

RAPPORT DE PROJET

**Application de gestion de location de
robes soirée**

Mai 2025

TABLE DES MATIÈRES

1. Introduction	1
le contexte	2
les besoins fonctionnels	3
les besoins non fonctionnels	4
2. Analyse des besoins	5
Modèle entité association	6
Modèle Relationnel	6
Requêtes SQL	7
PL/SQL	11
3. Analyse de domaine de conception	15
Diagramme de cas d'utilisation	16
4. Réalisation	17
La distribution des tâches	18
Les requêtes SQL Utilisées en Java	19
Interface java	33

INTRODUCTION

INTRODUCTION

LE CONTEXTE

La gestion des locations de robes de soirée au sein d'un magasin peut rapidement devenir complexe lorsqu'elle est effectuée manuellement. Les informations relatives aux clientes, aux robes disponibles, aux dates de location, aux tarifs et aux paiements sont nombreuses et doivent être constamment mises à jour pour garantir un bon fonctionnement de l'activité.

Actuellement, les procédures de gestion sont pour la plupart manuelles : les réservations se font sur papier, les fiches des clientes sont classées dans des dossiers physiques, et la disponibilité des robes est vérifiée à la main. Cette méthode entraîne plusieurs problèmes majeurs :

- Risque de perte ou de détérioration des documents.
- Difficulté à retrouver rapidement les informations.
- Mauvais archivage ou erreurs dans les réservations.
- Retards dans le traitement des demandes des clientes.
- Perte de temps considérable pour les employés.

Face à ces difficultés, nous avons envisagé la conception et le développement d'une application informatique dédiée à la gestion de la location de robes de soirée. L'objectif est de faciliter le travail des gérants et employés en automatisant les différentes tâches liées à la gestion de la boutique.

Deux espaces d'utilisation seront proposés :

- Un espace administrateur pour gérer l'ensemble des fonctionnalités (clients, robes, locations, facturation, etc.).
- Un espace employé, accessible via un compte configuré par l'administrateur, avec des priviléges limités selon le rôle de l'utilisateur.

INTRODUCTION

BESOINS FONCTIONNELS

Les besoins fonctionnels de notre application de gestion de location de robes de soirée se répartissent en six grandes catégories :

1. Authentification :

L'application devra permettre à chaque utilisateur (administrateur ou employé) d'accéder à son espace personnel après identification via un nom d'utilisateur et un mot de passe.

2. Recherche rapide :

Cette fonctionnalité permet aux utilisateurs d'effectuer une recherche par mot-clé sur les clientes, les robes ou les réservations.

3. Gestion des clientes :

Permet d'ajouter de nouvelles clientes, de modifier ou supprimer leurs informations, et de consulter leur historique de location.

4. Gestion des robes :

Cette fonctionnalité offre la possibilité de gérer l'inventaire des robes (ajout de nouveaux modèles, modification des caractéristiques comme la taille, la couleur, l'état, ou la disponibilité, suppression de robes hors service...).

5. Gestion des locations :

L'utilisateur pourra enregistrer de nouvelles locations ou retourner des robes. Il sera également possible de gérer les dates de début et de fin, les paiements associés.

6. Gestion des factures :

Cette fonctionnalité permet de générer automatiquement une facture numérique lors de la confirmation d'une location, et d'enregistrer ou modifier les paiements.

INTRODUCTION

BESOINS NON FONCTIONNELS

Ces besoins concernent la qualité du système et influencent fortement l'expérience utilisateur :

1. Fiabilité :

L'application doit fonctionner de manière stable et cohérente, sans interruption ni perte de données.

2. Gestion des erreurs :

En cas d'action incorrecte ou de donnée invalide, des messages d'erreurs clairs doivent guider l'utilisateur pour corriger l'erreur.

3. Ergonomie et interface intuitive :

L'application doit être facile à utiliser, avec une interface conviviale, bien organisée, et une navigation fluide. Les couleurs et la typographie doivent être harmonieuses et lisibles.

4. Sécurité :

Les données personnelles des clientes (coordonnées, historique de location, etc.) doivent être protégées. L'accès à l'application sera sécurisé par authentification et les rôles bien définis.

5. Maintenance et évolutivité :

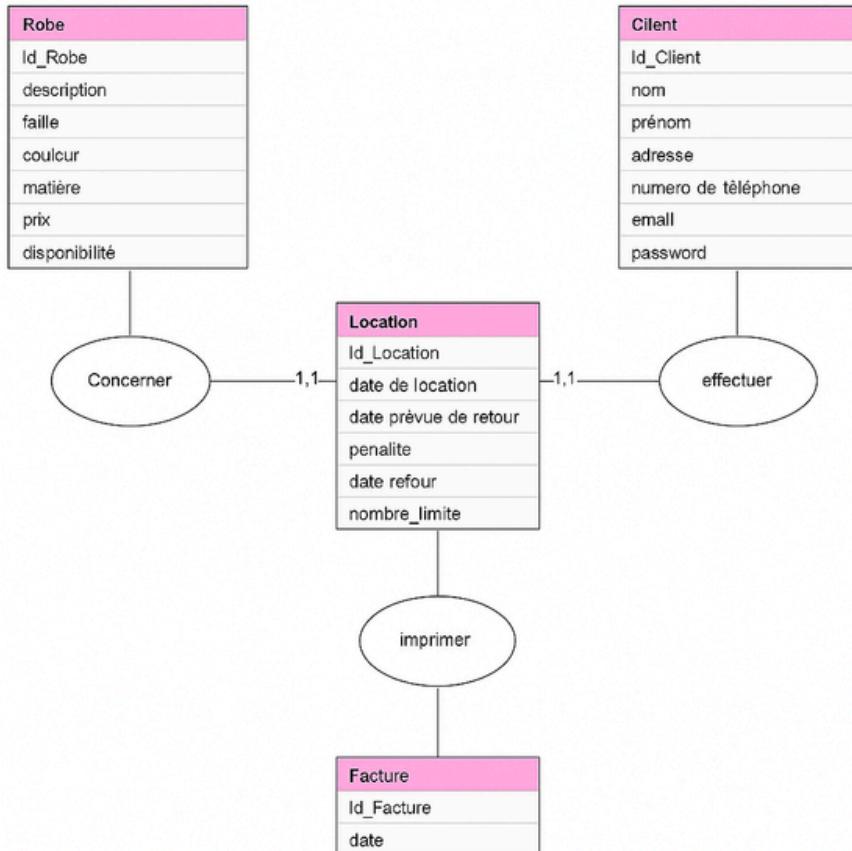
Le système devra être conçu avec une architecture modulaire et bien documentée, facilitant les corrections, les mises à jour et l'ajout futur de nouvelles fonctionnalités.

