|  |
| --- |
|  |
| Compte rendu Atelier SOA |
|  |

|  |
| --- |
| Wassim Dridi && Mohamed Marzouk |

[Atelier SOA - 01 Configuration de l’environnement de développement 5](#_Toc184200553)

[1. Utiliser les commandes de base de Maven pour gérer le cycle de vie du projet 5](#_Toc184200554)

[1.1 Commande «clean» : 5](#_Toc184200555)

[1.2 Commande «validate» 6](#_Toc184200556)

[1.3 Commande «compile» 6](#_Toc184200557)

[1.4 Commande «test» : 7](#_Toc184200558)

[1.5 Commande « package » 8](#_Toc184200559)

[1.6 Commande «install» 10](#_Toc184200560)

[1.7 Commande «site» 11](#_Toc184200561)

[1.8 Gérer le cycle de vie du projet dans Graphiquement 12](#_Toc184200562)

[2. Déployer une application web avec Tomcat en utilisant Maven 12](#_Toc184200563)

[2.1 Déployer le fichier WAR d’un dépôt local 12](#_Toc184200564)

[Développer un service Web SOAP avec JAX-WS 13](#_Toc184200565)

[1. Créer le service WEB 13](#_Toc184200566)

[1.1 Publier le WS à partir d’un serveur JAVA 13](#_Toc184200567)

[1.2 ajouter la dépendance 14](#_Toc184200568)

[1.3 Interroger le serveur via un navigateur Web 15](#_Toc184200569)

[Tester un service web avec Oxygen et SoapUI 16](#_Toc184200570)

[1. Tester les méthodes du SW via Oxygen XML 16](#_Toc184200571)

[1.1 Demarage de clé d’evaluation 16](#_Toc184200572)

[1.2 le lancement des methodes du SW 16](#_Toc184200573)

[2. Tester les méthodes d’un service web via SOAPUI 18](#_Toc184200574)

[2.1 le wsdl du service web 18](#_Toc184200575)

[2.2 les tests des methodes 18](#_Toc184200576)

[Développer et déployer un service web REST avec JERSEY et tomcat 23](#_Toc184200577)

[1. Déployer le service web en utilisant le plugin tomcat 23](#_Toc184200578)

[1.2 affichage les caractères spéciaux 23](#_Toc184200579)

[2. Développement d'un Service Web REST pour la Gestion d'Objets 23](#_Toc184200580)

[2.1 Appeler la méthode « ajouterCompte » du service web. 23](#_Toc184200581)

[2.2 Appeler la méthode « getCompte » du service web. 24](#_Toc184200582)

[2.3 Appeler la méthode « getComptes » du service web. 24](#_Toc184200583)

[2.4 Appeler la méthode « getListeComptes » du service web. 24](#_Toc184200584)

[2.5 Appeler la méthode « supprimerCompte » du service web. 25](#_Toc184200585)

[3. Creation une nouvelle méthode « modifierCompte » 26](#_Toc184200586)

[1.1 l’exécution de méthode modifierCompte 26](#_Toc184200587)

[**Projet De Gestion Des Produits** 27](#_Toc184200588)

[1. Développement de base de données 27](#_Toc184200589)

[1.1 Conception de la base de données "SOA" avec une table utilisateurs 27](#_Toc184200590)

[2. Développement d’application serveur 27](#_Toc184200591)

[2.1 Configuration d’un Projet pour le Développement et le Déploiement de Services Web REST avec Tomcat7 27](#_Toc184200592)

[2.1.1 Configuration de la Servlet JERSEY pour les Services Web REST 28](#_Toc184200593)

[2.2 Création des Composants de l'Application REST : Modèle, Réponse,DBconnexion,TestUnitaire et Services 28](#_Toc184200594)

[2.2.1 Création de la classe DBConnexion pour gérer la connexion à la base de données SOA 28](#_Toc184200595)

[2.2.2 Création de la classe Personne pour modéliser les entités dans le système SOA 30](#_Toc184200596)

[2.2.3 Création de l'interface PersonneService pour la gestion des entités Personne dans SOA 30](#_Toc184200597)

[2.2.4 Création de la classe PersonneServiceImpl pour implémenter les services RESTful de gestion des entités Personne 30](#_Toc184200598)

[2.2.5 Création de la classe Reponse pour gérer les retours des services REST 31](#_Toc184200599)

[2.2.6 Création de la classe PersonneServiceImplTest pour les tests unitaires 32](#_Toc184200600)

[2.4 Test Manuelle 34](#_Toc184200601)

[2.4.1 Installation et Utilisation de l’Outil ARC pour Tester les Services Web REST 34](#_Toc184200602)

[2.4.2Tester le Service REST avec la Méthode GET pour Récupérer Toutes les Personnes 35](#_Toc184200603)

[2.4.3Ajout d'une Personne via le Path "/add" 35](#_Toc184200604)

[2.4.4 Ajout du Paramètre "Accept" pour Spécifier le Format de Réponse en JSON et Gestion des Erreurs 36](#_Toc184200605)

[2.4.5 Ajout des Dépendances Jackson dans le Fichier "pom.xml" pour Supporter le Format JSON 37](#_Toc184200606)

[2.4.6 Réexécution du Serveur Tomcat7 et Envoi de la Requête pour Ajouter une Nouvelle Personne au Format JSON 37](#_Toc184200607)

[2.4.7 Exécution de la Requête pour Récupérer la Personne avec l'ID "1" via le Path "/{id}/get" 38](#_Toc184200608)

[2.4.8 Suppression d'une Personne avec le Path "/{id}/delete" : Cas Existant et Non Existant 40](#_Toc184200609)

[2.4.9 Mise à Jour d'une Personne via le Path "/update" avec la Méthode PUT 40](#_Toc184200610)

[2.5 Utilisation du Plugin Maven-Jar pour Générer un Fichier JAR à partir d'un Projet WebApp 41](#_Toc184200611)

[2.5.1Publication du Projet dans le Dépôt Local avec la Commande Maven Install 42](#_Toc184200612)

[3. Developper d’application client 43](#_Toc184200613)

[3.1Création d'un Projet Maven pour un Client JSP RESTful 43](#_Toc184200614)

[3.2Configuration des Dépendances pour le Support des Services Web REST 44](#_Toc184200615)

[3.3Configuration de la Connexion à la Ressource « /person » dans le Projet« jax-rs-arc » 45](#_Toc184200616)

[3.4 Appel du Chemin « /add » en POST avec l'Objet « Personne » Instancié" 45](#_Toc184200617)

[3.5 Appel du Chemin « /getAll » en GET pour Récupérer la Liste des Personnes au Format JSON 46](#_Toc184200618)

[3.6 Conversion du Tableau JSON en Liste d'Objets « Personne » avec Gson de Google 46](#_Toc184200619)

[3.7 Placer la Liste d'Objets dans la Session HTTP et Rediriger vers « listPersonnes.jsp » 46](#_Toc184200620)

[3.8 Création du Fichier « listPersonnes.jsp » pour Afficher la Liste des Personnes sous Forme Tabulaire" 46](#_Toc184200621)

[3.9 Modification du Formulaire « formPersonne.jsp » pour l'Édition et l'Ajout d'une Personne : Gestion des Attributs et Modes 48](#_Toc184200622)

[3.10 Gestion des Erreurs dans la Servlet de Formulaire de Personne 49](#_Toc184200623)

[3.11 Réalisation : 49](#_Toc184200624)

[3.11.1Gestion des Erreurs dans le Formulaire 50](#_Toc184200625)

[3.11.2 Affichage et Gestion de la Liste des Personnes 52](#_Toc184200626)

[Figure 1: structure de base de donnee 27](#_Toc184200507)

[Figure 2:Fichier pom.xml 28](#_Toc184200508)

[Figure 3:Fichier web.xml 28](#_Toc184200509)

[Figure 4:Classe DBConnexion 29](#_Toc184200510)

[Figure 5:Classe Personne 30](#_Toc184200511)

[Figure 6:Interface PersonneService 30](#_Toc184200512)

[Figure 7: Classe PersonneServiceImpl 31](#_Toc184200513)

[Figure 8:Classe Reponse 32](#_Toc184200514)

[Figure 9:Fichier pom.xml 33](#_Toc184200515)

[Figure 10:Classe Test 33](#_Toc184200516)

[Figure 11:TestUnitaire 34](#_Toc184200517)

[Figure 12:Excution Tompcat7 34](#_Toc184200518)

[Figure 13:TesteurArc 35](#_Toc184200519)

[Figure 14:TestGetAll 35](#_Toc184200520)

[Figure 15:TestAdd 36](#_Toc184200521)

[Figure 16:Test GetAll avec Json 36](#_Toc184200522)

[Figure 17:Dependency pour support json 37](#_Toc184200523)

[Figure 18:Test ajouter 38](#_Toc184200524)

[Figure 19:Test Get par Id 39](#_Toc184200525)

[Figure 20:Test Supprimer 40](#_Toc184200526)

[Figure 21:Test Modifier 40](#_Toc184200527)

[Figure 22:Plugin de deploiyement 41](#_Toc184200528)

[Figure 23:Mvn Install 42](#_Toc184200529)

[Figure 24:Depot Locale 42](#_Toc184200530)

[Figure 25:Creation partie client 43](#_Toc184200531)

[Figure 26:Configuration REST 44](#_Toc184200532)

[Figure 27:Declaration plugin Tomcat7 44](#_Toc184200533)

[Figure 28:Ajouter classifier 45](#_Toc184200534)

[Figure 29:Configuration de connexion 45](#_Toc184200535)

[Figure 30:Ajouter Personne 45](#_Toc184200536)

[Figure 31 :Recuperer la liste des personnes 46](#_Toc184200537)

[Figure 32:Analyse en java 46](#_Toc184200538)

[Figure 33:Passer la liste vers jsp 46](#_Toc184200539)

[Figure 34:Recuperer en jsp 47](#_Toc184200540)

[Figure 35:Ajout les liens hypertextes 47](#_Toc184200541)

[Figure 36:Suppression d'une Personne 47](#_Toc184200542)

[Figure 37:Traitement de l'Action « editer » 48](#_Toc184200543)

[Figure 38: Edition et Ajout d'une personne 48](#_Toc184200544)

[Figure 39:Traitement du Mode « edit » 49](#_Toc184200545)

[Figure 40:Gestion des Erreurs 49](#_Toc184200546)

[Figure 41:Architecture de partie web 50](#_Toc184200547)

[Figure 42:Page d'Accueil 50](#_Toc184200548)

[Figure 43:1Gestion des Erreurs 51](#_Toc184200549)

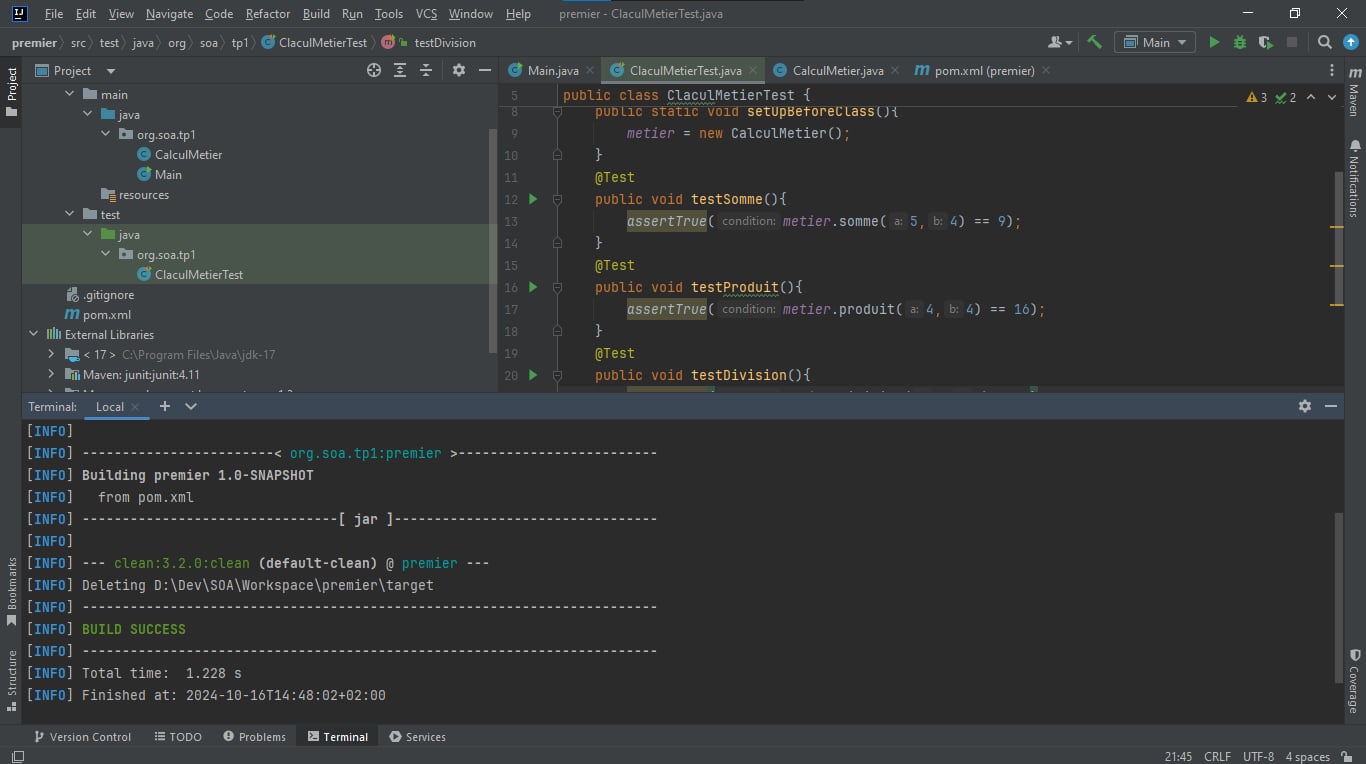
[Figure 44:l'effectation de l'ajout 52](#_Toc184200550)

[Figure 45:Affichage la liste de personne 52](#_Toc184200551)

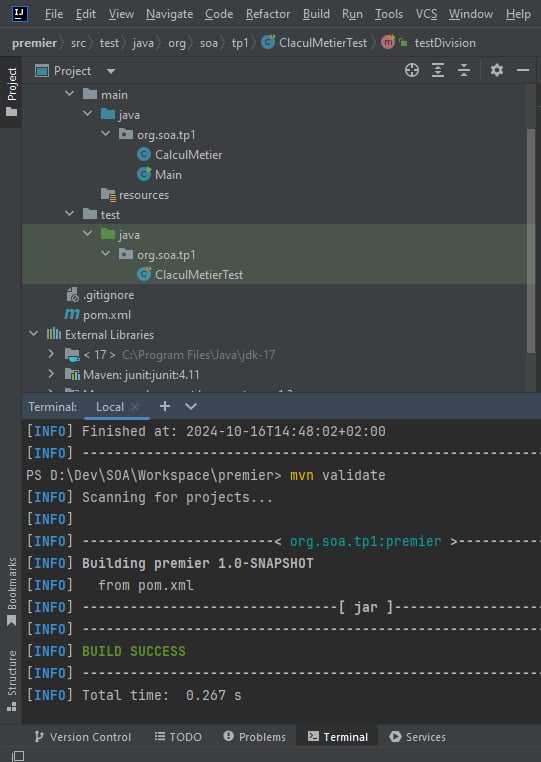
# Atelier SOA - 01 Configuration de l’environnement de développement

## Utiliser les commandes de base de Maven pour gérer le cycle de vie du projet

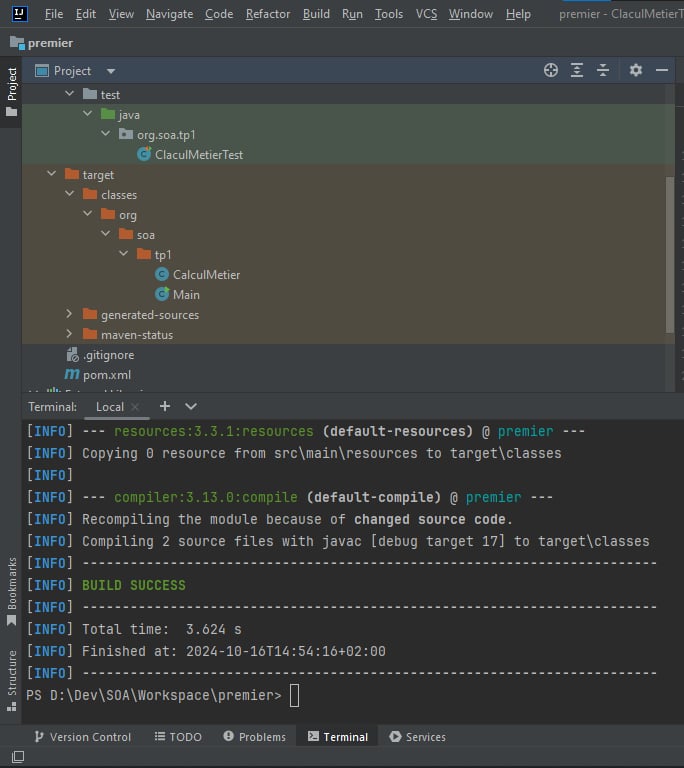
### Commande «clean» :



