

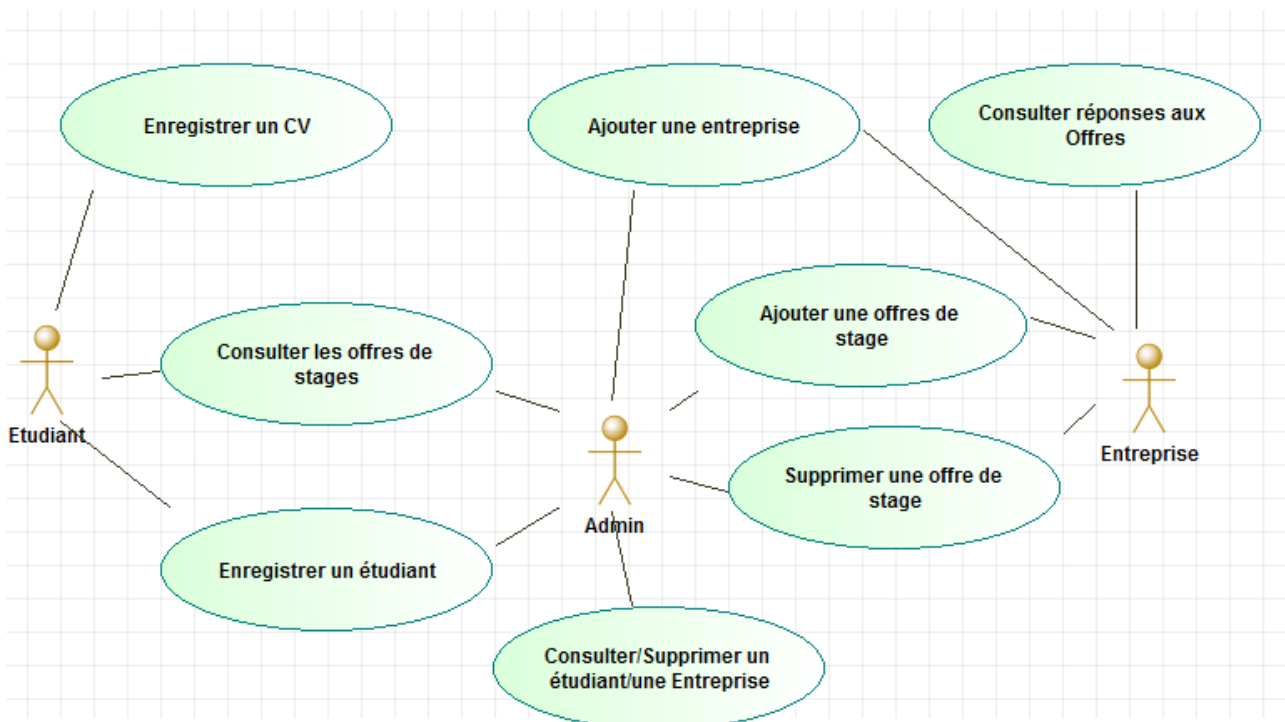
## Sommaire

Compte-rendu des séances	2
Diagramme des cas d'utilisation	3
Diagrammes de séquences	4
Diagramme de classes	12
Structure de la base de données	13
Explication du programme	14
Explication du code	27

## Compte-rendu des séances

Taches réalisées	Taches à faire	Difficultés
DATE: 09/03/17 -Diagramme de cas d'utilisation -Diagramme de classe -Base de données -Classe de gestion de la base de données -Interface graphique: -Page d'inscription -Page de Connexion -Page d'administration	-Interface Utilisateur -Coder les methodes -Pages de Recherches	
DATE: 17/03/17 -Inscription -Connexion -Modification du CV	-Entreprises et Offres de Stages	
DATE: 23/03 -Gestion de l'entreprise -Utilisation de Git	-Fonctions de recherches -Entreprises	-Apprendre Git
DATE:31/03 -Gestion des Offres de stages -Recherches de Stages	-Authentification des entreprises -Diagrammes de séquences	
DATE:06/04 -Gestion des Erreures d'identification -Diagrammes de séquences -Recherche des Stages	-Authentification des entreprises -Criptage des mot de passes	-Apprendre à faire un diagramme de séquence -Les bugs du logiciel modelio ont faire perdre beaucoup de temps avec les diagrammes de séquences
DATE:21/04 -Criptage -Mise à jour de la base de donnée finale	-Authentification des entreprises -Authentification de l'admin -Pages consulter entreprises -Pages consulter offres -Pages consulter Utilisateurs	-Adapter la nouvelle base de donnée au programme
DATE:23/04 -Authentification des entreprises -Fonctions des entreprises -Fonctions des utilisateurs	-Consultation et suppression des des entreprises,offres et utilisateurs par l'admin	-Fonction de recherche avancé qui tri les offres
DATE:24/04 -Consultation et suppression des des entreprises,offres et utilisateurs par l'admin -Finalisation		

## Diagramme des cas d'Utilisation

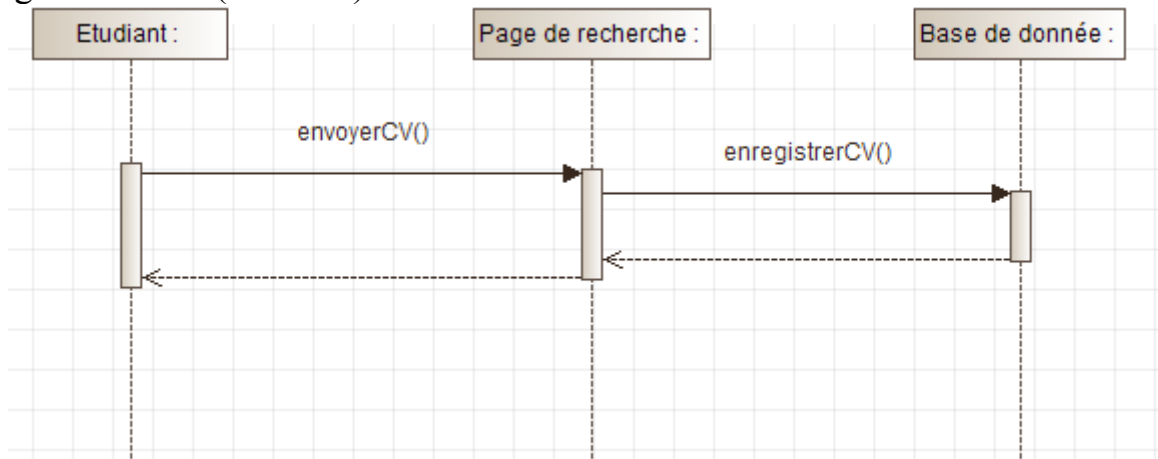


### Note:

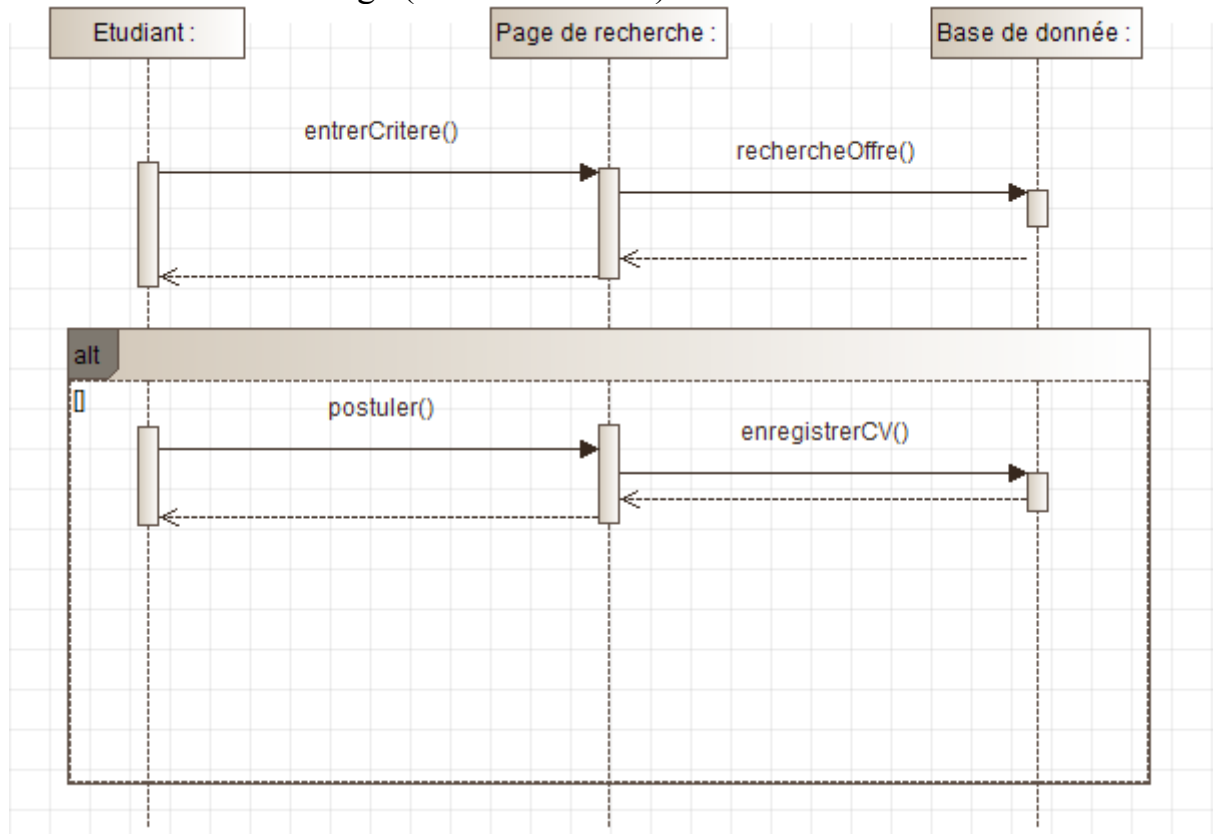
- Lorsque l'étudiant consulte les offres de stages, il peut s'y inscrire alors que l'admin quant à lui pourra les consulter et les supprimer.
- L'étudiant ne peut s'inscrire que lui-même alors que l'admin n'a pas de limite au nombre d'étudiants qu'il inscrit.
- L'Entreprise ne peut s'inscrire qu'elle-même alors que l'admin n'a pas de limite au nombre d'entreprises qu'il inscrit.
- L'Admin peut supprimer toutes les offres de stage alors que l'entreprise ne peut supprimer que celles qu'il a lui-même crée.
- De même, l'admin peut crée des offres de n'importe quelle entreprise.

## Diagrams de séquences

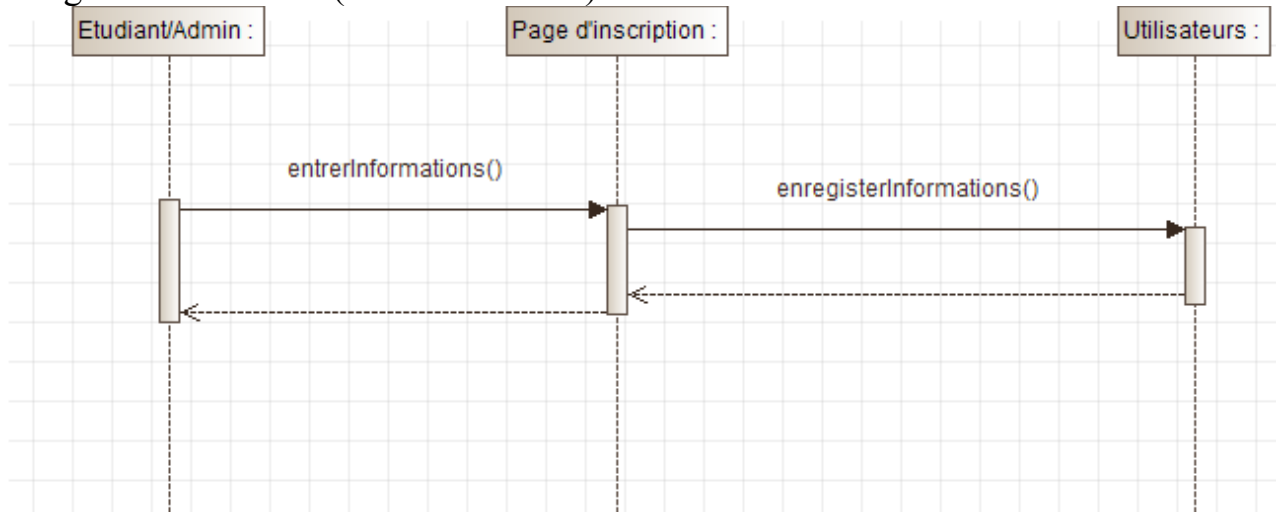
Enregistrer un CV (Etudiant):



