

**Algorithmique**

**S.A.É. 3.A.01** :

Développement d’application et Gestion de projet

**B.U.T. Informatique : semestre 3 et 4 (2022-2023)**

**I.U.T. de Bayonne et du Pays-Basque**

Tuteur : M. Chbeir – Projet : 13

Application destinée à gérer les parrainages du BDE

BRIÈRRE Titouan, DARGAZANLI Nicolas, ERREZARRET Leho, MAURICE Alexandre,

en TD I (parcours A).

# Pitch de l’application

Nous travaillons sur un site web permettant de gérer automatiquement les parrainages réalisés par le BDE.

Le parrainage est un évènement annuel réalisé par l'association étudiante Hego Berria (un BDE), dans lequel les étudiants de première année répondent d'abord à un questionnaire rédigé par les membres du BDE. Ensuite, les étudiants de deuxième année sont associés automatiquement (ou en choisissant, en fonction du paramétrage du questionnaire) à un étudiant de 1ère année, créant alors l'association entre le parrain et le filleul.

Le système permet aux utilisateurs de s'inscrire avec leur adresse mèl de l'UPPA et de conserver le même compte tout au long de leur cursus au sein de la structure. Le site internet est destiné aux étudiants et au BDE, qui ne dispose que d'un compte "administrateur".

Toute personne se connectant au site peut voir si le questionnaire est disponible ou non, modifier ses informations personnelles. On dispose alors de leur nom, leur prénom, leur date de naissance et leur sexe.

Les questionnaires sont rédigés et ouverts aux réponses par le BDE. Ils sont caractérisés par des questions et des réponses de différents types (Booléen, QCM, Réponse courte, Réponse longue), un type d'association (automatique, en faisant intervenir l'algorithme de l'application, ou manuelle, nécessitant l'intervention des étudiants de 2ème). Enfin, une date d'ouverture et de fermeture sont spécifiés.

[Justification du projet]

Le bureau des étudiants Hego Berria est une association avec une identité forte, qui commence à développer en parallèle (sans lien avec la SAE) un site web et une application de covoiturage. Faciliter ce système de parrainage peut alors le rendre plus fiable et avenant, en proposant des fonctionnalités ludiques dans un environnement maîtrisé (sans devoir recommencer à chaque fois tous les processus de création du formulaire ou devoir utiliser des applications tierces comme Excel ou Forms de chez Google, dénaturant l'association).

# Pitch du problème

L’accent est ici mis sur le problème algorithmique, portant dans notre cas sur l’association automatique des parrains/filleuls.

Il faudrait que notre algorithme puisse associer les étudiants grâce à leurs réponses au formulaire en question, en se basant sur un « score de compatibilité ». Les étudiants ayant le plus de réponses similaires seront alors associés entre eux.

En limitant le champ des possibles pour les réponses, notre algorithme est déjà sur la bonne voie. Cependant, nous souhaiterions laisser plus de liberté pour ces formulaires, en laissant la possibilité d’entrer des réponses libres, saisies au clavier. Nous savons d’expérience que les étudiants, s’ils répondent au questionnaire sérieusement, répondent plus ou moins des choses similaires. A la question « Décris-toi en 10 mots maximum » par exemple, il y a quelques chances qu’il y ait deux étudiants qui répondent « Timide » ou encore « Sportif », permettant de créer un lien fort et fiable entre ces étudiants, par rapport au susmentionné score de compatibilité.

