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° 2
# Systèmes Distribués

**Filière :**

**« Génie du Logiciel et des Systèmes Informatiques Distribués »**

**GLSID**

## Contrôle
## Architectures Micro-services

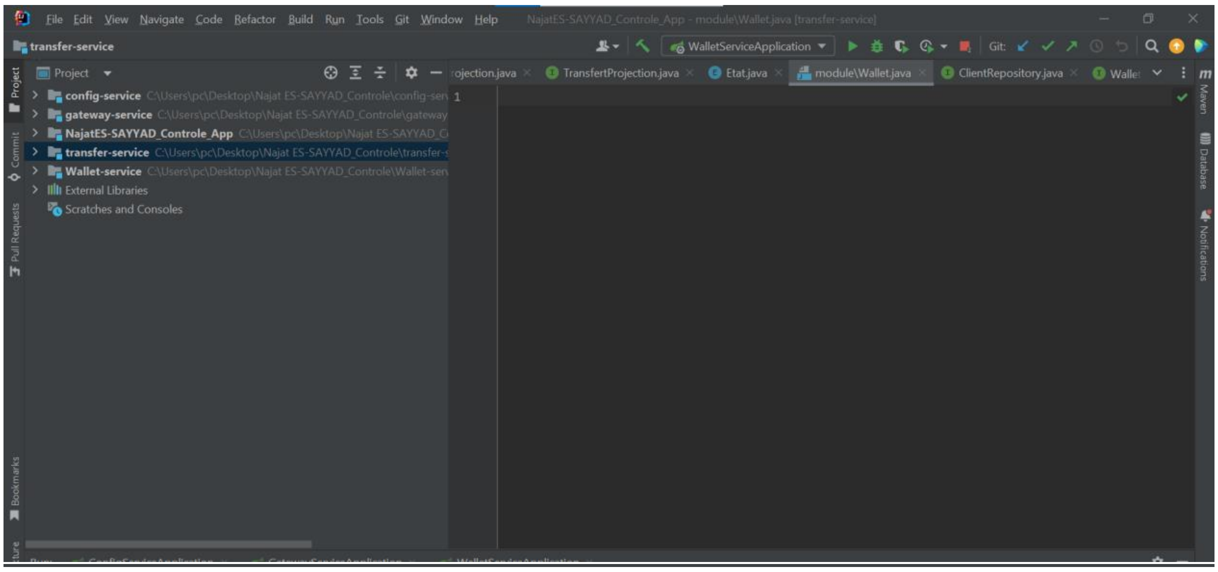
Réalisé par :

Najat ES-SAYYAD

**Année Universitaire : 2023-2024**

**1. Créer un Empty Project incluant les micro-services suivants : wallet-service, transfer-service, gateway-service, discovery-service et config-service**

## <u>Config service :</u>


File Edit View Navigate Code Refactor Build Run Tools Git Window Help
NajatES-SAYYAD_Controle_App - module\Wallet.java [transfer-service]
transfer-service
WalletServiceApplication
Project
config-service C:\Users\pc\Desktop\Najat ES-SAYYAD_Controle\config-ser
gateway-service C:\Users\pc\Desktop\Najat ES-SAYYAD_Controle\gateway
NajatES-SAYYAD_Controle_App C:\Users\pc\Desktop\Najat ES-SAYYAD_C
transfer-service C:\Users\pc\Desktop\Najat ES-SAYYAD_Controle\transfer-s
Wallet-service C:\Users\pc\Desktop\Najat ES-SAYYAD_Controle\Wallet-ser
External Libraries
Scratches and Consoles
rojection.java
TransfertProjection.java
Etat.java
module\Wallet.java
ClientRepository.java
Walle

ConfigServiceApplication :

```java
package org.sid.configservice;

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

@SpringBootApplication
@EnableConfigServer
@EnableDiscoveryClient
public class ConfigServiceApplication {

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

}
```

application.properties :

```properties
server.port=8888
spring.application.name=config-service
spring.cloud.config.server.git.uri=file:///C:/Users/pc/Desktop/Najat ES-
SAYYAD_Controle/NajatES-SAYYAD_Controle_App/config-reposit
```

## <u>Gateway service :</u>

GatewayServiceApplication :

```java
package org.sid.gatewayservice;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.cloud.client.discovery.ReactiveDiscoveryClient;
import
org.springframework.cloud.gateway.discovery.DiscoveryClientRouteDefinitionL
ocator;
import
org.springframework.cloud.gateway.discovery.DiscoveryLocatorProperties;
import org.springframework.context.annotation.Bean;

@SpringBootApplication
public class GatewayServiceApplication {

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

    @Bean
    DiscoveryClientRouteDefinitionLocator
dynamicRoutes(ReactiveDiscoveryClient rdc,

DiscoveryLocatorProperties dlp){
        return new DiscoveryClientRouteDefinitionLocator(rdc,dlp);
    }

}
```