6. Compatibilité et portabilité :

L'application doit être accessible sur différents systèmes d'exploitation (Windows, Linux, etc.) et adaptable à divers supports (ordinateurs de bureau, portables).

ANALYSE DES BESOINS

MODÈLE ENTITÉ ASSOCIATION / MODÈLE RELATIONNEL



Modèle relationnel :

Client(id_client, nom, prenom , adresse , num_tel, email , mdp)
Robe(id_robe, description, taille, couleur, matière, prix, disponibilité)
Location(id_location, #id_client, #id_robe, date_location,
date_prevu_retour, pénalité)
Facture (id_facture, #id_location, date)

REQUÊTES SQL

```
DROP TABLE clients CASCADE CONSTRAINTS;

CREATE TABLE clients (
    id NUMBER GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,
    nom VARCHAR2(50) NOT NULL,
    prenom VARCHAR2(50) UNIQUE NOT NULL ,
    adresse VARCHAR2(255)NOT NULL,
    email VARCHAR2(100) UNIQUE NOT NULL,
    tel VARCHAR2(20)NOT NULL,
    mdp VARCHAR2(20) NOT NULL
);
```

Supprimer table 'clients' si elle existe déjà
crée une nouvelle table 'clients' :

- colonne pour l'ID client auto-incrémentée
- colonne pour le nom du client, ne peut pas être vide
- colonne pour le prénom du client, unique (utilisée comme username)
- colonne pour l'adresse du client, ne peut pas être vide
- colonne pour l'adresse email, unique
- colonne pour le mot de passe

la clé primaire est l'ID du client

REQUÊTES SQL

```
DROP TABLE robes CASCADE CONSTRAINTS;
```

```
CREATE TABLE robes (
    id NUMBER GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,
    description VARCHAR2(255),
    taille VARCHAR2(50),
    couleur VARCHAR2(50),
    matiere VARCHAR2(100),
    prix NUMBER(10,2),
    disponibilite VARCHAR2(50)
);
```

Supprimer table 'robes' si elle existe déjà
crée une nouvelle table 'robes' :

- colonne pour l'ID robe auto-incrémentée
- colonne pour la description de la robe (longue, midi, courte)
- colonne pour la taille de la robe (XS, S, M, L, XL ...)
- colonne pour la matière de la robe
- colonne pour le prix de la robe avec 2 chiffres après la virgule
- colonne pour la disponibilité de la robe ("Oui" ou "Non")

la clé primaire est l'ID de la robe

REQUÊTES SQL

```
DROP TABLE locations CASCADE CONSTRAINTS;

CREATE TABLE locations (
    id_location NUMBER PRIMARY KEY,
    id_client NUMBER,
    id_robe NUMBER,
    date_location DATE,
    date_retour_due DATE,
    date_retour DATE,
    nombre_limite NUMBER,
    penalite FLOAT
);

ALTER TABLE locations
ADD CONSTRAINT fk_client FOREIGN KEY (id_client) REFERENCES clients(id);

ALTER TABLE locations
ADD CONSTRAINT fk_robe FOREIGN KEY (id_robe) REFERENCES robes(id);
```

Supprimer table 'locations' si elle existe déjà
crée une nouvelle table 'locations' :

- colonne pour l'ID location auto-incrémentée
- colonne pour l'ID client
- colonne pour l'ID robe
- colonne pour la date de location de la robe
- colonne pour la date prévues de retour de la robe
- colonne pour la date de retour
- colonne pour le nombre limite de location
- colonne pour la pénalité de la cliente

La clé primaire est l'ID de la location

Les clés étrangères sont :

- L'ID de la cliente
- L'ID de la robe

REQUÊTES SQL

```
DROP TABLE Factures CASCADE CONSTRAINTS;

CREATE TABLE Factures (
    idFacture NUMBER GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,
    idLocation NUMBER,
    FOREIGN KEY (idLocation) REFERENCES Locations(id_location)
);
```

Supprimer table 'FACTURES' si elle existe déjà
crée une nouvelle table 'factures':

- colonne pour l'ID factures auto-incrémentée
- colonne pour l'ID locations référence à la colonne id_locations dans table Locations

la clé primaire est l'ID de la facture

PL/SQL

```
CREATE OR REPLACE PROCEDURE verifier_robés_disponibles (
    p_taille IN VARCHAR2,
    p_couleur IN VARCHAR2,
    p_description IN VARCHAR2,
    p_result OUT SYS_REFCURSOR
) AS
BEGIN
    OPEN p_result FOR
        SELECT r.id, r.taille, r.couleur, r.prix, l.date_location
        FROM robes r
        LEFT JOIN locations l ON r.id = l.id_robe
        WHERE r.taille = p_taille
            AND r.couleur = p_couleur
            AND r.description = p_description
            AND (l.id_robe IS NULL OR l.date_retour IS NOT NULL);
END;
/

SET SERVEROUTPUT ON;

DECLARE
    v_cursor SYS_REFCURSOR;
    v_id robes.id%TYPE;
    v_taille robes.taille%TYPE;
    v_couleur robes.couleur%TYPE;
    v_prix robes.prix%TYPE;
    v_date_location locations.date_location%TYPE;
BEGIN
    verifier_robés_disponibles('M', 'Rouge', 'Longue', v_cursor);

    LOOP
        FETCH v_cursor INTO v_id, v_taille, v_couleur, v_prix, v_date_location;
        EXIT WHEN v_cursor%NOTFOUND;
        DBMS_OUTPUT.PUT_LINE('ID: ' || v_id || ', Taille: ' || v_taille ||
                            ', Couleur: ' || v_couleur || ', Prix: ' || v_prix ||
                            ', Date location: ' || v_date_location);
    END LOOP;

    CLOSE v_cursor;
END;
/
```

PL/SQL

La procédure `verifier_robés_disponibles` permet de rechercher les robes disponibles selon des critères spécifiques et une date de location donnée. Elle accepte cinq paramètres : la matière de robe, la taille, la couleur, la date de location souhaitée, et un curseur de sortie pour retourner les résultats.

Le fonctionnement interne de la procédure repose sur une requête SQL qui sélectionne toutes les robes correspondant aux critères fournis et qui ne sont pas déjà réservées pour la date spécifiée. La vérification des réservations existantes se fait en examinant la table des locations pour s'assurer que la date demandée ne se situe pas entre les dates de début et de retour d'une location existante pour chaque robe.

