Ingénierie des Systèmes

Informatiques et Logiciels

Mini Projet :

Gestion d'un Cabinet Médicale

Realisé par :

Khadija BOUHAMED

Année Universitaire 2019– 2020

Chapitre 1: Contexte générale du projet

1. Analyse des besoins

Nous allons identifier dans cette partie les acteurs et leurs rôles. Par la suite, nous répondrons aux questions suivantes « que doit faire le système ? » et « Quelles sont les contraintes ?» afin d'expliquer les différents besoins fonctionnels et non fonctionnels que notre application cherche à satisfaire.

➢ Besoins fonctionnels :

• s'authentifier : saisir le login et le mot de passe,

• gérer les clients : ajouter, modifier, supprimer et consulter,

• gérer les médecins : ajouter, modifier, supprimer,

• Gérer les créneaux : ajouter, modifier, supprimer,

• Gérer les rendez-vous : ajouter, modifier, supprimer,

➢ Besoins non fonctionnels :

• **Besoins de sécurité :** il faut que notre serveur garantisse la sécurité de notre application car son contenu ne doit être accessible qu'aux membres autorisés.

• **besoins de convivialité :** le système de centralisation doit être facile à comprendre Et à utiliser. En effet, l'interface de monitoring devra être conviviale, interactive, Cohérente du point de vue ergonomie et bien adaptée à l'utilisation

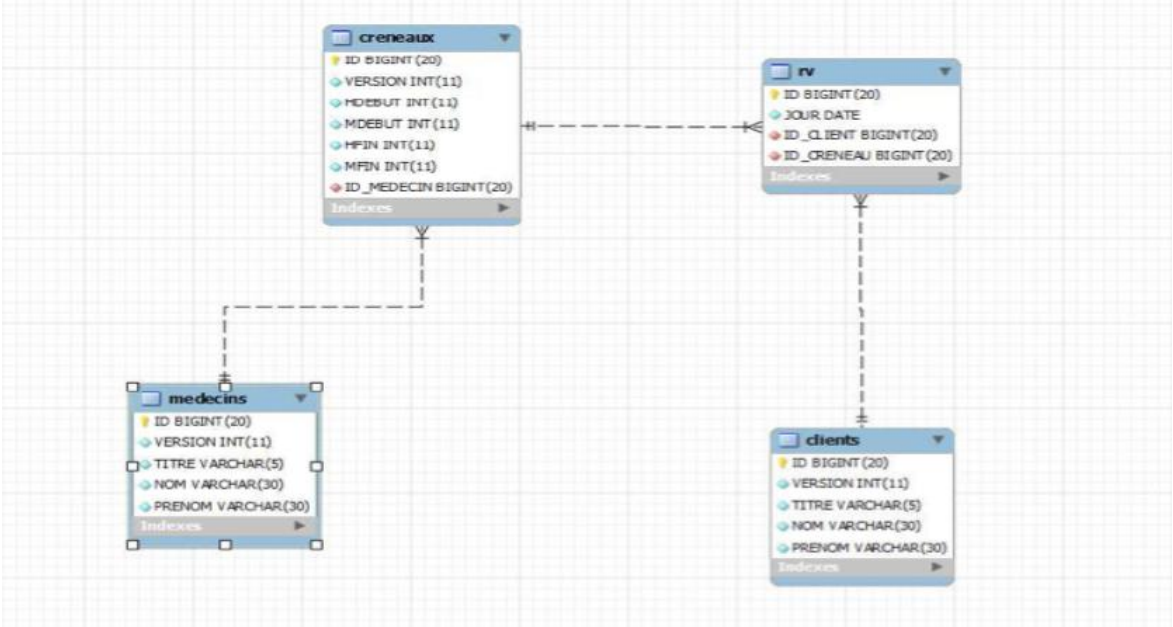


Figure 1 : Diagramme de classe

Chapitre 2: La réalisation

1. MVC

L'application utilise un MVC modèle, pattern MVC est utilisé pour l'interaction entre les couches métiers et présentation.