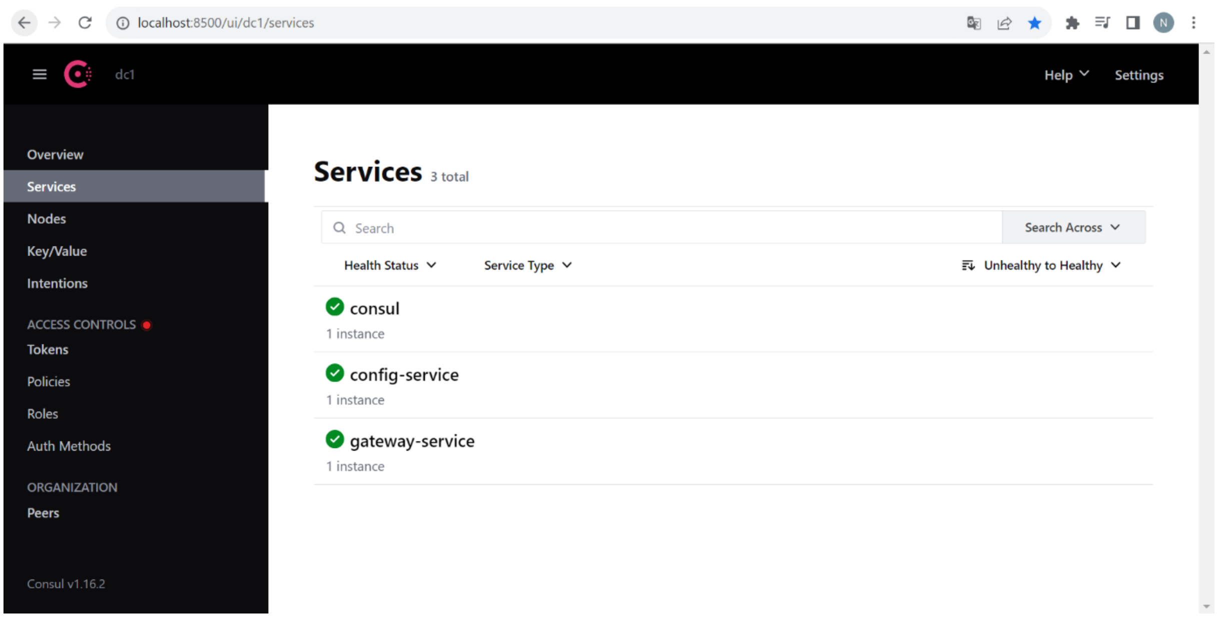
application.properties :

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

application.yml :

```yaml
spring:
  cloud:
    gateway:
      globalcors:
        corsConfigurations:
          '[/**]':
            allowedOrigins: "http://localhost:4200"
            allowedHeaders: "*"
            allowedMethods:
              - GET
              - POST
              - PUT
              - DELETE
```

## 2. Développer et tester les micro-services discovery-service et gateway-service et config-service



Services 3 total

consul
1 instance

config-service
1 instance

gateway-service
1 instance

## 3. Développer et tester le micro-service wallet-service

### Wallet-service

### Etities :

### EntitéClient ;

```
package org.sid.walletservice.entities;

import jakarta.persistence.*;
import lombok.AllArgsConstructor;
import lombok.Builder;
import lombok.Data;
import lombok.NoArgsConstructor;

import java.util.Collection;
import java.util.List;

@Data
@Entity @AllArgsConstructor @NoArgsConstructor @Builder
public class Client {
    @Id @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    private String name;
    private String email;
    @OneToMany(mappedBy = "client")
    private Collection<Wallet> wallets;


}
```

### entité Wallet :

```
package org.sid.walletservice.entities;

import com.fasterxml.jackson.annotation.JsonProperty;
import jakarta.persistence.*;
import lombok.AllArgsConstructor;
import lombok.Builder;
import lombok.Data;
```

```java
import lombok.NoArgsConstructor;

import java.util.Date;

@Data
@Entity @NoArgsConstructor @AllArgsConstructor @Builder
public class Wallet {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    private  Double solde;
    private Date dateCreated;
    private Double devise ;
    @ManyToOne
    private Client client;
}
```

