

Micro service Architecture

TP : Développement d'une application respectant l'architecture MSA avec Spring Cloud en utilisant : Spring Cloud Config, Spring Cloud Gateway, Spring Cloud Netflix Eureka Server, Spring Cloud OpenFeign et Resilience 4J.

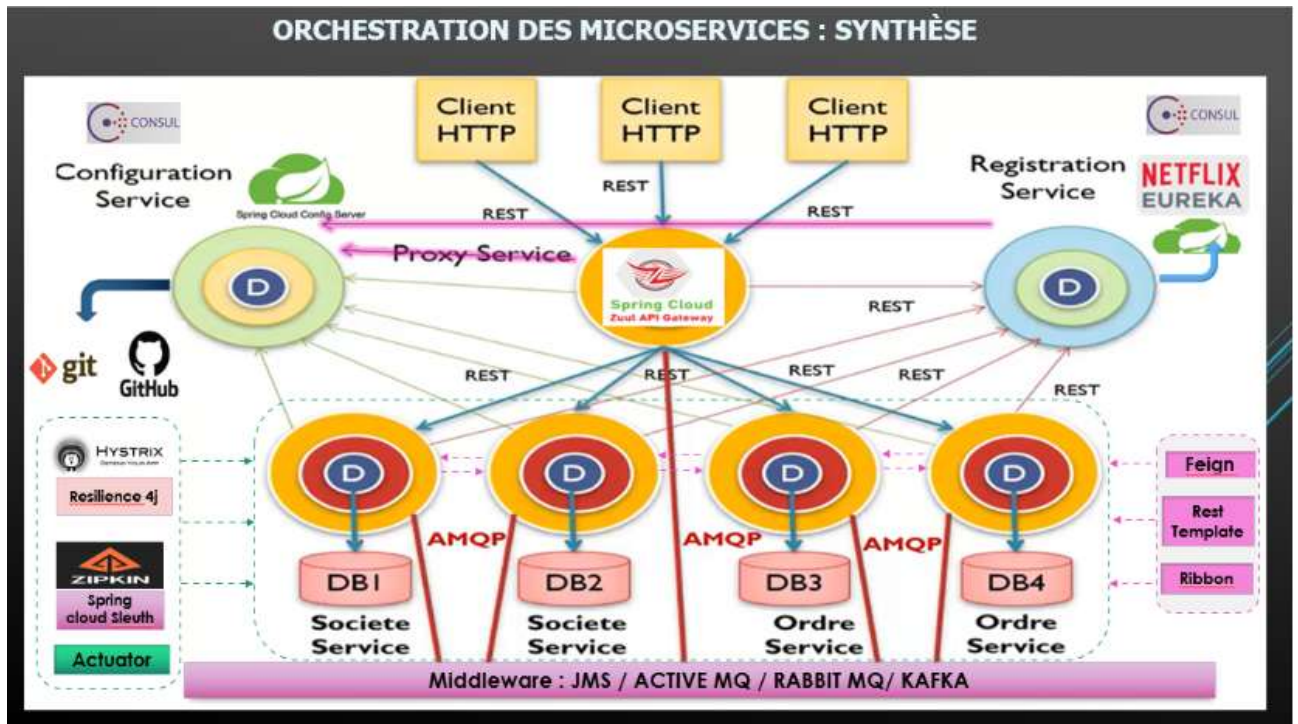
Plan

I. Objectif	3
II. Rappel de l'architecture MSA.....	3
III. Prérequis	4
IV. Architecture de notre application.....	4
V. Développement de l'application	4
a. Création du projet ainsi que les modules	4
b. Développement du MS config-service	6
c. Développement du MS discovery-service	9
d. Développement du MS gateway-service	11
e. Développement du MS customer-service.....	14
f. Développement du MS account-service	19
VI. Les tests.....	25
Conclusion :	Erreur ! Signet non défini.

I. Objectif

L'objectif de cet atelier est de montrer comment mettre en place une architecture MSA avec les Framework de Spring Cloud.

II. Rappel de l'architecture MSA



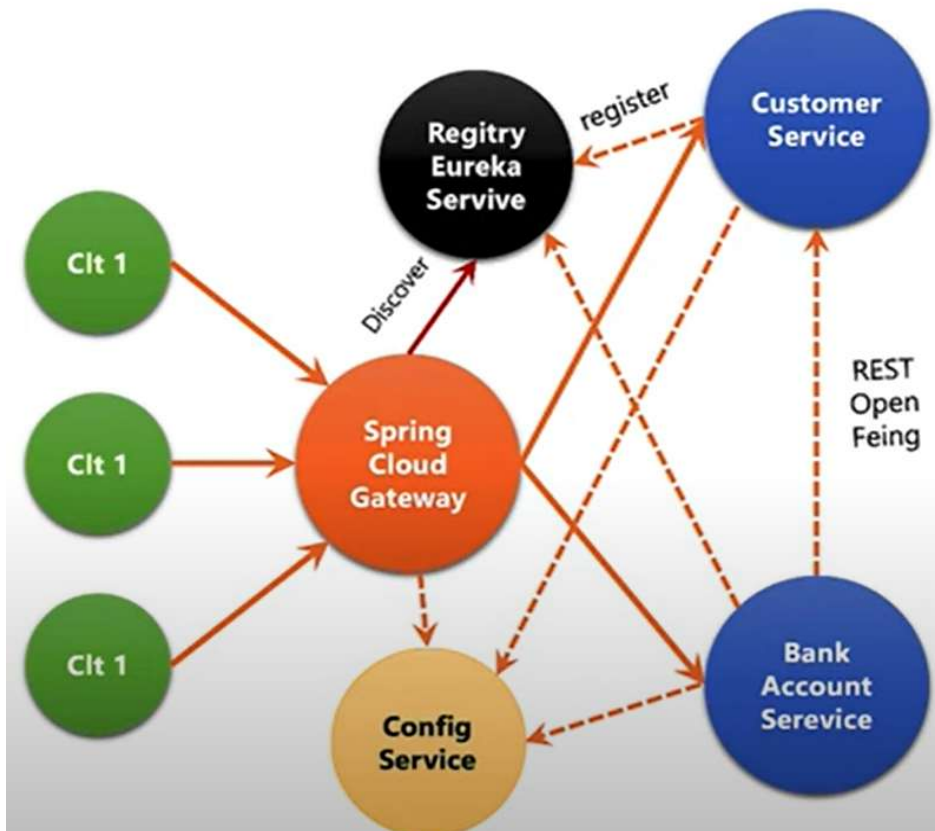
- Chaque requête émanant des clients passe par le même MS (Micro Service). Ce dernier qui joue le rôle de Gateway et ceci moyennant **Spring Cloud Gateway**.
- L'enregistrement des MS se fait moyennant **Spring Cloud Eureka Server**.
- Chaque MS s'enregistre au niveau du service d'enregistrement moyennant **Spring Cloud Eureka Client**.
- La configuration de chaque MS (Micro Service) est externalisée (par exemple sur GITHUB) moyennant **Spring Cloud Config Server**.
- Chaque MS collecte ses données de configuration moyennant **Spring Cloud Config Client**.
- Les MS peuvent communiquer entre eux moyennant **Spring Cloud OpenFeign**, ou bien **RestTemplate** ou bien **HttpClient**.
- Le Monitoring des MS se fait moyennant Spring Cloud Actuator par exemple.
- La traçabilité de chaque requête peut être réalisée par exemple via ZIPKIN.
- La gestion des pannes éventuelles peut être traitée moyennant les Framework comme **Hystrix** ou bien **Resilience 4J**.

