

Rapport de Projet JAVA

Sujet choisi : Gestion d'Offre de Stage.



BLANCHIS Aurèle
JEANNE DIT LE PAGE Florian

Sommaire :

I. Présentation de notre Projet

- A. Le diagramme de Classe
- B. Le diagramme de Cas d'Utilisation
- C. Les diagrammes de séquences

II. Explications sur le code

- A. La Connexion
- B. Les Panels Principaux
 - 1. Administrateur
 - 2. Entreprise
 - 3. Étudiant
- C. Les Panels avec Listes
- D. Les JDialogs
 - 1. JDialogCréer
 - 2. JDialogModifier
 - 3. JDialogProfil
- E. Les JTables
- F. Les DAO
- G. Fonctionnement du code et liens entre classes

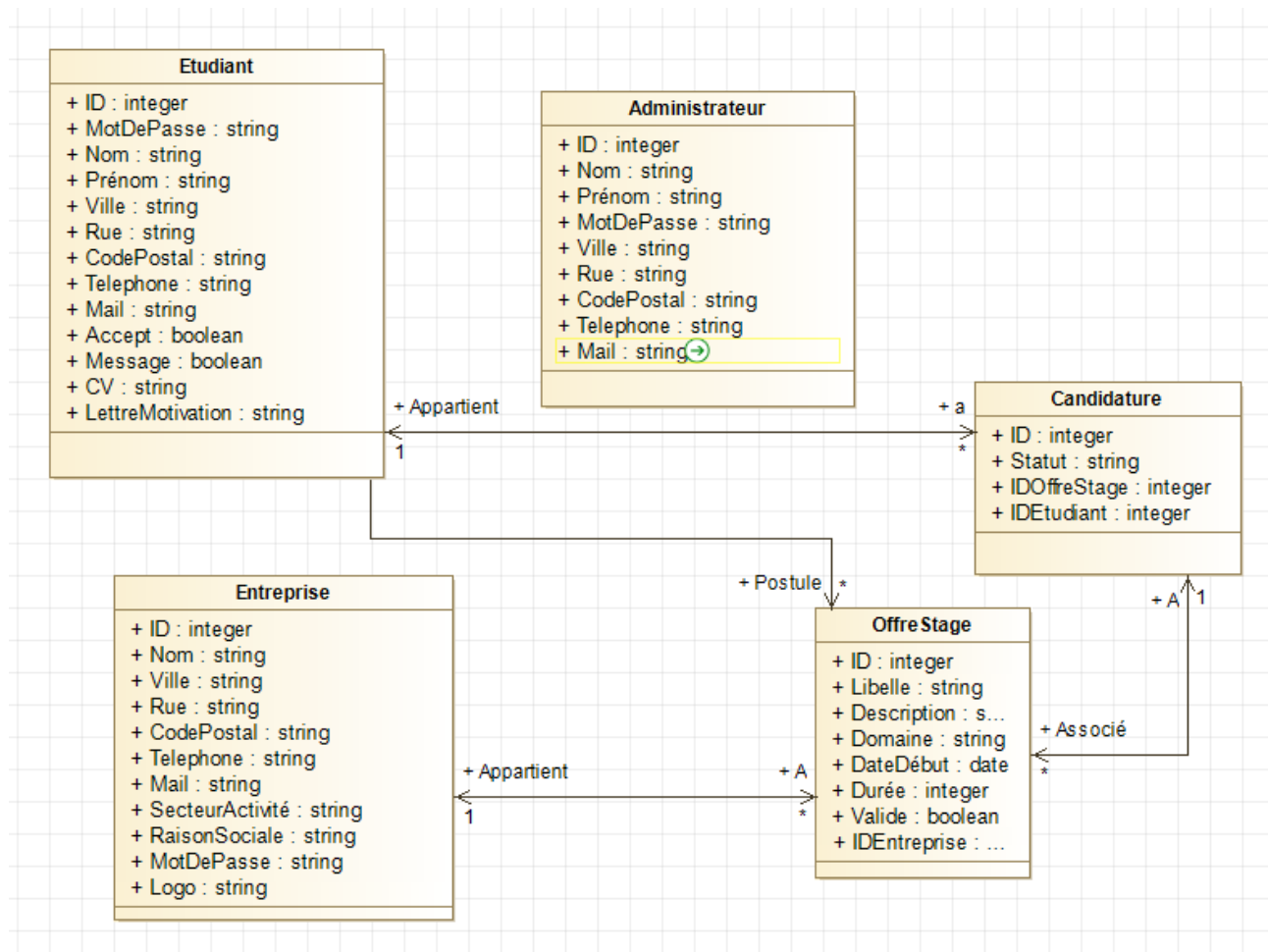
III. Le Journal de bord

- A. Déroulement du Projet
- B. Difficultés rencontrées

I. Présentation de notre Projet.

A. Le diagramme de Classe

Je vais commencer par vous expliquer comment fonctionne notre programme.
Pour illustrer mes propos dès le début, je vous joint notre diagramme de Classe :



Explications :

Un étudiant possède tout un tas d'attribut (nom, ID, mdp, etc).

(l'ID de l'étudiant est également son ID de connexion)

Une entreprise en possède également.

Une offre de stage en possède, mais elle en possède 1 important : IDEntreprise. En effet, cet attribut va nous permettre de relier les offres de stage aux entreprises auxquelles elles appartiennent. Ceci nous permet d'effectuer une liste d'offres par entreprise.

Nous avons ensuite les candidatures : elle possèdent un statut, pour savoir si un étudiant a été accepté par une entreprise, refusé, ou s'il est toujours en attente (de base). Elles possèdent surtout 2 attributs importants : IDOffreStage et IDEtudiant. Ces 2 attributs vont permettre le lien entre une offre de stage et un étudiant :

- 1 étudiant possède plusieurs candidatures (il postule où il veut)
- 1 offre de stage possède plusieurs candidatures (plusieurs étudiants)
- 1 candidature concerne 1 seul étudiant
- 1 candidature concerne 1 seule offre

Une offre de stage possède donc une liste de candidatures, qui elles-mêmes sont liées à 1 unique étudiant.

