ECOLE NORMALE SUPÉRIEURE DE
L'ENSEIGNEMENT TECHNIQUE DE MOHAMMEDIA

UNIVERSITÉ HASSAN II DE CASABLANCA

ENSET

المدرسة العليا لأساتذة التعليم التقني
المحمدية
جامعة الحسن الثاني بالدار البيضاء

DEPARTEMENT MATHEMATIQUES ET INFORMATIQUE

# Activité Pratique N°7

## Filière :
## « Génie du Logiciel et des Systèmes Informatiques Distribués »
## GLSID

# Mise en œuvre d'une architecture micro-services avec Spring cloud

Réalisé par :

Najat ES-SAYYAD

**Année Universitaire : 2022-2023**

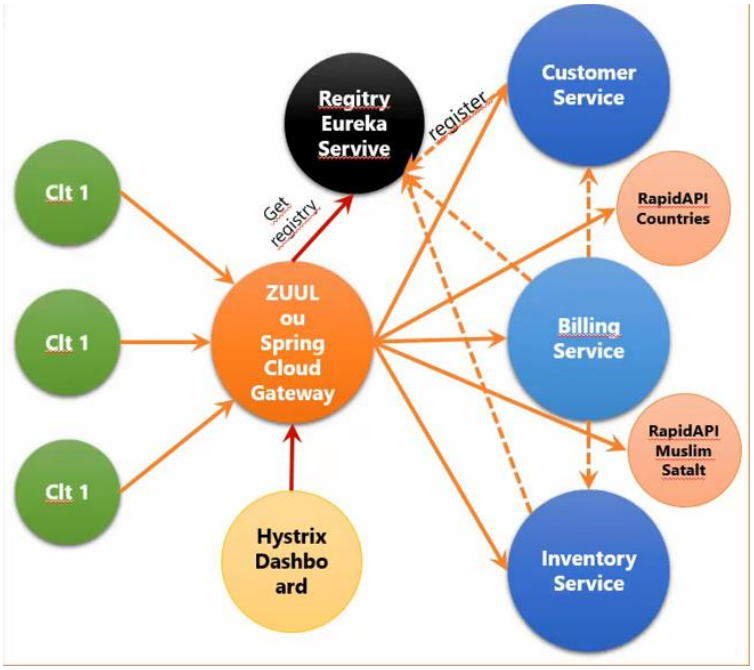
# Introduction

Au cours de cette activité pratique, je vais explorer les étapes clés pour mettre en place une architecture de micro-services avec Spring Cloud. Je vais apprendre à découper une application monolithique en services autonomes, où chaque service est responsable d'une fonctionnalité spécifique. Je vais également découvrir comment utiliser les fonctionnalités avancées de Spring Cloud pour gérer la communication entre les services, la découverte des services, la configuration distribuée, la tolérance aux pannes, et bien plus encore. Enfin, je vais avoir l'opportunité de mettre en pratique ces concepts en développant un projet concret basé sur une architecture de micro-services.

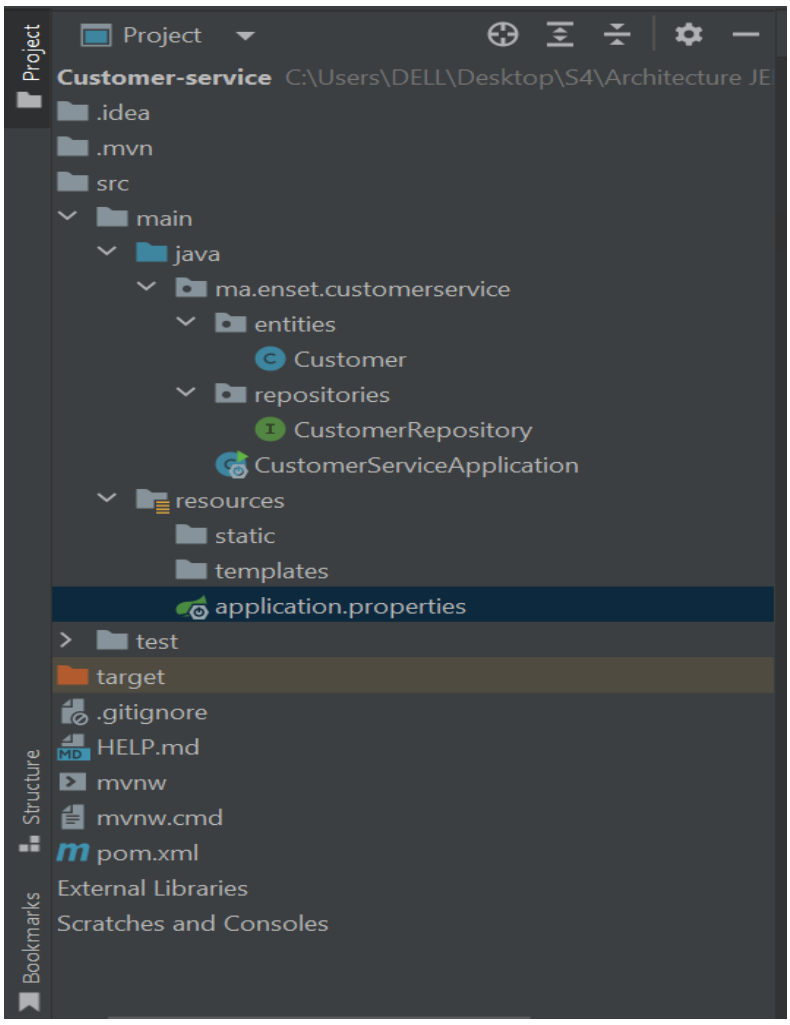
Cette activité pratique me permettra d'acquérir une compréhension approfondie des concepts fondamentaux des architectures de micro-services et de maîtriser l'utilisation de Spring Cloud pour les mettre en œuvre. Je développerai des compétences précieuses qui me permettront de concevoir et de construire des applications évolutives, résilientes et flexibles à l'avenir.

Je suis impatient de me plonger dans cette activité pratique et d'explorer les possibilités passionnantes offertes par les architectures de micro-services avec Spring Cloud. Ce devoir sera une expérience précieuse qui approfondira ma connaissance de l'architecture logicielle moderne et me préparera à relever les défis du développement d'applications distribuées.

