

UNIVERSITÉ CADI AYAD
ECOLE SUPERIEURE DE TECHNOLOGIE-SAFI
DUT GÉNIE INFORMATIQUE

Compte rendu

TP 3 : Système de Gestion des Congés
(Genericite, MVC, DAO, Swing, E/S)

Réalisée par :

SAISSI Zahra

Enseigné par :

M.EL ABDELLAOUI Said

Année universitaire :2024/2025

Table des matières

Introduction	2
Objectifs	2
Lien GitHub:	2
Étapes de Création du Projet	3
1. 1ère étape : Création de l'interface générique et son implémentation	3
1.1. Interface DataImportExport:	3
1.2. Importation des Données Employés depuis un Fichier Texte	3
1.3. Exportation des Données Employés depuis un Fichier Texte	4
2. 2ème étape : Couche Model -Logique metier	5
3. 3ème étape : Couche View - Interface graphique	5
3.1. Les boutons d'importation et d'exportation	5
3.2. L'action d'importation et d'exportation	6
3.3. Les méthodes d'exportation et d'importation des données (exportData et importData)	7
4. 4ème étape : Main - Application principale	8
Réalisation	10
1. Page Login:	10
2. Authentification:	10
3. Affichage des employes et congés:	10
4. Exportation des employes:	11
5. Exportation des congés:	13
6. Importation des employes:	14
7. Importation des congés:	16
Conclusion	19

Introduction

Dans le cadre de l'amélioration du système de gestion des congés, l'entreprise SEA a décidé d'intégrer des mécanismes d'entrées/sorties (E/S) afin d'ajouter des fonctionnalités d'importation et d'exportation des données. Ces nouvelles fonctionnalités, basées sur le modèle MVC et l'architecture DAO, permettront de renforcer la flexibilité et l'efficacité de l'application en facilitant la gestion des employés et des congés via des fichiers externes.

Objectifs

L'objectif principal est d'étendre l'application existante pour inclure des fonctionnalités avancées d'import/export des données. Plus précisément, il s'agit de permettre :

- L'exportation de la liste complète des employés.
- L'importation de nouvelles données concernant les employés.
- L'exportation ciblée des employés bénéficiant de congés.

Lien GitHub:

<https://github.com/saissizahra/GestionRessourcesHumain.git>

Étapes de Création du Projet

1. 1ère étape : Création de l'interface générique et son implémentation

1.1. Interface DataImportExport:

Dans cette première étape, nous mettons en place une interface générique DataImportExport qui définit les méthodes nécessaires pour importer et exporter des données. L'objectif est de permettre à l'application de lire des données à partir de fichiers (par exemple, CSV ou TXT) et d'exporter des informations vers ces fichiers, tout en restant flexible pour différents types de données grâce à la généricité.

Code :

```
1 package DAO;
2
3 import java.io.IOException;
4 import java.util.List;
5
6 //Interface générique pour l'importation et l'exportation des données.
7
8 public interface DataImportExport<T> {
9     void importData(String fileName) throws IOException;
10    void exportData(String fileName, List<T> data) throws IOException;
11 }
```

1.2. Importation des Données Employés depuis un Fichier Texte

Dans la classe EmployeDAOImpl, cette méthode permet d'importer les données des employés depuis un fichier texte dans la base de données. Le fichier est lu ligne par ligne avec un BufferedReader. Chaque ligne est ensuite divisée en différentes valeurs (nom, prénom, email, etc.) à l'aide de la méthode split. Si le format est valide (7 éléments), ces valeurs sont insérées dans la base de données via une requête INSERT préparée. Après avoir ajouté toutes les lignes valides au lot de la requête, l'insertion est effectuée en une seule opération avec ps.executeBatch(). Si une ligne a un format incorrect, un message d'erreur est affiché.

Code :

```

108 //Importation des données des employés depuis un fichier texte vers la base de données.
109
110 public void importData(String filePath) {
111     String query = "INSERT INTO Employee(nom, prenom, email, telephone, salaire, role, poste) VALUES (?, ?, ?, ?, ?, ?, ?)";
112     Connection conn = null;
113     try (BufferedReader reader = new BufferedReader(new FileReader(filePath));
114         PreparedStatement ps = conn.prepareStatement(query)) {
115
116         String line = reader.readLine();
117         while ((line = reader.readLine()) != null) {
118             String[] data = line.split(",");
119             if (data.length == 7) {
120                 ps.setString(1, data[0].trim()); // nom
121                 ps.setString(2, data[1].trim()); // prenom
122                 ps.setString(3, data[2].trim()); // email
123                 ps.setString(4, data[3].trim()); // telephone
124                 ps.setString(5, data[4].trim()); // salaire
125                 ps.setString(6, data[5].trim()); // role
126                 ps.setString(7, data[6].trim()); // poste
127                 ps.addBatch();
128             } else {
129                 System.err.println("Invalid data format:" + line);
130             }
131         }
132         ps.executeBatch();
133         System.out.println("Les employés ont été importés avec succès !");
134     } catch (IOException | SQLException e) {
135         e.printStackTrace();
136     }
137 }

```

1.3. Exportation des Données Employés depuis un Fichier Texte

Dans cette méthode, située également dans la classe `EmployeDAOImpl`, les données des employés sont exportées vers un fichier texte à l'aide d'un `BufferedWriter`. Le fichier est créé ou ouvert avec un `FileWriter`, et la première ligne contient les en-têtes des colonnes (nom, prénom, email, etc.). Ensuite, pour chaque employé dans la liste, une ligne est formatée avec les informations correspondantes (nom, prénom, email, etc.) et écrite dans le fichier. Après chaque ligne, un saut de ligne est ajouté pour séparer les enregistrements. Une fois l'exportation terminée, un message confirme le succès de l'opération.

Code :

```

138 //Exporter des données des employés dans un fichier texte
139
140 public void exportData(String fileName, List<Employe> data) throws IOException {
141     try (BufferedWriter writer = new BufferedWriter(new FileWriter(fileName))) {
142         writer.write("nom,prenom,email,telephone,role,poste,salaire");
143         writer.newLine();
144         for (Employe employee : data) {
145             String line = String.format("%s,%s,%s,%s,%s,%s,%.2f",
146                 employee.getNom(),
147                 employee.getPrenom(),
148                 employee.getEmail(),
149                 employee.getTelephone(),
150                 employee.getRole(),
151                 employee.getPoste(),
152                 employee.getSalaire());
153             writer.write(line);
154             writer.newLine();
155         }
156         System.out.println("Les employés ont été exportés avec succès !");
157     }
158 }
159 }

```

2. 2ème étape : Couche Model -Logique metier

Dans cette étape, située dans la couche **Model**, plusieurs vérifications sont mises en place pour garantir que le fichier spécifié est valide avant d'être utilisé pour l'import ou l'export des données.