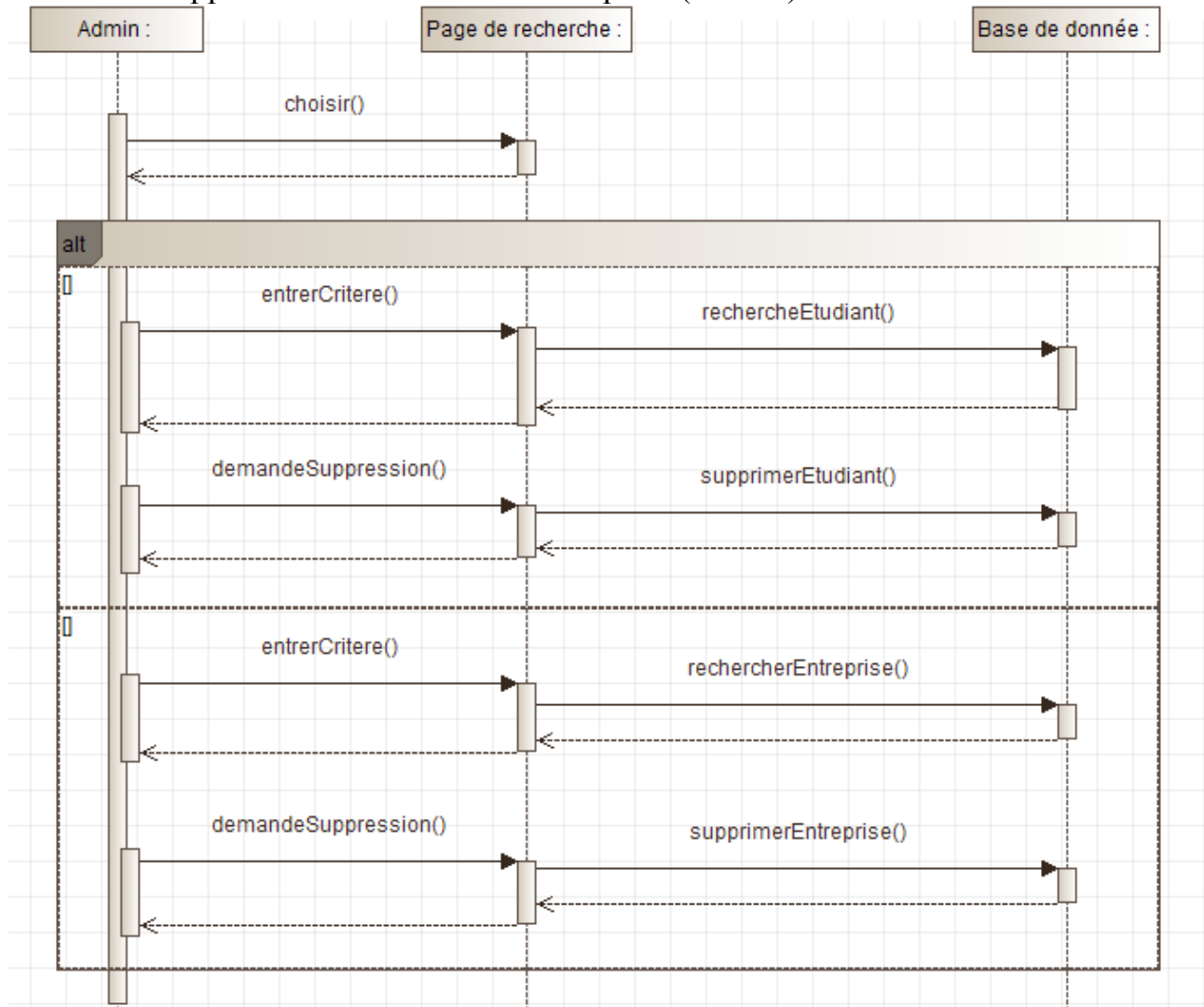
Consulter les offres de stage (Etudiant/Admin):



## Enregistrer un étudiant (Etudiant/Admin):



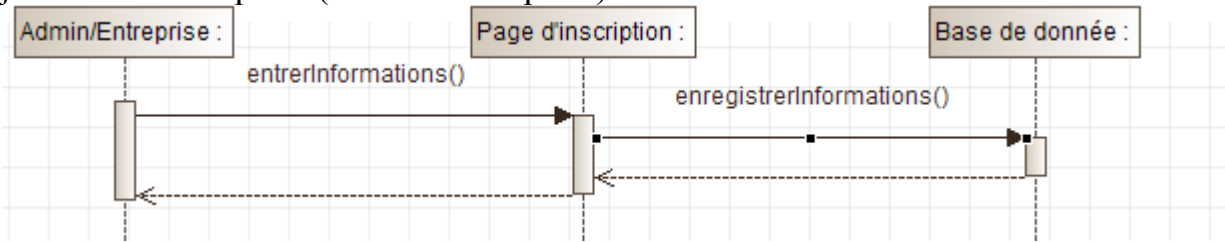
## Consulter/Supprimer un étudiant/une entreprise (Admin):



### Notes:

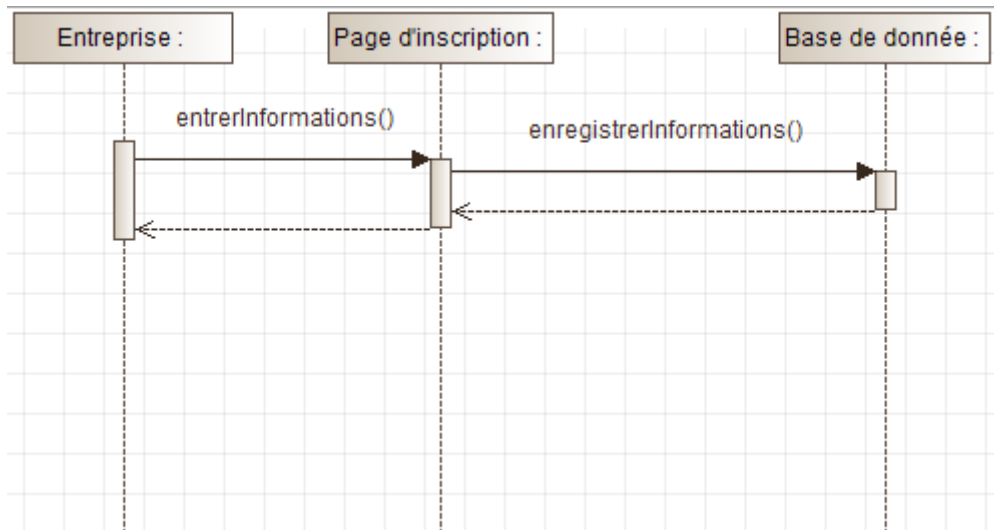
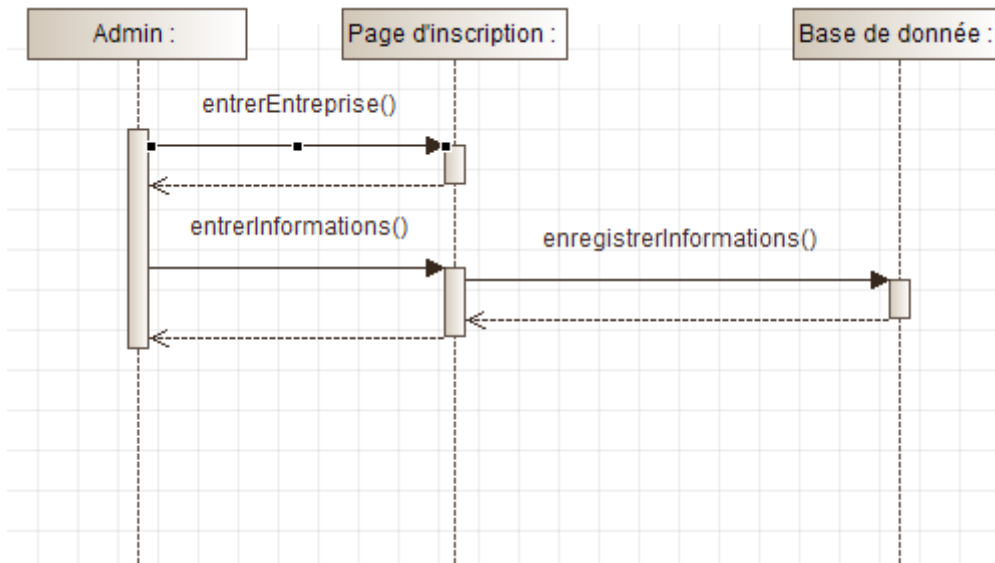
- Supprimer une entreprise implique aussi la suppression de ses offres et des demandes qui lui sont envoyés

Ajouter une entreprise (Admin/Entreprise):

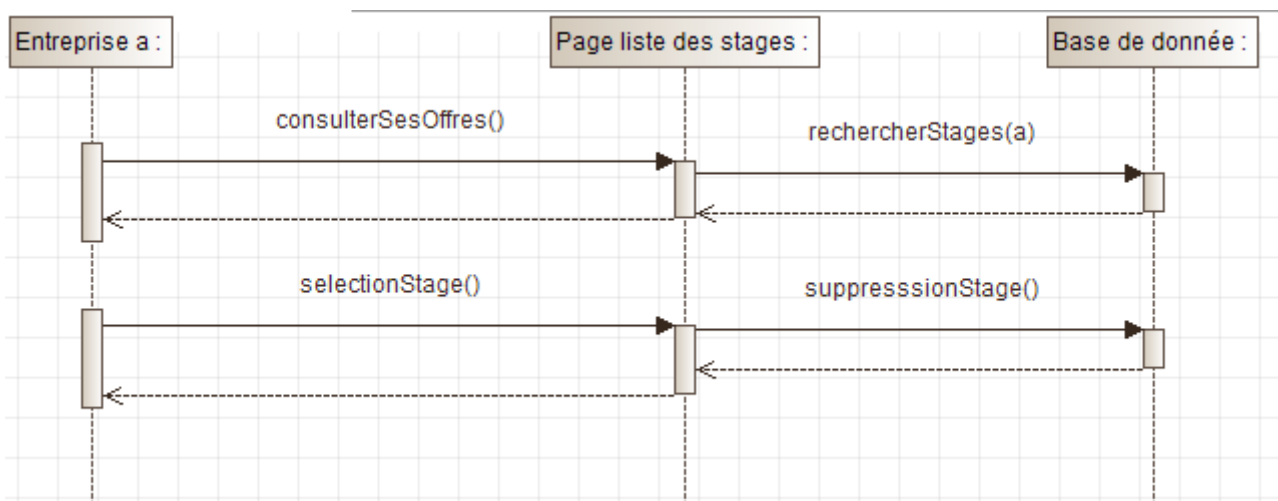
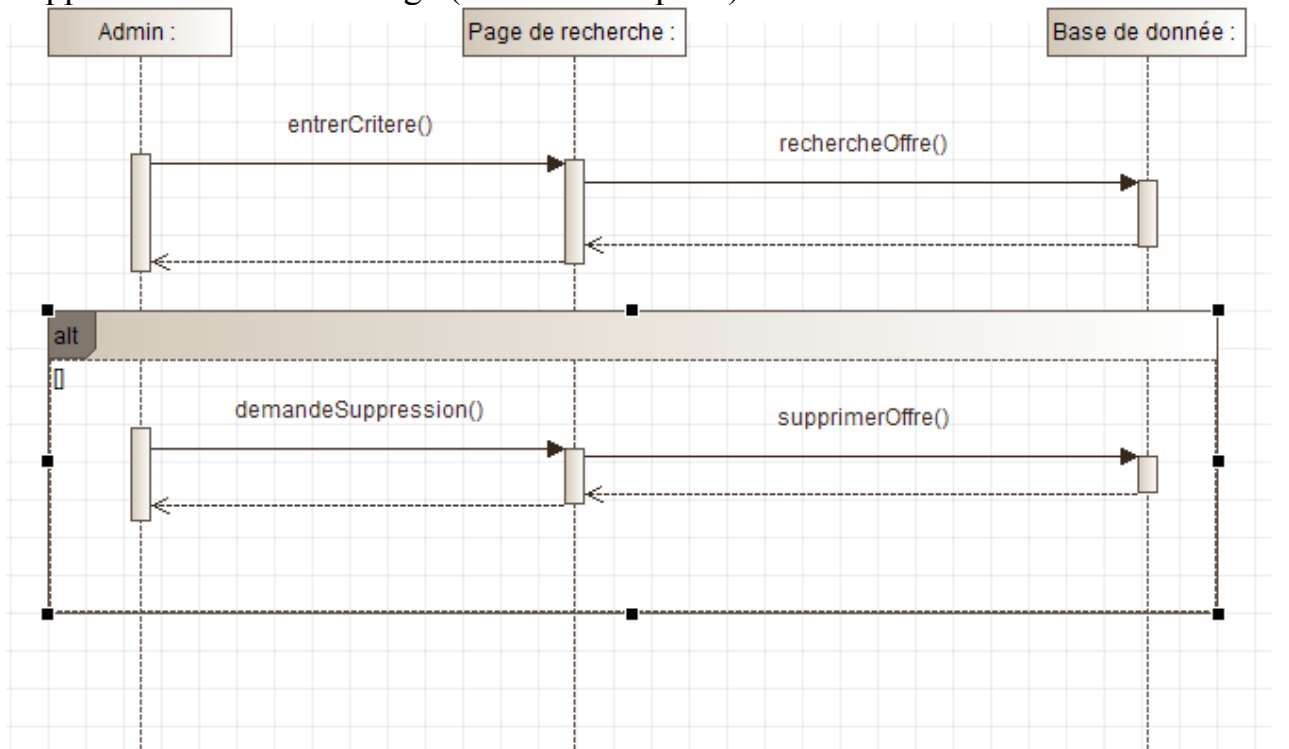




