Atelier 4 : Application Web JEE avec modèle MVC et base de données

**Objectifs de l’atelier :**

L’objectif principal est de **mettre en œuvre une application web Java** selon l'architecture **MVC (Modèle-Vue-Contrôleur)**, avec **accès à une base de données MySQL**. Le projet permet de gérer une liste d'utilisateurs (ajout, modification, suppression, affichage).

**Configuration de l’environnement**

1**-installation de XAMPP pour utiliser Apache et MYSQL.**

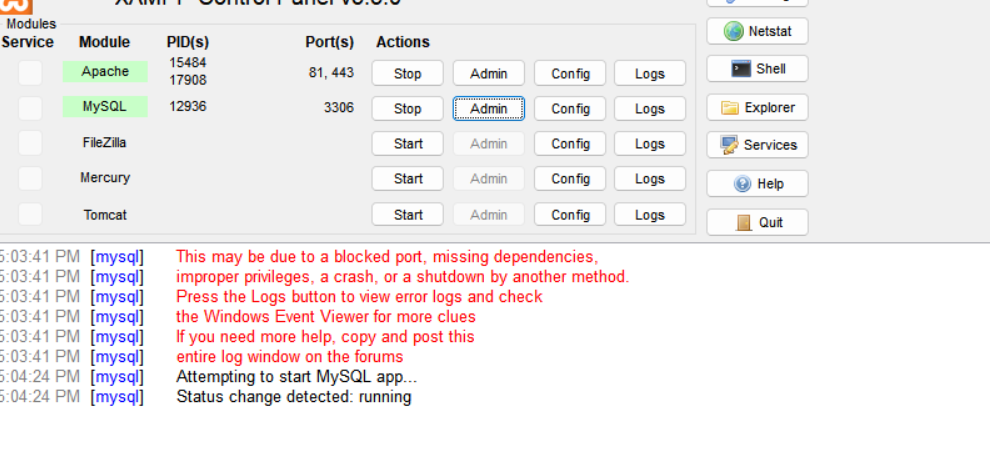


Figure 25 Xampp Controle panel

**2 Lancement de phpMyAdmin via** [**http://localhost:81/phpmyadmin/**](http://localhost:81/phpmyadmin/)

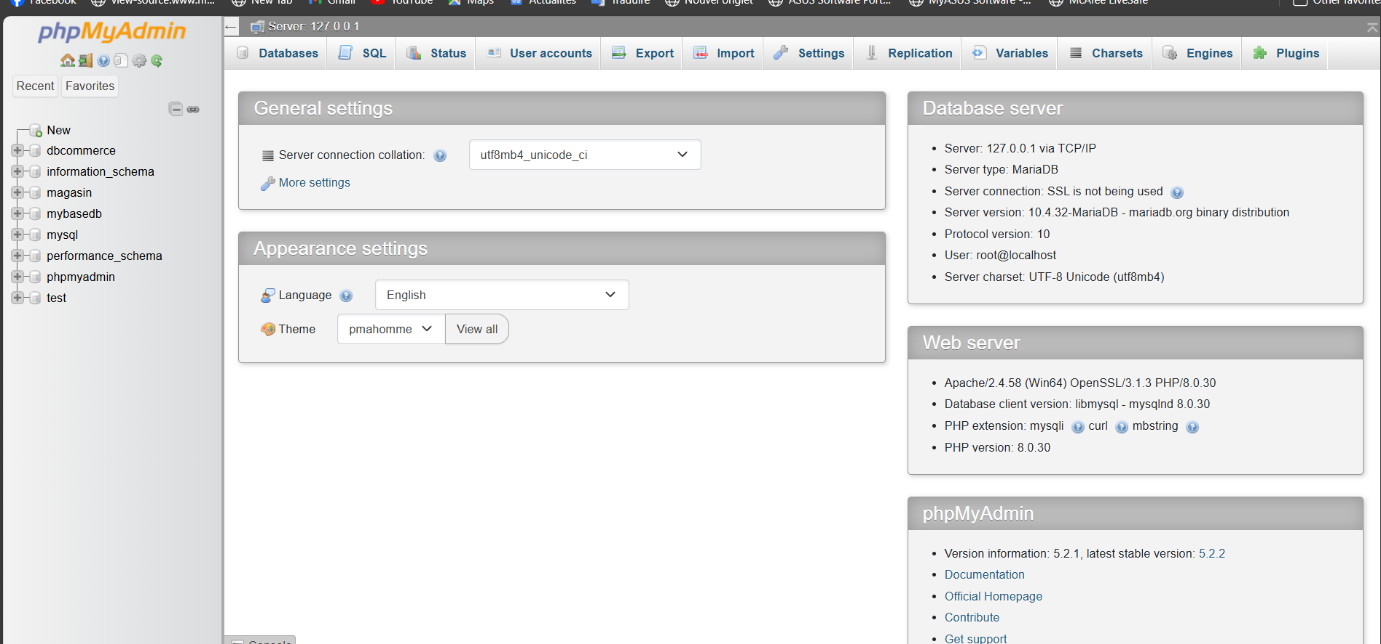


Figure 26 PhpMyAdmin

**3 Creation de base donnee la table magasin qui contient une table user :**



Figure 27Creation du Table User dans la base MAGASIN

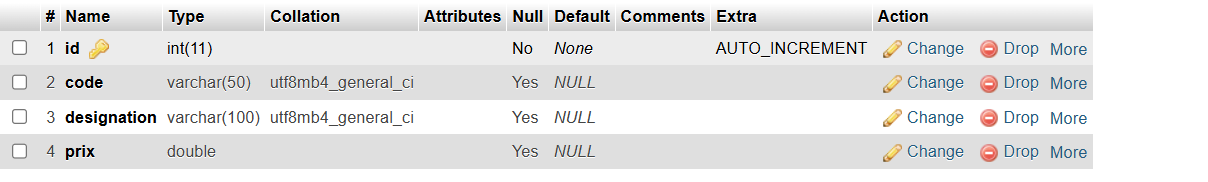


Figure 28Les champs de la Table User

**4 . insertion de 3 utilisateurs dans le tableau**

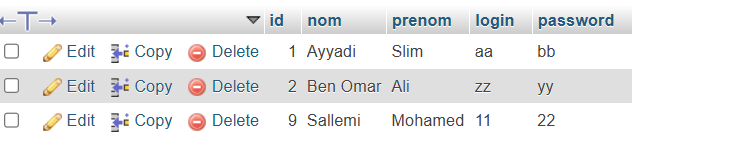


Figure 29 utilisateurs initiaux

**Partie B:couche Metier**

**1 creation d’un projet dynamique nommee web\_app\_tp04.**

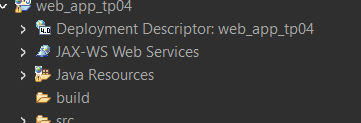


Figure 30 Projet dynamique web\_app\_tp04

**2 creation d’un package metier**

Classe java bean User qui contient les getters/ setters/ constructeurs, methode toString()

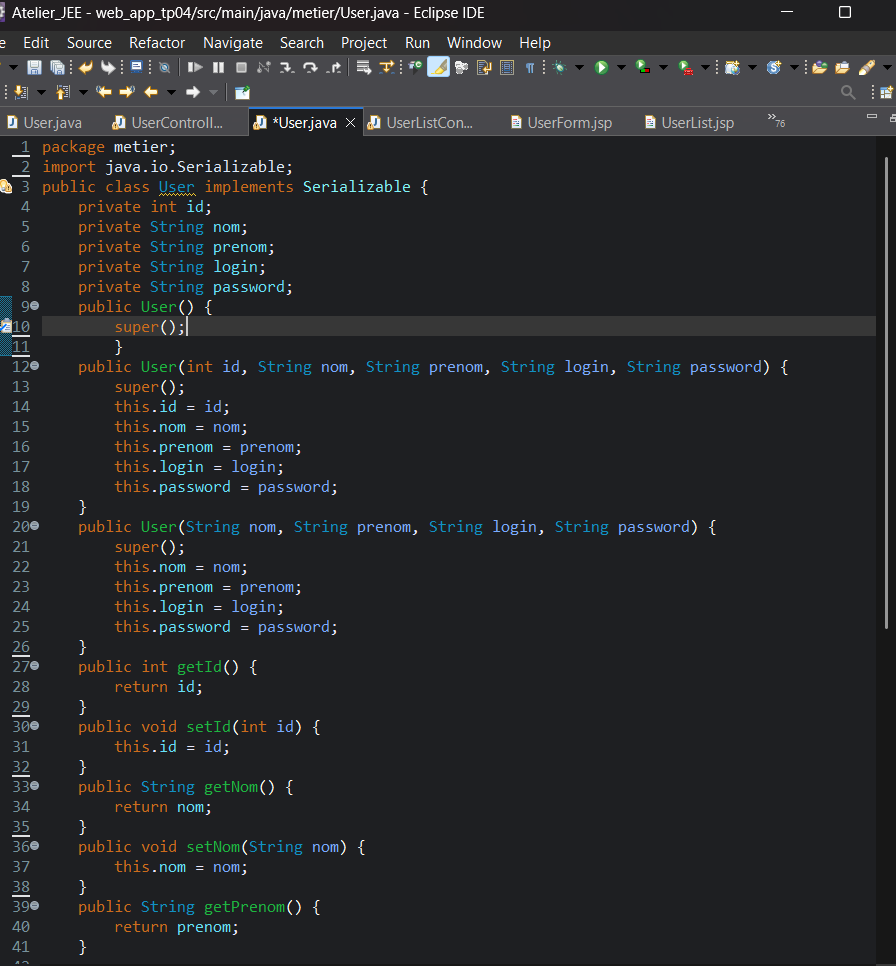


Figure 31Classe User

Interface UserMetierInterface avec les méthodes metier

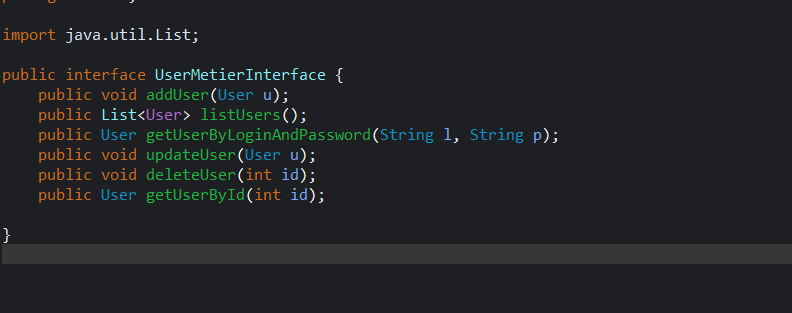


Figure 32Interface UserMetierInterface

Classe UserMetierImpl qui implemente ces méthodes (ajout,liste, recherche mise a jour , suppression, recherche par id)

A screen shot of a computer program

AI-generated content may be incorrect.

