# Rapport de projet Pharmagest

## Sommaire :

## Introduction :

Implantée à Port-Louis, Île Maurice, depuis 2020, la pharmacie PHARMAGEST exerce ses activités à l'aide d'un système de gestion traditionnel pour la vente de ses produits pharmaceutiques. Face à la complexité croissante de la gestion de plus de 5000 références de médicaments, et dans le but de prévenir les ruptures de stock tout en optimisant ses ventes, la direction a choisi de moderniser son système.

Dans cette optique, PHARMAGEST a décidé de numériser la gestion de son stock. Pour mener à bien ce projet, elle a fait appel à une Société de Services en Ingénierie Informatique (SSII), chargée de proposer une solution adaptée aux besoins de la pharmacie et d'en assurer l'implémentation. Cette SSII est représentée par mon binôme Anne-Sophie Montenot et moi-même, dans le cadre de notre projet de développement informatique.

## Démarche du projet

- **Objectifs du projet**

Dans le cadre de sa transformation digitale, **PHARMAGEST** souhaite développer sa propre application afin d'informatiser plusieurs de ses activités clés. L'objectif principal est d'optimiser la gestion interne de la pharmacie grâce à un outil numérique performant. L'application visera donc à couvrir les domaines suivants :

1. La gestion de l'approvisionnement des médicaments
2. Le processus de vente au comptoir

3. La gestion financière globale de l’établissement

- **Objectifs de l’entreprise**

À travers ce projet, PHARMAGEST ambitionne d’atteindre plusieurs objectifs stratégiques :

1. Digitaliser la gestion du stock de médicaments pour plus d’efficacité
2. Éviter les ruptures de stock grâce à un système d’approvisionnement automatisé
3. Faciliter la vente au comptoir tout en respectant les obligations légales, notamment liées aux ordonnances
4. Optimiser la gestion de la caisse
5. Mieux suivre les indicateurs financiers tels que les ventes, les achats et les marges

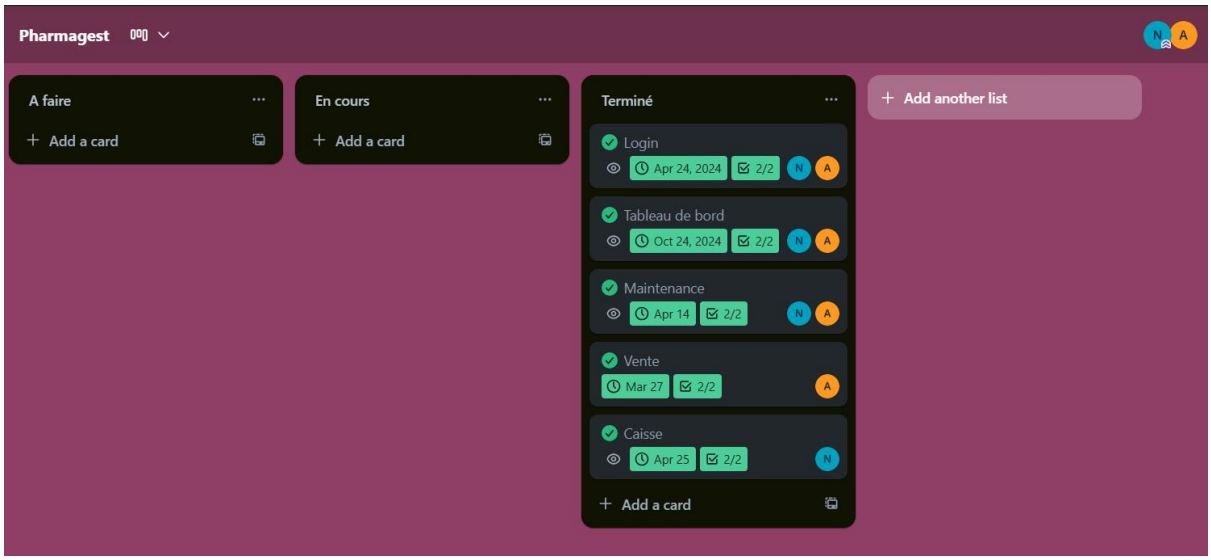
- **Mission**

Le prestataire informatique sélectionné joue un rôle central dans la réussite de ce projet. Il est chargé de :

- Analyser les besoins spécifiques de la pharmacie
- Définir les différentes étapes du projet (graphisme, ergonomie, contenu)
- Proposer un calendrier de réalisation réaliste
- Fournir les éléments de base nécessaires (textes, maquettes, logos, photos)
- Concevoir et développer l’application web selon les exigences établies
- Livrer le projet dans les délais convenus

- **Cahier de réalisation**

Pour réaliser ce projet, mon binôme et moi avons réparti les tâches en utilisant l'outil Trello ainsi qu'un tableau Excel.



