ENSAI
École nationale de la statistique et de l'analyse de l'information

PROJET INFORMATIQUE

Etudiants 2A

Rapport de projet : application de rencontre

Groupe 28 :
Adrien FERNANDEZ
Samuel GOUTIN
Sidi KADER TRAORE
Caleb CARLOSS AGUIDA
Lucie BOUIN

Encadrant :
Antoine BRUNETTI

2019 - 2020

Sommaire

Table des figures

Introduction

Depuis plusieurs années les phénomènes d'internet, suivis des réseaux sociaux et enfin des sites de rencontres explosent. Le site Match.com, créateur de la rencontre en ligne, voit le jour en 1995. Depuis des centaines de sites ont été créés. Age, situation professionnelle, région géographique ; nombreux sont les critères utilisés et mis en avant par chaque site. Longue ou ponctuelle, amoureuse ou amical, de nombreuses relations existent grâce aux sites de rencontres.

C'est à partir de cette grande idée que nous avons créé une application de rencontres comprenant selon nous les critères les plus essentiels : l'âge, l'orientation sexuelle, la situation géographique et les centres d'intérêts.

Notre objectif est de créer une application accessible à tous donc simple d'utilisation mais présentant les options élémentaires : la mise en relation avec des personnes compatibles géographiquement et l'échange de messages.

En tant qu'élève de deuxième année nous avons mis en oeuvre nos différentes connaissances informatiques acquises l'année précédente telles que le codage orienté-objet, le langage UML et SQL et la manipulation de bases de données. Nous nous sommes concentré particulièrement sur l'optimisation du langage et la rigueur dans nos pratiques. Nous avons également mis en application nos nouvelles connaissances à partir de l'utilisation d'API, de Git et d'une structure en couche pour notre codage.

Dans ce rapport, nous aborderons dans un premier temps l'organisation interne de notre groupe et la description des besoins. Dans un second temps nous vous présenterons notre méthode de conception de l'application que nous avons imaginé à travers notre conception générale ainsi que la création de nos bases de données. Enfin, dans un troisième temps, nous détaillerons les fonctinnalités de l'application, ses spécificités et son guide d'utilisation. Enfin nous conclurons en apportant chacun notre point de vue.

Chapitre 1

Organisation interne

Dans ce chapitre nous allons vous présenter comment nous avons organisé notre projet. C'est le premier projet par groupe de cinq que nous effectuons. Une répartition des tâches rigoureuse a donc du être mise en place pour optimiser notre temps et profiter des qualités de chacun.

1.1 Les différentes phases

La phase d'étude préalable a eu pour but de se mettre d'accord sur la solution envisagée et de planifier les grandes phases de la réalisation (diagramme de cas d'utilisation et diagramme de Gantt). Ensuite, la phase de conception générale de l'application a eu pour but de décrire les exigences fonctionnelles générales par la modélisation (diagramme d'activité ou d'états, diagramme de classes, modèle de données ...) et de planifier la mise en place des fonctionnalités (dépendances, priorités, ...).

1.2 Répartition du travail

Bien que nous avons réfléchi ensemble à ces deux phases, nous avons malgré tout réparti la production finale de chaque diagramme afin de diviser la charge de travail. C'est d'après ce fonctionnement que nous avons également abordé la phase de réalisation du projet. En effet, nous avons continué à prendre les décisions importantes, fondatrices du projet ensemble. Ainsi, chacun a pu donner son avis et choisir de produire les parties du codage sur lesquelles il serait le plus à l'aise. Nous avons malgré tout remarqué une forte hétérogénéité de niveau sur la partie codage. Ceci a compliqué la répartition de la charge de travail car certain d'entre nous se sont trouvés en partie dépendants pour faire avancer efficacement le codage. D'ailleurs, sur les parties les plus difficiles, un travail en commun a été fourni même si il est plus coûteux en terme de temps. Tous nos travaux sont coordonnés par notre chef de projet, Adrien Fernandez qui a rythmé notre travail et réparti les tâches si nécessaire.

Nous vous présentons dans l'annexe **A.1** un diagramme de Gantt réalisé sur GanttProject. Il représente de manière chronologique les différentes phases et les différentes tâches effectuées tout au long du projet ou à effectuer. Nous avons décidé de fixer nos deadline en fonction des temps de rencontre prévus avec notre encadrant.

1.3 Nos outils

Nous avons codé l'application dans le langage Python à l'aide de l'environnement de développement PyCharm. Nous avons également décidé d'utiliser le logiciel de gestion de versions Git afin de faciliter notre travail à plusieurs sur le même code source. L'intérêt est aussi de pouvoir conserver un historique de chaque modification des fichiers. L'utilisation de Git nous a permis de prendre en main ce logiciel même si l'intervention de notre tuteur a parfois été nécessaire. Concernant la gestion de base de données, pour la persistance des données nous utiliserons le système PostgreSQL.

Chapitre 2

Etude préalable

Nous allons dans un premier temps présenter l'application et les besoins à satisfaire de manière générale en nous appuyant sur différents diagrammes (**diagramme de menu**, **diagramme de cas d'utilisation**...).

Dans un second temps, nous nous concentrerons davantage sur la conception afin de répondre aux besoins qui auront été définis auparavant. Là aussi, nous nous appuierons sur quelques diagrammes(**diagrammes d'activité**, **diagramme de classes**, **diagramme de package**).

2.1 Description des besoins

Le but premier d'une application de rencontre est de pouvoir créer des relations (amicales, amoureuses...). Notre application se concentre sur la création de relations amoureuses. Pour cela il est nécessaire que chaque utilisateur qui se connecte à l'application puisse se démarquer et se mettre en avant. Chaque utilisateur dispose d'un **compte** qui lui est propre. Ce compte contient des informations qui sont modifiables et consultables par l'utilisateur, mais aussi consultables par d'autres personnes. Il contient des informations personnelles comme des photos de profil, une description et les différents centres d'intérêts de l'utilisateur.

Après avoir mis à jour son compte, l'utilisateur peut rechercher des profils d'autres utilisateurs. L'application met en relation des profils qui possèdent des des critères de recherche concordant qui sont proches afin de maximiser les chances de créer une relation. Puis, lorsqu'un utilisateur rencontre un profil qu'il considère intéressant, celui-ci peut mettre un **"like"** sur ce profil. Si deux comptes se donnent mutuellement un "like", on dit qu'un **"match"** se crée. Le "match" est la condition nécessaire pour établir la communication avec un autre utilisateur et proposer un rendez-vous. En effet, après avoir "matché" une personne, l'utilisateur peut utiliser sa **messagerie** pour lui envoyer un message ou organiser un rendez-vous.. La messagerie est donc un outil qui permet à l'utilisateur de consulter tous les profils avec qui il est en relation, c'est à dire ses **"matchs"** et ses **"like"** en attente. Il peut ainsi y consulter tous ses messages.

Lorsqu'un utilisateur veut proposer un rendez-vous, l'application peut, grâce à l'utilisation d'une API, proposer des adresses de restaurants, de bars, de parcs ou d'autres lieux au choix qui sont proches d'un point de référence de l'utilisateur (que l'utilisateur renseigne) afin de faciliter la rencontre.

2.2 Rapport d'analyse

La rédaction du rapport d'analyse nous a permis de nous faciliter la tâche lorsque nous avons commencé à coder l'application. En effet, cela nous a permis d'avoir une bonne vision globale du fonctionnement de l'application mais cela nous a aussi permis d'établir un ordre d'importance concernant le codage des différentes fonctionnalités.

Chapitre 3

Conception générale de l'application

L'objectif de ce chapitre est de créer un modèle de conception afin de répondre aux besoins décrits dans le chapitre précedent.

3.1 Deux profils différents

Comme nous pouvons le voir dans l'Annexe **B.2**, notre application propose des actions différentes en fonction de si vous êtes connecté à un compte ou non.

3.1.1 Profil non connecté

Pour commencer, nous allons décrire les actions possibles du profil **non connecté**. Ce profil représente les utilisateurs qui ne sont pas encore connectés ou qui ne possèdent pas encore de compte. Ils auront alors deux choix possibles, s'authentifier ou créer un compte. Lors de la création du compte, il sera demandé à l'utilisateur de renseigner des informations (prénom, nom, email, sexe,...). Une fois connecté ou le compte crée, l'utilisateur sera en possession de son propre compte, et il aura ainsi accès au menu principal de l'application.

3.1.2 Profil connecté

Une fois connecté, l'utilisateur peut effectuer plusieurs actions :

- Consulter/Modifier son profil : L'utilisateur peut ajouter ou modifier sa description, ses centres d'intérêts, ses photos de profil et ses préférences (orientation sexuelle).

-Proposer un rendez-vous : L'utilisateur peut dans un premier temps **consulter les matchs** qu'il possède. Dans un second temps, il a la possibilité de sélectionner le match à qui il souhaite proposer un rendez-vous. Puis, il va choisir le type de rendez-vous qu'il va proposer (bar, parc, restaurant ou autres). Grâce à **l'API**, l'application va retourner une liste de lieux possibles proches d'un lieu de référence que l'utilisateur aura précisé. Il peut ainsi choisir le lieu qu'il considère comme idéal pour son rendez-vous. Une fois le lieu choisi, l'application envoie automatiquement une proposition de rendez-vous à l'autre utilisateur. Si l'autre utilisateur accepte la proposition, ils pourront alors se mettre d'accord sur la date et l'heure de leur rencontre.

- Accéder à sa messagerie : L'utilisateur peut consulter sa boîte de réception où sont conservés tous les messages reçus et envoyés. Il peut voir si celle-ci contient de nouveaux messages non lus

grâce à une notification. Une fois dans sa messagerie, il peut sélectionner un message en particulier qu'il pourra lire, puis il a la possiblité d'y répondre ou de le supprimer (cf Annexe **B.1**). De plus, il peut communiquer avec une personne en particulier à l'aide d'une **messagerie isntantannée** en précisant le pseudo du destinataire des messages à l'application. Cette messagerie lui permet ainsi de communiquer avec ses différents matchs, et notamment d'organiser des rendez-vous comme on a vu ci-dessus.

