

Département Mathématiques et Informatique

« Ingénierie Informatique : Big Data et Cloud Computing »

II-BDCC

Module : Systèmes Distribués basés sur les Micro services

TAF : TP Micro Services

Réalisé par :

Hamza El Anssari

Année Universitaire : 2019-2020

Introduction :

Les micro services sont une approche d'architecture et de développement d'une application composées de petits services.

- L'idée étant de découper un grand problème en petites unités implémentée sous forme de micro-services
- Chaque service est responsable d'une fonctionnalité,
- Chaque micro-service est développé, testé et déployé séparément des autres.
- Chaque micro service est développé en utilisant une technologie qui peut être différente des autres. (Java, C++, C#, PHP, NodeJS, Python,...)
- Chaque service tourne dans un processus séparé.
- Utilisant des mécanismes de communication légers (REST)
- La seule relation entre les différents micro services est l'échange de données effectué à travers les différentes APIs qu'ils exposent. (SOAP, REST, RMI, CORBA, JMS, MQP, ...)
- Lorsqu'on les combine, ces micro services peuvent réaliser des opérations très complexes.
- Ils sont faiblement couplés puisque chaque micro service est physiquement séparé des autres,
- Indépendance relative entre les différentes équipes qui développent les différents micro services.
- Facilité des tests et du déploiement
- Livraison continue.
- S'apprête bien à au processus du GL : TDD (Test Driver Développement) et les méthodes agiles
- Comme pour le cas d'une application monolithique, un micro service peut être composé de plusieurs très petites couches:
 - Couche DAO
 - Couche Métier,
 - Couches Techniques (REST, SOAP, RMI, JMS, AMQP, Sécurité, etc...)

Travail à faire :

1. Créer le micro service Customer-service
 - Créer l'entité Customer
 - Créer l'interface Custom Repository basée sur Spring Data
 - Déployer l'API Restful du micro-service en utilisant Spring Data Rest
 - Tester le Micro service
2. Créer le micro service Inventory-service
 - Créer l'entité Product
 - Créer l'interface Product Repository basée sur Spring Data
 - Déployer l'API Restful du micro-service en utilisant Spring Data Rest
 - Tester le Micro service
3. Créer la Gateway service en utilisant Spring Cloud Gateway
 - Tester la Service proxy en utilisant une configuration Statique basée sur le fichier application.yml
 - Tester la Service proxy en utilisant une configuration Statique basée une configuration Java
4. Créer l'annuaire Discovery Service basé sur Netflix Eureka Server
5. Tester le proxy en utilisant une configuration dynamique de Gestion des routes vers les micro services enregistrés dans l'annuaire Eureka Server
6. Créer le service Billing Service
7. Créer un service d'authentification Stateless basé sur Spring Security et Json Web Token. Ce service devrait permettre de :
 - Gérer les utilisateurs et les rôles de l'application
 - Authentifier un utilisateur en lui délivrant un access Token et un refresh Token de type JWT
 - Gérer les autorisation d'accès
 - Renouveler l'access Token à l'aide du refresh Token

I. Créer le micro service Customer-service :

➤ Entité et l'interface :

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

Customer Entité

```
@RepositoryRestResource
interface CustomerRepository extends JpaRepository<Customer,Long> {
    @RestResource(path = "/byName")
    Page<Customer> findByNameContains(@Param("kw") String name, Pageable pageable);
}

@Projection(name = "FullCustomer",types = Customer.class)
interface FullCustomerProjection extends Projection{
    Long getId();
    String getName();
    String getEmail();
}

@Projection(name = "NameCustomer",types = Customer.class)
interface NameCustomerProjection extends Projection{
    String getName();
}
```

interface Custom Repository

```
spring.cloud.discovery.enabled=true
server.port=8081
spring.application.name=customer-service
#management.endpoints.web.exposure.include=*
eureka.client.service-url.defaultZone=http://localhost:8765/eureka
```

Application properties

➤ Code :

```
package org.sid.customerservice;

import lombok.AllArgsConstructor; import lombok.Data; import lombok.NoArgsConstructor;

import lombok.ToString; 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.domain.Page;

import org.springframework.data.domain.Pageable;

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

import org.springframework.data.repository.query.Param;

import org.springframework.data.rest.core.annotation.RepositoryRestResource;

import org.springframework.data.rest.core.annotation.RestResource;

import org.springframework.data.rest.core.config.Projection;

import javax.persistence.Entity; import javax.persistence.GeneratedValue;

import javax.persistence.GenerationType; import javax.persistence.Id;

@SpringBootApplication

public class CustomerServiceApplication {

    public static void main(String[] args) {

        SpringApplication.run(CustomerServiceApplication.class, args); }

    @Bean

    CommandLineRunner start(CustomerRepository customerRepository){

        return args -> {

            customerRepository.save(new Customer(null,"Enset1","contact@enset-media.ma"));

            customerRepository.save(new Customer(null,"FSTM1","contact@fstm.ma"));

            customerRepository.save(new Customer(null,"ENSAM1","contact@ensam.ma"));

            customerRepository.save(new Customer(null,"Enset2","contact@enset-media.ma"));
```

```

        customerRepository.save(new Customer(null,"FSTM2","contact@fstm.ma"));

        customerRepository.save(new Customer(null,"ENSAM2","contact@ensam.ma"));

        customerRepository.save(new Customer(null,"Enset3","contact@enset-media.ma"));

        customerRepository.save(new Customer(null,"FSTM3","contact@fstm.ma"));

        customerRepository.save(new Customer(null,"ENSAM3","contact@ensam.ma"));

        customerRepository.save(new Customer(null,"Enset3","contact@enset-media.ma"));

        customerRepository.save(new Customer(null,"FSTM3","contact@fstm.ma"));

        customerRepository.save(new Customer(null,"ENSAM3","contact@ensam.ma"));

        customerRepository.save(new Customer(null,"Enset4","contact@enset-media.ma"));

        customerRepository.save(new Customer(null,"FSTM4","contact@fstm.ma"));

        customerRepository.save(new Customer(null,"ENSAM4","contact@ensam.ma"));

        customerRepository.save(new Customer(null,"Enset5","contact@enset-media.ma"));

        customerRepository.save(new Customer(null,"FSTM5","contact@fstm.ma"));

        customerRepository.save(new Customer(null,"ENSAM5","contact@ensam.ma"));

        customerRepository.findAll().forEach(System.out::println);

    },}}

@Entity @Data @NoArgsConstructor @AllArgsConstructor @ToString
class Customer{

    @Id @GeneratedValue(strategy = GenerationType.IDENTITY)

    private Long id; private String name; private String email; }

@RepositoryRestResource

interface CustomerRepository extends JpaRepository<Customer,Long> {

    @RestResource(path = "/byName")

    Page<Customer> findByNameContains(@Param("kw") String name, Pageable pageable);}

    @Projection(name = "FullCustomer",types = Customer.class)

interface FullCustomerProjection extends Projection{

    Long getId();

    String getName();

```


```
String getEmail();}
```

```
@Projection(name = "NameCustomer",types = Customer.class)
```

```
interface NameCustomerProjection extends Projection{
```

```
String getName(); }
```

