

Université Cadi Ayyad  
École Supérieure De Technologie-Safi  
Filière : génie informatique

---

**Rapport du TP N°3 java avancée**  
**Gestion de congés E/S**

---

**Réalisé par :**  
SAAD FATI MA ZAHRA

**Encadré par :**  
Mme. ASMAA el kourchi

**ANNEE UNIVERSITAIRE : 2024/2025**

## Table des matières

<b>Introduction :</b>	<b>3</b>
<b>1. Outils &amp; environnement de travail</b>	<b>4</b>
1.1 Environnement de travail	4
1.2 Outils de travail	4
1.3 Language de Programmation	5
<b>Les codes :</b>	<b>6</b>
Package model :	6
• Employee et EmployeeModel et Holiday et HolidayModel :	6
Package DAO:	11
• DBconnction,les interfaces, implémentation des interfaces :	11
Package Controller:	17
• Employe Controller et Holiday Controller :	17
Package View :	22
• Employeview,HolidayView et ManagementInterfaces:	22
Main:	28
<b>Résultats</b>	<b>29</b>
4.TableBD :	29
5. View :	29
6. Exportemploye :	30
7. Import Employé :	32
<b>Conclusion générale</b>	<b>34</b>
<b>Références</b>	<b>35</b>

## *Introduction :*

Ce **TP3** porte sur la création d'une application Java axée sur la gestion des employés et gestion de congés, en adoptant une approche structurée selon le modèle d'architecture MVC (Modèle Vue Contrôleur). Ce projet s'inscrit dans le cadre de l'exploration des principes essentiels de la programmation orientée objet (POO) et de l'utilisation des interfaces graphiques avec la bibliothèque Swing. Il représente également une opportunité de renforcer les compétences en conception logicielle et en organisation méthodique du code afin d'assurer une séparation claire des rôles.

L'objectif fondamental est de concevoir une application conviviale et performante permettant de manipuler les informations liées aux employés et le congés. Cette application est élaborée pour gérer l'ajout, la mise à jour, la suppression ainsi que l'affichage des données des employés et de congés, tout en offrant une interface utilisateur fluide et engageante. En appliquant les principes de l'architecture MVC, ce projet favorise une meilleure maintenabilité et garantit la capacité d'évolution de l'application.

Les fonctionnalités clés incluent :

- L'ajout d'employés avec des détails complets et l'ajout de congé si le solde est suffisant.
- La mise à jour des informations des congés.
- La suppression des congés.
- L'affichage de la liste des congés.
- Export les informations d'employé.
- Import les informations d'employé.

# 1. Outils & environnement de travail

## 1.1 Environnement de travail



Figure 1 IntelliJ logo

- **IntelliJ IDEA** est un IDE puissant, principalement utilisé pour Java, mais supporte également plusieurs autres langages. Il offre des fonctionnalités avancées comme l'auto-complétion, le débogage, et le refactoring pour améliorer la productivité des développeurs.

## 1.2 Outils de travail



Figure 2 MySQL Workbench logo

- **MySQL Workbench** : un outil de travail graphique conçu pour faciliter la conception, l'administration, et la gestion des bases de données MySQL. Il fournit une interface utilisateur intuitive permettant de travailler avec des bases de données sans avoir à utiliser uniquement des commandes en ligne.



Figure 3 xampp logo

- **xampp** : En parallèle, le projet vise à fournir des outils de gestion robustes pour le corps administratif, avec une fonctionnalité de multi-rôle, permettant à chaque agent d'accéder à un compte adapté à ses responsabilités spécifique



Figure 4 java developpement kit logo

- **java developpement kit** : st un ensemble d'outils logiciels nécessaires pour développer des applications Java. Il inclut les composants essentiels pour coder, compiler, exécuter et déboguer des programmes Java.

## 1.3 Language de Programmation



Figure 5 java logo

- **Java** : un langage de programmation orienté objet et une plateforme largement utilisée pour le développement d'applications logicielles. Il a été créé par Sun Microsystems (maintenant propriété d'Oracle) en 1995 et reste l'un des langages les plus populaires au monde, notamment pour les applications d'entreprise, le développement mobile (Android) et les applications web

## *Les codes :*

### **Package model :**

- **Employee et EmployeeModel et Holiday et HolidayModel :**

```
package Model;

public class Employee {

    private int id;
    private String nom;
    private String prenom;
    private String email;
    private String telephone;
    private double salaire;
    private Role role;
    private Poste poste;
    private int solde;

    // Enum pour le rôle d'un employé
    public enum Role {
        MANAGER,
        DEVELOPER,
        DESIGNER,
        QA
    }

    // Enum pour le poste d'un employé
    public enum Poste {
        FULL_TIME,
        PART_TIME,
        INTERN
    }

    // Constructeur de l'employé
    public Employee(int id, String nom, String prenom, String email, String telephone, double
salaire, Role role, Poste poste, int solde) {
        this.id = id;
        this.nom = nom;
        this.prenom = prenom;
        this.email = email;
        this.telephone = telephone;
        this.salaire = salaire;
        this.role = role;
        this.poste = poste;
        this.solde = solde;
    }

    // Getters et setters pour tous les attributs
    public int getId() {
        return id;
    }

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

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

```

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

public String getPrenom() {
    return prenom;
}

public void setPrenom(String prenom) {
    this.prenom = prenom;
}

public String getEmail() {
    return email;
}

public void setEmail(String email) {
    this.email = email;
}

public String getTelephone() {
    return telephone;
}

public void setTelephone(String telephone) {
    this.telephone = telephone;
}

public double getSalaire() {
    return salaire;
}

public void setSalaire(double salaire) {
    this.salaire = salaire;
}

public Role getRole() {
    return role;
}

public void setRole(Role role) {
    this.role = role;
}

public Poste getPoste() {
    return poste;
}

public void setPoste(Poste poste) {
    this.poste = poste;
}

public int getSolde() {
    return solde;
}

public void setSolde(int solde) {
    this.solde = solde;
}
}

```

```

package Model;

import DAO.EmployeeDAOI;

import java.util.ArrayList;
import java.util.List;
import java.io.*;

public class EmployeeModel {
    private EmployeeDAOI employeeDAO;

    public EmployeeModel(EmployeeDAOI employeeDAO) {
        this.employeeDAO = employeeDAO;
    }

    private List<Employee> employees = new ArrayList<>();

    public boolean ajouterEmployee(int id, String nom, String prenom, String email, String
telephone, double salaire, Employee.Role role, Employee.Poste poste, int solde) {
        Employee newEmployee = new Employee(id, nom, prenom, email, telephone, salaire, role,
poste, solde);
        return employees.add(newEmployee);
    }

    public boolean modifierEmployee(int id, String nom, String prenom, String email, String
telephone, double salaire, Employee.Role role, Employee.Poste poste, int solde) {
        for (Employee emp : employees) {
            if (emp.getId() == id) {
                emp.setNom(nom);
                emp.setPrenom(prenom);
                emp.setEmail(email);
                emp.setTelephone(telephone);
                emp.setSalaire(salaire);
                emp.setRole(role);
                emp.setPoste(poste);
                emp.setSolde(solde);
                return true;
            }
        }
        return false;
    }

    public boolean supprimerEmployee(int id) {
        return employees.removeIf(emp -> emp.getId() == id);
    }

    public boolean importerEmployees(String filePath) {
        try (BufferedReader reader = new BufferedReader(new FileReader(filePath))) {
            String line;
            while ((line = reader.readLine()) != null) {
                String[] data = line.split(",");
                int id = Integer.parseInt(data[0]);
                String nom = data[1];
                String prenom = data[2];
                String email = data[3];
                String telephone = data[4];
                double salaire = Double.parseDouble(data[5]);
                Employee.Role role = Employee.Role.valueOf(data[6]);
                Employee.Poste poste = Employee.Poste.valueOf(data[7]);
                int solde = Integer.parseInt(data[8]);

                ajouterEmployee(id, nom, prenom, email, telephone, salaire, role, poste,
solde);
            }
        }
    }
}

```