- Consulter des nouveaux matchs : Il est possible d'accéder à l'onglet de recherche de matchs. Ensuite l'utilisateur peut afficher et modifier les critères de recherche (portée géographique, âge, sexe) ou bien lancer tout simplement la recherche. Il a ensuite des profils de différents utilisateurs qui lui seront proposés. L'utilisateur peut ainsi liker ou passer le profil, ce qui en fera apparaître un autre.

- Consulter les profils d'autres utilisateurs : Il peut consulter les profils des utilisateurs qui lui ont envoyé un message, mais aussi des profils qu'il rencontrera lors de sa recherche de nouveaux matchs.

- Supprimer son compte : Il est également possible de supprimer son compte si l'utilisateur considère qu'il ne souhaite définitivement plus utiliser l'application.

- Déconnexion : Et pour finir il est évidemment possible de quitter de l'application.

3.2 Diagramme de classes

Dans l'annexe **B.3** nous voyons que la classe centrale sera la classe `Compte`. Chaque utilisateur connecté sera donc affecté à une instance de classe Compte qui lui est propre. Cette classe possède en attributs les caractéristiques de chaque utilisateur (sexe, âge, nom, prénom, ...). Lors de chaque connexion, l'application va ainsi créer une instance de classe Compte grâce aux informations contenues dans la base de données.

On voit que chaque compte peut posséder plusieurs messages, et que chaque message est précisément lié à deux comptes. Donc chaque message correspondra à une instance de la classe messagerie et sera défini par un identifiant (clé primaire), les identifiants des deux comptes concernés, ainsi que la date et le contenu du message.

Un compte peut être lié à un ou plusieurs **like**, et chaque like est forcément lié à deux comptes. Un like et un match correspondront chacun à une instance de cette classe, et possédera en attributs les identifiants des deux utilisateurs concernés, la date du like ainsi que son **type**. Le type est un booléen qui nous indique si le like est à sens unique ou s'il est partagé. Dans le cas où le like est partagé, celui-ci sera considéré comme un **match**.

Enfin, chaque compte possède zéro, un ou plusieurs centres d'intérêts. Ces centres d'intérêts étaient à l'origine supposés nous permettre de sélectionner les comptes qui ont des passions en commun afin faciliter les mises en relation. En effet, l'objectif initial était que lorsqu'un utilisateur allait accéder à son onglet de recherche de match, l'application allait lui proposer les profils avec qui il partage des centres intérêts. Malheureusement, nous n'avons pas réussi à programmer cela dans le temps qui nous était imparti.

Même chose pour la classe Detail_recherche, cette classe était supposée nous permettre de sélectionner les profils à proposer à l'utilisateur en fonction de différents critères.

Enfin, chaque utilisateur a la possibilité d'organiser des rendez-vous. En effet, l'application, grâce à l'utilisation de l'API, propose différents lieux de rendez-vous aux utilisateurs qui le demandent. Ces lieux peuvent être très variés (bars, restaurants, parcs ou autres).

3.3 Diagramme de package

L'ensemble du code sera réparti dans trois grandes parties à savoir : la partie "Interface", la partie "Business_model" et la partie "Data Access".

3.3.1 L'Interface

Cette partie contient tous les fichiers qui serviront à l'élaboration du menu. Elle contient l'ensemble des menus qui s'afficheront à l'écran. Ainsi seront composés les fichiers suivants :

- Le fichier Accueil : qui est le menu d'accueil, s'affichera au démarrage de l'application. Il permet au consultant de se connecter ou de créer un compte.
- Le fichier bienvenue : il s'affichera lorsque nous lancerons l'application.
- Le fichier centreinteret : il affiche les différentes questions qui vont permettre de renseigner les centres d'intérêt de l'utilisateur.
- Le fichier compte_authentification : il affiche le menu de connexion.
- Le fichier compte_creation : il affiche le menu pour créer son compte.
- Le fichier menu_detail_recherche : il affiche les différents critères de recherches à choisir pour effectuer la recherche.
- Le fichier menu_rendezvous : il permettra à l'utilisateur de choisir quel type de rendez_vous vous souhaitez proposer.
- Le fichier menuAccederAsonProfil : il affiche le menu pour accéder à son profil.
- Le fichier MenuAccederMessagerie : il affiche le menu de la messagerie.
- Le fichier menuAccederOngletMatch : il affiche le menu pour rechercher des matchs.
- Le fichier menuModifierprofil : Il affiche les choix possibles de modification du profil.

3.3.2 La couche Business Model

Elle sera composée des fichiers suivant : le fichier Compte, detail_recherche, Interet, like, messagerie et rendezvous.

3.3.3 La partie Data Access

La partie Data Access contiendra deux dossiers : le dossier DAO, et le dossier service.

- le DAO : le dossier DAO comportera l'ensemble des classes dont les méthodes permettront à l'application d'accéder aux bases de données SQL et d'y effectuer des actions ou modifications.
- Service : le dossier Service contient la classe compte_service qui possède des méthodes faisant appel aux classes DAO.

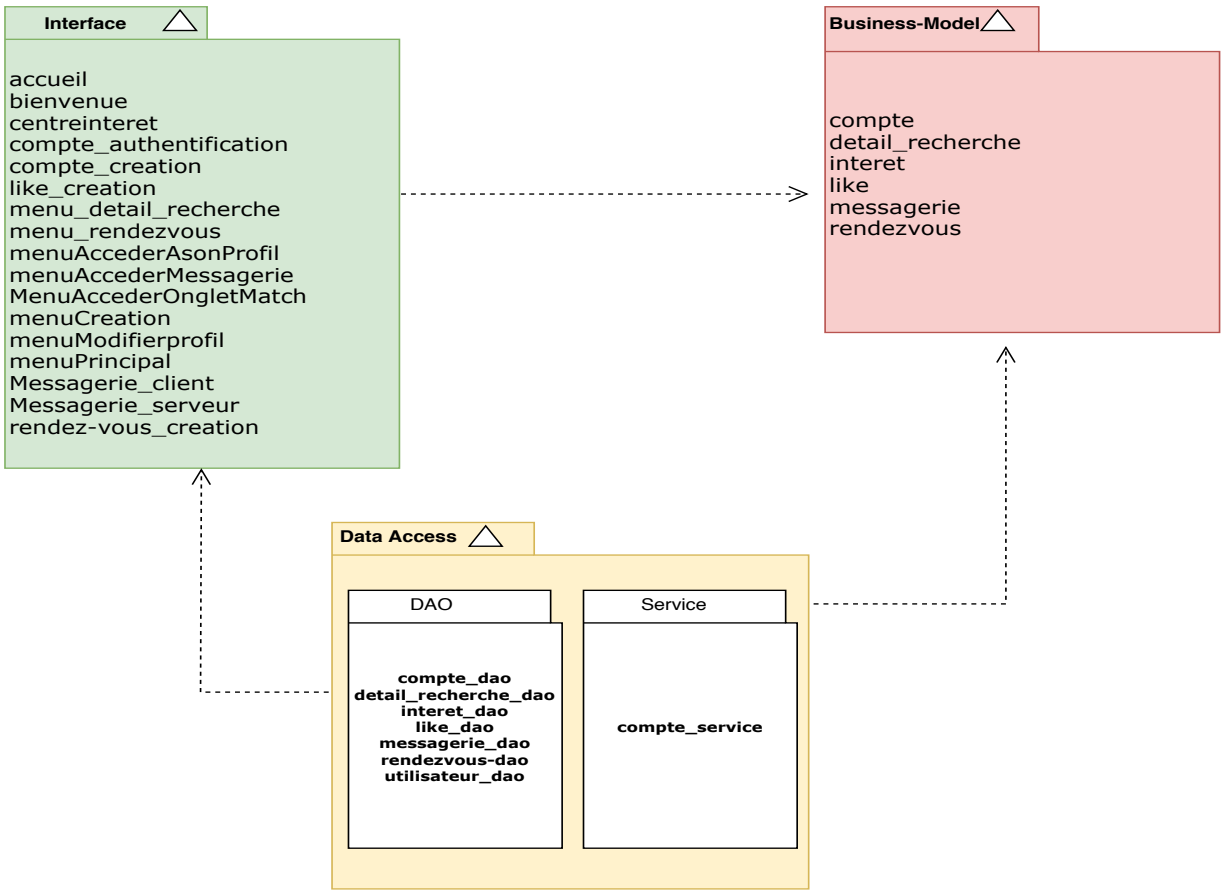


FIGURE 3.1 – Diagramme de package

Chapitre 4

La persistance des données

Comme introduit précédemment, dans ce projet, nous avons travaillé sur la persistance des données. En effet, ultérieurement nous avions l'habitude d'utiliser une application orientée objet où toutes les données existantes dans la mémoire vive de l'ordinateur sont perdues à la fermeture de la session de l'utilisateur. Ici nous avons sauvegardé toutes nos données sur un support non volatile, des bases de données relationnelles, de telle sorte qu'à l'ouverture de la session de l'utilisateur les données puissent être recréées. Elles peuvent alors être modifiées ou manipulée via notre application de rencontre, localement, sur la mémoire vive de l'ordinateur. De plus de nouvelles données peuvent être créées à condition qu'elles respectent le format de nos bases de données. Enfin, une fois que l'utilisateur ferme sa session les données sont de nouveau rendues persistantes. Toutes les modifications, suppressions ou créations sont donc sauvegardées à chaque fermeture de l'application. Plus concrètement, ici, les données manipulées, créées ou supprimées sont principalement les like, les messages et les comptes.

4.1 La créations des bases de données

Nous avons décidé de créer six bases de données. Une pour chaque type d'objet important :
— les comptes via la base de donnée "utilisateur",
— les like et les match via la base de donné "liker",
— les messages via la base de données "message",
— les rendez-vous via la base de données "rendezvous",
— les centres d'intérêt via la base "interet", et enfin,
— les détails sur les critères de l'utilisateur en terme de recherche via la base "detail_recherche".

