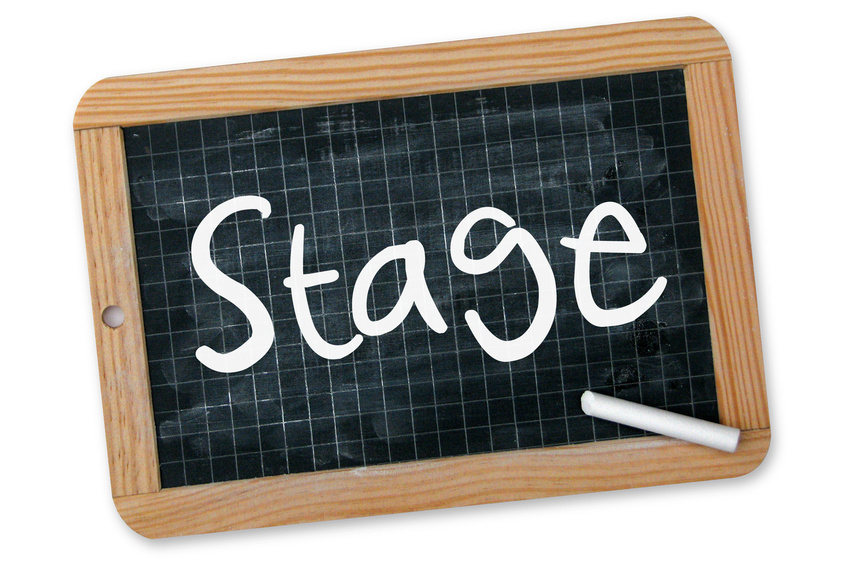
|  |  |
| --- | --- |
|  | **2017** |
|  | Guillemet Quentin & Cyril Bourget  User |

|  |
| --- |
| **[Gestion des Offres de Stage]** |
| Programme pour étudiant et entrepreneur, qui regroupe des offres de stage. |



**SOMMAIRE**

* Les différents diagrammes

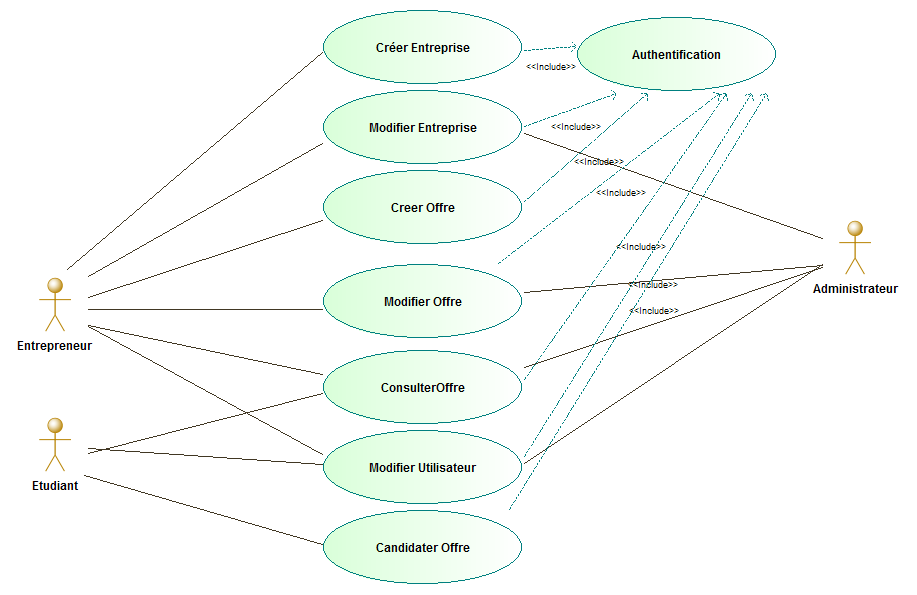
-Diagramme de cas d’utilisation

-Diagramme de classe

-Diagramme de séquence

* Fenêtres du programme
* La Base de données
* Petit passage dans le code

1. Les Différents Diagrammes
   1. Diagramme de cas d’utilisation



L’entrepreneur, l’étudiant ou l’administrateur doivent tous passé par l’Authentification pour arrivé à se connecter à la base de donné des offres de stage.

La création d’un nouvel utilisateur se fait avant tout cela, c'est-à-dire qu’ils doivent rentré leurs coordonnées adéquates aux champs indiqués.

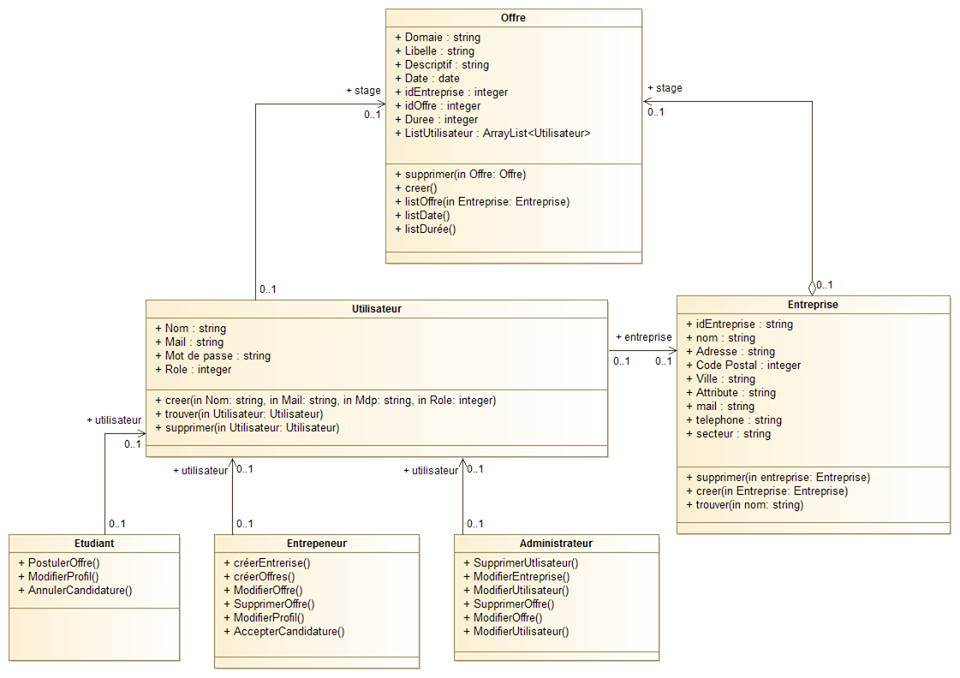
L’étudiant après sa connexion peut soit Consulter les offres avec possibilité de candidater pour une offre ou bien de Modifier son profil.

