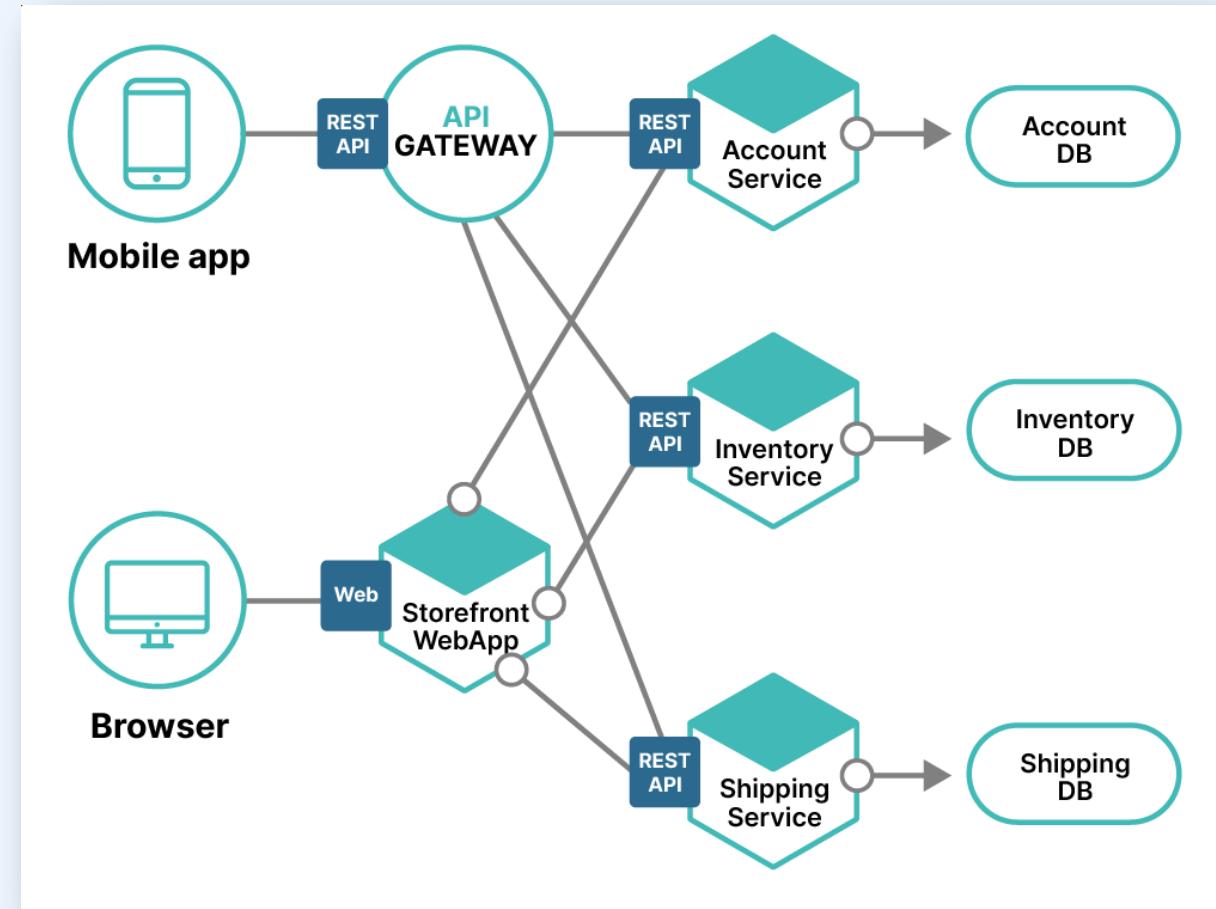




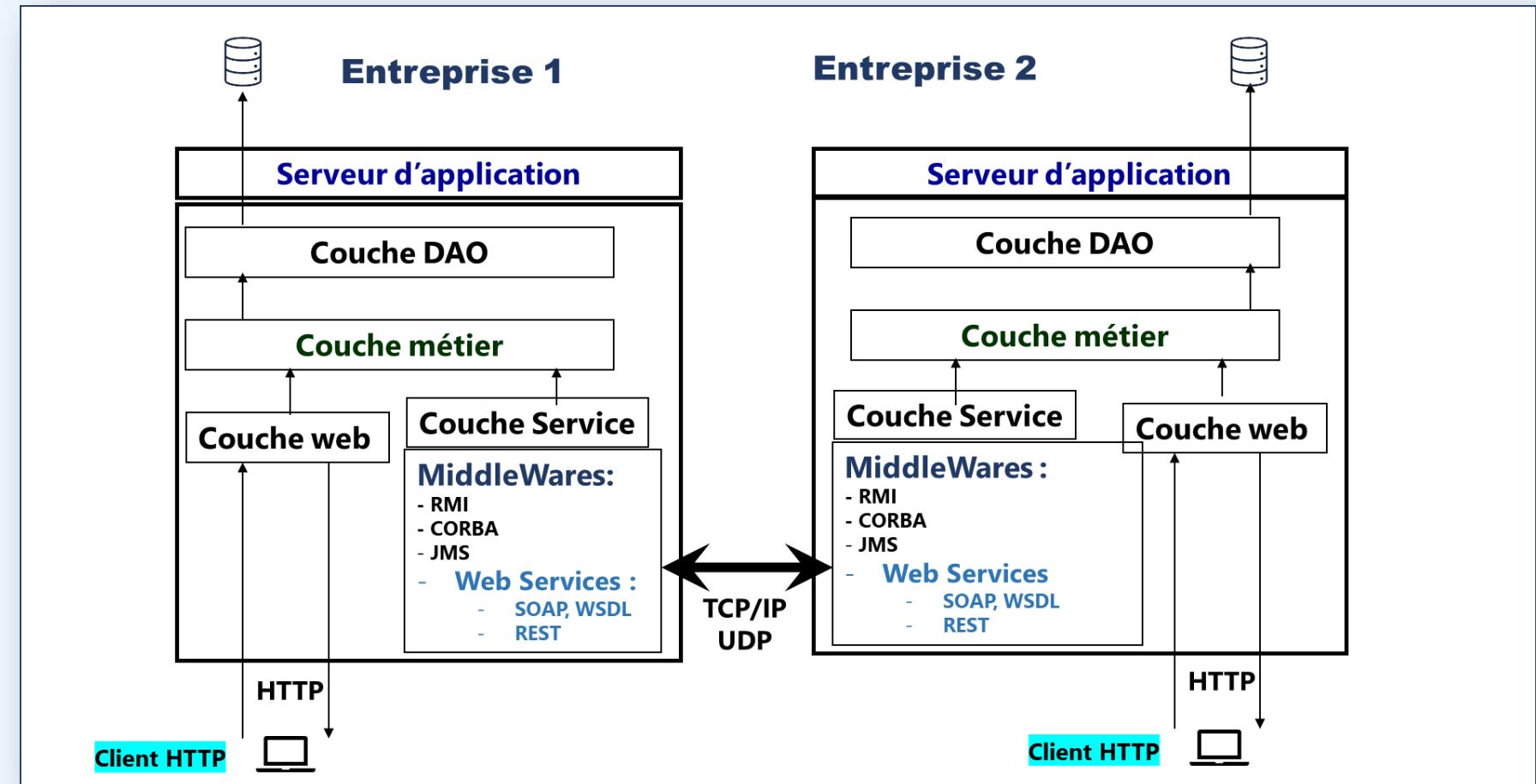
Web services (SOAP et REST)

Système distribué

Un système distribué est un ensemble de composants logiciels individuels, situés sur différents ordinateurs qui partagent des messages entre eux pour atteindre des objectifs communs.



