

Rapport de Stage

WEB APPLICATION : SYSTEME PACES

Zihao Hua

INSA DE LYON
DEPARTEMENT IF

Sommaire

| | |
|--|----|
| I. Introduction..... | 2 |
| II. Objectif du projet..... | 2 |
| III. Principe d'algorithme..... | 3 |
| IV. Architecture Back-end..... | 5 |
| V. Description de fonctionnalités | 7 |
| VI. Problèmes apparus et Solutions | 11 |
| VII.Conclusion | 12 |

I. Introduction

Dans le cas des affectations des étudiants en première année des études de santé (PACES), les candidats passeront plusieurs concours dans divers secteurs et obtiendront plusieurs résultats de classement différents. Alors que l'affectation de filière devient plus complexe car le rang de classement, autrement dit le critère d'attribution, n'est plus unique. Il faut donc trouver une méthode pour décider de l'affectation des étudiants dans les différentes filières en prenant en compte à la fois les classements des étudiants aux différents concours et leurs vœux entre les différentes filières.

II. Objectif du projet

À l'aide de l'algorithme Gale et Chapley (algorithme de mariage stable) , nous pouvons trouver une solution adéquate pour résoudre ce problème d'affectation multiple. L'objectif de ce projet est donc mettre en oeuvre cet algorithme en langage C++, ensuite développer une application web qui permet aux utilisateurs étudiants de :

- Gérer son compte personnel et mettre à jour sa liste des vœux avant la date de fermeture de la saisie des vœux.
- Consulter son résultat d'affectation après que le calcul se termine

III. Principe d'algorithme

D'après le pseudo code sur le [site wikipedia](https://fr.wikipedia.org/wiki/Algorithme_de_Gale_et_Shapley) de l'algorithme Gale et Shapley :

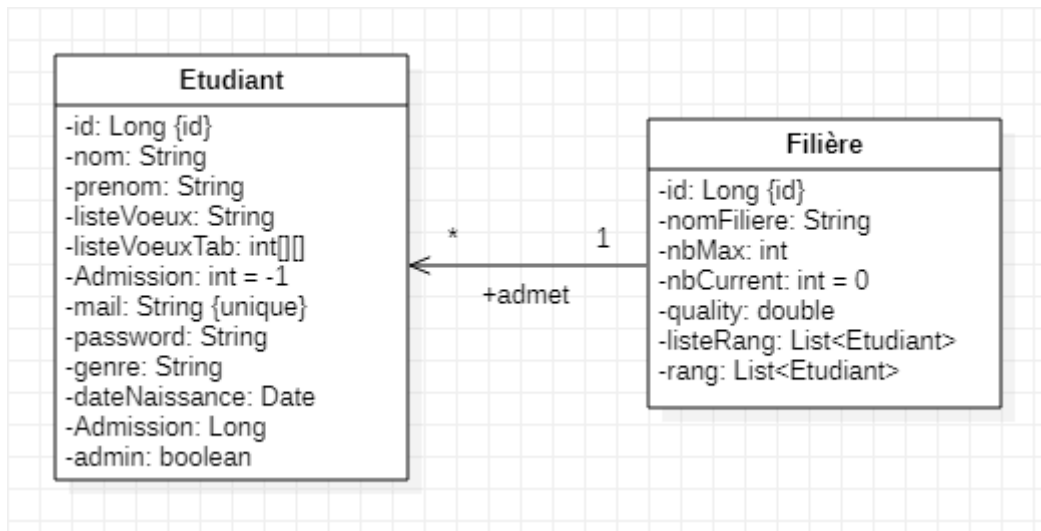
```
Entrée : Deux ensembles finis M (d'hommes) et W (de femmes) de cardinal n ;
        Une famille L de relations de préférences ;
Sortie : Un ensemble S de couples engagés (homme ; femme) ;
fonction mariageStable {
    Initialiser tous les  $m \in M$  et  $w \in W$  à célibataire
    tant que  $\exists$  homme célibataire  $m$  qui peut se proposer à une femme  $w$  {
         $w$  = femme préférée de  $m$  parmi celles à qui il ne s'est pas déjà proposé
        si  $w$  est célibataire
            ( $m, w$ ) forment un couple
        sinon un couple ( $m', w$ ) existe
            si  $w$  préfère  $m$  à  $m'$ 
                ( $m, w$ ) forment un couple
                 $m'$  devient célibataire
            sinon
                ( $m', w$ ) restent en couple
        }
    Retourner l'ensemble S des couples engagés
}
```

Chaque rôle dans le scénario doit posséder une liste de préférence pour commencer l'affectation. Du côté **Etudiant**, il s'agit d'un tableau `int[][]` mémorisant sa préférence (les identifiants des étudiants/filières) par ordre descendante. De plus, leur tableau de vœux est à deux dimension afin d'enregistrer les refus. Du côté **Filière**, la « listeRang » représente le classement de tous les étudiants par ordre décroissante.

