

# COMPTER RENDEZ TP1 SOA



# Sahar Mannai -3ING-IDL1

# NOTION DE BASE

## Notion du SOA:

SOA, Acronyme de Service Oriented Architecture, est traduit en français par Architecture Orientée Services. En fait , c'est un modèle de développement logiciel à base de composants applicatifs distribués et doté de fonctions de découverte, de contrôle d'accès, de mappage de données et de sécurité.

## Notion de service:

Défini comme un processus de production ayant un point de contact physique entre le client et le processus, et fournissant des biens intangibles ; à ce point de contact, la production et la consommation sont simultanées, un service revêt des spécificités qui posent problème au contrôle de gestion.

Ainsi, les attributs spécifiques des activités de services contraignent le contrôle de gestion à faire preuve d'adaptabilité pour être efficace et efficient.

## Notion de API:

API est l'acronyme d'**Application Programming Interface**, que l'on traduit en français par **interface de programmation applicative** ou **interface de programmation d'application**. L'API est une solution informatique qui permet à des applications de communiquer entre elles et de s'échanger mutuellement des services ou des données.

## Différence entre Rest et Soap:

SOAP est un protocole tandis que REST est un style d'architecture.

La différence majeure entre ces 2 éléments est le degré de liaison entre le client et le serveur. Un client développé avec le protocole SOAP ressemble à un logiciel d'ordinateur, car il est étroitement lié au serveur. Si une modification est effectuée d'un côté ou de l'autre, l'ensemble peut ne plus fonctionner. Il faut effectuer des mises à jour du client s'il y a des changements sur le serveur et vice-versa.

# PARTIE 1:

## Code :

- Classe Serveur.java :

```
serveur.java ×
1 package idl1;
2
3 import javax.xml.ws.Endpoint;
4
5 public class serveur {
6
7
8     static final String URL ="http://localhost:8082/";
9
10    public static void main(String[] args) {
11        // TODO Auto-generated method stub
12        System.out.println("running on"+URL);
13        Endpoint.publish(URL, new Usercontroller());
14    }
15
16 }
17
```

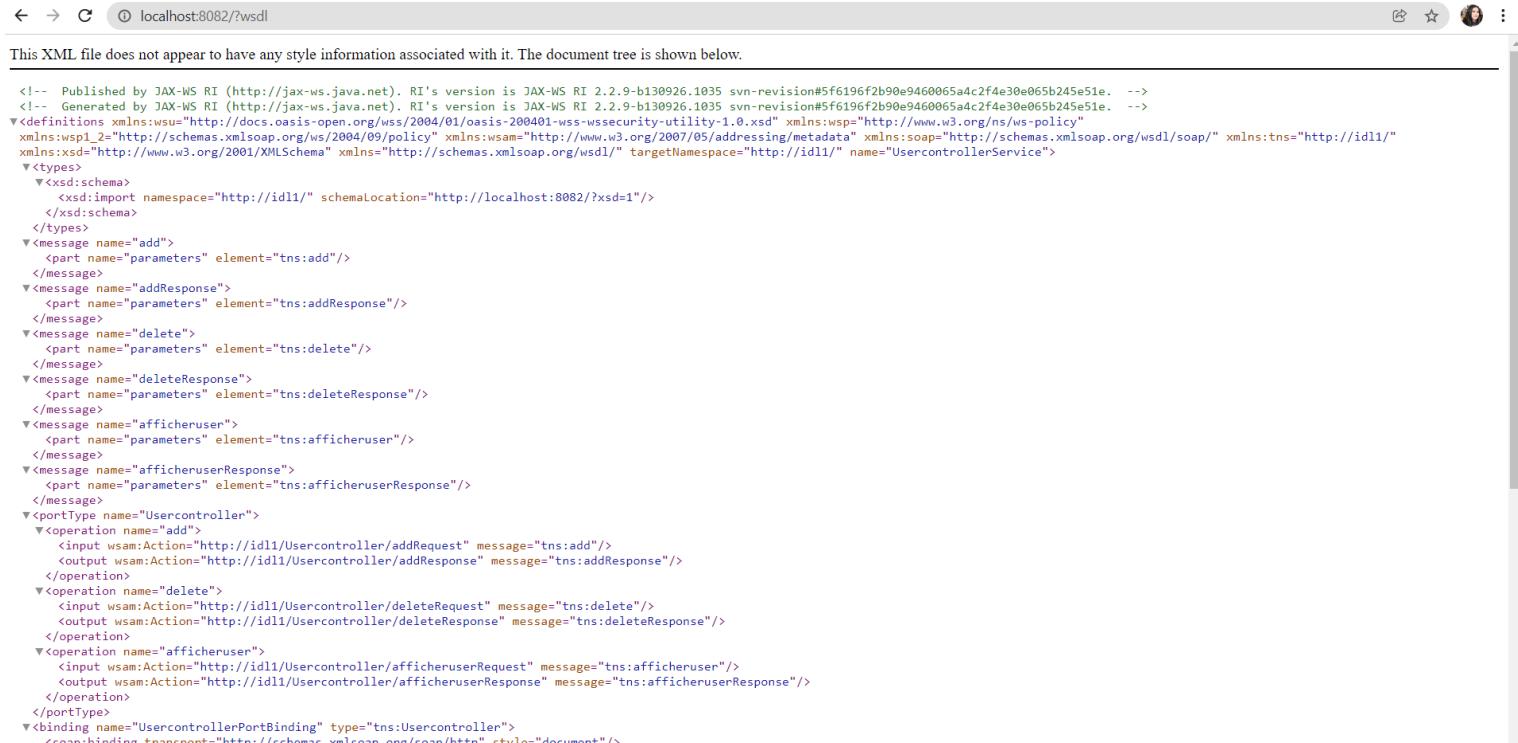
- Classe Hello.java :

```
serveur.java hello.java ×
1 package idl1;
2
3+ import java.util.ArrayList;...
4
5 @WebService
6 public class hello {
7     //exposé dans le web
8     @WebMethod
9     public String sayHello(){
10         return "HELLO WORLD";
11     }
12 }
```