III. Prérequis

- JDK 17.
- IntelliJ IDEA 2023.2.3 (Ultimate Edition) ou autre.
- Connexion Internet pour télécharger les dépendances via MAVEN.

IV. Architecture de notre application

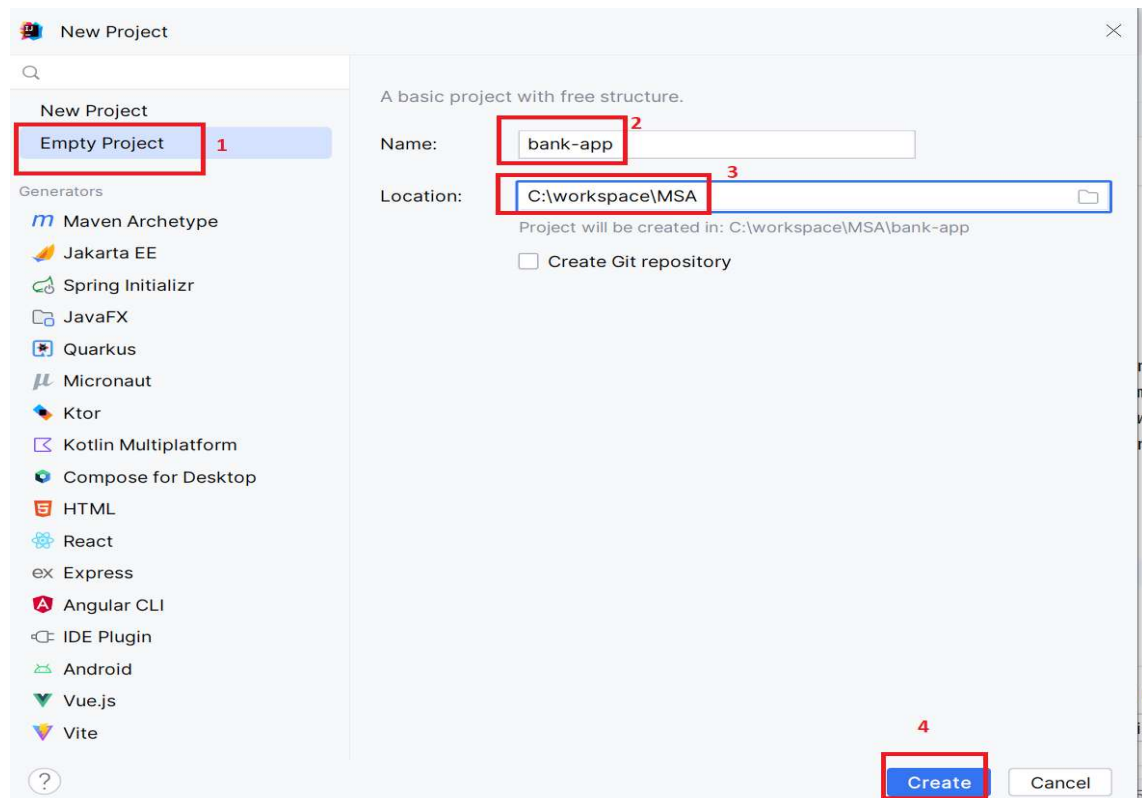
Dans cet atelier, nous allons implémenter l'architecture suivante :



V. Développement de l'application

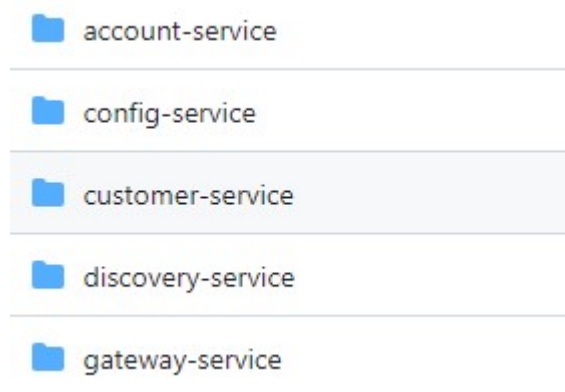
a. Création du projet ainsi que les modules

- Au niveau d'IntelliJ, créer un projet vide (Empty projet) comme expliqué ci-dessous :

