Sorbonne Université
Master ANDROIDE

Outil d'appariement pour l'attribution des projets Androide

UE de projet M1

Rayan Perotti-Valle – Félix Savarit – Malo Thomasson –
Encadrant : Olivier Spanjaard

2024

Table des matières

Chapitre 1

Introduction

Notre projet s'est articulé autour de la conception d'un outil d'appariement pour l'attribution de projets dans le cadre de l'UE Projet Android. Sous la supervision de Monsieur Spanjaard Olivier, nous avons entrepris cette initiative qui comportait plusieurs volets. Tout d'abord, nous avons développé une interface web permettant de collecter les choix de projets des étudiants, ainsi que les groupes avec lesquels ils souhaitaient collaborer. En parallèle, nous avons conçu un programme linéaire destiné à optimiser l'affectation des étudiants à leurs projets en fonction de leurs préférences et des contraintes spécifiques.

Ce projet a nécessité un suivi régulier, avec des réunions hebdomadaires dirigées par Monsieur Spanjaard Olivier, qui nous a donné conseils et orientations tout au long du processus. Ensuite, nous avons intégré une troisième composante au projet afin de générer des données fiables pour tester notre solution. Cette évolution a enrichi notre approche et renforcé la robustesse de notre système.

Pour en savoir davantage sur notre travail et explorer les détails de notre démarche, nous vous invitons à consulter le dépôt git du projet disponible via le lien suivant.

Chapitre 2

État de l'art

L'attribution de projets aux individus ou aux équipes est une tâche complexe et cruciale dans de nombreux domaines tels que l'éducation, la recherche, l'industrie, et bien d'autres. Pour optimiser ce processus, les outils d'appariement basés sur la programmation linéaire et les mécanismes de préférences et mariages stables sont largement utilisés. Ces outils visent à équilibrer les préférences des parties prenantes tout en garantissant des résultats équitables et stables.

Le modèle classique d'appariement stable, popularisé par le problème des résidents et des hôpitaux (hospital-resident), consiste à faire correspondre des résidents (étudiants, chercheurs, etc.) à des hôpitaux (écoles, laboratoires, etc.) en tenant compte de leurs préférences mutuelles. L'algorithme de Gale-Shapley, également connu sous le nom d'algorithme des mariages stables, fournit une méthode efficace pour résoudre ce problème en produisant un appariement stable optimal.

Dans le contexte des projets, des extensions de ce modèle classique sont souvent nécessaires pour tenir compte de facteurs spécifiques tels que l'égalité des chances et la diversité des équipes. Des mécanismes de pondération peuvent être introduits pour refléter l'importance relative des critères de sélection, tandis que des contraintes supplémentaires comme la capacité ou la stabilité peuvent être imposées pour garantir une représentation équitable des différents groupes.

Les techniques de programmation linéaire sont largement utilisées pour formuler et résoudre les problèmes d'appariement dans ce contexte. En définissant des variables de décision correspondant aux appariements possibles et en formulant des objectifs et des contraintes appropriés, il est possible de modéliser efficacement les préférences des parties prenantes et de trouver des solutions optimales ou approchées.

De nombreux outils logiciels et bibliothèques sont disponibles pour résoudre des problèmes d'appariement complexes. Des environnements de programmation tels que Python avec des bibliothèques comme PuLP ou Gurobi fournissent des fonctionnalités avancées pour la modélisation et la résolution de problèmes d'optimisation linéaire, y compris les problèmes d'appariement.

La génération realiste de données est aussi une problematique importante pour tester et verifier le bon fonctionnement du programme linéaire. Des outils deja existant comme Faker, Pandas, spicy sont des librairies python fonctionnelles.

Chapitre 3

Contribution

3.1 Interfaces

Cette partie decrit les interfaces utilisées, leurs fonctionnalités, ainsi que nos choix de présentation et d'utilisation.

Nous avons réalisés deux interfaces distinctes reliées à la meme base de données. Une interface pour les encadrants et une pour les étudiants.

