ECOLE MAROCAINE DES SCIENCES DE L'INGENIEUR
Membre de HONORIS UNITED UNIVERSITIES

RAPPORT D'ACTIVITÉ PRATIQUE

MICRO-SERVICES AVEC SPRING CLOUD GATEWAY

RÉALISÉ PAR : EL KHOULALY YOUSSEF
ENCADRÉ PAR : MR. YOUSSFI MOHAMED

5EME IIR G1

TABLE DES MATIERES

I. INTRODUCTION

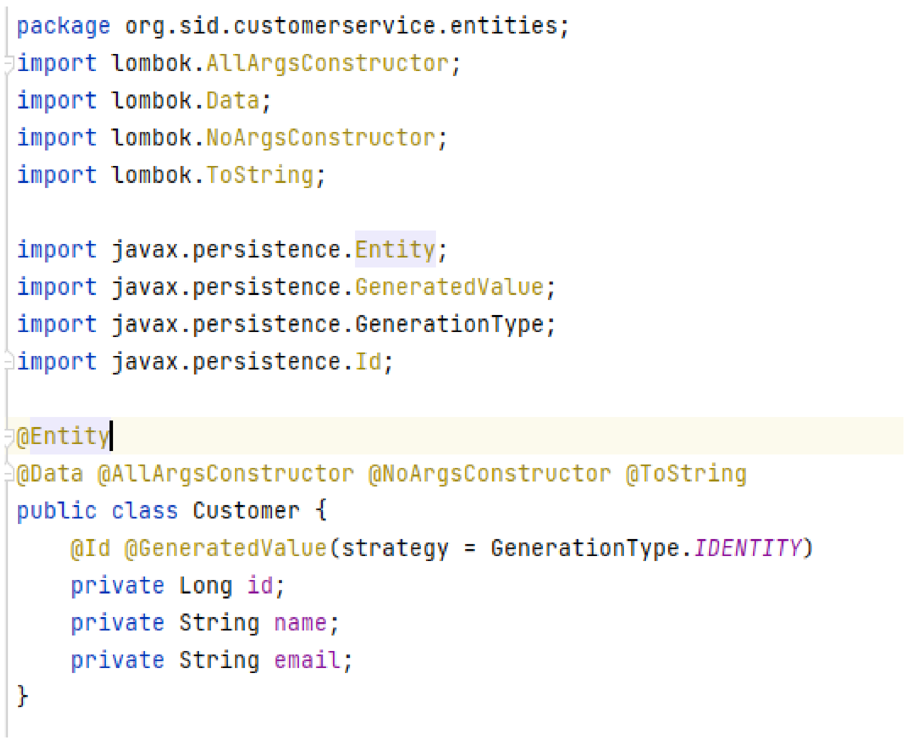
JAKARTA EE : (Java Entreprise Edition) est une spécification pour la plate-forme Java d'Oracle, destinée aux applications d'entreprise.
La plate-forme étend Java Platform, Standard Edition (Java SE) en fournissant une API de mapping objet-relationnel, des architectures distribuées et multitiers, et des services web3. La plate-forme se fonde principalement sur des composants modulaires exécutés sur un serveur d'applications.

Spring : est un framework open source pour construire et définir l'infrastructure
d'une application Java3, dont il facilite le développement et les tests.
En 2004, Rod Johnson a écrit le livre Expert One-on-One J2EE Design and Development4 qui explique les raisons de la création de Spring.
Spring Boot : est un framework qui facilite le développement d'applications fon-dées sur Spring en offrant des outils permettant d'obtenir une application packagée en jar , totalement autonome. Ce qui nous intéresse particulièrement, puisque nous essayons de développer des Microservices !

Spring Cloud fournit tous les services techniques nécessaires à la mise en place d'une architecture micro-service. Généralement associé à Spring Boot, il permet de rapidement composer des applications à partir de services unitaires et de les déployer sur une architecture de production nécessitant scalabilité et monitoring en temps- réel.

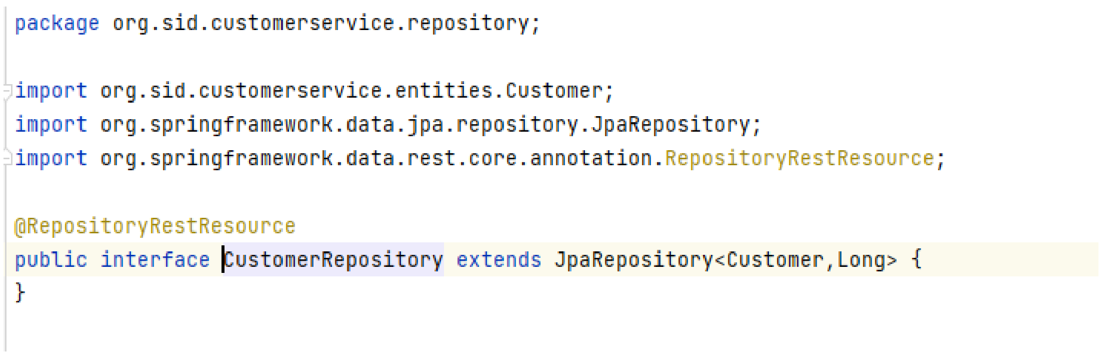
1. Micro service Customer-service

Entity Customer



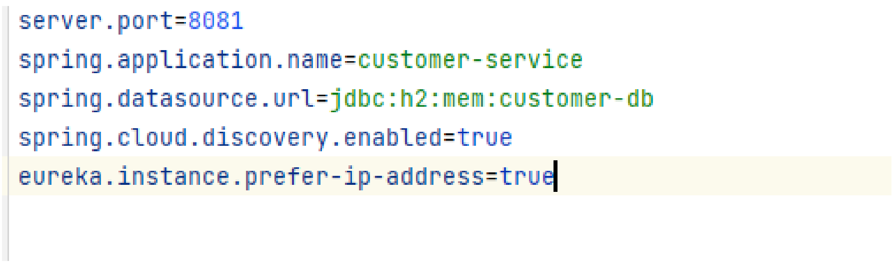
```java
package org.sid.customerservice.entities;
import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;
import lombok.ToString;

import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;

@Entity
@Data @AllArgsConstructor @NoArgsConstructor @ToString
public class Customer {
    @Id @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    private String name;
    private String email;
}
```

