Équipe 106

Projet 3 : Fais-moi un dessin
Document d'architecture logicielle

Version 1.7

Historique des révisions

Date	Version	Description	Auteur
2020-02-02	1.0	Complété sections 1, 2, 3 et 7	Karima Salem
2020-02-06	1.1	Ajouter diagramme de paquetages client léger	Nicole Joyal
2020-02-06	1.2	Ajouter diagramme de paquetage client lourd	Émilio Gagnon
2020-02-07	1.3	Ajouter diagramme de paquetage serveur	Hubert et Zakari
2020-02-07	1.4	Mise en page et révision finale	Équipe 5
2020-04-12	1.5	Révision diagramme client léger	Nicole Joyal
2020-04-13	1.6	Révision diagramme client lourd	Karima Salem
2020-04-13	1.7	Révision diagramme serveur	Hubert

Table des matières

Document d'architecture logicielle

1. Introduction

Ce document a pour but de détailler l'architecture du projet à réaliser. Pour ce faire, il y est énoncé les divers éléments ayant un impact concret sur la conception, la réalisation et l'utilisation du projet. Tout d'abord, il est question des objectifs et des contraintes quant à l'architecture. Cette section porte sur les limitations en ce qui concerne l'architecture du logiciel. La vue des cas d'utilisation présente les multiples scénarios dans lesquels les acteurs emploient directement le logiciel implémenté. Ainsi, il est possible de réaliser le projet en tenant compte des différentes façons dont il sera utilisé. Ensuite, la vue logique propose les différents paquetages utilisés pour le client lourd, le client léger ainsi que le serveur. La vue des processus permet de comprendre les différentes interactions entre les divers processus. Puis, la vue de déploiement donne un aperçu de la liaison entre le matériel logiciel et le matériel physique. Finalement, la section portant sur la taille et la performance présente les contraintes en ce qui a trait à la qualité du logiciel.

2. Objectifs et contraintes architecturaux

2.1. Objectifs

2.1.1. Sécurité et confidentialité

Tout au long du développement de l'application, les développeurs doivent garder en tête la confidentialité des utilisateurs. Il est primordial que les informations personnelles collectées lors de l'inscription et tout au long de l'utilisation de l'application demeurent privées. Cela ne tient pas compte des statistiques liées aux parties, au classement des usagers ainsi qu'aux informations que les utilisateurs partagent publiquement sur leur profil. Toutefois, une fois la suppression du compte confirmée, toutes les informations propres à l'utilisateur seront détruites.

2.1.2. Réutilisation

Il est important d'implémenter des fonctionnalités qui seront faciles à réutiliser dans d'autres parties du code de l'application. Cela est non seulement une bonne pratique de codage, mais simplifiera la tâche des développeurs quand l'application s'alourdit.

2.1.3. Portabilité

Selon les exigences du client, il est important de considérer la portabilité de l'application. Puisque l'appel d'offres spécifie qu'il soit au moins possible d'utiliser l'application sur un ordinateur Windows ainsi qu'une tablette Android.

2.1. Contraintes

2.2.1. Échéancier

Un prototype du projet contenant une page de connexion ainsi qu'une plateforme de

clavardage doit être livré d'ici le 7 février 2020. Le projet complet doit être rendu avant le 13 avril 2020. Entre temps, l'équipe de développement réalisera des fonctionnalités supplémentaires chaque semaine, tel que planifié dans l'échéancier du plan de projet.

2.2.2. Langage de développement

Les langages de développement sont imposés de sorte que le client lourd est implémenté en C# avec une interface utilisateur en WPF. Pour le client léger, les fonctionnalités sont faites en Kotlin. Pour le serveur, le langage utilisé est TypeScript.

2.2.3. Outil de développement

Parmi les outils de développement, on retrouve le cadriciel .NET qui nous a été imposé pour le client lourd. Ensuite, en ce qui concerne la communication du serveur avec le client léger et le client lourd nous avons priorisé l'usage de la librairie SocketIO. Enfin, NodeJS et Express seront utilisés pour le serveur.

3. Vue des cas d'utilisation

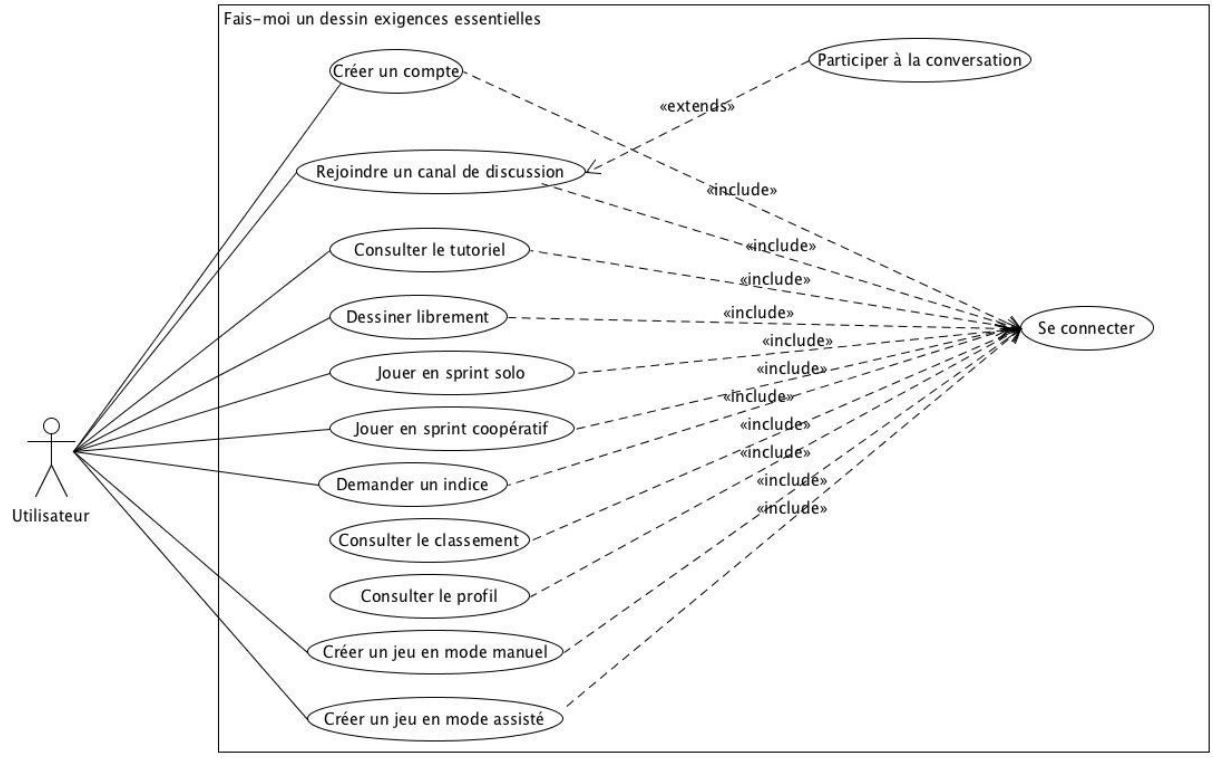

Figure 1: Diagramme de vue des cas d'utilisation des exigences essentielles du côté client lourd et client léger

