RECONNUE PAR L'ÉTAT

SYSTÈME DE GESTION D'INVENTAIRE AVEC ACCÈS DISTANT

UTILISANT RMI (REMOTE METHOD INVOCATION)

TAOUAF AHMED
2GI

Tables des Matières

Introduction

Le projet Système d'Inventaire a été conçu pour gérer efficacement les produits d'un inventaire à travers un système distribué utilisant RMI (Remote Method Invocation) en Java.

L'objectif principal de ce projet est de fournir une solution robuste et flexible pour l'ajout, la modification, la suppression, la recherche et l'affichage des produits dans un inventaire. Le système permet également une gestion centralisée de la base de données en utilisant MySQL, tout en garantissant la communication entre un serveur central et un client via des méthodes distantes.

Ce rapport décrit en détail le processus de développement du projet, les solutions mises en œuvre pour surmonter les défis rencontrés, ainsi que les résultats obtenus.

Architecture

L'architecture du projet repose sur une communication en client-serveur utilisant le protocole RMI (Remote Method Invocation) pour gérer un inventaire de produits. Cette architecture permet à un client de se connecter à un serveur distant pour effectuer des opérations telles que l'ajout, la modification, la suppression, la recherche et l'affichage des produits. Le projet est organisé en plusieurs composants:

Client : Une interface utilisateur qui permet d'interagir avec le serveur via des méthodes exposées par l'interface distante.

Serveur : La logique métier et le stockage des données. Il implémente les opérations disponibles pour gérer l'inventaire.

RMI : Le protocole de communication qui connecte le client et le serveur, permettant l'appel de méthodes distantes.

Les diagrammes UML (diagramme de classe et diagrammes de séquence) détaillent la structure et le fonctionnement de cette architecture.

01 Diagramme de Classes

Le diagramme de classe représente la structure statique de l'application. Les principales classes et leurs relations sont illustrées comme suit