Ainsi, une entreprise pourra voir la liste d'étudiants qui ont postulé à une offre de stage (par le biais de la candidature).

Un étudiant ne peut pas postuler plusieurs fois pour la même offre de stage.

Nous pouvons de même postuler en tant qu'étudiant pour une offre de stage, ce qui va créer une candidature à cette offre. Depuis le compte de l'entreprise, elle verra dans sa liste d'offres de stage la liste des étudiants qui y ont postulé. Ceci sera expliqué par des images dans le II.

Tous les autres attributs sont plutôt clairs, sauf 2 : les boolean Accept et Message dans Etudiant.

A quoi servent-ils ?

C'est simple, quand un étudiant se connecte et qu'il y a été accepté ou refusé pour une offre, un « Pop-up » va apparaître lors de sa connexion.

- Quand une entreprise accepte un étudiant : Message = true ; Accept = true.
- Quand une entreprise refuse un étudiant : Message = true ; Accept = false.

Dans les 2 cas, Message passe à true, ce qui affiche donc le pop-up lors de la prochaine connexion de l'étudiant (avec une condition if (message = true)).

Le boolean Accept sert simplement à modifier le contenu du message.

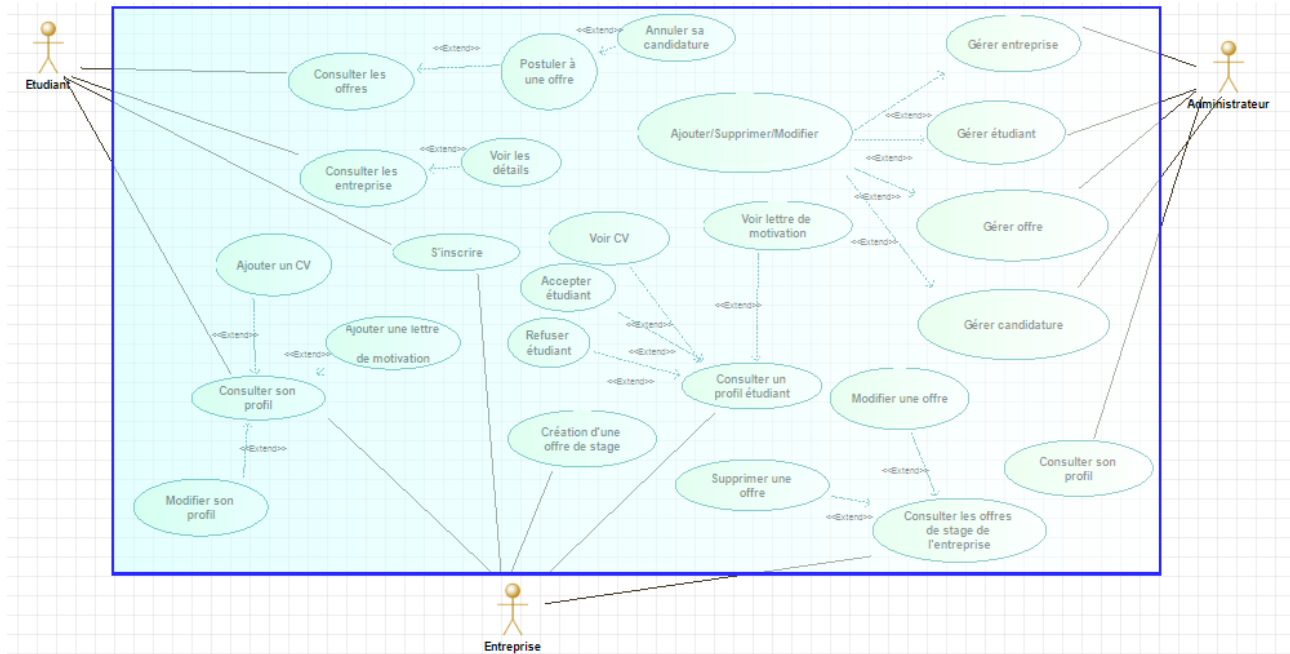
Pour que ce message ne s'affiche pas éternellement, on passe Message à false lors de l'apparition du pop-up, comme ça il ne réapparaîtra plus.

Le statut « En attente » de la candidature permet également d'afficher un pop-up « candidature en attente » lors de la connexion de l'entreprise.

L'administrateur possède des attributs (et tous les droits), mais il n'entre en lien avec aucune autre classe.

B. Le diagramme de Cas d'utilisation

Pour vous expliquer comment fonctionne notre programme grâce au diagramme de cas d'utilisation :



