Mini Projet Java

Gestion Pharmacie

Meriem Jribi « CPI2 C»

2023-2024

1 Table des matières

2 Introduction

Je vous présente mon projet de Système de Gestion de Pharmacie en Java, développé dans le cadre du module de Programmation Orientée Objet. Cette application offre une solution complète pour la gestion des clients, des ordonnances et des médicaments, en utilisant les technologies Swing, JDBC et MySQL.

3 Conception

Diagramme de cas d'utilisation :

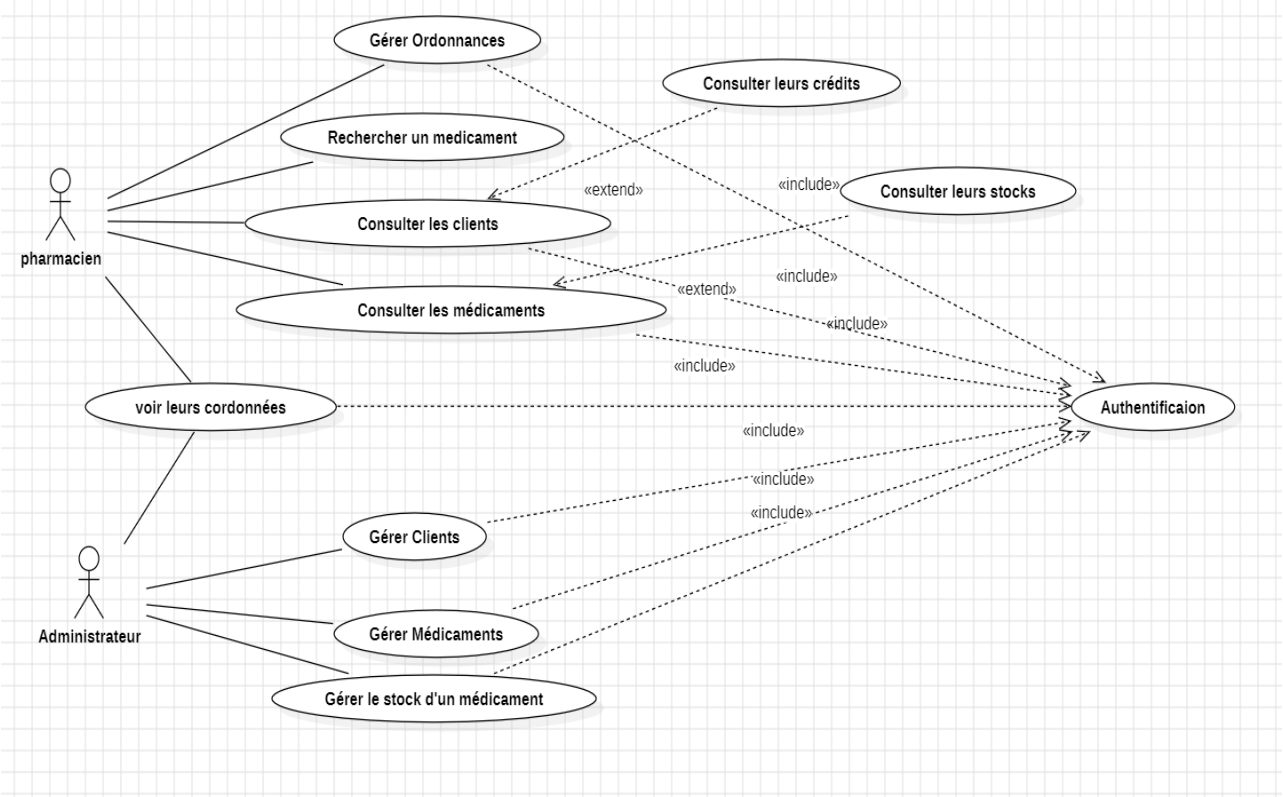
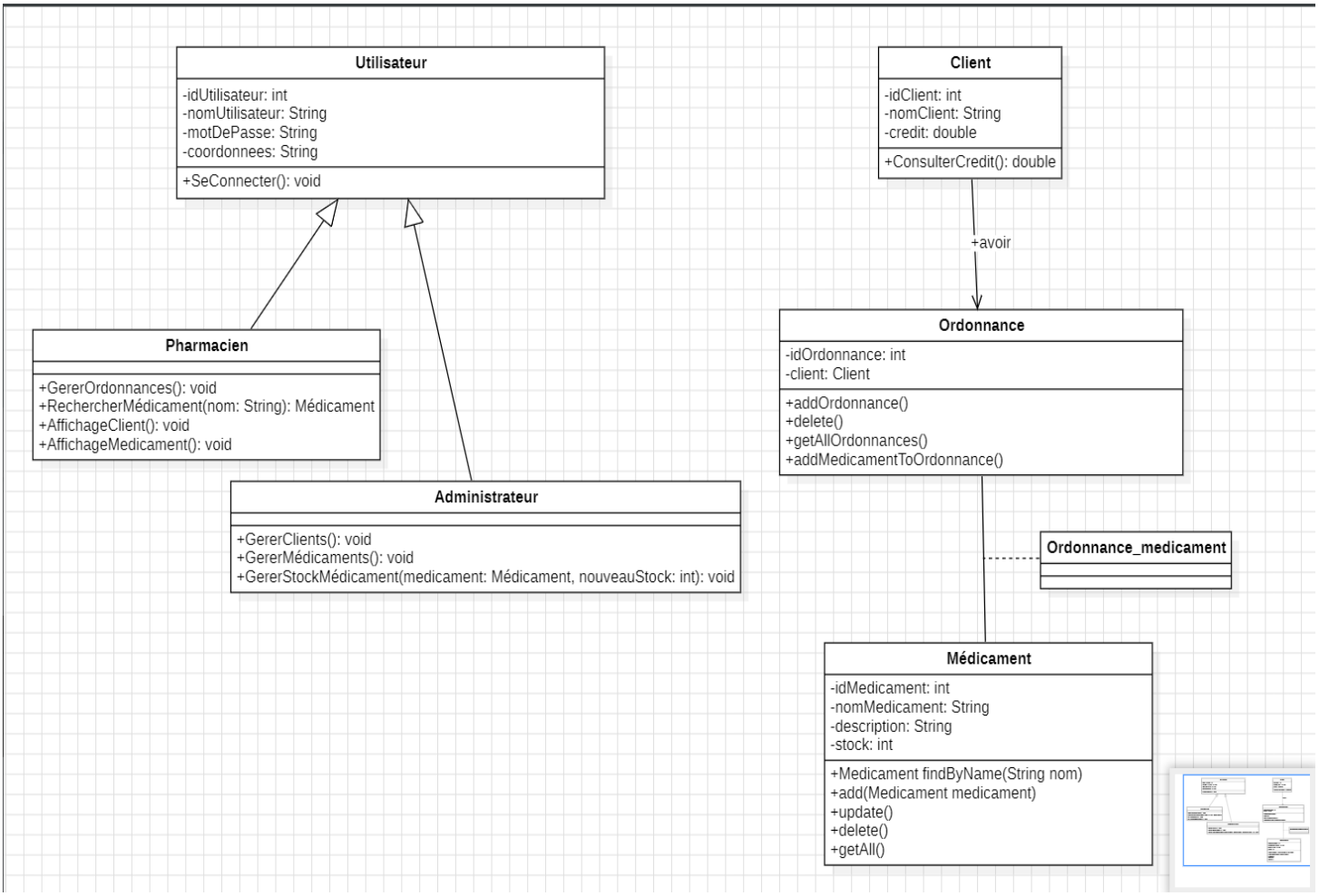


Diagramme de classe :



4 Base de données

4.1 Code SQL

```
CREATE TABLE IF NOT EXISTS `gestionpharmacie`.`utilisateur` (
  `idutilisateur` INT NOT NULL,
  `nomUtilisateur` VARCHAR(45) NULL DEFAULT NULL,
  `motdepasse` VARCHAR(45) NULL DEFAULT NULL,
  `mail` VARCHAR(45) NULL DEFAULT NULL,
  `adresse` VARCHAR(45) NULL DEFAULT NULL,
  `telephone` VARCHAR(45) NULL DEFAULT NULL,
  `role` VARCHAR(45) NULL DEFAULT NULL,
  PRIMARY KEY (`idutilisateur`))
```

```
CREATE TABLE IF NOT EXISTS `gestionpharmacie`.`administrateur` (
  `idadministrateur` INT NOT NULL,
  `AnnéEmbauche` INT NULL DEFAULT NULL,
  PRIMARY KEY (`idadministrateur`),
  CONSTRAINT `idadministrateur`
    FOREIGN KEY (`idadministrateur`)
    REFERENCES `gestionpharmacie`.`utilisateur` (`idutilisateur`))
```

```sql
CREATE TABLE IF NOT EXISTS `gestionpharmacie`.`client` (
  `idClient` INT NOT NULL,
  `nomClient` VARCHAR(45) NULL DEFAULT NULL,
  `credit` DOUBLE NULL DEFAULT NULL,
  PRIMARY KEY (`idClient`),
  UNIQUE INDEX `idClient_UNIQUE` (`idClient` ASC) VISIBLE)
```