```

    }
    return true;
} catch (IOException | NumberFormatException e) {
    e.printStackTrace();
    return false;
}
}

public boolean exporterEmployees(String filePath) {
    try (BufferedWriter writer = new BufferedWriter(new FileWriter(filePath))) {
        for (Employee emp : employees) {
            writer.write(emp.getId() + "," + emp.getNom() + "," + emp.getPrenom() + "," +
emp.getEmail() + "," +
                emp.getTelephone() + "," + emp.getSalaire() + "," + emp.getRole() +
"," + emp.getPoste() + "," + emp.getSolde());
            writer.newLine();
        }
        return true;
    } catch (IOException e) {
        e.printStackTrace();
        return false;
    }
}

public List<Employee> afficherEmployees() {
    return employees;
}
}

package Model;

import java.util.Date;

public class Holiday {

    private int id;
    private Date startDate, endDate;
    private HolidayType holidayType;
    private int employeeId;
    private String employeeNom;
    private String selectedItem;
    private int solde;

    public Holiday(int id, Date startDate, Date endDate, HolidayType holidayType, int
employeeId) {
        this.id = id;
        this.startDate = startDate;
        this.endDate = endDate;
        this.holidayType = holidayType;
        this.employeeId = employeeId;
    }

    public Holiday(int id, Date startDate, Date endDate, HolidayType holidayType, int
employeeId, String employeeNom) {
        this.id = id;
        this.startDate = startDate;
        this.endDate = endDate;
        this.holidayType = holidayType;
        this.employeeId = employeeId;
        this.employeeNom = employeeNom;
    }

    public Holiday(String employeeNom, Date startDate, Date endDate, HolidayType holidayType)
{

```

```

        this.employeeNom = employeeNom;
        this.startDate = startDate;
        this.endDate = endDate;
        this.holidayType = holidayType;
    }

    public Holiday(int id, Date startDate, Date endDate, HolidayType holidayType, String
employeeNom) {
        this.id = id;
        this.startDate = startDate;
        this.endDate = endDate;
        this.holidayType = holidayType;
        this.employeeNom = employeeNom;
    }

    public Holiday(int holidayId, Date startDate2, Date endDate2, HolidayType valueOf, int
employeeId2, String nomEmployee, int solde) {
        this.id = holidayId;
        this.startDate = startDate2;
        this.endDate = endDate2;
        this.holidayType = valueOf;
        this.employeeNom = nomEmployee;
        this.solde = solde;
    }

    public int getId() { return id; }
    public Date getStartDate() { return startDate; }
    public Date getEndDate() { return endDate; }
    public HolidayType getHolidayType() { return holidayType; }
    public int getEmployeeId() { return employeeId; }
    public String getEmployeeNom() { return employeeNom; }
    public String getSelectedItem() { return selectedItem; }
    public int getSolde() { return solde; }

    public enum HolidayType {
        CONGE_PAYEE,
        CONGE_NON_PAYEE,
        CONGE MALADIE
    }
}

package Model;

import DAO.HolidayDAOImpl;
import Model.Holiday.HolidayType;
import java.util.Date;
import java.util.List;

public class HolidayModel {

    private HolidayDAOImpl dao;

    public HolidayModel(HolidayDAOImpl dao) {
        this.dao = dao;
    }

    public boolean ajouterHoliday(String employeeNom, Date startDate, Date endDate,
HolidayType holidayType) {
        if (!isValidDateRange(startDate, endDate)) {
            System.out.println("Date invalide!");
            return false;
        }

        long days = calculateHolidayDays(startDate, endDate);

```

```

        System.out.println(days + " day(s)");

        Holiday newHoliday = new Holiday(employeeNom, startDate, endDate, holidayType);
        dao.ajouter(newHoliday);

        return true;
    }

    public boolean isValidDateRange(Date startDate, Date endDate) {
        return !startDate.after(endDate);
    }

    private long calculateHolidayDays(Date startDate, Date endDate) {
        long diffInMillis = endDate.getTime() - startDate.getTime();
        return diffInMillis / (1000 * 60 * 60 * 24);
    }

    public List<Holiday> afficherHolidays() {
        return dao.afficher();
    }

    public List<String> chargerNomsEmployes() {
        return dao.chargerNomsEmployes();
    }

    public boolean supprimerHoliday(int id) {
        if(id <= 0) {
            System.out.println("Id invalide!!");
            return false;
        }

        dao.supprimer(id);
        return true;
    }

    public boolean modifierHoliday(int id, String employeeNom, Date startDate, Date endDate,
        HolidayType holidayType) {
        if (id <= 0 || !isValidDateRange(startDate, endDate)) {
            System.out.println("Données invalides pour la modification.");
            return false;
        }

        Holiday updatedHoliday = new Holiday(id, startDate, endDate, holidayType,
        employeeNom);
        return dao.modifier(id, updatedHoliday);
    }
}

```

## Package DAO:

- DBconnexion, les interfaces, implémentation des interfaces :

```

package DAO;
import java.sql.*;

public class DBConnection {
    private static final String url="jdbc:mysql://localhost:3306/gestiondeconges";
    private static final String user="root";
    private static final String password="YES";
    static Connection connection=null;

    public static Connection getConnection() throws SQLException {

```

```

        if(connection==null|| connection.isClosed()) {

            try {

                Class.forName("com.mysql.cj.jdbc.Driver");
                connection=DriverManager.getConnection(url, user, password);
                System.out.println("Connexion reussie !");

            }catch(ClassNotFoundException |SQLException e) {
                e.printStackTrace();
                throw new RuntimeException("Erreur lors de la connexion a la base de donnees
!");
            }

        }
        return connection;

    }

    /*public static void closeConnection() {

        if(connection!=null) {
            try {

                connection.close();
                System.out.println("Connexion ferme.");

            }catch(SQLException e) {
                e.printStackTrace();
                throw new RuntimeException("Erreur lors de la fermeture de la connexion!");
            }
        }
    }*/
}

package DAO;

import java.io.IOException;
import java.util.List;

public interface DataImportExport<T> {
    void importData(String fileName) throws IOException;

    void exportData(String fileName, List<T> data) throws IOException;
}

```

```

package DAO;

import java.util.List;
import Model.Employee;

public interface EmployeeDAOI {

    public void ajouterEmployee(Employee newEmployee);
    public List<Employee> afficherEmployees();
    public void modifierEmployee(int id, Employee modifiedEmployee);
    public void supprimerEmployee(int id);

}

```

```

package DAO;

```

```

import java.util.List;

public interface GenericDAOI<T> {
    void ajouter(T entity);
    List<T> afficher();
    boolean modifier(int id, T entity); // Méthode modifiée
    void supprimer(int id);
}

package DAO;

import Model.Employee;
import Model.Employee.Poste;
import Model.Employee.Role;
import java.sql.*;
import java.util.ArrayList;
import java.util.List;
import java.io.*;

public class EmployeeDAOImpl implements EmployeeDAOI, DataImportExport<Employee> {

    @Override
    public List<Employee> afficherEmployees() {
        String query = "SELECT e.id, e.nom, e.prenom, e.email, e.telephone, e.salaire, r.nom AS
roleNom, p.nom AS posteNom " +
            "FROM employe e " +
            "JOIN role r ON e.role_id = r.id " +
            "JOIN poste p ON e.poste_id = p.id";
        List<Employee> employees = new ArrayList<>();

        try (Connection conn =
DriverManager.getConnection("jdbc:mysql://localhost:3306/gestiondeconges", "root", "YES");
            Statement stmt = conn.createStatement();
            ResultSet rs = stmt.executeQuery(query)) {

            while (rs.next()) {
                String roleNom = rs.getString("roleNom");
                String posteNom = rs.getString("posteNom");

                Role role = Role.valueOf(roleNom);
                Poste poste = Poste.valueOf(posteNom);

                Employee employee = new Employee(
                    rs.getInt("id"),
                    rs.getString("nom"),
                    rs.getString("prenom"),
                    rs.getString("email"),
                    rs.getString("telephone"),
                    rs.getDouble("salaire"),
                    role,
                    poste,
                    0
                );
                employees.add(employee);
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }

        return employees;
    }
}

```

```

@Override
public void ajouterEmployee(Employee employee) {
    String query = "INSERT INTO employe (nom, prenom, email, telephone, salaire, role_id,
poste_id) " +
        "VALUES (?, ?, ?, ?, ?, (SELECT id FROM role WHERE nom=?), (SELECT id FROM poste
WHERE nom=?))";

    try (PreparedStatement stmt = DBConnection.getConnection().prepareStatement(query)) {
        stmt.setString(1, employee.getNom());
        stmt.setString(2, employee.getPrenom());
        stmt.setString(3, employee.getEmail());
        stmt.setString(4, employee.getTelephone());
        stmt.setDouble(5, employee.getSalaire());
        stmt.setString(6, employee.getRole().name());
        stmt.setString(7, employee.getPoste().name());

        int rowAffected = stmt.executeUpdate();

        if (rowAffected == 0) {
            System.out.println("Échec de l'insertion !!");
        } else {
            System.out.println("Insertion réussie :)");
        }