L’entrepreneur a accès à son profil (donc possibilité de le modifier), Consulter les offres, Modifier ses offres et son entreprise. La création de son entreprise sera demandé juste après sa première connexion.

L’enregistrement d’un administrateur sera fait directement via la base de donnée et aura donc accès à tout (excepté les mots de passe des utilisateurs qui seront au préalable crypté).

Toute modification ou bien Consultation d’offre donnera court à une requête vers la base de donné.

1. Les Différents Diagrammes
   1. Diagramme de classe

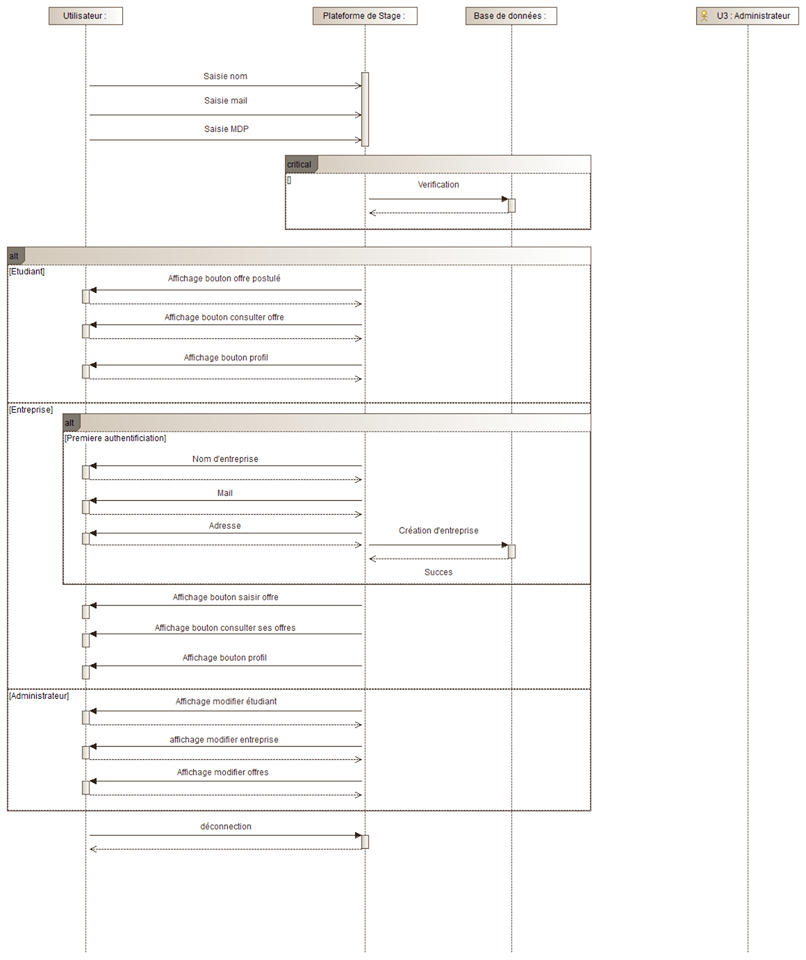


Voici les grandes classes implémentées dans le programme.

L’utilisateur « Etudiant », « Entrepreneur » ou bien « Administrateur » seront unique par leur mail

L’entreprise aura l’obligation de créer son entreprise et une seule après sa première connexion.

Une erreur de cardinalité entre Entreprise et Offres, car une Entreprise pourra créer une ou plusieurs offres (1-n).

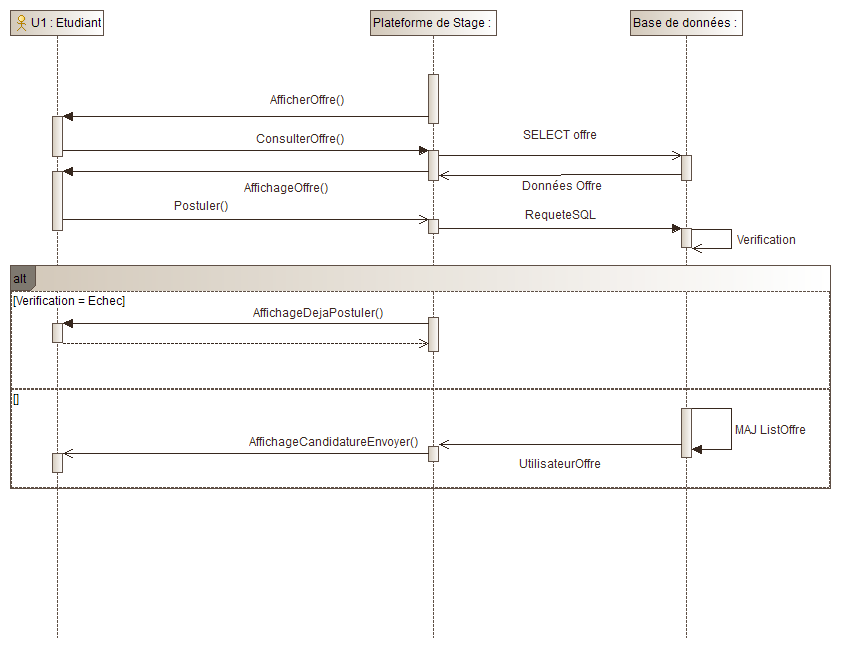
1. Les Différents Diagrammes
   1. Diagrammes de séquence 

L’utilisateur devra entrer ses logs de connexion. Une vérification se fera par le biais d’une requête SQL et d’une condition « IF » pour savoir s’il existe un Utilisateur qui a les mêmes logs. (Hashage du mot de passe inclus).