Pour mieux visualiser les bases de données nous avons réalisé un diagramme. Nous observons que notre diagramme est assez simple : la base de données utilisateur est au centre, c'est l'id_user qui est l'attribu omniprésent dans nos bases. En effet cet identifiant se retrouve en clé étrangère dans toutes nos bases de données. Il permet d'associer chaque message au compte de la personne qui l'envoie et au compte de la personne qui le reçoit. Il permet de relier chaque like à la personne qui like et à la personne qui a été liké. Aussi, il permet de relier un rendez-vous donné à l'iniciateur de ce rendez-vous et à l'invité. Enfin, il permet de relier la recherche détaillée et les centres d'intérêt au bon utilisateur afin d'assurer la précision de notre application de rencontre.

Une description de chaque attribut est également fourni ici pour la classe Utilisateur détaillant chaque nom, description, type et contraintes associés. Vosu pouvez retrouver en annexes C.1 le même

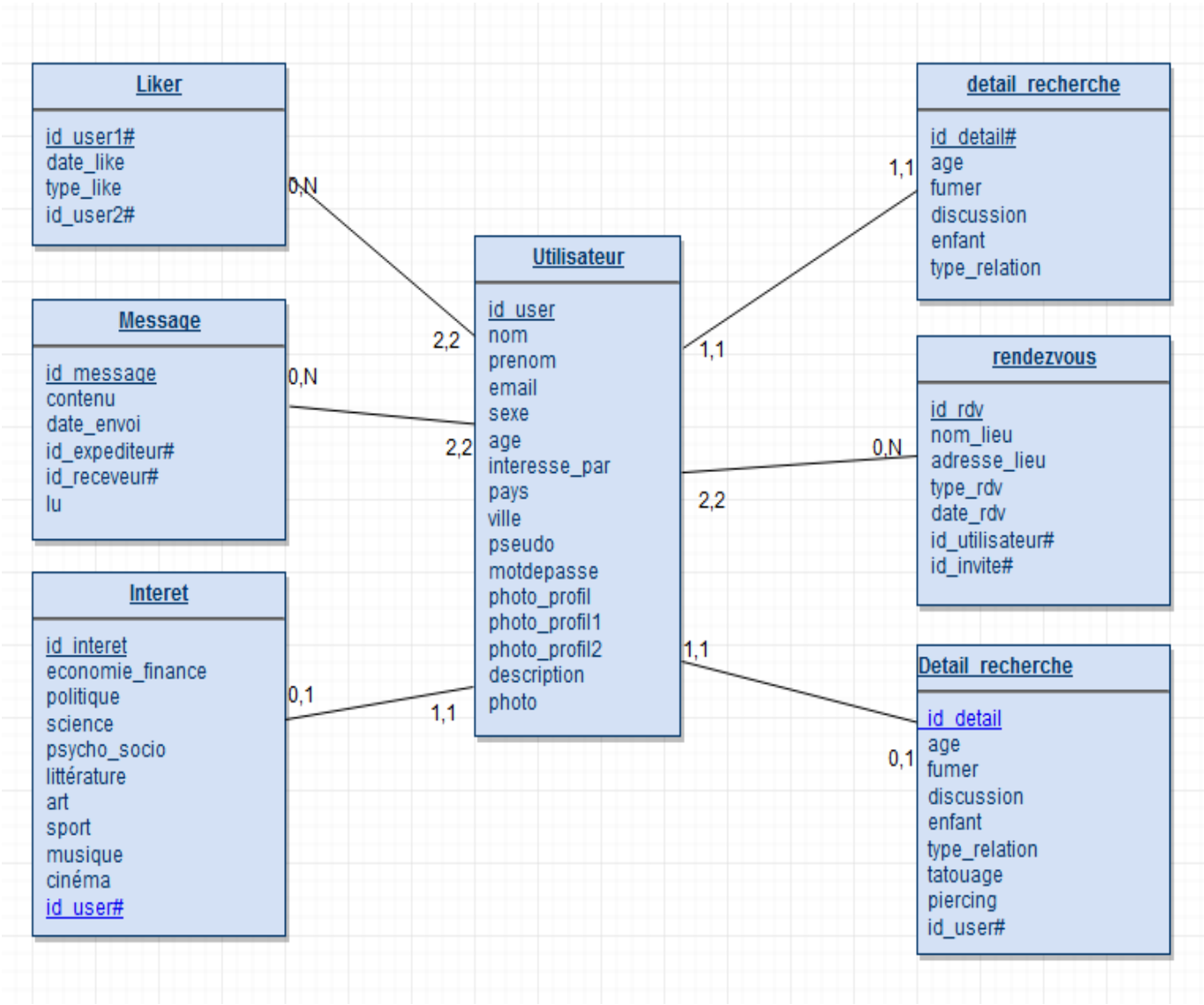


FIGURE 4.1 – Diagramme du modèle de base de données

détail pour les trois autres base de données.

Chacune de nos bases de données ont été créées sur PosteGre à partir de requêtes SQL. Ces requêtes sont jointes en annexes. On peut oberser qu'elles sont toute codées d'après une même structure : premièrement on indique le nom de la base de données, ensuite on fait la liste de chaque attribut en indiquant leur type et spécifiant des contraintes si nécessaire. Enfin, on indique la clé primaire et les clés étrangères en précisant la base de données de référence.

Ci-dessous se trouve la requête SQL utilisée pour créer la classe Utilisateur. Vous pouvez retrouver en annexe C.2 et C.3 les requêtes utilisées pour les trois autres base de données.

Nom de l'attribut	Description de l'attribut	Type	Contrainte
UTILISATEUR			
id_user	identifiant de l'utilisateur	int	NOT NULL
nom	nom de l'utilisateur	varchar	NOT NULL
prenom	prénom de l'utilisateur	varchar	NOT NULL
email	adresse email de l'utilisateur	varchar	NOT NULL
sexe	sexe de l'utilisateur	int	NOT NULL
age	âge de l'utilisateur	int	>= 18
interesse_par	orientation sexuel de l'utilisateur	int	NOT NULL
pays	pays de résidence de l'utilisateur	varchar	NOT NULL
ville	ville de résidencde de l'utilisateur	varchar	NOT NULL
pseudo	pseudo choisi par l'utilisateur	varchar	NOT NULL
motdepasse	mot de passe choisi par l'utilisateur	varchar	NOT NULL
photo_profil	photo de profil choisi par l'utilisateur	varchar	NOT NULL
photo_profil1	deuxième photo de profil choisi par l'utilisateur	varchar	
photo_profil2	troisième photo de profil choisi par l'utilisateur	varchar	
description	description écrite par l'utilisateur	varchar	

FIGURE 4.2 – Tableau des attributs de la base de données Utilisateur

```
CREATE TABLE utilisateur(
id_user SERIAL,
nom VARCHAR(20) not null,
prenom VARCHAR(20) not null,
email  VARCHAR(50) not null,
sexe integer not null,
age integer constraint val_age CHECK (age >= 18),
interesse_par integer not null,
pays VARCHAR(20) not null,
ville VARCHAR(20) not null,
pseudo  VARCHAR(20) not null,
motdepasse  VARCHAR(20) not null,
photo_profil  VARCHAR(200) not null,
photo_profil1  VARCHAR(200),
photo_profil2  VARCHAR(200),
description  VARCHAR(500),
PRIMARY KEY (id_user)
);
```

FIGURE 4.3 – Requête SQL pour créeer la base de données Utilisateur

4.2 Le remplissage des bases de données

Nous avons intégrées des données à nos bases de données uniquement via notre application python. Pour cela nous avons créé un code qui permet d'accéder aux donnéees et nous l'avons placé dans une couche d'accès aux données : la couche DAO. Notre architecture inclue ainsi un dossier

nommé DAO contenant nos fichiers avec ce code d'accès. Cette couche a de nombreux avantages. Elle permet tout d'avord d'ajouter un niveau d'abstraction entre les sources de données et leur utilisation. Elle garantit alors une liaison faible entre la couche métier et la base de données afin d'éviter des erreurs qui affecterai ces deux derniers éléments. La couche DAO permet aussi une simplification du code de la couche métier qui ne contiant aucune requête SQL, aucun code de connexion et pas de chaîne de connexion à la base de données.

Notre dossier DAO comporte sept fichiers. Dans ces sept fichiers sont codées une classe abstraite, quatre classes soit une part base de données et une fonction de connection. Premièrement on observe la fonction connection, indispensalbe pour accéder aux bases de données. Celle-ci utilise le package psycopg2 qui permet de faire le lien à notre application postegre à l'aide du fichier *properties* qui contient l'identifiant et le mot de passe de postegre. La classe abstraire initialise donc en premier lieu l'obejet *connection* qui lance la fonction de connection retournant les bases voulues.Dans un second temps, la classe abstraite permet de définir cinq méthodes de base pour les cinq actions de bases de la couche de persistance : *find_by_id*, *find_all*, *update*, *create*, *delete*. Ces méthodes sont vides, elle permettent simplement d'être hérité afin d'uniformiser notre code. Ainsi les quatre autres classes héritent de cette classe incluant l'objet *connection*. Elles contiennent chacune des méthodes répondant aux besoins particuliers en terme de manipulation de données de chaque fonctionnalité de la classe correspondante en couche métier. Par exemple, la classe DAO reliée à la base de données Utilisateur contient une méthode *Voirprofil* qui permet simpliment de récupéré toutes les données de la base pour les afficher proprement.

Chapitre 5

Fonctionnalités de l'application

5.1 Présentation des menus

Dans l'annexe **B.1** vous pouvez voir une représentation explicite du menu que nous souhaitons concevoir. Ce **diagramme de menu** représente les différentes possibilités d'actions d'un utilisateur connecté.

Après la connexion, les utilisateurs accèdent au menu principal qui propose 4 choix d'action. Par exemple, si un utilisateur souhaite mettre à jour son profil, il choisira dans un premier temps "Accéder à son profil", puis "Modifier le profil" et choisira pour finir l'élément qu'il souhaite modifier. A noter qu'il est possible, lors de chaque étape, de retourner au menu principal (ce qui n'est pas indiqué sur le diagramme).

Pour finir, l'utilisateur qui n'est pas encore connecté n'a bien sûr pas l'accès à ce menu principal. Il n'aura que deux choix possibles sur l'application, il pourra soit **créer un compte** s'il n'en possède aucun, soit **s'authentifier** pour ensuite accéder au menu principal.