Customer Repository



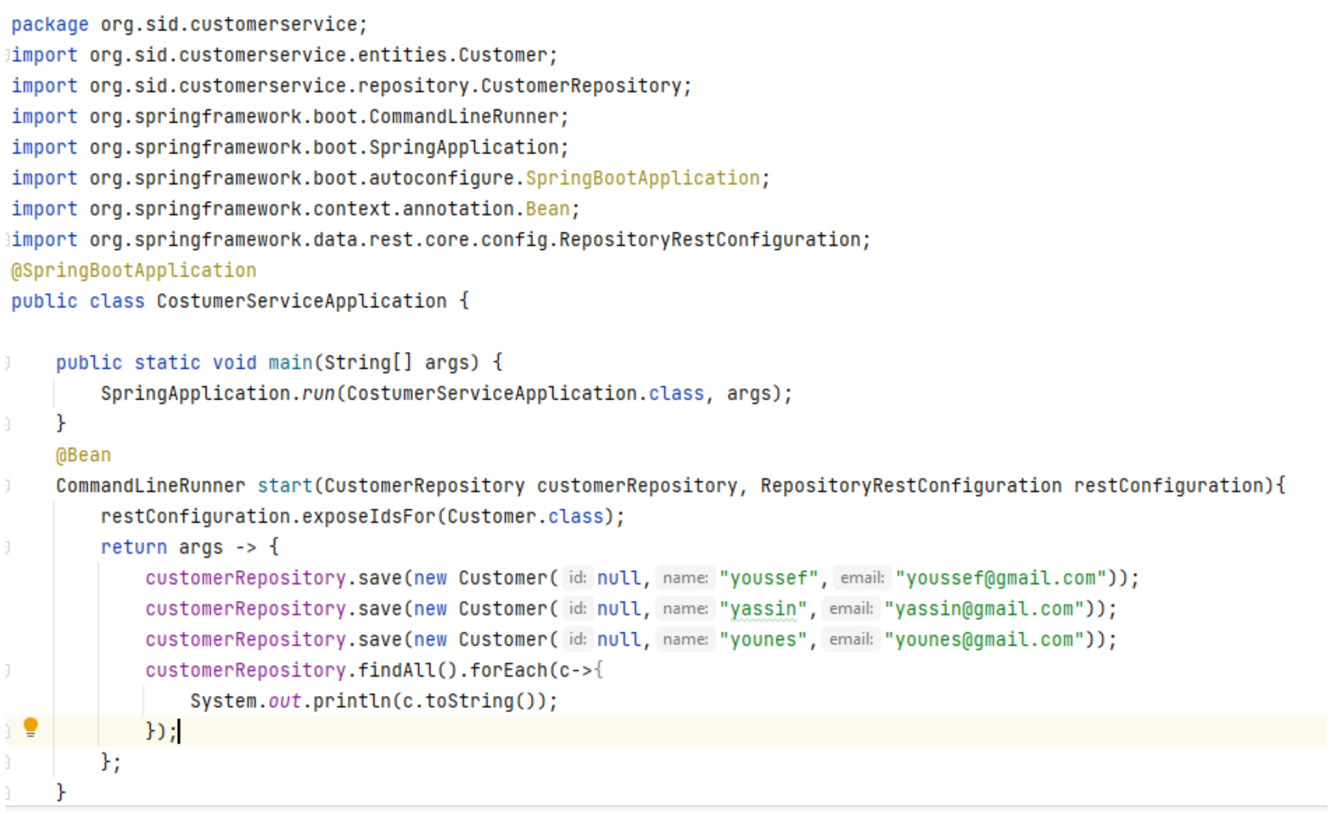
```java
package org.sid.customerservice.repository;

import org.sid.customerservice.entities.Customer;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.data.rest.core.annotation.RepositoryRestResource;

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

application properties



```properties
server.port=8081
spring.application.name=customer-service
spring.datasource.url=jdbc:h2:mem:customer-db
spring.cloud.discovery.enabled=true
eureka.instance.prefer-ip-address=true
```

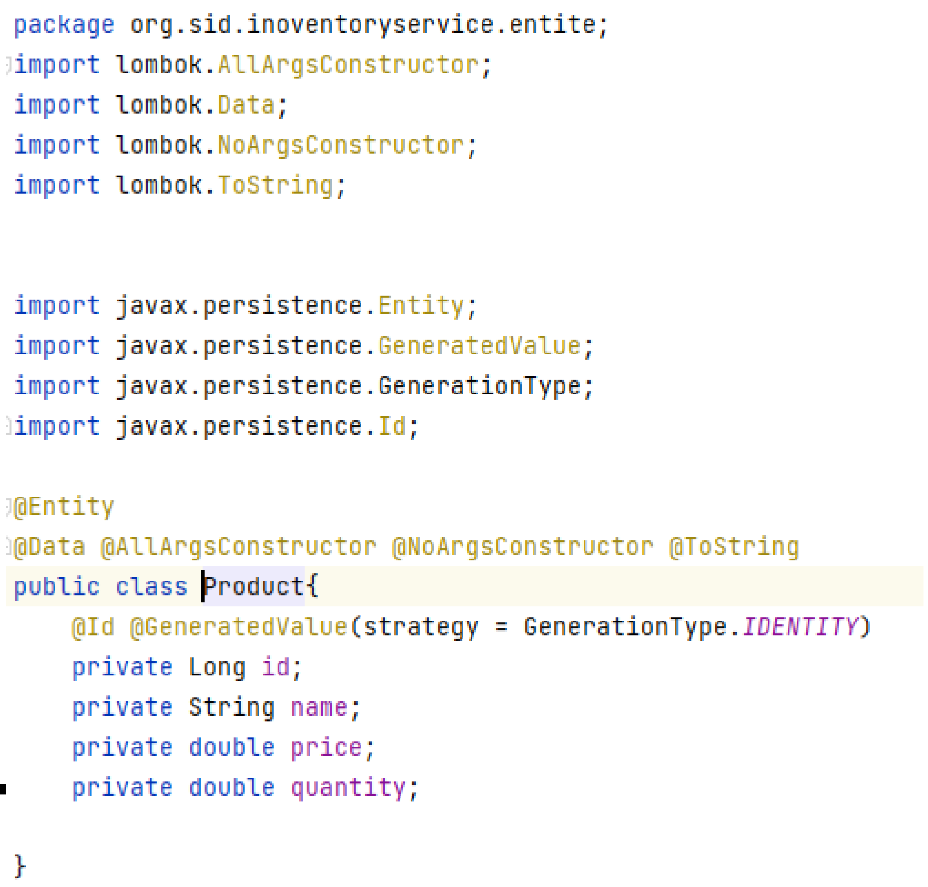
Costumer Service Application