➤ **Implémentation (localhost:8081):**



```
{
  "_embedded" : {
    "customers" : [ {
      "name" : "Enset1",
      "email" : "contact@enset-media.ma",
      "_links" : {
        "self" : {
          "href" : "http://localhost:8081/customers/1"
        },
        "customer" : {
          "href" : "http://localhost:8081/customers/1{?projection}",
          "templated" : true
        }
      }
    }, {
      "name" : "FSTM1",
      "email" : "contact@fstm.ma",
      "_links" : {
        "self" : {
          "href" : "http://localhost:8081/customers/2"
        },
        "customer" : {
          "href" : "http://localhost:8081/customers/2{?projection}",
          "templated" : true
        }
      }
    }, {
      "name" : "ENSAM1",
      "email" : "contact@ensam.ma",
      "_links" : {
        "self" : {
          "href" : "http://localhost:8081/customers/3"
        },
        "customer" : {
          "href" : "http://localhost:8081/customers/3{?projection}",
          "templated" : true
        }
      }
    }, {
      "name" : "Enset2",
      "email" : "contact@enset-media.ma",
      "_links" : {
        "self" : {
          "href" : "http://localhost:8081/customers/4"
        },
        "customer" : {
          "href" : "http://localhost:8081/customers/4{?projection}",
          "templated" : true
        }
      }
    }
  ]
}
```

All Customers


```
{
  "_embedded" : {
    "customers" : [ {
      "name" : "Enset1",
      "email" : "contact@enset-media.ma",
      "_links" : {
        "self" : {
          "href" : "http://localhost:8081/customers/1"
        },
        "customer" : {
          "href" : "http://localhost:8081/customers/1{?projection}",
          "templated" : true
        }
      }
    }, {
      "name" : "Enset2",
      "email" : "contact@enset-media.ma",
      "_links" : {
        "self" : {
          "href" : "http://localhost:8081/customers/4"
        },
        "customer" : {
          "href" : "http://localhost:8081/customers/4{?projection}",
          "templated" : true
        }
      }
    }, {
      "name" : "Enset3",
      "email" : "contact@enset-media.ma",
      "_links" : {
        "self" : {
          "href" : "http://localhost:8081/customers/7"
        },
        "customer" : {
          "href" : "http://localhost:8081/customers/7{?projection}",
          "templated" : true
        }
      }
    }, {
      "name" : "Enset3",
      "email" : "contact@enset-media.ma",
      "_links" : {
        "self" : {
          "href" : "http://localhost:8081/customers/10"
        },
        "customer" : {
          "href" : "http://localhost:8081/customers/10{?projection}",
          "templated" : true
        }
      }
    }
  ]
}
```

Full Customers avec projection

```
← → ↻ 🏠 ⓘ localhost:8081/customers/1
📱 Apps 🌐 DTD Examples 🌐 Marketing Dig... 🔥 localhost / 127...

{
  "name" : "Enset1",
  "email" : "contact@enset-media.ma",
  "_links" : {
    "self" : {
      "href" : "http://localhost:8081/customers/1"
    },
    "customer" : {
      "href" : "http://localhost:8081/customers/1{?projection}",
      "templated" : true
    }
  }
}
```

Customer id=1

```
← → ↻ 🏠 ⓘ localhost:8081/actuator/health
📱 Apps 🌐 DTD Examples 🌐 Marketing Dig... 🔥 localhost / 127...

{"status":"UP"}
```

Actuator health

← → ↻ 🏠 ⓘ localhost:8081/h2-console/login.do?jsessionId=8347a0029bb83a655f7529344d5dc7b5

🌐 Apps 📄 DTD Examples 🌐 Marketing Dig... 🚒 localhost / 127... 🚩 Java Swing Tu... 🗄️ MongoDB Co...

🔊 🔊 ☒ Auto commit 🔊 🔊 | Max rows: 1000 🔄 🔴 | 📄 | Auto complete Off 🔽 Auto select On 🔽 ?

🔑 jdbc:h2:mem:328f4c54-c351-458 Run Run Selected Auto complete Clear SQL statement:

🔑 CUSTOMER
🔑 INFORMATION_SCHEMA
🔑 Sequences
🔑 Users
🔑 H2 1.4.200 (2019-10-14)

SELECT * FROM CUSTOMER|

SELECT * FROM CUSTOMER;

ID	EMAIL	NAME
1	contact@enset-media.ma	Enset1
2	contact@fstm.ma	FSTM1
3	contact@ensam.ma	ENSAM1
4	contact@enset-media.ma	Enset2
5	contact@fstm.ma	FSTM2
6	contact@ensam.ma	ENSAM2
7	contact@enset-media.ma	Enset3
8	contact@fstm.ma	FSTM3
9	contact@ensam.ma	ENSAM3
10	contact@enset-media.ma	Enset3
11	contact@fstm.ma	FSTM3
12	contact@ensam.ma	ENSAM3
13	contact@enset-media.ma	Enset4
14	contact@fstm.ma	FSTM4
15	contact@ensam.ma	ENSAM4
16	contact@enset-media.ma	Enset5
17	contact@fstm.ma	FSTM5
18	contact@ensam.ma	ENSAM5

(18 rows, 7 ms)

Edit

Database

II. Créer le micro service Inventory-service :

➤ Entité et l'interface :

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

Product Entité

```
@RepositoryRestResource
interface ProductRepository extends JpaRepository<Product,Long> {
    @RestResource(path = "/byName")
    Page<Product> findByNameContains(@Param("kw") String name, Pageable pageable);
}

@Projection(name = "FullProduct",types = Product.class)
interface FullCustomerProjection extends Projection{
    Long getId();
    String getName();
    String getPrice();
}

@Projection(name = "NameProduct",types = Product.class)
interface NameCustomerProjection extends Projection{
    String getName();
}
```

interface Product Repository

```
spring.application.name=inventory-service
spring.cloud.discovery.enabled=true
server.port=8082
eureka.client.service-url.defaultZone=http://localhost:8765/eureka
```

Application properties

➤ Code :

```
package org.sid.customerservice;

import lombok.AllArgsConstructor;
import lombok.Data; import lombok.NoArgsConstructor;
import lombok.ToString;
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.domain.Page;
import org.springframework.data.domain.Pageable;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.data.repository.query.Param;
import
org.springframework.data.rest.core.annotation.RepositoryRestResource;
import org.springframework.data.rest.core.annotation.RestResource;
import org.springframework.data.rest.core.config.Projection;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;

@SpringBootApplication
public class InventoryServiceApplication {

    public static void main(String[] args) {

        SpringApplication.run(InventoryServiceApplication.class, args); }

    @Bean
    CommandLineRunner start(ProductRepository productRepository){

        return args -> {
```

```

        productRepository.save(new Product(null, "Computer Desk Top
HP", 900));

        productRepository.save(new Product(null, "Printer Epson", 80));

        productRepository.save(new Product(null, "MacBook Pro Lap
Top", 1800));

        productRepository.findAll().forEach(System.out::println);

    }; } }

@Entity @Data @NoArgsConstructor @AllArgsConstructor @ToString
class Product{

    @Id @GeneratedValue(strategy = GenerationType.IDENTITY)

    private Long id; private String name; private double price; }

@RepositoryRestResource

interface ProductRepository extends JpaRepository<Product, Long> {

    @RestResource(path = "/byName")

    Page<Product> findByNameContains(@Param("kw") String name, Pageable
pageable); }

@Projection(name = "FullProduct", types = Product.class)

interface FullCustomerProjection extends Projection{

    Long getId();

    String getName();

    String getPrice(); }

@Projection(name = "NameProduct", types = Product.class)

interface NameCustomerProjection extends Projection{

    String getName(); }