L'architecture du projet est de type MERN (**MongoDB**, **Express.js**, **React**, **Node.js**).

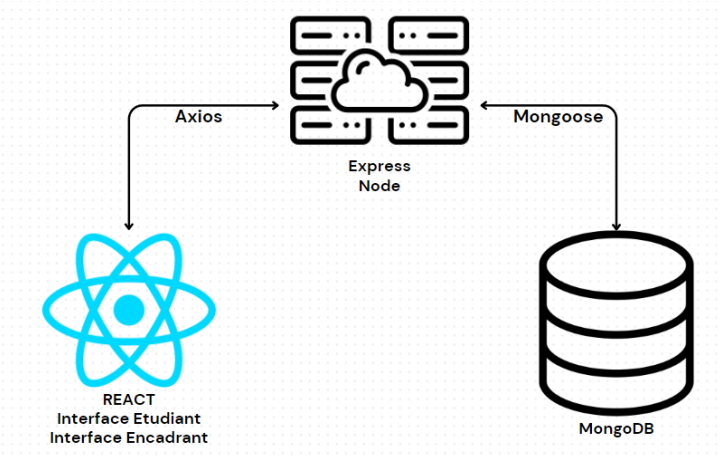

FIGURE 3.1 – Schema de l'architecture MERN pour notre projet

Nous avons choisi cette architecture notamment grace à l'UE technologie du WEB en L3 qui nous a appris à l'utiliser. Mais également car elle est tres pratique et adapté au projet.

3.1.1 Interface encadrant

L'interface encadrant est une interfaces utilisée et accessible uniquement par les encadrants. Elle a deux fonctionnalités principales. Dans un premier temps la création de nouveau projet par les encadrants.

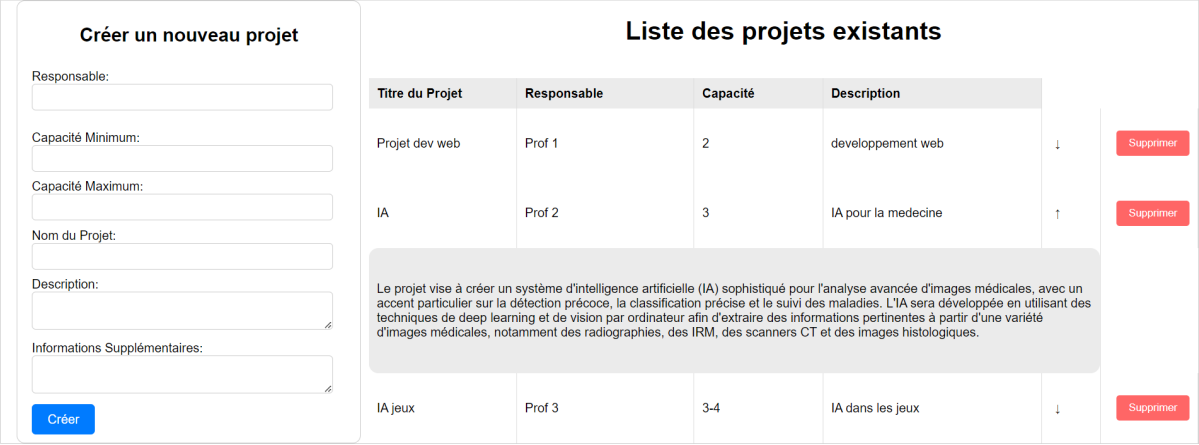

FIGURE 3.2 – Interface de création de projet par les encadrants

Sur la figure on voit qu'on peut creer son projet et qu'il s'affichera à droite une fois envoyé.

L'interface est tres simple d'utilisation, tout y est expliqué.

Le but n'étant pas de faire un site trop complexe, il faut qu'il soit adapté aux encadrants.

La deuxieme partie de l'interface encadrant apporte la possibilité de visualiser les groupes et membres des groupes ayant candidatés à chaque projet.

L'encadrant peut ensuite classer les groupes candidats selon un rang.