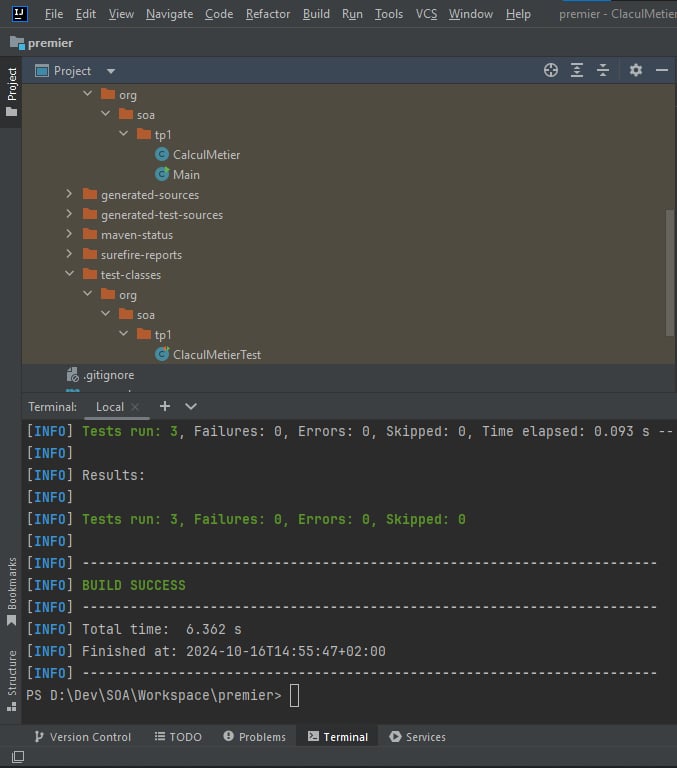
### Commande «validate»



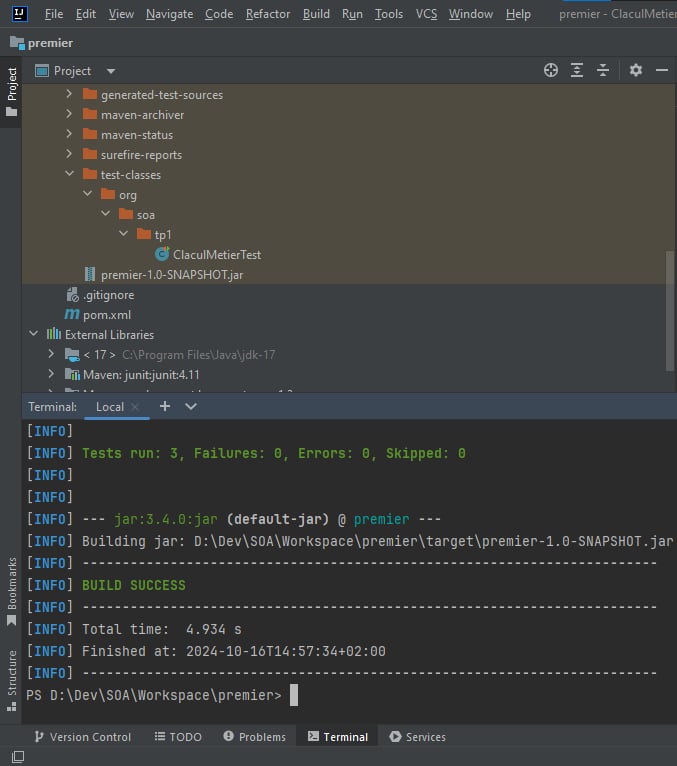
### Commande «compile»



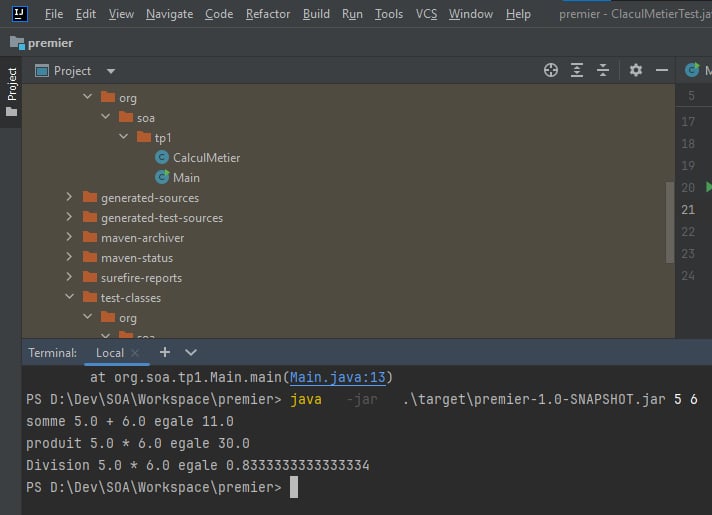
### Commande «test» :



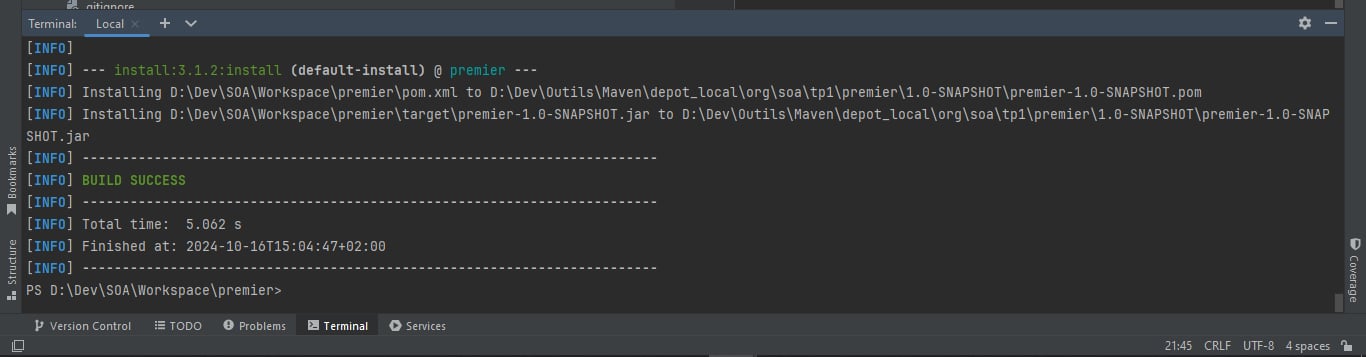
### Commande « package »

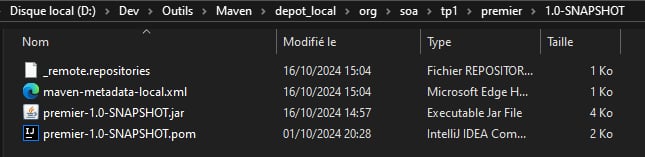


l’exécution du programme à travers le fichier d’archive

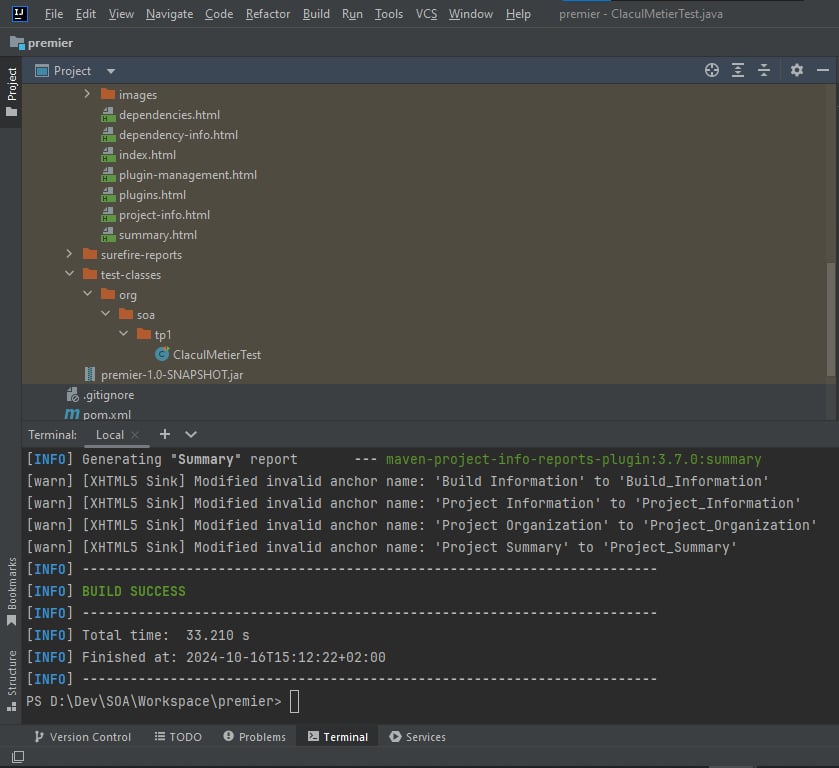


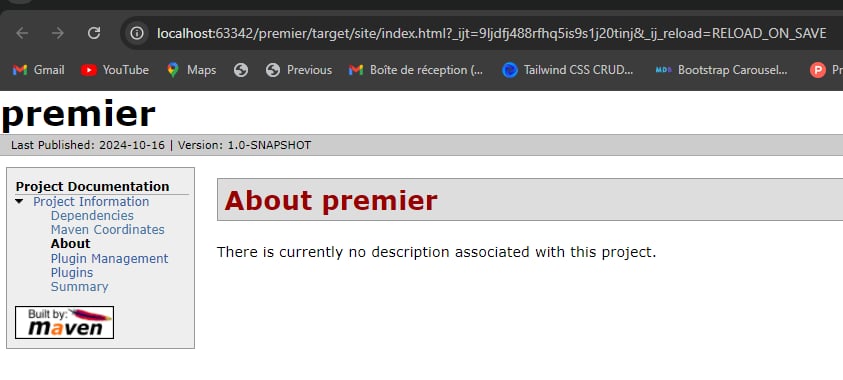
### Commande «install»



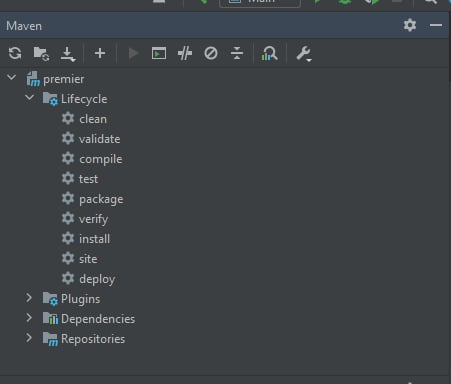


### Commande «site»



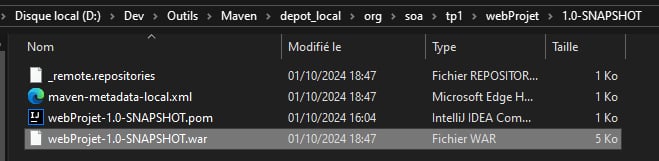


### Gérer le cycle de vie du projet dans Graphiquement



## Déployer une application web avec Tomcat en utilisant Maven

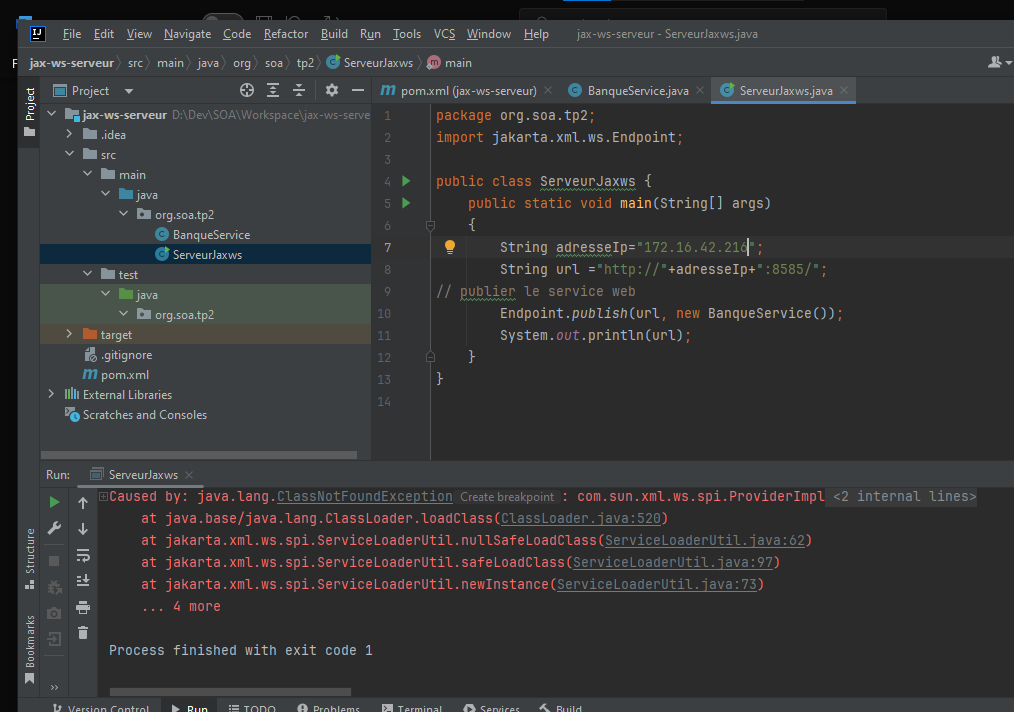
### Déployer le fichier WAR d’un dépôt local