Architecture Système distribué





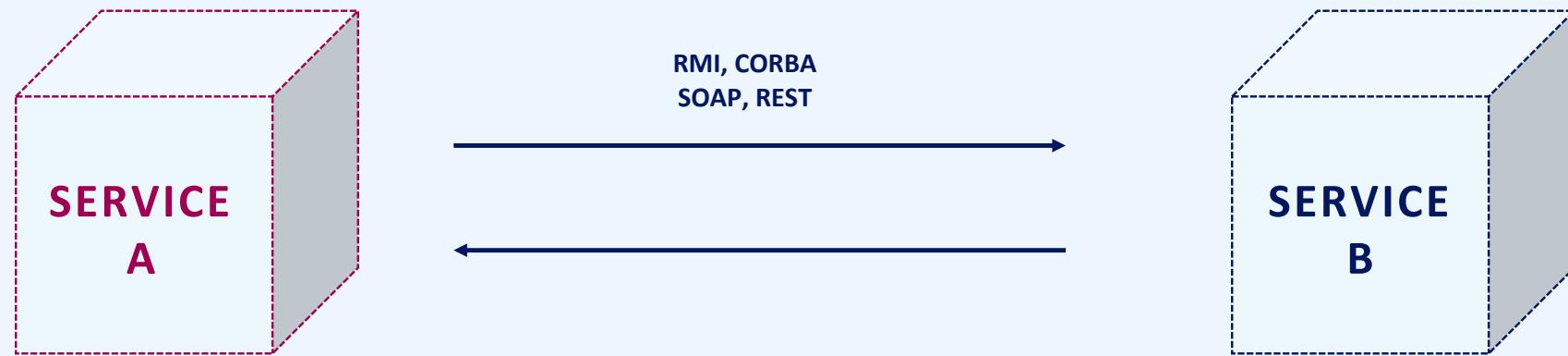
Quelques Middlewares

RMI (Remote Method Invocation) : est une API qui fournit un mécanisme pour créer une application distribuée en Java. Le RMI permet à un objet d'invoquer des méthodes sur un objet s'exécutant dans une autre JVM.

CORBA (Common Object Request Broker Architecture) : est une architecture logicielle qui permet de construire des applications complètes, peuvent être écrits dans des langages de programmation distincts, être exécutés dans des processus séparés, voire être déployés sur des machines distinctes.



Middlewares Synchrones



Communication synchrone



Middlewares Synchrones

Dans une communication synchrone :

Un service A dans une machine M1 établie une connexion avec un service B dans une autre machine M2.

Si le service B n'est pas disponible, il n'y aura pas de communication

Si oui, Le service A envoie ensuite une requête pour faire appel à une opération à distance du service B.

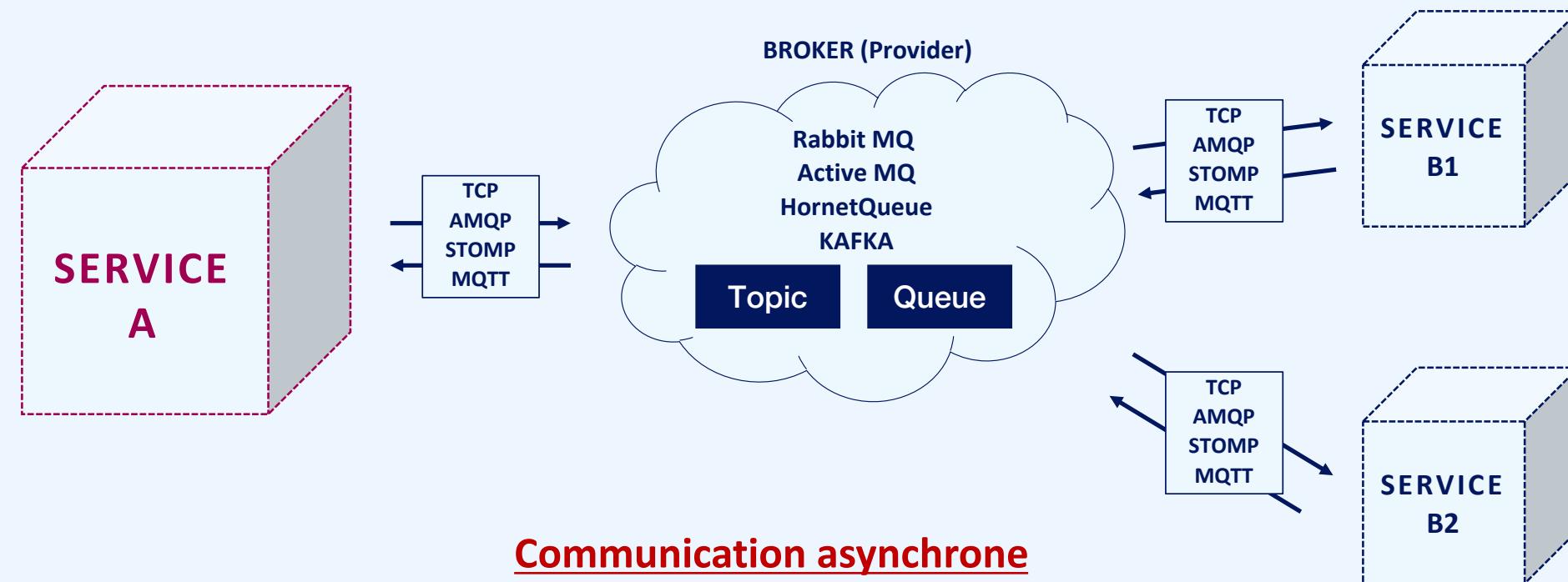
- Le service A est bloqué jusqu'à ce qu'il reçoit la réponse du service B.