5.2 Difficulté et criticité des fonctionnalités

Toutes les fonctionnalités de notre application ont été plus ou moins difficiles à coder, mais sont aussi plus ou moins importantes pour notre application. Par exemple, comme on peut le voir dans le tableau ci-dessous, l'authentification pour un utilisateur était la base pour notre application, et sa difficulté est faible. Nous avons donc privilégié cette fonctionnalité lors de la création du code. Ce tableau va nous permettre de savoir quelle fonctionnalité nous avons codé en priorité.

En effet, nous nous sommes dans un premier temps nous concentrés sur les fonctionnalités avec une forte criticité et une faible difficulté, c'est à dire la création d'un compte, l'authentification d'un utilisateur et la création de relations entre utilisateurs (like et match). Les fonctionnalités que nous considérions moins prioritaires sont celles avec une faible criticité et d'une grande difficulté. Un exemple de ces fonctonnalités peut être la création d'une messagerie instantannée par exemple.

Tâches	Criticité	Difficulté
Créer un compte	4	1
Authentification d'un utilisateur	4	1
Consommation d'une API de lieux pour la prise de rendez-vous	3	2
Création de relation entre les membres connectés	3	2
Création d'un système de sauvegarde de la base de données	3	2
Permettre à l'utilisateur de proposer un rendez vous près de sa position actuelle	3	3
Création d'un système de messagerie instantannée	3	3
Recupération de l'historique de messagerie en format JSON	2	2
Création d'un système de notifications	2	2

TABLE 5.1 – Classement des fonctionnalités en fonction de leur criticité et difficulté

5.3 Diagramme d'activité

5.3.1 Diagramme d'activité pour la création de compte

L'une des premières fonctionnalités que pourra effectuer un utilisateur non connecté sur l'application sera de se créer un compte. Cette fonctionnalité est décrite dans l'annexe **D.1**.

Lorsque l'utilisateur non connecté arrivera sur l'application, la classe **Accueil** va lui afficher un menu pour lui donner la possibilité de se connecter ou de se créer un compte. S'il n'a pas de compte, il va donc choisir de s'en créer un. Cela va faire appel à la classe **CreationCompte** du fichier **vue** qui demandera à l'utilisateur de taper son psoeudo et mot de passe. Ensuite, la méthode psoeudo_disponible de la classe **CompteService** va faire appel à la méthode fins_by_psoeudo de la classse **compte_dao**. L'application va ainsi pouvoir vérifier si le psoeudo proposé par l'utilisateur n'est pas déjà utilisé :

- Si les coordonnées correspondent à un compte déjà existant, il sera alors demandé à l'utilisateur de retaper de nouvelles coordonnées (psoeudo et mot de passe).

- Sinon, la méthode creer_compte de l'instance de classe **CompteService** va créer une instance de classe **Compte** avec le psoeudo et le mot de passe en attributs, puis va faire appel à la classe **CompteDao** afin d'enregistrer le compte dans la base de données. Pour finir, l'instance de la classe compte sera enregistrée dans l'attribut user de la classe **session**, puis le menu principal de l'application sera affiché. L'utilisateur sera ainsi considéré comme connecté.

5.3.2 Diagramme d'activité pour rechercher de nouveaux matchs

L'utilité première de notre application est de pouvoir faire de nouvelles rencontres. L'utilisateur pourra donc faire des recherches de profils qui peuvent potentiellement l'intéressé.

Dans l'annexe **D.2**, on voit qu'avant de lancer la recherche, l'utilisateur doit préciser ses critères de recherche. Une fois fait, il pourra ainsi lancer la recherche. Cela va enregistrer l'orientation sexuelle de l'utilisateur dans une variable, puis va lancer la méthode **Voirprofil_autre** de la classe **compte_dao** avec la variable de l'orientation sexuelle en attribut. Cette méthode va sélectionner les profils dont le sexe correspond au sexe de l'orientation sexuelle de l'utilisateur. De plus, elle choisira des profils qui n'ont pas encore été matché ou liké par notre utilisateur (bien sûr si l'autre l'a déjà liké, cela ne posera pas de problèmes). La méthode va ensuite ranger tous ces profils dans

une liste. Enfin, l'application va afficher un premier profil parmi ceux de la liste.

- Si l'utilisateur est intéressé par ce profil, il va pouvoir choisir de mettre un like. Cela va permettre la création d'une instance de la classe **Like** avec en attributs les identifiants des deux utilisateurs, ainsi que la date. Ensuite la méthode **create** de la classe **LikeDao** sera appelée. Elle va ainsi enregistrer le like dans la base de données. Enfin, après avoir mis son like, l'utilisateur aura le choix de poursuivre sa recherche, ou bien d'y mettre fin, ce qui le redirigera au menu principal.

- Si l'utilisateur n'est pas intéressé par le profil, il pourra le passer. S'il souhaite poursuivre sa recherche (et s'il reste encore des profils dans la liste), un autre profil de la liste lui sera alors proposé. Sinon, il pourra mettre fin à sa recherche et revenir au menu principal.

Attention toutefois, il faut préciser que même s'il y a des like réciproques, les **matchs** ne sont pas encore crées. La création des matchs se fait au moment où l'utilisateur va vouloir consulter ses matchs. A ce moment, la méthode DAO va vérifier s'il existe des matchs réciproques, et les matchs seront ainsi affichés à l'utilisateur.

Chapitre 6

Partis-pris

6.1 Le hachage des mots de passe

Il se peut que un individu mal intentionné infiltre notre base de données et accède aux données des utilisateurs. Pour pallier cela, nous avons protégé les mots de passe des utilisateurs en les cryptants.
Nous avons utilisé une fonction de hachage. Cette fonction va associer à un message en entrée (le mot de passe), une valeur de hachage de longueur fixe. Ainsi, les mots de passe ne seront pas stockés en clair dans notre base de données.

Dans python, cette fonctionnalité se traduit par la classe `ChiffrerMotPasse` qui possède une méthode `hash_password`, qui prend en paramètre la valeur d'entrée et la valeur de hashage. Cette méthode est appelée à chaque fois qu'un utilisateur crée un nouveau compte, autrement dit à chaque fois qu'un mot de passe est ajouté à la base de donnée.

6.2 Messagerie instantanée

Nous avons réussi à mettre en place un système de messagerie instantannée. Cela a été possible grâce à l'utilisation d'un serveur socket.

6.3 Le fonctionnement des différents menus

Nous avons décidé d'utiliser plusieurs modèles de menu. Certains fonctionnent avec des numéros, certains avec un curseur à pointer sur la fonctionnalité souhaité et certains avec un système de checkbox. Les principaux menus utilisent le système de curseur. Le système de numéro est utilisé dans la boîte de réception et le système de checkbox est utilisée pour remplir la partie détail de recherche. Présenter différents menu permet de rendre l'application maniable par tous, ce qui était l'un de nos premier objectif.

Chapitre 7

Guide d'utilisation

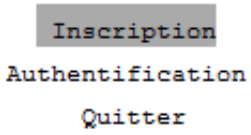
Bienvenue sur **You&Me**, l'application indispensable qui vous permettra de trouver (peut-être) l'amour !

Ce guide vous permettra d'utiliser au mieux **You&Me** pour vous préparer au mieux à rencontrer l'âme soeur.

Tout au long de votre experience avec **YouMe**, lorsque qu'il vous sera proposé de quitter l'application, vous répondrez *Oui* = *Y* pour fermer l'application, et *Non* = *n* pour revenir au menu précédent.

7.1 Lancement de l'application

Au lancement de **You&Me**, un joli écran vous accueillera en vous souhaitant la bienvenu. Vous serez alors invité à vous inscrire, en selectionnant le choix `Inscription` dans le menu principal.



7.2 L'inscription

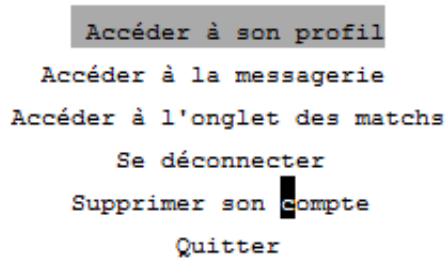
Pour utiliser **You&Me**, vous devez être inscrit. Il vous est donc demandé de renseigner un formulaire comprenant votre `nom`, votre `prénom`, votre `e-mail`, votre `age`, votre `pays`, votre `ville`, votre `pseudonyme`, votre `mot de passe` (champs ouverts), de séléctionner votre `sexe` (question fermée binaire). Il vous est aussi demandé de renseigner votre `préférence` : votre cible d'intérêt (homme, femme, ou les deux).

7.3 L'authentification

Une fois inscrit, vous pourrez vous authentifier dans le menu principal. Le choix `Authentification` vous renvoi vers un menu dans lequel il vous est demandé d'entrer votre pseudo et mot de passe. Si votre authentification est réussie, vous serez alors redirigée vers le menu des fonctionnalités.

7.4 Fonctionnalités

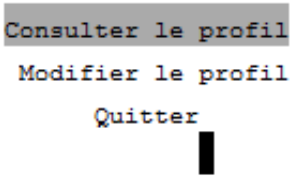
Félicitations ! Vous être maintenant authentifié, vous pouvez profiter pleinement des fonctionnalités de l'application.



7.4.1 Accéder à mon profil

Avant de se lancer pleinement dans la recherche de votre âme soeur, il vous faudra d'abord pimper votre profil. Pour cela, faites le choix `Accéder à mon profil`. Vous arriverez alors sur le menu de votre profil. Depuis celui-ci, vous pourrez consulter ou modifier votre profil.

En sélectionnant `Modifier votre profil`, vous aurez la possibilité de modifier votre description, centres d'intérêts et photo de profil séparément, ou alors de modifier l'entiéreté de votre profil d'un coup.

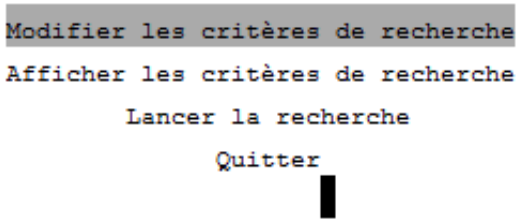


7.4.2 Accéder à la messagerie

Ca y est ! Vous être désormais paré pour partir chasser de la chair fraiche. Rendez-vous dans le menu `Accéder à la messagerie`. Vous aurez alors la possibilité de consulter vos matchs, d'accéder à votre boite de réception, de proposer un rendez-vous à l'un de vos matchs et d'exporter la messagerie au format Json.