InterfaceInventaire

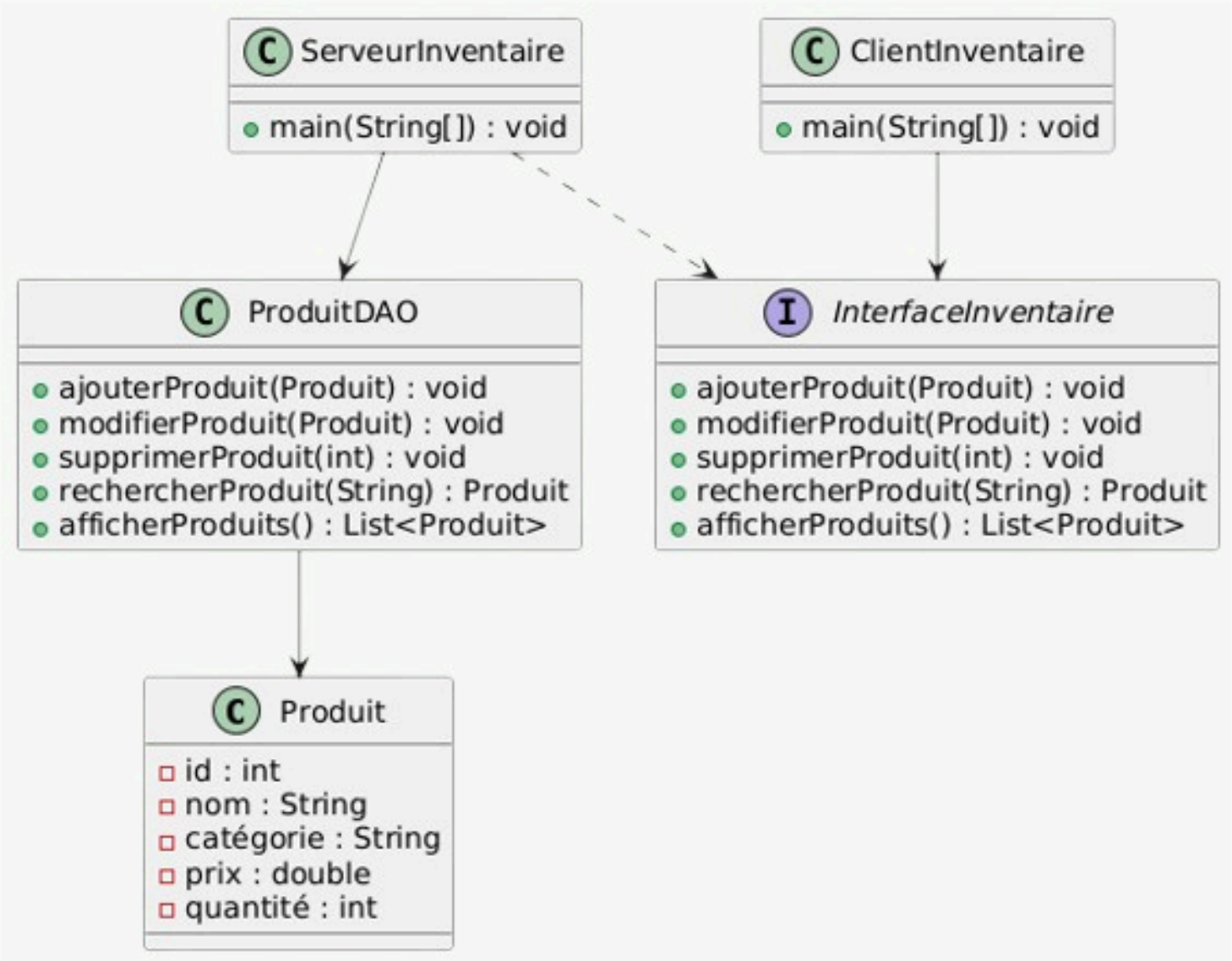
- C'est une interface distante (Remote) qui expose les méthodes disponibles pour la gestion de l'inventaire.
- Méthodes :
  - ajouterProduit : Permet d'ajouter un nouveau produit.
  - modifierProduit : Modifie un produit existant en fonction de son ID.
  - supprimerProduit : Supprime un produit en fonction de son ID.
  - rechercherProduits : Recherche des produits par catégorie ou autre critère.
  - afficherProduits : Affiche tous les produits de l'inventaire.

Produit

- Représente un produit dans l'inventaire avec ses attributs :
  - id : Identifiant unique.
  - nom : Nom du produit.
  - categorie : Catégorie à laquelle appartient le produit.
  - quantite : Quantité disponible.
  - prix : Prix unitaire.

ClientInventaire

- Représente le programme client qui utilise l'interface InterfaceInventaire pour interagir avec le serveur.
- Il contient les méthodes pour gérer l'entrée utilisateur et appeler les opérations distantes exposées par le serveur.