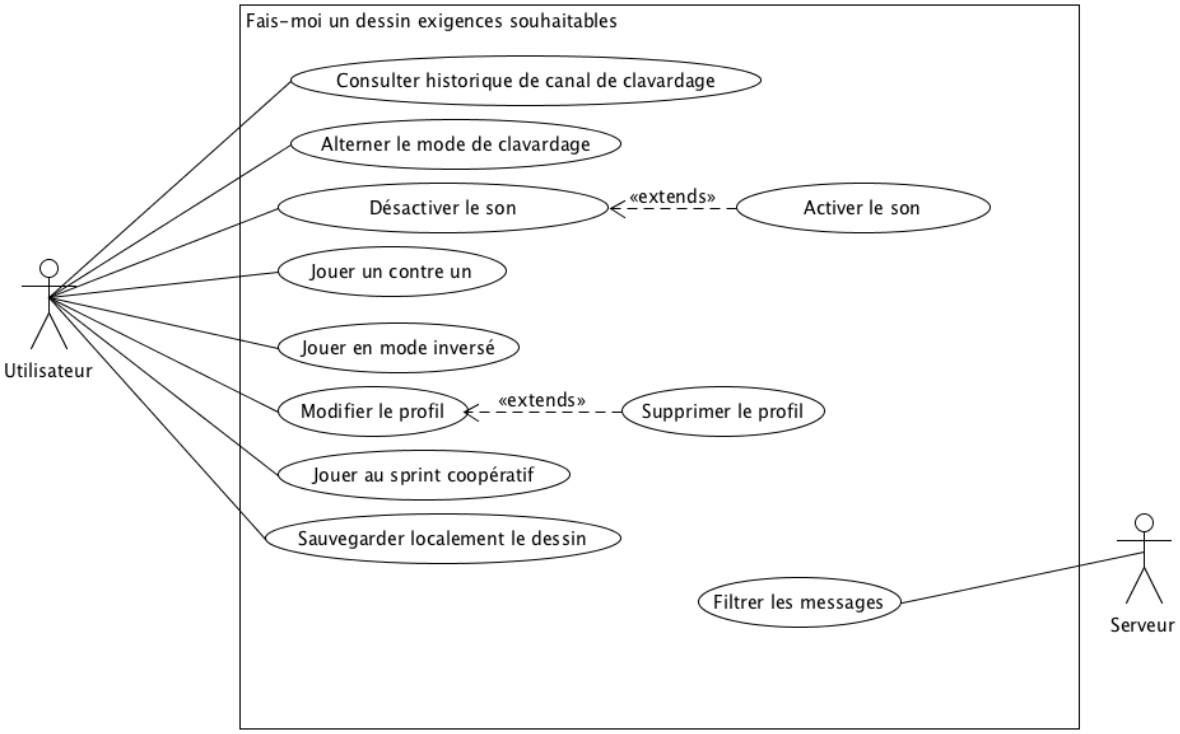

Figure 2: Diagramme de vue des cas d'utilisation des exigences souhaitables du côté client lourd et client léger

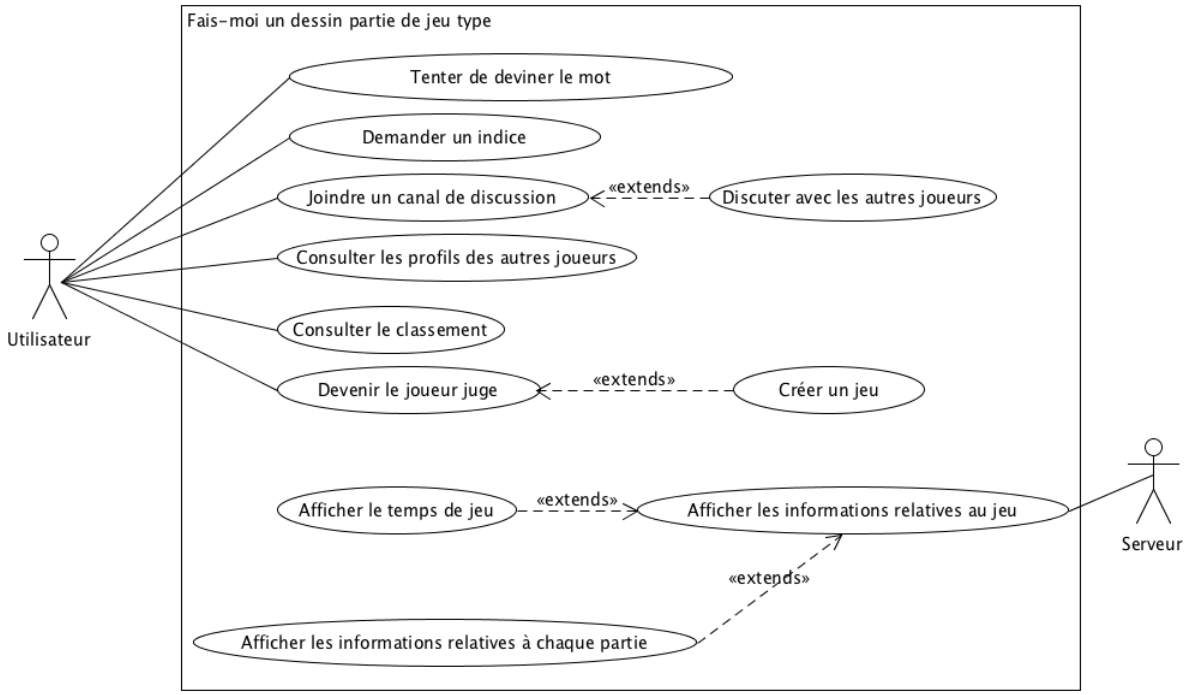

Figure 3: Diagramme de vue des cas d'utilisation d'une partie de jeu type du côté client lourd et client léger