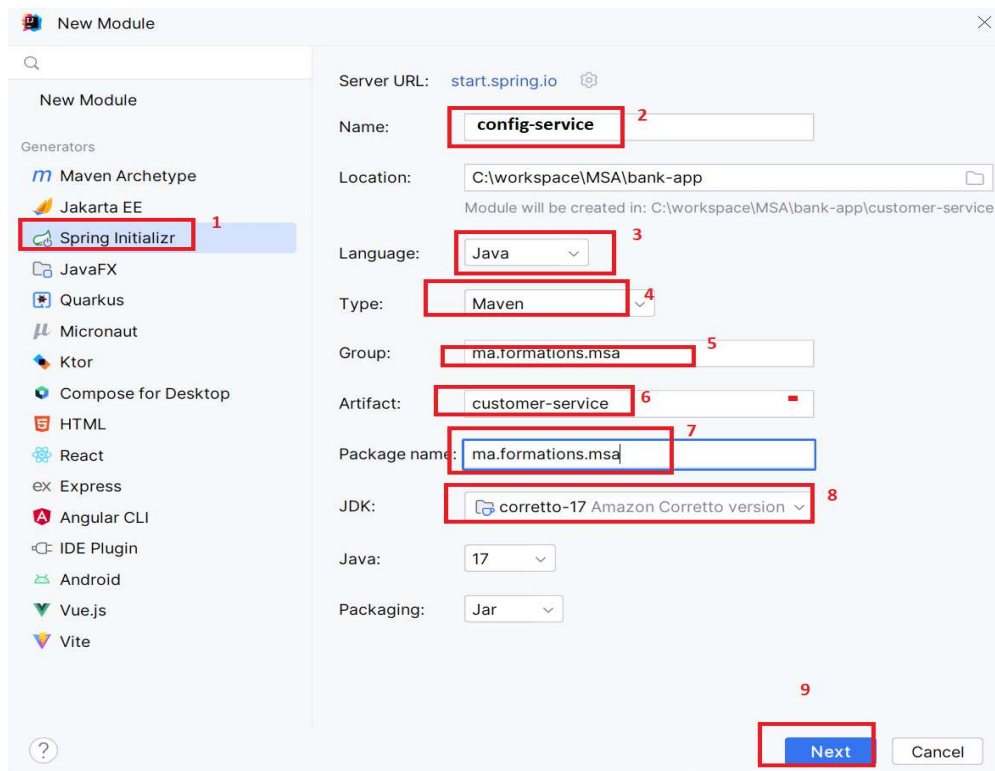


- ✓ 1- Cliquer sur « **Empty Project** ».
- ✓ 2- Dans Name : entrer le nom de votre projet, par exemple bank-app.
- ✓ 3- Préciser le dossier dans lequel sera créé votre projet.
- ✓ 4- cliquer sur le bouton **Create**.

- Dans votre projet bank-app, créer les modules suivants :



- Par exemple, pour créer le module config-service, suivre les étapes suivantes :
 - ✓ Sur votre projet, cliquer à droite de la souris ;
 - ✓ Cliquer sur **New->Module..** :



- 1- Cliquer sur « Spring Initializr ».
- 2- Dans Name, entrer le nom du service : config-service.
- 3- Dans Language, entrer Java.
- 4- Dans Type, entrer Maven.
- 5- Dans Group, entrer ma.formations.msa.
- 6- Dans Artifact, entrer customer-service.
- 7- Dans « Package name », entrer ma.foration.msa.
- 8- Dans JDK, choisir JDK 17.
- 9- Cliquer enfin sur le bouton Next.

b. Développement du MS config-service

- Ajouter les dépendances suivantes au niveau du fichier **pom.xml** :

```
<?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>config-service</artifactId>
<version>0.0.1-SNAPSHOT</version>
<name>config-service</name>
<description>config-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-config-server</artifactId>
  </dependency>
  <dependency>
    <groupId>org.springframework.cloud</groupId>
    <artifactId>spring-cloud-starter-netflix-eureka-client</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>

```

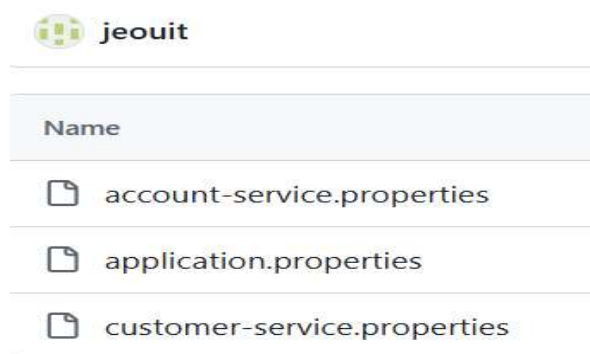
Explication :

- ✓ Remarquer que le Starter : spring-boot-starter-actuator sera rajoutée à l'ensemble des MS.
- ✓ Actuator offre les services de Monitoring. Par exemple, l'Endpoint /health (<http://localhost:8888/actuator/health>) permettra de connaître la santé du MS.

- Ajouter les lignes suivantes au niveau du fichier `application.properties` :

```
spring.application.name=config-service
server.port=8888
spring.cloud.config.server.git.uri= https://github.com/jeouit/mas-bank-config.git
```

- Par rapport à la valeur de la clé **`spring.cloud.config.server.git.uri`**, il faut configurer le lien de votre repository dans lequel Spring Cloud Config Server centralisera les données de configuration des MS. Dans cet atelier, la configuration des MS est centralisée au niveau du Repository **`msa-bank-config`** sur GITHB comme expliqué ci-dessous :



<i>application.properties</i>	<code>spring.cloud.discovery.enabled=true</code> <code>eureka.instance.prefer-ip-address=true</code> <code>spring.cloud.config.enabled=true</code> <code>springdoc.api-docs.path=/api-docs</code> <code>springdoc.swagger-ui.path=/swagger-ui.html</code> <code>springdoc.swagger-ui.enabled=true</code> <code>management.endpoints.web.exposure.include=*</code> <code>eureka.client.service-url.defaultZone=\${DISCOVERY_SERVICE_URL:http://localhost:8761/eureka}</code>
<i>account-service.properties</i>	<code>spring.datasource.url=jdbc:h2:mem:account-db</code> <code>spring.h2.console.enabled=true</code>
<i>customer-service.properties</i>	<code>spring.datasource.url=jdbc:h2:mem:customer-db</code> <code>spring.h2.console.enabled=true</code> <code>customer.service.default.location=Casablanca</code> <code>customer.service.default.currency=MAD</code> <code>customer.service.default.language=Arabic</code>

Explication :

- ✓ La propriété **`spring.cloud.discovery.enabled`** positionnée à « true » permet au MS de s'enregistrer au niveau de l'Eureka server.
- ✓ La propriété **`eureka.instance.prefer-ip-address`** positionnée à « true » permet au MS de s'enregistrer au niveau d'Eureka server avec son adresse IP.
- ✓ La propriété **`spring.cloud.config.enabled`** positionnée à « true » permet au MS d'activer sa configuration externe.