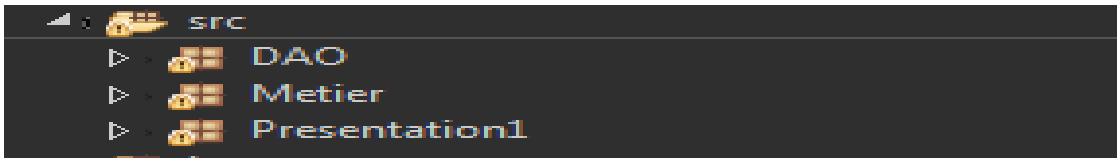
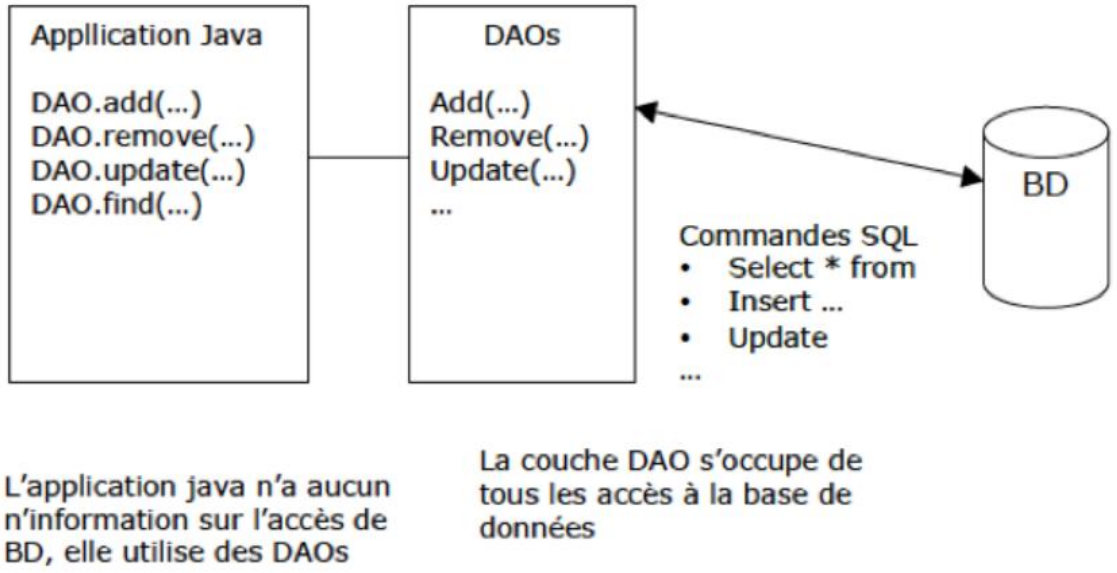


Figure 2 : Répertoire src d'application

2. DAO



En bref, le modèle de conception d'objet d'accès aux données ou DAO est un moyen de réduire le couplage entre la logique métier et la logique de persistance. La logique métier de l'application a souvent besoin d'objets de domaine persistants dans l'une ou l'autre des bases de données.

Si vous utilisez le modèle de conception DAO pour accéder à la base de données, ce serait relativement sûr car vous n'avez besoin que de modifier la couche d'accès aux données.

Dans L'application : chaque classe a une classe DAO and DAOImpl

```java
package DAO;

public class Client {

    private int id;
    private int version;
    private String titre;
    private String nom;
    private String prenom;


    public Client(int id, int version, String titre, String nom, String prenom) {
        this.setId(id);
        this.version = version;
        this.titre = titre;
        this.nom = nom;
        this.prenom = prenom;
    }

    public int getVersion() {
        return version;
    }
    public void setVersion(int version) {
        this.version = version;
    }


    public String getTitre() {
        return titre;
    }
    public void setTitre(String titre) {
        this.titre = titre;
    }


    public String getNom() {
        return nom;
    }
```

```java
public class ClientDaoImpl implements ClientDAO1 {

    private Connection cnx ;

    public ClientDaoImpl() {
        this.cnx = connection.getConx();

    }


    public List<Client> getAllClients()
    {
         List<Client> clients = new ArrayList<Client>();

        String sql = "select * from client";
        try {

            Statement ps = cnx.createStatement() ;
            ResultSet rs = ps.executeQuery(sql);
            Client C;
            while(rs.next())
            {
                C = new Client(rs.getInt("id"), rs.getInt("version"),rs.getString("titre"), rs.getString("nom"), rs.get

              clients.add(C);
            }

        }
        catch (Exception e)
        {
            System.out.println(e);
            e.getStackTrace();
        }
```