Cette section du site devient accessible une fois que tous les étudiants ont soumis leurs préférences. Les deux parties du site pour les encadrants ne sont pas directement liées. Elles sont accessibles chacune à des intervalles de temps différents.

FIGURE 3.3 – Interface de séléction des voeux pour les encadrants

Une fois les voeux envoyés, il est impossible de les modifier.

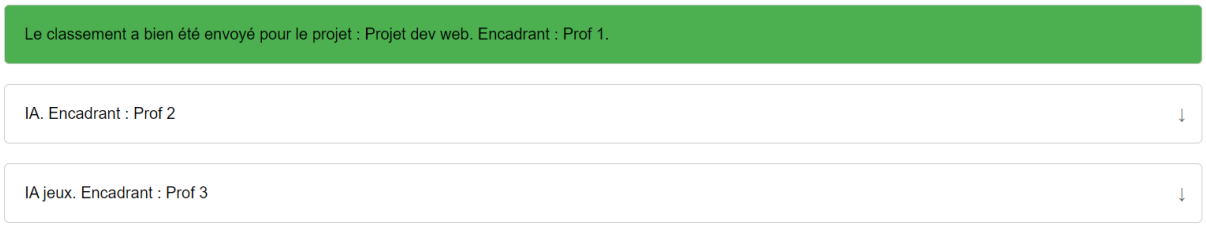

FIGURE 3.4 – Exemple d'envoie d'un classement pour un projet

3.1.2 Interface étudiante

L'interface étudiante est une interfaces utilisée et accessible uniquement par les étudiants. Elle presente 2 pages principales qui se suivent.

Dans un premier temps, les étudiants peuvent visualiser les projets avec leurs titres, responsables, capacités, descriptions.

Ils doivent ensuite choisir les projets souhaités et passer à la page suivante.

FIGURE 3.5 – Interface de séléction des voeux pour les étudiants

La page suivante est le remplissage des infos supplementaires pour chaque candidat du groupe.

Chaque candidat devra remplir cette page si un groupe postule à un projet. Par exemple si un groupe de 3 candidats postule à un projet, il devra etre rempli par les 3 candidats meme si des infos se repeteront. Mais cela est indispensable pour obtenir le score de chaque candidats pour un projet donné.

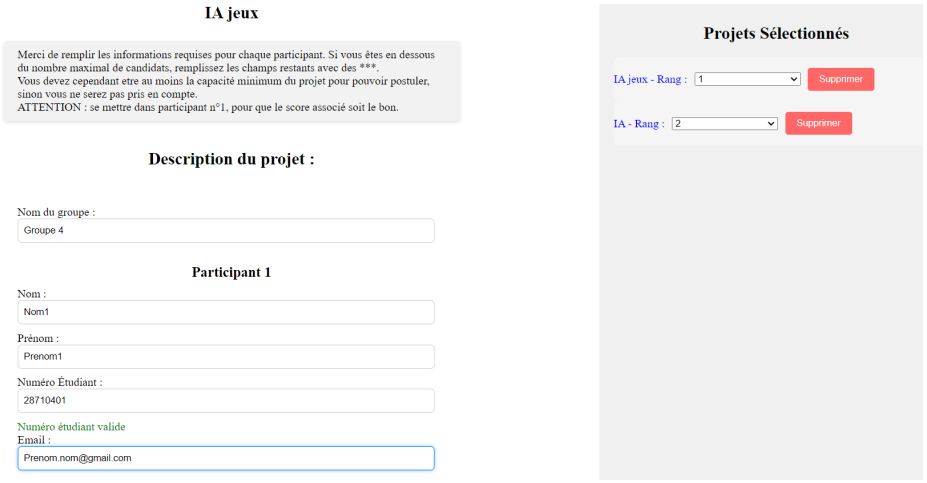

FIGURE 3.6 – Interface pour que l'étudiant remplisse les informations de son groupe pour un projet

Le site est fait pour au maximum eviter les données incoherentes dans les soumissions, créations, choix des rangs et autre problemes evitables en amont pour favoriser l'utilisation du programme linéaire.