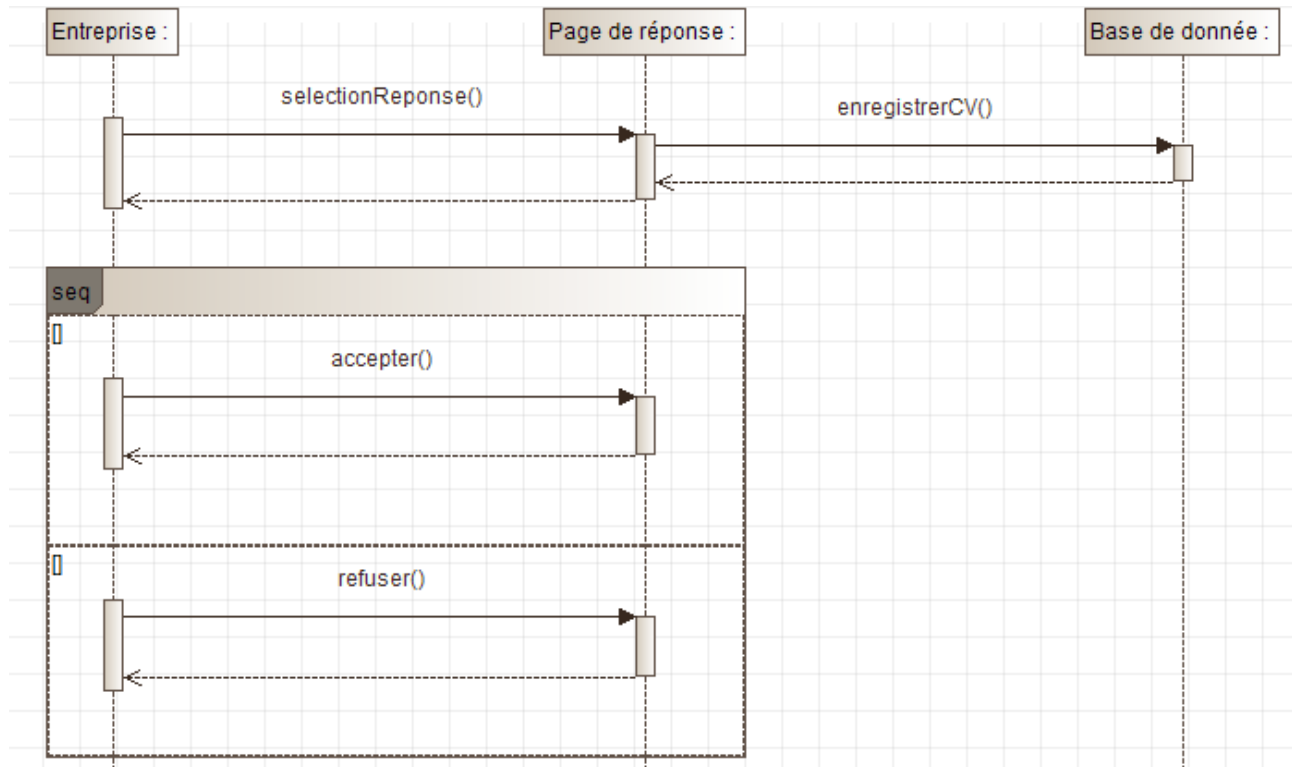
Ajouter une offre de stage (Admin/Entreprise):



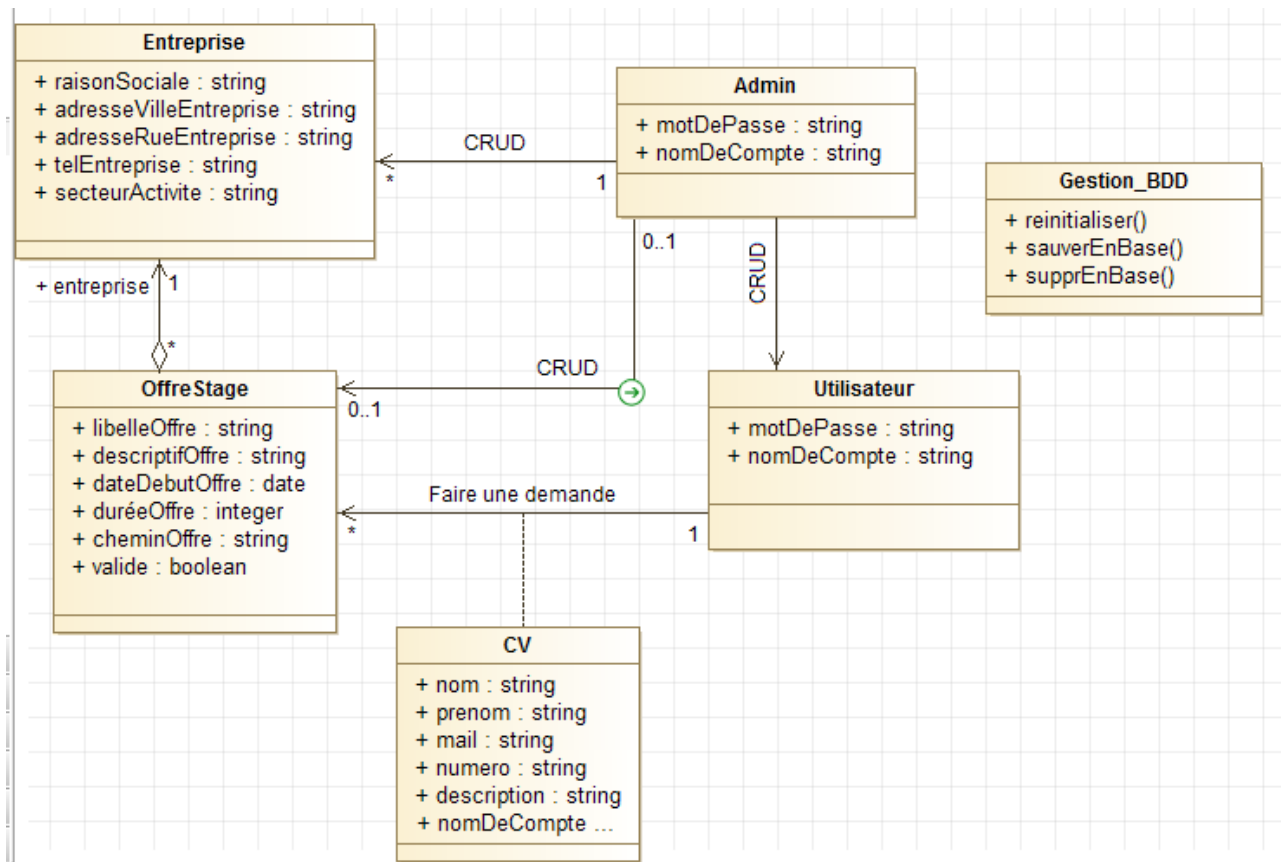
Supprimer une offre de stage (Admin/Entreprise):



## Consulter réponses aux offres (Entreprise):



## Diagramme de classes



Note:

- L'agrégation modélise le fait qu'il y a obligatoirement une entreprise pour chaque offre de stage
- L'envoi d'une demande par l'Utilisateur (l'étudiant) implique l'envoi de son CV
- CRUD veut dire que l'Admin a le droit de Créer, lire, modifier et supprimer les autres instances de classes en question.  
(CRUD=Create,Read,Update,Delete)
- Gestion\_BDD modélise le lien avec la base de donnée (c'est la base du DAO)

## Structure de la base de donnée

### Logiciels utilisés :

- MySQL
- Wampserver (phpmyadmin)

### Structure :

La base de donnée "gestionstages" contient 4 tables:

- entreprise (NomEntreprise, Mdp, Adresse, Telephone)
- offres (NomEntreprise, Duree, Poste, Places)
- postulation (Identifiant, NomEntreprise, Poste, Duree, Etat)
- utilisateur (Identifiant, Mdp, Telephone, Mail, Adresse, Nom, Prenom, Formation, Competence, Experience, Interet)

La table postulation fait le lien entre l'utilisateur et l'offre qu'il accepte

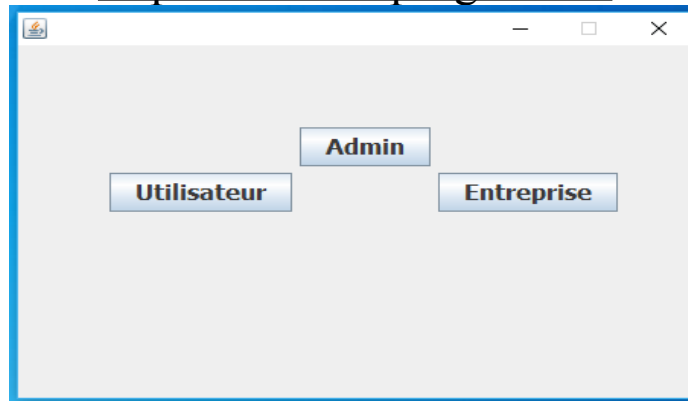
La table utilisateur contient aussi le CV de l'étudiant.

### Importation:

Dans le dossier ressources, il y a deux base de données mise à votre disposition :

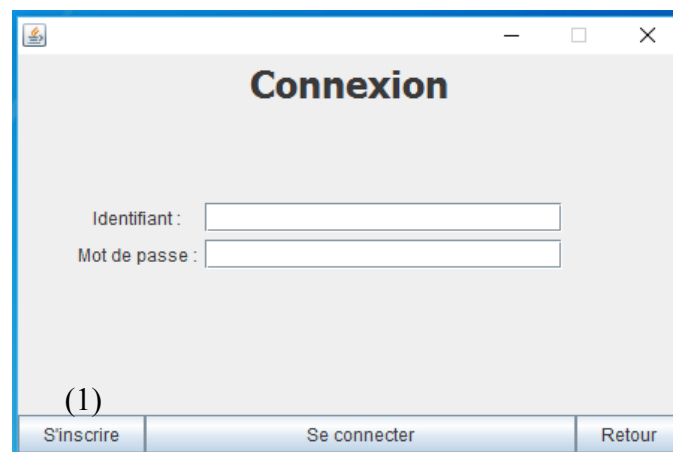
- Une rempli avec quelques exemples (Vous trouverez tout les mots de passes dans le meme dossier que ce rapport).
- Une vide que vous pouvez remplir vous-même ou **directement à l'aide de notre programme de gestion de stages.**

## Explication du programme

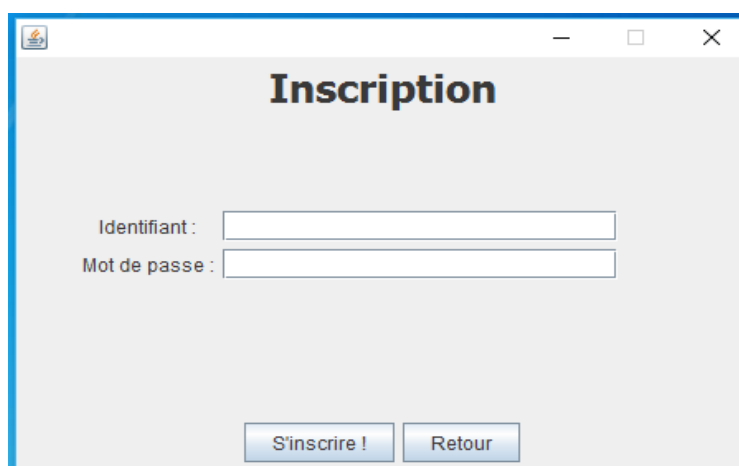


Lors du lancement du programme le premier panel affiché par notre application contient 3 boutons , chacun pour un des différents Utilisateurs du programme

## LE PANNEAU D'UTILISATEUR

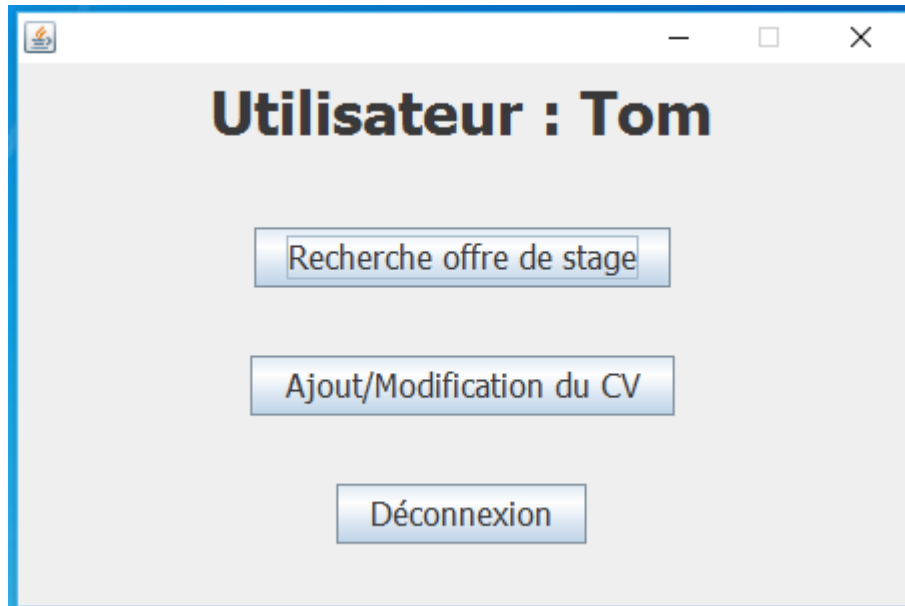


Il est utilisé par le stagiaire potentiel , qui va utiliser ces identifiants et mot de passe pour accéder à son compte dans le cas de la première connexion il va utiliser la fonction pour s'inscrire (1)



La page d'inscription contient les même champs que lors de la connexion c'est à dire Identifiant et mot de passe ;