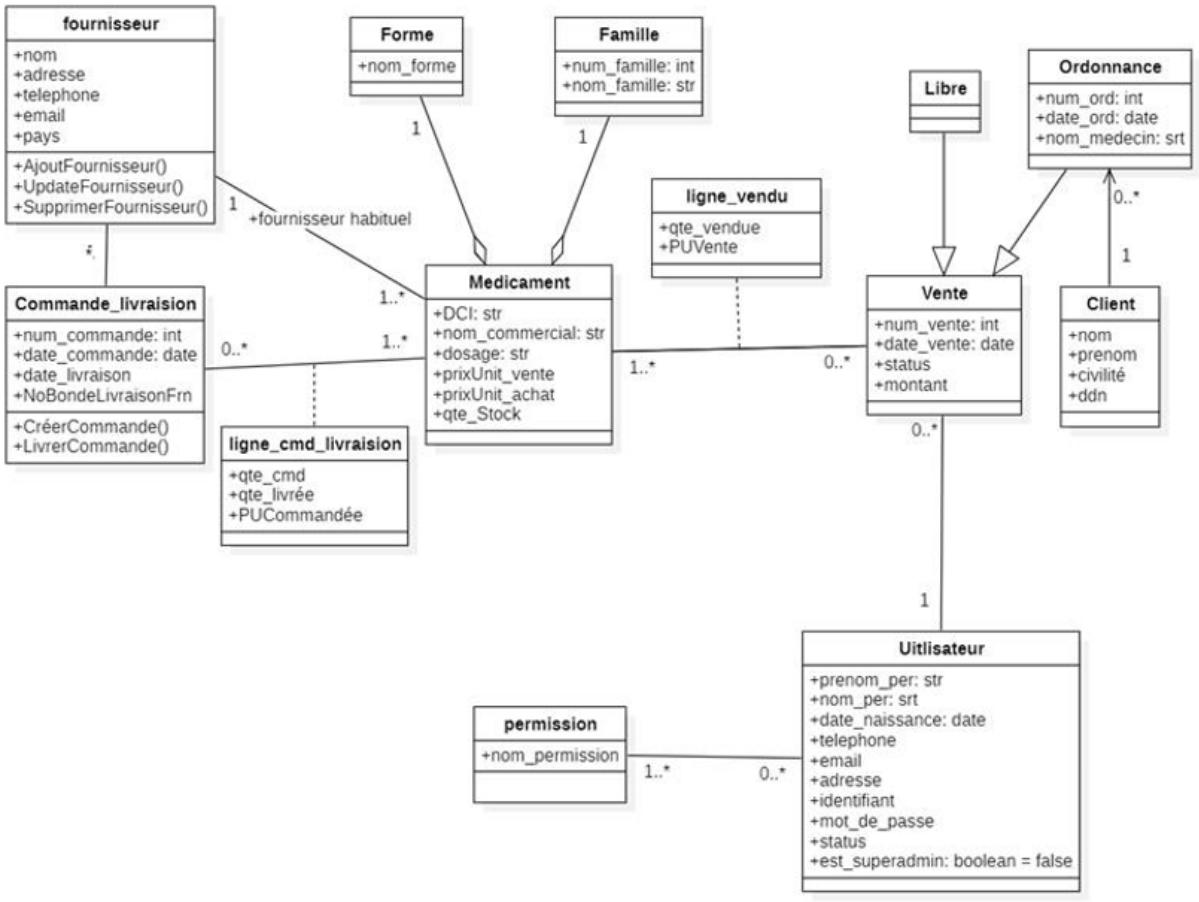

**PHARMAGEST**

| MENUS | SOUS-MENUS | TACHES | DATE DEBUT | RESPONSABLES | STATUT | DATE FIN |
|---|---|---|---|---|---|---|
| Login | | maquette | 25/03/2024 | Anne-Sophie | Terminer | 24/04/2024 |
| | | code | | Nanjanee | Terminer | |
| Dashboard | | maquette | 07/10/2024 | Anne-Sophie | Terminer | 24/10/2024 |
| | | code | | Nanjanee | Terminer | |
| Maintenance | CRUD médicaments | maquette | 11/04/2025 | Nanjanee | Terminer | 14/04/2025 |
| | | code | | | Terminer | |
| | CRUD utilisateurs | maquette | 05/04/2025 | Anne-Sophie | Terminer | 10/04/2025 |
| | | code | | | Terminer | |
| Vente | | maquette | 20/03/2025 | Anne-Sophie | Terminer | 27/03/2025 |
| | | code | | | Terminer | |
| Caisse | | marquette | 19/03/2025 | Nanjanee | Terminer | 25/04/2025 |
| | | code | | | Terminer | |