# Développer un service Web SOAP avec JAX-WS

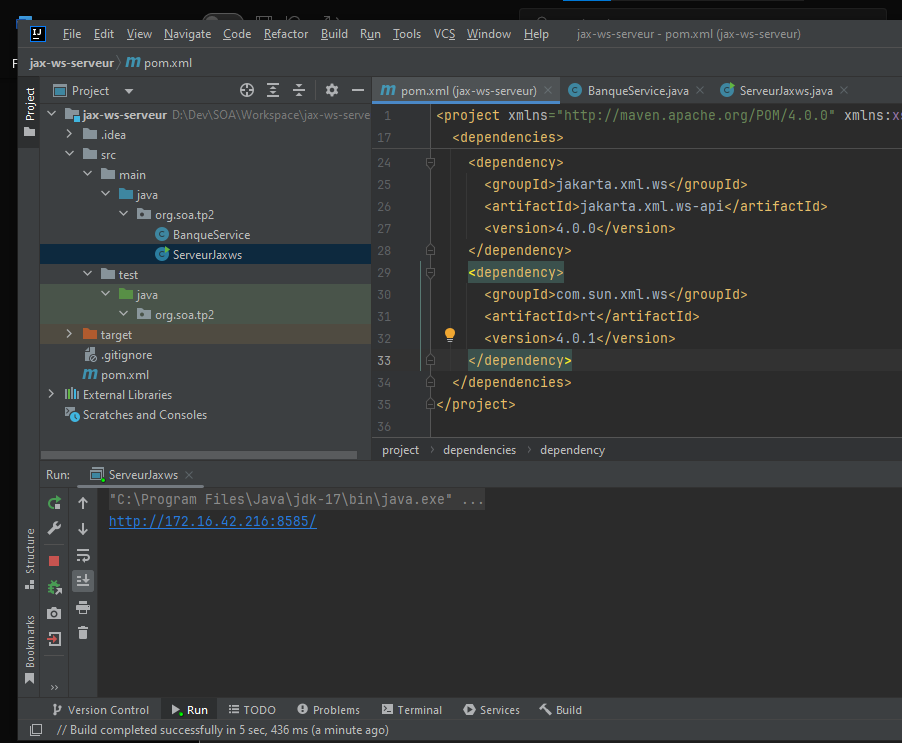
## Créer le service WEB

### 1.1 Publier le WS à partir d’un serveur JAVA



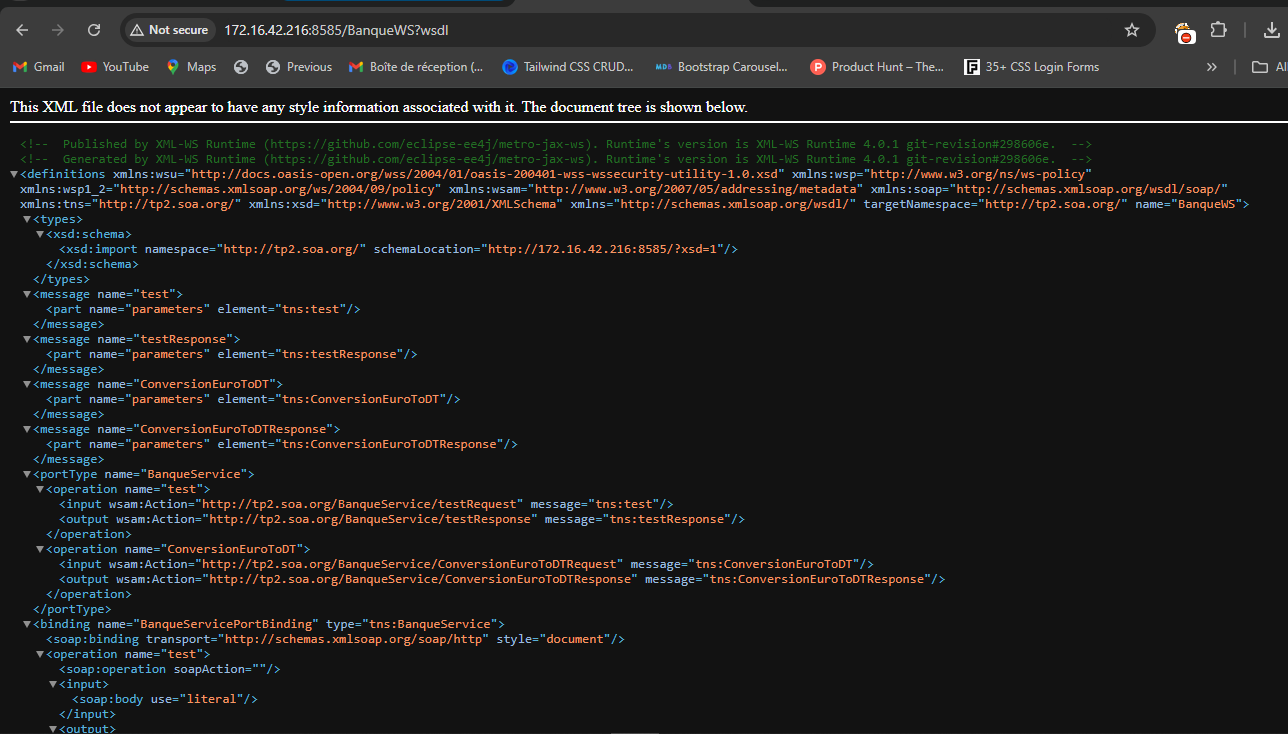
### 1.2 ajouter la dépendance

### 



### 1.3 Interroger le serveur via un navigateur Web

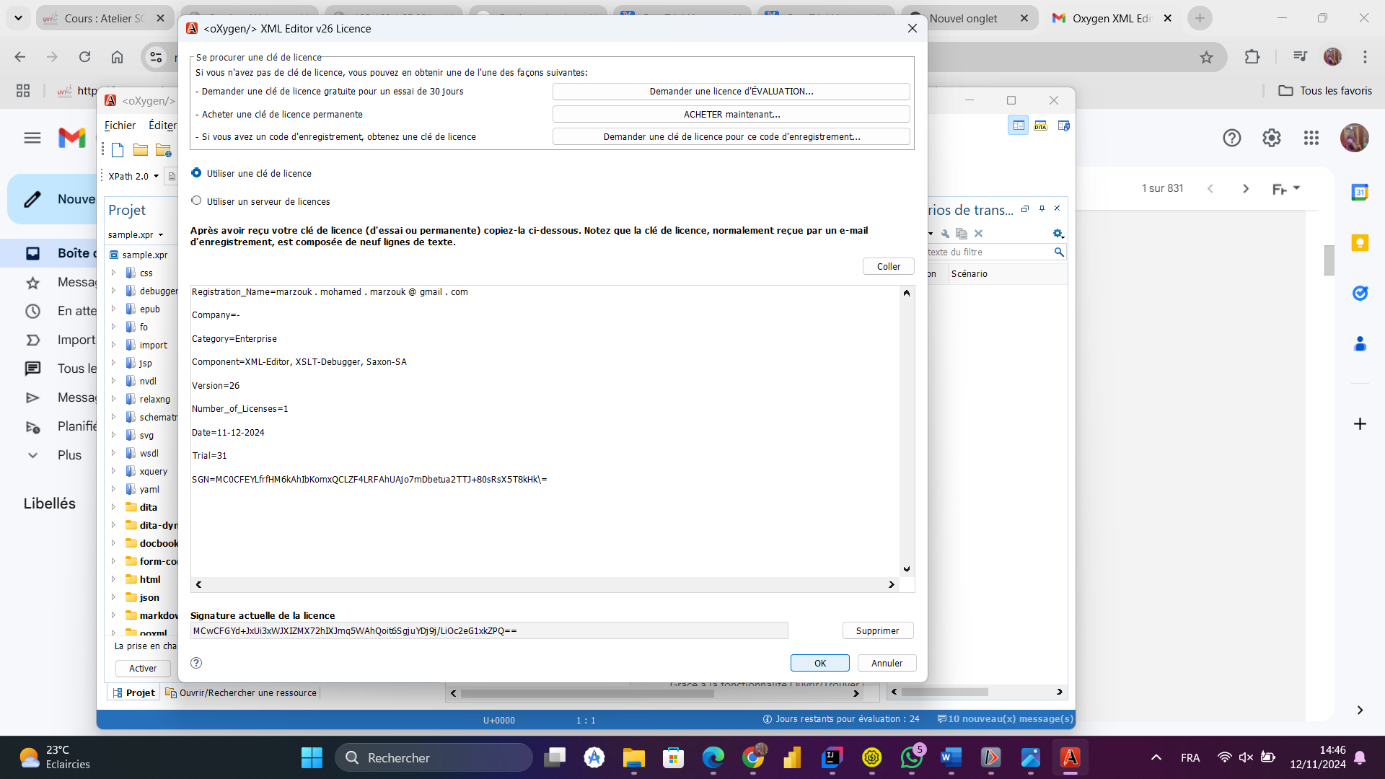
### 



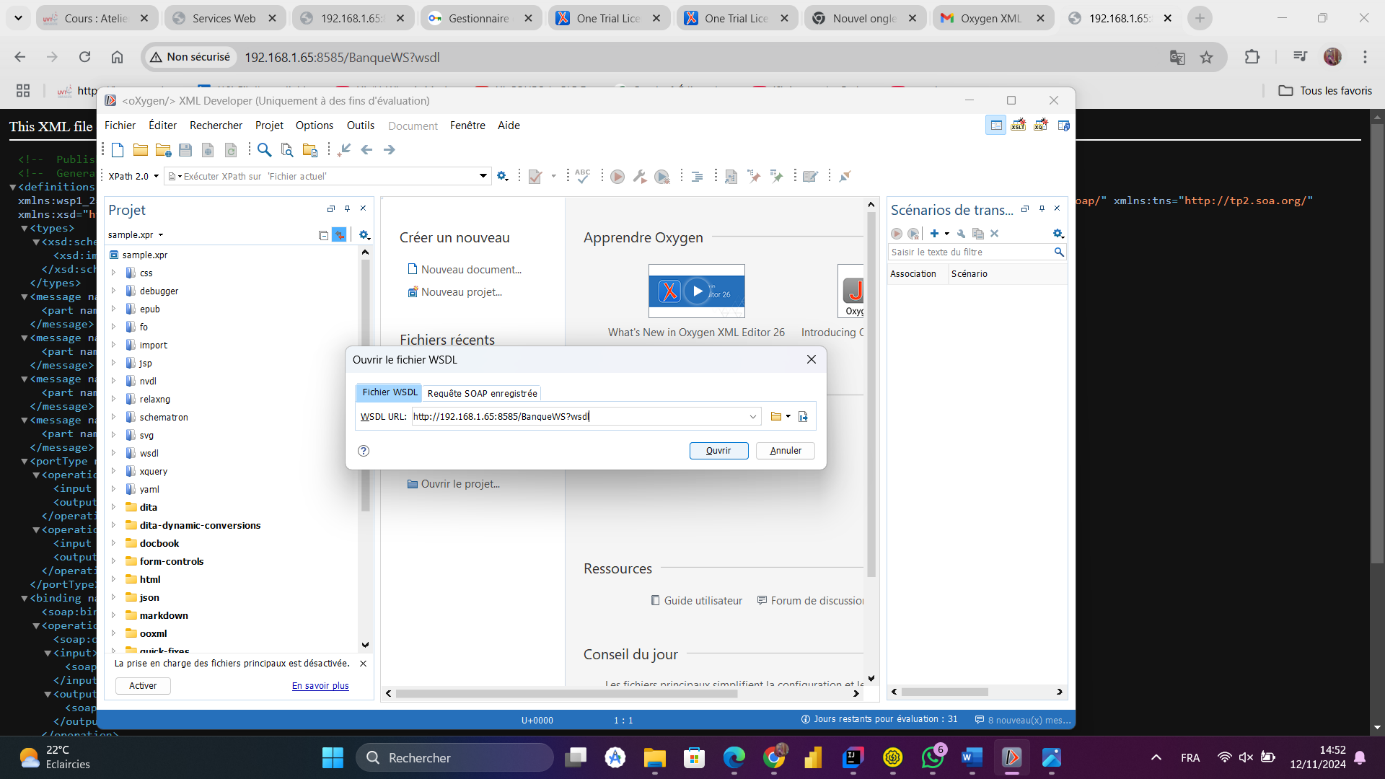
# Tester un service web avec Oxygen et SoapUI

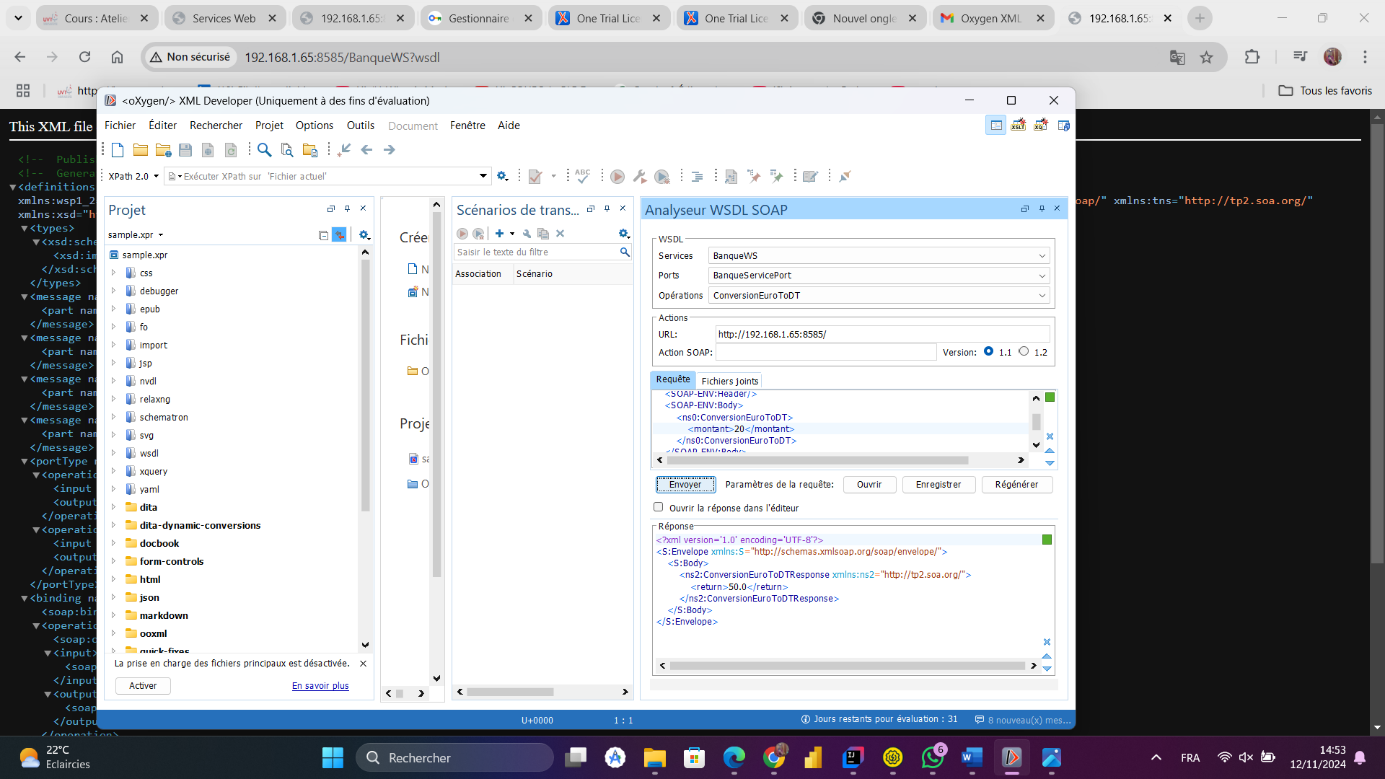
## Tester les méthodes du SW via Oxygen XML