Communication synchrone



Middlewares Asynchrones



Middlewares Asynchrones

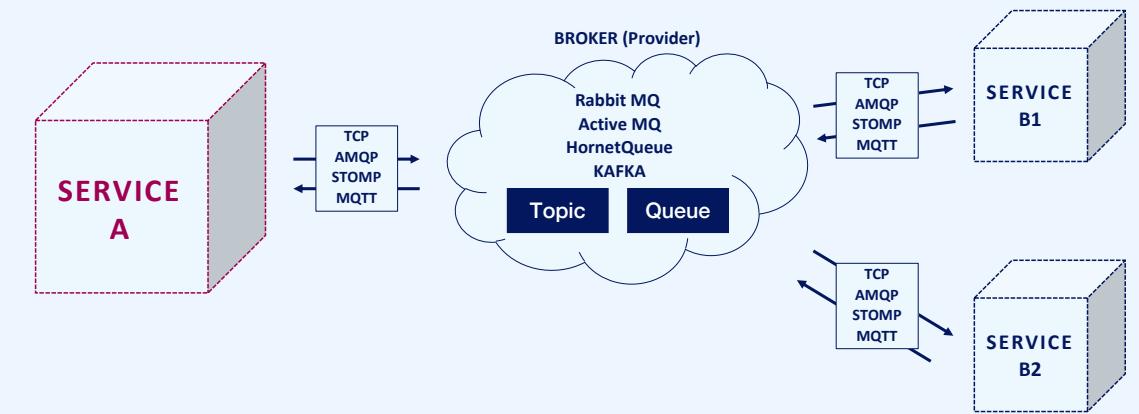
Dans une communication asynchrone :

Un service A établie une connexion vers un provider (Broker).

Le service A envoie le message à une file d'attente du Broker pour le délivrer au service B.

Si le service B n'est pas disponible, cela n'empêche pas le service A d'envoyer le message. Dans ce cas là message est conservé dans la file d'attente.

Si B est connecté à la file d'attente du broker, ce dernier lui délivre le message.



Web Services

Les Web Services sont des composants web basés sur Internet (HTTP) qui exécutent des tâches précises et qui respectent un format spécifique (XML).

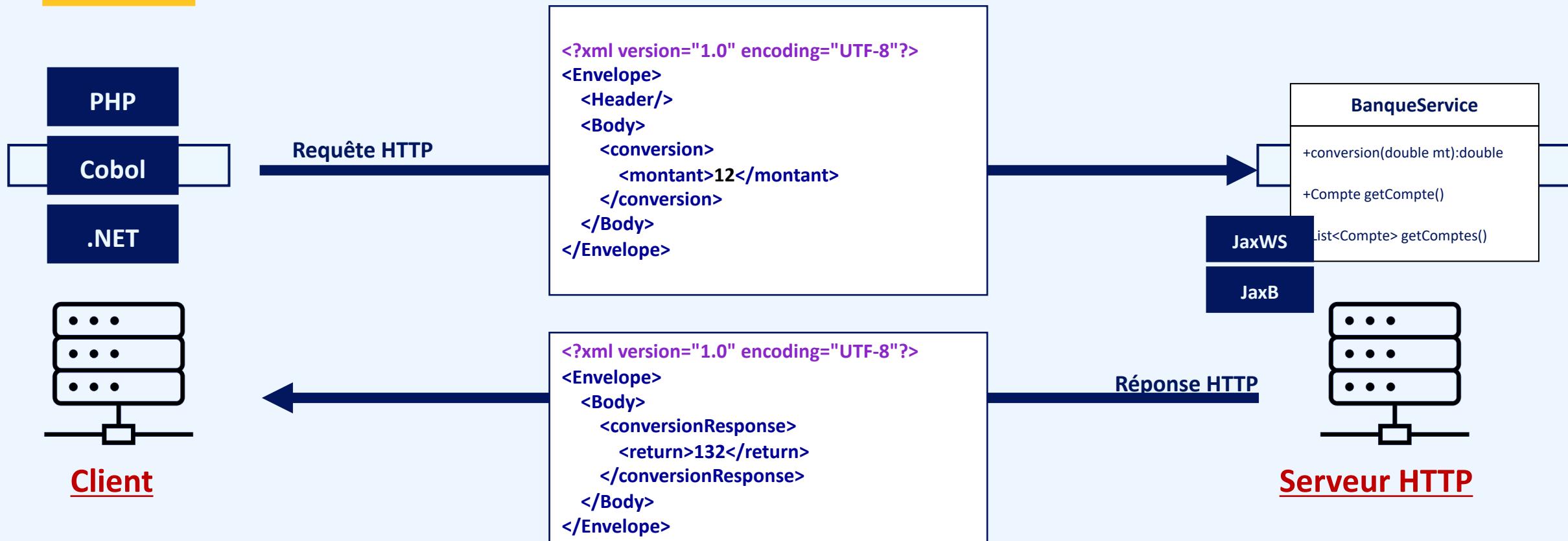
Ils permettent aux applications de faire appel à des fonctionnalités à distance en simplifiant ainsi l'échange de données.

Les Web Services permettent aux applications de dialoguer à travers le réseau, indépendamment de leur plate-forme d'exécution et de leur langage d'implémentation.

A voir: <https://www.youtube.com/watch?v=je9iCRcdJnc>



Architecture de base des Web Services



Requête SOAP avec POST

```
1 ⊞ <soapenv:Envelope
2     xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
3     xmlns:ws="http://ws/"
4     <soapenv:Header/>
5     <soapenv:Body>
6     <ws:somme>
7         <arg0>1</arg0>
8         <arg1>0</arg1>
9     </ws:somme>
10    </soapenv:Body>
11 </soapenv:Envelope>
```

```
POST http://localhost:8087/ HTTP/1.1
Accept-Encoding: gzip,deflate
Content-Type: text/xml;charset=UTF-8
SOAPAction: ""
Content-Length: 275
Host: localhost:8087
Connection: Keep-Alive
User-Agent: Apache-HttpClient/4.5.5 (Java/16.0.1)
```



Réponse SOAP avec POST

```
1 ⊞ <S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
2 ⊞   <S:Body>
3 ⊞     <ns2:sommeResponse xmlns:ns2="http://ws/">
4 ⊞       <return>1</return>
5 ⊞     </ns2:sommeResponse>
6 ⊞   </S:Body>
7 ⊞ </S:Envelope>
```

HTTP/1.1 200 OK
Date: Thu, 06 Oct 2022 11:47:34 GMT
Transfer-encoding: chunked
Content-type: text/xml; charset=utf-8



Concepts fondamentaux des web services

À l'aide d'un groupe émergent de normes, les organisations sont en mesure de rendre les logiciels disponibles en tant que service sur Internet, de sorte que lorsque vous utilisez un Web Service dans votre application, vous n'avez pas besoin de savoir comment il a été implémenté. Les normes sont :