Ensuite, pour enregistrer les « couples » formées et les demandes venant de tous les étudiants non-admis, il nous faut deux table de hachage. Ils nous permettent d'enregistrer et rechercher rapidement les couples.

```
1. // HashMap <Etudiant.id, Filiere.id>
2. Map<Integer, Integer> etud = new HashMap<>();
3. // hashmap représentant la situation d'admission
4. Map<Integer, Integer> luvletter = new HashMap<>();
5. // hashmap représentant les demandes des étudiants
```

Mais dans notre cas, une filière correspond à plusieurs étudiants, donc il nous faut encore 2 attributs pour décrire l'état de filière au cours du calcul : nombre d'étudiants maximum et nombre d'étudiants reçus.



Maintenant nous pouvons commencer l'affectation :

- ✧ Etape 1 : Parcourir tous les objets **Etudiant**, s'il n'est pas encore admis, on parcourt sa *listeVoeux* et envoyer la demande à sa filière préférée et disponible (put cette couple dans le hashmap *luvletter*).
- ✧ Etape 2 : Parcourir tous les objets **Filiere**, et vérifier s'il a reçu des demandes.
 Si oui et cette filière n'est pas pleine, on ajoute cet objet **Etudiant** dans le bon endroit du **Filiere.rang**, la liste représentant le rang des étudiants déjà affectés, et on insère en même temps cette couple dans *etud*.
 Si la filière est déjà pleine, on compare la note du dernier élément de **Filiere.rang** et celle de l'étudiant que l'on traite maintenant. Celui possédant de meilleurs notes restera et l'autre sera éliminé.
- ✧ Etape 3 : Une fois qu'un étudiant est éliminé par une certaine filière, son attribut *listeVoeux* sera modifié pour enregistrer ce refus. Cet étudiant ne pourra plus envoyer sa demande à cette filière.
- ✧ Etape 4 : Recommencer l'étape 1 s'il reste encore des étudiants non-admis.

Le résultat d'attribution et les classements de chaque étudiant seront paramétrés à la fin de l'exécution. En même temps, un attribut appelé « **quality** » est initialisé dans chaque entité **Filiere**. Il sert à représenter la qualité des étudiants admis par cette filières, il s'agit d'une valeur entre 0 et 1, calculée par le formule ci-dessous :

$$\frac{Max - Moy}{Max - Min}$$

Où :

- Max : La moyenne de classement dans le pire cas ;
- Min : La moyenne de classement dans le meilleur cas ;
- Moy : La moyenne de classement des étudiants admis ;

Exemple :

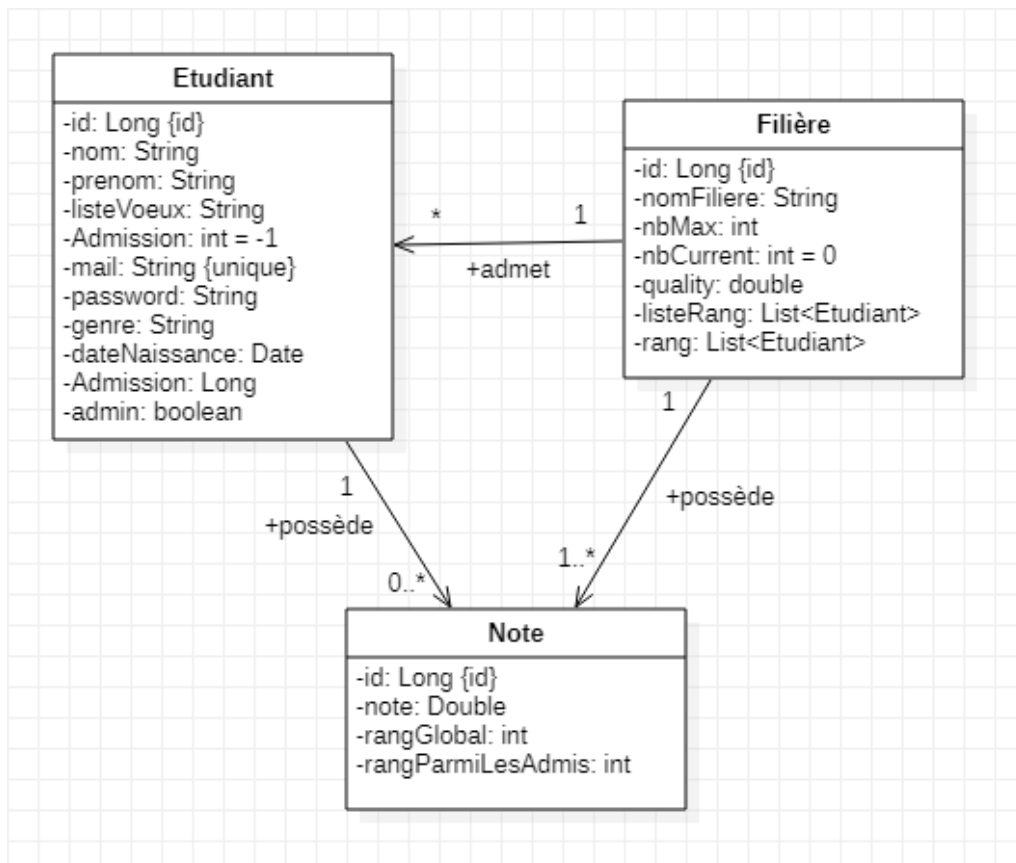
Si une filière reçoit les étudiants 1, 2 et 4 parmi sa liste de rang { 1, 2, 3, 4, 5 }, alors :