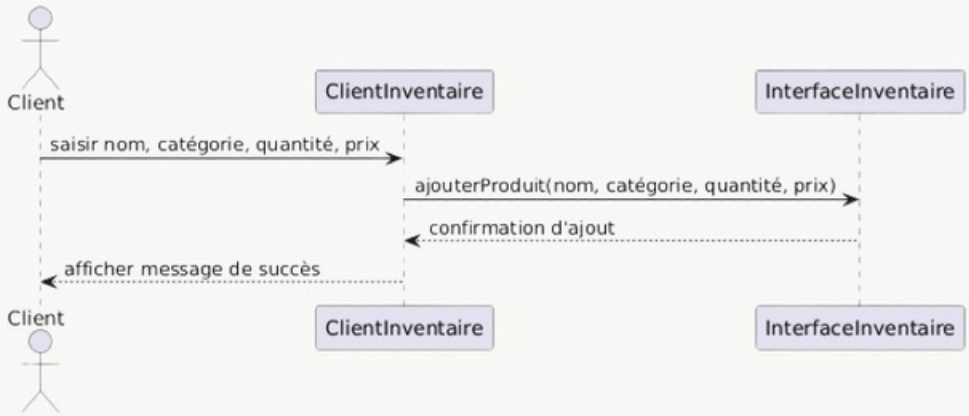

ServeurInventaire
main(String[]) : void
ClientInventaire
main(String[]) : void
ProduitDAO
ajouterProduit(Produit) : void
modifierProduit(Produit) : void
supprimerProduit(int) : void
rechercherProduit(String) : Produit
afficherProduits() : List<Produit>
InterfaceInventaire
ajouterProduit(Produit) : void
modifierProduit(Produit) : void
supprimerProduit(int) : void
rechercherProduit(String) : Produit
afficherProduits() : List<Produit>
Produit
id : int
nom : String
catégorie : String
prix : double
quantité : int

02 Diagrammes de Séquence

Les diagrammes de séquence montre l'interaction entre le client et le serveur distant, illustrant l'appel de méthodes à distance, la transmission des données et la réponse du serveur, avec les objets servant d'intermédiaires dans la communication.

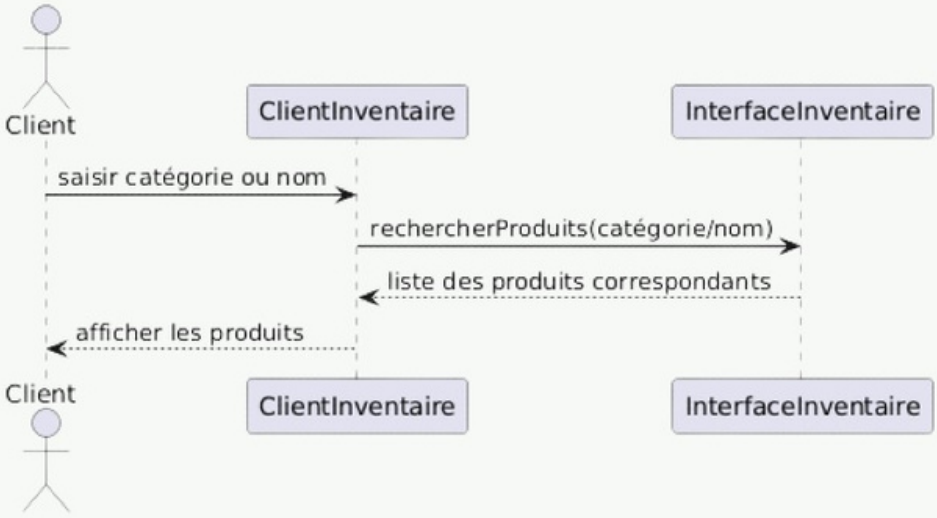
Ajout d’un produit

- Ce diagramme montre comment un client interagit avec le serveur pour ajouter un produit en spécifiant le nom, la catégorie, la quantité et le prix. Le serveur enregistre les informations dans l'inventaire.