Architecture du projet :




Structure du projet :



Créer l'entité JPA Customer

L'entité JPA Customer ayant les attributs :

- id de type Long
- name de type String
- email de type String

```java
package ma.enset.customerservice.entities;

import jakarta.persistence.Entity;
import jakarta.persistence.GeneratedValue;
import jakarta.persistence.GenerationType;
import jakarta.persistence.Id;
import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;
import lombok.ToString;

6 usages
@Entity
@Data @NoArgsConstructor @AllArgsConstructor @ToString
public class Customer {
    no usages
    @Id @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    no usages
    private  String name;
    no usages
    private String email ;
}
```

Créer l'interface CustomerRepository basée sur Spring Data

```java
package ma.enset.customerservice.repositories;

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


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

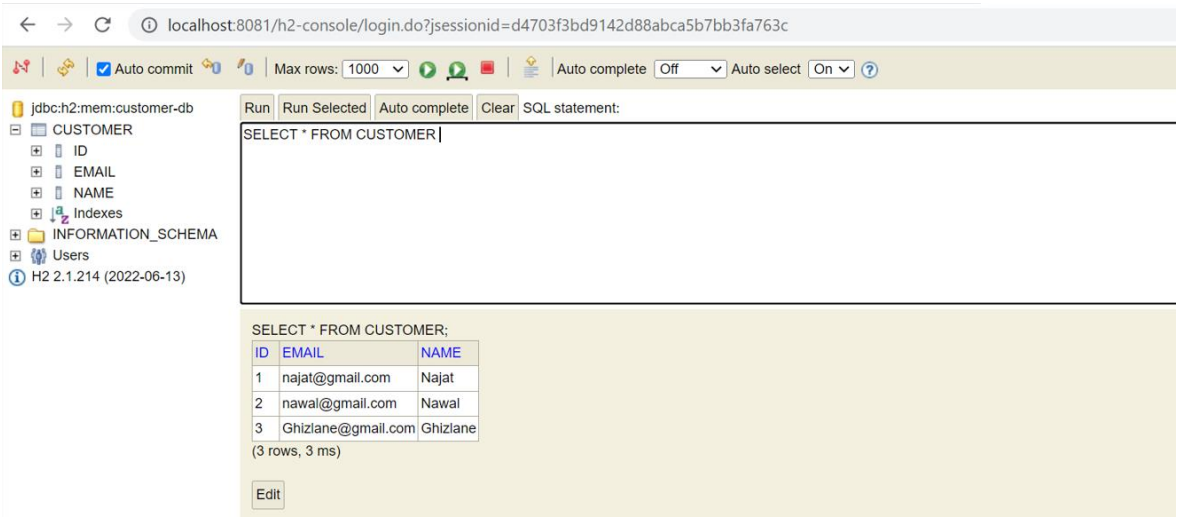
Configurer l'unité de persistance dans le ficher application.properties

```
server.port=8081
spring.application.name=customer-service
spring.datasource.url=jdbc:h2:mem:customer-db
spring.h2.console.enabled=true
#management.endpoints.web.exposure.include=*
```

Tester la couche DAO

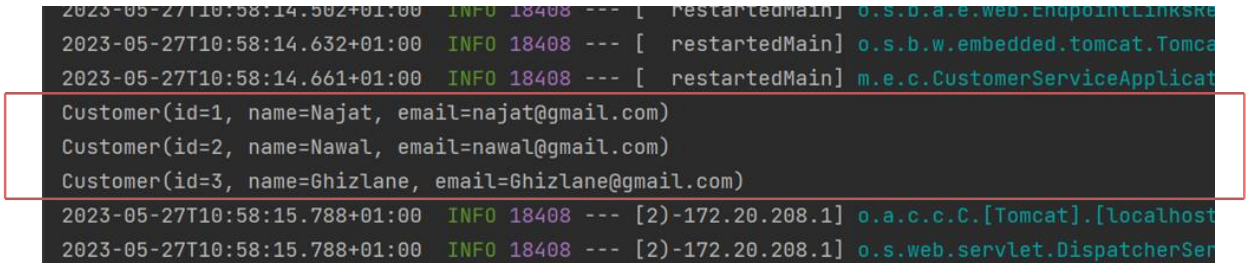
```
package ma.enset.customerservice;

import ...

1 usage
@SpringBootApplication
public class CustomerServiceApplication {
    no usages
    public static void main(String[] args) {
        SpringApplication.run(CustomerServiceApplication.class, args);
    }