- ✓ La propriété **eureka.client.service-url.defaultZone** précise l'URL du serveur Eureka.
- ✓ Remarquer l'utilisation de la variable d'environnement `DISCOVERY_SERVICE_URL` au niveau de l'URL : `#{DISCOVERY_SERVICE_URL:http://localhost:8761/eureka}`. En effet, lors du chargement du fichier `application.properties`, Spring commence par chercher la valeur de cette variable, si il ne la trouve pas il utilise l'URL précisée après les « : ». L'utilisation des variables d'environnement est nécessaire lorsque vous déployez les MS dans des conteneurs. En effet, deux MS déployés dans deux conteneurs différents n'auront pas la même adresse IP et donc l'utilisation de **localhost** ne va pas marcher.
- ✓ La propriété **management.endpoints.web.exposure.include** positionnée à « * » permet au Framework **Actuator** d'exposer l'ensemble des Endpoint fournis par ce dernier (en nombre de ~ 24).
- ✓ Au niveau de ce Repository, nous avons configuré 03 fichiers :
 - Le fichier **application.properties** : concerne la configuration commune de l'ensemble des MS.
 - Le fichier **account-service.properties** : concerne le MS account-service.
 - Le fichier **customer-service.properties** : concerne le MS customer-service.
- Annoter la classe de démarrage par **@EnableConfigServer** comme montré ci-dessous :

```
package ma.formation.msa.configservice;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.cloud.config.server.EnableConfigServer;

@SpringBootApplication
@EnableConfigServer
public class ConfigServiceApplication {

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

c. Développement du MS discovery-service

- Ajouter les dépendances suivantes au niveau du fichier `pom.xml` :

```
<?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>discovery-service</artifactId>
<version>0.0.1-SNAPSHOT</version>
<name>discovery-service</name>
<description>discovery-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-netflix-eureka-server</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>

```

- Ajouter les lignes suivantes au niveau du fichier ***application.properties*** :

```
server.port=8761
eureka.client.register-with-eureka=false
eureka.client.fetch-registry=false
```

Explication :

- ✓ Le port par défaut de Eureka server est 8761.
- ✓ Les deux propriétés : **eureka.client.register-with-eureka** et **eureka.client.fetch-registry** ont été positionnées à « false » afin Spring Cloud Eureka Server ignore ce MS lors de l'enregistrement étant donné que le MS **discovery-service** est lui-même un serveur d'enregistrement.

- Annoter la classe de démarrage par l'annotation **@EnableEurekaServer** comme montré ci-dessous :

```
package ma.formation.msa.discovery-service;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.cloud.netflix.eureka.server.EnableEurekaServer;

@SpringBootApplication
@EnableEurekaServer
public class DiscoveryServiceApplication {

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

d. Développement du MS gateway-service

- Ajouter les dépendances suivantes au niveau du fichier pom.xml :

```
<?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.formation</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*.

- Ajouter les lignes suivantes au niveau du fichier `application.properties` :

```
spring.application.name=gateway-service
server.port=9999
spring.config.import=optional:configserver:${CONFIG_SERVER_URL:http://localhost:8888/}
```

Explication :

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

- 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);
    }
}
```

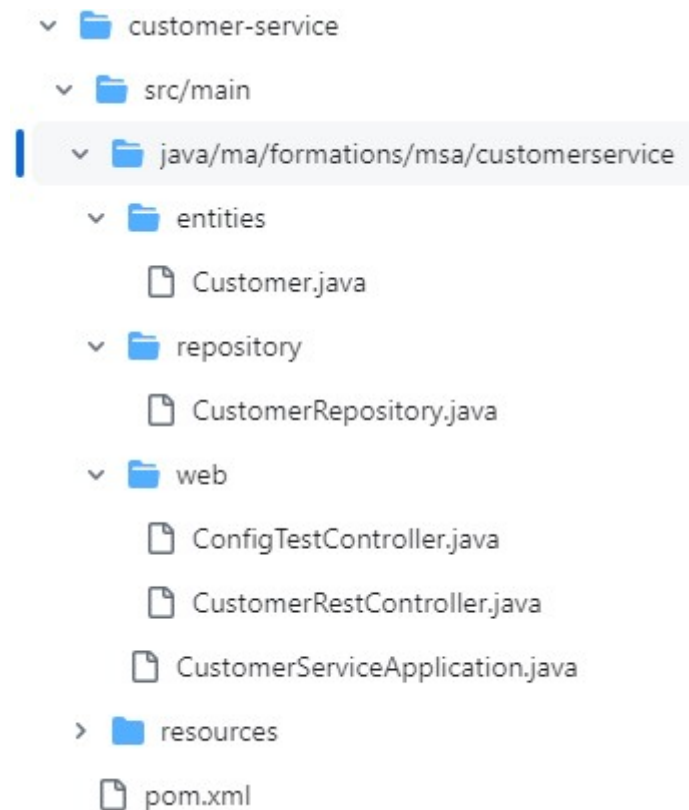
Explication :

- ✓ 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.

e. Développement du MS customer-service

- L'arborescence du module **customer-service** est la suivante :



- Ajouter les dépendances suivantes au niveau du fichier pom.xml :

```
<?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>3.1.2</version>
        <relativePath/>
    </parent>
    <groupId>ma.formations</groupId>
    <artifactId>customer-service</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <name>customer-service</name>
    <description>customer-service</description>
    <properties>
        <java.version>17</java.version>
        <spring-cloud.version>2022.0.4</spring-cloud.version>
    </properties>
</project>
```

```

<dependencies>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-actuator</artifactId>
  </dependency>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-data-jpa</artifactId>
  </dependency>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-web</artifactId>
  </dependency>
  <dependency>
    <groupId>org.springframework.cloud</groupId>
    <artifactId>spring-cloud-starter-config</artifactId>
  </dependency>
  <dependency>
    <groupId>org.springframework.cloud</groupId>
    <artifactId>spring-cloud-starter-netflix-eureka-client</artifactId>
  </dependency>

  <dependency>
    <groupId>org.springdoc</groupId>
    <artifactId>springdoc-openapi-starter-webmvc-ui</artifactId>
    <version>2.2.0</version>
  </dependency>

  <dependency>
    <groupId>com.h2database</groupId>
    <artifactId>h2</artifactId>
    <scope>runtime</scope>
  </dependency>
  <dependency>
    <groupId>org.projectlombok</groupId>
    <artifactId>lombok</artifactId>
    <optional>true</optional>
  </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>
    <configuration>
      <excludes>
        <exclude>
          <groupId>org.projectlombok</groupId>
          <artifactId>lombok</artifactId>
        </exclude>
      </excludes>
    </configuration>
  </plugin>
</plugins>
</build>
</project>

```

- Ajouter les lignes suivantes au niveau du fichier application.properties :

```

spring.application.name=customer-service
server.port=8084
spring.config.import=optional:configserver:${CONFIG_SERVER_URL:http://localhost:8888/}

```

- La classe **Customer** :

```

package ma.formations.msa.customerservice.entities;

import jakarta.persistence.Entity;
import jakarta.persistence.GeneratedValue;
import jakarta.persistence.GenerationType;
import jakarta.persistence.Id;
import lombok.*;

@Entity
@Getter @Setter
@NoArgsConstructor
@AllArgsConstructor
@Builder
public class Customer {
    @Id @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    private String firstName;
    private String lastName;
    private String email;
}