Après cela, l’interface changera par rapport au « Role » qui se connectera. Les rôles vont de 1 à 3. 1 = Etudiant, 2 = Entrepreneur, 3 = Administrateur.

Le clique sur l’une des images d’accueil changera le rôle interne qui sera aussi placé dans la requête SQL de connexion.

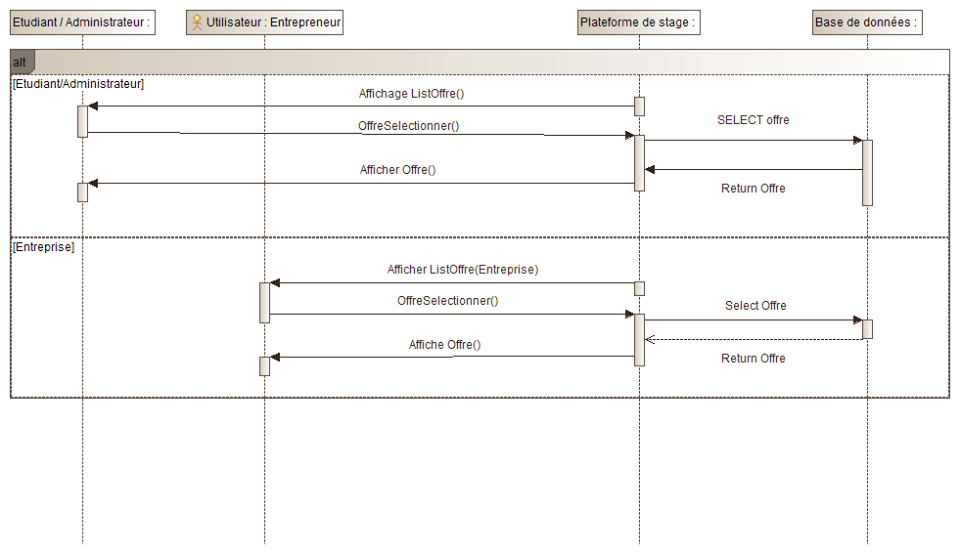
(Voir « Fenetre du programme »)

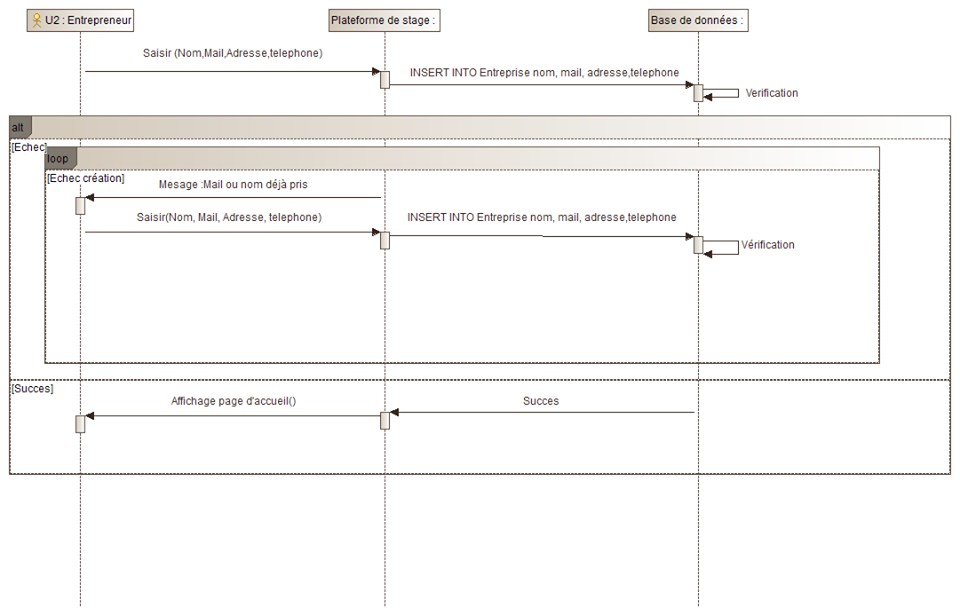


Un étudiant pourra ConsulterOffre() via un clique sur le bouton adéquate et une requete SQL se fera pour afficher toutes les offres de stage dans un Table().

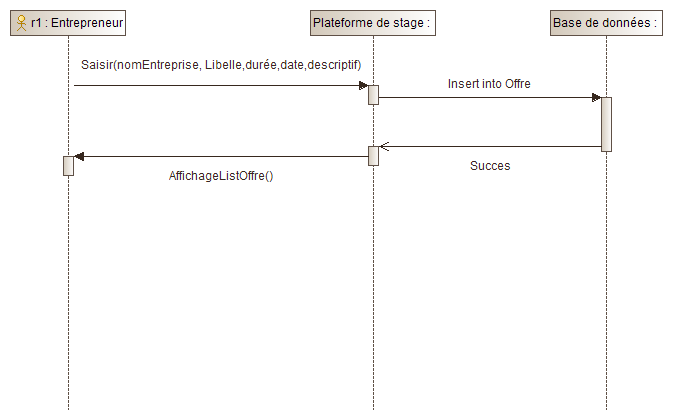
Il pourra par la suite en sélectionner une puis si il le veut postuler ou non.