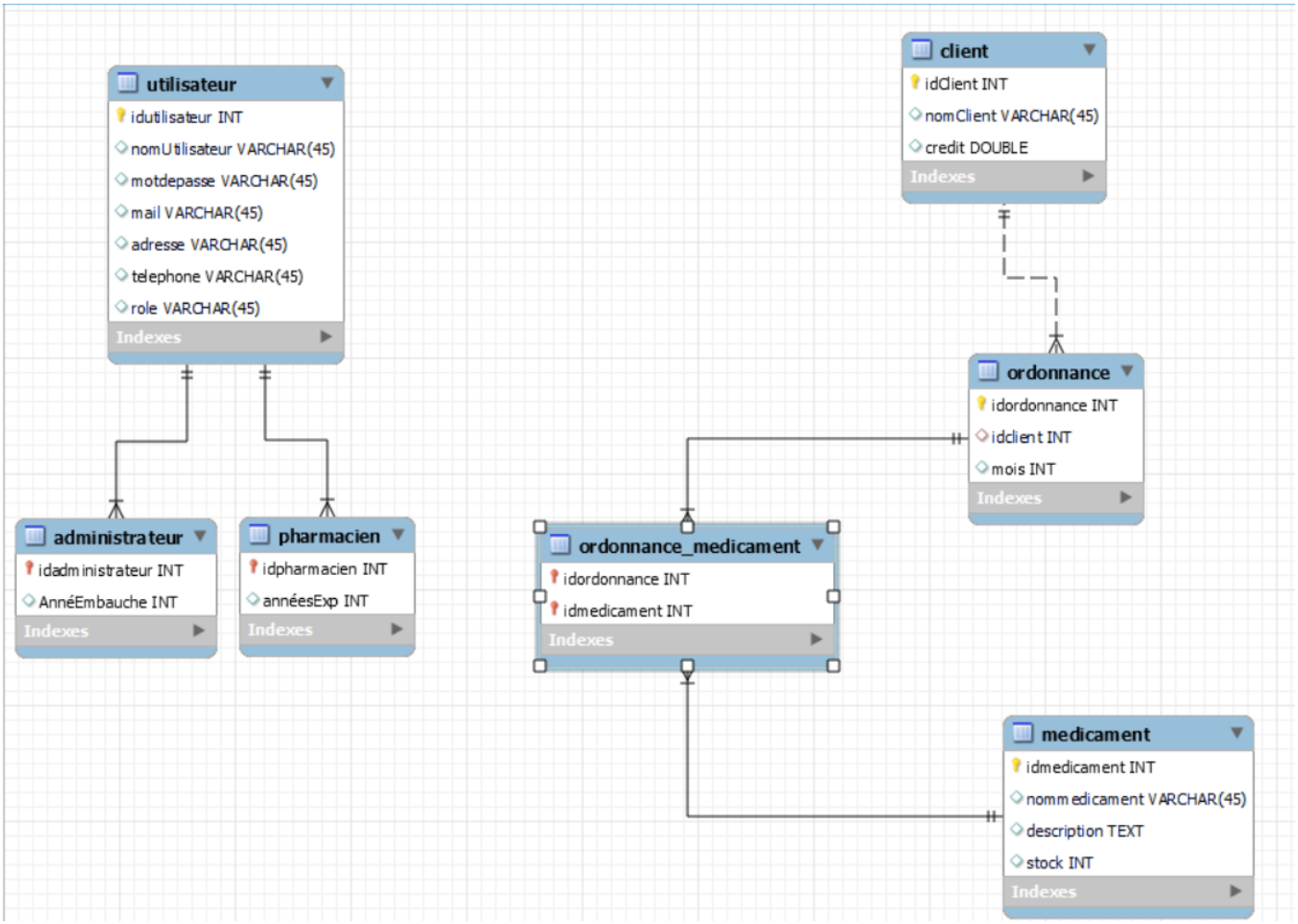
```sql
CREATE TABLE IF NOT EXISTS `gestionpharmacie`.`medicament` (
  `idmedicament` INT NOT NULL,
  `nommedicament` VARCHAR(45) NULL DEFAULT NULL,
  `description` TEXT NULL DEFAULT NULL,
  `stock` INT NULL DEFAULT NULL,
  PRIMARY KEY (`idmedicament`))
```

```sql
CREATE TABLE IF NOT EXISTS `gestionpharmacie`.`ordonnance` (
  `idordonnance` INT NOT NULL,
  `idclient` INT NULL DEFAULT NULL,
  `mois` INT NULL DEFAULT NULL,
  PRIMARY KEY (`idordonnance`),
  INDEX `idclient_idx` (`idclient` ASC) VISIBLE,
  CONSTRAINT `idclient`
    FOREIGN KEY (`idclient`)
    REFERENCES `gestionpharmacie`.`client` (`idClient`))
```

```sql
CREATE TABLE IF NOT EXISTS `gestionpharmacie`.`ordonnance_medicament` (
  `idordonnance` INT NOT NULL,
  `idmedicament` INT NOT NULL,
  PRIMARY KEY (`idordonnance`, `idmedicament`),
  INDEX `idmedicament_idx` (`idmedicament` ASC) VISIBLE,
  CONSTRAINT `idmedicament`
    FOREIGN KEY (`idmedicament`)
    REFERENCES `gestionpharmacie`.`medicament` (`idmedicament`),
  CONSTRAINT `idordonnance`
    FOREIGN KEY (`idordonnance`)
    REFERENCES `gestionpharmacie`.`ordonnance` (`idordonnance`))
```

```sql
CREATE TABLE IF NOT EXISTS `gestionpharmacie`.`pharmacien` (
  `idpharmacien` INT NOT NULL,
  `annéesExp` INT NULL DEFAULT NULL,
  PRIMARY KEY (`idpharmacien`),
  INDEX `idpharmacien_idx` (`idpharmacien` ASC) VISIBLE,
  CONSTRAINT `idpharmacien`
    FOREIGN KEY (`idpharmacien`)
    REFERENCES `gestionpharmacie`.`utilisateur` (`idutilisateur`))
```

4.2 Shéma Relationnel :



4.3 Les tables :

4.3.1 Table :Utilisateur

idutilisateur	nomUtilisateur	motdepasse	mail	adresse	telephone	role
111	Meriem	motdepasse123	meriem@gmail.com	Mahdia	11111111	administrateur
222	Ali	mdp456	Ali@email.com	Gabes	22222222	pharmacien
333	Asma	mdp	Asma@email.com	Sfax	33333333	administrateur
444	Youssef	mdp123	Youssef@email.com	Tunis	44444444	pharmacien
NULL	NULL	NULL	NULL	NULL	NULL	NULL

4.3.2 Table :Administrateur

idadministrateur	AnnéEmbauche
111	2013
333	20
NULL	NULL

4.3.3 Table :Pharmacien

idpharmacien	annéesExp
222	12
444	8
NULL	NULL

4.3.4 Table :Client

idClient	nomClient	credit
121	Ahmed	1000
122	Adem	500
123	Takwa	100.5
124	Safwen	1000
125	Aziza	650
127	Amine	1000
128	Yassine	3000.2
129	Fawzi	3000
130	Khalil	100
131	Farah	200
132	Meryem	4345
NULL	NULL	NULL

4.3.5 Table :Ordonnance

idordonnance	idclient	mois
10	123	9
20	121	3
30	125	6
40	131	10
50	128	1
60	127	7
70	130	11
90	123	8
NULL	NULL	NULL

4.3.6 Table :Medicament

	idmedicament	nommedicament	description	stock
▶	100	Paracétamol	Pour soulager la douleur et la fièvre	48
	200	Ibuprofène	Anti-inflammatoire	27
	300	Amoxicilline	Antibiotique , utilisé pour traiter diverses infecti...	40
	400	Loratadine	Antihistaminique utilisé pour soulager les sympt...	13
	500	Omeprazole	Inhibiteur de la pompe à protons	53
	600	Céfalexine	Antibiotique céphalosporine	59
	700	Aspirine	Analgesique, anti-inflammatoire	81
	800	Diazepam	Anxiolytique de la classe des benzodiazépines	21
	900	Atorvastatine	Inhibiteur de la HMG-CoA réductase	109
	1000	Metformine	Antidiabétique oral de la classe des biguanides	87
*	NULL	NULL	NULL	NULL

4.3.7 Table : Ordonnance_Medicament

	idordonnance	idmedicament
▶	20	100
	10	200
	30	400
	40	400
	30	500
	50	600
	50	700
	90	800
	60	900
	70	1000
*	NULL	NULL

5 Développement

5.1 Couche Métier

5.1.1 Classe :Utilisateur

```java
package Métiers;

public class Utilisateur {
    private int id;
    private String nomUtilisateur;
    private String motDePasse;
    private String mail;
    private String adresse;
    private String telephone;
    private String role;

    public Utilisateur(int id,String nomUtilisateur, String motDePasse, String mail, String adresse, String telephone,
            String role) {
        this.nomUtilisateur = nomUtilisateur;
        this.motDePasse = motDePasse;
        this.mail = mail;
        this.adresse = adresse;
        this.telephone = telephone;
        this.role = role;
        this.id = id;
    }


    public int getId() {
        return id;
    }
    public void setId(int id) {
        this.id = id;
    }
    public String getNomUtilisateur() {
        return nomUtilisateur;
    }
    public void setNomUtilisateur(String nomUtilisateur) {
        this.nomUtilisateur = nomUtilisateur;
    }
    public String getMotDePasse() {
        return motDePasse;
    }
```


```java
    }
    public String getMotDePasse() {
        return motDePasse;
    }
```


```java
    public void setMotDePasse(String motDePasse) {
        this.motDePasse = motDePasse;
    }
    public String getMail() {
        return mail;
    }
    public void setMail(String mail) {
        this.mail = mail;
    }
    public String getAdresse() {
        return adresse;
    }
    public void setAdresse(String adresse) {
        this.adresse = adresse;
    }
    public String getTelephone() {
        return telephone;
    }
    public void setTelephone(String telephone) {
        this.telephone = telephone;
    }
    public String getRole() {
        return role;
    }
    public void setRole(String role) {
        this.role = role;
    }
    @Override
    public String toString() {
        return "Utilisateur [id=" + id + ", nomUtilisateur=" + nomUtilisateur + ", motDePasse=" + motDePasse + ", mail="
                + mail + ", adresse=" + adresse + ", telephone=" + telephone + ", role=" + role + "]";}
```

5.1.2 Classe :Administrateur

```java
package Métiers;

public class Administrateur extends Utilisateur {
    private int AnnéEmbauche;

    public Administrateur(int id,String nomUtilisateur, String motDePasse, String mail, String adresse, String telephone,
            String role,int AnnéEmbauche) {
        super(id,nomUtilisateur,motDePasse,mail,adresse, telephone,role);
        this.AnnéEmbauche=AnnéEmbauche;
    }

    public int getAnnéEmbauche() {
        return AnnéEmbauche;
    }
    public void setAnnéEmbauche(int annéEmbauche) {
        AnnéEmbauche = annéEmbauche;
    }
}
```

5.1.3 Classe :Pharmacien

```java
package Métiers;

public class Pharmacien extends Utilisateur {
    private int annéesExp;

    public Pharmacien(int id,String nomUtilisateur, String motDePasse, String mail, String adresse, String telephone,
            String role,int annéesExp) {
        super(id,nomUtilisateur,motDePasse,mail,adresse, telephone,role);
        this.annéesExp=annéesExp;

    }
    public int getAnnéesExp() {
        return annéesExp;
    }
    public void setAnnéesExp(int annéesExp) {
        this.annéesExp = annéesExp;
    }
}
```