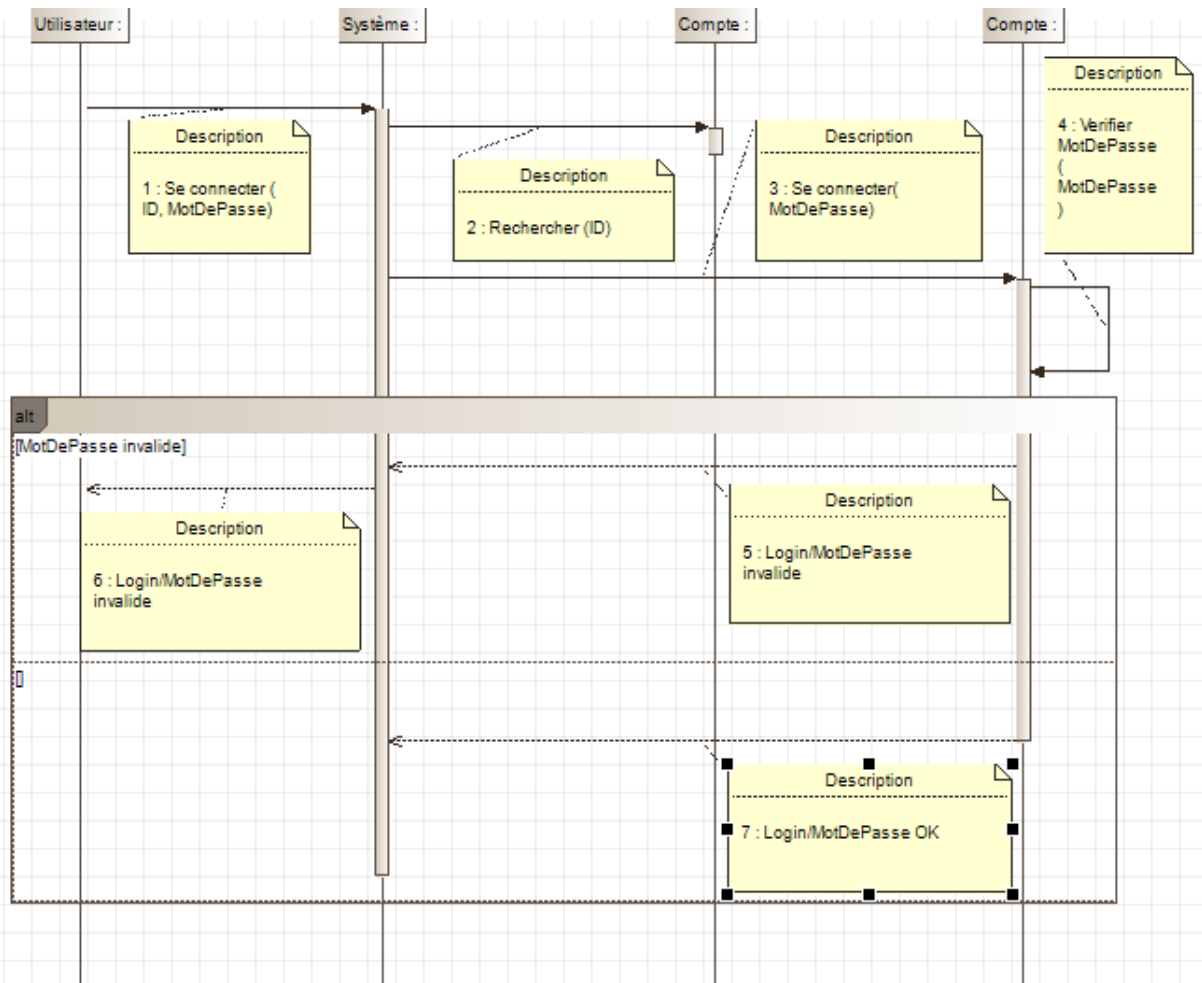
(Cette image sera donnée en pièce jointe, pour une meilleure visualisation)

- Un étudiant peut donc consulter les offres de stages par entreprise, y postuler et annuler sa candidature plus tard s'il le souhaite. Depuis son Profil il peut ajouter une CV ou une Lettre de Motivation (via le JFileChooser). Il peut également le modifier.
- Une entreprise peut créer une offre de stage (la modifier et la supprimer par la suite), et consulter ses candidatures à ses offres. Il peut ainsi Accepter/Resufer un étudiant, tout en consultant son profil auparavant.
- L'administrateur à accès à tout:il gère tout type de classe, avec la permission de suppression, création et modification.

C. Les diagrammes de séquences.

Les diagrammes de séquence sont la représentation graphique entre les utilisateurs (admin, entreprise, étudiant) et le système.

Sur ces diagrammes on représente chaque action possible dans notre programme.

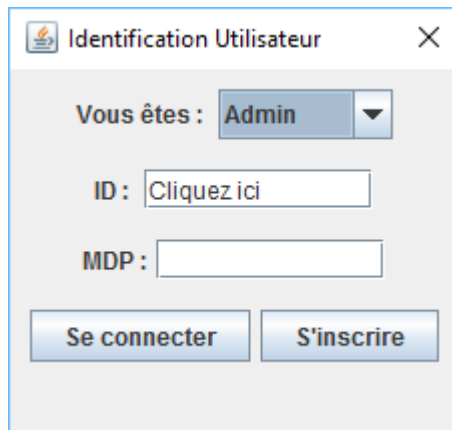


Ici le diagramme de cas d'utilisation de la connexion réalise en premier l'action de cliquer sur le bouton se connecter. Le système recherche ensuite l'ID et le compare avec le mot de passe. Si le mot de passe est invalide il envoie un message à l'utilisateur, sinon le système se connecte avec ce compte.

Chaque diagramme de cas d'utilisation est basé sur le même principe, je vous laisse les voir en pièce jointe du rapport.

II. Explications sur le Code

A. La Connection



Tout d'abord on sélectionne qui l'on est :

- L'**Administrateur**
- Une **Entreprise**
- Un **Etudiant**

En rentrant son ID (généré automatiquement par PhpMyAdmin) et le mot de passe correspondant tout en cliquant sur le JButton

« Se connecter », on va ouvrir une Fenetre avec comme Panel un Panel Principal (voir II. B) correspondant à qui l'on est.

Nous avons crypté le Mot de Passe avec la classe *Md5*. Quand on crée un compte, le mot de passe est crypté.

Quand on se connecte et qu'on rentre un mot de passe dans le JTextField, il va être crypté pour le comparer à celui de la base de données.

L'inscription se fera suivant qui l'on est, et ouvrira un formulaire d'inscription (voir II.D).

L'inscription en tant qu'Administrateur n'est pas possible (Unique).

Classe correspondante : *Log*, *Md5*.

B. Les Panels Principaux

Il en existe 3 : Pour l'Administrateur, l'Entreprise, l'Étudiant.

Pour ces classes, on a opté pour un placement des éléments par position (avec des SetBounds), pour une question d'apparence.

1. Administrateur



- La Version de l'application sous laquelle on est connecté est affichée en haut à gauche
- « Connecté en tant que ... » affiché en haut à droite
- Titre de l'Application au milieu
- Bouton du Profil en dessous : il affiche la page sur son profil (JDialog, voir II. D pour son contenu).
- Les 4 autres boutons servent à ouvrir des JTable pour tout gérer (création, modification, suppression, voir II.E).
- En bas à droite, bouton de déconnexion.
- Pour l'image de fond, on a utilisé une classe *ImageDeFond* qui extends de JPanel et qui prend en argument une image de notre ordinateur.