$$quality = \frac{(3 + 4 + 5) - (1 + 2 + 4)}{(3 + 4 + 5) - (1 + 2 + 3)} = 0,833$$

IV. Architecture Back-end

A. Vue d'ensemble

Les techniques utilisées pour construire la base de données sont identiques à celles du projet DASI : java avec maven dans l'IDE NetBeans + JPA.



Dans la base de données, on y trouvera 3 tableaux : **Etudiant**, **Filière** et **Note**, ce dernier est donc le tableau d'association entre les étudiants et les filières, il contient non seulement la note mais aussi le classement d'étudiant. Il sera complété après que l'algorithme se termine.

Il y a des attributs avec l'annotation `@Transient` qui sont utilisés uniquement pendant l'exécution de l'algorithme principal, tels `listeRang` et `rang` dans **Filière**.

B. Services

Les services consistent en 3 catégories : la modification, la requête de données et le programme principal.

```
public class Service {
    protected EtudiantDao etudiantDao = new EtudiantDao();
    protected FiliereDao filiereDao = new FiliereDao();
    protected NoteDao noteDao = new NoteDao();

    public Long authentifier(String login, String motDePasse) {...18 lines }
    public Etudiant rechercherEtudiantParId(Long id) {...13 lines }

    public int changerMotDePasse(Long id,String ancien,String nouveau) {...20 lines }
    public boolean changerMail(Long id,String mail) {...16 lines }
    public boolean changerListeVoeux(Long id,String liste) {...24 lines }

    public List<Note> obtenirBulletin(Long id) {...13 lines }
    public String obtenirListeVoeux(Etudiant e) {...19 lines }
    public String obtenirAdmission(Etudiant e) {...15 lines }

    // <editor-fold defaultstate="collapsed" desc="Main functions. Click on the + sign on th
    public List<Etudiant> listerEtudiant() {...13 lines }
    public List<Filiere> listerFiliere() {...13 lines }
    public void initialiserFiliere(List<Filiere> filiere) {...37 lines }
    public void initialiserEtudiant(List<Etudiant> etudiant) {...11 lines }

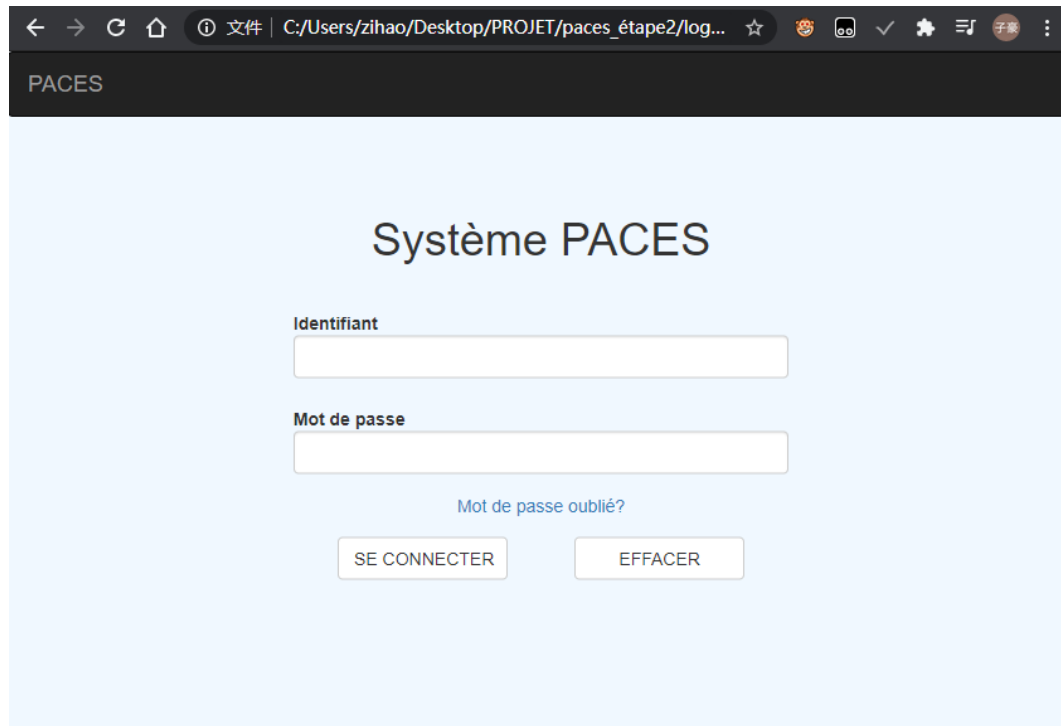
    public void mariage() {...122 lines }
    // </editor-fold>
}
```

