



UNIVERSITÉ TUNIS EL MANAR
FACULTÉ DES SCIENCES DE TUNIS

RAPPORT DU MINI PROJET :

PRÉSENTÉ PAR :
MAROUEN FOURAT
MEDIOUNI ADEM
HOUDI ABDERRAZEK

THÈME :

DÉVELOPPEMENT D'UNE APPLICATION JAVA DE GESTION
DE LOCATION DE VOITURE .

RESPONSABLE :

HOUDA ALAYA
MARIEM BOUSSAID

ANNÉE UNIVERSITAIRE : 2023/2024

Membres et Répartitions des Tâches :

Premièrement, la partie conception (diagramme EA /Diagramme de cas d'utilisation / Diagramme de classe) , le développement de la base de données et l'interface LOGIN / SIGNUP et l'interface Mécanicien ont été faites par tout le groupe ensemble.

Par contre, la partie application java a été créé chacun a sa propre tâche :

Pour les interfaces java la répartition est :

MEDIOUNI ADEM : interface ADMIN (gestion client , gestion mécanicien , afficher les listes de voitures avec critères de recherche)

HOUIDI ABDERRAZEK: interface ADMIN (gestion voiture , afficher client régulier , afficher les voitures les plus vendu)

MAROUEN FOURAT : interface CLIENT

Description de l'application :

Cette application permet de gérer les différentes activités au sein d'une agence de location de voiture.

Elle est utilisée par soit un admin ou un client ou un mécanicien .

Chaque membre a sa propre interface ou il a des fonctionnalités et des données propres à lui.

Tous les utilisateurs au début doivent s'identifier grâce à un identifiant et un mot de passe. Chaque membre a sa propre interface de login qui redirectionne le membre à sa propre interface.

Cette application permet à un visiteur de :

- s'inscrire pour devenir un client .

Cette application permet à l'admin de :

- Ajouter un mécanicien , un client et une voiture.
- supprimer un mécanicien , un client et une voiture.
- Modifier l'adresse , le tel , l'email de chaque client et mécanicien .
- Modifier l'état , la disponibilité et le prix de chaque voiture .
- enlever le ban d'un client .

L'interrogation de la base de données pour afficher :

- a. Les voitures existantes et leurs états (en panne ou en marche).
- b. Les voitures disponibles et celles qui sont louées.
- c. Les voitures les plus demandées (les trois premières).
- d. Le client régulier (en fonction du nombre de fois où il a loué des voitures).

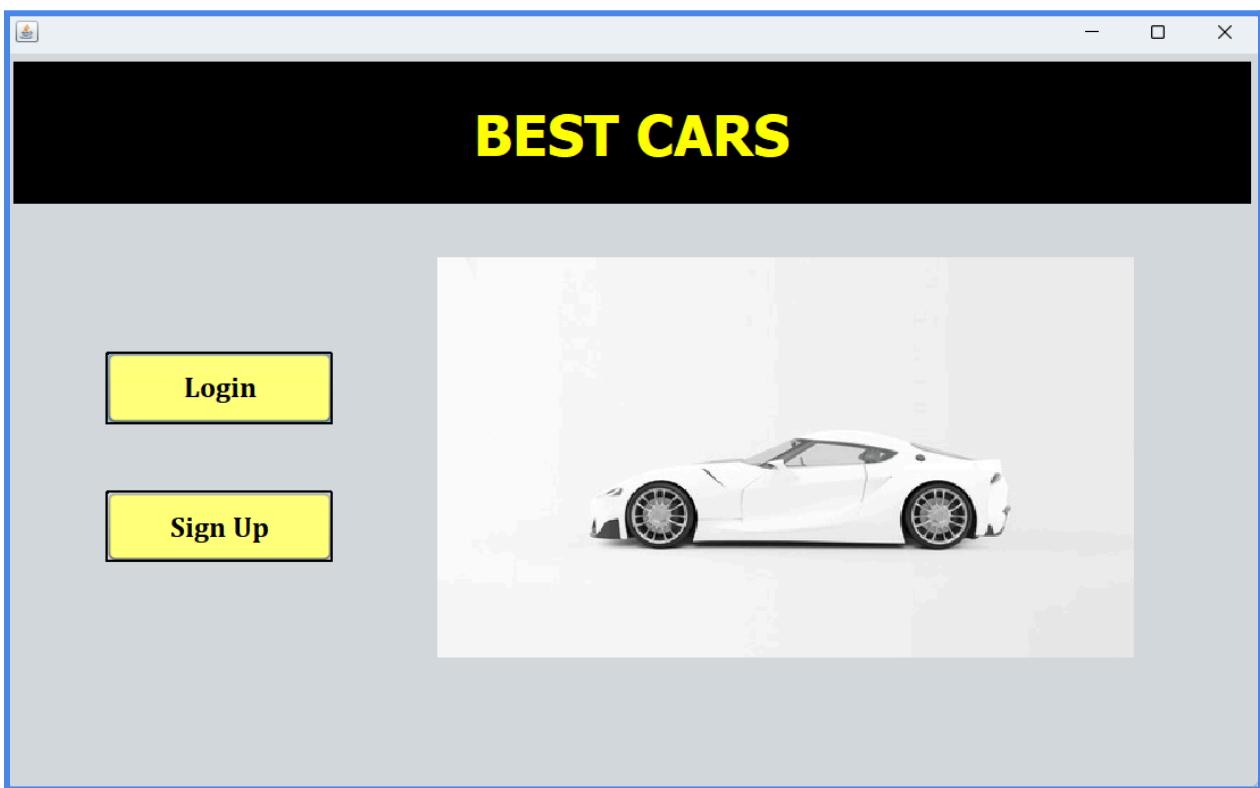
Cette application permet au Client de :

- réserver une voiture .
- prendre une voiture (après une réservation)
- payer location .
- retourner voiture (après la prendre) et en cas de retard :
 - il doit payer une pénalité .
 - Il va être banni si le retard dépasse 4 jours .

Cette application permet au mécanicien de :

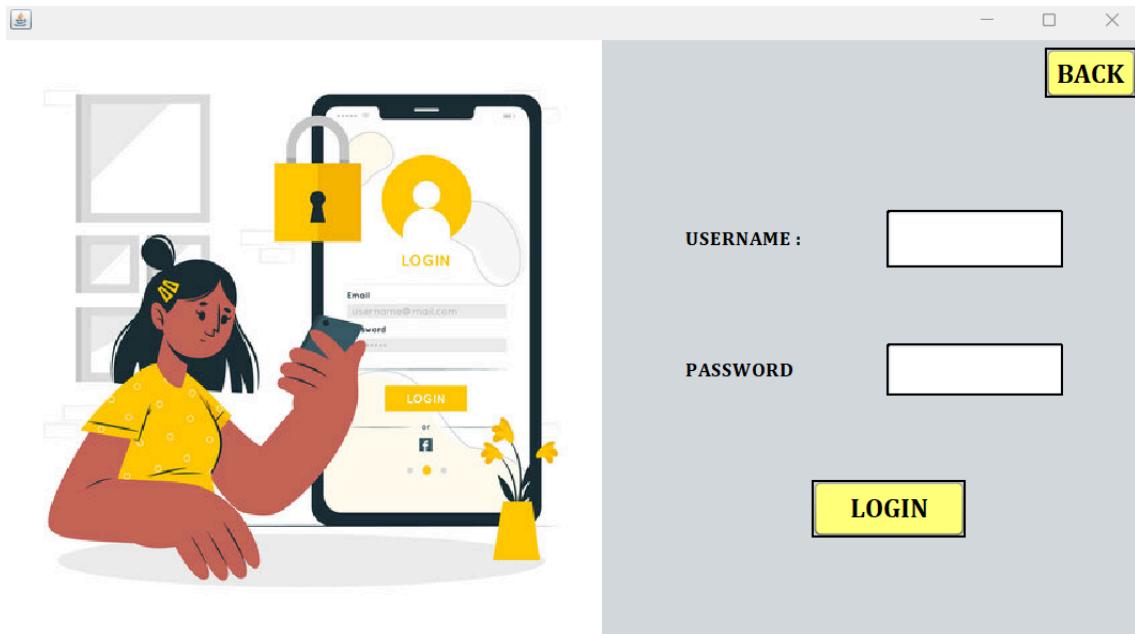
- Consulter la liste des voitures en panne et réparer la voiture qu'il choisit .
- Consulter la liste des voitures en attente et réviser leurs états .

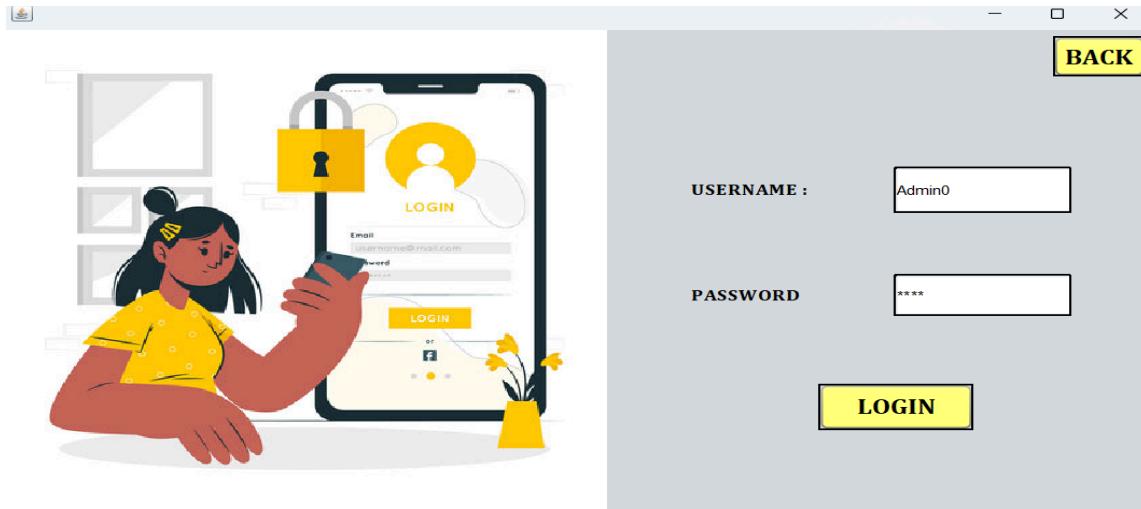
Page principale :



Interfaces ADMIN:

login admin:





code:

```

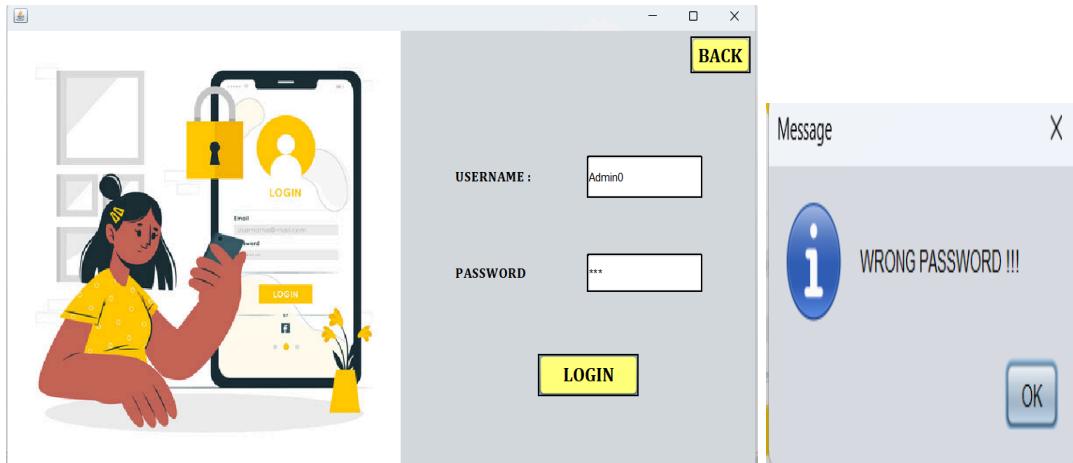
private void loginActionPerformed(java.awt.event.ActionEvent evt) {
    String username = username_admin.getText();
    char[] passwordChars = password.getPassword();
    String password_user = new String(value:passwordChars);
    if(username.equals(anObject: "") | password_user.equals(anObject: "")){
        JOptionPane.showMessageDialog(parentComponent:null,message:" VEILLEZ REMPLIRE TOUS LES CHAMPS      !!!");
    }else{
        Connecteur connect=new Connecteur();
        Connection con;
        con = connect.connecttodb();

        String query = ("SELECT * FROM administrateur WHERE username='"+username+"'");
        try {
            Statement stmt = con.createStatement(i: ResultSet.TYPE_SCROLL_INSENSITIVE,i: ResultSet.CONCUR_READ_ONLY);
            ResultSet rs = stmt.executeQuery(string: query);
            if(rs.next()){
                rs.first();
                if(rs.getString(string: "mot_de_pass").equals(anObject:password_user)){
                    this.dispose(); //tsakker
                    admin ad =new admin(); //thel wahda jdida
                    ad.setLocationRelativeTo(e: null);
                    ad.setVisible(b: true);
                }else{
                    JOptionPane.showMessageDialog(parentComponent:null,message:"WRONG PASSWORD !!!");
                }
            }else{
                JOptionPane.showMessageDialog(parentComponent:null,message:" NO ACCOUNT WITH THIS CIN !!!");
            }
        } catch (SQLException ex) {
            Logger.getLogger(LoginClient.class.getName()).log(level: Level.SEVERE, msg:null, thrown: ex);
        }
    }
}

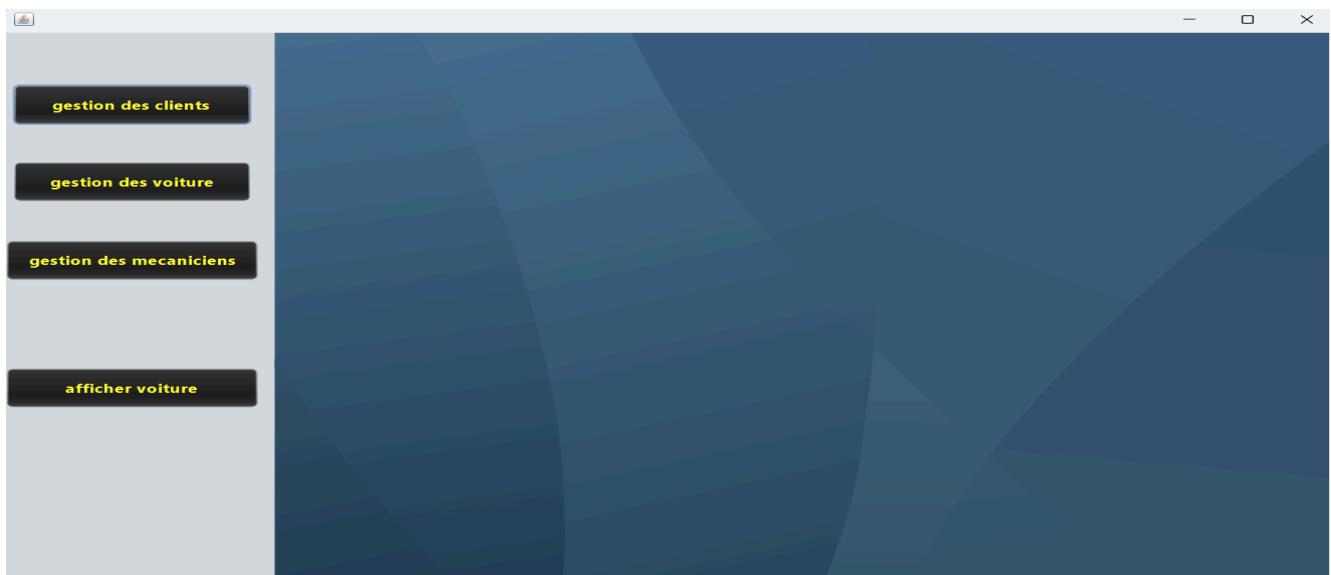
```

en cliquant le bouton login une requête SQL va être envoyé va tester l'existence du username correct de l'admin puis un test sur la validité du mot de pass.

et cela le cas ou en fait entre un mot de pass incorrect.



INTERFACE ADMIN PRINCIPALE:



en cliquant sur le bouton gestion de client on va aller a l'interface de gestion de client:

interface ajouter client:

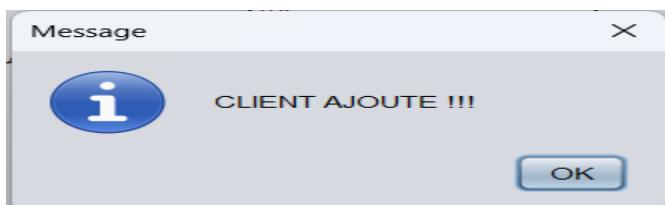
AJOUTER CLIENT

CIN	77777777
NOM	adem
PRENOM	medyouni
DATE DE NAISSANCE	17 avr. 2003
TEL	99887798
ADDRESS	jendouba
USERNAME	adem
EMAIL	emmedyouni@gmail.com
PASSWORD	****

AJOUTER

gestion des clients
gestion des voiture
gestion des mecaniciens
afficher voiture

ajouter
supprimer
modifier
list banned
best client



code:

```

private void confirmActionPerformed(java.awt.event.ActionEvent evt) {
    String cin_user = cin.getText();
    String nom_user = nom.getText();
    String username_user = username.getText();
    String prenom_user = prenom.getText();
    String tel_user = tel.getText();
    String address_user = address.getText();
    String email_user = email.getText();
    char[] passwordChars = password.getPassword();
    String password_user = new String(value: passwordChars);
    Date date_user = date.getDate();
    Client client = new Client(cin:cin_user, nom:nom_user, prenom:prenom_user, email:email_user, date_de_naissance:date_user, username:username_user, tel:tel_user, address:address_user, mot_de_passe:password_user);

    if(!verif(cin:cin_user) ) {
        if(cin_user.equals(anObject: "")){
            JOptionPane.showMessageDialog(parentComponent: null, message: "Veuillez remplir votre CIN !!!");
        }else{
            JOptionPane.showMessageDialog(parentComponent: null, message: "Veuillez verifier votre CIN !!!");
        }
    }else{
        if (nom_user.equals(anObject: "") | prenom_user.equals(anObject: "") | tel_user.equals(anObject: "") | date_user==null | address_user.equals(anObject: "")){
            JOptionPane.showMessageDialog(parentComponent: null, message: "Veuillez remplir tous les champs !!!");
        }else{
            java.sql.Date sqlDate1 = new java.sql.Date(date: date_user.getTime());
            String datel=sqlDate1.toString();
            if (!isValidEmailAddress(email: email_user)) {
                JOptionPane.showMessageDialog(parentComponent: null, message: "L'email est invalide !!!");
            }else{
                if (password_user.equals(anObject: "")){
                    JOptionPane.showMessageDialog(parentComponent: null, message: "Veuillez remplir le mot de passe !!!");

                }else{
                    Connecteur connect=new Connecteur();
                    Connection con;
                    con =connect.connecttodb();
                    try {
                        Statement stmt = con.createStatement();
                        String query="SELECT * FROM banned where cin ="+cin_user+"";
                        ResultSet rs = stmt.executeQuery(string: query);

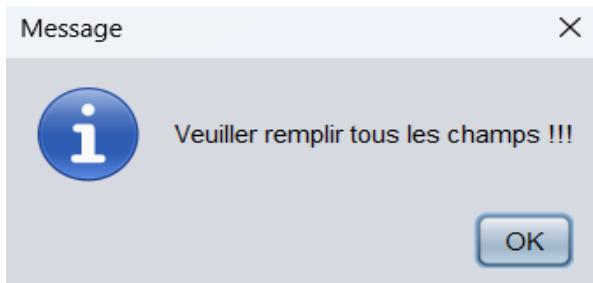
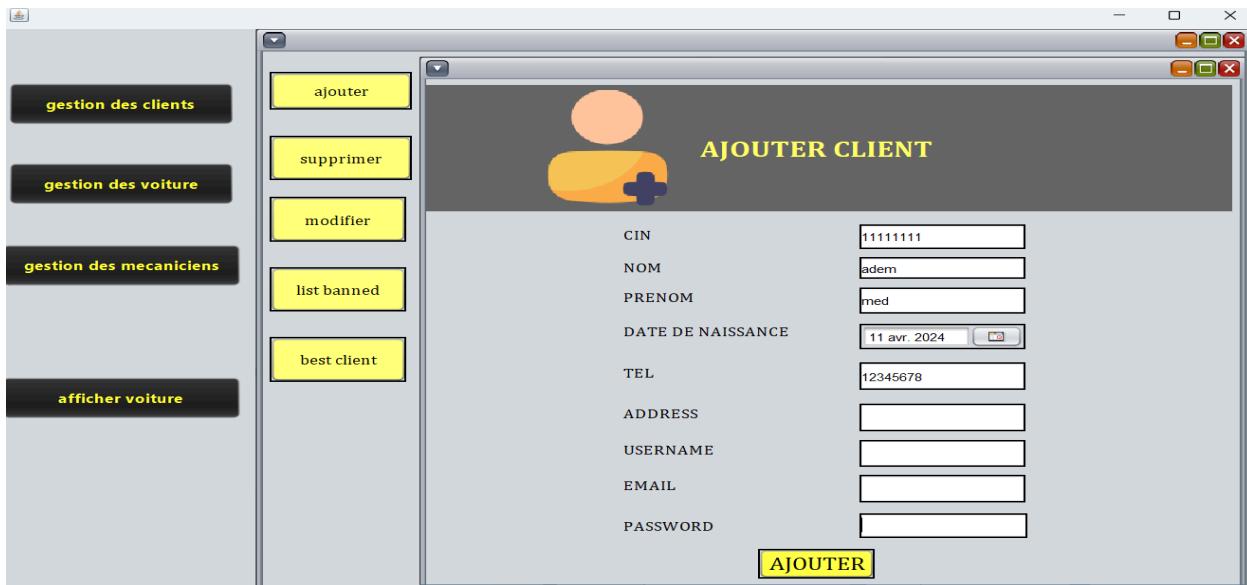
                        if(rs.next()){
                            JOptionPane.showMessageDialog(parentComponent: null, message: "CE CLIENT EST BANNEE !!!", title: "Error", messageType:JOptionPane.ERROR_MESSAGE);
                        }else{
                            query="SELECT * FROM client where cin ="+cin_user+"";
                            rs=stmt.executeQuery(string: query);
                        }
                    }
                }
            }
        }
    }
}

```

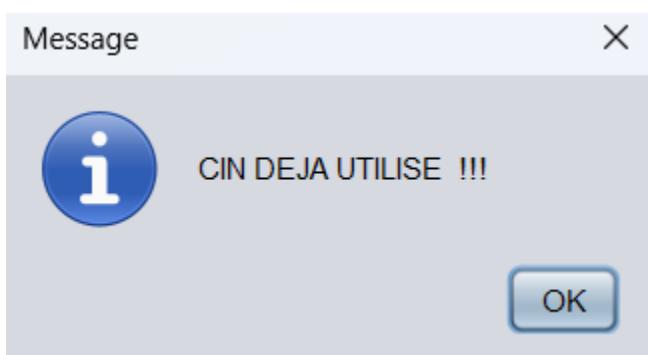

en cliquant sur le bouton ajouter une requête SQL va être envoyé pour tester

Premièrement que le client n'est pas banni puis qu'il n'existe aucun client identique à celui qu'on veut ajouter et aussi la validité de l'email qu'on va ajouter après vérifier les conditions on envoie une requête insert into table client pour ajouter le client.

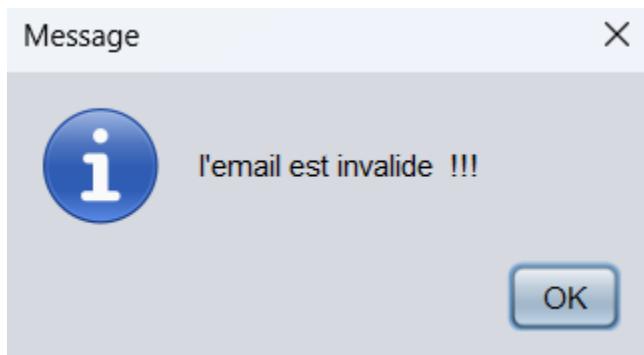
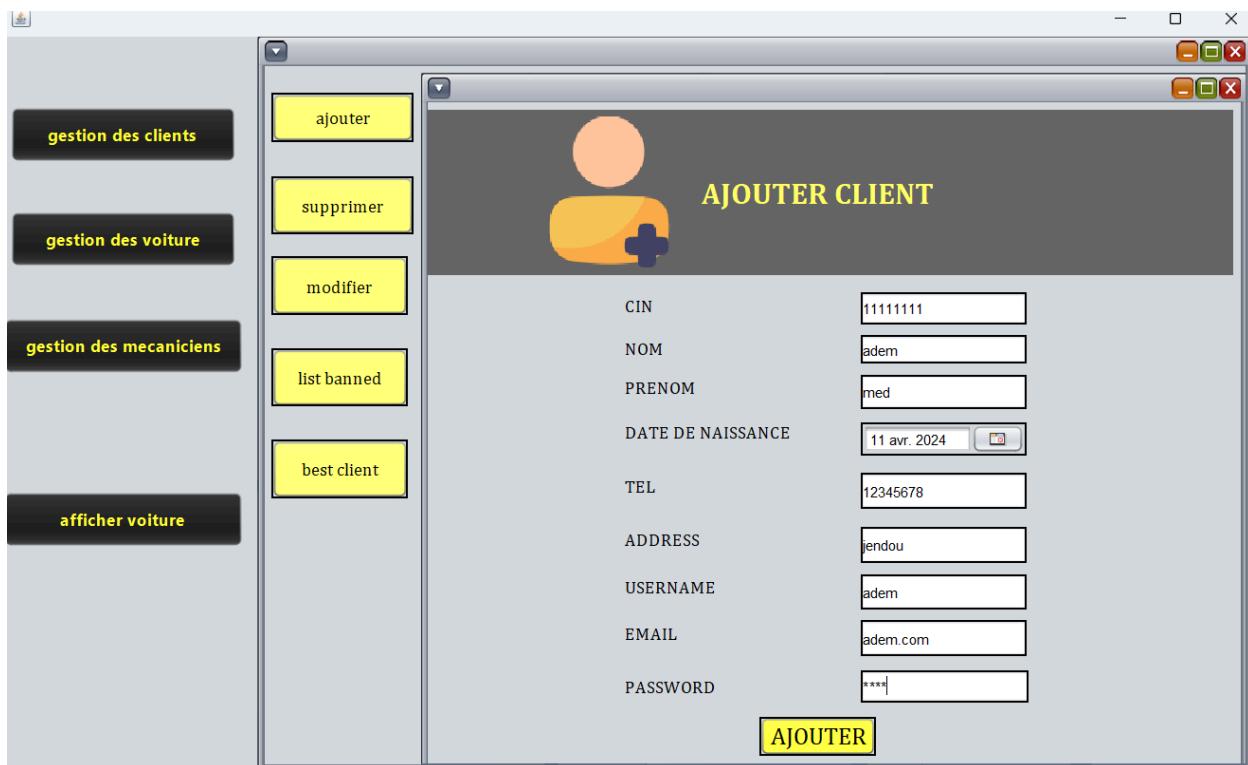
le cas ou en fait entrer des champs vide :