Si l’étudiant a déjà postulé à une offre donné, il recevra un message d’erreur. (pas encore implémenter IDEE : dans la requête faire une vérification dans la listOffres si il en existe pas déjà une postulation avec celle faite sur la sélection)

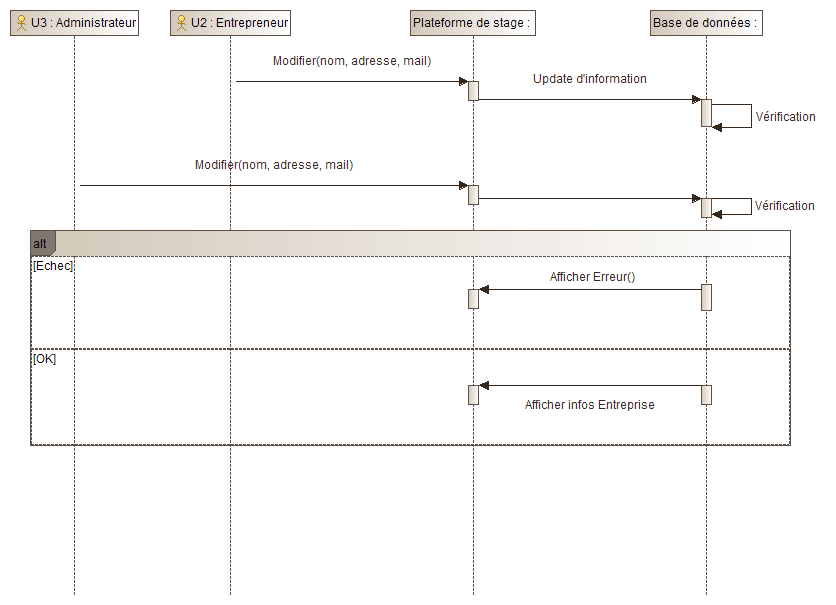


Quand une offre est sélectionnée, une fenetre sera afficher avec tout le descriptif de l’offre.

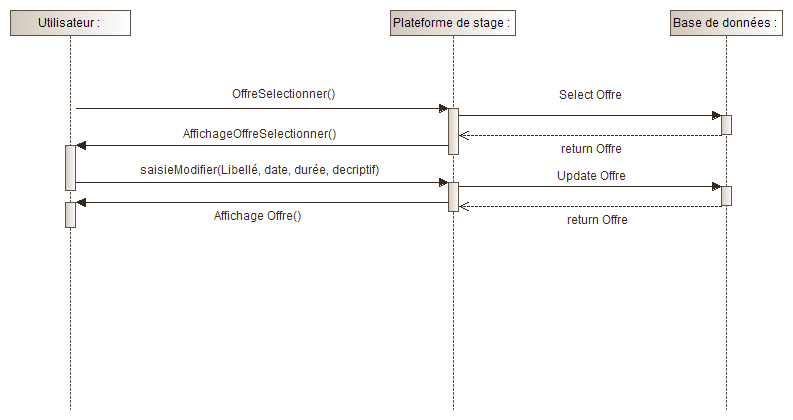
Voici la création d’une entreprise. Toute insertion d’une nouvelle entreprise sera verifier, en cas de duplicata par une autre entreprise, elle sera refuser. En cas de succès de la création de l’entreprise, il retournera a la fenêtre d’accueil avec Consultation Offres, son profil ainsi que Saisir Offre.

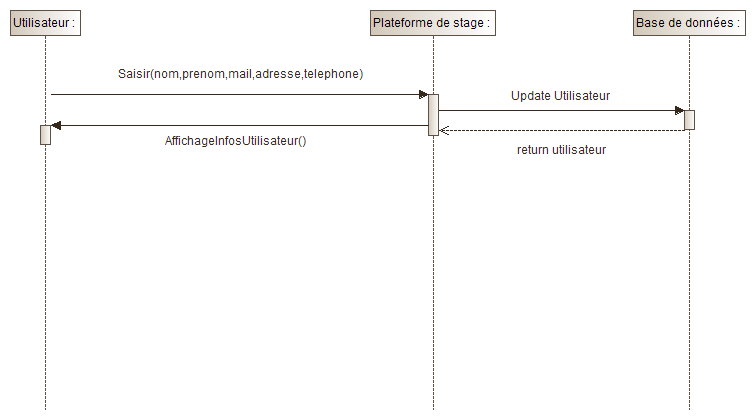


Une entreprise peut saisir une ou plusieurs offres. L’utilisateur devra saisir plusieurs champs demandé et sera inséré à la base de donné par la suite après l’appuie du bouton « Envoyer ».



Une entreprise et un administrateur pourront si ils le veulent changer de nom, adresse et/ou mail. Tout changera créera une requête de type UPDATE sur la base de donné au niveau de l’utilisateur concerné.

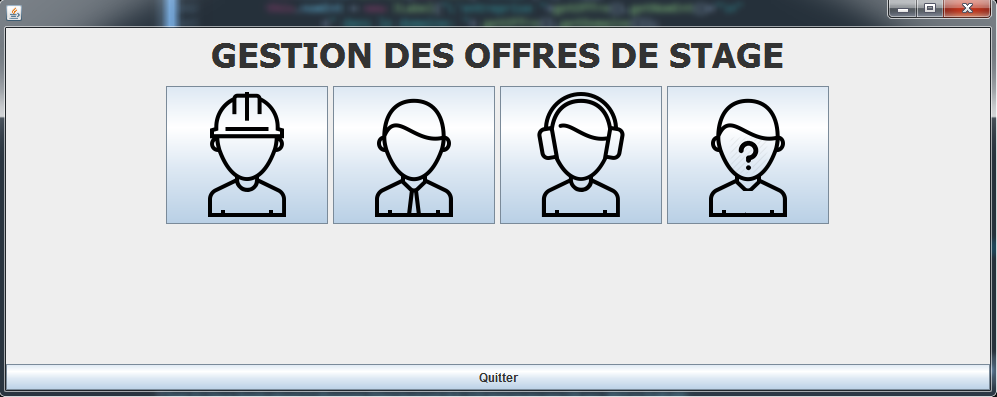


Ici nous pouvons voir après la sélection d’une offre, qu’il y a une requête qui se fait sur la base de donné dans la Table « offre » puis retourne tout le descriptif dans une fenêtre.

