Malatray Fabien
Sénéchal Damien

Rapport Mini-projet Client/Serveur

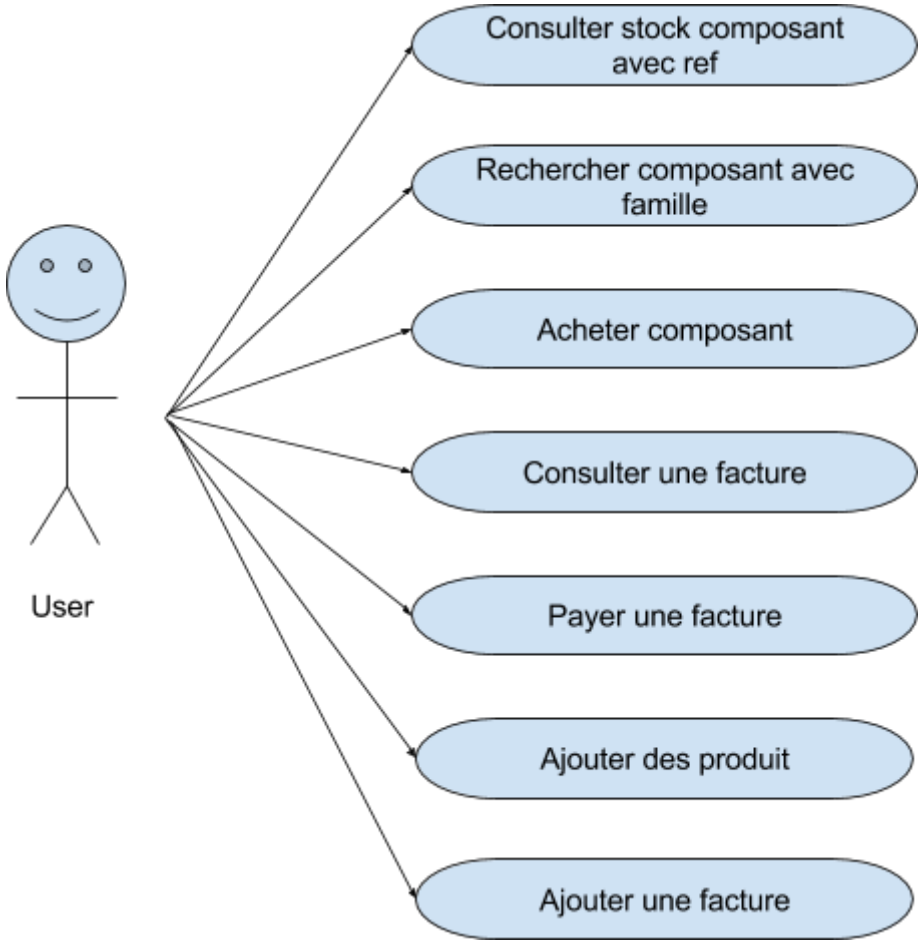
Le but de ce projet est la mise en place d'un système de gestion de stock pour un magasin de vente de marchandises.
Le vendeur possède plusieurs boutiques il faut donc séparer l'application en deux :
Un côté serveur, non accessible au public, contenant la base de données.
Un côté client accédant à distance au donné stocké côté serveur.

Analyse

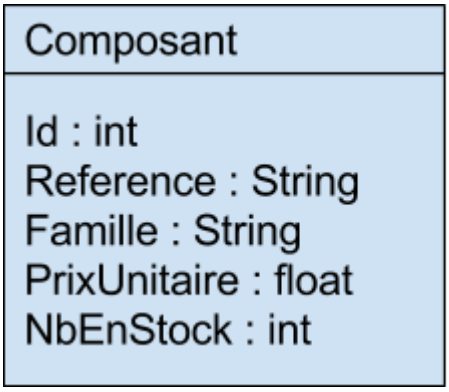
Conditions :

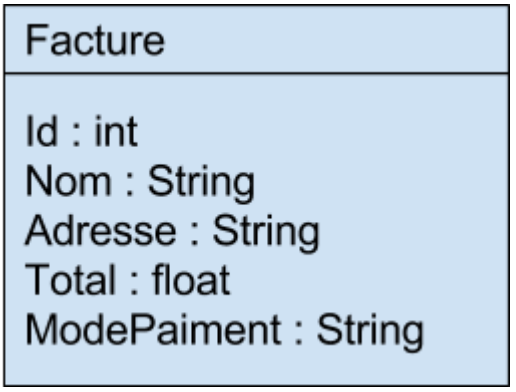
Client et serveur en Java
RMI comme middleware
MySQL comme SGBD