```java
package org.sid.customerservice;
import org.sid.customerservice.entities.Customer;
import org.sid.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;
import org.springframework.data.rest.core.config.RepositoryRestConfiguration;
@SpringBootApplication
public class CostumerServiceApplication {

    public static void main(String[] args) {
        SpringApplication.run(CostumerServiceApplication.class, args);
    }
    @Bean
    CommandLineRunner start(CustomerRepository customerRepository, RepositoryRestConfiguration restConfiguration){
        restConfiguration.exposeIdsFor(Customer.class);
        return args -> {
            customerRepository.save(new Customer( id: null, name: "youssef", email: "youssef@gmail.com"));
            customerRepository.save(new Customer( id: null, name: "yassin", email: "yassin@gmail.com"));
            customerRepository.save(new Customer( id: null, name: "younes", email: "younes@gmail.com"));
            customerRepository.findAll().forEach(c->{
                System.out.println(c.toString());
            });
        };
    }
```

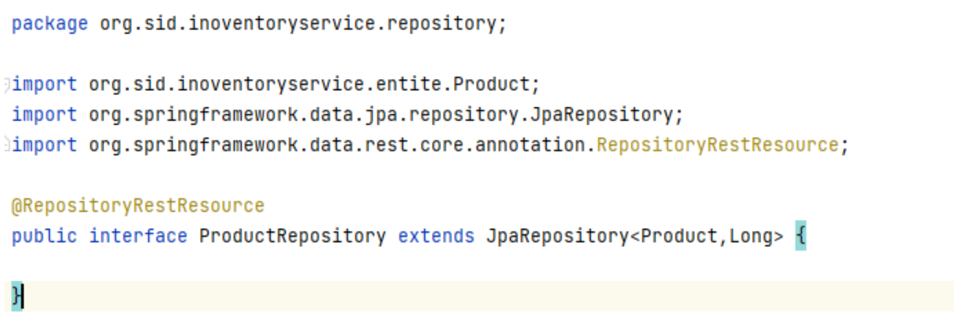
2. Micro service Inoventory service

Entity product



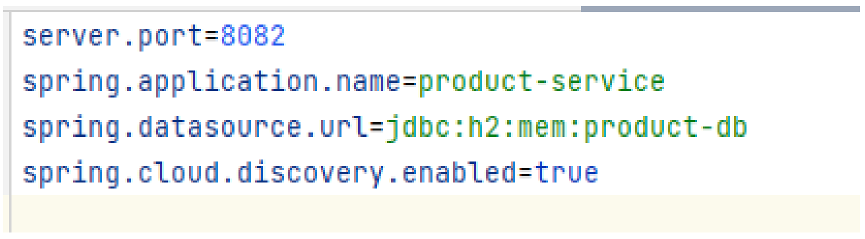
```java
package org.sid.inoventoryservice.entite;
import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;
import lombok.ToString;


import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;

@Entity
@Data @AllArgsConstructor @NoArgsConstructor @ToString
public class Product{
    @Id @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    private String name;
    private double price;
    private double quantity;

}
```

product Repository



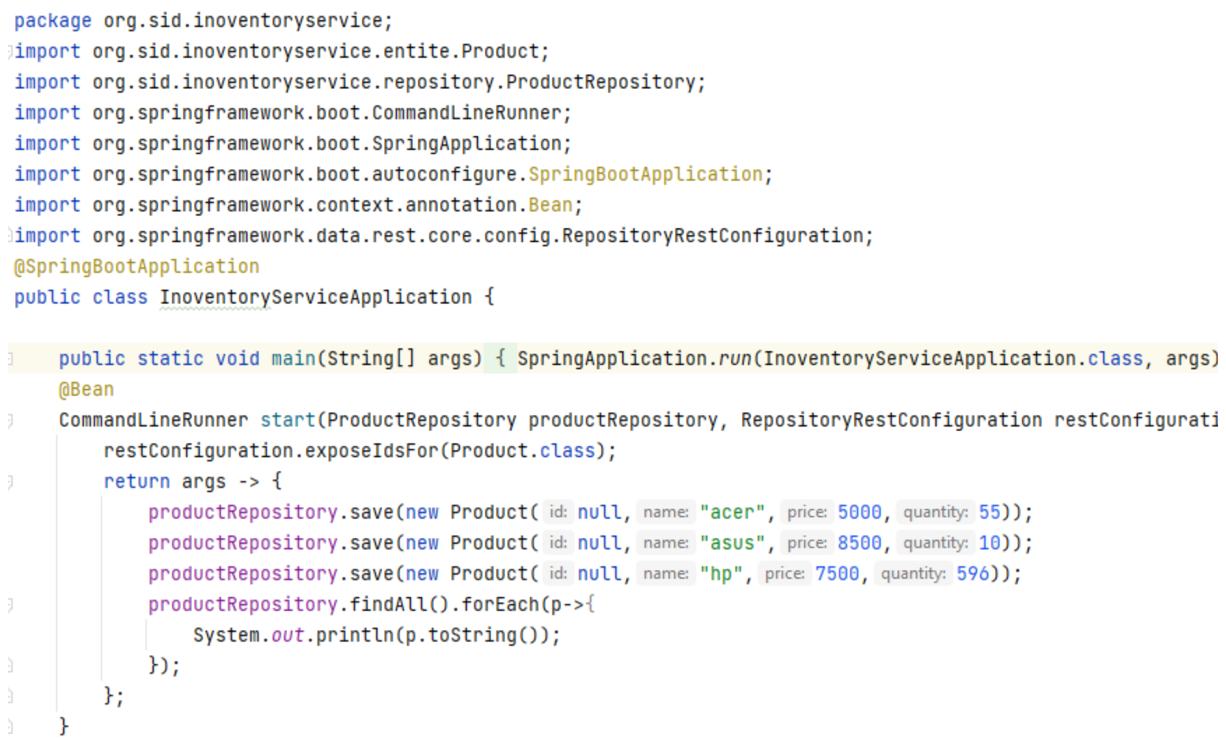
```
package org.sid.inoventoryservice.repository;

import org.sid.inoventoryservice.entite.Product;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.data.rest.core.annotation.RepositoryRestResource;

@RepositoryRestResource
public interface ProductRepository extends JpaRepository<Product,Long> {