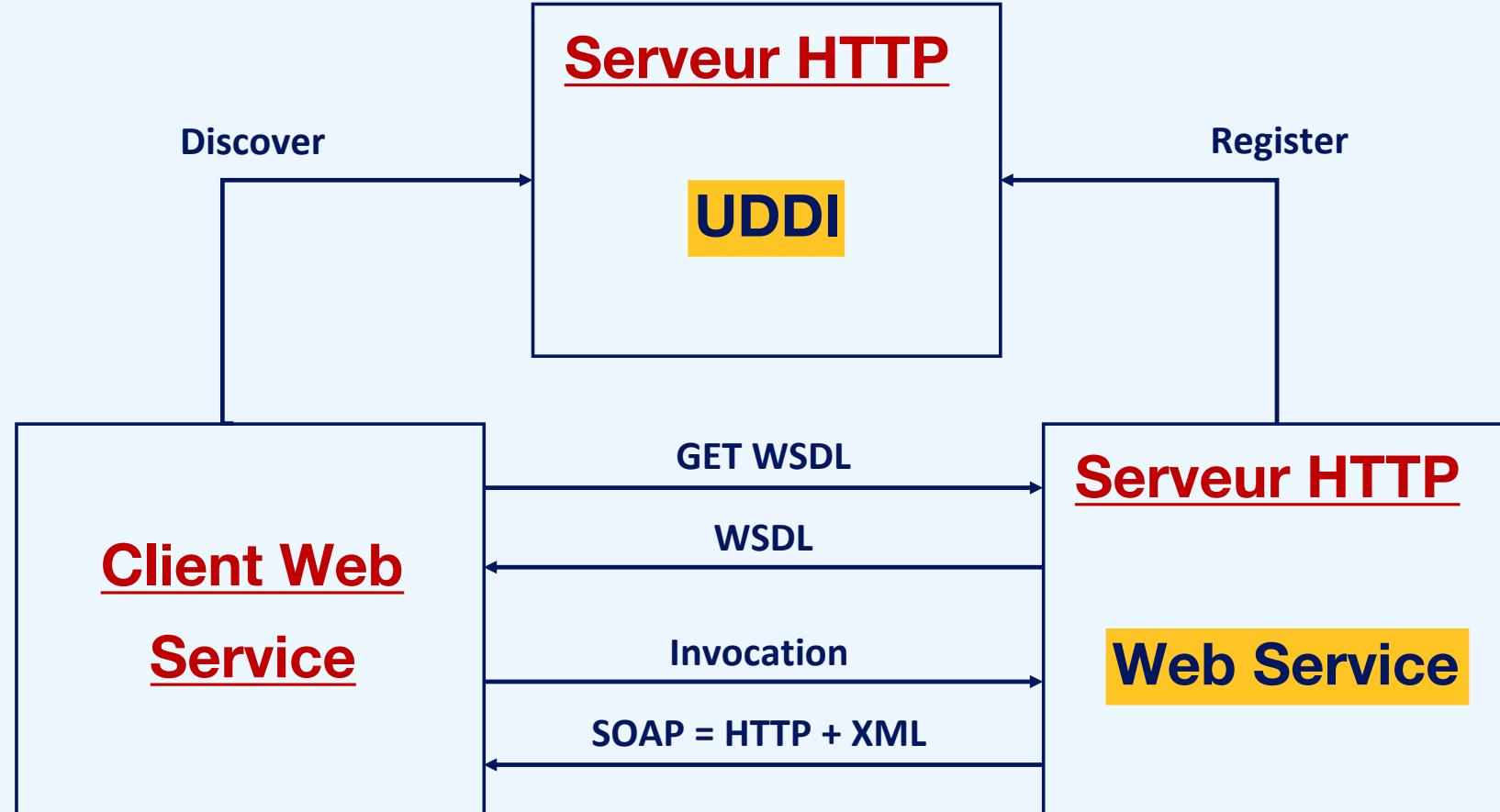
SOAP (Simple Object Access Protocol) : est un protocole d'échange inter-applications indépendant de toute plate-forme, basé sur le langage XML.

WSDL (Web Services Description Language) : donne la description au format XML des Web Services en précisant les méthodes pouvant être invoquées, leurs signatures et le point d'accès (URL, port, etc..).

UDDI (Universal Description, Discovery and Integration) : normalise une solution d'annuaire distribué de Web Services, permettant à la fois la publication et l'exploration (recherche) de Web Services.



UUDI



Installation d'outils

Installation d'outils -



Lien de téléchargement: <https://www.jetbrains.com/idea/download/>

Télécharger la version « Ultimate » (Utilisez votre adresse EMSI pour activer la licence)



Lien de téléchargement: <https://www.soapui.org/downloads/soapui/>





Activité Pratique N°:1

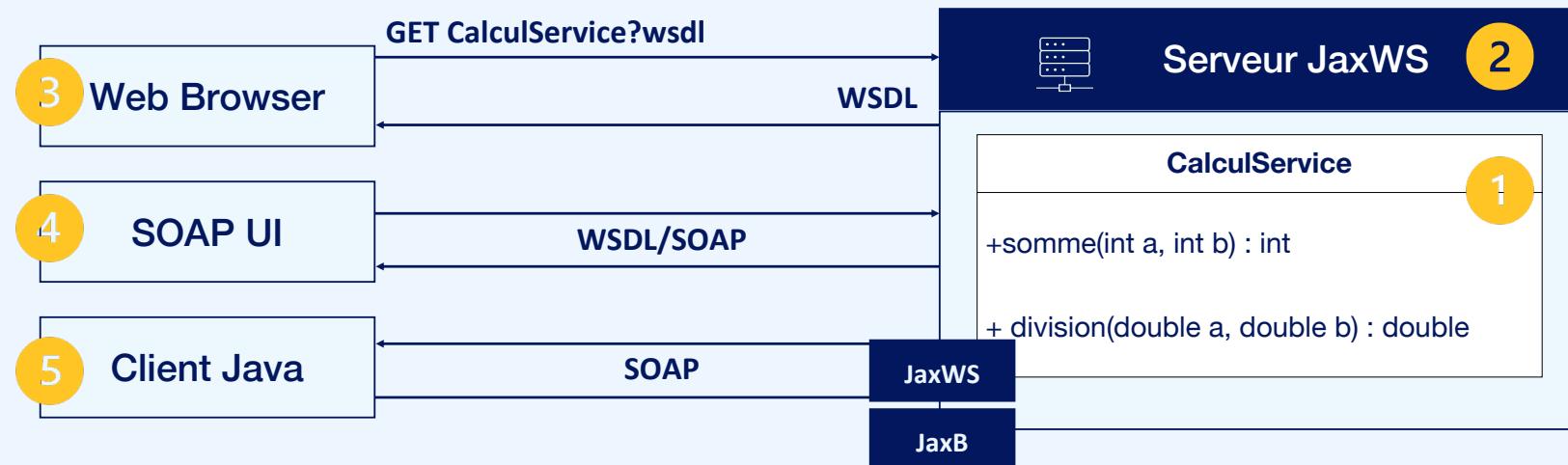
Activité Pratique N°:1

- ✓ Création d'un simple Web Service qui permet de:
 - Faire des calculs arithmétiques de base: Somme, soustraction...
- ✓ Déployer le web service avec un simple Serveur JaxWS
- ✓ Consulter et analyser le WSDL avec un Browser HTTP
- ✓ Tester les opérations du web service avec un outil comme SoapUI
- ✓ Créer un client SOAP Java