C. Initialisation

110 objets Etudiant (y compris un compte admin) et 8 objets Filiere seront créés dans la base de données après l'initialisation des entités. Pour les étudiants, leurs informations personnelles (genre, date de naissance, mot de passe et notes dont l'intervalle est de 8.0 à 20.0) sont générées au hasard.

V. Description de fonctionnalités

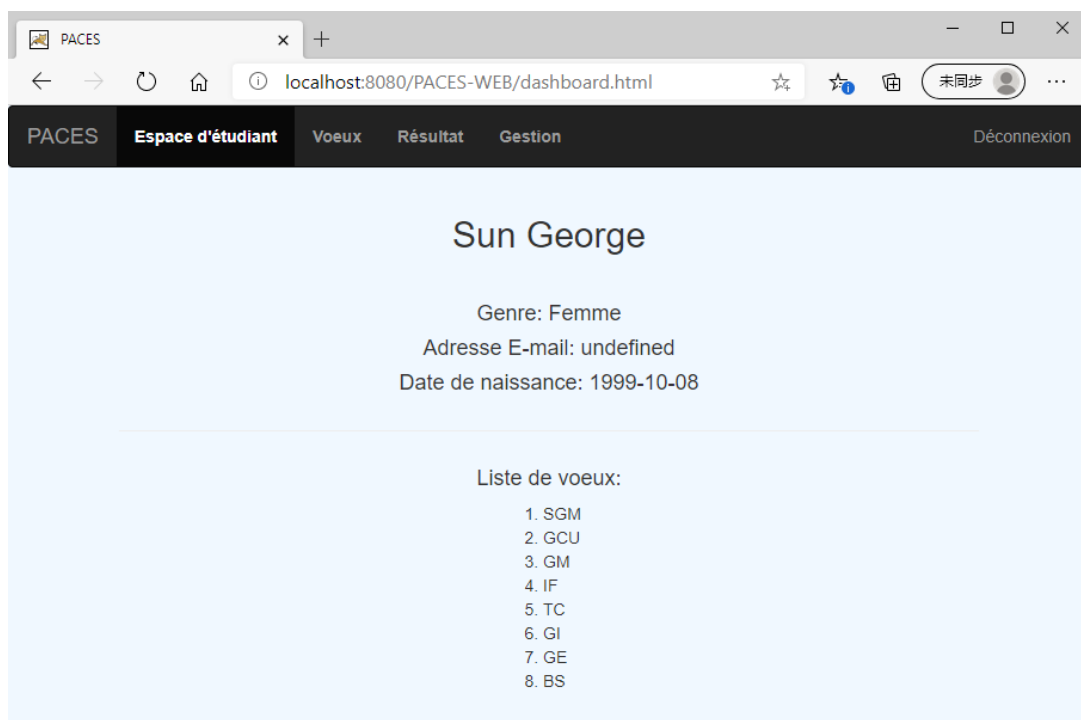
Les utilisateurs vont se connecter au système par le page de connexion (login.html) avec l'identifiant et le mot de passe fournis par l'organisation PACES, s'il ne souvient plus de son mot de passe, il peut envoyer une requête de réinitialisation en cliquant sur le lien en-dessous. Le bouton « EFFACER » permet de vider le formulaire.



The screenshot shows a web browser window with the address bar displaying 'C:/Users/zihao/Desktop/PROJET/paces_étape2/log...'. The page title is 'PACES'. The main content area has a light blue background and contains the following elements:

- Systeme PACES**: A large heading centered on the page.
- Identifiant**: A label above a text input field.
- Mot de passe**: A label above a text input field.
- Mot de passe oublié?**: A link centered below the password field.
- SE CONNECTER**: A button to the left of the 'EFFACER' button.
- EFFACER**: A button to the right of the 'SE CONNECTER' button.

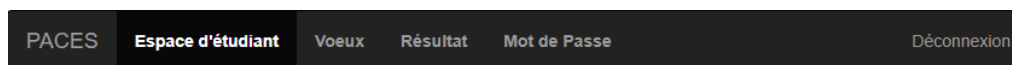
Une fois que l'utilisateur est connecté, il accède à son espace personnel où se trouve ses informations.