    no usages
    @Bean
    CommandLineRunner start(CustomerRepository customerRepository){
        return args -> {
            customerRepository.save(new Customer( id: null, name: "Najat", email: "najat@gmail.com"));
            customerRepository.save(new Customer( id: null, name: "Nawal", email: "nawal@gmail.com"));
            customerRepository.save(new Customer( id: null, name: "Ghizlane", email: "Ghizlane@gmail.com"));
            customerRepository.findAll().forEach(customer -> {
                System.out.println(customer.toString());
            });
        };
    }
}
```

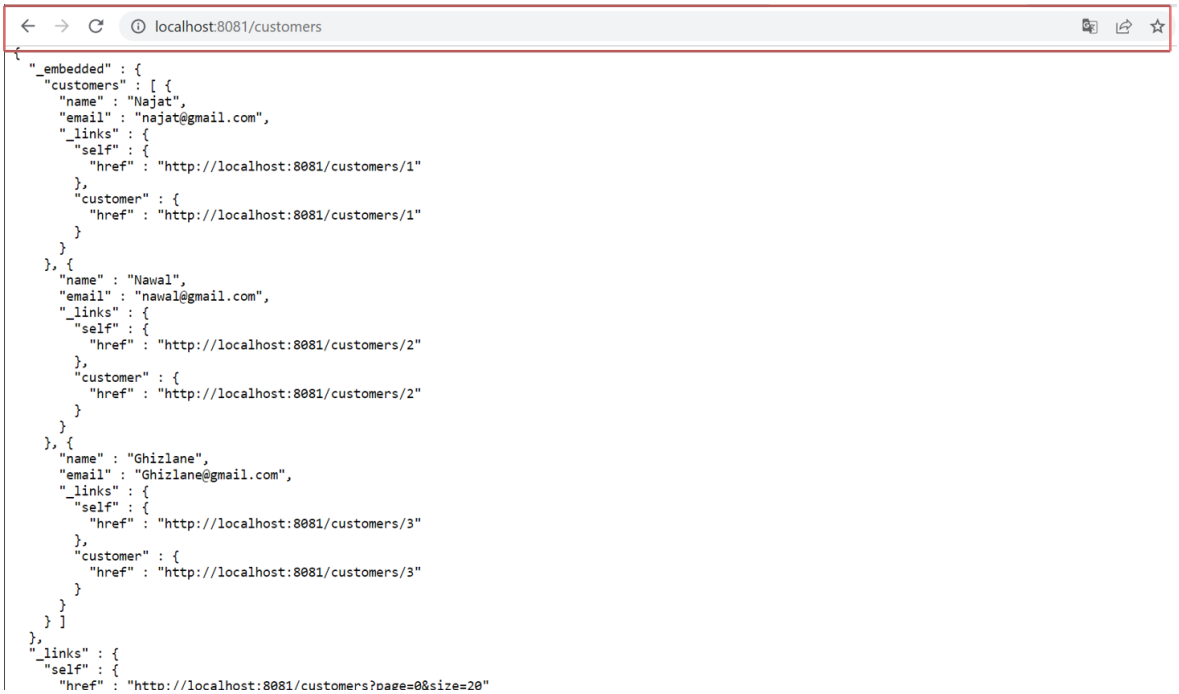
La base de données sous H2 :



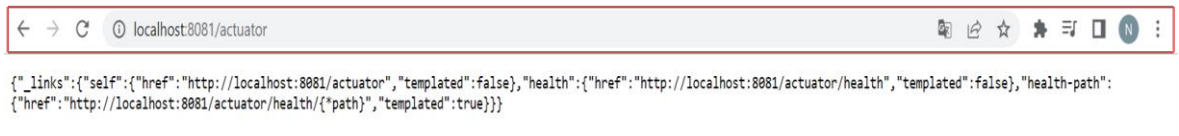
Résultat :

Terminal :



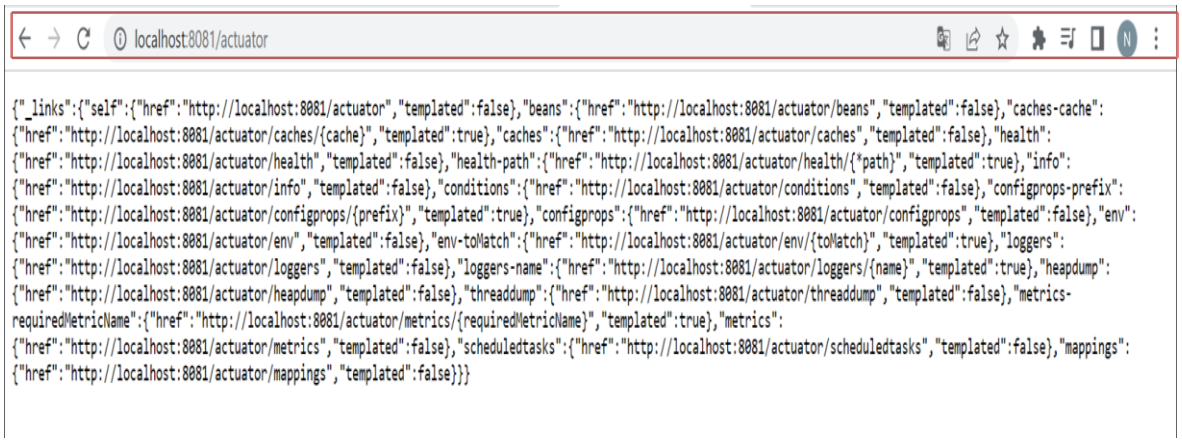







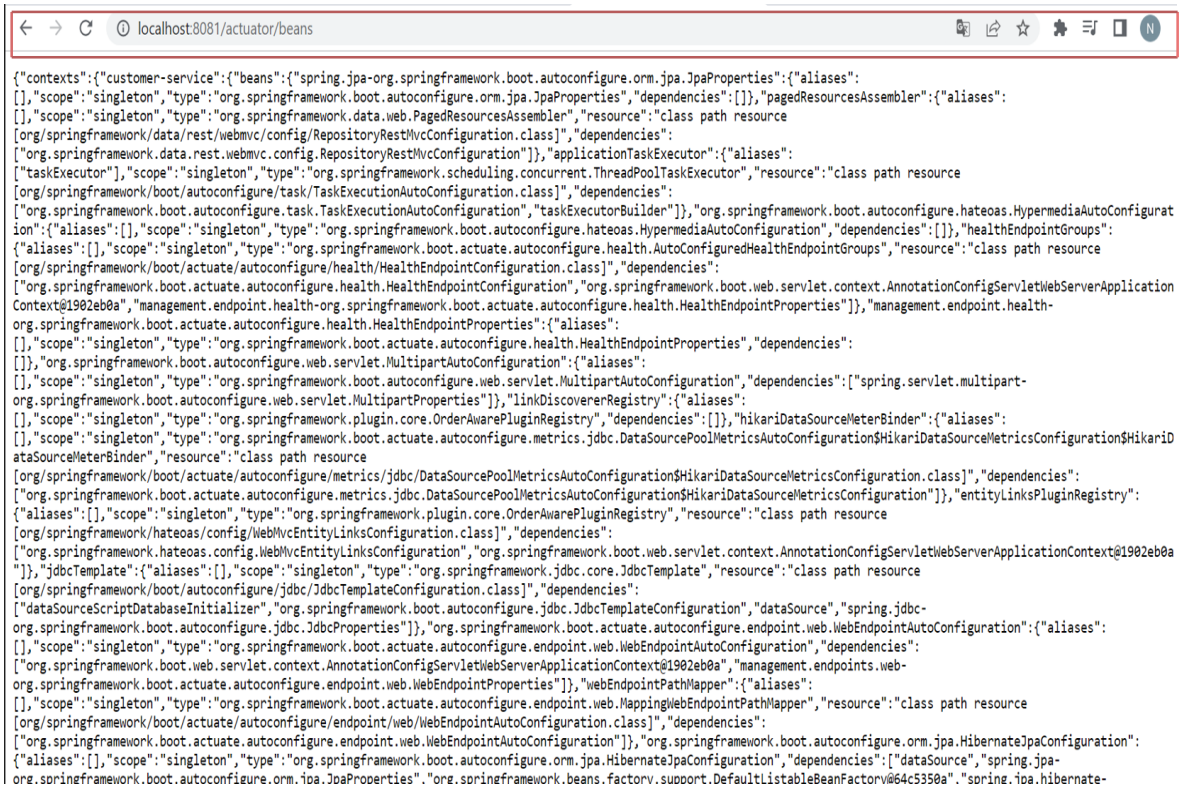

Configurer l'unité de persistance dans le ficher application.properties