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

@Override
public void modifierEmployee(int id, Employee modifiedEmployee) {
    String query = "UPDATE employe SET nom=?, prenom=?, email=?, telephone=?, salaire=?, "
+
        "role_id=(SELECT id FROM role WHERE nom=?), poste_id=(SELECT id FROM poste WHERE
nom=?) WHERE id=?";

    try (PreparedStatement stmt = DBConnection.getConnection().prepareStatement(query)) {
        stmt.setString(1, modifiedEmployee.getNom());
        stmt.setString(2, modifiedEmployee.getPrenom());
        stmt.setString(3, modifiedEmployee.getEmail());
        stmt.setString(4, modifiedEmployee.getTelephone());
        stmt.setDouble(5, modifiedEmployee.getSalaire());
        stmt.setString(6, modifiedEmployee.getRole().name());
        stmt.setString(7, modifiedEmployee.getPoste().name());
        stmt.setInt(8, id);

        int rowAffected = stmt.executeUpdate();

        if (rowAffected > 0) {
            System.out.println("L'employé a été modifié :)");
        } else {
            System.out.println("L'employé n'a pas été modifié :(");
        }

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

@Override
public void supprimerEmployee(int id) {
    String query = "DELETE FROM employe WHERE id=?";

```

```

try (PreparedStatement stmt = DBConnection.getConnection().prepareStatement(query)) {
    stmt.setInt(1, id);

    int rowAffected = stmt.executeUpdate();

    if (rowAffected > 0) {
        System.out.println("L'employé a été supprimé :)");
    } else {
        System.out.println("L'employé n'a pas été supprimé :(");
    }
} catch (SQLException e) {
    e.printStackTrace();
}

}

@Override
public void importData(String filePath) throws IOException {
    String query = "INSERT INTO employe (nom, prenom, email, telephone, salaire, role_id,
poste_id) VALUES (?, ?, ?, ?, ?, ?, ?)";

    try (BufferedReader reader = new BufferedReader(new FileReader(filePath));
        PreparedStatement pstmt = DBConnection.getConnection().prepareStatement(query)) {

        String line;
        while ((line = reader.readLine()) != null) {
            String[] data = line.split(",");
            if (data.length == 7) {
                pstmt.setString(1, data[0].trim());
                pstmt.setString(2, data[1].trim());
                pstmt.setString(3, data[2].trim());
                pstmt.setString(4, data[3].trim());
                pstmt.setDouble(5, Double.parseDouble(data[4].trim()));
                pstmt.setString(6, data[5].trim());
                pstmt.setString(7, data[6].trim());
                pstmt.addBatch();
            }
        }
        pstmt.executeBatch();
        System.out.println("Les employés ont été importés avec succès !");
    } catch (SQLException e) {
        e.printStackTrace();
    }
}

@Override
public void exportData(String fileName, List<Employee> employees) throws IOException {
    try (BufferedWriter writer = new BufferedWriter(new FileWriter(fileName))) {
        writer.write("Nom, Prenom, Email, Telephone, Salaire, Role, Poste");
        writer.newLine();
        for (Employee employee : employees) {
            writer.write(String.format("%s,%s,%s,%s,%.2f,%s,%s",
                employee.getNom(),
                employee.getPrenom(),
                employee.getEmail(),
                employee.getTelephone(),
                employee.getSalaire(),
                employee.getRole().name(),
                employee.getPoste().name()));
            writer.newLine();
        }
        System.out.println("Les employés ont été exportés avec succès !");
    }
}

```

```

    }
}
}

```

```

package DAO;

import Model.Holiday;
import java.sql.*;
import java.util.List;
import java.util.ArrayList;

public class HolidayDAOImpl {

    public void ajouter(Holiday holiday) {
        String sql = "INSERT INTO holiday (employeeNom, startDate, endDate, holidayTypeId)
VALUES (?, ?, ?, (SELECT id FROM holidaytype WHERE nom=?))";
        try (PreparedStatement stmt = DBConnection.getConnection().prepareStatement(sql)) {
            stmt.setString(1, holiday.getEmployeeNom());
            stmt.setDate(2, new java.sql.Date(holiday.getStartDate().getTime()));
            stmt.setDate(3, new java.sql.Date(holiday.getEndDate().getTime()));
            stmt.setString(4, holiday.getHolidayType().name());
            stmt.executeUpdate();
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }

    public boolean modifier(int id, Holiday holiday) {
        String sql = "UPDATE holiday SET startDate = ?, endDate = ?, holidayTypeId = (select
id from holidaytype where nom=?) WHERE id = ?";
        try (PreparedStatement stmt = DBConnection.getConnection().prepareStatement(sql)) {
            stmt.setDate(1, new java.sql.Date(holiday.getStartDate().getTime()));
            stmt.setDate(2, new java.sql.Date(holiday.getEndDate().getTime()));
            stmt.setString(3, holiday.getHolidayType().name());
            stmt.setInt(4, id); // Utilisation de l'ID ici
            return stmt.executeUpdate() > 0;
        } catch (SQLException e) {
            e.printStackTrace();
        }
        return false;
    }

    public List<Holiday> afficher() {
        List<Holiday> holidays = new ArrayList<>();
        String sql = "SELECT * FROM holiday";
        try (Statement stmt = DBConnection.getConnection().createStatement()) {
            ResultSet rs = stmt.executeQuery(sql);
            while (rs.next()) {
                // Charger les données d'un holiday
                // Utilisez rs.getXXX pour obtenir les valeurs des colonnes
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
        return holidays;
    }

    public List<String> chargerNomsEmployes() {
        List<String> employees = new ArrayList<>();
        String sql = "SELECT DISTINCT employeeNom FROM holiday";
        try (Statement stmt = DBConnection.getConnection().createStatement()) {
            ResultSet rs = stmt.executeQuery(sql);

```



```

        while (rs.next()) {
            employees.add(rs.getString("employeeNom"));
        }
    } catch (SQLException e) {
        e.printStackTrace();
    }
    return employees;
}

public void supprimer(int id) {
    String sql = "DELETE FROM holiday WHERE id = ?";
    try (PreparedStatement stmt = DBConnection.getConnection().prepareStatement(sql)) {
        stmt.setInt(1, id);
        stmt.executeUpdate();
    } catch (SQLException e) {
        e.printStackTrace();
    }
}
}

```

## Package Controller:

- **Employe Controller et Holiday Controller :**

```

package Controller;

import Model.EmployeeModel;
import Model.Employee;
import Model.Employee.Role;
import Model.Employee.Poste;
import View.EmployeeView;

import java.util.List;

public class EmployeeController {

    private EmployeeView view;
    private EmployeeModel model;

    public EmployeeController(EmployeeView view, EmployeeModel model) {
        this.model = model;
        this.view = view;
        afficherEmployees();

        // Add listeners to the view buttons
        this.view.ajouterButton.addActionListener(e -> {
            ajouterEmployee();
            clearFields();
            afficherEmployees();
        });

        this.view.modifierButton.addActionListener(e -> {
            modifierEmployee();
            clearFields();
            afficherEmployees();
        });

        this.view.supprimerButton.addActionListener(e -> {
            supprimerEmployee();
            afficherEmployees();
        });
    }
}

```

```

        this.view.afficherButton.addActionListener(e -> afficherEmployees());

        this.view.importButton.addActionListener(e -> importerEmployees());
        this.view.exportButton.addActionListener(e -> exporterEmployees());
    }

    // Method to add an employee
    public void ajouterEmployee() {
        int solde = 0;
        int id = 0;
        String nom = view.getNom(), prenom = view.getPrenom(), email = view.getEmail(),
telephone = view.getTelephone();
        double salaire = view.getSalaire();
        Role role = view.getRole();
        Poste poste = view.getPoste();

        // Ensure the employee is successfully added
        boolean ajoutReussi = model.ajouterEmployee(id, nom, prenom, email, telephone,
salaire, role, poste, solde);

        if (ajoutReussi) {
            view.afficherMessageSucces("Employee ajouté avec succès :)");
        } else {
            view.afficherMessageErreur("Échec de l'ajout :(");
        }
    }

    // Method to display the list of employees
    public void afficherEmployees() {
        List<Employee> employees = model.afficherEmployees();
        view.afficherEmployees(employees);
    }

    // Method to modify an employee
    public void modifierEmployee() {
        int solde = 0;
        int id = view.getId();

        if (id <= 0) {
            view.afficherMessageErreur("Sélectionnez un employé pour le modifier.");
            return;
        }

        String nom = view.getNom(), prenom = view.getPrenom(), email = view.getEmail(),
telephone = view.getTelephone();
        double salaire = view.getSalaire();
        Role role = view.getRole();
        Poste poste = view.getPoste();

        // Call the model to modify the employee
        boolean modificationReussie = model.modifierEmployee(id, nom, prenom, email,
telephone, salaire, role, poste, solde);

        if (modificationReussie) {
            view.afficherMessageSucces("L'employé a été modifié :)");
        } else {
            view.afficherMessageErreur("L'employé n'a pas été modifié :(");
        }
    }