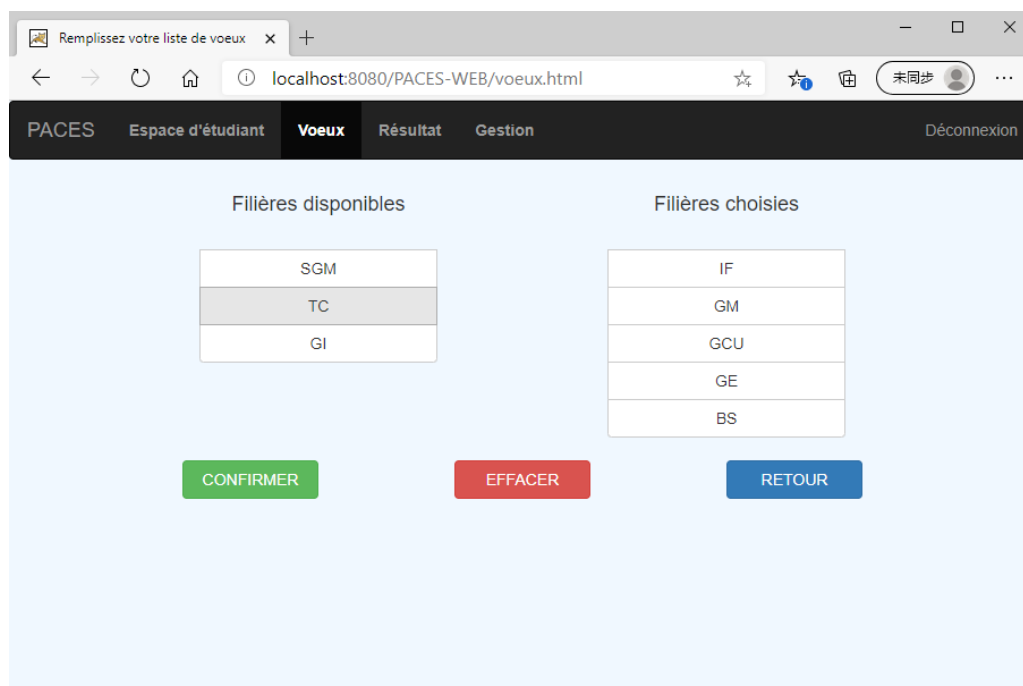
The screenshot shows a web browser window with the address bar displaying 'localhost:8080/PACES-WEB/dashboard.html'. The page title is 'PACES'. The main content area has a light blue background and contains the following elements:

- Espace d'étudiant**: A navigation menu item highlighted in the top bar.
- Sun George**: A large heading centered on the page.
- Genre: Femme**: A text label.
- Adresse E-mail: undefined**: A text label.
- Date de naissance: 1999-10-08**: A text label.
- Liste de vœux:**: A heading for a list of wishes.
- Wishes List**: A numbered list of wishes:
 1. SGM
 2. GCU
 3. GM
 4. IF
 5. TC
 6. GI
 7. GE
 8. BS

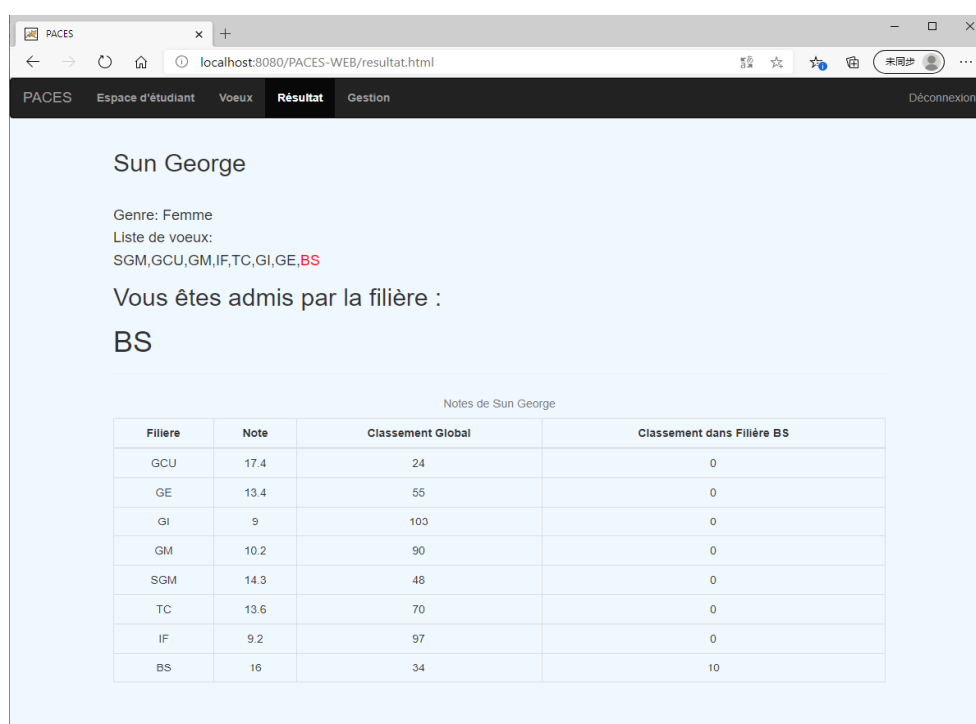
Il peut consulter et modifier sa liste vœux, consulter le résultat d'attribution, gérer son compte ou se déconnecter en cliquant les boutons sur la barre de navigation :



La structure principale du page « Vœux » consiste en deux groupes de boutons verticaux, ils correspondent aux filières choisies et non-choisies. En cliquant sur les boutons blancs, ils « sauteront » dans l'autre groupe. La liste gauche est initialisée par les données qui se trouve dans le serveur. Le bouton « effacer » sert à réinitialiser les deux groupes.