```
server.port=8081
spring.application.name=customer-service
spring.datasource.url=jdbc:h2:mem:customer-db
spring.h2.console.enabled=true
management.endpoints.web.exposure.include=*
```





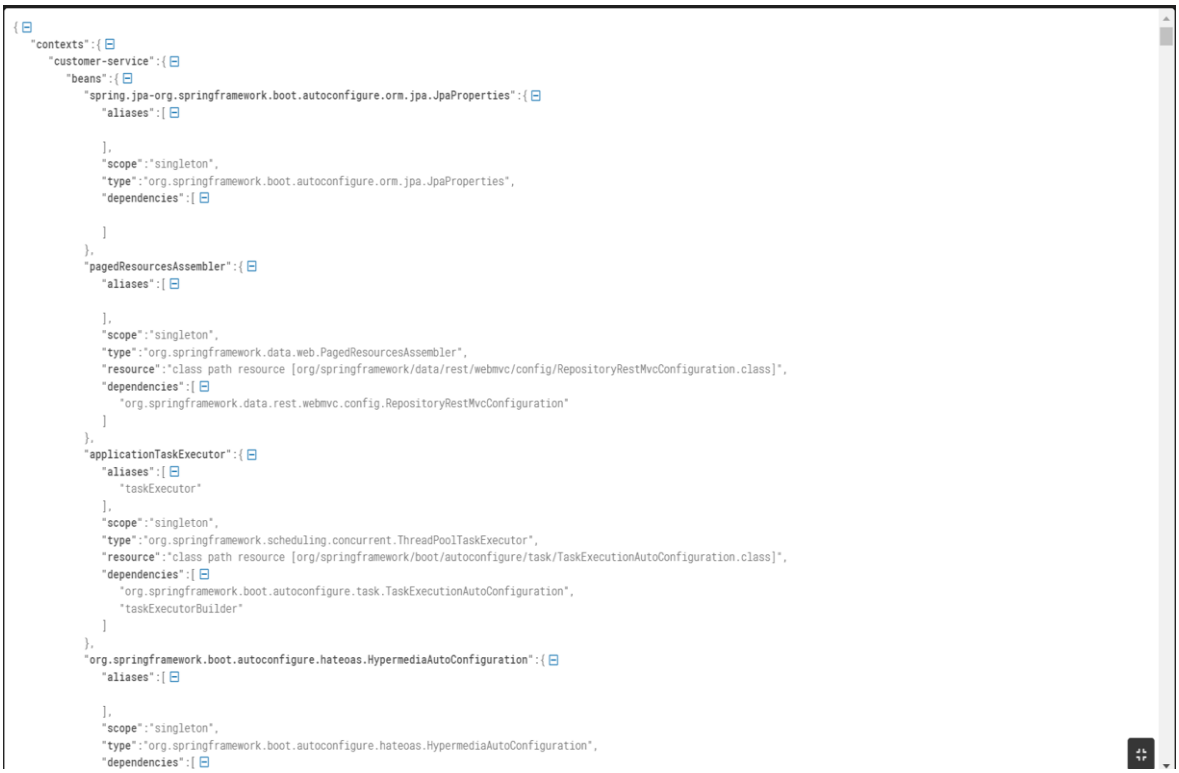
```
{"_links":{"self":{"href":"http://localhost:8081/actuator","templated":false},"beans":{"href":"http://localhost:8081/actuator/beans","templated":false},"caches-cache":
{"href":"http://localhost:8081/actuator/caches/{cache}","templated":true},"caches":{"href":"http://localhost:8081/actuator/caches","templated":false},"health":
{"href":"http://localhost:8081/actuator/health","templated":false},"health-path":{"href":"http://localhost:8081/actuator/health/{*path}","templated":true},"info":
{"href":"http://localhost:8081/actuator/info","templated":false},"conditions":{"href":"http://localhost:8081/actuator/conditions","templated":false},"configprops-prefix":
{"href":"http://localhost:8081/actuator/configprops/{prefix}","templated":true},"configprops":{"href":"http://localhost:8081/actuator/configprops","templated":false},"env":
{"href":"http://localhost:8081/actuator/env","templated":false},"env-toMatch":{"href":"http://localhost:8081/actuator/env/{toMatch}","templated":true},"loggers":
{"href":"http://localhost:8081/actuator/loggers","templated":false},"loggers-name":{"href":"http://localhost:8081/actuator/loggers/{name}","templated":true},"heapdump":
{"href":"http://localhost:8081/actuator/heapdump","templated":false},"threaddump":{"href":"http://localhost:8081/actuator/threaddump","templated":false},"metrics-
requiredMetricName":{"href":"http://localhost:8081/actuator/metrics/{requiredMetricName}","templated":true},"metrics":
{"href":"http://localhost:8081/actuator/metrics","templated":false},"scheduledtasks":{"href":"http://localhost:8081/actuator/scheduledtasks","templated":false},"mappings":
{"href":"http://localhost:8081/actuator/mappings","templated":false}}}
```



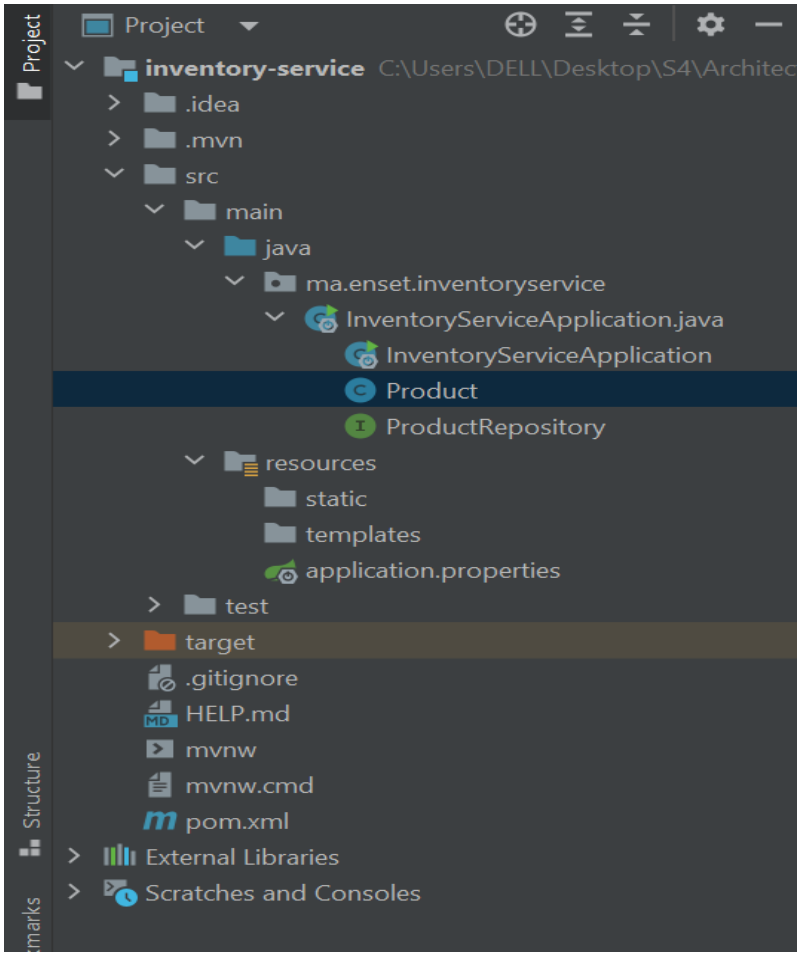
```
{"contexts":{"customer-service":{"beans":{"spring.jpa-org.springframework.boot.autoconfigure.orm.jpa.JpaProperties":{"aliases":
[],"scope":"singleton","type":"org.springframework.boot.autoconfigure.orm.jpa.JpaProperties","dependencies":[]},"pagedResourcesAssembler":{"aliases":
[],"scope":"singleton","type":"org.springframework.data.web.PagedResourcesAssembler","resource":"class path resource
[org/springframework/data/rest/webmvc/config/RepositoryRestMvcConfiguration.class]","dependencies":
["org.springframework.data.rest.webmvc.config.RepositoryRestMvcConfiguration"]},"applicationTaskExecutor":{"aliases":
["taskExecutor"],"scope":"singleton","type":"org.springframework.scheduling.concurrent.ThreadPoolTaskExecutor","resource":"class path resource
[org/springframework/boot/autoconfigure/task/TaskExecutionAutoConfiguration.class]","dependencies":
["org.springframework.boot.autoconfigure.task.TaskExecutionAutoConfiguration","taskExecutorBuilder"]},"org.springframework.boot.autoconfigure.hateoas.HypermediaAutoConfigurat
ion":{"aliases":[],"scope":"singleton","type":"org.springframework.boot.autoconfigure.hateoas.HypermediaAutoConfiguration","dependencies":[]},"healthEndpointGroups":
{"aliases":[],"scope":"singleton","type":"org.springframework.boot.actuate.autoconfigure.health.AutoConfiguredHealthEndpointGroups","resource":"class path resource
[org/springframework/boot/actuate/autoconfigure/health/HealthEndpointConfiguration.class]","dependencies":
["org.springframework.boot.actuate.autoconfigure.health.HealthEndpointConfiguration","org.springframework.boot.web.servlet.context.AnnotationConfigServletWebServerApplication
Context@1902eb0a","management.endpoint.health-org.springframework.boot.actuate.autoconfigure.health.HealthEndpointProperties"]},"management.endpoint.health-
org.springframework.boot.actuate.autoconfigure.health.HealthEndpointProperties":{"aliases":
[],"scope":"singleton","type":"org.springframework.boot.actuate.autoconfigure.health.HealthEndpointProperties","dependencies":
[]},"org.springframework.boot.autoconfigure.web.servlet.MultipartAutoConfiguration":{"aliases":
[],"scope":"singleton","type":"org.springframework.boot.autoconfigure.web.servlet.MultipartAutoConfiguration","dependencies":["spring.servlet.multipart-
org.springframework.boot.autoconfigure.web.servlet.MultipartProperties"]},"linkDiscovererRegistry":{"aliases":
[],"scope":"singleton","type":"org.springframework.plugin.core.OrderAwarePluginRegistry","dependencies":[]},"hikariDataSourceMeterBinder":{"aliases":
[],"scope":"singleton","type":"org.springframework.boot.actuate.autoconfigure.metrics.jdbc.DataSourcePoolMetricsAutoConfiguration$HikariDataSourceMetricsConfiguration$HikariD
ataSourceMeterBinder","resource":"class path resource
[org/springframework/boot/actuate/autoconfigure/metrics/jdbc/DataSourcePoolMetricsAutoConfiguration$HikariDataSourceMetricsConfiguration.class]","dependencies":
["org.springframework.boot.actuate.autoconfigure.metrics.jdbc.DataSourcePoolMetricsAutoConfiguration$HikariDataSourceMetricsConfiguration"]},"entityLinksPluginRegistry":
{"aliases":[],"scope":"singleton","type":"org.springframework.plugin.core.OrderAwarePluginRegistry","resource":"class path resource
[org/springframework/hateoas/config/WebMvcEntityLinksConfiguration.class]","dependencies":
["org.springframework.hateoas.config.WebMvcEntityLinksConfiguration","org.springframework.boot.web.servlet.context.AnnotationConfigServletWebServerApplicationContext@1902eb0a
"]},"jdbcTemplate":{"aliases":[],"scope":"singleton","type":"org.springframework.jdbc.core.JdbcTemplate","resource":"class path resource
[org/springframework/boot/autoconfigure/jdbc/JdbcTemplateConfiguration.class]","dependencies":
["dataSourceScriptDatabaseInitializer","org.springframework.boot.autoconfigure.jdbc.JdbcTemplateConfiguration","dataSource","spring.jdbc-
org.springframework.boot.autoconfigure.jdbc.JdbcProperties"]},"org.springframework.boot.actuate.autoconfigure.endpoint.web.WebEndpointAutoConfiguration":{"aliases":
[],"scope":"singleton","type":"org.springframework.boot.actuate.autoconfigure.endpoint.web.WebEndpointAutoConfiguration","dependencies":
["org.springframework.boot.web.servlet.context.AnnotationConfigServletWebServerApplicationContext@1902eb0a","management.endpoints.web-
org.springframework.boot.actuate.autoconfigure.endpoint.web.WebEndpointProperties"]},"webEndpointPathMapper":{"aliases":
[],"scope":"singleton","type":"org.springframework.boot.actuate.autoconfigure.endpoint.web.MappingWebEndpointPathMapper","resource":"class path resource
[org/springframework/boot/actuate/autoconfigure/endpoint/web/WebEndpointAutoConfiguration.class]","dependencies":
["org.springframework.boot.actuate.autoconfigure.endpoint.web.WebEndpointAutoConfiguration"]},"org.springframework.boot.autoconfigure.orm.jpa.HibernateJpaConfiguration":
{"aliases":[],"scope":"singleton","type":"org.springframework.boot.autoconfigure.orm.jpa.HibernateJpaConfiguration","dependencies":["dataSource","spring.jpa-
org.springframework.boot.autoconfigure.orm.jpa.JpaProperties","org.springframework.beans.factory.support.DefaultListableBeanFactory@64c5350a","spring.jpa.hibernate-
```


Si j’utilise jsonformatter :



