ECOLE NORMALE SUPÉRIEURE DE L'ENSEIGNEMENT
TECHNIQUE DE MOHAMMEDIA
UNIVERSITÉ HASSAN II DE CASABLANCA

Département Mathématique Informatique

Filière : Master-SDIA2

Modèle : Systèmes parallèle et Distribués

Rapport :

Examen Systèmes Distribués

Réalisé par :

- **Abdelkarim AGOUJIL**

Année Universitaire : 2022-2023

1. Introduction

On souhaite créer une application basée sur une architecture micro-service qui permet de gérer des réservations

concernant des ressources. Chaque réservation concerne une seule ressource. Une ressource est définie par son

id, son nom, son type (MATERIEL_INF0, MATERIEL_AUDIO_VUSUEL). Une réservation est définie par son id, son

nom, son contexte, , sa date, sa durée. Chaque réservation est effectuée par une personne. Une personne est

définie par son id, son nom, son email et sa fonction.

L'application doit permettre de gérer les ressources et les réservations. Pour faire plus simple, cette application

se composera de deux micro-services fonctionnels :

• Un Micro-service qui permet de gérer des « Resources-Service ».

• Un Micro-service qui permet de gérer les réservations effectuées par des personnes.

Les micro-services technique à mettre en place sont :

• Le service Gateway basé sur Spring cloud Gateway

• Le service Discovery base sur Eureka Server ou Consul Discovery (au choix)

• Le service de configuration basé sur Spring cloud config ou Consul Config (au choix)

Pour l'application, nous avons besoin de développer une frontend web, basé sur Angular Framework.

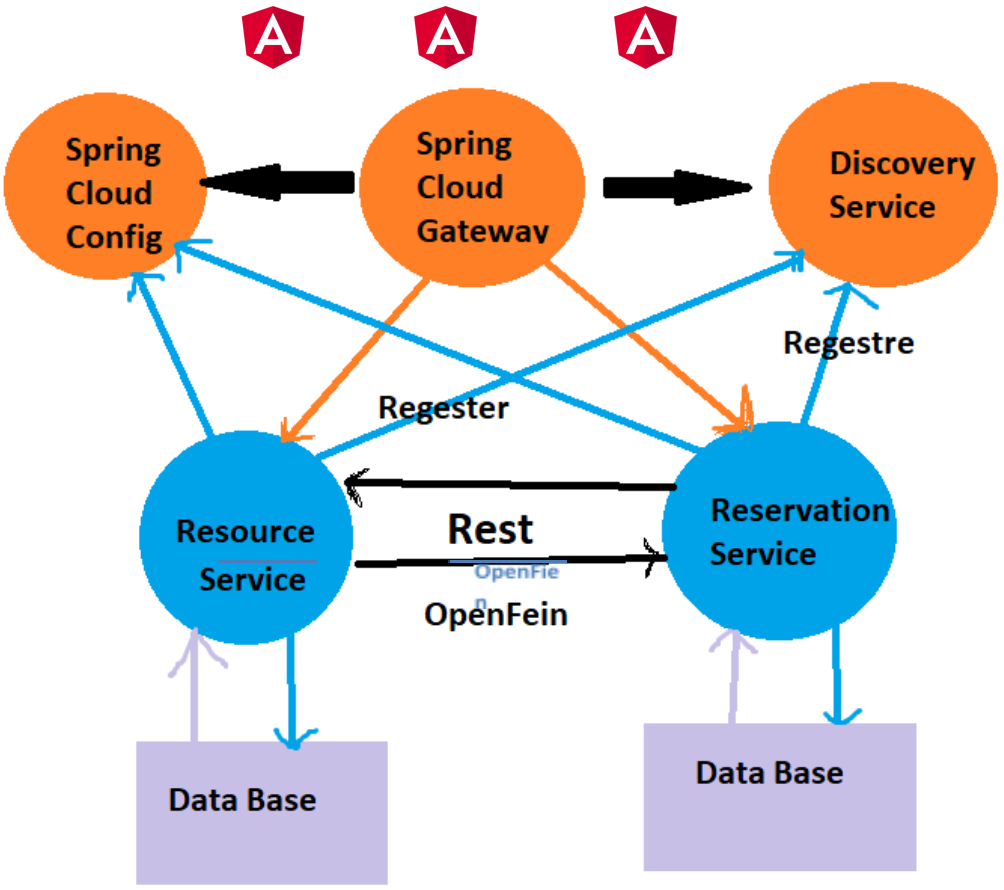
La sécurité de l'application est basée sur Oauth2 et OIDC avec Keycloak comme Provider

Pour les micro-services, il faut générer la documentation des web services Restfull en utilisant la spécification

OpenAPIDoc (Swagger). Prévoir aussi des circuit breakers basés sur Resilience4J comme solution de fault