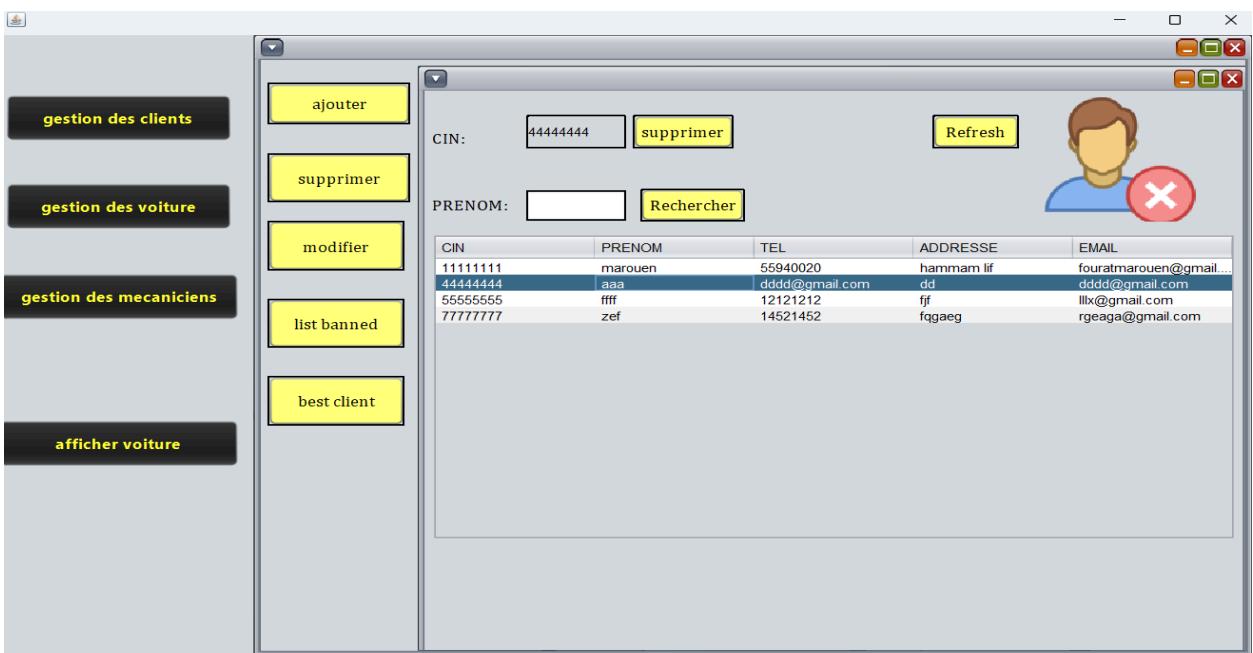
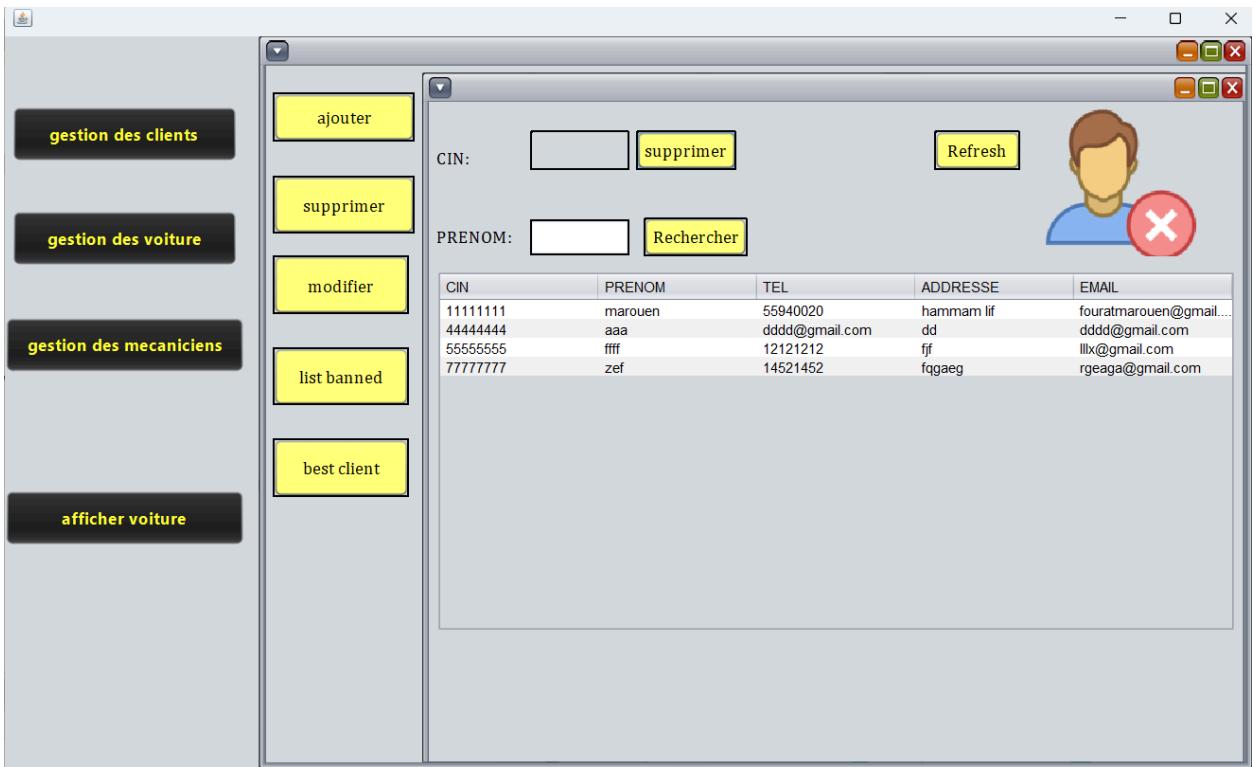
le cas ou en fait entrer un client déjà existant:

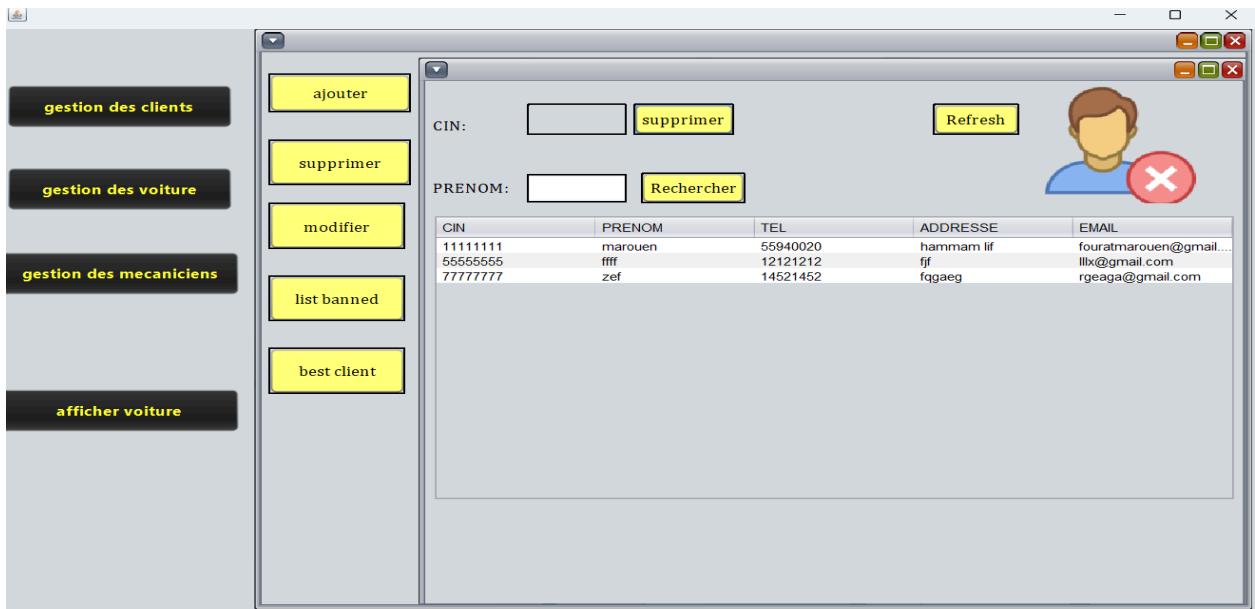


le cas où on fait entrer un email non valide :



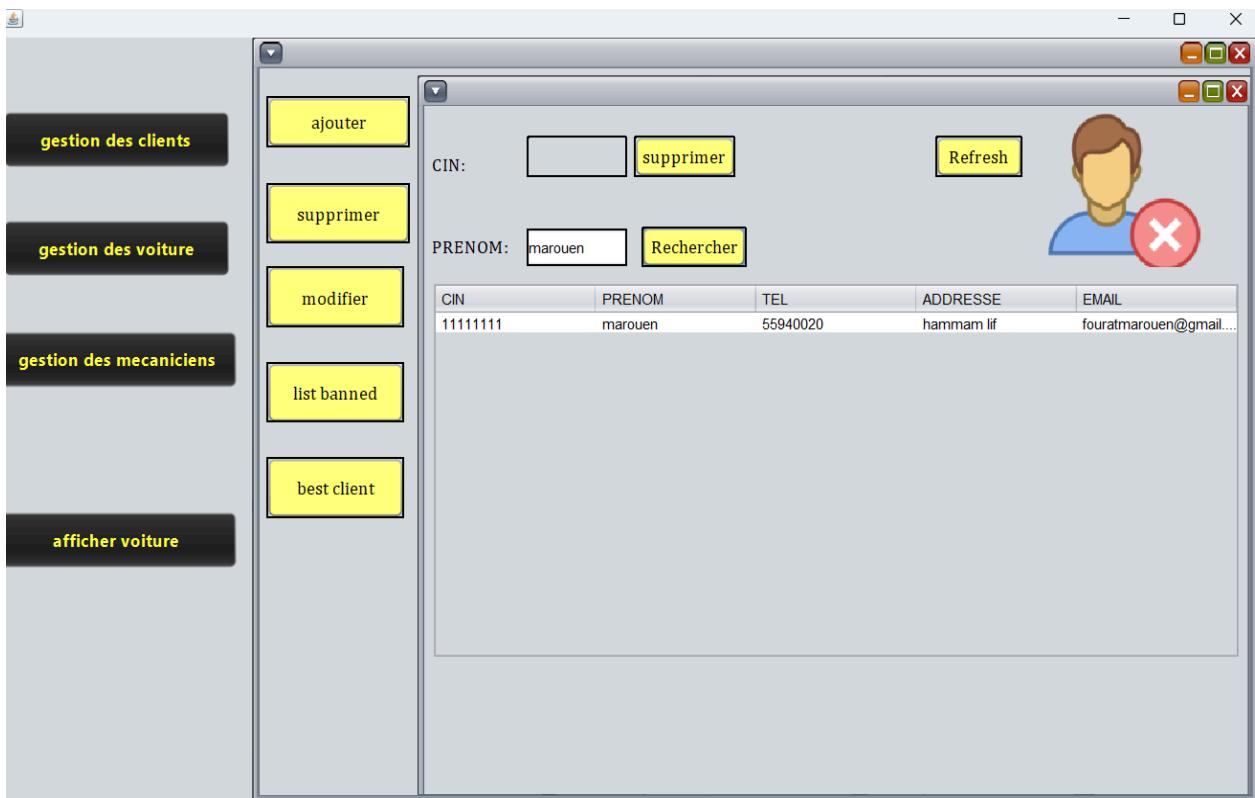
interface supprimer client:





on va trouver dans l'interface de supprimer un client un tableau de la liste de client en sélectionnant le client puis en appuyant sur le bouton supprimer

le client va être supprimée de la table client et aussi en peut chercher un client suivant son prénom.



code:

la fonction load qui fait afficher la liste de client est lancer dans le constructeur de la classe delete client

```
public class deleteclient extends javax.swing.JInternalFrame {
    private javax.swing.JDesktopPane jDesktopPane3;

    /**
     * Creates new form deleteclient
     */
    public deleteclient() {
        initComponents();
    }
    public deleteclient(javax.swing.JDesktopPane jDesktopPanel) {

        this.jDesktopPane3=jDesktopPanel;
        initComponents();
        load();
    }
    private Connection con ;
    public void load(){
        Connecteur connect=new Connecteur();

        con = connect.connecttodb();
        String query = "SELECT * FROM client";
        DefaultTableModel tablemodel =(DefaultTableModel) table.getModel();
        tablemodel.setRowCount(rowCount: 0);
        try {
            Statement stmt = con.createStatement();
            ResultSet res = stmt.executeQuery(string: query);

            while(res.next()){
                String cin = res.getString(string: "CIN");
                String prenom = res.getString(string: "prenom");
                String address = res.getString(string: "adresse");
                String tel = res.getString(string: "tel");
                String email =res.getString(string: "email");
                tablemodel.addRow(new Object[]{cin,prenom,tel,address,email});
            }
        } catch (SQLException ex) {
            Logger.getLogger(name:listban.class.getName()).log(level: Level.SEVERE, msg:null, thrown: ex);
        }
        this.name_field.setText(s: "");
        this.cin_field.setText(text: "");
    }
}
```

le code la fonction de recherche pour le client:

```
private void rechercheActionPerformed(java.awt.event.ActionEvent evt) {
    Connecteur connect=new Connecteur();
    con = connect.connecttodb();
    String prenom = name_field.getText();
    String query = "SELECT * FROM client WHERE prenom = '"+prenom+"'";
    DefaultTableModel tablemodel =(DefaultTableModel) table.getModel();
    tablemodel.setRowCount(rowCount: 0);
    try {
        Statement stmt = con.createStatement();
        ResultSet res = stmt.executeQuery(string: query);

        while(res.next()){
            String cin = res.getString(string: "cin");
            prenom = res.getString(string: "prenom");
            String tel = res.getString(string: "tel");
            String address = res.getString(string: "adresse");
            String email =res.getString(string: "email");
            tablemodel.addRow(new Object[]{cin,prenom,tel,address,email});
        }
    } catch (SQLException ex) {
        Logger.getLogger(name:updateclient.class.getName()).log(level: Level.SEVERE, msg:null, thrown: ex);
    }
}
```

enfin ceci est le code de suppression qui se fait en envoyant une requête delete from client.

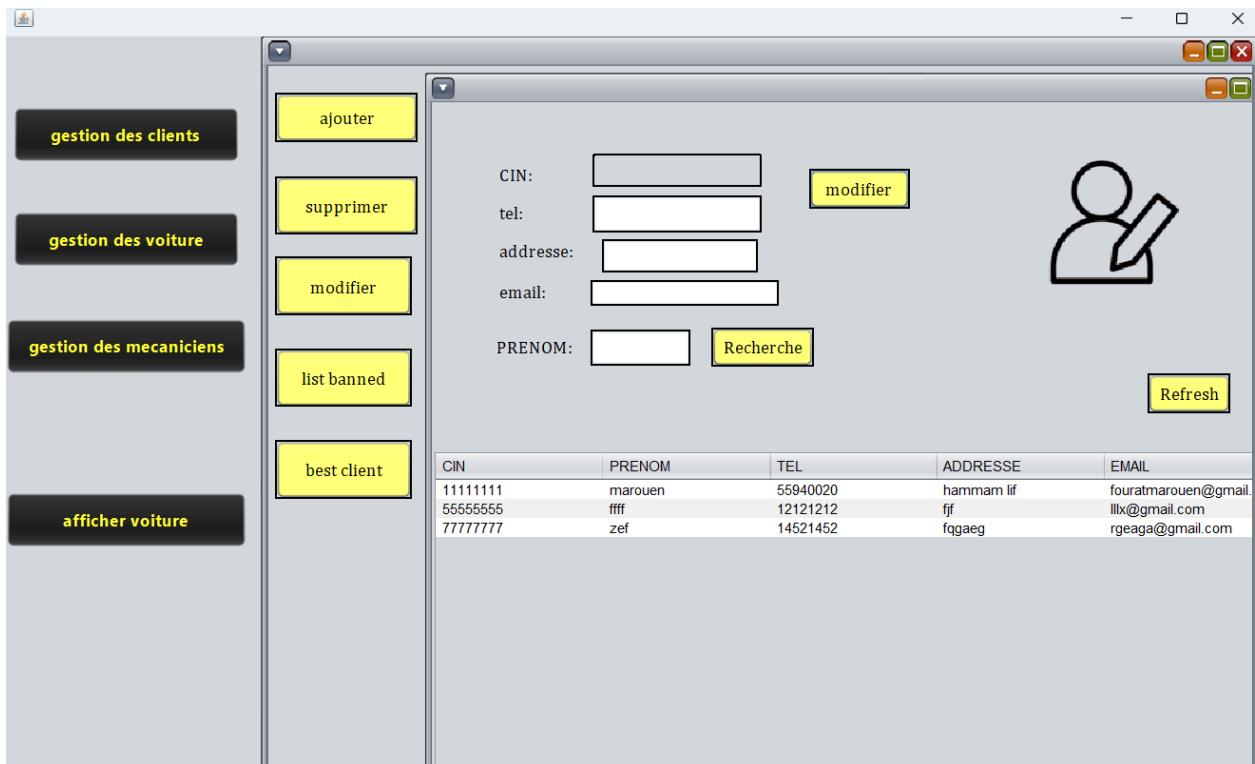
```
private void deleteActionPerformed(java.awt.event.ActionEvent evt) {
    String cin_user = cin_field.getText();
    String query = " DELETE FROM client where CIN=" + cin_user + "";
    Statement stmt;

    try
    {
        stmt = con.createStatement();
        stmt.execute(string: query);
    }
    catch (SQLException ex)
    {
        Logger.getLogger(name: updateclient.class.getName()).log(level: Level.SEVERE, msg: null, thrown: ex);
    }

    cin_field.setText(text: "");
    load();
}
```

interface modifier client:

cette interface permet de modifier le téléphone l'adresse et l'email de client.



Screenshot of the application interface showing the 'gestion des clients' module.

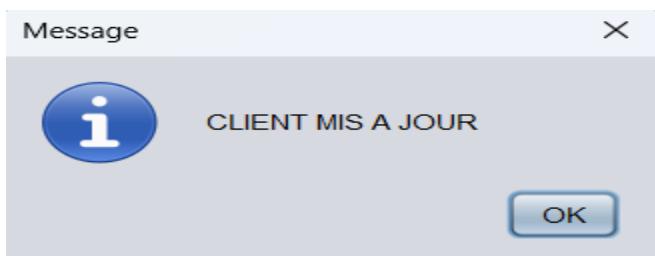
Left Sidebar:

- gestion des clients
- gestion des voiture
- gestion des mecaniciens
- afficher voiture

Central Panel:

- Buttons:** ajouter, supprimer, modifier, list banned, best client
- Text Fields:** CIN: 55555655, tel: 88888888, adresse: tunis, email: test@gmail.com
- Buttons:** modifier, Recherche
- Table:**

CIN	PRENOM	TEL	ADRESSE	EMAIL
11111111	marouen	55940020	hammam lif	fouratmarouen@gmail.com
55555555	ffff	12121212	fff	lllx@gmail.com
77777777	zef	14521452	fqgaeg	rgeaga@gmail.com
- Image:** A user icon with a pencil.
- Buttons:** Refresh



après la modification le client dont le cin est 12121212 va changer l'adresse le tel et l'email avec tous les test de validité de l'email.

Screenshot of the application interface showing the 'gestion des clients' module after modification.

Left Sidebar:

- gestion des clients
- gestion des voiture
- gestion des mecaniciens
- afficher voiture

Central Panel:

- Buttons:** ajouter, supprimer, modifier, list banned, best client
- Text Fields:** CIN: (empty), tel: (empty), adresse: (empty), email: (empty)
- Buttons:** modifier, Recherche
- Table:**

CIN	PRENOM	TEL	ADRESSE	EMAIL
11111111	marouen	55940020	hammam lif	fouratmarouen@gmail.com
55555555	ffff	88888888	tunis	test@gmail.com
77777777	zef	14521452	fqgaeg	rgeaga@gmail.com
- Image:** A user icon with a pencil.
- Buttons:** Refresh

code:

```
package adminCRUD;
] import ig.Connecteur;
import java.sql.Connection;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.swing.JOptionPane;
- import javax.swing.table.DefaultTableModel;

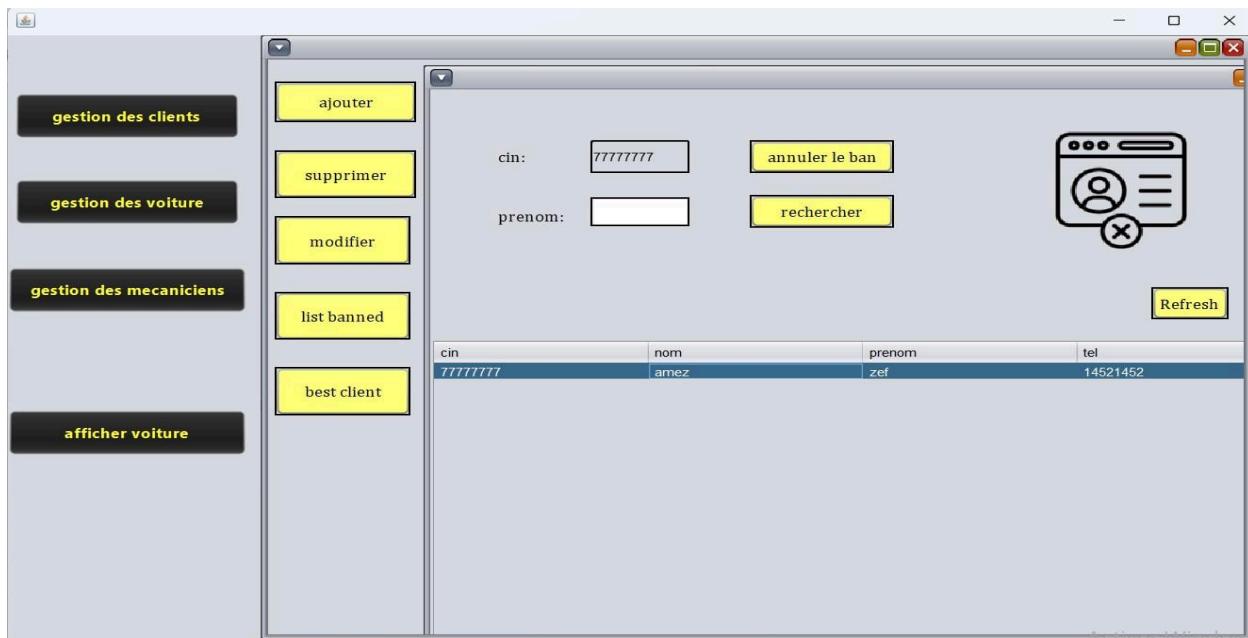
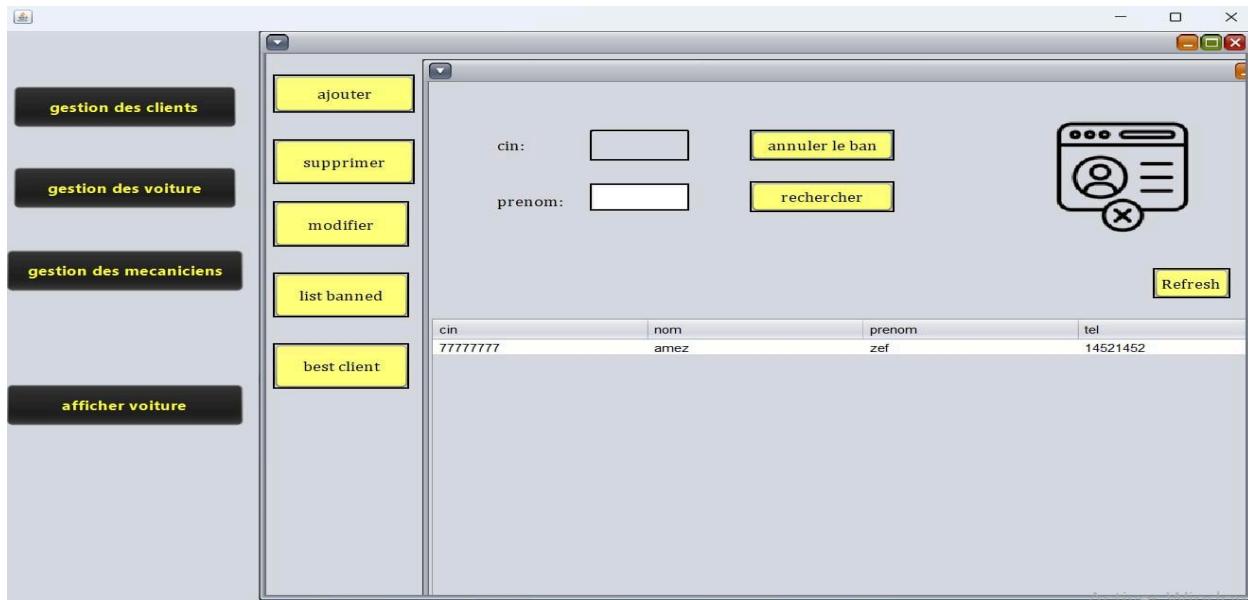
]
/***
 *
 * @author adem
 */
public class updateclient extends javax.swing.JInternalFrame {
    private javax.swing.JDesktopPane jDesktopPane3;

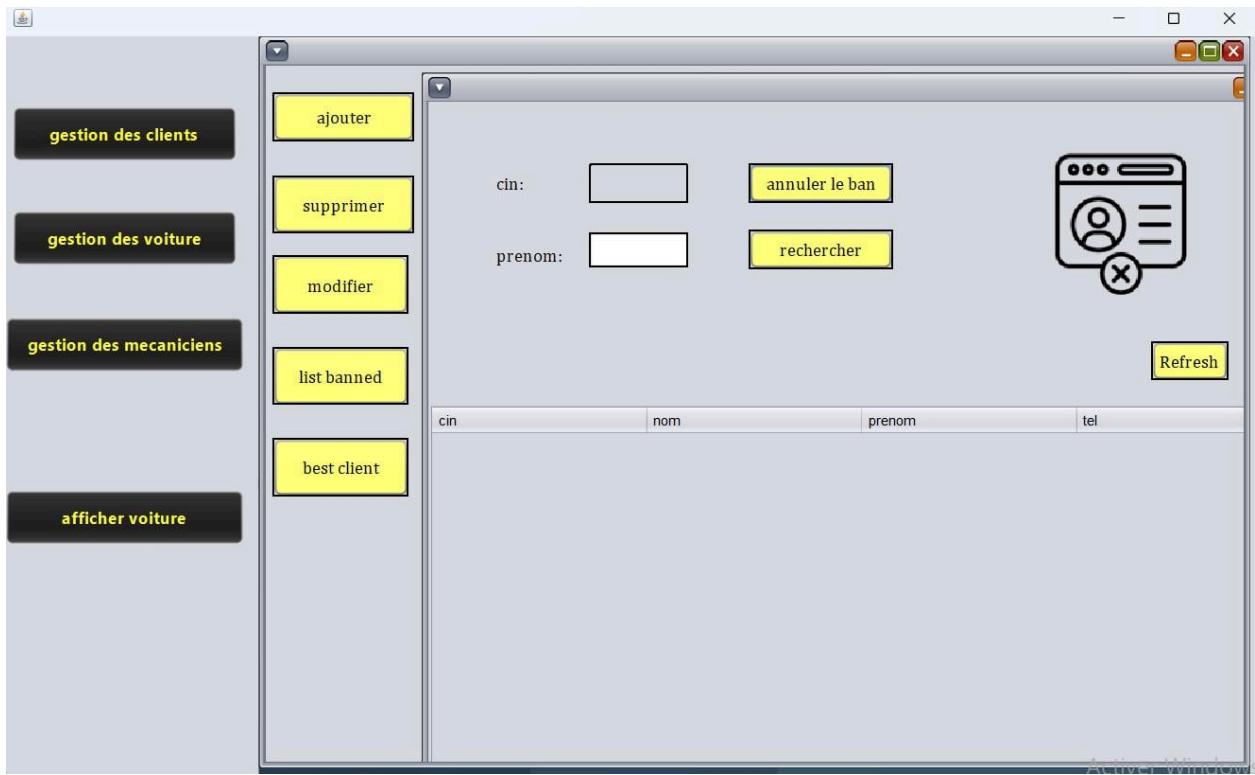
]
/***
 * Creates new form updateclient
 */
] public updateclient() {
    initComponents();
}
] public updateclient(javax.swing.JDesktopPane jDesktopPanel) {

    this.jDesktopPane3=jDesktopPanel;
    initComponents();
    load();
}
}
```

interface liste banned:

cette interface est faite pour afficher la liste des clients bannis et d'enlever le ban si l'admin veut le faire





code:

la fonction load pour afficher la liste des clients banni:

```

private Connection con ;
public void load(){
    Connecteur connect=new Connecteur();

    con = connect.connecttodb();
    String query = "SELECT * FROM banned b join client c on b.cin=c.cin";
    DefaultTableModel tablemodel =(DefaultTableModel) table.getModel();
    tablemodel.setRowCount(rowCount: 0);
    try {
        Statement stmt = con.createStatement();
        ResultSet res = stmt.executeQuery(string: query);

        while(res.next()){
            String cin = res.getString(string: "CIN");
            String nom = res.getString(string: "nom");
            String prenom = res.getString(string: "prenom");
            String tel = res.getString(string: "tel");
            tablemodel.addRow(new Object[]{cin,nom,prenom,tel});

        }
    } catch (SQLException ex) {
        Logger.getLogger(name: listban.class.getName()).log(level: Level.SEVERE, msg:null, thrown: ex);
    }
    this.prenom_field.setText(t: "");
    this.cin_field.setText(text: "");
}