PL/SQL

```
CREATE OR REPLACE TRIGGER check_robe_availability
BEFORE INSERT OR UPDATE ON location
FOR EACH ROW
DECLARE
    v_overlap_count NUMBER;
BEGIN
    SELECT COUNT(*)
    INTO v_overlap_count
    FROM locations l
    WHERE l.id_robe = :new.id_robe
    AND (
        (:new.date_location BETWEEN l.date_location AND l.date_retour)
        OR (:new.date_retour BETWEEN l.date_location AND l.date_retour)
        OR (l.date_location BETWEEN :new.date_location AND :new.date_retour)
        OR (l.date_retour BETWEEN :new.date_location AND :new.date_retour)
    )
    AND l.id_location <> NVL(:old.id_location, 0);

    IF v_overlap_count > 0 THEN
        RAISE_APPLICATION_ERROR(-20005, 'La robe est déjà louée pour cette période.');
    END IF;
END;
```

PL/SQL

Pour compléter le système de gestion des locations de robes, un déclencheur a été créé pour garantir l'intégrité des données. Ce déclencheur s'active automatiquement avant chaque insertion ou mise à jour dans la table des locations.

Le déclencheur fonctionne en vérifiant systématiquement s'il existe des locations en conflit pour la même robe. Il examine scénarios possibles de chevauchement de dates :

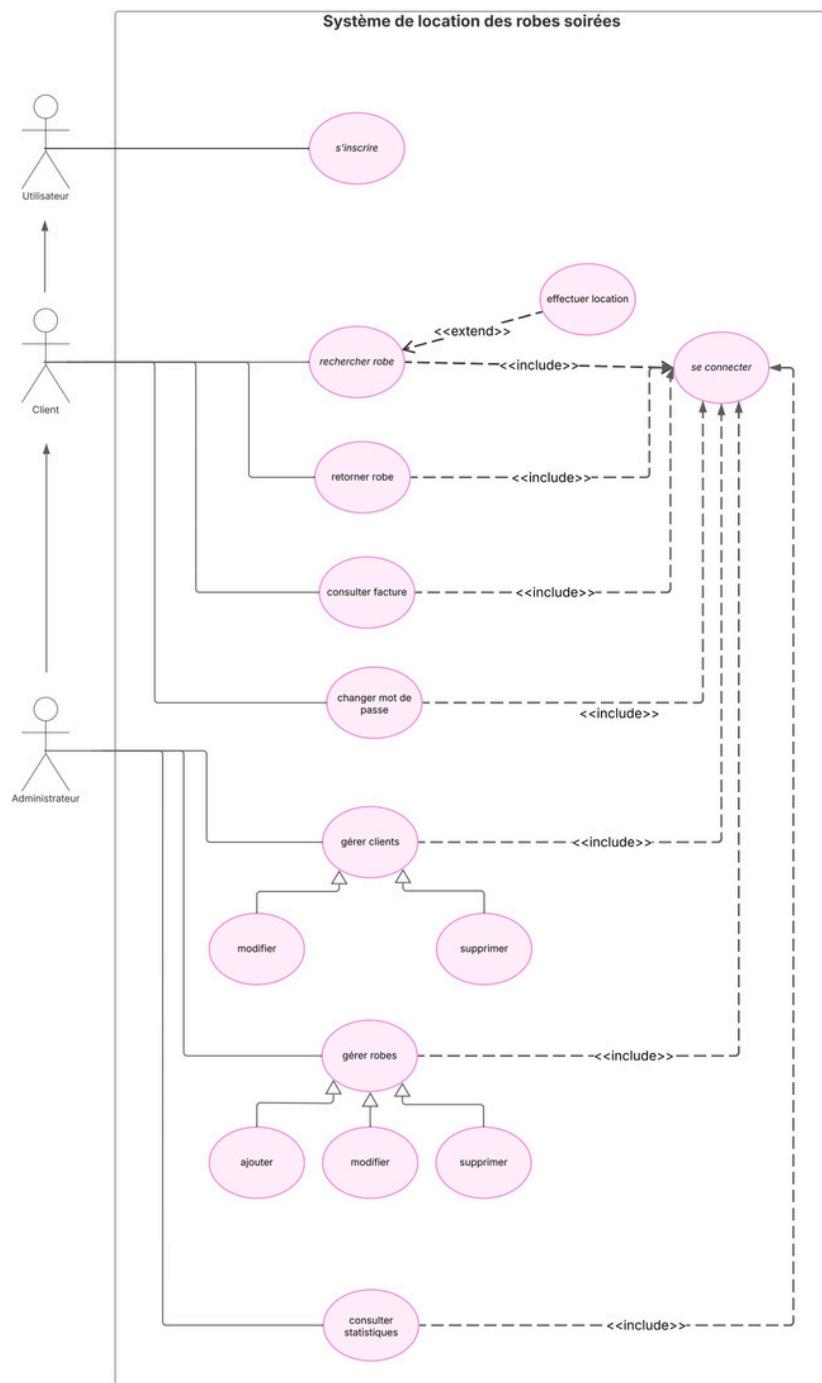
- lorsque la nouvelle date de début se situe dans une période déjà réservée
- lorsque la nouvelle date de retour tombe pendant une location existante
- lorsque les dates d'une location existante s'étendent sur la nouvelle période demandée.

Une attention particulière est portée aux opérations de mise à jour en excluant la location actuellement modifiée de cette vérification.

Si un conflit est détecté, le déclencheur interrompt l'opération en cours et renvoie un message d'erreur clair indiquant que la robe n'est pas disponible pour la période demandée. Cette approche préventive permet d'éviter les doubles réservations et garantit la cohérence des données dans le système.

