App Desktop pour la Gestion des Ressources Humaines

Réalisé par :

ELAMRANI ABOUELASSAD Dauha

EL AICH Housni

EL AICH Houssam

EL AZM Ayman

Responsable :

M. NEJEOUI

SOMMAIRE

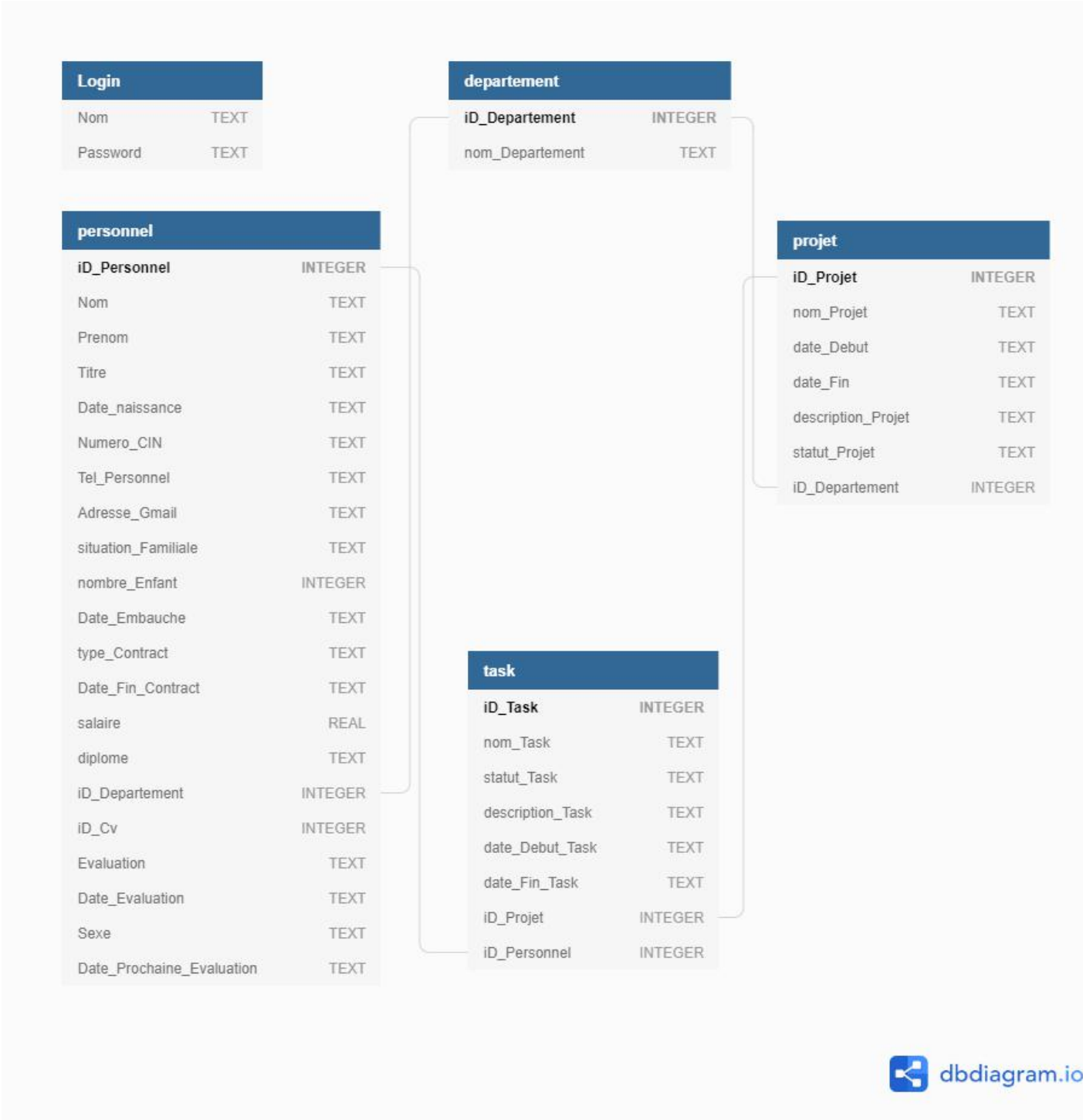
REMERCIEMENT

Avant tout propos, nous rendons grâce à Dieu pour notre réussite à ce projet. Il est souvent difficile de remercier les gens qui vous aident à accomplir les tâches qui vous sont données, et pourtant nous nous devons d'exprimer l'entière gratitude que nous ressentons envers eux. Nous tenons donc à présenter un remerciement bien distingué à notre encadrant Mr abderrazzak NEJEOUI pour son soutien, son aide, et ses conseils qui nous ont guidés durant l'élaboration de ce projet. C`est ici l`occasion pour nous de remercier toutes les personnes physiques et morales qui ont contribué à sa réalisation. Nous voudrons aussi remercier la direction de l'ENSA-M ainsi que tout le corps professoral et administratif de l'école pour les efforts qu'ils fournissent afin de nous garantir une bonne formation. Egalement, Nous tenons à signaler que l'étude approfondie de ce travail a fait naître en nous un véritable intérêt pour la gestion des ressources humaines et les stratégies de collaboration.

