UNIVERSITÉ CADI AYAD

ECOLE SUPERIEURE DE TECHNOLOGIE-SAFI

DUT GÉNIE INFORMATIQUE

Compte rendu

TP : Login

(Genericite,MVC, DAO, Swing)

Réalisée par :

SAISSI Zahra

Enseigné par :

M.EL ABDELLAOUI Said

Année universitaire :2024/2025

Table des matières

Introduction

Le système de login est essentiel pour sécuriser l'accès aux applications et protéger les données sensibles. Cette partie a pour objectif de mettre en place un système de connexion simple utilisant l'architecture MVC (Modèle-Vue-Contrôleur). L'objectif principal est de développer une application qui permet à un utilisateur de se connecter en utilisant un nom d'utilisateur et un mot de passe. Le système repose sur l'utilisation de DAO pour gérer l'accès aux données, un modèle pour la logique métier, une vue pour l'interface utilisateur et un contrôleur pour gérer les interactions entre ces composants.

Objectifs

· Créer une vue de connexion permettant à l'utilisateur d'entrer ses identifiants.

· Implémenter un modèle de gestion des identifiants avec une logique d'authentification.

· Utiliser un DAO pour récupérer les informations de l'utilisateur et vérifier son mot de passe.

· Développer un contrôleur pour gérer les actions de l'utilisateur et la navigation entre les vues en fonction du résultat de l'authentification.

· Garantir une gestion des erreurs en cas d'échec de l'authentification.

Étapes de Création du Projet

1. 1ère étape : Couche Model - Implémentation du Modèle

1.1. Class Login.java:

La classe Login gère les informations d'identification d'un utilisateur, comprenant un nom d'utilisateur (username) et un mot de passe (password). Elle inclut des méthodes pour définir et récupérer ces données, ainsi qu'une fonction verifyCredentials qui compare le mot de passe saisi avec un mot de passe stocké afin de valider l'accès. Avec une structure simple et efficace, cette classe constitue un élément clé pour l'authentification dans une application.

Code :

```java
package Model;

public class Login {

    private String username;
    private String password;

    public Login(String username, String password) {
        this.username = username;
        this.password = password;
    }

    public String getUsername() {
        return username;
    }

    public void setUsername(String username) {
        this.username = username;
    }

    public String getPassword() {
        return password;
    }

    public void setPassword(String password) {
        this.password = password;
    }

    public boolean verifyCredentials(String storedPassword) {
        return this.password.equals(storedPassword);
    }

    @Override
    public String toString() {
        return "Login [username=" + username + ", password=" + password + "]";
    }
}
```

2. 2ème étape : gestion des donnée (DAO):

2.1. Création de la table dans la base de données

Ce code SQL crée une table login avec trois colonnes : id comme clé primaire unique auto-incrémentée, username pour les noms d'utilisateur uniques, et password pour les mots de passe. Cette table est utilisée pour gérer les connexions des utilisateurs.

Code SQL :

```sql
USE gestionressourceshumaines;

CREATE TABLE login (
  id INT(11) NOT NULL AUTO_INCREMENT,
  username VARCHAR(50) NOT NULL UNIQUE,
  password VARCHAR(255) NOT NULL,
  PRIMARY KEY (id)
);


INSERT INTO login (username, password) VALUES ('user', 'user');

```

gestionressourceshumaines.login: 1 ligne(s) au total (environ)

id	username	password
1	user	user

2.2. Implémentation de LoginDAOImpl:

La classe LoginDAOImpl contient une méthode principale, authenticate, qui vérifie si les informations de connexion d'un utilisateur sont valides. Cette méthode utilise une requête SQL pour récupérer le mot de passe stocké dans la base de données correspondant au nom d'utilisateur fourni. Elle compare ensuite ce mot de passe avec celui entré par l'utilisateur. En cas de problème, une exception SQL est capturée et un message d'erreur est affiché.