2. Entreprise



- Même chose pour la plupart des boutons et éléments
- Image de l'entreprise (modifiable sur « **Mon Profil** ») au milieu du Panel.
- « **Créer une Offre** » ouvre un formulaire pour créer une nouvelle offre pour cette entreprise.
- « **Mes Offres** » Permet de consulter ses offres, son nombre de candidats, le modifier, etc (voir II.C)

3. Étudiant



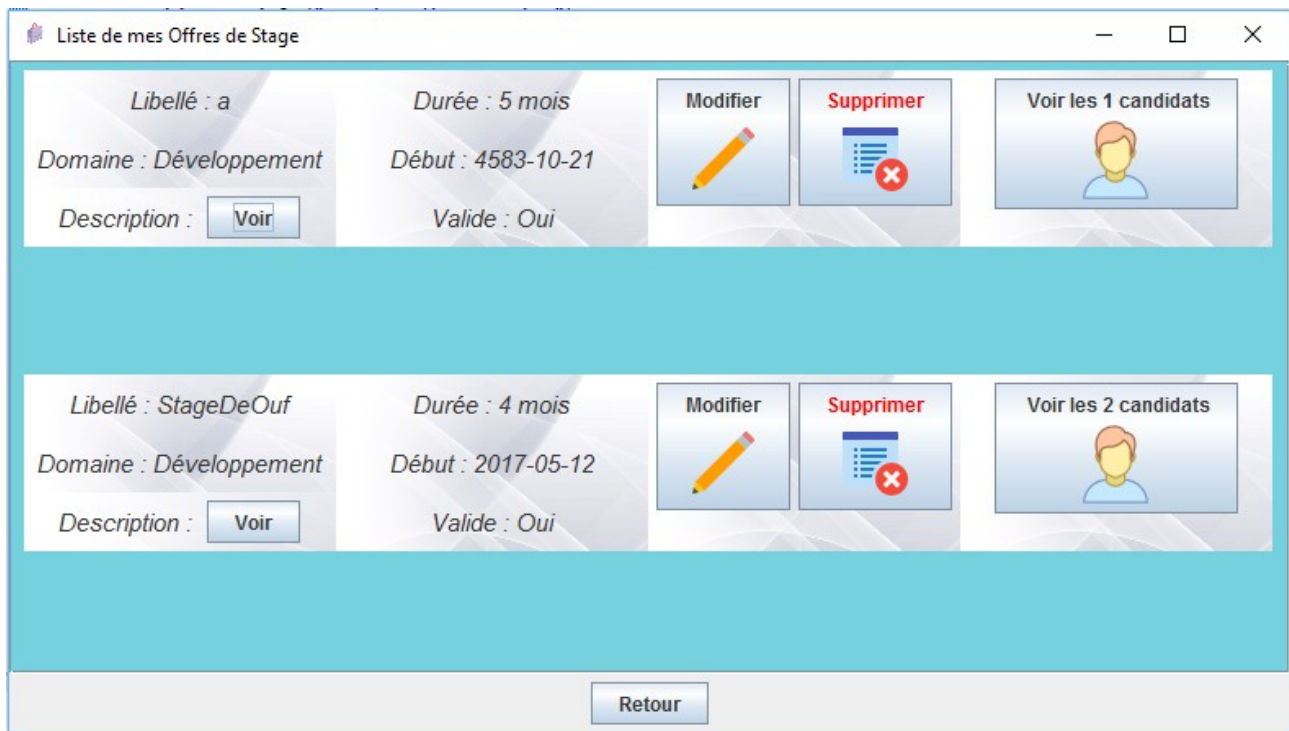
- Grosse ressemblance avec le Panel Entreprise
- « **Liste des Entreprises** » permet de consulter la liste de toutes les entreprises existantes, ainsi que leurs offres, et y postuler par la suite (Voir II.C).
- « **Mes Candidatures** » permet de consulter l'état de ses candidatures, ainsi que leurs informations.

Classes correspondantes : Voir le package *IHM.PanelPrincipaux* (3 classes).

C. Les Panels avec Liste

Ces panels s'ouvrent avec une nouvelle Fenêtre. Ils contiennent un Panel Général nommé par la classe par exemple « *JPanelListeCandidature* », qui lui-même contient plein de « petits » panels nommé par exemple « *PanelUniqueCandidature* ».

Il en contient autant qu'il en existe. Pour *JPanelListeCandidature* par exemple, on va construire un `GridLayout` de la taille du nombre de candidatures existantes pour une offre de stage, et dans chacun des panels du `GridLayout` on va ajouter un « *PanelUniqueCandidature* ».



Voici un exemple d'un Panel avec Liste. Ici on est connecté en tant qu'entreprise, on a choisi d'aller consulter ses Offres. On peut ainsi les modifier, les supprimer, voir les candidats. Ici l'entreprise n'en a que 2, le GridLayout est donc construit avec 2 panels. (un scroll apparaîtra si la taille dépasse la fenêtre).

Je ne vais pas tous les énumérer, c'est toujours le même principe pour les autres. J'en parlerai lors de l'explication du fonctionnement du programme (II.F).

Classes correspondantes : voir le package *IHM.PanelListe* (8 classes : 4 « liste » et 4 « unique »).

D. Les JDialogs

Il en existe plusieurs types. Voici les 3 principaux :

1. JDialogCréer

Ce JDialog est un formulaire qui permet de créer par exemple une nouvelle entreprise, une nouvelle offre, un nouvel étudiant... Elles sont toutes accessibles à l'Administrateur, et l'Entreprise peut elle créer une Offre seulement.

Il vérifie qu'aucun champ n'est vide lors de la création, et ajoute l'objet à la base de donnée une fois qu'il est créé.

Exemple : *JDialogCréerOffreStage*.