```

- L'interface **CustomerRepository** :

```

package ma.formations.msa.customerservice.repository;

import ma.formations.msa.customerservice.entities.Customer;

```



```
import org.springframework.data.jpa.repository.JpaRepository;

public interface CustomerRepository extends JpaRepository<Customer, Long> {
}
```

- La classe **CustomerRestController** :

```
package ma.formationen.msa.customerservice.web;

import ma.formationen.msa.customerservice.entities.Customer;
import ma.formationen.msa.customerservice.repository.CustomerRepository;
import org.springframework.web.bind.annotation.*;

import java.util.List;

@RestController
public class CustomerRestController {
    private CustomerRepository customerRepository;

    public CustomerRestController(CustomerRepository customerRepository) {
        this.customerRepository = customerRepository;
    }

    @GetMapping("/customers")
    public List<Customer> listCustomers(){
        return customerRepository.findAll();
    }

    @GetMapping("/customers/{id}")
    public Customer customerById(@PathVariable Long id){
        return customerRepository.findById(id).get();
    }

    @PostMapping("/customers")
    public Customer saveCustomer(@RequestBody Customer customer){
        return customerRepository.save(customer);
    }

    @PutMapping("/customers/{id}")
    public Customer updateCustomer(@RequestBody Customer customer, @PathVariable Long id){
        customer.setId(id);
        return customerRepository.save(customer);
    }

    @DeleteMapping("/customers/{id}")
    public void deleteCustomer(@PathVariable Long id){
        customerRepository.deleteById(id);
    }
}
```

- La classe **ConfigTestController** :

```
package ma.formationen.msa.customerservice.web;
```

```

import org.springframework.beans.factory.annotation.Value;
import org.springframework.cloud.context.config.annotation.RefreshScope;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RestController;

import java.util.Map;

@RestController
@RefreshScope
public class ConfigTestController {
    @Value("${customer.service.default.location}")
    private String defaultLocation;
    @Value("${customer.service.default.language}")
    private String defaultLanguage;
    @Value("${customer.service.default.currency}")
    private String defaultCurrency;
    @GetMapping("/configTest")
    public Map<String, String> config(){
        return Map.of(
            "defaultLocation",defaultLocation,
            "defaultLanguage",defaultLanguage,
            "defaultCurrency",defaultCurrency
        );
    }
}

```

- La classe de démarrage :

```

package ma.formations.msa.customerservice;

import ma.formations.msa.customerservice.entities.Customer;
import ma.formations.msa.customerservice.repository.CustomerRepository;
import org.springframework.boot.CommandLineRunner;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.context.annotation.Bean;

@SpringBootApplication
public class CustomerServiceApplication {

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

    @Bean
    CommandLineRunner commandLineRunner(CustomerRepository customerRepository){
        return args -> {
            customerRepository.save(Customer.builder()

```

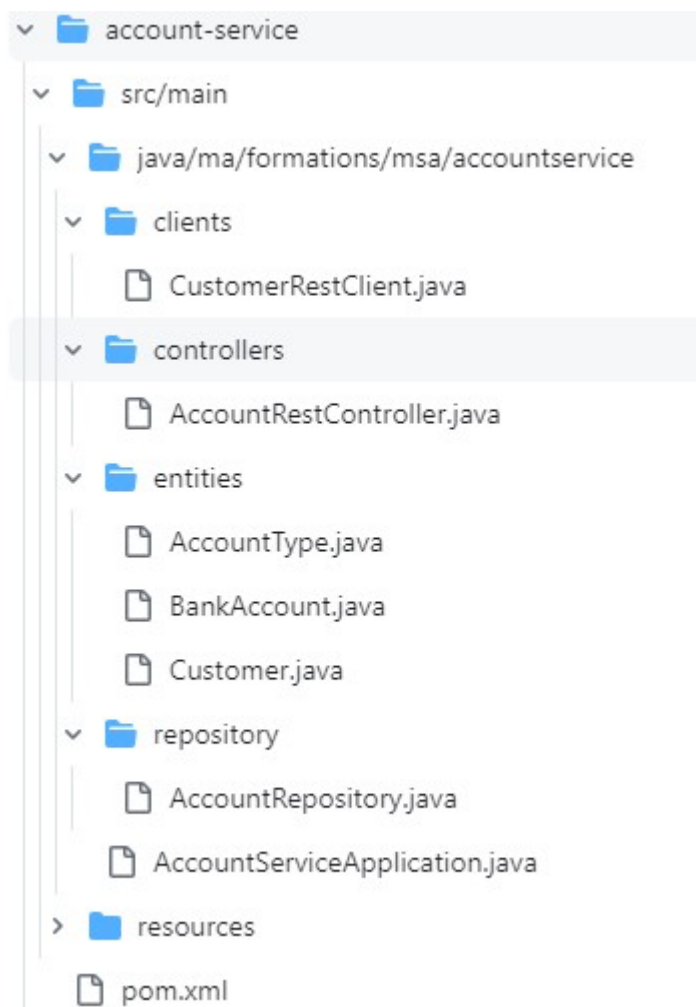
```

        .firstName("Mohammadi")
        .lastName("Imane")
        .email("imane@gmail.com")
        .build();
customerRepository.save(Customer.builder()
        .firstName("Ismaïli")
        .lastName("Aymane")
        .email("aymane@gmail.com")
        .build());
    };
}
}

```

f. Développement du MS account-service

- L'arborescence du module **account-service** est la suivante :



- Ajouter les dépendances suivantes au niveau du fichier pom.xml :

```
<?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>account-service</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <name>account-service</name>
    <description>account-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-data-jpa</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>
        <dependency>
            <groupId>org.springframework.cloud</groupId>
            <artifactId>spring-cloud-starter-openfeign</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework.cloud</groupId>
            <artifactId>spring-cloud-starter-config</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-circuitbreaker-resilience4j</artifactId>
        </dependency>
    </dependencies>
</project>
```

```
</dependency>
```

```
<dependency>
  <groupId>com.h2database</groupId>
  <artifactId>h2</artifactId>
  <scope>runtime</scope>
</dependency>
<dependency>
  <groupId>org.projectlombok</groupId>
  <artifactId>lombok</artifactId>
  <optional>true</optional>
</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>
      <configuration>
        <excludes>
          <exclude>
            <groupId>org.projectlombok</groupId>
            <artifactId>lombok</artifactId>
          </exclude>
        </excludes>
      </configuration>
    </plugin>
  </plugins>
</build>

</project>
```