Un étudiant ne peut par exemple pas rentrer le meme rang pour 2 projets, les numeros etudiants sont verifiés sur une base de données, il ne peuvent pas soummettre deux meme

groupes pour un projet etc...

Ainsi voici un exemple de fichier json recuperé après soumissions de voeux coté étudiant et encadrant.

On récupere les groupes candidats à chaque projet, avec dans chaque groupe les candidats et leur voeux pour le projet.

Le rang du groupe sera calculé à l'aide des rangs de tous les candidats d'un groupe.

Dans l'exemple suivant on a un exemple d'un projet *developpement web* qui contient 2 groupes candidats avec leurs scores.

Avant d'être soumis au solveur, ce fichier sera préparé à l'aide d'un script Python. Ce script réorganisera le fichier en une bibliothèque bien structurée, permettant ainsi de trier les listes de souhaits en vue de leur utilisation par le programme informatique.

```
[{
    "_id": { "$oid": "663e09ef0b938ed7cdf7e570" //id du projet},
    "responsable": "Prof 1",
    "capaciteMin": 2,
    "capaciteMax": 2,
    "nom": "Projet dev web",
    "description": "developpement web",
    "informationsSupplementaires": "",
    "submitted": false,
    "groupes": [
    {
      "nom": "Groupe 1",
      "rang": "1", //Rang du candidat 1
      "candidats": [
        {//candidat 1}
        },
        {//candidat 2}
      ],
      //id du groupe
      "_id": {"$oid": "663e0a890b938ed7cdf7e57c"}
    },
    {
      "nom": "Groupe 2",
      "rang": "1",
      "candidats": [
        //candidats du groupe 2
      ],
    },
    {
      "nom": "Groupe 1",
      "rang": "2", //Rang du candidat 2
      "candidats": [
        {//candidat 2},
        {//Candidat 1}
        ],
    }
    ],
    //classement de l'encadrant pour les groupes
    "classement": {
        "663e0a890b938ed7cdf7e57c": "1", //Rang du groupe 1
        "663e0cca0b938ed7cdf7e598": "2", //Rang du groupe 2
        },
    },
]
```

Listing 3.1 – Exemple de fichier JSON utile pour le solveur

3.2 Génération de données

Pour l'élaboration de notre projet, nous devons utiliser beaucoup de données pour tester notre programme linéaire et avoir de bons résultats. Ces données représentent les voeux des étudiants et les voeux des encadrants. Cependant comme nous n'avons pas accès à une base de données precises, nous devons générer nous même nos données. Pour cela nous avons exploré diverses approches et utilisé différentes méthodes de tirage et de rangement afin de générer des données cohérentes et pertinentes pour notre projet.

3.2.1 Méthodes utilisées

Pour notre solveur, nous avions besoin de différents types de données ;

1. Un grand nombre de projets.
2. Les préférences des étudiants pour les projets
3. Le rang des étudiants
4. Les groupes d'étudiants
5. Les voeux des groupes pour les projets
6. Les préférences des encadrants des projets

Dans un premier temps nous avons créé des classes représentant un type de préférence et un certain nombre de projets fictifs, pour les projets nous en avons juste créé un nombre n avec une taille prédéfini puis pour les classes on a simplement tirer des suites de nombres alétoires allant de 0 à n.

Nous avons pris en charge nos étudiants dans leur recherche du projet idéal en leur attribuant des préférences parmi une sélection de projets. Nous avons veillé à ce que ces préférences soient alignées avec les classes préalablement définies. Ensuite, nous avons regroupé les étudiants ayant des préférences similaires, tout en permettant la possibilité que des étudiants avec des préférences différentes puissent également se retrouver ensemble. Pour cela, nous avons utilisé la distance de Kendall-Tau dans le processus de répartition des étudiants dans les groupes. Cette approche a permis d'associer les étudiants en fonction de leurs classements individuels et d'obtenir des groupes présentant une homogénéité relative et une pertinence accrue.