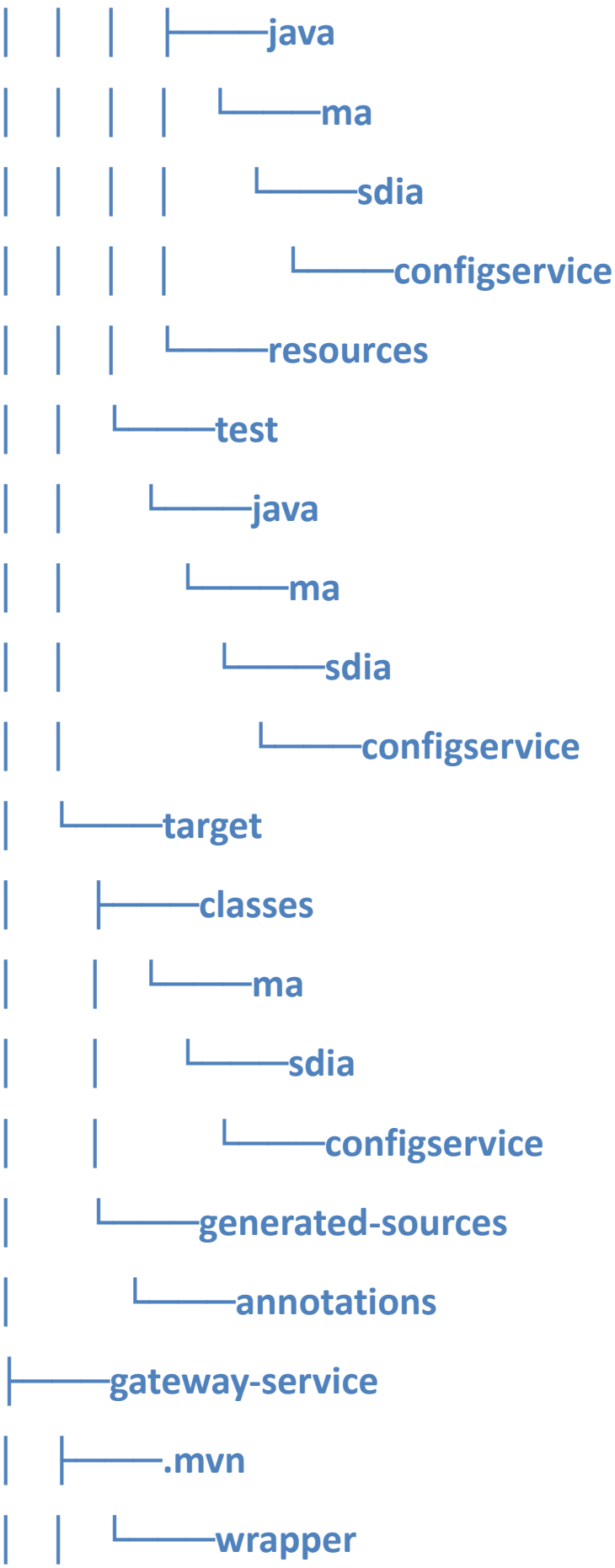
tolerence

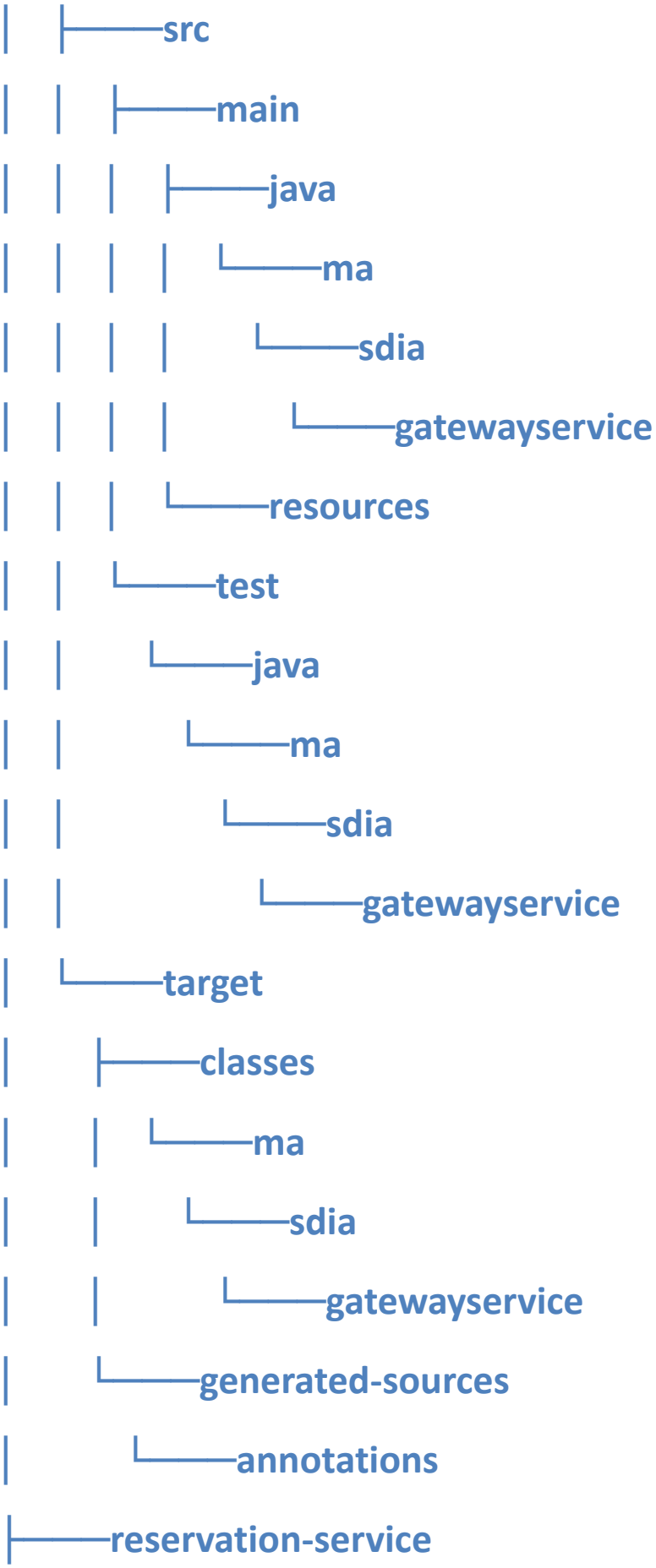