- Le fichier application.properties :

```
spring.application.name=account-service
server.port=8083
spring.config.import=optional:configserver:${CONFIG_SERVER_URL:http://localhost:8888/}
```

- La classe **Customer** :

```
package ma.formationen.msa.accountservice.entities;

import lombok.*;

@NoArgsConstructor @AllArgsConstructor @Getter @Setter @Builder @ToString
public class Customer {
    private Long id;
    private String firstName;
    private String lastName;
}
```

- La classe **BankAccount** :

```
package ma.formationen.msa.accountservice.entities;

import jakarta.persistence.*;
import lombok.*;

import java.time.LocalDate;
@Getter @Setter @AllArgsConstructor @NoArgsConstructor @Builder
@Entity
public class BankAccount {
    @Id
    private String id;
    private Double balance;
    private LocalDate createdAt;
    @Enumerated(EnumType.STRING)
    private AccountType type;
    private String currency;
    private Long customerId;
    @Transient
    private Customer customer;
}
```

- La classe **AccountType** :

```
package ma.formationen.msa.accountservice.entities;

public enum AccountType {
    CURRENT_ACCOUNT, SAVING_ACCOUNT
}
```

- L'interface **AccountRepository** :

```

package ma.formations.msa.accountservice.repository;

import ma.formations.msa.accountservice.entities.BankAccount;
import org.springframework.data.jpa.repository.JpaRepository;

public interface AccountRepository extends JpaRepository<BankAccount,String> {
}

```

- La classe **CustomerRestClient** :

```

package ma.formations.msa.accountservice.clients;

import io.github.resilience4j.circuitbreaker.annotation.CircuitBreaker;
import ma.formations.msa.accountservice.entities.Customer;
import org.springframework.cloud.openfeign.FeignClient;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;

import java.util.List;

@FeignClient(name = "CUSTOMER-SERVICE")
public interface CustomerRestClient {
    @GetMapping("/customers/{id}")
    @CircuitBreaker(name = "customerService", fallbackMethod = "getDefaultCustomer")
    Customer getCustomerById(@PathVariable Long id);
    @GetMapping("/customers")
    @CircuitBreaker(name = "customerService", fallbackMethod = "getDefaultCustomers")
    List<Customer> getAllCustomers();

    default Customer getDefaultCustomer(Long id, Exception e){
        return Customer.builder()
            .id(id)
            .firstName("Source not available")
            .lastName("Source Not Available")
            .build();
    }
    default List<Customer> getDefaultCustomers(Exception e){
        return List.of();
    }
}

```

- La classe **AccountRestController** :

```

package ma.formations.msa.accountservice.controllers;

import ma.formations.msa.accountservice.clients.CustomerRestClient;
import ma.formations.msa.accountservice.entities.BankAccount;
import ma.formations.msa.accountservice.entities.Customer;
import ma.formations.msa.accountservice.repository.AccountRepository;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;

```

```

import java.util.List;
import java.util.Map;

@RestController
@RequestMapping("/api")
public class AccountRestController {
    private AccountRepository accountRepository;
    private CustomerRestClient customerRestClient;

    public AccountRestController(AccountRepository accountRepository, CustomerRestClient customerRestClient) {
        this.accountRepository = accountRepository;
        this.customerRestClient = customerRestClient;
    }

    @GetMapping("/accounts/{accountId}")
    public ResponseEntity getBankAccountById(@PathVariable String accountId){
        BankAccount bankAccount = accountRepository.findById(accountId).orElse(null);
        if(bankAccount==null) {
            return ResponseEntity.internalServerError().body(Map.of("errorMessage","account not found"));
        }
        Customer customer=customerRestClient.getCustomerById(bankAccount.getCustomerId());
        bankAccount.setCustomer(customer);
        return ResponseEntity.ok(bankAccount);
    }
    @GetMapping("/accounts")
    public List<BankAccount> accountList(){
        return accountRepository.findAll();
    }
}

```

- La classe de démarrage :

```

package ma.formations.msa.accountservice;

import ma.formations.msa.accountservice.clients.CustomerRestClient;
import ma.formations.msa.accountservice.entities.AccountType;
import ma.formations.msa.accountservice.entities.BankAccount;
import ma.formations.msa.accountservice.entities.Customer;
import ma.formations.msa.accountservice.repository.AccountRepository;
import org.springframework.boot.CommandLineRunner;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.cloud.openfeign.EnableFeignClients;
import org.springframework.context.annotation.Bean;

import java.time.LocalDate;
import java.util.List;
import java.util.UUID;

@SpringBootApplication
@EnableFeignClients

```



```

public class AccountServiceApplication {

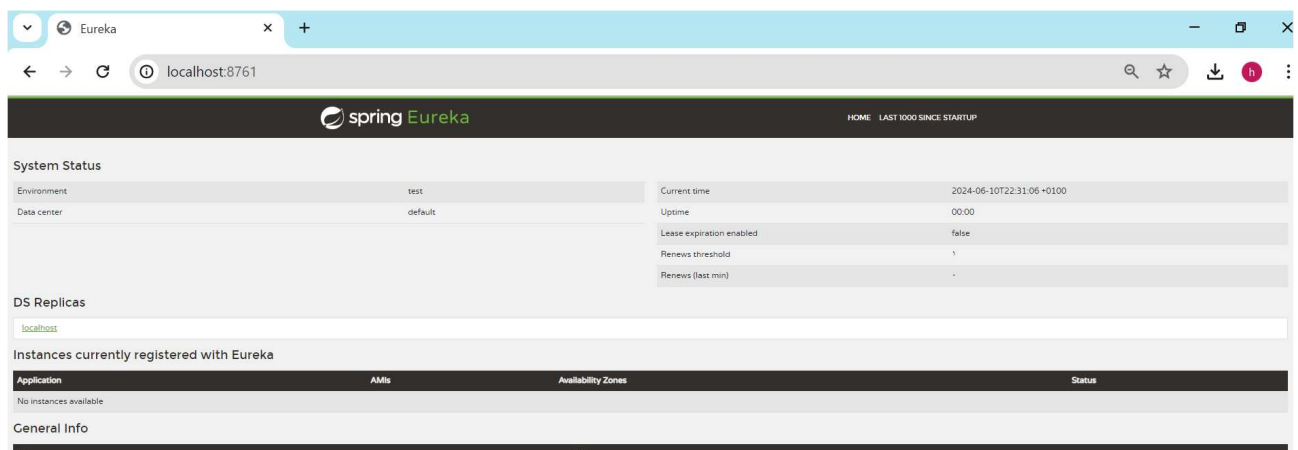
    public static void main(String[] args) {
        SpringApplication.run(AccountServiceApplication.class, args);
    }

    @Bean
    CommandLineRunner start(AccountRepository accountRepository, CustomerRestClient customerRestClient){
        return args -> {
            List<Customer> allCustomers = customerRestClient.getAllCustomers();
            allCustomers.forEach(customer -> {
                for (AccountType accountType:AccountType.values()){
                    BankAccount bankAccount = BankAccount.builder()
                        .id(UUID.randomUUID().toString())
                        .balance(Math.random()*80000)
                        .createdAt(LocalDate.now())
                        .type(accountType)
                        .currency("MAD")
                        .customerId(customer.getId())
                        .build();
                    accountRepository.save(bankAccount);
                }
            });
        };
    }
}