### 1.1 Demarage de clé d’evaluation



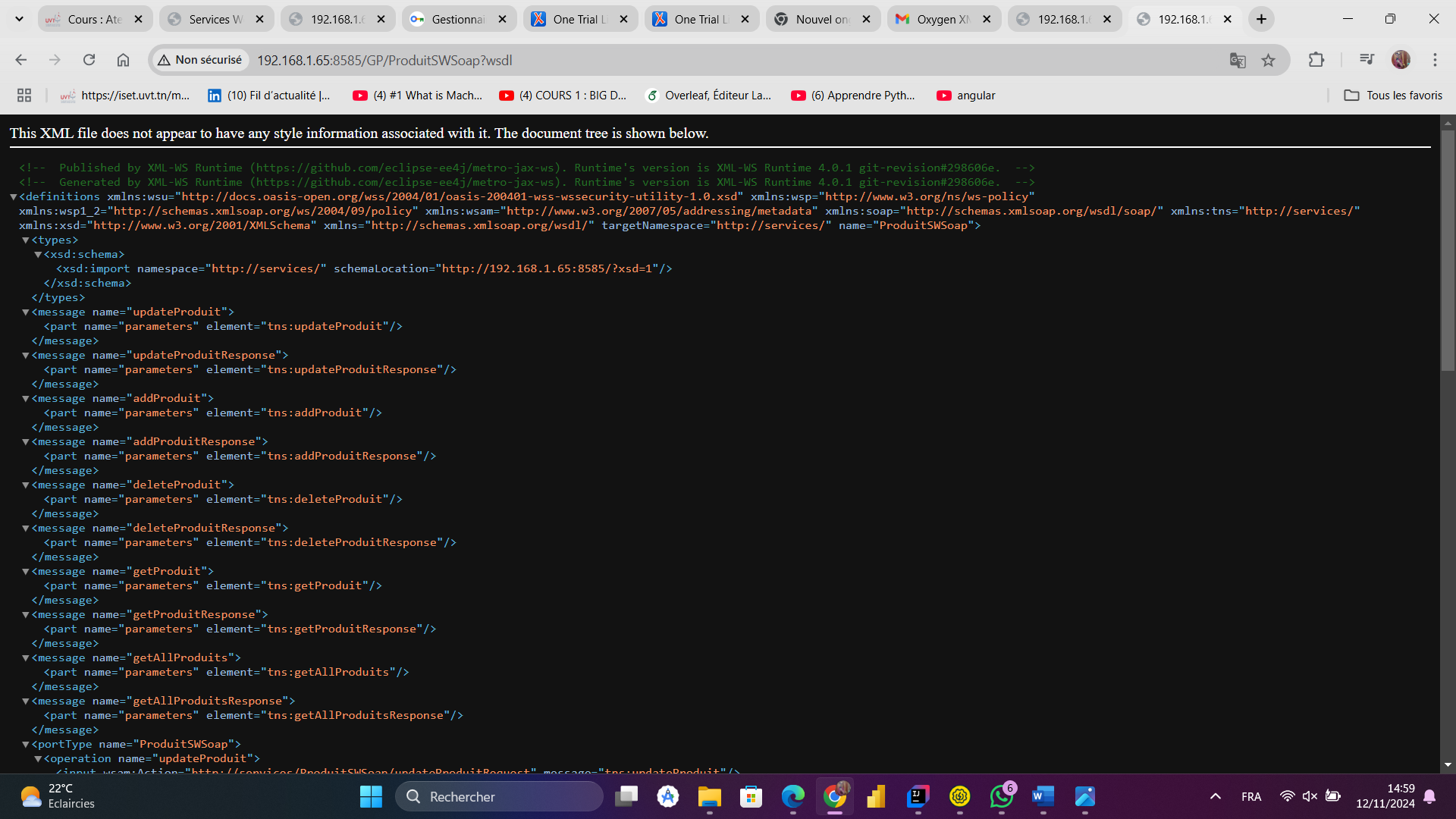
### le lancement des methodes du SW



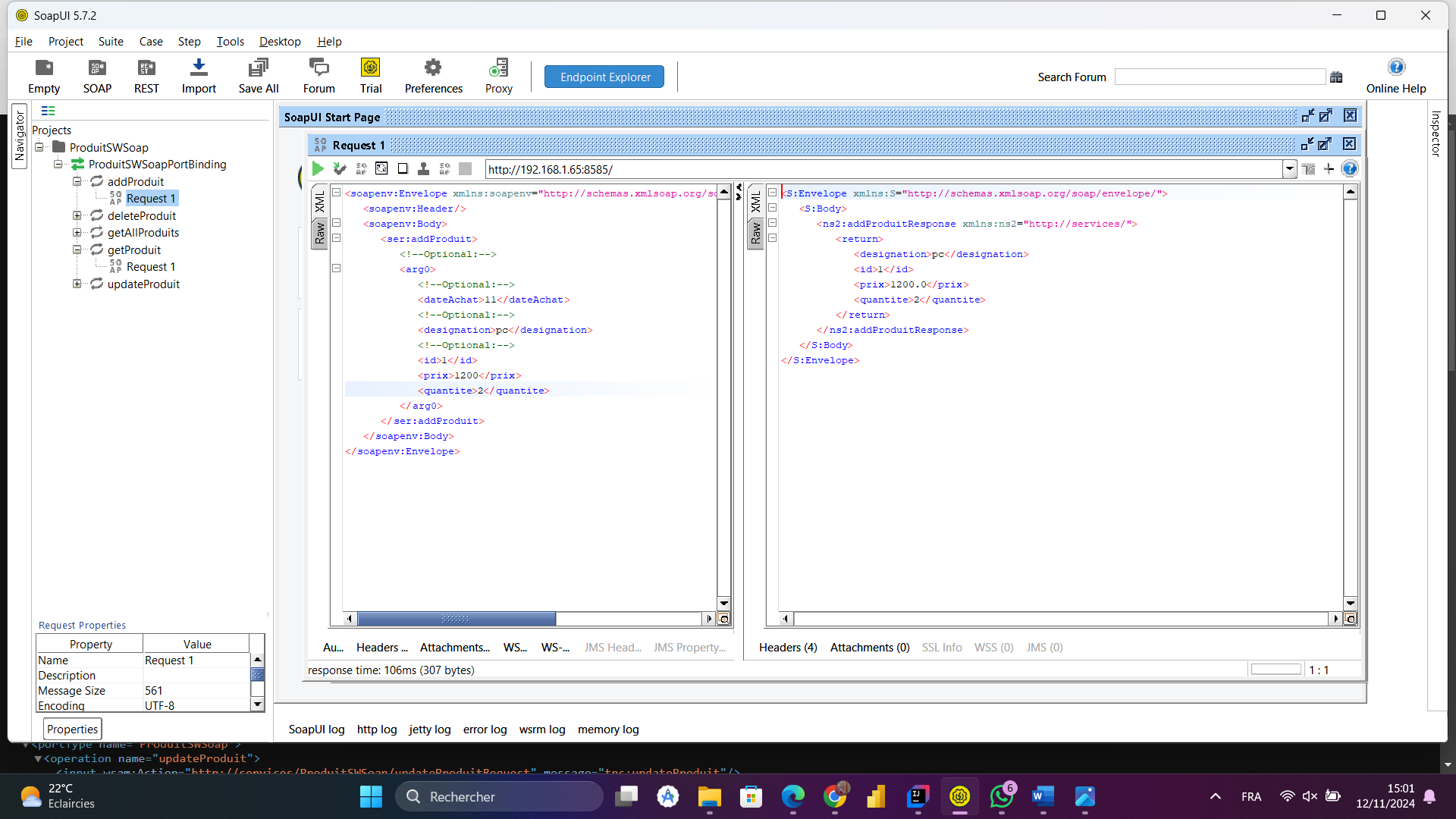


## Tester les méthodes d’un service web via SOAPUI

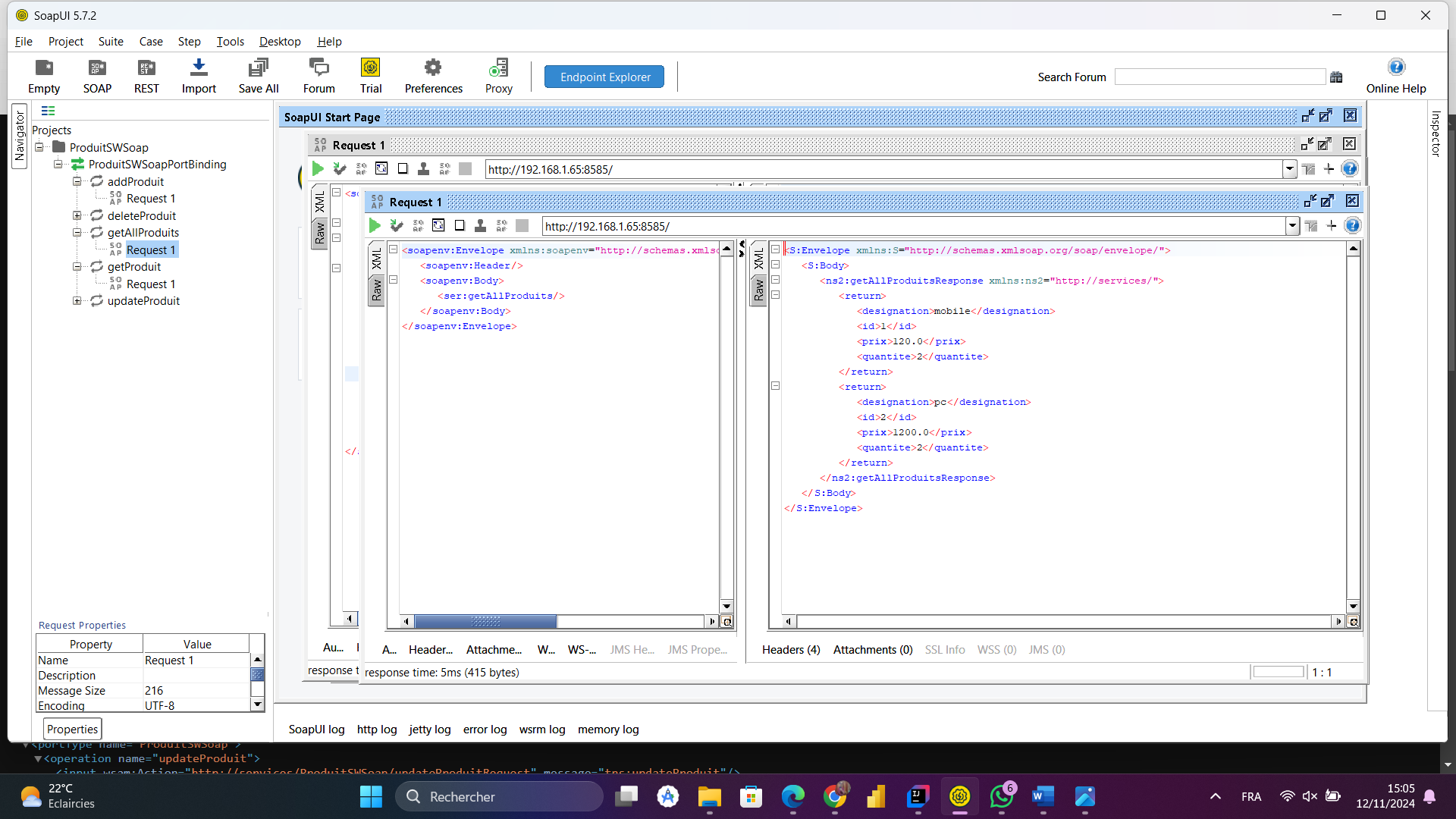
### 2.1 le wsdl du service web

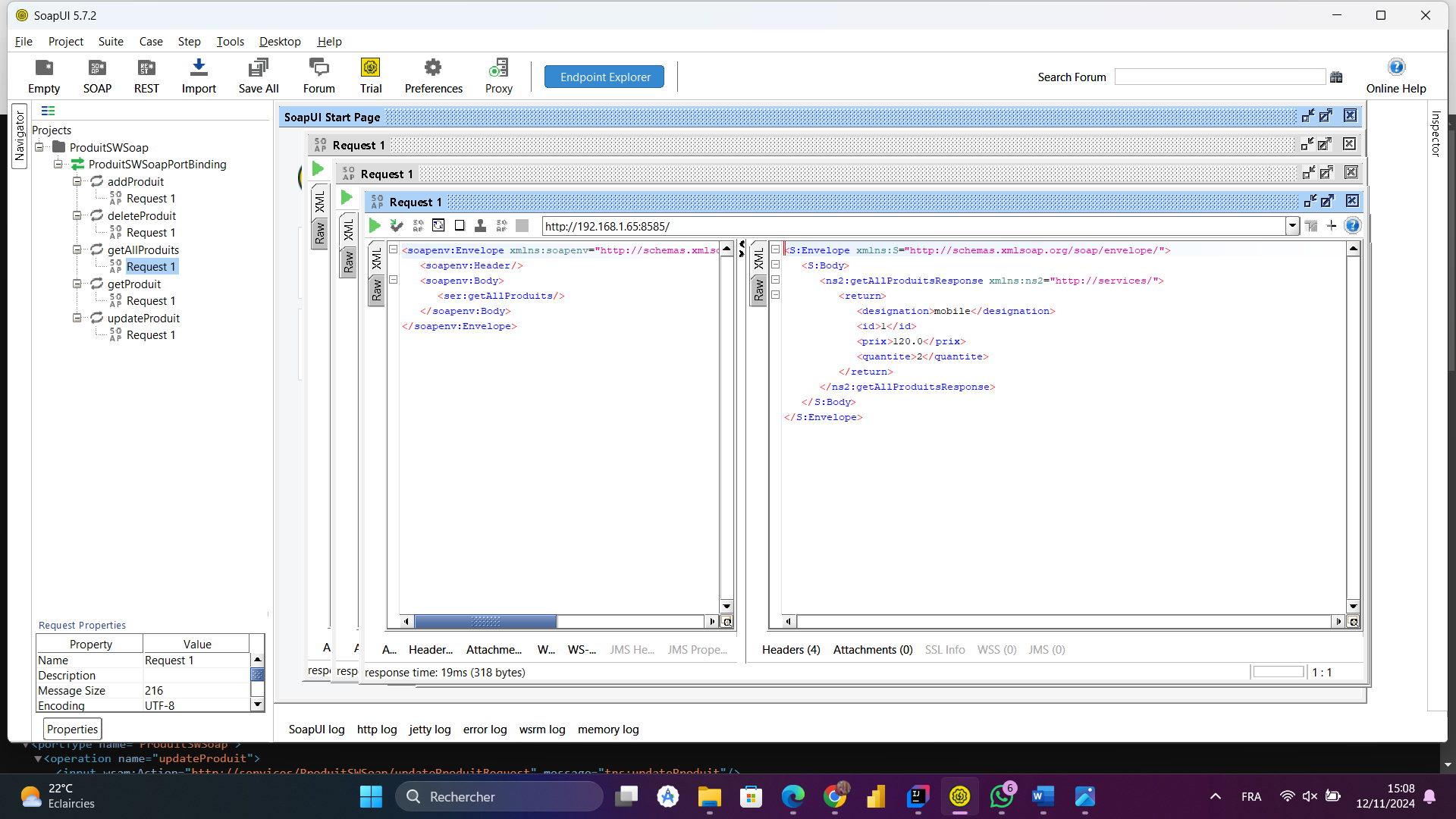
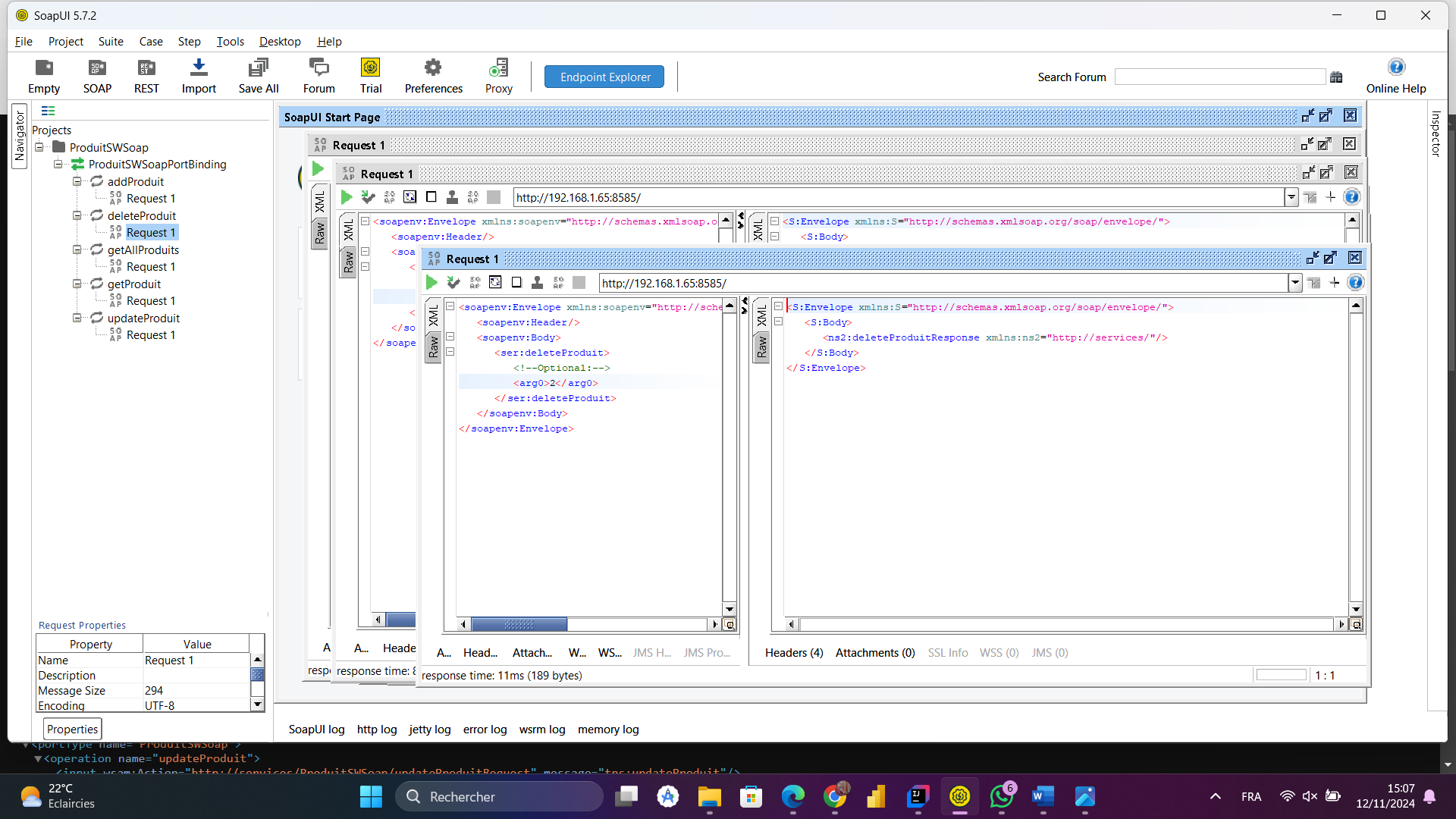
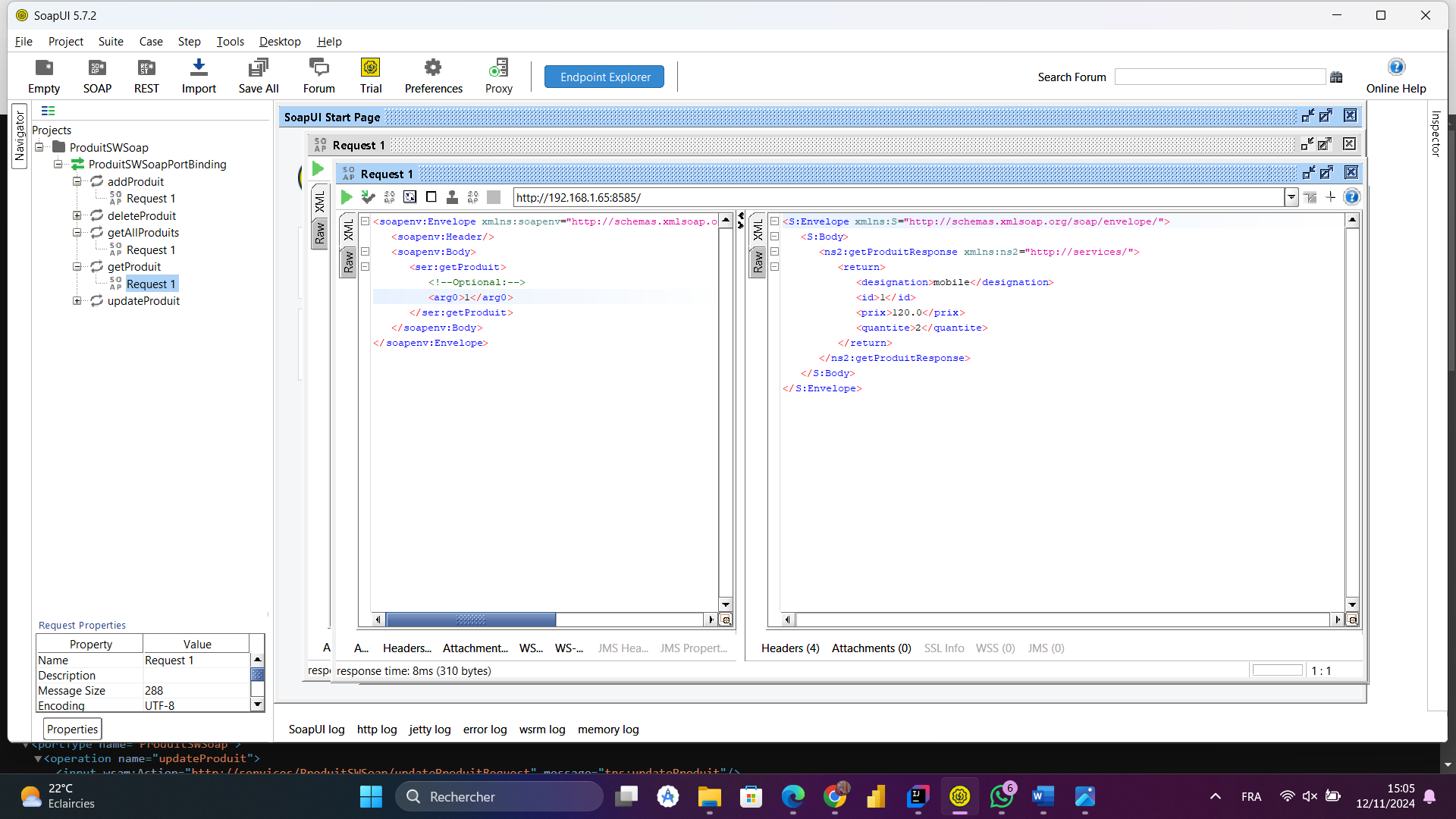