# PARTIE 1:

## Exécution :

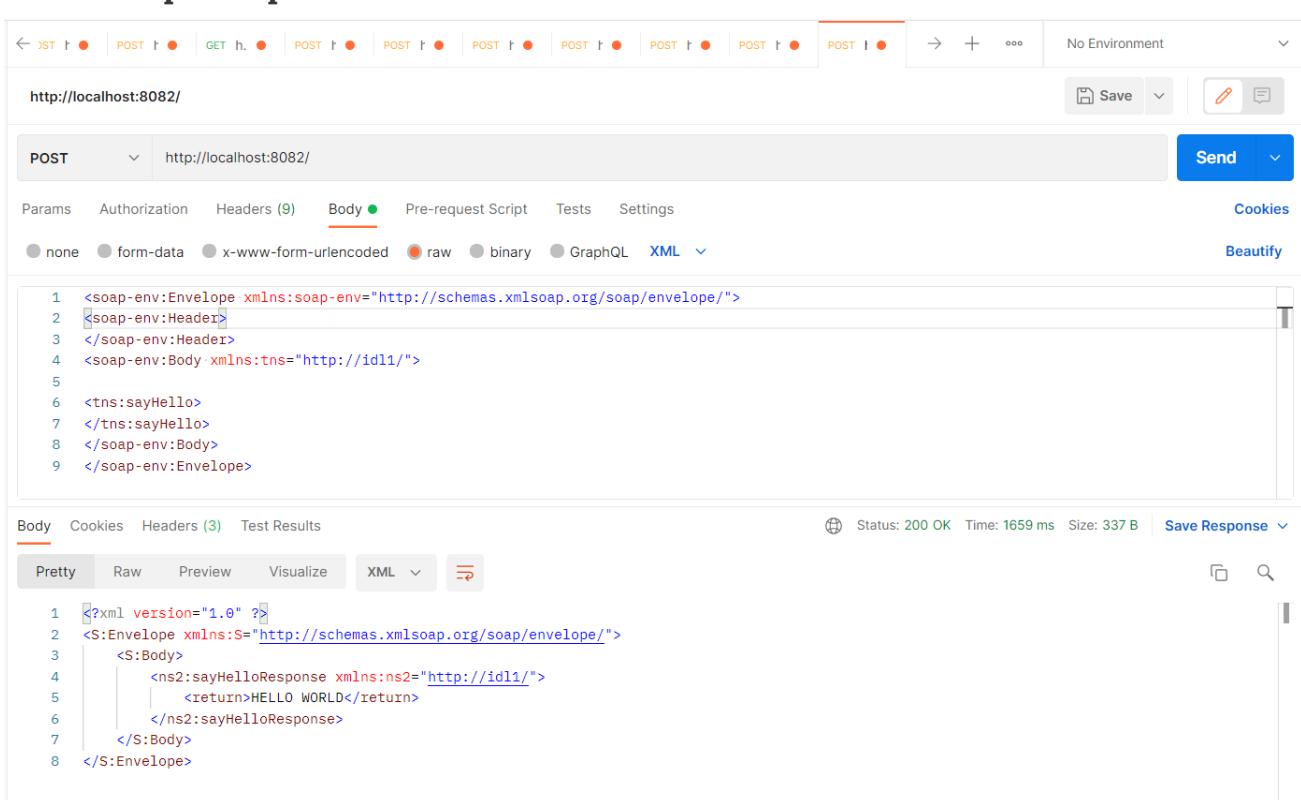
- Génération du fichier WSDL :