4. Vue logique

4.1. Client léger

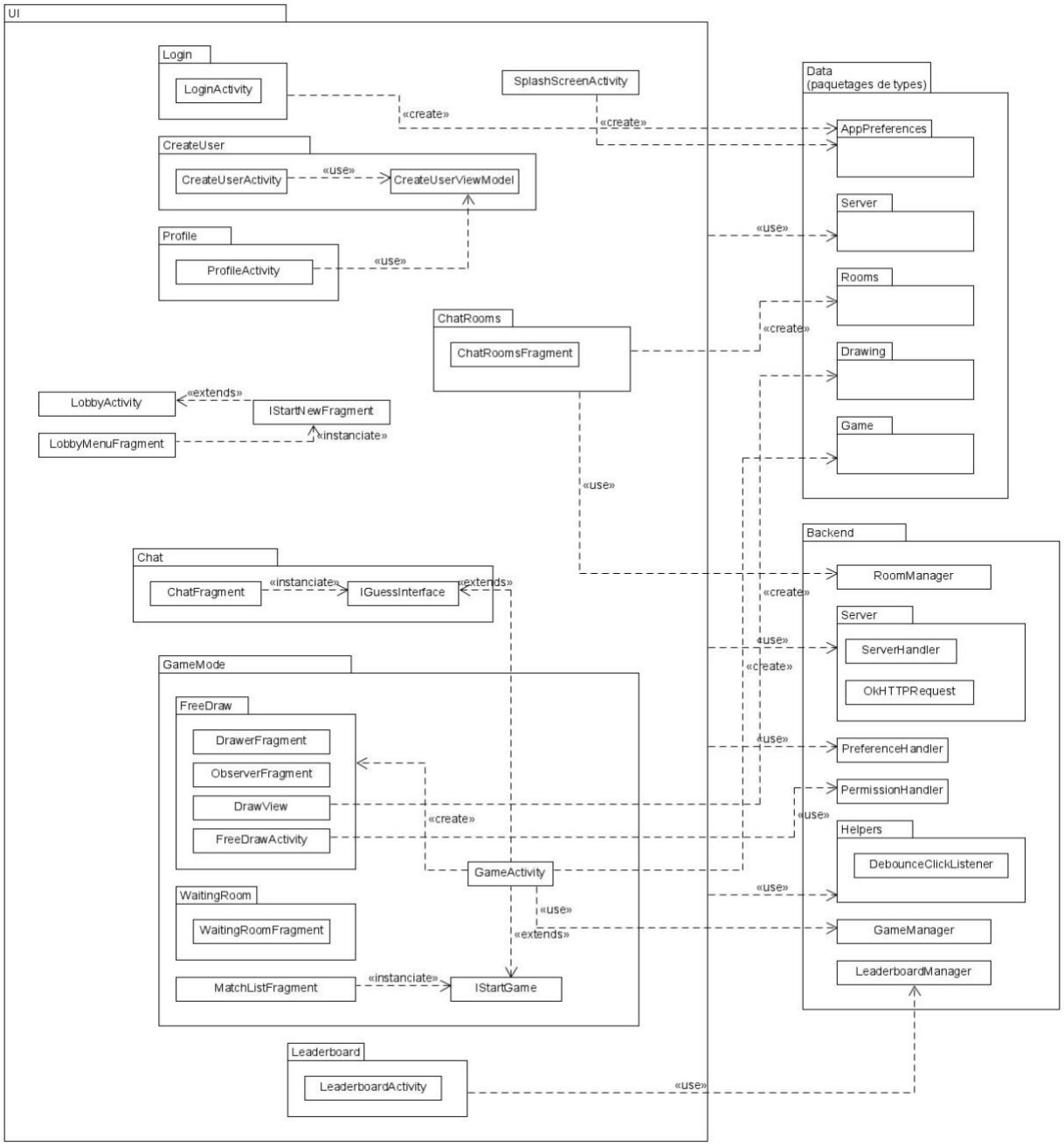

Figure 4 : Diagramme de paquetage du client léger

UI

Le paquet UI contient toutes les activités (ou fragments) de l'application. Les activités et les fragments Android s'occupent de la gestion d'évènements de l'utilisateur. Dans ce paquetage, nous retrouvons les différentes parties de l'interface essentielles à l'application.

SplashScreen

L'activité SplashScreen est la première classe instanciée à l'ouverture de l'application. Elle fait la préparation initiale des connexions et redirige l'utilisateur vers la vue appropriée (LoginActivity ou LobbyActivity).

Login

Le paquetage Login comporte les composantes reliées à l'interface de connexion de l'application.

LoginActivity

Classe qui gère l'affichage de la page de connexion ainsi que la validation de données avec le serveur.

CreateUser

Le paquetage CreateUser comporte les composantes reliées à la création d'un profil utilisateur de l'application. Il est utilisé notamment lorsque l'utilisateur choisit l'option de se créer un compte à la page de connexion.

CreateUserActivity

Activité qui gère la vue et la validation des données entrées pour la création d'un nouveau profil.

CreateUserViewModel

Classe qui fait la vérification initiale du forum de création de profil. Utilisée dans ProfileActivity aussi afin de valider les changements de profil avant d'envoyer au serveur.

Profile

Le paquetage Profile gère les évènements reliés à la récupération de statistiques de l'utilisateur, ainsi que ceux reliés à la modification du profil.

ProfileActivity

La classe ProfileActivity se charge de l'affichage des vues reliées à la modification du profil et les statistiques d'utilisation.

ChatRooms

Ce paquetage englobe les classes reliées à la gestion des canaux de discussion, notamment, la création, la suppression, l'affichage, les invitations, etc.

ChatRoomsFragment

Cette classe se charge de la vue et la gestion des évènements reliés aux canaux de discussion.

Chat

Le paquetage Chat gère les évènements reliés à la messagerie de l'application. Il gère l'affichage des messages et l'affichage de l'historique des messages (significatif).

ChatFragment

Classe qui gère les entrées de l'utilisateur et les changements de vues de l'interface de discussion.

IGuessInterface

L'interface est implémentée dans l'activité GameActivity pour traiter l'affichage d'une partie lors de la devinette d'un mot. Elle est utilisée dans le ChatFragment afin d'appeler les fonctions appropriées.

GameMode

Le paquetage GameMode englobe les différents affichages de chaque mode de jeu. Plus particulièrement, il implémente les évènements reliés au dessin en ligne et la gestion des vues dans une partie.

FreeDraw

Ce paquetage contient les fragments et activités pour différents états de dessin, ainsi que la vue permettant le dessin sur un canevas.

DrawerFragment

Classe qui gère la vue pour un joueur qui dessine.

ObserverFragment

Classe qui gère la vue pour un joueur qui observe le dessin entrant.

DrawView

Classe qui gère les actions de l'utilisateur afin de les traduire en traits de dessin.

FreeDrawActivity

Activité qui utilise le DrawView et qui implémente les fonctionnalités additionnelles de sauvegarde, mais qui ne communique pas avec le serveur.

WaitingRoom

Le paquetage WaitingRoom gère les états intermédiaires avant le commencement d'une partie (affichage des joueurs, ajout de joueurs virtuels, etc.).

WaitingRoomFragment

La classe WaitingRoomFragment gère l'affichage de composantes de la salle d'attente avant le départ d'un match. Elle communique au GameActivity avec l'interface IStartGame lorsque l'hôte part la partie.