5.1.4 Classe :Client

```java
package Métiers;

public class Client {
    private int id;
    private String nom;
    private double credit;

    public Client(int id ,String nom, double credit) {
        this.id=id;
        this.nom = nom;
        this.credit = credit;
    }
    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public String getNom() {
        return nom;
    }

    public void setNom(String nom) {
        this.nom = nom;
    }

    public double getCredit() {
        return credit;
    }

    public void setCredit(double credit) {
        this.credit = credit;
    }
    @Override
    public String toString() {
        return "Client [id=" + id + ", nom=" + nom + ", credit=" + credit + "]";
```

5.1.5 Classe :Ordonnance

```java
package Métiers;

import java.util.ArrayList;

public class Ordonnance {
    private int id;
    private Client client;
    private int mois;
    private List<Medicament> medicaments=new ArrayList<Medicament>();

    public Ordonnance(int id,  int mois, Client c) {
        this.id = id;
        this.mois = mois;
        this.client=c;}

    public List<Medicament> getMedicaments() {
        return medicaments;
    }
    public void setMedicaments(List<Medicament> medicaments) {
        this.medicaments = medicaments;
    }
    public void setMois(int mois) {
        this.mois = mois;
    }

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }
    public int getMois() {
        return mois;
    }
    public void setDate(int mois) {
        this.mois =mois;
    }
```

```java
public void setDate(int mois) {
    this.mois =mois;
}
public Client getClient() {
    return client;
}

public void setClient(Client client) {
    this.client = client;
}

@Override
public String toString() {
    return "Ordonnance [id=" + id + ", client=" + client + ", mois=" + mois + ", medicaments=" + medicaments + "]";
}
```

5.1.6 Classe :Medicament

```java
package Métiers;

public class Medicament {
    private int id;
    private String nom;
    private String description;
    private int stock;
    public Medicament(int id,String nom, String description, int stock) {
        this.id=id;
        this.nom = nom;
        this.description = description;
        this.stock = stock;
    }
     public int getId() {
        return id;
    }
    public void setId(int id) {
        this.id = id;
    }
    public String getNom() {
        return nom;
    }
    public void setNom(String nom) {
        this.nom = nom;
    }
    public String getDescription() {
        return description;
    }
    public void setDescription(String description) {
        this.description = description;
    }
    public int getStock() {
        return stock;
    }
    public void setStock(int stock) {
        this.stock = stock;
    }
    @Override
    public String toString() {
        return "Medicament [nom=" + nom + ", description=" + description + ", stock=" + stock + ", id=" + id + "]";
    }

}
```

5.2 Couche DAO :

5.2.1 SingletonConnexion

j'ai utilisé un fichier « config.properties » ou j'ai mis mes données pour se connecter à la base

```properties
jdbc.url=jdbc:mysql://localhost/gestionpharmacie?serverTimezone=Europe/Paris
jdbc.user=root
jdbc.password=1234
```

```java
package DAO;
import java.io.FileInputStream;
import java.io.IOException;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import java.util.Properties;

public class SingletonConnection {
Properties props=new Properties();
private static String user;
private static String password;
private static String url;
//Objet Connection
private static Connection connect;
//Constructeur privé
private SingletonConnection(){
try {

    props.load(new FileInputStream("config.properties"));
    url=props.getProperty("jdbc.url");
    user=props.getProperty("jdbc.user");
    password=props.getProperty("jdbc.password");
    connect = DriverManager.getConnection(url, user, password);
     System.out.println("connecte");
}
catch (SQLException e)
{ e.printStackTrace();
}
catch(IOException e)
{
    e.printStackTrace();
}
}
//Méthode qui retourne l'instance et la créer si elle n'existe pas
public static Connection getInstance(){
if(connect == null){

new SingletonConnection();
}
return connect;
}
}
```

5.2.2 Classe :UtilisateurDAO :

```
public class UtilisateurDAO {
    // Se connecter à un compte selon le role indiquer dans la table && voir les cordonnées
    public Utilisateur seConnecter(int idUtilisateur, String motdepasse) {
        Connection conn = SingletonConnection.getInstance();
        String queryAll = "SELECT * FROM utilisateur WHERE idutilisateur = ? AND motdepasse = ?";
        String queryPharmacien = "SELECT * FROM pharmacien WHERE idpharmacien = ?";
        String queryAdministrateur = "SELECT * FROM administrateur WHERE idadministrateur = ?";

        try {
            PreparedStatement ps = conn.prepareStatement(queryAll);
            ps.setInt(1, idUtilisateur);
            ps.setString(2, motdepasse);

            ResultSet rs = ps.executeQuery();

            if (rs.next()) {
                int id = rs.getInt("idutilisateur");
                String nom = rs.getString("nomUtilisateur");
                String mdp = rs.getString("motdepasse");
                String mail = rs.getString("mail");
                String adresse = rs.getString("adresse");
                String telephone = rs.getString("telephone");
                String role = rs.getString("role");

                // si il est pharmacien
                if (role.equals("pharmacien")) {
                    PreparedStatement psPharmacien = conn.prepareStatement(queryPharmacien);
                    psPharmacien.setInt(1, idUtilisateur);
                    ResultSet rsPharmacien = psPharmacien.executeQuery();

                    if (rsPharmacien.next()) {
                        int annéesExperience = rsPharmacien.getInt("annéesExp");
                        return new Pharmacien(id, nom, mdp, mail, adresse, telephone, role, annéesExperience);
                    }
                 // si administrateur
                } else if (role.equals("administrateur")) {
                    PreparedStatement psAdministrateur = conn.prepareStatement(queryAdministrateur);
                    psAdministrateur.setInt(1, idUtilisateur);
                    ResultSet rsAdministrateur = psAdministrateur.executeQuery();

                    if (rsAdministrateur.next()) {
                        int annéeEmb = rsAdministrateur.getInt("AnnéEmbauche");

                        return new Administrateur(id, nom, mdp, mail, adresse, telephone, role, annéeEmb);
                    }
                }
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }

        return null;
    }
```

5.2.3 Classe :ClientDAO

```java
public class ClientDAO {
    //affichge des clients
    public List<Client> getAll() {
        Connection conn=SingletonConnection.getInstance();
        List<Client> clients = new ArrayList<>();
        String instrc = "select * from client";

        try (PreparedStatement ps = conn.prepareStatement(instrc);
             ResultSet rs = ps.executeQuery()) {

            while (rs.next()) {
                int id=rs.getInt("idClient");
                String nom = rs.getString("nomClient");
                double credit = rs.getDouble("credit");
                Client client = new Client(id,nom, credit);
                clients.add(client);
            }

        } catch (SQLException e) {
            e.printStackTrace();
        }

        return clients;
    }
    //get by id
    public Client getid(int id) {
        Connection conn = SingletonConnection.getInstance();
        String instrc = "SELECT * FROM client WHERE idClient=?";

        try (PreparedStatement ps = conn.prepareStatement(instrc)) {
            ps.setInt(1, id); // Passer le paramètre id à la requête SQL
            try (ResultSet rs = ps.executeQuery()) {
                if (rs.next()) {
                    String nom = rs.getString("nomClient");
                    double credit = rs.getDouble("credit");
                    Client client = new Client(id, nom, credit);
                    return client;
                }
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
```

```java
    //ajout d'un Client
    public boolean add(Client client) {
        Connection conn=SingletonConnection.getInstance();
        String instrc = "insert into client (idClient ,nomClient, credit) values (?,?, ?)";

        try (PreparedStatement ps = conn.prepareStatement(instrc)) {
            ps.setInt(1,client.getId());
            ps.setString(2, client.getNom());
            ps.setDouble(3, client.getCredit());

            ps.executeUpdate();
            return true;

        } catch (SQLException e) {
            e.printStackTrace();
            return false;
        }
    }
    //modifier un client
    public boolean update(Client client) {
        Connection conn=SingletonConnection.getInstance();
        String instrc = "update client set nomClient = ?, credit = ? where idClient = ?";

        try (PreparedStatement ps = conn.prepareStatement(instrc)) {

            ps.setString(1, client.getNom());
            ps.setDouble(2, client.getCredit());
            ps.setInt(3, client.getId());

            ps.executeUpdate();

            return true;

        } catch (SQLException e) {
            e.printStackTrace();
            return false;
        }
    }
    //Supprimer Client
    public boolean delete(Client client) {
        Connection conn=SingletonConnection.getInstance();
        String instrc = "delete from client where idClient = ?";

    //Supprimer Client
    public boolean delete(Client client) {
        Connection conn=SingletonConnection.getInstance();
        String instrc = "delete from client where idClient = ?";
        try (PreparedStatement ps = conn.prepareStatement(instrc)) {

            ps.setInt(1, client.getId());

            ps.executeUpdate();

            return true;

        } catch (SQLException e) {
            e.printStackTrace();
            return false;
        }
    }
}
```