Figure 33UserMetierImpl

**Classe DBConnexion**

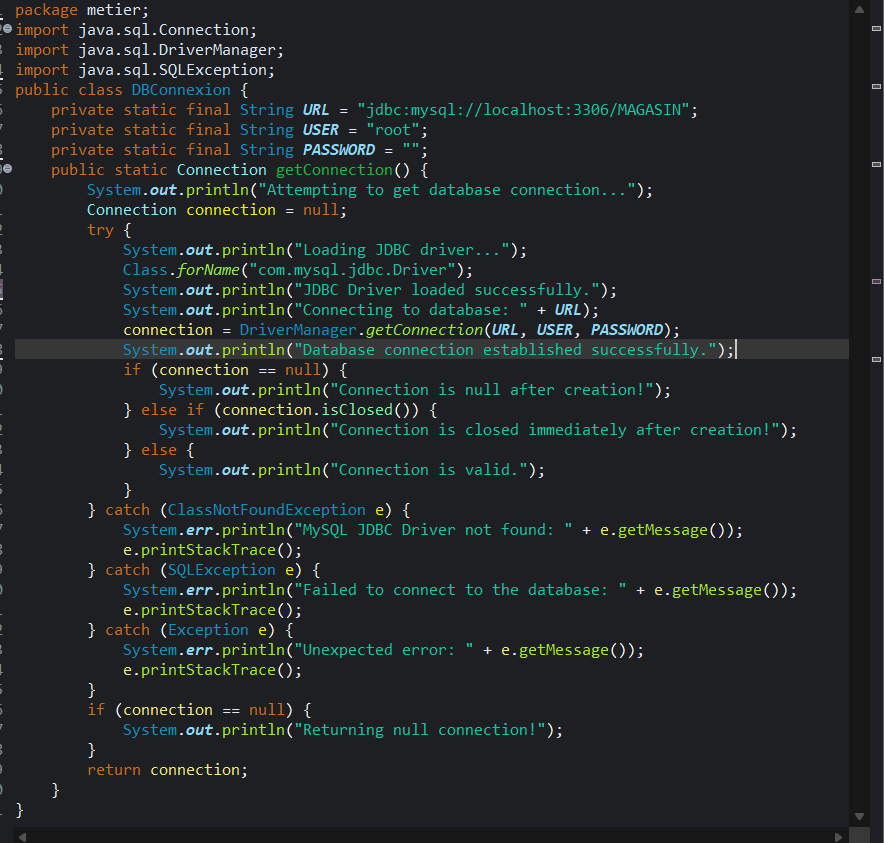


Figure 34DBConnexion

**Classe testMetier pour vérifier toutes les operations**

A screen shot of a computer program

AI-generated content may be incorrect.

Figure 35TestMetier

**Partie C interface web**

Creation du package WEB CONTENANT LES SERVELETS

UserController(connexion)

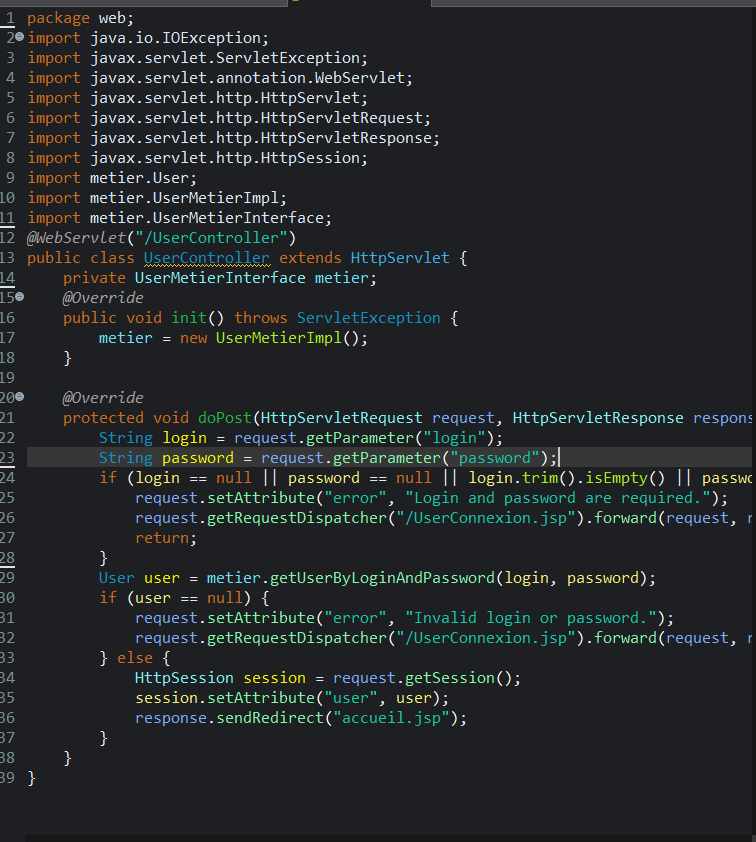


Figure 36UserController

**UserEditioncontroller(ajout/modification)**

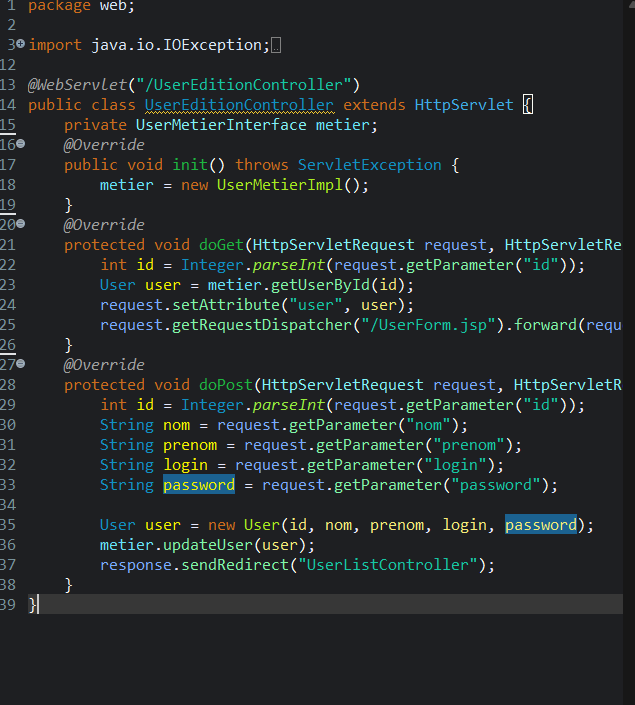
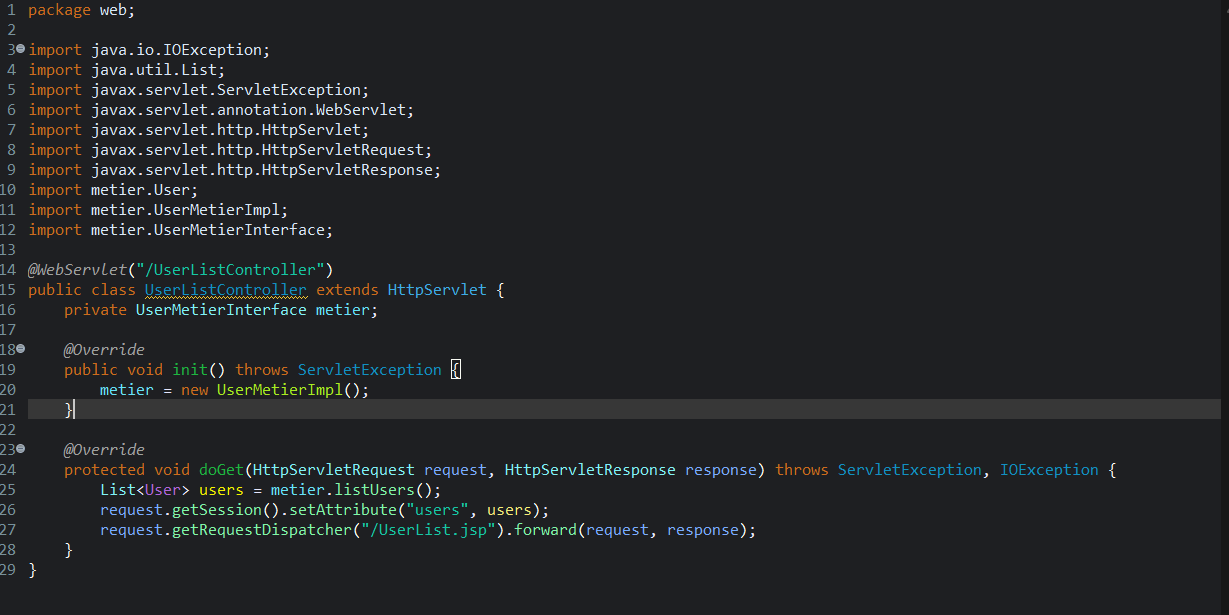


Figure 37UserEditionController

**UserList controller**



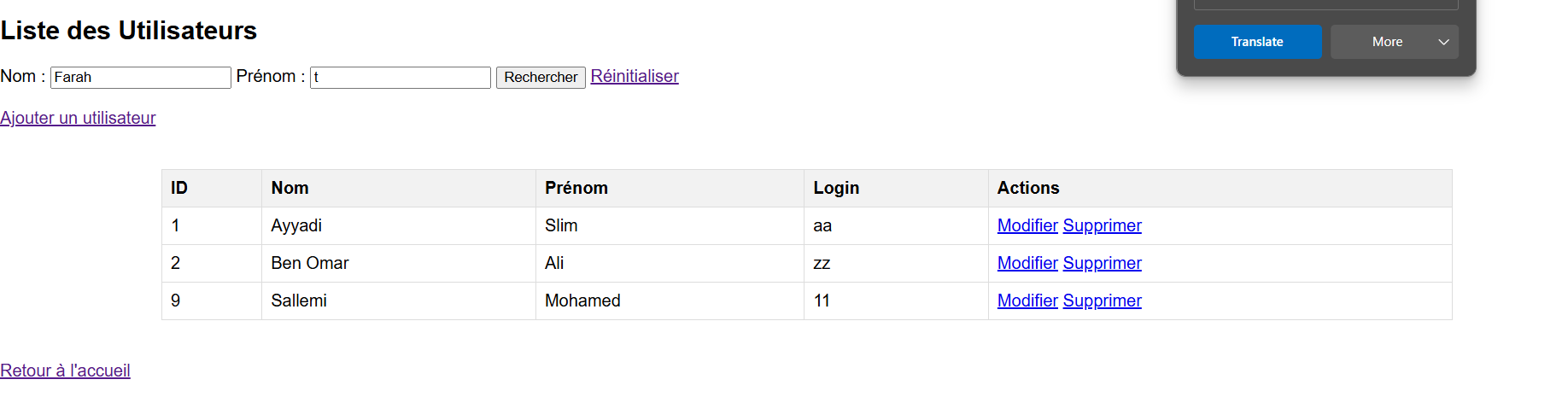


Figure 38UserListController

**Deploiement des vues JSP**

UserConnexion.jsp

A screenshot of a computer program

AI-generated content may be incorrect.

Figure 39UserConnexion.jsp

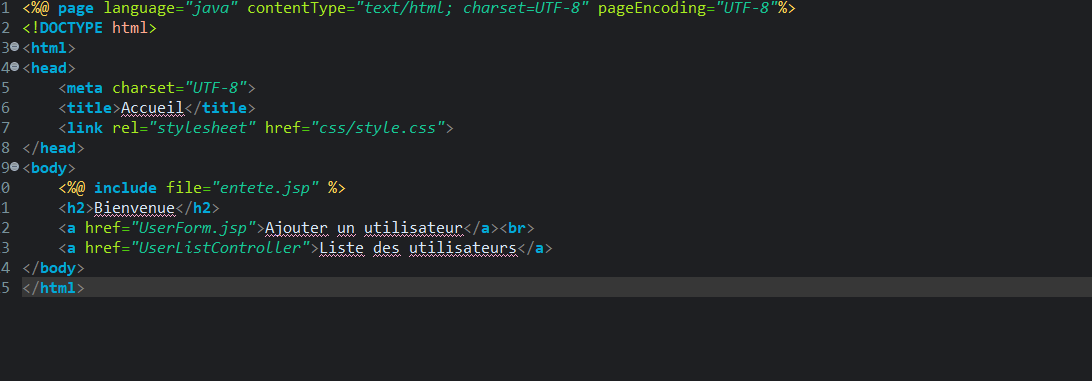


Figure 40Acceuil.jsp



**Entete.jsp**

A screen shot of a computer code

AI-generated content may be incorrect.

Figure 41 entete.jsp

**Userform.jsp**

A screen shot of a computer program

AI-generated content may be incorrect.

Figure 42UserForm



A screen shot of a computer code

AI-generated content may be incorrect.