```

fonction du bouton rechercher:

```
private void rechercheActionPerformed(java.awt.event.ActionEvent evt) {
    Connecteur connect=new Connecteur();

    con = connect.connecttodb();
    String prenom = prenom_field.getText();
    String query = "SELECT * FROM banned join client WHERE prenom = '"+prenom+"'";
    DefaultTableModel tablemodel = (DefaultTableModel) table.getModel();
    tablemodel.setRowCount(rowCount: 0);
    try {
        Statement stmt = con.createStatement();
        ResultSet res = stmt.executeQuery(string: query);

        while(res.next()){
            String cin = res.getString(string: "CIN");
            String nom = res.getString(string: "nom");
            prenom = res.getString(string: "prenom");
            String tel = res.getString(string: "tel");
            tablemodel.addRow(new Object[]{cin,nom,prenom,tel});

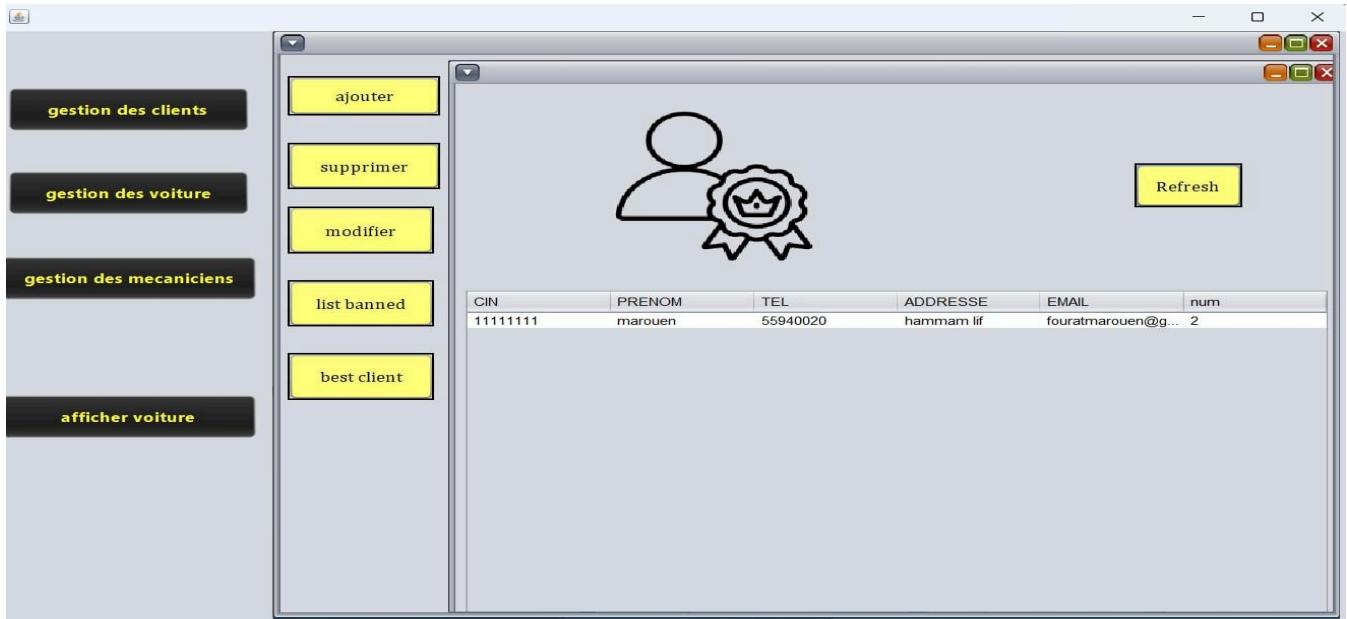
        }
    } catch (SQLException ex) {
        Logger.getLogger(name: listban.class.getName()).log(level: Level.SEVERE, msg:null, thrown: ex);
    }
}
```

le code du bouton annuler le ban qui fait supprimer le client de la liste de ban

```
private void unbanActionPerformed(java.awt.event.ActionEvent evt) {
    Connecteur connect=new Connecteur();
    con = connect.connecttodb();
    String cin = cin_field.getText();
    String queryl=" DELETE from banned where cin=?";
    try (PreparedStatement preparedStatement = con.prepareStatement(string: queryl))
    {
        preparedStatement.setString(i: 1, string: cin);
        preparedStatement.executeUpdate();
    }
    catch (SQLException ex)
    {
        ex.printStackTrace();
    }
    load();
}
```

interface best client:

Cette interface est faite pour afficher le meilleur client.



code:

```
public void load(){
    Connecteur connect=new Connecteur();

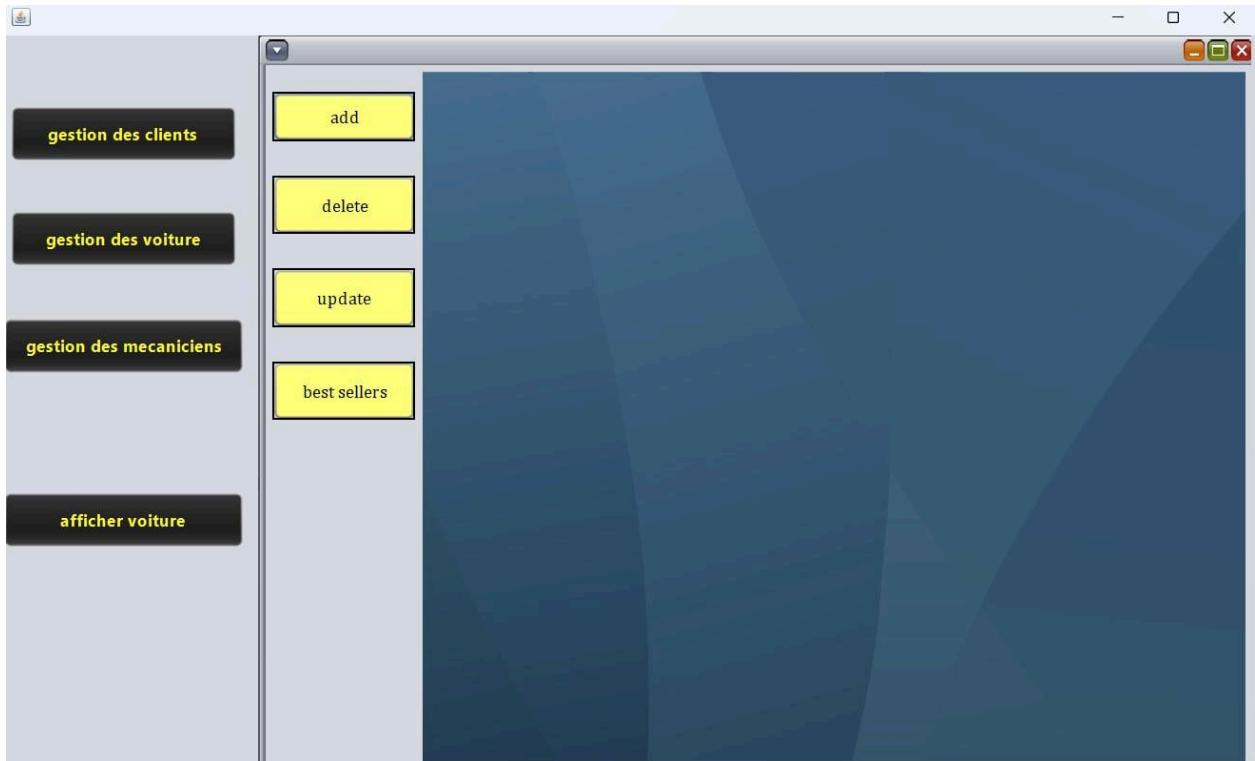
    con = connect.connecttodb();
    String query = "SELECT cin,count(*) as count FROM location group by cin order by count(*) desc  ";
    DefaultTableModel tablemodel =(DefaultTableModel) table.getModel();
    tablemodel.setRowCount(0);
    try {
        Statement stmt = con.createStatement();
        ResultSet res = stmt.executeQuery(query);

        while(res.next()){
            String cin = res.getString("CIN");
            String num = res.getString("count");
            String queryl = "SELECT * FROM client WHERE cin = '"+cin+"'";
            Statement stmtl;
            stmtl = con.createStatement();
            ResultSet resl = stmtl.executeQuery(queryl);
            resl.next();

            String prenom = resl.getString("prenom");
            String address = resl.getString("adresse");
            String tel = resl.getString("tel");
            String email =resl.getString("email");
            tablemodel.addRow(new Object[]{cin,prenom,tel,address,email,num});

        }
    } catch (SQLException ex) {
        Logger.getLogger(listban.class.getName()).log(Level.SEVERE, null, ex);
    }
}
```

interface gestion voiture:



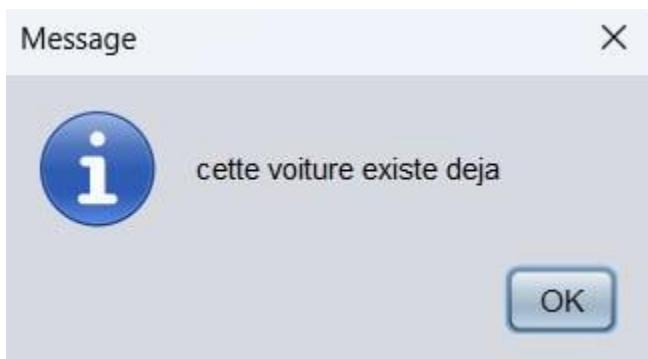
interface ajouté voiture:



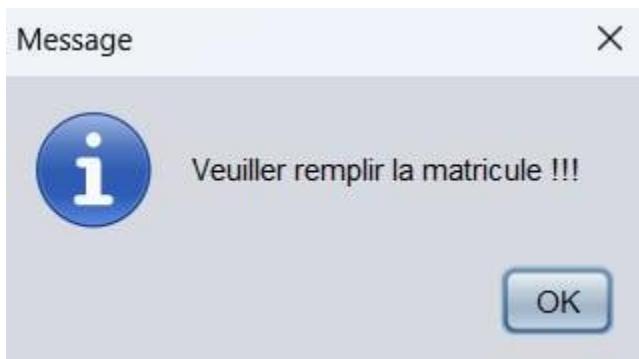
le cas idéal:



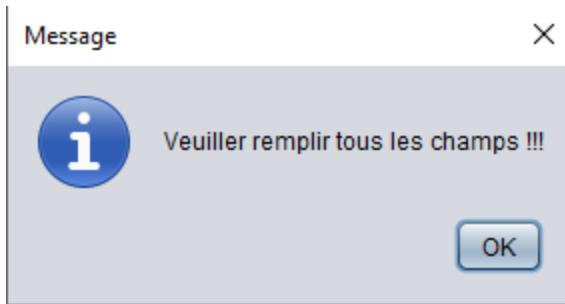
le cas ou la matricule existe:



le cas le champs matricule est vide:



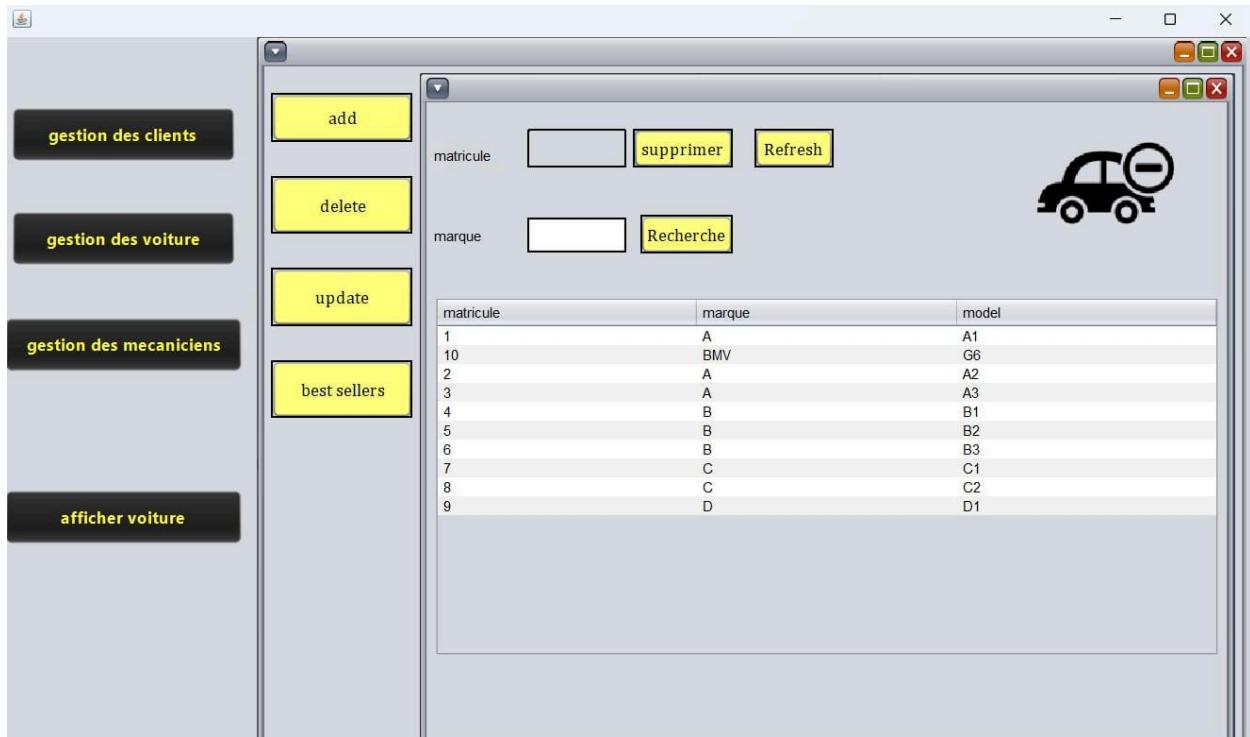
le cas les autres champs sont vides:



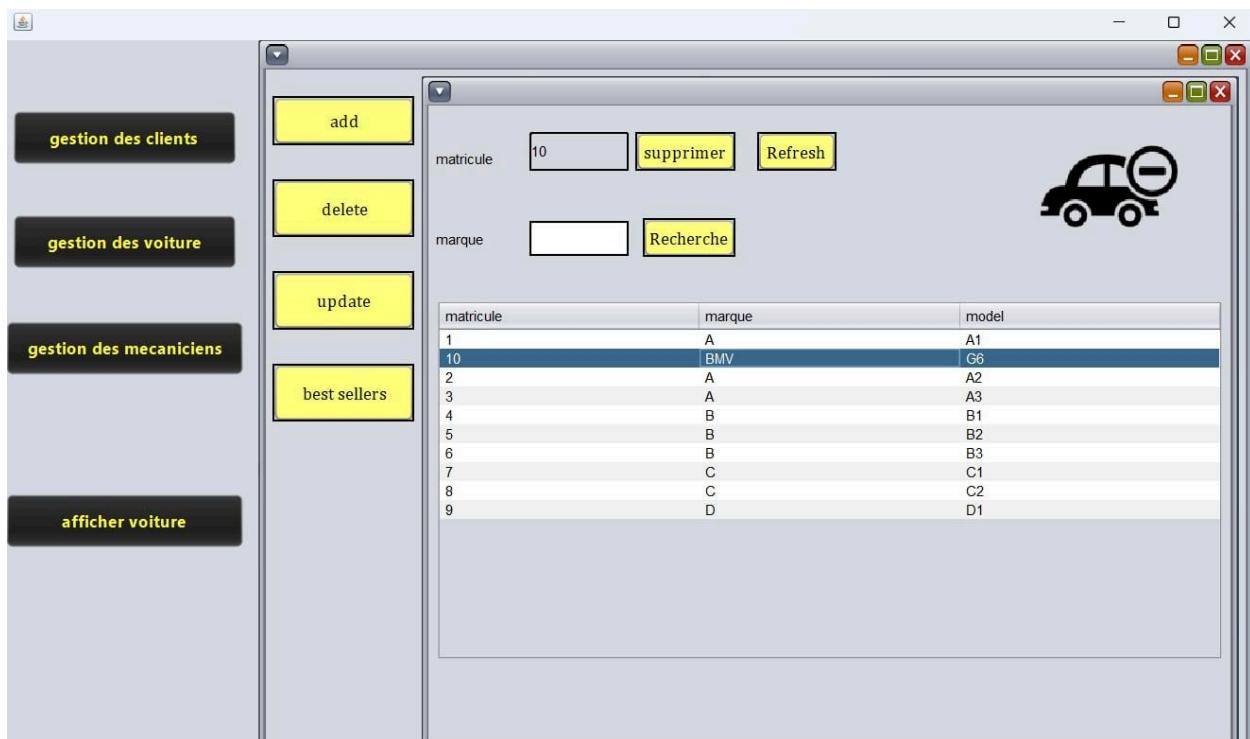
le code:

```
private void confirmActionPerformed(java.awt.event.ActionEvent evt) {
    String matriculeV = matricule.getText();
    String marqueV = marque.getText();
    String modelV = model.getText();
    String etat = "marche";
    String disp = "disponible";
    String prixV = prix.getText();
    if(!verif(matriculeV) ) {
        if(matriculeV.equals("")) {
            JOptionPane.showMessageDialog(null,"Veuillez remplir la matricule !!!");
        }else{
            JOptionPane.showMessageDialog(null,"Veuillez verifier la matricule !!!");
        }
    }else{
        if (marqueV.equals("") | modelV.equals("") | prixV.equals("")){
            JOptionPane.showMessageDialog(null,"Veuillez remplir tous les champs !!!");
        }else{
            if(!isNumeric(prixV))
            {
                JOptionPane.showMessageDialog(null,"Veuillez entrer un prix numeric!!!");
            }else{
                Connecteur connect=new Connecteur();
                Connection con;
                con =connect.connecttodb();
                try {
                    Statement stmt = con.createStatement();
                    String query="SELECT * FROM voiture where matricule ='"+matriculeV+"'";
                    ResultSet rs = stmt.executeQuery(query);
                    if(rs.next() ){
                        JOptionPane.showMessageDialog(null, "THIS CAR ALRAEDY EXISTS");
                    }else{
                        String query2 = "INSERT INTO voiture VALUES (?,?,?,?,?,?)";
                        try (PreparedStatement stml = con.prepareStatement(query2)) {
                            stml.setString(1, matriculeV);
                            stml.setString(2, marqueV);
                            stml.setString(3, modelV);
                            stml.setString(4, etat);
                            stml.setString(5, disp);
                            stml.setString(6, prixV);
                            stml.setString(7, "0");
                            stml.executeUpdate();
                            System.out.println("Data inserted successfully into car table.");
                            JOptionPane.showMessageDialog(null," CAR ADDED !!!");
                        } catch (SQLException e) {
                            System.out.println("Error inserting data into voiture table: " + e.getMessage());
                        }
                    }
                } catch (SQLException ex) {
                    Logger.getLogger(advoiture.class.getName()).log(Level.SEVERE, null, ex);
                }}}
```

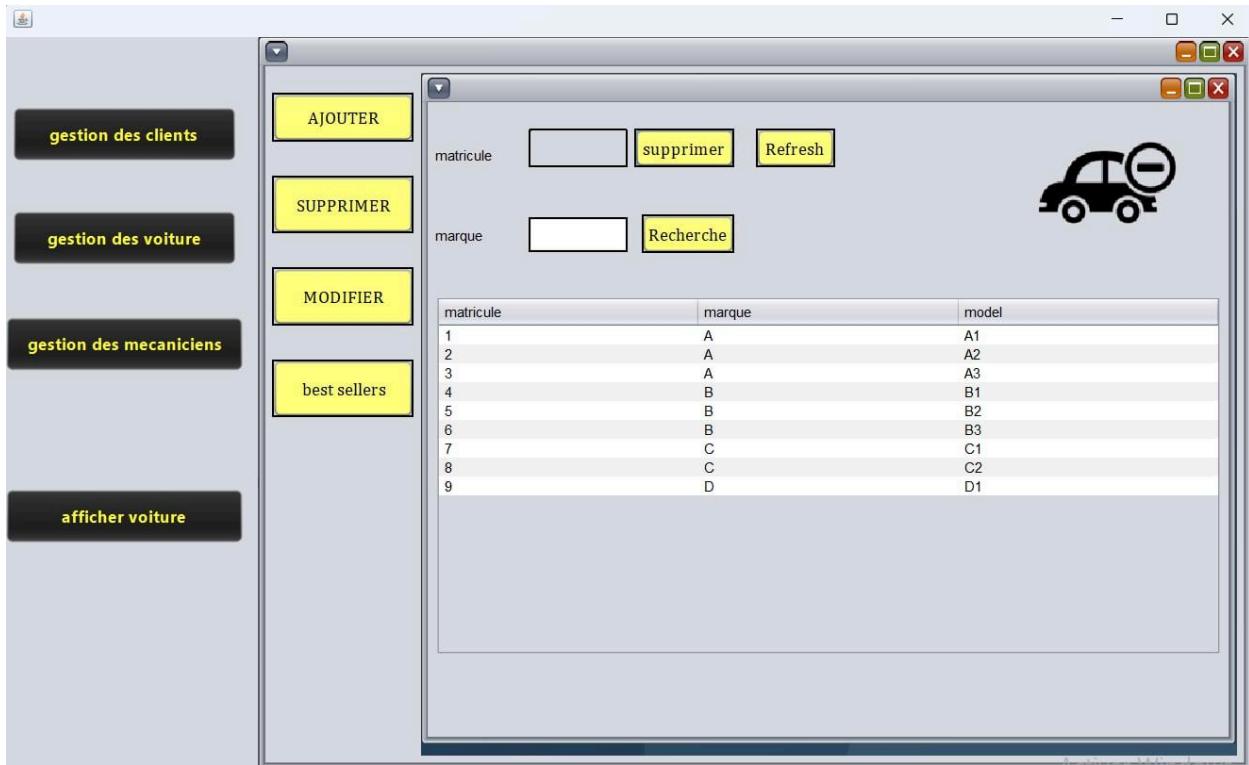
interface supprimer voiture:



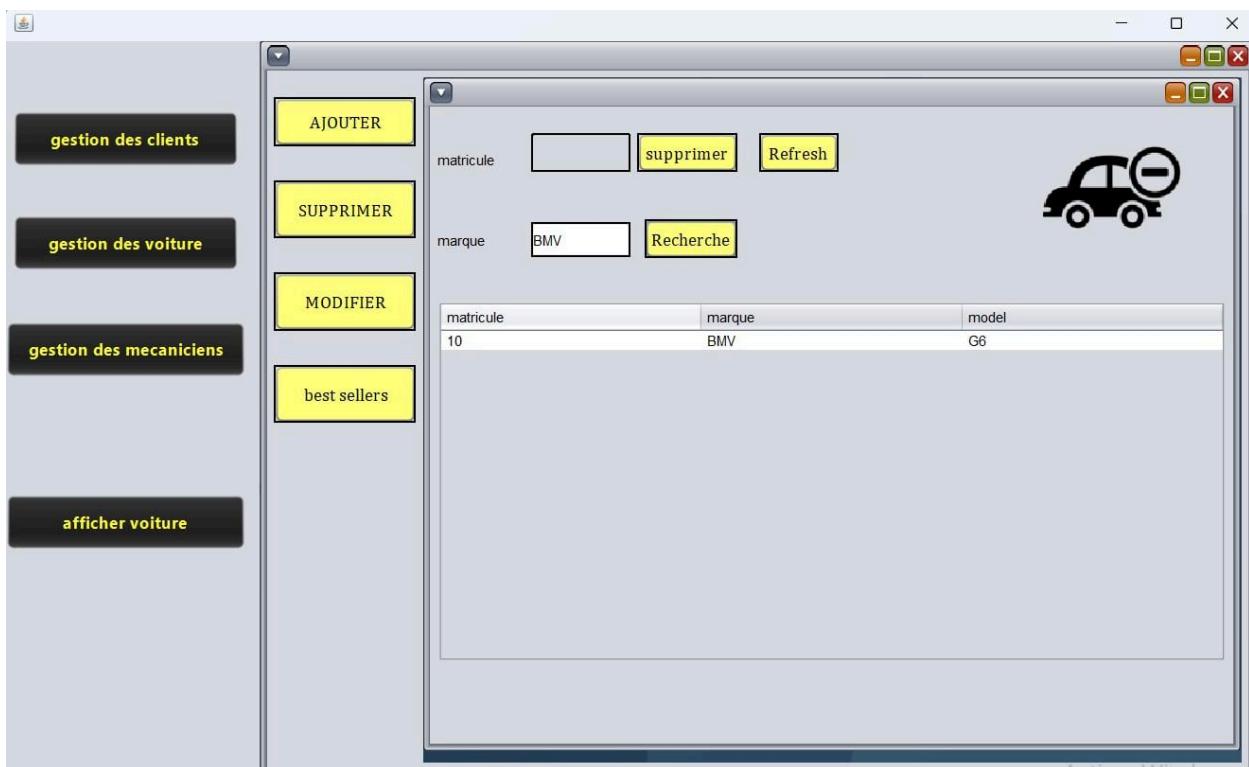
sélectionné une voiture rempli automatiquement le champ matricule