Ajouter une Offre de Stage

Veuillez saisir les informations nécessaires

Libellé : Domaine :

Date de Début (jj/mm/aaaa) :

Durée (mois) : Valide : ☒ Disponible ☐ Indisponible

Description :

Il contient des JLabels suivis de JTextField. Pour le domaine on a opté pour un JComboBox, pour valide on a opté pour un RadioButton.
La description est une JTextArea.

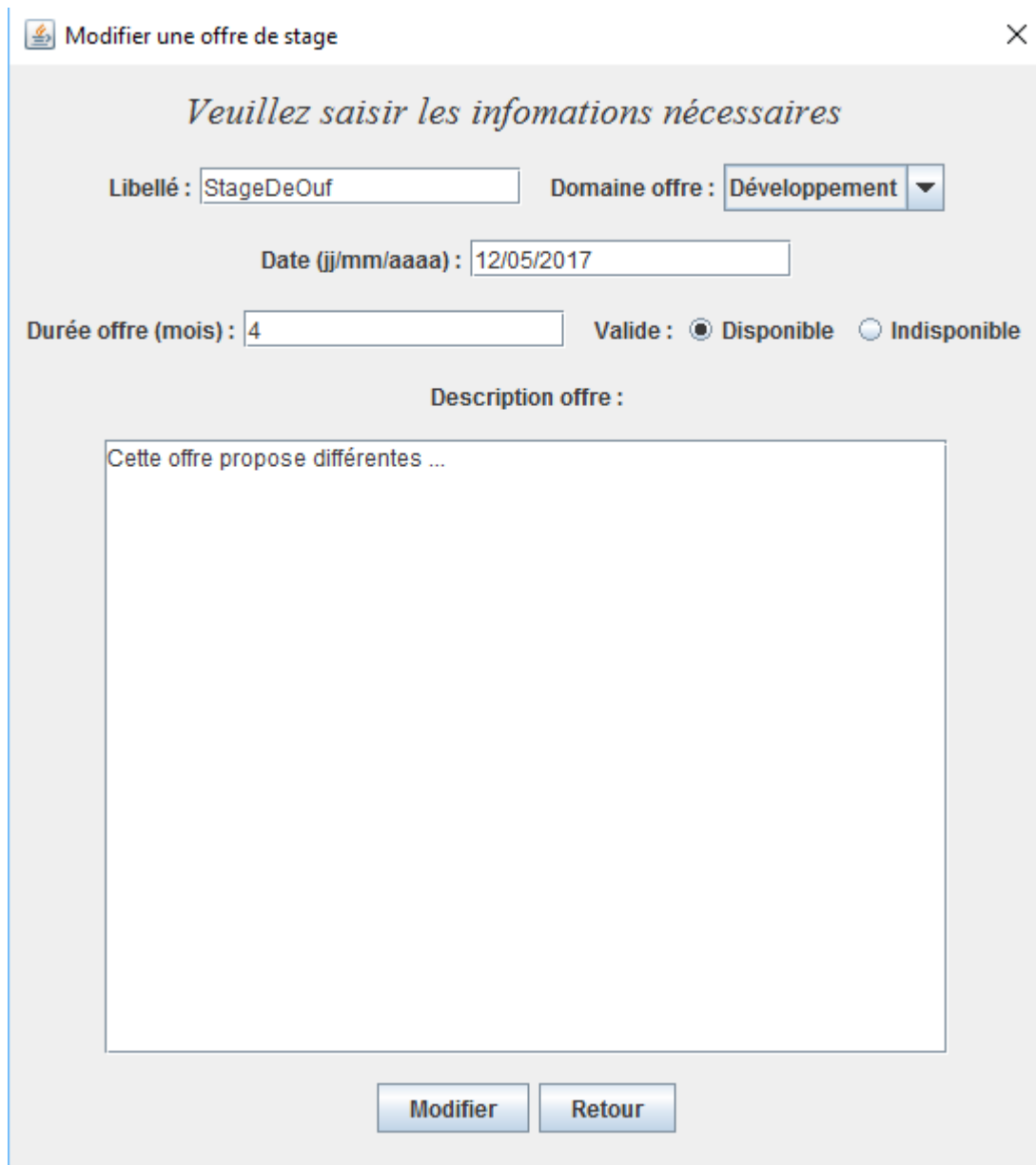
Le champ « **Disponible** » sert à dire si elle sera visible par les étudiants directement. L'entreprise peut l'initialiser à false pour la créer sans la rendre visible tout de suite, puis la modifier plus tard dans sa liste d'Offres.

Classes correspondantes : *JdialogCreerEtudiant*, *JdilaogCreerEntreprise*, *JdialogCreerOffreStage*, *JdialogCreerCandidature*.

2. *JDialogModifier*

C'est exactement la même chose que *JDialogCréer*, à la seule différence que les *TextField* contiendront les infos précédentes.

Exemple : *JDialogModifierOffreStage*.



The screenshot shows a Java Swing dialog box titled "Modifier une offre de stage". The dialog has a close button (X) in the top right corner. The main content area has a light gray background and contains the following elements:


- A title text: *Veillez saisir les informations nécessaires*
- A label "Libellé :" followed by a text field containing "StageDeOuf".
- A label "Domaine offre :" followed by a dropdown menu showing "Développement".
- A label "Date (jj/mm/aaaa) :" followed by a text field containing "12/05/2017".
- A label "Durée offre (mois) :" followed by a text field containing "4".
- A label "Valide :" followed by two radio buttons: "Disponible" (selected) and "Indisponible".
- A label "Description offre :" followed by a large text area containing the text "Cette offre propose différentes ...".
- At the bottom, there are two buttons: "Modifier" and "Retour".

Classes correspondantes : *JDialogModifierEntreprise*, *JDialogModifierEtudiant*, *JDialogModifierOffreStage*, *JDialogModifierCandidature*.





3. *JDialogProfil*

Ce JDialog s'ouvre quand on clique sur « **Mon Profil** » sur un des 3 Panels Principaux.

Exemple : *JDialogProfilEtudiant*.