```

➤ **Implémentation (localhost:8082):**



```
{
  "_embedded" : {
    "products" : [ {
      "name" : "Computer Desk Top HP",
      "price" : 900.0,
      "_links" : {
        "self" : {
          "href" : "http://localhost:8082/products/1"
        },
        "product" : {
          "href" : "http://localhost:8082/products/1{?projection}",
          "templated" : true
        }
      }
    }, {
      "name" : "Printer Epson",
      "price" : 80.0,
      "_links" : {
        "self" : {
          "href" : "http://localhost:8082/products/2"
        },
        "product" : {
          "href" : "http://localhost:8082/products/2{?projection}",
          "templated" : true
        }
      }
    }, {
      "name" : "MacBook Pro Lap Top",
      "price" : 1800.0,
      "_links" : {
        "self" : {
          "href" : "http://localhost:8082/products/3"
        },
        "product" : {
          "href" : "http://localhost:8082/products/3{?projection}",
          "templated" : true
        }
      }
    }
  ]
},
}
```

All Products


```
{
  "_embedded" : {
    "products" : [ {
      "name" : "Computer Desk Top HP",
      "_links" : {
        "_self" : {
          "href" : "http://localhost:8082/products/1"
        },
        "product" : {
          "href" : "http://localhost:8082/products/1{?projection}",
          "templated" : true
        }
      }
    }, {
      "name" : "Printer Epson",
      "_links" : {
        "_self" : {
          "href" : "http://localhost:8082/products/2"
        },
        "product" : {
          "href" : "http://localhost:8082/products/2{?projection}",
          "templated" : true
        }
      }
    }
  ]
}, {
  "_links" : {
    "_first" : {
      "href" : "http://localhost:8082/products?projection=NameProduct&page=0&size=2"
    },
    "_self" : {
      "href" : "http://localhost:8082/products?projection=NameProduct&size=2"
    },
    "_next" : {
      "href" : "http://localhost:8082/products?projection=NameProduct&page=1&size=2"
    },
    "_last" : {
      "href" : "http://localhost:8082/products?projection=NameProduct&page=1&size=2"
    },
    "profile" : {
      "href" : "http://localhost:8082/profile/products"
    },
    "search" : {
      "href" : "http://localhost:8082/products/search"
    }
  }
},
```

Affiche le nom de chaque produits (Projection)


```
localhost:8082/products/3

{
  "name" : "MacBook Pro Lap Top",
  "price" : 1800.0,
  "_links" : {
    "self" : {
      "href" : "http://localhost:8082/products/3"
    },
    "product" : {
      "href" : "http://localhost:8082/products/3{?projection}",
      "templated" : true
    }
  }
}
```

Product id=3

localhost:8082/h2-console/login.do?jsessionId=5919ec392db3331d05824e5947c767c5

Apps DTD Examples Marketing Dig... localhost / 127... Java Swing Tu... MongoDB

Auto commit Max rows: 1000 Auto complete Off Auto select On

jdbc:h2:mem:eead2b8e-3663-4fc Run Run Selected Auto complete Clear SQL statement:

PRODUCT
INFORMATION_SCHEMA
Sequences
Users
H2 1.4.200 (2019-10-14)

SELECT * FROM PRODUCT

SELECT * FROM PRODUCT;

ID	NAME	PRICE
1	Computer Desk Top HP	900.0
2	Printer Epson	80.0
3	MacBook Pro Lap Top	1800.0

(3 rows, 6 ms)

Edit

Database

III. Créer la Gateway service en utilisant Spring Cloud Gateway :

Statique Routes Configuration

```
GatewayServiceApplication.java x application.yml x
1  spring:
2    cloud:
3      gateway:
4        routes:
5          - id: r1
6            uri: http://localhost:8081/
7            predicates:
8              - Path= /customers/**
9          - id : r2
10             uri: http://localhost:8082/
11             predicates:
12               - Path= /products/**
13        discovery:
14          enabled: false
15  server:
16    port: 8088
```

Application properties

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

Statique Configuration

➤ Code :

```
package org.sid.gatewayservice;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
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 gatewayRoutes(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();

    }

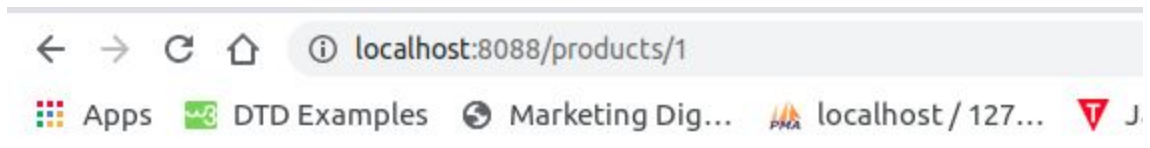
}
```

➤ **Implémentation (localhost:8088):**

```
localhost:8088/products/

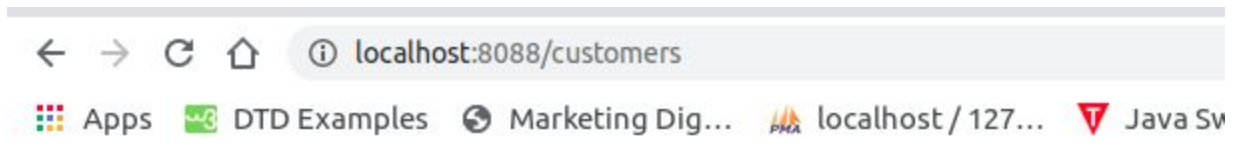
{
  "_embedded" : {
    "products" : [ {
      "name" : "Computer Desk Top HP",
      "price" : 900.0,
      "_links" : {
        "self" : {
          "href" : "http://localhost:8082/products/1"
        },
        "product" : {
          "href" : "http://localhost:8082/products/1{?projection}",
          "templated" : true
        }
      }
    }, {
      "name" : "Printer Epson",
      "price" : 80.0,
      "_links" : {
        "self" : {
          "href" : "http://localhost:8082/products/2"
        },
        "product" : {
          "href" : "http://localhost:8082/products/2{?projection}",
          "templated" : true
        }
      }
    }, {
      "name" : "MacBook Pro Lap Top",
      "price" : 1800.0,
      "_links" : {
        "self" : {
          "href" : "http://localhost:8082/products/3"
        },
        "product" : {
          "href" : "http://localhost:8082/products/3{?projection}",
          "templated" : true
        }
      }
    }
  ]
}
```

All Products



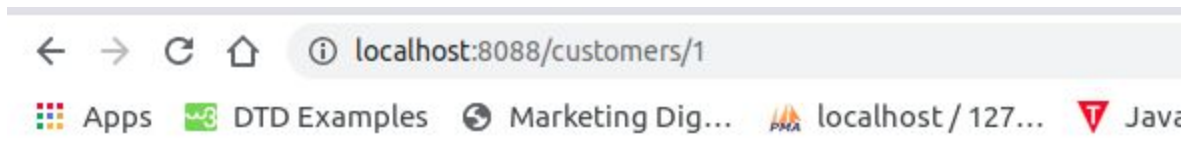
```
{
  "name" : "Computer Desk Top HP",
  "price" : 900.0,
  "_links" : {
    "self" : {
      "href" : "http://localhost:8082/products/1"
    },
    "product" : {
      "href" : "http://localhost:8082/products/1{?projection}",
      "templated" : true
    }
  }
}
```