3. Pattern Singleton

Singleton Pattern dit qu'il suffit de "définir une classe qui n'a qu'une seule instance et lui fournit un point d'accès global".

En d'autres termes, une classe doit garantir que seule une instance unique doit être créée et qu'un seul objet peut être utilisé par toutes les autres classes.

```java
package DAO;

import java.sql.Connection;

public class connection {


        private static Connection cnx ;

        static {
            String url = "jdbc:sqlserver://USER-PC\\SQLEXPRESS;"+"databaseName=cabinetMedicale;integratedSecurity=true";

            try {
                 Class.forName("com.microsoft.sqlserver.jdbc.SQLServerDriver");
                 cnx = DriverManager.getConnection(url);
                 System.out.println("connecter avec succes");
            }
            catch (Exception e)
            {
                 System.out.println(e);
            }

        }

        public static Connection getConx() {
            return cnx;
        }


}
```

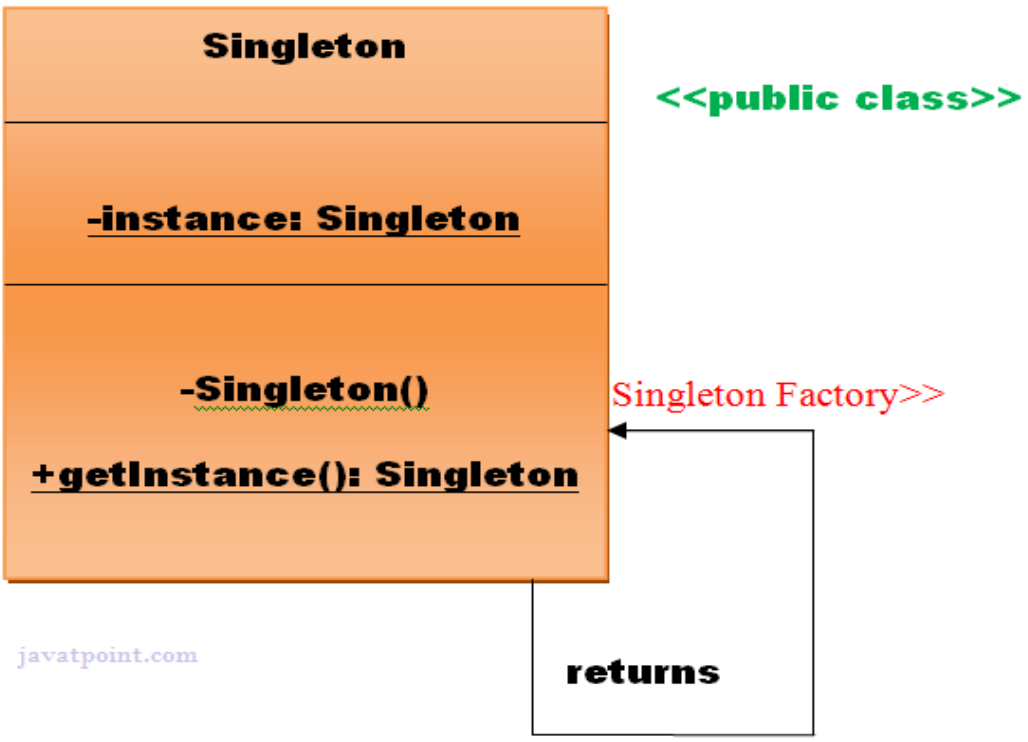


Figure 3 : UML de pattern Singleton

4. Présentation de l'application

a. Interface d'authentification

À la lance de mon application, une fenêtre s'affiche à l'écran, elle demandera d'introduire le nom d'utilisateur et le mot de passe déjà attribué par l'administrateur pour commencer à utiliser notre application.