Diagramme d'utilisation :



Base de donnée :





Test :

En parallèle de ces analyses nous avons également effectué des tests sur le fonctionnement du middleware RMI et du SGBD MySQL.
Cela nous a permis de savoir comment initialiser un serveur RMI :

```
try {
    java.rmi.registry.LocateRegistry.createRegistry(1099);
    System.out.println("RMI registry ready.");
} catch (Exception e) {
    System.out.println("Exception starting RMI registry:");
    e.printStackTrace();
}
try {
    Addition Hello = new Addition();
    Naming.rebind("rmi://localhost/ABC", Hello);

    System.out.println("Addition Server is ready.");
} catch (Exception e) {
    System.out.println("Addition Server failed: " + e);
}
```

De savoir comment accéder au serveur depuis le client :

```
AdditionInterface hello;
try {
    hello = (AdditionInterface) Naming.lookup("rmi://localhost/ABC");
    int result = hello.add(9, 10);
    System.out.println("Result is :" + result);

} catch (Exception e) {
    System.out.println("HelloClient exception: " + e);
}
```

De savoir comment accéder à la base de données:

```
String result = "\n";
String url = "jdbc:mysql://localhost:3306/stock";
String username = "root";
String password = "";

System.out.println("Connecting database...");

try (Connection connection = DriverManager.getConnection(url, username, password)) {
    System.out.println("Database connected!");
    Statement stmt = connection.createStatement();
    String query = "SELECT * FROM composant ;";
    ResultSet rs = stmt.executeQuery(query);
    while (rs.next()) {
        result=result+rs.getString(1)+"-";
        result=result+rs.getString(2)+"-";
        result=result+rs.getString(3)+"-";
        result=result+rs.getString(4)+"-";
        result=result+rs.getString(5)+"\n";
    }
} catch (SQLException e) {
    throw new IllegalStateException("Cannot connect the database!", e);
}
```

Objets :

Pour que le client et le serveur manipulent les mêmes objets pour éviter les problèmes nous les avons défini lors de la phase d'analyse.
Nous avons définis deux objets :

- Facture
- Stock

Le client et le serveur disposent chacun d'une copie similaire de ces classes pour pouvoir travailler sur les mêmes objets et utiliser les mêmes fonctions.
Pour que le client et le serveur puissent échanger ces deux objet héritent de la classe serializable.

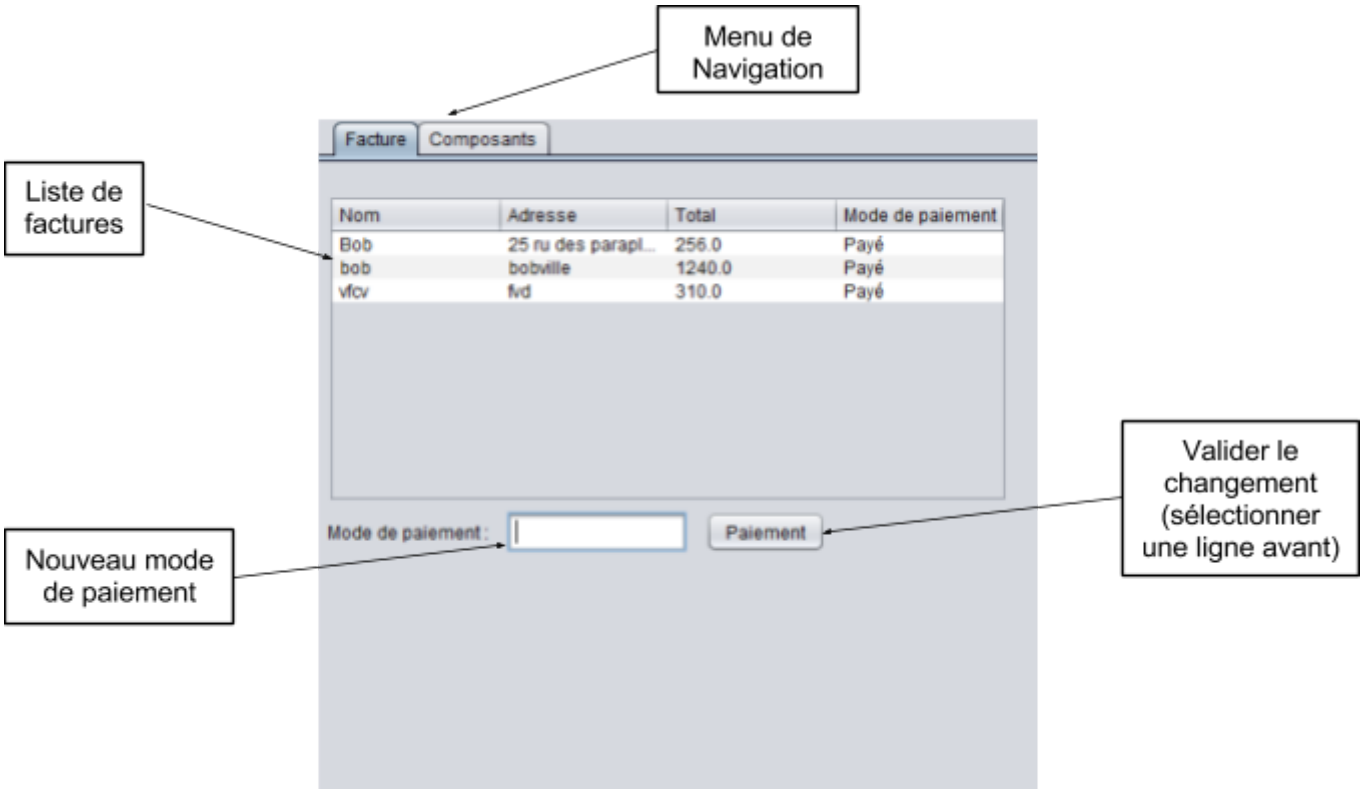
Réalisation :