ANALYSE DE DOMAINE DE CONCEPTION

DIAGRAMME DE CAS D'UTILISATION



RÉALISATION

LA DISTRIBUTION DES TÂCHES

Rached Sima :

Diagramme de cas d'utilisation

diagramme d'entité/association +modèle relationnel

interface de page welcome

interface de page signup

interface de page Changer le mot de passe

Rapport de projet

Msaddek Ilef :

diagramme d'entité/association

interface de page espace client

interface de page espace admin

interface de page gestion clients

interface de page gestion robes

implémentation des classes

Classe connexion de la base données avec le jdbc

Le débogage

Frikha Nadia :

Design des interfaces graphique en 'Canva'

interface de page login

interface de page statistiques

procédure pl/sql

rapport de projet

Tili Maha:

interface de page location robe

interface de page retour robe

implementation de classe facture

Le débogage

LES REQUÊTES SQL UTILISÉES EN JAVA CONNECTION

```
public class ConnectionProvider {

    public static Connection orclConnection() {
        try {
            Class.forName("oracle.jdbc.driver.OracleDriver");

            Connection conn = DriverManager.getConnection(
                "jdbc:oracle:thin:@localhost:1521/XEPDB1", "HR", "hr"
            );
            return conn;
        }
        catch (ClassNotFoundException | SQLException e) {
            System.out.println(e);
        }

        return null;
    }

    public static ResultSet operations(String query, String msg) {
        Connection connection = null;
        ResultSet rs = null;
        connection = orclConnection();

        try {
            if (connection != null) {
                Statement st = connection.createStatement();
                rs = st.executeQuery(query);
                if (!msg.equals(""))
                    JOptionPane.showMessageDialog(null, msg);
            }
        }

        } else {
            JOptionPane.showMessageDialog(null, "Database connection failed.");
        }
    catch (Exception e) {
        JOptionPane.showMessageDialog(null, "Error: " + e.getMessage(), "Error", JOptionPane.ERROR_MESSAGE);
    }
    return rs;
}
}
```

LES REQUÊTES SQL UTILISÉES EN JAVA

SIGN UP

```
try {
    // Load the JDBC driver (for Oracle, use: oracle.jdbc.OracleDriver)
    Class.forName("oracle.jdbc.OracleDriver");

    // Prepare SQL statement
    try { // Connect to the database
        Connection con = DriverManager.getConnection(
            "jdbc:oracle:thin:@//localhost:1521/XEPDB1","HR", "hr")) (
    // Prepare SQL statement
    String query = "INSERT INTO clients (nom, prenom, adresse, email, tel, mdp) VALUES (?, ?, ?, ?, ?, ?)";
    PreparedStatement pst = con.prepareStatement(query);

    pst.setString(1, nom);
    pst.setString(2, prenom);
    pst.setString(3, adresse);
    pst.setString(4, email);
    pst.setString(5, tel);
    pst.setString(6, motDePasse);

    // Execute update
    pst.executeUpdate();

    JOptionPane.showMessageDialog(this, "Inscription réussie!");
    txtnom.setText("");
    txtprenom.setText("");
    txtadr.setText("");
    txtmdp.setText("");
    txttel.setText("");
    txtemail.setText("");

    pst.close();
}

} catch (ClassNotFoundException e) {
    e.printStackTrace();
    JOptionPane.showMessageDialog(this, "Erreur lors de l'inscription: " + e.getMessage());
} catch (SQLException ex) {
    Logger.getLogger(signupinterface.class.getName()).log(Level.SEVERE, null, ex);
}

private boolean emailExists(String email) {
    boolean exists = false;
    try {
        Connection con = DriverManager.getConnection(
            "jdbc:oracle:thin:@//localhost:1521/XEPDB1", "HR", "hr");
        String query = "SELECT COUNT(*) FROM clients WHERE email = ?";
        PreparedStatement pst = con.prepareStatement(query);
        pst.setString(1, email);
        ResultSet rs = pst.executeQuery();
        if (rs.next()) {
            exists = rs.getInt(1) > 0;
        }
        rs.close();
        pst.close();
        con.close();
    } catch (SQLException ex) {
        ex.printStackTrace();
    }
    return exists;
}
```

LES REQUÊTES SQL UTILISÉES EN JAVA LOGIN

```
private void btnLoginActionPerformed(java.awt.event.ActionEvent evt) {  
    // TODO add your handling code here:  
    String email = txtEmail.getText().trim();  
    String mdp = new String(txtmdp.getPassword()).trim();  
    String admin = txtAdmin.getText().trim();  
  
    try {  
        java.sql.Connection con = ConnectionProvider.getConnection();  
        String query = "SELECT * FROM clients WHERE email = ? AND mdp = ?";  
        PreparedStatement ps = con.prepareStatement(query);  
        ps.setString(1, email);  
        ps.setString(2, mdp);  
  
        ResultSet rs = ps.executeQuery();  
        if (rs.next()) {  
            // Vérifie le code admin  
            if (admin.equals("1234")) {  
                JOptionPane.showMessageDialog(this, "Bienvenue Admin ! Redirection vers la page Admin.", "Admin", JOptionPane.INFORMATION_MESSAGE);  
                setVisible(false);  
                new mainAdminInterface().setVisible(true);  
  
            } else {  
                JOptionPane.showMessageDialog(this, "Connexion réussie ! Redirection vers la page Client.", "Client", JOptionPane.INFORMATION_MESSAGE);  
                setVisible(false);  
                new mainInterface().setVisible(true);  
            }  
        } else {  
            JOptionPane.showMessageDialog(this, "Email ou mot de passe incorrect", "Erreur", JOptionPane.ERROR_MESSAGE);  
        }  
  
        rs.close();  
        ps.close();  
        con.close();  
    } catch (Exception e) {  
        JOptionPane.showMessageDialog(this, "Erreur de connexion à la base : " + e.getMessage(), "Erreur", JOptionPane.ERROR_MESSAGE);  
    }  
}
```

Activate Wind

LES REQUÊTES SQL UTILISÉES EN JAVA CHANGER MDP

```
String ancienMDP = new String (txtamdp.getPassword()).trim();
String nouveauMDP = new String (txtnmdp.getPassword()).trim();
String confirmationMDP = new String (txtcmdp.getPassword()).trim();

if (!nouveauMDP.equals(confirmationMDP)) {
    JOptionPane.showMessageDialog(null, "Le mot de passe de confirmation ne correspond pas.");
    return;
}

if (nouveauMDP.length() < 8) {
    JOptionPane.showMessageDialog(null, "Le nouveau mot de passe doit contenir au moins 8 caractères.");
    return;
}

try {
    Class.forName("oracle.jdbc.OracleDriver");
    try (Connection con = DriverManager.getConnection(
        "jdbc:oracle:thin:@//localhost:1521/XEPDB1","HR","hr")) {

        // Vérification de l'ancien mot de passe
        String checkQuery = "SELECT * FROM clients WHERE email = ? AND mdp = ?";
        try (PreparedStatement checkStmt = con.prepareStatement(checkQuery)) {
            checkStmt.setString(1, emailUtilisateur);
            checkStmt.setString(2, ancienMDP);
            try (ResultSet rs = checkStmt.executeQuery()) {
                if (rs.next()) {
                    // Mise à jour du mot de passe
                    String updateQuery = "UPDATE clients SET mdp = ? WHERE email = ?";
                    try (PreparedStatement updateStmt = con.prepareStatement(updateQuery)) {
                        updateStmt.setString(1, nouveauMDP);
                        updateStmt.setString(2, emailUtilisateur);
                        updateStmt.executeUpdate();

                        JOptionPane.showMessageDialog(null, "Mot de passe modifié avec succès.");

                } else {
                    JOptionPane.showMessageDialog(null, "Ancien mot de passe incorrect.");
                }
            }
        }
    }
} catch (HeadlessException | ClassNotFoundException | SQLException e) {
    e.printStackTrace();
    JOptionPane.showMessageDialog(null, "Erreur : " + e.getMessage());
}
```