Avant d'aborder en détail le reste du processus de génération des données, prenons un moment pour discuter de la distance de Kendall-Tau, un concept clé qui a été utilisé pour déterminer les préférences des étudiants et les regrouper dans les groupes.

La distance de Kendall-Tau mesure le nombre de paires discordantes entre deux permutations. Dans notre contexte, nous avons calculé le nombre de paires discordantes entre une classe donnée et les préférences des étudiants. Ensuite, nous avons regroupé les étudiants ayant un faible nombre de discordances avec une même classe dans des groupes communs. Pour mieux comprendre ce concept, voici un exemple illustratif de la distance de Kendall-Tau :

Soit π_1 et π_2 deux permutations d'une suite de chiffres. La distance de Kendall-Tau entre π_1 et π_2 correspond au nombre de paires en désaccord parmi toutes les paires possibles.

Soit $\pi_1 = [1, 2, 3, 4]$ et $\pi_2 = [2, 3, 1, 4]$ comme permutations.

Les paires possibles sont $(1,2),(1,3),(1,4),(2,3),(2,4),(3,4)$.

Ici, les paires $(1,2)$ et $(1,3)$ sont en désaccord. Par conséquent, la distance de Kendall-Tau entre π_1 et π_2 est de 2.

Maintenant que nous avons formé nos groupes et créé nos projets, il est temps de générer les choix des groupes pour les projets. Pour cela, nous parcourons tous les projets dans l'ordre de préférence du groupe, et dès que nous trouvons un projet auquel aucun des étudiants du groupe n'a postulé, le groupe le choisit. Pour finaliser la génération de données, nous allons maintenant créer les préférences des encadrants de projets pour les groupes.

3.3 Programmation linéaire

L'objectif principal de ce projet est d'associer les étudiants aux projets créés par les encadrants, en visant une répartition stable ou aussi proche que possible de la stabilité. Pour atteindre cet objectif, nous avons développé un solveur capable de résoudre le programme mathématique correspondant. Dans cette section, nous allons vous présenter ce solveur ainsi que le programme mathématique qu'il suit.

3.3.1 Programme mathématique

L'élaboration du programme mathématique pour le solveur a été la première étape de ce projet. En effet avant même de commencer à coder l'interface et le solveur, nous avions besoin du programme mathématiques. Pour cela, il nous fallait dans un premier temps déterminer les variables qui entrent en jeu dans notre problème, ici nos variables étaient les projets ; les groupes ; les étudiants.

Variables :

$$x_{eP}, \text{ l'étudiant } e \text{ affecté au projet } P$$
$$x_{GP}, \text{ le groupe } G \text{ affecté au projet } P$$

De plus, nous avons ajouté un nombre k représentant le nombre cible d'étudiants affectés, ainsi qu'une variable score correspondantà la somme du score pondérée d'un groupe et du score de l'enseignant. Une fois nos variables déterminer, nous avons dans un second temps mis en place la fonction objectif de notre problème, le but de ce projet étant de répartir les projets dans des groupes d'étudiants, la fonction objectif de notre programme est donc la maximisation des scores des groupes affectés à des projets.

$$\text{Maximiser} \sum_{G} \sum_{P} x_{GP} \cdot \text{score}(G,P)$$

Enfin, il nous reste le choix des contraintes, nous en avons donc créé 7 :

1. Il faut au moins k étudiants affectés à un projet :

$$\sum_{e} \sum_{P} x_{eP} \geq k$$

3. Chaque groupe d'étudiants est affecté à au plus un projet :

$$\forall G \sum_{P} x_{GP} \leq 1$$

4. Chaque projet doit être affecté à au plus 1 groupe :

$$\forall P \sum_{G} x_{GP} \leq 1$$

5. un étudiant ne doit pas avoir plusieurs projet :

$$\forall P, \forall e \in G, x_{GP} = x_{eP}$$

6. Contrainte de stabilité

$$\forall g, \forall p \sum_{g' \succeq_p g} x_{gp} + \sum_{e \in g} \sum_{p' \succ_e p} x_{ep'} \geq 1$$