- L'administrateur va cliquer sur le bouton Accueil pour afficher cette dernière. .

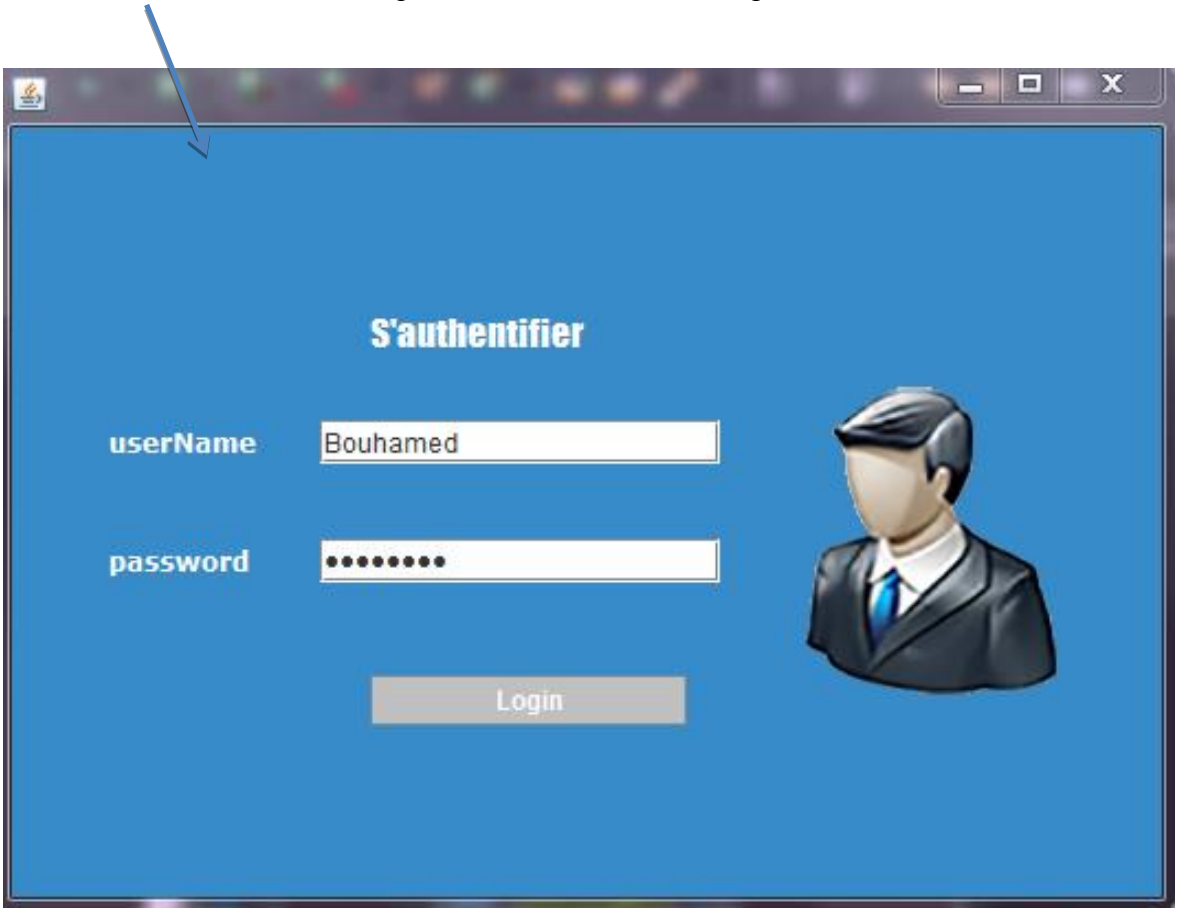
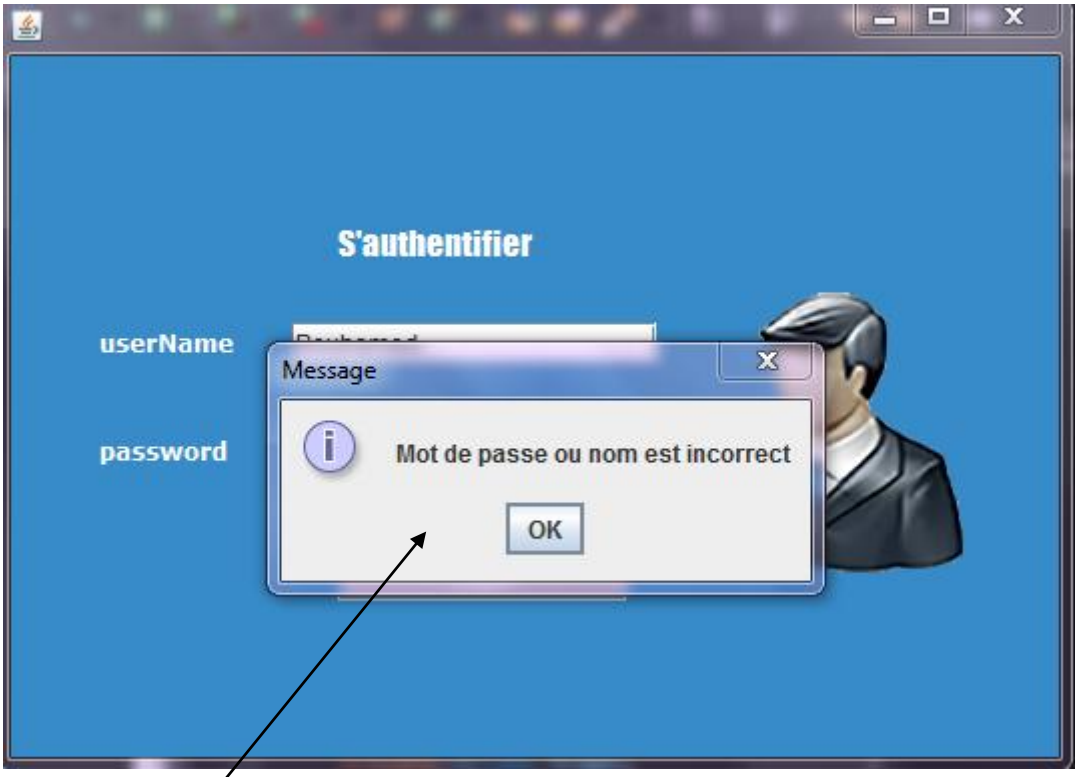