# Conception :

- **UML**

L'UML (Unified Modeling Language) est un **langage de modélisation graphique** utilisé en génie logiciel pour représenter visuellement la conception d'un système. L'objectif est de faciliter la **compréhension, la communication et le développement** d'un système informatique.




Ce diagramme UML modélise un système complet de gestion pharmaceutique avec gestion :

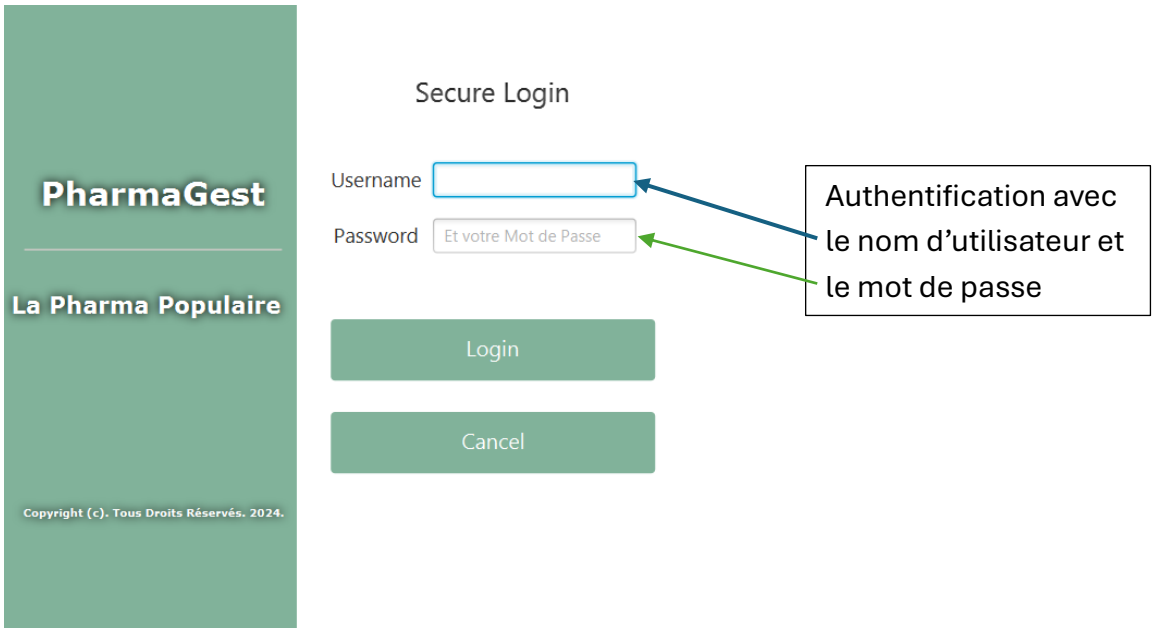
- **Des ventes,**
- **Des commandes de médicaments,**
- **Des clients et ordonnances,**
- **Des utilisateurs du système (avec rôles/permissions),**
- **Des fournisseurs et médicaments bien classés.**

- **Fonctionnelle**

  **Fonctionnalités de l'application**
  L'application développée pour **PHARMAGEST** comprend plusieurs interfaces essentielles qui permettent une gestion complète et intuitive des opérations internes.

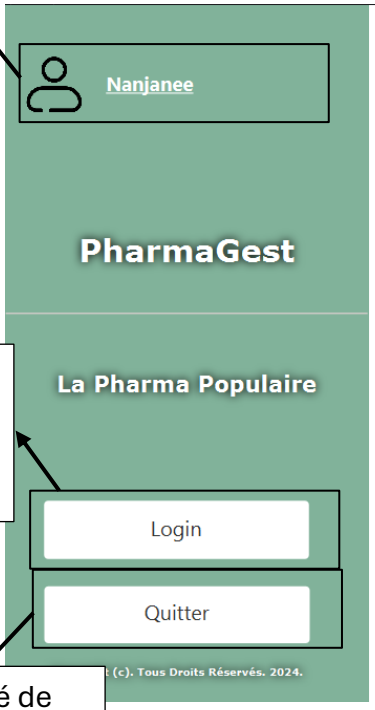
Voici un aperçu des principales fonctionnalités :

- **Page de connexion (Login)**
  Une interface d'authentification sécurisée permettant aux utilisateurs autorisés d'accéder aux différentes fonctionnalités du site.