Lorsque vous proposerez un rendez-vous à l'un des pseudos renseigné, il vous sera proposé de choisir

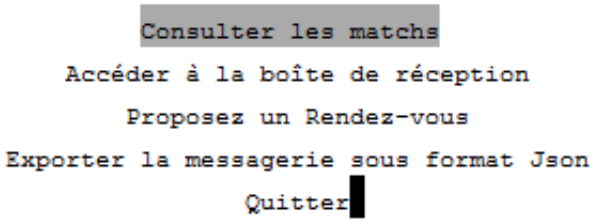
un type d'activité parmis `parc, drink` ou `food`. Ensuite, il vous sera demandé de choisir un lieu à proximité duquel établir le rendez-vous. Après validation, il vous sera proposé un liste de lieu qui correspondent à vos critères. Il ne vous restera qu'a en choisir l'un d'entre eux pour que votre proposition de rendez-vous soit validée.



7.4.3 Accéder à l'onglet des matchs

Avant de pouvoir prétendre à proposer un rendez-vous, il vous faut d'abord un/une prétendante. Cette étape d'une relation s'apelle le game. Autrement dit, il va vous falloir choisir une cible et soigner votre approche.

Pour cela, rendez-vous dans l'onglet des matchs. Vous avez alors deux possibilités : modifier les critères de recherche et lancer la recherche. Les critères de recherche sur lequels vous allez pouvoir agir sont le profil recherché (l'age et le sexe) ainsi que la portée de recherche.
Et enfin, une fois cette étape réalisée, vous pourrez lancer la recherche de match. Je ne vous expliquerez pas comment cet outil fonctionne, je sais que vous le connaissez à la perfection.



Chapitre 8

Conclusion

Après avoir abordé notre étude du sujet, la conception de l'application, le codage sur les trois couches structurant notre application, notre couche DAO permettant la persistance des données, les spécifités de notre programme et le guide d'utilisation de notre application, nous allons maintenant décrire les choses que l'on aurait aimer ajouter à ce projet (ou qu'on aurait souhaiter mieux faire !). Après ça, nous vous invitons tout de même à tester cette application et pourquoi pas à essayer de faire des rencontres.

8.1 Pistes d'amélioration

8.1.1 Critères de recherches

Actuellement, lorsque nous accédons à l'onglet des match et que nous lançons la recherche de profils, l'application nous affiche des profils de manière aléatoire, en se basant seulement sur le sexe de la personne et de son orientation sexuelle. Par exemple, si une femme est intéressée seulement par d'autres femmes, l'application va lui proposer des profils de femmes qui sont également intéressées par des femmes, ou qui sont intéressées par des hommes et des femmes.

Nous aurions aimé pouvoir utiliser les centres d'intérêt ainsi que les critères de recherche de match pour sélectionner les profils potentiels qui auront le plus de chance de plaire à l'utilisateur (en utilisant une ACM sur les variables qualitatives par exemple).

8.1.2 messagerie instantanée entre plusieurs ordinateurs

Une autre piste d'amélioration concerne la messagerie instantanée. En effet, notre messagerie instantanée actuelle nous permet de communiquer sur deux consoles différentes d'un même ordinateur. Nous aurions aimé pouvoir communiquer de manière instantanée mais entre plusieurs ordinateurs connectés simultanément à l'application.

8.1.3 Gestion à partir d'un administrateur

Nous aurions également aimé pouvoir coder des fonctionnalités telles que "mot de passe oublié", "signaler un compte". Cette dernière option aurait été gérable en ajoutant une classe et un profil administrateur qui pourrait observer les signalements, les vérifier et supprimer les comptes concernés si nécessaire.

Pour conclure ce rapport de projet nous avons pensé que le plus intéressant était de vous présenter le retour personnel de chaque membre de notre groupe.

8.2 Note personnelle : Adrien FERNANDEZ

Ce projet d'informatique nous semblait à première vue similaire à celui de l'année dernière, mais les outils que nous devions utiliser et la taille du groupe étaient différents. Tout d'abord, j'aimerai être honnête et préciser que l'informatique n'est pas mon domaine de prédilection et ne m'intéresse pas particulièrement. Cependant, je trouve tout de même intéressant de s'immerger dans un projet tel que celui-là pour plusieurs raisons.

Points positifs :

La première raison est **l'aspect organisationnel** au sein du groupe. Le groupe étant formé de manière aléatoire (même si supposée être en fonction des niveaux, sélection qui d'ailleurs se base sur des notes, donc n'est pas forcément bien représentative du niveau de la personne ou de son investissement), il a fallu s'adapter en fonction des autres, c'est-à-dire en fonction de leurs compétences en informatique, de leur manière de travailler. De plus, nous pouvions parfois être en désaccord entre membres, il était donc important de garder une bonne cohésion au sein du groupe afin de travailler de manière optimale.

La deuxième raison est **la découverte de nouveaux outils** tels que GitLab ou PyCharm et la pratique du langage python (langage indispensable pour notre future profession). Je trouve que ces outils sont très intéressants et je suis plutôt satisfait d'avoir pu les découvrir.

Les difficultés rencontrées :

Notre principal problème rencontré est le problème avec le package **PSYCOPG2** qui était indispensable pour faire fonctionner nos méthodes DAO, et ce problème nous empêchait de RUN le code. Par conséquent, après avoir demandé maintes fois de l'aide à notre chargé de TD et aux autres élèves (personnes que je remercie d'ailleurs pour leur aide), le problème réapparaissait régulièrement et nous nous sommes en quelques sortes résignés à vivre avec (car oui il fallait bien avancer et coder quelque chose !). Nous étions alors 4 personnes sur 5 à ne pas pouvoir RUN le code, ce qui était très, mais alors très très très handicapant pour notre projet comme vous pouvez l'imaginer.

Au bout d'un moment, nous commencions malheureusement à nous décourager et à perdre petit à petit notre motivation, mais nous nous sommes ressaisit. Dans ces conditions, nous avons donc opté pour une répartition du groupe par binôme pour coder à l'aveugle, c'est-à-dire sans moyen de pouvoir tester si notre code fonctionne ou non. Puis, nous envoyions notre « code brouillon » à Caleb, notre seul membre du groupe qui pouvait RUN le code. Je tiens d'ailleurs à souligner que Caleb s'est beaucoup investi dans le codage de l'application, dû à cette situation.

Un autre point négatif est que nous avons peu utilisé GitLab car nous avions des difficultés à comprendre son fonctionnement, donc nous avons préféré partager notre code via le dossier projet qui est partagé (cela nous était plus intuitif et plus rapide). En effet nous avions conscience que ce n'était pas la méthode demandée mais le problème précédent nous a beaucoup retardé sur notre organisation, donc le temps courrait contre nous !

Un autre problème rencontré était le temps de connexion sur nos machine virtuelle. Je pense que vous le savez déjà car tout le monde a dû en parler, mais bon, même s'il est évident, il ne faudrait

pas oublier de rappeler ce problème.

Conclusion :

On peut donc dire que le cadre de ce projet n'était pas idéal et je trouve cela bien dommage car, comme je l'ai précisé au début, je pense que ce projet aurait pu être très intéressant. Même si par moment je ressentais une certaine forme de satisfaction lorsque je constatais que notre application commençait à fonctionner (en fait c'est sympa l'informatique !), les conditions m'ont globalement fait garder un petit goût amer vis à vis de ce projet, un goût d'inachevé, un goût de « on aurait pu mieux faire si ... ».

8.3 Note personnelle : Lucie BOUIN

Je vais essayé d'exprimer au mieux mon ressenti lors des différentes étapes de notre projet.

Appréhension du projet

Tout d'abord, notre premier voeu de sujet nous a été attribué. Un sujet intéressant est la première source de motivation selon moi. Coder une application de rencontre m'interessait parce que ce type d'application est très répendu. Voir les dessous d'une application de ce genre et pouvoir l'agrémenter avec nos idées me semblait être une belle opportunité. Nous avons alors commencé l'étude préalable sans difficultés.
Pour la partie technique du sujet mes appréhensions étaient plus grande. L'utilisation du nouveau logiciel gitlab et d'une API en plus de la structure en couche m'ont semblé être de grandes nouveautés et un grand challenge. Cependant j'avais déjà codé sur le logiciel PyCharm qui m'était donc familié. Concernant le groupe, j'ai vite remarqué qu'une importante organisation serait nécessaire étant donné que nous étions cinq et que le niveau en informatique semblait très hétérogène.

Le travail de groupe

Le travail de groupe s'est bien passé, nous arrivions facilement a tous s'exprimer et à trouver un terrain d'entente lors de prises de décisions. J'aurai aimé que la répartition des tâches soit équitable entre le codage et la rédaction du rapport mais malheureusement nos niveaux très différents en informatique ne l'ont pas permis. Bien que j'ai codé certaine partie, je n'avais pas encore assimilé tout le squelette au début du codage. Parallèlement à mon travail mes coéquipiers ont rapidement pris en main la couche DAO et la partie métier au point qu'une messagerie utilisant socket a été mise en place. J'ai donc rapidement pris un rôle d'appui : j'ai pu aider à coder tout en guidant sur la forme et le contenu des menus.

Les difficultés rencontrées

Comme tous les groupes, les principales difficultés rencontrées sont liées au matériel mis à notre disposition. Les temps d'attente de chargement, les erreurs non identifiées sur Git Bash et le package psycopg2 nous ont fait perdre beaucoup de temps et de motivation. Nous avons d'ailleurs dû nous résigner à ne plus utiliser Git Lab ce qui est vraiment dommage. Nous nous sommes aperçus que, malgré plusieurs résolution du problème lié package psycopg2, le problème revenait après chaque pull. Ainsi seul un de nos coéquipiers avait la possibilité de lancer le code pour vérifier les fonctions. La version à jour du code était accessible à tous sur le dossier de groupe partagé. Bien que demandant une organisation plus grande ce fonctionnement nous à permis de ne plus perdre de temps sur les problèmes de Git Lab et de package.
D'un point de vue personnel la plus grande difficulté a été d'être confrontée a certaines lacunes majeures en informatique que j'ai accumulé. Le fonctionnement cahotique des TP en Python orienté objet de l'année dernière et de complément informatique en début d'année a été difficile à palier par le travail personnel. Malgré mon enthousiasme j'ai passé beaucoup plus de temps que la normal à faire de petite chose ce qui est assez démoralisant. Heureusement notre avancement en groupe m'a permis de continuer à contribuer au projet.