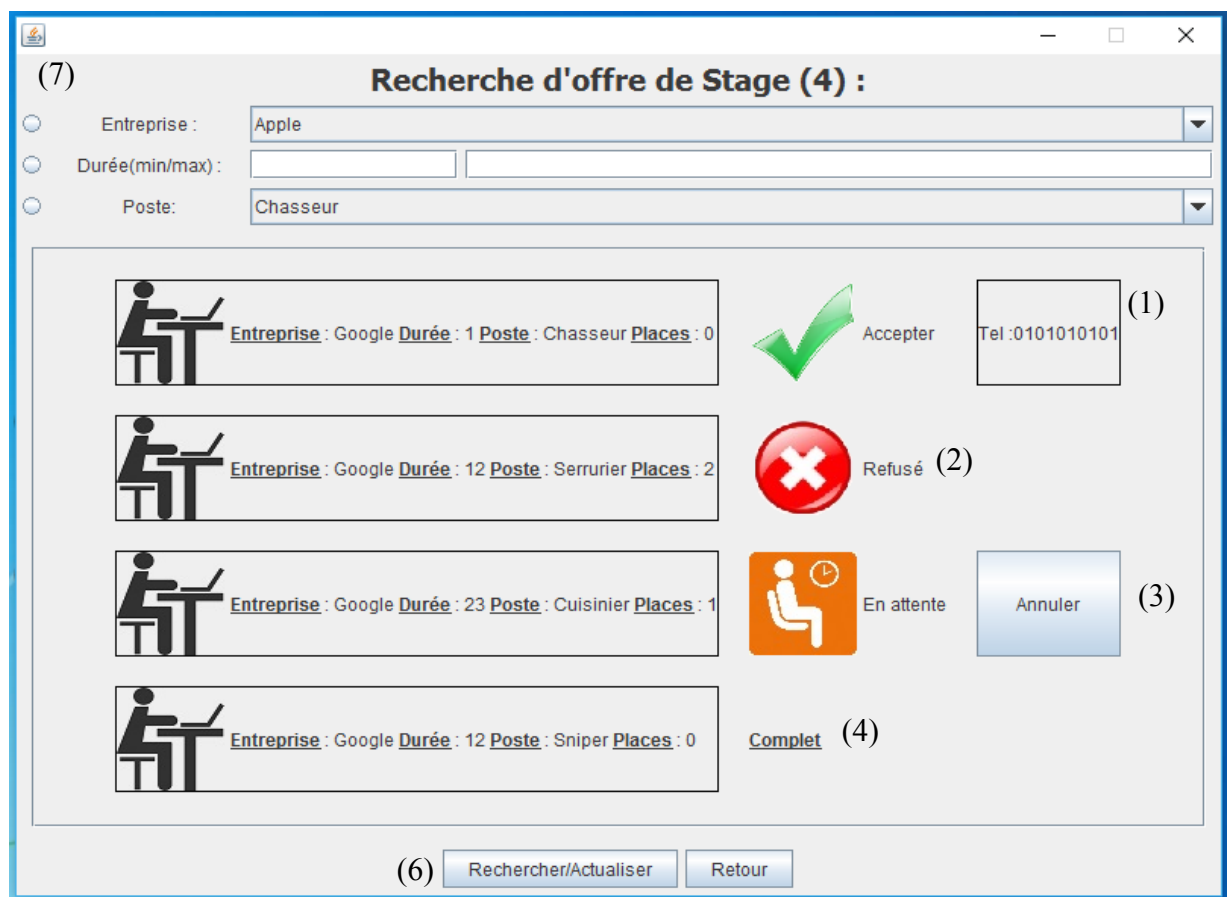
Après l'inscription et la connexion l'utilisateur peut enfin accéder à son panneau d'utilisateur qui lui permettra d'accéder aux différentes offres et de modifier son CV



L'utilisateur a accès à 3 fonctions : La recherche de stage , l'initialisation/modification du CV et enfin la déconnexion.

Chaque bouton modifie le panel actuel de notre application .

### I.Recherche offre de stage :





Entreprise : Google Durée : 23 Poste : Cuisinier Places : 1

Postuler

(5)

Dans la première fonction l'utilisateur a accès à toutes les offres de stages qui ont chacune un des cinq statuts suivant : En attente de réponse(3) .Accepté (1) .Refusé(2),Complet (4) et En attente de postulation (5)

L'utilisateur peut trier ces offres à l'aide des 3 critères en haut de la fenêtre (7) et du bouton de recherche (6).Les 3 critères sont l'entreprise qui propose l'offre , la durée et le poste de l'offre.

Lorsque que l'utilisateur postule il crée une postulation dans la base de données, cette postulation à pour paramètres l'identifiant de l'utilisateur et les paramètres de l'offre sélectionnée.

## **II.Initialisation / Modification du CV:**

**Modification CV :**

Nom : Bruno Prenom : Durand

Téléphone : 0645874741 Adresse : 71 rue de jonville, St-Fargeau Ponthierry

Mail : burno.durand@hotmail.fr

Formation : Bravet des colleges  
CAP menuiserie

Compétences : -Bricoler  
-Decouper  
-Sculter  
-Leadership

Expérience : 2015-Mcdo: premier pas dans le monde du travail

Interets: -Je fais du tennis de table  
-J'aime bricoler des choses en bois et en metal

Valider Retour

Dans cette seconde fonction l'utilisateur peut modifier son CV à sa guise , voir le crée si ce n'est pas déjà fait , lorsque que l'utilisateur à fini ces modifications il peut valider le tout ce qui le renvoie aux Panneaux d'Utilisateur .

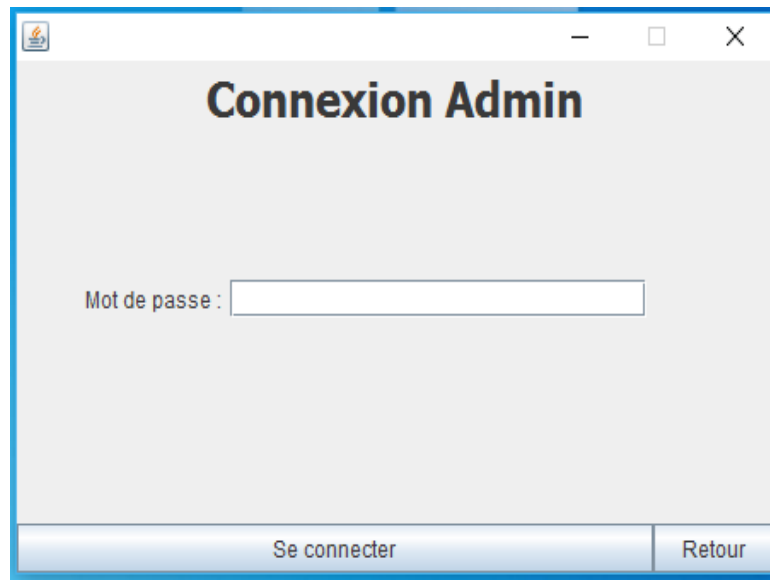
La 3 eme fonction: la déconnexion renvoie l'utilisateur à la page de connexion pour son panneau il est donc obligé de s'identifier à nouveau auprès du programme si il veut ré-accéder à son compte.



# LE PANNEAU D'ADMINISTRATEUR

Le panneau d'administrateur permet de modifier les 3 principales classes du programme c'est à dire Les entreprises , les offres et les utilisateurs.

Pour y accéder il suffit d'entrer le code unique connu par chaque administrateur  
(lire le **Mot de passe .txt**)

A screenshot of a web application window titled "Connexion Admin". It features a light gray background. In the center, there is a label "Mot de passe :" followed by a white text input field. At the bottom, there are two buttons: "Se connecter" on the left and "Retour" on the right, both with a blue gradient and white text.

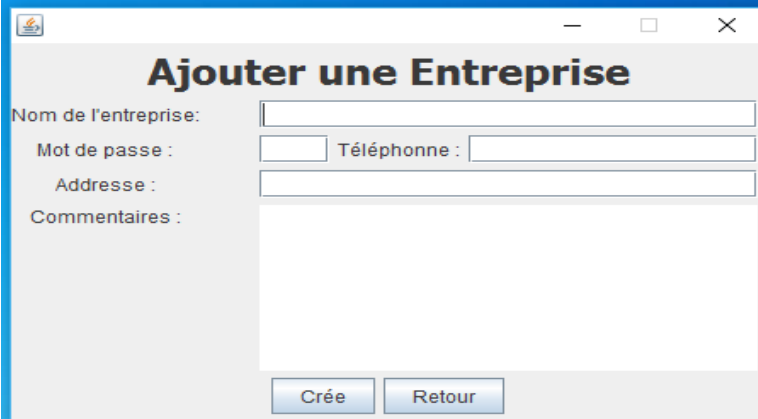
Après avoir entré le mot de passe l'utilisateur se retrouve dans la fenêtre suivante :

A screenshot of the admin dashboard window. It has a light gray background and a blue border. At the top, there are three main sections: "Entreprises", "Offres", and "Utilisateurs". Each section contains two buttons: "Ajouter" and "Consulter", arranged vertically. At the bottom center, there is a "Retour" button. All buttons have a blue gradient and white text.

C'est ici que l'administrateur va gérer directement la base de données à partir des différentes fonctions disponibles sur le panneau d'administrateur.

# 1 ENTREPRISES

## A-Ajouter des entreprises



**Ajouter une Entreprise**

Nom de l'entreprise:

Mot de passe :  Téléphone :

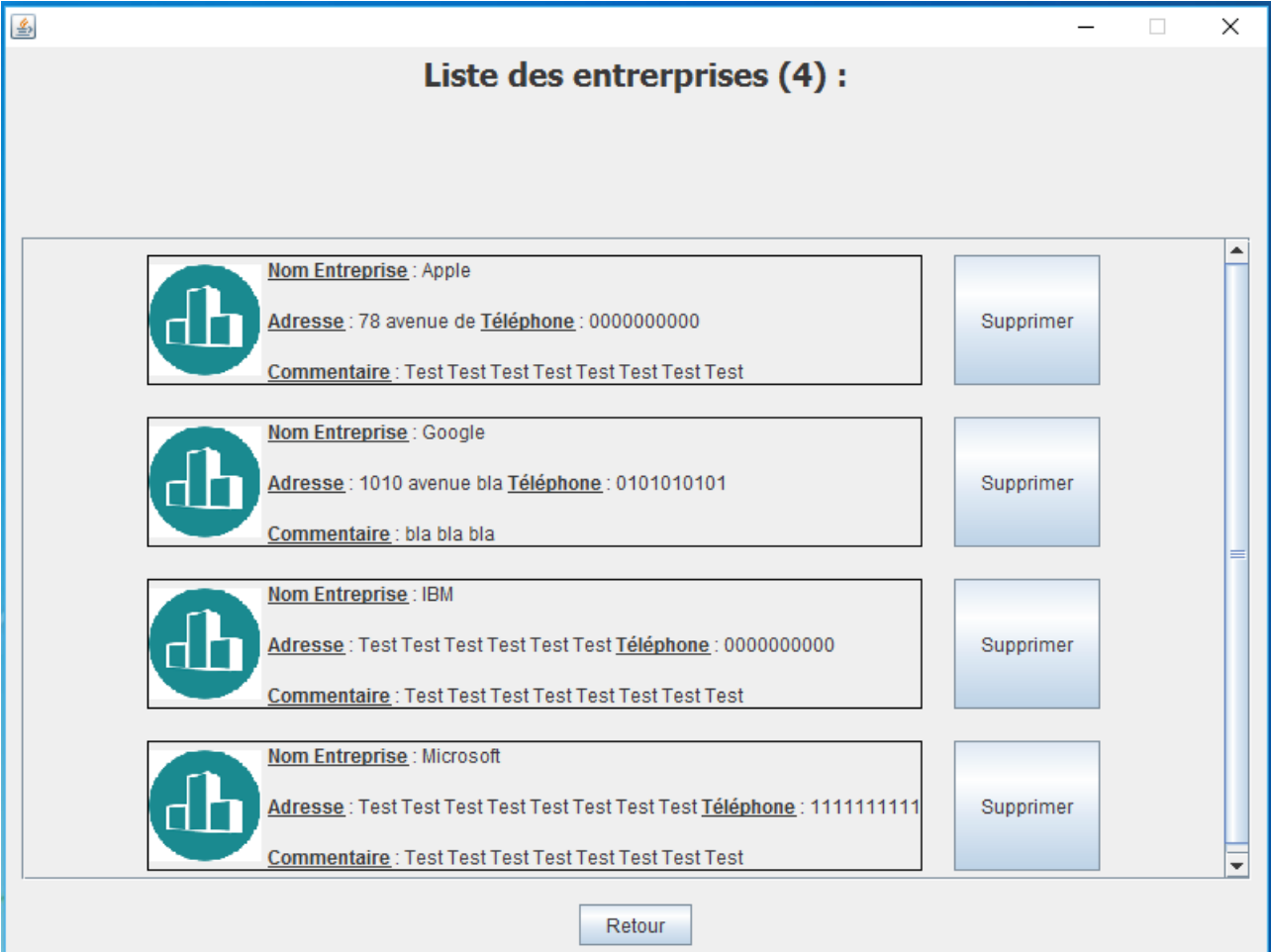
Adresse :

Commentaires :





Dans cette fenêtre l'administrateur inscrit directement une entreprise avec tout les paramètres requis à l'inscription d'une entreprise dans la base de données.

## B-Consulter les entreprises

Dans cette seconde fonction, l'administrateur à accès à toutes les entreprises pressentes dans la base de données et il peut les supprimer. Supprimer une entreprise revient à supprimer les offres qu'elle proposait et aussi les postulations envoyées par les utilisateurs à ces offres.