Produit id=1



```
{
  "_embedded" : {
    "customers" : [ {
      "name" : "Enset1",
      "email" : "contact@enset-media.ma",
      "_links" : {
        "self" : {
          "href" : "http://localhost:8081/customers/1"
        },
        "customer" : {
          "href" : "http://localhost:8081/customers/1{?projection}",
          "templated" : true
        }
      }
    }, {
      "name" : "FSTM1",
      "email" : "contact@fstm.ma",
      "_links" : {
        "self" : {
          "href" : "http://localhost:8081/customers/2"
        },
        "customer" : {
          "href" : "http://localhost:8081/customers/2{?projection}",
          "templated" : true
        }
      }
    }, {
      "name" : "ENSAM1",
      "email" : "contact@ensam.ma",
      "_links" : {
        "self" : {
          "href" : "http://localhost:8081/customers/3"
        },
        "customer" : {
          "href" : "http://localhost:8081/customers/3{?projection}",
          "templated" : true
        }
      }
    }
  ]
}
```

All Customers

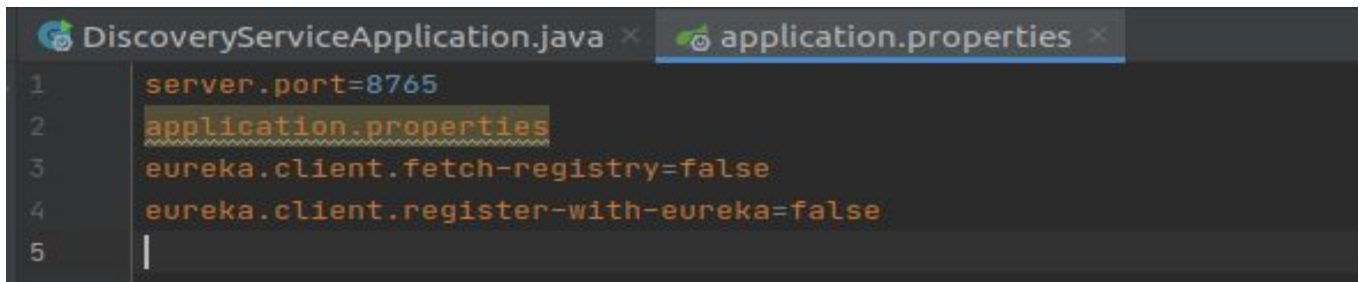


```
{
  "name" : "Enset1",
  "email" : "contact@enset-media.ma",
  "_links" : {
    "self" : {
      "href" : "http://localhost:8081/customers/1"
    },
    "customer" : {
      "href" : "http://localhost:8081/customers/1{?projection}",
      "templated" : true
    }
  }
}
```

Customer id=1

IV. Créer l'annuaire Discovery Service basé sur Netflix Eureka Server :

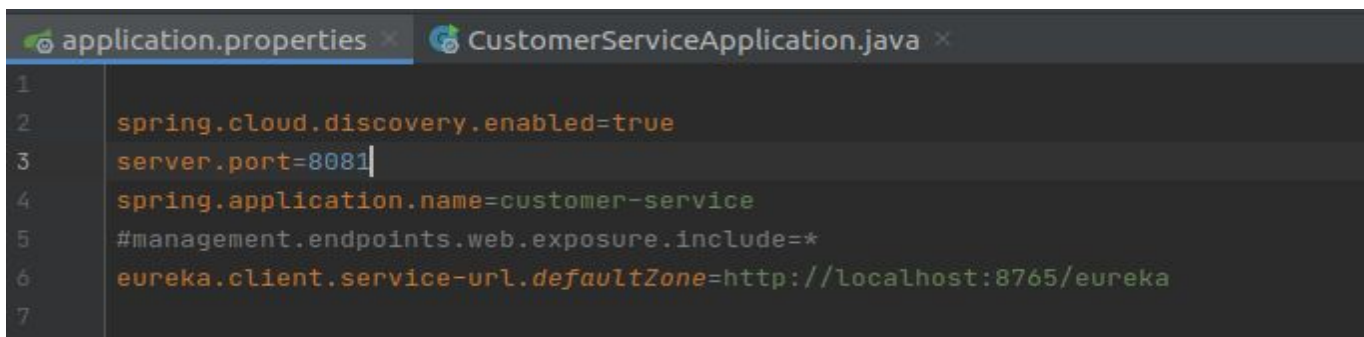
Dynamique Routes Configuration



The screenshot shows an IDE with two tabs: 'DiscoveryServiceApplication.java' and 'application.properties'. The 'application.properties' tab is active, displaying the following configuration:

```
1 server.port=8765
2 application.properties
3 eureka.client.fetch-registry=false
4 eureka.client.register-with-eureka=false
5 |
```

Application properties



The screenshot shows an IDE with two tabs: 'application.properties' and 'CustomerServiceApplication.java'. The 'application.properties' tab is active, displaying the following configuration:

```
1
2 spring.cloud.discovery.enabled=true
3 server.port=8081
4 spring.application.name=customer-service
5 #management.endpoints.web.exposure.include=*
6 eureka.client.service-url.defaultZone=http://localhost:8765/eureka
7
```

Customer application



The screenshot shows an IDE with two tabs: 'application.properties' and 'InventoryServiceApplication.java'. The 'application.properties' tab is active, displaying the following configuration:

```
1
2 spring.application.name=inventory-service
3 spring.cloud.discovery.enabled=true
4 server.port=8082
5 eureka.client.service-url.defaultZone=http://localhost:8765/eureka
```

Inventory application

➤ Code :

```
package or.sid.discoveryervice;  
  
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);  
  
    }  
  
}
```

➤ Implémentation :

localhost:8765

Apps DTD Examples Marketing Dig... localhost / 127... Java Swing Tu... MongoDB Co... Medium - Get... Hacker Noon DZone: Progra... App Not Found

Renews threshold	5
Renews (last min)	2

DS Replicas

localhost

Instances currently registered with Eureka

Application	AMIs	Availability Zones	Status
CUSTOMER-SERVICE	n/a (1)	(1)	UP (1) - 192.168.8.122:customer-service:8081
INVENTORY-SERVICE	n/a (1)	(1)	UP (1) - 192.168.8.122:inventory-service:8082

General Info

Name	Value
total-avail-memory	234mb
num-of-cpus	4
current-memory-usage	53mb (22%)
server-uptime	00:01
registered-replicas	http://localhost:8761/eureka/
unavailable-replicas	http://localhost:8761/eureka/
available-replicas	