RÉSUMÉ

Le but de ce projet est de développer une application Desktop permettant de faciliter les procédures de 3 fonctions essentielles des ressources humaines et à la mise en place de la stratégie et au pilotage de la performance ; gestion de recrutement, gestion de planning et gestion de compétences (évaluation des compétences) . Au fil de ce rapport nous essaierons d'indiquer les différentes technologies utilisées et d'expliquer la fonctionnalité de chaque package intégré dans notre projet .

Base de Donnés Utilisée



PARTIE I

TECHNOLOGIES ADOPTÉES

A- BIBLIOTHÉQUES

1- Lombok :

Pour faciliter l'écriture de nos classes , par une génération automatique des méthodes les plus communes , par exemple : "setter" , "getter" et "equals" . La dépendance associée à Lombok ne marche pas , alors pour l'inclure il faut ajouter dans le "Java Build Path" de projet un fichier jar : lombok.jar (Emplacement : Bibliothèques) .

2- JFoenix :

Permet de fournir une interface dans le style Material design , ce dernier est un ensemble de règles de design proposées par Google et qui s'appliquent à l'interface graphique des logiciels et applications . Pour l'utiliser, il faut ajouter le fichier "jfoenix.jar" au "Java Build Path" de projet . (Emplacement : Bibliothèques)

3- SQLite JDBC :

Permet d'accéder et créer des fichiers de bases de données SQLite ; bases de données crées avec le système de gestion de base de données **SQLite** . Pour ajouter cette bibliothèque, il faut ajouter le fichier "sqlite-jdbc-3.20.1.jar" au "Java Build Path" de projet . (Emplacement : Bibliothèques)

B- FRAMEWORKS

1- Bean validation 2.0 (JSR 380) :

Pour la validation des données d'un bean, ce dernier est une classe Java qui respecte diverses règles de syntaxe et qui possède des accesseurs pour tous ses attributs, en utilisant la dépendance "Javax Validation 2.0.1.Final" .

2- JavaFX :

JavaFX est un framework Java permettant de construire des RIA (Rich Internet Application) ou des interfaces graphiques, créé par Sun Microsystems et désormais publié par Oracle. Ce framework est destiné à remplacer AWT/Swing en tant que bibliothèque graphique de Java SE (Standard Edition - la version client de la JVM pour ordinateurs de bureau) . Pour pouvoir l'utiliser, il faut télécharger JavaFX SDK et ajouter au "Java Build Path" de projet les fichiers «jar» existant dans le dossier "lib" .

C- AUTRES

1- Les spécifications de la JSR 380 définissent un petit ensemble de contraintes que chaque implémentation doit fournir , mais pour pouvoir utilisées ces spécifications il faut qu'elles soit implémentées par un fournisseur . Ce dernier sera "Hibernate Validator" , pour l'utiliser nous devons ajouter la dépendance associer avec un fichier jar : hibernate-validator-6.1.7.Final.jar (Emplacement : Bibliothèques) .

2- De plus, JSR 380 autorise les expressions à l'intérieur des messages de violation, et pour les analyser nous allons ajouter la dépendance "javax.el" de GlassFish, qui contient une implémentation de la spécification Expression Language .

PARTIE II

PACKAGES

A- PACKAGE “BaseDonnees”

1-Classe “Connect ” : Générer la connexion avec la base de données (SGBD : sqlite).

2-Classe “Inserer ” : Gérer l’insertion des infos du personnel dans la base de données .Methode principale :

public void insert(String champs,String valeur) : Permet d’insérer un nouvel enregistrement au table “Personnel” en spécifiant les colonnes (champs) et leurs valeurs (valeur).

3-Classe “ Select_Supprimer_Modifier ” : Méthodes principaux :

a)public ArrayList<Personnel> selectAll() :Retourne une liste des objets de type “Personnel” remplie avec les informations de la table “personnel” de la base de données .

b)public int selectIdFromDepartement(String departement) : Retourne, à partir du base de données, l’identificateur du nom de département donné en paramètre .

c)public ArrayList<Departement> selectAlllDepartement() :Retourne, à partir du base de données,une liste des objets de type “Departement” remplie avec les informations de la table “departement” de la base de données .

d)public String selectDepartementFromId(int id) :Retourne, à partir du base de données,le nom de département à partir l’identificateur donné en paramètre .

e)public Personnel_Evaluation selectEvaluation(String CIN) : Retourne, à partir du base de données, un objet de type “Personnel_Evaluation” selon le numéro de CIN entré en paramètre en sélectionner ,depuis la base de données, tous les informations sur l'évaluation du personnel .

f)public void update(String numeroCIN,String champs,String valeur) : Mettre à jour les données du personnel associé au numéro de CIN passé en paramètre .

g)public void delete_prs(String numeroCIN) : Supprime tous les données du personnel associé au numéro de CIN passé en paramètre .