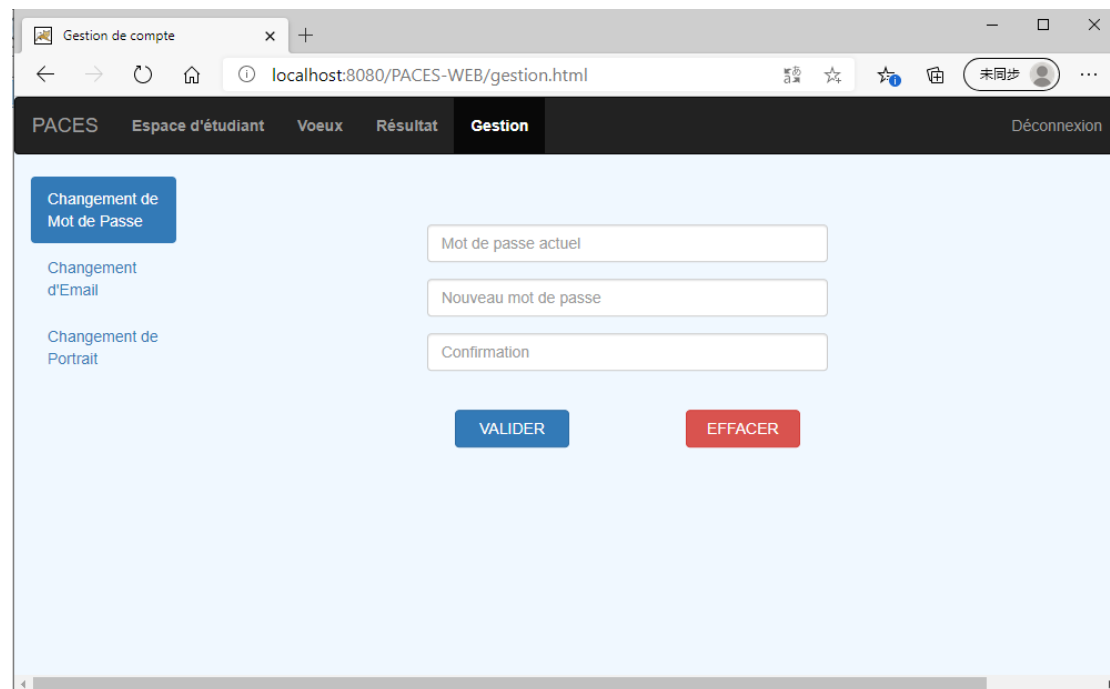


Ensuite, l'onglet « Résultat » sert à montrer le résultat d'admission et le détail de ses



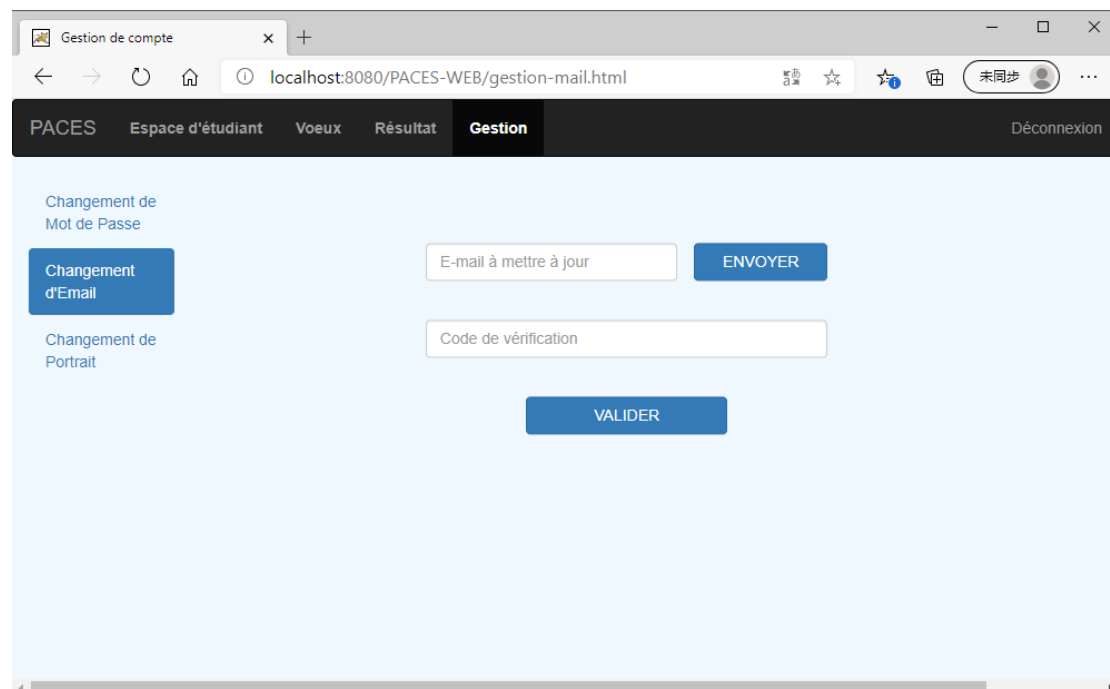
notes. Si malheureusement, cet élève a essuyé un échec, ce page affichera « non-admission » au lieu du nom de filière.