2. Architecture technique

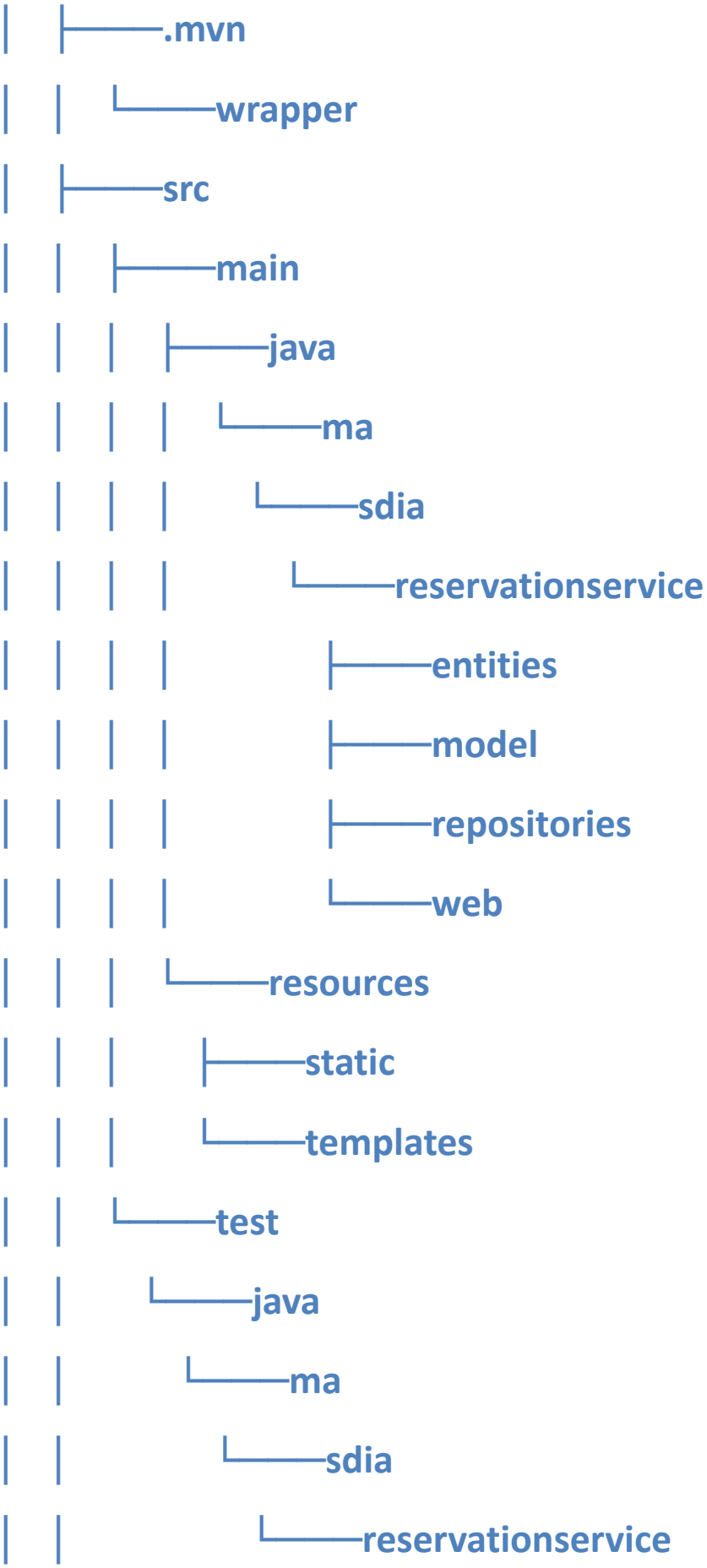


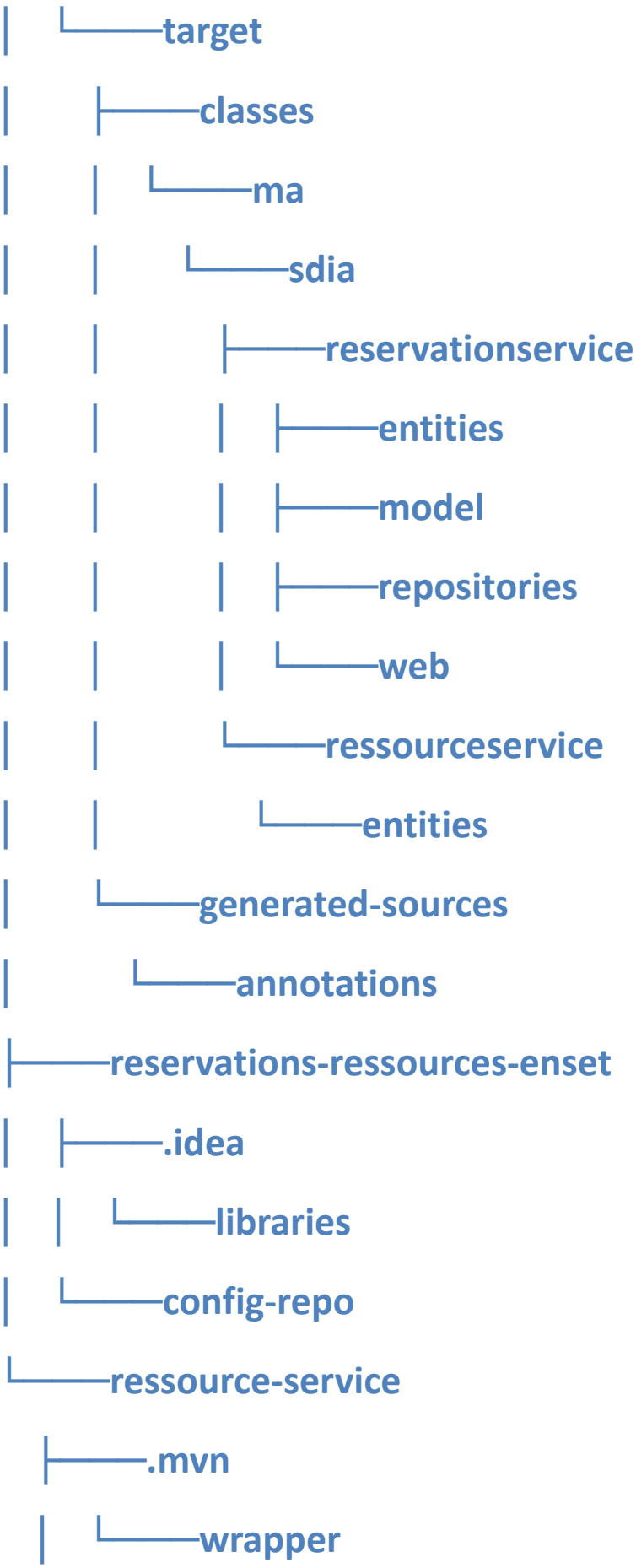
C:.