5.2.4 Classe :OrdonnanceDAO

```java
public class OrdonnanceDAO {

    //Affichage Ordonnances
    public ResultSet getAllOrdonnances() {
        Connection conn=SingletonConnection.getInstance();

        try  {
            Statement statement = conn.createStatement();
            return statement.executeQuery("SELECT o.idordonnance,c.idClient, c.nomClient,m.idmedicament, m.nommedicament ,o.mois "
                    + "FROM ordonnance o, client c, ordonnance_medicament om, medicament m "
                    + "WHERE o.idclient = c.idClient "
                    + "AND o.idordonnance = om.idordonnance "
                    + "AND om.idmedicament = m.idmedicament"
                    + " ORDER BY o.idordonnance");}

        catch (SQLException e) {
            e.printStackTrace();
            return null;
        }}
    //Ajout Ordonnance

    public void addOrdonnance(Ordonnance ordonnance) {
        Connection conn = SingletonConnection.getInstance();
        String query = "INSERT INTO ordonnance (idordonnance, idclient, mois) VALUES (?, ?, ?)";

        try (PreparedStatement statement = conn.prepareStatement(query)) {
            statement.setInt(1, ordonnance.getId());
            statement.setInt(2, ordonnance.getClient().getId());
            statement.setInt(3, ordonnance.getMois());

            statement.executeUpdate();

            // Vérification si une ligne a été insérée avec succès

                }
         catch (SQLException e) {
            e.printStackTrace();
        }


    }


      //ajout d'un  medicament dans ordonnance
      public boolean addMedicamentToOrdonnance(int id, int medicamentId) {
          Connection conn=SingletonConnection.getInstance();
          String instrc1 = "insert into ordonnance_medicament (idordonnance, idmedicament) values (?, ?)";
          String instrc2="update medicament set stock=stock-1 where idmedicament=?";
          try {
              PreparedStatement ps = conn.prepareStatement(instrc1);
              ps.setInt(1, id);
              ps.setInt(2, medicamentId);
              ps.executeUpdate();

              PreparedStatement ps1 = conn.prepareStatement(instrc2);
              ps1.setInt(1, medicamentId);
              ps1.executeUpdate();
              return true;

          } catch (SQLException e) {
              e.printStackTrace();
              return false;
          }
      }

  // Méthode pour supprimer un médicament spécifique d'une ordonnance
      public boolean deleteMedicamentFromOrdonnance(int idOrdonnance, int idMedicament) {
          Connection conn = SingletonConnection.getInstance();
          String deleteOrdonnanceMedicamentQuery = "DELETE FROM ordonnance_medicament WHERE idordonnance = ? AND idmedicament = ?";
          String increaseStockQuery = "UPDATE medicament SET stock = stock + 1 WHERE idmedicament = ?";

          try {
              // Supprimer le médicament de l'ordonnance
              PreparedStatement ps1 = conn.prepareStatement(deleteOrdonnanceMedicamentQuery);
              ps1.setInt(1, idOrdonnance);
              ps1.setInt(2, idMedicament);
              ps1.executeUpdate();

              // Augmenter le stock du médicament
                  PreparedStatement ps2 = conn.prepareStatement(increaseStockQuery);
                  ps2.setInt(1, idMedicament);
                  ps2.executeUpdate();
                  return true; // La suppression et la mise à jour du stock ont réussi
          } catch (SQLException e) {
              e.printStackTrace();
```

```java
        //Modifier une ordonnance et les médicament qu'elle contient
        public boolean updateOrdonnance(Ordonnance ordonnance) {
            Connection conn=SingletonConnection.getInstance();
            String updateinstrc = "update ordonnance set idclient = ?, mois = ? where idordonnance = ?";
            try {
                PreparedStatement updatePS = conn.prepareStatement(updateinstrc);
                updatePS.setInt(1, ordonnance.getClient().getId());
                updatePS.setInt(2, ordonnance.getMois());
                updatePS.setInt(3, ordonnance.getId());


                updatePS.executeUpdate();
                return true;}


             catch (SQLException e) {
                e.printStackTrace();
            }
            return false;
        }

    //Supprimer une ordonnance
        public boolean delete(int ordonnanceId) {
            Connection conn=SingletonConnection.getInstance();
            String OrdonnanceMedicamentinstrc = "delete from ordonnance_medicament where  idordonnance= ?";
            String Ordonnanceinstrc = "delete from ordonnance where idordonnance= ?";


            try {
                // Suppression des liens entre l'ordonnance et les médicaments
                PreparedStatement OrdonnanceMedicamentPS = conn.prepareStatement(OrdonnanceMedicamentinstrc);
                OrdonnanceMedicamentPS.setInt(1, ordonnanceId);
                OrdonnanceMedicamentPS.executeUpdate();


                // Suppression de l'ordonnance
                PreparedStatement OrdonnancePS = conn.prepareStatement(Ordonnanceinstrc);
                OrdonnancePS.setInt(1, ordonnanceId);
                OrdonnancePS.executeUpdate();


                return true;
        }

    //Supprimer une ordonnance
        public boolean delete(int ordonnanceId) {
            Connection conn=SingletonConnection.getInstance();
            String OrdonnanceMedicamentinstrc = "delete from ordonnance_medicament where  idordonnance= ?";
            String Ordonnanceinstrc = "delete from ordonnance where idordonnance= ?";


            try {
                // Suppression des liens entre l'ordonnance et les médicaments
                PreparedStatement OrdonnanceMedicamentPS = conn.prepareStatement(OrdonnanceMedicamentinstrc);
                OrdonnanceMedicamentPS.setInt(1, ordonnanceId);
                OrdonnanceMedicamentPS.executeUpdate();


                // Suppression de l'ordonnance
                PreparedStatement OrdonnancePS = conn.prepareStatement(Ordonnanceinstrc);
                OrdonnancePS.setInt(1, ordonnanceId);
                OrdonnancePS.executeUpdate();


                return true;
                }
             catch (SQLException e) {
                e.printStackTrace();
            }
            return false;
        }
}
```

5.2.5 Classe :MedicamentDAO

```
public class MedicamentDAO  {

 // recherche d'un médicament par nom
    public Medicament findByName(String nom) {
        Connection conn=SingletonConnection.getInstance();
        String instrc = "select * from medicament where nommedicament = ?";
        Medicament medicament=null;

        try (PreparedStatement ps = conn.prepareStatement(instrc);) {
            ps.setString(1, nom);
            ResultSet rs = ps.executeQuery();

            if (rs.next()) {
                int id = rs.getInt("idmedicament");
                String description = rs.getString("description");
                int stock = rs.getInt("stock");

               medicament = new Medicament( id,nom, description, stock);
            }

        } catch (SQLException e) {
            e.printStackTrace();
        }

        return medicament;
    }
    // afficher tous les médicaments

    public List<Medicament> getAll() {
        Connection conn=SingletonConnection.getInstance();
        List<Medicament> medicaments = new ArrayList<>();
        String instrc = "select * from medicament";

        try (PreparedStatement ps = conn.prepareStatement(instrc);
             ResultSet rs = ps.executeQuery()) {

            while (rs.next()) {
                int id = rs.getInt("idmedicament");
                String nom = rs.getString("nommedicament");
```

```java
    }
    // afficher tous les médicaments

    public List<Medicament> getAll() {
        Connection conn=SingletonConnection.getInstance();
        List<Medicament> medicaments = new ArrayList<>();
        String instrc = "select * from medicament";

        try (PreparedStatement ps = conn.prepareStatement(instrc);
             ResultSet rs = ps.executeQuery()) {

            while (rs.next()) {
                int id = rs.getInt("idmedicament");
                String nom = rs.getString("nommedicament");
                String description = rs.getString("description");
                int stock = rs.getInt("stock");

                Medicament medicament = new Medicament(id,nom, description, stock);
                medicaments.add(medicament);
            }

        } catch (SQLException e) {
            e.printStackTrace();
        }

        return medicaments;
    }
    //afficher liste des nom des médicament
    public List<String> getAllMedicamentNom() {
        List<Medicament> medicaments = getAll();
        List<String> medicamentNames = new ArrayList<>();

        for (Medicament medicament : medicaments) {
            medicamentNames.add(medicament.getNom());
        }

        return medicamentNames;
    }
```

```java
    //ajout medicament
    public boolean add(Medicament medicament) {
        Connection conn=SingletonConnection.getInstance();
        String instrc = "insert into medicament (idmedicament,nommedicament, description, stock) values (?, ?, ?,?)";

        try (PreparedStatement ps = conn.prepareStatement(instrc)) {
            ps.setInt(1,medicament.getId());
            ps.setString(2, medicament.getNom());
            ps.setString(3, medicament.getDescription());
            ps.setInt(4, medicament.getStock());

            ps.executeUpdate();
            return true;

        } catch (SQLException e) {
            e.printStackTrace();
            return false;
        }
    }
    //modifier médicament
    public boolean update(Medicament medicament) {
        Connection conn=SingletonConnection.getInstance();
        String instrc = "update medicament set nommedicament = ?, description = ?, stock = ? where idmedicament = ?";

        try (PreparedStatement ps = conn.prepareStatement(instrc)) {

            ps.setString(1, medicament.getNom());
            ps.setString(2, medicament.getDescription());
            ps.setInt(3, medicament.getStock());
            ps.setInt(4, medicament.getId());

            ps.executeUpdate();

            return true;

        } catch (SQLException e) {
            e.printStackTrace();
            return false;
        }
    }



    //Supprimer medicament
    public boolean delete(int id) {
        Connection conn=SingletonConnection.getInstance();
        String instrc = "delete from medicament where idmedicament = ?";
        try (PreparedStatement ps = conn.prepareStatement(instrc)) {

            ps.setInt(1, id);

            ps.executeUpdate();

            return true;

        } catch (SQLException e) {
            e.printStackTrace();
            return false;
        }
    }

}
```

5.3 Couche view :

5.3.1 Table Model Medicament :

```
public class TableModelMed1 extends AbstractTableModel {
    private String[] nomColonnes=new String[] {"ID","NOM Med","DESCRIPTION","STOCK"};
    private Vector<String[]> rows=new Vector<String[]>();
    @Override
    public int getColumnCount() {
        // TODO Auto-generated method stub
        return nomColonnes.length;
    }

    @Override
    public int getRowCount() {
        // TODO Auto-generated method stub
        return rows.size();
    }

    @Override
    public Object getValueAt(int rowIndex, int columnIndex) {
        // TODO Auto-generated method stub
        return rows.get(rowIndex)[columnIndex];
    }
    public String getColumnName(int column)
    {
        return nomColonnes[column];
    }
    public void loadData(List<Medicament> medicaments)
    {
        rows=new Vector<String[]>();
        for(Medicament m : medicaments)
        {
            rows.add(new String[] {
                    String.valueOf(m.getId()),
                    m.getNom(),
                    m.getDescription(),
                    String.valueOf(m.getStock())});
        }
        fireTableChanged(null);
    }
 // Méthode pour effacer les données actuellement affichées dans le tableau
    public void clearData() {
    }
 // Méthode pour effacer les données actuellement affichées dans le tableau
    public void clearData() {
        rows.clear();
        fireTableDataChanged();
    }

    // Méthode pour ajouter une seule ligne de données à afficher dans le tableau
    public void addData(Medicament medicament) {
        rows.add(new String[] {
                String.valueOf(medicament.getId()),
                medicament.getNom(),
                medicament.getDescription(),
                String.valueOf(medicament.getStock())});
        fireTableRowsInserted(rows.size() - 1, rows.size() - 1);
    }
}
```

5.3.2 Table Model Ordonnance :

```java
public class TableModelOrdonnance extends AbstractTableModel {


private String[] nomColonnes = new String[] {"idordonnance","idCilent", "nomClient","idmedicament", "nommedicament","mois"};
private Vector<String[]> rows = new Vector<String[]>();

@Override
public int getColumnCount() {
    return nomColonnes.length;
}

@Override
public int getRowCount() {
    return rows.size();
}

@Override
public Object getValueAt(int rowIndex, int columnIndex) {
    return rows.get(rowIndex)[columnIndex];
}

@Override
public String getColumnName(int column) {
    return nomColonnes[column];
}
```

```java
public void loadData(ResultSet rs) {
    rows.clear(); // Effacer les données existantes
    try {
        while (rs.next()) {
            rows.add(new String[] {
                String.valueOf(rs.getInt("idordonnance")),
                String.valueOf(rs.getInt("idClient")),
                rs.getString("nomClient"),
                String.valueOf(rs.getInt("idmedicament")),
                rs.getString("nommedicament"),
                rs.getString("mois")
            });
        }
    } catch (SQLException e) {
        e.printStackTrace();
    }
    fireTableChanged(null);

}}
```