Localhost:8765

V. Créer le service Billing Service :

➤ Code :

```
package org.sid.billingervice;

import com.fasterxml.jackson.annotation.JsonProperty;

import lombok.AllArgsConstructor;import lombok.Data; import
lombok.NoArgsConstructor;

import org.springframework.beans.factory.annotation.Autowired;
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.cloud.openfeign.FeignClient;
import org.springframework.context.annotation.Bean;
import org.springframework.data.jpa.repository.JpaRepository;

import
org.springframework.data.rest.core.annotation.RepositoryRestResource;

import org.springframework.hateoas.PagedModel;

import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RestController;

import javax.persistence.*;import java.util.Collection; import
java.util.Date;import java.util.List;

@SpringBootApplication

@EnableFeignClients

public class BillingServiceApplication {

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

    @Bean

    CommandLineRunner start(BillRepository billRepository,
ProductItemRepository productItemRepository, InventoryServiceClient
inventoryServiceClient, CustomerServiceClient customerServiceClient){
```

```

        return args -> {

            Bill bill=new Bill();

            bill.setBillingDate(new Date());

            Customer customer=customerServiceClient.findCustomerById(1L);

            bill.setCustomerID(customer.getId());

            billRepository.save(bill);

            inventoryServiceClient.findAll().getContent().forEach(p->{

                productItemRepository.save(new
                ProductItem(null,null,p.getId(),p.getPrice(),(int)(1+Math.random()*1000)
                ,bill));

            });

        };

    }

}

@Entity
@Data
@NoArgsConstructor
@AllArgsConstructor
class Bill{

    @Id

    @GeneratedValue(strategy = GenerationType.IDENTITY)

    private Long id; private Date billingDate;

    @Transient

    @OneToMany(mappedBy = "bill")

    private Collection<ProductItem> productItems;

    @Transient private Customer customer;

    private long customerID;

}

@RepositoryRestResource

interface BillRepository extends JpaRepository<Bill,Long> {}

@Entity @Data @NoArgsConstructor @AllArgsConstructor

```

```

class ProductItem{

    @Id @GeneratedValue(strategy = GenerationType.IDENTITY)

    private Long id;

    @Transient

    private Product product; private long productID;

    private double price; private double quantity;

    @ManyToOne

    @JsonProperty(access = JsonProperty.Access.WRITE_ONLY)

    private Bill bill;

}

@RepositoryRestResource

interface ProductItemRepository extends

    JpaRepository<ProductItem,Long>{

    List<ProductItem> findByBillId(Long billID);

}

```

```

@Data

class Product{

    private Long id; private String name; private double price;

}

```

```

@Data

class Customer{

    private Long id; private String name; private String email;

}

```

```

@FeignClient(name="inventory-service")

interface InventoryServiceClient{

    @GetMapping("/products/{id}?projection=FullProduct")

    Product findProductById(@PathVariable("id") Long id);

}

```

```

    @GetMapping("/products?projection=FullProduct")
    PagedModel<Product> findAll();
}

@FeignClient(name="customer-service")
interface CustomerServiceClient{

    @GetMapping("/customers/{id}?projection=FullCustomer")
    Customer findCustomerById(@PathVariable("id") Long id);
}

@RestController
class BillRestController{

    @Autowired private BillRepository billRepository;

    @Autowired private ProductItemRepository productItemRepository;

    @Autowired private CustomerServiceClient customerServiceClient;

    @Autowired private InventoryServiceClient inventoryServiceClient;

    @GetMapping("/bills/Full/{id}")
    Bill getBill(@PathVariable(name="id") Long id){

        Bill bill=billRepository.findById(id).get();

        bill.setCustomer(customerServiceClient.findCustomerById(bill.getCustomer
ID()));

        bill.setProductItems(productItemRepository.findByBillId(id));

        bill.getProductItems().forEach(pi->{

            pi.setProduct(inventoryServiceClient.findProductById(pi.getProductID()))
;

        });

        return bill; }
}

```

➤ Implémentation :

The screenshot shows the H2 console interface. The left sidebar displays the database schema with tables: BILL, PRODUCT_ITEM, and INFORMATION_SCHEMA. The main area shows the SQL statement `SELECT * FROM PRODUCT_ITEM;` being executed. The results are displayed in a table with 5 columns: ID, PRICE, PRODUCTID, QUANTITY, and BILL_ID. The results show 3 rows of data.

ID	PRICE	PRODUCTID	QUANTITY	BILL_ID
1	900.0	1	794.0	1
2	80.0	2	751.0	1
3	1800.0	3	454.0	1

(3 rows, 2 ms)

Products Table

The screenshot shows the H2 console interface. The left sidebar displays the database schema with tables: BILL, PRODUCT_ITEM, and INFORMATION_SCHEMA. The main area shows the SQL statement `SELECT * FROM BILL;` being executed. The results are displayed in a table with 3 columns: ID, BILLING_DATE, and CUSTOMERID. The results show 1 row of data.

ID	BILLING_DATE	CUSTOMERID
1	2020-12-12 01:56:48.57	1

(1 row, 1 ms)

Bills Table

```
{
  "_embedded" : {
    "bills" : [ {
      "billingDate" : "2020-12-12T00:56:48.570+00:00",
      "productItems" : null,
      "customer" : null,
      "customerID" : 1,
      "_links" : {
        "self" : {
          "href" : "http://localhost:8083/bills/1"
        },
        "bill" : {
          "href" : "http://localhost:8083/bills/1"
        }
      }
    } ]
  },
  "_links" : {
    "self" : {
      "href" : "http://localhost:8083/bills"
    },
    "profile" : {
      "href" : "http://localhost:8083/profile/bills"
    }
  },
  "page" : {
    "size" : 20,
    "totalElements" : 1,
    "totalPages" : 1,
    "number" : 0
  }
}
```

All Bills

```

localhost:8083/bills/1

Apps DTD Examples Marketing Dig... localhost / 127...

{
  "billingDate" : "2020-12-12T00:56:48.570+00:00",
  "productItems" : null,
  "customer" : null,
  "customerID" : 1,
  "_links" : {
    "self" : {
      "href" : "http://localhost:8083/bills/1"
    },
    "bill" : {
      "href" : "http://localhost:8083/bills/1"
    }
  }
}

```

Bill id=1

Instances currently registered with Eureka

Application	AMIs	Availability Zones	Status
CUSTOMER-SERVICE	n/a (1)	(1)	UP (1) - 192.168.8.122:customer-service:8081
INVENTORY-SERVICE	n/a (1)	(1)	UP (1) - 192.168.8.122:inventory-service:8082
UNKNOWN	n/a (2)	(2)	UP (2) - 192.168.8.122:8088 , 192.168.8.122:8761

Eureka

```

localhost:8083/bills/Full/1?projection=FullCustomer

Apps DTD Examples Marketing Dig... localhost / 127... Java Swing Tu... MongoDB Co... Medium - Get... Hacker Noon DZone: Progra... App Not Found

{"id":1,"billingDate":"2020-12-12T01:23:40.255+00:00","productItems":[{"id":1,"product":{"id":1,"name":"Computer Desk Top HP","price":900.0},"productID":1,"price":900.0,"quantity":365.0},{"id":2,"product":{"id":2,"name":"Printer Epson","price":80.0},"productID":2,"price":80.0,"quantity":632.0},{"id":3,"product":{"id":3,"name":"MacBook Pro Lap Top","price":1800.0},"productID":3,"price":1800.0,"quantity":780.0}], "customer":{"id":1,"name":"Enset1","email":"contact@enset-media.ma"},"customerID":1}