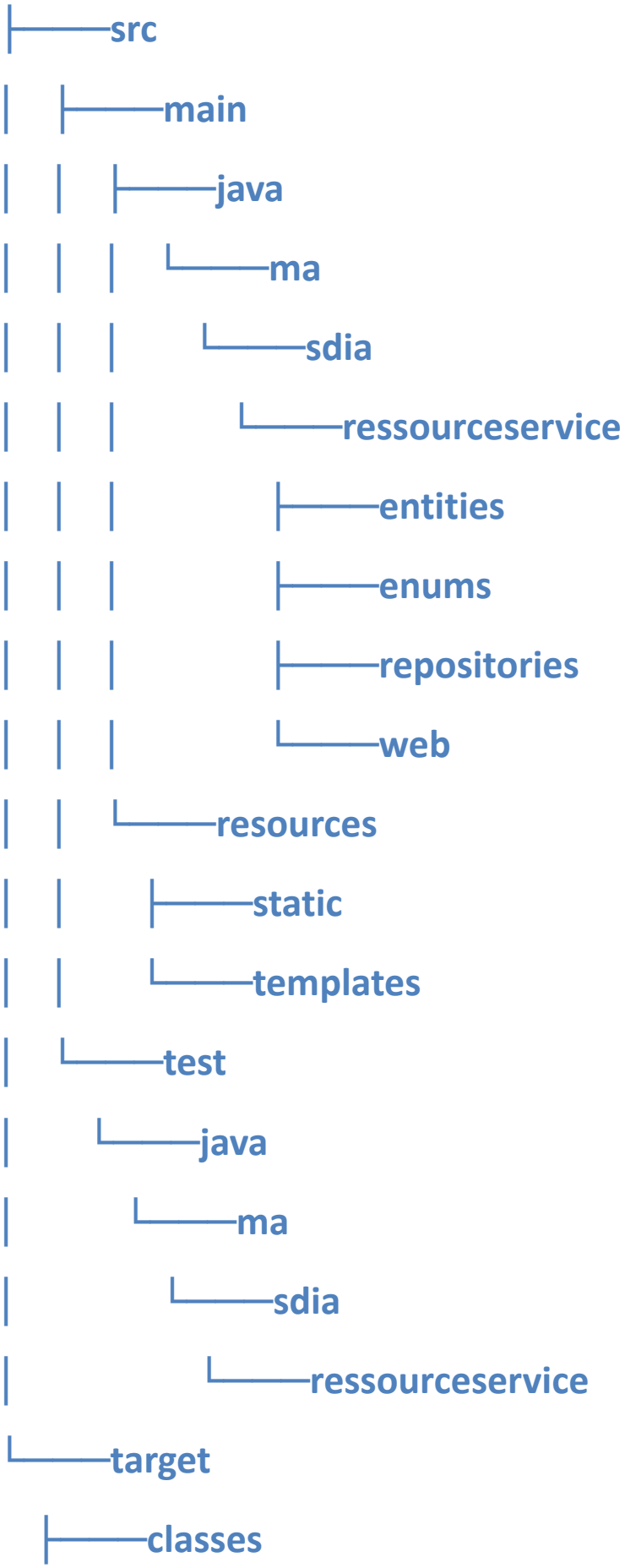
```
├───config-service
│   ├───.mvn
│   │   └───wrapper
│   ├───src
│   │   ├───main
```

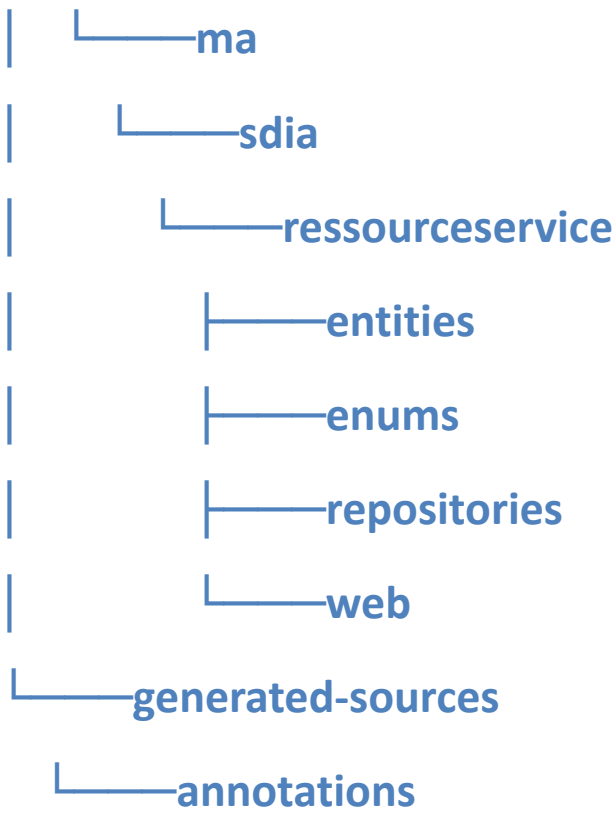

java
ma
sdia
configservice
resources
test
java
ma
sdia
configservice
target
classes
ma
sdia
configservice
generated-sources
annotations
gateway-service
.mvn
wrapper