    // Method to delete an employee
    public void supprimerEmployee() {
        int id = view.getId();

```

```

        if (id <= 0) {
            view.afficherMessageErreur("Sélectionnez un employé pour le supprimer.");
            return;
        }

        // Call the model to delete the employee
        boolean suppressionReussi = model.supprimerEmployee(id);

        if (suppressionReussi) {
            view.afficherMessageSucces("L'employé a été supprimé :)");
        } else {
            view.afficherMessageErreur("L'employé n'a pas été supprimé :(");
        }
    }

    // Method to import employee data from a file
    public void importerEmployees() {
        String filePath = view.showFileChooser("Importer des employés");
        if (filePath != null) {
            boolean importationReussie = model.importerEmployees(filePath);
            if (importationReussie) {
                view.afficherMessageSucces("Les employés ont été importés avec succès :)");
                afficherEmployees();
            } else {
                view.afficherMessageErreur("Échec de l'importation des employés :(");
            }
        }
    }

    // Method to export employee data to a file
    public void exporterEmployees() {
        String filePath = view.showFileChooser("Exporter des employés");
        if (filePath != null) {
            boolean exportationReussie = model.exporterEmployees(filePath);
            if (exportationReussie) {
                view.afficherMessageSucces("Les employés ont été exportés avec succès :)");
            } else {
                view.afficherMessageErreur("Échec de l'exportation des employés :(");
            }
        }
    }

    // Method to clear input fields after an operation
    private void clearFields() {
        view.jtfNom.setText(null);
        view.jtfPrenom.setText(null);
        view.jtfEmail.setText(null);
        view.jtfTelephone.setText(null);
        view.jtfSalaire.setText(null);
    }
}

```

```

package Controller;

import java.util.Date;
import java.util.List;

import Model.HolidayModel;
import View.HolidayView;

```

```

import Model.Holiday;
import Model.Holiday.HolidayType;

public class HolidayController {

    private HolidayView view;
    private HolidayModel model;

    public HolidayController(HolidayView view, HolidayModel model){
        this.model=model;
        this.view=view;
        afficherHolidays();
        view.ajouterButton.addActionListener(e -> {
            ajouterHoliday();
            afficherHolidays();
        });

        chargerNomsEmployes();

        view.afficherButton.addActionListener(e -> afficherHolidays());

        this.view.supprimerButton.addActionListener(e -> {
            supprimerHoliday();
            afficherHolidays();
        });

        view.modifierButton.addActionListener(e -> {
            modifierHoliday();
            afficherHolidays();
        });
    }

    public void modifierHoliday() {
        int id = view.getId();

        if (id <= 0) {
            view.afficherMessageErreur("Sélectionnez un congé à modifier.");
            return;
        }

        String employeeName = (String) view.employeeNameComboBox.getSelectedItem();
        Date startDate = view.getStartDate();
        Date endDate = view.getEndDate();
        HolidayType holidayType = view.getHolidayType();

        if (!model.isValidDateRange(startDate, endDate)) {
            view.afficherMessageErreur("Plage de dates invalide.");
            return;
        }

        boolean modificationReussie = model.modifierHoliday(id, employeeName, startDate,
        endDate, holidayType);
        if (modificationReussie) {
            view.afficherMessageSucces("Congé modifié avec succès.");
        } else {
            view.afficherMessageErreur("La modification a échoué.");
        }

        afficherHolidays(); // Recharge la table
    }
}

```

```

public void ajouterHoliday() {

    String employeeName = (String) view.employeeNameComboBox.getSelectedItemAt();
    Date startDate=view.getStartDate(), endDate=view.getEndDate();
    HolidayType holidayType=view.getHolidayType();

    boolean ajoutResult=model.ajouterHoliday(employeeName, startDate, endDate,
holidayType);

    if (ajoutResult) {
        view.afficherMessageSucces("Ajout reussi :");
    }else {
        view.afficherMessageErreur("Ajout failed :");
    }
}

public void afficherHolidays(){

    List<Holiday> holidays=model.afficherHolidays();
    view.afficherHolidays(holidays);
}

public void chargerNomsEmployes() {
    view.employeeNameComboBox.removeAllItems();

    List<String> names = model.chargerNomsEmployes();

    if (names.isEmpty()) {
        System.out.println("Aucun employe trouve.");
    } else {

        for (String name : names) {
            view.employeeNameComboBox.addItem(name);
        }

    }
}

public void supprimerHoliday(){

    int id=view.getId();

    if(id<=0) {
        view.afficherMessageErreur("Selectionner un employee pour le supprimer.");
        return;
    }

    boolean suppressionReussi=model.supprimerHoliday(id);

    if(suppressionReussi) {
        view.afficherMessageSucces("Le conges a ete supprimer :");
    }else {
        view.afficherMessageErreur("Le conges n'a pas ete supprime :");
    }
}
}

```

## Package View :

- **EmployeeView, HolidayView et ManagementInterfaces:**

```
package View;

import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import javax.swing.table.DefaultTableModel;
import java.io.File;
import java.util.List;

import Model.Employee;
import Model.Employee.Role;
import Model.Employee.Poste;

public class EmployeeView extends JFrame {
    private JPanel jp1 = new JPanel(), jp2 = new JPanel(), jp3 = new JPanel(), jp4 = new JPanel();
    private JLabel jlNom = new JLabel("Nom : "), jlPrenom = new JLabel("Prenom : "),
    jlEmail = new JLabel("Email"),
    jlTelephone = new JLabel("Téléphone : "), jlSalaire = new
    JLabel("Salaire : "), jlRole = new JLabel("Role : "),
    jlPoste = new JLabel("Poste : ");
    public JTextField jtfNom = new JTextField(), jtfPrenom = new JTextField(), jtfEmail
    = new JTextField(),
    jtfTelephone = new JTextField(), jtfSalaire = new JTextField();
    private JComboBox<Role> comboboxRole = new JComboBox<>(Role.values());
    private JComboBox<Poste> comboboxPoste = new JComboBox<>(Poste.values());
    private DefaultTableModel tableModel = new DefaultTableModel(new Object[][] {}, new
    String[] { "Id", "Nom", "Prenom", "Email", "Telephone", "Salaire" });
    private JTable jt = new JTable(tableModel);
    public JButton ajouterButton = new JButton("Ajouter"), modifierButton = new
    JButton("Modifier"),
    supprimerButton = new JButton("Supprimer"), afficherButton = new
    JButton("Afficher"),
    importButton = new JButton("Importer"), exportButton = new
    JButton("Exporter");

    public EmployeeView() {
        setTitle("Gestion des employés");
        setSize(600, 400);
        setLocationRelativeTo(null);
        setDefaultCloseOperation(EXIT_ON_CLOSE);

        add(jp1);
        jp1.setLayout(new BorderLayout());
        jp1.add(jp2, BorderLayout.NORTH);
        jp2.setLayout(new GridLayout(7, 2));
        jp2.add(jlNom);
        jp2.add(jtfNom);
        jp2.add(jlPrenom);
        jp2.add(jtfPrenom);
        jp2.add(jlEmail);
        jp2.add(jtfEmail);
        jp2.add(jlTelephone);
        jp2.add(jtfTelephone);
        jp2.add(jlSalaire);
        jp2.add(jtfSalaire);
        jp2.add(jlRole);
        jp2.add(comboboxRole);
        jp2.add(jlPoste);
        jp2.add(comboboxPoste);
    }
}
```

```

jp1.add(jp3, BorderLayout.CENTER);
jp3.setLayout(new BorderLayout());
JScrollPane scrollPane = new JScrollPane(jt);
jp3.add(scrollPane, BorderLayout.CENTER);

jp1.add(jp4, BorderLayout.SOUTH);
jp4.setLayout(new FlowLayout());
jp4.add(ajouterButton);
jp4.add(modifierButton);
jp4.add(supprimerButton);
jp4.add(afficherButton);
jp4.add(importButton); // New Import button
jp4.add(exportButton); // New Export button

jt.addMouseListener(new MouseAdapter() {
    public void mouseClicked(MouseEvent e) {
        int selectedRow = jt.getSelectedRow();
        if (selectedRow != -1) {
            jtfNom.setText(tableModel.getValueAt(selectedRow, 1).toString());
            jtfPrenom.setText(tableModel.getValueAt(selectedRow,
2).toString());
            jtfEmail.setText(tableModel.getValueAt(selectedRow, 3).toString());
            jtfTelephone.setText(tableModel.getValueAt(selectedRow,
4).toString());
            jtfSalaire.setText(tableModel.getValueAt(selectedRow,
5).toString());
        }
    }
});