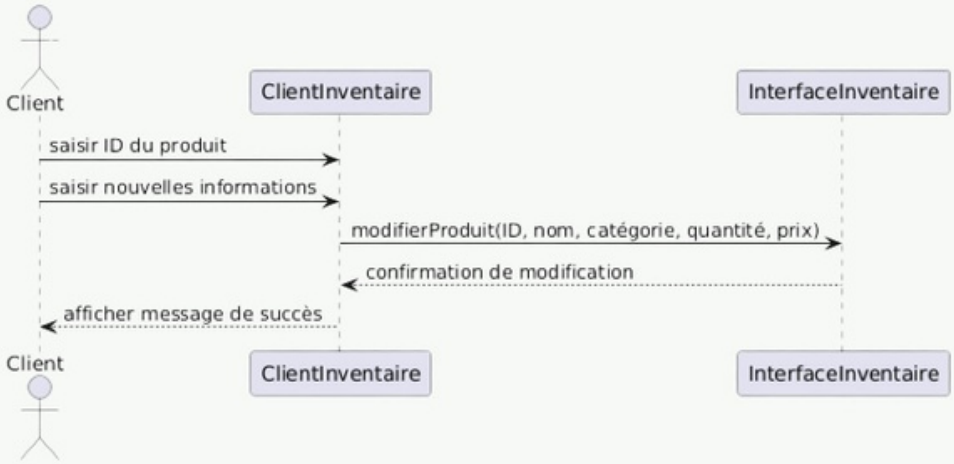
Recherche d’un Produit

- Ce diagramme illustre comment un client peut rechercher des produits dans l'inventaire en saisissant un nom ou une catégorie. Le serveur retourne les résultats correspondants.



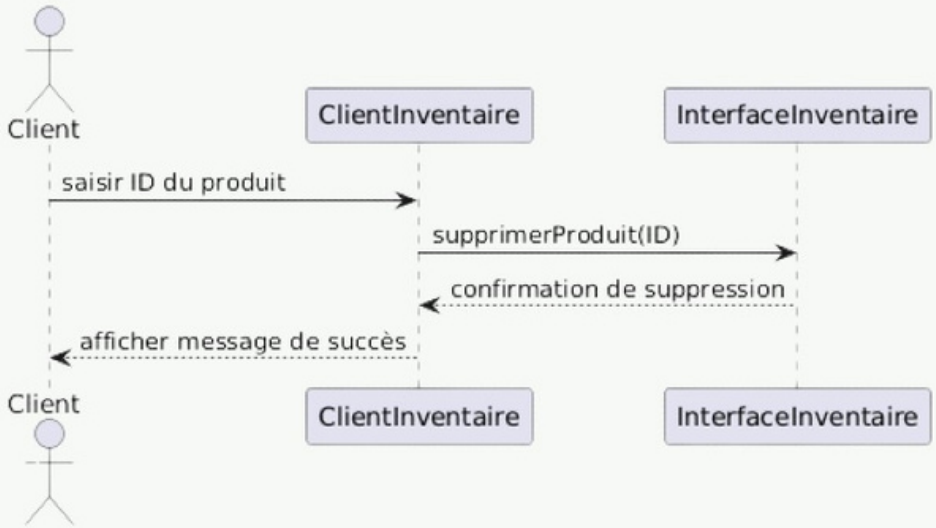
Modification d’un produit

- Ce diagramme décrit le processus par lequel un client modifie un produit existant. Le client fournit l'ID du produit et les nouvelles informations (nom, catégorie, quantité, prix), que le serveur met à jour dans l'inventaire.



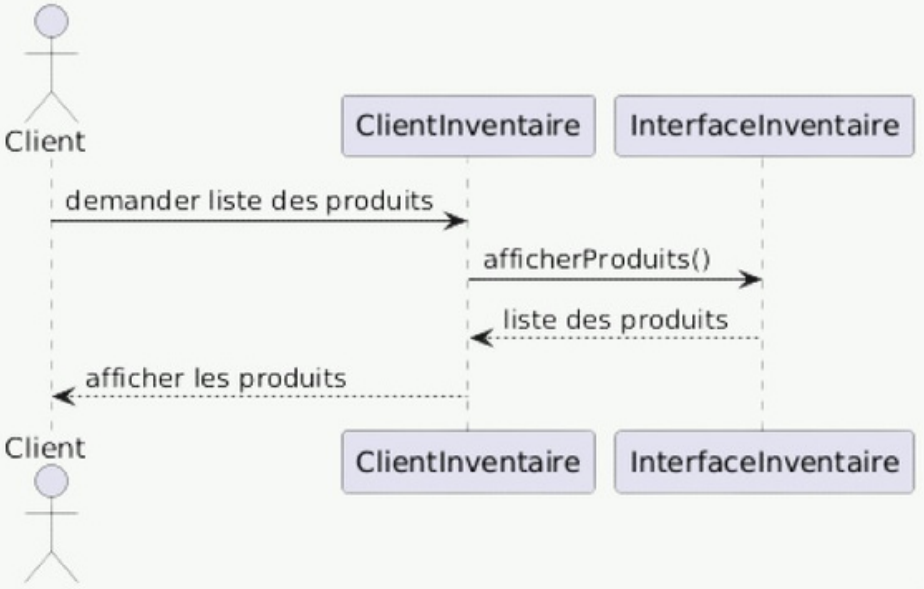
Suppression d’un Produit

- Ce diagramme détaille comment un client peut supprimer un produit en fournissant son ID. Le serveur identifie le produit et le supprime de l'inventaire.