This XML file does not appear to have any style information associated with it. The document tree is shown below.

```
<!-- Published by JAX-WS RI (http://jax-ws.java.net). RI's version is JAX-WS RI 2.2.9-b130926.1035 svn-revision#5f6196f2b90e9460065a4c2f4e30e065b245e51e. -->
<!-- Generated by JAX-WS RI (http://jax-ws.java.net). RI's version is JAX-WS RI 2.2.9-b130926.1035 svn-revision#5f6196f2b90e9460065a4c2f4e30e065b245e51e. -->
<definitions xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd" xmlns:wsps="http://www.w3.org/ns/ws-policy"
  xmlns:wspl_2="http://schemas.xmlsoap.org/ws/2004/09/policy" xmlns:wsam="http://www.w3.org/2007/05/addressing/metadata" xmlns:soap="http://schemas.xmlsoap.org/wsdl/" xmlns:tns="http://id11/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns="http://schemas.xmlsoap.org/wsdl/" targetNamespace="http://id11/" name="UsercontrollerService">
  <types>
    <xsd:schema>
      <xsd:import namespace="http://id11/" schemaLocation="http://localhost:8082/?xsd=1"/>
    </xsd:schema>
  </types>
  <message name="add">
    <part name="parameters" element="tns:add"/>
  </message>
  <message name="addResponse">
    <part name="parameters" element="tns:addResponse"/>
  </message>
  <message name="delete">
    <part name="parameters" element="tns:delete"/>
  </message>
  <message name="deleteResponse">
    <part name="parameters" element="tns:deleteResponse"/>
  </message>
  <message name="afficheruser">
    <part name="parameters" element="tns:afficheruser"/>
  </message>
  <message name="afficheruserResponse">
    <part name="parameters" element="tns:afficheruserResponse"/>
  </message>
  <portType name="Usercontroller">
    <operation name="add">
      <input wsam:Action="http://id11/Usercontroller/addRequest" message="tns:add"/>
      <output wsam:Action="http://id11/Usercontroller/addResponse" message="tns:addResponse"/>
    </operation>
    <operation name="delete">
      <input wsam:Action="http://id11/Usercontroller/deleteRequest" message="tns:delete"/>
      <output wsam:Action="http://id11/Usercontroller/deleteResponse" message="tns:deleteResponse"/>
    </operation>
    <operation name="afficheruser">
      <input wsam:Action="http://id11/Usercontroller/afficheruserRequest" message="tns:afficheruser"/>
      <output wsam:Action="http://id11/Usercontroller/afficheruserResponse" message="tns:afficheruserResponse"/>
    </operation>
  </portType>
  <binding name="UsercontrollerPortBinding" type="tns:Usercontroller">
    <soap:binding transport="http://schemas.xmlsoap.org/soap/http" style="document"/>
    <operation name="add">
      <soap:operation soapAction="" />
```

- Exécution par le postman :



http://localhost:8082/

POST http://localhost:8082/

Params Authorization Headers (9) Body (green dot) Pre-request Script Tests Settings Cookies Beautify

Body

```
1 <soap-env:Envelope xmlns:soap-env="http://schemas.xmlsoap.org/soap/envelope/">
2   <soap-env:Header>
3   </soap-env:Header>
4   <soap-env:Body xmlns:tns="http://id11/">
5     6     <tns:sayHello>
7     </tns:sayHello>
8   </soap-env:Body>
9 </soap-env:Envelope>
```

Status: 200 OK Time: 1659 ms Size: 337 B Save Response

Pretty Raw Preview Visualize XML

```
1 <?xml version="1.0" ?>
2 <S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
3   <S:Body>
4     <ns2:sayHelloResponse xmlns:ns2="http://id11/">
5       <return>HELLO WORLD</return>
6     </ns2:sayHelloResponse>
7   </S:Body>
8 </S:Envelope>
```