5.3.3 Table Model Client

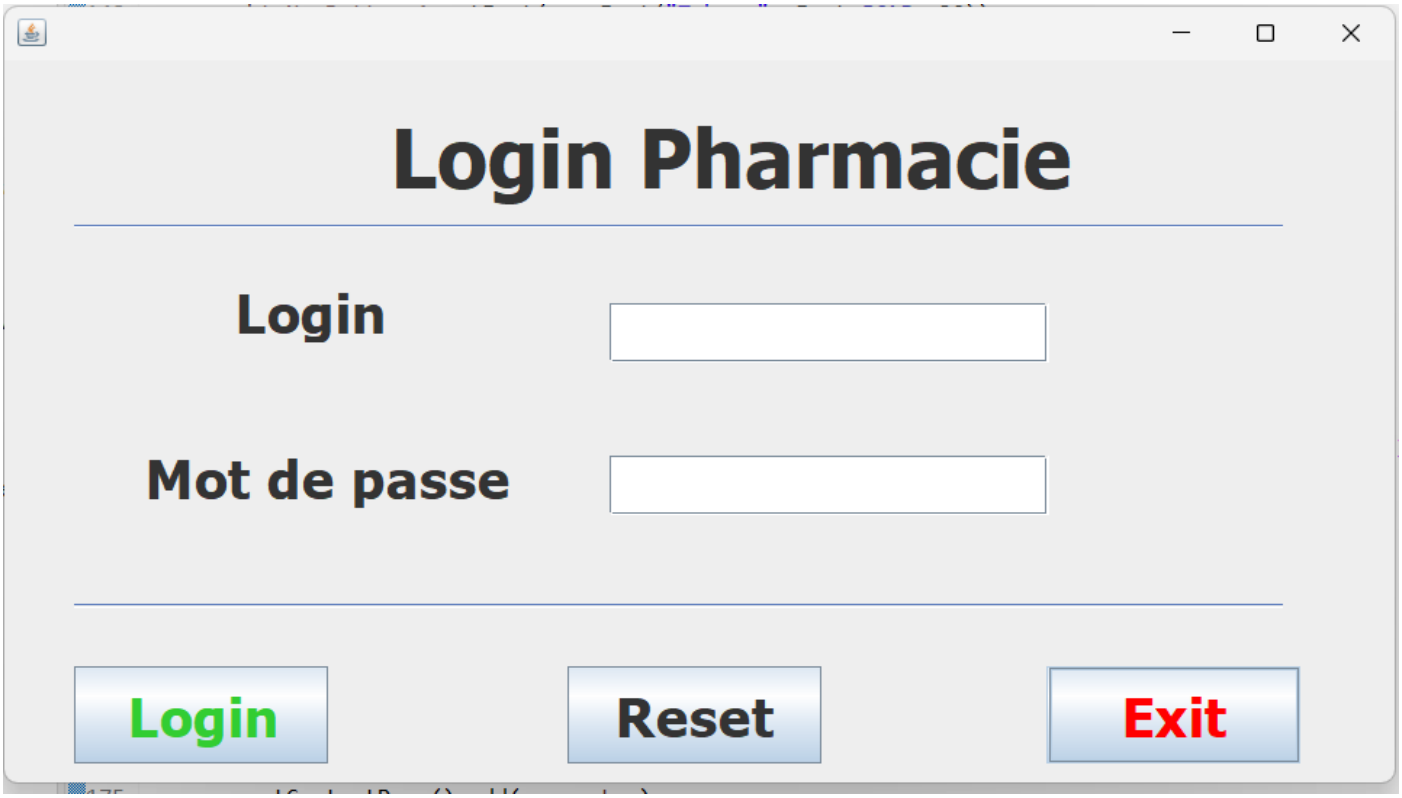
```java
public class tablemodelclient extends AbstractTableModel {
    private String[] nomColonnes=new String[] {"ID","NOM","CREDIT"};
    private Vector<String[]> rows=new Vector<String[]>();
    @Override
    public int getColumnCount() {
        // TODO Auto-generated method stub
        return nomColonnes.length;
    }

    @Override
    public int getRowCount() {
        // TODO Auto-generated method stub
        return rows.size();
    }

    @Override
    public Object getValueAt(int rowIndex, int columnIndex) {
        // TODO Auto-generated method stub
        return rows.get(rowIndex)[columnIndex];
    }
    public String getColumnName(int column)
    {
        return nomColonnes[column];
    }
    public void loadData(List<Client> clients)
    {
        rows=new Vector<String[]>();
        for(Client c : clients)
        {
            rows.add(new String[] {
                    String.valueOf(c.getId()),
                    c.getNom(),
                    String.valueOf(c.getCredit())});
        }
        fireTableChanged(null);
    }

}
```

5.3.4 Login Interface

```java
public class LoginInterface extends JFrame {

    /**
     *
     */

    private JTextField txt_login;
    private JPasswordField txtpasswrd;
    private JFrame fermer_Login;
    static private String password;
    public static String getPassword() {
        return password;
    }

    public static int getIntLog() {
        return intLog;
    }


    static private int intLog;

    /**
     * Launch the application.
     */
    public static void main(String[] args) {
        EventQueue.invokeLater(new Runnable() {
            public void run() {
                try {
                    LoginInterface window = new LoginInterface();
                    window.setVisible(true);
                } catch (Exception e) {
                    e.printStackTrace();
                }
            }
        });
    }
```

```java
/**
 * Create the application.
 */
public LoginInterface() {
    initialize();
}

/**
 * Initialize the contents of the frame.
 */
public int getIdentifiant() {
    return intLog;
}

public String getMotDePasse() {
    return password;
}
private void initialize() {
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    setBounds(200, 200, 781, 435);
    getContentPane().setLayout(null);

    JLabel lblNewLabel = new JLabel("Login Pharmacie");
    lblNewLabel.setIcon(null);
    lblNewLabel.setFont(new Font("Tahoma", Font.BOLD, 45));
    lblNewLabel.setHorizontalAlignment(SwingConstants.CENTER);
    lblNewLabel.setBounds(166, 25, 469, 56);
    getContentPane().add(lblNewLabel);

    JLabel login_lbl = new JLabel("Login");
    login_lbl.setFont(new Font("Tahoma", Font.BOLD, 30));
    login_lbl.setHorizontalAlignment(SwingConstants.CENTER);
    login_lbl.setBounds(104, 123, 128, 33);
    getContentPane().add(login_lbl);

    JLabel passwrd_lbl = new JLabel("Mot de passe");
    passwrd_lbl.setFont(new Font("Tahoma", Font.BOLD, 30));
    passwrd_lbl.setHorizontalAlignment(SwingConstants.CENTER);
    passwrd_lbl.setBounds(50, 214, 259, 33);
    getContentPane().add(passwrd_lbl);
```

```java
txt_login = new JTextField();
txt_login.setFont(new Font("Tahoma", Font.BOLD, 20));
txt_login.setBounds(334, 134, 242, 33);
getContentPane().add(txt_login);
txt_login.setColumns(10);

txtpasswrd = new JPasswordField();
txtpasswrd.setFont(new Font("Tahoma", Font.BOLD, 20));
txtpasswrd.setBounds(334, 218, 242, 33);
getContentPane().add(txtpasswrd);

JButton btnNewButton = new JButton("Login");
btnNewButton.setFont(new Font("Tahoma", Font.BOLD, 30));
btnNewButton.setForeground(new Color(50, 205, 50));
btnNewButton.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        password=txtpasswrd.getText();
        String login=txt_login.getText();
        intLog = Integer.parseInt(login);

        UtilisateurDAO u=new UtilisateurDAO();
        if(u.seConnecter(intLog, password) instanceof Pharmacien) {
            // on ouvre l'interface Pharmacien
            setVisible(false);
            new PharmacienInterface().setVisible(true);

        }
        else if(u.seConnecter(intLog, password) instanceof Administrateur) {
            // on ouvre l'interface Administrateur
            setVisible(false);
            new AdminInterface().setVisible(true);

        }
        else {
            JOptionPane.showMessageDialog(null, "Invalid Login Details","Login Error",JOptionPane.ERROR_MESSAGE);
            txtpasswrd.setText(null);
            txt_login.setText(null);
        }
    }
});

        }
    });
    btnNewButton.setBounds(39, 334, 140, 54);
    getContentPane().add(btnNewButton);

    JButton btnNewButton_1 = new JButton("Reset");
    btnNewButton_1.setFont(new Font("Tahoma", Font.BOLD, 30));
    btnNewButton_1.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            txt_login.setText(null);
            txtpasswrd.setText(null);
        }
    });
    btnNewButton_1.setBounds(311, 334, 140, 54);
    getContentPane().add(btnNewButton_1);

    JButton btnNewButton_2 = new JButton("Exit");
    btnNewButton_2.setFont(new Font("Tahoma", Font.BOLD, 30));
    btnNewButton_2.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            fermer_Login = new JFrame("Exit");
            if(JOptionPane.showConfirmDialog(fermer_Login, "Confirmez si vous voulez fermer la fenetre","Login",JOptionPane.YES_NO_OPTION)
                    ==0) {
                System.exit(0);
            }
        }
    });
    btnNewButton_2.setForeground(new Color(255, 0, 0));
    btnNewButton_2.setBounds(575, 334, 140, 54);
    getContentPane().add(btnNewButton_2);

    JSeparator separator = new JSeparator();
    separator.setBounds(39, 300, 666, 2);
    getContentPane().add(separator);

    JSeparator separator_1 = new JSeparator();
    separator_1.setBounds(39, 91, 666, 2);
    getContentPane().add(separator_1);
}
}
```