src
main
java
ma
sdia
gatewayservice
resources
test
java
ma
sdia
gatewayservice
target
classes
ma
sdia
gatewayservice
generated-sources
annotations
reservation-service


.mvn
wrapper
src
main
java
ma
sdia
reservationservice
entities
model
repositories
web
resources
static
templates
test
java
ma
sdia
reservationservice


target
classes
ma
sdia
reservationservice
entities
model
repositories
web
ressourceservice
entities
generated-sources
annotations
reservations-ressources-enset
.idea
libraries
config-repo
ressource-service
.mvn
wrapper

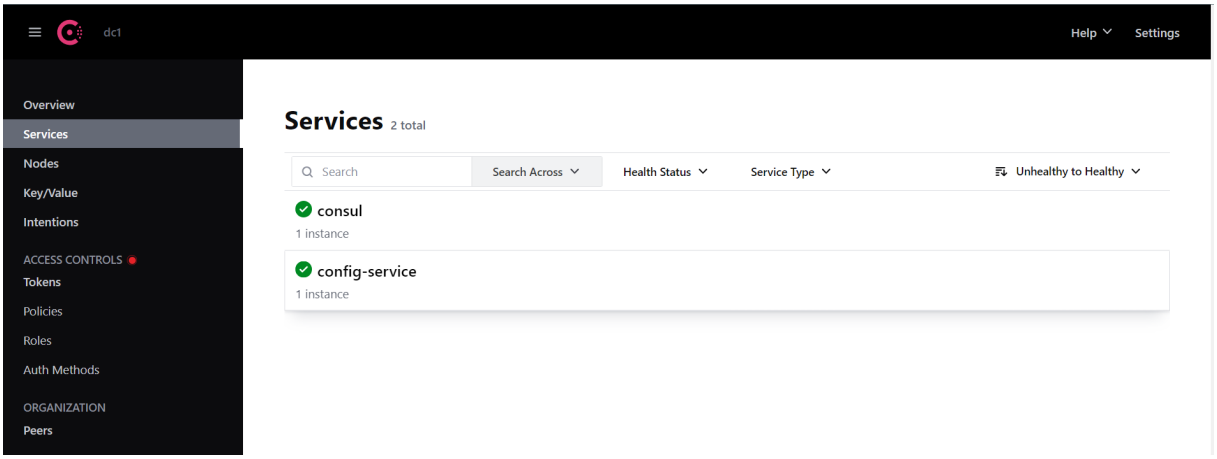

src
main
java
ma
sdia
ressourceservice
entities
enums
repositories
web
resources
static
templates
test
java
ma
sdia
ressourceservice
target
classes

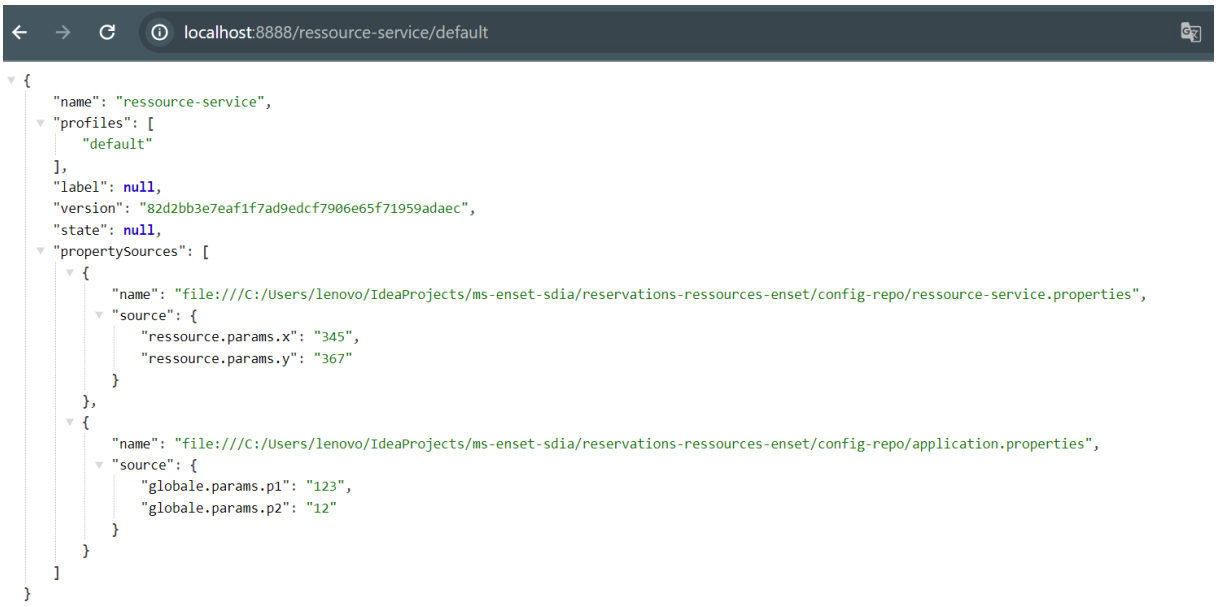


3. Développement du service config :

1) application.properties:

```
server.port=8888
spring.application.name=config-service
spring.cloud.config.server.git.uri=file:///C:/Users/lenovo/IdeaProjects/ms-enset-sdia/reservations-ressources-enset/config-repo
```