h)public int selectIdDepartementFromidProjet(int id) : Retourne, à partir du base de données, l’identificateur de département qui gère le projet d’identificateur entré en paramètre .

i)public ArrayList<String> selectAllPersonnel(int id) :Retourne, à partir de la base de données, le nom et le prénom des personnels appartenant au département qui gère le projet d’identificateur entré en paramètre

j)public ArrayList<Projet> selectAllPrj() :Retourne une liste des objets de type “Projet” remplie avec les informations de la table “projet” de la base de données .

k)public ArrayList<Task> selectAllProjet_Tasks(int id_Projet) :Retourne une liste des objets de type “Task” ,associé au projet d’identificateur entré en paramètre ,remplie avec les informations de la table “task” de la base de données .

l)public int selectIdFromPersonnel(String nom, String prenom):Retourne, à partir du base de données, l’identificateur de personnel ayant le nom et le prénom entrant en paramètre .

m)public void updateProjet(int id, String champs, String valeur):Mettre à jour les données du projet associé au identificateur passé en paramètre .

n)public void updateTask(int id, String champs, String valeur) :Mettre à jour les données du personnel associé au identificateur passé en paramètre .

o)public void delete_Task(int id_Task) :Supprimer toutes les données du tâche associé au identificateur passé en paramètre .

4-Classe " Login " : Le mode de chiffrement utilisé pour le mot de passe est basé sur le mode CBC; Chiffrement par blocs de 4 octets à clef secrète "soad" .

->Principe :

Le mode CBC "Cipher Block Chaining" a été introduit pour qu'un bloc ne soit pas codé de la même manière s'il apparaît dans deux messages différents ou s'il apparaît deux fois dans un message.

Le message, M, est découpé en blocs , (mi)i≥1, et chaque bloc est crypté de la manière suivante. On commence par choisir un bloc initial c0. Chaque bloc clair mi est d'abord modifié en faisant un XOR de ce bloc avec le bloc crypté précédent, ci−1 puis on crypte le résultat obtenu par une la fonction de codage Ek :

$c1 = Ek(m1 \oplus c0)$

$c2 = Ek(m2 \oplus c1)$

...

...

$ci = Ek(mi \oplus ci-1)$

Note : Ek, dans ce cas est une addition de bloc entrée en paramètre et de la clef secrète .On obtient le message crypté c1|| . . . ||cn .

Méthodes principaux :

a)public static String XOR(String var,String var1) :Applique l'opération XOR sur les deux mots entrer en paramètre et retourner le résultat de l'opération .

b)public static String Addition(String var,String var1) :Applique l'opération d'addition sur les deux mots entrer en paramètre et retourner le résultat de l'opération .

c)public String[] Crypter(String var) : Chiffrer le mot entrer en paramètre et retourner une séquence de message crypté en mode CBC.

d)public static String decimal_positive(byte[] tab_oct) :Lorsque l'addition est appliquée ,des octets prennent des valeurs négatives ,puisque les valeurs négatives ne peuvent pas être représentés par des caractères ,la méthode les rend positives et retourne le résultat d'affectation sous forme d'un mot .

e)public static String supprime_espaces(byte[] tab_oct) :Supprime les 26 premiers octets (code ASCII), qui représentent par exemple l'espace,tabulation . Afin d'obtenir des caractères alphanumériques .

B- PACKAGE “fx_Recrutement”

Ce package va se charger de la gestion du recrutement d’un personnel , il contient l’interface du recrutement “Recrutement.fxml” et des classes qui s’occupent de vérifier la validité des données saisies et insérer le personnel dans la table “Personnel” .

1- Recrutement.fxml : C’est notre scène de recrutement .

2- Classe “Departement” : Cette classe était principalement ajouter pour obtenir un objet contenant id et nom du département d’un personnel . (Il est utilisé dans l’affichage du tableau des personnels et leurs départements)

3- Classe “Personnel” : Ses attributs représentent les colonnes du tableau Personnel, sauf celles liées à l’évaluation (ils seront nulles après l’insertion) . Cette classe était créée pour gérer les données saisies à l’interface , de telle façons que ces derniers doivent vérifier certaines contraintes pour qu'elles soient validés .(Avant d’insérer dans la table “Personnel”, on instancie un objet de cette classe)

4- Classe “RecrutementController” : C’est la classe qui contrôle la scène du recrutement .

-> Attributs (autres que celles du FXML):

a) HashMap<String, Label> map_Labels : Pour associer les attributs et quelques méthodes de classe "Personnel" (exemple getCheckDate_naissance) avec leurs labels d'erreurs, qui affiche les messages d'erreurs définies par les annotations de lombok. Exemple : map_Labels("Adresse_Gmail",Label : emailErreur) . (Remplie dans la fonction remplir())

b) HashMap<String, JFXCheckBox> map_Checkbox : Les clés sont les choix possible du sexe (homme et femme) et les élément sont les deux checkbox définis. Exemple : map_Checkbox("homme",JFXCheckBox : homme) . (Remplie dans la fonction remplir())

c) HashMap<String,String> champetval : Il sera remplie dans la fonction ChargerElement(), telle que les clés seront les attributs du classe "Personnel" et leurs valeurs sont celles retenues par les getters de chaque attribut du classe "Personnel".

d) ArrayList<String> champs : Tableau des champs (attributs du "Personnel"). Remplie à partir du HashMap champetval et prend que les champs avec valeur définis .

e) ArrayList<String> valeurs : Tableau des valeurs (valeur du chaque champ). Remplie à partir du HashMap champetval .