// Add ActionListener for Import button
importButton.addActionListener(e -> {
    JFileChooser fileChooser = new JFileChooser();
    int result = fileChooser.showOpenDialog(this);
    if (result == JFileChooser.APPROVE_OPTION) {
        File selectedFile = fileChooser.getSelectedFile();
        System.out.println("Selected file for import: " +
selectedFile.getAbsolutePath());
        // Trigger import logic here (to be connected to controller)
    }
});

// Add ActionListener for Export button
exportButton.addActionListener(e -> {
    JFileChooser fileChooser = new JFileChooser();
    int result = fileChooser.showSaveDialog(this);
    if (result == JFileChooser.APPROVE_OPTION) {
        File selectedFile = fileChooser.getSelectedFile();
        System.out.println("Selected file for export: " +
selectedFile.getAbsolutePath());
        // Trigger export logic here (to be connected to controller)
    }
});

// setVisible(true); Uncomment this when testing the UI
}

public int getId() {
    int selectedRow = jt.getSelectedRow();
    if (selectedRow != -1) {

```

```

        Object value = tableModel.getValueAt(selectedRow, 0);
        return Integer.parseInt(value.toString());
    }
    return -1;
}

public String getNom() { return jtfNom.getText(); }
public String getPrenom() { return jtfPrenom.getText(); }
public String getEmail() { return jtfEmail.getText(); }
public String getTelephone() { return jtfTelephone.getText(); }
public double getSalaire() { return Double.parseDouble(jtfSalaire.getText()); }
public Role getRole() { return (Role) comboBoxRole.getSelectedItem(); }
public Poste getPoste() { return (Poste) comboBoxPoste.getSelectedItem(); }

public void afficherEmployees(List<Employee> employees) {
    tableModel.setRowCount(0);
    for (Employee elm : employees) {
        tableModel.addRow(new Object[] {
            elm.getId(),
            elm.getNom(),
            elm.getPrenom(),
            elm.getEmail(),
            elm.getTelephone(),
            elm.getSalaire()
        });
    }
}

public void afficherMessageErreur(String message) {
    JOptionPane.showMessageDialog(this, message, "Erreur",
JOptionPane.ERROR_MESSAGE);
}

public void afficherMessageSucces(String message) {
    JOptionPane.showMessageDialog(this, message, "Succes",
JOptionPane.INFORMATION_MESSAGE);
}

public String showFileChooser(String title) {
    JFileChooser fileChooser = new JFileChooser();
    fileChooser.setDialogTitle(title);
    int userSelection = fileChooser.showOpenDialog(this);

    if (userSelection == JFileChooser.APPROVE_OPTION) {
        File selectedFile = fileChooser.getSelectedFile();
        return selectedFile.getAbsolutePath();
    }
    return null;
}
}

```

```

package View;

import java.awt.*;
import java.util.Calendar;
import java.util.Date;
import java.util.List;
import java.awt.event.MouseAdapter;
import java.awt.event.MouseEvent;
import java.text.SimpleDateFormat;

```



```

import javax.swing.*;
import javax.swing.table.DefaultTableModel;

import Model.Employee;
import Model.Holiday;
import Model.Holiday.HolidayType;

public class HolidayView extends JFrame{

    JPanel jp1=new JPanel(), jp2=new JPanel(), jp3=new JPanel(), jp4=new JPanel();

    public JLabel jlNom=new JLabel("Id de l'employe : "), jlType=new JLabel("Type : "),
    jlDateDebut=new JLabel("Date de debut : "), jlDateFin=new JLabel("Date de fin : ");
    public JComboBox<String> employeeNameComboBox = new JComboBox<>();

    public JComboBox<HolidayType> holidayType=new JComboBox<>(HolidayType.values());
    public Calendar calendar=Calendar.getInstance();

    public JSpinner startDateSpinner=new JSpinner(new SpinnerDateModel(calendar.getTime(),
    null, null, Calendar.DAY_OF_MONTH));
    private JSpinner.DateEditor startDateEditor=new JSpinner.DateEditor(startDateSpinner,
    "dd/MM/yyyy");//startDateSpinner is going to be added

    public JSpinner endDateSpinner=new JSpinner(new SpinnerDateModel(calendar.getTime(), null,
    null, Calendar.DAY_OF_MONTH));
    private JSpinner.DateEditor endDateEditor=new JSpinner.DateEditor(endDateSpinner,
    "dd/MM/yyyy");

    public DefaultTableModel tableModel=new DefaultTableModel(new Object[] [] {}, new String[]
    {"Id", "Employe", "Date de bebut", "Date de fin", "Type", "solde"});
    public JTable jt=new JTable(tableModel);

    public JButton ajouterButton=new JButton("Ajouter"), modifierButton=new
    JButton("Modifier"), supprimerButton=new JButton("Supprimer"), afficherButton=new
    JButton("Afficher");

    public HolidayView(){

        setTitle("Gestion des congés");
        setLocationRelativeTo(null);
        setSize(600, 400);
        setDefaultCloseOperation(EXIT_ON_CLOSE);

        add(jp1);
        jp1.setLayout(new BorderLayout());
        jp1.add(jp2, BorderLayout.NORTH);
        jp2.setLayout(new GridLayout(4, 2));
        jp2.add(jlNom);
        jp2.add(employeeNameComboBox);
        jp2.add(jlType);
        jp2.add(holidayType);
        startDateSpinner.setEditor(startDateEditor);
        jp2.add(jlDateDebut);
        endDateSpinner.setEditor(endDateEditor);
        jp2.add(startDateSpinner);
        jp2.add(jlDateFin);
        jp2.add(endDateSpinner);

        jp1.add(jp3, BorderLayout.CENTER);
        jp3.setLayout(new BorderLayout());
        JScrollPane jsp=new JScrollPane(jt);
        jp3.add(jsp, BorderLayout.CENTER);
    }
}

```

```

jp1.add(jp4, BorderLayout.SOUTH);
jp4.setLayout(new FlowLayout());
jp4.add(ajouterButton);
jp4.add(modifierButton);
jp4.add(supprimerButton);
jp4.add(afficherButton);

jt.addMouseListener(new MouseAdapter() {
    @Override
    public void mouseClicked(MouseEvent e) {
        if (e.getClickCount() == 2) {
            int selectedRow = jt.getSelectedRow();
            if (selectedRow != -1) {
                try {
employeeNameComboBox.setSelectedItem(tableModel.getValueAt(selectedRow, 1).toString());

                    Object startDateObj = tableModel.getValueAt(selectedRow, 2);
                    if (startDateObj instanceof Date) {
                        startDateSpinner.setValue(startDateObj);
                    } else {
                        SimpleDateFormat sdf = new SimpleDateFormat("dd/MM/yyyy");
                        startDateSpinner.setValue(sdf.parse(startDateObj.toString()));
                    }

                    Object endDateObj = tableModel.getValueAt(selectedRow, 3);
                    if (endDateObj instanceof Date) {
                        endDateSpinner.setValue(endDateObj);
                    } else {
                        startDateSpinner.setValue(new
SimpleDateFormat("dd/MM/yyyy").parse(endDateObj.toString()));
                    }

holidayType.setSelectedItem(HolidayType.valueOf(tableModel.getValueAt(selectedRow,
4).toString()));
                } catch (Exception ex) {
                    afficherMessageErreur("Erreur lors de la récupération des données :
" + ex.getMessage());
                }
            }
        }
    }
});

//setVisible(true);

}

public void afficherHolidays(List<Holiday> holidays) {
    tableModel.setRowCount(0);
    for (Holiday elm : holidays) {
        tableModel.addRow(new Object[] {
            elm.getId(),
            elm.getEmployeeNom(),
            elm.getStartDate(),
            elm.getEndDate(),
            elm.getHolidayType().name(),
            elm.getSolde()
        });
    }
}
}

```

```

public int getId() {
    int selectedRow = jt.getSelectedRow();
    if (selectedRow != -1) {
        Object value = tableModel.getValueAt(selectedRow, 0);
        return Integer.parseInt(value.toString());
    }
    return -1;
}

public HolidayType getHolidayType() {
    return (HolidayType) holidayType.getSelectedItem();
}

public Date getStartDate() {
    return (Date) startDateSpinner.getValue();
}

public Date getEndDate() {
    return (Date) endDateSpinner.getValue();
}

public void afficherMessageErreur(String message) {
    JOptionPane.showMessageDialog(this, message, "Erreur", JOptionPane.ERROR_MESSAGE);
}

public void afficherMessageSucces(String message) {
    JOptionPane.showMessageDialog(this, message, "Succes",
JOptionPane.INFORMATION_MESSAGE);
}
}

```

```

package View;