4. Développement du micro-service Ressource:

1) application.properties:

```
server.port=8081
spring.application.name=ressource-service
spring.config.import=optional:configserver:http://loc
alhost:8888
```

2) entites:

A. Resource:

```
package ma.sdia.ressourceservice.entities;


import jakarta.persistence.Entity;
import jakarta.persistence.GeneratedValue;
import jakarta.persistence.GenerationType;
import jakarta.persistence.Id;
import lombok.*;
import ma.sdia.ressourceservice.enums.ResourceType;
@Entity
@Getter @Setter @ToString @NoArgsConstructor
@AllArgsConstructor @Builder
public class Ressource {
   @Id
   @GeneratedValue(strategy =
GenerationType.IDENTITY)
   private Long id;
   private String name;
   private ResourceType type;


}
```

2. enum:

A. ResourceType:

```
package ma.sdia.ressourceservice.enums;


public enum ResourceType {
   MATERIEL_INFO, MATERIEL_AUDIO_VISUEL
```

```
}
```

3. repositories:
    B. ResourceType:

```
package ma.sdia.ressourceservice.repositories;


import ma.sdia.ressourceservice.entities.Ressource;
import
org.springframework.data.jpa.repository.JpaRepository
;
@RepositoryRestResource
public interface ResourceRepository extends
JpaRepository<Ressource,Long> {
}
```

4. web:
    C. ResourceRestController:

```
package ma.sdia.ressourceservice.web;


import ma.sdia.ressourceservice.entities.Ressource;
import
ma.sdia.ressourceservice.repositories.ResourceReposit
ory;
import
org.springframework.web.bind.annotation.GetMapping;
import
org.springframework.web.bind.annotation.PathVariable;
import
org.springframework.web.bind.annotation.RestControlle
r;


import java.util.List;


@RestController
public class RessourceRestController {
   ResourceRepository resourceRepository;
   public RessourceRestController(ResourceRepository
```

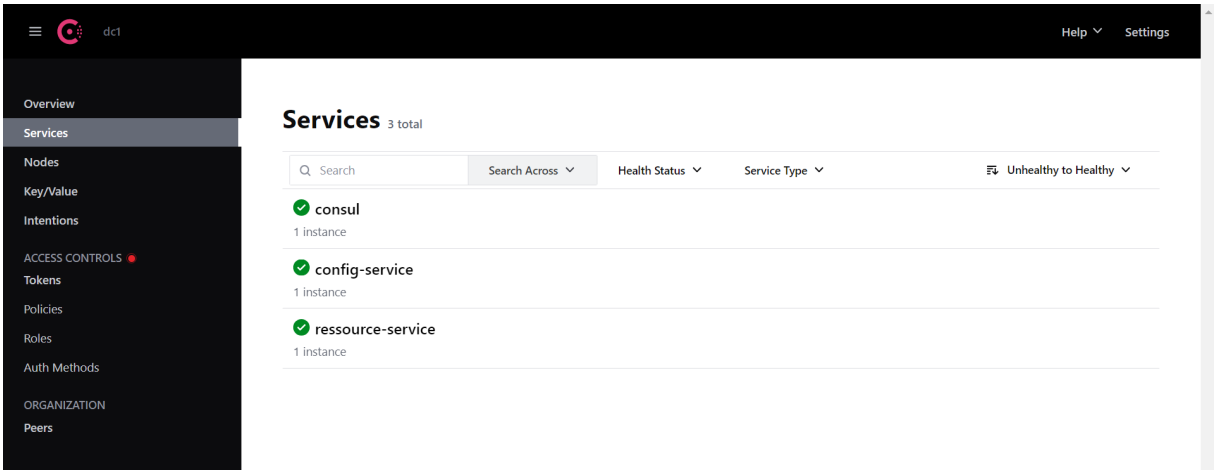
```
resourceRepository) {
        this.resourceRepository = resourceRepository;
    }
    @GetMapping("/ressources")
    public List<Ressource> ressourceList(){
        return resourceRepository.findAll();
    }
    @GetMapping("/ressources/{id}")
    public Ressource resourceById(@PathVariable Long
id){
        return resourceRepository.findById(id).get();
    }

}
```

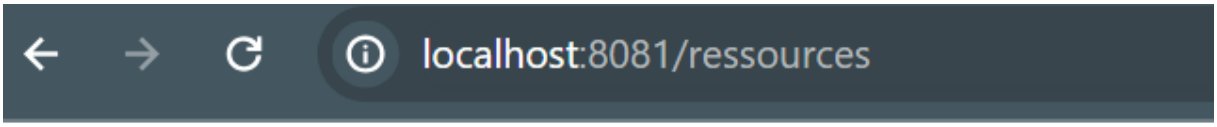
5. application:

D. RessourceServiceApplication:

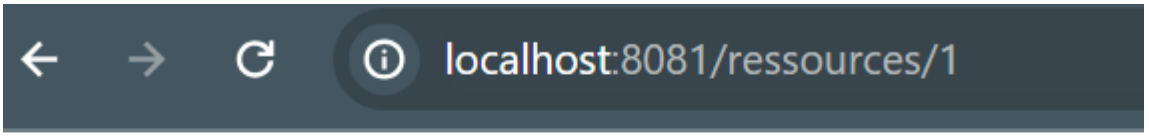
```
package ma.sdia.ressourceservice;


import ma.sdia.ressourceservice.entities.Ressource;
import ma.sdia.ressourceservice.enums.ResourceType;
import
ma.sdia.ressourceservice.repositories.ResourceReposit
ory;
import org.springframework.boot.CommandLineRunner;
import org.springframework.boot.SpringApplication;
import
org.springframework.boot.autoconfigure.SpringBootAppl
ication;
import org.springframework.context.annotation.Bean;


import java.util.List;


@SpringBootApplication
public class RessourceServiceApplication {
```

```
  public static void main(String[] args) {

SpringApplication.run(RessourceServiceApplication.cla
ss, args);
  }



  @Bean
  CommandLineRunner
commandLineRunner(ResourceRepository
resourceRepository){
     return args -> {

        List<Ressource> customerList=List.of(
              Ressource.builder()
                    .name("Resource1")
                    .type(ResourceType.MATERIEL_INFO)
                    .build(),
              Ressource.builder()
                    .name("Resource2")

.type(ResourceType.MATERIEL_AUDIO_VISUEL)
                    .build()
        );
        resourceRepository.saveAll(customerList);
     };
  }

}
```





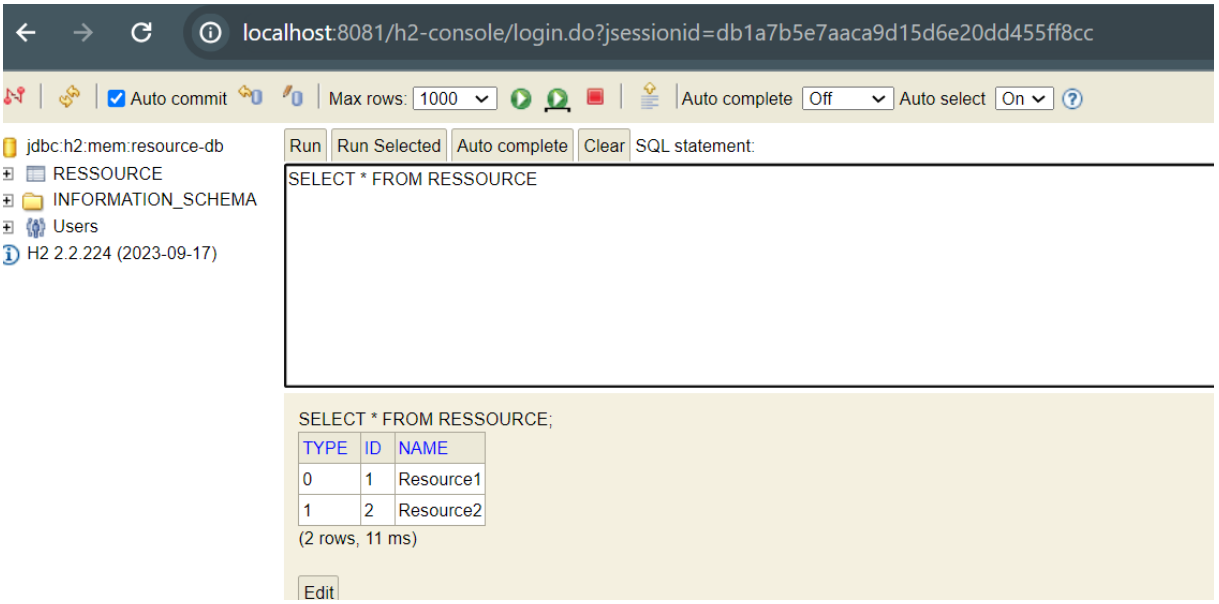

```
[
  {
    "id": 1,
    "name": "Resource1",
    "type": "MATERIEL_INFO"
  },
  {
    "id": 2,
    "name": "Resource2",
    "type": "MATERIEL_AUDIO_VISUEL"
  }
]
```




```
{
    "id": 1,
    "name": "Resource1",
    "type": "MATERIEL_INFO"
}
```



```
management.endPoints.web.exposure.include=*
spring.datasource.url=jdbc:h2:mem:resource-db
spring.h2.console.enabled=true
```



6. Développement du service gateway :

3) Application:

```
package ma.sdia.gatewayservice;


import org.springframework.boot.SpringApplication;
import
org.springframework.boot.autoconfigure.SpringBootAppl
ication;
import
org.springframework.cloud.client.discovery.ReactiveDi
scoveryClient;
import
org.springframework.cloud.gateway.discovery.Discovery
ClientRouteDefinitionLocator;
import
org.springframework.cloud.gateway.discovery.Discovery
LocatorProperties;
import org.springframework.context.annotation.Bean;


@SpringBootApplication
public class GatewayServiceApplication {


  public static void main(String[] args) {
```

```
SpringApplication.run(GatewayServiceApplication.class
, args);
  }


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




}
```

2) application.properties:

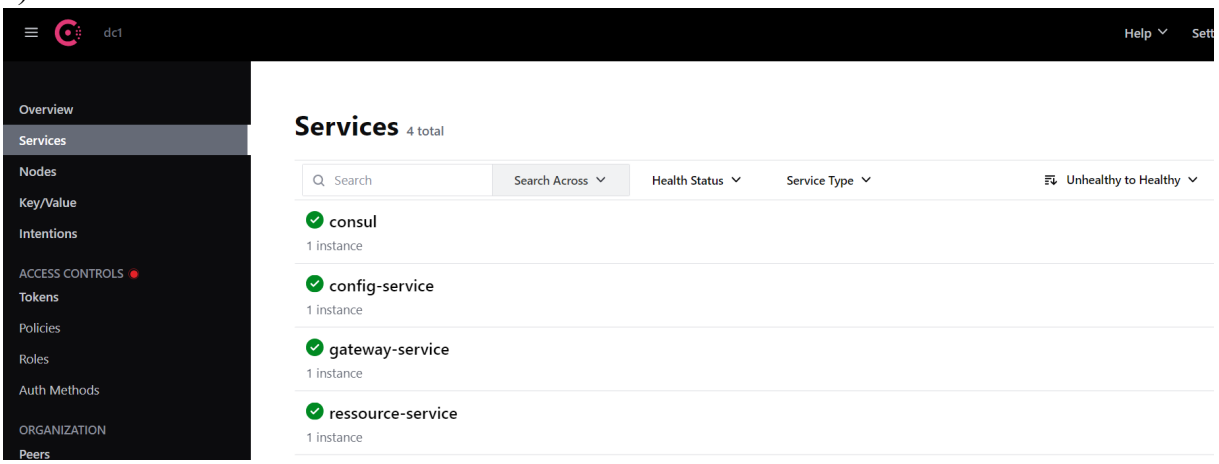
```
server.port=9999
spring.application.name=gateway-service
spring.config.import=optional:configserver:http://loc
alhost:8888
```

3) app.yml:

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

```
            - Path=/reservations/**
 application:
   name: gateway-service
server:
 port: 8888
```

4)Test:



7. Développement du micro-service Reservation :

4) Entités JPA et Interface JpaRepository basées sur Spring data

a) Entité Propriétaire

```
package ma.sdia.reservationservice.entities;

import jakarta.persistence.*;
import lombok.*;
import ma.sdia.ressourceservice.entities.Ressource;

import java.time.LocalDateTime;
import java.util.Date;

@Entity
@Getter @Setter @AllArgsConstructor @NoArgsConstructor
@Builder
public class Reservation {
   @Id
```

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

    private String name;

    private String context;

    private Date date;
    private int duration;
    @Transient
    private Ressource ressource;
    private Long ressourceId;


}
```

b) model:

```
package ma.sdia.ressourceservice.entities;

import jakarta.persistence.Entity;
import jakarta.persistence.GeneratedValue;
import jakarta.persistence.GenerationType;
import jakarta.persistence.Id;
import lombok.*;
import
org.springframework.data.rest.core.mapping.ResourceTyp
e;

@Getter @Setter @ToString @NoArgsConstructor
@AllArgsConstructor @Builder
public class Ressource {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
```

```
    private String name;
    private ResourceType type;

}
```

c) RessourceRestClient:

```
package ma.sdia.reservationservice.web;


import
io.github.resilience4j.circuitbreaker.annotation.Circu
itBreaker;
import ma.sdia.ressourceservice.entities.Ressource;
import
org.springframework.cloud.openfeign.FeignClient;
import
org.springframework.data.rest.core.mapping.ResourceTyp
e;
import
org.springframework.web.bind.annotation.GetMapping;
import
org.springframework.web.bind.annotation.PathVariable;


import java.util.List;


@FeignClient("RESSOURCE-SERVICE")
public interface ResourceRestClient {
    @GetMapping("/ressources")
    @CircuitBreaker(name = "ressourceService",
fallbackMethod = "getAllRessources")
    public List<Ressource> allRessource();
    @GetMapping("/ressources/{id}")
    @CircuitBreaker(name = "ressource-service",
fallbackMethod = "getDefaultRessource")
    public Ressource findRessourceById(@PathVariable
Long id);
```

```
    default Ressource getDefaultCustomer(Long id,
Exception exception){
        Ressource ressource=new Ressource();
        ressource.setId(id);
        ressource.setName("Ressource x");
        return ressource ;
    }
    default List<Ressource> getAllRessources(Exception
exception){
        return List.of();
    }
}
```

d) Interface Propriétaire Repository

```
package ma.sdia.reservationservice.repositories;

import ma.sdia.reservationservice.entities.Reservation;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.data.rest.core.annotation.RepositoryRestResource;

@RepositoryRestResource
public interface ReservationRepository extends
JpaRepository<Reservation,Long> {
}


}
```

e) ReservationServiceApplication

```
package ma.sdia.reservationservice;



import
ma.sdia.reservationservice.entities.Reservation;

import
ma.sdia.reservationservice.repositories.ReservationRe
pository;

import
ma.sdia.reservationservice.web.ResourceRestClient;
```

```
import org.springframework.boot.CommandLineRunner;

import org.springframework.boot.SpringApplication;

import
org.springframework.boot.autoconfigure.SpringBootAppl
ication;

import
org.springframework.cloud.openfeign.EnableFeignClient
s;

import org.springframework.context.annotation.Bean;

import java.util.Date;

import java.util.Random;


@SpringBootApplication

@EnableFeignClients

public class ReservationServiceApplication {


  public static void main(String[] args) {


SpringApplication.run(ReservationServiceApplication.c
lass, args);

  }


  @Bean

  CommandLineRunner
commandLineRunner(ReservationRepository
reservationRepository, ResourceRestClient
resourceRestClient){

     return args -> {
```

```
resourceRestClient.allRessource().forEach(r->{
            Reservation reservation1 =
Reservation.builder()
                    .name("reservation1")
                    .context("context1")
                    .date(new Date())
                    .duration(new
Random().nextInt(2,30))
                    .ressourceId(r.getId())
                    .build();
            Reservation reservation2 =
Reservation.builder()
                    .name("reservation1")
                    .context("context1")
                    .date(new Date())
                    .duration(new
Random().nextInt(2,30))
                    .ressourceId(r.getId())
                    .build();
            reservationRepository.save(reservation1);
            reservationRepository.save(reservation2);
        });



    };
```

```
}
```