Remarque : "champs" et "valeurs" seront remplies dans la fonction ChargerElement() et à partir de ces tableaux on crée deux chaînes de caractères que l'on utilisent dans la fonction d'insertion (du classe "Inserer" du package "baseDonnes") dans la base de données.

f) static HashMap<ArrayList<String>, ArrayList<String>> champsetvaleurs_Evaluation : Cette HashMap est static pour qu'on puisse passer les données saisies à la classe "EvaluationController" du package "fx_Evaluation". (clé : "champs" | valeur : "valeurs")

g) String Sexe : Dans la classe "Personnel" , il existe une attribut "Sexe" de type string, pour la donner une valeur on est besoin d'un string qui prend comme valeur le label du case coché, et cela le rôle du "Sexe".

-> Méthodes :

a)public void initialize(URL local,ResourceBundle resources) : Initialise les éléments de l'interface ; remplir les "JFXComboBox" et map_Labels, map_Checkbox en appelant la méthode remplir() , définir des valeurs initiales aux quelques éléments de l'interface .

b) public void remplir() : Remplir les “HashMap” : "map_Labels" et "map_Checkbox" .

c) public void chargerDonnées(HashMap<String, String> valeur) : Cette fonction est utile lorsqu'on sélectionne à modifier les données d'un employé sélectionné à partir du tableau, et on veut charger ses données à partir du base de données et les mettre dans leurs champs. On le fait appeler dans la classe "Affichage_Personnels" du package “fx_Affichage” lorsqu'on clique sur le bouton "modifier" (fonction Modification). (valeur{clés : champs , éléments : valeurs})

d) public void messageSucces() : Affiche un pop-up qui indique que l’opération du recrutement était bien effectuée .

e) public void reset() : Rendre les champs de l’interface à leur état initial .

f) public int ChargerElement() : Elle instancie un objet de type "Personnel" et définit des valeurs aux attributs associés à partir des champs . Après, elle vérifie la validité des données , s’ils sont validés, elle remplit “champetval”, “champs”, “valeurs”, “champsetvaleurs_Evaluation” et retourne 1, sinon elle affiche des messages d’erreurs sur les labels d’erreurs et retourne 0 . Cette fonction s'exécute une fois qu'on clique sur le bouton "submit" ou bien sur "suivant" .

g) public void ActionReset(ActionEvent e) : Fait appeler au fonction reset() . S'exécute lorsque le bouton de reset est cliqué.

h) public void ActionSubmit(ActionEvent e) : Fait appeler au fonction ChargerElement() , si cette dernière retourne 1, alors elle insère le personnel dans la table “Personnel” en gérant l’exception du sql , par exemple si l’un des données ne vérifie pas la contrainte d’unicité , elle informe l’utilisateur en affichant un message dans label correspondant . Elle s'exécute lorsque le bouton submit est cliqué.

i) public void ActionContrat(ActionEvent e) : Cette fonction est exécutée lors de la sélection d'une option du “JFXComboBox” "type_Contrat". Si l'option choisie est "CDD" donc le champ date fin contrat sera activé , sinon il sera désactivé.

j) public void ActionCheck(ActionEvent e) : Cette fonction est exécutée si l'un des cases est coché. Elle permet de sélectionner qu'une seule valeur.

k) public void ActionSuivant(ActionEvent e) : Si le bouton suivant est appuyé , la scène change et on passe au scène d'évaluation.

Remarque : Le bouton “suivant” n’est pas visible lorsqu’on est en train de recruter un personnel . Il sera visible et les deux autres boutons ne seront pas visibles dans le cas de modification des données d’un personnel existant .

C- PACKAGE “fx_Evaluation”

Gérer l’évaluation d’un employé. Elle suit le même principe que celui du package “fx_Recrutement” .

1- Evaluation.fxml : c’est l’interface de l’évaluation .

2- Classe “Personnel_Evaluation” : Ses attributs sont les colonnes du tableau Personnel lié à l’évaluation (Evaluation, Date_Evaluation, Date_Prochaine_Evaluation). Un objet de cette classe est instancié avant l’insertion de l’évaluation dans la table Personnel .

3- Classe “EvaluationController” : C’est la classe qui contrôle la scène du l’évaluation .