}
```

application properties



```
server.port=8082
spring.application.name=product-service
spring.datasource.url=jdbc:h2:mem:product-db
spring.cloud.discovery.enabled=true
```

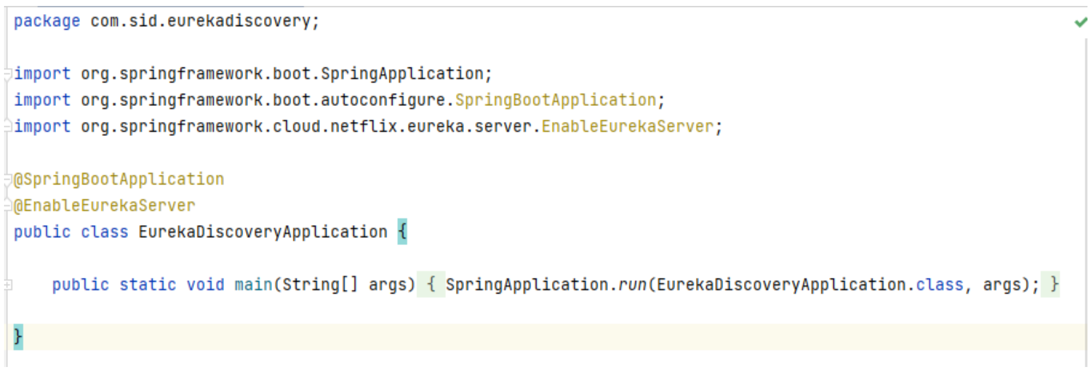
Product Service Application



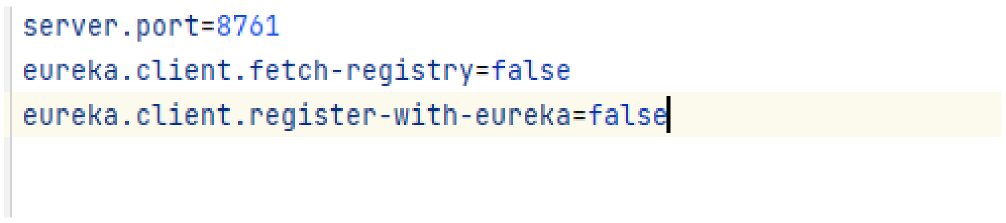
```
package org.sid.inoventoryservice;
import org.sid.inoventoryservice.entite.Product;
import org.sid.inoventoryservice.repository.ProductRepository;
import org.springframework.boot.CommandLineRunner;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.context.annotation.Bean;
import org.springframework.data.rest.core.config.RepositoryRestConfiguration;
@SpringBootApplication
public class InoventoryServiceApplication {

    public static void main(String[] args) { SpringApplication.run(InoventoryServiceApplication.class, args)
    @Bean
    CommandLineRunner start(ProductRepository productRepository, RepositoryRestConfiguration restConfigurati
        restConfiguration.exposeIdsFor(Product.class);
        return args -> {
            productRepository.save(new Product( id: null, name: "acer", price: 5000, quantity: 55));
            productRepository.save(new Product( id: null, name: "asus", price: 8500, quantity: 10));
            productRepository.save(new Product( id: null, name: "hp", price: 7500, quantity: 596));
            productRepository.findAll().forEach(p->{
                System.out.println(p.toString());
            });
        };
    }
```

3.EUREKA DISCOVERY

EUREKA DISCOVERY APPLICATION

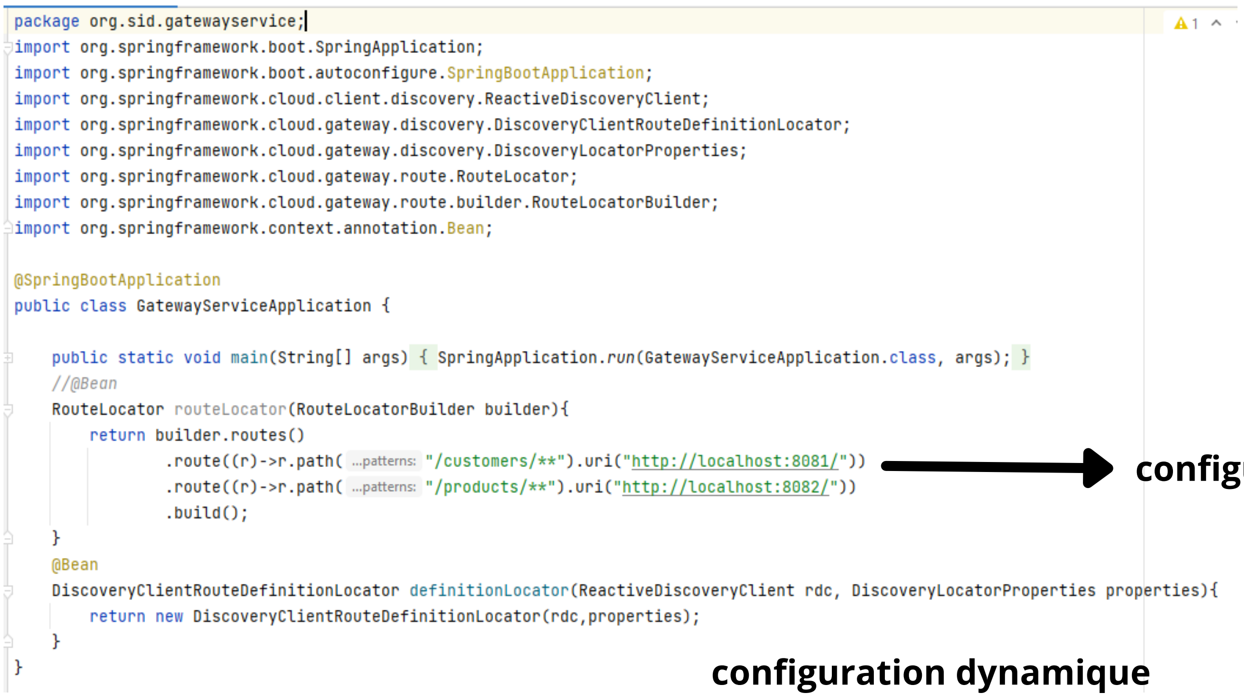

```java
package com.sid.eurekadiscovery;

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

@SpringBootApplication
@EnableEurekaServer
public class EurekaDiscoveryApplication {