IStartGame

L'interface qui fait le lien de communication entre l'activité de jeu (GameActivity) et la vue de la salle d'attente (WaitingRoomFragment).

MatchListFragment

Cette classe (fragment) gère les listes de parties à rejoindre.

GameActivity

Cette classe (activité) gère les évènements de déroulement d'une partie. Elle gère l'affichage des vues selon l'état de la partie (dessinateur, observateur, clavardage).

Leaderboard

Le paquetage Leaderboard présente les informations relatives au classement des utilisateurs.

LeaderboardActivity

Cette activité gère l'affichage du classement des joueurs selon le mode de jeu. Elle communique avec le serveur afin d'afficher les résultats des requêtes.

LobbyActivity

La classe LobbyActivity se charge de l'affichage des vues principales de l'application. Elle représente la classe principale qui redirige les évènements de l'utilisateur vers les vues appropriées.

LobbyMenuFragment

La classe LobbyMenuFragment implémente la vue du menu de départ de l'application. Elle se sert de l'interface IStartNewFragment pour communiquer les actions de l'utilisateur vers le LobbyActivity pour traitement.

IStartNewFragment

Interface qui est implémentée dans la classe LobbyActivity, lorsqu'appelée dans LobbyMenuFragment. Elle sert de pont de communication entre le fragment et l'activité principale.

Data

Ce paquetage englobe tous les types de données (*data class*) afin d'améliorer la lisibilité du code et de faciliter la gestion de données avec le serveur.

AppPreferences

Paquetage qui englobe les structures et les constantes reliées à la récupération des données propres à l'application.

Server

Ce paquetage regroupe les constantes reliées aux requêtes avec le serveur.

Rooms

Le paquetage Room contient tous les types utilisés durant l'envoi de données et la récupération de celles-ci quant à la gestion des canaux de discussion.

Drawing

Le paquetage Drawing contient tous les types utilisés durant l'envoi de données et la récupération de celles-ci quant à la gestion des dessins en ligne.

Game

Le paquetage Game contient tous les types utilisés durant l'envoi de données et la récupération de celles-ci quant à la gestion des parties de jeu.

Backend

Le paquet Backend contient les classes relatives aux modifications et à la récupération des données externes, serveur, ainsi que l'appareil propre.

RoomManager

La classe RoomManager se charge de la préservation de l'état des canaux de discussion au travers l'application.

Socket

SocketHandler

SocketHandler est une classe qui se charge de la gestion de la connexion avec le serveur. Elle gère les appels au serveur.

OkHTTPRequest

OkHTTPRequest contrôle les requêtes HTTP nécessaires à l'application. Souvent, il sera utile pour la récupération et la modification de données de la base de données (ex. historique des messages).

PreferenceHandler

La classe PreferenceHandler gère les préférences de l'utilisateur dans l'application (ex. l'état de connexion, etc.).

PermissionHandler

Cette classe effectue les demandes de permissions de l'utilisateur lorsque pertinente (ex. demande d'accès aux images pour la sauvegarde de dessin).

Helpers

Ce paquet englobe les classes utilitaires de l'application.

DebounceClickListener

Classe qui dérive de l'écoute de clics utilisateurs et qui impose un intervalle de temps avant de soumettre un autre clic.

GameManager

La classe GameManager se charge de la préservation de l'état de la partie jouée au travers l'application.

LeaderboardManager

La classe LeaderboardManager se charge de la préservation de l'état des listes de classement au travers l'application.

4.2. Client lourd

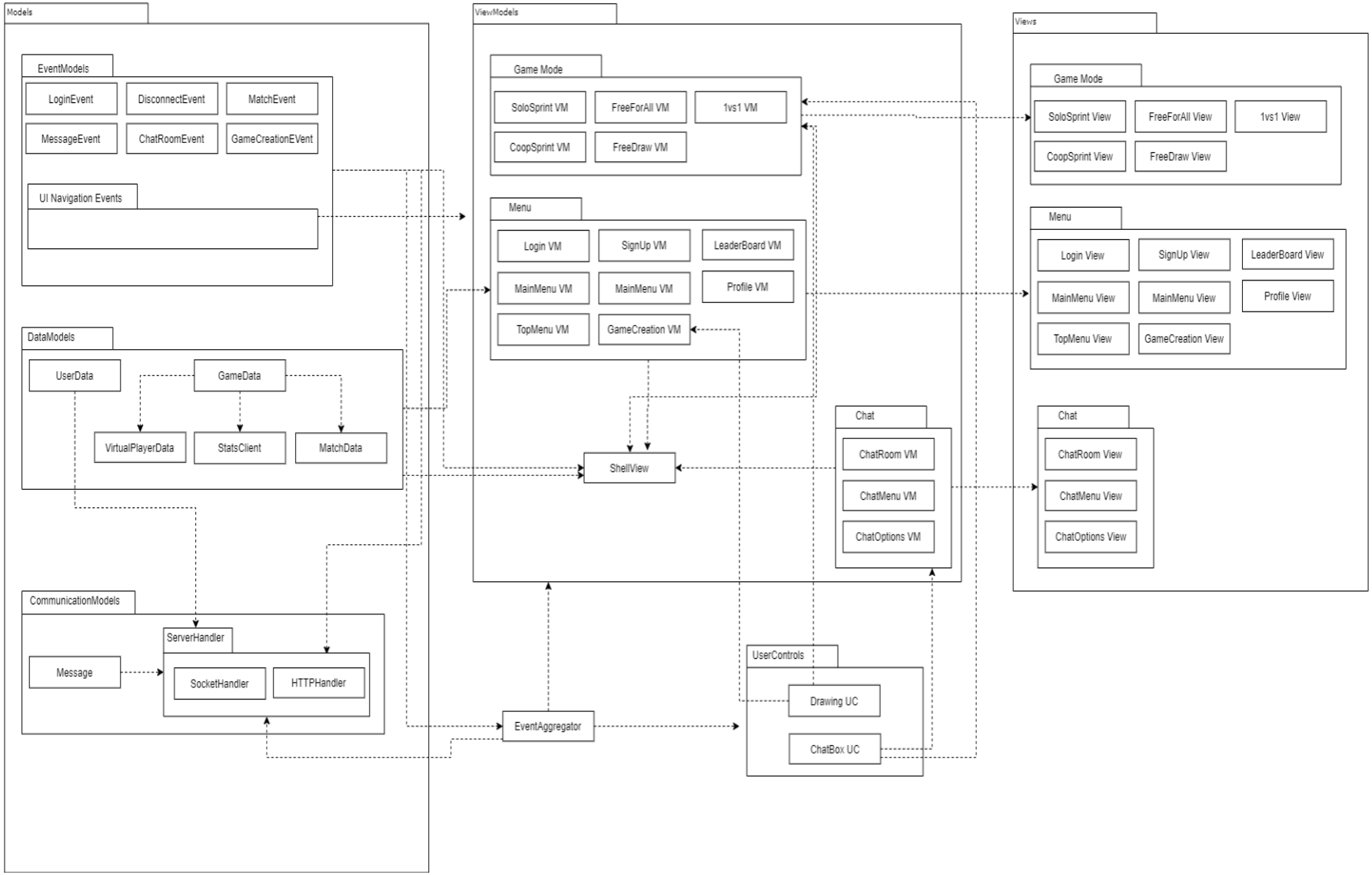

Figure 5 : Diagramme de paquetage du client lourd

Model

Le paquetage Model regroupe les paquets relatifs aux modèles d'évènements, de communication et de données.