**WalletProjection :**

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

import com.fasterxml.jackson.annotation.JsonProperty;
import jakarta.persistence.GeneratedValue;
import jakarta.persistence.GenerationType;
import jakarta.persistence.Id;
import jakarta.persistence.ManyToOne;
import org.springframework.data.rest.core.config.Projection;

import java.util.Date;

@Projection(name="fullWallet",types = Wallet.class)
public interface WalletProjection {

    Long getId();
    Double getSolde();
    Date getDateCeated();
    Double getDevise();
    Client getClient();


}
```

**Repository :**

**ClientRepository :**

```java
package org.sid.walletservice.Repository;

import org.sid.walletservice.entities.Client;
import org.springframework.data.jpa.repository.JpaRepository;
import
org.springframework.data.rest.core.annotation.RepositoryRestResource;

@RepositoryRestResource
public interface ClientRepository extends JpaRepository<Client,Long> {

}
```

**WalletRepository :**

```java
package org.sid.walletservice.Repository;

import org.sid.walletservice.entities.Wallet;
import org.springframework.data.jpa.repository.JpaRepository;
import
org.springframework.data.rest.core.annotation.RepositoryRestResource;

@RepositoryRestResource
public interface WalletRepository extends JpaRepository<Wallet,Long> {
}
```

### Application test :

```java
package org.sid.walletservice;

import jakarta.validation.constraints.Null;
import org.sid.walletservice.Repository.ClientRepository;
import org.sid.walletservice.Repository.WalletRepository;
import org.sid.walletservice.entities.Client;
import org.sid.walletservice.entities.Wallet;
import org.springframework.boot.CommandLineRunner;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.context.annotation.Bean;

import java.util.Date;
import java.util.List;

@SpringBootApplication
public class WalletServiceApplication {

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

   @Bean
   CommandLineRunner start(ClientRepository
clientRepository,WalletRepository walletRepository){
      return args -> {

         Client client1=new Client(null,"najat","najat@gmail.com", null);
         Client client2=new Client(null,"Salma","salma@gmail.com",null);
         Client client3=new Client(null,"ahmed","ahmed@gmail.com",null);


         Wallet wallet1=new Wallet(null,129.23,new Date(),12983.42,null);
         Wallet wallet2=new Wallet(null,23784.2,new Date(),1291.32,null);
         Wallet wallet3=new Wallet(null,29482.2,new Date(),939.3,null);

         walletRepository.save(wallet1);
         walletRepository.save(wallet2);
         walletRepository.save(wallet3);



         clientRepository.save(client1);
         clientRepository.save(client2);
         clientRepository.save(client3);
```

```
            client1.setWallets(walletRepository.findAll());
            client2.setWallets(walletRepository.findAll());

            walletRepository.save(wallet1);
            walletRepository.save(wallet2);
            walletRepository.save(wallet3);

            clientRepository.findAll().forEach(c->
                    System.out.println(c.toString()));

            walletRepository.findAll().forEach(w->
                    System.out.println(w.toString()));

        };

    }


}
```

**application.properties :**

```
server.port=8081
spring.application.name=wallet-service
spring.config.import=optional:configserver:http://localhost:8888
```

**4. Développer et tester le micro-service transfer-service**

**Etities :**

**Transfer :**

```
package org.sid.transferservice.entities;

import jakarta.persistence.*;
import lombok.AllArgsConstructor;
import lombok.Builder;
import lombok.Data;
import lombok.NoArgsConstructor;
import org.sid.transferservice.Enums.Etat;
import org.sid.transferservice.module.Wallet;

import java.util.Date;

@Data
@Entity @NoArgsConstructor @AllArgsConstructor @Builder
public class Transfer {
    @Id @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    private Date date;
    private Long walletId;
    @Transient
    private Wallet walletSource;
    @Transient
    private Wallet walletDestination;
    private Double montant ;
    private Etat etat ;

}
```

**TransferProjection :**

```
package org.sid.transferservice.entities;

import jakarta.persistence.Transient;
import org.sid.transferservice.Enums.Etat;
import org.sid.transferservice.module.Wallet;
import org.springframework.data.rest.core.config.Projection;

import java.util.Date;

@Projection(name="fullTransfert",types = Transfer.class)
public interface TransfertProjection {
    Long getId();
    Date getDate();
    Long getWalletId();
    Double getMontant();
    Etat getEtat();