```
{
  "contexts":{
    "customer-service":{
      "beans":{
        "spring.jpa-org.springframework.boot.autoconfigure.orm.jpa.JpaProperties":{
          "aliases":[

          ],
          "scope":"singleton",
          "type":"org.springframework.boot.autoconfigure.orm.jpa.JpaProperties",
          "dependencies":[

          ]
        },
        "pagedResourcesAssembler":{
          "aliases":[

          ],
          "scope":"singleton",
          "type":"org.springframework.data.web.PagedResourcesAssembler",
          "resource":"class path resource [org/springframework/data/rest/webmvc/config/RepositoryRestMvcConfiguration.class]",
          "dependencies":[
            "org.springframework.data.rest.webmvc.config.RepositoryRestMvcConfiguration"
          ]
        },
        "applicationTaskExecutor":{
          "aliases":[
            "taskExecutor"
          ],
          "scope":"singleton",
          "type":"org.springframework.scheduling.concurrent.ThreadPoolTaskExecutor",
          "resource":"class path resource [org/springframework/boot/autoconfigure/task/TaskExecutionAutoConfiguration.class]",
          "dependencies":[
            "org.springframework.boot.autoconfigure.task.TaskExecutionAutoConfiguration",
            "taskExecutorBuilder"
          ]
        },
        "org.springframework.boot.autoconfigure.hateoas.HypermediaAutoConfiguration":{
          "aliases":[