Figure 4 : Interface d'authentification



Si le nom d'utilisateur introduit, ou le mot de passe, n'est pas valide, alors l'application renvoi le message d'erreur suivant

b. Gestion des Clients

Cette interface est muni de 4 boutons commande (ajouter, supprimer, modifier et Actualiser Client) qui gère quelque fonctionnalités de notre base de donné.et aussi on peut L'Administrateur peut trouver un client l'aide de l'action recherche.

- l'administrateur va cliquer sur le bouton ajouter pour afficher une fenêtre client (figure 4: ajouter client).

- Pour modifier un client. L'administrateur remplit les champs à modifier puis il clique sur le bouton modifier, enfin le système va afficher un message pour dire que la modification et bien fait.

- Pour supprimer un client. L'administrateur va entrer seulement id client, puis il va cliquer sur le bouton supprimer .le système va afficher un message pour dire que la suppression et bien fait.

- L'administrateur va cliquer sur le bouton actualiser tous pour voir les informations du sous formulaire apparaître.

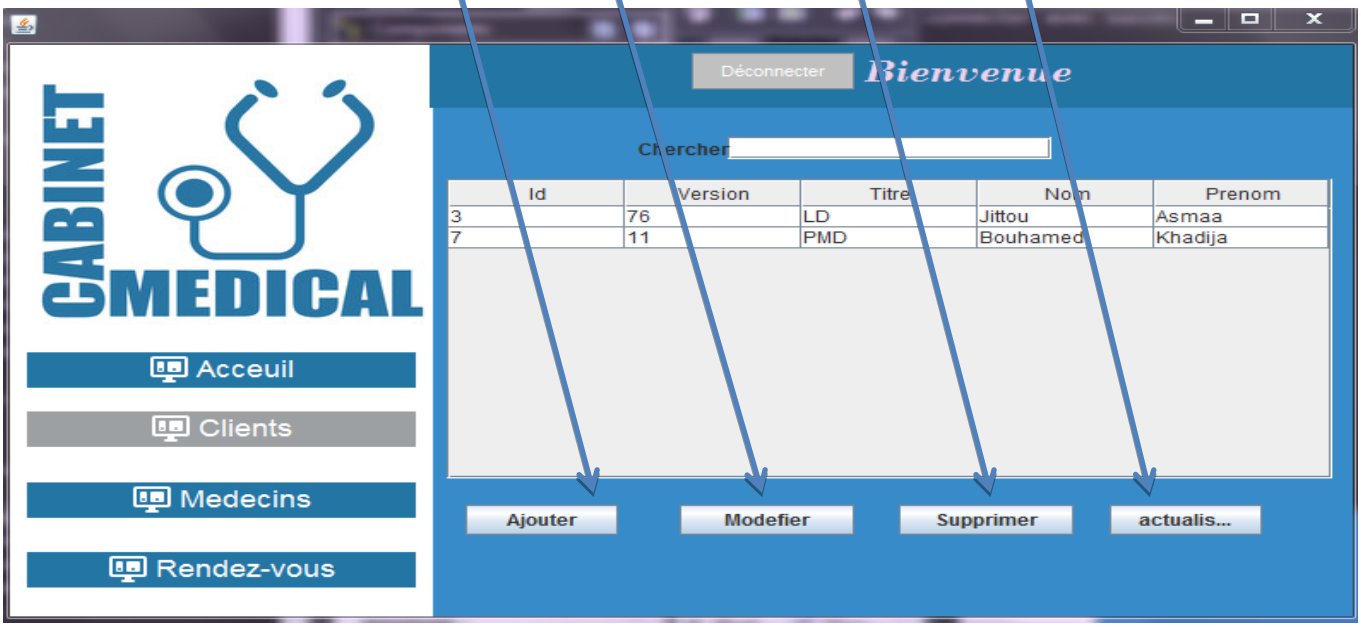


Figure 5 : Gestion des Clients

- Dans cette fenêtre client, l'administrateur peut ajouter un client en remplissant tous les champs pour gérer les nouveaux clients, et puis appuyer sur le bouton ajouter pour enregistrer ces informations dans la table client de notre base de données.

- l'administrateur a voulu annuler l'ajout d'un client en cliquant sur le bouton fermer.