LES REQUÊTES SQL UTILISÉES EN JAVA GESTION DES ROBES

```
public static void creerTable() {
    try {
        java.sql.Connection con = orclConnection();
        if (con == null) return;

        DatabaseMetaData dbm = con.getMetaData();
        ResultSet tables = dbm.getTables(null, null, "ROBES", null);

        if (!tables.next()) {
            String robeTable = "CREATE TABLE robes (" +
                "id NUMBER GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY, " +
                "description VARCHAR2(50), " +
                "taille VARCHAR2(50), " +
                "couleur VARCHAR2(50), " +
                "matiere VARCHAR2(100), " +
                "prix NUMBER(10,2), " +
                "disponibilite VARCHAR2(3))"; // Disponibilite "Oui" ou "Non"
            ConnectionProvider.operations(robeTable, "robeTable created successfully");
            insererRobes();
        }
    } catch (Exception e) {
        JOptionPane.showMessageDialog(null, "Erreur: " + e.getMessage(), "Erreur", JOptionPane.ERROR_MESSAGE);
    }
}

public static void insererRobes() {
    try {
        String q1 = "INSERT INTO robes (description, taille, couleur, matiere, prix, disponibilite) VALUES " +
                    "('Longue', 'M', 'Rouge', 'Soie', 199.99, 'Oui')";
        String q2 = "INSERT INTO robes (description, taille, couleur, matiere, prix, disponibilite) VALUES " +
                    "('Longue', 'L', 'Noir', 'Dentelle', 249.50, 'Oui')";
        String q3 = "INSERT INTO robes (description, taille, couleur, matiere, prix, disponibilite) VALUES " +
                    "('Courte', 'S', 'Bleu', 'Satine', 179.00, 'Non')";
        String q4 = "INSERT INTO robes (description, taille, couleur, matiere, prix, disponibilite) VALUES " +
                    "('Midi', 'M', 'Doré', 'Polyester', 299.99, 'Oui')";
        String q5 = "INSERT INTO robes (description, taille, couleur, matiere, prix, disponibilite) VALUES " +
                    "('Midi', 'L', 'Vert', 'Coton', 189.75, 'Oui')";

        ConnectionProvider.operations(q1, "");
        ConnectionProvider.operations(q2, "");
        ConnectionProvider.operations(q3, "");
        ConnectionProvider.operations(q4, "");
        ConnectionProvider.operations(q5, "");

    } catch (Exception e) {
        JOptionPane.showMessageDialog(null, "Erreur: " + e.getMessage(), "Erreur", JOptionPane.ERROR_MESSAGE);
    }
}
```

LES REQUÊTES SQL UTILISÉES EN JAVA GESTION DES ROBES

```
public void save(){
String query = "INSERT INTO robes (description, taille, couleur, matiere, prix, disponibilite) " +
    "VALUES (" + description + ", " + taille + ", " + couleur + ", " + matiere + ", " + prix + ", " + disponibilite + ")";
try{
    ConnectionProvider.operations(query, "Registered successfully");
}
catch (Exception e) {
    JOptionPane.showMessageDialog(null, "Error: " + e.getMessage(), "Error", JOptionPane.ERROR_MESSAGE);
}

public void update(){
String query = "UPDATE robes SET description='"+ description + "', prix='"+ prix + "', matiere='"+ matiere +"', couleur='"+ couleur + "'";
try{
    ConnectionProvider.operations(query, "updated successfully");
}
catch (Exception e) {
    JOptionPane.showMessageDialog(null, "Error: " + e.getMessage(), "Error", JOptionPane.ERROR_MESSAGE);
}

public void delete(){
String query = "delete from robes where id ='" + id + "' ";
try{
    ConnectionProvider.operations(query, "deleted successfully");
}
catch (Exception e) {
    JOptionPane.showMessageDialog(null, "Error: " + e.getMessage(), "Error", JOptionPane.ERROR_MESSAGE);
}

public static void inserer() {
try {
    String q1 = "INSERT INTO clients (nom, prenom, adresse, email, tel, password) VALUES " +
        "('Ben Mahmoud', 'Leila', 'Rue des Jasmins, Sfax', 'leila.bm@email.com', '987654321', 'pass123')";
    String q2 = "INSERT INTO clients (nom, prenom, adresse, email, tel, password) VALUES " +
        "('Al Fassi', 'Samira', 'Avenue Habib Bourguiba, Tunis', 'samira.fassi@email.com', '53426789', 'samira456')";
    String q3 = "INSERT INTO clients (nom, prenom, adresse, email, tel, password) VALUES " +
        "('Cherif', 'Yasmina', 'Rue Ibn Khaldoun, Kairouan', 'yasmina.cherif@email.com', '95674321', 'yas789')";
    String q4 = "INSERT INTO clients (nom, prenom, adresse, email, tel, password) VALUES " +
        "('Zouari', 'Nour', 'Quartier Ennassim, Sousse', 'nour.zouari@email.com', '54329876', 'nour321')";
    String q5 = "INSERT INTO clients (nom, prenom, adresse, email, tel, password) VALUES " +
        "('Bensalem', 'Hiba', 'Route de La Marsa, Tunis', 'hiba.bensalem@email.com', '96543210', 'hiba654')";

    ConnectionProvider.operations(q1, "");
    ConnectionProvider.operations(q2, "");
    ConnectionProvider.operations(q3, "");
    ConnectionProvider.operations(q4, "");
    ConnectionProvider.operations(q5, "");

} catch (Exception e) {
    JOptionPane.showMessageDialog(null, "Erreur: " + e.getMessage(), "Erreur", JOptionPane.ERROR_MESSAGE);
}

public static int getRobeId(String Robe) {
    for (Robe r : new Robe().getAllRecords()) {
        if (r.description.equals(Robe)) {
            return r.id;
        }
    }
    return -1; // Si non trouvée
}
```