}
```

**Enums :**

**Etat :**

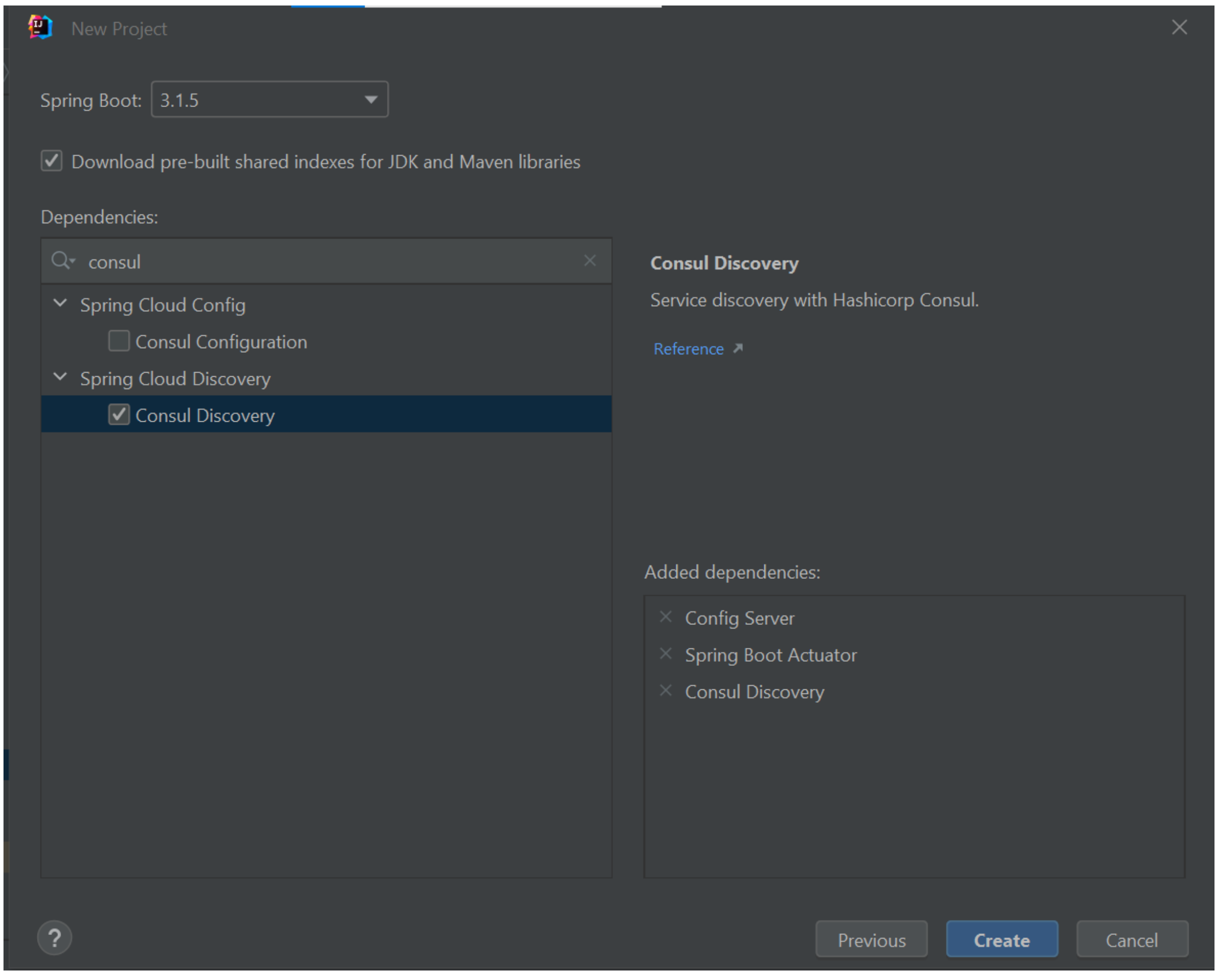
```
package org.sid.transferservice.Enums;

public enum Etat {
    PENDIND, VALIDATED, REJECTED

}
```

**Module :**

```
package org.sid.transferservice.module;

import jakarta.persistence.*;
import lombok.AllArgsConstructor;
import lombok.Builder;
import lombok.Data;
import lombok.NoArgsConstructor;

import java.util.Date;

public class Wallet {
        public Long id;
        public   Double solde;
        public Date dateCreated;
        public Double devise ;

    }

}
```

## 5. Développer un simple frontend web pour l’application

## 6. Proposer une solution pour sécuriser l’application



Spring Boot: 3.1.5

☑ Download pre-built shared indexes for JDK and Maven libraries

Dependencies:

consul

Spring Cloud Config
- ☐ Consul Configuration

Spring Cloud Discovery
- ☑ Consul Discovery

**Consul Discovery**

Service discovery with Hashicorp Consul.

Reference ↗

Added dependencies:
- Config Server
- Spring Boot Actuator
- Consul Discovery