Affichage des Produits

- Ce diagramme montre comment un client demande la liste complète des produits, et le serveur retourne les informations des produits disponibles dans l'inventaire.



Explication Technique

L'architecture utilisée dans ce projet est basée sur Remote Method Invocation (RMI), qui permet à un programme Java d'invoquer des méthodes sur un objet distant.

Serveur

- Le serveur implémente l'interface **InterfaceInventaire** et expose des méthodes telles que **ajouterProduit**, **supprimerProduit**, **rechercherProduits**, **afficherProduits**, et **modifierProduit**. Ces méthodes permettent au client d'interagir avec l'inventaire, en ajoutant, modifiant, supprimant ou recherchant des produits. Le serveur utilise RMI pour rendre ces méthodes accessibles à distance.

- Le serveur est configuré pour écouter sur un port spécifique (dans ce cas, le port 1099) et attendre les connexions entrantes des clients. Lorsqu'un client se connecte, il peut appeler des méthodes sur le serveur via les objets distants, tels que **ServeurInventaire**, qui sont créés et liés au registre RMI via **Naming.rebind()**.

Client

- Le client se connecte au serveur en utilisant l'URL RMI du serveur (par exemple, **rmi://localhost:1099/inventaire**). Le client peut invoquer des méthodes distantes sur le serveur, telles que l'ajout, la modification, la suppression ou l'affichage des produits, en appelant les méthodes de l'interface **InterfaceInventaire**. Le client dispose d'un menu interactif qui permet à l'utilisateur de sélectionner différentes actions à effectuer. Le client communique avec le serveur via des appels RMI, et l'interface utilisateur est implémentée dans la classe **ClientInventaire** avec des entrées de texte pour collecter les informations des utilisateurs.

01 Fonctionnement du RMI

Le RMI permet une communication transparente entre le client et le serveur, même s'ils sont exécutés sur des machines différentes.

RMI Registre

- Le serveur crée un objet distant et le lie au registre RMI sous le nom inventaire. Le registre RMI est un service qui permet aux clients de rechercher et de se connecter à des objets distants en utilisant un nom logique.

Binding

- Le serveur lie l'objet **ServeurInventaire** (qui implémente l'interface **InterfaceInventaire**) au registre RMI à l'adresse **rmi://localhost:1099/inventaire**.

Communication Client-Serveur

- Le client, lorsqu'il souhaite effectuer une action (ajouter, modifier, supprimer, rechercher ou afficher des produits), utilise l'interface **InterfaceInventaire** pour accéder aux méthodes du serveur via RMI.

02 Base de Données

Le projet utilise une base de données MySQL pour stocker les informations relatives aux produits de l'inventaire. La base de données permet de maintenir une persistance des données et d'assurer une gestion des produits efficaces

Structure de la base de données

- La base de données contient une table nommée produits qui contient les informations des produits. Cette table est définie avec les colonnes suivantes :

    - **id**: identifiant unique du produit (clé primaire).
    - **nom**: le nom du produit.
    - **categorie**: la catégorie à laquelle appartient le produit.
    - **quantite**: la quantité disponible du produit.
    - **prix**: le prix du produit.

La base de données est hébergée localement sur le serveur MySQL, et le serveur Java communique avec cette base pour effectuer des opérations telles que l'ajout, la modification, la suppression, et la recherche de produits.