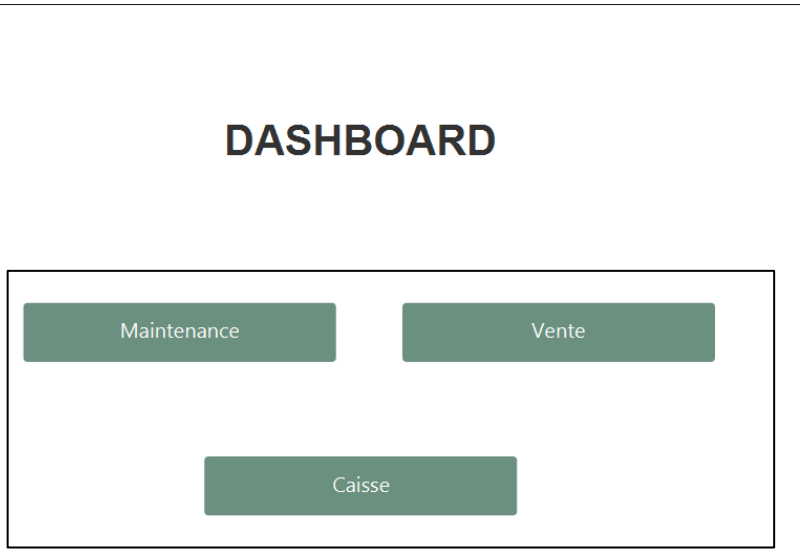

- **Tableau de bord (Dashboard)**
  Cette page centrale présente un résumé des fonctionnalités disponibles telles que la maintenance, la gestion de la caisse et les opérations de vente.