> Attributs (autres que celles du FXML):

a) HashMap<String,Label> map_Labels : Associer les attributs et quelques méthodes de classe "Personnel_Evaluation" avec leurs labels d'erreurs, qui affiche les messages d'erreurs définies par les annotations de lombok. Exemple : map_Labels("Date_Evaluation", DateEvErreur) .

b) HashMap<String,String> champetval : Sera remplie dans la fonction "MiseAJour” , telle que les clés seront les attributs du classe "Personnel_Evaluation” et leurs valeurs sont celles retenues par les getters de chaque attribut de classe "Personnel_Evaluation" .

c) ArrayList<String> champsEvaluation : Tableau des champs, remplie par les clés du “champetval”.

d) ArrayList<String> valeursEvaluation : Tableau des valeurs, remplie par les valeurs du “champetval”.

e) ArrayList<String> champPersonnel : Tableau des champs (attributs de classe "Personnel").

f) ArrayList<String> valeurPersonnel : Tableau des valeurs (valeur du chaque champ de personnel).

Remarque : “champPersonnel” et “valeurPersonnel” remplie dans la fonction "initialize" d'après la variable static de classe "RecrutementController" : "champsetvaleurs_Evaluation" .

-> Méthodes :

a) public void initialize(URL local,ResourceBundle resources) : Initialise les éléments de l’interface ; remplir les “JFXComboBox” et map_Labels . Remplie en avance les champs par les données d'évaluation extrait du base de données , du personnel sélectionné , par la méthode “chargerEvaluation”.

b) public void chargerEvaluation(String CIN) : Charger les données à partir du base de données et les mettre dans leurs champs . CIN : le numéro de CIN obtenu lorsqu'on clique sur un élément du liste des employés afficher .

c) public void messageSucces() : Fonction pour indiquer que les modifications étaient bien effectuées en affichant un pop-up qui se ferme après 3 secondes et ferme la fenêtre d'évaluation .

d) public void ActionSubmit(ActionEvent e) : Elle instancie un objet de type "Personnel_Evaluation" et définit des valeurs aux attributs associés à partir des champs . Après, elle vérifie la validité des données , s'ils sont validés ou les champs sont tous vides, elle fait appelle au fonction "MiseAJour", sinon elle affiche des messages d'erreurs sur les labels d'erreurs. Elle s'exécute lorsque le bouton submit est cliqué.

e) public void MiseAJour(ArrayList<String> champPersonnel, ArrayList<String> valeurPersonnel, Personnel_Evaluation PE) : Cette fonction s'occupe de remplir "champsEvaluation" et "valeursEvaluation" , après ajouter "champsEvaluation" au "champPersonnel" et "valeursEvaluation" au "valeurPersonnel", et finalement mettre à jour la base de données par la fonction update(String numeroCIN, String champs, String valeur) du classe "Select_Supprimer_Modifier" de package "baseDonnees".

D- PACKAGE “fx_Affichage”

Gérer l’affichage des listes de personnels , projets et tasks sous forme des tableaux (objets “TableView”) dans l’interface graphique ,afin de créer un système de recherche avec un moteur de recherche .

Le contrôle TableView est conçu pour visualiser un nombre illimité de lignes de données, réparties en colonnes.Prend en paramètre le type d’objet voulons nous le visualiser ,telle que les colonnes sont représentées par les attributs de cet objet .

1-Classe “ Affichage_Personnels_Class ” :

Fournit l’objet traité par “TableView” manipulé dans la classe «Affichage_Personnels» .Contient deux attributs de type “Personnel” et “Departement” ,afin d’associer à chaque personnel le nom de son département .

2-Classe “ Affichage_Personnels ” :

La classe qui contrôle la scène d’affichage et dans laquelle les évènements sont traités (Action de clicker sur les buttons , selectionner un ligne de TableView) .

a)public void initialize(URL location,ResourceBundle resources) :Initialisation de la scène ; remplissage du tableau par les données des personnels .

b)public void ActionActualiser(ActionEvent e) : Action associé au bouton “Actualiser” ;actualiser la scène en remplissant le tableau par les données des personnels .

c)public void Search(ActionEvent e) : Action de cliquer sur le bouton de recherche. Parcours la liste de tous les objets “Personnel” (si le text sélectionner dans “JFXComboBox” est personnel) et sélection ceux que l’un de leurs attributs contient le text saisie (dans la zone de recherche) .Parcours la liste de tous les objets “Departement” (si le text sélectionner dans “JFXComboBox” est departement) et sélection ceux que l’un de leurs attributs contient le text saisie (dans la zone de recherche) .

d)public void affiche_pop_up(MouseEvent e) : Action de cliquer sur un ligne de tableau,affiche un “AnchorPane” autant une fenêtre pop-up ;contenant les buttons de modification et de suppression .

e)public void fermer_pop_up(ActionEvent e) : Action de fermer pop-up fenêtre.

f)public void Modification(ActionEvent e) : Action de cliquer sur le bouton “Modifier” ,recharger les données du personnel sélectionnées dans le tableau en ouvrant la même scène utilisée pour le recrutement et on remplissant tous les champs avec ces données .

g)public void Suppression(ActionEvent e) : Action de cliquer sur le bouton “Supprimer” ,supprime tous les données du personnel sélectionnées dans le tableau .