LES REQUÊTES SQL UTILISÉES EN JAVA GESTION DES CLIENTS

```
public static void creerTable() {
    try {
        java.sql.Connection con = orclConnection();
        if (con == null) return;

        DatabaseMetaData dbm = con.getMetaData();
        ResultSet tables = dbm.getTables(null, null, "CLIENTS", null);

        if (!tables.next()) {
            String clientTable = "CREATE TABLE clients (" +
                "id NUMBER GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY, " +
                "nom VARCHAR2(50), " +
                "prenom VARCHAR2(50), " +
                "adresse VARCHAR2(255), " +
                "email VARCHAR2(100), " +
                "tel VARCHAR2(20), " +
                "password VARCHAR2(100))";
            ConnectionProvider.operations(clientTable, "clients table created successfully");
            inserer();
        }
    } catch (Exception e) {
        JOptionPane.showMessageDialog(null, "Erreur: " + e.getMessage(), "Erreur", JOptionPane.ERROR_MESSAGE);
    }
}

public void update() {
    String query = "UPDATE clients SET nom='" + nom + "', prenom='" + prenom + "', adresse='" + adresse +
        "', email='" + email + "', tel='" + tel + "' WHERE id=" + id;
    try {
        ConnectionProvider.operations(query, "updated successfully");
    } catch (Exception e) {
        JOptionPane.showMessageDialog(null, "Erreur: " + e.getMessage(), "Erreur", JOptionPane.ERROR_MESSAGE);
    }
}

public void delete() {
    String query = "DELETE FROM clients WHERE id=" + id;
    try {
        ConnectionProvider.operations(query, "deleted successfully");
    } catch (Exception e) {
        JOptionPane.showMessageDialog(null, "Erreur: " + e.getMessage(), "Erreur", JOptionPane.ERROR_MESSAGE);
    }
}

public static int getClientId(String prenom) {
int id = -1;
try {
    String query = "SELECT id FROM clients WHERE prenom=?";
    java.sql.Connection connection = ConnectionProvider.orclConnection();
    PreparedStatement statement = connection.prepareStatement(query);
    statement.setString(1, prenom);
    ResultSet rs = statement.executeQuery();
    if (rs.next()) {
        id = rs.getInt("id");
    }
} catch (Exception e) {
    JOptionPane.showMessageDialog(null, "Erreur lors de la récupération de l'ID du client : " + e.getMessage());
}
return id;
}
```

LES REQUÊTES SQL UTILISÉES EN JAVA STATISTIQUES

```
String idClient = jTextField1.getText();
if (idClient.isEmpty()) {
    JOptionPane.showMessageDialog(this, "Veuillez saisir l'ID de la cliente.");
    return;
}

try {
    Connection con = DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe", "HR", "hr");
    String query = "SELECT r.id_robe, r.description, r.taille, r.couleur, r.matiere, r.prix , r.disponibilite" +
        "FROM location l JOIN robes r ON l.id_robe = r.id_robe " +
        "WHERE l.id_client = ?";
    PreparedStatement pst = con.prepareStatement(query);
    pst.setString(1, idClient);
    ResultSet rs = pst.executeQuery();

    DefaultTableModel model = (DefaultTableModel) jTable1.getModel();
    model.setRowCount(0); // vider la table

    boolean found = false;

    while (rs.next()) {
        found = true;
        model.addRow(new Object[]{
            rs.getInt("id_robe"),
            rs.getString("description"),
            rs.getString("taille"),
            rs.getString("couleur"),
            rs.getString("matiere"),
            rs.getDouble("prix"),
            rs.getString("disponibilite"),
            "" // pas de "nb" ici
        });
    }
}

if (!found) {
    JOptionPane.showMessageDialog(this, "Cette cliente n'a loué aucune robe.");
}

con.close();
} catch (Exception e) {
    JOptionPane.showMessageDialog(this, "Erreur : " + e.getMessage());
}
```

LES REQUÊTES SQL UTILISÉES EN JAVA STATISTIQUES

```
try {
Connection con = DriverManager.getConnection("jdbc:oracle:thin:@//localhost:1521/XEPDB1", "HR", "hr");
String query = "SELECT r.id_robe, r.description, r.taille, r.couleur, r.matiere, r.prix,r.disponibilite, COUNT(*) as nb " +
    "FROM location l JOIN robes r ON l.id_robe = r.id_robe " +
    "GROUP BY r.id_robe, r.description, r.taille, r.couleur, r.matiere, r.prix " +
    "ORDER BY nb DESC";
Statement st = con.createStatement();
ResultSet rs = st.executeQuery(query);

DefaultTableModel model = (DefaultTableModel) jTable1.getModel();
model.setRowCount(0); // vider la table

while (rs.next()) {
    model.addRow(new Object[]{
        rs.getInt("id_robe"),
        rs.getString("description"),
        rs.getString("taille"),
        rs.getString("couleur"),
        rs.getString("matiere"),
        rs.getDouble("prix"),
        rs.getString("disponibilite"),
        rs.getInt("nb")
    });
}

con.close();
} catch (Exception e) {
    JOptionPane.showMessageDialog(this, "Erreur : " + e.getMessage());
}

try {
Connection con = DriverManager.getConnection("jdbc:oracle:thin:@//localhost:1521/XEPDB1", "HR", "hr");
String query = "SELECT id, description, taille, couleur, matiere, prix, disponibilite FROM robes";
Statement st = con.createStatement();
ResultSet rs = st.executeQuery(query);

DefaultTableModel model = (DefaultTableModel) jTable1.getModel();
model.setRowCount(0); // Vider la table avant d'ajouter les nouvelles données

while (rs.next()) {
    model.addRow(new Object[] {
        rs.getInt("id"),
        rs.getString("description"),
        rs.getString("taille"),
        rs.getString("couleur"),
        rs.getString("matiere"),
        rs.getDouble("prix"),
        rs.getString("disponibilite"),
        "" // colonne "nb" vide ici car ce n'est pas pertinent dans cette vue
    });
}

con.close();
} catch (Exception e) {
    JOptionPane.showMessageDialog(this, "Erreur : " + e.getMessage());
}
```