Affichage du prénom de l'utilisateur connecté




La possibilité de changer d'utilisateur

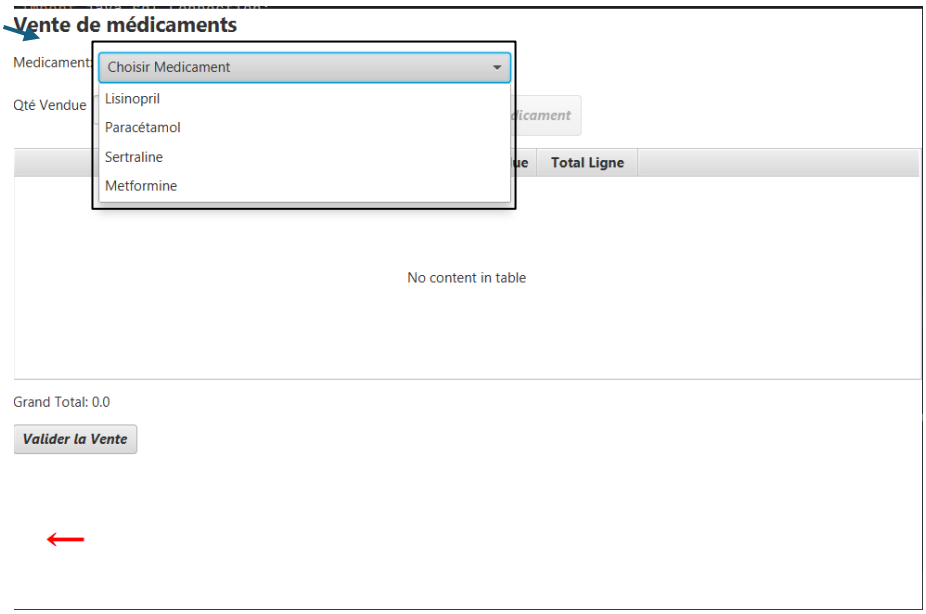
La possibilité de quitter l'application



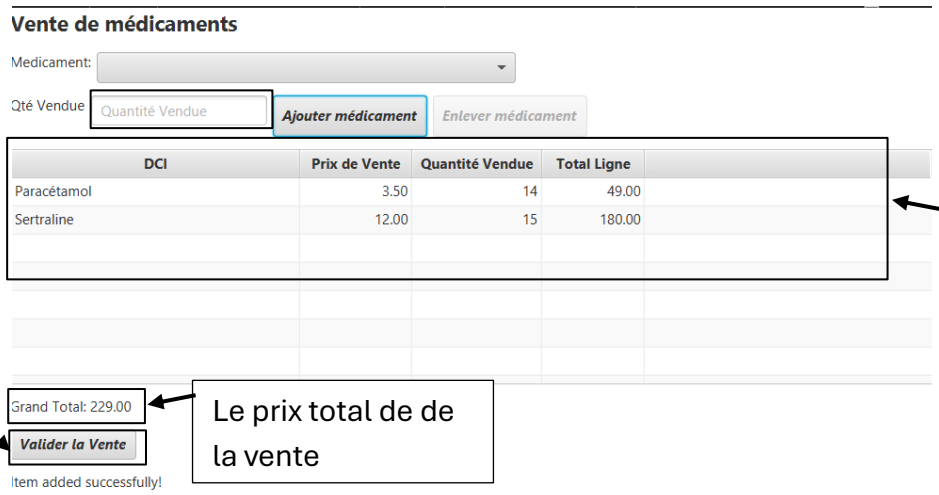

Les fonctionalitées du site.

- **Page de vente**
  Interface dédiée à la vente de médicaments. L'utilisateur peut effectuer une transaction ayant la possibilité de choisir les médicaments, de sélectionner la quantité souhaitée et percevoir le total du prix.

Les médicaments présents



Insérer la quantité voulue pour le médicament choisi.

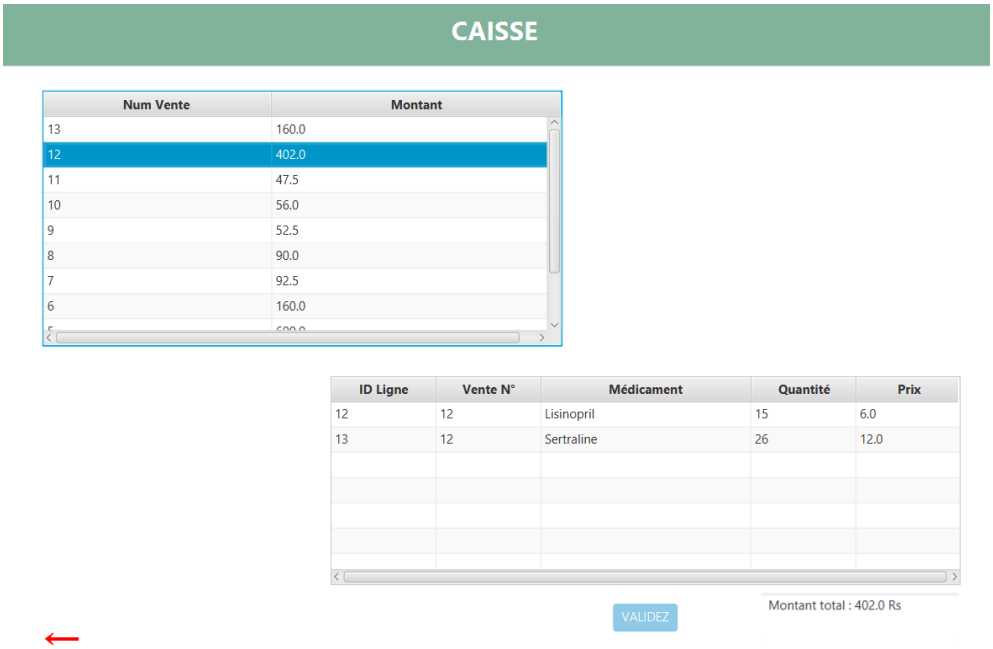



Les médicaments choisis sont affichés dans une table.

Bouton valider pour validez la vente qui sera ensuite afficher dans la caisse.

