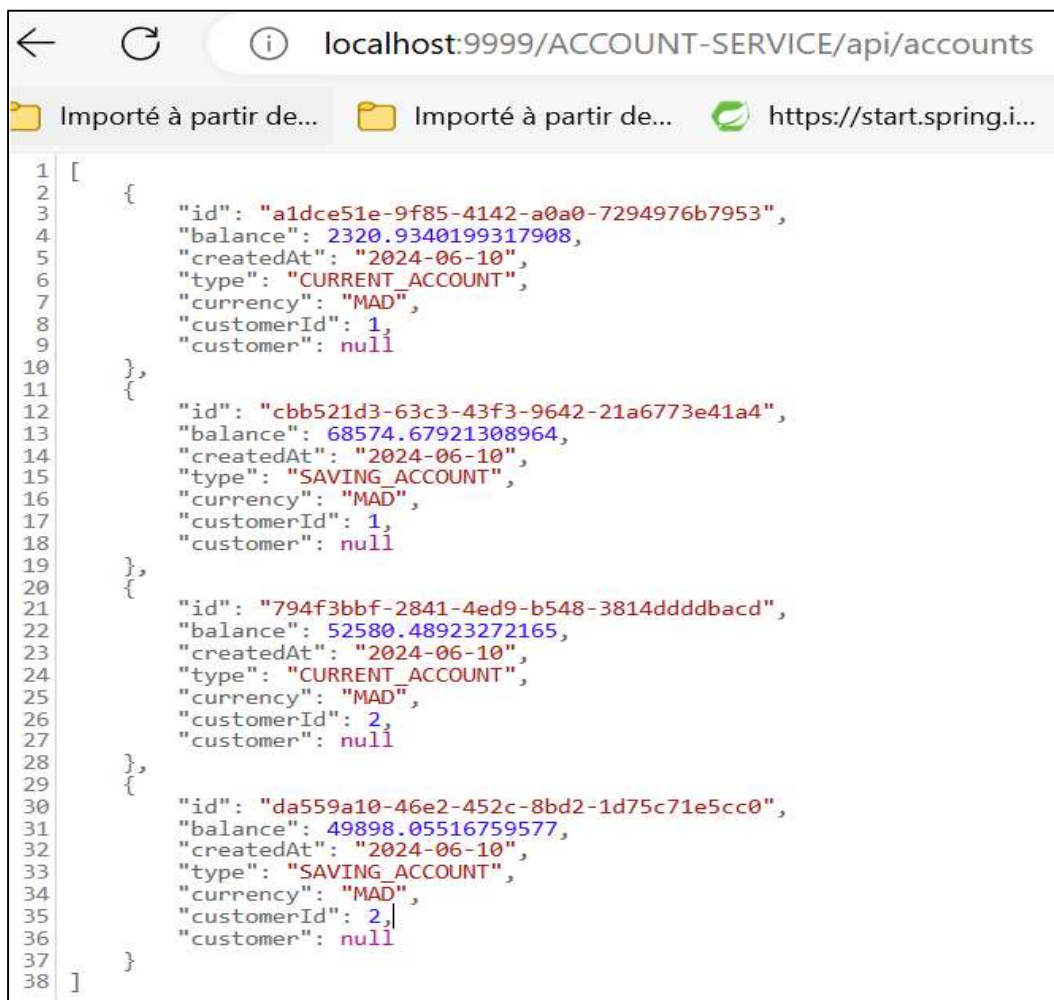
- Consulter un compte par son ID et vérifier que le client du compte a été bien récupéré via le Framework Spring Cloud OpenFeign



The screenshot shows a web browser window with the address bar displaying `localhost:8083/api/accounts/a1dce51e-9f85-4142-a0a0-7294976b7953`. The browser's address bar also shows several tabs: 'Importé à partir de...', 'Importé à partir de...', 'https://start.spring.i...', and 'Graphity Diagram E...'. The main content area displays a JSON response for the account with ID `a1dce51e-9f85-4142-a0a0-7294976b7953`. The JSON is as follows:

```
1 {
2   "id": "a1dce51e-9f85-4142-a0a0-7294976b7953",
3   "balance": 2320.9340199317908,
4   "createdAt": "2024-06-10",
5   "type": "CURRENT_ACCOUNT",
6   "currency": "MAD",
7   "customerId": 1,
8   "customer": {
9     "id": 1,
10    "firstName": "Mohammadi",
11    "lastName": "Imane"
12  }
13 }
```

- Accéder aux mêmes services mais cette fois-ci moyennant la Gateway :



The screenshot shows a web browser window with the address bar displaying `localhost:9999/ACCOUNT-SERVICE/api/accounts`. The browser's address bar also shows several tabs: 'Importé à partir de...', 'Importé à partir de...', and 'https://start.spring.i...'. The main content area displays a JSON response for all accounts, which is an array of four account objects. The JSON is as follows:

```
1 [
2   {
3     "id": "a1dce51e-9f85-4142-a0a0-7294976b7953",
4     "balance": 2320.9340199317908,
5     "createdAt": "2024-06-10",
6     "type": "CURRENT_ACCOUNT",
7     "currency": "MAD",
8     "customerId": 1,
9     "customer": null
10  },
11  {
12    "id": "cbb521d3-63c3-43f3-9642-21a6773e41a4",
13    "balance": 68574.67921308964,
14    "createdAt": "2024-06-10",
15    "type": "SAVING_ACCOUNT",
16    "currency": "MAD",
17    "customerId": 1,
18    "customer": null
19  },
20  {
21    "id": "794f3bbf-2841-4ed9-b548-3814dddbacd",
22    "balance": 52580.48923272165,
23    "createdAt": "2024-06-10",
24    "type": "CURRENT_ACCOUNT",
25    "currency": "MAD",
26    "customerId": 2,
27    "customer": null
28  },
29  {
30    "id": "da559a10-46e2-452c-8bd2-1d75c71e5cc0",
31    "balance": 49898.05516759577,
32    "createdAt": "2024-06-10",
33    "type": "SAVING_ACCOUNT",
34    "currency": "MAD",
35    "customerId": 2,
36    "customer": null
37  }
38 ]
```