Dans l'onglet « Gestion », l'utilisateur peut changer son mot de passe ou relier son compte à une nouvelle adresse mail pour faciliter la connexion.



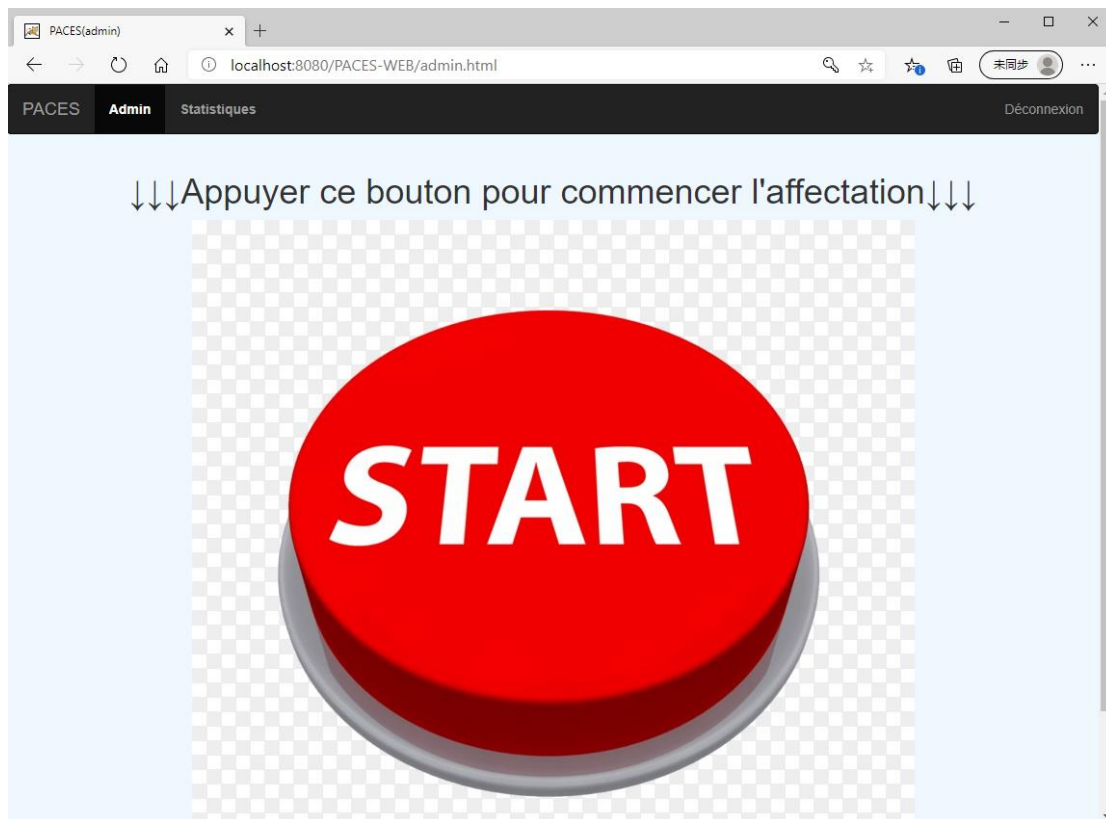
The screenshot shows a web browser window with the address bar displaying 'localhost:8080/PACES-WEB/gestion.html'. The page has a dark navigation bar with the following links: PACES, Espace d'étudiant, Voeux, Résultat, and Gestion (which is highlighted). A 'Déconnexion' link is on the right. On the left side of the page, there are three links: 'Changement de Mot de Passe' (highlighted in blue), 'Changement d'Email', and 'Changement de Portrait'. The main content area contains three input fields: 'Mot de passe actuel', 'Nouveau mot de passe', and 'Confirmation'. Below these fields are two buttons: 'VALIDER' (blue) and 'EFFACER' (red).