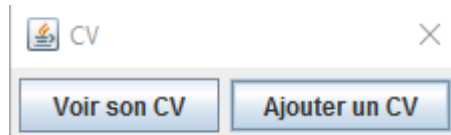
 Mon Profil ×

Informations du profil

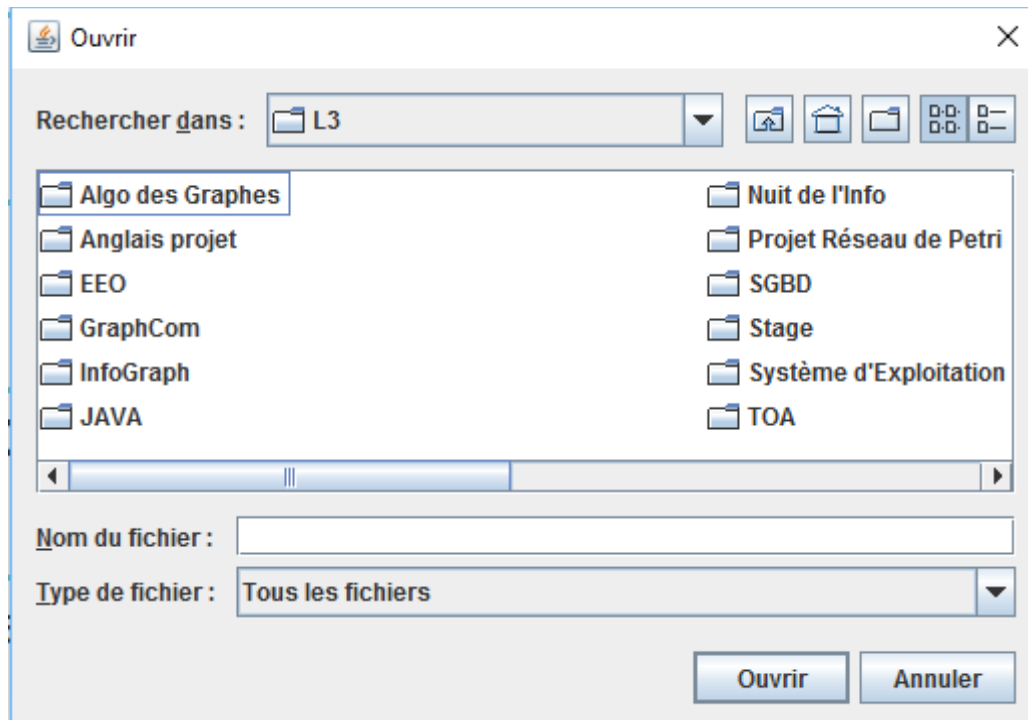
ID : 1	Nom : <i>Blanchis</i>
Prénom : <i>Aurèle</i>	Mot de Passe: *****
Ville : <i>Yerres</i>	Rue : <i>3 rue des dieux</i>
Code Postal : 91330	Téléphone : 0687952416
Mail : <i>aur09@gmail.com</i>	<div><div>Modifier </div><div>Retour </div></div>
<div>Consulter le CV </div>	<div>Lettre de Motivation </div>

Toutes ses informations sont affichées ici. En cliquant sur « **Modifier** » il peut tout modifier sauf son ID. Bouton « **Retour** » pour revenir en arrière, classique.

Les 2 boutons « **Consulter le CV** » et « **Lettre de Motivation** » ouvrent un JDialog « *JDialogChoixCV* » par exemple, qui permet de consulter ou d'en ajouter un (qui le remplacera s'il en existe déjà un). Même principe pour la lettre de motivation.



- « Voir son CV » ouvrira un PDF pour consulter son CV.
- « Ajouter un CV » va activer un JFileChooser pour choisir dans ses fichiers. Il sera ensuite mis à jour dans la BDD.



On va enregistrer le chemin du fichier dans un String, ce String étant un attribut de la classe Étudiant, il sera enregistré dans la BDD.

Classes correspondantes : Le reste de *IHM.JDialog*.

E. Les JTables.

Les JTables sont utilisées seulement pour l'Administrateur. Elles lui permettent de créer, modifier, supprimer tout type d'objet (étudiant, entreprise, offre de stage, candidature). Il a ainsi accès à tout sur le programme.

Elle fonctionne en 2 étapes :

- La classe « *tabledebut* » qui crée l'apparence du JTable : les entêtes, la méthode add, la méthode delete.
- La classe « *tablefinal* » qui ajoute les boutons « modifier », « ajouter » et

« supprimer », définit la taille du JTable, son titre. Elle prend en argument la classe « *tabledebut* ».

Quand on créera un JTable dans le code, c'est la classe « *tablefinal* » uniquement qui sera utilisé.

Liste des Etudiants										
ID	MDP	NOM	PRENOM	VILLE	RUE	CP	TEL	MAIL	CV	LETTRE
1	mdp	Blanchis	Aurèle	Yerres	3 rue des dieux	91330	0687952416	aur09@gmail.com	C:\Users\Aurèle...	C:\Users\Aurèle...
3	a	a	a	a	a	a	a	a		
4	b	b	b	b	b	b	b	b		
5	p	p	p	p	p	p	p	p		
6	s	s	s	s	s	s	s	s		
7	s	s	s	s	s	d	s	m		

(Désolé, ma BDD contient sur cette image beaucoup d'étudiant « test »).

Classes correspondantes : Voir *IHM.JTable* (8 classes : 4 debut et 4 final).

F. Les DAO

Nos classes DAO servent à relier notre code JAVA à notre base de données (phpMyAdmin).