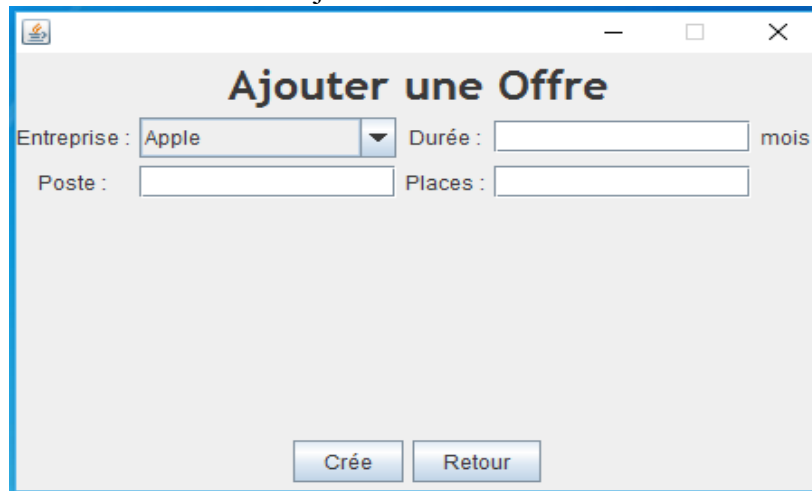
**Liste des entreprises (4) :**

	<b>Nom Entreprise :</b> Apple <b>Adresse :</b> 78 avenue de <b>Téléphone :</b> 0000000000 <b>Commentaire :</b> Test Test Test Test Test Test Test	<input type="button" value="Supprimer"/>
	<b>Nom Entreprise :</b> Google <b>Adresse :</b> 1010 avenue bla <b>Téléphone :</b> 0101010101 <b>Commentaire :</b> bla bla bla	<input type="button" value="Supprimer"/>
	<b>Nom Entreprise :</b> IBM <b>Adresse :</b> Test Test Test Test Test Test <b>Téléphone :</b> 0000000000 <b>Commentaire :</b> Test Test Test Test Test Test Test	<input type="button" value="Supprimer"/>
	<b>Nom Entreprise :</b> Microsoft <b>Adresse :</b> Test Test Test Test Test Test Test Test <b>Téléphone :</b> 1111111111 <b>Commentaire :</b> Test Test Test Test Test Test Test	<input type="button" value="Supprimer"/>

## 2 OFFRES

### A-Ajouter des offres

Dans cette deuxième colonne on s'attaque à l'administration des offres, et pour la troisième fonction du panneau cela concerne leur ajout.



The screenshot shows a window titled "Ajouter une Offre". It contains the following fields: "Entreprise" with a dropdown menu showing "Apple", "Durée" with a text input field followed by "mois", "Poste" with a text input field, and "Places" with a text input field. At the bottom, there are two buttons: "Crée" and "Retour".

Ici, l'administrateur remplit les différents paramètres qui caractérise une offre, son entreprise, sa durée, le poste occupé et enfin le nombre de places.

### A-Supprimer des offres

Dans cette seconde fonction, en ce qui concerne les offres tout comme pour les entreprises on gère ici la suppression des offres



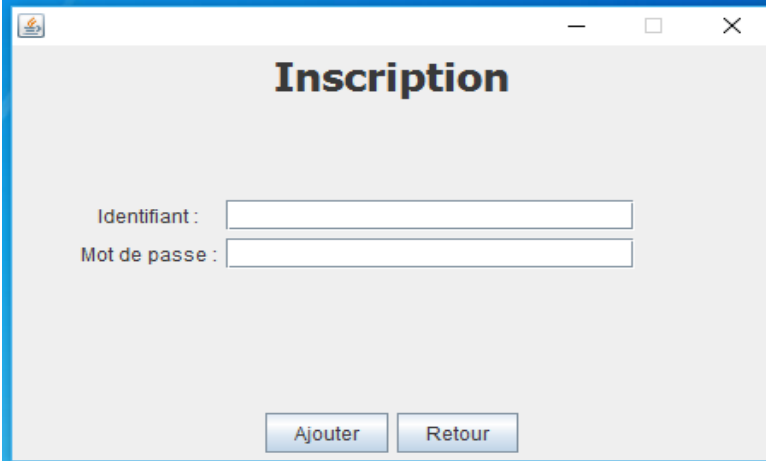
The screenshot shows a window titled "Liste des offres de Stage (2) :". It has three radio buttons on the left, each followed by a label and a dropdown menu: "Entreprise : Apple", "Durée(min/max) : [empty] [empty]", and "Poste : Serrurier". Below these is a large list area containing two entries. Each entry consists of a small icon of a person at a desk, followed by text: "Entreprise : Google", "Durée : 12", "Poste : Serrurier", and "Places : 2" for the first entry, and "Entreprise : Google", "Durée : 12", "Poste : Sniper", and "Places : 0" for the second. To the right of each entry is a "Supprimer" button. At the bottom of the window are two buttons: "Rechercher/Actualiser" and "Retour".

Tout comme l'utilisateur, l'administrateur peut trier ces offres grâce aux trois options de recherche sur le haut de la fenêtre, il pourra ensuite supprimer les offres qu'il souhaite, ce qui entraînera la suppression des postulations qui concerne l'offre choisie.

### **3 Utilisateurs**

#### **A-Ajouter des Utilisateurs**

Enfin la dernière colonne concerne les Utilisateurs, la première fonction dont dispose l'administrateur est d'ajouter un utilisateur comme pour l'inscription il n'y a que deux paramètres à remplir pour exécuter une inscription.



## **B-Supprimer des Utilisateurs**

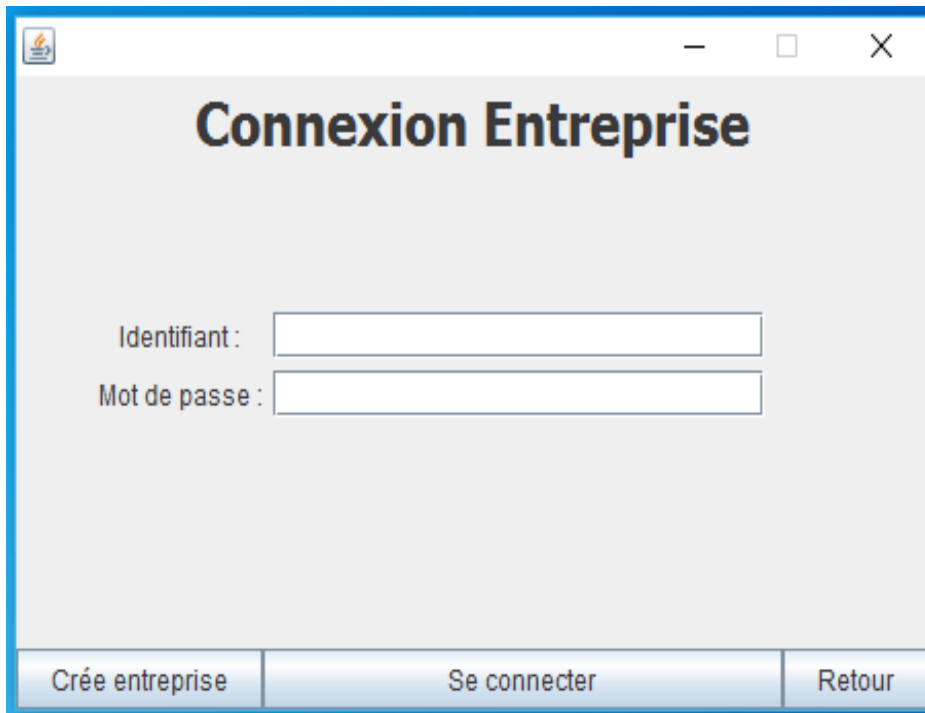
Enfin la dernière fonction concernant les utilisateurs et le panneau d'administration sert à supprimer les utilisateurs. Une liste des utilisateurs est affichée avec les informations dont dispose l'administrateur, chaque utilisateur a un bouton à côté de lui qui enclenche sa suppression. La suppression d'un utilisateur entraîne seulement la suppression de ses postulations, si il a été accepté, la place qu'il a prise est redonné aux offres où il était accepté.



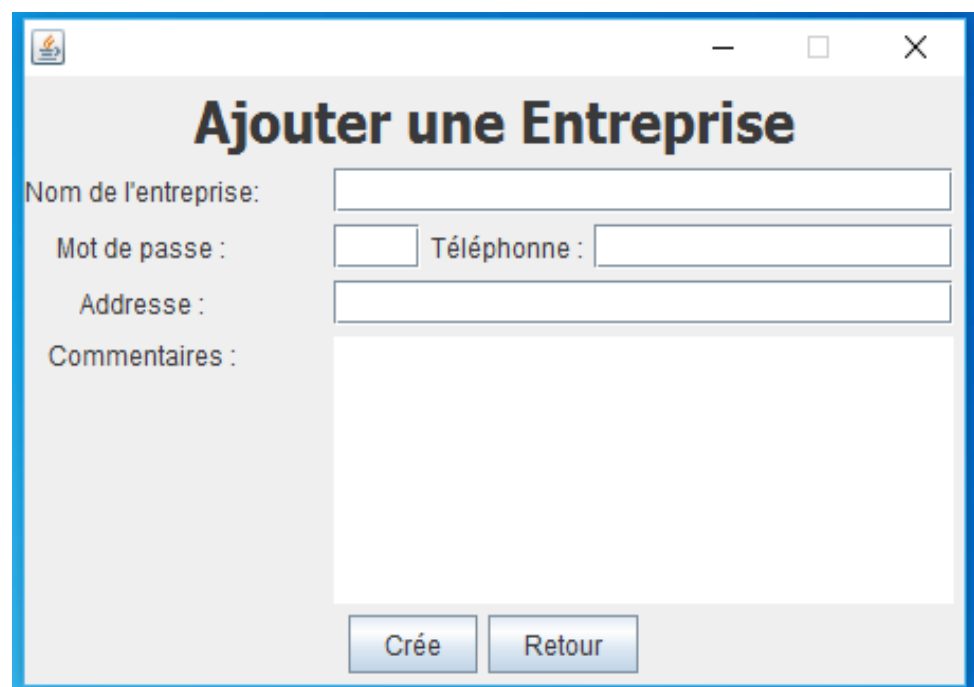
## LE PANNEAU DES ENTREPRISES

Enfin le dernier panneau concerne les entreprises, c'est dans cette partie du programmes que les entreprises gère leur offres et les candidats qui postulent.

Tout comme l'utilisateur lorsque l'entreprise se dirige vers son panneau et il peut soit s'identifier et y accéder dans le cas ou il est déjà inscrit dans la base de données soit s'inscrire avant de s'identifier.

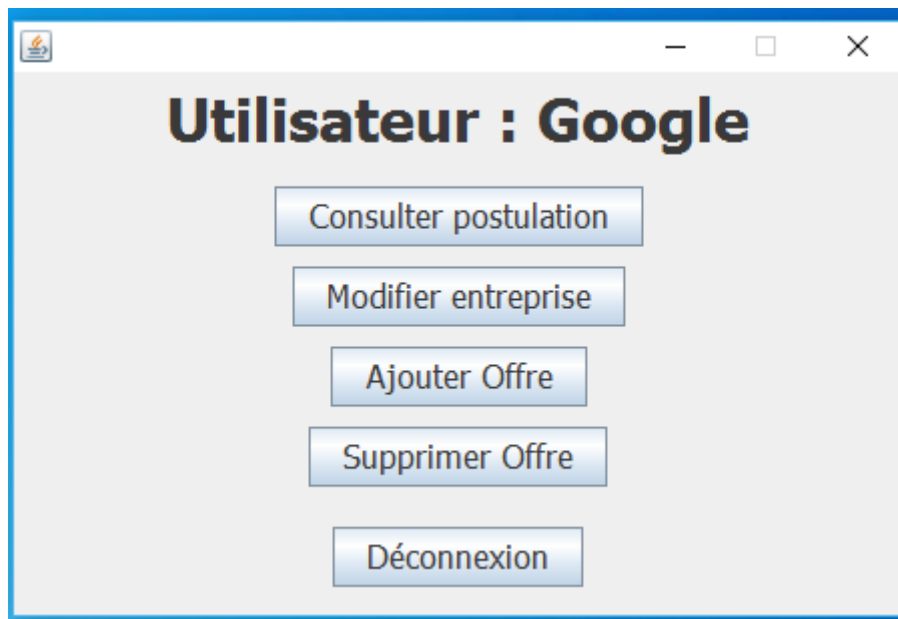


The screenshot shows a window titled "Connexion Entreprise". It has a standard Windows window frame with a minimize button, a maximize button, and a close button. The window contains two input fields: "Identifiant :" and "Mot de passe :". At the bottom, there are three buttons: "Crée entreprise", "Se connecter", and "Retour".



The screenshot shows a window titled "Ajouter une Entreprise". It has a standard Windows window frame with a minimize button, a maximize button, and a close button. The window contains four input fields: "Nom de l'entreprise:", "Mot de passe :", "Téléphone :", and "Adresse :". There is also a larger text area for "Commentaires :". At the bottom, there are two buttons: "Crée" and "Retour".