1. **Vérification de l'existence du fichier** : La méthode `checkFileExists` vérifie si le fichier existe bien dans le système. Si le fichier n'existe pas, une exception est levée avec un message d'erreur approprié.
2. **Vérification du type de fichier** : La méthode `checkIsFile` s'assure que le chemin spécifié pointe bien vers un fichier et non un répertoire. Si ce n'est pas le cas, une exception est levée.
3. **Vérification des droits de lecture** : La méthode `checkIsReadable` vérifie que le fichier est lisible par l'application. Si les droits de lecture sont insuffisants, une exception est levée.

Code :

```
68 // Vérifie que le fichier existe dans le système.
69 private boolean checkFileExists(File file) {
70     if (!file.exists()) {
71         throw new IllegalArgumentException("Le fichier n'existe pas " + file.getPath());
72     }
73     return true;
74 }
75
76 // Vérifie que le chemin spécifié est bien un fichier et non un répertoire.
77 private boolean checkIsFile(File file) {
78     if (!file.isFile()) {
79         throw new IllegalArgumentException("Le chemin spécifié n'est pas un fichier " + file.getPath());
80     }
81     return true;
82 }
83
84 // Vérifie que l'application a les droits de lecture sur le fichier.
85 private boolean checkIsReadable(File file) {
86     if (!file.canRead()) {
87         throw new IllegalArgumentException("Le fichier spécifié n'est pas lisible " + file.getPath());
88     }
89     return true;
90 }
91 }
```

3. 3ème étape : Couche View - Interface graphique

3.1. Les boutons d'importation et d'exportation

Dans l'interface de gestion des employés, deux boutons ont été ajoutés pour faciliter les opérations d'importation et d'exportation des données. Le bouton "Importer" permet à l'utilisateur de charger un fichier contenant des informations, comme des données sur les employés, et de les ajouter au modèle de la table en affichant les nouvelles informations dans l'interface. De même, le bouton "Exporter" permet de sauvegarder les données actuelles de la table dans un fichier.

Code:

```
65
66 // Les boutons pour gérer les employés
67 private JButton addButton_employe = new JButton("Ajouter");
68 private JButton updateButton_employe = new JButton("Modifier");
69 private JButton deleteButton_employe = new JButton("Supprimer");
70 private JButton displayButton_employe = new JButton("Afficher");
71 public JButton importButton_employe = new JButton("Importer");
72 public JButton exportButton_employe = new JButton("Exporter");
73
74 // Les boutons pour gérer les congés
75 private JButton addButton_holiday = new JButton("Ajouter");
76 private JButton updateButton_holiday = new JButton("Modifier");
77 private JButton deleteButton_holiday = new JButton("Supprimer");
78 private JButton displayButton_holiday = new JButton("Afficher");
79 public JButton importButton_holiday = new JButton("Importer");
80 public JButton exportButton_holiday = new JButton("Exporter");
81
82
```

3.2. L'action d'importation et d'exportation

L'action d'importation est déclenchée lorsque l'utilisateur clique sur le bouton "Importer". Cela ouvre une fenêtre de sélection de fichier, et une fois un fichier choisi, les données sont lues ligne par ligne et ajoutées à la table. Un message de succès informe l'utilisateur de la réussite de l'importation.

L'action d'exportation se lance lorsque l'utilisateur clique sur le bouton "Exporter". Il ouvre une fenêtre pour sélectionner l'emplacement et le nom du fichier, puis les données de la table sont sauvegardées dans le fichier choisi. Un message de confirmation est affiché pour informer l'utilisateur de la réussite de l'exportation.

Code:

```

181 // Action pour importer des données des employés
182 importButton_employe.addActionListener(e -> {
183     JFileChooser fileChooser = new JFileChooser();
184     if (fileChooser.showOpenDialog(this) == JFileChooser.APPROVE_OPTION) {
185         importData(tableModel, fileChooser.getSelectedFile().getPath());
186     }
187 });
188
189 // Action pour exporter les données des employés
190 exportButton_employe.addActionListener(e -> {
191     JFileChooser fileChooser = new JFileChooser();
192     if (fileChooser.showSaveDialog(this) == JFileChooser.APPROVE_OPTION) {
193         exportData(tableModel, fileChooser.getSelectedFile().getPath());
194     }
195 });
196
197 // Action pour importer des données des congés
198 importButton_holiday.addActionListener(e -> {
199     JFileChooser fileChooser = new JFileChooser();
200     if (fileChooser.showOpenDialog(this) == JFileChooser.APPROVE_OPTION) {
201         importData(tableModel1, fileChooser.getSelectedFile().getPath());
202     }
203 });
204 // Action pour exporter les données des congés
205 exportButton_holiday.addActionListener(e -> {
206     JFileChooser fileChooser = new JFileChooser();
207     if (fileChooser.showSaveDialog(this) == JFileChooser.APPROVE_OPTION) {
208         exportData(tableModel1, fileChooser.getSelectedFile().getPath());
209     }
210 });
211

```

3.3. Les méthodes d'exportation et d'importation des données

(exportData et importData)

Les méthodes exportData et importData gèrent respectivement l'exportation et l'importation des données dans l'application. La méthode exportData prend en charge l'enregistrement des données actuelles de la table dans un fichier. Elle commence par écrire les noms des colonnes, suivis des valeurs des lignes, séparées par des virgules. Une fois l'exportation terminée, un message de succès s'affiche pour informer l'utilisateur de la réussite de l'opération. En revanche, la méthode importData s'occupe de charger les données depuis un fichier. Elle lit chaque ligne du fichier, extrait les valeurs séparées par des virgules et les ajoute à la table. Les anciennes données sont supprimées avant l'importation pour éviter toute confusion avec les nouvelles données.