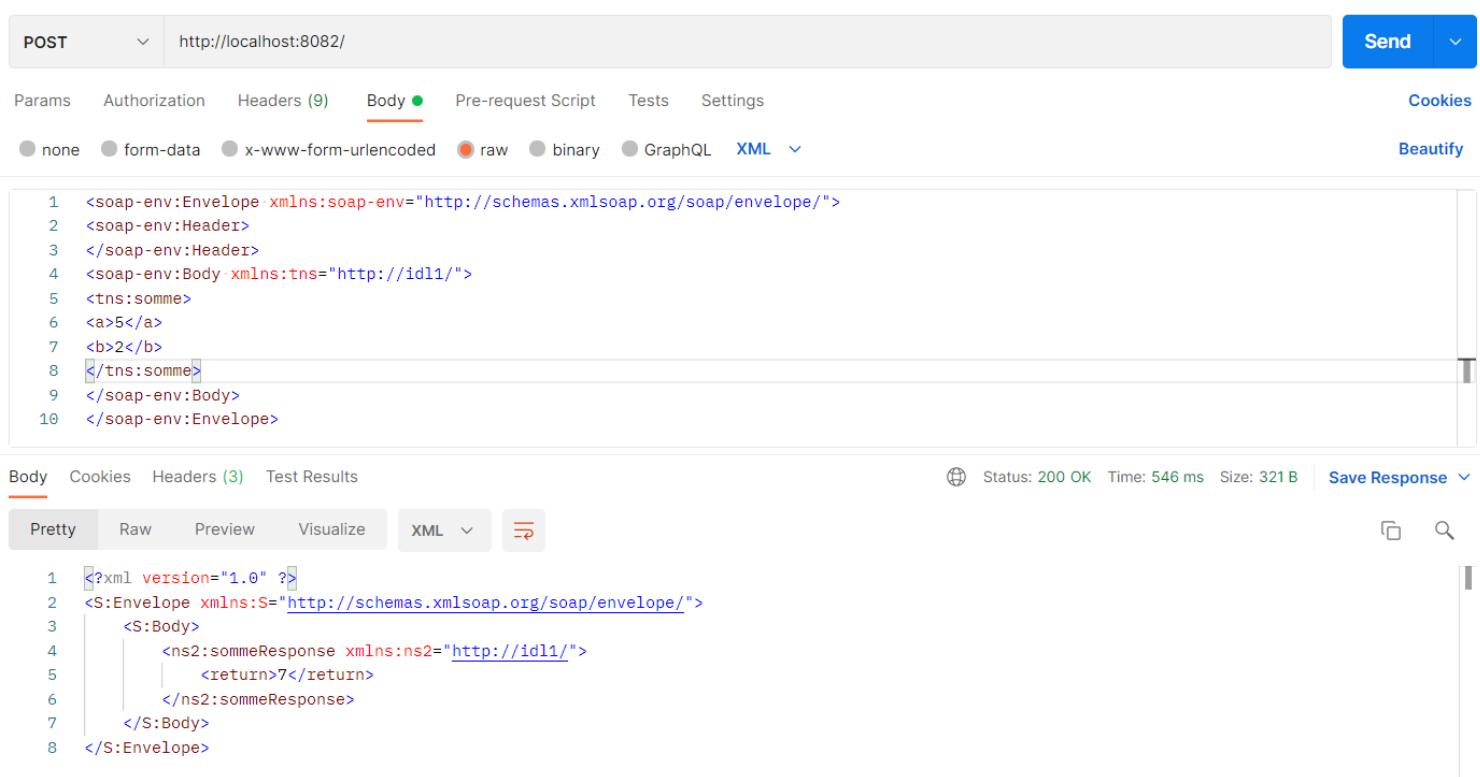
# PARTIE 2:

## Code de la méthode d'addititon :



```
serveur.java  hello.java X User.java  UserController.java
1 package idl1;
2
3 import java.util.ArrayList;
4
5 @WebService
6 public class hello {
7     //exposé dans le web
8     @WebMethod
9     public String sayHello(){
10         return "HELLO WORLD";
11     }
12     @WebMethod
13     public int somme(@WebParam(name="a")int a ,@WebParam(name="b") int b ){
14         return a+b;
15     }
16 }
```

## Exécution par le Postman :



POST http://localhost:8082/

Params Authorization Headers (9) Body Pre-request Script Tests Settings Cookies

none form-data x-www-form-urlencoded raw binary GraphQL XML Beautify

```
1 <soap-env:Envelope xmlns:soap-env="http://schemas.xmlsoap.org/soap/envelope/">
2 <soap-env:Header>
3 </soap-env:Header>
4 <soap-env:Body xmlns:tns="http://idl1/">
5   <tns:somme>
6     <a>5</a>
7     <b>2</b>
8   </tns:somme>
9 </soap-env:Body>
10 </soap-env:Envelope>
```