```

Bill id=1 Full Customer

VI. service d'authentification Stateless :

➤ Code de billing service après l'ajout de service d'authentification :

```
package org.sid.billingervice;

import com.fasterxml.jackson.annotation.JsonProperty;
import com.fasterxml.jackson.databind.ObjectMapper;
import io.jsonwebtoken.Claims;
import io.jsonwebtoken.Jwts;
import io.jsonwebtoken.SignatureAlgorithm;
import lombok.AllArgsConstructor; import lombok.Data; import
lombok.NoArgsConstructor;

import org.springframework.beans.factory.annotation.Autowired;
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.cloud.openfeign.FeignClient;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.data.jpa.repository.JpaRepository;

import
org.springframework.data.rest.core.annotation.RepositoryRestResource;
import org.springframework.hateoas.PagedModel;
import org.springframework.http.HttpMethod;

import
org.springframework.security.authentication.AuthenticationManager;

import
org.springframework.security.authentication.UsernamePasswordAuthenticati
onToken;

import
org.springframework.security.config.annotation.authentication.builders.A
uthenticationManagerBuilder;
```

```
import
org.springframework.security.config.annotation.web.builders.HttpSecurity
;

import
org.springframework.security.config.annotation.web.configuration.EnableWebSecurity;

import
org.springframework.security.config.annotation.web.configuration.WebSecurityConfigurerAdapter;

import org.springframework.security.config.http.SessionCreationPolicy;

import org.springframework.security.core.Authentication;

import org.springframework.security.core.AuthenticationException;

import org.springframework.security.core.GrantedAuthority;

import
org.springframework.security.core.authority.SimpleGrantedAuthority;

import org.springframework.security.core.context.SecurityContextHolder;

import org.springframework.security.core.userdetails.User;

import org.springframework.security.core.userdetails.UserDetails;

import org.springframework.security.core.userdetails.UserDetailsService;

import
org.springframework.security.core.userdetails.UsernameNotFoundException;

import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;

import
org.springframework.security.web.authentication.UsernamePasswordAuthenticationFilter;

import org.springframework.stereotype.Service;

import org.springframework.ui.Model;

import org.springframework.validation.BindingResult;

import org.springframework.web.bind.annotation.*;

import org.springframework.web.filter.OncePerRequestFilter;

import org.springframework.web.servlet.ModelAndView;

import org.springframework.web.servlet.view.RedirectView;


import javax.persistence.*;

import javax.servlet.FilterChain;
```

```

import javax.servlet.ServletException;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.transaction.Transactional;
import javax.validation.Valid;
import java.io.IOException;
import java.util.*;

@SpringBootApplication
@EnableFeignClients
public class BillingServiceApplication {

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

    @Bean
    public BCryptPasswordEncoder bCryptPasswordEncoder() {
        return new BCryptPasswordEncoder();
    }

    @Bean
    CommandLineRunner start(AccountService accountService, BillRepository
billRepository, ProductItemRepository productItemRepository,
InventoryServiceClient inventoryServiceClient, CustomerServiceClient
customerServiceClient) {
        return args -> {
            Bill bill = new Bill();
            bill.setBillingDate(new Date());

            Customer customer =
customerServiceClient.findCustomerById(1L);

            System.out.println(customer.toString());

```

```

        bill.setCustomerId(customer.getId());

        billRepository.save(bill);

        inventoryServiceClient.findAll().getContent().forEach(p -> {
            productItemRepository.save(new ProductItem(null, null,
p.getId(), p.getPrice(), (int) (1 + Math.random() * 1000), bill));
        });

        accountService.saveRole(new AppRole(null, "USER"));

        accountService.saveRole(new AppRole(null, "ADMIN"));

        accountService.saveUser(new AppUser(null, "user", "1234",
null));

        accountService.saveUser(new AppUser(null, "admin", "1234",
null));

        accountService.addRoleToUser("user", "USER");

        accountService.addRoleToUser("admin", "USER");

        accountService.addRoleToUser("admin", "ADMIN");

    };
}
}

```

@Entity

@Data

@NoArgsConstructor

@AllArgsConstructor

class Bill{

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id; private Date billingDate;

@Transient

@OneToMany(mappedBy = "bill")

private Collection<ProductItem> productItems;

@Transient private Customer customer;

```

        private long customerID;
    }

@RepositoryRestResource
interface BillRepository extends JpaRepository<Bill,Long> {}

@Entity @Data @NoArgsConstructor @AllArgsConstructor
class ProductItem{

    @Id @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    @Transient
    private Product product; private long productID;

    private double price; private double quantity;

    @ManyToOne

    @JsonProperty(access = JsonProperty.Access.WRITE_ONLY)
    private Bill bill;
}

@RepositoryRestResource
interface ProductItemRepository extends

    JpaRepository<ProductItem,Long>{

    List<ProductItem> findByBillId(Long billID);
}

@Data
class Product{

    private Long id; private String name; private double price;
}

@Data
class Customer{

    private Long id; private String name; private String email;

```

```
}
```

```
@FeignClient(name="inventory-service")
```

```
interface InventoryServiceClient{
```

```
    @GetMapping("/products/{id}?projection=FullProduct")
```

```
    Product findProductById(@PathVariable("id") Long id);
```

```
    @GetMapping("/products?projection=FullProduct")
```

```
    PagedModel<Product> findAll();
```

```
}
```

```
@FeignClient(name="customer-service")
```

```
interface CustomerServiceClient{
```

```
    @GetMapping("/customers/{id}?projection=FullCustomer")
```

```
    Customer findCustomerById(@PathVariable("id") Long id);
```

```
}
```

```
@RestController
```

```
class BillRestController{
```

```
    @Autowired private BillRepository billRepository;
```

```
    @Autowired private ProductItemRepository productItemRepository;
```

```
    @Autowired private CustomerServiceClient customerServiceClient;
```

```
    @Autowired private InventoryServiceClient inventoryServiceClient;
```

```
    @GetMapping("/bills")
```

```
    public List<Bill> listTasks(){
```

```
        return billRepository.findAll();
```

```
    }
```

```
    @PostMapping("/bills")
```

```
    public Bill save(@RequestBody Bill task){
```

```
        return billRepository.save(task);
```

```
    }
```

```

@GetMapping("/bills/Full/{id}")

Bill getBill(@PathVariable(name="id") Long id){

    Bill bill=billRepository.findById(id).get();

    bill.setCustomer(customerServiceClient.findCustomerById(bill.getCustomer
ID()));

    bill.setProductItems(productItemRepository.findByBillId(id));

    bill.getProductItems().forEach(pi->{

pi.setProduct(inventoryServiceClient.findProductById(pi.getProductID()))
;

    });

    return bill; }

@RequestMapping(value = "/admin/bills/delete",method=
RequestMethod.GET)

    public RedirectView delete(Model model,
@RequestParam(name="id",required = true)Long id){

        billRepository.deleteById(id);

        return new RedirectView("/bills");

    }

@RequestMapping(value="/admin/form",method= RequestMethod.GET)

    public ModelAndView form(Model model,
@RequestParam(name="id",defaultValue = "0") Long id){

        Bill p = (billRepository.existsById(id))?
billRepository.getOne(id):new Bill();

        model.addAttribute("bill", p);

        return new ModelAndView("form");

    }

@RequestMapping(value="/admin/save",method=RequestMethod.POST)

    public ModelAndView save(Model model, @Valid Bill p, BindingResult
bindingResult){

        if(bindingResult.hasErrors()) return new ModelAndView("form");