Code:


```

368 //exporter les données contenues dans un DefaultTableModel vers un fichier
369= public void exportData(DefaultTableModel model, String fileName) {
370     try (PrintWriter writer = new PrintWriter(new FileWriter(fileName))) {
371         for (int i = 0; i < model.getColumnCount(); i++) {
372             writer.print(model.getColumnModel().getColumnName(i));
373             if (i < model.getColumnCount() - 1) writer.print(",");
374         }
375         writer.println();
376         for (int i = 0; i < model.getRowCount(); i++) {
377             for (int j = 0; j < model.getColumnCount(); j++) {
378                 writer.print(model.getValueAt(i, j));
379                 if (j < model.getColumnCount() - 1) writer.print(",");
380             }
381             writer.println();
382         }
383         JOptionPane.showMessageDialog(this, "Exportation avec succès.");
384     } catch (Exception e) {
385         JOptionPane.showMessageDialog(this, "Erreur lors de l'exportation : " + e.getMessage(), "Erreur", JOptionPane.ERROR_MESSAGE);
386     }
387 }
388
389 //importer des données à partir d'un fichier et de les ajouter à un DefaultTableModel.
390= public void importData(DefaultTableModel model, String fileName) {
391     try (BufferedReader reader = new BufferedReader(new FileReader(fileName))) {
392         model.setRowCount(0);
393         String line = reader.readLine();
394         while ((line = reader.readLine()) != null) {
395             String[] data = line.split(",");
396             model.addRow(data);
397         }
398         JOptionPane.showMessageDialog(this, "Importation avec succès.");
399     } catch (Exception e) {
400         JOptionPane.showMessageDialog(this, "Erreur lors de l'importation : " + e.getMessage(), "Erreur", JOptionPane.ERROR_MESSAGE);
401     }
402 }
403 }

```

4. 4ème étape : Main - Application principale

La classe Main initialise l'application en créant les objets nécessaires pour la gestion de la connexion des employés et leurs congés. Il crée des instances des classes de gestion des données (DAO), des modèles qui traitent la logique métier, et des vues pour afficher l'interface utilisateur. Lorsqu'un utilisateur entre ses informations de connexion, le contrôleur vérifie si celles-ci sont correctes. Si la connexion réussit, la vue principale apparaît, permettant de gérer les employés et les congés. En cas d'échec, un message d'erreur est affiché.

Code :

```

15 public class Main {
16     public static void main(String[] args) {
17
18         // Crée des instances des DAO pour la gestion des données
19         LoginDAOImpl loginDAO = new LoginDAOImpl();
20         EmployeDAOImpl employeDAO = new EmployeDAOImpl();
21         HolidayDAOImpl holidayDAO = new HolidayDAOImpl();
22
23         // Crée des instances des modèles pour gérer la logique des employés, des congés et de la connexion
24         LoginModel loginModel = new LoginModel(loginDAO);
25         EmployeModel employeModel = new EmployeModel(employeDAO);
26         HolidayModel holidayModel = new HolidayModel(holidayDAO);
27
28         // Crée la vue de connexion et la vue principale de l'application
29         LoginView loginView = new LoginView();
30         MainView employeHolidayView = new MainView();
31
32         // Crée le contrôleur de connexion pour gérer les actions liées à la connexion
33         new LoginController(loginView, loginModel);
34
35         loginView.setVisible(true);
36
37         // Écoute les événements de connexion et affiche la vue principale si la connexion est réussie
38         loginView.addLoginListener(e -> {
39
40             // Si l'authentification est réussie
41             if (loginModel.authenticate(loginView.getUsername(), loginView.getPassword())) {
42                 loginView.setVisible(false);
43
44                 // Initialise les contrôleurs pour la gestion des employés et des congés
45                 new EmployeController(employeHolidayView, employeModel);
46                 new HolidayController(employeHolidayView, holidayModel);
47
48                 // Affiche la vue principale de l'application
49                 employeHolidayView.setVisible(true);
50             } else {
51                 loginView.showError("Nom d'utilisateur et mot de passe incorrects. Essayez à nouveau.");
52             }
53         });
54     }
55 }

```

Réalisation

1. Page Login:

The screenshot shows the 'Gestion des employes et des congés' application window. The 'Employee' tab is selected. A 'Login' dialog box is open in the center, prompting for 'Username:' and 'Password:' with a 'Login' button. The background form contains fields for 'Nom', 'Prenom', 'Email', 'Telephone', 'Salaire', 'Role', and 'Poste'. Below these fields is a table with columns 'ID', 'Nom', 'Prenom', 'Role', 'Poste', and 'solde'. At the bottom are buttons: 'Ajouter', 'Modifier', 'Supprimer', 'Afficher', 'Importer', and 'Exporter'.

2. Authentification:

The screenshot shows the same application window after a successful login. The 'Role' dropdown is now set to 'ADMIN'. An 'Error' dialog box (with a red 'X' icon) displays the message 'Login successful!' and an 'OK' button. The rest of the interface, including the form fields and the bottom buttons, remains the same.

3. Affichage des employes et congés:

Gestion des employes et des congés

Employe Holiday

Nom

Prenom

Email

Telephone

Salaire

Role

Poste

ADMIN

INGENIEURE_ETUDE_ET_DEVELOPPEMENT

ID	Nom	Prenom	Email	Telephone	Salaire	Role	Poste	solde
1	SAISSI	Zahra	zahra@gmail.com	06123456789	1000.0	ADMIN	PILOTE	20
2	KOUIS	khaoula	khaoula@gmail.c.	06234567890	1500.0	EMPLOYE	TEAM_LEADER	22
3	ouhidous	inas	inas@gmail.com	06345678901	1200.0	EMPLOYE	INGENIEURE_E...	12
4	elouati	nissrin	nissrin@gmail.co.	06456789012	1300.0	ADMIN	PILOTE	23

Ajouter Modifier Supprimer Afficher Importer Exporter

Gestion des employes et des congés

Employe Holiday

Nom de l'employe

Date de debut (YYYY-MM-DD)

Date de fin (YYYY-MM-DD)