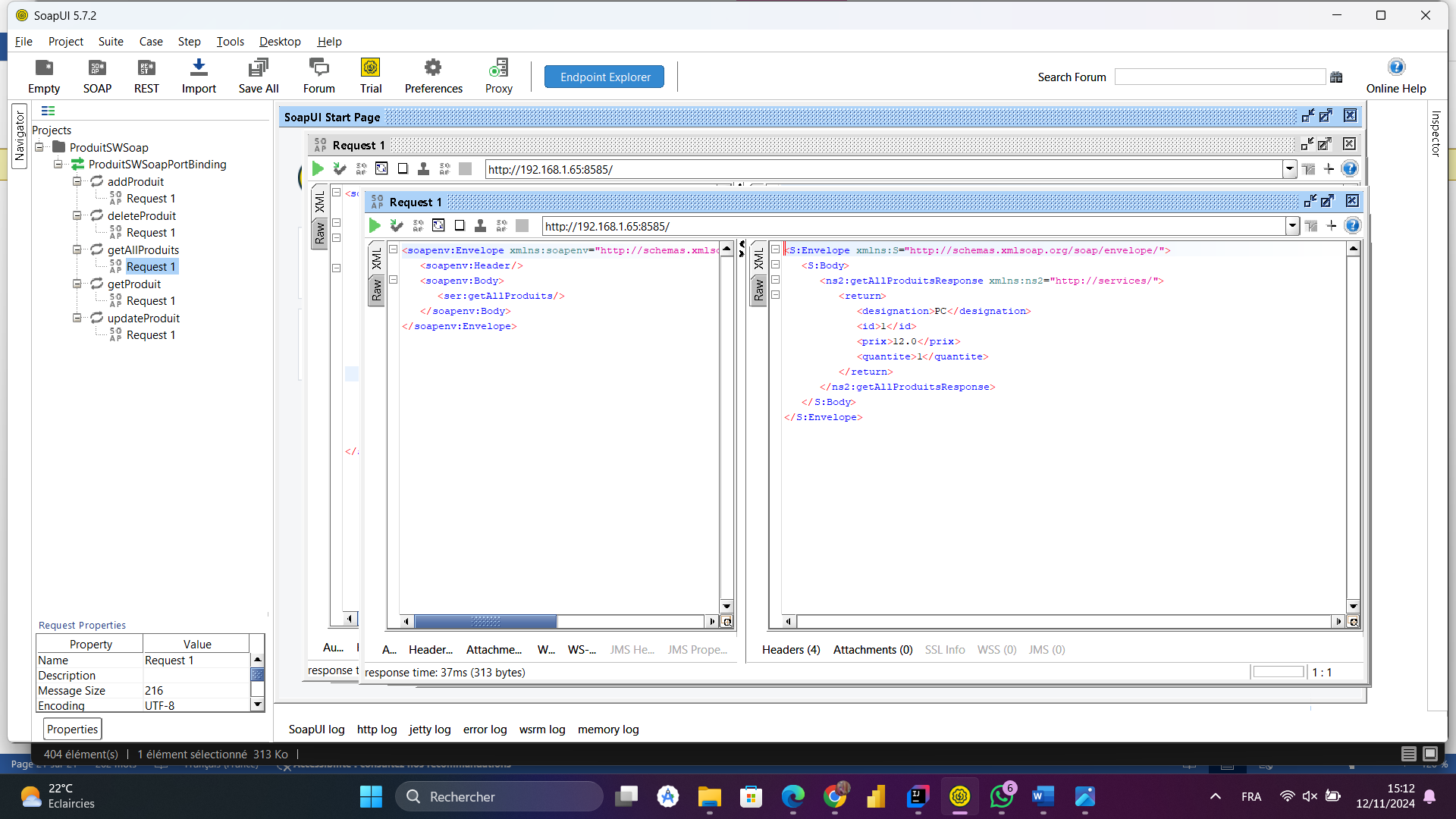
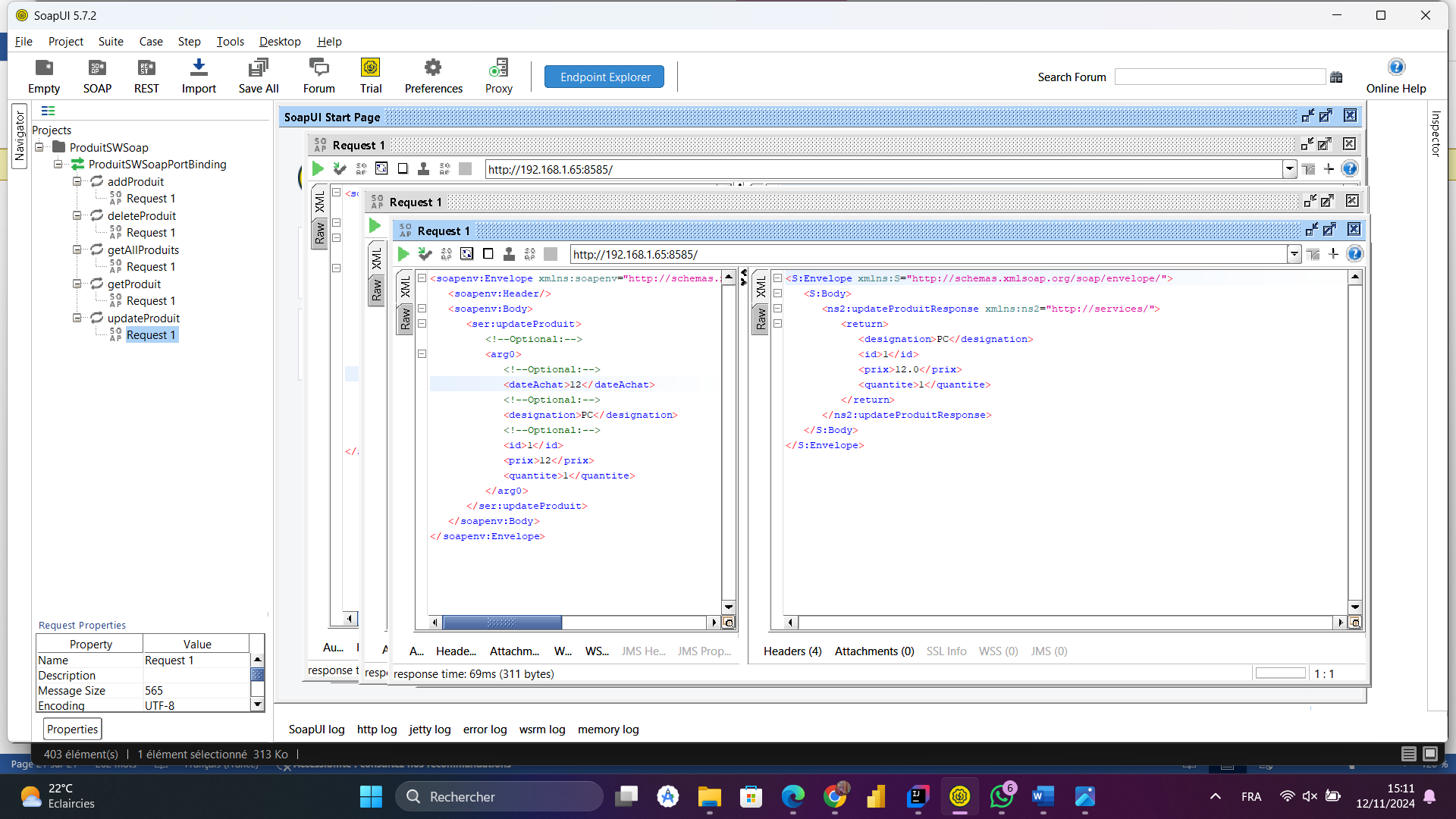
### 2.2 les tests des methodes





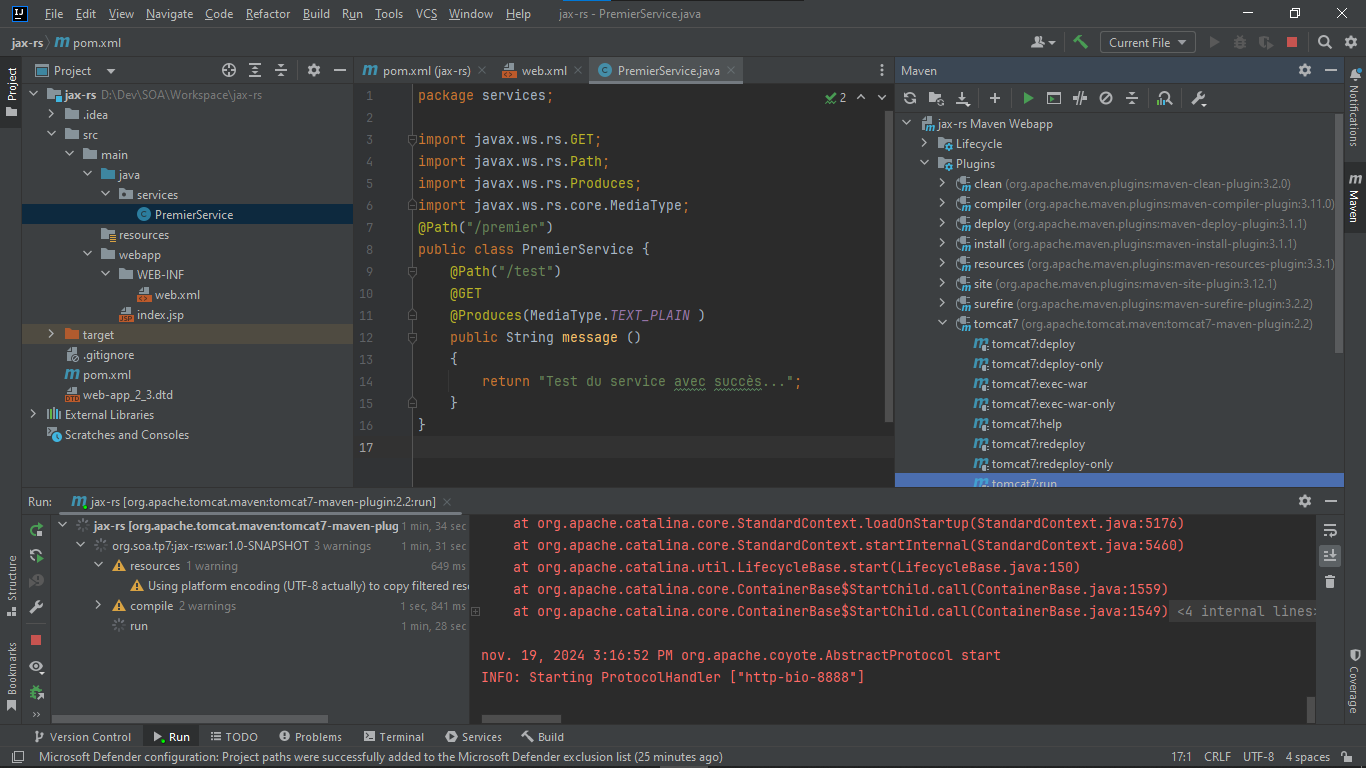


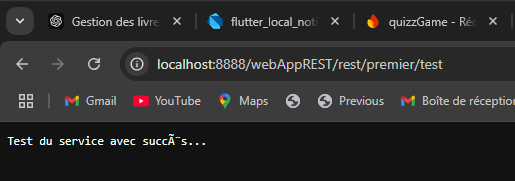




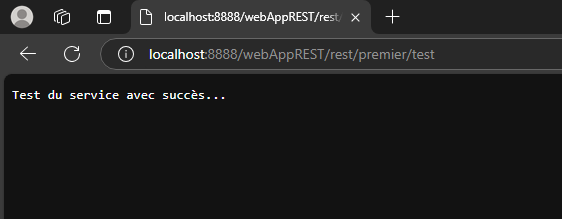
# Développer et déployer un service web REST avec JERSEY et tomcat

## Déployer le service web en utilisant le plugin tomcat



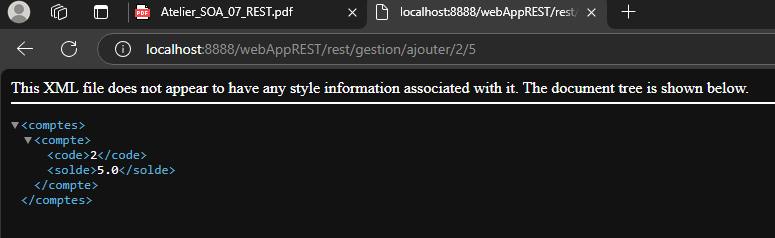
* 1. Premier résultat
  2. 

### 1.2 affichage les caractères spéciaux

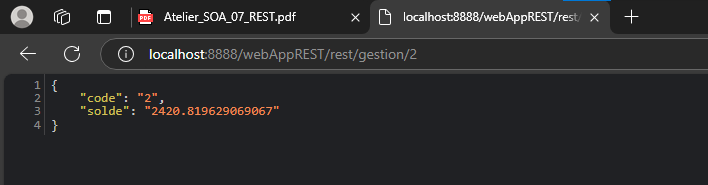


## Développement d'un Service Web REST pour la Gestion d'Objets

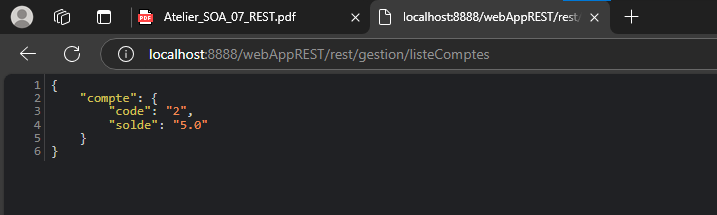
### 2.1 Appeler la méthode « ajouterCompte » du service web.



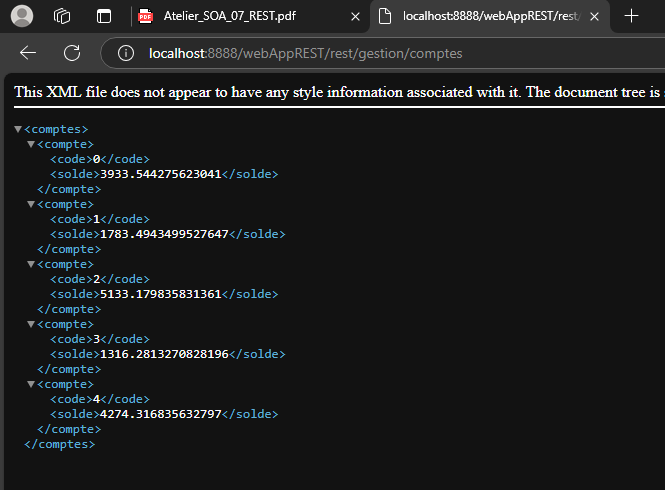
### 2.2 Appeler la méthode « getCompte » du service web.



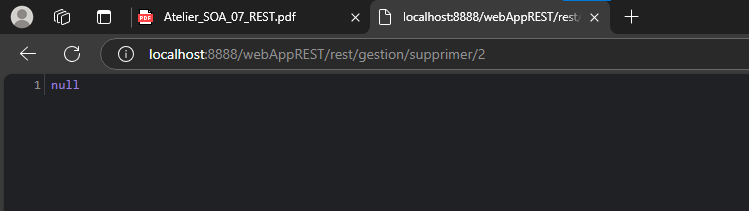
### 2.3 Appeler la méthode « getComptes » du service web.



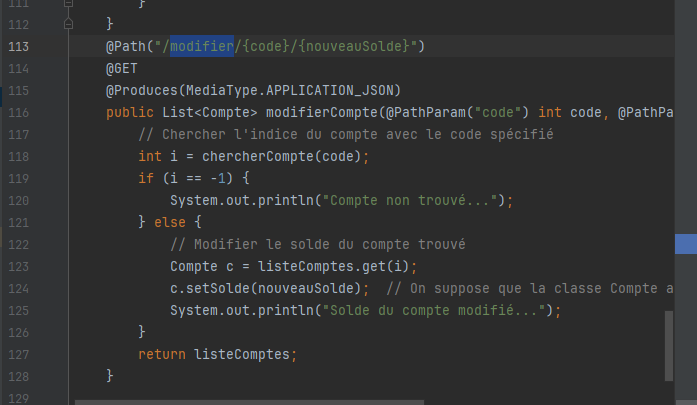
### 2.4 Appeler la méthode « getListeComptes » du service web.



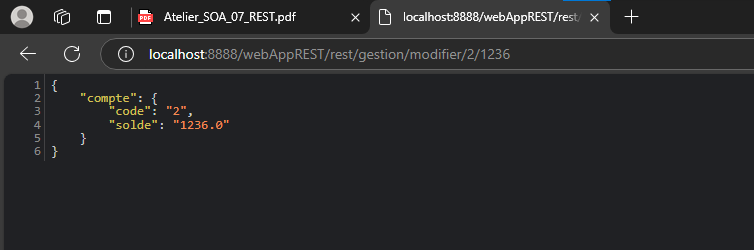
### 2.5 Appeler la méthode « supprimerCompte » du service web.