Les points positifs

Ce projet a, malgré ces dernières difficultés rencontrées, été intéressant et riche. J'ai maintenant une notion plus précise du fonctionnement d'un projet informatique construit avec une structure en

couche, très fonctionnelle. J'ai aussi découvert le logiciel Git qui sera sûrement réutilisé à l'avenir et dans le cadre scolaire et dans un milieu professionel. J'ai aussi agrandi mes connaissances en codage orienté objet : j'ai davantage assimilé l'intérêt de classe abstraite et le fonctionnement d'héritage.

Conclusion

Bien que la frustration nous a suivi tout au long de ce projet nous sommes parvenu a créer une application de rencontre qui fonctionne. Travailler avec mes coéquipiers ayant d'importants savoirs-faire en informatique a été appréciable et enrichissant.

8.4 Note personnelle : Samuel GOUTIN

L'organisation

Ce projet fut le premier de notre formation à être réalisé en plutôt grand effectif (5 personnes). Cela demandait donc une organisation adaptée en conséquent : La décision de l'organisation à adopter était le résultat de longues discussions durant les séquences de projet. Durant ces discussions, nous prenions le temps d'échanger nos idées en considérant toutes celles qui étaient porposées. D'un côté, cette manière de procéder nous a demandé beaucoup de temps pour construire nos diagrammes et se lancer pleinement dans la programmation. Mais d'un autre côté, cela nous permettait de choisir un plan d'étude qui convient à tout le monde.

Impossible de travailler dans de bonnes conditions

Une fois l'organisation determinée, nous avons fait face à des problèmes techniques qui ralentissaient notre avancée. D'abord, il y a eu les problèmes de lenteur des machines virtuelles (entre 1h et 2h pour lancer sa session). Cela a raccourci considérablement notre productivité, car les séances de travail étaient plus courtes. Etant très consommateur pour nos machine, `Pycharm` prenait environs 30 minutes à se lancer et à se mettre à jour. Ajoutons à cela que nous ne pouvions pas executer notre projet depuis notre ordinateur personnel (sans passer par la VM), car Pycharm était trop complexe à configurer. Lorsque nous parvenions enfin à avoir un environement de travail fonctionnel, les soucis n'étaient pas terminés pour autant. A cause d'une erreur de package, le projet ne s'executait que sur une seule de nos 5 sessions. A cause de ces problèmes, notre avancée fut très lente et nous avons perdu de nombreux jours de travail.

Pour pouvoir participer convenablement au projet malgré l'impossibilité de pouvoir coder convenablement, nous avons pris certaines habitudes. D'abord, nous codions de petite parties (sans compilation), puis nous les assemblions sur la session qui fonctionne. Ensuite, nous avons appris à travailler à plusieurs sur la même tâche (même celles pour lesquelles ce n'était pas nécessaire).

Apports personnels

A travers ce projet, j'ai pu me développer techniquement et humainement. Techniquement, j'ai découvert de manière plus approfondie le language python et l'utilisation d'API. J'ai également appris à utiliser un gestionnaire de versions comme git lab qui est un outil indispensable pour un datascientist qui souhaite s'orienter vers l'informatique comme moi. Humainement, j'ai appris, avec mes camarades, à m'adapter face à un problème (cf impossible de travailler dans de bonnes conditions) pour mener le projet à terme.

Conclusion

Ainsi, ce projet m'a permis de développer mes compétences techniques, et humaines. Bien que les problème rencontrés l'ai rendu pénible, ce travail nous prépare bien au monde de l'entreprise, car tout ne se passe jamais comme on nous l'explique dans les livres. Malgré tout, surmonter ces problèmes à plusieurs me permet de garder un bon souvenir de ce projet.

8.5 Note personnelle : Sidi TRAORE

Appréhension du projet

Très souvent, les projets de classes constituent à la fois une source d'angoisse et un très bon cadre d'apprentissage où l'on acquiert beaucoup de connaissances à la fois techniques et humaines. Ainsi, avant de commencer le projet, chaque fois que je lisais le sujet j'étais très inquiet à l'idée que moi et mes camarades de groupe ne puissions le mener à bien malgré l'ensemble des connaissances que nous aurions mobilisées. Notre sujet était de concevoir une application de rencontre. Pour cela, nous avions à mobiliser des connaissances en UML, en sql pour les bases de données et en python. Aussi la configuration du groupe faisait que j'étais amené à travailler avec des personnes que je n'avais pas l'habitude de côtoyer. Je ne connaissais pas leur humeur et leur degré d'engagement pour les projets de classe.

Vécu de l'expérience : bel esprit d'équipe

Dès l'entame du projet, j'ai pu voir à quel point mes camarades de groupes étaient très motivés. Chacun donnait le meilleur de lui-même pour que nous puissions avoir une application fonctionnelle avec toutes les fonctionnalités de base dans un premier temps, et dans un second temps que nous puissions aller au-delà de ces fonctionnalités basiques pour avoir une très belle application de rencontre. La phase analyse conceptuelle a pris plus de temps qu'il ne fallait à mon avis, mais par la suite j'ai pu comprendre à quel point il était nécessaire d'être d'accord sur le fonctionnement général de l'application.

Connaissances acquises

Le projet informatique a été pour moi un cadre d'apprentissage technique et humain. Ainsi sur le plan technique, tout d'abord j'ai pu améliorer mes connaissances en programmation orientée objet avec python. Ensuite, j'ai appris à utiliser une API (foursquare) pour la prise de rendez-vous, j'ai appris à utiliser de nouveaux modules comme PyInquirer pour la programmation du menu dans la console, PsychoPg2 pour les requêtes sql, PIL pour afficher les images dans la visionneuse d'image, STAMP pour envoyer des messages à une adresse mail depuis python. Aussi, J'ai amélioré mes connaissances en modélisation UML avec les différents diagrammes de classe, d'activité, de menu, de package et de base de données que nous avons réalisés. Enfin, J'ai appris à mieux structurer mon code en utilisant l'architecture, vue, service, dao, et business modèle. Sur le plan humain, le projet m'a permis de travailler, de discuter et de connaître plus amplement d'autres camarades de classe.

Difficultés rencontrées

Malgré le travail réalisé avec mes camarades, personnellement à l'instar des autres membres du groupe, j'ai rencontré beaucoup de difficultés avec pycharm et le module psychoPg2. J'étais dans l'incapacité de « débugger » les programmes que j'écrivais ou de les exécuter afin de savoir s'ils marchaient comme je le souhaiterais. En effet, chaque fois que je lançais l'exécution de mon code soit je recevais un message m'indiquant qu'il n'y avait pas d'interpréteur python ou je recevais une erreur m'indiquant que le module PsychoPg2 n'a pas été retrouvé. Pour pallier à ce problème, durant les vacances j'ai installé « pycharm » et Postegré SQL sur mon ordinateur personnel afin de pouvoir avancer sur le projet. Malgré cela, le problème persistait. Pendant longtemps, avec mes camarades de groupes, les réunions de groupe ou les suivis de projet se résumaient plus à résoudre les problèmes techniques de pycharm qu'à programmer et avancer sur la création de notre application de rencontre. Au bout de vaines tentatives, je me suis résigné à écrire des lignes de code sans les

exécuter. J'envoyais les programmes à Caleb pour qu'il exécute et me fasse un retour sur les erreurs observées.

Conclusion

Toutefois à la fin du projet informatique, je tire un bilan très positif. Dans un premier temps, je suis satisfait non seulement du personnel que j'ai pu réaliser dans le cadre de ce projet, mais aussi du travail que mes camarades ont pu réaliser. Dans un second temps, je suis très heureux de toutes les nouvelles connaissances que j'ai pu acquérir à travers ce projet. Enfin, je suis très heureux d'avoir travaillé avec Caleb Carlos, Adrien Fernandez, Lucie Bouin et Samuel Goutin. Sans oublier aussi Antoine Brunetti notre chargé de suivit de projet qui répondait toujours à nos mails et qui n'hésitait pas à nous aider chaque fois que rencontrions des problèmes techniques avec Pycharm ou PsychoPg2 .

8.6 Note personnelle : Caleb Carloss AGUIDA

Travailler ce projet informatique a été l'occasion pour moi d'asseoir les diverses connaissances acquises lors du cours d'Informatique et de pouvoir travailler au sein d'une équipe aussi intéressante qu'organisée, à qui j'adresse mes satisfactions.

Au terme de ce projet, à l'exception des problèmes techniques rencontrés qui nous ont fait perdre assez de temps (près de 70% du temps de projet), je me réjouis du travail abattu dans le peu de temps et des connaissances acquises (structurer un projet, webscraping, hachage , JSON...) ainsi que de l'apport de notre mentor.

L'objectif de pouvoir réaliser une application de rencontre qui marche a été pour chacun de nous un défi personnel tant sur le plan organisationnel (la compréhension du sujet, les méthodes d'approches, le choix et la définition des tâches critiques, la répartition des tâches, le respect du temps, l'élaboration du diagramme de Gantt) que sur le plan technique où nous avons appris à nous servir de github, d'une API...

La forte hétérogenéité en compétence au sein du groupe a constitué au départ un véritable goulot d'étranglement que nous avons pu surmonter par la suite. Le temps qui nous a été accordé par l'administration, bien que considérable, n'a pas été suffisant au vue de nos ambitions pour le projet. Il nous a fallu trouver des horaires supplémentaires.

J'ai beaucoup apprécié la qualité des discussions, les implications des uns et des autres pour la rédaction des divers rapports, la création des bases de données, les requêtages, la définition du diagramme de classe, l'intégration de l'API foursquare, la création d'un service de discussion instantanée, l'utilisation des sockets qui nous ont permis d'acquérir des notions sur les systèmes de réseaux informatiques, l'exportation de données sous format JSON...