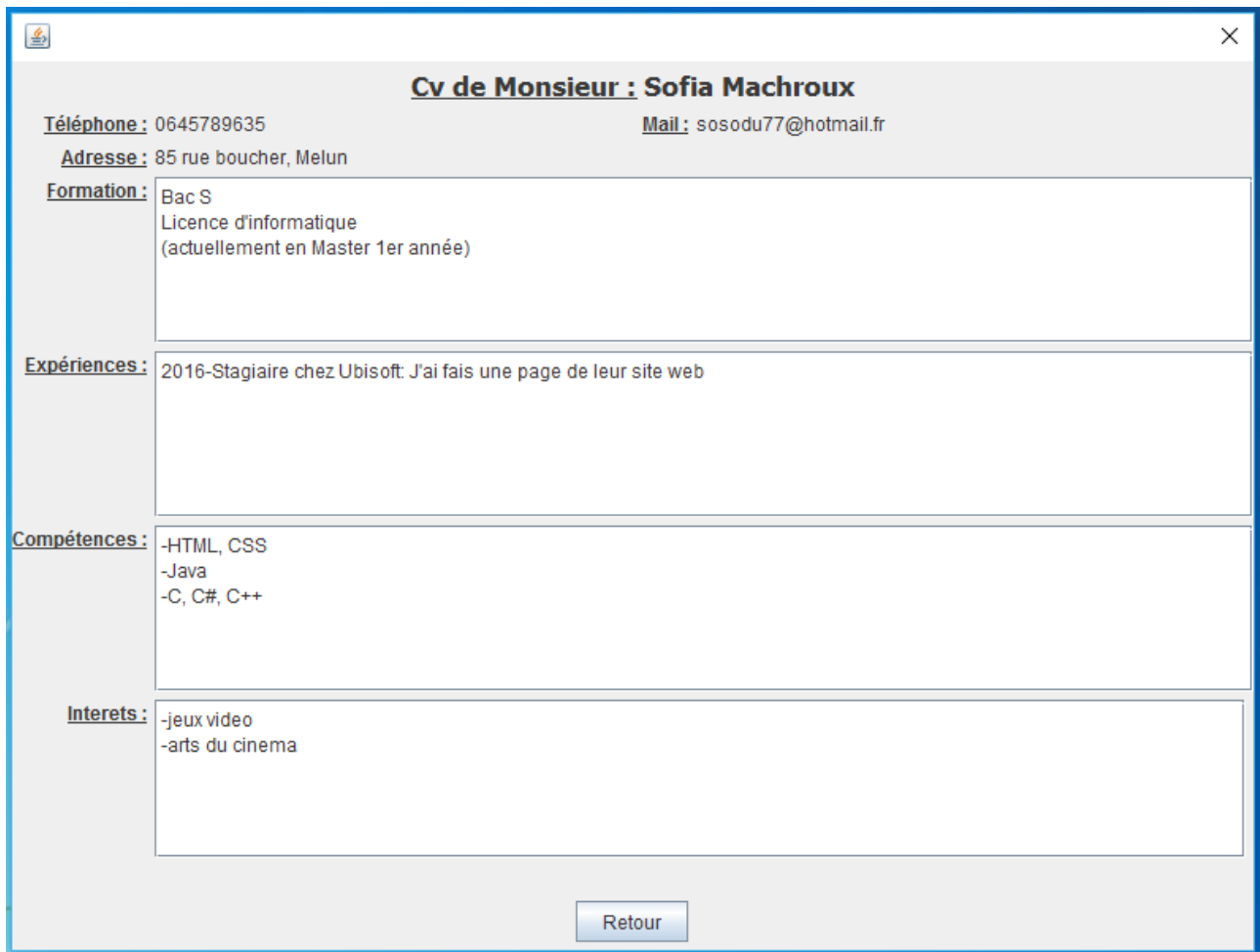
Après s'être identifié, l'entreprise accède à son panneau pour gérer les demandes qu'elle reçoit et en proposer ou supprimé d'autres. L'entreprise a accès à 4 fonction sans compter la déconnexion la première concerne les postulations envoyées pour l'entreprise, la seconde fonction permet de modifier certains paramètres de son entreprise et enfin les deux dernières permettent de gérer les offres proposées par l'entreprise.



## 1 CONSULTER POSTULATION

Lorsque l'entreprise accède aux postulations, trois options s'offre à elle, voir le CV du candidat, Accepter la candidature ou la Refusé. Le gérant du compte de l'entreprise peut revenir sur sa décision et accepté quelqu'un après l'avoir refusé et vice versa.

Lorsque le gerant de l'entreprise tente d'accéder au CV de l'un des candidats, une fenetre s'ouvre avec le profil détaillé du candidat, voir si dessous.



**Cv de Monsieur : Sofia Machroux**

**Téléphone :** 0645789635      **Mail :** sosodu77@hotmail.fr

**Adresse :** 85 rue boucher, Melun

**Formation :** Bac S  
Licence d'informatique  
(actuellement en Master 1er année)

**Expériences :** 2016-Stagiaire chez Ubisoft. J'ai fait une page de leur site web

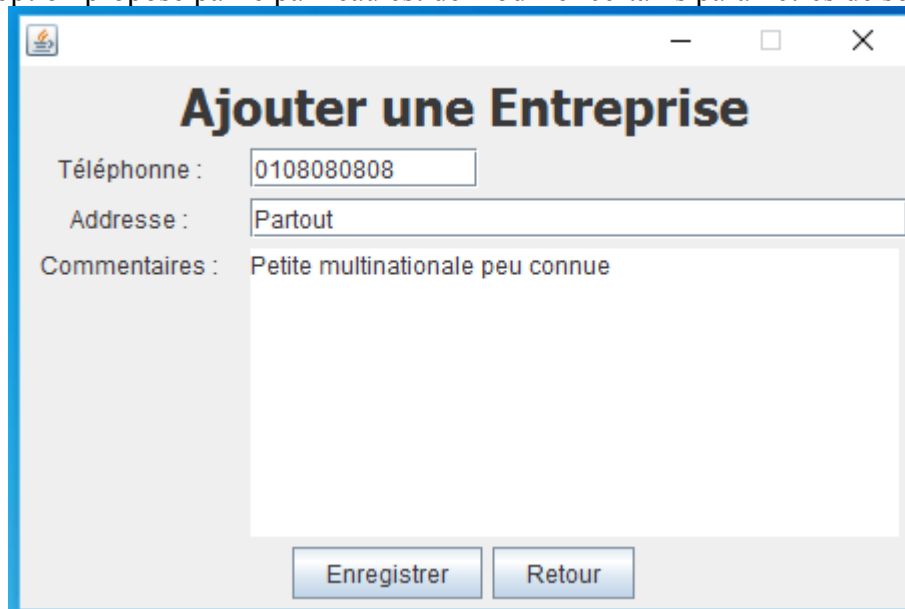
**Compétences :** -HTML, CSS  
-Java  
-C, C#, C++

**Interets :** -jeux video  
-arts du cinema

Retour

## 2 MODIFIER ENTREPRISE

La deuxième option proposé par le panneau est de modifier certains paramètres de son entreprise.



**Ajouter une Entreprise**

**Téléphone :** 0108080808

**Adresse :** Partout

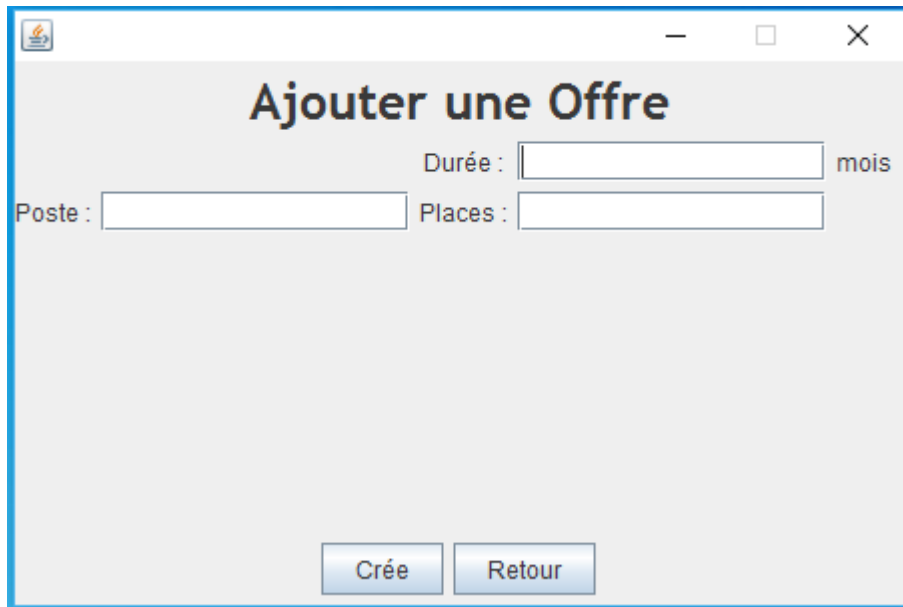
**Commentaires :** Petite multinationale peu connue

Enregistrer    Retour

Les valeurs actuels de l'entreprise sont pré-rentrées dans les champs qui permettent la modification, lorsque les modifications sont faites on valide avec le bouton Enregistrer.



### 3 AJOUTER OFFRE

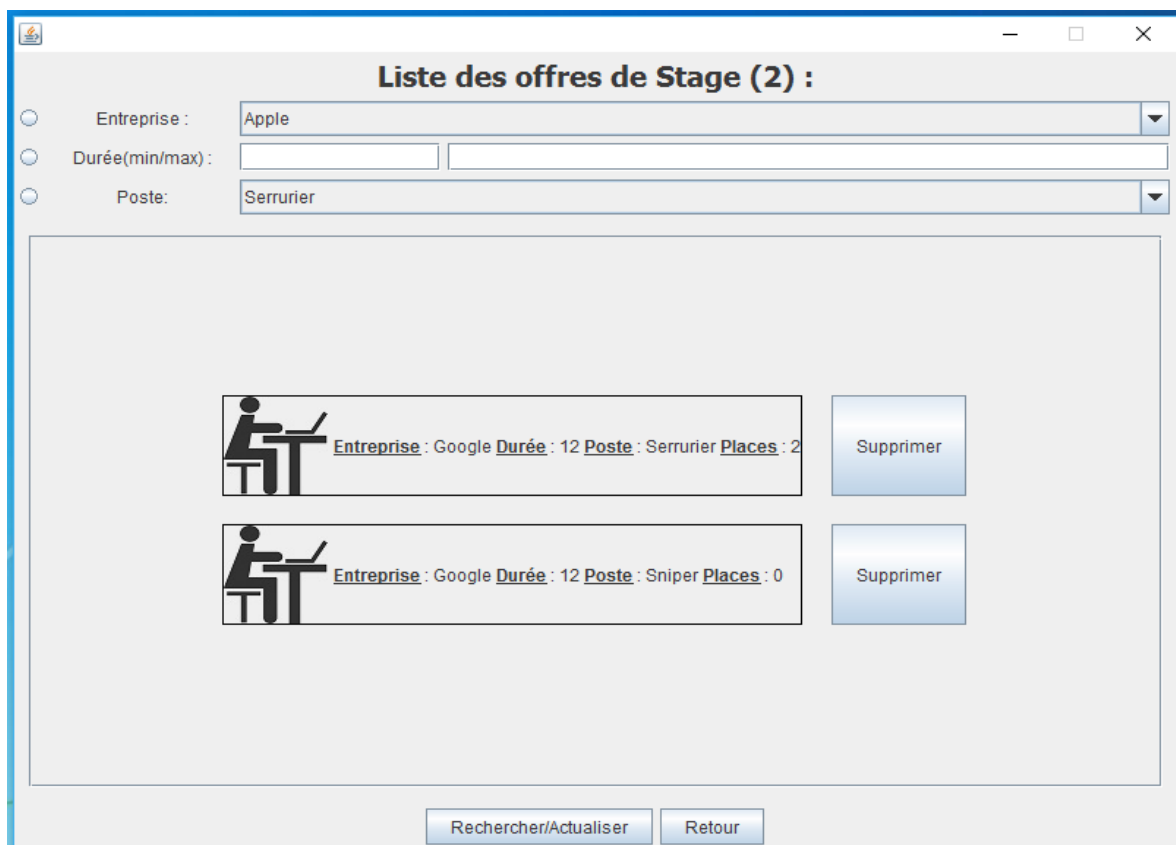


A screenshot of a web application window titled "Ajouter une Offre". The window has a light gray background and a blue border. At the top, there is a title bar with standard window controls (minimize, maximize, close). Below the title bar, the title "Ajouter une Offre" is centered in a large, bold, black font. Underneath the title, there are three input fields: "Durée : [ ] mois" on the right, "Poste : [ ]" on the left, and "Places : [ ]" on the right. At the bottom of the window, there are two buttons: "Crée" and "Retour".

Lorsque qu'une entreprise veut ajouter une offre à ses offres disponibles, la fenêtre est la même que dans le panneau d'administration sauf le fait que l'entreprise ne peut pas être modifiée c'est l'entreprise actuellement connecté qui va gérer cette offre sinon une entreprise pourrait créer une offre pour une autre entreprise et ce n'est pas logique.

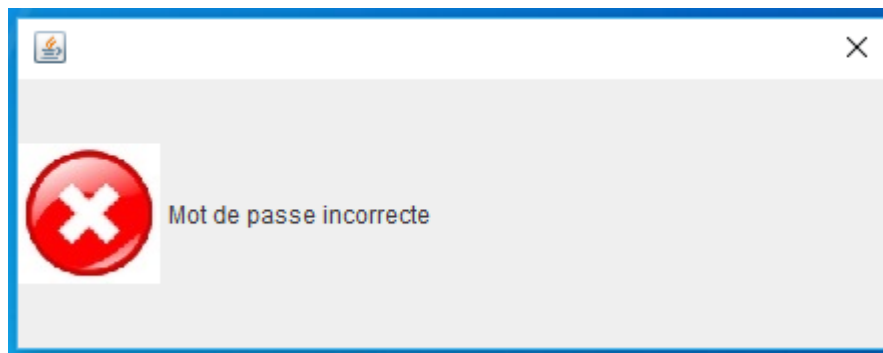
### 4 SUPPRIMER OFFRE

Enfin la dernière fonction du panneau d'entreprise permet de supprimer ses propres offres. Comme depuis le début du programme lorsque l'on affiche des offres elles peuvent être trier avec les options. La suppression d'une offre entraîne la suppression de ces postulations.