Figure 43UserList.jsp

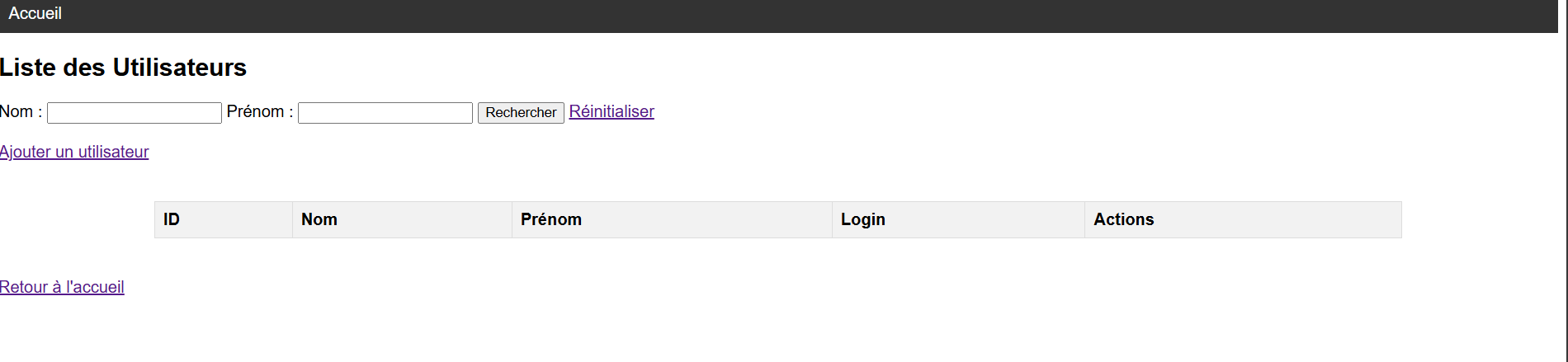


Figure 44UserList.jsp

**Activite 1**

Créant une copie nommée "web\_app\_tp04\_bis".

2. Découpant le package `metier` en trois sous-packages : `modele.metier`, `modele.domaine`, et `modele.dao`.

3.Renommant le package `web` en `controleur`.

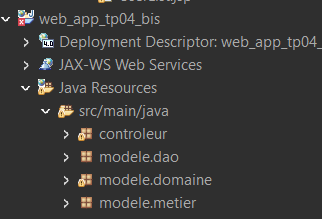


Figure 45 Les packages de L'activite 1

1. package "modele.metier": contient l'interface "UserMetierInterface" et son implémentation "UserMetierImpl".

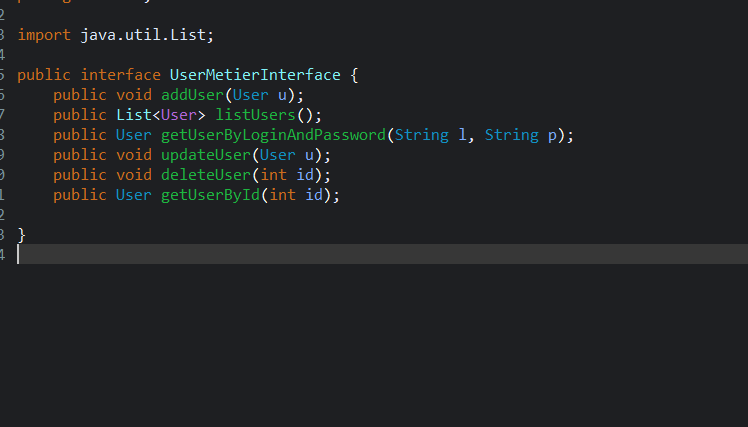


Figure 46UserMetierInterface

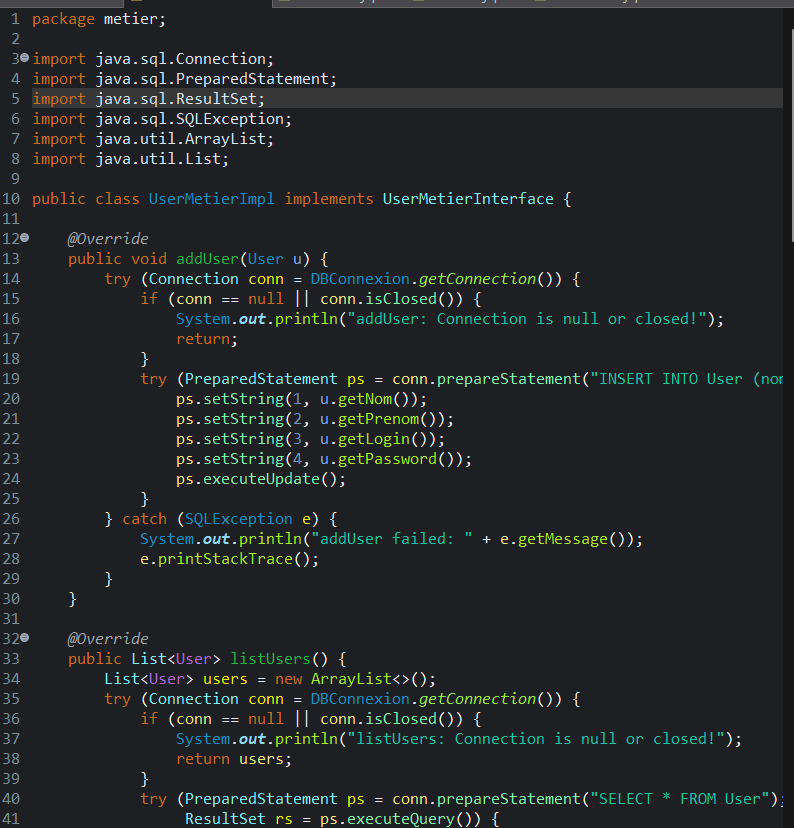


Figure 47UserMetierImpl

1. package "**modele.domaine**" : contient la classe "**User**"
2. package "**modele.dao**: contient la classe "**DBConnexion**" et une nouvelle interface "**UserDaoInterface**" et son implémentation "**UserDaoImpl**" qui présentent deux méthodes:
   1. une pour lire des données à partir de la BD (**select**)
   2. et une autre pour écrire des données dans la BD (**insert**, **update**, **delete**)

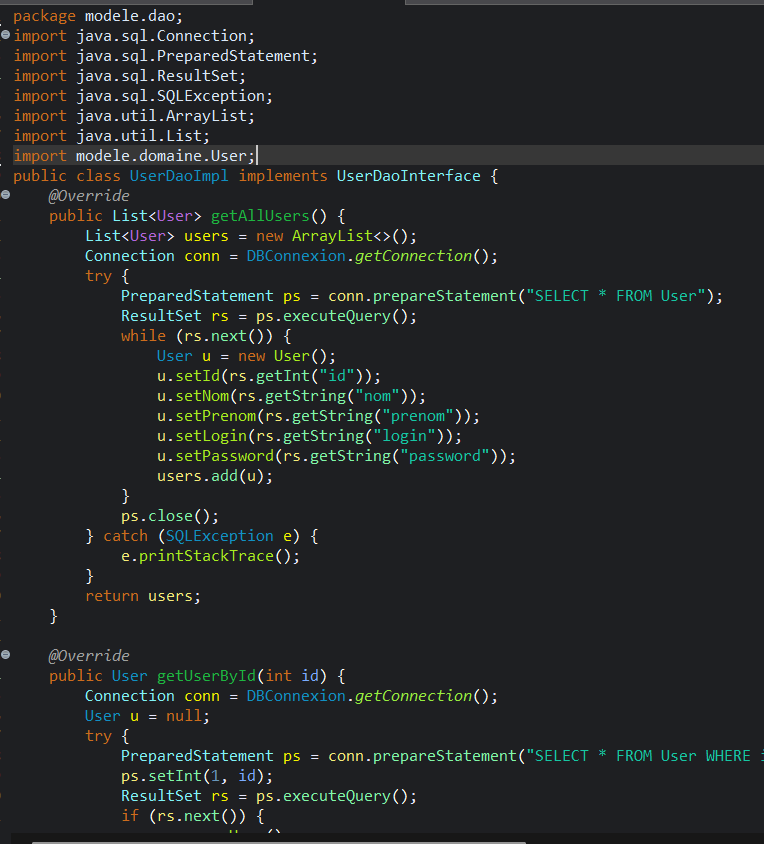


Figure 48UserDaoImpl

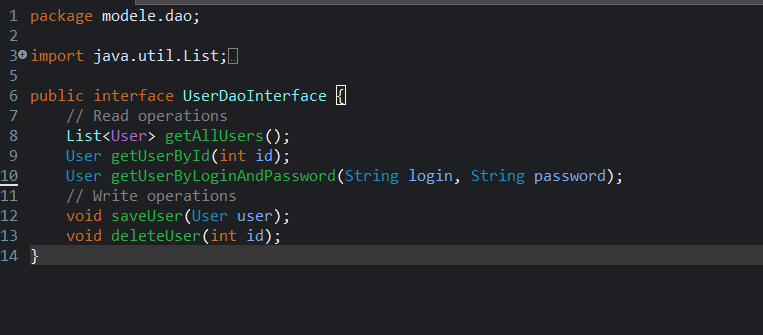


Figure 49UserDaoInterface

Changer le nom du package "**web**" par "**controleur**"

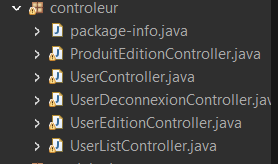


Figure 50Le package controlleur dans l'activite 1

**Activite 2**

L'objectif de cette activité était d'étendre le projet "web\_app\_tp04\_bis" pour ajouter la gestion des produits, en suivant une approche similaire à celle utilisée pour les utilisateurs.

Création de la Classe `Produit`

- Une nouvelle classe a été ajoutée dans le package `modele.domaine` pour représenter un produit.

- Cette classe inclut les attributs suivants : un identifiant entier, un code sous forme de chaîne, une désignation sous forme de chaîne, et un prix sous forme de nombre réel.

- Les constructeurs, getters, setters et une méthode de représentation textuelle ont été définis pour cette classe.

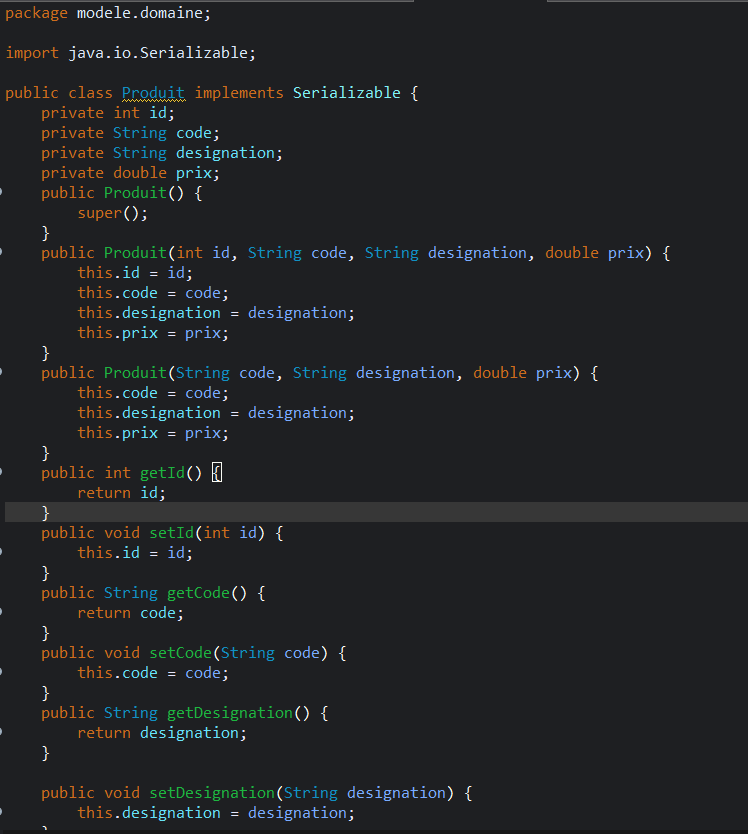


Figure 51Le classe Bean Produit

Création de la Table `produit` dans la Base de Données

- Une table correspondant à la classe `Produit` a été ajoutée dans la base de données `MAGASIN` via phpMyAdmin.

- La table a été définie avec des colonnes pour l’identifiant (clé primaire avec auto-incrémentation), le code, la désignation et le prix.