EventModels

EventModels est un paquetage qui contient les modèles des différents événements qui seront transmis entre les vues et les classes de l'application via le singleton EventAgreggator.

LoginEvent

Événement associé à l'appui du bouton connexion dans la vue Login. Permets d'engendrer un changement de vue via le Shell ViewModel.

DisconnectEvent

Événement associé à l'appui du bouton déconnexion dans la vue MainMenu. Permets d'engendrer un changement de vue via le Shell ViewModel.

ConnectEvent

Événement utilisé par le SocketHandler pour vérifier la disponibilité d'un compte dans le serveur.

sendMessageEvent

Événement associé à l'envoi d'un message, permet d'avertir le SocketHandler de faire l'émission du message écrit dans le ChatBox UC.

UI Navigation Events

Paquetage regroupant le reste des événements permettant de changer de vue via la vue modèle squelette (Shell ViewModel).

DataModel

Le paquet DataModel regroupe les paquets relatifs aux modèles de gestion des données dans l'application.

UserData

Classe regroupant les données de l'utilisateur actif. Par exemple son nom d'utilisateur, son avatar, ses préférences utilisateur, ses messages, etc.

GameData

Classe regroupant les données d'un jeu. Par exemple les dessins, les indices, le nombre de tours, le score final pour chaque joueur, etc.

VirtualPlayerData

Classe regroupant les données d'un joueur virtuel. Par exemple, ses traits de personnalité, la liste de son pointage des parties où il a participé, etc.

Score

Classe regroupant les données d'un pointage. Par exemple: le type de partie joué, le nombre de points, l'identifiant unique du joueur qui a réalisé le pointage, etc.

CommunicationModels

Paquet regroupant tous les paquets essentiels à la communication avec le serveur, notamment la classe Message, la classe APIHelper et le paquet ServerHandler.

Message

Classe décrivant un message, c'est-à-dire un contenu, une estampille de temps et un nom d'utilisateur.

ServerHandler

Paquetage regroupant les classes permettant de gérer les interactions avec le serveur.

SocketHandler

Classe permettant de gérer la connexion socket entre le client et le serveur ainsi que la réception de messages et les événements des vues de jeu. Permets notamment de gérer les événements sur le socket.

HTTPHandler

Classe permettant de gérer les appels HTTP entre le client et le serveur. Utile pour les requêtes de données.

ViewModel

Le paquetage regroupe les vues modèles faisant le lien entre l'affichage (Views) et les différents modèles logiques de l'application (Models).

GameMode

Paquetage regroupant les vues modèles des modes de jeu de l'application.

SprintSoloVM

Vue modèle du mode de jeu sprint solo. Contiens une variable pointage en cours.

SprintCoopVM

Vue modèle du mode de sprint coopératif. Contiens une variable pointage en cours.

OneOnOneVM

Vue modèle du mode de jeu 1v1. Contiens une variable pointage en cours.

FreeDrawVM

Vue modèle du mode de jeu dessin libre. Contiens une variable booléenne permettant de cacher ou non les boutons de sauvegarde du dessin.

Shell ViewModel

Classe de la vue-modèle squelette de l'application. C'est-à-dire que toutes les vues de l'application seront contenues dans cette vue. Cette classe permet aussi de gérer les événements de changement de vue (UI Navigation Events) émis par les vues.

Menu

Paquetage regroupant les vues-modèles des vues ayant un lien avec le menu principal de l'application.

MainMenu VM

Vue modèle de la vue du menu principal.

Login VM

Vue modèle de la vue de connexion.

LeaderBoard VM

Vue modèle de la vue de classement. Contiens une liste de pointage.

EventAggregator

Classe permettant de centraliser tous les événements de l'application, par exemple un changement de

vue, la réception d'un message, la réception de données, etc.

UserControls

Paquetage regroupant les classes de contrôle utilisateur, celles-ci pouvant être intégrées dans les vues ou affichées dans une vue séparée (utile par exemple pour la boîte de chat ou les modes de jeu).

ChatBox UC

Classe de contrôle utilisateur d'une boîte de chat.

Drawing UC

Classe de contrôle utilisateur d'une boîte de dessin.

View

Paquetage regroupant les paquets de vue de l'application.

GameMode

Paquetage regroupant les vues modèles des modes de jeu de l'application.

SprintSolo View

Vue du mode de jeu sprint solo. Contiens l'UC Drawing, l'UC Chatbox, l'affichage des points, le chronomètre, etc.

SprintCoop View

Vue du mode du sprint coopératif. Contiens l'UC Drawing, l'UC Chatbox, l'affichage des points, le chronomètre, le mot décrivant le dessin demandé, etc.

OneOnOne View

Vue du mode de jeu 1 contre 1. Contiens l'UC Drawing, l'UC Chatbox, l'affichage des points, le chronomètre, les indices, etc.

FreeDraw View

Vue du mode de jeu dessin libre. Contiens l'UC Drawing.

Menu

Paquetage regroupant les vues-modèles des vues ayant un lien avec le menu principal de l'application.

MainMenu View

Vue modèle de la vue du menu principal. Contiens des boutons dirigeant aux autres vues menues.

Login View

Vue modèle de la vue de connexion.

LeaderBoard View

Vue modèle de la vue de classement. Affiche une liste des 10 premiers meilleurs pointages d'un mode de jeu, ainsi que la position du joueur actif.

Tutorial View

Vue modèle de la vue d'un tutoriel. Permets de passer des images une par une avec un bouton suivant et précédent.

Preferences View

Vue modèle de la vue des préférences de l'application. Permets de changer les paramètres de compte utilisateur et un bouton pour sauvegarder les changements.