Type

1 - SAISSI Zahra

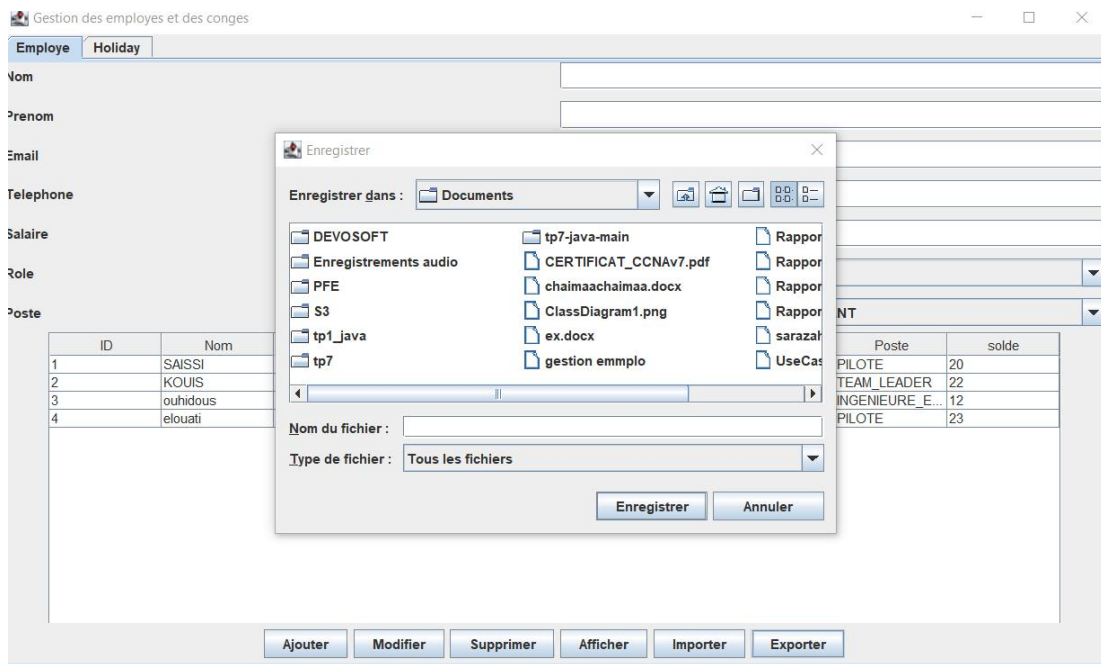
CONGE_PAYE

ID	nom_employe	date_debut	date_fin	type
1	1 - SAISSI Zahra	2025-01-05	2025-01-10	CONGE_PAYE
2	3 - ouhidous inas	2025-01-12	2025-01-15	CONGE_NON_PAYE
3	2 - KOUIS khaoula	2025-03-22	2025-03-25	CONGE_NON_PAYE
4	3 - ouhidous inas	2025-05-18	2025-05-28	CONGE_MALADIE
5	4 - elouati nissrin	2025-01-02	2025-01-04	CONGE_MALADIE

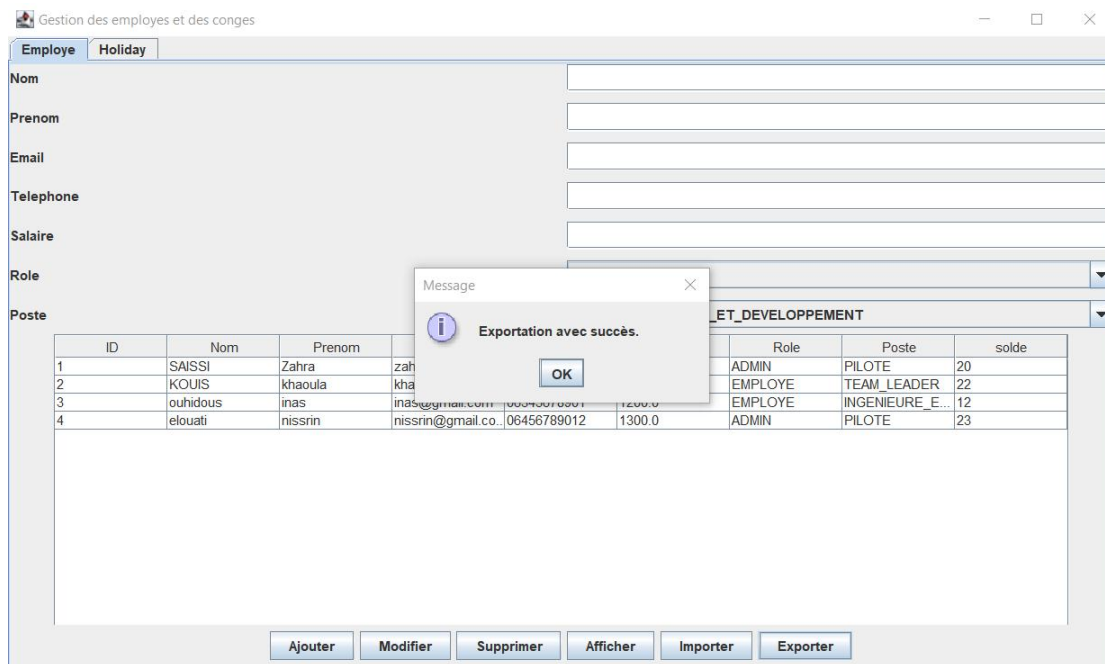
Ajouter Modifier Supprimer Afficher Importer Exporter

4. Exportation des employes:

- Une fois en clique sur le bouton exporter



-En l'enregistre sous le nom test:



- Le fichier test s'affiche dans le bloc note comme si-dessous:

test - Bloc-notes

Fichier Edition Format Affichage Aide

ID,Nom,Prenom,Email,Telephone,Salaire,Role,Poste,solde

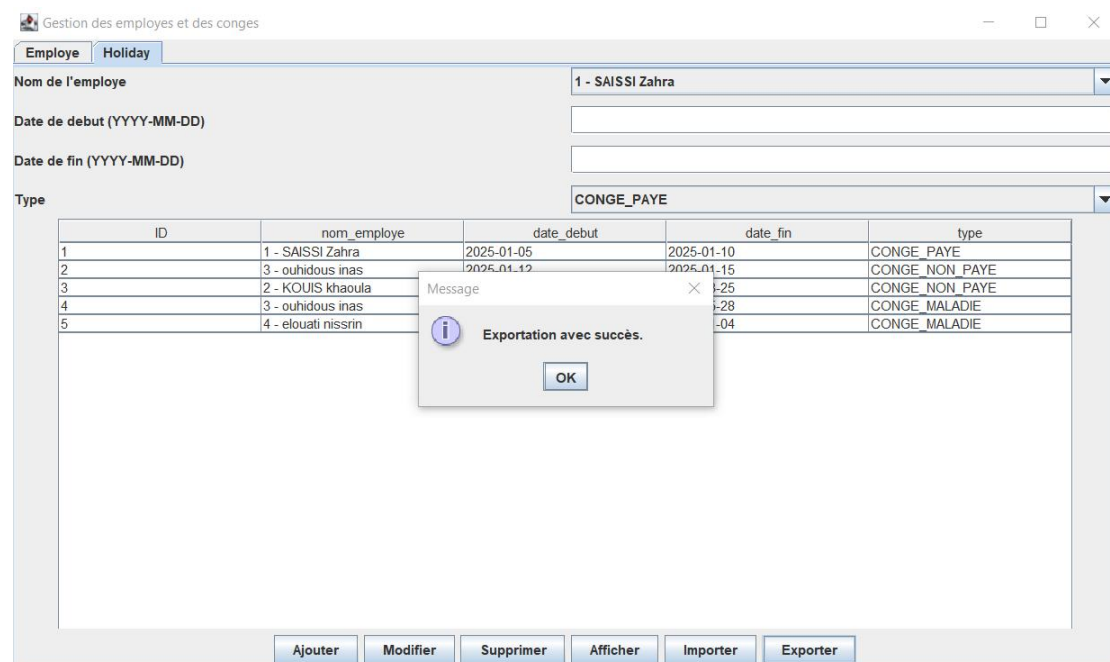
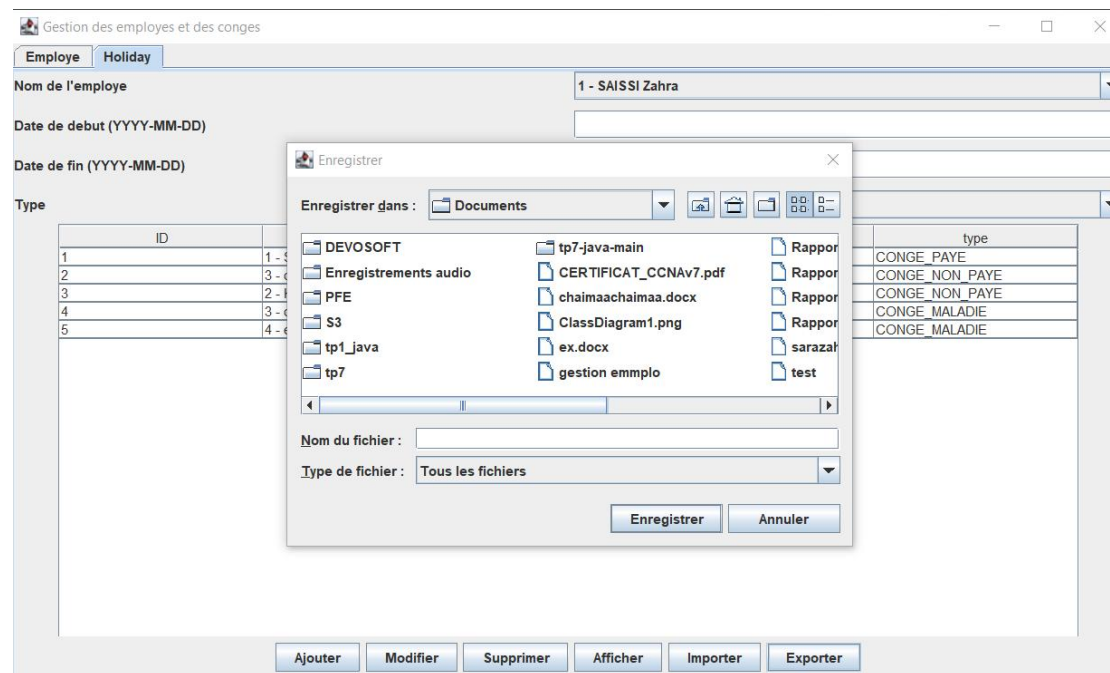
1,SAISSI,Zahra,zahra@gmail.com,06123456789,1000.0,ADMIN,PILOTE,20

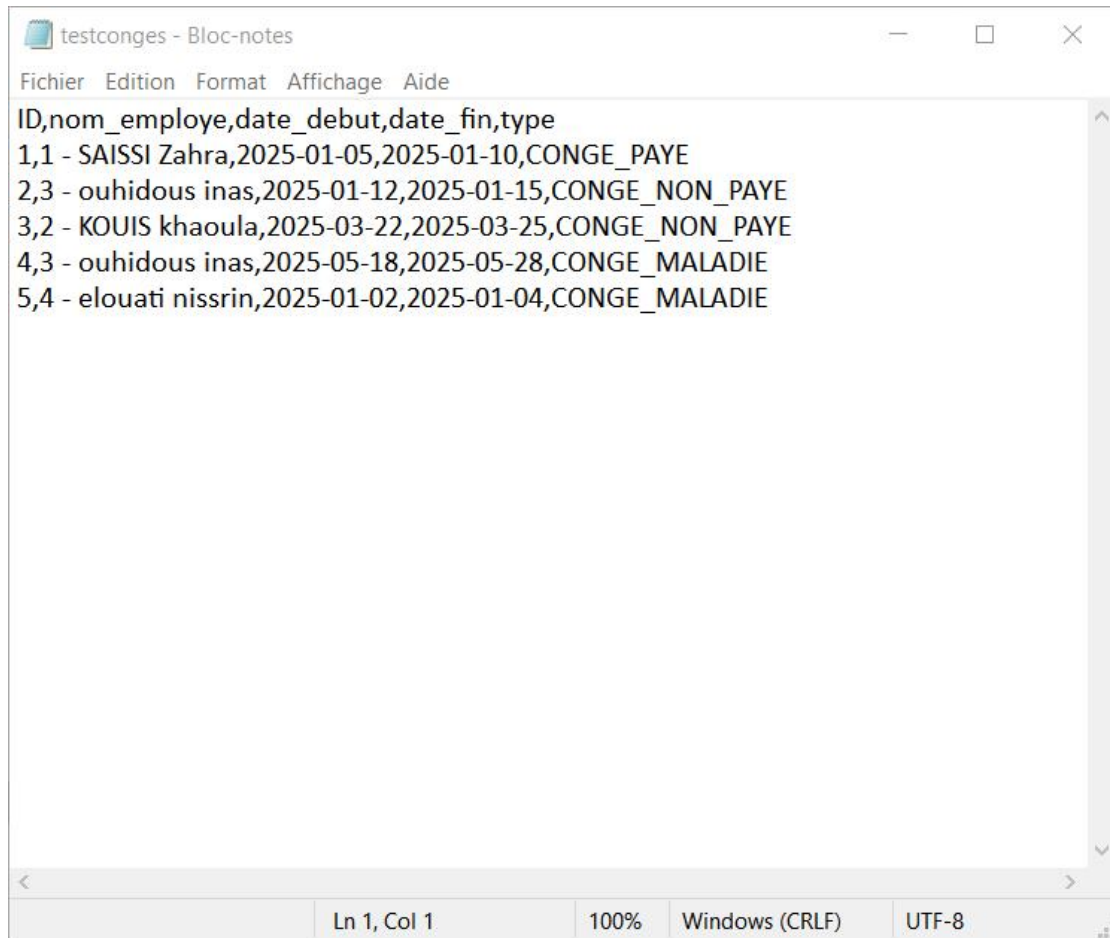
2,KOUIS,khaoula,khaoula@gmail.com,06234567890,1500.0,EMPLOYE,TEAM_LEADER,22