Body Cookies Headers (3) Test Results Status: 200 OK Time: 546 ms Size: 321 B Save Response

Pretty Raw Preview Visualize XML

```
1 <?xml version="1.0" ?>
2 <S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
3   <S:Body>
4     <ns2:sommeResponse xmlns:ns2="http://idl1/">
5       <return>7</return>
6     </ns2:sommeResponse>
7   </S:Body>
8 </S:Envelope>
```

# PARTIE 2:

## Code de la méthode de la soustraction:

```
serveur.java  hello.java X User.java  UserController.java
1 package idl1;
2
3 import java.util.ArrayList;
4
5 @WebService
6 public class hello {
7     //exposé dans le web
8     @WebMethod
9     public String sayHello(){
10         return "HELLO WORLD";
11     }
12     @WebMethod
13     public int somme(@WebParam(name="a")int a ,@WebParam(name="b") int b ){
14         return a+b;
15     }
16
17     @WebMethod
18     public int soustraction(@WebParam(name="a")int a ,@WebParam(name="b") int b ){
19         return a-b;
20     }
21
22 }
```

## Exécution par le Postman :

POST <http://localhost:8082/> Send

Params Authorization Headers (9) Body  Pre-request Script Tests Settings Cookies

none form-data x-www-form-urlencoded raw binary GraphQL **XML** Beautify

```
1 <soap-env:Envelope xmlns:soap-env="http://schemas.xmlsoap.org/soap/envelope/">
2 <soap-env:Header>
3 </soap-env:Header>
4 <soap-env:Body xmlns:tns="http://idl1/">
5 <tns:soustraction>
6 <a>5</a>
7 <b>2</b>
8 </tns:soustraction>
9 </soap-env:Body>
10 </soap-env:Envelope>
```

Body Cookies Headers (3) Test Results Status: 200 OK Time: 519 ms Size: 335 B Save Response

Pretty Raw Preview Visualize XML

```
1 <?xml version="1.0" ?>
2 <S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
3   <S:Body>
4     <ns2:soustractionResponse xmlns:ns2="http://idl1/">
5       <return>3</return>
6     </ns2:soustractionResponse>
7   </S:Body>
8 </S:Envelope>
```

# PARTIE 2:

## Code de la méthode de multiplication:

```
serveur.java  hello.java x User.java  UserController.java
1 package idl1;
2
3 import java.util.ArrayList;
4
5 @WebService
6 public class hello {
7     //exposé dans le web
8     @WebMethod
9     public String sayHello(){
10         return "HELLO WORLD";
11     }
12     @WebMethod
13     public int somme(@WebParam(name="a")int a ,@WebParam(name="b") int b ){
14         return a+b;
15     }
16     @WebMethod
17     public int soustraction(@WebParam(name="a")int a ,@WebParam(name="b") int b ){
18         return a-b;
19     }
20     @WebMethod
21     public int multiplication(@WebParam(name="a")int a ,@WebParam(name="b") int b ){
22         return a*b;
23     }
24 }
```

## Exécution par le Postman :

The screenshot shows the Postman interface with a SOAP request and its corresponding response.

**Request (POST):**

- URL: <http://localhost:8082/>
- Body tab selected.
- Raw XML content:

```
1 <soap-env:Envelope xmlns:soap-env="http://schemas.xmlsoap.org/soap/envelope/">
2   <soap-env:Header>
3   </soap-env:Header>
4   <soap-env:Body xmlns:tns="http://idl1/">
5     <tns:multiplication>
6       <a>5</a>
7       <b>2</b>
8     </tns:multiplication>
9   </soap-env:Body>
10  </soap-env:Envelope>
```