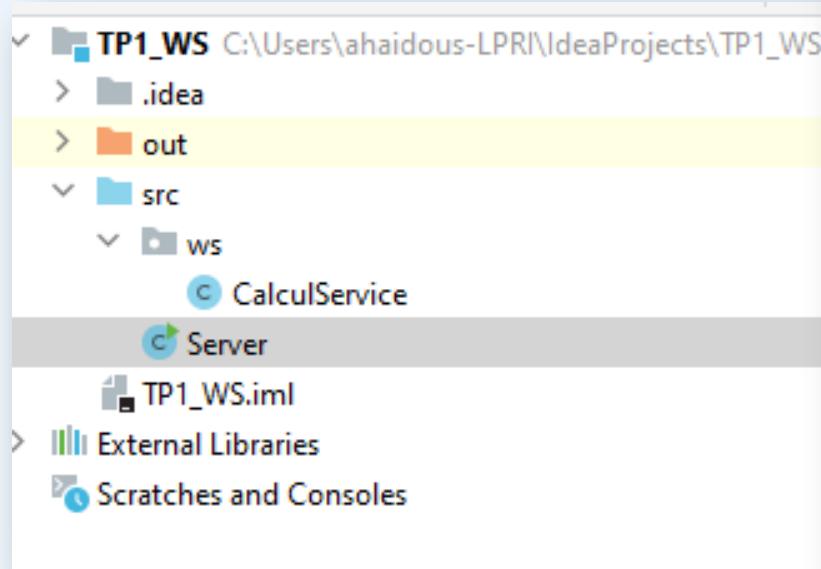
Activité Pratique N°:1

- ✓ Création d'un simple Web Service qui permet de:
 - Faire des calculs arithmétiques de base: Somme, soustraction...
- ✓ Déployer le web service avec un simple Serveur JaxWS
- ✓ Consulter et analyser le WSDL avec un Browser HTTP
- ✓ Tester les opérations du web service avec un outil comme SoapUI
- ✓ Crée un client SOAP Java



Activité Pratique N°:1

→ Création des classes



```
package ws;

import javax.jws.WebMethod;
import javax.jws.WebParam;
import javax.jws.WebService;

2 usages
@WebService
public class CalculService {

    @WebMethod
    public int somme(@WebParam int a, @WebParam int b) { return a+b; }

    @WebMethod
    public int soustraction(@WebParam int a, @WebParam int b) { return a-b; }

    @WebMethod
    public long multiplication(@WebParam long a,@WebParam long b) { return a*b; }

    @WebMethod
    public int inverser(@WebParam int a){
        return -a;
    }
}
```

Annotations JaxWS

Annotation	Description
@WebService	Elle marque une classe Java comme implémentant d'un service Web.
@WebMethod	Elle désigne une méthode qui est une opération de service Web.
@WebParam	Elle personnalise le mappage d'un paramètre individuel à une partie de message de service web et à un élément XML.

Activité Pratique N°:1

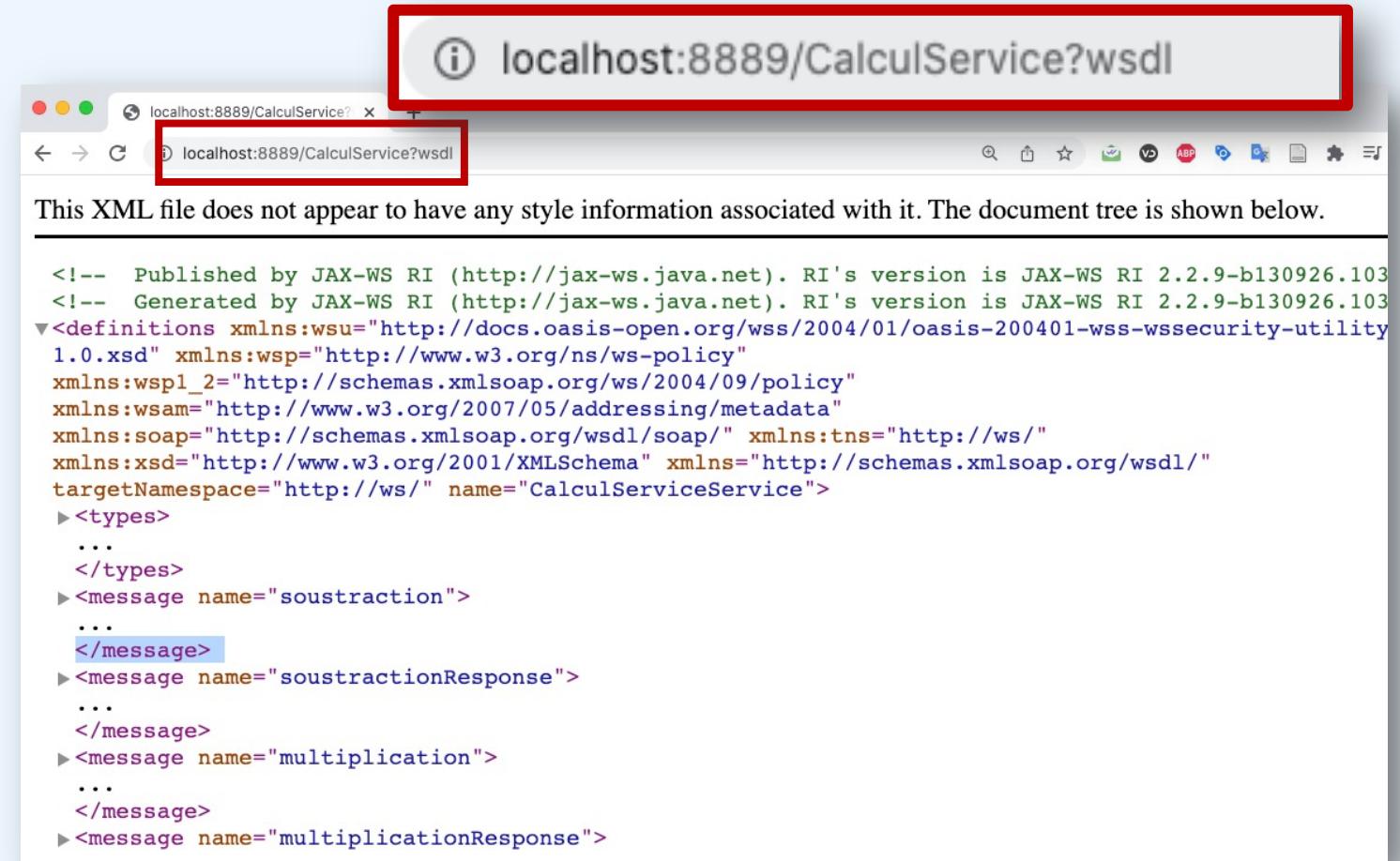
→ Déploiement du web service avec JaxWS

```
import ws.CalculService;  
  
import javax.xml.ws.Endpoint;  
  
public class Server {  
    public static void main(String[] args) {  
  
        String url ="http://0.0.0.0:8087/";  
        Endpoint.publish(url, new CalculService());  
  
    }  
}
```