bouton supprimer



bouton rechercher



le code:

```
private void rechercheActionPerformed(java.awt.event.ActionEvent evt) {
    Connecteur connect=new Connecteur();
    con = connect.connecttodb();
    String marque = name_field.getText();
    String query = "SELECT * FROM voiture WHERE marque = '"+marque+"'";
    DefaultTableModel tablemodel =(DefaultTableModel) table.getModel();
    tablemodel.setRowCount(0);
    try {
        Statement stmt = con.createStatement();
        ResultSet res = stmt.executeQuery(query);

        while(res.next()){
            String matricule = res.getString("matricule");
            marque = res.getString("marque");
            String model = res.getString("model");

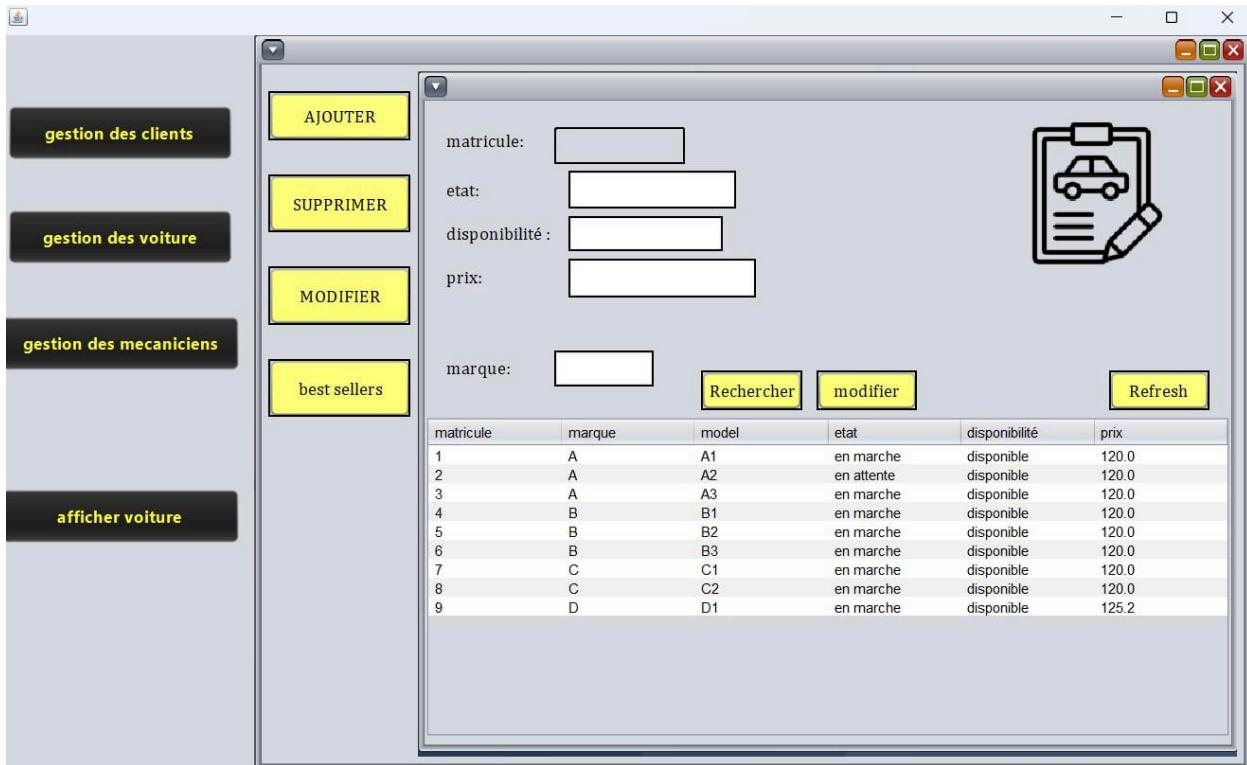
            tablemodel.addRow(new Object[]{matricule,marque,model});
        }
    } catch (SQLException ex) {
        Logger.getLogger(updateclient.class.getName()).log(Level.SEVERE, null, ex);
    }
}
```

```
private void deleteActionPerformed(java.awt.event.ActionEvent evt) {
    String matricule = matricule_field.getText();
    String query=" DELETE FROM voiture where matricule='"+matricule+"'";
    Statement stmt;

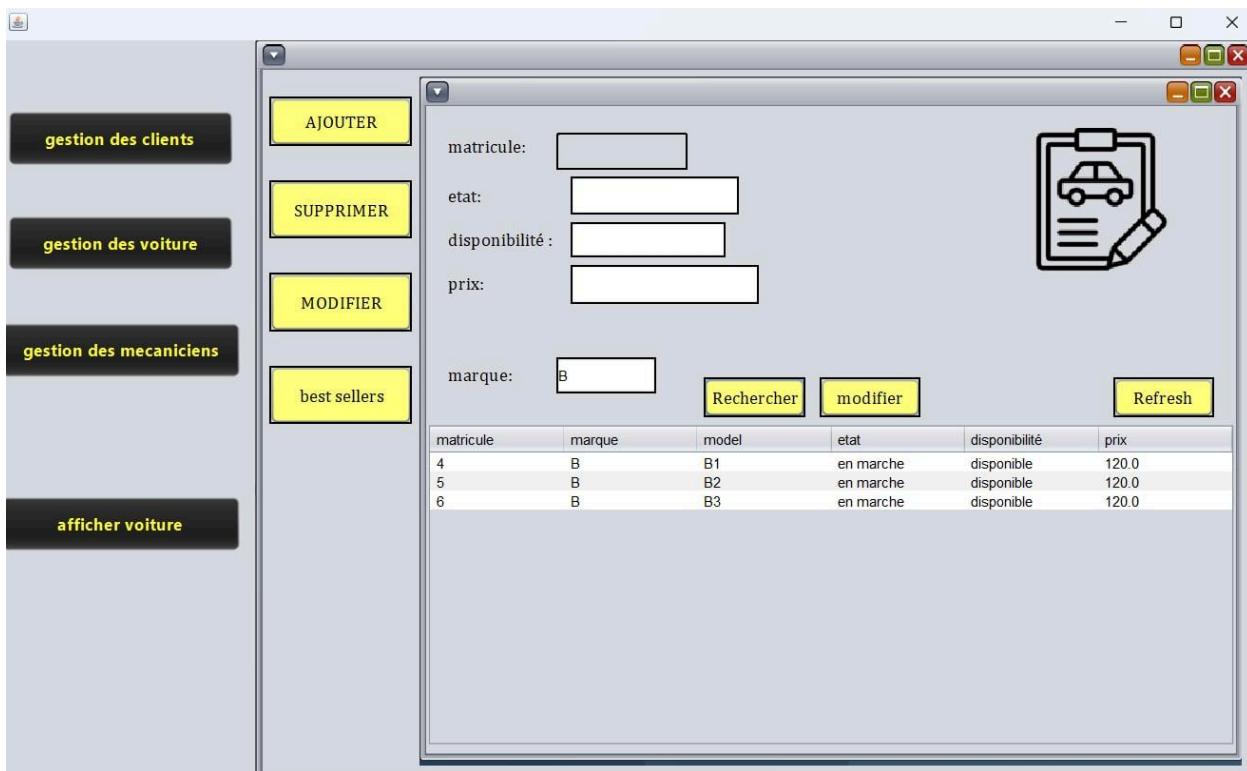
    try
    {
        stmt = con.createStatement();
        stmt.execute(query);
    }
    catch (SQLException ex)
    {
        Logger.getLogger(updateclient.class.getName()).log(Level.SEVERE, null, ex);
    }

    matricule_field.setText("");
    load();
}
```

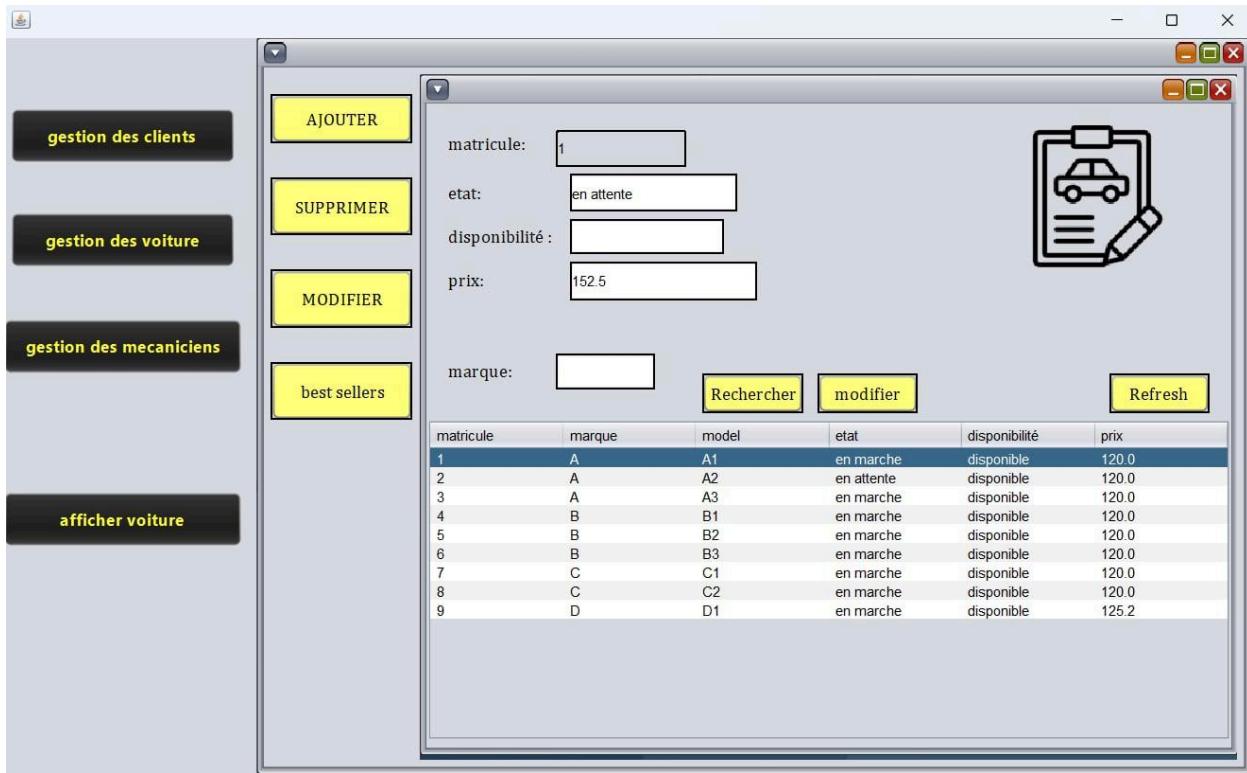
interface modifier voiture



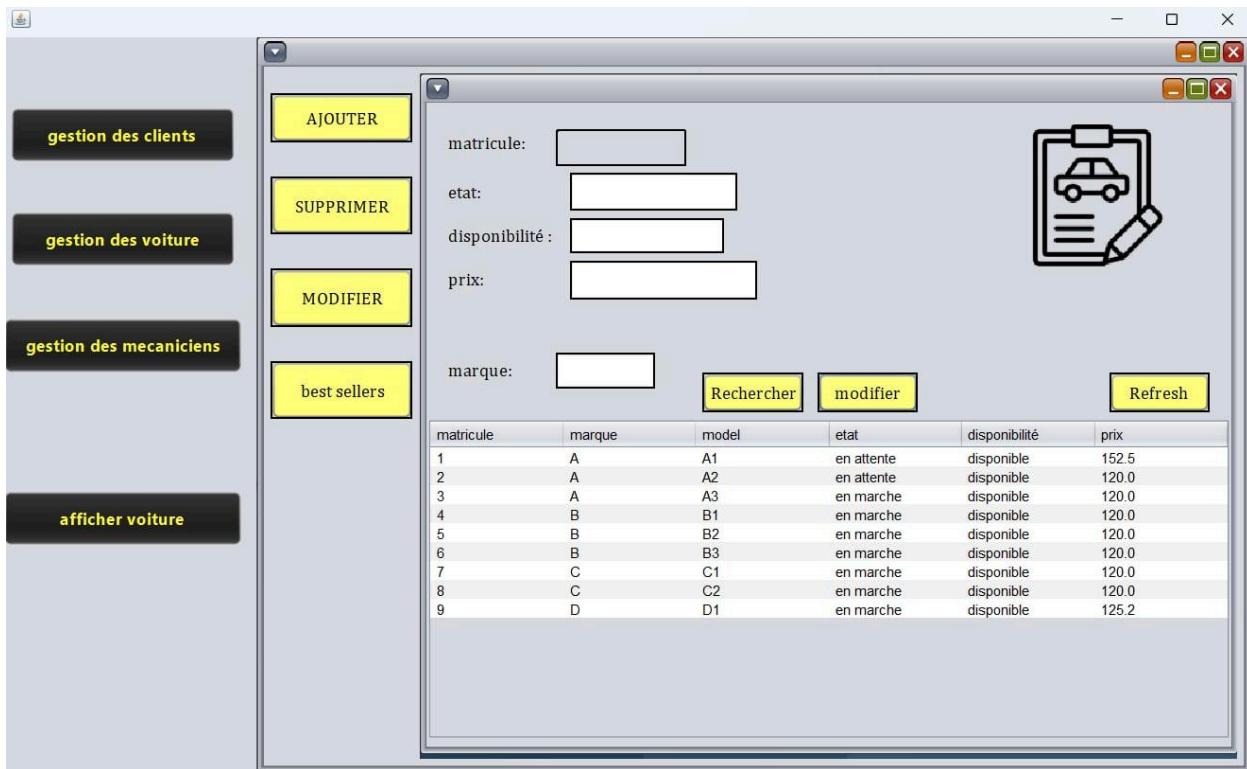
le bouton recherche



sélectionnée voiture



le bouton modifier:



code:

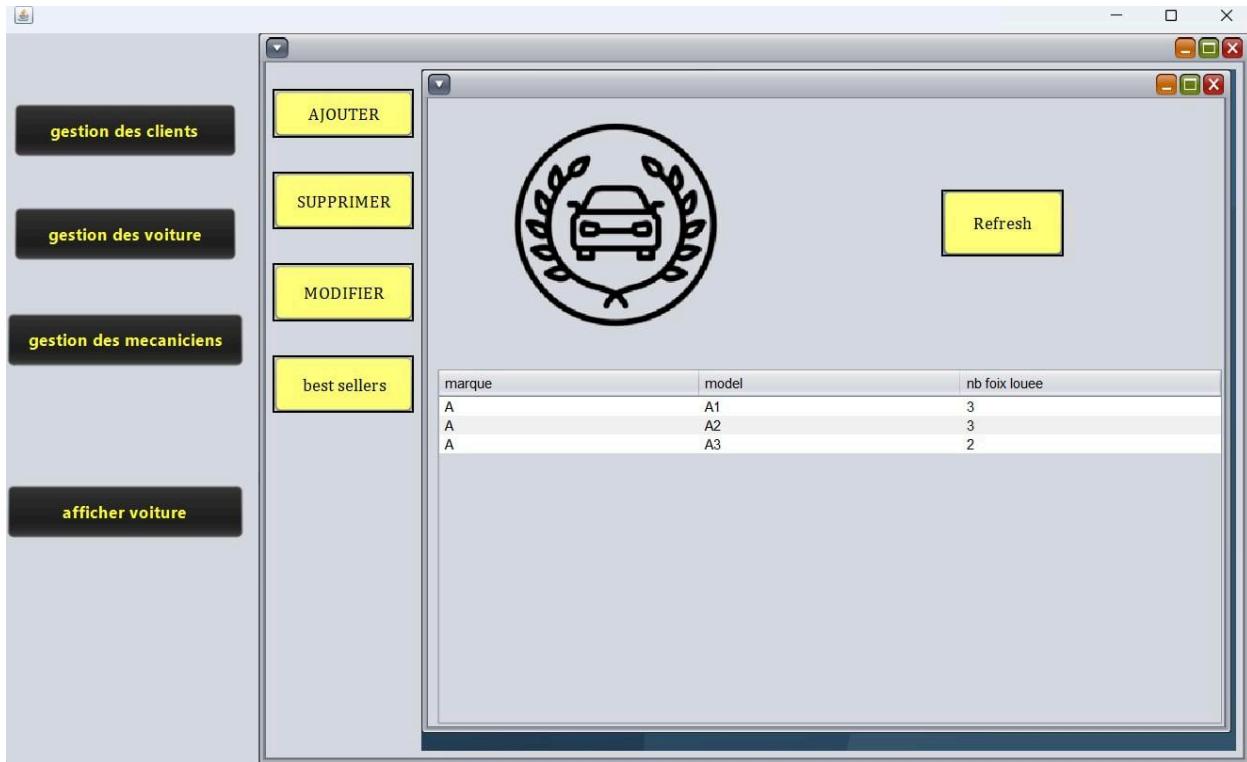
```
private void rechercheActionPerformed(java.awt.event.ActionEvent evt) {  
    Connecteur connect=new Connecteur();  
    con = connect.connecttodb();  
    String marquev = marque.getText();  
    String query = "SELECT * FROM voiture WHERE prenom = '"+marquev+"'";  
    DefaultTableModel tablemodel =(DefaultTableModel) table.getModel();  
    tablemodel.setRowCount(0);  
    try {  
        Statement stmt = con.createStatement();  
        ResultSet res = stmt.executeQuery(query);  
  
        while(res.next()){  
            String matriculev = res.getString("matricule");  
            marquev = res.getString("marque");  
            String modelv = res.getString("model");  
            String etatv = res.getString("etat");  
            String disponibilityv =res.getString("diponibilite");  
            String prixv =res.getString("prix");  
            tablemodel.addRow(new Object[]{matriculev,marquev,modelv,etatv,disponibilityv,prixv});  
        }  
    } catch (SQLException ex) {  
        Logger.getLogger(updatevoiture.class.getName()).log(Level.SEVERE, null, ex);  
    }  
}
```

```

    private void editActionPerformed(java.awt.event.ActionEvent evt) {
        String matriculev =matricule.getText();
        String etatv =etat.getText();
        String disponibilityv =disponibility.getText();
        String prixv =prix.getText();
        if(etatv.equals("") && disponibilityv.equals("") && prixv.equals(""))
        {
            JOptionPane.showMessageDialog(null, "all the fields are empty!", "Error", JOptionPane.ERROR_MESSAGE);
        }else{
            if(!(etatv.equals(""))){
                String query=" UPDATE client SET tel='"+etatv+'' WHERE CIN='"+matriculev+"'";
                Statement stmt;
                try
                {
                    stmt = con.createStatement();
                    stmt.execute(query);
                }
                catch (SQLException ex)
                {
                    Logger.getLogger(updatevoiture.class.getName()).log(Level.SEVERE, null, ex);
                }
            }
            if(!disponibilityv.equals("")){
                String queryl=" UPDATE client SET addresse='"+disponibilityv+"' WHERE CIN='"+matriculev+"'";
                Statement stmtl;
                try
                {
                    stmtl = con.createStatement();
                    stmtl.execute(queryl);
                }
                catch (SQLException ex)
                {
                    Logger.getLogger(updatevoiture.class.getName()).log(Level.SEVERE, null, ex);
                }
            }
            if(!(prixv.equals(""))){
                String query2=" UPDATE client SET email='"+prixv+"'" WHERE CIN='"+matriculev+"'";
                Statement stmt2;
                try
                {
                    stmt2 = con.createStatement();
                    stmt2.execute(query2);
                }
                catch (SQLException ex)
                {
                    Logger.getLogger(updatevoiture.class.getName()).log(Level.SEVERE, null, ex);
                }
            }
            JOptionPane.showMessageDialog(null,"UPDATED CLIENT");
            matricule.setText("");
        }
    }

```

interface best sellers:



le code:

```

public void load(){
    Connecteur connect=new Connecteur();

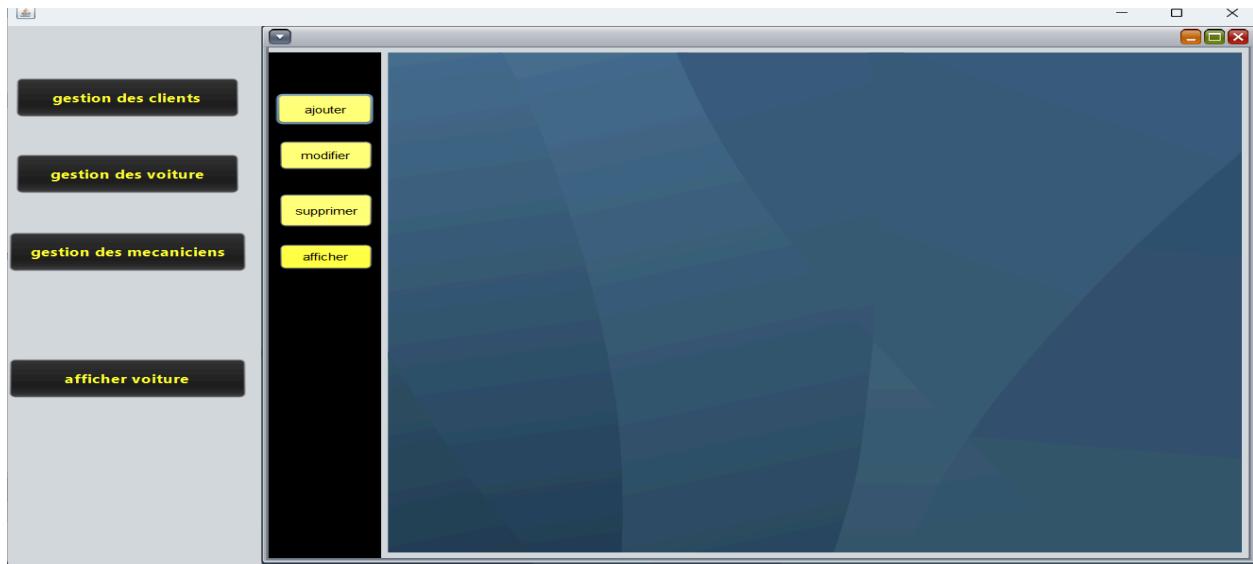
    con = connect.connecttodb();
    String query = "SELECT marque, model, sum(nb_fois_louee) as summ FROM voiture group by marque, model order by sum(nb_fois_louee) desc LIMIT 3";
    DefaultTableModel tablemodel =(DefaultTableModel) table.getModel();
    tablemodel.setRowCount(0);
    try {
        Statement stmt = con.createStatement();
        ResultSet res = stmt.executeQuery(query);

        while(res.next()){
            String nb = res.getString("summ");
            String marque = res.getString("marque");
            String model = res.getString("model");
            tablemodel.addRow(new Object[]{marque,model,nb});

        }
    } catch (SQLException ex) {
        Logger.getLogger(listban.class.getName()).log(Level.SEVERE, null, ex);
    }
}

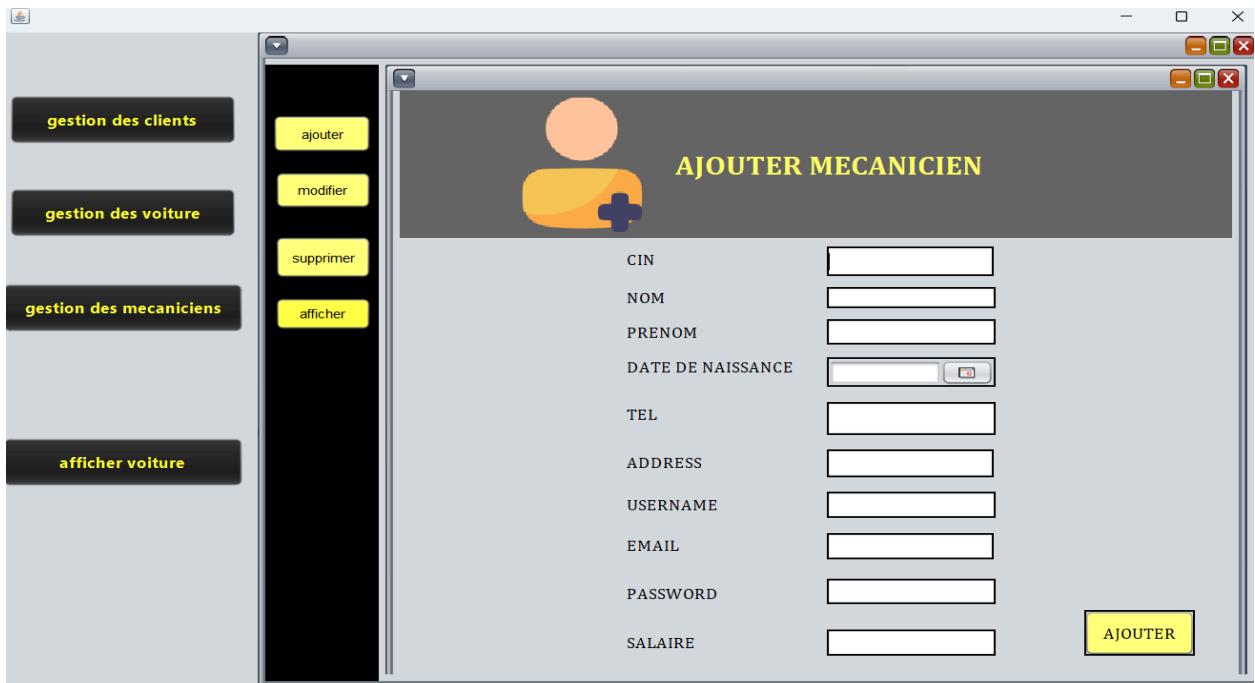
```

interface gestion mécanicien:



interface ajouter mécanicien:

interface ajouter mécanicien avec les contrôles sur les données saisies(email invalide mechanicien déjà existant)



code:

```

private void confirmActionPerformed(java.awt.event.ActionEvent evt) {
    String cin_user = cin.getText();
    String nom_user = nom.getText();
    String username_user = username.getText();
    String prenom_user = prenom.getText();
    String tel_user = tel.getText();
    String address_user = address.getText();
    String email_user = email.getText();
    char[] passwordChars = password.getPassword();
    String password_user = new String(passwordChars);
    Date date_user = date.getDate();
    String sal_user = sal_field.getText();

    Mecanicien mec = new Mecanicien(cin:cin_user, nom:nom_user, prenom:prenom_user, email:email_user, date_de_naissance:date_user,username:username_user, tel:tel_user,address:address_user,
    mot_de_passe:password_user,salaire:Double.parseDouble(s: sal_user));

    if(!verif(cin:cin_user) ) {
        if(cin_user.equals(anObject:"")){
            JOptionPane.showMessageDialog(parentComponent:Null,message:"Veuillez remplir le CIN !!!");
        }else{
            JOptionPane.showMessageDialog(parentComponent:Null,message:"Veuillez verifier le CIN !!!");
        }
    }else{
        if (nom_user.equals(anObject:"") | prenom_user.equals(anObject:"") | tel_user.equals(anObject:"") | date_user==null | address_user.equals(anObject:"")){
            JOptionPane.showMessageDialog(parentComponent:Null,message:"Veuillez remplir tous les champs !!!");
        }else{
            java.sql.Date sqlDate1 = new java.sql.Date(date:date_user.getTime());
            String date1=sqlDate1.toString();
            if (!isValidEmailAddress(email:email_user)) {
                JOptionPane.showMessageDialog(parentComponent:Null,message:"L'email est invalide !!!");
            }else{
                if (password_user.equals(anObject:"")){
                    JOptionPane.showMessageDialog(parentComponent:Null,message:"Veuillez remplir le mot de passe !!!");
                }else{
                    Connecteur connect=new Connecteur();
                    Connection con;
                    con =connect.connecttodb();
                    try {
                        Statement stmt = con.createStatement();
                        String query="SELECT * FROM mechanicien where cin ='"+cin_user+"'";
                        ResultSet rs=stmt.executeQuery(string: query);

```

```

try {
    Statement stmt = con.createStatement();
    String query="SELECT * FROM mechanicien where cin ='"+cin_user+"'";
    ResultSet rs=stmt.executeQuery(string: query);
    if(rs.next() )
    {
        JOptionPane.showMessageDialog(parentComponent:null,message:"CIN DEJA UTILISE !!!");
    }
    else{

        String query2 = "INSERT INTO mechanicien VALUES (?,?,?,?,?,?,?,?,?,?)";

        try (PreparedStatement stmt1 = con.prepareStatement(string: query2)) {
            stmt1.setString(1, string: cin_user);
            stmt1.setString(2, string: nom_user);
            stmt1.setString(3, string: prenom_user);
            stmt1.setString(4, string: username_user);
            stmt1.setDate(5, date: sqlDate1); // Assuming sqlDate1 is your java.sql.Date object
            stmt1.setString(6, string: email_user);
            stmt1.setString(7, string: tel_user);
            stmt1.setString(8, string: address_user);
            stmt1.setString(9, string: password_user);
            stmt1.setString(10, string: sal_user);

            stmt1.executeUpdate();
            System.out.println("Data inserted successfully into mechanicien table.");
            JOptionPane.showMessageDialog(parentComponent:null,message:" MECHANICIEN AJOUTE !!!");

            this.nom.setText(string: "");
            this.prenom.setText(string: "");
            this.date.setDate(date: null);
            this.tel.setText(string: "");
            this.address.setText(string: "");
            this.username.setText(string: "");
            this.email.setText(string: "");
            this.password.setText(string: "");
            this.sal_field.setText(string: "");

        } catch (SQLException e) {
            System.out.println("Error inserting data into mechanicien table: " + e.getMessage());
        }
    }
}

catch (SQLException ex)
{
    Logger.getLogger(name: addmec.class.getName()).log(level: Level.SEVERE, msg: null, thrown: ex);
}

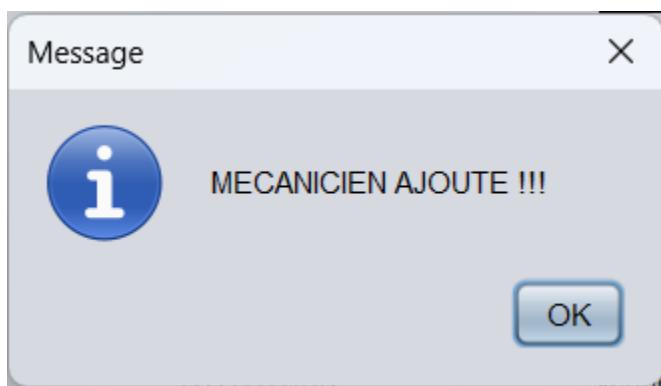
```

cas d'ajout:

AJOUTER MECANICIEN

CIN	44444444
NOM	adem
PRENOM	madyouni
DATE DE NAISSANCE	12 avr. 2024 <input style="width: 20px; height: 20px;" type="button" value="..."/>
TEL	14587452
ADDRESS	jandouba
USERNAME	adem
EMAIL	adem@gmail.com
PASSWORD	***
SALAIRE	1254

AJOUTER

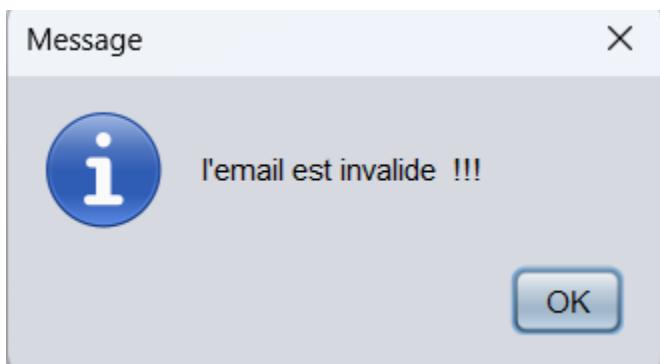


cas d'email invalide:

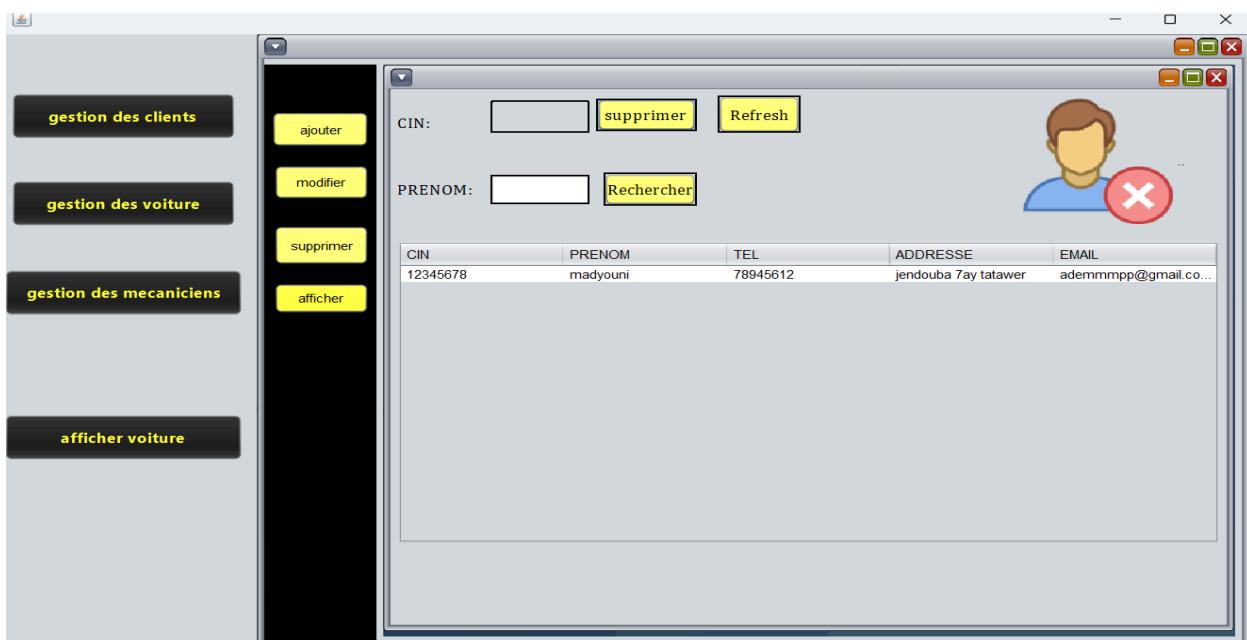
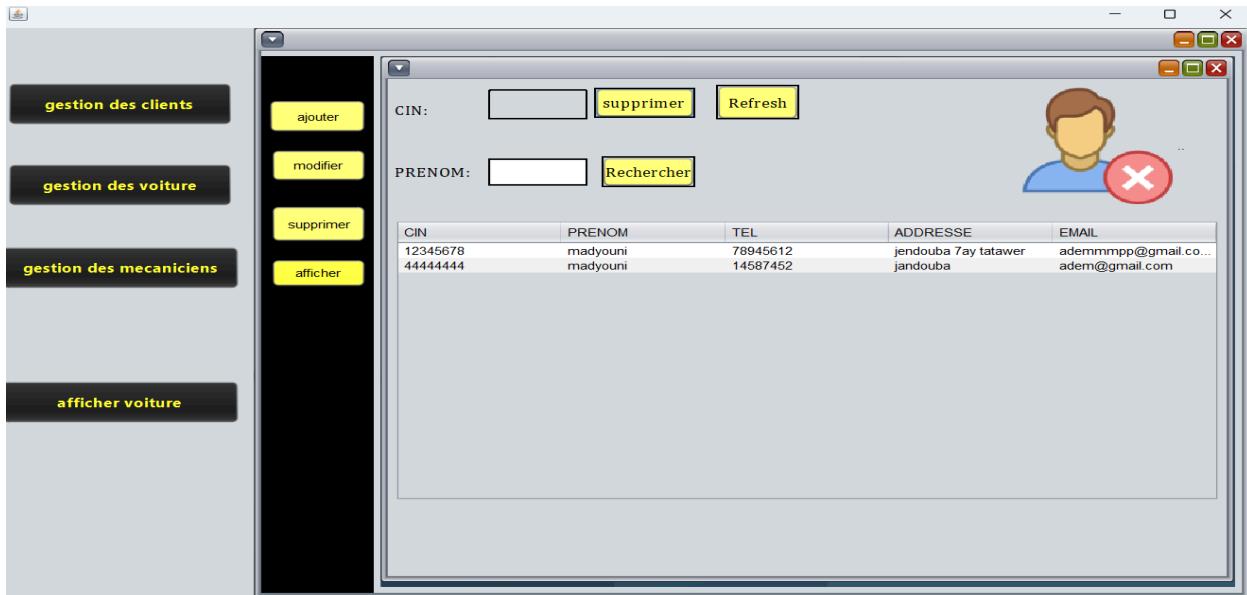
AJOUTER MECANICIEN

CIN	4444444
NOM	adem
PRENOM	adem
DATE DE NAISSANCE	5 avr. 2024
TEL	154251
ADDRESS	tunis
USERNAME	adem
EMAIL	dfz.com
PASSWORD	****
SALAIRE	1200

AJOUTER



interface supprimer un mécanicien:



code:

code de fonction load qui affiche la liste des mécanicien

```
private Connection con ;
public void load(){
    Connecteur connect=new Connecteur();

    con = connect.connecttodb();
    String query = "SELECT * FROM mechanicien";
    DefaultTableModel tablemodel =(DefaultTableModel) table.getModel();
    tablemodel.setRowCount(rowCount: 0);
    try {
        Statement stmt = con.createStatement();
        ResultSet res = stmt.executeQuery(string: query);

        while(res.next()){
            String cin = res.getString(string: "CIN");
            String prenom = res.getString(string: "prenom");
            String address = res.getString(string: "adresse");
            String tel = res.getString(string: "tel");
            String email =res.getString(string: "email");
            tablemodel.addRow(new Object[]{cin,prenom,tel,address,email});

        }
    } catch (SQLException ex) {
        Logger.getLogger(name: listban.class.getName()).log(level: Level.SEVERE, msg:null, thrown: ex);
    }
    this.cin_field.setText(text: "");
    this.name_field.setText(: "");

}
```

code de fonction recherche:

```
private void rechercheActionPerformed(java.awt.event.ActionEvent evt) {
    Connecteur connect=new Connecteur();
    con = connect.connecttodb();
    String prenom = name_field.getText();
    String query = "SELECT * FROM mechanicien WHERE prenom = '"+prenom+"'";
    DefaultTableModel tablemodel =(DefaultTableModel) table.getModel();
    tablemodel.setRowCount(rowCount: 0);
    try {
        Statement stmt = con.createStatement();
        ResultSet res = stmt.executeQuery(string: query);

        while(res.next()){
            String cin = res.getString(string: "cin");
            prenom = res.getString(string: "prenom");
            String tel = res.getString(string: "tel");
            String address = res.getString(string: "adresse");
            String email =res.getString(string: "email");
            tablemodel.addRow(new Object[]{cin,prenom,tel,address,email});

        }
    } catch (SQLException ex) {
        Logger.getLogger(name: updateclient.class.getName()).log(level: Level.SEVERE, msg:null, thrown: ex);
    }
}
```

code de fonction delete pour effacer le mécanicien de la table mécanicien:

```

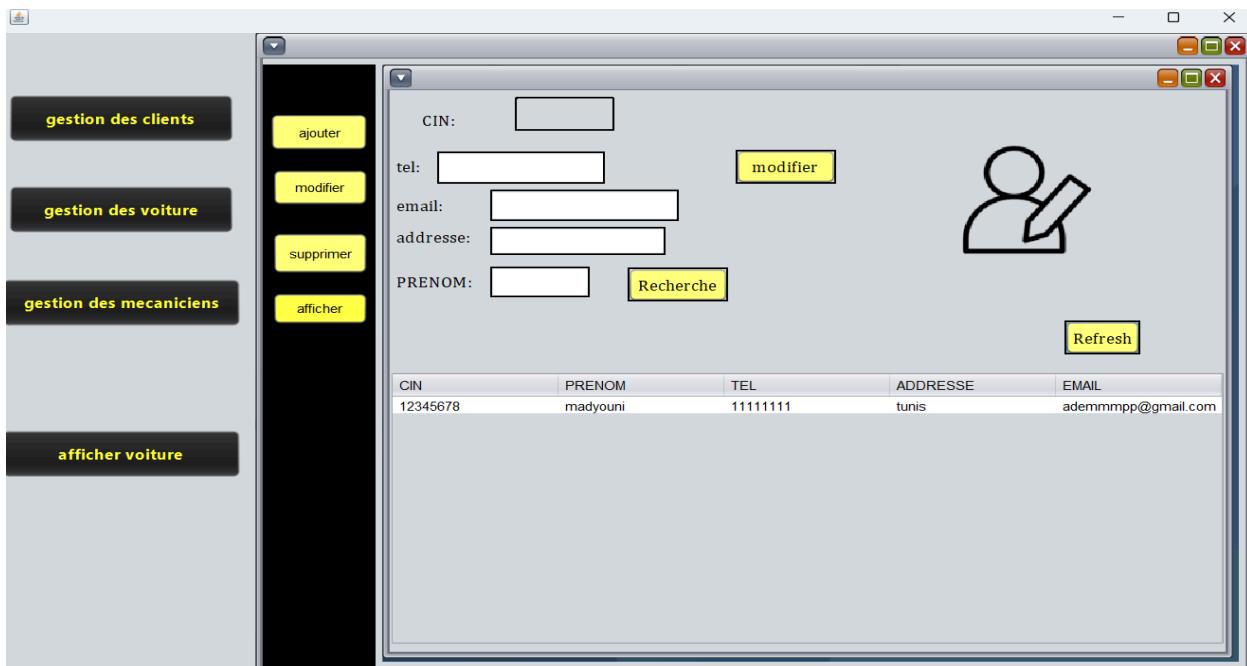
private void deleteActionPerformed(java.awt.event.ActionEvent evt) {
    String cin_user =cin_field.getText();
    String query=" DELETE FROM mechanicien where CIN='"+cin_user+"'";
    Statement stmt;

    try
    {
        stmt = con.createStatement();
        stmt.execute(string: query);
    }
    catch (SQLException ex)
    {
        Logger.getLogger(name:updateclient.class.getName()).log(level: Level.SEVERE, msg:null, thrown: ex);
    }

    cin_field.setText(text: "");
    load();
}

```

interface modifier mécanicien:



gestion des clients

ajouter

modifier

supprimer

afficher

gestion des voiture

gestion des mecaniciens

afficher voiture

CIN: 12345678

tel: 99999999

email: adem@gmail.com

adresse: jendouba

PRENOM: madyouni

modifier

Recherche

Refresh

CIN	PRENOM	TEL	ADRESSE	EMAIL
12345678	madyouni	11111111	tunis	ademmmpp@gmail.com



Message



MECANICIEN MIS A JOUR

OK

gestion des clients

ajouter

modifier

supprimer

afficher

gestion des voiture

gestion des mecaniciens

afficher voiture

CIN:

tel:

email:

adresse:

PRENOM: madyouni

modifier

Recherche

Refresh

CIN	PRENOM	TEL	ADRESSE	EMAIL
12345678	madyouni	99999999	jendouba	adem@gmail.com



code:

fonction du bouton recherche

```
private void rechercheActionPerformed(java.awt.event.ActionEvent evt) {
    Connecteur connect=new Connecteur();
    con = connect.connecttodb();
    String prenom = name_field.getText();
    String query = "SELECT * FROM mechanicien WHERE prenom = '"+prenom+"'";
    DefaultTableModel tablemodel =(DefaultTableModel) table.getModel();
    tablemodel.setRowCount(0);
    try {
        Statement stmt = con.createStatement();
        ResultSet res = stmt.executeQuery(string: query);

        while(res.next()){
            String cin = res.getString(string: "cin");
            prenom = res.getString(string: "prenom");
            String tel = res.getString(string: "tel");
            String address = res.getString(string: "adresse");
            String email =res.getString(string: "email");
            tablemodel.addRow(new Object[]{cin,prenom,tel,address,email});

        }
    } catch (SQLException ex) {
        Logger.getLogger(name: updateclient.class.getName()).log(level: Level.SEVERE, msg:null, thrown: ex);
    }
}
```

la requête pour modifier le mécanicien avec les tests sur la validité de l'email

```
private void editActionPerformed(java.awt.event.ActionEvent evt) {
    String cin_user =cin_field.getText();
    String telp =tel.getText();
    String address =adresse.getText();
    String mail =email.getText();
    if(telp.equals(anObject: "") && address.equals(anObject: "") && mail.equals(anObject: ""))
    {
        JOptionPane.showMessageDialog(parentComponent:null, message:"tous les champs sont vides", title: "Error", messageType:JOptionPane.ERROR_MESSAGE);
    }
    else
    {
        if(!(telp.equals(anObject: "")))
        {
            String query=" UPDATE mechanicien SET tel='"+telp+'' WHERE CIN='"+cin_user+"'";
            Statement stmt;

            try
            {
                stmt = con.createStatement();
                stmt.execute(string: query);
            }
            catch (SQLException ex)
            {
                Logger.getLogger(name: updateclient.class.getName()).log(level: Level.SEVERE, msg:null, thrown: ex);
            }
        }
        if(!(address.equals(anObject: "")))
        {
            String query1=" UPDATE mechanicien SET adresse='"+address+"'WHERE CIN='"+cin_user+"'";
            Statement stmt1;
            try
            {
                stmt1 = con.createStatement();
                stmt1.execute(string: query1);
            }
            catch (SQLException ex)
            {
                Logger.getLogger(name: updateclient.class.getName()).log(level: Level.SEVERE, msg:null, thrown: ex);
            }
        }
    }
}
```

```

        if(!mail.equals(anObject))
        {
            if(!isValidEmailAddress(email: mail))
            {
                JOptionPane.showMessageDialog(parentComponent:null, message:"address mail invalide !!!!!!", title: "Error", messageType:JOptionPane.ERROR_MESSAGE);
                if(!address.equals(anObject)||!telp.equals(anObject))
                    JOptionPane.showMessageDialog(parentComponent:null,message:" MECANICIEN MIS A JOUR");
            }
        }
        else
        {
            String query2=" UPDATE mechanicien SET email='"+mail+"' WHERE CIN='"+cin_user+"'";
            Statement stmt2;

            try
            {
                stmt2 = con.createStatement();
                stmt2.execute(string: query2);
            }
            catch (SQLException ex)
            {
                Logger.getLogger(name:updateclient.class.getName()).log(level: Level.SEVERE, msg:null, thrown: ex);
            }
            JOptionPane.showMessageDialog(parentComponent:null,message:" MECANICIEN MIS A JOUR");
        }
    }
    cin_field.setText(text: "");
}

load();
}

```

L'interface afficher liste mécanicien:



code:

```
public void load(){
    Connecteur connect=new Connecteur();

    con = connect.connecttodb();
    String query = "SELECT * FROM mechanicien";
    DefaultTableModel tablemodel =(DefaultTableModel) table.getModel();
    tablemodel.setRowCount(rowCount: 0);
    try {
        Statement stmt = con.createStatement();
        ResultSet res = stmt.executeQuery(string: query);

        while(res.next()){
            String cin = res.getString(string: "CIN");
            String prenom = res.getString(string: "prenom");
            String address = res.getString(string: "adresse");
            String tel = res.getString(string: "tel");
            String email =res.getString(string: "email");
            String sal=res.getString(string: "salaire");
            tablemodel.addRow(new Object[]{cin,prenom,tel,address,email,sal});

        }
    } catch (SQLException ex) {
        Logger.getLogger(name: listban.class.getName()).log(level: Level.SEVERE, msg:null, thrown: ex);
    }
}
```

interface afficher voiture:

cette interface permet d' afficher la liste de s voiture et aussi de selectionner selon le critere de l'etat.

afficher toutes les voiture:



afficher seule qui sont en panne en appuyant le bouton en panne:



afficher seules qui sont en marche en appuyant le bouton en marche:



afficher seules qui sont en attente en appuyant le bouton en attente:



afficher seules qui sont en disponibles en appuyant le bouton disponible:



code:

afficher toute les voiture avec requete SQL select * from voiture.

```

private Connection con ;
public void load(){
    Connecteur connect=new Connecteur();

    con = connect.connecttodb();
    String query = "SELECT * FROM voiture ";
    DefaultTableModel tablemodel =(DefaultTableModel) table.getModel();
    tablemodel.setRowCount(rowCount: 0);
    try {
        Statement stmt = con.createStatement();
        ResultSet res = stmt.executeQuery(string: query);

        while(res.next()){
            String matricule = res.getString(string: "matricule");
            String marque = res.getString(string: "marque");
            String model = res.getString(string: "model");
            String etat = res.getString(string: "etat");
            String dispo = res.getString(string: "disponibilite");
            tablemodel.addRow(new Object[]{matricule,marque,model,etat,dispo});

        }
    } catch (SQLException ex) {
        Logger.getLogger(name: affichervoiture.class.getName()).log(level: Level.SEVERE, msg:null, thrown: ex);
    }
}
  
```

afficher les voiture en marche select from voiture where etat= en marche.

```

private void marchActionPerformed(java.awt.event.ActionEvent evt) {
    Connecteur connect=new Connecteur();

    con = connect.connecttodb();
    String query = "SELECT * FROM voiture where etat='"+ "en marche'";
    DefaultTableModel tablemodel =(DefaultTableModel) table.getModel();
    tablemodel.setRowCount(rowCount: 0);
    try {
        Statement stmt = con.createStatement();
        ResultSet res = stmt.executeQuery(string: query);

        while(res.next()){
            String matricule = res.getString(string: "matricule");
            String marque = res.getString(string: "marque");
            String model = res.getString(string: "model");
            String etat = res.getString(string: "etat");
            String dispo = res.getString(string: "disponibilite");
            tablemodel.addRow(new Object[]{matricule,marque,model,etat,dispo});
        }
    } catch (SQLException ex) {
        Logger.getLogger(name: affichervoiture.class.getName()).log(level: Level.SEVERE, msg:null, thrown: ex);
    }
}

```

afficher les voiture en attente select from voiture where etat= en attente.

```

private void attActionPerformed(java.awt.event.ActionEvent evt) {
    Connecteur connect=new Connecteur();

    con = connect.connecttodb();
    String query = "SELECT * FROM voiture where etat='"+ "en attente'";
    DefaultTableModel tablemodel =(DefaultTableModel) table.getModel();
    tablemodel.setRowCount(rowCount: 0);
    try {
        Statement stmt = con.createStatement();
        ResultSet res = stmt.executeQuery(string: query);

        while(res.next()){
            String matricule = res.getString(string: "matricule");
            String marque = res.getString(string: "marque");
            String model = res.getString(string: "model");
            String etat = res.getString(string: "etat");
            String dispo = res.getString(string: "disponibilite");
            tablemodel.addRow(new Object[]{matricule,marque,model,etat,dispo});
        }
    } catch (SQLException ex) {
        Logger.getLogger(name: affichervoiture.class.getName()).log(level: Level.SEVERE, msg:null, thrown: ex);
    }
}

```

afficher les voiture en panne select from voiture where etat= en panne.

```

private void panneActionPerformed(java.awt.event.ActionEvent evt) {
    Connecteur connect=new Connecteur();

    con = connect.connecttodb();
    String query = "SELECT * FROM voiture where etat='"+ "en panne'";
    DefaultTableModel tablemodel =(DefaultTableModel) table.getModel();
    tablemodel.setRowCount(rowCount: 0);
    try {
        Statement stmt = con.createStatement();
        ResultSet res = stmt.executeQuery(string: query);

        while(res.next()){
            String matricule = res.getString(string: "matricule");
            String marque = res.getString(string: "marque");
            String model = res.getString(string: "model");
            String etat = res.getString(string: "etat");
            String dispo = res.getString(string: "disponibilite");
            tablemodel.addRow(new Object[]{matricule,marque,model,etat,dispo});

        }
    } catch (SQLException ex) {
        Logger.getLogger(name: affichervoiture.class.getName()).log(level: Level.SEVERE, msg:null, thrown: ex);
    }
}

```

afficher les voiture disponibles select from voiture where disponibilité= disponible.

```

private void dispoActionPerformed(java.awt.event.ActionEvent evt) {
    Connecteur connect=new Connecteur();

    con = connect.connecttodb();
    String query = "SELECT * FROM voiture where disponibilite='"+ "disponible'";
    DefaultTableModel tablemodel =(DefaultTableModel) table.getModel();
    tablemodel.setRowCount(rowCount: 0);
    try {
        Statement stmt = con.createStatement();
        ResultSet res = stmt.executeQuery(string: query);

        while(res.next()){
            String matricule = res.getString(string: "matricule");
            String marque = res.getString(string: "marque");
            String model = res.getString(string: "model");
            String etat = res.getString(string: "etat");
            String dispo = res.getString(string: "disponibilite");
            tablemodel.addRow(new Object[]{matricule,marque,model,etat,dispo});

        }
    } catch (SQLException ex) {
        Logger.getLogger(name: affichervoiture.class.getName()).log(level: Level.SEVERE, msg:null, thrown: ex);
    }
}

```

afficher les voitures louées select from voiture where disponibilité= louée.

```
private void loueeActionPerformed(java.awt.event.ActionEvent evt) {
    Connecteur connect=new Connecteur();

    con = connect.connecttodb();
    String query = "SELECT * FROM voiture where disponibilite='"+ "louee'";
    DefaultTableModel tablemodel =(DefaultTableModel) table.getModel();
    tablemodel.setRowCount(rowCount: 0);
    try {
        Statement stmt = con.createStatement();
        ResultSet res = stmt.executeQuery(string: query);

        while(res.next()){
            String matricule = res.getString(string: "matricule");
            String marque = res.getString(string: "marque");
            String model = res.getString(string: "model");
            String etat = res.getString(string: "etat");
            String dispo = res.getString(string: "disponibilite");
            tablemodel.addRow(new Object[]{matricule,marque,model,etat,dispo});

        }
    } catch (SQLException ex) {
        Logger.getLogger(name: affichervoiture.class.getName()).log(level: Level.SEVERE, msg:null, thrown: ex);
    }
}
```

interface mécanicien:

voiture en attente:

The screenshot shows a software interface for managing a fleet of vehicles. On the left, there is a dark sidebar with two yellow buttons: "AFFICHER LISTE PANNE" and "AFFICHER LISTE D'ATTENTE". The main window has a title bar with standard window controls. It contains several buttons: "Matricole : [input field]", "METTRE EN PANNE" (yellow button), "METTRE EN MARCHE" (yellow button), "Marque : [input field]", "RECHERCHE" (yellow button), and "REFRESH" (yellow button). Below these buttons is a table with columns: Matricole, Marque, Model, and Etat. The data in the table is as follows:

Matricole	Marque	Model	Etat
1	A	A1	en attente
2	A	A2	en attente
3	A	A3	en attente
4	B	B1	en attente
5	B	B2	en attente
6	B	B3	en attente
7	C	C1	en attente
8	C	C2	en attente
9	D	D1	en attente

la recherche par matricule:

This screenshot shows the same software interface as the previous one, but with a specific search applied. The "Marque : [input field]" contains the value "A", and the "RECHERCHE" button is highlighted in yellow. The table below shows the results of this search, which are the same three cars from the previous list, all belonging to Marque A.