## Creation une nouvelle méthode « modifierCompte »



### 1.1 l’exécution de méthode modifierCompte



# **Projet De Gestion Des Produits**

## Développement de base de données

### Conception de la base de données "SOA" avec une table utilisateurs

La base de données SOA a été créée pour stocker et gérer les informations des utilisateurs, avec une table principale contenant les attributs suivants : id, nom, et âge



Figure : structure de base de donnee

## Développement d’application serveur

### 2.1 Configuration d’un Projet pour le Développement et le Déploiement de Services Web REST avec Tomcat7

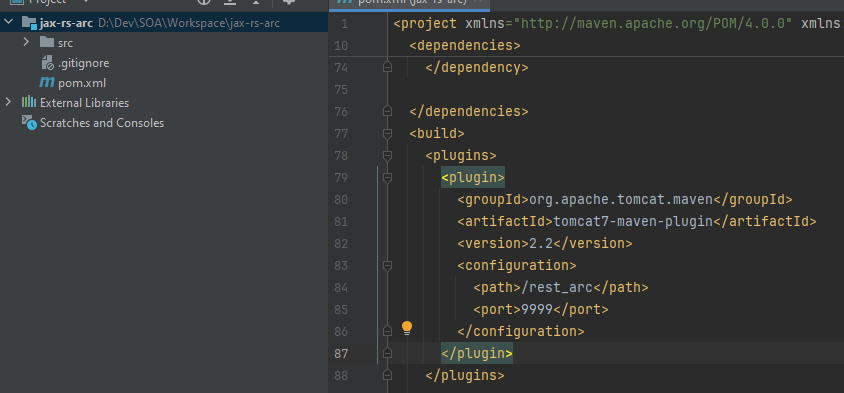


Figure :Fichier pom.xml

### 2.1.1 Configuration de la Servlet JERSEY pour les Services Web REST

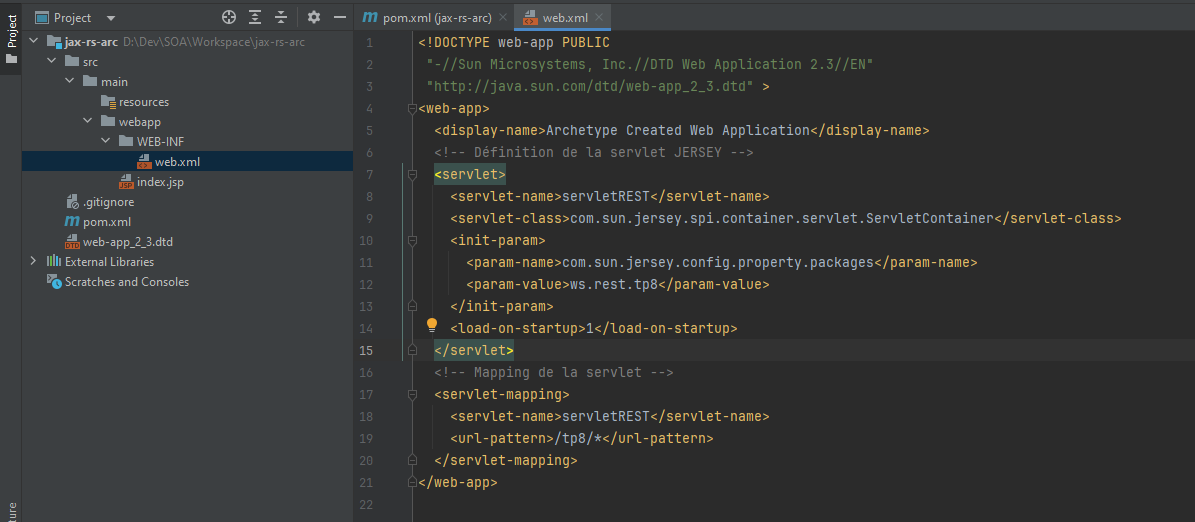


Figure :Fichier web.xml

### Création des Composants de l'Application REST : Modèle, Réponse,DBconnexion,TestUnitaire et Services

#### Création de la classe DBConnexion pour gérer la connexion à la base de données SOA

Méthode getConnection()

Cette méthode retourne l'objet de connexion statique, permettant aux autres parties de l'application d'accéder à la base de données SOA.

Méthode setConnection(Connection conn)

Cette méthode permet de remplacer la connexion existante par une nouvelle connexion passée en paramètre.

Bloc static

Le bloc static initialise la connexion à la base de données en chargeant le driver MySQL et en établissant une connexion avec les paramètres spécifiés. En cas d'échec, une exception est capturée et affichée.

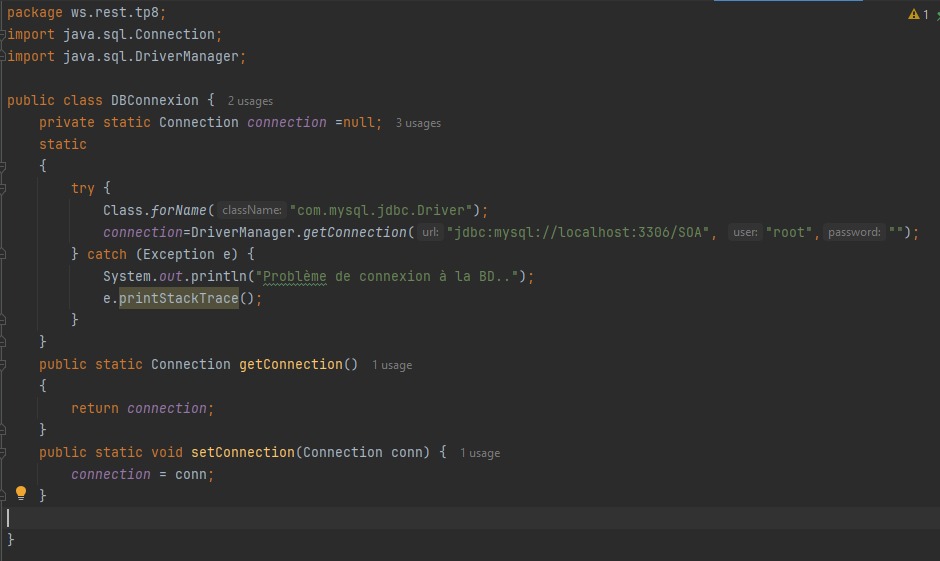


Figure :Classe DBConnexion

#### 2.2.2 Création de la classe Personne pour modéliser les entités dans le système SOA

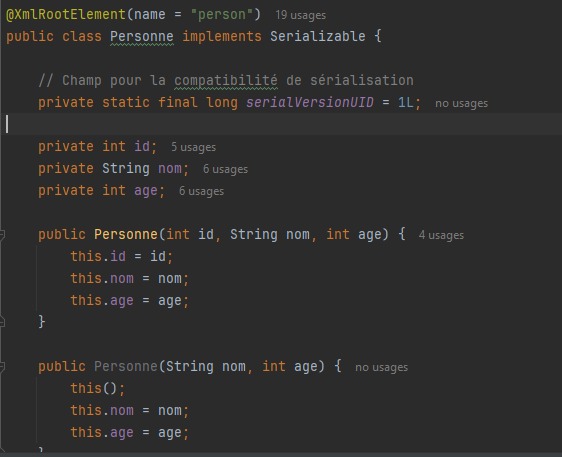


Figure :Classe Personne

#### 2.2.3 Création de l'interface PersonneService pour la gestion des entités Personne dans SOA



Figure :Interface PersonneService

#### 2.2.4 Création de la classe PersonneServiceImpl pour implémenter les services RESTful de gestion des entités Personne

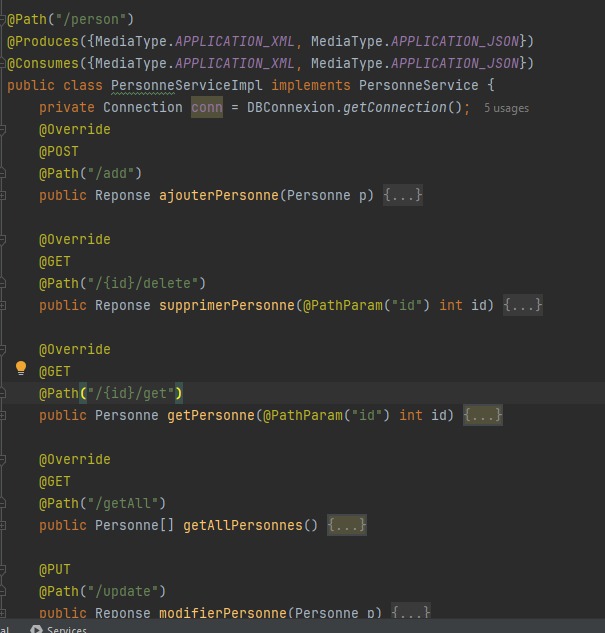


Figure : Classe PersonneServiceImpl

#### 2.2.5 Création de la classe Reponse pour gérer les retours des services REST

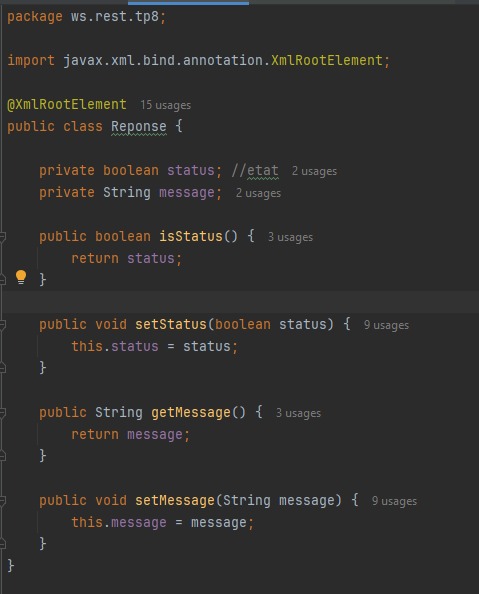


Figure :Classe Reponse

#### 2.2.6 Création de la classe PersonneServiceImplTest pour les tests unitaires

Les dépendances suivantes sont utilisées pour automatiser et exécuter les tests unitaires de la classe PersonneServiceImpl :



Figure :Fichier pom.xml

d’automatiser la procédure de test avec des tests unitaires en utilisant la bibliothèque JUnit et mockito et Maven.

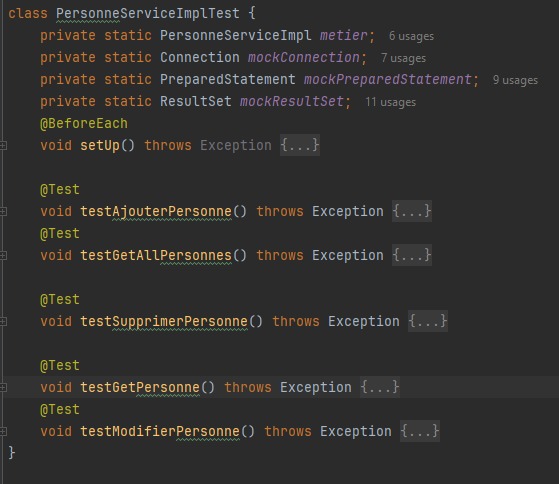


Figure :Classe Test

Remarquer que le test est réalisé avec succès (sans erreur)