A screenshot of a web application window titled "Liste des offres de Stage (2) : ". The window has a light gray background and a blue border. At the top, there is a title bar with standard window controls (minimize, maximize, close). Below the title bar, the title "Liste des offres de Stage (2) : " is centered in a bold, black font. Underneath the title, there are three search filters: "Entreprise : [Apple] [v]", "Durée(min/max) : [ ] [ ]", and "Poste : [Serrurier] [v]". Below the search filters, there is a large gray rectangular area containing two job offers. Each offer is represented by a small icon of a person sitting at a desk with a laptop, followed by text: "Entreprise : Google", "Durée : 12", "Poste : Serrurier", and "Places : 2" for the first offer, and "Entreprise : Google", "Durée : 12", "Poste : Sniper", and "Places : 0" for the second offer. To the right of each offer is a blue button labeled "Supprimer". At the bottom of the window, there are two buttons: "Rechercher/Actualiser" and "Retour".

Dans notre programme lorsque que l'on entre un paramètre qui n'est pas valide ou un identifiant/mot de passe incorrect une fenêtre d'erreur comme celle ci-dessous, avec un message personnalisé en fonction de la situation



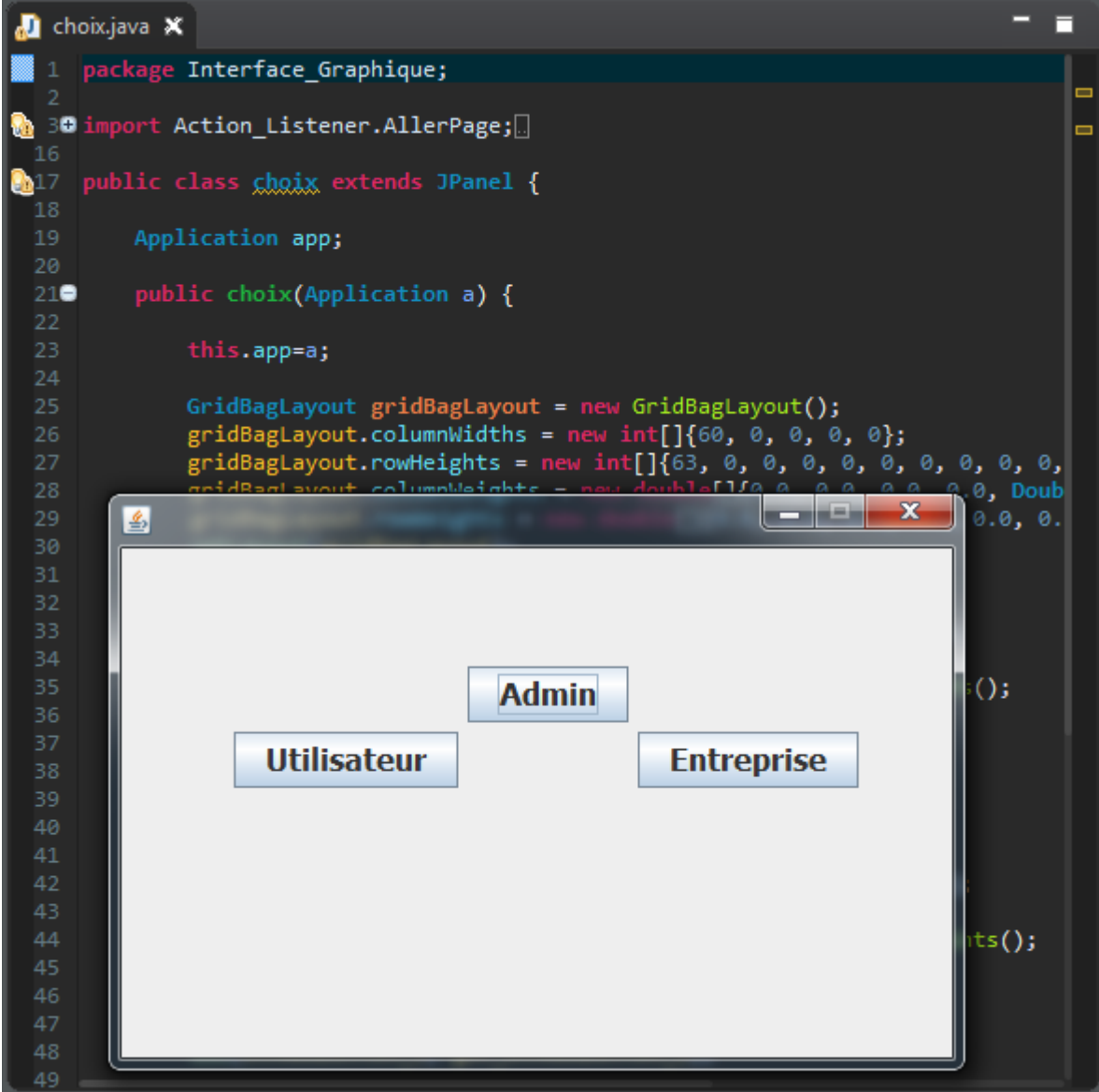
## Explication du Code

```
1 package Principal;
2
3 import javax.swing.UIManager;
4
5 public class main {
6
7     public static void main(String[] args)
8     {
9         UIManager.put("swing.boldMetal", Boolean.FALSE);
10        Application application = new Application();
11    }
12 }
13
14
15 }
```

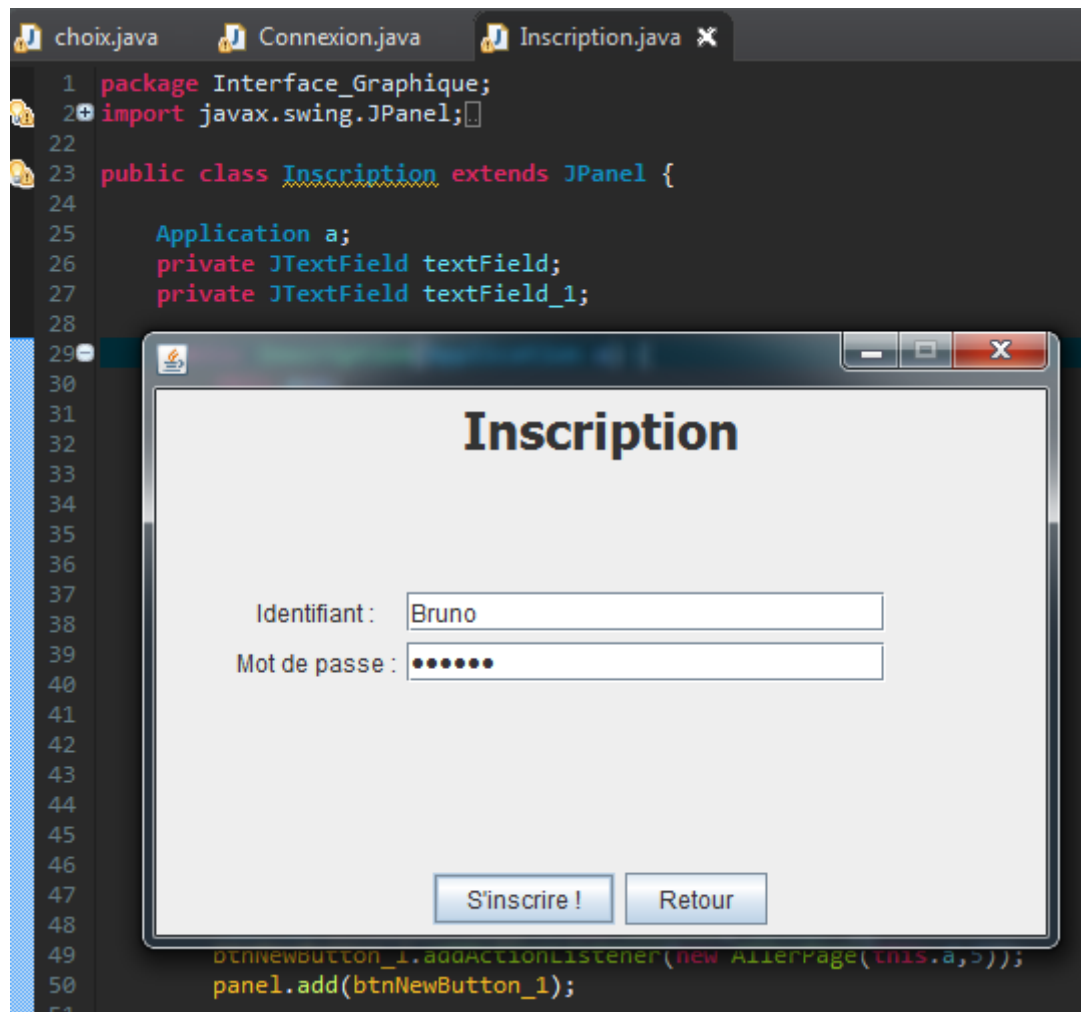
Le main ne fait que crée l'application.

```
1 package Principal;
2
3 import java.awt.BorderLayout;
4
5
6
7
8
9
10
11 public class Application extends JFrame {
12
13     choix choix = new choix();
14
15
16
17
18     public Application() {
19         this.setResizable(false);
20         this.setVisible(true);
21
22
23         choix = new choix(this);
24         setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
25         setBounds(100, 100, 450, 300);
26         setContentPane(choix);
27     }
28
29
30
31 }
```

Et l'application crée son interface graphique avec tout les paramètre de la fenêtre qui vont avec.



On ne va pas entrer dans les détails du code qui se trouve derrière l'interface graphique et expliquer comment fonctionne Java Swing, ce serait long et très répétitif.



Ce qui est plus intéressant c'est ce qu'il se passe lorsque l'on appui sur le bouton "s'incrيره" par exemple.

La classe Inscription qui correspond à la page ci-dessus va faire appelle à la classe AjouterU ci-dessous.

```

1 package Action_Listener;
2 import java.awt.event.ActionEvent;
13
14 public class AjouterU implements ActionListener {
15
16     Application a ;
17     String user ;
18     String mdp ;
19     int z ;
20     Inscription b;
21
22     AjouterUtilisateur c;
23
24     public AjouterU(Application a , Inscription b)
25     {
26         this.a=a;
27         this.b=b;
28     }
29
30     public AjouterU(Application a , AjouterUtilisateur c)
31     {
32         this.a=a;
33         this.c=c;
34         this.b=null;
35     }
36
37     @Override
38     public void actionPerformed(ActionEvent arg0) {
39
40
41         //difference entre admin et utilisateur
42         if(b!=null){
43             this.user=b.getTextField().getText();
44             this.mdp=b.getTextField_1().getText();
45         }
46         else{
47             this.user=c.getTextField().getText();
48             this.mdp=c.getTextField_1().getText();
49         }

```

Cette classe peut être constituée de deux façons (il y a deux constructeurs).

Les constructeurs prennent en attribut l'Application comme toutes les autres classes pour la garder et travailler sur la même fenêtre.

Le deuxième attribut c'est la page, il y a un deuxième constructeur parce que l'admin fait aussi appel à cette classe pour créer des utilisateurs, et donc deux pages différentes pouvant y faire appel, ainsi on peut faire la différence entre la page de l'admin et celle de l'utilisateur.

```

38 public void actionPerformed(ActionEvent arg0) {
39
40
41     //difference entre admin et utilisateur
42     if(b!=null){
43         this.user=b.getTextField().getText();
44         this.mdp=b.getTextField_1().getText();
45     }
46     else{
47         this.user=c.getTextField().getText();
48         this.mdp=c.getTextField_1().getText();
49     }
50     //cryptage du mot de passe
51     if(DAO.dispoU(user)){
52         Erreur erreur=new Erreur("Ce nom d'utilisateur n'est pas disponible");
53     }
54     else if(user.equals("")){
55         Erreur erreur=new Erreur("Veuillez entrer un nom d'utilisateur");
56     }
57     else if(mdp.equals("")){
58         Erreur erreur=new Erreur("Veuillez entrer un mot de passe");
59     }
60     else{
61
62         Md5 criptage=new Md5(mdp);
63         mdp=criptage.getCode();
64         DAO.Inscrire(user, mdp);
65
66         //difference entre admin et utilisateur
67         if(b!=null){
68             a.setContentPane(new Connexion(this.a));
69         }
70         else{
71             a.setContentPane(new AjouterUtilisateur(this.a));
72         }
73
74         a.repaint();
75         a.revalidate();
76     }
}

```

Pour commencer, l'actionPerformed récupère ce qu'il y a écrit sur la page (que ce soit utilisateur ou admin) dans la case "identifiant" et "Mot de passe".

Ensuite on teste si le nom d'utilisateur est disponible à l'aide de la fonction "dispoU()" qui vient du DAO. (on en reparle un peu plus loin)

Si l'utilisateur n'est pas disponible, on affiche un message d'erreur.  
On affiche d'autres messages d'erreur si un des deux champs est vide.

Si l'identifiant et le mot de passe sont en règles, on crypte le mot de passe avec la classe Md5 (dont on parle aussi plus tard), puis on va l'inscrire dans la base de données avec une autre fonction du DAO "inscrire()".