DAO (Data Access Object)

- Le projet utilise un modèle DAO pour séparer la logique de persistance des données de la logique métier. La classe ProduitDAO est responsable des interactions avec la base de données. Les principales opérations que la classe ProduitDAO effectue sont :

    - Ajout de produit : La méthode ajouterProduit() insère un nouveau produit dans la base de données avec les informations fournies (nom, catégorie, quantité, prix).
    - Suppression de produit : La méthode supprimerProduit() supprime un produit de la base de données en fonction de son identifiant.
    - Recherche de produits : La méthode rechercherProduits() permet de rechercher des produits par catégorie.
    - Affichage de tous les produits : La méthode afficherProduits() récupère et affiche tous les produits de la base de données.
    - Modification de produit : La méthode modifierProduit() met à jour les informations d'un produit existant (nom, catégorie, quantité, prix).

03 Justification Des Choix Technologiques

JDBC (Java Database Connectivity)

- JDBC a été choisi pour la gestion de la communication entre l'application et la base de données MySQL. Ce choix permet une interaction simple et efficace avec la base de données relationnelle, permettant d'exécuter des requêtes SQL pour manipuler les données. JDBC est une technologie robuste et largement utilisée dans le développement Java, garantissant ainsi une compatibilité étendue et une gestion transparente des connexions à la base de données.

RMI (Remote Method Invocation)

- RMI a été choisi pour permettre l'accès distant entre le client et le serveur. Il facilite l'appel de méthodes sur un objet distant comme si elles étaient locales. Cela permet de concevoir un système distribué où les clients peuvent interagir avec les services du serveur à distance.

MySQL

- MySQL a été choisi comme système de gestion de base de données relationnelle en raison de sa fiabilité, de ses performances et de son large écosystème. Il offre une gestion efficace des données structurées, ce qui est essentiel pour une application de gestion d'inventaire nécessitant des requêtes rapides et des transactions fiables. MySQL est également bien supporté et couramment utilisé dans des projets de taille petite à moyenne, avec une communauté active et une documentation complète.

Installation Et Exécution

L'installation et l'exécution de l'application nécessitent plusieurs étapes pour s'assurer que tous les pré-requis sont en place et que l'application fonctionne correctement. Cette application repose sur plusieurs technologies clés, notamment Java, MySQL, et le mécanisme RMI (Remote Method Invocation).

Afin de garantir une configuration correcte, vous devrez installer Java, configurer une base de données MySQL, et vous assurer que le serveur RMI est opérationnel pour la communication entre le client et le serveur. Une fois ces éléments en place, vous pourrez facilement démarrer le projet sur votre machine en suivant quelques étapes simples.

Dans cette section, nous détaillons les pré-requis nécessaires, ainsi que les étapes à suivre pour installer et exécuter l'application de manière fluide.

01 Pré-requis

Java Development Kit (JDK)

- Version : JDK 8 ou supérieure
- Description : Le projet nécessite le JDK pour compiler et exécuter le code Java. Il est nécessaire d'installer le JDK sur votre machine avant de pouvoir lancer l'application.
- Téléchargement : Vous pouvez télécharger le JDK depuis le site officiel de Oracle ou utiliser un JDK open source tel que OpenJDK.

MySQL

- Version : MySQL 5.7 ou supérieure
- Description : MySQL est utilisé comme système de gestion de base de données pour stocker les informations sur les produits. Assurez-vous que MySQL est installé et en fonctionnement sur votre machine.
- Téléchargement : MySQL peut être téléchargé à partir du site officiel de MySQL.

Bibliothèque JDBC

- Description : Le projet utilise JDBC pour se connecter à la base de données MySQL. Le fichier JAR pour JDBC est inclus dans les bibliothèques de votre JDK, mais assurez-vous d'avoir également le Connecteur JDBC MySQL si nécessaire.

IDE (environnement de développement intégré)

- Recommandation : Utilisez un IDE comme IntelliJ IDEA, Eclipse ou NetBeans pour faciliter le développement et l'exécution du projet. Ces IDEs offrent une intégration facile avec les bibliothèques et permettent de gérer le projet Java.