Nous ne pensons pas avoir atteint tous nos objectifs, mais nous sommes satisfaits du travail accompli et des résultats atteints qui, pour ma part, sera très utile dans mon projet professionnel.

Annexe A

Organisation interne

FIGURE A.1 – Diagramme de Gantt

Annexe B

Conception générale de l'application

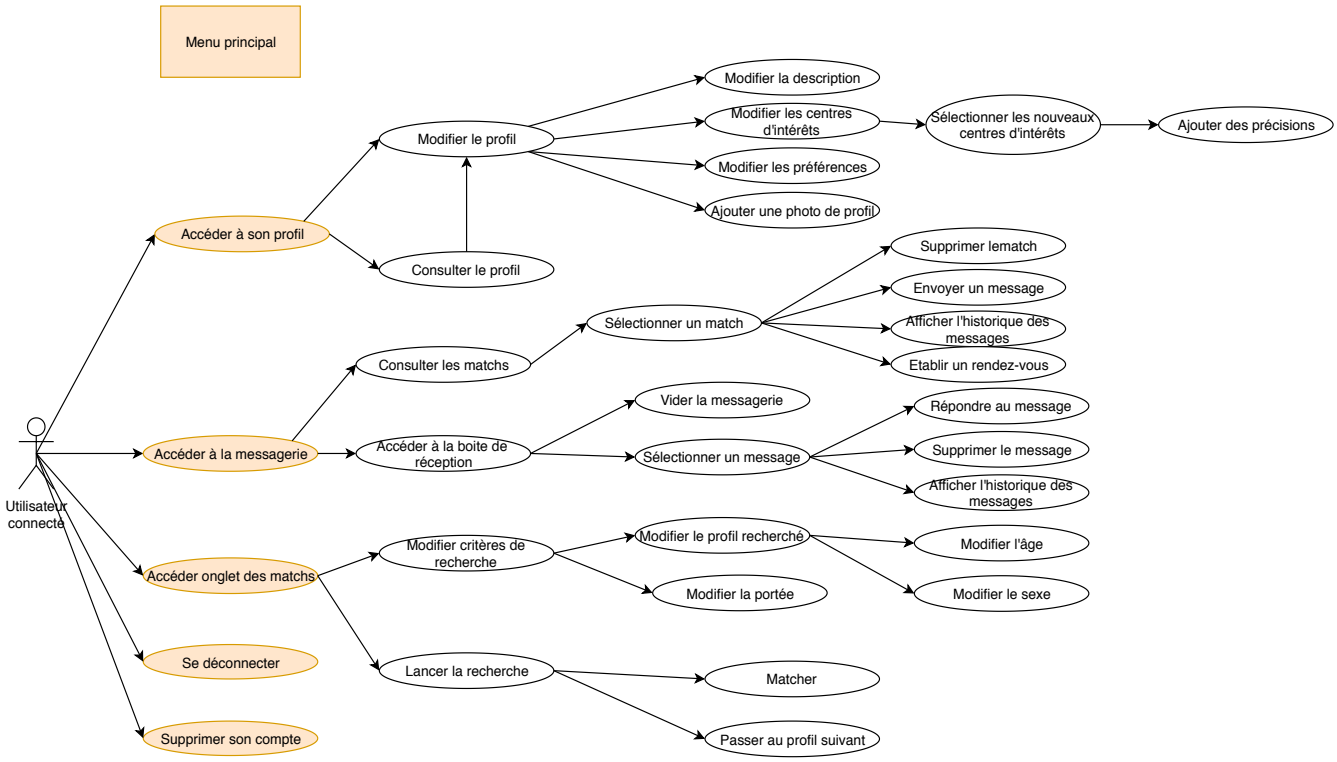


FIGURE B.1 – Diagramme de Menus

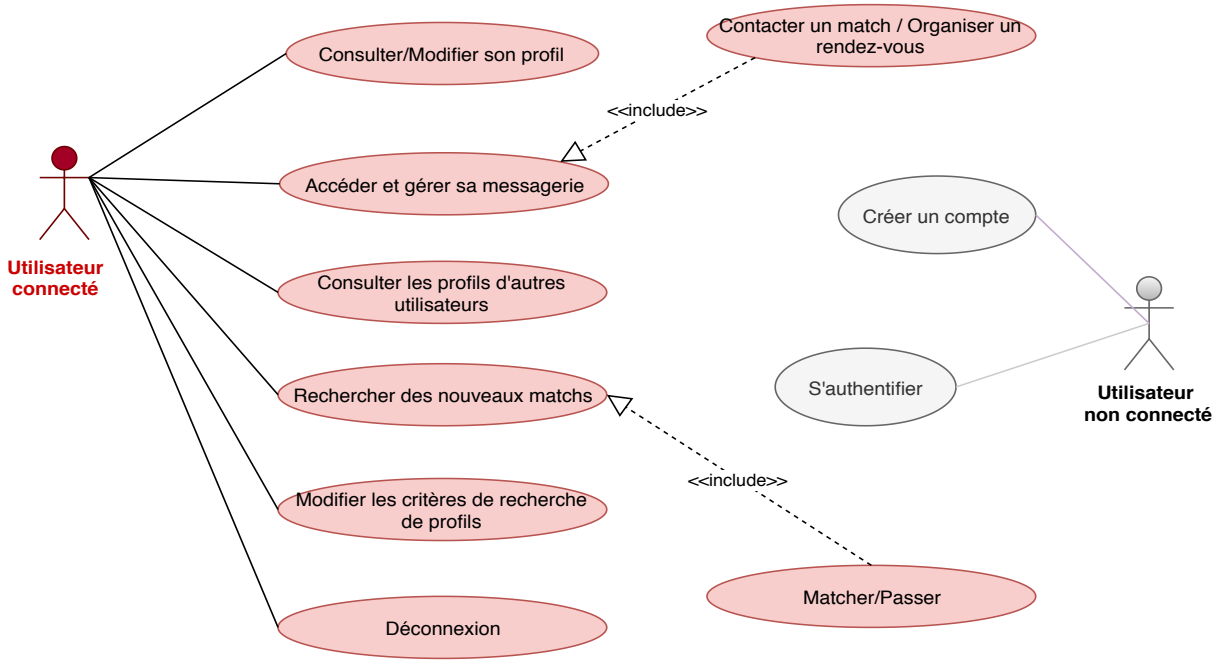


FIGURE B.2 – Diagramme de Cas d'Utilisations

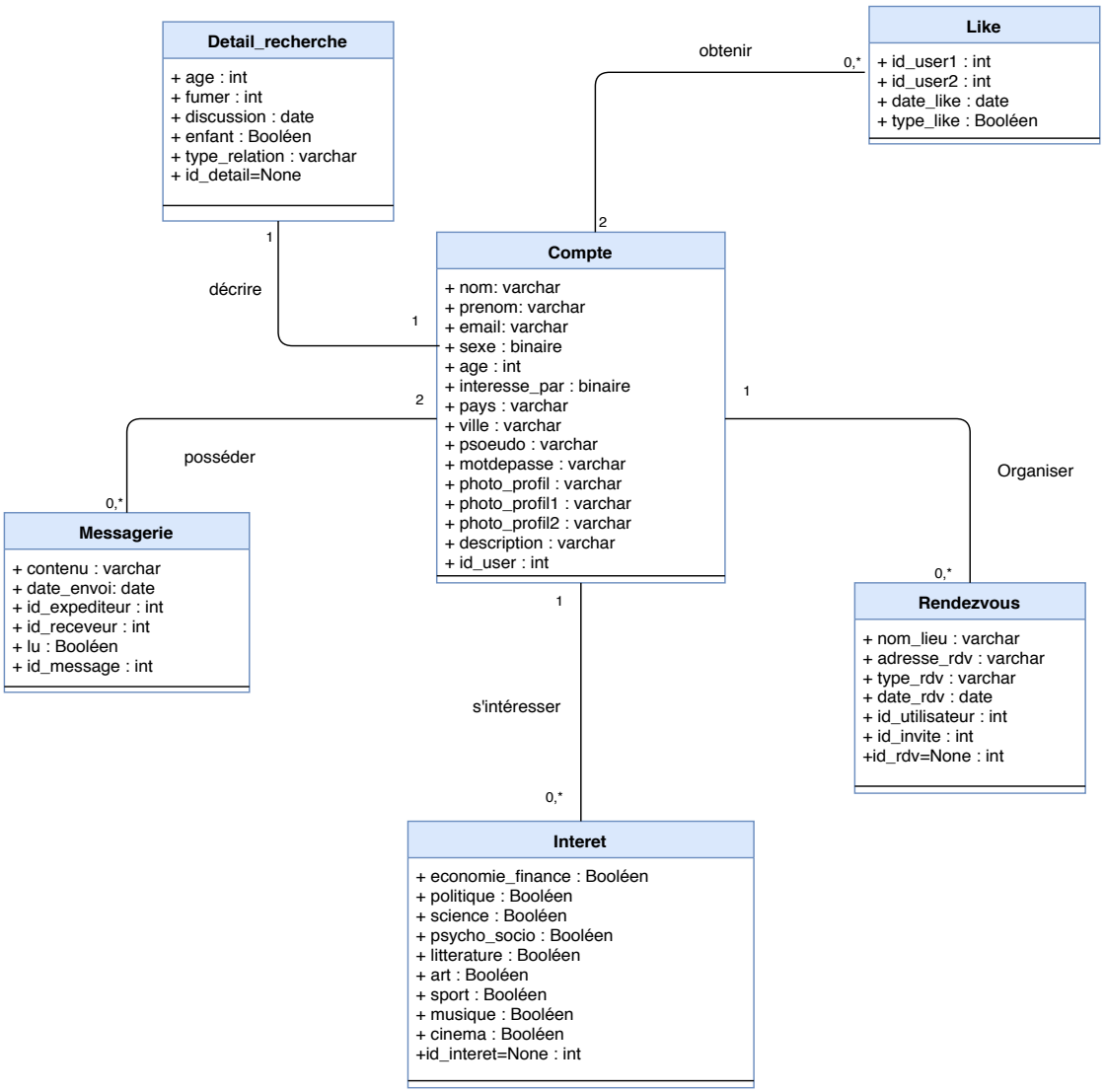


FIGURE B.3 – Diagramme de Classes

Annexe C

La persistence des données

Nom de l'attribut	Description de l'attribut	Type	Contrainte
	LIKER		
id_user1 #	identifiant de l'utilisateur	int	NOT NULL
date_like	date du like	varchar	NOT NULL
type_like	ke ou match (deuxième like réciproque entre les utilisateurs	varchar	NOT NULL
id_user2#	identifiant de l'utilisateur liké	int	NOT NULL
	MESSAGE		
id_message	identifiant du message	int	NOT NULL
contenu	contenu du message	varchar	
date_envoi	date d'envoi du message	date	NOT NULL
id_expediteur#	identifiant de l'expéditeur du message	int	NOT NULL
id_receveur#	identifiant du destinataire du message	int	NOT NULL
lu	massage lu ou non lu	boolean	NOT NULL
	RENDEEZVOUS		
id_rdv	identifiant du rendez-vous	int	NOT NULL
nom_lieu	nom du lieu de rendez-vous choisi par l'utilisateur	character	
adresse_rdv	adresse du lieu de rendez-vous choisi par l'utilisateur	character	NOT NULL
type_rdv	type de rendez-vous (boire un verre, manger, parc,...)	character	
date_rdv	date de la proposition du rendez-vous	date	NOT NULL
id_utilisateur#	identifiant de l'utilisateur proposant le rendez-vous	int	NOT NULL
id_invite#	identifiant de l'invité de l'utilisateur	int	NOT NULL
	DETAIL_RECHERCHE		
id_detail	identifiant des détails de recherche	int	NOT NULL
age	catégorie d'âge qui intéresse l'utilisateur	character	NOT NULL
fumer	préférence de l'utilisateur concernant le tabac	character	NOT NULL
discussion	préférence de l'utilisateur concernant le caractère bavard	character	NOT NULL
enfant	volonté de l'utilisateur d'avoir des enfants ou non	character	NOT NULL
type_relation	type de relation recherché par l'utilisateur	character	NOT NULL
tatouage	préférence de l'utilisateur concernant les tatouages	character	NOT NULL
piercing	préférence de l'utilisateur concernant les piercing	character	NOT NULL
id_user#	identifiant de l'utilisateur	int	NOT NULL
	INTERET		
id_interet	identifiant des intérêts	int	NOT NULL
economie_finance	l'utilisateur est-il intéressé par l'économie ou la finance ?	int	NOT NULL
politique	l'utilisateur est-il intéressé par la politique ?	int	NOT NULL
science	l'utilisateur est-il intéressé par la science ?	int	NOT NULL
psycho_socio	tilisateur est-il intéressé par la psychologie ou la sociologie	int	NOT NULL
littérature	l'utilisateur est-il intéressé par la littérature ?	int	NOT NULL