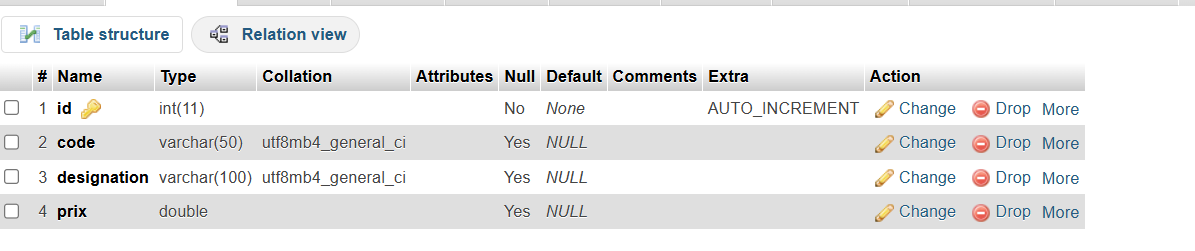


Figure 52Les champs de table Produit

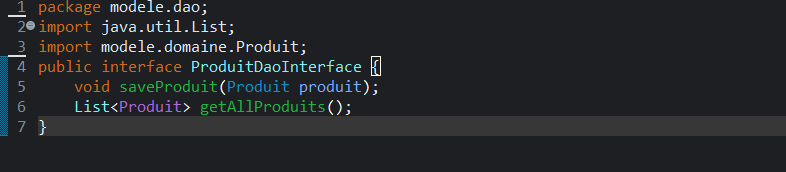


Figure 53ProduitDaoInterface

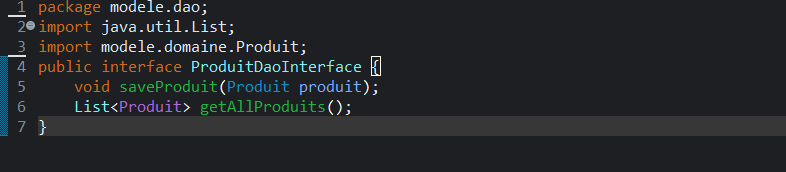


Figure 54ProduitDaoInterface

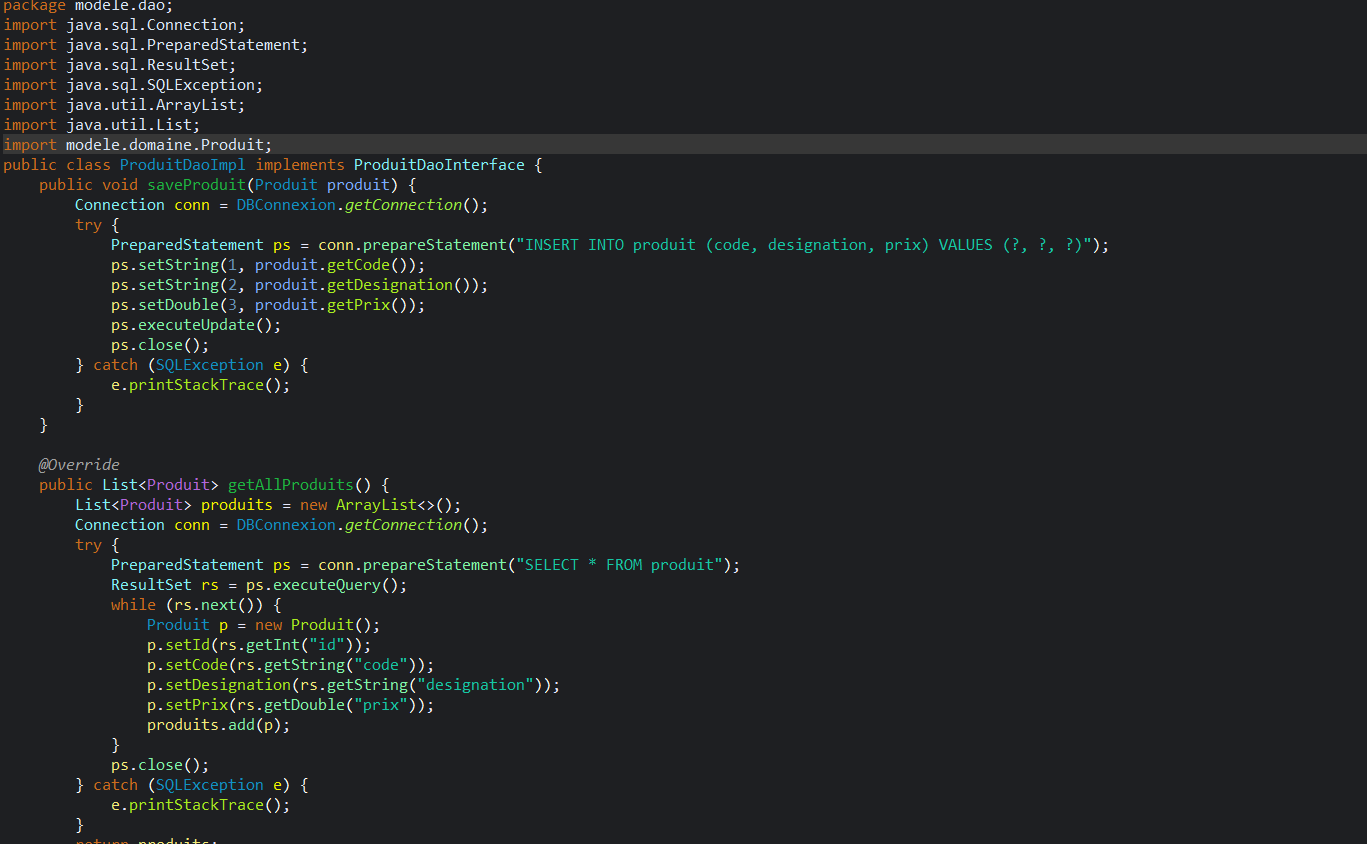


Figure 55ProduiDaoImpl

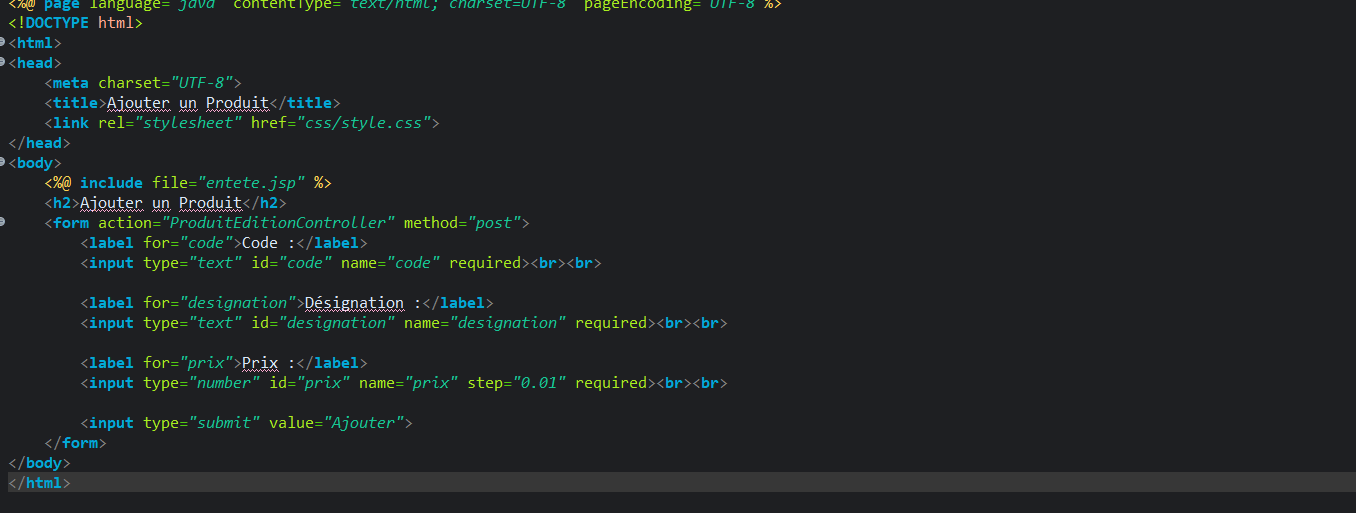


Figure 56FormProduit.jsp

**4.3 : Gestion des relations (Produit/Categorie)**

**Objectifs**

L'objectif de cet atelier était de gérer les produits stockés physiquement dans la table "Produit" en établissant une relation avec une nouvelle table "Categorie". Les tâches principales comprenaient la création de la table "Categorie", la modélisation en Java, la réalisation des fonctionnalités métier, et l'ajout d'une clé étrangère dans la table "Produit".

**I. Réalisation**

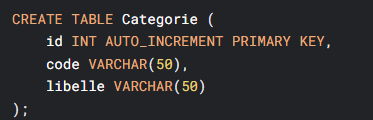
**1. Création de la table "Categorie"**

La table "Categorie" a été créée dans la base de données avec les colonnes suivantes :

id : integer (clé primaire AUTO-INCREMENT)

code : varchar2 (50)

libelle : varchar2 (50)

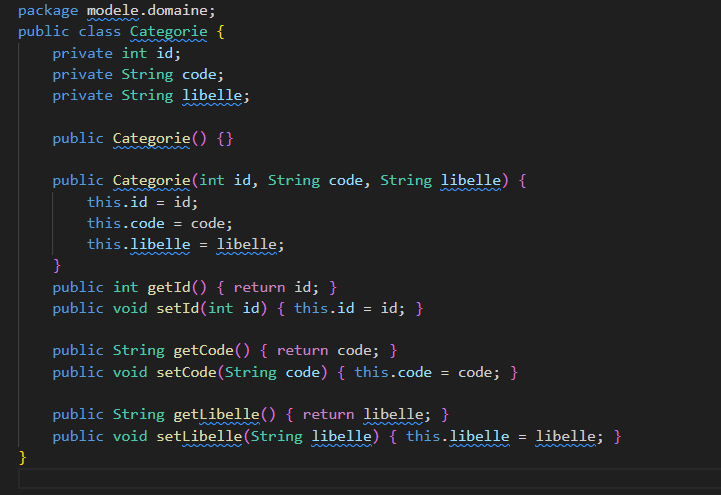


57Script SQL : CREATE Categorie

**2. Modélisation en Java**

Un Java Bean "Categorie" a été défini dans le package modele.domaine pour représenter la table "Categorie".

**Classe Categorie.java :**

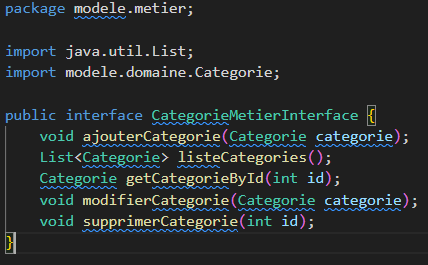
****

58 Categorie.java

**3. Fonctionnalités métier**

Les opérations CRUD pour la gestion des catégories ont été implémentées dans le package modele.metier.

**Interface CategorieMetierInterface.java :**



59CategorieMetierInterface.java

**Implémentation CategorieMetierImpl.java :**

****

60 CategorieMetierImpl.java

**4. Vues pour les catégories**

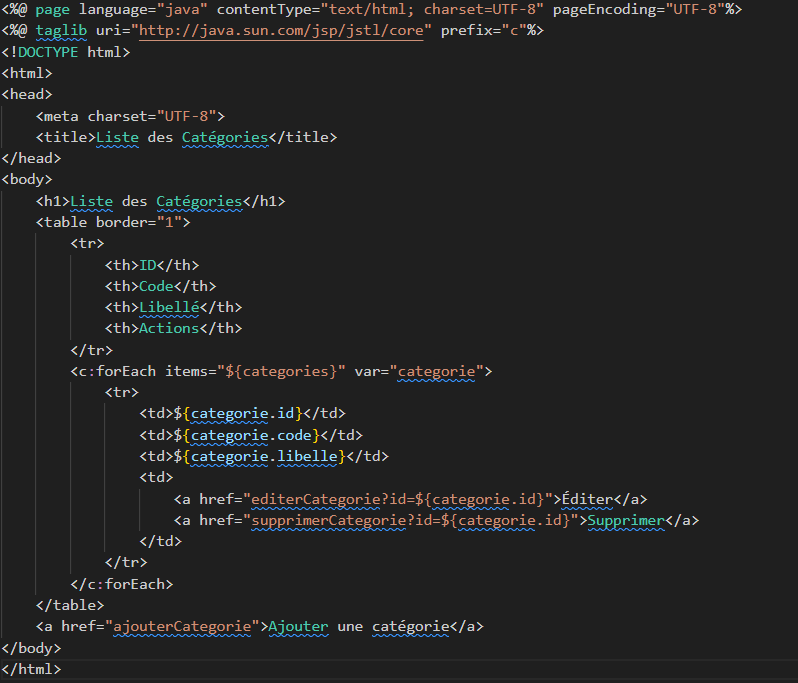
Les pages JSP suivantes ont été créées pour gérer les catégories :

ajouterCategorie.jsp : Formulaire d'ajout d'une catégorie.

editerCategorie.jsp : Formulaire de modification d'une catégorie.

listeCategories.jsp : Affichage de la liste des catégories.