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

public class ManagementInterfaces extends JFrame {
    private JTabbedPane tabbedPane = new JTabbedPane();

    public ManagementInterfaces(EmployeeView employeeView, HolidayView holidayView) {
        setTitle("Gestion des Employes et Conges");
        setSize(800, 600);
        setLocationRelativeTo(null);
        setDefaultCloseOperation(EXIT_ON_CLOSE);

        tabbedPane.addTab("Gestion des Employes", employeeView.getContentPane());
        tabbedPane.addTab("Gestion des Conges", holidayView.getContentPane());

        add(tabbedPane);

        setVisible(true);
    }

    public static void main(String[] args) {
        new ManagementInterfaces(new EmployeeView(), new HolidayView());
    }
}

```

## Main:

```
import View.*;
import Model.*;
import Controller.*;
import DAO.*;

public class Main {

    public static void main(String[] args) {

        EmployeeDAOImpl employeeDAO = new EmployeeDAOImpl();
        EmployeeModel employeeModel = new EmployeeModel(employeeDAO);

        HolidayDAOImpl holidayDAO = new HolidayDAOImpl();
        HolidayModel holidayModel = new HolidayModel(holidayDAO);

        EmployeeView employeeView = new EmployeeView();
        HolidayView holidayView = new HolidayView();

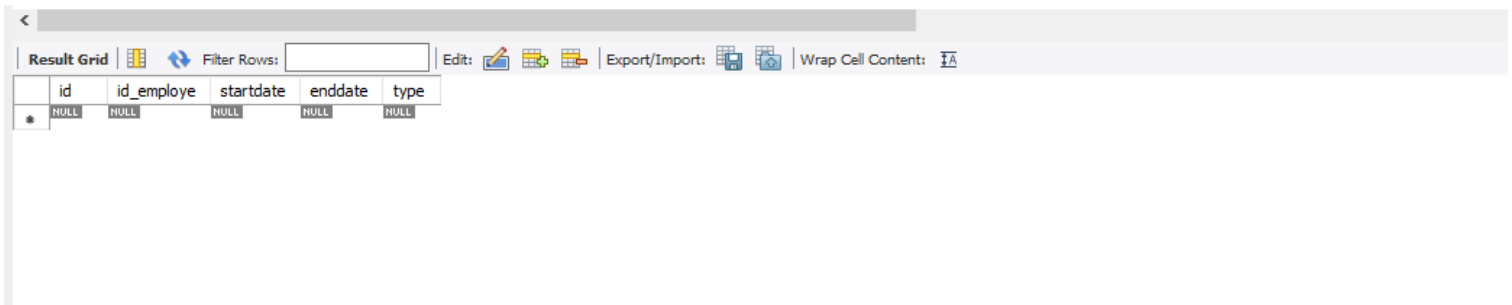
        new EmployeeController(employeeView, employeeModel);
        new HolidayController(holidayView, holidayModel);

        ManagementInterfaces combinedView = new ManagementInterfaces(employeeView,
holidayView);