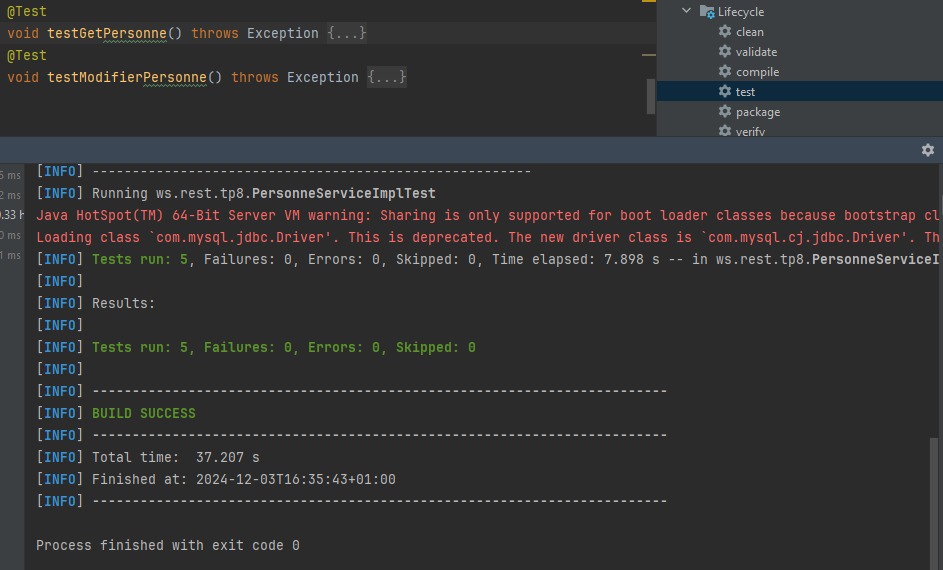


Figure :TestUnitaire

### 2.4 Test Manuelle

Déploiement et Test Initial du Service Web REST

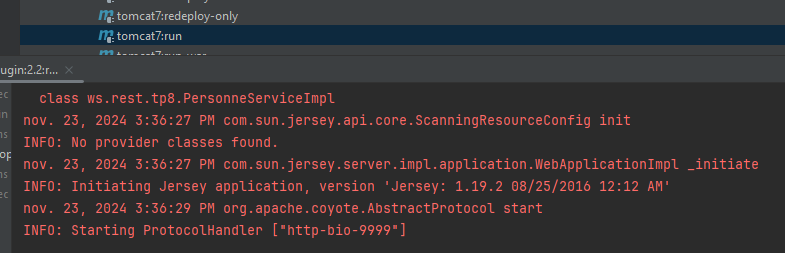


Figure :Excution Tompcat7

#### 2.4.1 Installation et Utilisation de l’Outil ARC pour Tester les Services Web REST

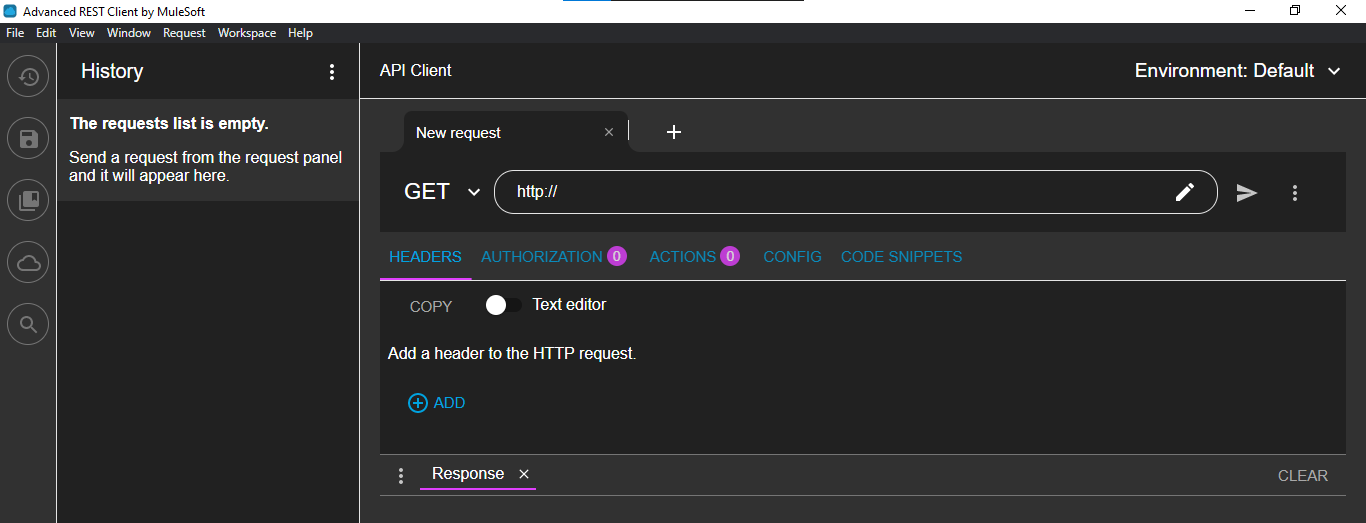


Figure :TesteurArc

#### 2.4.2Tester le Service REST avec la Méthode GET pour Récupérer Toutes les Personnes

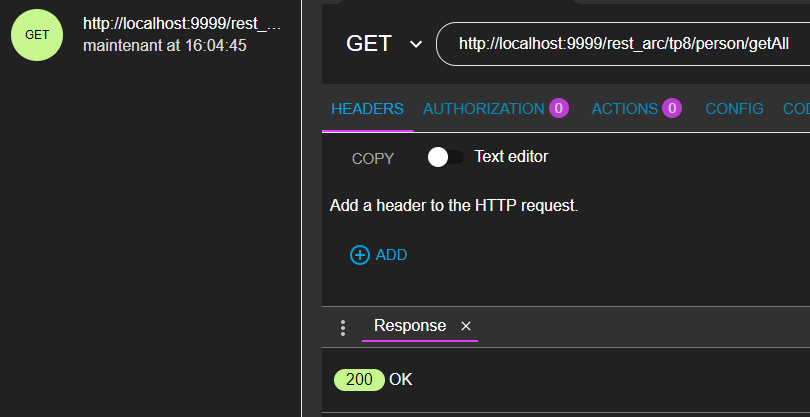


Figure :TestGetAll

#### 2.4.3Ajout d'une Personne via le Path "/add"

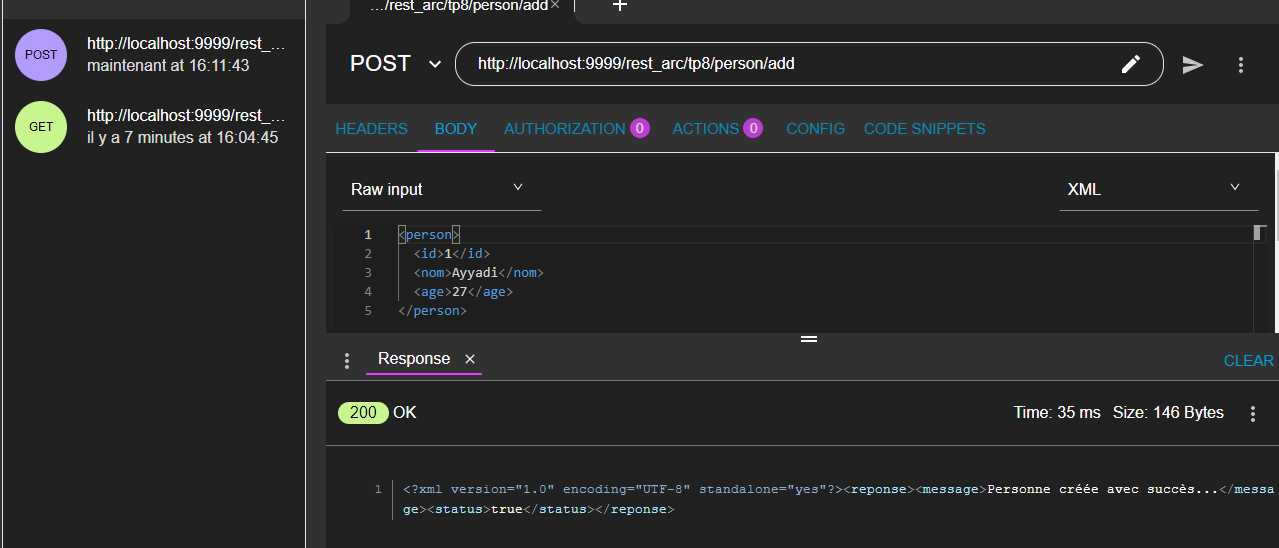


Figure :TestAdd

#### 2.4.4 Ajout du Paramètre "Accept" pour Spécifier le Format de Réponse en JSON et Gestion des Erreurs

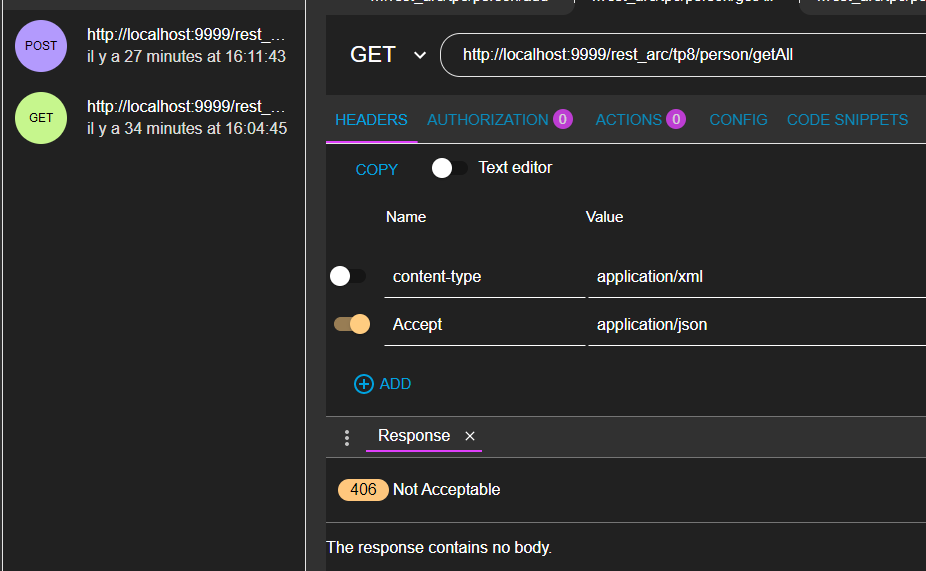


Figure 16:Test GetAll avec Json

#### 2.4.5 Ajout des Dépendances Jackson dans le Fichier "pom.xml" pour Supporter le Format JSON

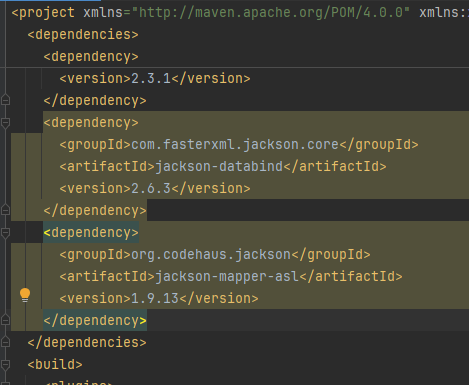


Figure :Dependency pour support json

#### 2.4.6 Réexécution du Serveur Tomcat7 et Envoi de la Requête pour Ajouter une Nouvelle Personne au Format JSON

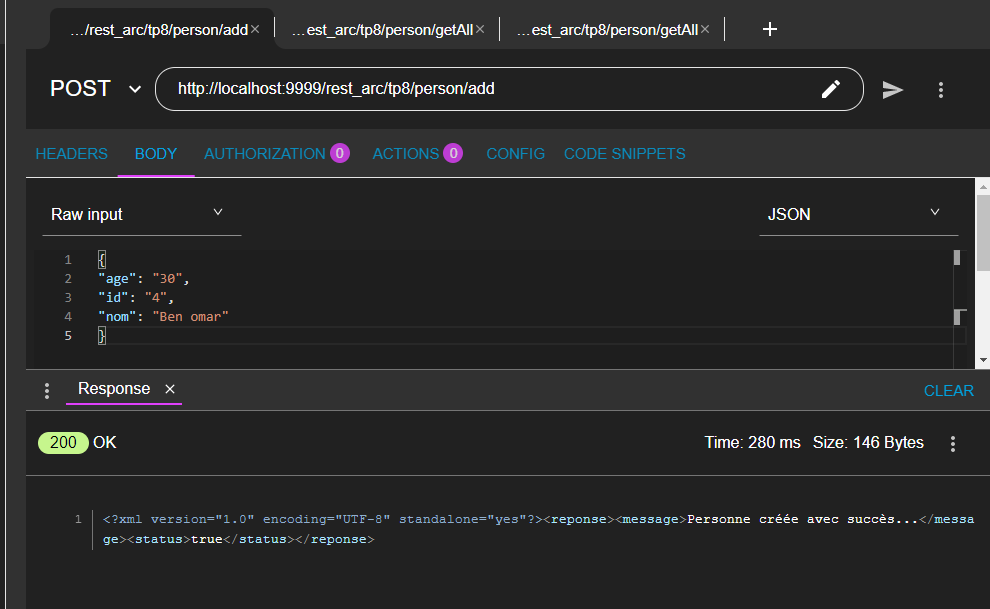


Figure :Test ajouter

#### 2.4.7 Exécution de la Requête pour Récupérer la Personne avec l'ID "1" via le Path "/{id}/get"

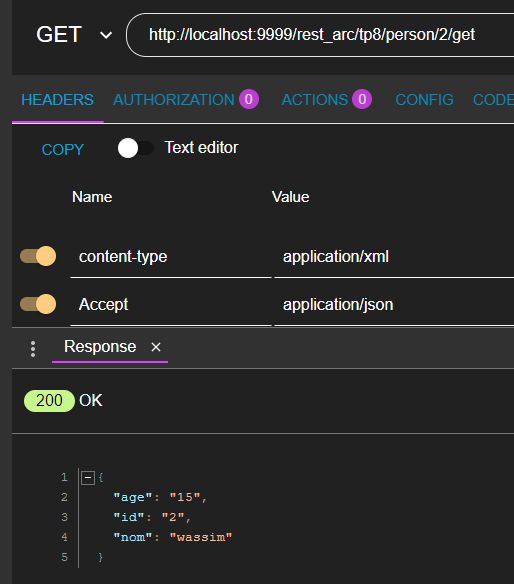


Figure 19:Test Get par Id

#### 2.4.8 Suppression d'une Personne avec le Path "/{id}/delete" : Cas Existant et Non Existant

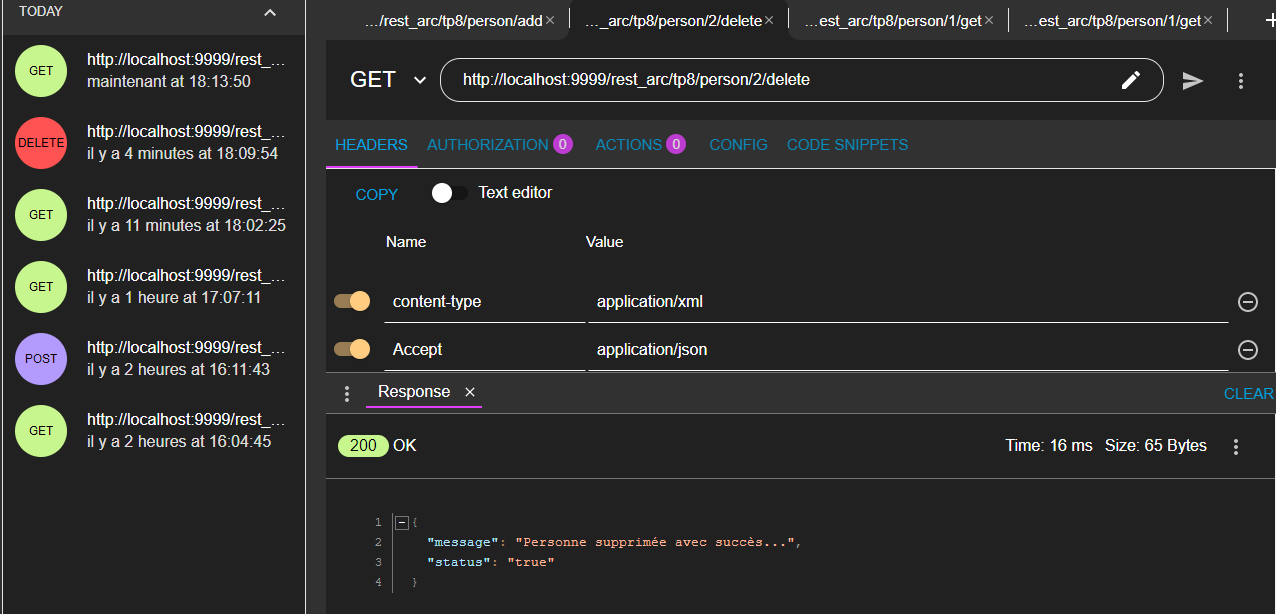


Figure :Test Supprimer

#### 2.4.9 Mise à Jour d'une Personne via le Path "/update" avec la Méthode PUT

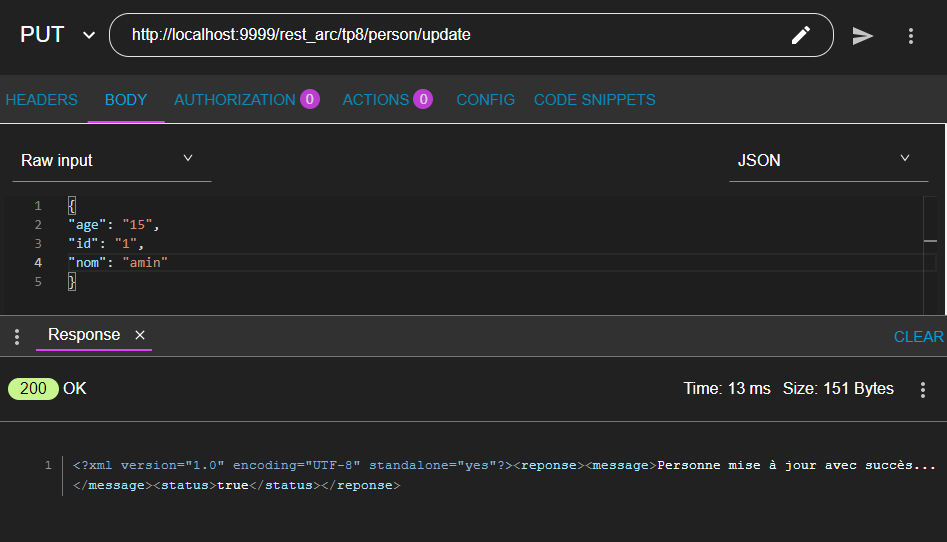


Figure :Test Modifier

### 2.5 Utilisation du Plugin Maven-Jar pour Générer un Fichier JAR à partir d'un Projet WebApp

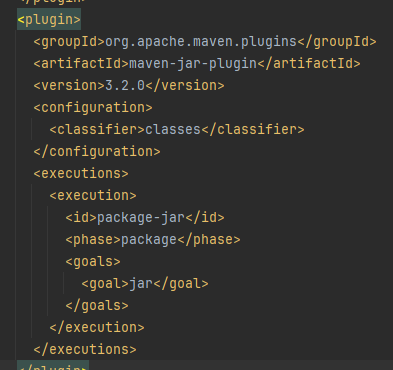


Figure :Plugin de deploiyement

#### 2.5.1Publication du Projet dans le Dépôt Local avec la Commande Maven Install

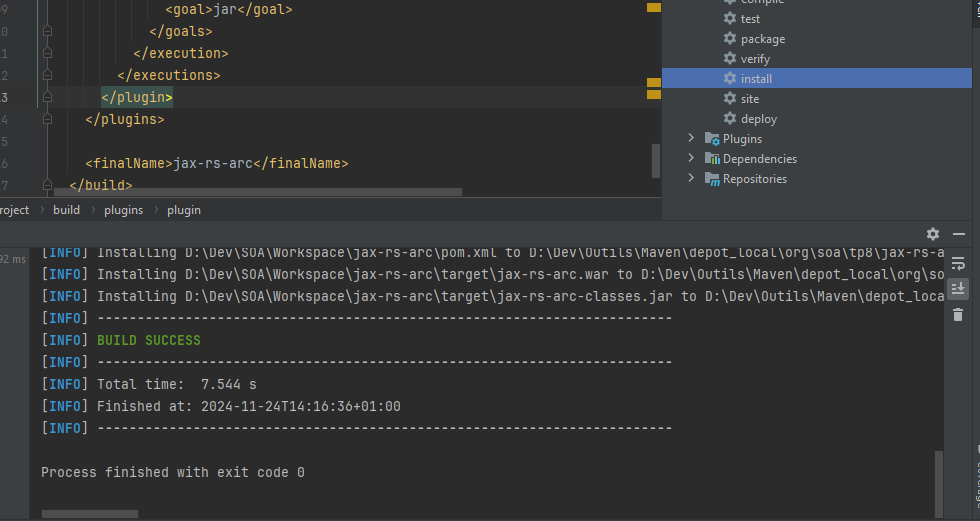


Figure :Mvn Install

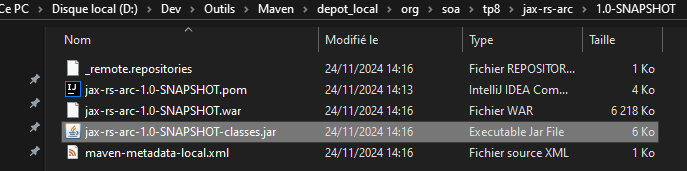


Figure :Depot Locale

## Developper d’application client

### 3.1Création d'un Projet Maven pour un Client JSP RESTful

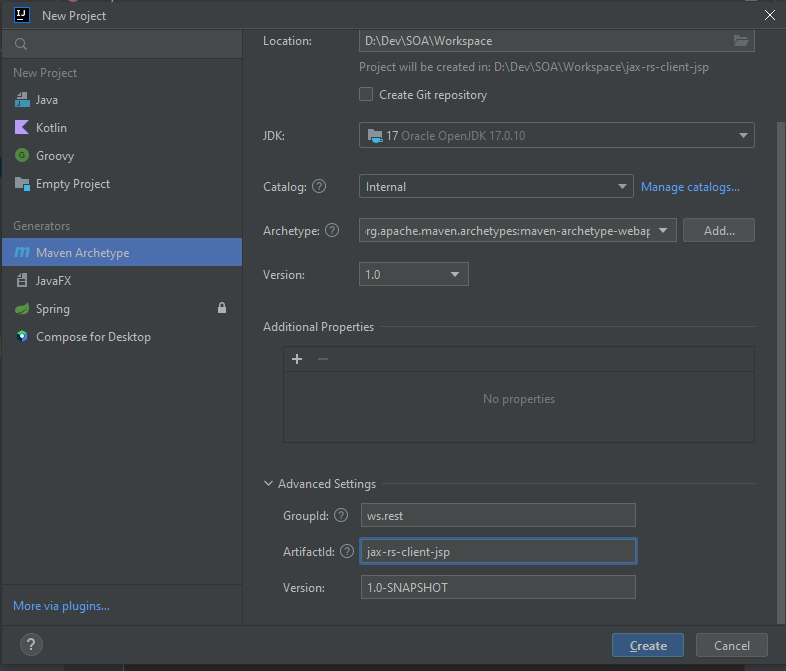


Figure :Creation partie client

### 3.2Configuration des Dépendances pour le Support des Services Web REST

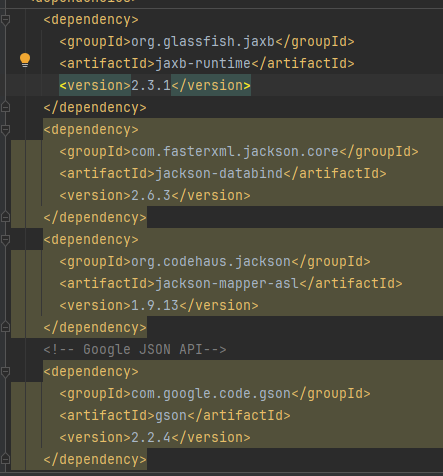


Figure :Configuration REST

Déclaration du Plugin Tomcat7 pour l'Exécution sur le Port 9997

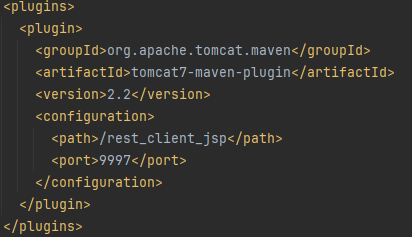


Figure :Declaration plugin Tomcat7

**Ajout d'une Dépendance REST Client avec Balise Classifier dans un Projet Maven**

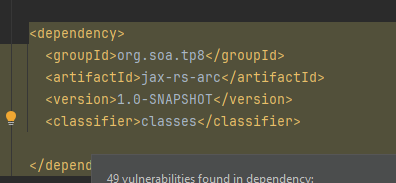


Figure :Ajouter classifier

### 3.3Configuration de la Connexion à la Ressource « /person » dans le Projet« jax-rs-arc »

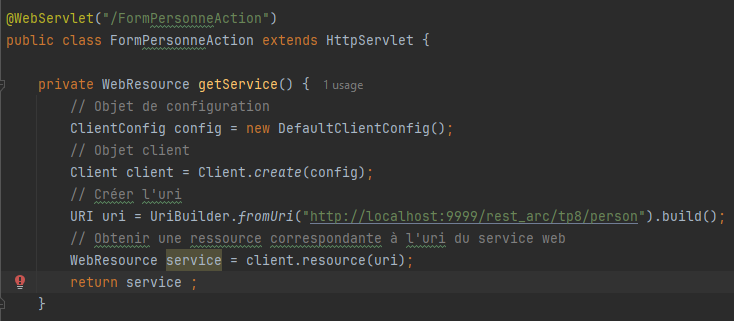


Figure :Configuration de connexion

### 3.4 Appel du Chemin « /add » en POST avec l'Objet « Personne » Instancié"

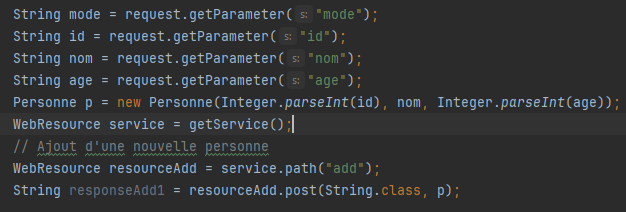


Figure :Ajouter Personne

### 3.5 Appel du Chemin « /getAll » en GET pour Récupérer la Liste des Personnes au Format JSON

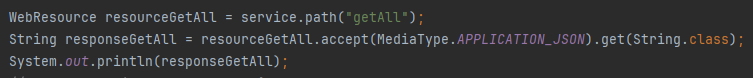


Figure  :Recuperer la liste des personnes

### 3.6 Conversion du Tableau JSON en Liste d'Objets « Personne » avec Gson de Google

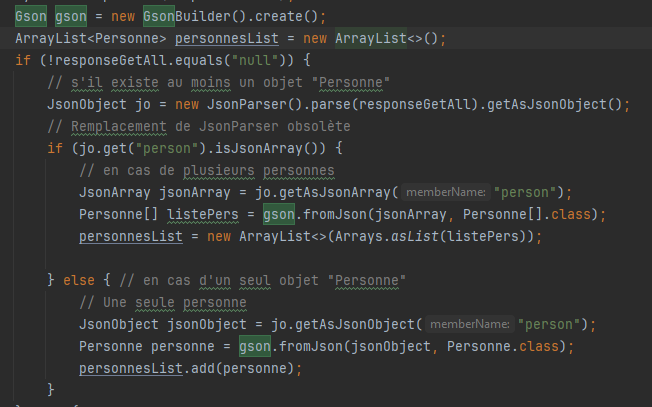


Figure :Analyse en java