Activité Pratique N°:1

→ Consultation du WSDL sur un browser



localhost:8889/CalculService?wsdl

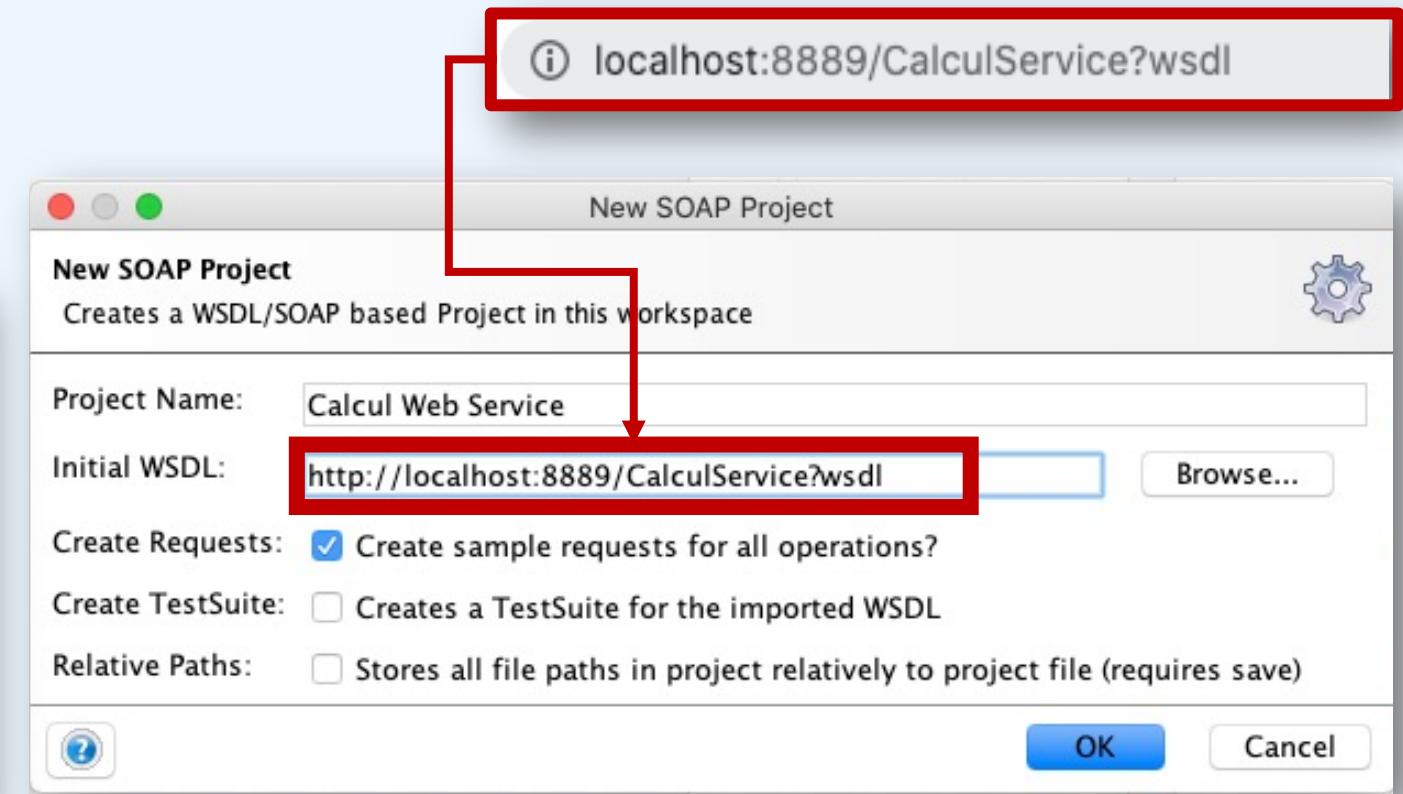
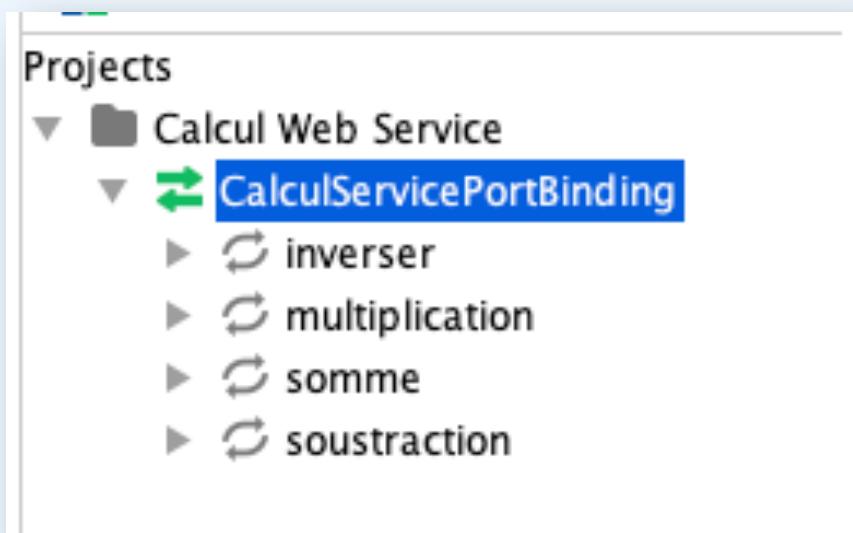
This XML file does not appear to have any style information associated with it. The document tree is shown below.

```
<!-- Published by JAX-WS RI (http://jax-ws.java.net). RI's version is JAX-WS RI 2.2.9-b130926.103
<!-- Generated by JAX-WS RI (http://jax-ws.java.net). RI's version is JAX-WS RI 2.2.9-b130926.103
<?xml version="1.0" encoding="UTF-8"?
<definitions xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd" xmlns:wsp="http://www.w3.org/ns/ws-policy"
  xmlns:wsp1_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/soap/" xmlns:tns="http://ws/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns="http://schemas.xmlsoap.org/wsdl/"
  targetNamespace="http://ws/" name="CalculServiceService">
  <types>
    ...
  </types>
  <message name="soustraction">
    ...
    </message>
  <message name="soustractionResponse">
    ...
  </message>
  <message name="multiplication">
    ...
  </message>
  <message name="multiplicationResponse">
    ...
  </message>
</definitions>
```