**Exemple de listeCategories.jsp :**

****

61 listeCategories.jsp

**5. Ajout de la clé étrangère dans "Produit"**

Une colonne categorie\_id a été ajoutée à la table "Produit" pour établir la relation avec la table "Categorie".

Script SQL :

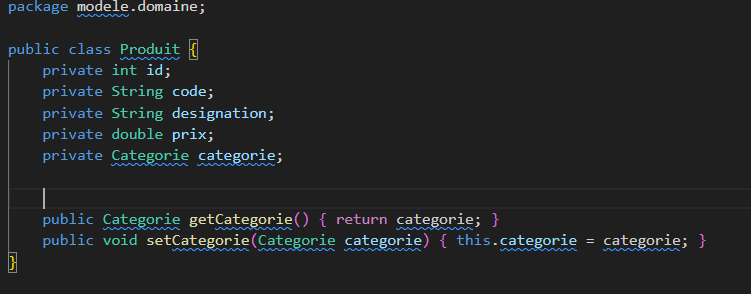


62 cle etrangere

**6. Modifications côté Java**

La classe Produit a été mise à jour pour inclure une référence à Categorie.

**Classe Produit.java :**

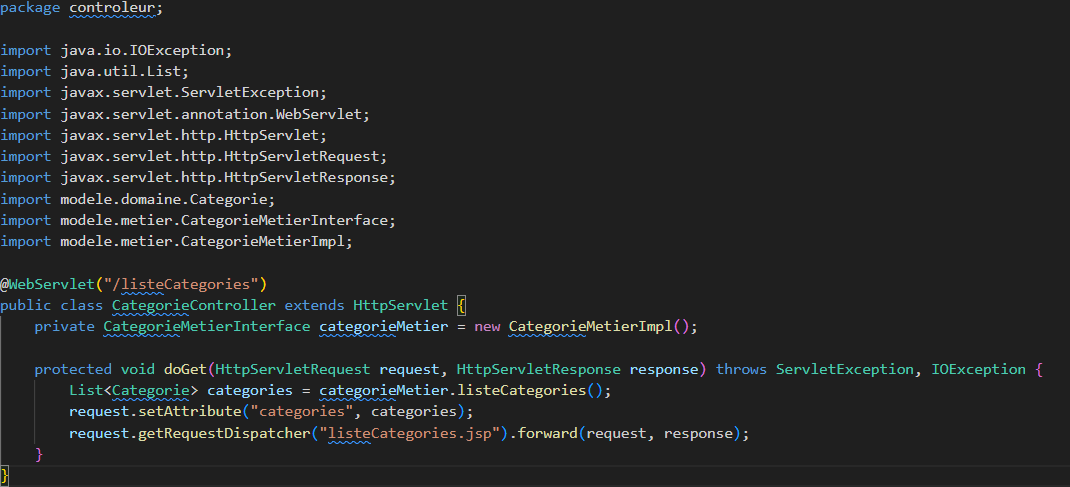
****

63Produit.java

**7. Contrôleurs et Servlets**

Les contrôleurs pour la gestion des catégories ont été ajoutés dans le package controleur.

**Exemple de CategorieController.java :**

****

64CategorieController.java

**II. Conclusion**

Cet atelier a permis de :

Créer une nouvelle table "Categorie" et établir une relation avec la table "Produit".

Implémenter les fonctionnalités métier pour gérer les catégories.

Développer les vues nécessaires pour interagir avec les catégories.

Structurer le projet selon une architecture MVC propre.

Les compétences acquises incluent la gestion des relations entre tables, la manipulation des données via Java, et la création d'interfaces utilisateur dynamiques avec JSP. Ce projet constitue une base solide pour le développement d'applications web plus complexes.

**Compte Rendu du TP07 - Les composants "EJB Entity"**

Ce TP traite de la persistance des données en utilisant des EJB Entity (composants Java Enterprise Edition) avec le serveur JBoss WildFly et le standard JPA (Java Persistence API).

**Objectifs du TP**

* Préparer l'environnement de développement EJB
* Mettre en place la persistance des données

**Partie A : Préparation de l'environnement de développement**

1. **Installation du serveur WildFly**
   * Téléchargement et extraction de WildFly 18 dans le dossier D:\Atelier\_JEE\Servers



Figure 65 instalation wildfly

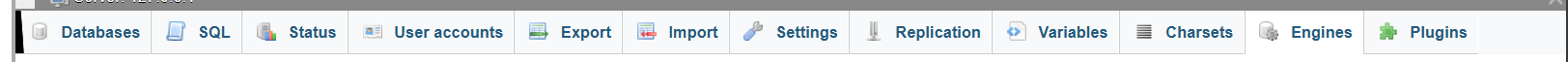
* + Installation du plugin JBoss Tools via Eclipse Marketplace
  + Configuration du runtime environment WildFly 18 dans Eclipse

A screenshot of a computer

AI-generated content may be incorrect.

Figure 66o Configuration du runtime environment WildFly

* + Démarrage du serveur WildFly depuis Eclipse

**Partie B : Configuration d'une source de données (datasource) **

1. **Création de la base de données**
   * Création d'une base MySQL nommée "MySqlDB" via phpMyAdmin



Figure 67o Création d'une base MySQL

* + Configuration d'un compte utilisateur avec les identifiants admin/admin

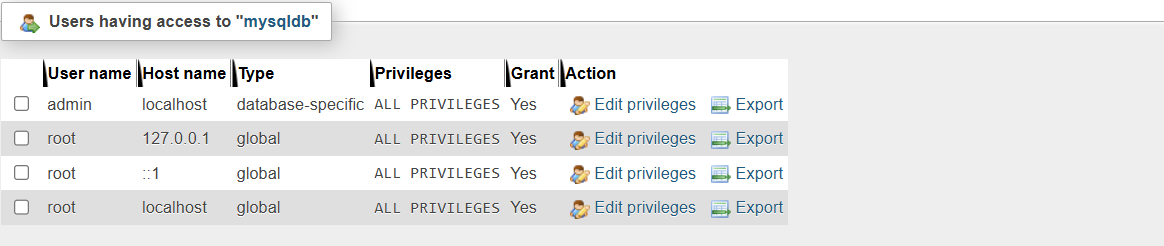


Figure 68o Créatin compte ADMIN

**Configuration de WildFly pour MySQL**

* Création d'un module MySQL dans WildFly (ajout du driver JDBC)

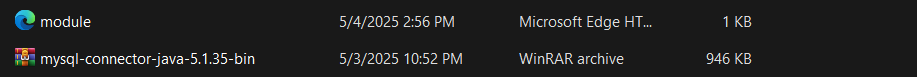


Figure 69• Création d'un module MySQL dans WildFly

* Configuration du driver JDBC via la console d'administration
* Création d'un compte administrateur (jbossadmin/jbossadmin) pour la console d'administration

A screenshot of a computer error

AI-generated content may be incorrect.

Figure 70• Création d'un compte administrateur

**Création de la datasource**

* Création d'une source de données "MySqlDS" pointant vers la base "MySqlDB"

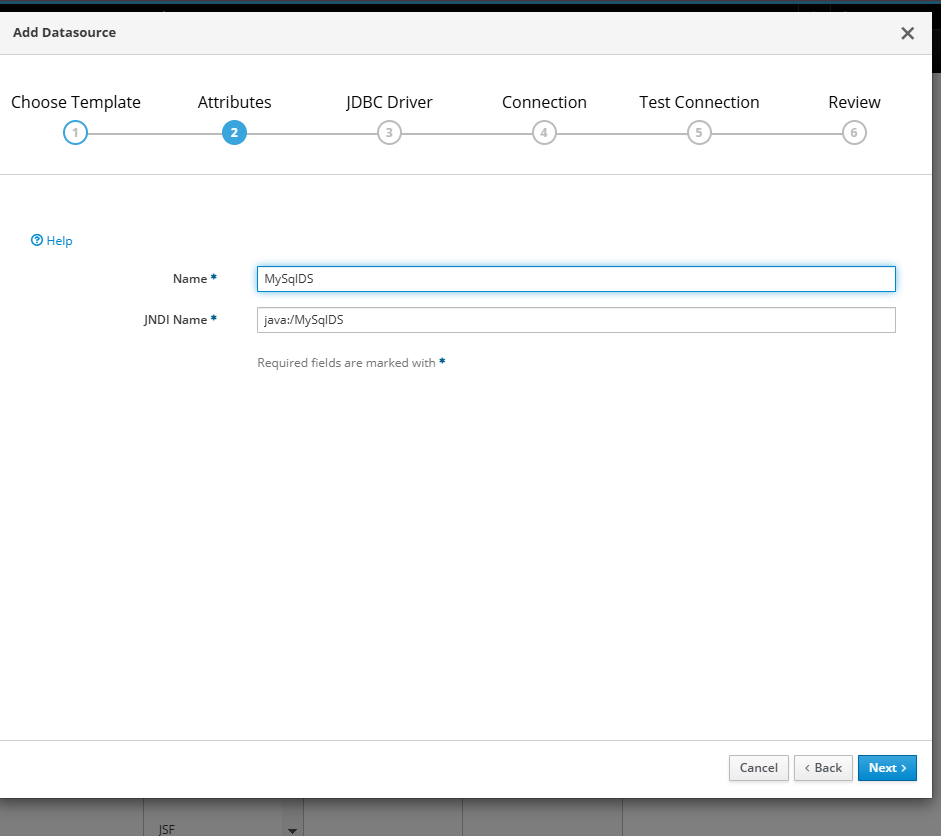


Figure 71• Création d'une source de données "MySqlDS"

* Ouvrir le wildfly via localhost:9090



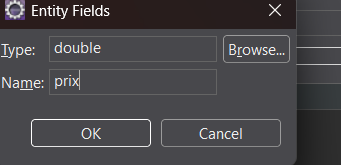
Figure 72• Ouvrir le wildfly

* Test de la connexion à la base de données

Partie C : Premier projet EJB

**Création du projet**

* Création d'un projet EJB "ejb\_app\_tp07" associé au serveur WildFly 18



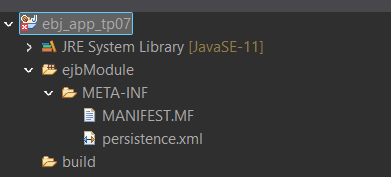
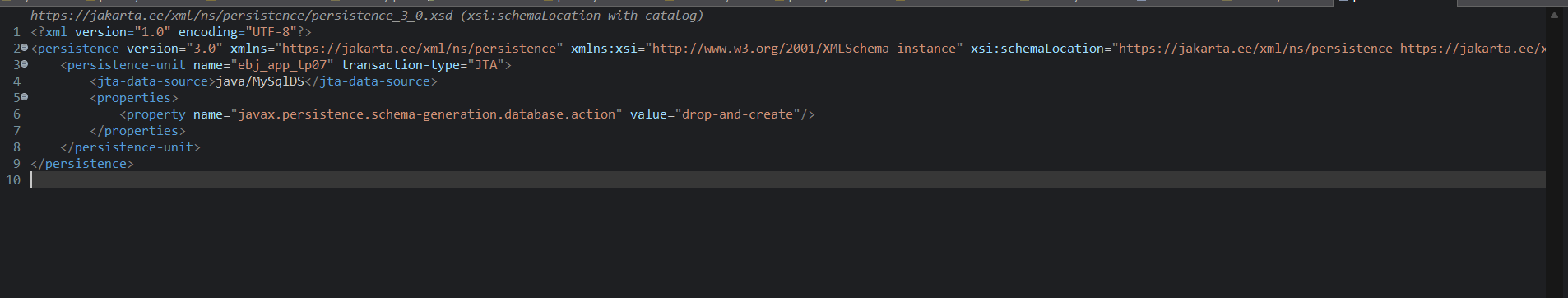