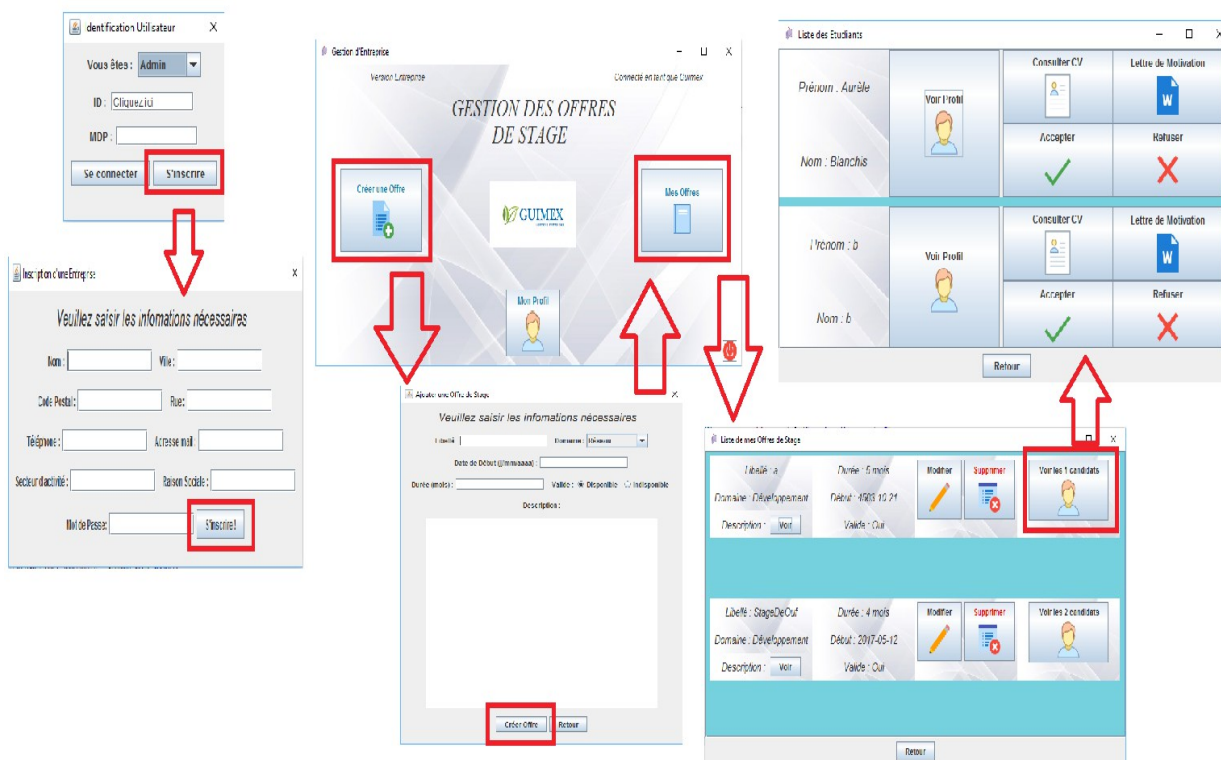
Il existe autant de classe DAO que de table dans notre base de données : *AdminDAO*, *CandidatureDAO*, *EntrepriseDAO*, *EtudiantDAO*, *OffreStageDAO*.

Tout d'abord on utilise la classe *Connexion* qui fait le lien avec la base de données. Ensuite ces classes DAO ont des méthodes comme *Create()*, *Delete()*, *Update()*, qui sont basiques. Et d'autres comme *Find()*, *Relier()*, *Lister()*, qui permettent par exemple de faire le lien entre l'entreprise et ses offres de stage, puis de les lister toutes.

G. Fonctionnement du code et liens entre classes

Le programme débute tout d'abord avec l'inscription d'au moins 1 étudiant et 1 entreprise avec les classes *JDialoCréer* vu précédemment. L'entreprise va ensuite créer sa première offre de stage avec le *JDialoCréerOffreStage*.

Une fois créé, il peut aller la consulter pour la modifier ou voir ses candidats (pour l'instant 0 puisqu'il vient de la créer). Voici comment ça fonctionne sur notre code :

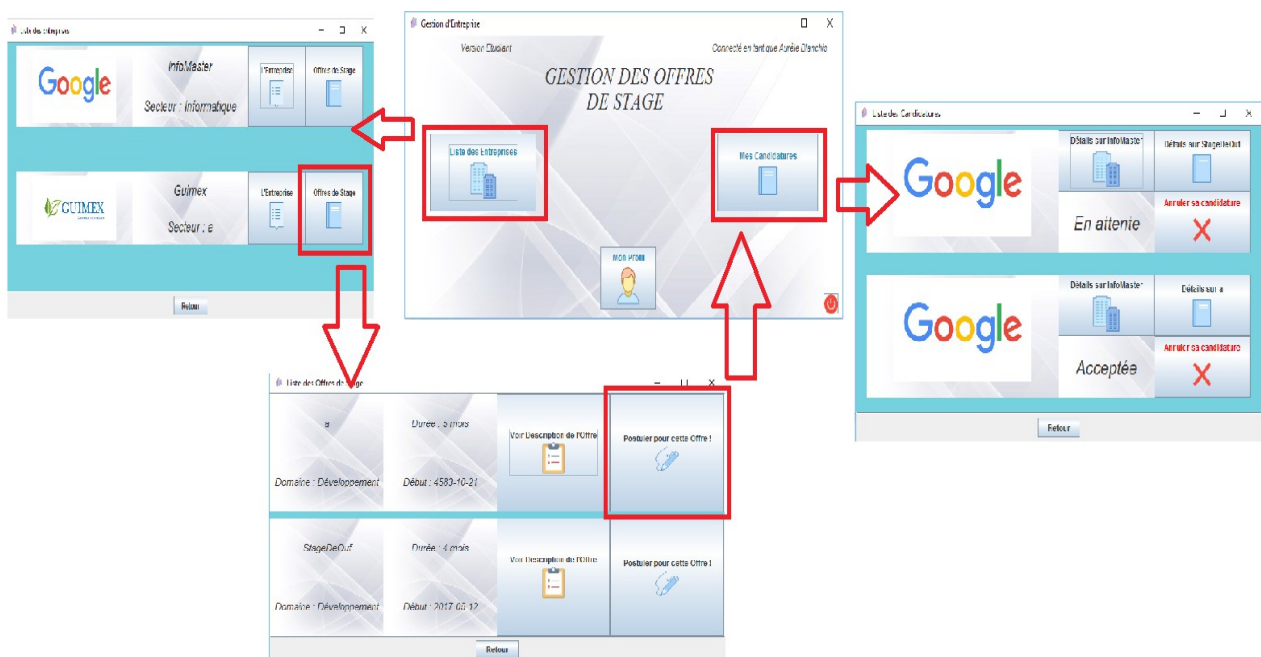


Quand il y aura des candidats, l'entreprise pourra choisir de voir son CV, Lettre de Motivation, son Profil, l'accepter ou le refuser.