### 3.7 Placer la Liste d'Objets dans la Session HTTP et Rediriger vers « listPersonnes.jsp »

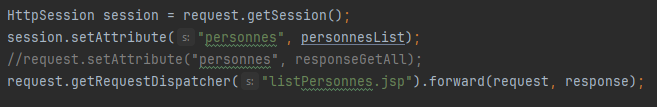


Figure :Passer la liste vers jsp

### 3.8 Création du Fichier « listPersonnes.jsp » pour Afficher la Liste des Personnes sous Forme Tabulaire"

Récupérer la Liste des Personnes Définie comme Attribut dans l'Objet « Session»

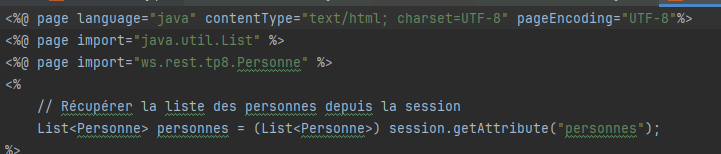


Figure :Recuperer en jsp

Ajout des Liens Hypertextes pour Éditer et Supprimer une Personne avec Redirection vers la Servlet « ListPersonnesAction »

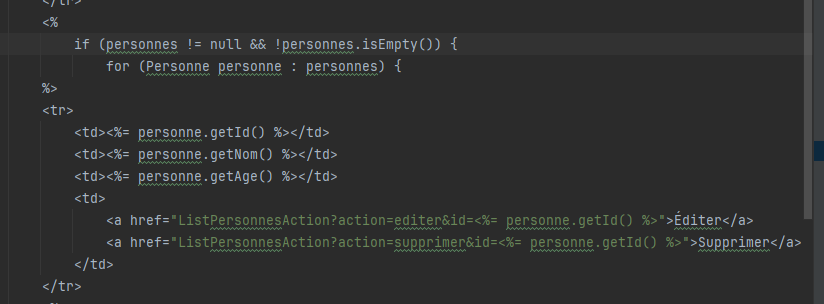


Figure :Ajout les liens hypertextes

Suppression d'une Personne via la Ressource « /person/{id}/delete » et Redirection vers la Servlet « /FormPersonneAction »

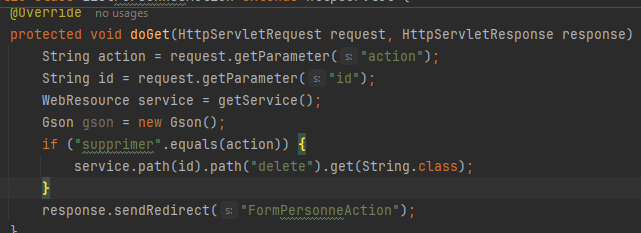


Figure :Suppression d'une Personne

Traitement de l'Action « editer » dans la Servlet « ListPersonnesAction » : Récupération, Conversion et Redirection vers « formPersonne.jsp »

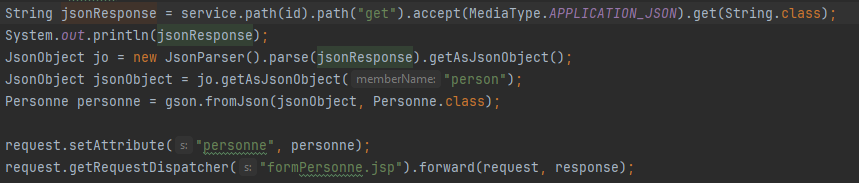


Figure :Traitement de l'Action « editer »

### 3.9 Modification du Formulaire « formPersonne.jsp » pour l'Édition et l'Ajout d'une Personne : Gestion des Attributs et Modes

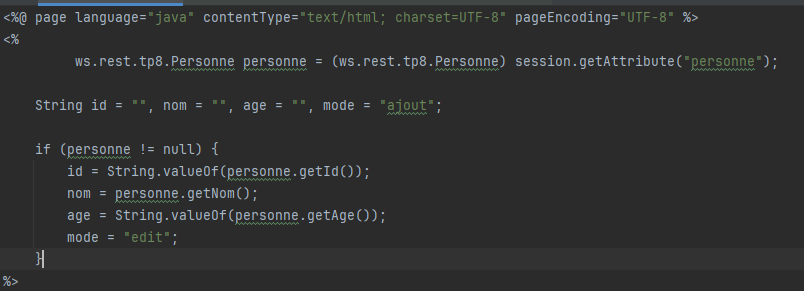


Figure : Edition et Ajout d'une personne

Traitement du Mode « edit » dans la Servlet « FormPersonneAction » : Mise à Jour d'une Personne et Redirection vers « listPersonnes.jsp »

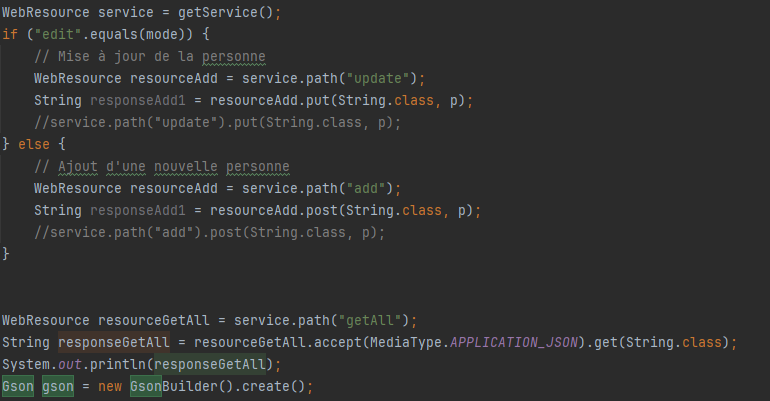


Figure :Traitement du Mode « edit »

### 3.10 Gestion des Erreurs dans la Servlet de Formulaire de Personne

Ce code implémente une gestion des erreurs lors de la soumission du formulaire pour l'ajout ou la mise à jour d'une personne. Il vérifie la validité des paramètres comme l'ID, le nom et l'âge, et affiche des messages d'erreur lorsque des valeurs manquantes ou incorrectes sont détectées. En cas d'erreur de format ou d'exception, les messages appropriés sont ajoutés à une liste d'erreurs et redirigent l'utilisateur vers la page de formulaire avec des informations détaillées sur l'erreur rencontrée.

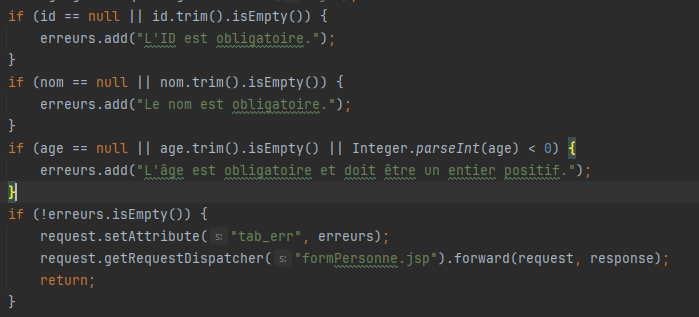


Figure :Gestion des Erreurs

### 3.11 Réalisation :

Organisation des Dossiers « images » et « style » pour la Réalisation des Pages Web sur WebApp

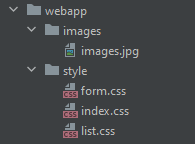


Figure :Architecture de partie web

Page d'Accueil pour la Gestion des Personnes : Navigation entre Ajout et Liste

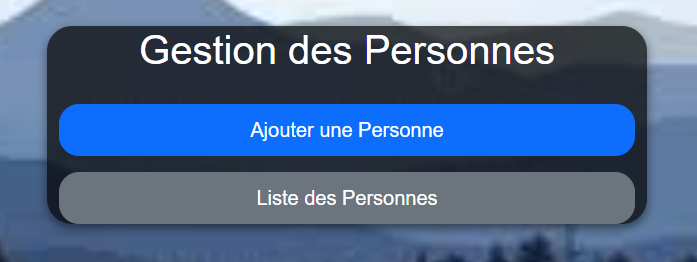


Figure :Page d'Accueil

#### 3.11.1Gestion des Erreurs dans le Formulaire

Affichez clairement les erreurs détectées pour guider l'utilisateur dans la correction des champs invalides

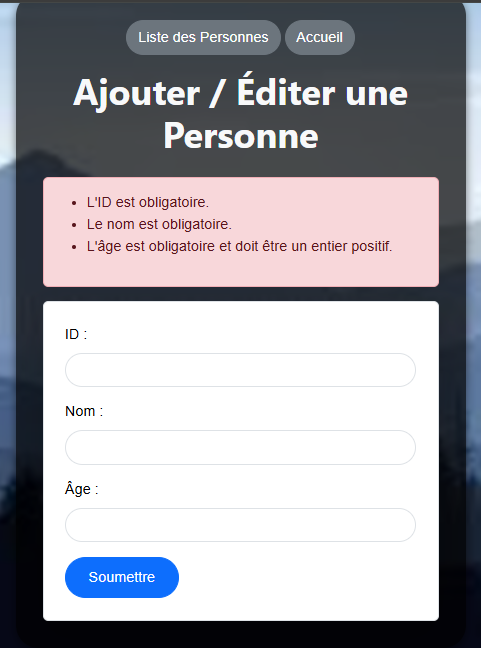


Figure :1Gestion des Erreurs

Message de Succès lors de l'Ajout ou de la Modification

Indiquez à l'utilisateur que l'opération a été effectuée avec succès grâce à un message de confirmation visible

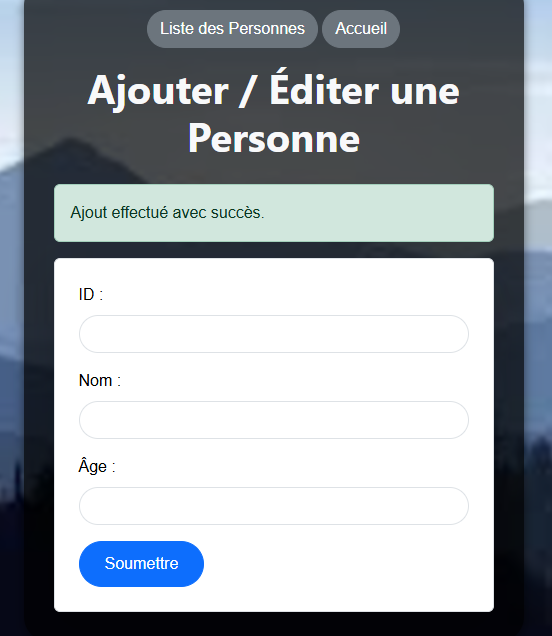


Figure :l'effectation de l'ajout

#### 3.11.2 Affichage et Gestion de la Liste des Personnes

Cette page vous permet de consulter, modifier ou supprimer des personnes enregistrées. Utilisez les actions disponibles pour gérer chaque entrée ou ajoutez une nouvelle personne depuis le menu.

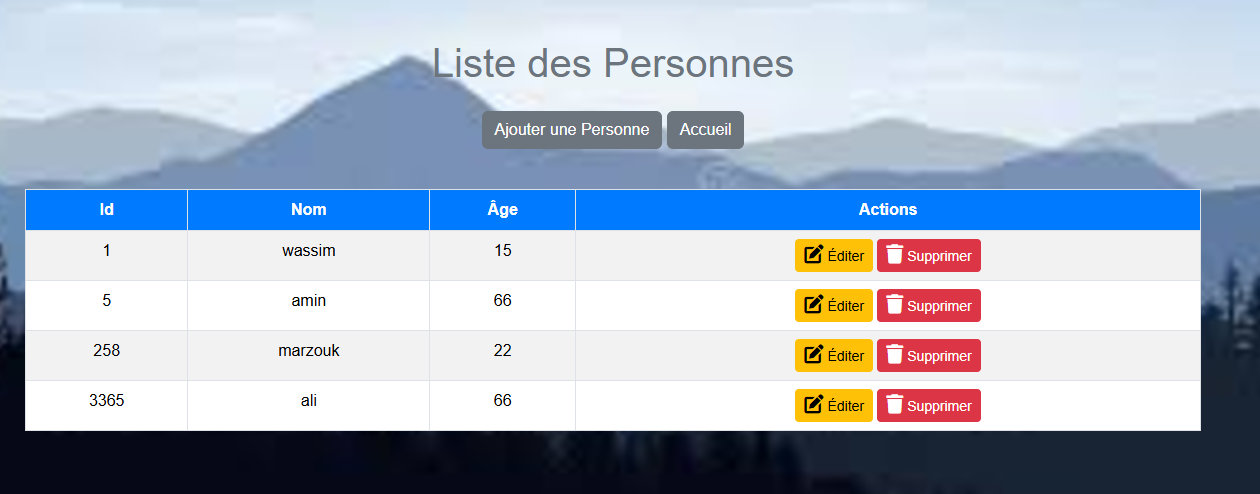


Figure :Affichage la liste de personne