Code :

```java
package DAO;

import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;

public class LoginDAOImpl {

    // Méthode pour vérifier si le nom d'utilisateur et le mot de passe sont valides
    public boolean authenticate(String username, String password) {
        String sql = "SELECT password FROM login WHERE username = ?";
        try (PreparedStatement stmt = DBConnexion.getConnexion().prepareStatement(sql)) {
            stmt.setString(1, username);
            try (ResultSet rs = stmt.executeQuery()) {
                // Si un utilisateur est trouvé avec ce nom d'utilisateur
                if (rs.next()) {
                    String storedPassword = rs.getString("password");
                    return storedPassword.equals(password);
                }
            }
        } catch (SQLException exception) {
            System.err.println("Erreur lors de l'authentification : " + exception.getMessage());
            exception.printStackTrace();
        }
        return false;
    }
}
```

3. 3ème étape : Logique métier:

Classe LoginModel

La classe LoginModel sert de couche intermédiaire entre le contrôleur et le DAO. Elle contient une méthode authenticate qui utilise le DAO LoginDAOImpl pour vérifier si le nom d'utilisateur et le mot de passe sont corrects. Cette architecture permet de séparer la logique métier de l'accès aux données. Le constructeur prend un objet LoginDAOImpl, et la méthode authenticate transmet simplement la vérification des informations de connexion au DAO pour exécuter la logique de la base de données.

Code :

```java
package Model;

import DAO.LoginDAOImpl;

public class LoginModel {

    private LoginDAOImpl loginDAO;

    // Constructeur qui initialise le DAO de connexion
    public LoginModel(LoginDAOImpl loginDAO) {
        this.loginDAO = loginDAO;
    }

    // Méthode d'authentification qui appelle la méthode d'authentification du DAO
    // Et en meme temps vérifie si les informations de connexion (nom d'utilisateur et mot de passe) sont justes
    public boolean authenticate(String username, String password) {
        return loginDAO.authenticate(username, password);
    }
}

```

4. 4ème étape : Couche View - Interface graphique

La classe LoginView représente la vue de l'interface graphique pour la page de connexion. Elle comprend deux champs de saisie pour le nom d'utilisateur et le mot de passe, ainsi qu'un bouton de connexion. Les méthodes getUsername() et getPassword() permettent d'accéder aux informations saisies par l'utilisateur, tandis que addLoginListener() permet d'ajouter un écouteur d'événements pour le bouton de connexion. La méthode showError() affiche un message d'erreur en cas de connexion incorrecte, et la méthode close() ferme la fenêtre de connexion.

Code:

```java
package View;

import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionListener;

public class LoginView extends JFrame {

    private JTextField usernameField;
    private JPasswordField passwordField;
    private JButton loginButton;

    public LoginView() {
        setTitle("Login");
        setSize(300, 200);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        usernameField = new JTextField(20);
        passwordField = new JPasswordField(20);
        loginButton = new JButton("Login");

        setLayout(new FlowLayout());
        add(new JLabel("Username:"));
        add(usernameField);
        add(new JLabel("Password:"));
        add(passwordField);
        add(loginButton);

        setLocationRelativeTo(null);
    }

    // Retourne le nom d'utilisateur saisi
    public String getUsername() {
        return usernameField.getText();
    }

    // Retourne le mot de passe saisi
    public String getPassword() {
        return new String(passwordField.getPassword());
    }



    // Ajoute un ActionListener au bouton de connexion
    public void addLoginListener(ActionListener listener) {
        loginButton.addActionListener(listener);
    }

    // Affiche un message d'erreur dans une fenêtre pop-up
    public void showError(String message) {
        JOptionPane.showMessageDialog(this, message, "Error", JOptionPane.ERROR_MESSAGE);
    }

    // Ferme la fenêtre de login
    public void close() {
        this.setVisible(false);
    }
}
```

5. 5ème étape : Couche Controller - Implémentation du Contrôleur

La classe LoginController gère l'interaction entre la vue (LoginView) et le modèle (LoginModel) pour le processus de connexion. Lorsqu'un utilisateur clique sur le bouton, la méthode actionPerformed() récupère les informations saisies (nom d'utilisateur et mot de passe), valide si ces informations ne sont pas vides, puis tente d'authentifier l'utilisateur en utilisant le modèle. Si l'authentification réussit, un message de succès est affiché, sinon un message d'erreur est montré.