Nous avons décidé d'utiliser l'ide netBeans pour réaliser le projet, car nous l'avions déjà tout les deux utilisé.
Pour pouvoir travailler sur le même projet simultanément nous avons créé un git avec GitHub.
Nous nous sommes globalement réparti le travail de façon suivante:

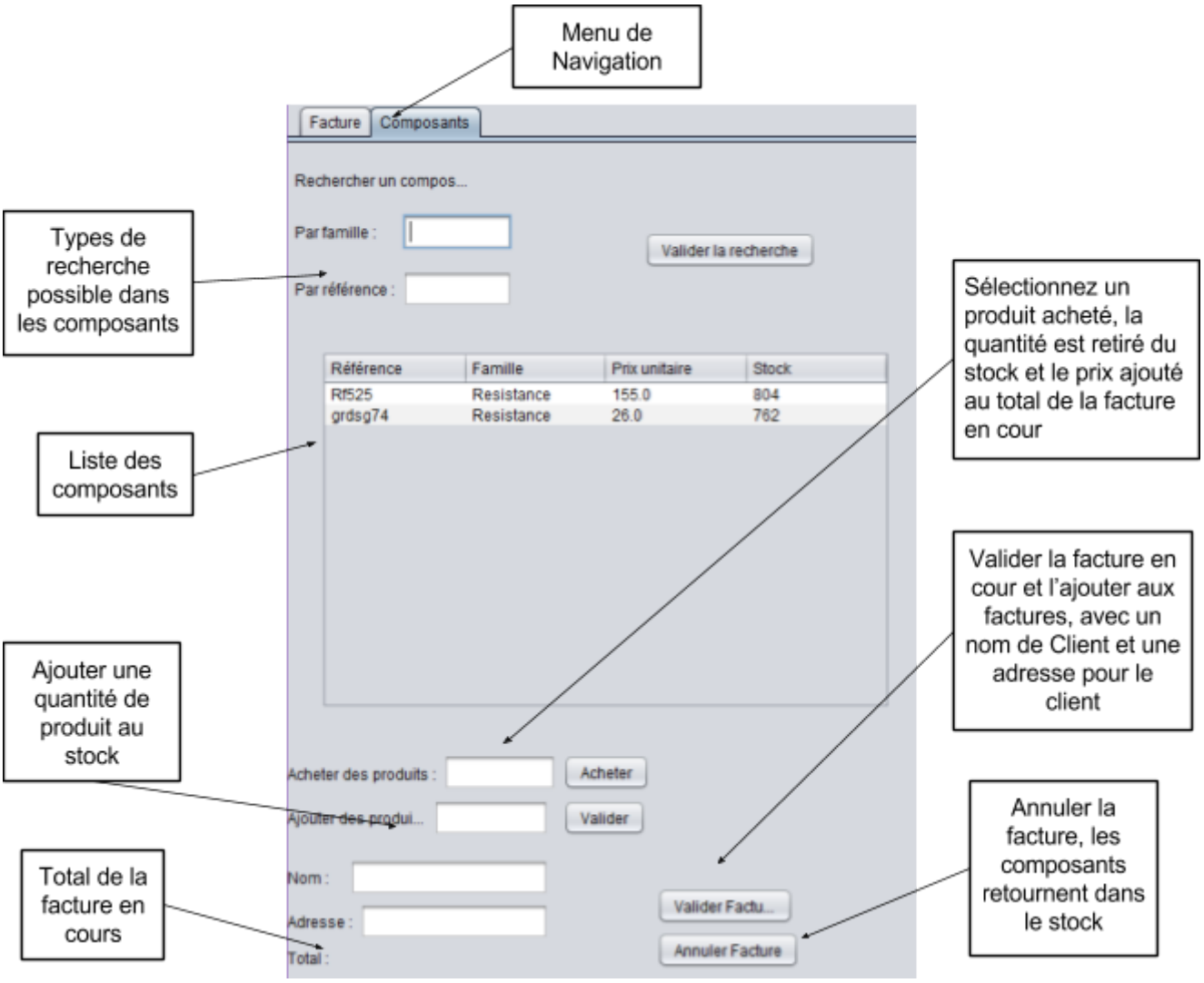
- Damien le serveur
- Fabien le client

Guide Utilisation

Factures :



Composants :


Menu de Navigation
Facture
Composants
Rechercher un compos...
Par famille :
Valider la recherche
Par référence :
Types de recherche possible dans les composants
Référence
Famille
Prix unitaire
Stock
Rf525
Resistance
155.0
804
grdsg74
Resistance
26.0
762
Liste des composants
Sélectionnez un produit acheté, la quantité est retiré du stock et le prix ajouté au total de la facture en cour
Valider la facture en cour et l'ajouter aux factures, avec un nom de Client et une adresse pour le client
Ajouter une quantité de produit au stock
Acheter des produits :
Acheter
Ajouter des produi...