Et pour finir, on revient sur la page de connexion si l'on vient du côté utilisateur ou sur une autre page d'ajout d'utilisateur si l'on vient du côté admin.

```

public static boolean dispoU(String user) {

    // Information d'accès à la base de données
    String url = "jdbc:mysql://localhost/gestionstages?useSSL=false";
    String login = "root";
    String passwd = "";
    Connection cn = null;
    Statement st = null;
    ResultSet rs = null;
    boolean existe = false;
    try {
        // Etape 1 : Chargement du driver
        Class.forName("com.mysql.jdbc.Driver");
        // Etape 2 : récupération de la connexion
        cn = DriverManager.getConnection(url, login, passwd);
        // Etape 3 : Création d'un statement
        st = cn.createStatement();
        String sql = "SELECT `Identifiant`, `Mdp`, `Telephone`, `Mail`, `Adresse`, `Nom`, `Prenom`, `Formation`, "
            + "`Competences`, `Experience`, `Interets` FROM `Utilisateur` WHERE `Identifiant`='" + user + "'";
        // Etape 4 : exécution requête
        rs = st.executeQuery(sql);
        // Si récun données alors étapes 5 (parcours ResultSet)
        while (rs.next()) {
            if(rs.getString("Identifiant").equals(""))
            {
                existe=false;
            }
            else
            {
                if(rs.getString("Identifiant").equals(user))
                {
                    existe = true;
                }
            }
        }
    } catch (SQLException e) {
        e.printStackTrace();
    } catch (ClassNotFoundException e) {

```

DispoU() est un boolean qui prend l'identifiant en attribut et renvoie **false** si il est disponible et **true** sinon. (c'est un peu contre-intuitif, on a mal choisi le nom de la methode)

Pour cela, on ouvre la base de données "gestionstages" qui doit se trouver sur le serveur local (localhost) sur la session root.

Une fois connecter, on entre une commande SQL qui selectionne toutes les lignes du tableau "Utilisateur" où l'identifiant est identique à celui qui est pris en attribut.

Si il n'y a en a pas, cela veux dire que l'identifiant est dispoinible, donc on renvoie **false**. Sinon, l'identifiant est disponible et on renvoie **true**.

*Remarque: La plupart des autres méthode du DOA qui fonctionne de la même façon avec une requête sql sont nombreuses, donc on n'en reparlera pas beaucoup par la suite.*



```

18 public static void Inscrire(String user,String mdp) {
19     String url = "jdbc:mysql://localhost/gestionstages?useSSL=false";
20     String login = "root";
21     String passwd = "";
22     Connection cn =null;
23     Statement st =null;
24     try{
25         Class.forName("com.mysql.jdbc.Driver");
26         cn = DriverManager.getConnection(url,login,passwd);
27         st= cn.createStatement();
28
29
30         String sql = "INSERT INTO `Utilisateur` (`Identifiant`, `Mdp`, `Telephone`, `Mail`, `Adresse`, `Nom`, "
31             + " `Prenom`, `Formation`, `Competences`, `Experience`, `Interets`)"
32             + " VALUES ('"+user+"', '"+mdp+"', '', '', '', '', '', '', '', '', '')";
33
34         st.executeUpdate(sql);
35     }
36     catch (SQLException e){
37         e.printStackTrace();
38     }
39     catch (ClassNotFoundException e){
40         e.printStackTrace();
41     }
42     finally {
43         try {
44             cn.close();
45             st.close();
46         }
47         catch (SQLException e)
48         {
49             e.printStackTrace();
50         }
51     }
52 }
53 }

```

Cette methode fonction légèrement diffèrent de la méthode dispoU()

La méthode Inscrire() ne retourne rien mais enregistre une nouvelle ligne dans la base de données avec une commande sql.

```

public ConnexionU(Application a , Connexion c)
{
    this.a=a;
    this.c=c;
}

public void actionPerformed(ActionEvent arg0) {
    // TODO Auto-generated method stub

    this.user=c.getTextField().getText();
    this.mdp=c.getTextField_1().getText();
    //cryptage du mot de passe

    if(this.user.equals("")||this.mdp.equals(""))
    {
        Erreur mdpInconnu= new Erreur("Mot de passe incorrect ou nom d'utilisateur inconnu");
    }
    else
    {
        Md5 cryptage=new Md5(mdp);
        mdp=cryptage.getCode();
        if(DAO.connexionU(user,mdp))
        {
            a.setContentPane(new PanneauUser(this.a,user));
            a.repaint();
            a.revalidate();
        }
        else{
            Erreur mdpInconnu= new Erreur("Mot de passe incorrect ou nom d'utilisateur inconnu");
        }
    }
}

```

Une fois que l'utilisateur est crée et que l'on veut se connecter sur la page Connexion, on utilise la classe ConnexionU qui fonctionne à peu de chose près comme AjoutU vu précédemment.

Si tout est en règle on va encore une fois crypter le mot de passe, puis on utilise la methode connexionU() du DAO pour savoir si l'identifiant et le mot de passe correspondent pour ensuite accéder à la page du compte de l'utilisateur.

Sinon, on affiche un message d'erreur.

```

String sql = "SELECT `Identifiant`, `Mdp`, `Telephone`, `Mail`, `Adresse`, `Nom`, `Prenom`, "
+ " `Formation`, `Competences`, `Experience`, `Interets` "
+ "FROM `Utilisateur` "
+ "WHERE `Identifiant`='"+user+"' AND `Mdp`='"+mdp+"'";

// Etape 4 : exécution requête
rs = st.executeQuery(sql);
// Si récup données alors étapes 5 (parcours ResultSet)
while (rs.next()) {
    if(rs.getString("Identifiant").equals(""))
    {
        existe=false;
    }
    else
    {
        if(rs.getString("Identifiant").equals(user)&&rs.getString("Mdp").equals(mdp))
        {
            existe = true;
        }
    }
}

```

Ci-dessus, la methode connexionU() du DAO qui fonction comme dispoU().  
 On va faire une requete sql pour voir si le mot de passe correspond à l'identifiant.  
 Si l'on en trouve pas, on renvoie **false**, si on est trouve on renvoie **true**.

**Remarque:** On vient de voir comment l'on faisait "dans le code" pour s'incrيره (incluant l'enregistrement dans la base de donnée). On procède de la même façon pour ajouter/modifier des utilisateurs, des entreprises et des offres mais avec plus d'attributs. Donc on reparlera pas plus loin.

On s'intéresse maintenant à la classe de cryptage Md5

```
9 public Md5(String md5) {
10     Passe(md5);
11     // TODO Auto-generated constructor stub
12 }
13
14 public void Passe(String pass){
15     byte[] passBytes = pass.getBytes();
16     try {
17         MessageDigest algorithm = MessageDigest.getInstance("MD5");
18         algorithm.reset();
19         algorithm.update(passBytes);
20         MessageDigest md = MessageDigest.getInstance("MD5");
21         byte[] messageDigest = md.digest(passBytes);
22         BigInteger number = new BigInteger(1, messageDigest);
23         this.code= number.toString(16);
24     } catch (NoSuchAlgorithmException e) {
25         throw new Error("invalid JRE: have not 'MD5' impl.", e);
26     }
27 }
28 public String getCode(){
29     return code;
30 }
```

Tout d'abord, cet algorithme ne nous appartient pas (nous l'avons récupéré sur un forum).

Il nous permet de coder les mots de passes avant de les entrer dans la base de données.

Par exemple, sur notre application le mot "azerty" devient "ab4f63f9ac65152575886860dde480a1".

Le code diffère selon l'application, donc on ne peut pas retrouver le mot correspondant à partir du code.

Pour se connecter, on va donc récupérer la chaîne de caractères en entrée, puis la coder avec Md5 et comparer ce code avec celui que l'on a écrit dans la base de données précédemment (on compare grâce une méthode du DAO, comme connexionU() vu précédemment par exemple).

Ci-dessous la classe AllerPage:

```
1 package Action_Listener;
2 import java.awt.Dimension;
30
31 public class AllerPage implements ActionListener {
32
33     private int b;
34     private Application a ;
35     private String user;
36
37     public AllerPage(Application app,int b){
38         // TODO Auto-generated constructor stub
39         this.a=app;
40         this.b=b;
41
42     }
43
44     public AllerPage(Application a2, int i, String user) {
45         this.a=a2;
46         this.b=i;
47         this.user=user;
48     }
49
50     @Override
51     public void actionPerformed(ActionEvent arg0) {
52         a.setPreferredSize(new Dimension(400,300));
53         if(b==1){
54             a.setContentPane(new choix(this.a));
55         }
56         if(b==2){
57             a.setContentPane(new PanneauAdmin(this.a));
58         }
59         if(b==3){
60             a.setContentPane(new AjouterEntreprise(this.a));
61         }
62         if(b==4){
63             a.setContentPane(new AjouterOffre(this.a));
64         }
65         if(b==5){
66             a.setContentPane(new Connexion(this.a));
67     }
```

Cette classe joue le rôle d'intermédiaire entre les différentes pages de l'interface graphique.

Elle a deux constructeurs:

- Un pour les pages "générales"
- Un pour les pages où un utilisateur (ou entreprise) est connecté, on doit alors récupérer son identifiant (user).

Dans cette classe, la méthode actionPerformed reconnaît chaque page selon un numéro b qui lui a été attribué.

Ainsi, pour aller à la page de l'admin PanneauAdmin() en cliquant sur un bouton, on écrit par exemple:

```
JButton btnNewButton_1 = new JButton("Retour");
btnNewButton_1.addActionListener(new AllerPage(this.a,2));
panel.add(btnNewButton_1);
```

```

ListeCv lcv;
boolean e;
boolean d;
boolean p;
GererOffr ge;

boolean admin = false;

public Recherche(ListeCv listeCv) {
    // TODO Auto-generated constructor stub
    this.lcv=listeCv;
}

public Recherche(GererOffr GererOffr) {
    // TODO Auto-generated constructor stub
    this.ge=GererOffr;
    admin=true;
}

```

Une autre classe intéressante à décrire est la classe Recherche() elle est utilisé dans le tri des offres. Les booléens "e", "d" et "p" représente les RadioButton devant Entreprise, Durée et Poste dans les panel qui traitent des offres, il y a deux constructeurs car on utilise cette classe dans deux cas différents dans ce programme: le premier dans le panneau d'utilisateur et le second dans le panneau d'administrateur.

```

this.e=lcv.getRdbtnNewRadioButtonE().isSelected();
this.d=lcv.getRdbtnNewRadioButton_D().isSelected();
this.p=lcv.getRdbtnNewRadioButton_p().isSelected();
ArrayList<Offres> offres = new ArrayList<Offres>();
boolean faire = true;

```

Au tout debut on initialise les trois booleans en fonction de l'etat des RadioButton, on cree une liste d'offres vide pour l'instant qui seras remplie et affichée à la fin, et on crée un boolean "faire" qui nous servira plus tard.

```

if(e)
{
    if(d)
    {
        if(p)
        {
            offres=Rechercheedp();
        }
        else
        {
            offres=Rechercheed();
        }
    }
    else
    {
        if(p)
        {
            offres=Rechercheep();
        }
        else
        {
            offres=Recherchee();
        }
    }
}
else
{

```

En fonction des valeurs de "e", "d" et "p" on rempli la liste d'offres differement. Par exemple ici on voit que quand les trois variables sont vrais, (sous entendu les trois boutons sont cochés) les offres seront donc triés en fonction de l'entreprise, la durée et du poste et donc la liste d'offres qui va etre affichée depends de la methode Rechercheedp().

La methode rechercheedp va appliquer les trois recherches sur trois listes d'offres contenant toute les offres puis ensuite prendre seulement les offres en commun grâce à la methode Commun3()

```
private ArrayList<Offres> Rechercheedp() {  
    // TODO Auto-generated method stub  
    ArrayList<Offres> offresd = Recherched();  
  
    ArrayList<Offres> offresp = Recherchep();  
  
    ArrayList<Offres> offrese = Recherchee();  
  
    ArrayList<Offres> offres = Commun3(offresd,offresp,offrese);  
  
    return offres;  
}
```

Pour la methode Recherchep() on prend une liste de toute les offres et si l'offre ne correspond pas au poste contenu dans le ComboBox en face du RadioButton qui était activé alors on l'enleve de la liste.

```
private ArrayList<Offres> Recherchep() {  
    // TODO Auto-generated method stub  
    ArrayList<Offres> offres = DAO.Liste0();  
    for(int i = 0;i<offres.size();i++)  
    {  
        if(!admin){  
            if(!offres.get(i).poste().equals(this.lcv.getComboBox_1().getSelectedItem().toString()))  
            {  
                offres.remove(i);  
                i--;  
            }  
        }  
        else  
        {  
            if(!offres.get(i).poste().equals(this.ge.getComboBox_1().getSelectedItem().toString()))  
            {  
                offres.remove(i);  
                i--;  
            }  
        }  
    }  
    return offres;  
}
```

Apres avoir appliqué ça sur 3 listes on obtient 3 listes repondant à 3 criteres differents. Lorsque que l'on prend les elements en communs, on obtient une liste repondant aux 3 criteres en même temps.

```

    }
    else
    {
        faire=false;
        this.ge.setNbr(DAO.listeO().size());
        this.ge.getPanel_2().remove(this.ge.getScrollPane());
        this.ge.afficherliste(DAO.listeO());
    }
}

if(faire){
    this.ge.setNbr(offres.size());
    this.ge.getPanel_2().remove(this.ge.getScrollPane());
    this.ge.afficherliste(offres);
}
}

```

La variable "faire" sert dans le cas où aucun des boutons n'est coché. On affiche alors toutes les offres, dans le cas contraire, on affiche la liste d'offres que l'on a obtenu en fonction de la méthode qui est exécutée elle-même en fonction des boutons activés.