Pour changer l'adresse e-mail, si l'utilisateur ne l'a pas encore définie, une fois que le bouton « envoyer » est cliqué, le système enverra un mail contenant le code de vérification à l'adresse qui se trouve dans le champs à côté. Ensuite, l'adresse sera mise à jour en soumettant le bon code dans le champs suivant.



The screenshot shows the same web browser window, but the address bar now displays 'localhost:8080/PACES-WEB/gestion-mail.html'. The navigation bar and left sidebar are identical. The main content area now shows the email change form. It includes an input field 'E-mail à mettre à jour' with an 'ENVOYER' button next to it. Below this is a 'Code de vérification' input field and a 'VALIDER' button. The 'Changement d'Email' link in the sidebar is now highlighted in blue.

De plus, ce système dispose d'un compte administratif. L'admin peut consulter les statistiques d'admission et démarrer le programme principal qui affecte les étudiants.



The screenshot shows a web browser window with the address bar displaying 'localhost:8080/PACES-WEB/stats.html'. The page has a dark header with 'PACES', 'Admin', and 'Statistiques' tabs. Below the header, there is a table with the following data:

| Filière | Note Moyenne | Objectif | Situation Réelle | Qualité |
|---------|--------------|----------|------------------|---------|
| IF | 17.86 | 15 | 15 | 0.867 |
| SGM | 19.22 | 12 | 12 | 0.949 |
| GE | 18.72 | 15 | 15 | 0.892 |
| GCU | 19.07 | 10 | 10 | 0.937 |
| BS | 18.09 | 10 | 10 | 0.916 |
| TC | 15.74 | 15 | 15 | 0.683 |
| GM | 15.94 | 20 | 20 | 0.747 |
| GI | 19.05 | 10 | 10 | 0.928 |

Page statistique des filières après l'affectation

VI. Problèmes apparus et Solutions

A. Notes identiques

En application réelle, il y a toujours des étudiants possédant les mêmes notes. Si leur classements semblaient défavorables, la situation serait plus délicate. Il se peut que l'un parmi eux est accepté mais l'autre est éliminé. Pour l'égalité, nous devons améliorer les critères de classement : Si deux élèves ont des notes identiques, l'élève ayant la moyenne (la moyenne de toutes ses notes) plus élevée sera classé devant l'autre.

B. Fuite de données

Lorsque j'utilise la fonction `gson.toJson()` pour convertir directement une liste de type `List<Note>` en un `String` représentant un objet json. Je constate que toutes les information de l'objet **Etudiant** associé à ce **Note** ont été enregistrées dedans, y compris le mot de passe. Et cette chaîne de caractère sera transmis jusqu'au navigateur pour créer le tableau de notes. Il est donc très dangereux de procéder ainsi, il va falloir tronquer toutes les données concernant l'étudiant.

```
1. public class AfficherResultatAction extends Action {
2.     @Override
3.     public void executer(HttpServletRequest request) {
4.         HttpSession session = request.getSession();
5.         Service service = new Service();
6.         Etudiant etudiant = null;
7.         List<Note> notes = null;
8.         /*String nom;
9.         String genre;
10.        String liste;
11.        String note;
12.        String admission;*/
13.        Gson gson = new Gson();
14.        Long id = (Long) session.getAttribute("idEtudiant");
15.        if(id!=null) {
16.            etudiant = service.rechercherEtudiantParId(id);
17.            // obtenir le List contenant les notes de cet étudiant
18.            notes = service.obtenirBulletin(id);
19.        } else {
20.            request.setAttribute("notLoggedIn", true);
21.        }
22.        /*
23.        Paramétrer les autres attributs
24.        */
```

```

25.         if(notes!=null) {
26.             // convertir les objets en format json de type String
27.             note = gson.toJson(notes);
28.             int begin;
29.             int end = 0;
30.             int lastend;
31.             // enlever tous les attributs "etudiant"
32.             while(note.contains("\netudiant\"){
33.                 begin = note.indexOf("\netudiant\");
34.                 lastend = end;
35.                 end = note.indexOf("\nfiliere\"",lastend+1);
36.                 note = note.substring(0,begin) + note.substring(end);
37.             }
38.             request.setAttribute("notes",note);
39.         }
40.     }
41. }

```

VII.Conclusion

A priori, ce projet se fonde sur les techniques apprises au cours des séances DASI. Mais cette fois j'ai approfondi mes connaissances de jquery et bootstrap qui facilite la création du site. Et il est très intéressant d'implémenter un algorithme qui me semble assez abstrait dans une web-application. En fait, l'algorithme de Gale et Shapley a beaucoup de propriétés à étudier, ce système pourrait aussi être amélioré en détaillant plus les configurations d'algorithme. Pour qu'elle puisse fonctionner dans les situations spéciales, par exemple si le directeur de la filière veut mettre un seuil d'admission ou les étudiants veulent abandonner son admission. Je vais continuer le développement de cette application parce qu'elle est quand même très primitive.