Matricole	Marque	Model	Etat
1	A	A1	en attente
2	A	A2	en attente
3	A	A3	en attente

sélectionne la voiture:

AFFICHER LISTE PANNE

AFFICHER LISTE D'ATTENTE

Matricule : 1

Marque : A

METTRE EN PANNE

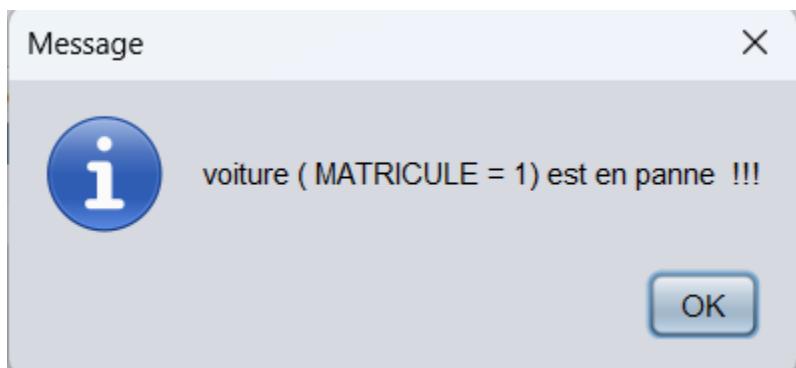
METTRE EN MARCHÉ

RECHERCHE

REFRESH

Matricule	Marque	Model	Etat
1	A	A1	en attente
2	A	A2	en attente
3	A	A3	en attente

mettre l'état du véhicule vers panne:



sélectionne une autre:

AFFICHER LISTE PANNE

AFFICHER LISTE D'ATTENTE

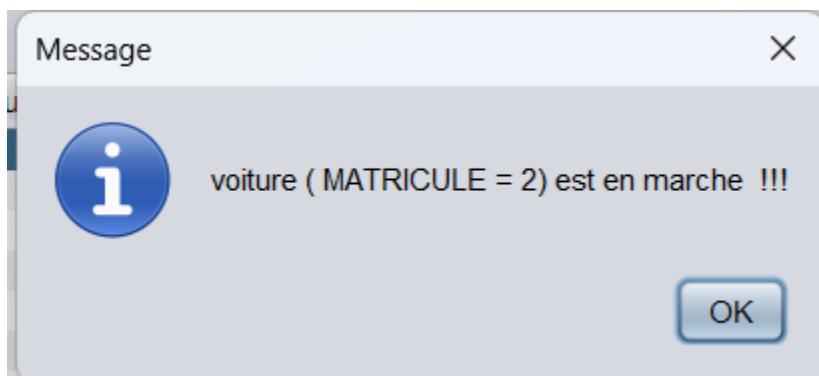
Matricule : 2 METTRE EN PANNE METTRE EN MARCHE

Marque : RECHERCHE

REFRESH

Matricule	Marque	Model	Etat
2	A	A2	en attente
3	A	A3	en attente
4	B	B1	en attente
5	B	B2	en attente
6	B	B3	en attente
7	C	C1	en attente
8	C	C2	en attente
9	D	D1	en attente

mettre l'état du voiture vers marche:



code:

```
public void load() {
    Connecteur connect=new Connecteur();

    con = connect.connecttodb();
    String query = "SELECT * FROM voiture WHERE ETAT ='en attente'";
    DefaultTableModel tablemodel =(DefaultTableModel) table.getModel();
    tablemodel.setRowCount(0);
    try {
        Statement stmt = con.createStatement();
        ResultSet res = stmt.executeQuery(query);

        while(res.next()){
            String matricule = res.getString("matricule");
            String marque = res.getString("marque");
            String model = res.getString("model");
            String etat = res.getString("etat");
            tablemodel.addRow(new Object[]{matricule,marque,model,etat});

        }
    } catch (SQLException ex) {
        Logger.getLogger(ListePanne.class.getName()).log(Level.SEVERE, null, ex);
    }
}
```

```
private void rechercheActionPerformed(java.awt.event.ActionEvent evt) {
    String marque = marque_field.getText();
    String query = "SELECT * FROM voiture WHERE ETAT ='en attente' and marque = '"+marque+"'";
    DefaultTableModel tablemodel =(DefaultTableModel) table.getModel();
    tablemodel.setRowCount(0);
    try {
        Statement stmt = con.createStatement();
        ResultSet res = stmt.executeQuery(query);

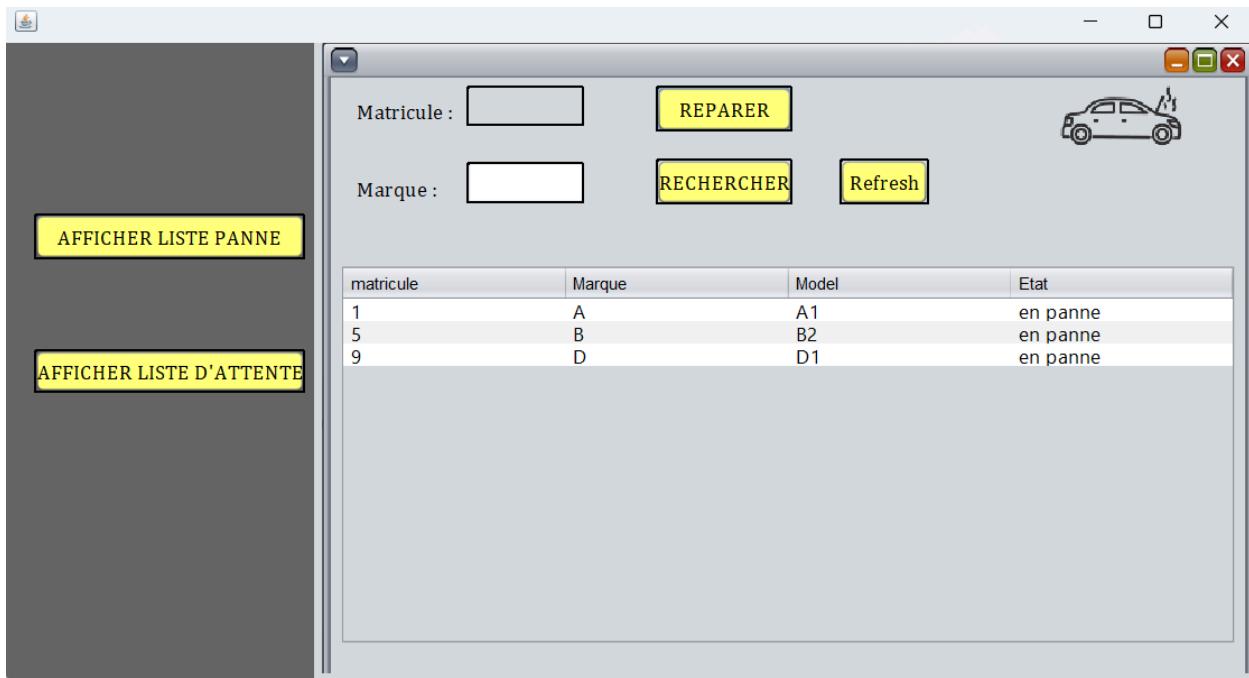
        while(res.next()){
            String matricule = res.getString("matricule");
            marque = res.getString("marque");
            String model = res.getString("model");
            String etat = res.getString("etat");
            tablemodel.addRow(new Object[]{matricule,marque,model,etat});

        }
    } catch (SQLException ex) {
        Logger.getLogger(ListePanne.class.getName()).log(Level.SEVERE, null, ex);
    }
}
```

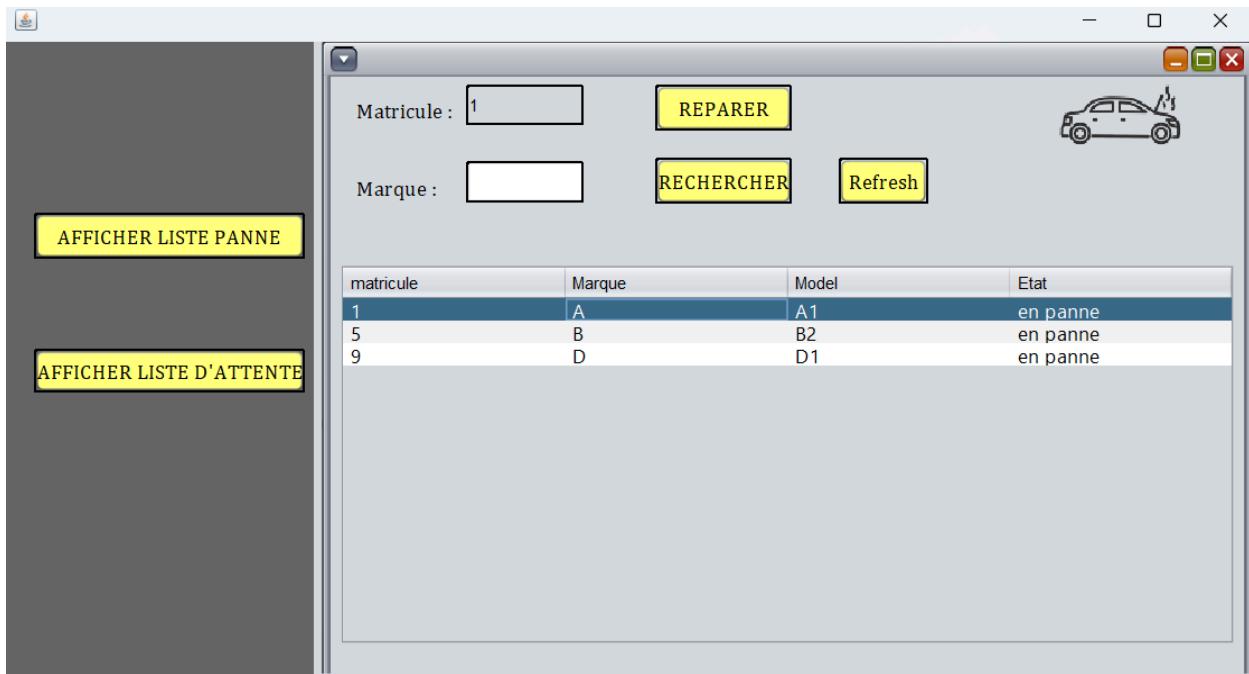
```
private void setdispoActionPerformed(java.awt.event.ActionEvent evt) {
    String matricule =matricule_field.getText();
    String query=" UPDATE voiture SET etat='disponible' WHERE matricule='"+matricule+"'";
    Statement stmt;
    try {
        stmt = con.createStatement();
        stmt.execute(query);
        JOptionPane.showMessageDialog(null,"CAR ( MATRICULE = "+matricule+) IS NOW AVAILABLE !!!");
        matricule_field.setText("");
    } catch (SQLException ex) {
        Logger.getLogger(ListePanne.class.getName()).log(Level.SEVERE, null, ex);
    }
    load();
}

private void setpanneActionPerformed(java.awt.event.ActionEvent evt) {
    String matricule =matricule_field.getText();
    String query=" UPDATE voiture SET etat='panne' WHERE matricule='"+matricule+"'";
    Statement stmt;
    try {
        stmt = con.createStatement();
        stmt.execute(query);
        JOptionPane.showMessageDialog(null,"CAR ( MATRICULE = "+matricule+) IS BROKEN DOWN !!!");
        matricule_field.setText("");
    } catch (SQLException ex) {
        Logger.getLogger(ListePanne.class.getName()).log(Level.SEVERE, null, ex);
    }
    load();
}
```

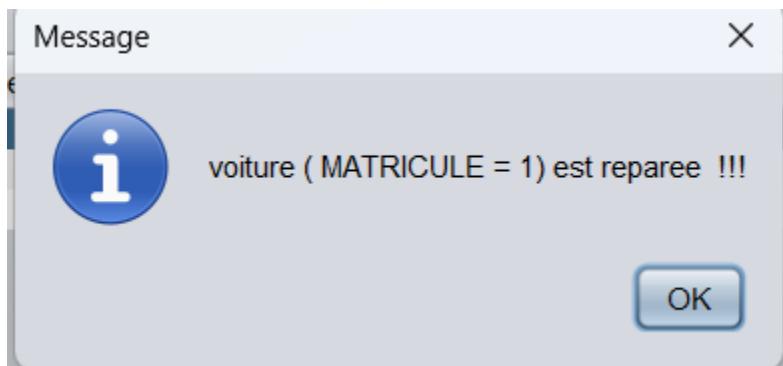
voiture en panne:



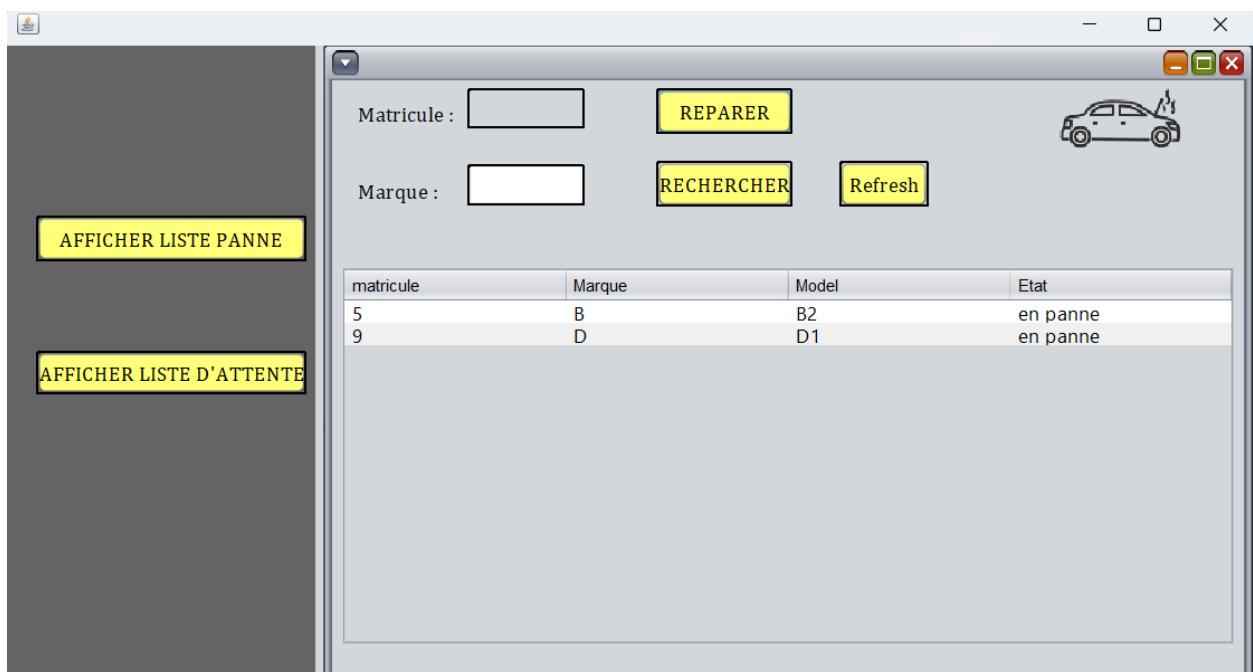
sélectionne la voiture:



répare la voiture



la liste se rafraîchit automatiquement:



code:

```
public void load() {
    Connecteur connect=new Connecteur();

    con = connect.connecttodb();
    String query = "SELECT * FROM voiture WHERE ETAT ='panne'";
    DefaultTableModel tablemodel =(DefaultTableModel) table.getModel();
    tablemodel.setRowCount(0);
    try {
        Statement stmt = con.createStatement();
        ResultSet res = stmt.executeQuery(query);

        while(res.next()){
            String matricule = res.getString("matricule");
            String marque = res.getString("marque");
            String model = res.getString("model");
            String etat = res.getString("etat");
            tablemodel.addRow(new Object[]{matricule,marque,model,etat});

        }
    } catch (SQLException ex) {
        Logger.getLogger(ListePanne.class.getName()).log(Level.SEVERE, null, ex);
    }
}
```

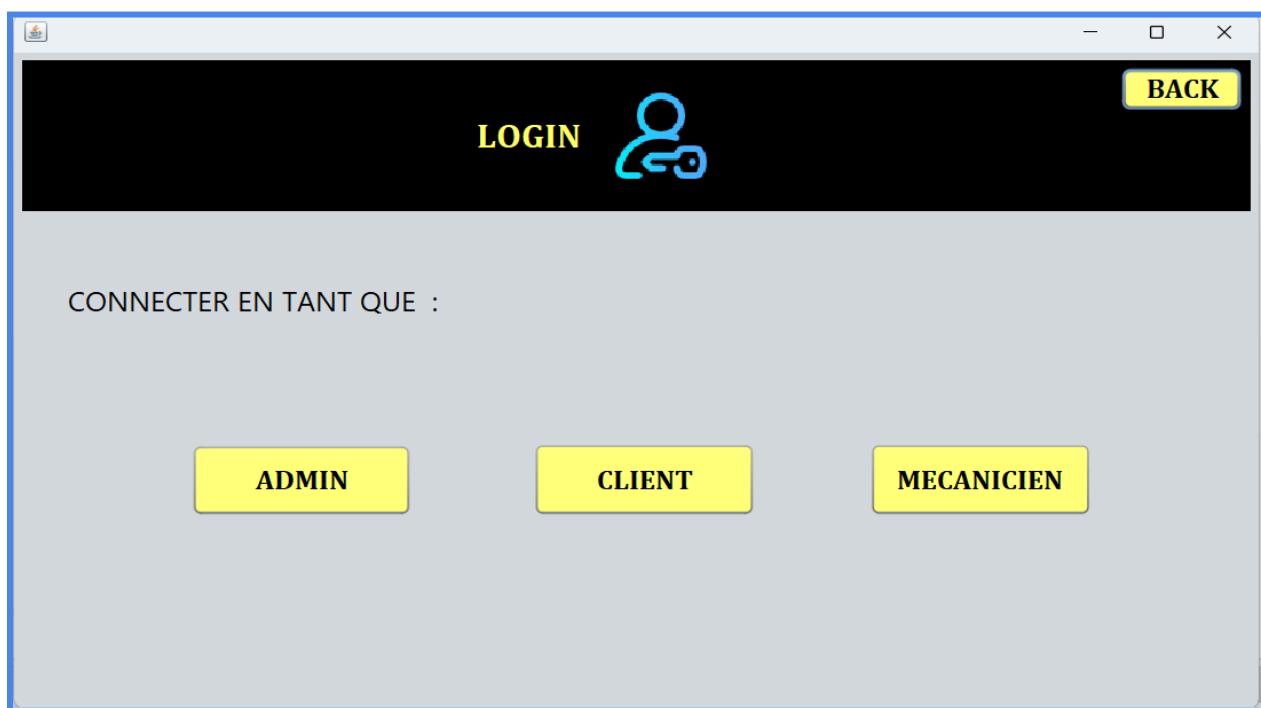
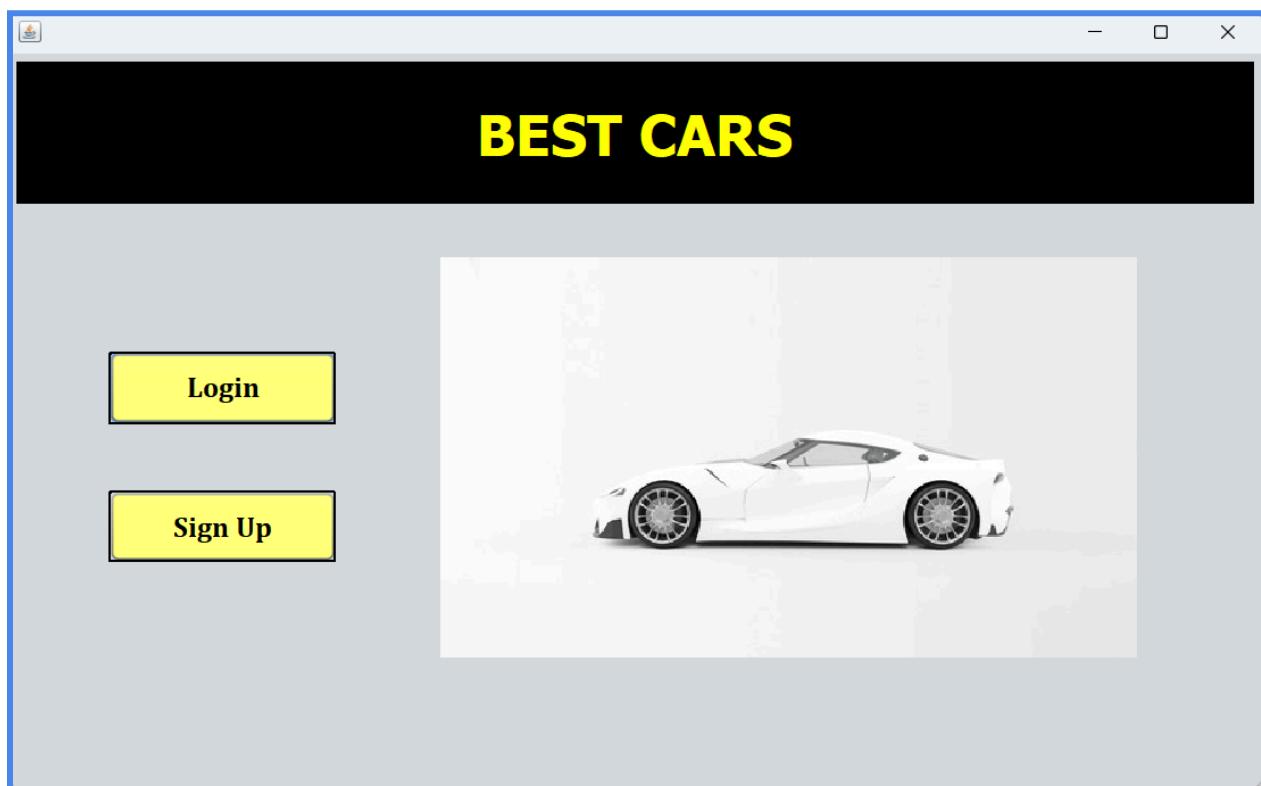
```
private void rechercheActionPerformed(java.awt.event.ActionEvent evt) {
    String marque = marque_field.getText();
    String query = "SELECT * FROM voiture WHERE ETAT ='panne' and marque = '"+marque+"'";
    DefaultTableModel tablemodel =(DefaultTableModel) table.getModel();
    tablemodel.setRowCount(0);
    try {
        Statement stmt = con.createStatement();
        ResultSet res = stmt.executeQuery(query);

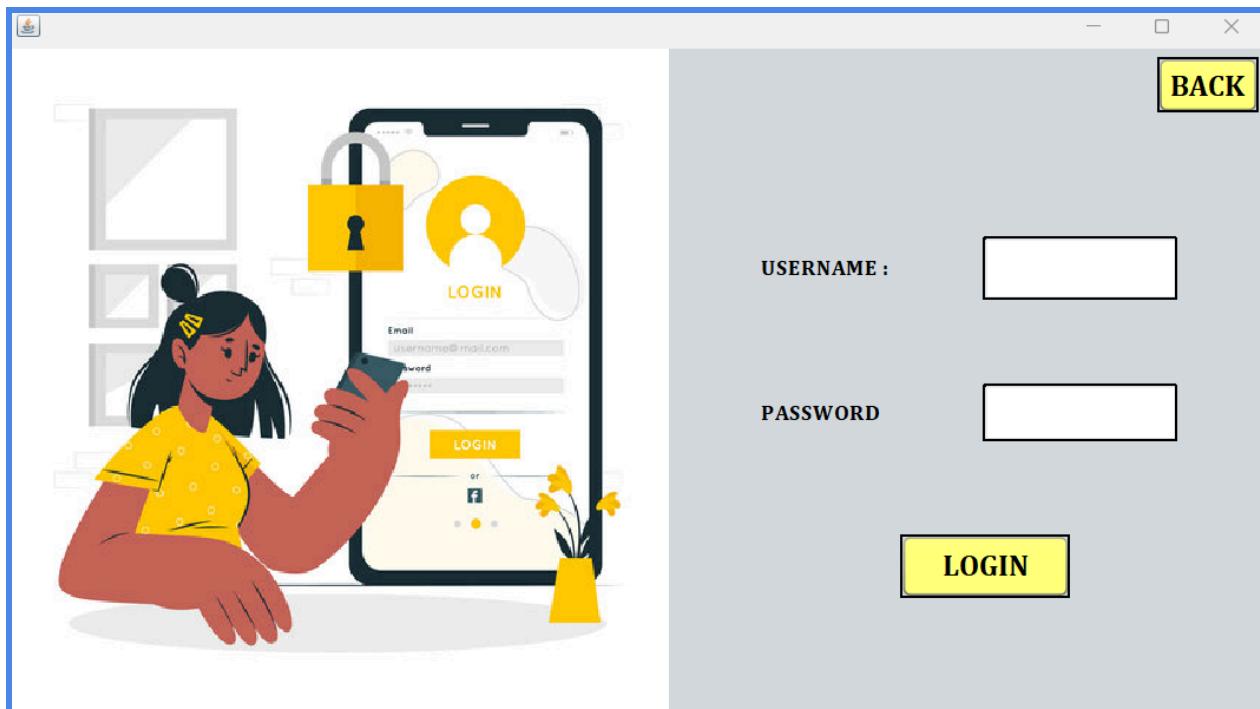
        while(res.next()){
            String matricule = res.getString("matricule");
            marque = res.getString("marque");
            String model = res.getString("model");
            String etat = res.getString("etat");
            tablemodel.addRow(new Object[]{matricule,marque,model,etat});

        }
    } catch (SQLException ex) {
        Logger.getLogger(ListePanne.class.getName()).log(Level.SEVERE, null, ex);
    }
}
```

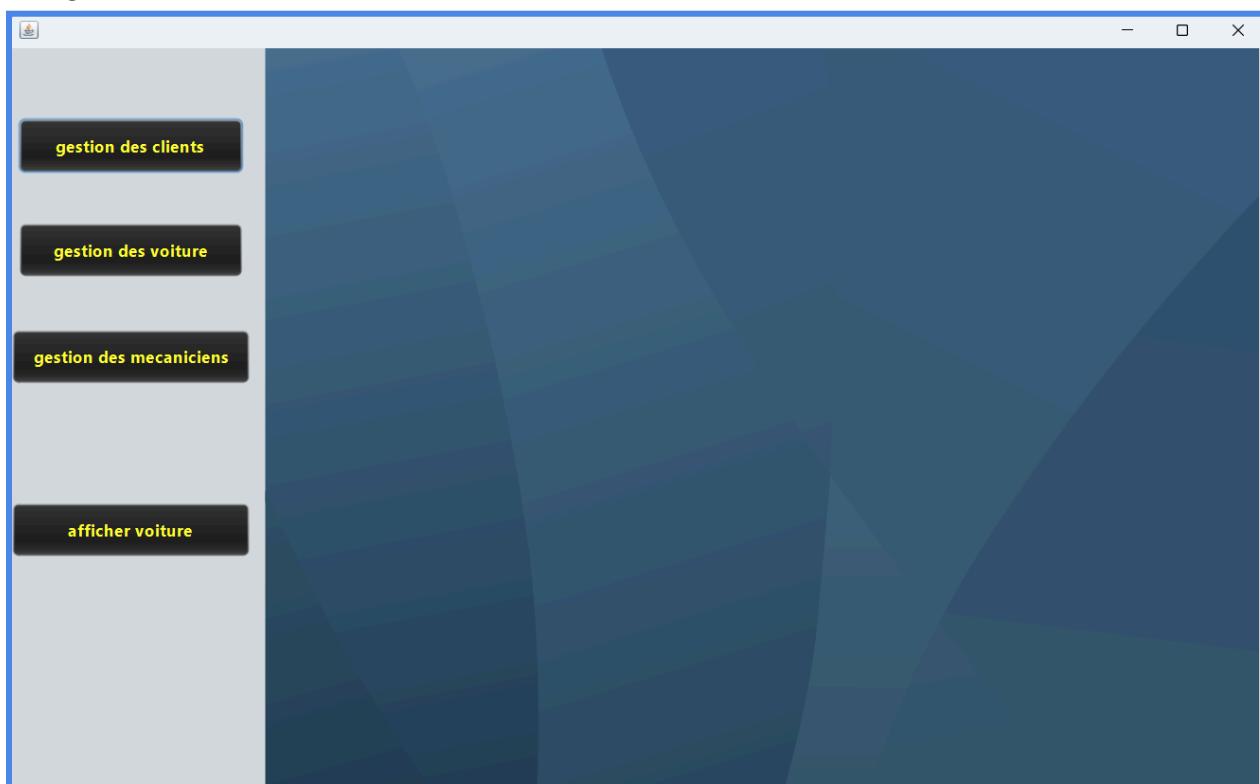
```
private void reparerActionPerformed(java.awt.event.ActionEvent evt) {
    String matricule =matricule_field.getText();
    String query=" UPDATE voiture SET etat='disponible' WHERE matricule='"+matricule+"'";
    Statement stmt;
    try {
        stmt = con.createStatement();
        stmt.execute(query);
        JOptionPane.showMessageDialog(null,"CAR ( MATRICULE = "+matricule+) REPAIRED !!!");
        matricule_field.setText("");
    } catch (SQLException ex) {
        Logger.getLogger(ListePanne.class.getName()).log(Level.SEVERE, null, ex);
    }
    load();
}
```

Interfaces CLIENT:

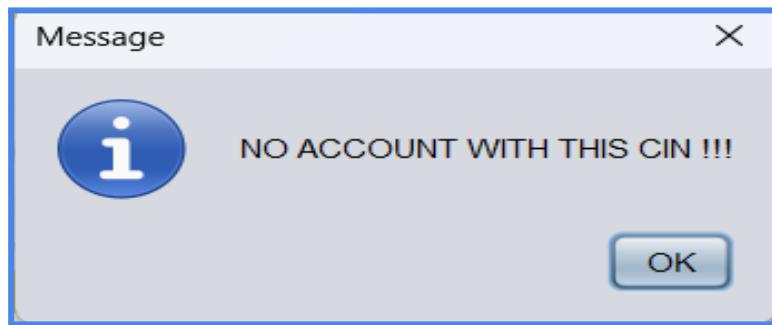




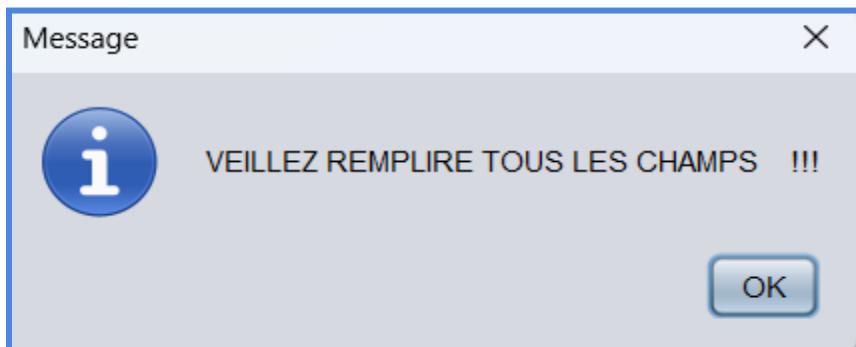
si login est terminée avec succès :



sinon : si le cin est faux (n'existe pas dans la base de donnée) :



si les champs sont vides (aux moin un champ vide) :

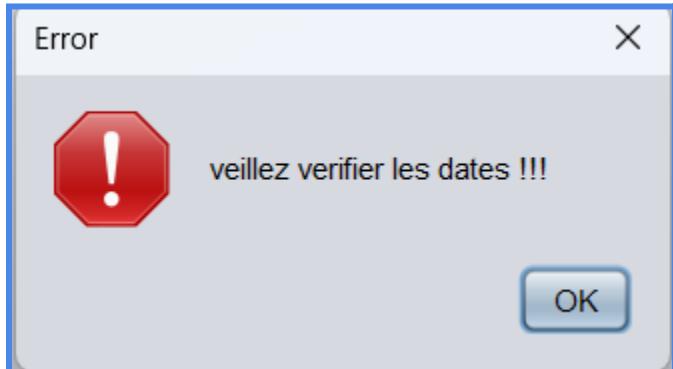


1)Louer voiture :

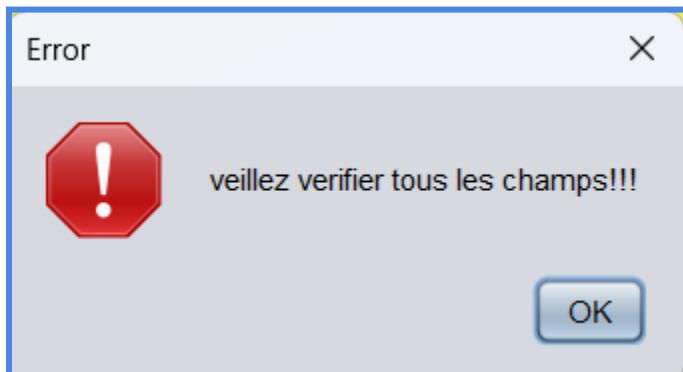
A screenshot of a Windows application window titled "LOUER VOITURE :". The window has a yellow header bar. On the left, there's a sidebar with buttons: "BIENVENUE , fourat", "LOUER VOITURE" (highlighted in yellow), "PRENDRE VOITURE", "RETOURNER VOITURE", and "ANNULER RESERVATION". The main area contains form fields: "Marque =" with dropdown menu showing "A", "Model =" with dropdown menu showing "A1", "Date Début =" with a date picker, and "Date Fin =" with a date picker. To the right of these fields is a cartoon illustration of a car and coins. At the bottom center is a yellow "RESERVER" button.