3-Classe “ Affichage_Projets_Class ” :

Fournit l’objet traité par “TableView” manipulé dans la classe «Controller_Projet_scene».Contient deux attributs de type “Projet” et “Departement” ,afin d’associer à chaque projet son département .

4-Classe “ Controller_Projet_scene” :

La classe qui contrôle la scène d’affichage «Affichage_Projets.fxml» et dans laquelle les évènements sont traités (Action de clicker sur les buttons , selectionner un ligne de TableView) .

A)public void initialize(URL location,ResourceBundle resources):Initialisation de la scène ; remplissage du tableau par les données des personnels .

B)public void Choix_Type_chercher(ActionEvent e) : Action de sélectionner une valeur dans le «JFXComboBox» .Activer le champ de text ,où on saisie le mot à chercher, lorsqu’une valeur est sélectionner .

C)public void ActionActualiser(ActionEvent e) : Action de cliquer sur le bouton actualiser;Actualiser le Tableau d'affichage .

D)public void Action_chercher(ActionEvent e) : Action de cliquer sur le bouton de recherche. Parcours la liste de tous les objets “Projet” (si le text sélectionner dans “JFXComboBox” est "Nom Projet" ou "Statut Projet") et sélection ceux que l’un de leurs attributs contient le text saisie (dans la zone de recherche) .Parcours la liste de tous les objets “Departement” (si le text sélectionner dans “JFXComboBox” est "Departement") et sélection ceux que l’un de leurs attributs contient le text

saisie (dans la zone de recherche) .

E)public void affiche_Tasks(MouseEvent e) : Action de cliquer sur un ligne de tableau;remplacer la scène actuel par «Affichage_Tasks.fxml» en affichant liste des tasks associé au projet sélectionner .

5-Classe “ Affichage_Tasks_Class ” :

Fournit l’objet traité par “TableView” manipulé dans la classe «Controller_Tasks_scene» .Contient deux attributs de type “Personnel” et “Task” ,afin d’associer à chaque task son responsable .

6-Classe “ Controller_Tasks_scene ” :

La classe qui contrôle la scène d’affichage «Affichage_Tasks.fxml» et dans laquelle les évènements sont traités (Action de clicker sur les buttons , selectionner un ligne de TableView) .

A)public void initialize(URL location,ResourceBundle resources):Initialisation de la scène ; remplissage du tableau par les données des personnels .

B)public void Choix_Type_chercher(ActionEvent e) : Action de sélectionner une valeur dans le «JFXComboBox» .Activer le champ de text ,où on saisie le mot à chercher, lorsqu’une valeur est sélectionner .

C)public void Action_chercher(ActionEvent e) : Action de cliquer sur le bouton de recherche. Parcours la liste de tous les objets “Task” (si le text sélectionner dans “JFXComboBox” est "Nom Task" ou "Statut Task") et sélection ceux que l’un de leurs

attributs contient le text saisie (dans la zone de recherche) .Parcours la liste de tous les objets “Personnel” (si le text sélectionner dans “JFXComboBox” est "Personnel") et sélection ceux que l’un de leurs attributs contient le text saisie (dans la zone de recherche) .

D)public void affiche_pop_up(MouseEvent e) : Action de cliquer sur un ligne de tableau,affiche un “AnchorPane” autant une fenêtre pop-up ;contenant les buttons de modification et de suppression .

E)public void fermer_pop_up(ActionEvent e) : Action de fermer pop-up fenêtre.

F)public void Modification(ActionEvent e) : Action de cliquer sur le bouton “Modifier” ,recharger les données du Task sélectionnées dans le tableau en ouvrant la même scène utilisée pour l’ajout des Tasks et on remplissant tous les champs avec ces données .

G)public void Suppression(ActionEvent e) : Action de cliquer sur le bouton “Supprimer” ,supprime tous les données du Task sélectionnées dans le tableau .

H)public void Action_Ajouter_Task(ActionEvent e) : Action de cliquer sur le bouton “+” .Ajouter un nouveau task au projet actuel en faisant appelé la scène qui contrôle l’ajout des tasks «Tasks_inputs.fxml»

I)public void Action_Modifier_Projet (ActionEvent e) : Action de cliquer sur le bouton “Modifier” ,recharger les données du Projet sélectionnées dans le tableau des projets précédent en ouvrant la même scène utilisée pour l’ajout des Projets et on remplissant tous les champs avec ces données .

J)public void retourner_scene_projets(ActionEvent e) : Action de cliquer sur le bouton “<” ;retourner à la scène précédent «Affichage_Projets.fxml» en affichant liste des projets .

E- PACKAGE “fx_Login”

Gérer la scène d’identification .

1-Classe “ LoginController ” : La classe qui contrôle la scène d’identification et dans laquelle les évènements sont traités (Action de clicker sur les buttons) .

a)public void initialize(URL location,ResourceBundle resources) : Initialiser la scène en associant elle à une autre “window.fxml” qui represent la barre de la fenêtre ,après que le style de scène ait été défini non décoré .

b)public void Login(ActionEvent event) : Action de cliquer sur le bouton de “login” .Tester si le nom d’utilisateur saisie existe et que le mot de passe saisie est celui associé aux utilisateur ,s’ils valident ,on passe à la page d'accueil “Accueil.fxml”.