4.3. Serveur

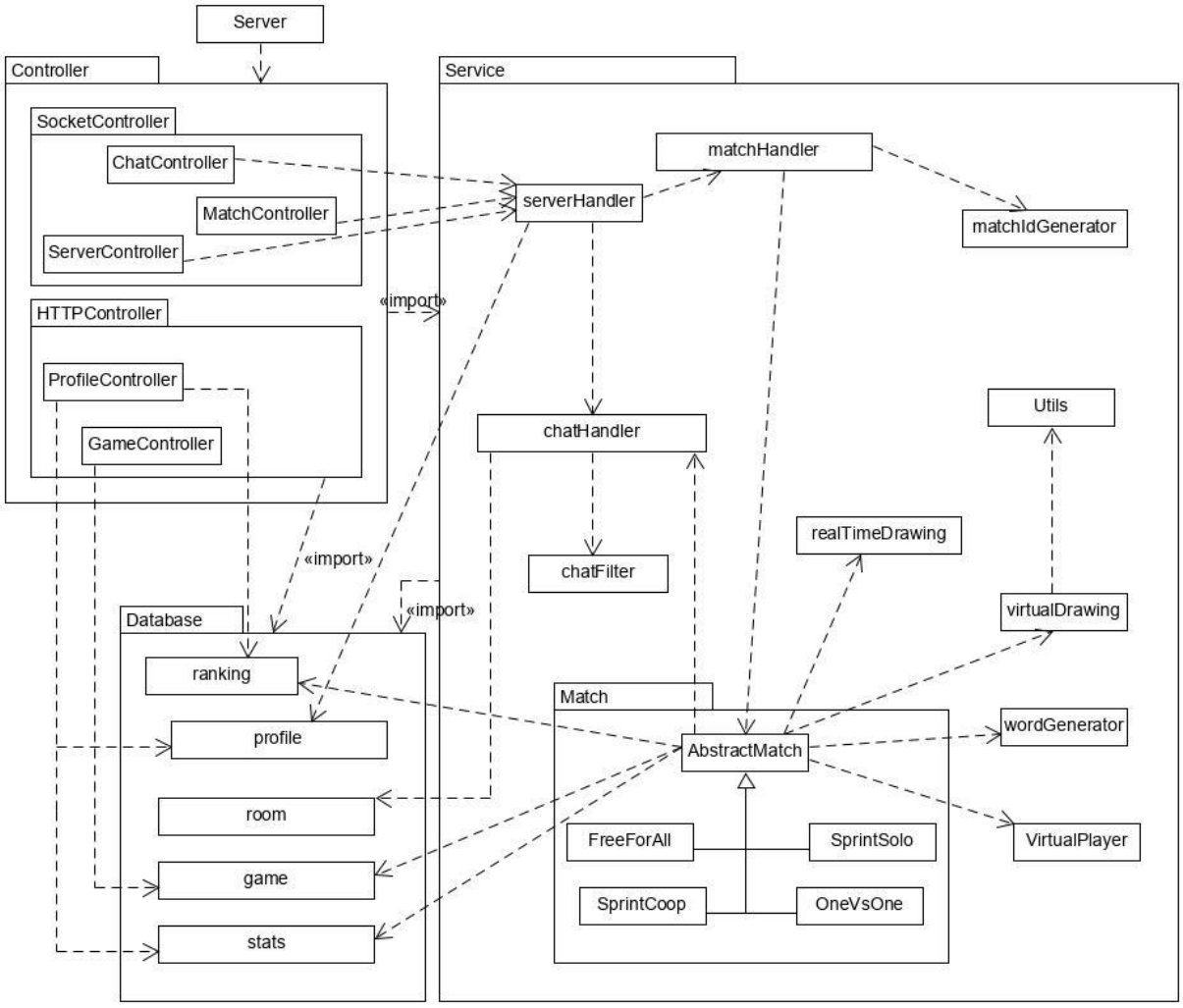

Figure 6 : Diagramme de paquetage du serveur

Server
Point de départ de l'application. C'est de cette classe que les *controllers* sont démarrés et que le serveur se met en écoute des requêtes réseau imminentes.

Controller
Englobe les paquetages SocketController et HTTPController

SocketController
Englobe les classes qui s'occupent de la connexion Socket.io

ChatController

Classe qui s'occupe de recevoir les messages concernant le clavardage.

ServerController

Classe qui s'occupe de la connexion de l'utilisateur au serveur.

MatchController

Classe qui s'occupe des messages envoyés durant une partie.

HTTPController

Englobe les classes qui s'occupent de la connexion avec requêtes HTTP.

Profile controller

Classe qui gère les changements liés aux profils à la demande de l'utilisateur.

GameController

Classe qui s'occupe d'insérer les jeux créés dans la base de données.

Database

Englobe les classes permettant d'ajouter, modifier et supprimer les éléments des différentes tables de la base de données MongoDB. Chaque classe ci-dessous correspond à une table dans la base de données.

ranking

Classe permettant l'accès aux tables regroupant les classements des joueurs pour chaque mode de jeu de la base de données.

profile

Classe permettant l'accès à la table regroupant les informations sur les profils des utilisateurs.

game

Classe permettant l'accès à la table regroupant les jeux créés par les usagers sur le client lourd.

stats

Classe permettant l'accès à la table regroupant les statistiques de partie des profils utilisateurs.

room

Classe permettant l'accès à la table regroupant les canaux de discussions publiques créés.

Service

ChatHandler

Service qui s'occupe de faire le traitement des canaux de discussion et des messages.

ChatFilter

Service qui s'occupe de filtrer les mauvais mots des messages envoyés par les utilisateurs.

MatchHandler

Service qui s'occupe d'effectuer toutes les actions déclenchées par les évènements dans le *controller* lié à une partie, tels que début d'une partie, début d'une manche, fin d'une manche, fin d'une partie.

MatchIdGenerator

Service qui s'occupe de générer un identifiant aléatoire pour un match.

ServerHandler

Service qui s'occupe de gérer les connexions et les déconnexions des utilisateurs.

VirtualPlayer

Service qui s'occupe de répertorier tous les comportements des joueurs virtuels selon leur personnalité.

RealTimeDrawing

Service qui s'occupe de gérer le dessin en temps réel vu par tous les joueurs d'une partie.

WordGenerator

Service qui s'occupe de générer trois mots aléatoires pour le joueur dessinateur.

VirtualDrawing

Service qui s'occupe d'envoyer les évènements de dessin au bon moment pour recréer un dessin auparavant créé.

Utils

Service qui s'occupe d'implémenter les algorithmes de tri selon le mode de dessin ainsi que d'uniformiser les dessins selon le temps disponible.

VirtualPlayer

Service qui s'occupe de créer des joueurs virtuels aléatoires et de récupérer leurs messages personnalisés.

Match

Paquetage regroupant les classes qui s'occupent des différents fonctionnements des parties.

AbstractMatch

Classe abstraite avec les méthodes de base à implémenter pour le fonctionnement d'une partie.

FreeForAll

Classe dérivant de AbstractMatch s'occupant du fonctionnement d'une partie du mode de jeu FreeForAll.

SprintSolo

Classe dérivant de AbstractMatch s'occupant du fonctionnement d'une partie du mode de jeu SprintSolo.

SprintCoop

Classe dérivant de AbstractMatch s'occupant du fonctionnement d'une partie du mode de jeu SprintCoop.