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

}
```

application properties


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

4. Micro service Gateway service

Gateway Service Application

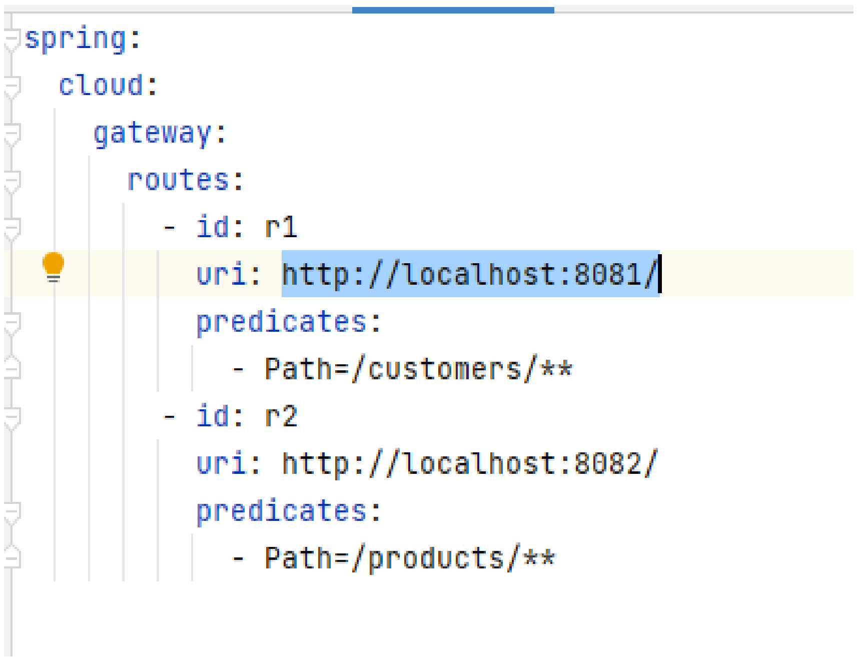


```java
package org.sid.gatewayservice;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.cloud.client.discovery.ReactiveDiscoveryClient;
import org.springframework.cloud.gateway.discovery.DiscoveryClientRouteDefinitionLocator;
import org.springframework.cloud.gateway.discovery.DiscoveryLocatorProperties;
import org.springframework.cloud.gateway.route.RouteLocator;
import org.springframework.cloud.gateway.route.builder.RouteLocatorBuilder;
import org.springframework.context.annotation.Bean;

@SpringBootApplication
public class GatewayServiceApplication {

    public static void main(String[] args) { SpringApplication.run(GatewayServiceApplication.class, args); }
    //@Bean
    RouteLocator routeLocator(RouteLocatorBuilder builder){
        return builder.routes()
                .route((r)->r.path( ...patterns: "/customers/**").uri("http://localhost:8081/"))
                .route((r)->r.path( ...patterns: "/products/**").uri("http://localhost:8082/"))
                .build();
    }
    @Bean
    DiscoveryClientRouteDefinitionLocator definitionLocator(ReactiveDiscoveryClient rdc, DiscoveryLocatorProperties properties){
        return new DiscoveryClientRouteDefinitionLocator(rdc,properties);
    }
}
```

configuration Java

configuration dynamique

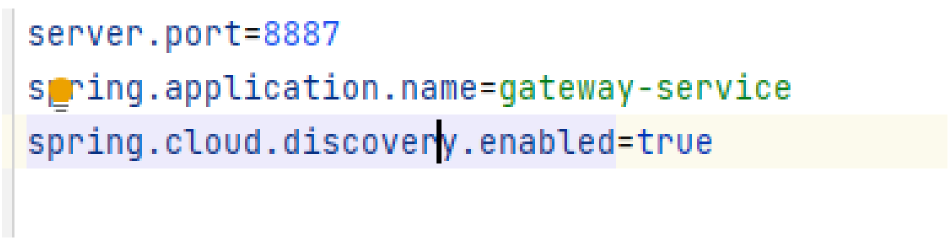
application yml