-sélectionner la Marque , Modèle , Date Début , Date Fin

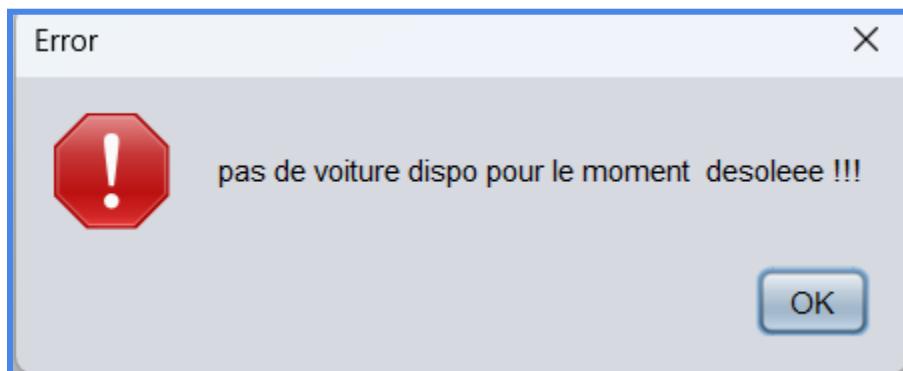
si il y a un problème dans le choix des dates (date debut apres date fin / date début avant aujourd'hui)



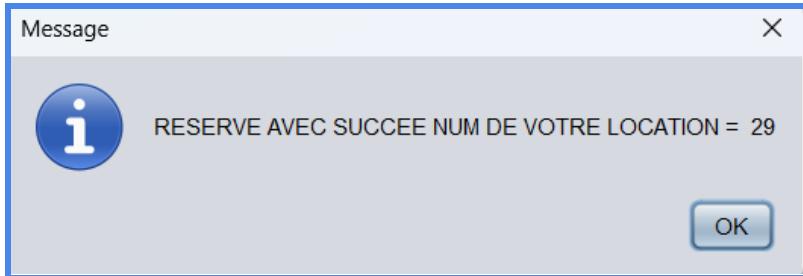
si les champs sont vides :



si on a pas de voiture disponible :



si la reservation est passé avec succès (on a de voiture disponible)



Algorithme de recherche des voitures disponible :

- 1 - il favorise les voiture neuves (qui n'ont pas de location auparavant)**
- 2 - sinon : on favorise les voiturs qui sont libre dans la période de location est qui sont disponible (disponibilite = disponible)**
- 3 - lors de la recherche on etendue la periode de reservation d' un jour avant et un jour après pour minimiser les erreurs**

CODE :

```
private void reserverbtnActionPerformed(java.awt.event.ActionEvent evt) {  
    if(!verifydates()){  
        //verifier si les date sont correctes  
        //elle entre la si les date sont incorrectes  
    }else{  
        if(Marque.getSelectedItem()==null || Model.getSelectedItem()==null || DateDebut.getDate()== null || DateFin.getDate()==null ){  
            JOptionPane.showMessageDialog(null, " veillez verifier tous les champs!!! ", "Error", JOptionPane.ERROR_MESSAGE);  
  
        }else{  
            String marque=Marque.getSelectedItem().toString();  
            String model=Model.getSelectedItem().toString();  
  
            java.util.Date datedeb = DateDebut.getDate();  
            java.util.Date datefin = DateFin.getDate();  
  
            // Create a Calendar instance and set it to the datedeb  
            Calendar calendar = Calendar.getInstance();  
            calendar.setTime(datedeb);  
            // Subtract one day from the calendar  
            calendar.add(Calendar.DAY_OF_MONTH, -1);  
            // Get the updated date from the calendar  
            java.util.Date updatedDate = calendar.getTime(); // date debut - 1 day  
  
            //to get datefin_new = datefin + 1 day  
            // Create a Calendar instance and set it to the datedeb  
            Calendar calendarfin = Calendar.getInstance();  
            calendarfin.setTime(datefin);  
            // Subtract one day from the calendar  
            calendarfin.add(Calendar.DAY_OF_MONTH, +1);  
            // Get the updated date from the calendar  
            java.util.Date datefin_new = calendarfin.getTime(); // date fin +1 day  
  
            // Define the date format pattern  
            String pattern = "yyyy-MM-dd";  
        }  
    }  
}
```

```

// Create a SimpleDateFormat object with the pattern
SimpleDateFormat dateFormat = new SimpleDateFormat(pattern);
String datedeb_initial = dateFormat.format(datedeb); // get a string value of datedebut initial
// string datedebut = datedebut - 1 day
// Format the date to a string
String datedebut = dateFormat.format(updatedDate);
String datefin_newstring = dateFormat.format(datefin_new);

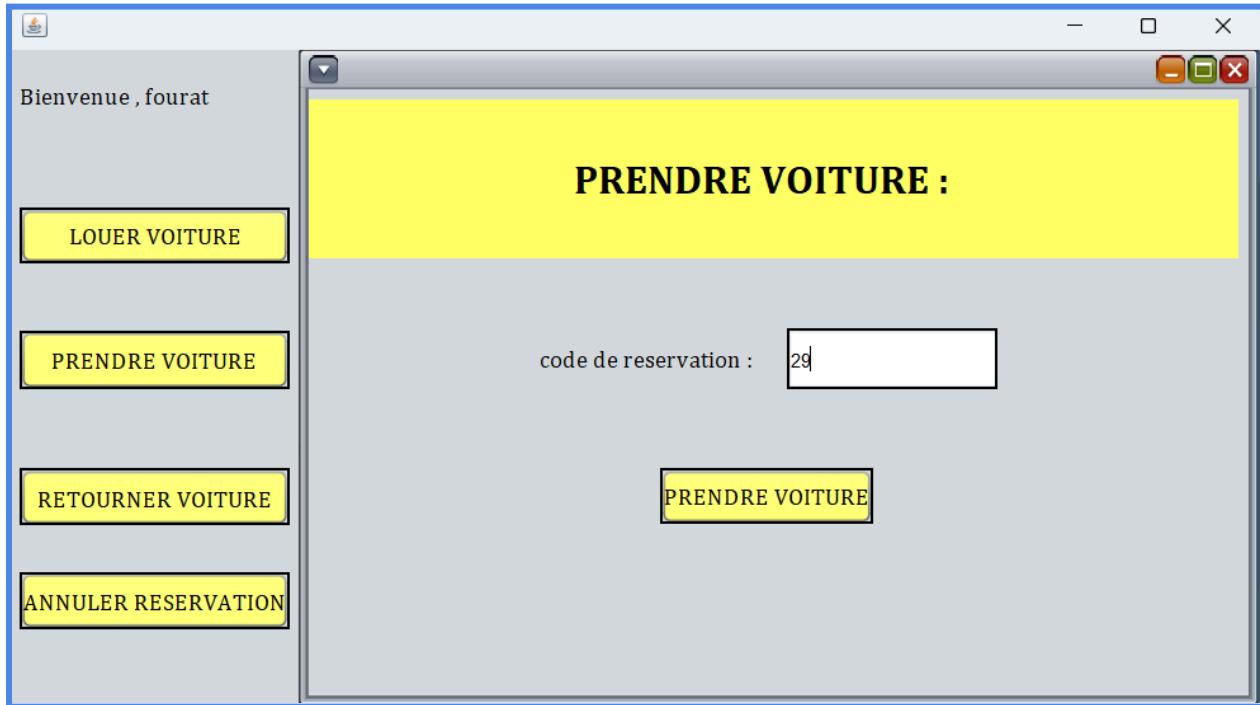
//ALGO DE RECHERCHE D'UNE VOITURE QUI FAVORISE LES VOITURES NOUVEAU (AUCUNE LOCATION AUPARAVANT)

String sql = "SELECT V.matricule FROM voiture V LEFT JOIN location L on V.matricule = L.matricule WHERE V.marque ='" +marque+
"' and V.model = '" +model+" AND V.disponibilite = 'disponible' AND V.etat='en marche' AND L.id IS null ";
try {
    Statement stmt = con.createStatement();
    ResultSet res = stmt.executeQuery(sql);

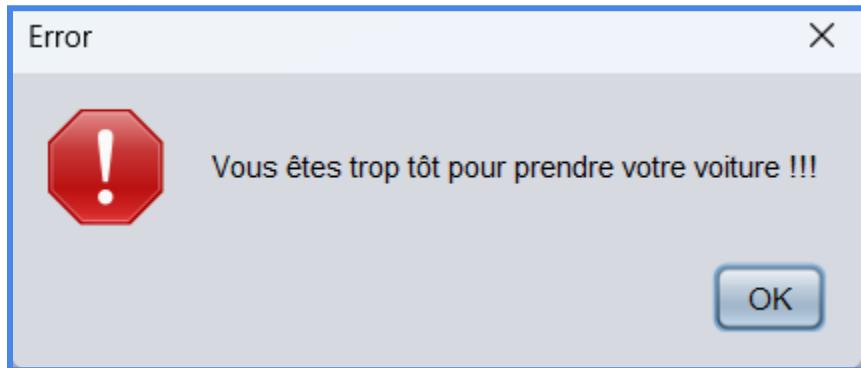
    if(res.next()){
        String matricule_voiture = res.getString("matricule");
        Location location = new Location(0,client.getCin(),matricule_voiture,datedeb,datefin,null,0,null,null);
        String id_location = addlocation(location.getCin(),location.getMatricule(),location.getDateDebut(),location.getDateFin());
        if(!id_location.isEmpty()){
            JOptionPane.showMessageDialog(null,"RESERVE AVEC SUCCÉE "
                + "NUM DE VOTRE LOCATION = "+id_location);
            resetbtn();
            //LA RESERVATION EST TERMINEE AVEC SUCCÉE
        }else{
            JOptionPane.showMessageDialog(null,"erreur de reservation !!!");
        }
    }else{
        //si il n y a pas des voitures nouveau on CHERCHE LES VOITURES A ETAT ( en marche ) QUI SONT DISPONIBLES ENTRE LES DATES DE RESERVATION PRESISER PAR LE CLIENT
        //ON FAVORISE LES VOITURE DISPONIBLE QUE LES VOITURE LOUEE (V.disponibilite DESC)
        sql = "SELECT V.matricule FROM voiture V LEFT JOIN location L on V.matricule = L.matricule WHERE V.marque ='" +marque+
"' and V.model = '" +model+
"' AND V.etat='en marche' AND L.date_retour IS NULL AND V.matricule NOT IN (SELECT L2.matricule FROM location L2 WHERE (L2.date_debut <= '" +
+datedebut+" AND "+datedebut+" <= L2.date_fin ) OR ( L2.date_debut <= '" +
+datefin_newstring+" AND "+datefin_newstring+" <= L2.date_fin) OR (L2.date_debut <= '" +
+datedebut+" AND "+datefin_newstring+" <= L2.date_fin) OR(L2.date_debut >= '" +
+datedebut+" AND "+datefin_newstring+" >= L2.date_fin)) ORDER BY V.disponibilite DESC";
        res.close();
        /date_retour = null
        res= stmt.executeQuery(sql);
        if(res.next()){
            String matricule_voiture = res.getString("matricule");
            Location location = new Location(0,client.getCin(),matricule_voiture,datedeb,datefin,null,0,null,null);
            String id_location = addlocation(location.getCin(),location.getMatricule(),location.getDateDebut(),location.getDateFin());
            if(!id_location.isEmpty()){
                JOptionPane.showMessageDialog(null,"RESERVE AVEC SUCCÉE "
                    + "NUM DE VOTRE LOCATION = "+id_location);
                resetbtn();
            }else{
                JOptionPane.showMessageDialog(null,"erreur de reservation !!!");
            }
        }else{
            JOptionPane.showMessageDialog(null, " pas de voiture dispo pour le moment desoleee !!! ", "Error", JOptionPane.ERROR_MESSAGE);
        }
    }
} catch (SQLException ex) {
    Logger.getLogger(LouerVoiture.class.getName()).log(Level.SEVERE, null, ex);
}
}
}

```

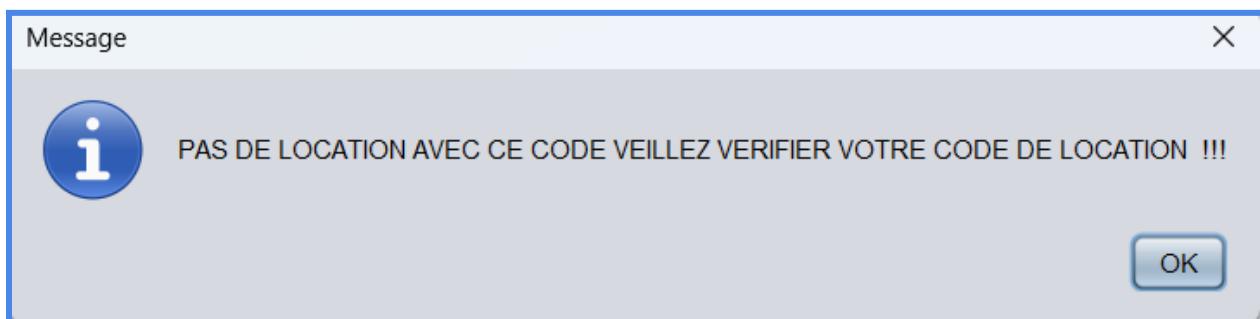
2) Prendre Voiture :



si le client est veut prendre sa voiture avant le début de sa réservation :



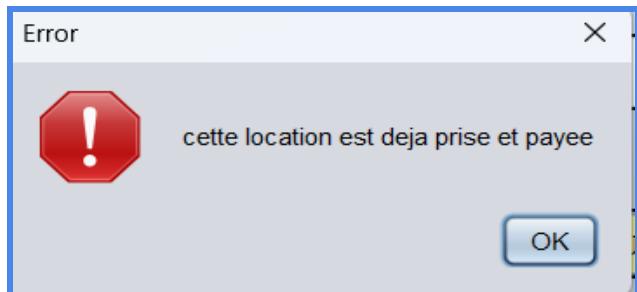
si le code est incorrect ou le champ est vide :



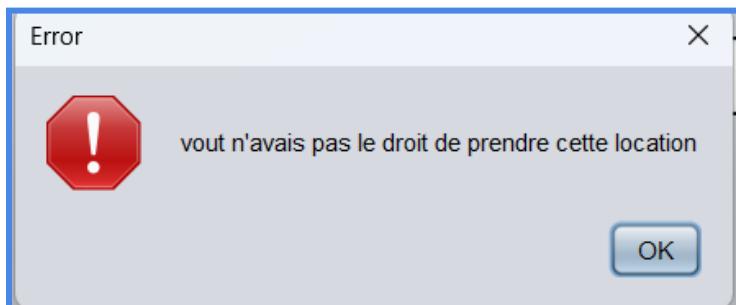
cas idéal :

diriger vers [Payer location](#)

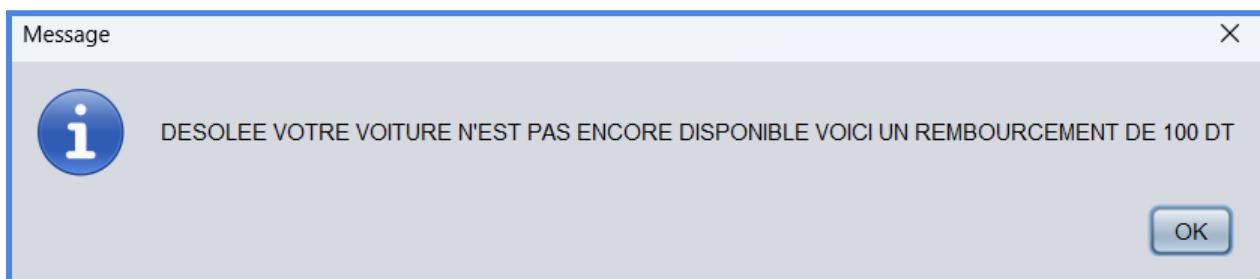
si la voiture déjà prise par le client :



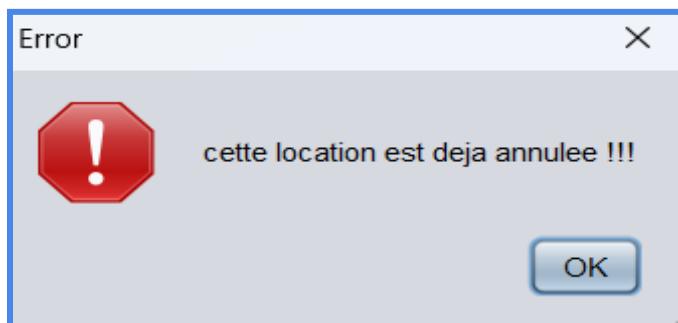
si le client veut prendre la voiture après la période de réservation (il est en retard)



si la voiture n'est pas disponible : on lui donne un remboursement



si la location est annulée :



CODE :

```
private void prendrebtnActionPerformed(java.awt.event.ActionEvent evt) {  
    String id_location = code.getText();  
    String sql = "SELECT * FROM location WHERE ID='"+id_location+"'";  
    try {  
        Statement stmt = con.createStatement();  
        ResultSet res= stmt.executeQuery(sql);  
        if(res.next()){  
            String cin=res.getString("CIN");  
            java.sql.Date datedeb=res.getDate("date_debut");  
            java.sql.Date datefin=res.getDate("date_fin");  
            //verifier si elle est deja payee dans le tableau paiement |  
            if(res.getString("num_carte_bancaire")!=null){  
                JOptionPane.showMessageDialog(null, " cette location est deja prise et payee ", "Error", JOptionPane.ERROR_MESSAGE);  
            }else{  
                if(cin.equals(client.getCin())){  
                    if(verifdate(datedeb,datefin)){  
                        String matricule = res.getString("matricule");  
                        System.out.println(matricule);  
                        sql="SELECT * FROM voiture WHERE matricule = '"+matricule+"'";  
                        res.close();  
                        res = stmt.executeQuery(sql);  
                        String etat="";  
                        String disponibilite="";  
                        int nb_fois_louee =0;  
                        float prix=0;  
                        String marque="";  
                        String model="";  
                        if(res.next()){  
                            etat = res.getString("etat");  
                            disponibilite=res.getString("disponibilite");  
                            nb_fois_louee = res.getInt("nb_fois_louee");  
                            prix=res.getFloat("prix");  
                            marque=res.getString("marque");  
                            model=res.getString("model");  
                        }  
                    }  
                }  
            }  
        }  
    }  
}
```

Activer Windo

```

        if(etat.equals("en marche") && disponibilite.equals("disponible")){
            nb_fois_louee++;
            // Convert to milliseconds
            long millisecondsInDay = 1000 * 60 * 60 * 24; // Number of milliseconds in a day
            long datedebMillis = datedeb.getTime();
            long datefinMillis = datefin.getTime();
            // Calculate the difference in days
            long differenceInMillis = datefinMillis - datedebMillis;
            int differenceInDays = (int) (differenceInMillis / millisecondsInDay)+1;
            PayerLocation adc=new PayerLocation(jDesktopPanel1,con,client,nb_fois_louee,id_location,differenceInDays,prix,matricule,marque,model
            ,datedeb.toString(),datefin.toString());
            jDesktopPanel1.removeAll();
            jDesktopPanel1.updateUI(); //bech tsaker w thel haja jdida
            jDesktopPanel1.add(adc);
            adc.show();
        }else{
            Remboursement remboursement = new Remboursement(id_location, 100);
            // insertion du donner de remboursement
            sql = "INSERT INTO remboursement (id_location,montant,date_remboursement) VALUE (?,?,CURDATE())";
            PreparedStatement pstmtt = con.prepareStatement(sql);
            pstmtt.setString(1,remboursement.getIdLocation());
            pstmtt.setDouble(2, remboursement.getMontant());
            pstmtt.executeUpdate();
            System.out.println("Data inserted successfully into remboursement table.");
            //modification des donner dans la table location ( rendre les champs null sauf ID et CIN)
            sql = "UPDATE location SET matricule = null, date_debut = null, date_fin = null WHERE ID = '" + id_location + "'";
            stmt.executeUpdate(sql);
            JOptionPane.showMessageDialog(null," DESOLEE VOTRE VOITURE N'EST PAS ENCORE DISPONIBLE "+ "VOICI UN REMBOURSEMENT DE 100 DT");
        }
    }
} else{
    JOptionPane.showMessageDialog(null," CETTE LOCATION N'EST PAS RESERVEE PAR VOUS "+ "VEILLER VERIFIER VOTRE CODE DE LOCATION ");
}
} else{JOptionPane.showMessageDialog(null,"PAS DE LOCATION AVEC CE CODE "+ "VEILLESZ VERIFIER VOTRE CODE DE LOCATION !!!");}
} catch (SQLException ex) {
    Logger.getLogger(PrendreVoiture.class.getName()).log(Level.SEVERE, null, ex);
}

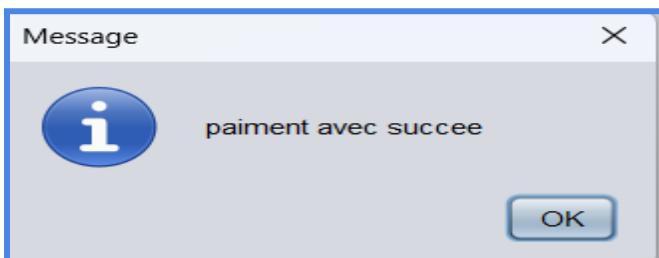
```

Activer Windows
Accédez aux paramètres

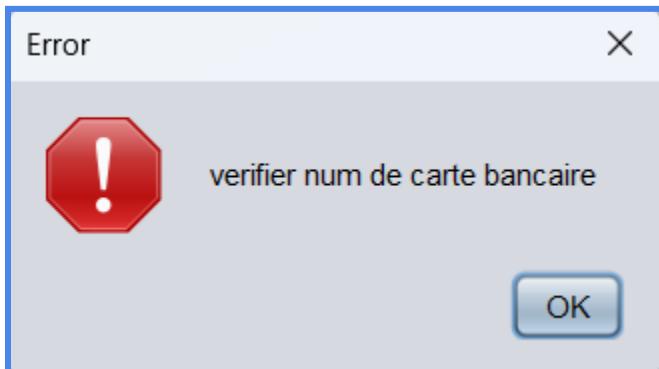
3) Payer location :



cas idéal :



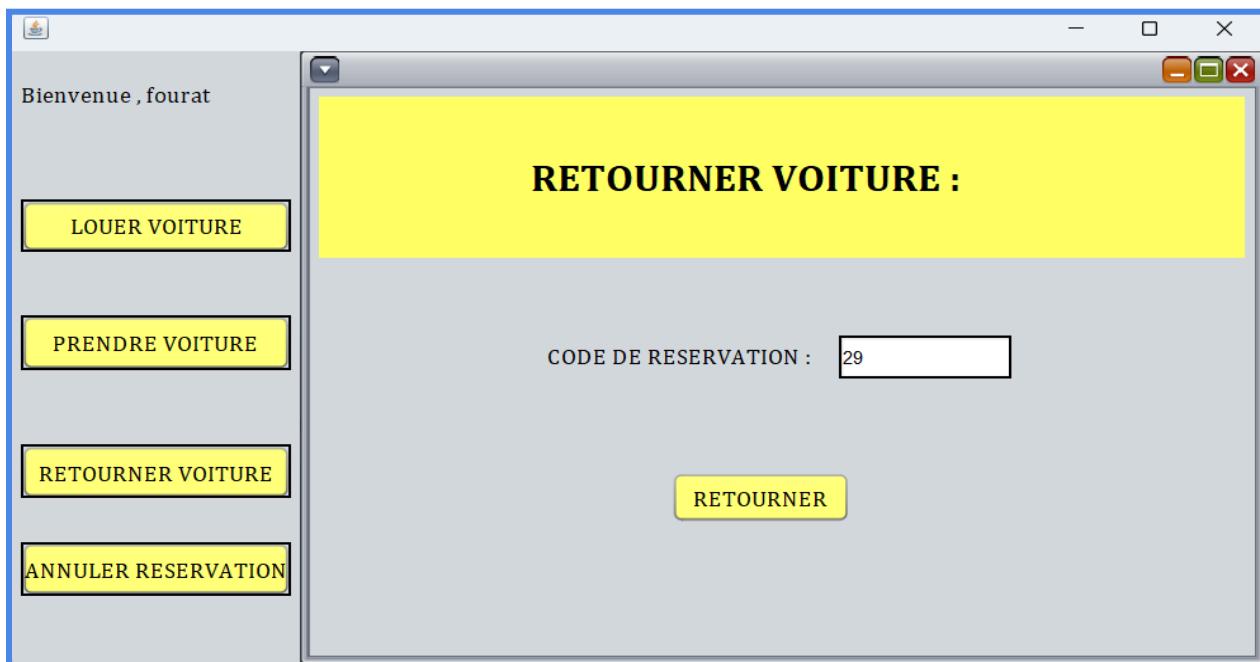
si le client ne fournit pas le num de carte bancaire :