Code :

```java
package Controller;

import Model.LoginModel;
import View.LoginView;

import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

public class LoginController {

    private final LoginView loginView;
    private final LoginModel loginModel;

    public LoginController(LoginView loginView, LoginModel loginModel) {
        this.loginView = loginView;
        this.loginModel = loginModel;

        // Attach the login button listener
        this.loginView.addLoginListener(new LoginListener());
    }

    private class LoginListener implements ActionListener {

        @Override
        public void actionPerformed(ActionEvent e) {
            // Get the username and password from the login view
            String username = loginView.getUsername();
            String password = loginView.getPassword();

            // Validate input fields
            if (username.isEmpty() || password.isEmpty()) {
                loginView.showError("Username or password cannot be empty.");
                return;
            }

            // Attempt to authenticate the user using the model
            boolean isAuthenticated = loginModel.authenticate(username, password);

            if (isAuthenticated) {
                // If authentication is successful, notify the user
                loginView.showError("Login successful!");
                loginView.close(); // Close the login view or proceed to the next view
            } else {
                // If authentication fails, show an error message
                loginView.showError("Invalid username or password. Please try again.");
            }
        }
    }
}
```

6. 6ème étape : Main - Application principale

La classe Main initialise l'application en créant les objets nécessaires pour la gestion de la connexion. Il instancie la vue de connexion (LoginView) et le modèle de

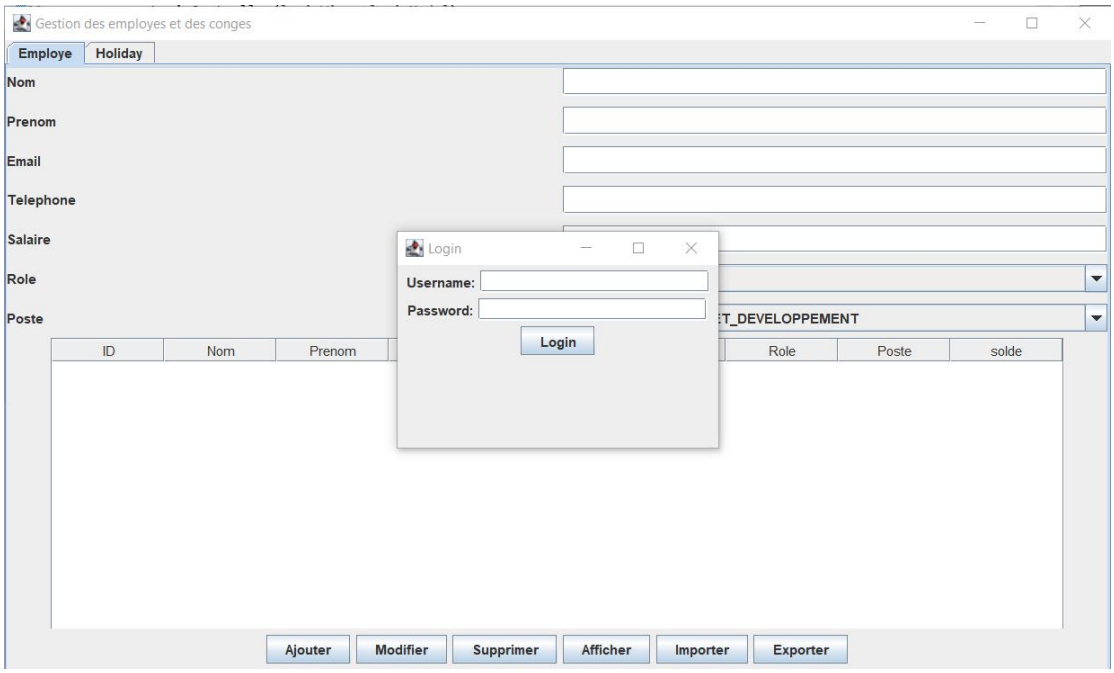
connexion (LoginModel) associé au DAO (LoginDAOImpl). Le contrôleur LoginController est ensuite créé pour gérer l'interaction entre la vue et le modèle de connexion.