3,ouhidous,inas,inas@gmail.com,06345678901,1200.0,EMPLOYE,INGENIEURE_ETUDE_ET_DEVELOPPEMENT,12

4,elouati,nissrin,nissrin@gmail.com,06456789012,1300.0,ADMIN,PILOTE,23

5. Exportation des congés:

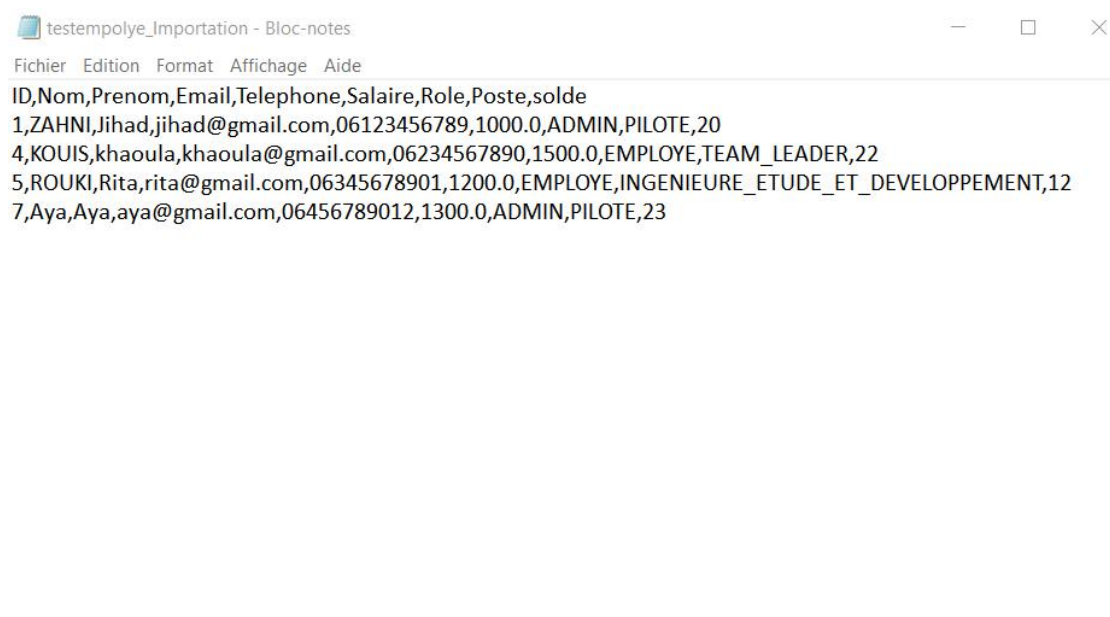




```
testconges - Bloc-notes
Fichier Edition Format Affichage Aide
ID,nom_employe,date_debut,date_fin,type
1,1 - SAISSI Zahra,2025-01-05,2025-01-10,CONGE_PAYE
2,3 - ouhidous inas,2025-01-12,2025-01-15,CONGE_NON_PAYE
3,2 - KOUIS khaoula,2025-03-22,2025-03-25,CONGE_NON_PAYE
4,3 - ouhidous inas,2025-05-18,2025-05-28,CONGE_MALADIE
5,4 - elouati nissrin,2025-01-02,2025-01-04,CONGE_MALADIE
Ln 1, Col 1 100% Windows (CRLF) UTF-8
```

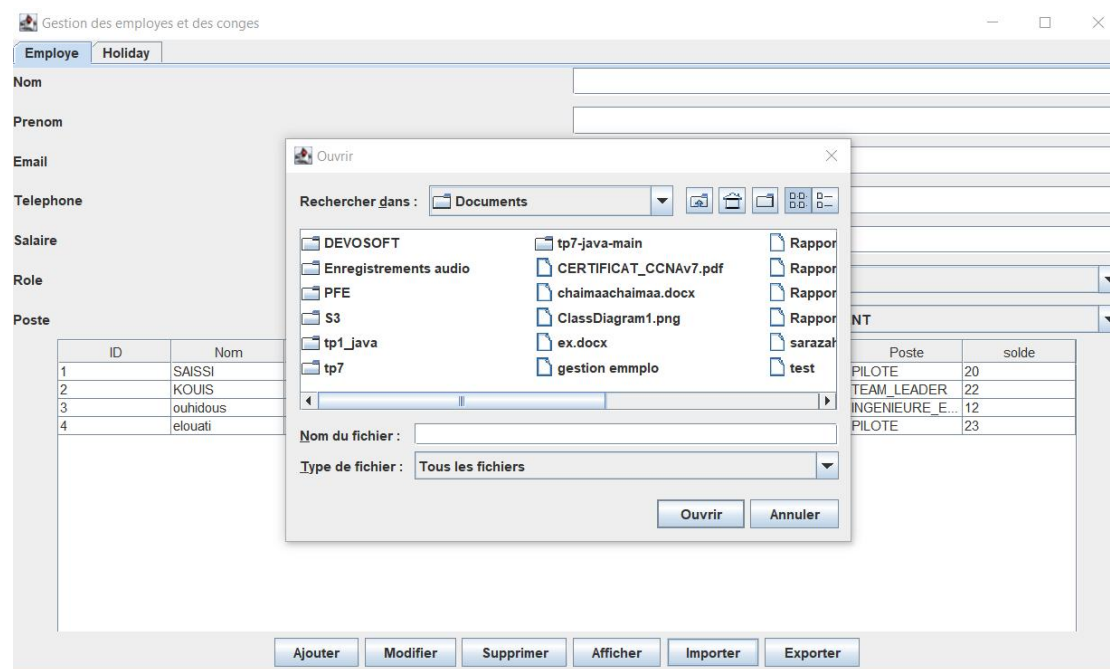
6. Importation des employes:

-On importe un fichier testemplye_Importation:

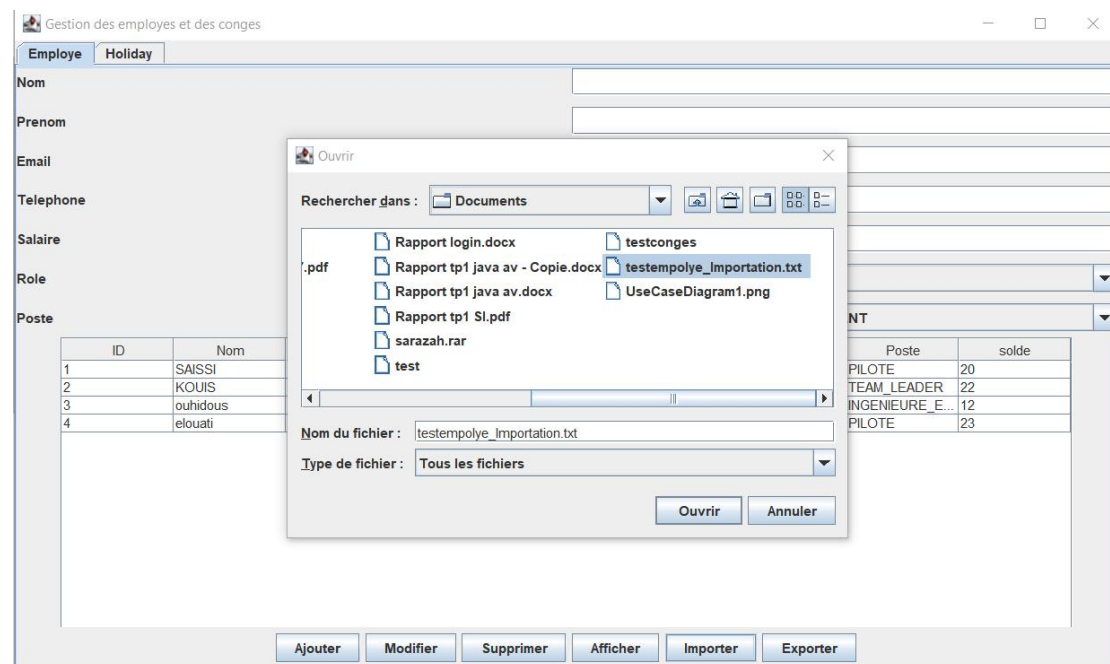


```
testempolye_Importation - Bloc-notes
Fichier Edition Format Affichage Aide
ID,Nom,Prenom,Email,Telephone,Salaire,Role,Poste,solde
1,ZAHNI,Jihad,jihad@gmail.com,06123456789,1000.0,ADMIN,PILOTE,20
4,KOUIS,khaoula,khaoula@gmail.com,06234567890,1500.0,EMPLOYE,TEAM_LEADER,22
5,ROUKI,Rita,rita@gmail.com,06345678901,1200.0,EMPLOYE,INGENIEURE_ETUDE_ET_DEVELOPPEMENT,12
7,Aya,Aya,aya@gmail.com,06456789012,1300.0,ADMIN,PILOTE,23
```


-On clique sur importer:



-On choisi notre fichier:



-on l'ouvre:

Gestion des employes et des congés

Employee Holiday

Nom

Prenom

Email

Telephone

Salaire

Role

Poste

Message

Importation avec succès.

OK

ID	Nom	Prenom	Email	Telephone	Salaire	Role	Poste	solde
1	ZAHNI	Jihad	jihad@gmail.com	06123456789	1000.0	ADMIN	PILOTE	20
4	KOUIS	khaoula	khaoula@gmail.com	06234567890	1500.0	EMPLOYE	TEAM_LEADER	22
5	ROUKI	Rita	rita@gmail.com	06345678901	1200.0	EMPLOYE	INGENIEURE_E...	12
7	Aya	Aya	aya@gmail.com	06456789012	1300.0	ADMIN	PILOTE	23