7. Les variables de décision sont binaires :

$$x_{eP} \in \{0, 1\}$$
$$x_{GP} \in \{0, 1\}$$

3.3.2 Implantation en machine

Pour tester notre solveur, nous avons utilisé les données qu'on a généré cependant en situation réelle, le solveur devra utiliser les données récupérer par notre interface, on a donc du mettre en place des fonctions pour convertir les données à leur sortie de l'interface vers le format que l'on souhaite pour notre solveur.

```
{'Projet 0': ['Groupe 0', 'Groupe 2'],
'Projet 1': ['Groupe 1', 'Groupe 2']
}
{'Groupe 0': ['Etudiant 0', 'Etudiant 1'],
'Groupe 1': ['Etudiant 0', 'Etudiant 2'],
'Groupe 2': ['Etudiant 3', 'Etudiant 4']
}
{'Etudiant 0': ['Projet 0', 'Projet 1'],
'Etudiant 1': ['Projet 0'],
'Etudiant 2': ['Projet 1'],
'Etudiant 3': ['Projet 0'],
'Etudiant 4': ['Projet 1']
}
```

Listing 3.2 – Format des données pour notre solveur

Dans les projets et dans les groupes les éléments sont rangé par ordre de préférences.

Pour faire tourner notre porgramme linéaire, nous avons utilisé le solveur **GLPK**. Nous avons rentré tous les variables , les données, la fonction objectif et les contraintes dans le solveur et on a obtenu des affectations des groupes à des projets.

Chapitre 4

Conclusion

À travers ce projet d'outil d'appariement pour l'attribution des projets Androide, nous avon été confronté à des différents aspect techniques à développer en parallèle. Travailler sur ce projet a été une expérience enrichissante, nous permettant d'explorer en profondeur différents domaines, de la conception d'interfaces utilisateur web, à la génération de données pertinente en utilisant des méthodes mathématiques et enfin en passant par l'utilisation de solveur pour résoudre un programme linéaire, sans oublier l'organisation de groupe.

La conception et le développement de l'interface utilisateur, en utilisant une architecture MERN, ont été des défis passionnants. Nous avons dû jongler avec des technologies modernes telles que React et Node.js pour créer une expérience utilisateur fluide et intuitive.

La génération de données a été une étape importante. En simulant divers scénarios et en générant des ensembles de données réalistes, nous avons pu mettre notre solveur à l'épreuve dans des conditions variées. Cette phase nous a également permis de mieux comprendre les défis pratiques auxquels nous pourrions être confrontés dans un contexte réel.

En parlant du solveur, sa conception et son implémentation ont été le cœur de notre projet. À l'aide de Python et de la bibliothèque Pyomo, nous avons traduit notre modèle mathématique en code exécutable.

Au final, ce projet est le résultat d'un long proccessus de travail complet de plusieurs mois, de collaboration et d'apprentissage. Tout au long de ce projet nous avons acquis de nouvelles compétences.

Cependant, tant le site que le solveur peuvent bénéficier d'optimisations supplémentaires. Cela pourrait impliquer la conception d'une interface plus conviviale et interactive pour le site, ou encore l'amélioration des performances du solveur afin de maximiser le nombre k d'étudiants affectés à des projets.

Chapitre 5

Bibliographie

A. Kwanashie, D.F. Manlove An integer programming approach to the Hospitals/-Residents problem with ties. 20 AUgust 2013

Mallows model

Voici les differentes bibliotheques python utilisées :

PYOMO Solveur

Ranky

GLPK génération de données

Chapitre 6

Cahier des charges

6.1 Définition du problème

Ce projet à pour but de récupérer les préférences des étudiants pour leur choix des projets Androide dans le cadre de leur UE de projet. Dans un premier temps nous récolterons tous les choix des étudiants et professeurs à l'aide d'une interface web liée à une base de données. Cela nous donnera une liste. Dans un second temps nous utiliserons un programme linéaire sur les listes de préférences des étudiants et professeurs pour maximiser le nombre d'étudiants affectés à un projet tout en maximisant la somme des **scores** des étudiants et des **scores** des encadrants. On appelle le **score** d'un étudiant (resp encadrant) la position de son projet (resp groupe) dans sa liste de préférences, le projet (resp groupe) préféré d'un étudiant (resp encadrant) est n avec n son nombre de projet (resp groupe) dans sa liste de préférence. Son dernier projet (resp groupe) préféré a donc un score de 1.