          ],
          "scope":"singleton",
          "type":"org.springframework.boot.autoconfigure.hateoas.HypermediaAutoConfiguration",
          "dependencies":[
```

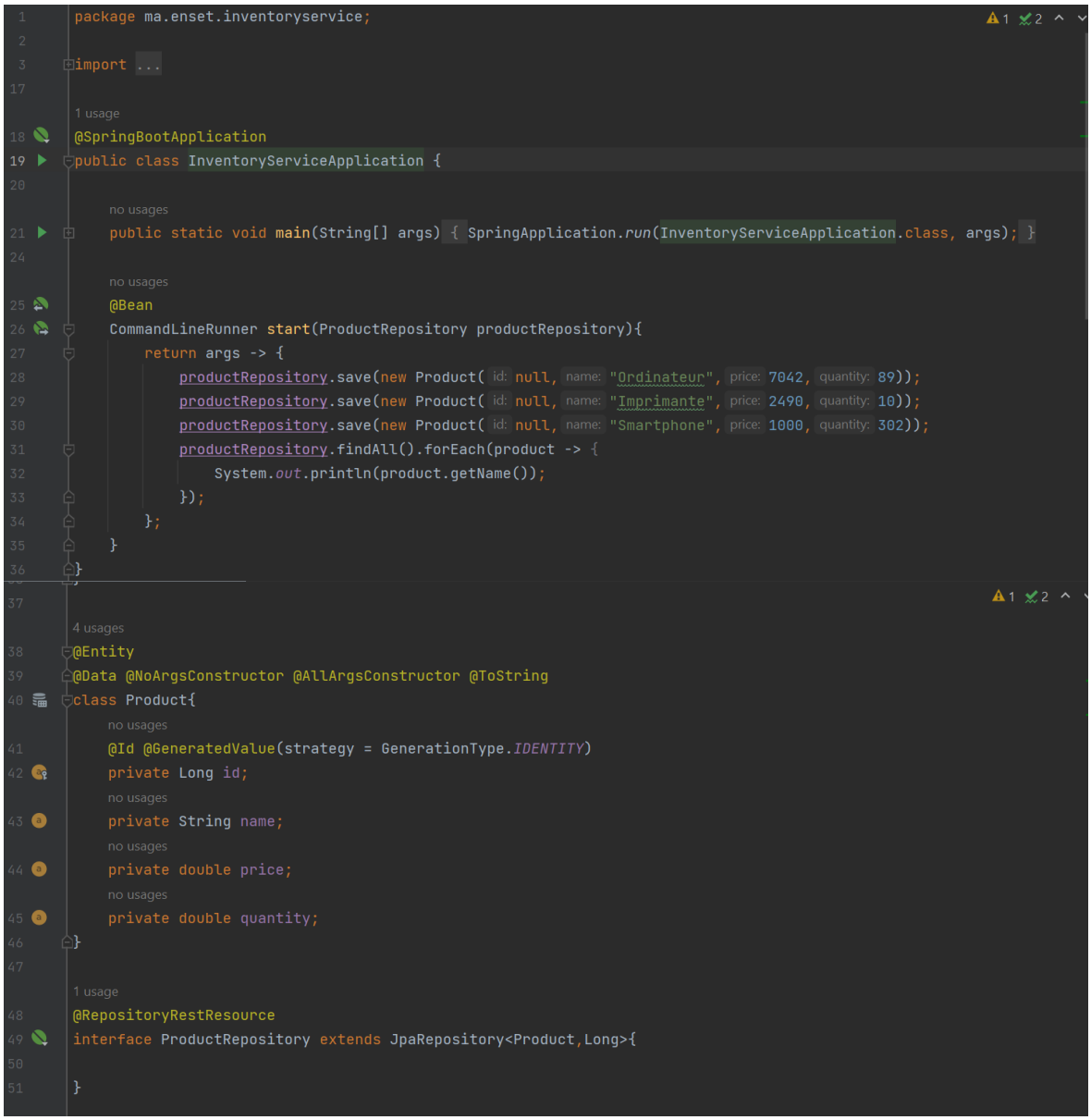
Structure du projet :




Créer l'entité JPA Product et Créer l'interface ProductRepository basée sur Spring Data et Tester dans une seule classe InventoryServiceApplication :

L'entité JPA Product ayant les attributs :

- id de type Long
- name de type String
- price de type double
- quantity de type double



```java
package ma.enset.inventoryservice;

import ...

1 usage
@SpringBootApplication
public class InventoryServiceApplication {

    no usages
    public static void main(String[] args) { SpringApplication.run(InventoryServiceApplication.class, args); }

    no usages
    @Bean
    CommandLineRunner start(ProductRepository productRepository){
        return args -> {
            productRepository.save(new Product( id: null, name: "Ordinateur", price: 7042, quantity: 89));
            productRepository.save(new Product( id: null, name: "Imprimante", price: 2490, quantity: 10));
            productRepository.save(new Product( id: null, name: "Smartphone", price: 1000, quantity: 302));
            productRepository.findAll().forEach(product -> {
                System.out.println(product.getName());
            });
        };
    }
}
```

```java
4 usages
@Entity
@Data @NoArgsConstructor @AllArgsConstructor @ToString
class Product{
    no usages
    @Id @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    no usages
    private String name;
    no usages
    private double price;
    no usages
    private double quantity;
}

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

}
```

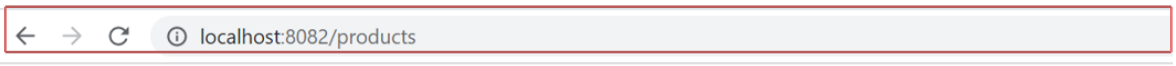
Configurer l'unité de persistance dans le ficher application.properties

```
server.port=8082
spring.application.name=product-service
spring.datasource.url=jdbc:h2:mem:product-db
spring.h2.console.enabled=true
#management.endpoints.web.exposure.include=*
```

Résultat :

Terminal :

```
2023-05-27T11:35:05.573+01:00  INFO 13220 --- [nio-8082-exec-1] o.s.
Ordinateur
Imprimante
Smartphone
```



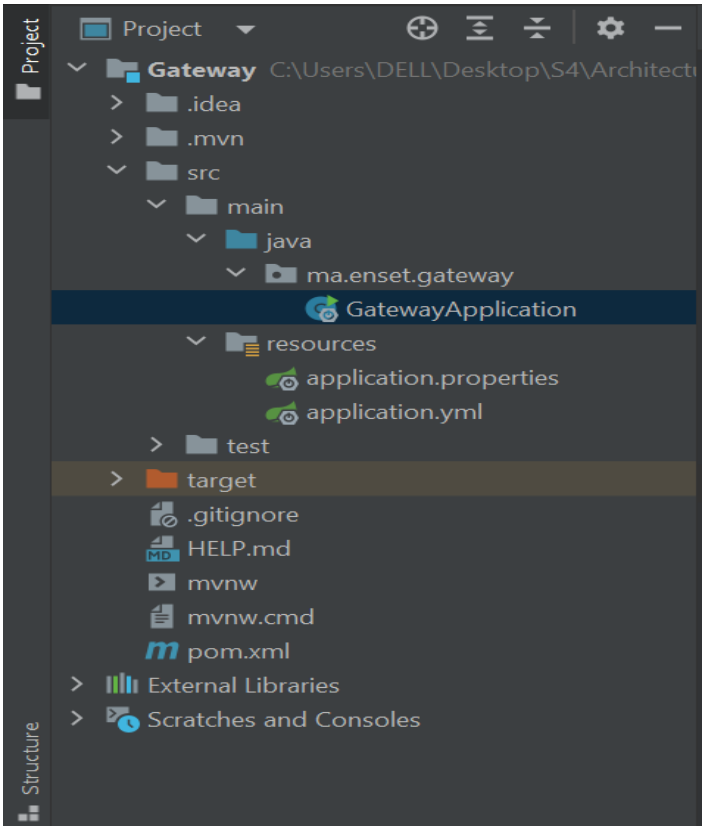


```
{
  "_embedded" : {
    "products" : [ {
      "name" : "Ordinateur",
      "price" : 7042.0,
      "quantity" : 89.0,
      "_links" : {
        "self" : {
          "href" : "http://localhost:8082/products/1"
        },
        "product" : {
          "href" : "http://localhost:8082/products/1"
        }
      }
    }, {
      "name" : "Imprimante",
      "price" : 2490.0,
      "quantity" : 10.0,
      "_links" : {
        "self" : {
          "href" : "http://localhost:8082/products/2"
        },
        "product" : {
          "href" : "http://localhost:8082/products/2"
        }
      }
    }, {
      "name" : "Smartphone",
      "price" : 1000.0,
      "quantity" : 302.0,
      "_links" : {
        "self" : {
          "href" : "http://localhost:8082/products/3"
        },
        "product" : {
          "href" : "http://localhost:8082/products/3"
        }
      }
    } ]
```




```
{
  "name" : "Ordinateur",
  "price" : 7042.0,
  "quantity" : 89.0,
  "_links" : {
    "self" : {
      "href" : "http://localhost:8082/products/1"
    },
    "product" : {
      "href" : "http://localhost:8082/products/1"
    }
  }
}
```

Structure du projet :




Créer la classe GatewayApplication :

```java
package ma.enset.gateway;

import ...