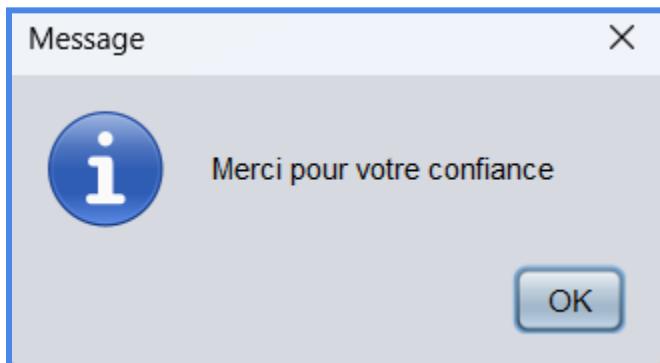
CODE :

```
private void payerActionPerformed(java.awt.event.ActionEvent evt) {
    String num_carte= carteBancaire.getText();
    if(num_carte.isEmpty()){
        JOptionPane.showMessageDialog(null, " verifier num de carte bancaire ", "Error", JOptionPane.ERROR_MESSAGE);
        return ;
    }
    if(!verify_deja_payee()){
        JOptionPane.showMessageDialog(null, " cette location est deja payee ", "Error", JOptionPane.ERROR_MESSAGE);
    }else{
    String sql = "UPDATE location SET total=?, num_carte_bancaire=?, date_paiement=CURDATE() WHERE ID=?";
    try {
        PreparedStatement pstmt = con.prepareStatement(sql);
        pstmt.setFloat(1, this.total);
        pstmt.setString(2, num_carte);
        pstmt.setString(3, this.id_location);
        pstmt.executeUpdate();
    } catch (SQLException ex) {
        Logger.getLogger(PayerLocation.class.getName()).log(Level.SEVERE, null, ex);
    }
    sql="UPDATE voiture SET nb_fois_louee=?, disponibilite='louee' WHERE matricule=?";
    PreparedStatement pstmt2;
    try {
        pstmt2 = con.prepareStatement(sql);
        pstmt2.setInt(1, this.nb_fois_louee);
        pstmt2.setString(2, this.matricule);
        pstmt2.executeUpdate();
        JOptionPane.showMessageDialog(null," paiement avec succee ");
        JInternalFrame[] frames = jDesktopPanel1.getAllFrames();
        for (JInternalFrame frame : frames) {
            frame.dispose(); // Close the internal frame
        }
    } catch (SQLException ex) {
        Logger.getLogger(PayerLocation.class.getName()).log(Level.SEVERE, null, ex);
    }
}
}
```

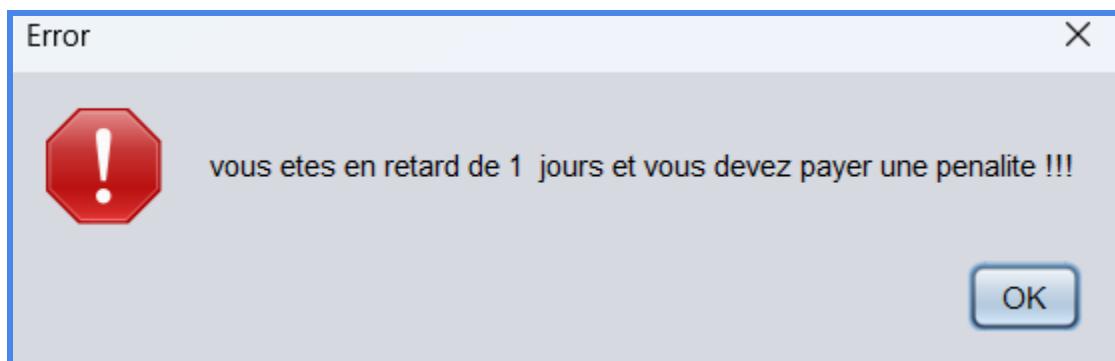
4) Retourner Voiture :



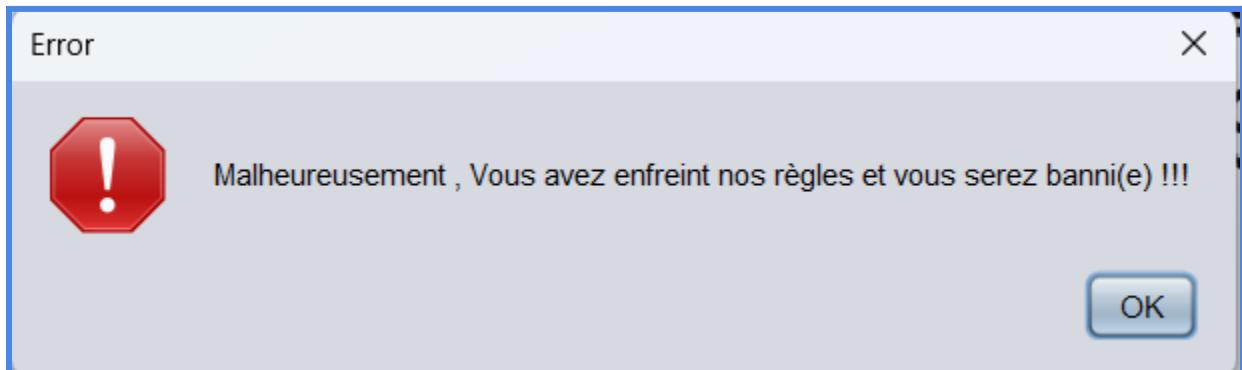
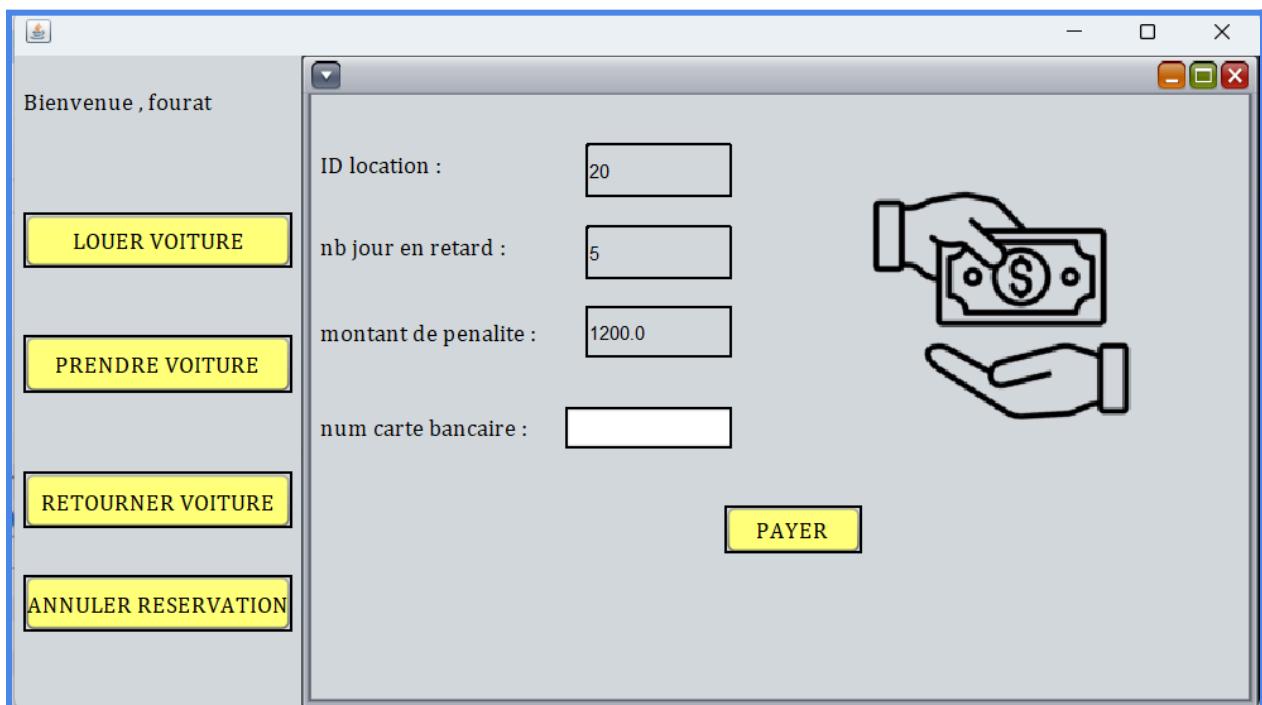
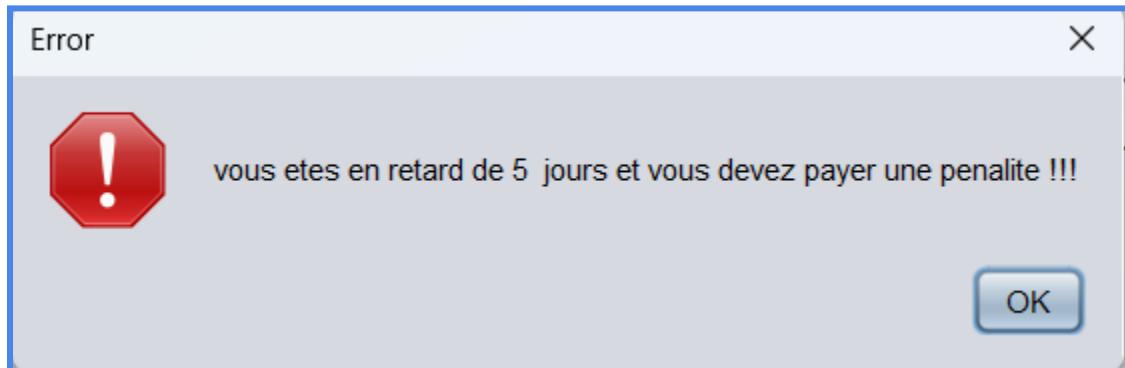
cas idéal : le client retourne la voiture avant ou à la date de fin de sa réservation



si il est en retard (nbr de jours en retard < 5) il doit payer une pénalité :

A screenshot of a car rental application window. On the left, there's a sidebar with buttons: "LOUER VOITURE", "PRENDRE VOITURE", "RETOURNER VOITURE", and "ANNULER RESERVATION". The main area shows a form with fields: "ID location : 32", "nb jour en retard : 1", "montant de penalite : 240.0", and "num carte bancaire : 12121212". To the right of the form is an illustration of two hands exchanging a dollar bill. A yellow "PAYER" button is located below the form.

si il est en retard (nbr de jours en retard ≥ 5) il doit payer une pénalité et va etre bannie:



CODE :

```
private void retournerbtnActionPerformed(java.awt.event.ActionEvent evt) {  
    String sql ="SELECT * FROM location WHERE ID=?";  
    String id_location=id_location_field.getText();  
    try {  
        PreparedStatement pstmt = con.prepareStatement(sql);  
        pstmt.setString(1, id_location);  
        ResultSet res = pstmt.executeQuery();  
        if(res.next()) {  
            if(res.getString("CIN").equals(client.getCin())) {  
                if(res.getString("date_retour")!=null){  
                    JOptionPane.showMessageDialog(null, " cette voiture est déjà rentrée !!! ", "Error", JOptionPane.ERROR_MESSAGE);  
                }else{  
                    if(res.getString("num_carte_bancaire")==null){  
                        JOptionPane.showMessageDialog(null, " il faut payer les frais de location avant !!! ", "Error", JOptionPane.ERROR_MESSAGE);  
                    }else{  
                        Date date_fin=res.getDate("date_fin");  
                        LocalDate currentDate = LocalDate.now();  
                        //LocalDate date_fin_new = date_fin.toInstant().atZone(ZoneId.systemDefault()).toLocalDate();  
                        LocalDate date_fin_new = date_fin.toLocalDate();  
                        long differenceInDays = ChronoUnit.DAYS.between(date_fin_new,currentDate);  
                        System.out.println("nb jour en retard = "+differenceInDays);  
                        //update etat car  
                        String matricule=res.getString("matricule");  
                        sql="UPDATE voiture SET etat='en attente', disponibilite='disponible' WHERE matricule=?";  
                        PreparedStatement pstmt2 = con.prepareStatement(sql);  
                        pstmt2.setString(1, matricule);  
                        pstmt2.executeUpdate();  
                        pstmt2.clearParameters();  
                        //Update location ajouter date de retour  
                        sql="UPDATE location SET date_retour=CURDATE() WHERE ID=?";  
                        pstmt2=con.prepareStatement(sql);  
                        pstmt2.setString(1, id_location);  
                        pstmt2.executeUpdate();  
                        if(differenceInDays>0){  
                            if(differenceInDays>0){  
                                JOptionPane.showMessageDialog(null, " la date de retour est incorrecte !!! ", "Error", JOptionPane.ERROR_MESSAGE);  
                            }else{  
                                JOptionPane.showMessageDialog(null, " la date de retour est correcte !!! ", "Success", JOptionPane.SUCCESS_MESSAGE);  
                            }  
                        }  
                    }  
                }  
            }  
        }  
    } catch (SQLException ex) {  
        JOptionPane.showMessageDialog(null, " Erreur lors de l'exécution de la requête SQL !!! ", "Error", JOptionPane.ERROR_MESSAGE);  
    }  
}
```

Autorisé par l'administrateur

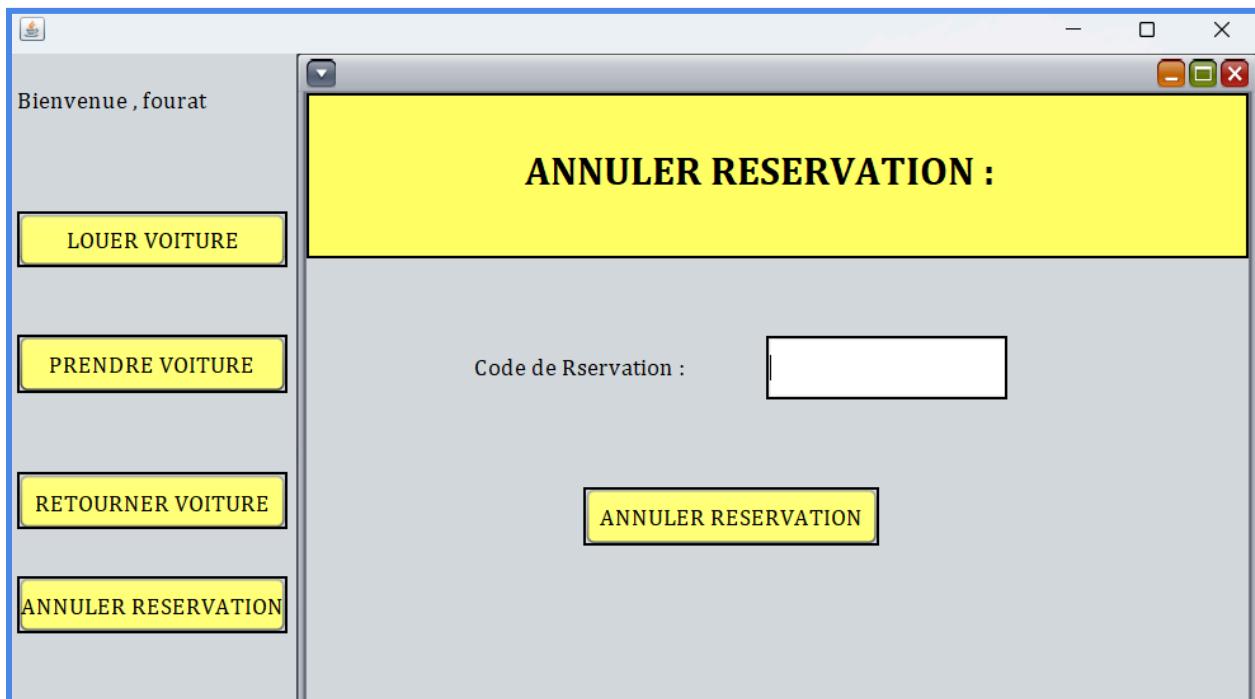
```

        if(differenceInDays>0) {
            JOptionPane.showMessageDialog(null, " vous etes en retard de "+differenceInDays+" jours et vous devez payer une penalite !!! ",
                    "Error", JOptionPane.ERROR_MESSAGE);
            //calculer le montant de peanite
            sql="SELECT prix FROM voiture WHERE matricule = ?";
            pstmt2.clearParameters();
            pstmt2=con.prepareStatement(sql);
            pstmt2.setString(1, matricule);
            res = pstmt2.executeQuery();
            float prix=0;
            if(res.next()){
                prix=res.getFloat("prix");
            }
            double monant_penalite = prix* differenceInDays*2;
            //payer penalite
            PayerPenalite adc=new PayerPenalite(jDesktopPanel,con,client,differenceInDays,monant_penalite,id_location);
            jDesktopPanel.removeAll();
            jDesktopPanel.updateUI(); //bech tsaker w thel haja jdida
            jDesktopPanel.add(adc);
            adc.show();
        }else{
            // si le client retourne la voiture dans le temps propre ou meme avant
            JOptionPane.showMessageDialog(null, " Merci pour votre confiance ");
            JInternalFrame[] frames = jDesktopPanel.getAllFrames();
            for (JInternalFrame frame : frames) {
                frame.dispose(); // Close the internal frame
            }
        }
    }
} else{
    JOptionPane.showMessageDialog(null, " cette location n'est pas pour vous !!! ", "Error", JOptionPane.ERROR_MESSAGE);
} else{
    JOptionPane.showMessageDialog(null, " Veuillez verifier votre code de location ", "Error", JOptionPane.ERROR_MESSAGE);
} catch (SQLException ex) {
}

```

Activer Windows

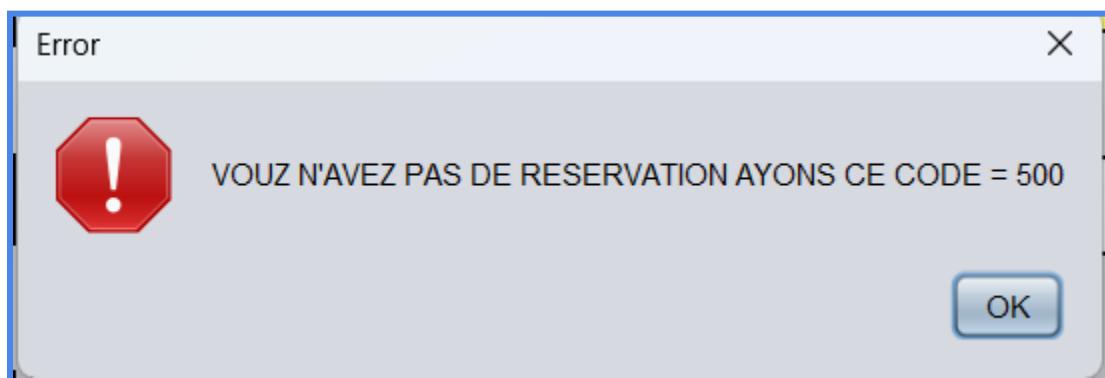
5) annuler reservation :



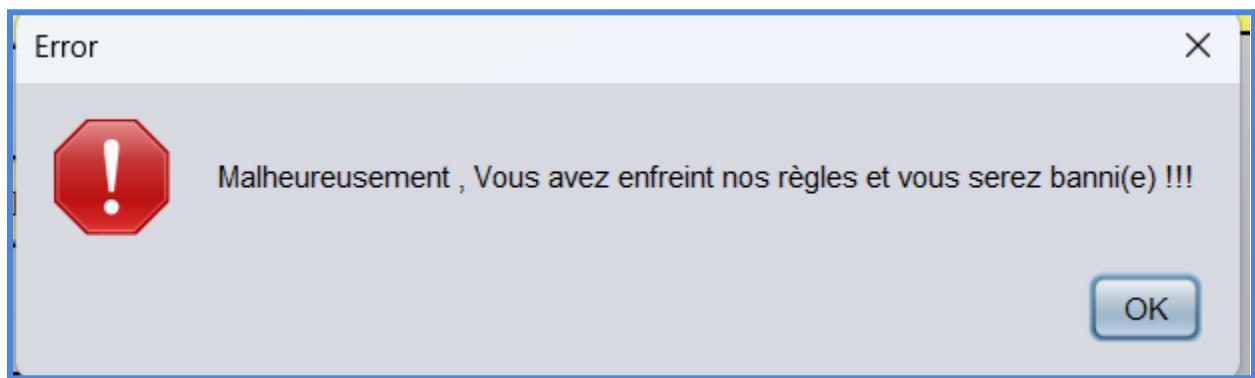
cas idéal (annuler la reservation avant la date de début de location) :



code incorrect :



si le client annule sa reservation apres le date de debut de sa location : client sera bannee



CODE :

```
private void annulerActionPerformed(java.awt.event.ActionEvent evt) {  
    String id = id_location.getText();  
    try {  
        Statement stmt = con.createStatement();  
        String sql = "SELECT * FROM location WHERE ID='"+id+"' AND CIN='"+client.getCin()+"';  
        ResultSet res = stmt.executeQuery(sql);  
        if(res.next()){  
            Date datedeb = res.getDate("date_debut");  
            // Convert java.sql.Date to LocalDate  
            LocalDate localDateDebut = datedeb.toLocalDate();  
            // Get the current date  
            LocalDate currentDate = LocalDate.now();  
            // Now you can compare the two LocalDate objects  
            if (currentDate.isBefore(localDateDebut)) {  
                sql = "DELETE FROM LOCATION WHERE ID = '"+id+"';  
                stmt.executeUpdate(sql);  
                JOptionPane.showMessageDialog(null,"VOTRE RESERVATION EST ANNULER ");  
                // Close all internal frames  
                JInternalFrame[] frames = jDesktopPanel.getAllFrames();  
                for (JInternalFrame frame : frames) {  
                    frame.dispose(); // Close the internal frame  
                }  
            } else {  
                if(res.getString("num_carte_bancaire") != null){  
                    JOptionPane.showMessageDialog(null, "VOUS AVEZ DEJA PAYER CETTE LOCATION ", "Error", JOptionPane.ERROR_MESSAGE);  
                }else{  
                    //si le client veut annuler une location (une reservation ) le jour de la reservation ou même apres il sera BANNED car il nous a perdu de l'agent  
                    JOptionPane.showMessageDialog(null, " Malheureusement , Vous avez enfreint nos règles et vous serez banni(e) !!! ", "Error", JOptionPane.ERROR_MESSAGE);  
                    //insérer le client dans le table banned  
                    sql="INSERT INTO banned VALUE (?)";  
                    PreparedStatement pstmt = con.prepareStatement(sql);  
                    pstmt.setString(1, client.getCin());  
                    pstmt.executeUpdate();  
                    //delete the location from the table location  
                    sql = "DELETE FROM location WHERE ID='"+id+"'";  
                }  
            }  
        }  
    } catch (SQLException ex) {  
        Logger.getLogger(AnnulerReservation.class.getName()).log(Level.SEVERE, null, ex);  
    }  
}
```

Activer Windows

```
//close all  
// Close all internal frames  
JInternalFrame[] frames = jDesktopPanel.getAllFrames();  
for (JInternalFrame frame : frames) {  
    frame.dispose(); // Close the internal frame  
}  
  
// Get the parent frame and dispose of it  
Window parentWindow = SwingUtilities.windowForComponent(jDesktopPanel);  
parentWindow.dispose();  
MainPage log =new MainPage();//thel wahda jdida  
log.setLocationRelativeTo(null);  
log.setVisible(true);//tkhalifa todhher  
}}  
} else{JOptionPane.showMessageDialog(null, "VOUZ N'AVEZ PAS DE RESERVATION AYONS CE CODE = "+id, "Error", JOptionPane.ERROR_MESSAGE);}  
} catch (SQLException ex) {  
    Logger.getLogger(AnnulerReservation.class.getName()).log(Level.SEVERE, null, ex);  
}
```

6) payer pénalité :

Bienvenue , fourat

LOUER VOITURE

PRENDRE VOITURE

RETOURNER VOITURE

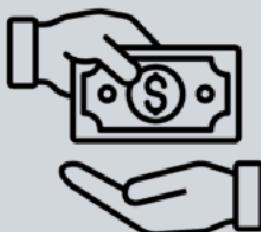
ANNULER RESERVATION

ID location :

nb jour en retard :

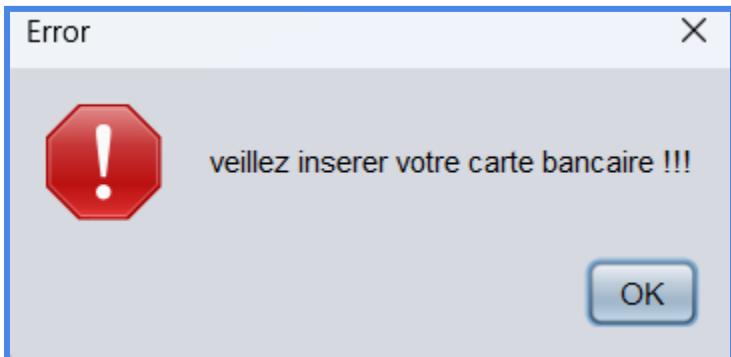
montant de penalite :

num carte bancaire :



PAYER

si champ de num de carte bancaire est vide :



CODE :

```
if(num_carte_bancaire_field.getText().isEmpty()){
    JOptionPane.showMessageDialog(null, " veillez inserer votre carte bancaire !!! ", "Error", JOptionPane.ERROR_MESSAGE);
}else{
    String carte=num_carte_bancaire_field.getText();
    String sql="INSERT INTO penalite VALUE(?, ?, ?)";
    try {
        PreparedStatement pstmt = con.prepareStatement(sql);
        pstmt.setString(1, this.id_location);
        pstmt.setDouble(2, montant_penalite);
        pstmt.setString(3, carte);
        pstmt.executeUpdate();
        if(days_en_retard>=5){
            JOptionPane.showMessageDialog(null, " Malheureusement , Vous avez enfreint nos règles et vous serez banni(e) !!! ", "Error", JOptionPane.ERROR_MESSAGE);
            //inserer le client dans le table banned
            sql="INSERT INTO banned VALUE (?)";
            pstmt.clearParameters();
            pstmt=con.prepareStatement(sql);
            pstmt.setString(1, this.client.getCin());
            pstmt.executeUpdate();
            //close all
            // Close all internal frames
            JInternalFrame[] frames = jDesktopPanel.getAllFrames();
            for (JInternalFrame frame : frames) {
                frame.dispose(); // Close the internal frame
            }
            // Get the parent frame and dispose of it
            Window parentWindow = SwingUtilities.windowForComponent(jDesktopPanel);
            parentWindow.dispose();
            MainPage log=new MainPage();//thel wahda jdida
            log.setLocationRelativeTo(null);
            log.setVisible(true);//tkhalifa todher
        }else{
            // Close all internal frames
            JInternalFrame[] frames = jDesktopPanel.getAllFrames();
            for (JInternalFrame frame : frames) {
```

Activer Windows

Diagramme De Classes