```

```

        billRepository.save(p);

        model.addAttribute("bill", p);

        return new ModelAndView("confirmation");
    }

    @RequestMapping(value="/403",method= RequestMethod.GET)

    public ModelAndView error(){

        return new ModelAndView("error/403");
    }
}

```

```

/* Security */

```

```

@Entity
@Data @AllArgsConstructor @NoArgsConstructor
class AppRole {

    @Id @GeneratedValue

    private Long id;

    private String role;
}

```

```

@Entity
@Data @AllArgsConstructor @NoArgsConstructor
class AppUser {

    @Id @GeneratedValue

    private Long id;

    private String username;

    private String password;

    @ManyToMany(fetch=FetchType.EAGER)

```



```

        private Collection<AppRole> roles=new ArrayList<>();
    }

    interface AppUserRepository extends JpaRepository<AppUser, Long> {

        AppUser findByUsername(String username);
    }

    interface AppRoleRepository extends JpaRepository<AppRole,Long>{

        AppRole findByRole(String role);
    }

    interface AccountService {

        AppUser saveUser(AppUser u);

        AppRole saveRole(AppRole r);

        AppUser findUserByUsername(String username);

        void addRoleToUser(String username,String role);
    }

    @Service
    @Transactional
    class AccountServiceImpl implements AccountService {

        @Autowired
        private AppUserRepository userRepository;

        @Autowired
        private AppRoleRepository roleRepository;

        @Autowired
        private BCryptPasswordEncoder bCryptPasswordEncoder;

        @Override
        public AppUser saveUser(AppUser u) {

            u.setPassword(bCryptPasswordEncoder.encode(u.getPassword()));

            return userRepository.save(u);
        }
    }

```

```

@Override
public AppRole saveRole(AppRole r) {
    return roleRepository.save(r);
}

@Override
public AppUser findUserByUsername(String username) {
    return userRepository.findByUsername(username);
}

@Override
public void addRoleToUser(String username, String roleName) {
    AppUser user=userRepository.findByUsername(username);
    AppRole role=roleRepository.findByRole(roleName);
    user.getRoles().add(role);
}
}

```

```

@Data @AllArgsConstructor @NoArgsConstructor
class RegistrationForm {
    private String username;
    private String password;
    private String repassword;
}

```

```

@RestController
class UserController {
    @Autowired
    private AccountService accountService;

    @PostMapping("/users")
    public AppUser signUp(@RequestBody RegistrationForm data) {
        String username=data.getUsername();
        AppUser user=accountService.findUserByUsername(username);
    }
}

```

```

        if(user!=null) throw new RuntimeException("This user already
exists, Try with an other username");

        String password=data.getPassword(); String
repassword=data.getRepassword();

        if(!password.equals(repassword))

            throw new RuntimeException("You must confirm your password");

        AppUser u=new AppUser(); u.setUsername(username);
u.setPassword(password);

        accountService.saveUser(u);

        accountService.addRoleToUser(username, "USER");

        return (u);
    }
}

@Configuration
@EnableWebSecurity

class SecurityConfig extends WebSecurityConfigurerAdapter {

    @Autowired

    private UserDetailsService userDetailsService;

    @Autowired

    private BCryptPasswordEncoder bCryptPasswordEncoder;


    @Override

    protected void configure(AuthenticationManagerBuilder auth) throws
Exception {

        auth.userDetailsService(userDetailsService)

            .passwordEncoder(bCryptPasswordEncoder);

    }


    @Override

    protected void configure(HttpSecurity http) throws Exception {

        http.csrf().disable()

        // don't create session

```

```

.sessionManagement().sessionCreationPolicy(SessionCreationPolicy.STATELESS)

.and()

.authorizeRequests()

.antMatchers("/users/**", "/login/**")

.permitAll()

.antMatchers(HttpMethod.POST,
"/bills/**").hasAuthority("ADMIN")

.anyRequest().authenticated()

.and()

.addFilter(new
JWTAuthenticationFilter(authenticationManager()))

.addFilterBefore(new JWTAuthorizationFilter(),
UsernamePasswordAuthenticationFilter.class);
}

}

class SecurityConstants {

    public static final String SECRET = "elanssarihamza@gmail.com";

    public static final long EXPIRATION_TIME = 864_000_000;

    public static final String TOKEN_PREFIX = "Bearer ";

    public static final String HEADER_STRING = "Authorization";

}

@Service

class UserDetailsServiceImpl implements UserDetailsService {

    @Autowired

    private AccountService accountService;

    @Override

    public UserDetails loadUserByUsername(String username) throws
UsernameNotFoundException {

        AppUser u=accountService.findUserByUsername(username);

        if(u==null) throw new UsernameNotFoundException(username);

```

```

        Collection<GrantedAuthority> authorities=new ArrayList<>();

        u.getRoles().forEach(r->{

            authorities.add(new SimpleGrantedAuthority(r.getRole()));

        });

        return new User(u.getUsername(), u.getPassword(), authorities);

    }

}

```

```

class JWTAuthenticationFilter extends
UsernamePasswordAuthenticationFilter {

    private AuthenticationManager authenticationManager;

    public JWTAuthenticationFilter(AuthenticationManager
authenticationManager) {

        super();

        this.authenticationManager = authenticationManager;

    }

    @Override

    public Authentication attemptAuthentication(HttpServletRequest
request,

                                                HttpServletResponse
response) throws AuthenticationException {

        AppUser user=null;

        try {

            user = new ObjectMapper().readValue(request.getInputStream(),
AppUser.class);

        } catch (Exception e) {

            throw new RuntimeException(e);

        }

        return authenticationManager.authenticate(

            new UsernamePasswordAuthenticationToken(

                user.getUsername(),

                user.getPassword()

            ));

    }

}

```

```

    }

    @Override
    protected void successfulAuthentication(HttpServletRequest request,
        HttpServletResponse
            response, FilterChain chain,
                Authentication authResult)
        throws IOException, ServletException {
        User springUser=(User)authResult.getPrincipal();
        String jwtToken= Jwts.builder()
            .setSubject(springUser.getUsername())
            .setExpiration(new
                Date(System.currentTimeMillis()+SecurityConstants.EXPIRATION_TIME))
            .signWith(SignatureAlgorithm.HS512,
                SecurityConstants.SECRET)
            .claim("roles", springUser.getAuthorities())
            .compact();
        response.addHeader(SecurityConstants.HEADER_STRING,
            SecurityConstants.TOKEN_PREFIX+jwtToken);
    }
}

```

```

class JWTAuthorizationFilter extends OncePerRequestFilter {

    @Override
    protected void doFilterInternal(HttpServletRequest request,
        HttpServletResponse response,
            FilterChain chain)
        throws IOException, ServletException {
        response.addHeader("Access-Control-Allow-Origin", "*");
        response.addHeader("Access-Control-Allow-Headers", "Origin,
            Accept, X-Requested-With, Content-Type, Access-Control-Request-Method,
            Access-Control-Request-Headers,authorization");
    }
}