Voici la modification d’un utilisateur en général à partir de son profil.

Requête Update sur la Table Utilisateur.

1. Fenêtres du programme



Voici l’accueil, au lancement du programme vous serez envoyé sur cette page.

Il y a 4 boutons qui symbolisent chacun une action.

-le premier : Connexion en tant qu’Administrateur



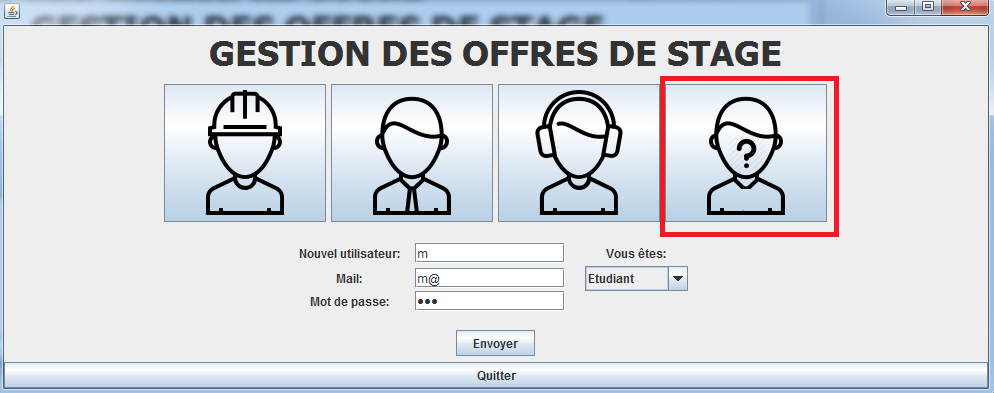
-le second : Connexion en tant qu’Entrepreneur



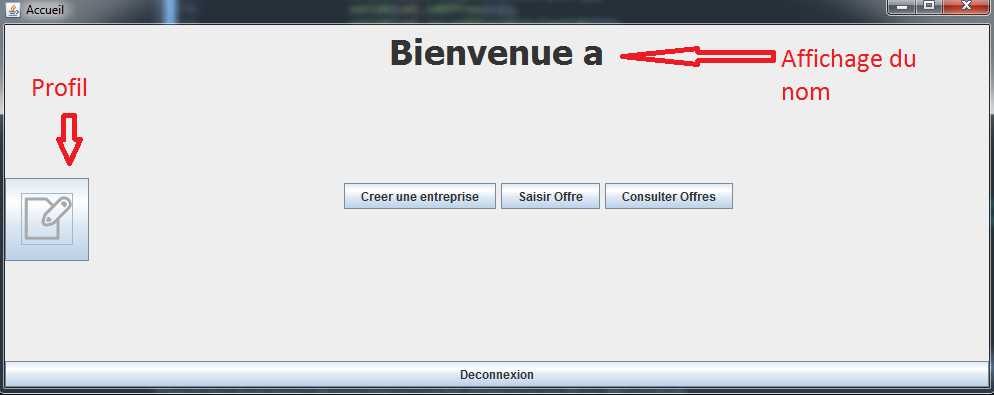
-le troisième : Connexion en tant qu’Etudiant



-le quatrième : Création d’un nouvel Utilisateur



A la création d’un nouvel Utilisateur, vous allez être mis sur la fenêtre de votre « Rôle » choisit.

Une connexion en tant qu’Administrateur vous emmènera sur cette fenêtre : 

Le profil est représenté par un bouton à gauche du panel(en BorderLayout.EST) de la fenêtre, le titre en haut (BorderLayout.NORTH), les boutons centraux (BorderLayout.CENTER) et pour finir en bas de fenêtre, le bouton « Deconnexion » (BorderLayout.SOUTH).

Toutes les fenêtre de « ChoixFrame » seront identique excepté les boutons centraux qui changeront en fonction du Rôle.

NB : Nous n’avons pas totalement implémenté le role de l’administrateur, mais on a toutes les classes, méthodes déjà prête à utilisation.

Passons maintenant a l’entrepreneur :

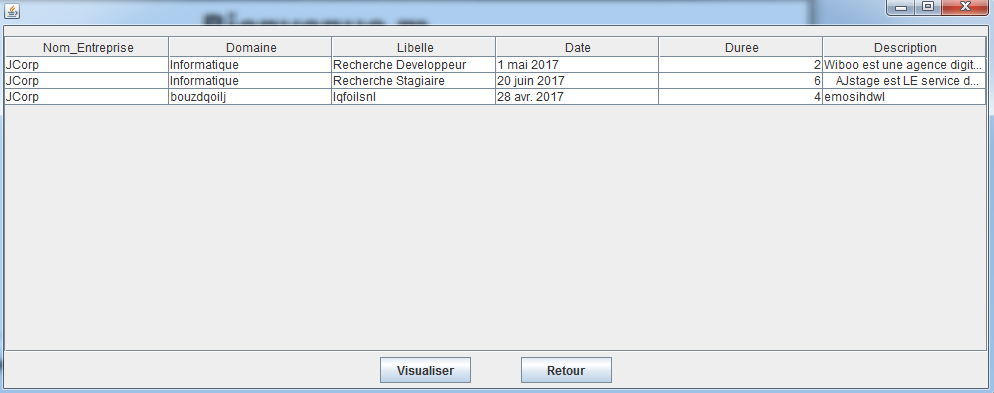
Arrivé sur cette page de ChoixFrame() vous pouvez cliquer sur SaisirOffre, ConsulterOffres, Profil, ou bien Déconnexion.