        combinedView.setVisible(true);
    }
}
```

# Résultats

## 4.TableBD :

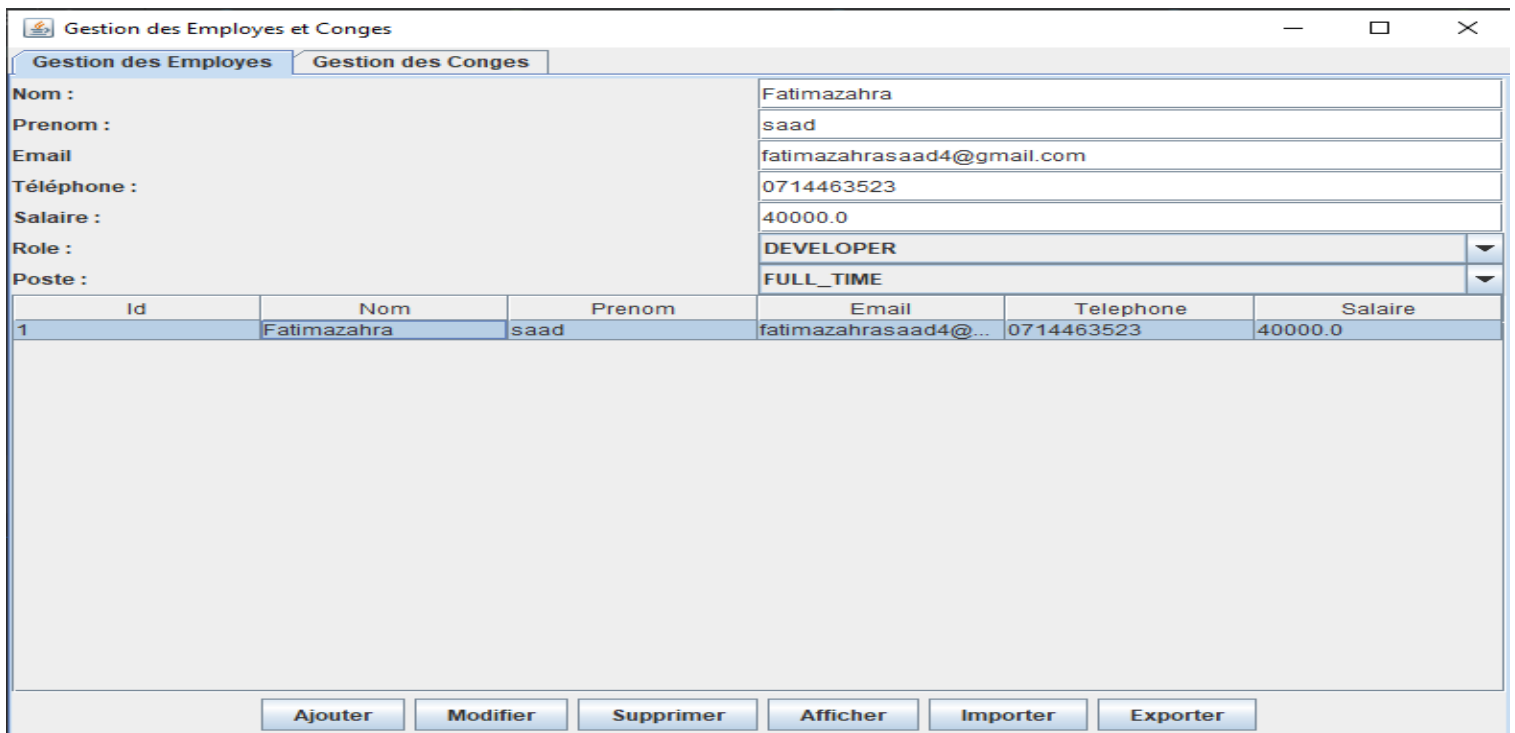


Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content: |

id	id_employe	startdate	enddate	type
NULL	NULL	NULL	NULL	NULL

Figure 6 database

## 5. View :



Gestion des Employes et Conges

Gestion des Employes | Gestion des Conges

Nom : Fatimazahra

Prenom : saad

Email : fatimazahrasaad4@gmail.com

Téléphone : 0714463523

Salaire : 40000.0

Role : DEVELOPER

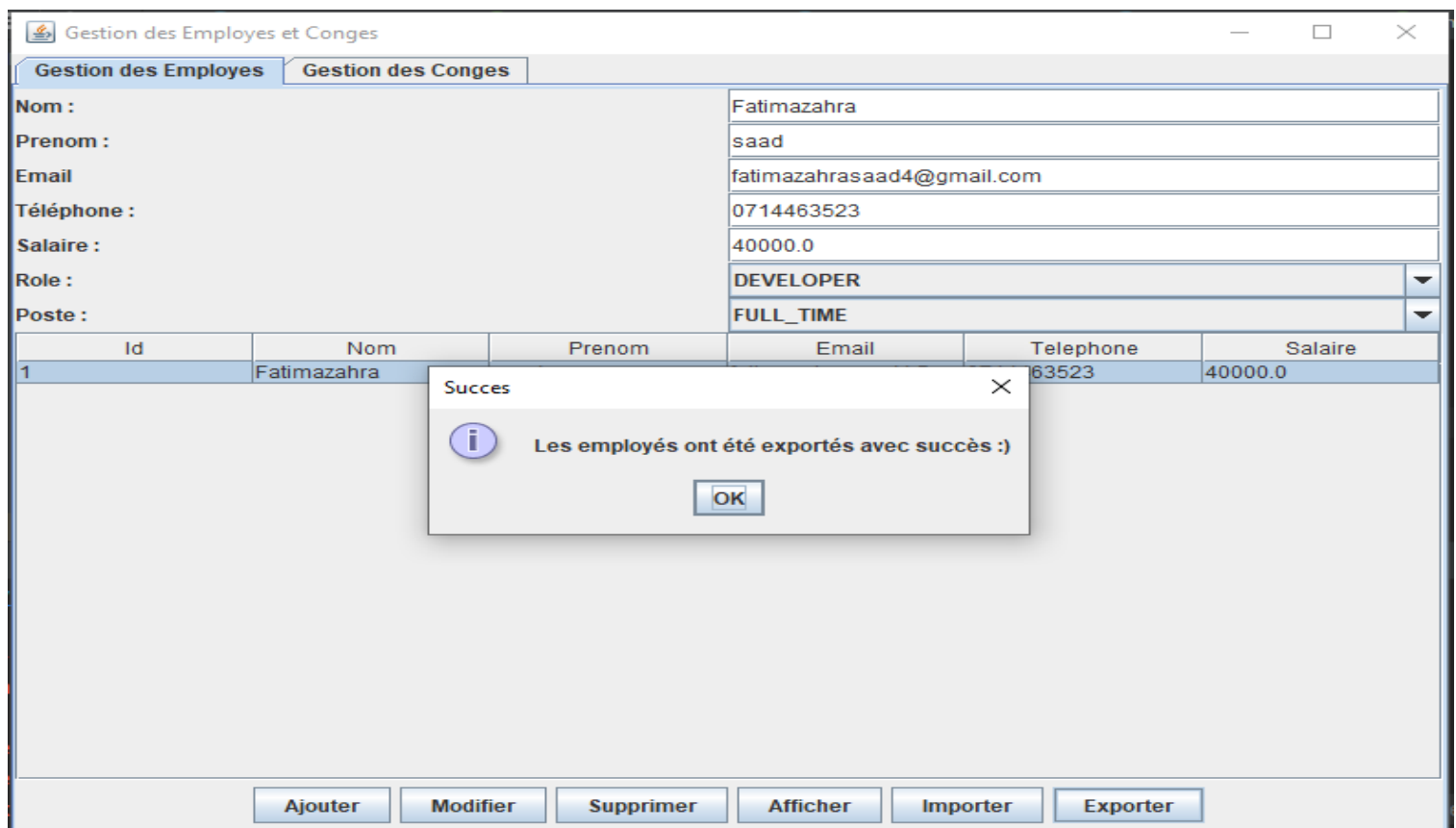
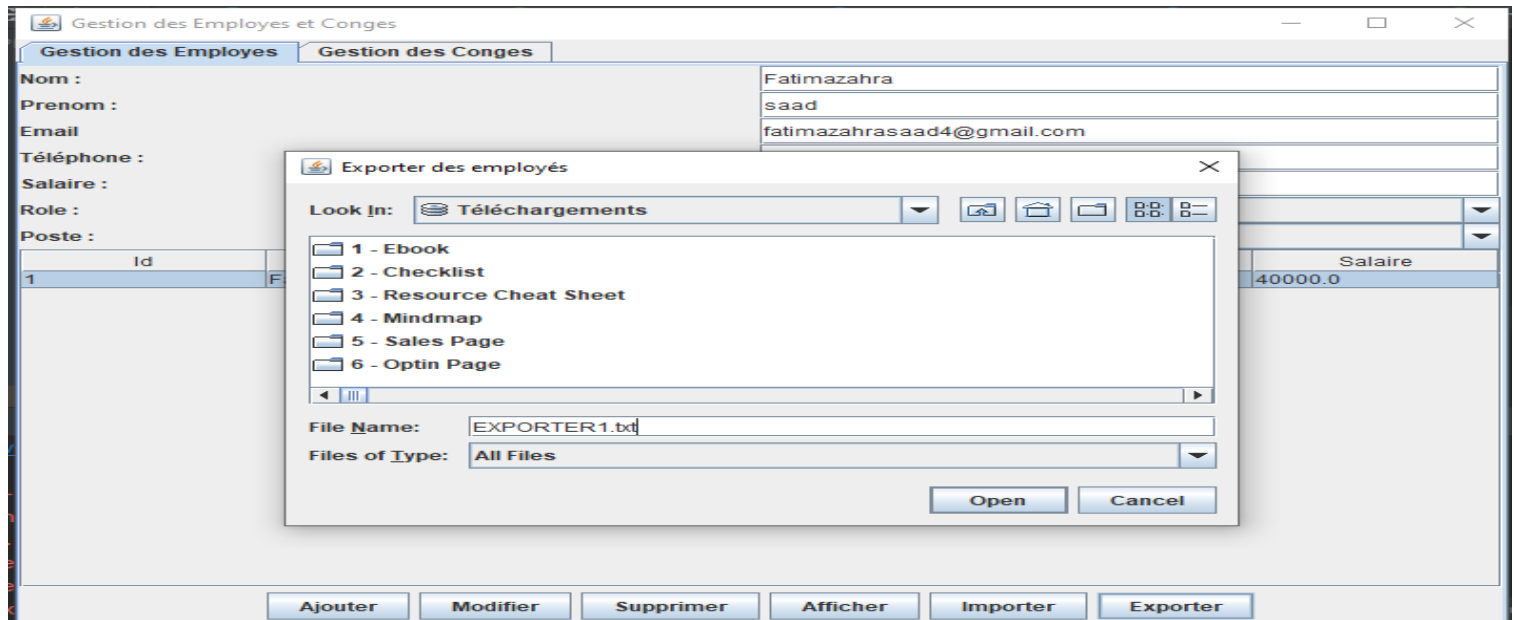
Poste : FULL\_TIME

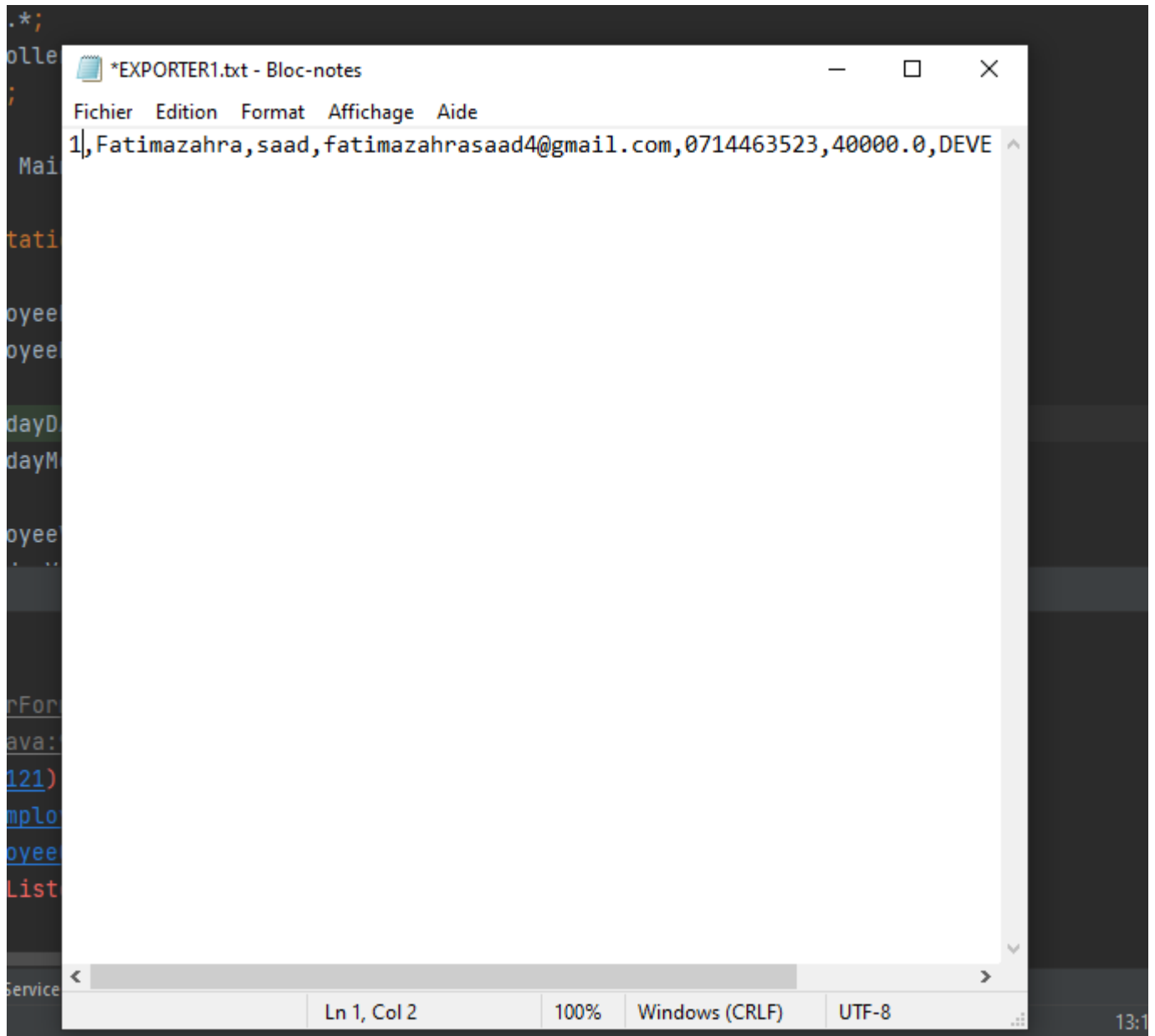
Id	Nom	Prenom	Email	Telephone	Salaire
1	Fatimazahra	saad	fatimazahrasaad4@...	0714463523	40000.0

Ajouter | Modifier | Supprimer | Afficher | Importer | Exporter

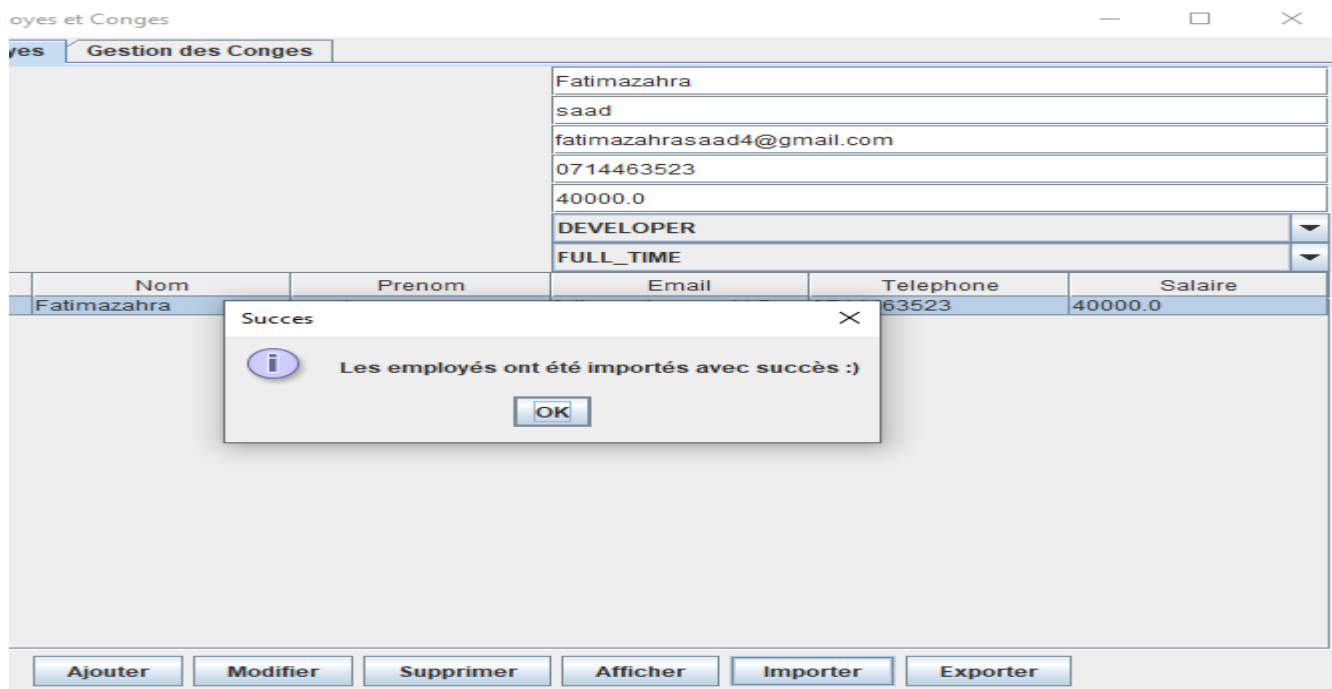
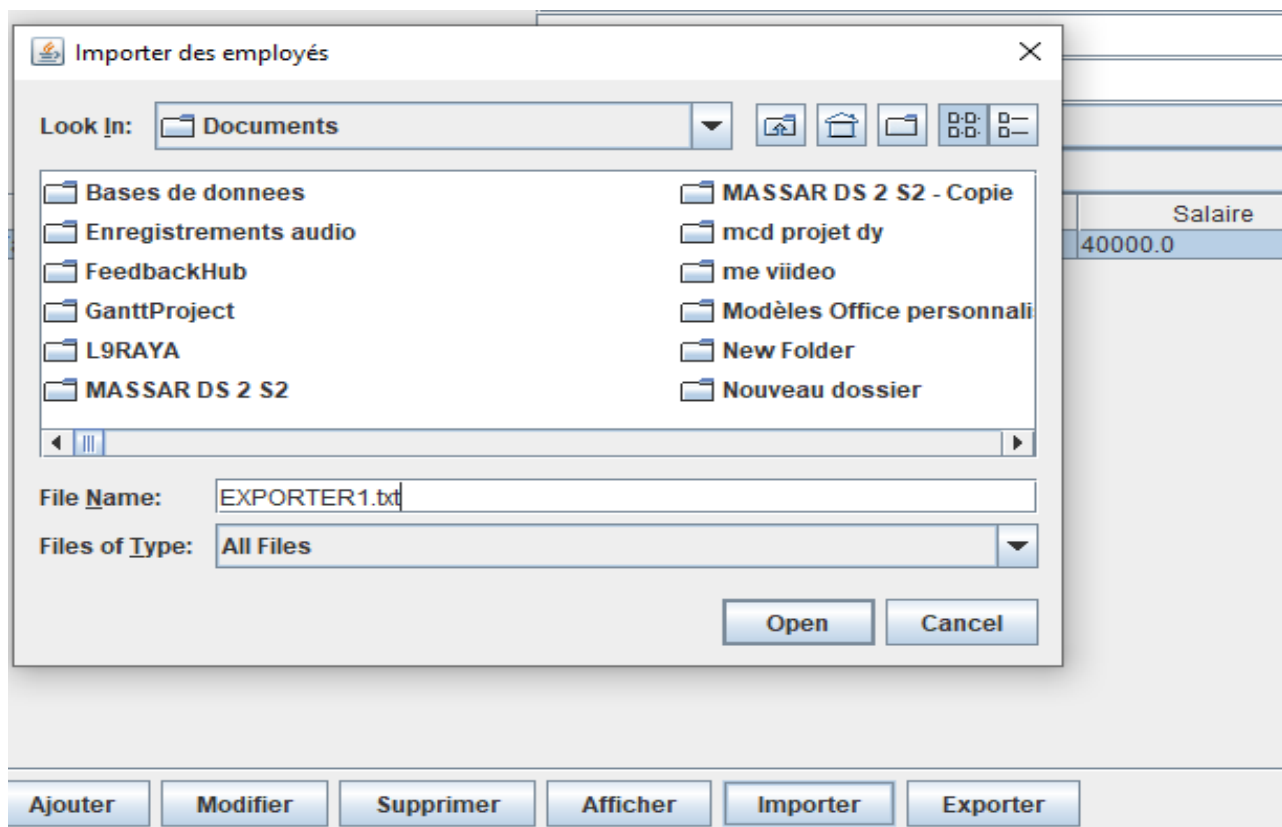
Figure 7 view

## 6. Exportemploye :






## 7. Import Employé :





 Gestion des Employes et Conges

Gestion des Employes

Gestion des Conges

Nom :

Fatimazahra

Prenom :

saad

Email

fatimazahrasaad4@gmail.com

Téléphone :

0714463523

Salaire :

40000.0

Role :

DEVELOPER

Poste :

FULL\_TIME

Id	Nom	Prenom	Email	Telephone	Salaire
1	Fatimazahra	saad	fatimazahrasaad4@...	0714463523	40000.0
1	Fatimazahra	saad	fatimazahrasaad4@...	0714463523	40000.0

Ajouter

Modifier

Supprimer

Afficher

Importer

Exporter

## *Conclusion générale*

En résumé, ce TP a permis de développer une application d'export et import les informations de l'employé en adoptant l'architecture **MVC**. Cette approche a permis de bien séparer les responsabilités entre la logique métier, l'interface utilisateur et la gestion des données d'employé et de congés, assurant ainsi une solution modulaire, facile à maintenir et évolutive. L'ajout de fonctionnalités telles que la création, la modification et la suppression d'employés et de congé a renforcé notre maîtrise des principes de la programmation orientée objet ainsi que de la gestion des interfaces graphiques en Java. Ce projet met en évidence l'importance d'une structure claire et d'une organisation rigoureuse du code pour concevoir des applications fiables et performantes.

## *Références*

*java :*

— <https://www.java.com/en/download/>

*Intellig :*

— <https://www.jetbrains.com/idea/>

*XAMPP :*

— <https://www.apachefriends.org/fr/index.html>

*jdk 23 :*

— <https://www.oracle.com/java/technologies/downloads/>