```yaml
spring:
  cloud:
    gateway:
      routes:
        - id: r1
          uri: http://localhost:8081/
          predicates:
            - Path=/customers/**
        - id: r2
          uri: http://localhost:8082/
          predicates:
            - Path=/products/**
```

application properties

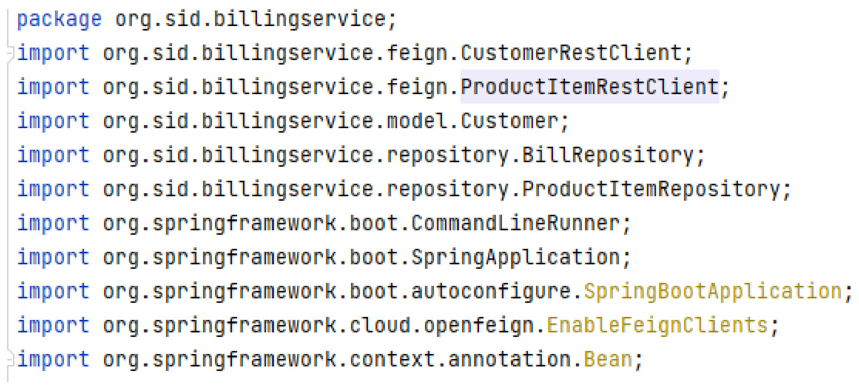
application properties



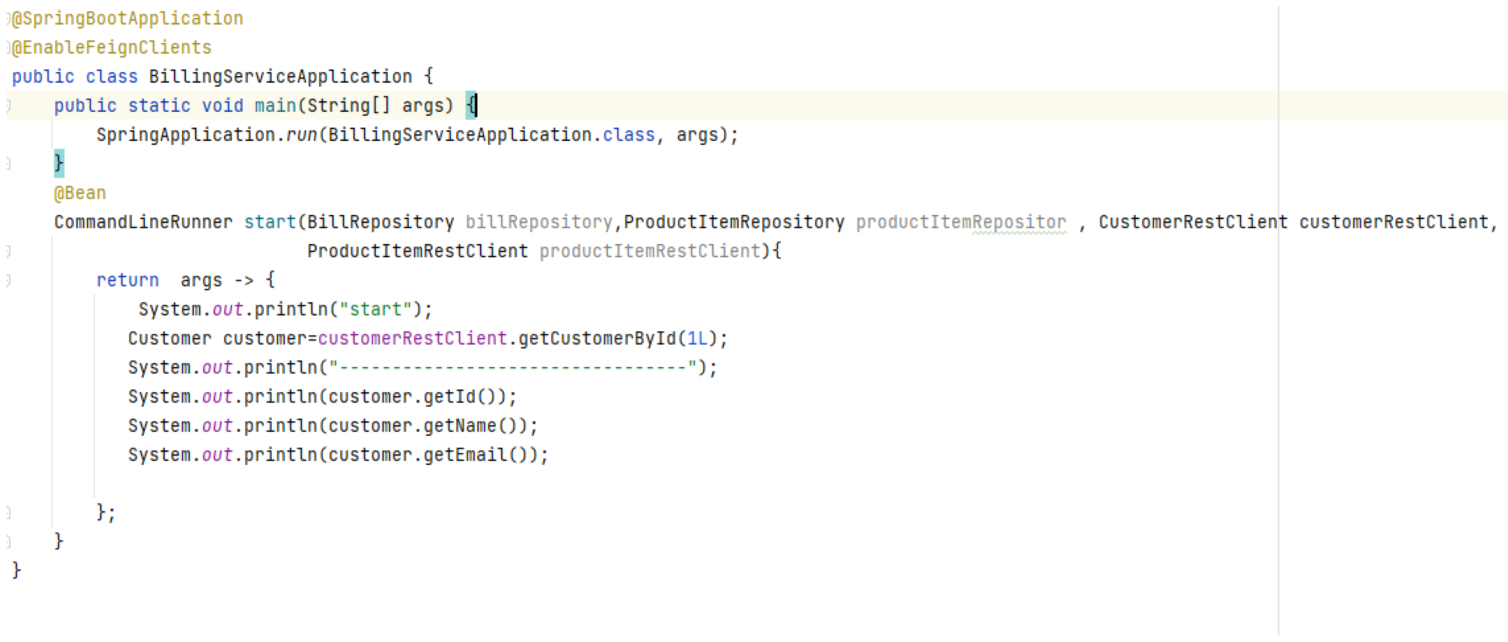
```
server.port=8887
spring.application.name=gateway-service
spring.cloud.discovery.enabled=true
```

5. Micro service Billing Service

Billing Service Application

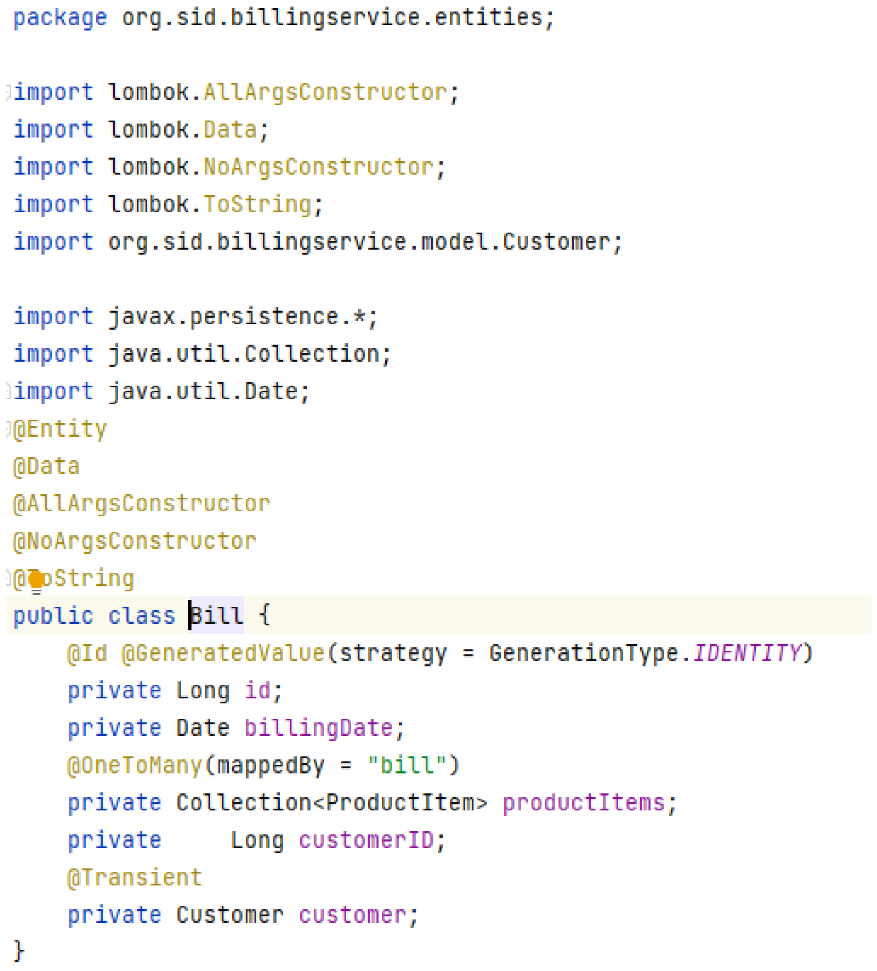


```java
package org.sid.billingservice;
import org.sid.billingservice.feign.CustomerRestClient;
import org.sid.billingservice.feign.ProductItemRestClient;
import org.sid.billingservice.model.Customer;
import org.sid.billingservice.repository.BillRepository;
import org.sid.billingservice.repository.ProductItemRepository;
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;
```



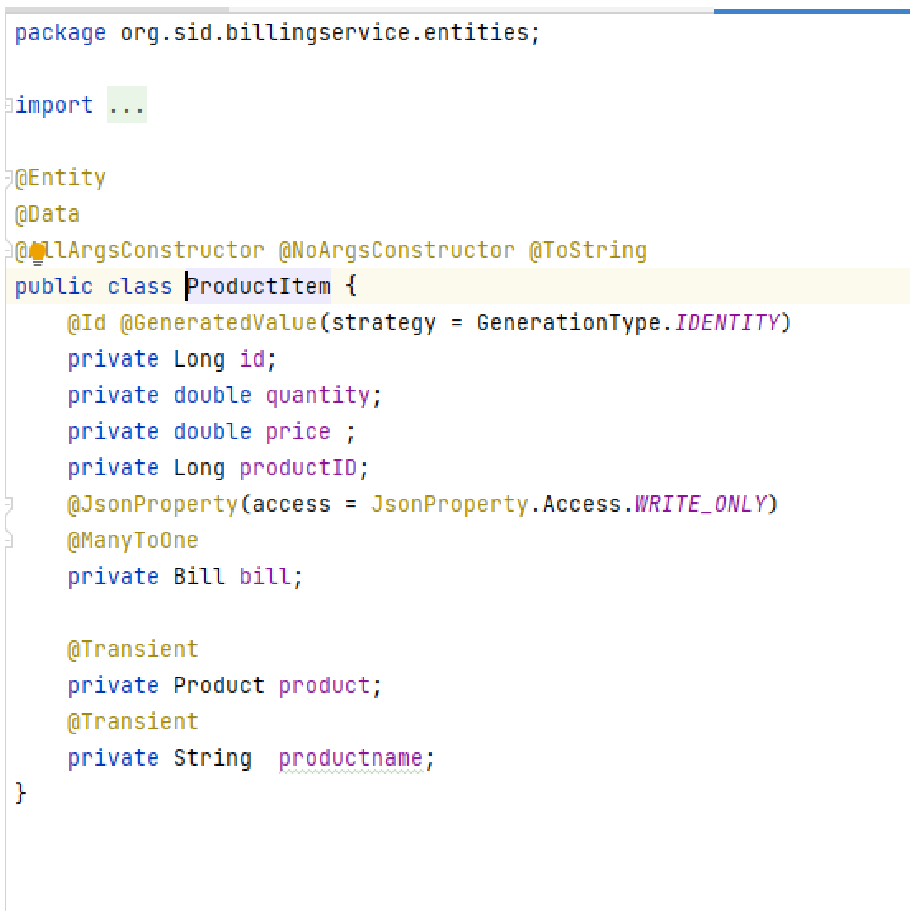
```java
@SpringBootApplication
@EnableFeignClients
public class BillingServiceApplication {
    public static void main(String[] args) {
        SpringApplication.run(BillingServiceApplication.class, args);
    }
    @Bean
    CommandLineRunner start(BillRepository billRepository,ProductItemRepository productItemRepositor , CustomerRestClient customerRestClient,
                            ProductItemRestClient productItemRestClient){
        return  args -> {
            System.out.println("start");
           Customer customer=customerRestClient.getCustomerById(1L);
           System.out.println("--------------------------------");
           System.out.println(customer.getId());
           System.out.println(customer.getName());
           System.out.println(customer.getEmail());