SaisirOffre : 

Tout les Label et TextField sont placé dans un panel GridBagLayout

Le bouton du chemin de stockage ne renvoi rien pour l’instant, mais le bouton du chemin fonctionne bien, il instancie la classe EssaiChoixFichier() qui ouvre une fenêtre du système.

ConsulterOffre :



Voici l’affichage de toutes les offres de la table Offre de notre base de donné.

Il peut sélectionner(Selection unique et non multiple, et l’une des options d’un Table()) une offre et l’afficher :



Voici une offre entré aléatoirement. Il y a en premier le Libellé de l’offre, puis le nom de l’entreprise, avec la durée du stage, et la date du début puis pour finir sa descritpion dans son intégralité placé dans un TextArea avec un ScrollBar mis en Vertical si besoin.

Passons maintenant à l’Etudiant :

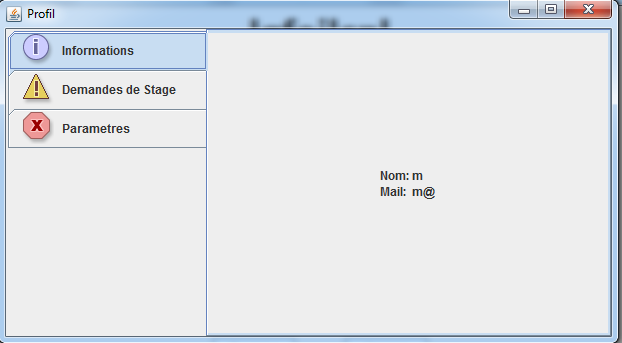


Même fenêtre que les deux autres excepté les boutons centraux. Ici nous pouvons voir que l’etudiant ne pourra que consulter les offres ou bien son profil.

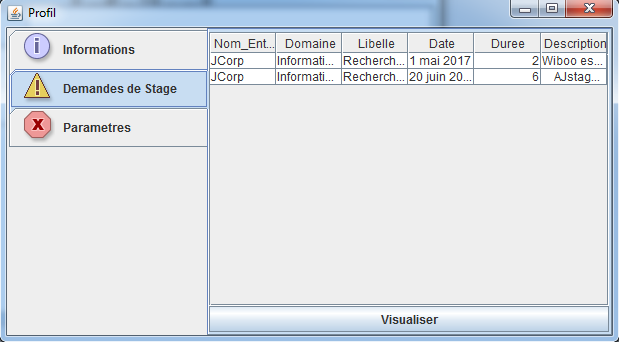
Pour la consultation d’offre rien de change, sauf pour la visualisation d’une offre ou un bouton « Postuler » apparait :



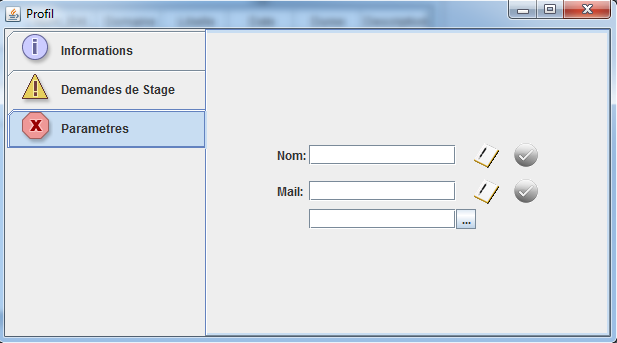
Le profil qui est plus ou moins identique avec les 2 autres rôles, 3 onglets se présente sur la gauche.



Informations : C’est ici ou est récapitulé toutes les informations sur l’Utilisateur

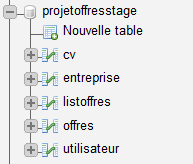


Demande de Stage : recupère tout les stages postuler par l’étudiant.



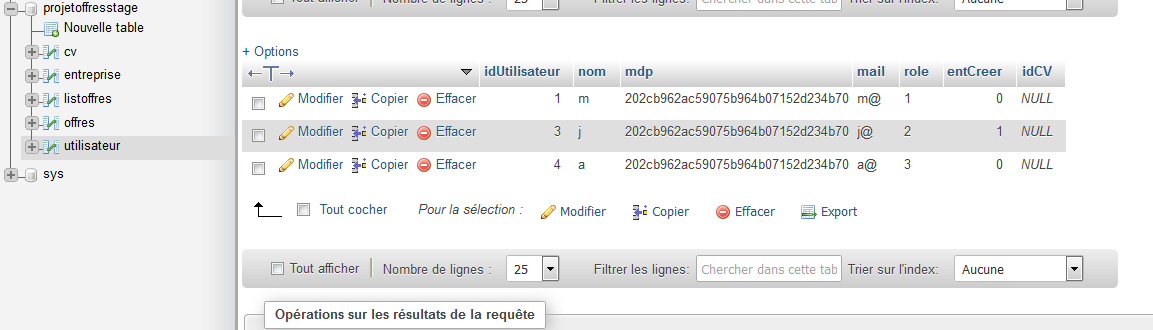
Paramètres : Permet de modifier les informations de l’utilisateur en question ainsi que son CV(qui n’a pas encore été implémenté mais à l’ajout d’un cv, celui-ci sera mis dans l’onglet Information en image simplement par manque de temps et de notion sur les buffer/transformation d’une image en BYTES pour la placer dans la BDD).

1. La Base de données



Notre base de données se compose de 5 Tables.