LES REQUÊTES SQL UTILISÉES EN JAVA STATISTIQUES

```
try {
Connection con = DriverManager.getConnection("jdbc:oracle:thin:@//localhost:1521/XEPDB1", "HR", "hr");

String query = "SELECT r.id, r.description, r.taille, r.couleur, r.matiere, r.prix " +
    "FROM robes r " +
    "WHERE r.id IN (SELECT l.id_robe FROM location l)";

Statement st = con.createStatement();
ResultSet rs = st.executeQuery(query);

DefaultTableModel model = (DefaultTableModel) jTable1.getModel();
model.setRowCount(0); // vider la table

while (rs.next()) {
    model.addRow(new Object[] {
        rs.getInt("id"),
        rs.getString("description"),
        rs.getString("taille"),
        rs.getString("couleur"),
        rs.getString("matiere"),
        rs.getDouble("prix"),
        "", // disponibilité vide
        "" // nb vide
    });
}

con.close();
} catch (Exception e) {
    JOptionPane.showMessageDialog(this, "Erreur : " + e.getMessage());
}
```

LES REQUÊTES SQL UTILISÉES EN JAVA LOUER ROBE

```
public List <String> get Client() {
    List<String> clientNames = new ArrayList<>();
    String query = "SELECT prenom FROM clients";

    try (Connection connection = ConnectionProvider.orclConnection();
        Statement statement = connection.createStatement();
        ResultSet resultSet = statement.executeQuery(query)) {

        // Loop through the result set and add client names to the list
        while (resultSet.next()) {
            clientNames.add(resultSet.getString("prenom"));
        }

    } catch (SQLException e) {
        System.out.println("erreur de connexion");

    }
    return clientNames;
}

public static List<String> get_robès() {
    List<String> Robe_disc = new ArrayList<>();
    String query = "SELECT name FROM robes";

    try (Connection connection = ConnectionProvider.orclConnection();
        Statement statement = connection.createStatement();
        ResultSet resultSet = statement.executeQuery(query)) {

        // Loop through the result and add Robe discription to the list
        while (resultSet.next()) {
            Robe_disc.add(resultSet.getString("name"));
        }

    } catch (SQLException e) {
        // Handle exceptions
    }
    return Robe_disc;
}
```

LES REQUÊTES SQL UTILISÉES EN JAVA LOUER ROBE

```
// Vérifier si le client a déjà effectué une location
String checkSql = "SELECT COUNT(*) FROM locations WHERE id_client = ?";
boolean isNewClient = false;

try (PreparedStatement checkStmt = conn.prepareStatement(checkSql)) {
    checkStmt.setInt(1, idClient);
    try (ResultSet checkRs = checkStmt.executeQuery()) {
        if (checkRs.next()) {
            isNewClient = checkRs.getInt(1) == 0; // si 0 => nouveau client
        }
    }
}

// Si nouveau client - insérer directement
if (isNewClient) {
    String insertSql = "INSERT INTO locations (id_location, id_client, id_robe, date_location, date_retour_due, nombre_limite) VALUES (location_seq.nextval, ?, ?, ?, ?, ?)";
    try (PreparedStatement insertStmt = conn.prepareStatement(insertSql)) {
        insertStmt.setInt(1, idClient);
        insertStmt.setInt(2, selectedRobe);
        insertStmt.setDate(3, new java.sql.Date(currentDate.getTime()));
        insertStmt.setDate(4, new java.sql.Date(returnDate.getTime()));
        insertStmt.setInt(5, maxAllowed);
        int rows = insertStmt.executeUpdate();
        if (rows > 0) {
            JOptionPane.showMessageDialog(null, "Robe louée avec succès (nouveau client) !");
        }
    }
    return; // Fin de l'exécution ici
}

// Sinon - client existant, faire les vérifications
String activeSql = "SELECT COUNT(*) FROM locations WHERE id_client = ? AND date_retour IS NULL";
int activeLocations = 0;
try (PreparedStatement stmt = conn.prepareStatement(activeSql)) {
    stmt.setInt(1, idClient);
    try (ResultSet rs = stmt.executeQuery()) {
        if (rs.next()) {
            activeLocations = rs.getInt(1);
        }
    }
}

String limitSql = "SELECT nombre_limite FROM locations WHERE id_client = ? FETCH FIRST 1 ROWS ONLY";
int maxAllowed = 0;
try (PreparedStatement stamt = conn.prepareStatement(limitSql)) {
    stamt.setInt(1, idClient);
    try (ResultSet res = stamt.executeQuery()) {
        if (res.next()) {
            maxAllowed = res.getInt("nombre_limite");
        }
    }
}

String sql = "INSERT INTO locations (id_location, id_client, id_robe, date_location, date_retour_due, nombre_limite) VALUES (location_seq.nextval, ?, ?, ?, ?, ?)";
try (PreparedStatement stamt = conn.prepareStatement(sql)) {
    stamt.setInt(1, idClient);
    stamt.setInt(2, selectedRobe);
    stamt.setDate(3, new java.sql.Date(currentDate.getTime()));
    stamt.setDate(4, new java.sql.Date(returnDate.getTime()));
    stamt.setInt(5, maxAllowed); // Utilise la même limite
    int rowsAffected = stamt.executeUpdate();
    if (rowsAffected > 0) {
        JOptionPane.showMessageDialog(null, "Robe louée avec succès !");
    } else {
        System.out.println("Erreur lors de la location de la robe.");
    }
}
```