OneVsOne

Classe dérivant de AbstractMatch s'occupant du fonctionnement d'une partie du mode de jeu OneVsOne.

5. Vue des processus

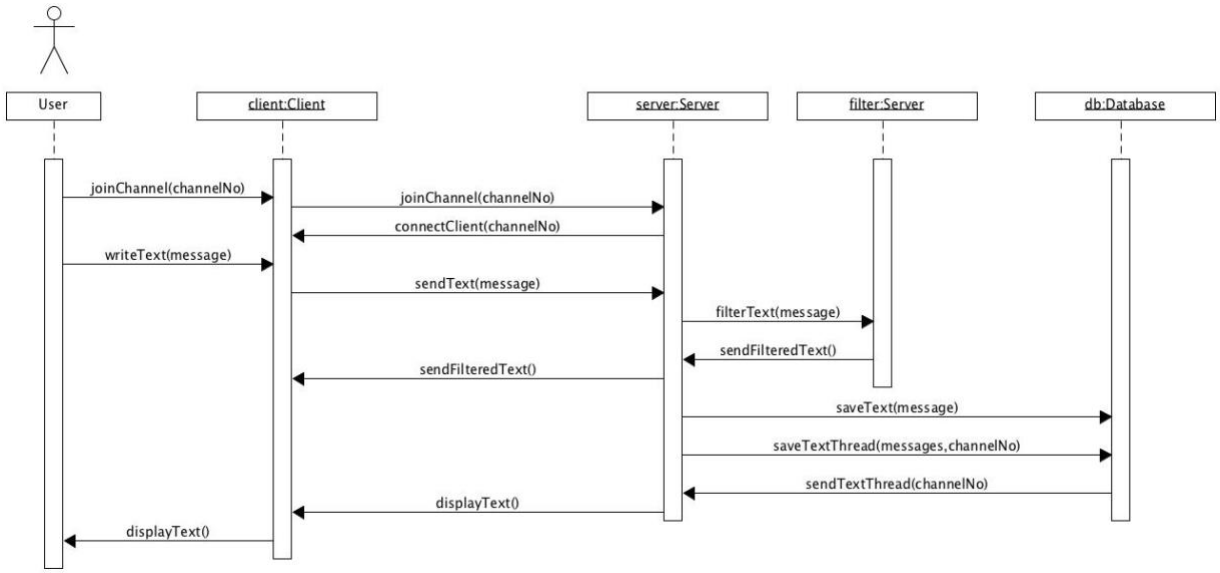

Figure 7: Diagramme de séquence du clavardage

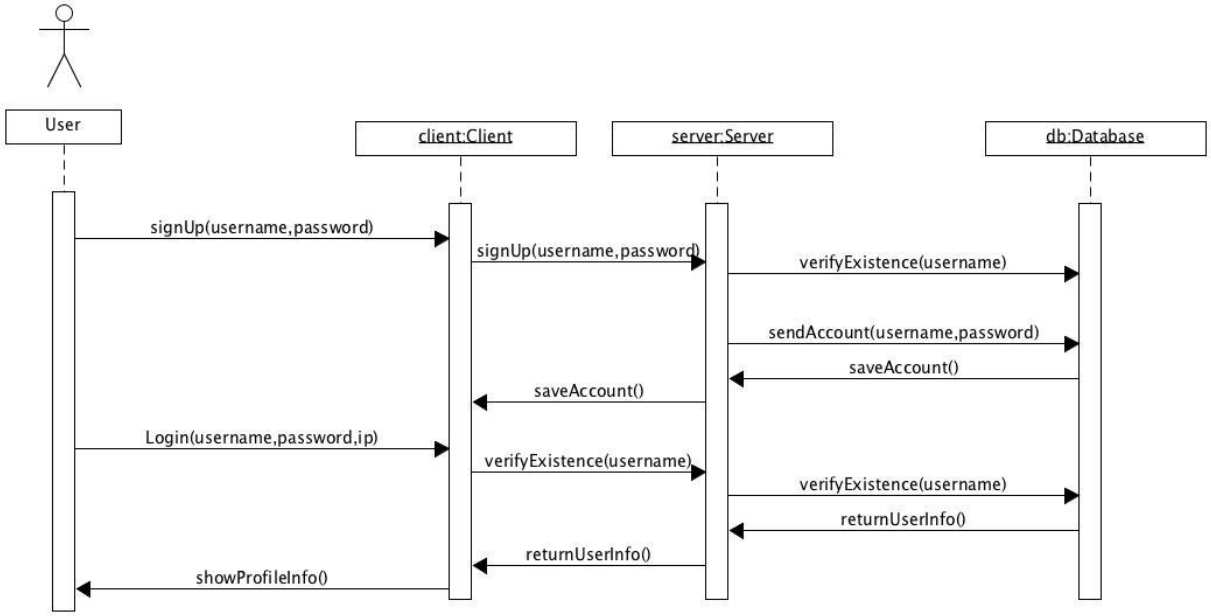

Figure 8: Diagramme de séquence de l'inscription et de la connexion

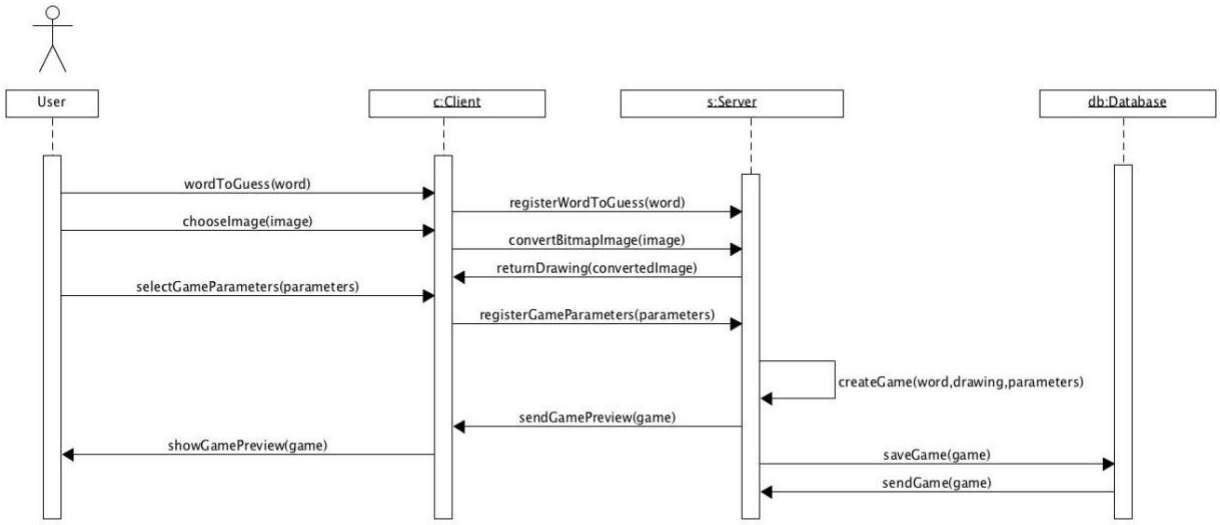

Figure 9: Diagramme de séquence de la création de jeu en mode Assisté I

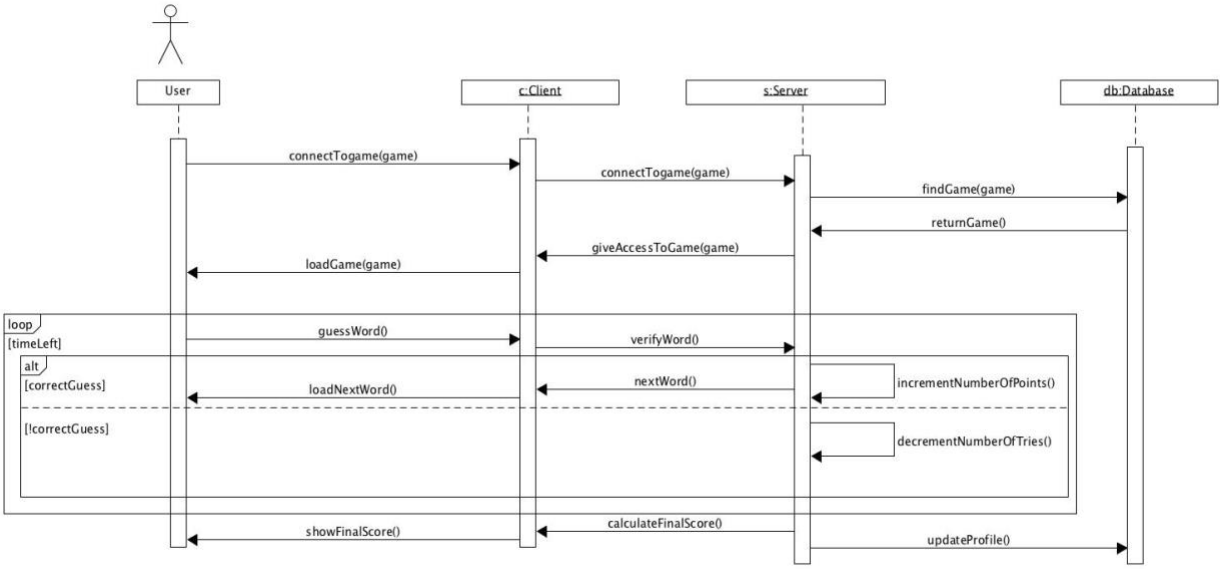

Figure 10: Diagramme de partie de jeu en sprint solo

6. Vue de déploiement

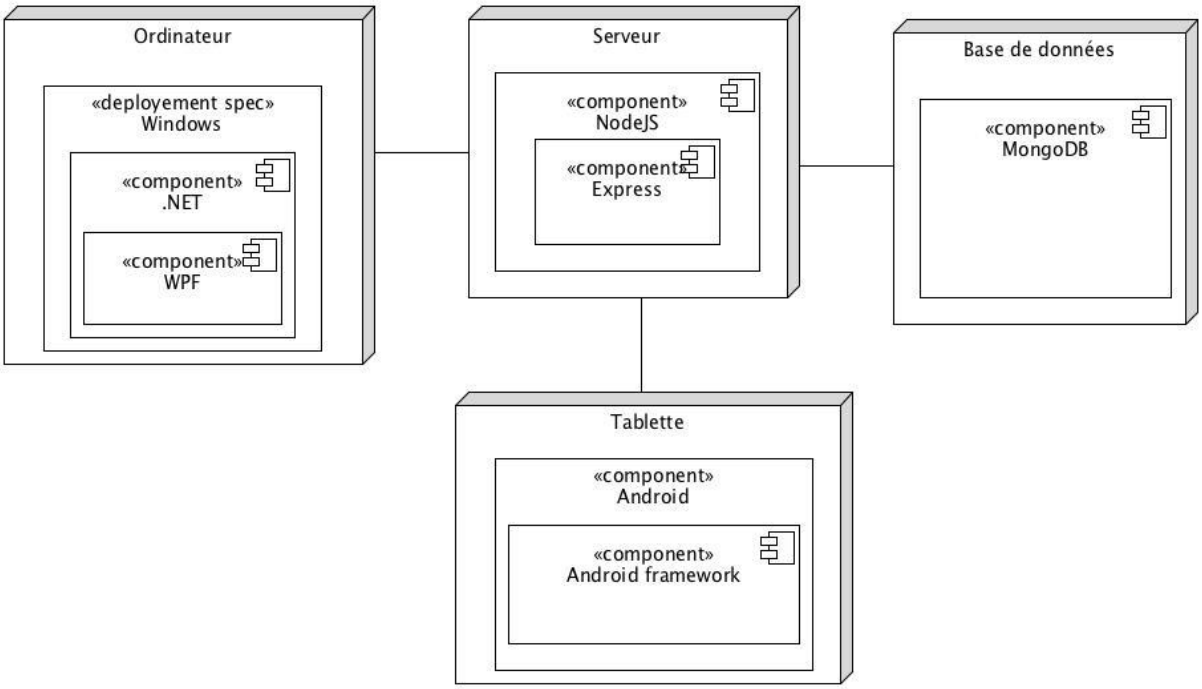

Figure 11: Diagramme de vue de déploiement du système

7. Taille et performance

Il est impératif de considérer la taille et la performance de l'application lors de sa planification. Donc, pour le client lourd il serait important d'imposer à l'utilisateur de libérer au moins 1 Go d'espace sur sa machine. Du côté performance, l'application ne devrait nécessiter que 1 Go de mémoire vive pour bien fonctionner. L'exploitation du processeur ne devrait jamais surpasser 25 %. Pour assurer le moins de frustration possible de la part de l'utilisateur, le lancement de l'application devrait prendre au maximum 15 secondes, mais il serait idéal que cela prenne moins de 5 secondes. Ensuite, pour le client léger, la tablette pour laquelle les dimensions de l'application ont été optimisées est la Galaxy Tab A. Afin d'assurer un bon fonctionnement adéquat de l'application sur la tablette, cette dernière nécessite au moins 1 Go d'espace mémoire. Pour ce qui a trait au serveur, le temps de réponse lors d'une requête ne devrait pas dépasser une demi-seconde (environ 0.5s). Autrement dit, le temps total pour la réception, le traitement et le retour de données aux clients doit demeurer minimal. Finalement, le serveur doit être capable de supporter au moins 100 canaux de discussions simultanés et 25 parties jouées en même temps en n'importe quel mode de jeu.