**Response:**

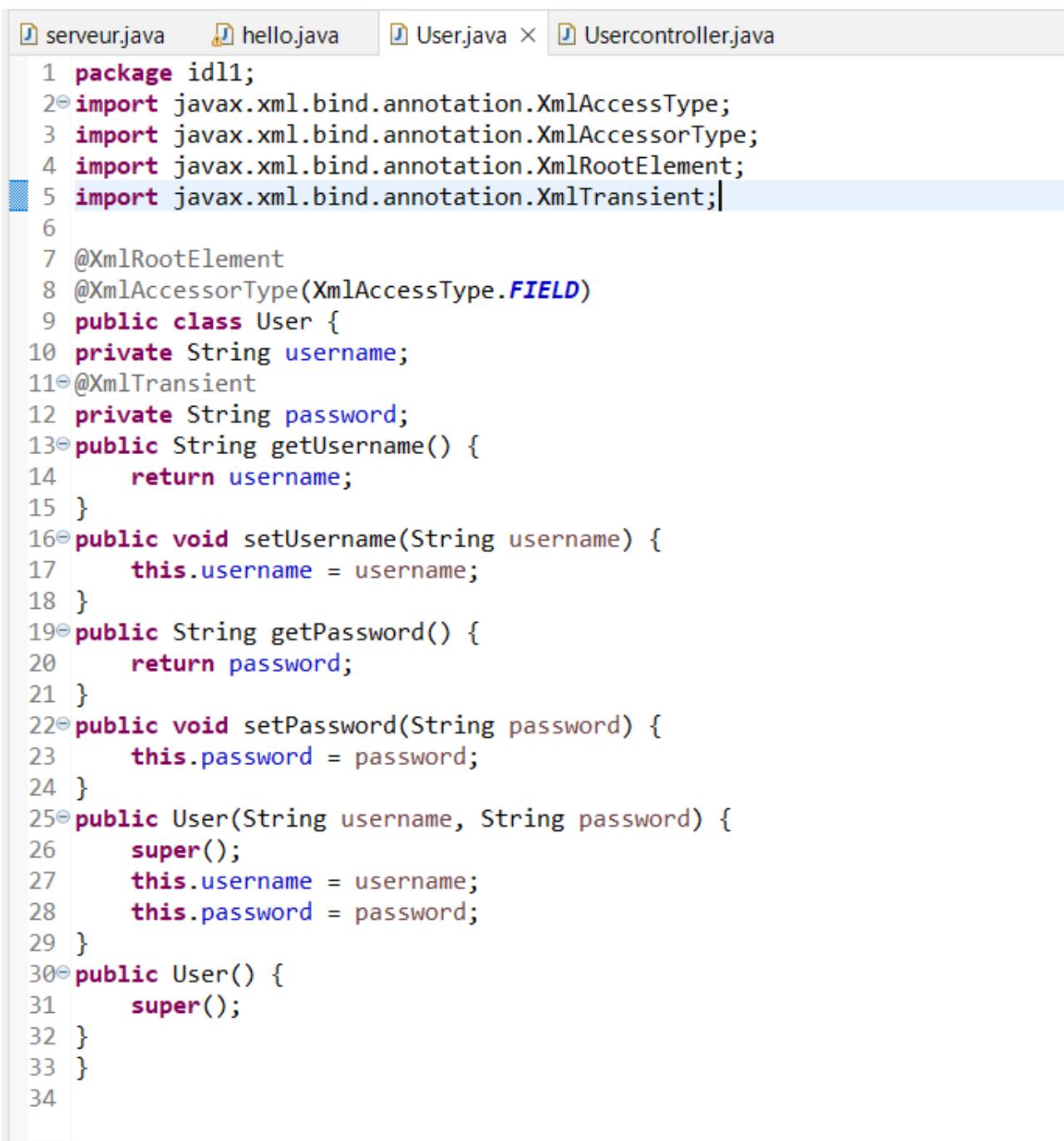
- Status: 200 OK
- Time: 524 ms
- Size: 340 B
- XML Content:

```
1 <?xml version="1.0" ?>
2 <S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
3   <S:Body>
4     <ns2:multiplicationResponse xmlns:ns2="http://idl1/">
5       <return>10</return>
6     </ns2:multiplicationResponse>
7   </S:Body>
8 </S:Envelope>
```

# PARTIE 3:

## Code :

- Classe User :



The screenshot shows a Java code editor with the User.java file open. The tabs at the top include serveur.java, hello.java, User.java (which is the active tab), and UserController.java. The code itself is a Java class named User with various methods and annotations for XML serialization.

```
1 package idl1;
2 import javax.xml.bind.annotation.XmlAccessType;
3 import javax.xml.bind.annotation.XmlAccessorType;
4 import javax.xml.bind.annotation.XmlRootElement;
5 import javax.xml.bind.annotation.XmlTransient;
6
7 @XmlRootElement
8 @XmlAccessorType(XmlAccessType.FIELD)
9 public class User {
10     private String username;
11     @XmlTransient
12     private String password;
13     public String getUsername() {
14         return username;
15     }
16     public void setUsername(String username) {
17         this.username = username;
18     }
19     public String getPassword() {
20         return password;
21     }
22     public void setPassword(String password) {
23         this.password = password;
24     }
25     public User(String username, String password) {
26         super();
27         this.username = username;
28         this.password = password;
29     }
30     public User() {
31         super();
32     }
33 }
34
```