Figure 73• Création d'un projet EJB "ejb\_app\_tp07"

Avec persistence.xml



* Configuration du niveau de conformité du compilateur Java à 1.8

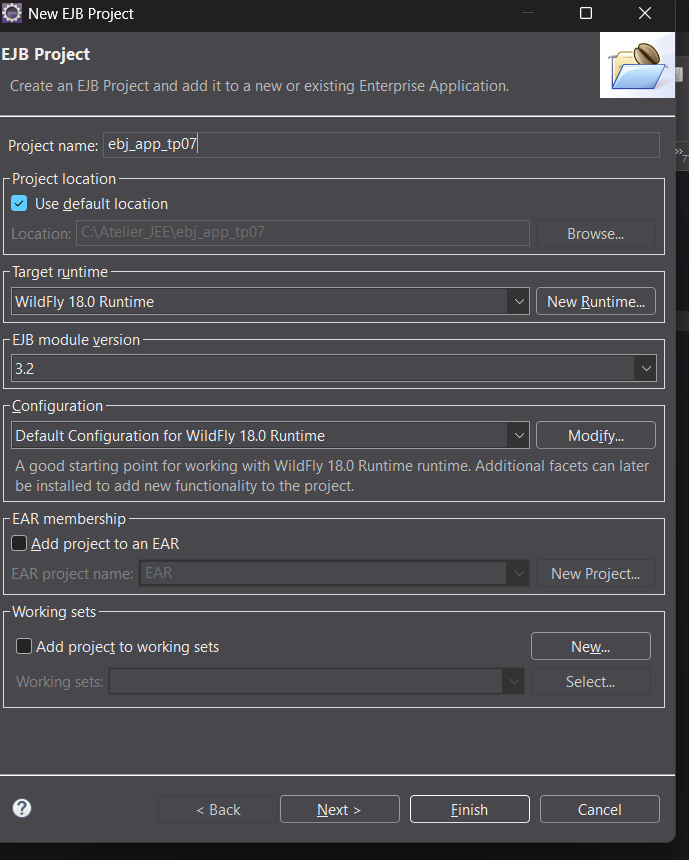


Figure 74Création d'un projet EJB "ejb\_app\_tp07"

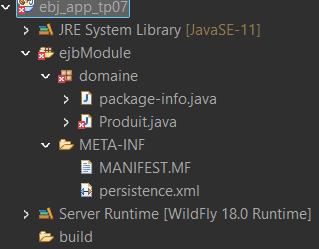
* Ajouter de la facet JPA
* A screenshot of a computer

  AI-generated content may be incorrect.

Figure 75• Ajouter de la facet JPA

**Partie D : Premier composant EJB Entity**

1. **Création de l'entité JPA**
   * Création d'une classe "Produit" dans le package "domaine"



* + Définition des attributs : id (Long, clé primaire), designation (String), prix (double), quantite (int)

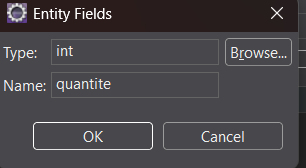


Figure 76CREATION DE ATTRRIBUT



Figure 77 les attribut de produit

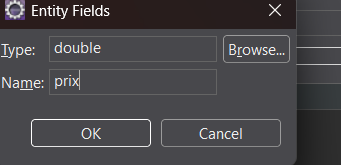


Figure 78 creation attribut prix

* + Déploiement du projet pour vérifier la création automatique de la table "produit"

## Conclusion

Ce TP nous a permis de mettre en place un environnement complet pour le développement d'applications JEE avec persistance des données. Nous avons configuré le serveur WildFly, créé une source de données, puis développé une première entité JPA capable de se mapper automatiquement à une table dans la base de données. Les annotations JPA nous ont permis de personnaliser ce mapping selon nos besoins spécifiques.

Ce travail constitue une base solide pour le développement d'applications d'entreprise Java utilisant la persistance JPA pour la gestion des données.

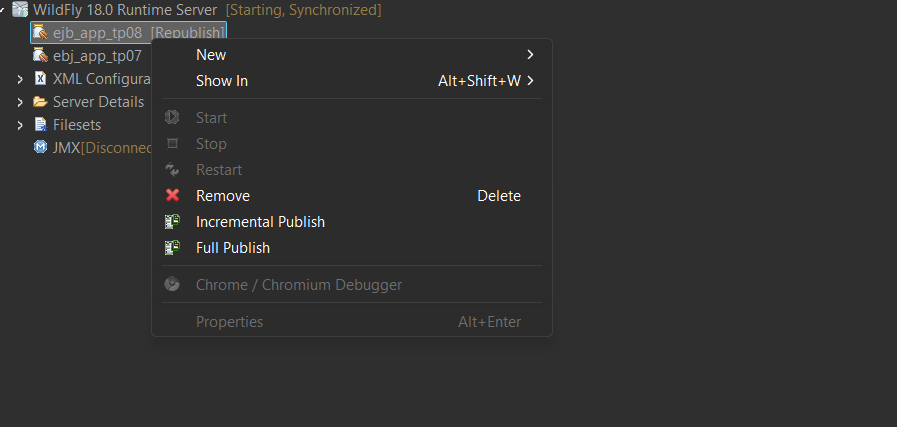
**Compte rendu : Atelier JEE - TP08 - Les composants « EJB Session »**

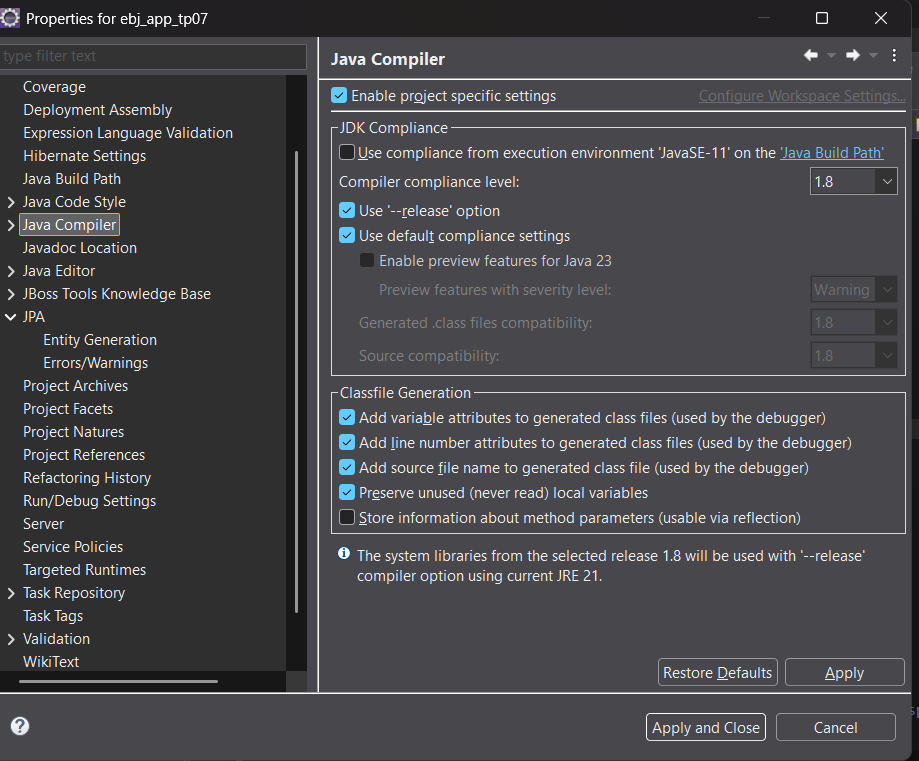
**Introduction**

L'atelier JEE TP08, réalisé à l’Institut Supérieur des Études Technologiques de Sfax, se concentre sur l’utilisation des composants **EJB Session** pour effectuer des traitements sur des données. L’objectif principal est de créer, configurer et invoquer des services EJB à partir de clients Java (lourd) et web, en utilisant le serveur **WildFly 18** et, dans un cas spécifique, **Tomcat**. Ce compte rendu résume les étapes clés, les concepts abordés et les résultats obtenus.

**A. Création du projet EJB**

* Un projet EJB nommé **ejb\_app\_tp08** a été créé dans Eclipse, associé à **WildFly 18**.



* Le niveau de conformité du compilateur a été configuré à **Java 1.8**.

A screenshot of a computer program

AI-generated content may be incorrect.

**B. Création de l’entité JPA (couche domaine)**

* Les composants JPA ont été ajoutés au projet, incluant le fichier **persistence.xml** configuré avec la datasource **MySQLDS**.
* Un package **domaine** a été créé, contenant la classe **Produit.java**, représentant une entité JPA.

A black background with white text

AI-generated content may be incorrect.

* Lors de l’exécution du projet sur WildFly, la table **produit** a été automatiquement créée dans la base de données MySQL grâce à la propriété drop-and-create.

**. Création de l’EJB Session (couche services)**

* Un package **services** a été créé pour accueillir les interfaces et la classe EJB.

A screen shot of a computer

AI-generated content may be incorrect.

* Deux interfaces ont été définies :
  + **ProduitSessionRemote** (pour un accès distant).

A screen shot of a computer program

AI-generated content may be incorrect.

* + **ProduitSessionLocal** (pour un accès local).

A screen shot of a computer screen

AI-generated content may be incorrect.

* + Ces interfaces exposent des méthodes CRUD : addProduit, updateProduit, deleteProduit, getProduit, getAllProduits.
* Une classe **ProduitSession** a été implémentée, marquée comme **@Stateless** avec le nom JNDI **PR**. Elle utilise **EntityManager** pour gérer la persistance des entités **Produit**.

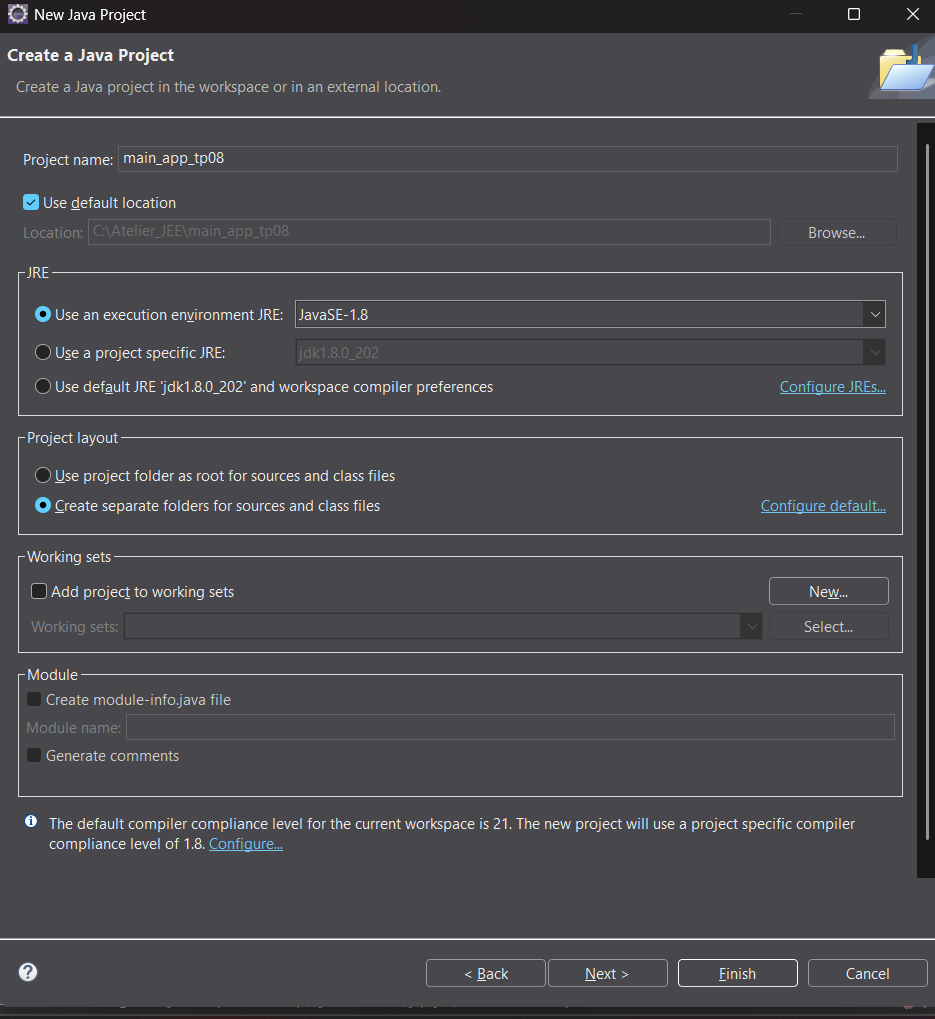
A screen shot of a computer program

AI-generated content may be incorrect.