TABLE utilisateur :



IdUtilisateur : (INT) qui s’incrémente tout seul à chaque ajout d’un nouvel utilisateur

Nom : (VARCHAR) simple nom

Mdp : (VARCHAR) qui sera crypté par notre programme en « md5 »

Mail : (VARCHAR) qui sera unique

Rôle : (INT) de 1 à 3

entCreer : (BOOLEAN/TinyInt) utile seulement pour les entreprises, si elle on créé ou non leur entreprise après leur premiere connexion, si oui le boolean passera à vrai donc à sa prochaine connexion l’entreprise ne repassera pas par la fenêtre de création d’une entreprise.

idCV : (INT) Aurait servi a lié avec l’idCV de la Table CV.

TABLE offres :



idOffre : (INT)

idEntreprise : (INT) relié avec l’idEntreprise de la table entreprise

NomEnt : (VARCHAR) nom de l’entreprise

Domaine : (VARCHAR) domaine de l’offre

Libelle : (VARCHAR) titre de l’offre

Date : (DATE) date de l’offre formatter en (yyyy-MM-dd) dans la BDD

Duree : (int) durée de l’offre

Descriptif : (VARCHAR) Description de l’offre

TABLE listoffres :



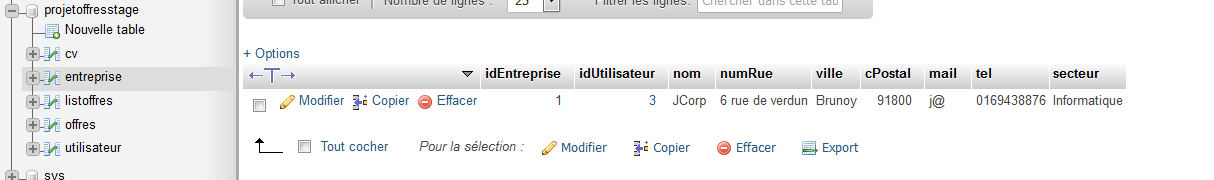
idUtilisateur : (INT) lié avec l’idUtilisateur de la table utilisateur

idOffre : (INT) lié avec l’idOffre de la table offres

idEntreprise : (INT) lié avec l’idEntreprise de la table entreprise

Une spécification avec cette table qui est fait dans la partie requête SQL de notre programme c’est qu’un utilisateur Etudiant ne peut pas postuler 2 fois à la même offre, il faut que l’idOffre soit différent pour une idUtilisateur donné pour chaque nouvelle entré dans cette table listoffres.

TABLE entreprise :



idEntreprise : (INT) auto-incrémentation

idUtilisateur : (INT) lié avec l’idUtilisateur de la table utilisateur

Nom : (VARCHAR) nom de l’entreprise

NumRue : (VARCHAR) adresse de l’entreprise

Ville : (VARCHAR) ville de l’entreprise

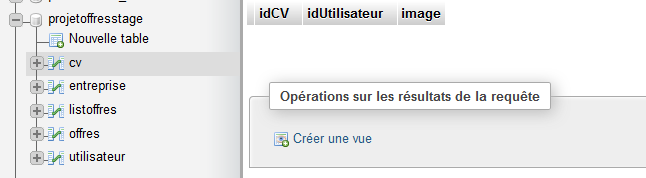
cPostale : (INT) code postale de l’entreprise

Mail : (VARCHAR) mail de l’entreprise

Tel : (VARCHAR) numéro de téléphone de l’entreprise

Secteur : (VARCHAR) secteur de l’entreprise

TABLE cv :



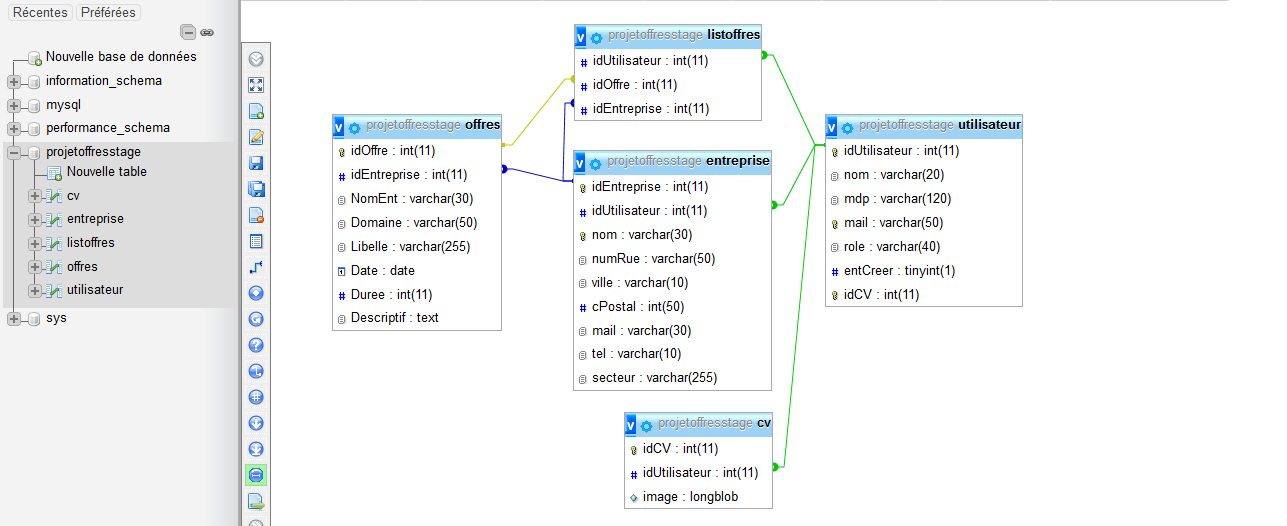
idCV : (INT) auto-incrémentation

idUtilisateur : (INT) lié à l’idUtilisateur de la table utilisateur

image : (LONGBLOB) d’après nos recherche, ce type servirait à stocker une image qui aurait été préalablement mis en BYTES pour être stocker ici même dans cette table.