Login Pharmacie
Login
Mot de passe
Login
Reset
Exit

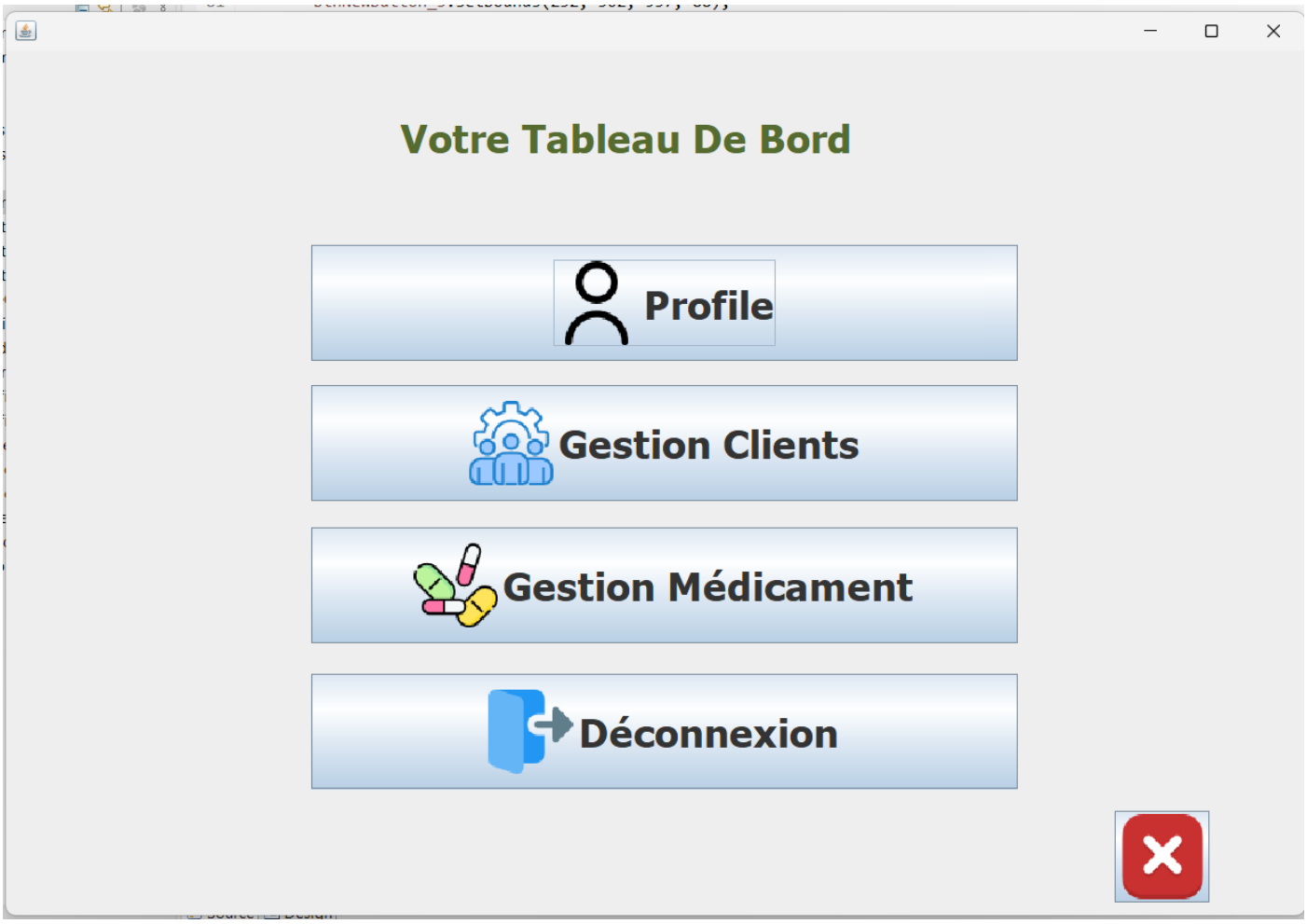
5.3.5 Administrateur Interface

```java
public class AdminInterface extends JFrame {

    private static final long serialVersionUID = 1L;
    private JPanel contentPane;

    /**
     * Launch the application.
     */
    public static void main(String[] args) {
        EventQueue.invokeLater(new Runnable() {
            public void run() {
                try {
                    AdminInterface frame = new AdminInterface();
                    frame.setVisible(true);
                } catch (Exception e) {
                    e.printStackTrace();
                }
            }
        });
    }

    /**
     * Create the frame.
     */
    public AdminInterface() {
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setBounds(100, 100, 1002, 694);
        contentPane = new JPanel();
        contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));

        setContentPane(contentPane);

        JButton btnNewButton = new JButton("Profile");
        btnNewButton.setBounds(232, 148, 537, 88);
        btnNewButton.setIcon(new ImageIcon(AdminInterface.class.getResource("/images/user (4).png")));
        btnNewButton.setFont(new Font("Tahoma", Font.BOLD, 30));
        btnNewButton.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                setVisible(false);
                new Profile2().setVisible(true);
            }
```


```java
            new Profile2().setVisible(true);
        }
    });
```

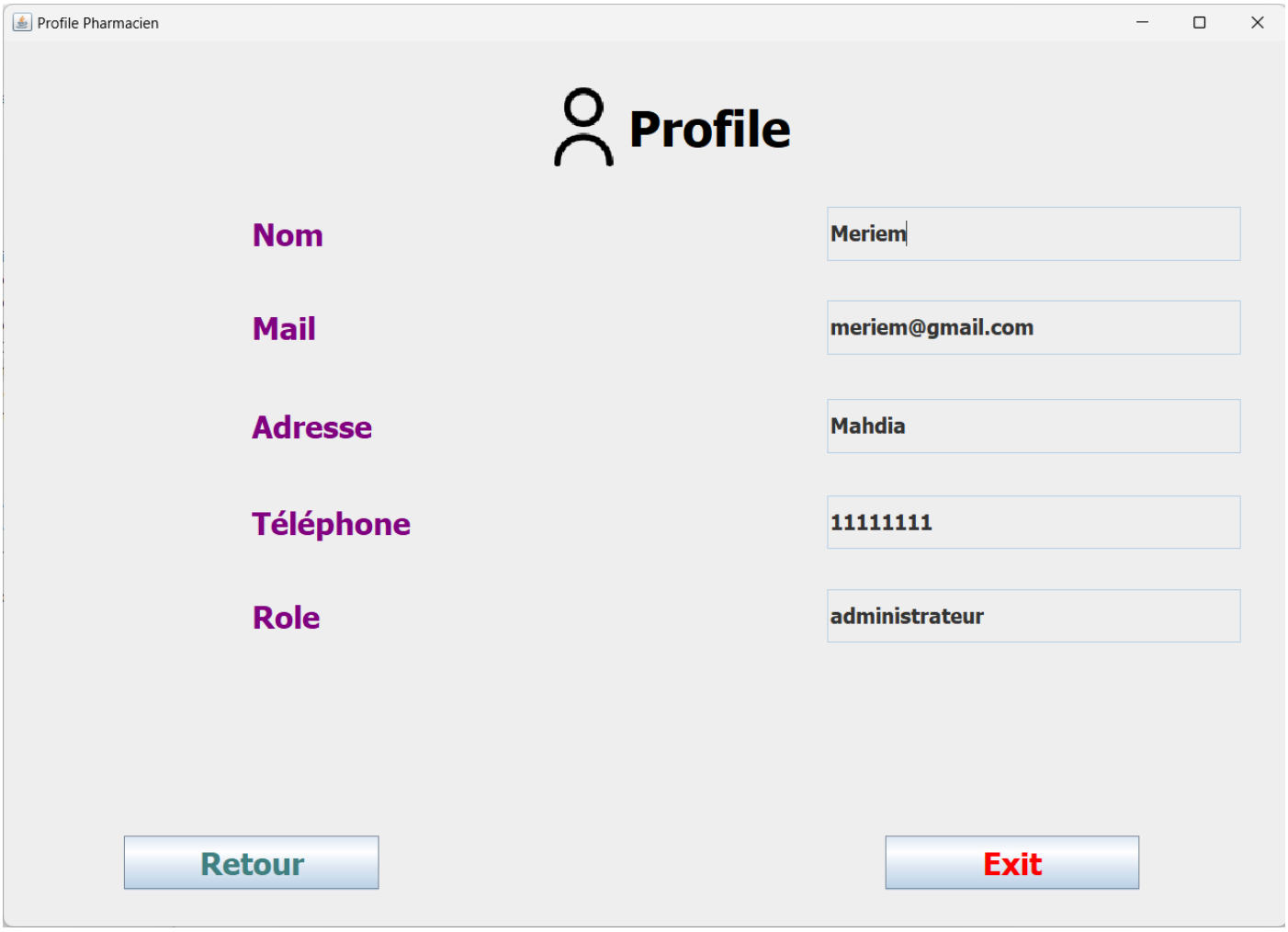

```java
    contentPane.setLayout(null);
    contentPane.add(btnNewButton);

    JButton btnNewButton_2 = new JButton("Gestion Clients");
    btnNewButton_2.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            setVisible(false);
            new GestionClient().setVisible(true);
        }
    });
    btnNewButton_2.setBounds(232, 254, 537, 88);
    btnNewButton_2.setFont(new Font("Tahoma", Font.BOLD, 30));
    btnNewButton_2.setIcon(new ImageIcon(AdminInterface.class.getResource("/images/management.png")));
    contentPane.add(btnNewButton_2);

    JButton btnNewButton_3 = new JButton("Gestion Médicament");
    btnNewButton_3.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            setVisible(false);
            new GestionMed().setVisible(true);
        }
    });
    btnNewButton_3.setBounds(232, 362, 537, 88);
    btnNewButton_3.setFont(new Font("Tahoma", Font.BOLD, 30));
    btnNewButton_3.setIcon(new ImageIcon(AdminInterface.class.getResource("/images/pills (1).png")));
    contentPane.add(btnNewButton_3);
```

```java
JButton btnNewButton_5 = new JButton("Déconnexion");
btnNewButton_5.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        if(JOptionPane.showConfirmDialog(null, "Voulez vous vraiment Déconnecter ?","Déconnecter",JOptionPane.YES_NO_OPTION)
                ==0) {
        setVisible(false);
        new LoginInterface().setVisible(true);
    }
    }
});
btnNewButton_5.setBounds(232, 473, 537, 88);
btnNewButton_5.setIcon(new ImageIcon(AdminInterface.class.getResource("/images/logout.png")));
btnNewButton_5.setFont(new Font("Tahoma", Font.BOLD, 30));
contentPane.add(btnNewButton_5);

JLabel lblNewLabel = new JLabel("Votre Tableau De Bord");
lblNewLabel.setHorizontalAlignment(SwingConstants.CENTER);
lblNewLabel.setForeground(new Color(85, 107, 47));
lblNewLabel.setFont(new Font("Tahoma", Font.BOLD, 30));
lblNewLabel.setBounds(274, 31, 394, 71);
contentPane.add(lblNewLabel);

JButton btnNewButton_1 = new JButton("");
btnNewButton_1.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        if(JOptionPane.showConfirmDialog(null, "Confirmez si vous voulez fermer la fenetre","fermer",JOptionPane.YES_NO_OPTION)
                ==0) {
            System.exit(0);
        }
    }
});
btnNewButton_1.setIcon(new ImageIcon(AdminInterface.class.getResource("/images/button.png")));
btnNewButton_1.setBounds(843, 577, 72, 70);
contentPane.add(btnNewButton_1);
```