Valider
Nom :
Adresse :
Total :
Valider Factu...
Annuler Facture
Total de la facture en cours
Annuler la facture, les composants retournent dans le stock

Guide D'installation (Sous Windows)

Tout d'abord il faut télécharger l'archive contenant tous les composants du projet.

Côté Serveur :

- Lancer un serveur pour base de données Mysql (Wamp par exemple) .
- Importer la base de données (avec phpMyAdmin par exemple).
- Placer le fichier ServeurGestionStock sur le serveur.

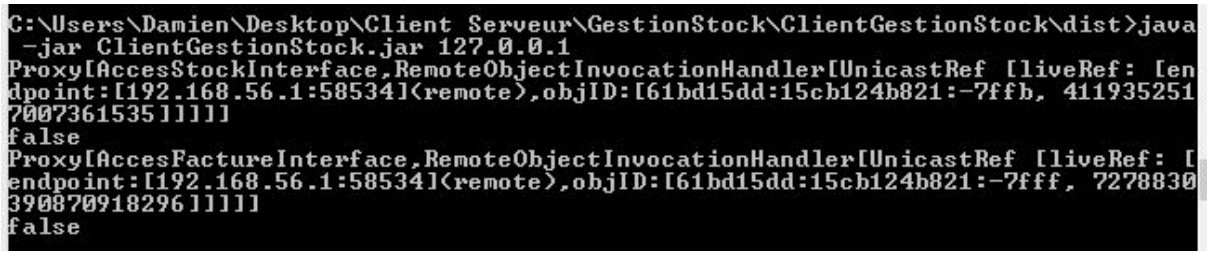
Pour lancer le serveur ouvrir une invite de commande et de commande et se placer dans le répertoire ServeurGestionStock\dist.
Entrer la commande : **java -jar ServeurGestionStock.jar [ip]**

Le serveur est lancé !

Côté Client :

- Placer le fichier ClientGestionStock sur le client

Pour lancer le client, ouvrir une invite de commande et se placer dans le répertoire ClientGestionStock\dist.
Entre la commande **: java -jar ClientGestionStock.jar [ip]**



Le client est lancé !