Ajouter Modifier Supprimer Afficher Importer Exporter

Gestion des employes et des congés

Employee Holiday

Nom

Prenom

Email

Telephone

Salaire

Role

Poste

ADMIN

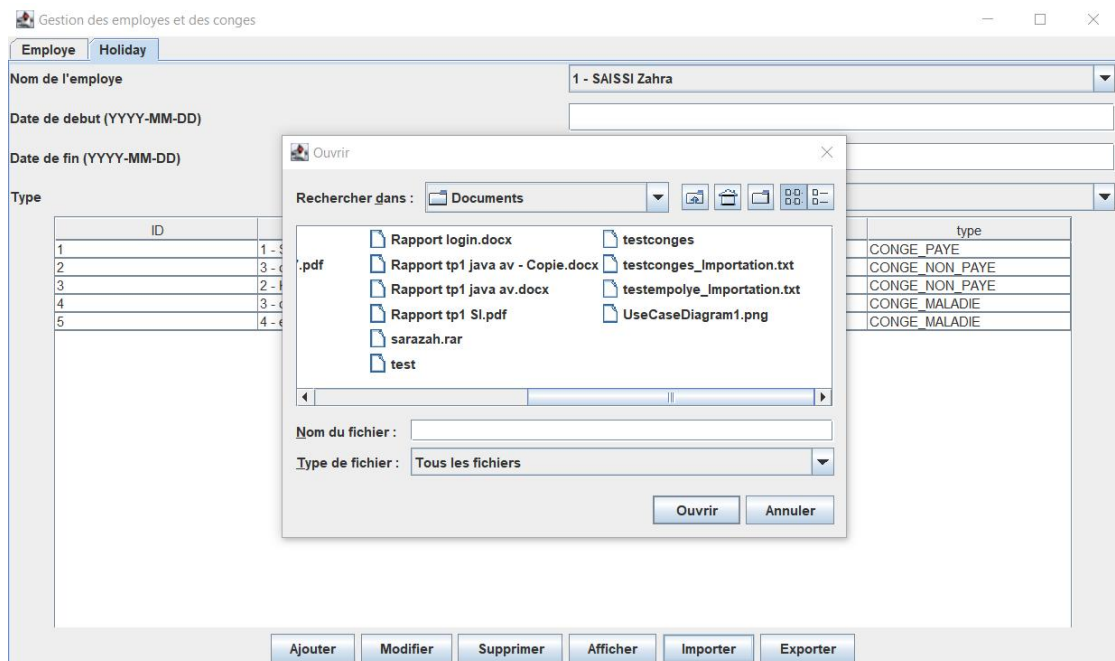
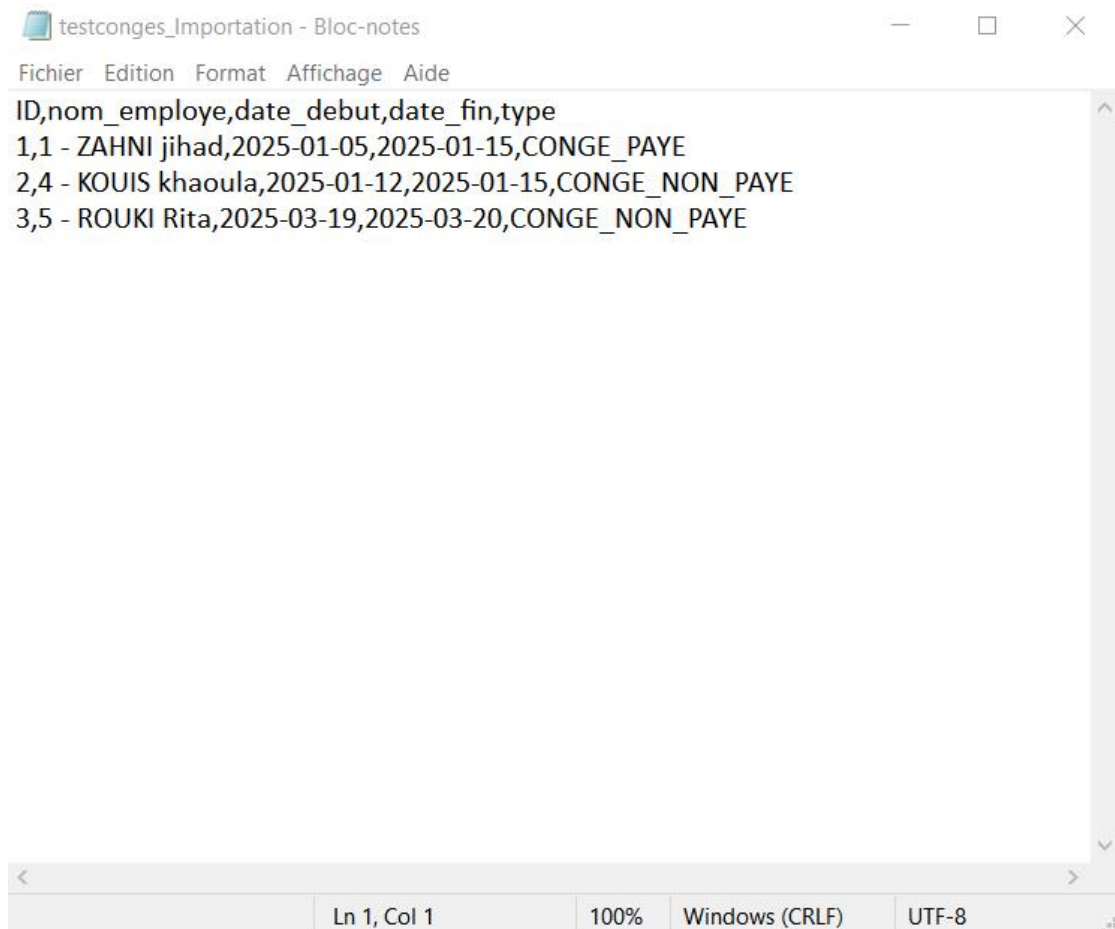
INGENIEURE_ETUDE_ET_DEVELOPPEMENT

ID	Nom	Prenom	Email	Telephone	Salaire	Role	Poste	solde
1	ZAHNI	Jihad	jihad@gmail.com	06123456789	1000.0	ADMIN	PILOTE	20
4	KOUIS	khaoula	khaoula@gmail.c...	06234567890	1500.0	EMPLOYE	TEAM_LEADER	22
5	ROUKI	Rita	rita@gmail.com	06345678901	1200.0	EMPLOYE	INGENIEURE_E...	12
7	Aya	Aya	aya@gmail.com	06456789012	1300.0	ADMIN	PILOTE	23

Ajouter Modifier Supprimer Afficher Importer Exporter

7. Importation des congés:

-On suit les memes etapes:



Gestion des employes et des congés

Employee Holiday

Nom de l'employe: 1 - SAISSI Zahra

Date de debut (YYYY-MM-DD):

Date de fin (YYYY-MM-DD):

Type: CONGE_PAYE

ID	nom_employe	date_debut	date_fin	type
1	1 - ZAHNI jhad	2025-01-05	2025-01-15	CONGE_PAYE
2	4 - KOUIS khaoula	2025-01-12	2025-01-15	CONGE_NON_PAYE
3	5 - ROUKI Rita		2025-03-20	CONGE_NON_PAYE

Message

Importation avec succès.

OK

Ajouter Modifier Supprimer Afficher Importer Exporter

Gestion des employes et des congés

Employee Holiday

Nom de l'employe: 1 - SAISSI Zahra

Date de debut (YYYY-MM-DD):

Date de fin (YYYY-MM-DD):

Type: CONGE_PAYE

ID	nom_employe	date_debut	date_fin	type
1	1 - ZAHNI jhad	2025-01-05	2025-01-15	CONGE_PAYE
2	4 - KOUIS khaoula	2025-01-12	2025-01-15	CONGE_NON_PAYE
3	5 - ROUKI Rita	2025-03-19	2025-03-20	CONGE_NON_PAYE

Ajouter Modifier Supprimer Afficher Importer Exporter

Conclusion

L'ajout des fonctionnalités d'E/S à l'application de gestion des congés représente une étape cruciale pour améliorer son efficacité et sa convivialité. Grâce à l'intégration des mécanismes d'import/export, l'application devient plus flexible et adaptée aux besoins des utilisateurs. Cette extension, conforme au modèle MVC et à l'approche DAO, garantit une architecture bien structurée et maintenable, tout en offrant une gestion optimisée des données liées aux employés et à leurs congés.