Le prix total de de la vente

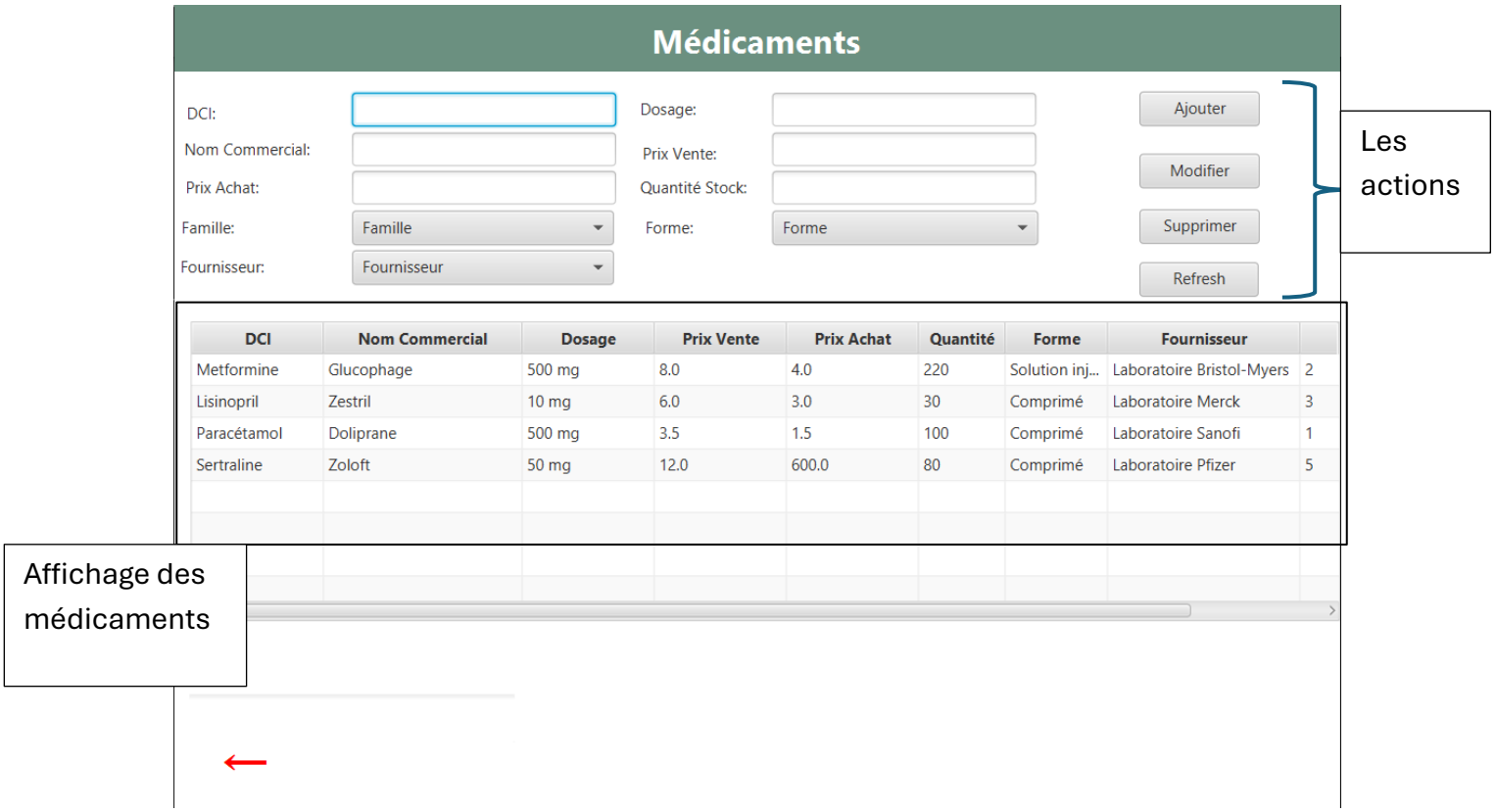
- **Page de caisse**
  Cette section affiche les détails de la vente facilitant ainsi la gestion du paiement et l'édition de reçus.