F- PACKAGE “fx_Projet”

Ce package va se charger d’ajouter un projet , il contient l’interface d’addition “Projet_inputs.fxml” et des classes qui s’occupent de vérifier la validité des données saisies et insérer le projet dans la table “projet” .

1- Projet_inputs.fxml : C’est notre scène .

3- Classe “Projet” : Ses attributs représentent les colonnes du tableau Projet . Cette classe était créée pour gérer les données saisies à l’interface , de telle façons que ces derniers doivent vérifier certaines contraintes pour qu'elles soient validés .(Avant d’insérer dans la table “projet”, on instancie un objet de cette classe)

4- Classe “Projet_inputs” : C’est la classe qui contrôle la scène .

-> Attributs (autres que celles du FXML):

a) HashMap<String, Label> map_Labels : Pour associer les attributs et quelques méthodes de classe "Projet" avec leurs labels d'erreurs, qui affiche les messages d'erreurs définies par les annotations de lombok. (Remplie dans la fonction initialize)

c) HashMap<String,String> champetval : Il sera remplie dans la fonction ChargerElement(), telle que les clés seront les attributs du classe "Projet” et leurs valeurs sont celles retenues par les getters de chaque attribut du classe "Projet".

d) ArrayList<String> champs : Tableau des champs (attributs du "Projet"). Remplie à partir du HashMap champetval et prend que les champs avec valeur définis .

e) ArrayList<String> valeurs : Tableau des valeurs (valeur du chaque champ). Remplie à partir du HashMap champetval .

Remarque : "champs" et "valeurs" seront remplies dans la fonction ChargerElement() et à partir de ces tableaux on crée deux chaînes de caractères que l'on utilisent dans la fonction d'insertion ("insertProjet" du classe "Inserer" de package "baseDonnes") ou fonction de mise à jour ("updateProjet" du classe "Select_Supprimer_Modifier" de package "baseDonnes") dans la base de données.

f)int modifier : Par cette valeur on peut contrôler si submit va effectuer une insertion ou bien une mise à jour .

-> Méthodes :

a)public void initialize(URL local,ResourceBundle resources) : Initialise les éléments de l'interface ; remplir les "JFXComboBox" et map_Labels, définir des valeurs initiales aux quelques éléments de l'interface .

b) public void chargerDonnées(HashMap<String, String> valeur) : Cette fonction est utile lorsqu'on sélectionne à modifier les données d'un projet, et on veut charger ses données à partir du base de données et les mettre dans leurs champs. On le fait appeler dans la classe "Controller_Tasks_scene" du package “fx_Affichage” lorsqu'on clique sur le bouton "modifier" (fonction Action_Modifier_Projet). Une fois cette fonction est exécuté, l’entier "modification" égale à 1. (valeur{clés : champs , éléments : valeurs})

c) public void messageSucces() : Affiche un pop-up qui indique que l’opération d’ajout ou de modification était bien effectuée .

d) public void reset() : Rendre les champs de l’interface à leur état initial .

e) public int ChargerElement() : Elle instancie un objet de type "Projet" et définit des valeurs aux attributs associés à partir des champs . Après, elle vérifie la validité des données , s’ils sont validés, elle remplit “champetval”, “champs”, “valeurs” et retourne 1, sinon elle affiche des messages d’erreurs sur les labels d’erreurs et retourne 0 . Cette fonction s'exécute lorsqu'on clique sur le bouton "submit".

f) public void ActionReset(ActionEvent e) : Fait appeler au fonction reset() . S'exécute lorsque le bouton de reset est cliqué.

g) public void ActionSubmit(ActionEvent e) : Elle voir tout d’abord la valeur du variable “modification“,si elle égale à 0 alors c’est une insértion, sinon c’est une modification .Après, fait appeler au fonction ChargerElement() , si cette

dernière retourne 1, alors elle applique l'opération en gérant l'exception du sql . Elle s'exécute lorsque le bouton submit est cliqué.

G- PACKAGE "fx_Task"

Ce package va se charger d'ajouter une tâche , il contient l'interface d'addition "Tasks_inputs.fxml" et des classes qui s'occupent de vérifier la validité des données saisies et insérer la tâche dans la table "task" .

1- Tasks_inputs.fxml : C'est notre scène .

3- Classe "Task" : Ses attributs représentent les colonnes du tableau Task . Cette classe était créée pour gérer les données saisies à l'interface , de telle façons que ces derniers doivent vérifier certaines contraintes pour qu'elles soient validés .(Avant d'insérer dans la table "task", on instancie un objet de cette classe)

4- Classe "Tasks_inputs" : C'est la classe qui contrôle la scène .