```

```

        response.setHeader("Access-Control-Expose-Headers",
"Access-Control-Allow-Origin, Access-Control-Allow-Credentials,
authorization");

        if(request.getMethod().equals("OPTIONS")){

            response.setStatus(HttpServletResponse.SC_OK);

        }

        else {

            String
jwtToken=request.getHeader(SecurityConstants.HEADER_STRING);

            if(jwtToken==null ||
!jwtToken.startsWith(SecurityConstants.TOKEN_PREFIX)) {

                chain.doFilter(request, response); return;

            }

            Claims claims=Jwts.parser()

                .setSigningKey(SecurityConstants.SECRET)

.parseClaimsJws(jwtToken.replace(SecurityConstants.TOKEN_PREFIX,""))

                .getBody();

            String username=claims.getSubject();

            ArrayList<Map<String, String>> roles=(ArrayList<Map<String,
String>>)

                claims.get("roles");

            Collection<GrantedAuthority> authorities=new ArrayList<>();

            roles.forEach(r->{

                authorities.add(new
SimpleGrantedAuthority(r.get("authority")));

            });

            UsernamePasswordAuthenticationToken authenticationToken=

                new UsernamePasswordAuthenticationToken(username,
null,authorities);

SecurityContextHolder.getContext().setAuthentication(authenticationToken
);

            chain.doFilter(request, response);

        }}}

```

➤ Implémentation :

The screenshot shows a web browser's developer tools interface. At the top, the 'Method' is 'GET' and the 'Request URL' is 'http://localhost:8083/bills'. A 'SEND' button is visible. Below this, the 'Parameters' section is collapsed. The main area shows a '403 Forbidden' status with a response time of '11.12 ms'. A 'DETAILS ^' link is on the right. Below the status bar, there are four tabs: 'Response headers' (12), 'Request headers' (4), 'Redirects' (0), and 'Timings'. The 'Response headers' tab is selected, showing a list of headers including 'access-control-allow-origin: *', 'access-control-allow-headers: Origin, Accept, X-Requested-With, Content-Type, Access-Control-Request-Method, Access-Control-Request-Headers, authorization', 'access-control-expose-headers: Access-Control-Allow-Origin, Access-Control-Allow-Credentials, authorization', 'x-content-type-options: nosniff', 'x-xss-protection: 1; mode=block', 'cache-control: no-cache, no-store, max-age=0, must-revalidate', 'pragma: no-cache', 'expires: 0', 'x-frame-options: DENY', 'content-type: application/json', 'transfer-encoding: chunked', and 'date: Sat, 12 Dec 2020 20:31:46 GMT'. Below the headers, there is a JSON response body:

```
{
  "timestamp": "2020-12-12T20:31:46.370+00:00",
  "status": 403,
  "error": "Forbidden",
  "message": "Access Denied",
  "path": "/bills"
}
```

Test : Consulter les listes des bills

Method POST Request URL http://localhost:8083/users Show panel SEND ⋮

Parameters ⌵

200 OK 178.19 ms DETAILS ^

POST http://localhost:8083/users

Response headers 12 Request headers 4 Redirects 0 Timings

access-control-allow-origin: *
access-control-allow-headers: Origin, Accept, X-Requested-With, Content-Type, Access-Control-Request-Method, Access-Control-Request-Headers, authorization
access-control-expose-headers: Access-Control-Allow-Origin, Access-Control-Allow-Credentials, authorization
x-content-type-options: nosniff
x-xss-protection: 1; mode=block
cache-control: no-cache, no-store, max-age=0, must-revalidate
pragma: no-cache
expires: 0
x-frame-options: DENY
content-type: application/json
transfer-encoding: chunked
date: Sat, 12 Dec 2020 20:38:23 GMT

📄 🔍 <> ⋮

```
{
  "id": 5,
  "username": "hamza",
  "password": "$2a$10$WSp4wmvwiBH6uEfrH7hB7On0zzJBnmDjdPD88iAbEUxGF6g7E1cKW",
  "roles": [Array(1)]
  -0: {
    "id": 1,
    "role": "USER"
  }
}
```

Ajouter un utilisateur

Header name Header value
authorization Bearer eyJhbGciOiJIUzUxMiJ9.eyJzdWIiOiJhZG1pbilslmV4cCI6MTYwODY2OTY1Nywicm9sZXMiOiI7ImF1dGhvcm

Method Request URI
GET http://localhost:8083/bills

Parameters

200 OK 16.83 ms

DETAILS ^

GET http://localhost:8083/bills

Response headers 13

Request headers 4

Redirects 0

Timings

```
access-control-allow-origin: *
access-control-allow-headers: Origin, Accept, X-Requested-With, Content-Type, Access-Control-Request-Method, Access-Control-Request-Headers, authorization
access-control-expose-headers: Access-Control-Allow-Origin, Access-Control-Allow-Credentials, authorization
set-cookie: JSESSIONID=D6D660A8597550C261C6B5F5F6FE7A9B; Path=/; HttpOnly
x-content-type-options: nosniff
x-xss-protection: 1; mode=block
cache-control: no-cache, no-store, max-age=0, must-revalidate
pragma: no-cache
expires: 0
x-frame-options: DENY
content-type: application/json
transfer-encoding: chunked
date: Sat, 12 Dec 2020 20:45:20 GMT
```

Copy Download Expand Collapse

```
[Array(1)]
  0: {
    "id": 1,
    "billingDate": "2020-12-12T20:08:27.249+00:00",
    "productItems": null,
    "customer": null,
    "customerID": 1
  }
],
```

Liste des bills

Method POST Request URL http://localhost:8083/bills

Parameters

Headers Body Variables

Body content type application/json Editor view Raw input

FORMAT JSON MINIFY JSON

```
{
  "productItems": null,
  "customer": null,
  "customerID": 2
}
```

200 OK 24.67 ms DETAILS

Ajouter un bill

Method POST Request URL http://localhost:8083/login

Parameters

Headers Body Variables

Body content type application/json Editor view Raw input

FORMAT JSON MINIFY JSON

```
{
  "username" : "user",
  "password" : "1234"
}
```

200 OK 168.62 ms DETAILS

Authentification avec le rôle user

POST http://localhost:8083/login

Response headers **12** Request headers **4** Redirects **0** Timings

Response headers **12** Request headers **4** Redirects **0** Timings

Response headers **12** Request headers **4** Redirects **0** Timings

Response headers **12** Request headers **4** Redirects **0** Timings

```
accept: application/json
content-type: application/json
cookie: JSESSIONID=6FB544C3D61233598E4ADCA9284412BC; JSESSIONID=1A5858065A7D50FA99820748FEE5E7B2
authori Bearer
zation: eyJhbGciOiJIUzUxMiJ9.eyJzdWIiOiJhZG1ibG1iIiwiaWF0IjE5MTYwODY2OTY1Nywicm9sZXMiOiJ0LTImF1dGhvcml0eSI6IkFETU0In0seyJhdXRob3JpdHkiOiJVV0VSInldfQ.Ufn3t0REZP5mR3bT8V6PLC-Fuj99HlRXCsv56RQ_UlAA-heGoVZPX9QK2ZRZq9aDdGM34LXYLsujya05kCEf7Q
```

Code d'authentification

Conclusion :

Ce TP est très important pour moi car j'ai utilisé le Framework Spring (Spring Cloud Gateway, Spring Security et Spring Boot) et Eureka Server, aussi la différence entre la configuration Statique qui basée sur le fichier application.yml et la configuration dynamique des routes avec Eureka.cela m'a donné une idée sur les micro services et Spring Cloud, aussi tester et améliorer mes connaissances dans le Framework Spring.