* Le projet a été publié sur WildFly, et les bindings JNDI ont été vérifiés dans la console, notamment : ejb:/ejb\_app\_tp08/PR!services.ProduitSessionRemote.

**. Création d’un client Java lourd**

* Un projet Java nommé **main\_app\_tp08** a été créé.



* La bibliothèque **jboss-client.jar** a été ajoutée pour permettre l’accès au serveur WildFly.

A black screen with white text

AI-generated content may be incorrect.

* Deux fichiers de configuration ont été inclus dans le dossier src :
  + **jboss-ejb-client.properties** : pour les paramètres de connexion au serveur.
  + **jndi.properties** : pour spécifier le contexte JNDI.

A screenshot of a computer

AI-generated content may be incorrect.

* Une classe **ClientEJBRemote** a été créée pour invoquer les méthodes du bean **ProduitSessionRemote**. Elle a permis d’ajouter un produit et d’afficher la liste des produits.
* Une erreur initiale NoInitialContextException a été corrigée en ajoutant les fichiers de propriétés.

**E. Création d’un client web avec WildFly**

* Un projet web **web\_app\_tp08** a été créé, configuré pour WildFly 18.

A screenshot of a computer program

AI-generated content may be incorrect.

* Le projet EJB **ejb\_app\_tp08** a été exporté sous forme de JAR et placé dans le dossier WebContent/WEB-INF/lib.

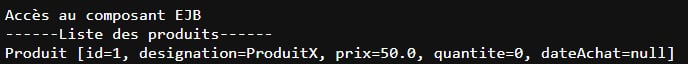
A screenshot of a computer

AI-generated content may be incorrect.

* Une servlet **ClientWebEJB** a été créée dans le package **controleur**. Elle utilise l’annotation @EJB pour injecter une référence au bean **ProduitSessionRemote** et appelle les méthodes addProduit et getAllProduits.

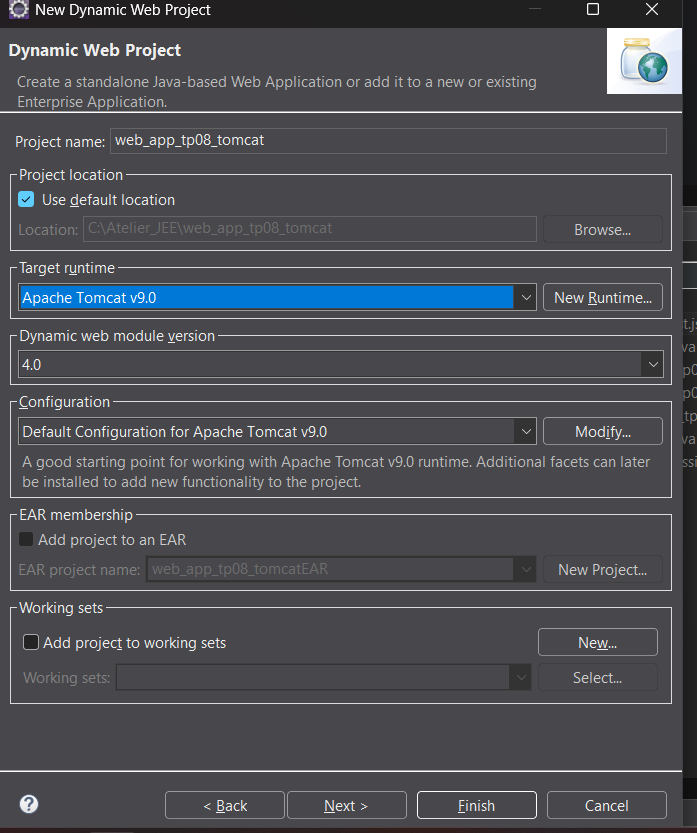
A screen shot of a computer

AI-generated content may be incorrect.

* L’exécution de la servlet via l’URL http://localhost:8080/web\_app\_tp08/ClientWebEJB a affiché la liste des produits.

**F. Création d’un client web avec Tomcat**

* Un projet web **web\_app\_tp08\_tomcat** a été créé, configuré pour **Tomcat 9.0**.



* Comme pour le client WildFly, le JAR du projet EJB et **jboss-client.jar** ont été ajoutés au dossier WebContent/WEB-INF/lib
* Une servlet **ClientWebEJBTomcat** a été créée. Contrairement à la version WildFly, elle utilise **InitialContext** et un objet **Hashtable** pour charger les propriétés de connexion, car Tomcat ne supporte pas l’annotation @EJB.

A screenshot of a computer program

AI-generated content may be incorrect.

A computer screen shot of text

AI-generated content may be incorrect.

* L’exécution de la servlet a affiché la liste des produits, confirmant l’accès distant au bean EJB sur WildFly.

A math equation with numbers

AI-generated content may be incorrect.

# Compte Rendu : Atelier JEE – TP09

## Objectif du TP

L'objectif de ce TP est de :

* Manipuler la relation N-1 entre deux entités EJB.
* Utiliser l’annotation `@ManyToOne`.
* Comprendre et implémenter les comportements de cascade : `MERGE`, `PERSIST`, `REMOVE`.

## A. Création du projet EJB

Projet EJB nommé `ejb\_app\_tp09`, associé à WildFly 18.

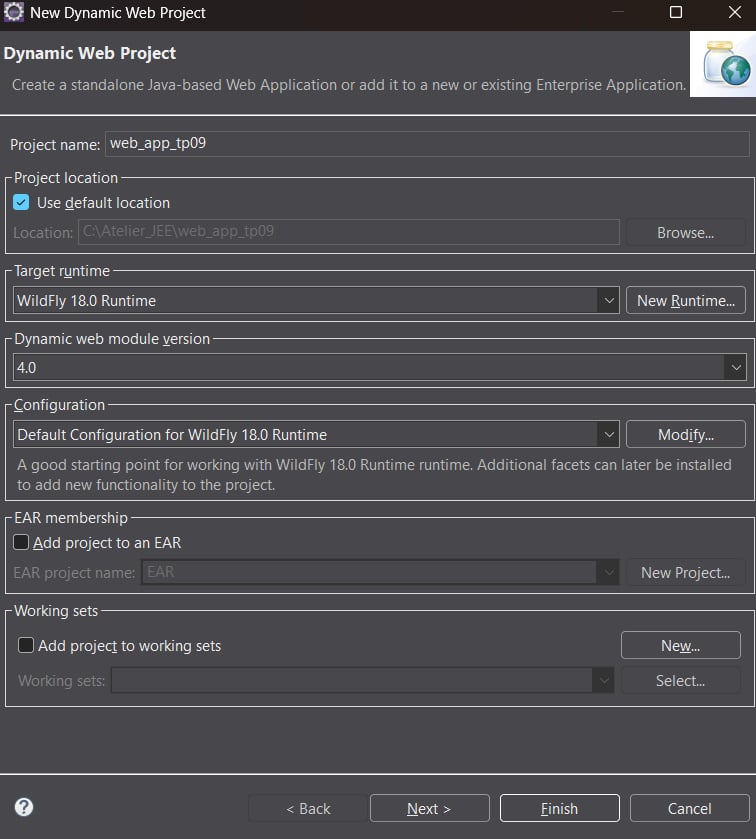


Figure 79 Projet EJB nommé `ejb\_app\_tp09`

* Configuration du compilateur à Java 1.8.

Ajout du Facet JPA et remplacement du fichier `persistence.xml`.\*

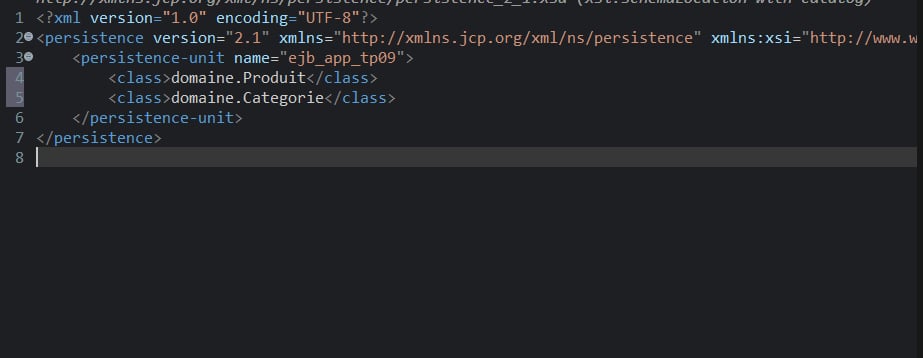


Figure 80 persistence.xml

Création d'une base de données `mysqldb\_relations` et configuration d'une datasource `MySqlDSRelations` via la console d'administration WildFly.

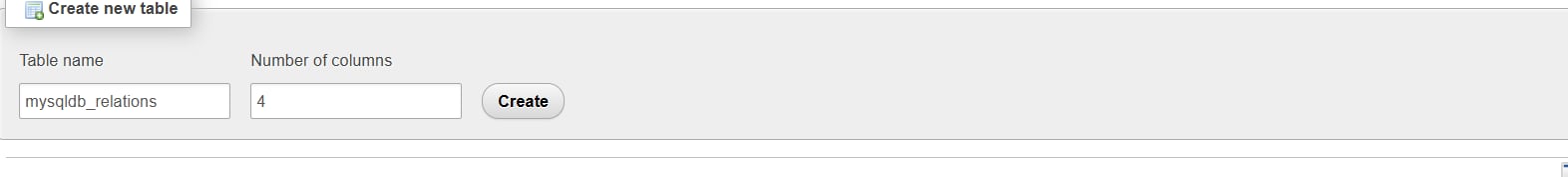


Figure 81 mysqldb\_relations

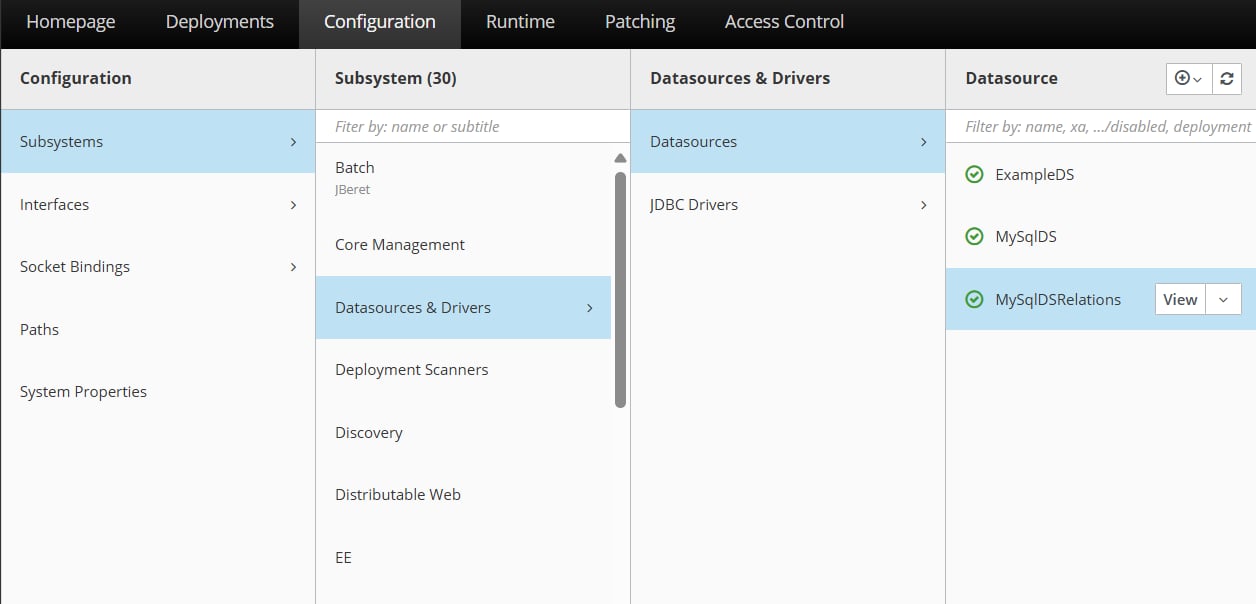


Figure 82 configuration d'une datasource `MySqlDSRelations`

## B. Conception des entités

Produit : `id`, `designation`, `prix`, `quantite`, `dateAchat`, `categorie`.

