SUJET DE RATTRAPAGE WEBSERCVICES

CODE SOURCES

1. pom.xml - Dépendances Maven

xml

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>

    <groupId>com.cmu.bridge</groupId>
    <artifactId>cmu-bridge-rest</artifactId>
    <version>1.0-SNAPSHOT</version>
    <packaging>jar</packaging>

    <properties>
        <java.version>11</java.version>
    </properties>

    <dependencies>
        <!-- Spring Boot Web for REST API -->
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-web</artifactId>
        </dependency>

        <!-- Spring Boot DevTools for easier development -->
```

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-devtools</artifactId>
    <scope>runtime</scope>
</dependency>

<!-- Spring Web Services for SOAP -->
<dependency>
    <groupId>org.springframework.ws</groupId>
    <artifactId>spring-webservices</artifactId>
    <version>4.0.0</version>
</dependency>

<!-- H2 Database (in-memory) for simulating database -->
<dependency>
    <groupId>com.h2database</groupId>
    <artifactId>h2</artifactId>
    <scope>runtime</scope>
</dependency>

<!-- Spring Data JPA for database access -->
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-data-jpa</artifactId>
</dependency>

<!-- Spring Boot Starter Test (for unit testing) -->
<dependency>
```

```
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-test</artifactId>
        <scope>test</scope>
    </dependency>
  </dependencies>

  <build>
    <plugins>
      <plugin>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-maven-plugin</artifactId>
      </plugin>
    </plugins>
  </build>
</project>
```

---

2. application.properties - Configuration Spring Boot

properties

```
# Configuration de la base de données H2
spring.datasource.url=jdbc:h2:mem:testdb
spring.datasource.driverClassName=org.h2.Driver
spring.datasource.username=sa
spring.datasource.password=password
spring.jpa.database-platform=org.hibernate.dialect.H2Dialect
spring.jpa.hibernate.ddl-auto=update
spring.h2.console.enabled=true

# Port de l'application REST
```

```
server.port=8080
```

---

3. Classe principale de l'application Spring Boot - Application.java

java

```
package com.cmu.bridge.application;

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

@SpringBootApplication
public class Application {
    public static void main(String[] args) {
        SpringApplication.run(Application.class, args);
    }
}
```

---

4. Entité Patient - Patient.java

java

```
package com.cmu.bridge.model;

import javax.persistence.Entity;
import javax.persistence.Id;

@Entity
public class Patient {

    @Id
```

```
    private String phoneNumber;
    private boolean eligible;

    public String getPhoneNumber() {
        return phoneNumber;
    }

    public void setPhoneNumber(String phoneNumber) {
        this.phoneNumber = phoneNumber;
    }

    public boolean isEligible() {
        return eligible;
    }

    public void setEligible(boolean eligible) {
        this.eligible = eligible;
    }
}
```

---

5. Repository Patient - PatientRepository.java

java

```
package com.cmu.bridge.repository;

import com.cmu.bridge.model.Patient;
import org.springframework.data.jpa.repository.JpaRepository;

public interface PatientRepository extends JpaRepository<Patient, String> {
```

```
    Patient findByPhoneNumber(String phoneNumber);
}
```

---

6. Service SOAP - SOAPService.java

java

```
package com.cmu.bridge.service;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import com.cmu.bridge.repository.PatientRepository;

@Service
public class SOAPService {

    @Autowired
    private PatientRepository patientRepository;

    public String checkEligibility(String phoneNumber) {
        // Vérifier dans la base de données (H2)
        Patient patient = patientRepository.findByPhoneNumber(phoneNumber);

        if (patient != null) {
            // Simulation de la réponse SOAP basée sur les données du patient
            return simulateSOAPResponse(patient);
        } else {
            return "<soapenv:Envelope
xmlns:soapenv=\"http://schemas.xmlsoap.org/soap/envelope/\"
xmlns:web=\"http://webservice.cmu.com/\">\n"  +
```

```
            "   <soapenv:Header/>\n" +
            "   <soapenv:Body>\n" +
            "      <web:CheckEligibilityResponse>\n" +
            "         <status>NonEligible</status>\n" +
            "         <message>Patient non trouvé.</message>\n" +
            "      </web:CheckEligibilityResponse>\n" +
            "   </soapenv:Body>\n" +
            "</soapenv:Envelope>";
    }
  }

  private String simulateSOAPResponse(Patient patient) {
    String status = patient.isEligible() ? "Eligible" : "NonEligible";
    String message = patient.isEligible() ?
                "Le patient avec le numéro " + patient.getPhoneNumber() + " est éligible." :
                "Le patient avec le numéro " + patient.getPhoneNumber() + " n'est pas
éligible.";

    return "<soapenv:Envelope
xmlns:soapenv=\"http://schemas.xmlsoap.org/soap/envelope/\"
xmlns:web=\"http://webservice.cmu.com/\">\n"  +
            "   <soapenv:Header/>\n" +
            "   <soapenv:Body>\n" +
            "      <web:CheckEligibilityResponse>\n" +
            "         <status>" + status + "</status>\n" +
            "         <message>" + message + "</message>\n" +
            "      </web:CheckEligibilityResponse>\n" +
            "   </soapenv:Body>\n" +
            "</soapenv:Envelope>";
```

```
    }
}
```

---

7. Contrôleur REST - PatientController.java

java

```
package com.cmu.bridge.controller;

import com.cmu.bridge.service.SOAPService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.*;

@RestController
@RequestMapping("/api/patient")
public class PatientController {

    @Autowired
    private SOAPService soapService;

    @GetMapping("/eligibility")
    public String checkEligibility(@RequestParam String phoneNumber) {
        try {
            // Appel du service SOAP pour vérifier l'éligibilité
            return soapService.checkEligibility(phoneNumber);
        } catch (Exception e) {
            return "Erreur lors de la vérification de l'éligibilité : " + e.getMessage();
        }
    }
}
```

---

8. Base de données H2 - Initialisation avec des données

J’ initialise la base de données H2 en créant un fichier de script dans src/main/resources/data.sql pour peupler la base avec quelques données de test.

sql

```
INSERT INTO patient (phone_number, eligible) VALUES ('0123456789', true);

INSERT INTO patient (phone_number, eligible) VALUES ('0987654321', false);
```

---

9. Démarrer l'application

- **Lancer l'application** : Dans votre terminal, à la racine du projet, exécutez la commande suivante pour démarrer l'application Spring Boot :

bash

CopierModifier

```
mvn spring-boot:run
```

- **Accéder à l'API REST** : Une fois l'application lancée, vous pouvez interroger l'API à l'endpoint http://localhost:8080/api/patient/eligibility?phoneNumber=0123456789.

10. Exemple de réponse JSON

Si un patient est éligible, la réponse sera :

json

CopierModifier

```
{
  "status": "Eligible",
  "message": "Le patient avec le numéro 0123456789 est éligible."
}
```

Si le patient n'est pas trouvé ou non éligible :

json

CopierModifier

```
{
```

```
  "status": "NonEligible",
  "message": "Le patient avec le numéro 0987654321 n'est pas éligible."
}
```