Serveur RMI

- Description : L'application utilise le mécanisme RMI (Remote Method Invocation) pour la communication entre le client et le serveur. Assurez-vous que le serveur RMI est correctement configuré et fonctionne avant de lancer l'application. Le serveur doit être en marche pour accepter les requêtes du client.

02 Étapes D'installation Et D'exécution

1- Installer MySQL

- Téléchargez et installez MySQL.
- Créez une base de données inventaire dans MySQL en exécutant la commande suivante dans le terminal MySQL :

```
CREATE DATABASE inventaire;
```

2- Configurer la base de données

- Dans la base de données **inventaire**, créez une table pour stocker les produits :

```
CREATE TABLE produits (
id INT AUTO_INCREMENT PRIMARY KEY,
nom VARCHAR(255) NOT NULL,
categorie VARCHAR(255) NOT NULL,
quantite INT NOT NULL,
prix DECIMAL(10, 2) NOT NULL);
```

Cela permet de stocker les informations sur les produits dans la base de données.

3- Télécharger et configurer le projet

- Clonez ou téléchargez le code source du projet sur votre machine.
- Importez le projet dans votre IDE préféré.

4- Configurer le fichier de connexion JDBC

- Dans le fichier JDBCUtil.java, modifiez les paramètres de connexion si nécessaire:

```
private static final String URL = "jdbc:mysql://localhost:3306/inventaire";
private static final String USER = "root";
private static final String PASSWORD = "";
```

5- Compilation Du Code

- Naviguez dans le répertoire principal du projet et exécutez la commande suivante dans votre invite de commandes pour compiler le code dans :

javac -d out src/client/ClientInventaire.java src/serveur/ServeurInventaire.java src/serveur/InterfaceInventaire.java src/serveur/DAO/ProduitDAO.java src/serveur/DAO/JDBCUtil.java

- L'option -d out crée un dossier “out” pour stocker les fichiers .class compilés.

6- Démarrage du Registre RMI

- Tout d’abord on doit démarrer le Registre RMI avec la commande suivante :

start rmiregistry

- Exécutez la classe **ServeurInventaire** dans la meme chemin pour démarrer le serveur RMI avec spécification du Driver JBDC mySQL. Le serveur attendra les connexions des clients sur **rmi://localhost:1099/inventaire.**

java -cp .;../lib/mysql-connector-j-8.0.31.jar serveur.ServeurInventaire

6- Lancement du Client

- Exécutez la classe ClientInventaire pour démarrer l'application cliente. Vous pourrez interagir avec le serveur à travers un menu de gestion des produits.

java client.ClientInventaire

Conclusion Et Perspective

Le système de gestion d'inventaire avec accès distant via RMI a permis de démontrer la puissance des technologies Java en réseau, couplée à une gestion efficace des données via MySQL.

Ce projet a mis en œuvre une architecture client-serveur robuste et flexible, permettant de gérer les produits en toute simplicité et de répondre aux besoins d'une entreprise souhaitant centraliser son inventaire.

En s'appuyant sur des concepts tels que l'accès distant sécurisé et la manipulation des données relationnelles, le projet a fourni une solution fiable, performante, et extensible. Comme perspectives on peut :

- **Amélioration de l'interface utilisateur** : Actuellement basé sur une interface console, le système pourrait évoluer vers une interface graphique conviviale (JavaFX, Swing ou une interface web).
- **Support multiplateforme** : Développement d'applications mobiles et web pour permettre un accès encore plus flexible et adapté aux utilisateurs.
- **Fonctionnalités avancées** : Ajout de fonctionnalités telles que les alertes de stock faible, les rapports personnalisés, ou encore une gestion des utilisateurs avec rôles et permissions.
- **Intégration avec des systèmes externes** : Connexion avec des logiciels tiers, tels que des ERP ou des systèmes de gestion de la chaîne logistique.