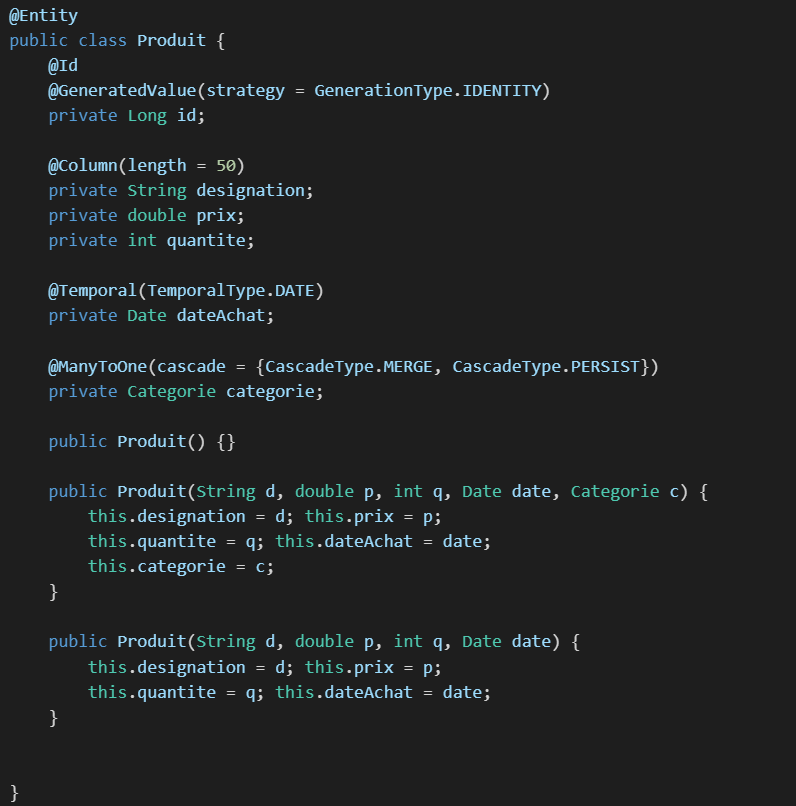


Figure 83 Produit.java

Categorie : `id`, `libelle`.

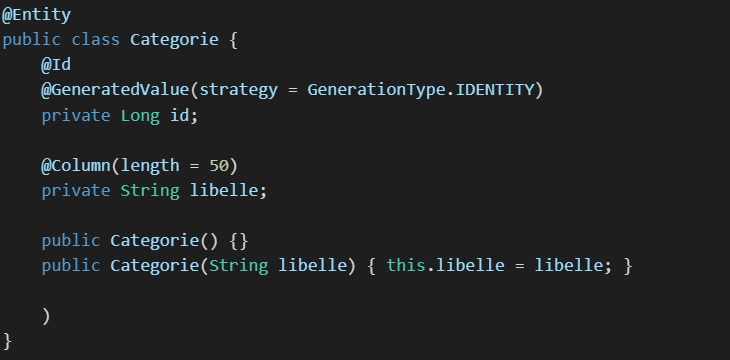
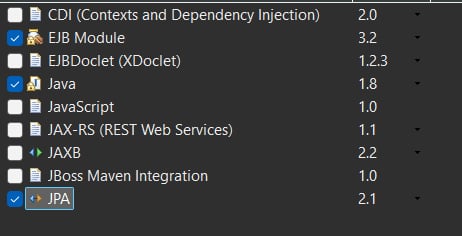


Figure 84 categorie.java

* Relation `@ManyToOne` dans Produit vers Categorie.
* Personnalisation des entités avec annotations JPA (@Id, @GeneratedValue, @Column, @Temporal).



* Ajout de constructeurs et méthodes `toString()`.

## C. Création des EJB Session

* EJB créés : `ProduitSession`, `CategorieSession` (et interfaces Remote/Local).

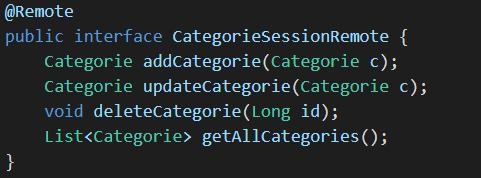


Figure 85 Interfaces CategorieSession

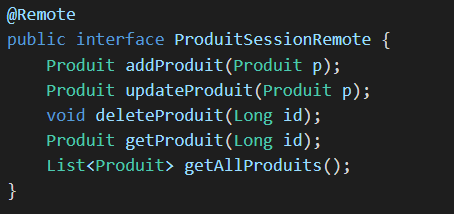


Figure 86Interfaces ProduitSession



Figure 87 Implémentation CategorieSession

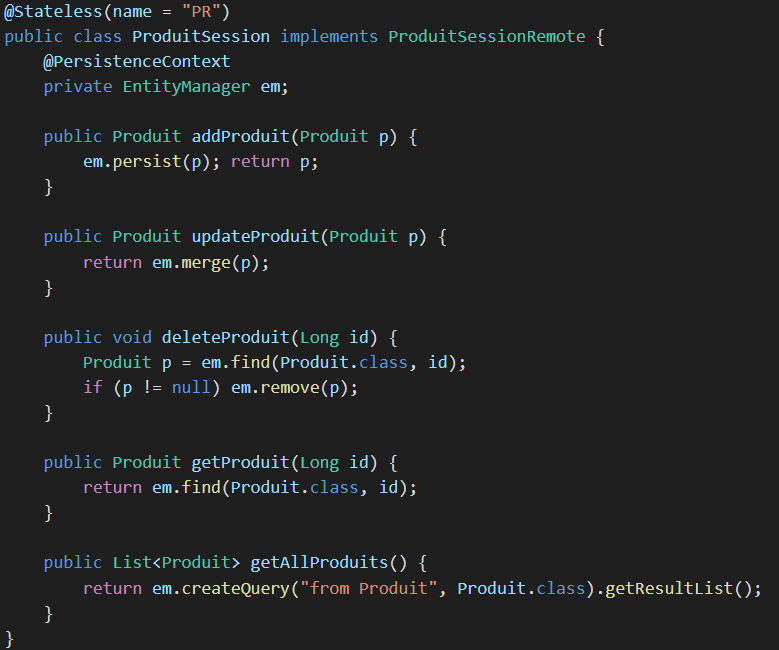


Figure 88 Implémentation ProduitSession

* Publication sur WildFly et vérification via la console (JNDI).

## D. Client Java (Lourd)

* Projet client `main\_app\_tp09` :

- Dépendance vers le projet EJB.

- Ajout du `jboss-client.jar` et fichiers de configuration.

- Classe `ClientEJBRemote` : test des méthodes distantes d’ajout, mise à jour, suppression.

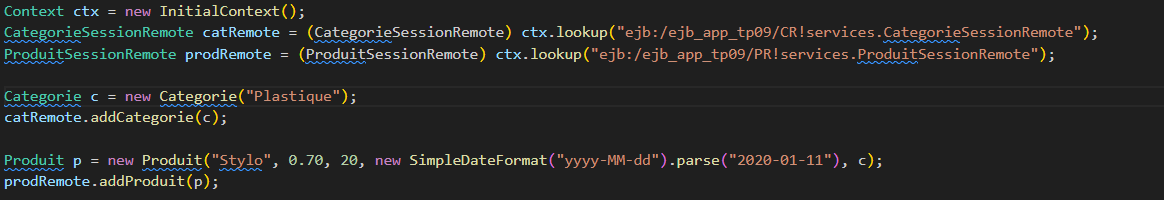


Figure 89 Extrait de ClientEJBRemote.java

* Test des relations produit/categorie et comportement de cascade.

## E. Comportements de Cascade

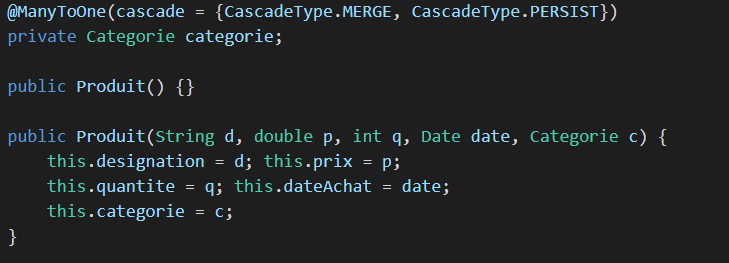


Figure 90 Comportements de Cascade

* CascadeType.MERGE : mise à jour d'une catégorie non enregistrée.
* CascadeType.PERSIST : insertion d'une nouvelle catégorie avec un nouveau produit.
* CascadeType.REMOVE : suppression d'un produit et de sa catégorie (non recommandée car risque de violations d'intégrité).

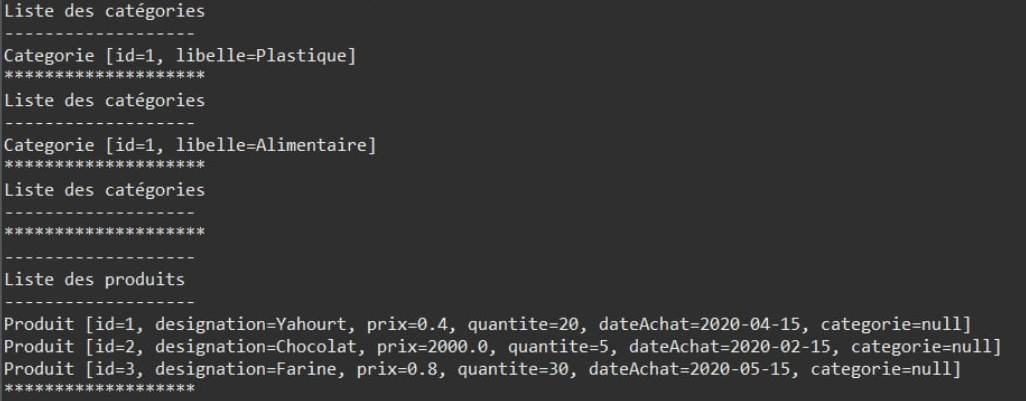
## F. Client Web

* Projet web `web\_app\_tp09` associé à WildFly.
* Export du projet EJB dans `WEB-INF/lib`.
* Création de pages JSP pour :

- Ajout de catégories et produits.

- Affichage des listes avec options de modification et suppression.

- Association de produits à des catégories.



## Compétences Acquises

* Maîtrise des entités JPA et de la relation @ManyToOne.
* Mise en place de comportements de cascade.
* Utilisation des EJB Session en local et distant.
* Création de clients lourds et web pour tester les EJB.
* Déploiement sur WildFly.