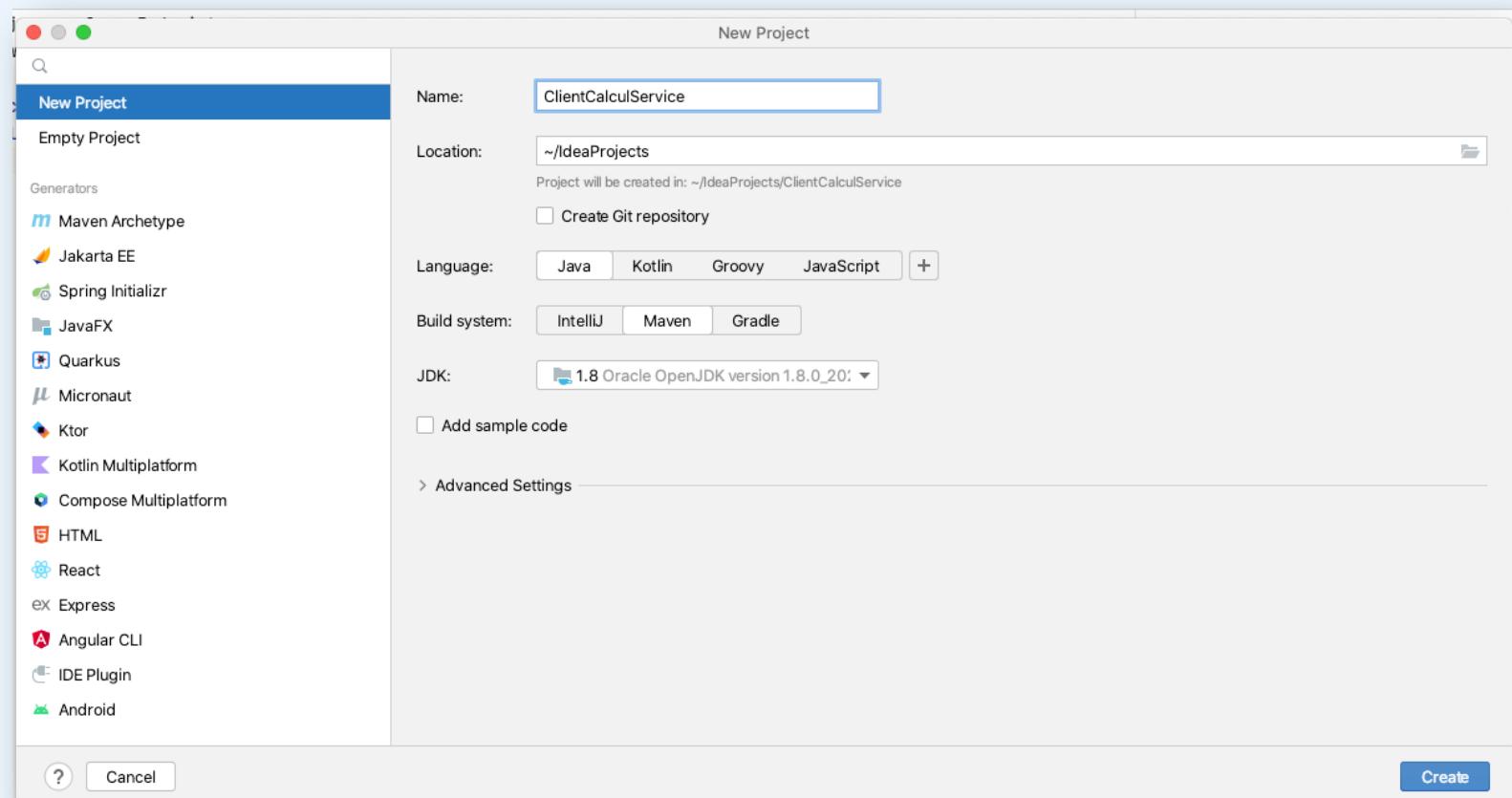
Activité Pratique N°:1

→ Tester les opérations avec SOAP UI



Activité Pratique N°:1

→ Création d'un client Java



Activité Pratique N°:1

→ Ajout de la dépendance MAVEN

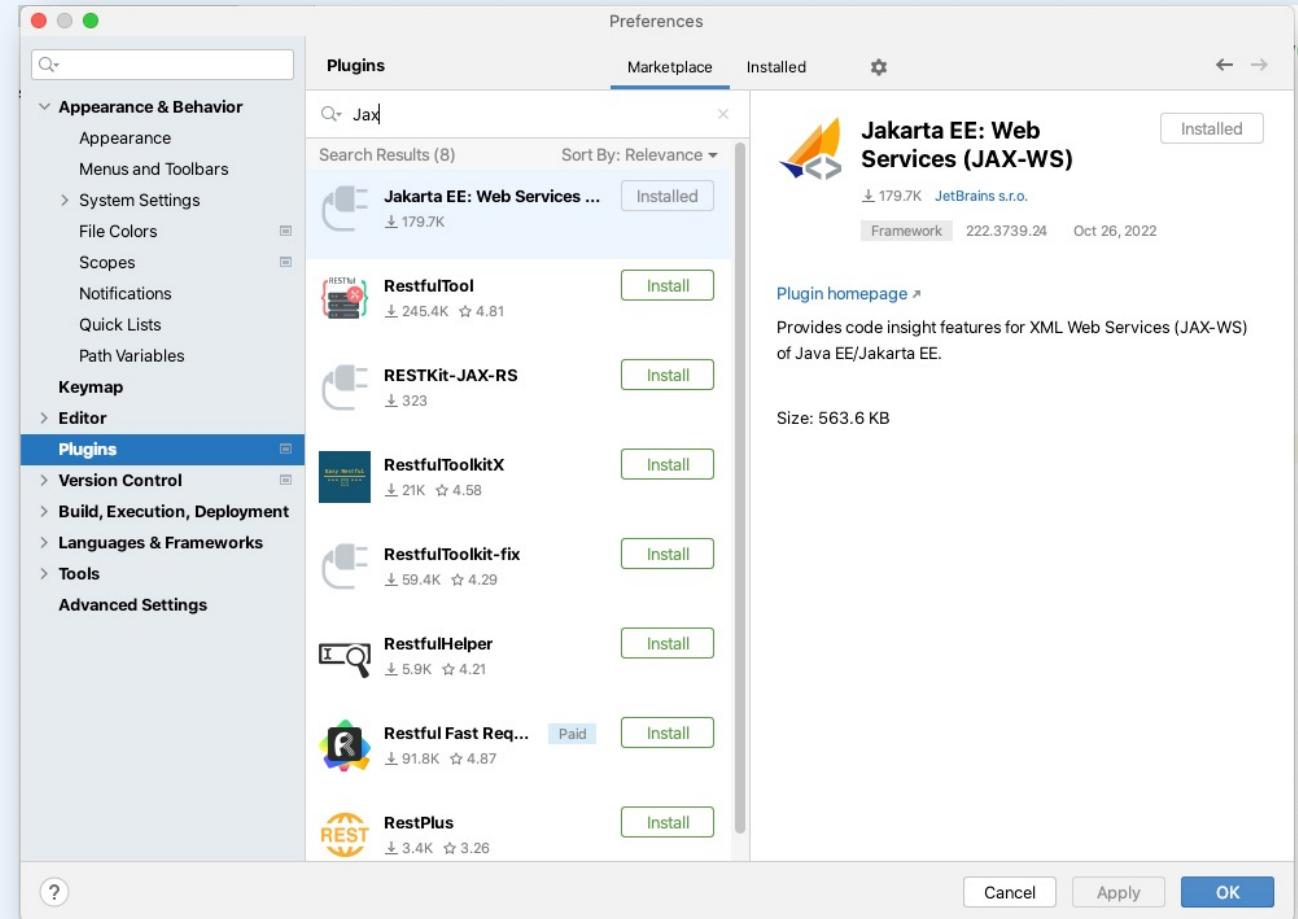
```
<dependencies>
    <!-- https://mvnrepository.com/artifact/javax.xml.ws/jaxws-api -->
    <dependency>
        <groupId>javax.xml.ws</groupId>
        <artifactId>jaxws-api</artifactId>
        <version>2.3.1</version>
    </dependency>
</dependencies>
```