-> Attributs (autres que celles du FXML):

a) HashMap<String, Label> map_Labels : Pour associer les attributs et quelques méthodes de classe "Task" avec leurs labels d'erreurs, qui affiche les messages d'erreurs définies par les annotations de lombok. (Remplie dans la fonction initialize)

c) HashMap<String,String> champetval : Il sera remplie dans la fonction ChargerElement(), telle que les clés seront les attributs du classe "Task" et leurs valeurs sont celles retenues par les getters de chaque attribut du classe "Task".

d) ArrayList<String> champs : Tableau des champs (attributs du "Task"). Remplie à partir du HashMap champetval et prend que les champs avec valeur définis .

e) ArrayList<String> valeurs : Tableau des valeurs (valeur du chaque champ). Remplie à partir du HashMap champetval .

Remarque : "champs" et "valeurs" seront remplies dans la fonction ChargerElement() et à partir de ces tableaux on crée deux chaînes de caractères que l'on utilisent dans la fonction d'insertion ("insertTask" du classe "Inserer" de package "baseDonnes") ou fonction de mise à jour ("updateTask" du classe "Select_Supprimer_Modifier" de package "baseDonnes") dans la base de données.

f)int modifier : Par cette valeur on peut contrôler si submit va effectuer une insertion ou bien une mise à jour .

-> Méthodes :

a)public void initialize(URL local,ResourceBundle resources) : Initialise les éléments de l'interface ; remplir les "JFXComboBox" et map_Labels, définir des valeurs initiales aux quelques éléments de l'interface .

b) public void chargerDonnées(HashMap<String, String> valeur) : Cette fonction est utile lorsqu'on sélectionne à modifier les données d'une tâche sélectionné à partir du tableau, et on veut charger ses données à partir du base de données et les mettre dans leurs champs. On le fait appeler dans la classe "Controller_Tasks_scene" du package "fx_Affichage" lorsqu'on clique sur le bouton "modifier" du pop-up (fonction Modification). Une fois cette fonction est exécuté, l'entier "modification" égale à 1. (valeur{clés : champs , éléments : valeurs})

c) public void messageSucces() : Affiche un pop-up qui indique que l'opération d'ajout ou de modification était bien effectuée .

d) public void reset() : Rendre les champs de l'interface à leur état initial .

e) public int ChargerElement() : Elle instancie un objet de type "Taskt" et définit des valeurs aux attributs associés à partir des champs . Après, elle vérifie la validité des données , s'ils sont validés, elle remplit "champetval", "champs", "valeurs" et retourne 1, sinon elle affiche des messages d'erreurs sur les labels d'erreurs et retourne 0 . Cette fonction s'exécute lorsqu'on clique sur le bouton "submit".

f) public void ActionReset(ActionEvent e) : Fait appeler au fonction reset() . S'exécute lorsque le bouton de reset est cliqué.

g) public void ActionSubmit(ActionEvent e) : Elle voir tout d'abord la valeur du variable "modification",si elle égale à 0 alors c'est une insértion, sinon c'est une modification .Après, fait appeler au fonction ChargerElement() , si cette dernière retourne 1, alors elle applique l'opération en gérant l'exception du sql . Elle s'exécute lorsque le bouton submit est cliqué.

H- PACKAGE "fx_Accueil"

1- Classe "Main" : C'est la classe principale de projet entier, charge la fenêtre d'authentification .

2- "Accueil.fxml" : scène d'accueil .

3- Classe "AccueilController" :

Contrôle les boutons définis dans la scène d'accueil :

- "Recruter" : Permet de charger la scène "Recrutement.fxml" de package "fx_Recrutement" .

- "Liste des employées" : Permet de charger la scène "Affichage_tabPersonnels.fxml" de package "fx_Affichage" .

- "Ajouter Projet" : Permet de charger la scène "Projet_inputs.fxml" de package "fx_Projet" .

- "Liste des projets" : Permet de charger la scène "Affichage_Projets.fxml" de package "fx_Affichage" .

4- "Window.fxml" : scène représentant la barre de titre avec le bouton fermer de fenêtre d'authentification .

5- Classe "WindowController" : Contrôle la scène "Window.fxml" .

Conclusion & Perspectives

Cette expérience était, sans aucun doute, une occasion non seulement pour découvrir le travail en groupe, mais aussi afin de développer une vision claire et nette sur les problèmes rencontrés lors du développement d'une application pour satisfaire des besoins réels et tangibles sous la surveillance des supérieurs dans l'hiérarchie des entreprises ou des administrations. Les aspects précités dans l'introduction ont tous vu la lumière, surtout sur les deux volets organisationnel et technique. Les obstacles que nous avons affrontés le long du projet nous ont empêchés de développer beaucoup plus notre application et de diversifier ses formes. Ainsi, seulement si nous pouvions ajouter une forme « statistiques » qui servira à faire une synthèse de toutes les opérations et livraison faites antérieurement, ce qui sera de grande importance pour le contrôle des demandes de document et de la surveillance des employés au cours de l'année, mais la contrainte du temps ne nous a pas permis.

Enfin, nous souhaitons que ce modeste travail soit à la hauteur des espérances de notre honorable encadrant à qui nous devons toute notre gratitude et notre respect.