# PARTIE 3:

- Classe UserController :

```
serveur.java hello.java User.java UserController.java
1 package idl1;
2
3 import java.util.ArrayList;
4 @WebService
5 public class Usercontroller {
6     public List<User> listU=new ArrayList<>();
7
8     @WebMethod
9     public User add(@WebParam(name="u")User u ){
10         listU.add(u);
11         return u;
12     }
13
14     @WebMethod
15     public List<User> afficheruser(){
16
17         return listU;
18     }
19
20     @WebMethod
21     public List<User> delete(@WebParam(name="id")int id) {
22         listU.remove(id);
23         return listU ;
24     }
25
26 }
27
28 }
```

- Exécution de AddUser :

- Ajout de l'utilisateur "Sahar" avec le password "Sahar" :

POST <http://localhost:8082/> Send

Params Authorization Headers (9) Body ● Pre-request Script Tests Settings Cookies

none form-data x-www-form-urlencoded raw binary GraphQL XML Beautify

```
<soap-env:Envelope xmlns:soap-env="http://schemas.xmlsoap.org/soap/envelope/">
<soap-env:Header>
</soap-env:Header>
<soap-env:Body xmlns:tns="http://idl1/">
<tns:add>
<u>
<username>
sahar
</username>
<password>
sahar
</password>
</u>
</tns:add>
</soap-env:Body>
</soap-env:Envelope>
```

# PARTIE 3:



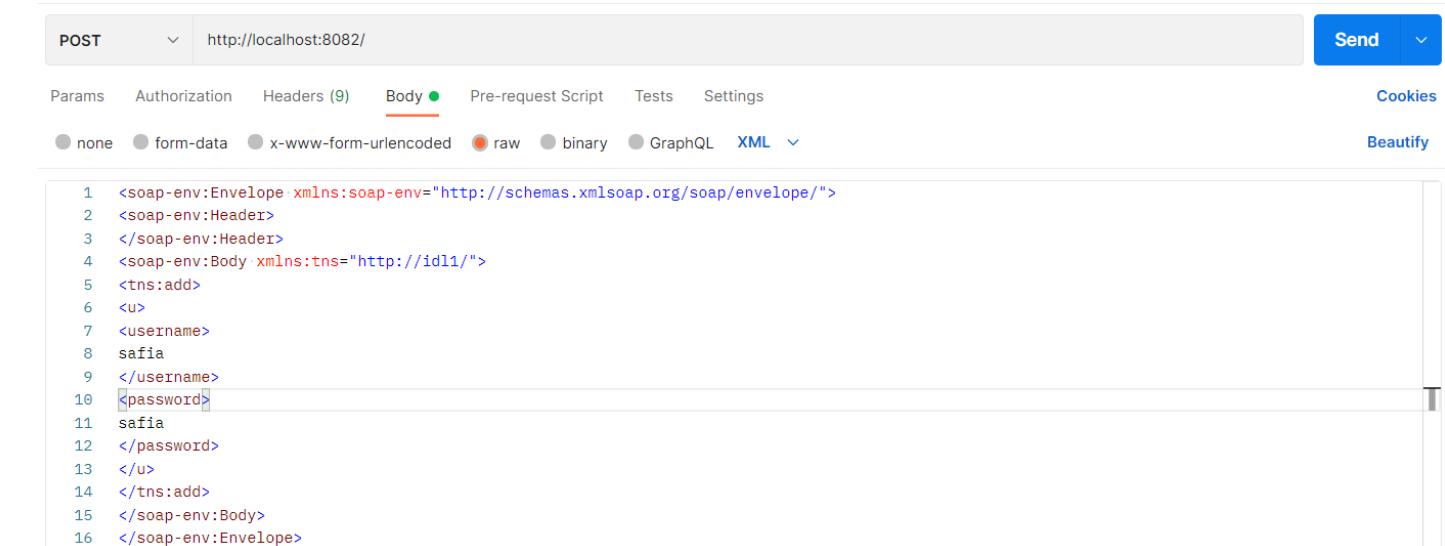
Body Cookies Headers (3) Test Results

Status: 200 OK Time: 653 ms Size: 344 B Save Response

Pretty Raw Preview Visualize XML

```
1 <?xml version="1.0" ?>
2 <S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
3   <S:Body>
4     <ns2:addResponse xmlns:ns2="http://idl1/">
5       <return>
6         |   <username>
7           sahar
8         </username>
9       </return>
10      </ns2:addResponse>
11    </S:Body>
12  </S:Envelope>
```