        };
    }
}
```

Bill



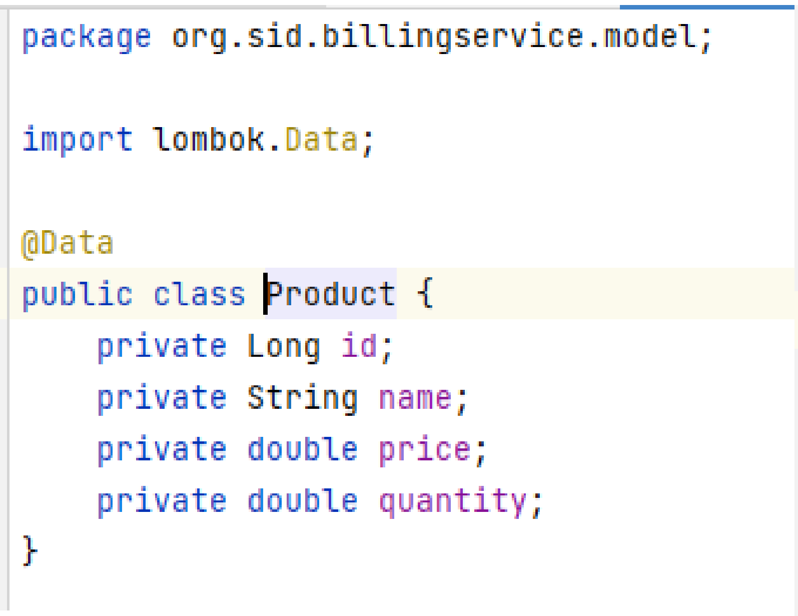
```java
package org.sid.billingservice.entities;

import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;
import lombok.ToString;
import org.sid.billingservice.model.Customer;

import javax.persistence.*;
import java.util.Collection;
import java.util.Date;
@Entity
@Data
@AllArgsConstructor
@NoArgsConstructor
@ToString
public class Bill {
    @Id @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    private Date billingDate;
    @OneToMany(mappedBy = "bill")
    private Collection<ProductItem> productItems;
    private     Long customerID;
    @Transient
    private Customer customer;
}
```

Prodectlems

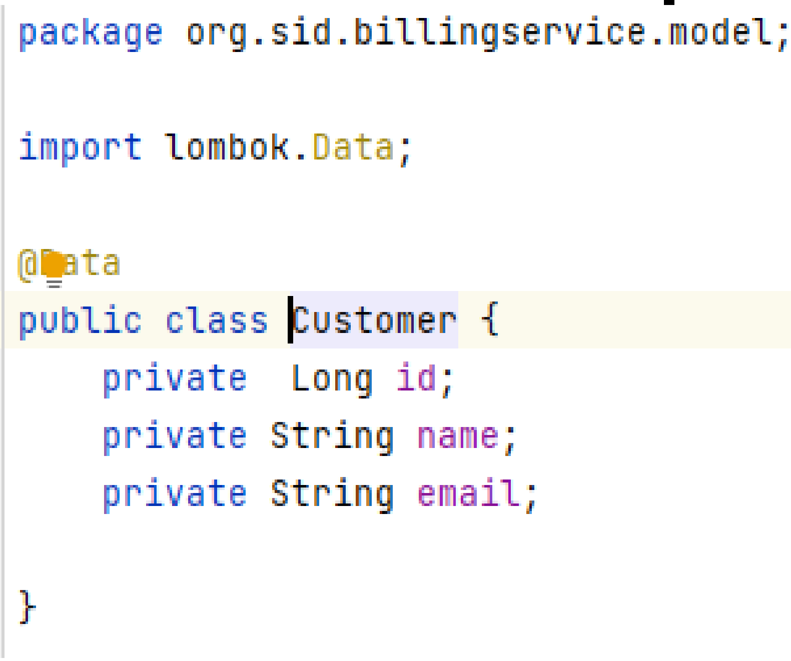


```java
package org.sid.billingservice.entities;

import ...

@Entity
@Data
@AllArgsConstructor @NoArgsConstructor @ToString
public class ProductItem {
    @Id @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    private double quantity;
    private double price ;
    private Long productID;
    @JsonProperty(access = JsonProperty.Access.WRITE_ONLY)
    @ManyToOne
    private Bill bill;

    @Transient
    private Product product;
    @Transient
    private String  productname;
}
```

Models

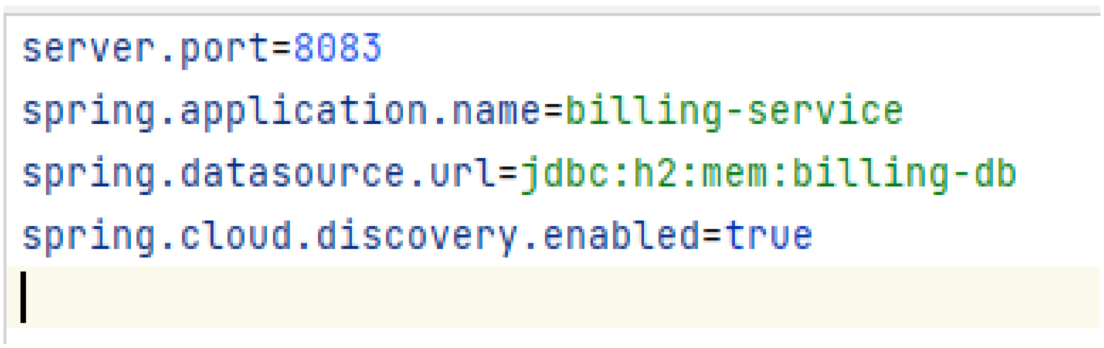