Activité Pratique N°:1

→ Création d'un client Java

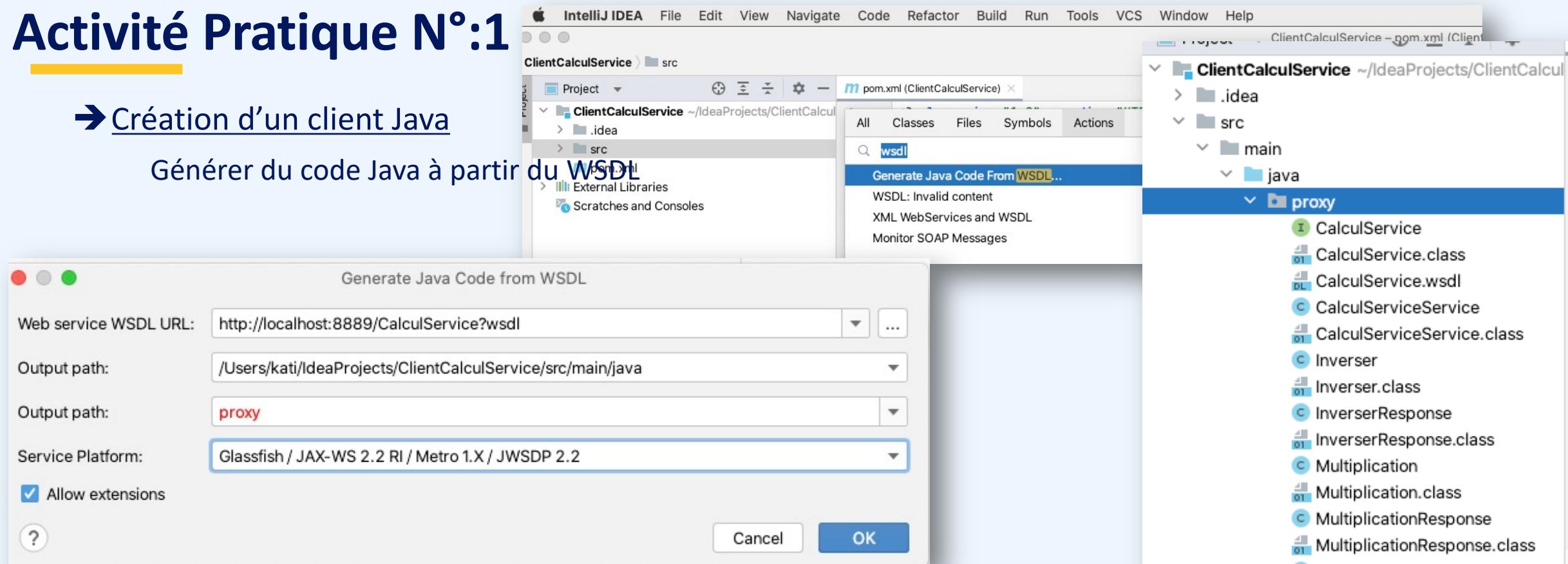
Installation du plugin JAX-WS (*Optional*)



Activité Pratique N°:1

→ Création d'un client Java

Générer du code Java à partir du **WSDL**



Activité Pratique N°:1

→ Création d'un client Java

Création du code client

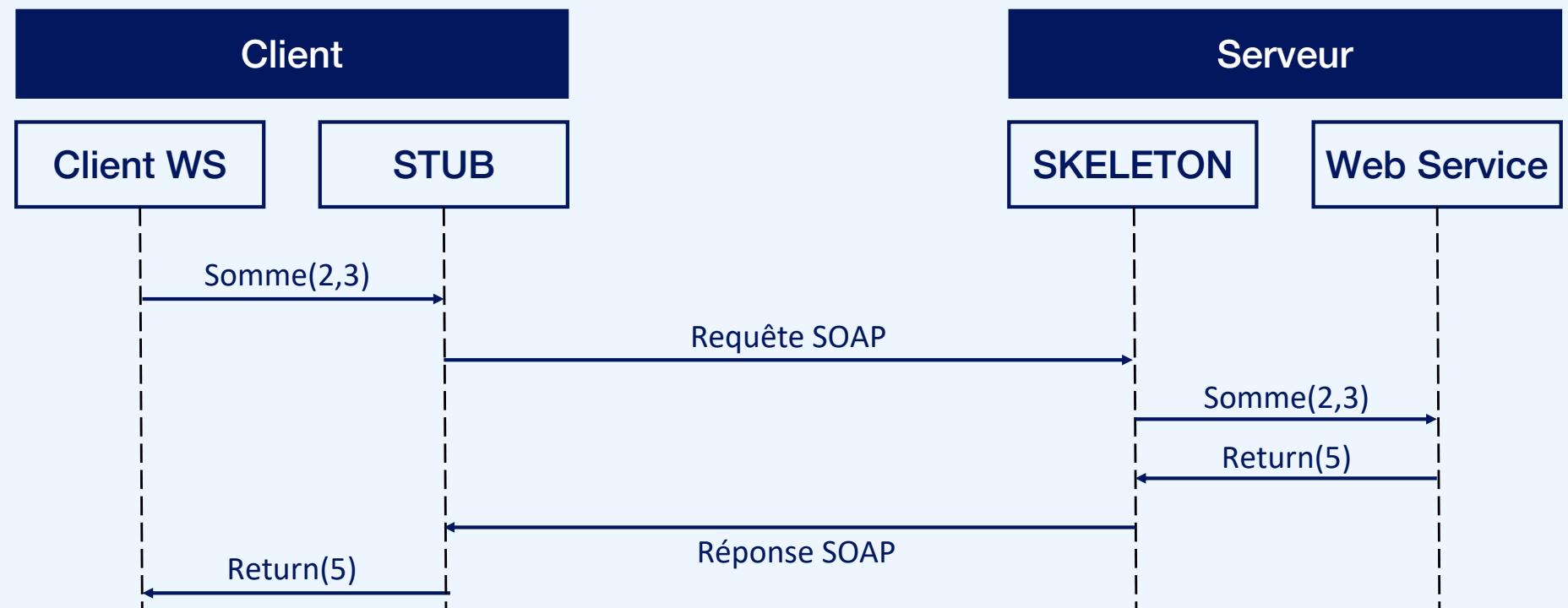
```
import proxy.CalculService;
import proxy.CalculServiceService;

public class ClientWS {
    public static void main(String[] args) {
        CalculService stub = new CalculServiceService().getCalculServicePort();
        System.out.println(stub.somme(2,3));
        System.out.println(stub.multiplication(5,3));
        System.out.println(stub.inverser( arg0: 2022));
    }
}
```



Activité Pratique N°:1

→ Architecture



Activité supplémentaire

- ✓ Créer un Web Service qui permet de:
 - Récupérer l'ensemble des étudiants dans la base de données
 - Récupérer un nombre limité des étudiants (à préciser en paramètre)
 - Récupérer un étudiant à partir de son CIN

Un étudiant est représenté par: ID, nom complet, CIN, date de naissance, filière

Tester le WS par SOAP UI et créer un client (Java ou autre) qui pourra consommer ce WS.