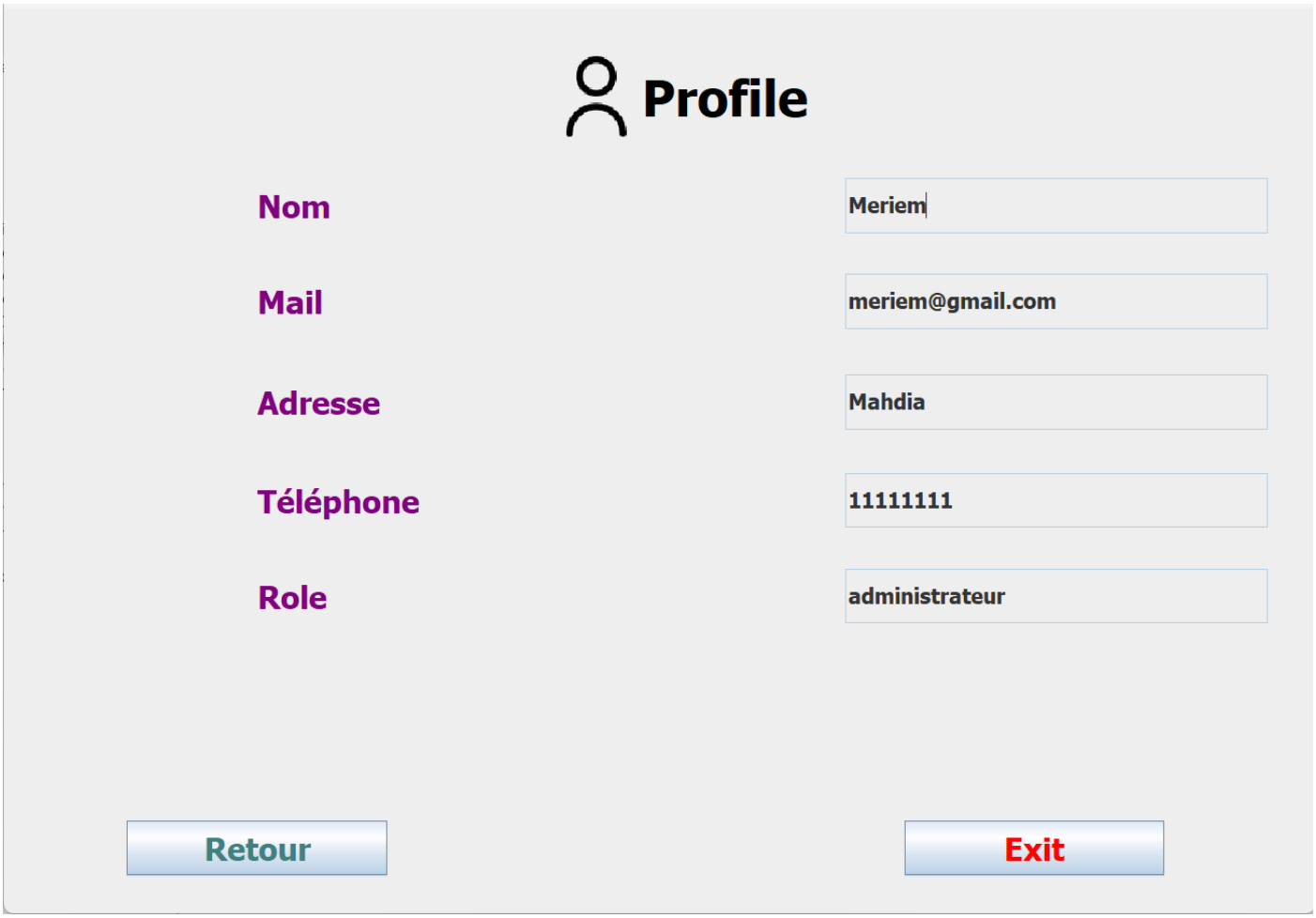


5.3.6 Pharmacien Interface

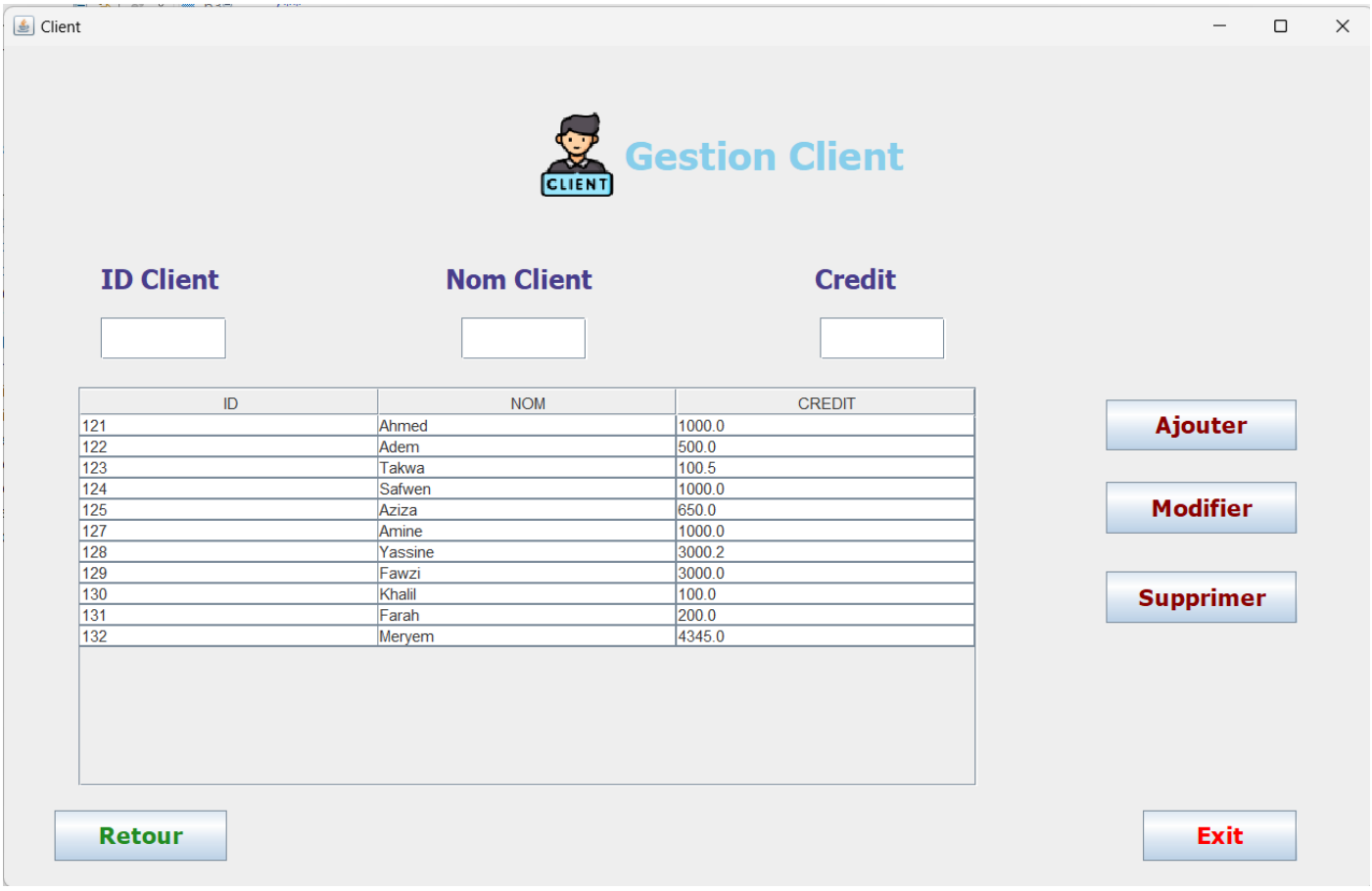
5.3.7 Profile Pharmacien Interface



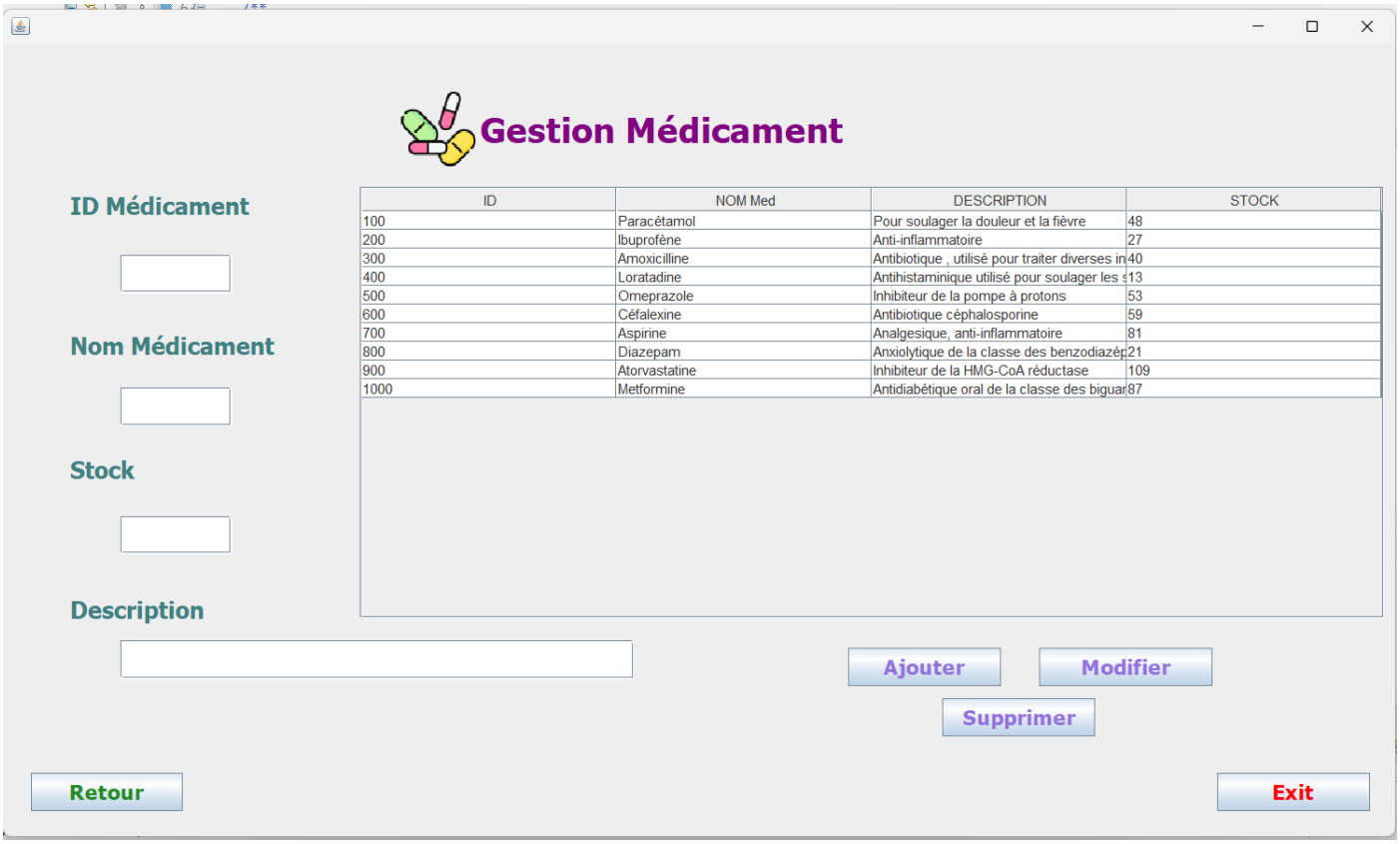
5.3.8 Profile Administrateur Interface



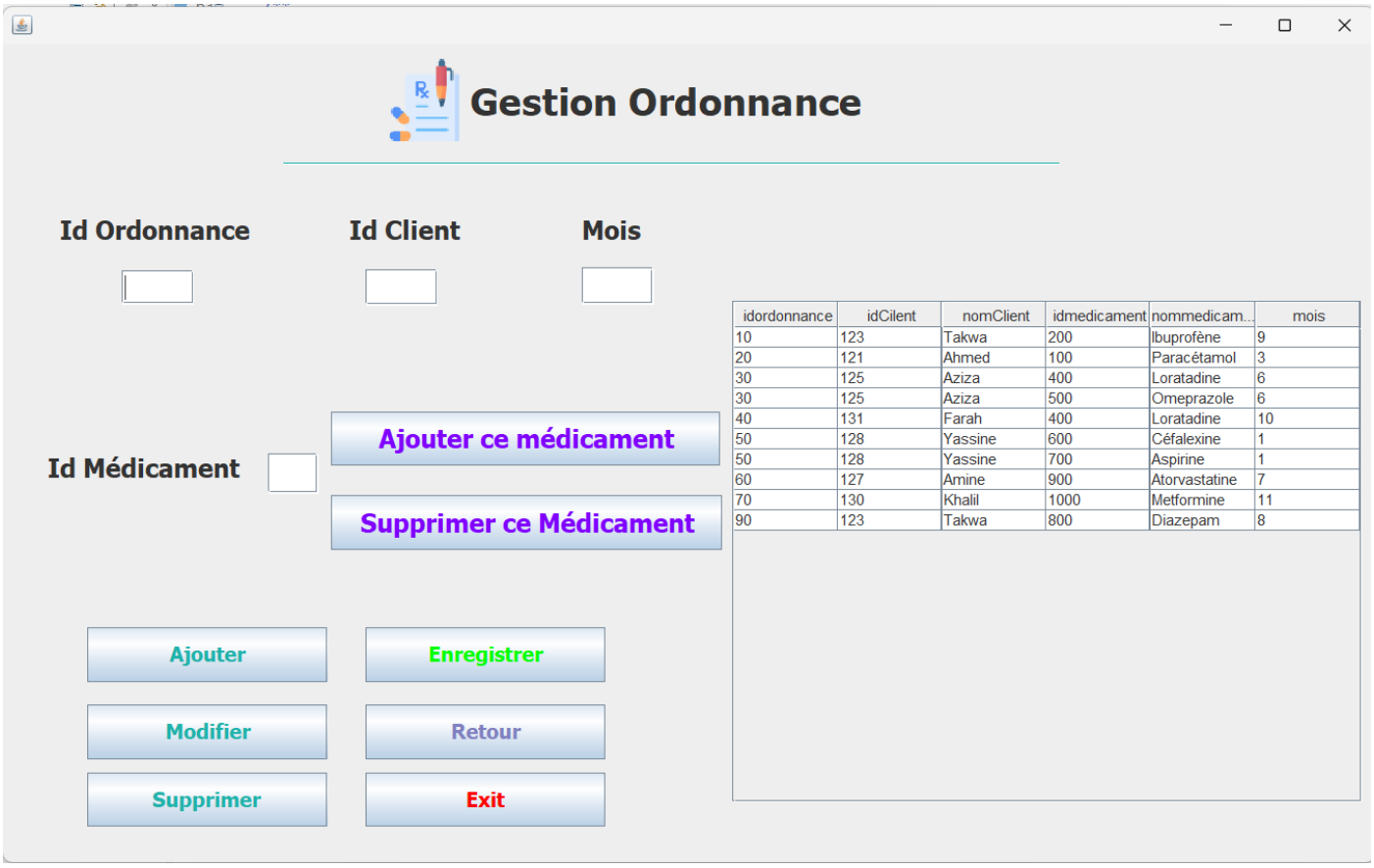
5.3.9 Gestion Client Interface



5.3.10 Gestion Medicament Interface



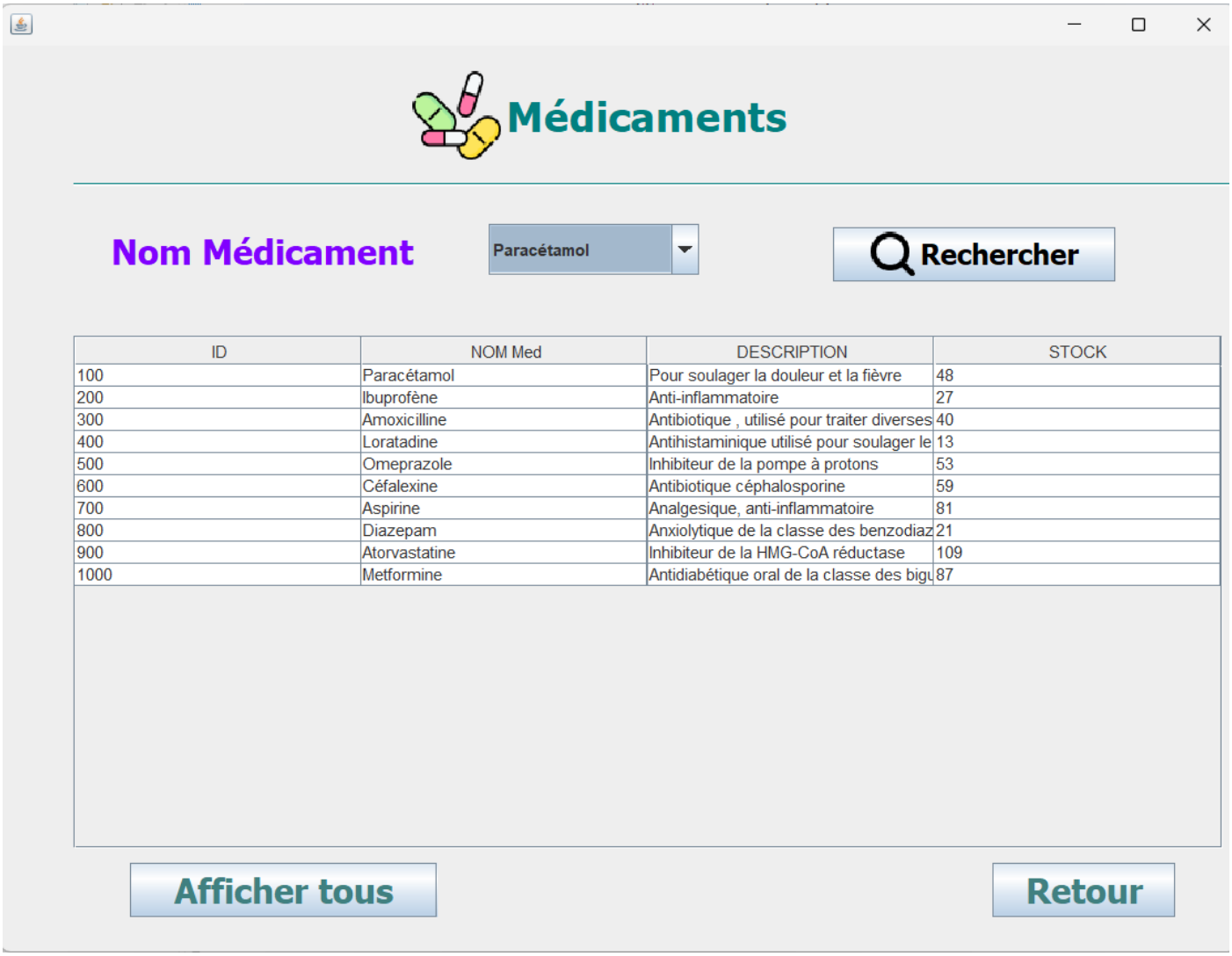
5.3.11 Gestion Ordonnance Interface



5.3.12 Liste des clients Interface

ID	NOM	CREDIT
121	Ahmed	1000.0
122	Adem	500.0
123	Takwa	100.5
124	Safwen	1000.0
125	Aziza	650.0
127	Amine	1000.0
128	Yassine	3000.2
129	Fawzi	3000.0
130	Khalil	100.0
131	Farah	200.0
132	Meryem	4345.0

5.3.13 Liste des Médicaments Interface



ID	NOM Med	DESCRIPTION	STOCK
100	Paracétamol	Pour soulager la douleur et la fièvre	48
200	Ibuprofène	Anti-inflammatoire	27
300	Amoxicilline	Antibiotique , utilisé pour traiter diverses	40
400	Loratadine	Antihistaminique utilisé pour soulager le	13
500	Omeprazole	Inhibiteur de la pompe à protons	53
600	Céfalexine	Antibiotique céphalosporine	59
700	Aspirine	Analgesique, anti-inflammatoire	81
800	Diazepam	Anxiolytique de la classe des benzodiaz	21
900	Atorvastatine	Inhibiteur de la HMG-CoA réductase	109
1000	Metformine	Antidiabétique oral de la classe des bigu	87