- Ajout de l'utilisateur "Safia" avec le password "Safia" :



POST http://localhost:8082/ Send

Params Authorization Headers (9) Body Pre-request Script Tests Settings Cookies Beautify

none form-data x-www-form-urlencoded raw binary GraphQL XML

```
1 <soap-env:Envelope xmlns:soap-env="http://schemas.xmlsoap.org/soap/envelope/">
2 <soap-env:Header>
3 </soap-env:Header>
4 <soap-env:Body xmlns:tns="http://idl1/">
5   <tns:add>
6     <u>
7       <username>
8         safia
9       </username>
10      <password>
11        safia
12      </password>
13    </u>
14  </tns:add>
15 </soap-env:Body>
16 </soap-env:Envelope>
```



Cookies Headers (3) Test Results

Status: 200 OK Time: 529 ms Size: 344 B Save Response

Pretty Raw Preview Visualize XML

```
<?xml version="1.0" ?>
<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
  <S:Body>
    <ns2:addResponse xmlns:ns2="http://idl1/">
      <return>
        |   <username>
safia
</username>
      </return>
    </ns2:addResponse>
  </S:Body>
</S:Envelope>
```

# PARTIE 3:

- Exécution de ListUser :

The screenshot shows the Postman application interface. A POST request is being made to the URL `http://localhost:8082/`. The 'Body' tab is selected, showing the following XML payload:

```
1 <soap-env:Envelope xmlns:soap-env="http://schemas.xmlsoap.org/soap/envelope/">
2 <soap-env:Header>
3 </soap-env:Header>
4 <soap-env:Body xmlns:tns="http://id11/">
5 <tns:afficheruser>
6 </tns:afficheruser>
7 </soap-env:Body>
8 </soap-env:Envelope>
```

The response status is 200 OK, with a response time of 1674 ms and a size of 407 B. The response body is displayed in pretty XML format:

```
1 <?xml version="1.0" ?>
2 <S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
3   <S:Body>
4     <ns2:afficheruserResponse xmlns:ns2="http://id11/">
5       <return>
6         |   <username>
7           sahar
8         </username>
9       </return>
10      <return>
11        |   <username>
12          safia
13        </username>
14      </return>
15    </ns2:afficheruserResponse>
16  </S:Body>
17 </S:Envelope>
```

The screenshot shows the Postman application interface. A POST request is being made to the URL `http://localhost:8082/`. The 'Body' tab is selected, showing the following XML payload:

```
1 <soap-env:Envelope xmlns:soap-env="http://schemas.xmlsoap.org/soap/envelope/">
2 <soap-env:Header>
3 </soap-env:Header>
4 <soap-env:Body xmlns:tns="http://id11/">
5 <tns:afficheruser>
6 </tns:afficheruser>
7 </soap-env:Body>
8 </soap-env:Envelope>
```

The response status is 200 OK, with a response time of 1674 ms and a size of 407 B. The response body is displayed in pretty XML format:

```
1 <?xml version="1.0" ?>
2 <S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
3   <S:Body>
4     <ns2:afficheruserResponse xmlns:ns2="http://id11/">
5       <return>
6         |   <username>
7           sahar
8         </username>
9       </return>
10      <return>
11        |   <username>
12          safia
13        </username>
14      </return>
15    </ns2:afficheruserResponse>
16  </S:Body>
17 </S:Envelope>
```

# PARTIE 3:

- Exécution de DeleteUser :

The screenshot shows a Postman interface with the following details:

- Method:** POST
- URL:** http://localhost:8082/
- Body:** XML (selected)
- XML Body Content:**

```
1 <soap-env:Envelope xmlns:soap-env="http://schemas.xmlsoap.org/soap/envelope/">
2   <soap-env:Header>
3   </soap-env:Header>
4   <soap-env:Body xmlns:tns="http://id11/">
5     <tns:delete>
6       <id>
7         0
8       </id>
9     </tns:delete>
10    </soap-env:Body>
11  </soap-env:Envelope>
```

- Response Status:** 200 OK
- Response Time:** 800 ms
- Response Size:** 350 B
- Response Preview:** XML (Pretty)

```
1 <?xml version="1.0" ?>
2 <S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
3   <S:Body>
4     <ns2:deleteResponse xmlns:ns2="http://id11/">
5       <return>
6         <username>
7           safia
8         </username>
9       </return>
10      </ns2:deleteResponse>
11    </S:Body>
12  </S:Envelope>
```