1 usage
@SpringBootApplication
public class GatewayApplication {

    no usages
    public static void main(String[] args) {
        SpringApplication.run(GatewayApplication.class, args);
    }
}
```

Configurer l'unité de persistance dans le fichier application.properties

```
server.port=8888
spring.application.name=gateway-service
spring.h2.console.enabled=true

```

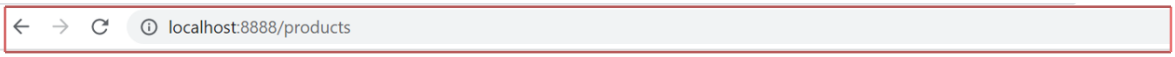
Créer et Configurer l'unité de persistance dans le fichier application.yml

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

Ou sur la clasee GatewayApplication :

```java
/*@Bean
RouteLocator routeLocator(RouteLocatorBuilder builder){
    return builder.routes()
            .route((r)->r.path("/customers/**").uri("http://localhost:8081/").id("r1"))
            .route((r)->r.path("/products/**").uri("http://localhost:8082/").id("r2"))
            .build();
    return
}*/
```

Résultat :

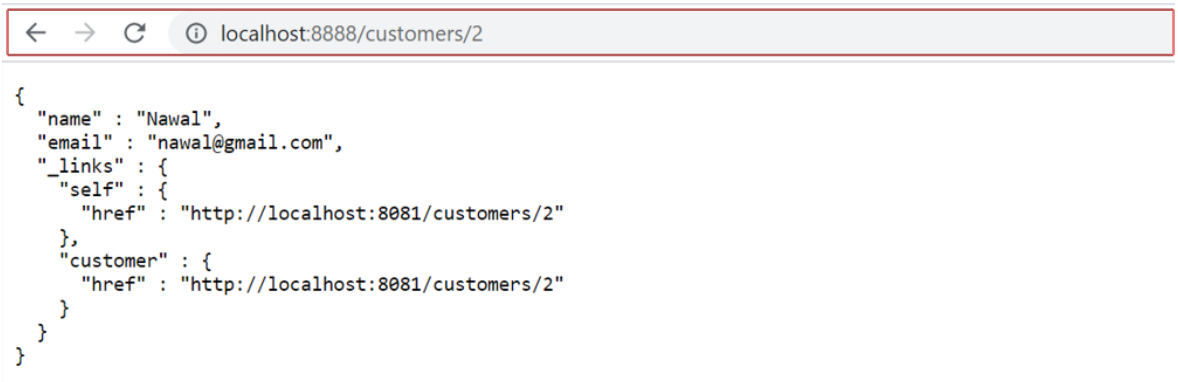




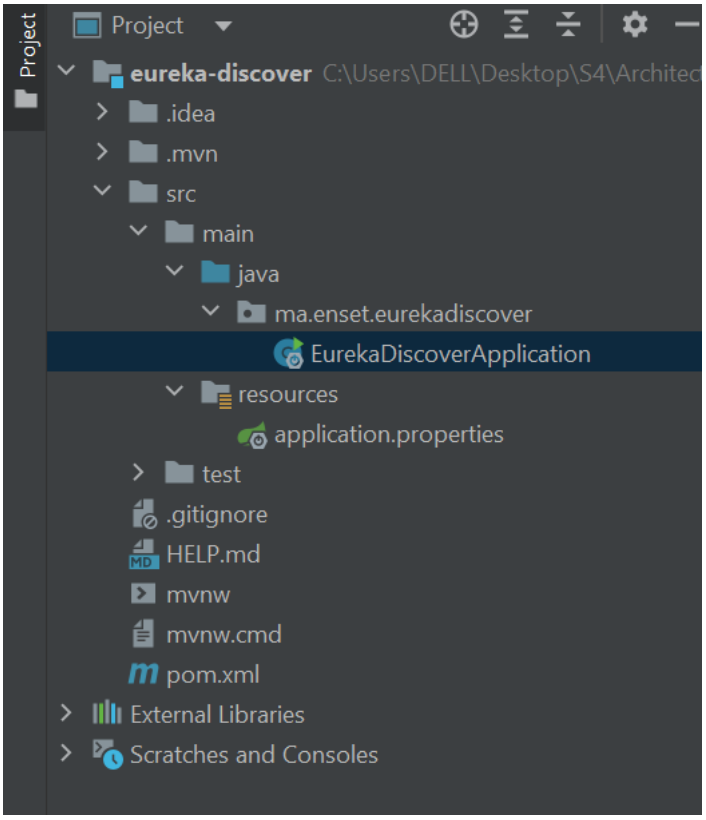
```
{
  "_embedded" : {
    "products" : [ {
      "name" : "Ordinateur",
      "price" : 7042.0,
      "quantity" : 89.0,
      "_links" : {
        "self" : {
          "href" : "http://localhost:8082/products/1"
        },
        "product" : {
          "href" : "http://localhost:8082/products/1"
        }
      }
    }, {
      "name" : "Imprimante",
      "price" : 2490.0,
      "quantity" : 10.0,
      "_links" : {
        "self" : {
          "href" : "http://localhost:8082/products/2"
        },
        "product" : {
          "href" : "http://localhost:8082/products/2"
        }
      }
    }, {
      "name" : "Smartphone",
      "price" : 1000.0,
      "quantity" : 302.0,
      "_links" : {
        "self" : {
          "href" : "http://localhost:8082/products/3"
        },
        "product" : {
          "href" : "http://localhost:8082/products/3"
        }
      }
    } ]
```