Cette table n’est pas du tout utilisé par manque de temps ainsi que de savoir comme nous l’avons précisé plus haut.

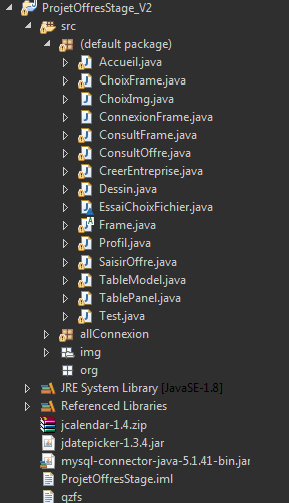
Voici les relations (KEY\_FOREIGN, PRIMARY…) sur tout cette base de données :



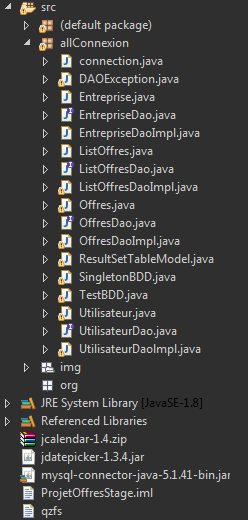
1. Petit passage dans le code

Nous avons séparé nos classes en 2 packages, l’une avec les frames et l’autre avec les interfaces (UtilisateurDao, OffresDao…), les classes primaires importantes (Utilisateurs, Offres, Entreprise..) et nos implémentation à partir de ces classes primaires et leur interface (UtilisateurDaoImpl, OffresDaoImpl..).

PACKAGE DEFAULT :

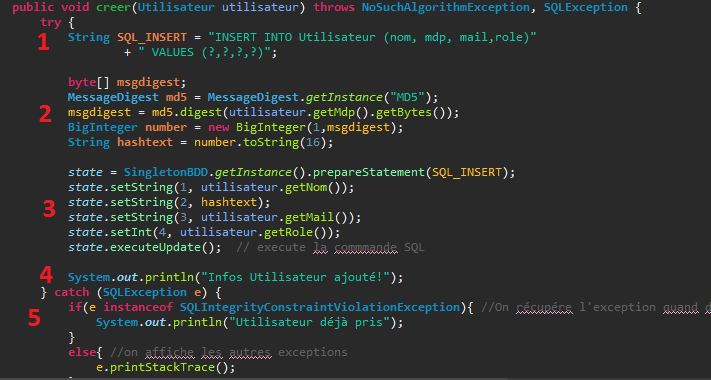


PACKAGE ALL\_CONNEXION :



La classe SingletonBDD, sera appelé à chaque fois qu’il y aura une interaction avec notre base de données, pour faire un SELECT par exemple. Elle est en Static car elle est appelé qu’une fois avant d’être refermé après la requête pour plus de sécurité.

DAOException qui extends d’Exception, qui permettera d’afficher un message d’erreur si besoin.

Les \*\*\*Dao sont les interfaces et \*\*\*DaoImpl sont leur implémentation avec les PrepareStatement pour la préparation de chaque requête SQL 

Voici un exemple : la méthode créer() dans la classe UtilisateurDaoImpl

Qui prend en argument un Utilisateur. L’Utilisateur utilisateur de cette méthode sera crée à partir de la fenêtre d’accueil de notre programme. Qui prendra le nom, le mot de passe, le mail et son role choisit.

Nous préparons la requête dans un String, ici en (1)

Bien évidemment le mot de passe entré dans le champs adéquate n’est pas crypté, donc on le crype dans le bloc (2) même que l’on va placé dans un String appelé hashtext dans ce cas là.

En (3) l’appel du SingeltonBDD qui va préparer la requête SQL préalablement ecrite dans le String précédemment. Qui va prendre plusieurs valeur par rapport au « ? » dans VALUES de la requête SQL.

Exemple :

State.set\*\*\*\*(1, \*\*\*) : 1 signifie que c’est la 1ere « ? » valeur donc pour « nom », puis après la virgule c’est ce que l’on veut mettre dans la requête à cette valeur là et ainsi de suite pour le reste du bloc (3).

En (4) le System.out.print est mal placé, il devrait être dans le bloc catch.

En (5) le catch permet en cas d’erreur de faire appel à l’Exception en question est d’afficher si il y a erreur que l’utilisateur entré est déjà utilisé (vérifié par rapport a la clé unique dans la table utilisateur).

1. Conclusion

Nous n’avons tout à fait fini le programme, mais le plus gros est fait (ajout d’un utilisateur, connexion à partir de logs, consultation des offres, saisie d’une offre, création d’une entreprise, modification du profil pour l’étudiant, affichage des données personnelles par le biais du profil, consultation des offres postulé pour l’étudiant..)

On a pas mal ralentit avec l’acquisition de nouvelle connaissance qui, grâce à ce projet nous montre beaucoup de fonctionnalité en plus et très utile. Les différents Panel, avec leurs spécifications comme pour le GridBagLayout qui est très utile pour la mise en forme d’un formulaire, mais aussi pour l’implémentation des requêtes SQL avec le Hashage du mot de passe qui pour nous était nouveau.

On aurait pu avec le temps faire une meilleur interface graphique, mettre au clair notre programme (car il y a pas mal de coquille), mettre beaucoup plus de fonctionnalitées.

Les Fonctionnalitées manquantes :

* Dans le profil Entreprise : modification des données personnelles (entreprise et utilisateur)
* Toujours dans le profil Entreprise : Affichage de leur offre puis visualisation des étudiants qui ont postulé, et y ajouter les boutons « accepter », « refuser » ou « en attente ».
* La partie de l’administrateur doit être revus
* Au niveau du profil, modification des données personnelles
* Peut etre ajouter la liste des utilisateurs (etudiant et entreprise) puis de choisir de les supprimer ou pas. (ceci reste qu’une suggestion).