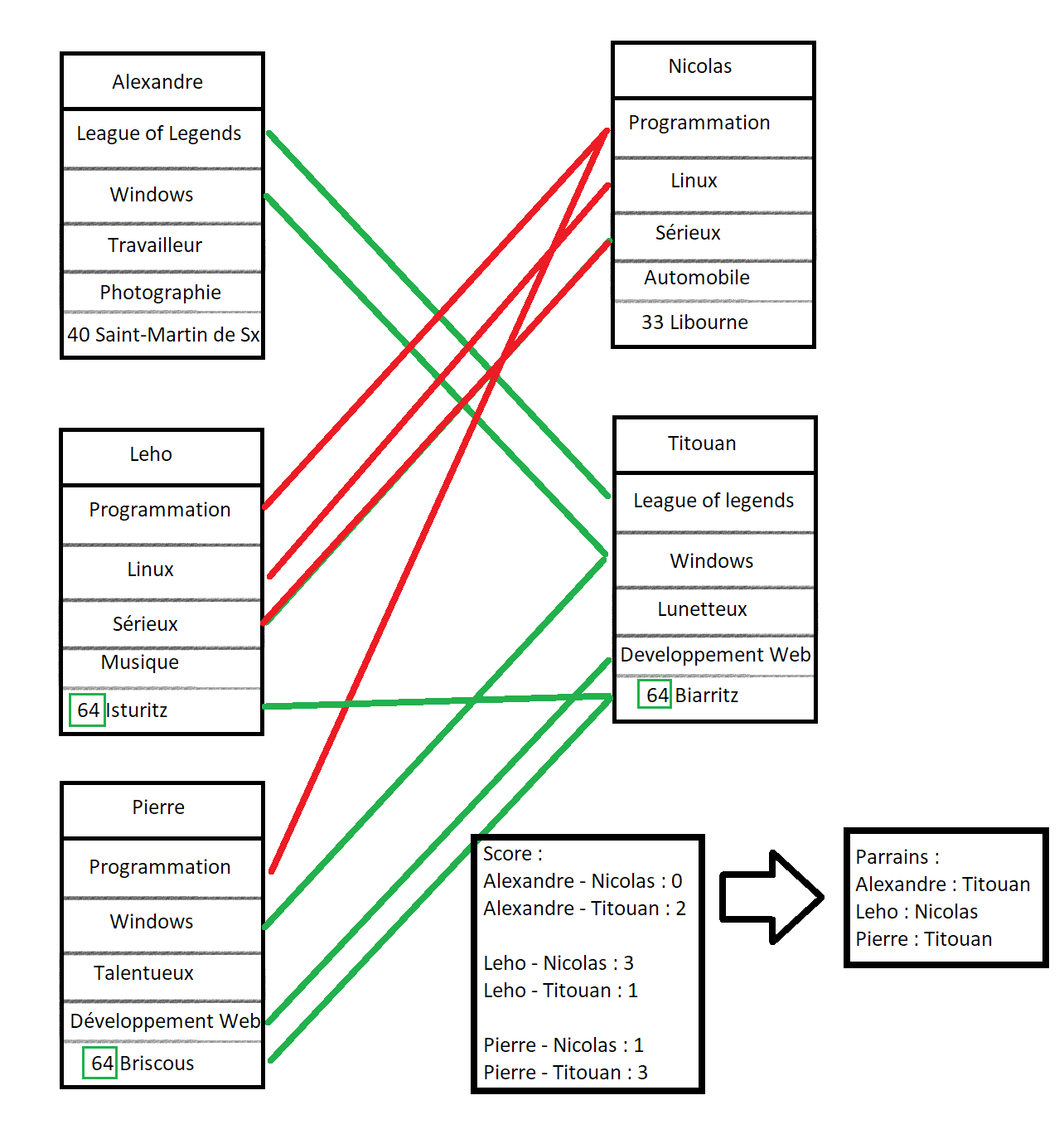
# Maquettes et/ou schémas explicatifs

Ci-dessous un exemple sous forme de table des réponses à un formulaire du BDE.  
Représentation d’un échantillon de 5 étudiants, avec les étudiants à gauche (Alexandre, Leho et Pierre), des étudiants de 1ère année, et ceux de droite (Nicolas et Titouan), des étudiants de 2ème année.

Chaque lien similaire ajoute un point au score de compatibilité, et on pourrait imaginer une analyse des mots de chaque réponse libre, comme c’est montré dans cet exemple avec la détection du département, et l’association avec un numéro de département similaire.

Le questionnaire est le suivant :

1. Tu préfères League of Legends ou la programmation ?
2. Windows ou Linux ?
3. Décris-toi en un adjectif.
4. Quelle est ta passion ?
5. Où habites-tu ? (pour du covoiturage par exemple) (Numéro de département + Ville)



# Informations manipulées (données et résultat)

Les informations que notre algorithme reçoit en entrée est la liste des étudiants de 1ère année ainsi que leurs réponses associées, et la liste des étudiants de 2ème année ainsi que leurs réponses associées également.