```
{
  "_embedded" : {
    "customers" : [ {
      "name" : "Najat",
      "email" : "najat@gmail.com",
      "_links" : {
        "self" : {
          "href" : "http://localhost:8081/customers/1"
        },
        "customer" : {
          "href" : "http://localhost:8081/customers/1"
        }
      }
    }, {
      "name" : "Nawal",
      "email" : "nawal@gmail.com",
      "_links" : {
        "self" : {
          "href" : "http://localhost:8081/customers/2"
        },
        "customer" : {
          "href" : "http://localhost:8081/customers/2"
        }
      }
    }, {
      "name" : "Ghizlane",
      "email" : "Ghizlane@gmail.com",
      "_links" : {
        "self" : {
          "href" : "http://localhost:8081/customers/3"
        },
        "customer" : {
          "href" : "http://localhost:8081/customers/3"
        }
      }
    } ]
  },
  "_links" : {
    "self" : {
```




Structure du projet :

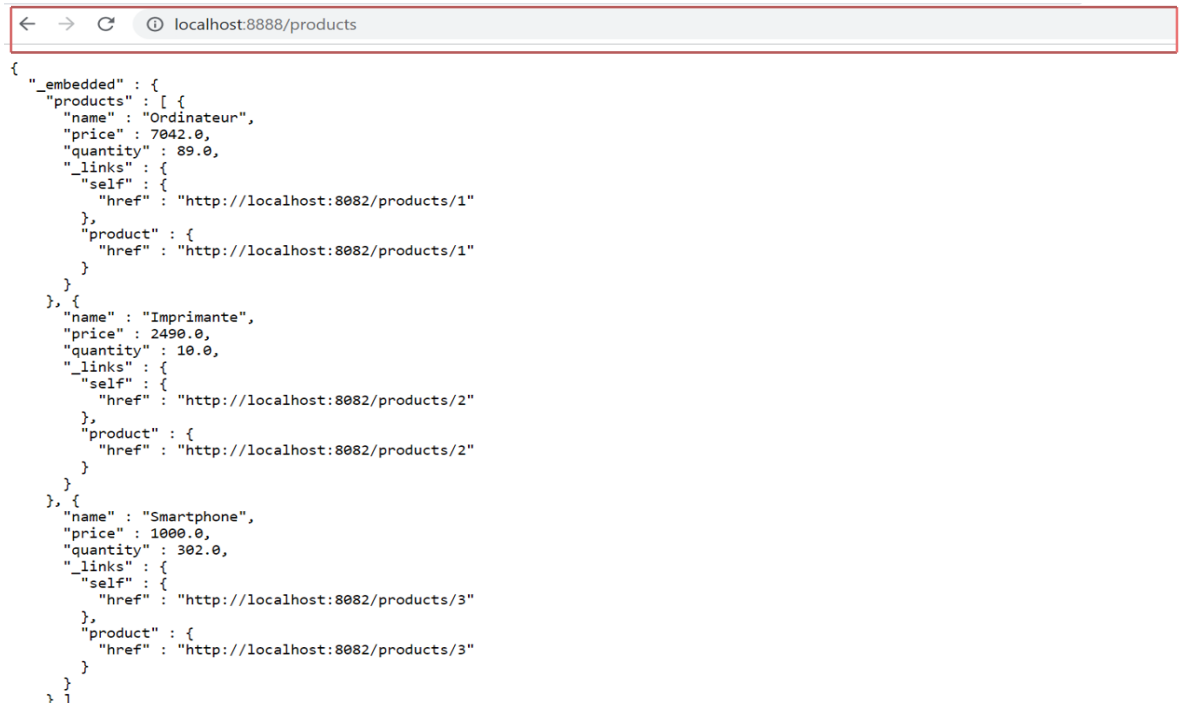


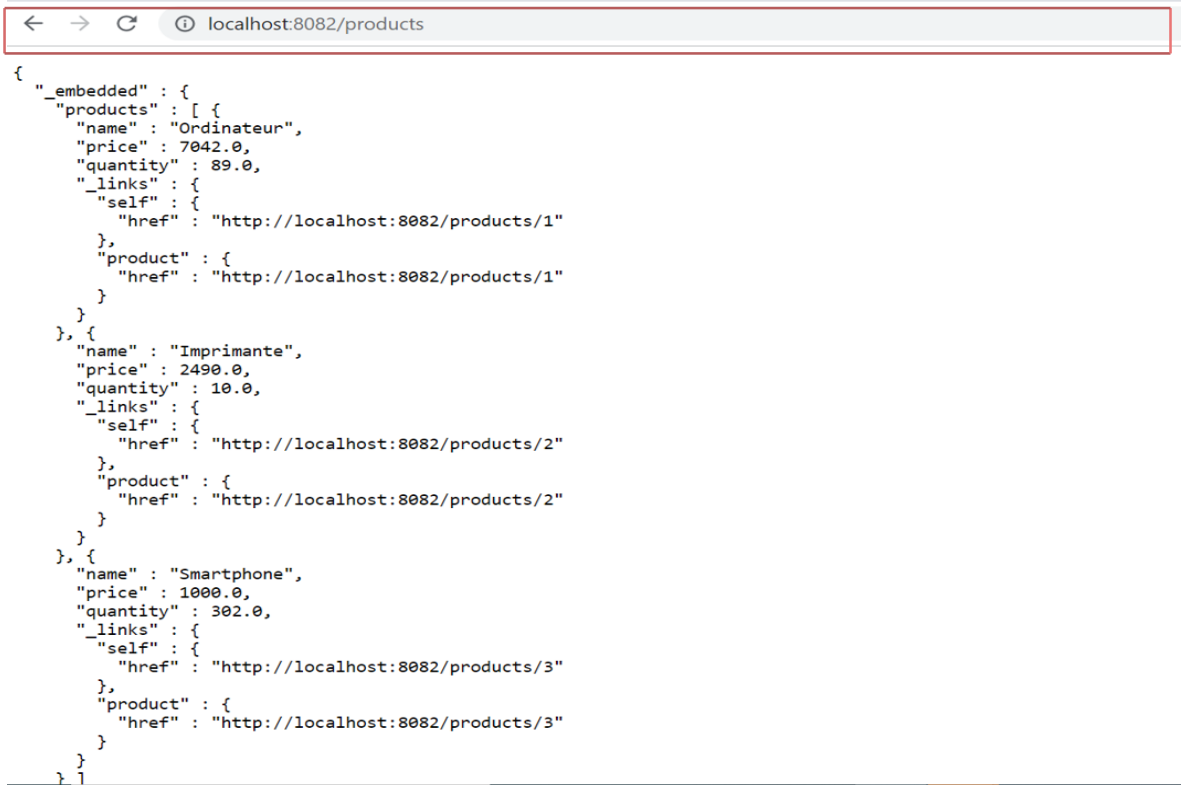

Créer la classe EurekaDiscoverApplication :

```
package ma.enset.eurekadiscover;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
@EnableEurekaService
public class EurekaDiscoverApplication {
```

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

}
```

Résultat :



## Conclusion

En conclusion, l'activité pratique "Mise en œuvre d'une architecture micro-services avec Spring Cloud" a été une expérience enrichissante qui m'a permis d'explorer et de comprendre les concepts fondamentaux des architectures de micro-services. Grâce à l'utilisation de Spring Cloud, j'ai pu découvrir comment découper une application monolithique en services indépendants, favorisant ainsi la scalabilité, la maintenabilité et la résilience de mes systèmes distribués.

L'implémentation de cette architecture a été rendue possible grâce aux fonctionnalités puissantes offertes par Spring Cloud, telles que la gestion de la communication inter-services, la découverte de services, la configuration distribuée et la tolérance aux pannes. J'ai appris à utiliser ces fonctionnalités pour construire des applications évolutives et résilientes, capables de s'adapter aux besoins changeants de l'entreprise.

Cette activité pratique m'a également permis de développer des compétences pratiques en matière de développement avec Spring Cloud. J'ai pu appliquer les concepts théoriques que j'ai appris à travers la réalisation d'un projet concret basé sur une architecture de micro-services. Cette expérience m'a donné une vision concrète des défis et des opportunités liés à la mise en œuvre d'une telle architecture.

Enfin, cette activité pratique a renforcé ma passion pour le développement d'applications modernes et m'a ouvert de nouvelles perspectives quant à la façon dont je peux concevoir des systèmes logiciels plus flexibles et résilients. Je suis reconnaissant envers mon professeur de m'avoir donné cette opportunité d'apprentissage stimulante et je suis impatient d'appliquer les connaissances acquises dans mes futurs projets.