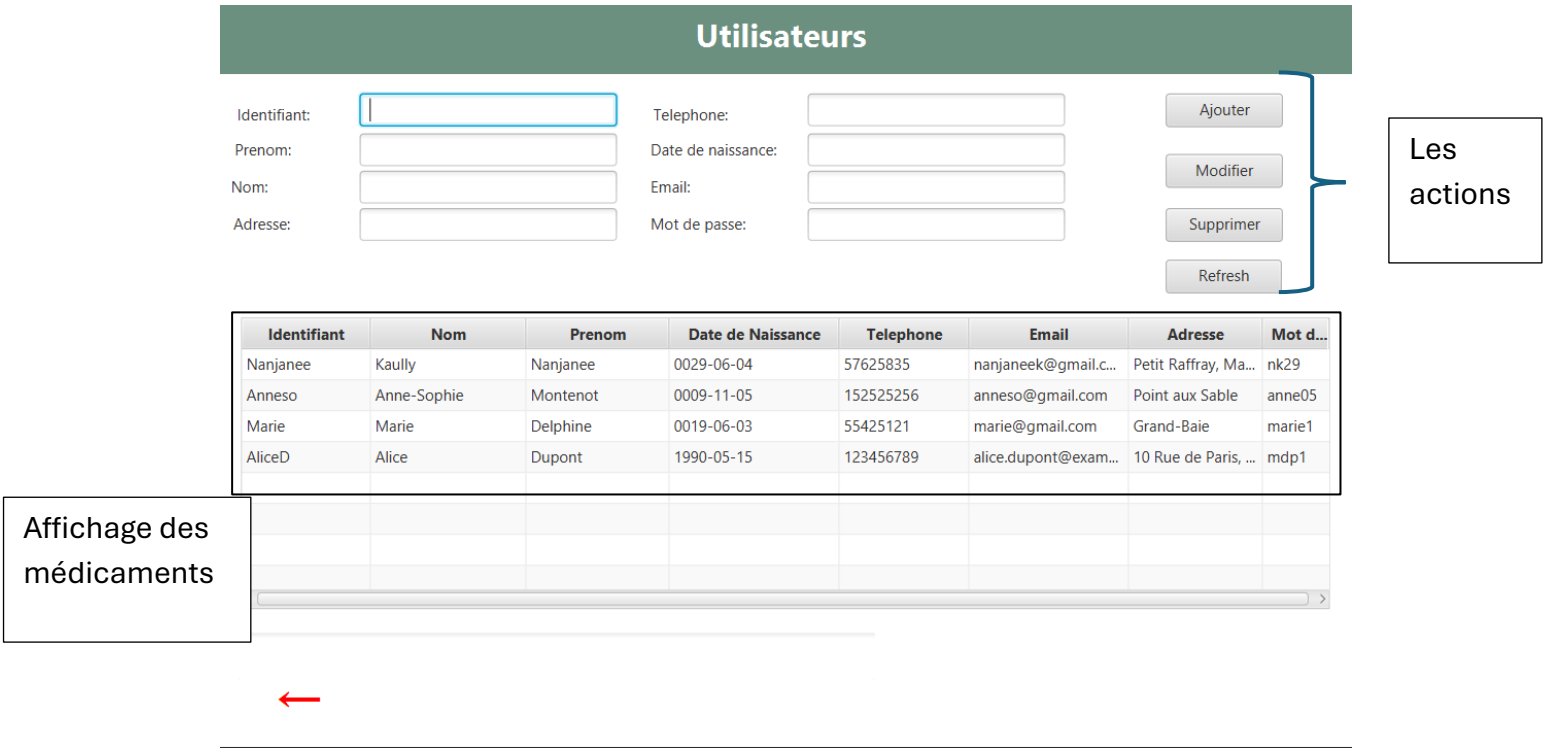

Affichage détaillé des médicaments associés à un numéro de vente spécifique.

- **Dashboard de maintenance**
  Permet la gestion des utilisateurs et des médicaments via des opérations CRUD (Créer, Lire, Mettre à jour, Supprimer). Deux sections y sont intégrées :

    - **Médicaments** : possibilité d'ajouter, de modifier ou de supprimer des médicaments du stock



- **Utilisateurs** : gestion des comptes utilisateurs avec les mêmes opérations CRUD




## Développement :

- **Outils (langue, Framework et autres…)**

Langage : Java, JavaFx

Base de données : Utilisation de PostgreSQL comme SGBD avec l'interface graphique PgAdmin.

Scene Builder : Outil graphique utilisé pour la création des interfaces

- **Programmation**

L'application couvre principalement la **gestion des ventes**, la **caisse**, ainsi que la **maintenance des utilisateurs et des médicaments**. Voici les règles de gestion retenues :

1. Un **médicament** est considéré comme un **produit** appartenant à une famille spécifique (ex. : psychotrope, analgésique, anti-inflammatoire, etc.).

2. Chaque **médicament** possède une **quantité courante en stock**, un **seuil minimal** (pour alerter d'un niveau bas), et une **quantité maximale** (pour la gestion optimale du stock).

3. Le stock est géré **en unités** (par exemple, une boîte de 10 comprimés est équivalente à 10 unités en stock).

4. Les unités de gestion peuvent être : **comprimés**, **suppositoires**, **flacons**, **crèmes**, etc.

5. La **concentration** et la **contenance** des médicaments sont incluses dans leur nom (ex. : *Panadol 500mg*, *Panadol 200mg/100ml*).