```

VI. Les tests

- Démarrer le service : discovery-service et accéder au lien <http://localhost:8761> :



- Remarquer qu'aucune instance n'est enregistrée au niveau d'Eureka.
- Démarrer le service : config-service et accéder au lien <http://localhost:8761>. Vérifier que le service config-service a été bien enregistré au niveau du serveur Eureka :

Instances currently registered with Eureka

Application	AMIs	Availability Zones	Status
CONFIG-SERVICE	n/a (')	(')	UP (') - host.docker.internal:config-service:8888

- Démarrer le service : gateway-service et vérifier que le service gateway-service a été bien enregistré au niveau d'Eureka :

Instances currently registered with Eureka

Application	AMIs	Availability Zones	Status
CONFIG-SERVICE	n/a (')	(')	UP (') - host.docker.internal:config-service:8888
GATEWAY-SERVICE	n/a (')	(')	UP (') - host.docker.internal:gateway-service:9999

- Démarrer les deux services : customer-service et account-service et vérifier que ces services ont été bien enregistrés au niveau d'Eureka :

Instances currently registered with Eureka

Application	AMIs	Availability Zones	Status
ACCOUNT-SERVICE	n/a (')	(')	UP (') - host.docker.internal:account-service:8083
CONFIG-SERVICE	n/a (')	(')	UP (') - host.docker.internal:config-service:8888
CUSTOMER-SERVICE	n/a (')	(')	UP (') - host.docker.internal:customer-service:8084
GATEWAY-SERVICE	n/a (')	(')	UP (') - host.docker.internal:gateway-service:9999

- Accéder au lien <http://localhost:8084/customers> pour consulter la liste des clients :



```
1  [  
2    {  
3      "id": 1,  
4      "firstName": "Mohammadi",  
5      "lastName": "Imane",  
6      "email": "imane@gmail.com"  
7    },  
8    {  
9      "id": 2,  
10     "firstName": "Ismaili",  
11     "lastName": "Aymane",  
12     "email": "aymane@gmail.com"  
13   }  
14 ]
```

- 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 ]
```

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

```
localhost:8083/api/accounts/a1dce51e-9f85-4142-a0a0-7294976b7953

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 :

```
localhost:9999/ACCOUNT-SERVICE/api/accounts

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 ]
```

```
localhost:9999/ACCOUNT-SERVICE/api/accounts/a1dce51e-9f85-4142-a0a0-7294976b7953

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 }
```

```
localhost:9999/CUSTOMER-SERVICE/customers

1 [
2   {
3     "id": 1,
4     "firstName": "Mohammadi",
5     "lastName": "Imane",
6     "email": "imane@gmail.com"
7   },
8   {
9     "id": 2,
10    "firstName": "Ismaili",
11    "lastName": "Aymane",
12    "email": "aymane@gmail.com"
13  }
14 ]
```

```
localhost:9999/CUSTOMER-SERVICE/customers/1

1 {
2   "id": 1,
3   "firstName": "Mohammadi",
4   "lastName": "Imane",
5   "email": "imane@gmail.com"
6 }
```

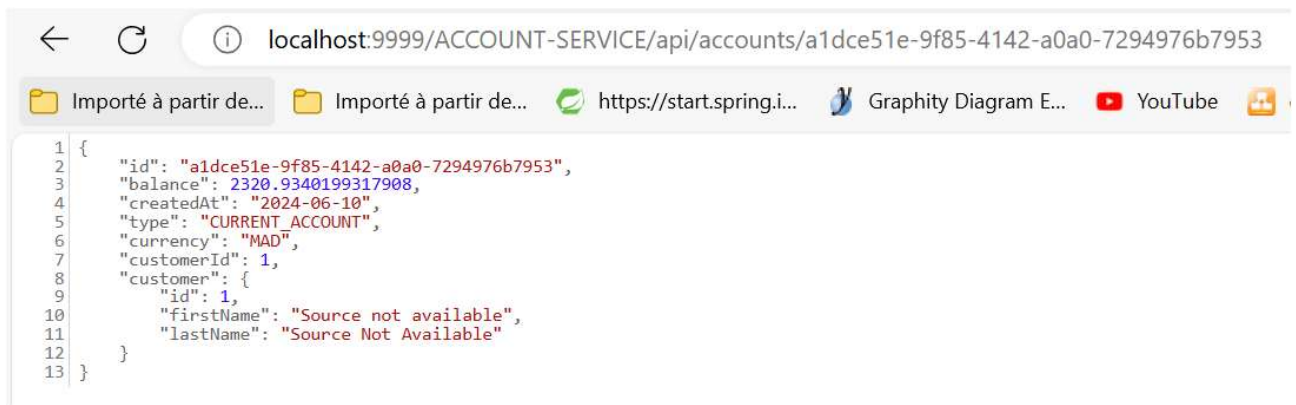
Noter que le nom du service au niveau de l'URL en utilisant la Gateway doit être en majuscule :

<http://localhost:9999/CUSTOMER-SERVICE/customers/1>

Arrêter le service customer-service et accéder au lien :

<http://localhost:9999/ACCOUNT-SERVICE/api/accounts/a1dce51e-9f85-4142-a0a0-7294976b7953>

et vérifier que le principe de **Circuit Breaker** a été bien exécuté. En effet, la fallbackMethod définie au niveau de la classe **CustomerRestClient** a été exécutée grâce au Framework **Resilience 4J** utilisé.