LES REQUÊTES SQL UTILISÉES EN JAVA RETOURNER ROBE

```
String id = (String) robe_louer.getSelectedItem();
int idLocation=Integer.parseInt(id);
Date dateRetourReelle = new Date(); // aujourd'hui
long joursRetard=0;
try (Connection conn = ConnectionProvider.orclConnection()) {
    // 1. Récupérer la date_retour_due
    PreparedStatement ps1 = conn.prepareStatement("SELECT date_retour_due FROM locations WHERE id_location = ?");
    ps1.setInt(1, idLocation);
    ResultSet rs = ps1.executeQuery();

    if (rs.next()) {
        Date dateRetourDue = rs.getDate("date_retour_due");

        // 2. Mettre à jour la date de retour réelle
        PreparedStatement ps2 = conn.prepareStatement("UPDATE locations SET date_retour = ? WHERE id_location = ?");
        ps2.setDate(1, new java.sql.Date(dateRetourReelle.getTime()));
        ps2.setInt(2, idLocation);
        ps2.executeUpdate();
        JOptionPane.showMessageDialog(null, "Date de retour mise à jour avec succès !");
    }

    // 3. Vérifier le retard
    long diffMillis = dateRetourReelle.getTime() - dateRetourDue.getTime();
    joursRetard = diffMillis / (1000 * 60 * 60 * 24);

    if (joursRetard > 0) {
        double montantPenalite = joursRetard * FINALITE_VALUE;

        // 4. Ajouter la pénalité
        PreparedStatement ps3 = conn.prepareStatement(
            "update locations set penalite =? where id_location=?"
        );
        ps3.setDouble(1, montantPenalite);
        ps3.setDouble(2, idLocation);

        ps3.executeUpdate();
        JOptionPane.showMessageDialog(null,
            "Retour en retard !\nPénalité appliquée : " + montantPenalite + " $");
    }
    else{
        JOptionPane.showMessageDialog(null, "Retour à temps. Aucune pénalité.");
    }
}
System.out.println("Bouton cliqué !");
setVisible(false);
new RetournerRobe().setVisible(true);
} catch (Exception e) {
    JOptionPane.showMessageDialog(null, "Erreur lors du retour de la robe : " + e.getMessage());
}
```

LES REQUÊTES SQL UTILISÉES EN JAVA RETOURNER ROBE

```
public void remplirComboClient(JComboBox<String> comboBox) {
    comboBox.removeAllItems();
    try{
        Connection con=ConnectionProvider.orclConnection();
        int id=LoginInetrface.clientConnecte.id;
        String query ="select prenom from clients where id=? ";
        System.out.println(LoginInetrface.clientConnecte.id);
        PreparedStatement pst = con.prepareStatement(query);
        pst.setInt(1, id);
        ResultSet rs = pst.executeQuery();
        while(rs.next()){
            comboBox.addItem(rs.getString("prenom"));
        }
        con.close();
    } catch(Exception e){
        JOptionPane.showMessageDialog(null, "Erreur: " + e.getMessage());
    }
}
```

INTERFACE JAVA WELCOME PAGE



Bienvenue à Glamour & Élégance

Glamour & Élégance est une boutique dédiée à la location de robes de soirée sophistiquées.
Bienvenue dans notre univers raffiné, explorez l'application et trouvez la robe de vos rêves.
Nous vous souhaitons une expérience aussi inoubliable que votre soirée.

s'inscrire

se connecter

+(216) 99 111 222

glamouretelegance@gmail.com

tous droits réservés

INTERFACE JAVA SIGN UP PAGE

Glamour & Élégance
s'inscrire

Nom

Prénom

Adresse

Email

Tél

Mot de passe

 Enregistrer

 Reculer



INTERFACE JAVA LOGIN PAGE



*Glamour & Élégance
se connecter*

Email

mot de passe

code admin

Login

clear

exit

[Changer mot de passe](#)

[s'inscrire](#)

INTERFACE JAVA CHANGE PASSWORD PAGE



*Glamour & Élégance
changer mot de passe*

ancien mot de passe

nouveau mot de passe

confirmation de mot de passe

 Enregistrer

 Reculer

INTERFACE JAVA ESPACE ADMIN



*Glamour & Élégance
Espace Admin*

gérer les robes

gérer les clients

voir statistiques

se déconnecter

INTERFACE JAVA GESTION DES ROBES

Glamour & Élégance Gestion des robes

Id : --

Description : courte midi longue

Taille :

Couleur :

Matière :

Prix :

Disponibilité : oui non

 ADD  UPDATE  DELETE...
 CLEAR

ID	Description	Taille	Couleur	Matière	Prix	Disponibilité
5	Midi	L	Verte	Coton	189.75	Oui
6	Courte	M	Verte	Velour	199.9	Oui
68	Longue	XL	Bleue	Sole	120.0	Oui
73	Midi	M	Jaune	Satin	90.0	Oui
3	Courte	S	Bleue	Satin	179.0	Oui
64	Midi	XL	Rouge	Satin	99.9	Oui
70	Courte	S	Noire	Dentelle	90.0	Oui
75	Longue	M	Rose	Sole	100.0	Oui
67	Longue	L	Bleue	Sole	120.0	Oui
62	Midi	M	Rouge	Satin	99.9	Oui
72	Courte	L	Noire	Dentelle	90.0	Oui
1	Longue	M	Rouge	Sole	199.99	Oui
63	Midi	L	Rouge	Satin	99.9	Oui
69	Courte	XS	Noire	Dentelle	90.0	Oui
74	Longue	XL	Rose	Sole	100.0	Oui
2	Longue	L	Noire	Dentelle	249.5	Oui
21	Longue	XL	Orange	Polyester	100.0	Oui
41	Longue	XS	Jaune	Satin	120.0	Oui
61	Midi	XS	Rouge	Satin	99.9	Oui
65	Midi	S	Rouge	Satin	99.9	Oui
66	Longue	M	Bleue	Sole	120.0	Oui
71	Courte	M	Noire	Dentelle	90.0	Oui

 EXIT  RECULER

INTERFACE JAVA GESTION DES CLIENTS

*Glamour & Élégance
Gestion des clients*

Id	Nom	Prenom	adresse	email	tel

Id --
nom
prenom
adresse
email
tel

 UPDATE
 DELETE
 CLEAR

 RESUMER  EXIT

INTERFACE JAVA STATISTIQUES

Glamour & Élégance Statistiques

ID Client :

 Afficher les robes louées par cette cliente

 Afficher les robes

 Afficher les robes en location

 Afficher les robes les plus demandées

 Reculer

ID	Description	Taille	Couleur	Matière	Prix	nb

INTERFACE JAVA ESPACE CLIENT

*Glamour & Élégance
Espace Client*

 effectuer location

 retourner robe

 se déconnecter



INTERFACE JAVA LOUER ROBE

*Glamour & Élégance
Effectuer location*

Prenom

Taille

 Item 1

Couleur

 Item 1

Date de Location

 2025-27-11

Durée (jours)

 0

Choisir robe

 Item 1

Robes disponibles

Id Robe	Description	Taille	Couleur	Matière	Prix



Afficher les robes disponibles



Louer

INTERFACE JAVA RETOURNER ROBE

*Glamour & Élégance
Retourner robe*

Client

Id robe louée

Date retour

Table de Robes louées par vous

Description	Taille	Couleur	Nom du Client	Date de Location

Table de penalité

Nom du Client	Date de Location	Date de Retour Prévue	Date de Retour Effectuée	Pénalité

 Retourner robe

 Facture