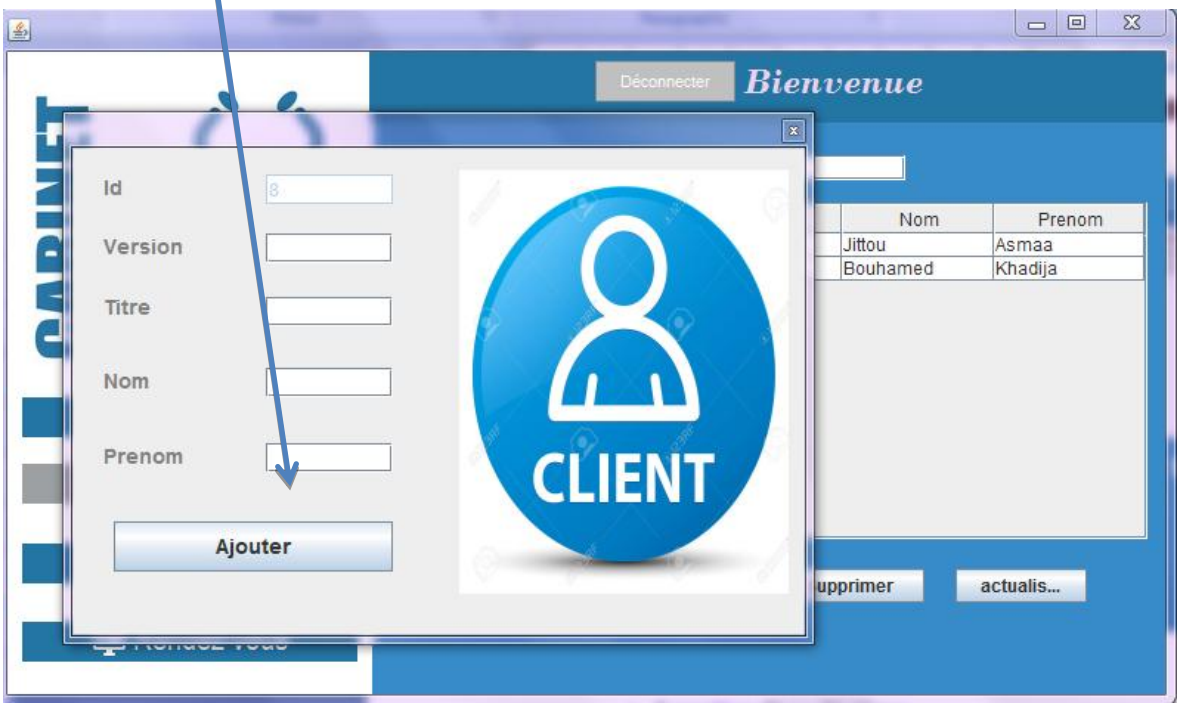


Figure 6 : Ajouter Client

c. Gestion des RDVs

La gestion des rendez-vous est une tâche essentielle de la secrétaire, celle-ci vérifie la disponibilité de la date demandée et par la suite elle ajoute un rendez-vous en saisissant les renseignements nécessaires

- Dans cette interface, l'administrateur peut ajouter un rendez-vous en remplissant tous les champs, et puis appuyer sur le bouton valider pour enregistrer ces informations dans la table voiture de notre base de données