```java
package org.sid.billingservice.model;

import lombok.Data;

@Data
public class Product {
    private Long id;
    private String name;
    private double price;
    private double quantity;
}
```



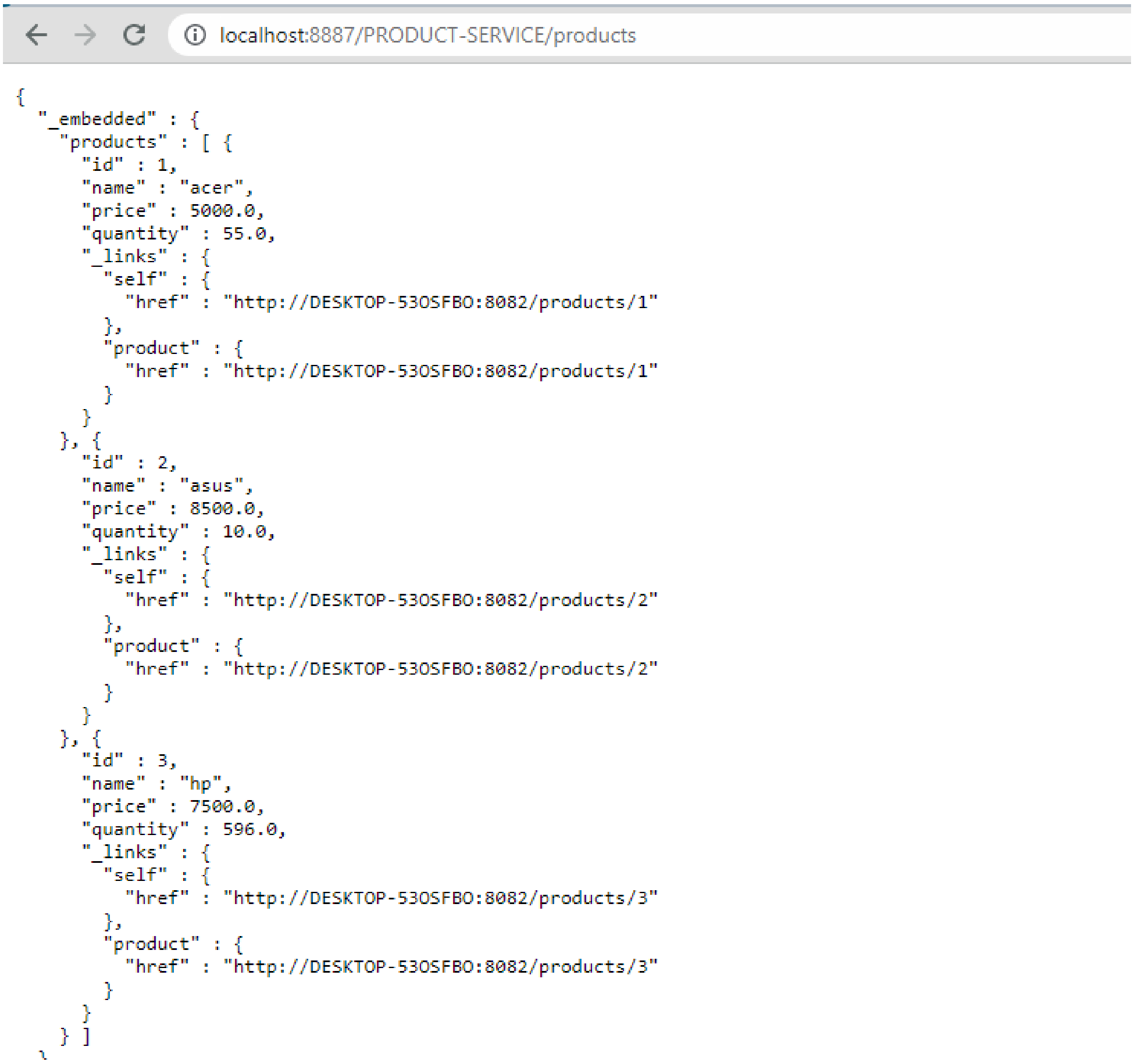
```java
package org.sid.billingservice.model;

import lombok.Data;

@Data
public class Customer {
    private  Long id;
    private String name;
    private String email;

}
```

application properties



```
server.port=8083
spring.application.name=billing-service
spring.datasource.url=jdbc:h2:mem:billing-db
spring.cloud.discovery.enabled=true
```

6. DEMO



localhost:8887/PRODUCT-SERVICE/products

```
{
  "_embedded" : {
    "products" : [ {
      "id" : 1,
      "name" : "acer",
      "price" : 5000.0,
      "quantity" : 55.0,
      "_links" : {
        "self" : {
          "href" : "http://DESKTOP-53OSFBO:8082/products/1"
        },
        "product" : {
          "href" : "http://DESKTOP-53OSFBO:8082/products/1"
        }
      }
    }, {
      "id" : 2,
      "name" : "asus",
      "price" : 8500.0,
      "quantity" : 10.0,
      "_links" : {
        "self" : {
          "href" : "http://DESKTOP-53OSFBO:8082/products/2"
        },
        "product" : {
          "href" : "http://DESKTOP-53OSFBO:8082/products/2"
        }
      }
    }, {
      "id" : 3,
      "name" : "hp",
      "price" : 7500.0,
      "quantity" : 596.0,
      "_links" : {
        "self" : {
          "href" : "http://DESKTOP-53OSFBO:8082/products/3"
        },
        "product" : {
          "href" : "http://DESKTOP-53OSFBO:8082/products/3"
        }
      }
    } ]
```

IV. CONCLUSION

L’ architecture des Micro-services s’est révélée très enrichissante dans la mesure où ce projet nous a permis d’appliquer nos connaissances acquises en JEE et spring .
Je voudrais adresser mes remerciements à mon professeur **M. YOUSSFI Mohamed**, pour sa confiance, sa disponibilité et sa ponctualité et son dynamisme.