6.2 Objectifs

6.2.1 Méthode de résolution

Programme mathématiques :

Variables :

$$x_{eP}$$

$$x_{GP}$$

- P projet
- G groupe
- e étudiant
- k nombre cible d'étudiants affectés
- score(G,P) : la somme du score pondérée des étudiants du groupe G et du score de l'enseignant du projet P.

Maximiser la somme de tous les scores :

$$\text{Maximiser} \sum_{G} \sum_{P} x_{GP} \cdot \text{score}(G, P)$$

Contraintes :

1. Il faut au moins k étudiants affectés à un projet :

$$\sum_{e} \sum_{P} x_{eP} \geq k$$

3. Chaque groupe d'étudiants est affecté à au plus un projet :

$$\forall G \sum_{P} x_{GP} \leq 1$$

4. Chaque projet doit être affecté à au plus 1 groupe :

$$\forall P \sum_{G} x_{GP} \leq 1$$

5. un étudiant ne doit pas avoir plusieurs projet :

$$\forall P, \forall e \in G, x_{GP} = x_{eP}$$

6. Contrainte de stabilité

$$\forall g, \forall p \sum_{g' \succeq_p g} x_{gp} + \sum_{e \in g} \sum_{p' \succ_e p} x_{ep'} \geq 1$$

7. Les variables de décision sont binaires :

$$x_{eP} \in \{0, 1\}$$
$$x_{GP} \in \{0, 1\}$$

6.2.2 Outils de recueil des voeux étudiants/encadrants

On développera une interface web en javascript/html/CSS qui aura un coté recueil des voeux étudiant et un coté recueil des voeux encadrant. Le but étant de recueillir les préférences de chacun dans une base de données. Une fois les données récupérées on fait tourner l'algo dessus en local. On pourra si nécessaire ajouter des fonctionnalités supplémentaires comme le dépôt de CV ou lettre de motivation sur demande des encadrants.

6.3 Délais

Semaine du 20 Mai

Chapitre 7

Manuel utilisateur

7.1 Site

Chaque fichier du code contient des commentaires permettant de comprendre chaque fonctionnalité.

Le site contient 3 dossiers :

- Dossier serveur
- Dossier encadrant
- Dossier étudiant

Les 3 dossiers contiennent chacun un fichier README donnant les instructions pour installer les prérequis, les instructions pour lier le frontEnd et backEnd et enfin les instructions pour exécuter chaque partie du code.

7.2 Génération de données

Le code pour la génération de données, présent dans le dossier **RandGen**, est divisé en 2 fichiers.

- **class.py**
- **RandGen.py**

Le fichier **class.py** contient toutes les classes utilisées pour la génération de données.

RandGen.py contient le main et les fonctions de génération.

Un fichier README est présent pour expliquer les informations d'exécution du fichier **RandGen.py**, avec les paramètres et leur signification.

7.3 Solveur

Pour exécuter le solveur, préparez d'abord un fichier JSON contenant les données dans le format spécifié.
Ensuite, ouvrez le fichier **solveur.py** et assurez-vous d'ajuster manuellement le nom du

fichier JSON à utiliser dans la fonction principale. Si nécessaire, modifiez également la valeur de k pour représenter le nombre minimum d'étudiants devant être affectés à un projet.
Enfin, lancez l'exécution du fichier principal pour obtenir les résultats du solveur.

Dans **solveur.py**, des instructions pour vous aiguillez sont ecrites.