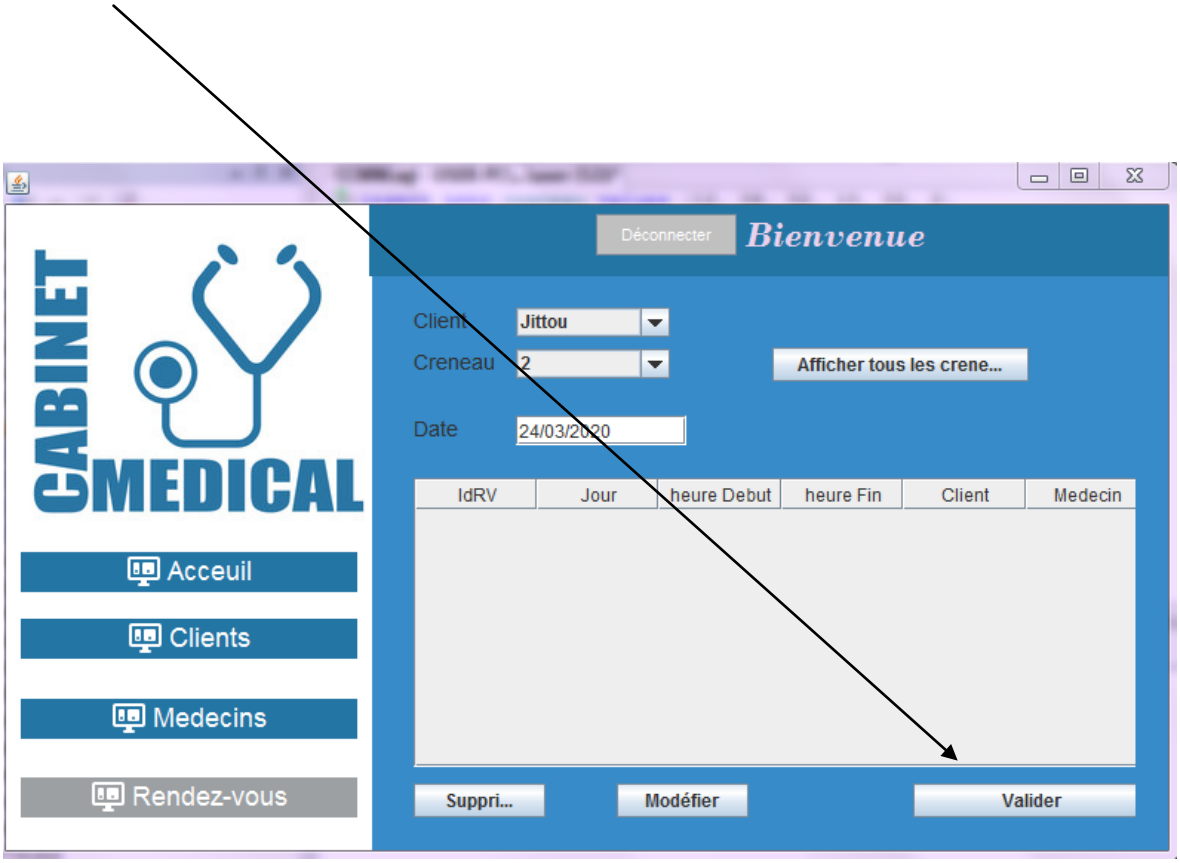
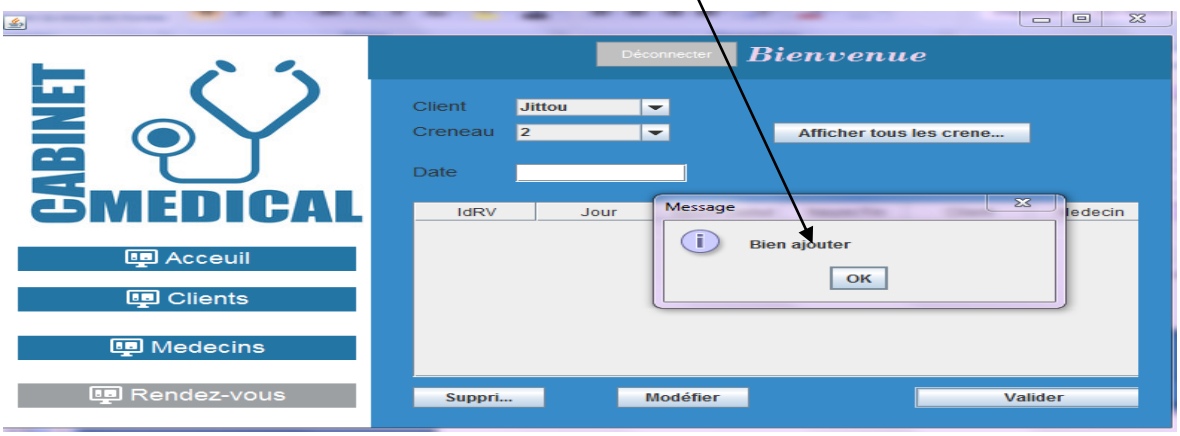


Figure 7 : Location d'une Voiture

- le système va afficher un message pour dire que l'ajout et bien fait



Chapitre 3: Conclusion

Conclusion général :

Dans le cadre de notre projet, nous avons conçu et développé une application qui assure la gestion d'un cabinet médicale.

Ce projet nous a donné de plus l'occasion de maitriser Eclipse, SQL Server et de maitriser les langages de programmation JAVA, en utilisant les pattern de conception DAO et Singleton.