art	l'utilisateur est-il intéressé par l'art ?	int	NOT NULL
sport	l'utilisateur est-il intéressé par le sport ?	int	NOT NULL
musique	l'utilisateur est-il intéressé par la musique ?	int	NOT NULL
cinema	l'utilisateur est-il intéressé par le cinéma ?	int	NOT NULL
id_user#	identifiant de l'utilisateur	int	NOT NULL

FIGURE C.1 – Tableau des attributs des bases de données

```
CREATE TABLE liker (
    id_user1 integer NOT NULL,
    date_like date NOT NULL,
    type_like boolean DEFAULT false NOT NULL,
    id_user2 integer NOT NULL,
    PRIMARY KEY (id_user1, id_user2),
    FOREIGN KEY (id_user1) REFERENCES utilisateur(id_user) ON DELETE CASCADE,
    FOREIGN KEY (id_user2) REFERENCES utilisateur(id_user) ON DELETE CASCADE

)

CREATE TABLE rendezvous (
    id_rdv SERIAL,
    nom_lieu character(100),
    adresse_rdv character(100) NOT NULL,
    type_rdv character(100),
    date_rdv timestamp without time zone NOT NULL,
    id_utilisateur integer NOT NULL,
    id_invite integer NOT NULL,
    PRIMARY KEY (id_rdv),
    FOREIGN KEY (id_utilisateur) REFERENCES utilisateur(id_user) ON DELETE CASCADE,
    FOREIGN KEY (id_invite) REFERENCES utilisateur(id_user) ON DELETE CASCADE

);

CREATE TABLE detail_recherche(
    id_detail SERIAL,
    age character(20) NOT NULL,
    fumer character(20) NOT NULL,
    discussion character(20) NOT NULL,
    enfant character(20) NOT NULL,
    type_relation character(20) NOT NULL,
    tatouage character(20) NOT NULL,
    piercing character(20) NOT NULL,
    id_user integer NOT NULL,
    PRIMARY KEY (id_detail ),
    FOREIGN KEY (id_user) REFERENCES utilisateur(id_user) ON DELETE CASCADE

);
```

FIGURE C.2 – Tableau des attributs des bases de données

```
CREATE TABLE interet(
    id_interet SERIAL,
    economie_finance integer NOT NULL,
    politique integer NOT NULL,
    science integer NOT NULL,
    psycho_socio integer NOT NULL,
    litterature integer NOT NULL,
    art integer NOT NULL,
    sport integer NOT NULL,
    musique integer NOT NULL,
    cinema integer NOT NULL,
    id_user integer NOT NULL,
    PRIMARY KEY (id_interet),
    FOREIGN KEY (id_user) REFERENCES utilisateur(id_user) ON DELETE CASCADE

);

CREATE TABLE message (
    id_message integer NOT NULL,
    contenu character varying(500),
    date_envoi timestamp without time zone NOT NULL,
    id_expediteur integer NOT NULL,
    id_receveur integer NOT NULL,
    lu boolean DEFAULT true NOT NULL,
    PRIMARY KEY (id_message ),
    FOREIGN KEY (id_expediteur) REFERENCES utilisateur(id_user) ON DELETE CASCADE,
    FOREIGN KEY (id_receveur) REFERENCES utilisateur(id_user) ON DELETE CASCADE

);
```

FIGURE C.3 – Tableau des attributs des bases de données

Annexe D

Fonctionnalités de l'application

FIGURE D.1 – Diagramme d'activités pour la création de compte

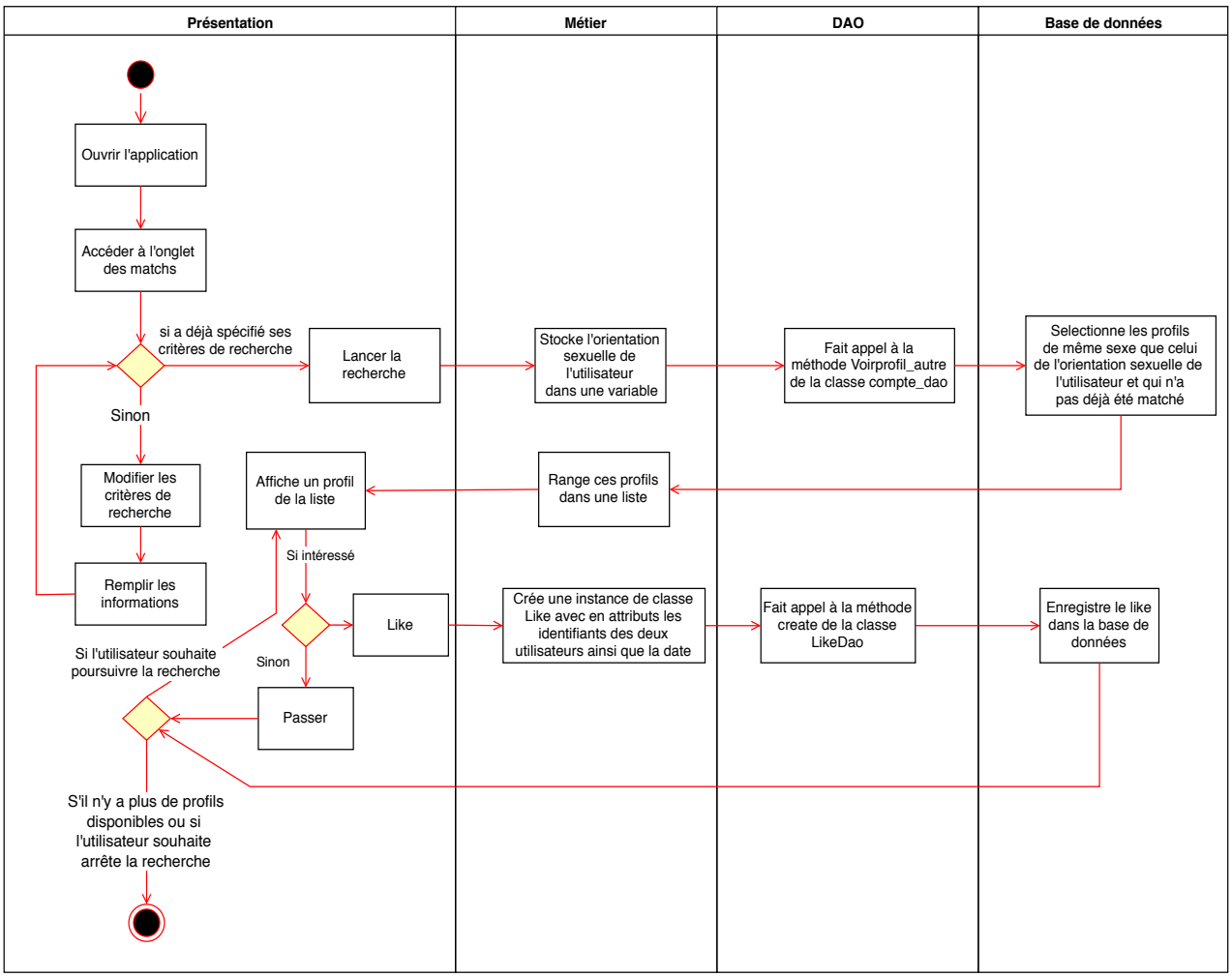


FIGURE D.2 – Diagramme d'activités pour rechercher des matchs