Mais comment postuler ? C'est ce que nous allons voir tout de suite :

Une fois connecté en tant qu'étudiant, on a juste à aller voir la liste des Entreprises (et leurs détails) , puis leur liste d'Offre de stage (et leur description). Une fois cela fait, on peut choisir de postuler.

Une fois qu'on aura postulé, la candidature s'ajoutera à la liste de candidature de l'étudiant. Voici comment ça fonctionne sur notre code :



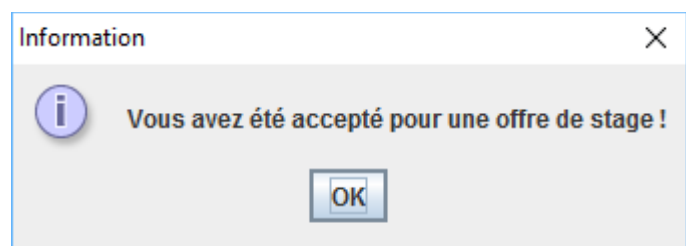
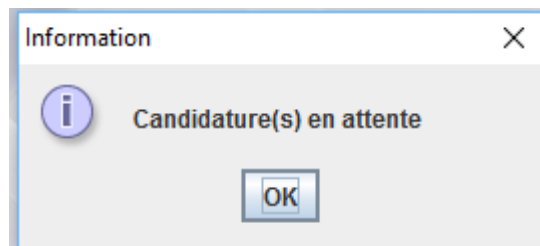
Dans sa liste de candidature, l'Étudiant verra « En attente » si l'Entreprise n'a toujours pas répondu, ou Acceptée s'il est accepté.

S'il a été refusé, sa candidature pour cette offre va disparaître de sa liste de candidatures.

Les Pop-ups :

Une fois qu'on a postulé, l'Entreprise va recevoir un « Pop-up » à chacune de ses connexions qui indiquera « Candidatures en attente », quand un candidat a postulé à une de ses offres de stage sans qu'elle ait encore prit la décision de l'accepter ou de le refuser.

Si elle choisi de l'accepter, lors que la prochaine connexion de l'Etudiant un « pop-up » apparaîtra pour lui indiquer qu'il a été accepté pour une offre. Et de même s'il a été refusé.



Si une Entreprise accepte un étudiant pour une offre de stage, cette offre n'apparaîtra plus dans la liste des offres de l'Entreprise.

Le seul moyen qu'elle y revienne est que soit l'Étudiant annule sa candidature (l'Entreprise serait alors averti), soit que l'Entreprise passe elle-même le boolean valide à true dans « *JDialogModifierOffreStage* ».



Pour finir, nous avons bien fait attention à supprimer les offres de stage de l'Entreprise concernée lorsqu'on supprime cette Entreprise. De même pour lorsqu'on supprime une offre de Stage, toutes les candidatures associées sont supprimées.

III. Le Journal de Bord

A. *Déroulement du Projet.*

On a commencé par choisir notre Sujet : la Gestion d'offre de stage.

On a ensuite débuté par créer un diagramme de classe sur papier pour avoir une idée de la construction de notre programme. Une fois terminé, nous avons créé nos classes Objets : *Admin*, *Candidature*, *Entreprise*, *Etudiant*, *OffreStage*, *Utilisateur*.

Une fois leur liens mis en place, on a commencé à créer notre interface graphique.

Une fois l'interface principale mise en place (les 3 panels principaux), on s'est attaqué à la connexion, mais pour cela il fallait qu'on ait une base de donnée.

Comme on ne s'y connaissait pas en base de donnée, on s'est renseigné et on a choisi MySQL (WAMP) avec PhpMyAdmin. On a ensuite créé notre base de données avec autant de table qu'on avait de classe Objet. Une fois cela fait, on est allé se renseigné sur comment fonctionnait les DAO.

On a commencé par la classe *Connexion*, puis par tous les autres DAO. Au début, seules les méthodes *Create()*, *Update()* et *Delete()* étaient utiles.

On a ensuite mis en place le reste de notre interface graphique avec tous les JDialogs (pour créer, modifier) en faisant des formulaires.

On a aussi crypté les mots de passe dans notre BDD avec la classe *Md5*.

On est ensuite parti se renseigner sur les JTable puisqu'on ne connaissait pas du tout. Une fois maîtrisé, on a créé un JTable pour presque chaque table de notre base de données, pour pouvoir gérer chaque données de notre programme depuis l'interface graphique (ceci a été très utile pour faire des tests par la suite).

On a ensuite créé tous les Panels avec Liste (ce sont eux qui relient l'entreprise et l'étudiant entre eux principalement).

On a donc ensuite créé d'autre méthodes un peu plus complexes sur nos DAO pour pouvoir bien relier nos tables de notre base de données ensemble.

Une fois ceci terminé on a rendu notre interface graphique plus belle.

Pour finir, on a créé le diagramme de cas d'utilisation et les diagrammes de séquences.

B. Difficultés rencontrées

- Nous avons tout d'abord eu beaucoup de mal avec les DAO, puisqu'on paraît de 0. Comprendre comment relier une base de donnée à son code JAVA, comment et où utiliser les DAO, etc. Ceci nous a prit une semaine à peu près.
- Les JTable. On ne s'en était jamais servi auparavant, on a donc du aller se renseigner sur la façon dont s'en servir. Ceci n'a prit qu'une demi-journée.

Sinon, l'ensemble du projet s'est déroulé plutôt bien, on a évidemment rencontré des difficultés par-ci par-là, mais rien d'insurmontable.