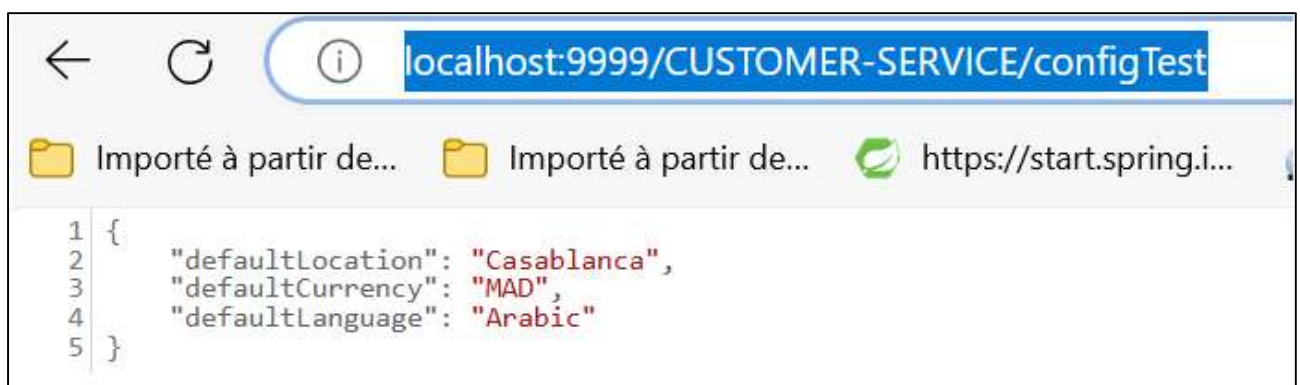


The screenshot shows a web browser window with the address bar displaying `localhost:9999/ACCOUNT-SERVICE/api/accounts/a1dce51e-9f85-4142-a0a0-7294976b7953`. The browser's developer tools are open, showing a JSON response. The JSON object contains the following fields: `id` (a1dce51e-9f85-4142-a0a0-7294976b7953), `balance` (2320.9340199317908), `createdAt` (2024-06-10), `type` (CURRENT_ACCOUNT), `currency` (MAD), `customerId` (1), and `customer` (an object with `id` 1, `firstName` "Source not available", and `lastName` "Source Not Available").

```
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": "Source not available",
11    "lastName": "Source Not Available"
12  }
13 }
```

Accéder au lien <http://localhost:9999/CUSTOMER-SERVICE/configTest> et vérifier que vous avez accédé aux valeurs définies dans le fichier customer-service.properties au niveau du Repository GITHUB :

```
customer.service.default.location=Casablanca
customer.service.default.currency=MAD
customer.service.default.language=Arabic
```



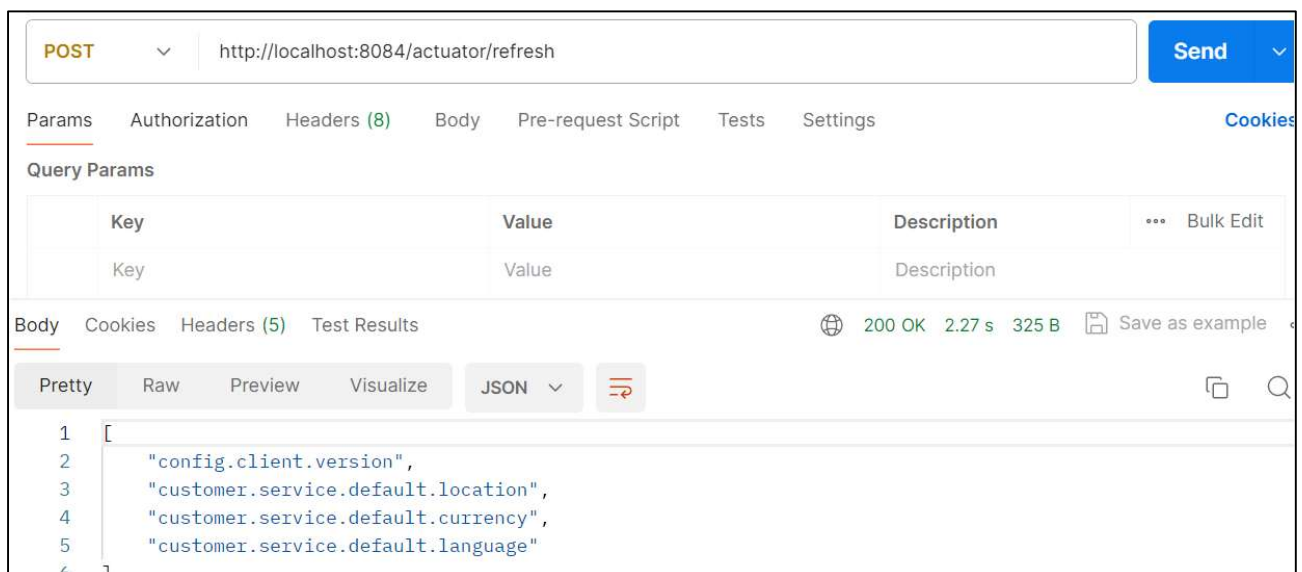
The screenshot shows a web browser window with the address bar displaying `localhost:9999/CUSTOMER-SERVICE/configTest`. The browser's developer tools are open, showing a JSON response. The JSON object contains the following fields: `defaultLocation` (Casablanca), `defaultCurrency` (MAD), and `defaultLanguage` (Arabic).

```
1 {
2   "defaultLocation": "Casablanca",
3   "defaultCurrency": "MAD",
4   "defaultLanguage": "Arabic"
5 }
```

Dans le Repository « msa-bank-config », modifier les propriétés du service customer-service, par exemple :

```
1  spring.datasource.url=jdbc:h2:mem:customer-db
2  spring.h2.console.enabled=true
3  customer.service.default.location=Paris
4  customer.service.default.currency=EUR
5  customer.service.default.language=French
6  management.endpoints.web.exposure.include=*
```

Au niveau de postman, envoyer une méthode POST au lien <http://localhost:8084/actuator/refresh> comme expliqué ci-dessous :



Accéder à nouveau au lien <http://localhost:8084/configTest> et vérifier que les données ont été bien rafraîchies :



The screenshot shows a web browser window with the address bar displaying `localhost:8084/configTest`. Below the address bar, there are two tabs labeled "Importé à partir de...". The main content area displays a JSON configuration with line numbers 1 through 5 on the left. The JSON object contains three key-value pairs: `"defaultCurrency": "EUR"`, `"defaultLanguage": "Frensh"`, and `"defaultLocation": "Paris"`.

```
1 {  
2   "defaultCurrency": "EUR",  
3   "defaultLanguage": "Frensh",  
4   "defaultLocation": "Paris"  
5 }
```