En sortie, notre algorithme doit retourner sous forme de liste les couples qu’il a calculé, avec éventuellement leur score de compatibilité pour du débogage par exemple. (filleul, parrain, [score]).

Il y a plusieurs types de question que notre algorithme doit prendre en compte :

*Questionnaire :*

*- Un texte (pour la question)*

*- Une liste de cases à cocher (avec un texte associé) (par défaut 2) (minimum 2)*

*- Un nombre maximum de cases à cocher (par défaut 1) (minimum 1)*

*Exemple :*

*Texte : Êtes-vous intéressés à l'idée d'avoir un parrain ?*

*Liste des cases : Oui, Non, Indifférent*

*Nombre maximum : 1*

*Affichage :*

*Si un choix max :*

*Forme : boutons*

*Afficher côte à côte si deux choix seulement, et si on a la place. En colonne sinon.*

*Sinon : cases à cocher et afficher en colonne*

*Question libre :*

*- Un texte (pour la question)*

*- Un nombre de caractères maximum (par défaut 100)*

*Exemple :*

*Texte : Quels sont vos loisirs ?*

*Maximum : 400*

*Slider :*

*- Un texte (pour la question)*

*- Une valeur minimum (par défaut 0)*

*- Une valeur maximum (par défaut 100)*

*Exemple :*

*Texte : résistance à l'alcool*

*Minimum : 0*

*Maximun : 100*

*Affichage : ON PENSE à un affichage dynamique, qui change en fonction d'à quel point on pousse le slider*

*Ce que "ON PENSE" signifie :*

*- si on n’a pas le temps, on ne fait pas*

*- si on a le temps, on fait*

*- si on a trop de temps, on fait plus*

**Fonction** associerParrainFilleul(reponses1eAnnees, reponses2eAnnees)

matriceScore = creerMatriceScore(reponses1eAnnees, reponses2eAnnees)

tableauAssociation = appliquerMethodeHongroise(matriceScore, reponses1eAnnees, reponses2eAnnees)

**Retourner** tableauAssociation

**Fonction** creerMatriceScore(reponses1eAnnees, reponses2eAnnees)

**Récupérer** liste1eAnnees, liste2eAnnees

scoreMaximal = nombreDeQuestions

**Initialiser** matriceScore[nbPremiereAnnees,nbSecondeAnnee] avec chaque case valant scoreMaximal

**Pour chaque** ligne

**Pour chaque** colonne

**Pour chaque** reponse

matriceScore[ligne][colonne] = matriceScore[ligne][colonne] -calculerDistanceReponse(reponses1eAnnees[liste1eAnnees[ligne]][reponse], reponses2eAnnees[liste2eAnnees[j]][k])

**Pour chaque** ligne de matriceScore

**Pour chaque** colonne de matriceScore

matriceScore[ligne][colonne] = entier(100 \* arrondir(matriceScore[ligne][colonne]))

matriceScore, liste1eAnnees, liste2eAnnees = dupliquerEtudiants(matriceScore, reponses1eAnnees, reponses2eAnnees)

**Fonction** dupliquerEtudiants(matriceScore, reponses1eAnnees, reponses2eAnnees)

**Si** il y a plus de réponses de 1e années que de 2e années

**Initialiser** listeEtudiantsADupliquer[]

**Pour chaque** reponses2eAnnee

**Si** reponses2eAnnee[Etudiant][reponseNbFilleul] = "plusieurs"

**Ajouter** Etudiant à listeEtudiantsADupliquer

**Initialiser** listeMoyenneScore[]

**Pour chaque** etudiantADupliquer

somme = 0

**Pour chaque** ligne de matriceScore

somme += matriceScore[ligne][colonne]

**Ajouter** à listeMoyenneScore ([somme / taille(matriceScore), 0])

**Pour** i allant de taille(reponses1eAnnees)-taille(reponses2eAnnees)

**Initialiser** valeurMin à l’infini

**Pour chaque** etudiantADupliquer

**Fonction** appliquerMethodeHongroise(matriceScore)

**Initialiser** tableauAssociation[nbPremiereAnnees,2]

**Démonstration au tableau**

**Fonction** match(reponsePremiereAnnee, reponseSecondeAnnee)

**Retourner** nombreReponsesCochéesEnCommun/nombreReponsesCochéesMaximum