Pour pouvoir faire fonctionner le code nous avons créé un data base connexion qui permet de connecter à la base de données.

```java
package com.example.login;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;

public class DatabaseConnection {

    public Connection databaselink;  2 usages

    public Connection getConnection() {  8 usages
        String databaseName ="PharmaGest";
        String databaseUser="postgres";
        String databasePassword="091105";
        String url="jdbc:postgresql://localhost:5432/" + databaseName;


        try {
            Class.forName( className: "org.postgresql.Driver");
            databaselink=DriverManager.getConnection(url,databaseUser,databasePassword);
        } catch (Exception e) {
            e.printStackTrace();
        }
        return databaselink;
    }
}
```

**Controller :**

Un **controller** joue le rôle de **pont entre la vue (interface utilisateur) et le modèle (données)**. Il contient toute la **logique métier** et les **actions** que l'utilisateur peut effectuer via l'interface.

Exemple d'un controller pour medicament qui permet de faire le lien entre le fxml *Medicaments.fxml* et le DAO (**Data Access Object**, c'est-à-dire un **objet d'accès aux données**.)

DAO :

```java
// Supprimer un médicament de la base de données
public void deleteMedicament(Medicament medicament) {  1 usage
    String query = "DELETE FROM medicament WHERE DCI = ?";

    try (PreparedStatement stmt = connection.prepareStatement(query)) {
        stmt.setString( parameterIndex: 1, medicament.getDCI());
        stmt.executeUpdate();
    } catch (SQLException e) {
        e.printStackTrace();
    }
}
```

La requête SQL qui permet de supprimer un médicament où le champ DCI correspond.

Controller :

```java
@FXML
private void deleteMedicament() {
    Medicament selectedMed = medicamentTable.getSelectionModel().getSelectedItem();
    if (selectedMed != null) {
        medicamentDAO.deleteMedicament(selectedMed);
        consoleOutput.setText("Médicament supprimé !");
        loadMedicaments();
    } else {
        consoleOutput.setText("Sélectionnez un médicament !");
    }
}
```

Récupération du médicament sélectionné dans la table

Appelle de cette méthode DAO pour pouvoir supprimer

**Fxml :**

```
Button fx:id="addButton" layoutX="736.0" layoutY="15.0" onAction="#addMedicament" prefHeight="26.0" prefWidth="91.0" text="Ajouter" />
Button fx:id="updateButton" layoutX="736.0" layoutY="62.0" onAction="#updateMedicament" prefHeight="26.0" prefWidth="91.0" text="Modifier" />
Button fx:id="deleteButton" layoutX="736.0" layoutY="104.0" onAction="#deleteMedicament" prefHeight="26.0" prefWidth="91.0" text="Supprimer" />
Button fx:id="showAllButton" layoutX="668.0" layoutY="193.0" onAction="#showAllMedicaments" prefHeight="26.0" prefWidth="68.0" text="Afficher tout" />
```

OnAction qui fais appel deleteMedicament pour pour exécuter cette demande lorsque le bouton est actionné.

```java
@FXML
private Button addButton, updateButton, deleteButton, searchButton, showAllButton, validerButton, refreshButton;
```

@FXML est une annotation utilisée en JavaFX elle permet de faire le lien dans le fichier **FXML** avec le **Controller**. Et ainsi le buton a comme **fx:id deleteButton** pour que l'action fonctionne correctement.

## Déploiement :

La base de données a été mis ligne sur *alwaysdata* afin qu'elle soit accessible.

Et pour cela, il faut changer cependant la connexion pour que le projet soit connecté avec la base de données en ligne et non en local.

Connexion à la base de données en ligne :

```
public Connection getConnection() {  9 usages
 /*   String databaseName ="PharmaGest";
    String databaseUser="postgres";
    String databasePassword="091105";
    String url="jdbc:postgresql://localhost:5432/" + databaseName; */
    String databaseName ="supercarnanjanee_pharmagestdb";
    String databaseUser="supercarnanjanee_admin";
    String databasePassword="nanjaneeK29";
    String url="jdbc:postgresql://postgresql-supercarnanjanee.alwaysdata.net:5432/" + databaseName;

    try {
        Class.forName( className: "org.postgresql.Driver");
        databaselink=DriverManager.getConnection(url,databaseUser,databasePassword);
    } catch (Exception e) {
        e.printStackTrace();
    }
    return databaselink;
}
```

Et pour pourvoir accéder a la base de données en ligne il faut : accéder sur phppgadmin | alwaysdata avec nom d'utilisateur : supercarnanjanee_admin) et mot de passe: (nanjaneeK29).

## Conclusion :

Le projet Pharmagest a permis de moderniser efficacement la gestion interne de la pharmacie en remplaçant le système traditionnel par une application numérique.

Grâce à cet outil, la gestion des ventes et des utilisateurs est désormais centralisée et rapide.

Ce projet marque ainsi une avancée importante dans la digitalisation des services de PHARMAGEST.