Code :

```java
public class Main {
    public static void main(String[] args) {
        // Crée des instances des DAO pour la gestion des données
        LoginDAOImpl loginDAO = new LoginDAOImpl();
        EmployeDAOImpl employeDAO = new EmployeDAOImpl();
        HolidayDAOImpl holidayDAO = new HolidayDAOImpl();

        // Crée des instances des modèles pour gérer la logique des employés, des congés et de la connexion
        LoginModel loginModel = new LoginModel(loginDAO);
        EmployeModel employeModel = new EmployeModel(employeDAO);
        HolidayModel holidayModel = new HolidayModel(holidayDAO);

        // Crée la vue de connexion et la vue principale de l'application
        LoginView loginView = new LoginView();
        MainView employeHolidayView = new MainView();

        // Crée le contrôleur de connexion pour gérer les actions liées à la connexion
        new LoginController(loginView, loginModel);

        loginView.setVisible(true);

        // Écoute les événements de connexion et affiche la vue principale si la connexion est réussie
        loginView.addLoginListener(e -> {
            // Vérifie si l'authentification est réussie
            if (loginModel.authenticate(loginView.getUsername(), loginView.getPassword())) {
                // Si la connexion est réussie, cache la vue de connexion
                loginView.setVisible(false);

                // Initialise les contrôleurs pour la gestion des employés et des congés
                new EmployeController(employeHolidayView, employeModel);
                new HolidayController(employeHolidayView, holidayModel);

                // Affiche la vue principale de l'application
                employeHolidayView.setVisible(true);
            } else {
                // Si la connexion échoue, affiche un message d'erreur
                loginView.showError("Nom d'utilisateur et mot de passe incorrects. Essayez à nouveau.");
            }
        });
    }

}
```

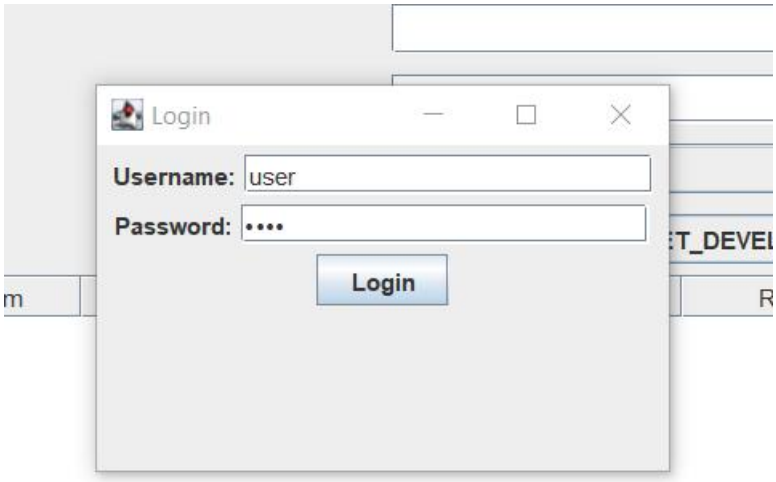
Réalisation

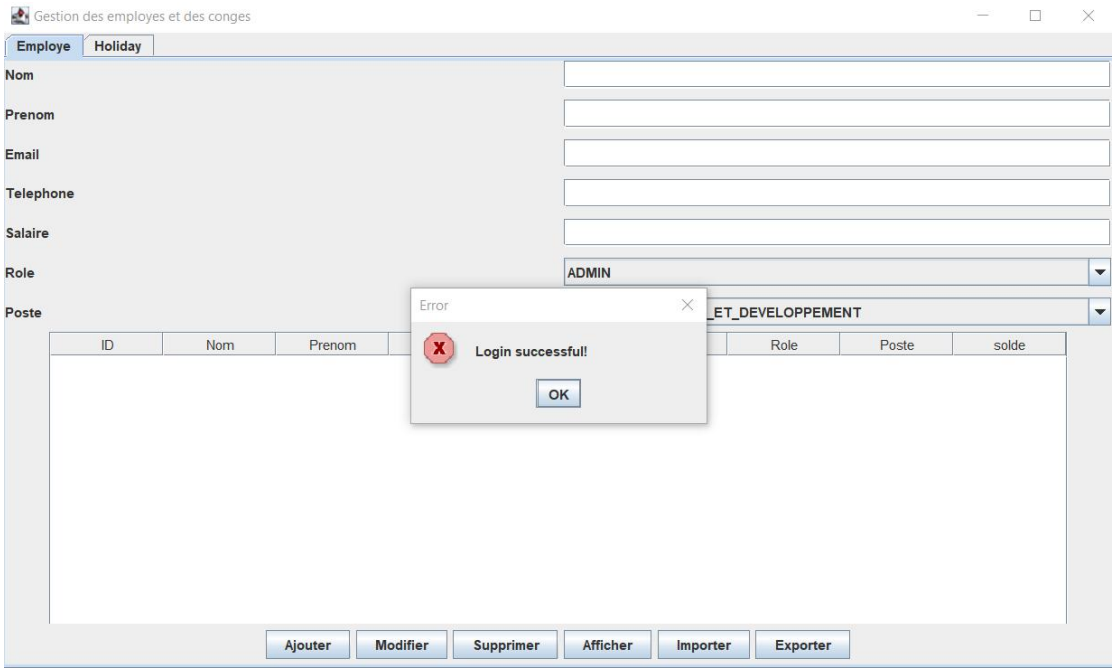
1. Page Login:



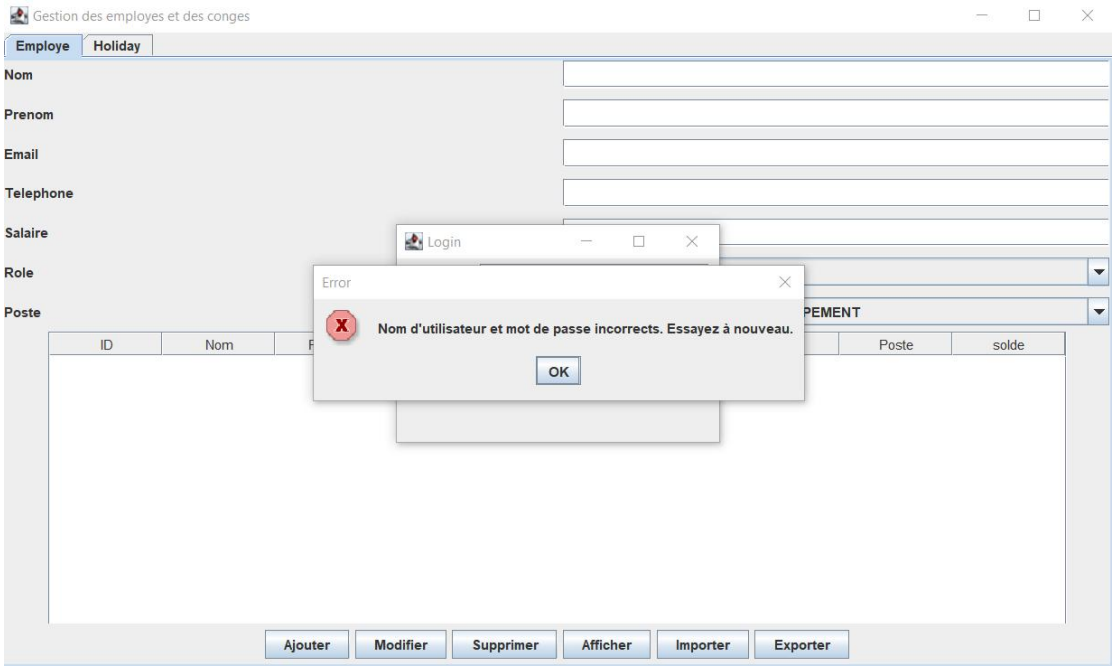
2. Entrer le nom utilisateur et mot de passe:

D'après la base de données le nom d'utilisateur est user et le mot de passe est user





Si les info sont incorrect:



Conclusion

Ce TP permet de comprendre et de mettre en pratique l'architecture MVC dans le contexte d'une application de gestion de connexion. Il met l'accent sur l'interaction entre la vue, le modèle et le contrôleur, ainsi que sur la gestion de l'authentification utilisateur à travers une structure modulaire et maintenable. Ce travail offre une base solide pour des projets nécessitant une gestion sécurisée des accès.