Équipe 207

Fais-moi un dessin
Document d'architecture logicielle

Version 2.2

Historique des révisions

Date	Version	Description	Auteur
2021-02-17	1.0	Première rédaction	Équipe 207
2021-02-18	1.1	Révision de la structure du document	Marc-Olivier Riopel
2021-04-11	2.0	Révision avant la remise finale	Samuel Ouvrard
2021-04-19	2.1	Correction	Julien Witty
2021-04-19	2.2	Mise en page finale	Samuel Ouvrard

Table des matières

Document d'architecture logicielle

1. Introduction

Le document d'architecture logicielle qui suit présente en détail le plan d'architecture pour la soumission de l'équipe 207. Les différentes sections présentent l'ensemble du système sous différentes vues (cas d'utilisation, logique, processus et déploiement) ainsi que les objectifs, les contraintes, les caractéristiques de tailles et de performance du projet ayant un impact sur l'architecture du logiciel. La section 2 traite d'abord des objectifs et des contraintes du logiciel qui influence l'architecture. Ensuite, les sections 3 à 6 présentent les différentes vues du système dans l'ordre suivant: vue des cas d'utilisation, vue logique, vue des processus et vue de déploiement. Puis, la section 7 traite des caractéristiques de taille et de performance du logiciel qui influence l'architecture.

2. Objectifs et contraintes architecturaux

L'architecture devra permettre la communication entre différents clients par un intermédiaire servant aussi de source de vérité pour la logique derrière le fonctionnement des modes de jeu. Les données pour lesquelles l'intégrité est d'une grande importance devront être transformées seulement par le serveur. Le client pourra uniquement afficher ces données. Cela empêchera les clients de tricher le mécanisme de jeu, de clavardage ou de classement. Certaines données devront être conservées dans une base de données qui sera seulement accessible du serveur. Pour afficher ces données, les clients devront utiliser le serveur comme intermédiaire à l'aide du protocole HTTP ou WebSocket, selon le contexte. Finalement, un service de stockage infonuagique devra être utilisé pour sauvegarder efficacement les images qui seront utilisées dans le fil d'image du client lourd, ainsi que dans les votes sur les dessins de joueurs virtuels en fin de partie.

2.1 Serveur

2.1.1 Objectif

L'objectif de notre architecture est que le serveur serve de seule source de vérité. Cette structure permet à tous les clients lourds et légers d'avoir les mêmes informations puisque celles-ci sont maintenues à jour dans le serveur et peuvent être demandées indépendamment par tous les clients. Dans le serveur l'objectif de notre architecture à trois niveaux (contrôleurs, services, modèles) est de réduire la possibilité de dépendance circulaire en séparant les responsabilités de chaque niveau très clairement. Les contrôleurs reçoivent les requêtes et appellent les services, les services traitent l'information et appellent les modèles alors que les modèles ont l'unique responsabilité d'interagir avec les différentes collections de la base de données.

2.1.2 Disponibilité

Pour assurer le fonctionnement de notre application en production, nous devrons rendre le serveur accessible aux clients à distance. Pour ce faire, le service EC2 d'AWS sera utilisé. Le code du serveur sera déployé sur une instance t2.small et disponible en tout temps.

2.1.3 Fiabilité

Le serveur doit être en mesure de supporter un minimum de deux parties simultanément (8 joueurs). En dessous de cette limite, le serveur doit offrir une grande fiabilité. Il ne doit pas avoir de problèmes de latence, de *crashs* ou toutes autres situations similaires.

2.1.3 Intégrité des données

Le serveur doit assurer la persistance des données importantes. Pour ce faire, une base de données MongoDB sera utilisée pour conserver ces données. Également, le serveur devra être développé de manière à ce que les clients ne puissent pas manipuler de données importantes de leur côté. Il faudra donc implémenter la validation côté serveur et éviter le traitement des données importantes du côté client.

2.2 Client léger

2.2.1 Objectif

L'architecture du client léger est une architecture à trois niveaux soit modèle, vue, vue modèle (MVVM). L'objectif de cette architecture est de pouvoir créer des vues qui sont gérées par les vues-modèles. Les vues-modèles peuvent aussi interagir avec la couche modèle de l'application où sont stockées les informations accessibles dans l'ensemble des vues. Ces vues et vues-modèles sont divisées en fragments et en activités. Les fragments servent de sous-ensembles pour les activités et peuvent être réutilisés dans l'application. Le but de cette pratique est de diviser la logique des différents composants de l'application pour pouvoir s'en resservir ailleurs.

2.2.2 Portabilité

Le client léger doit être disponible sur le système d'exploitation Android.

2.2.3 Cadriciel de développement

Le client léger sera développé avec le cadriciel AndroidX.

2.3 Client lourd

2.3.1 Objectif

L'objectif de notre architecture dans le client lourd est de séparer la vue du modèle. On retrouve donc une architecture orientée sur des composantes qui contiennent une vue et un élément comparable à un contrôleur pour gérer dynamiquement celle-ci. Les contrôleurs peuvent interagir avec les services qui ont donc la responsabilité de traiter les données fournies par les composantes avant de les rendre disponibles au reste de l'application.

2.3.2 Portabilité

Le client lourd doit être disponible pour les systèmes d'exploitation Windows.

2.3.3 Cadriciel de développement

Le client lourd sera développé avec le cadriciel Angular et utilisera Electron pour rendre l'application disponible pour ordinateur de bureau.

3. Vue des cas d'utilisation

3.1 Diagramme de cas d'utilisation du clavardage

Figure 1. Diagramme de cas d'utilisation du clavardage

3.2 Cas d'utilisation pour le profil utilisateur

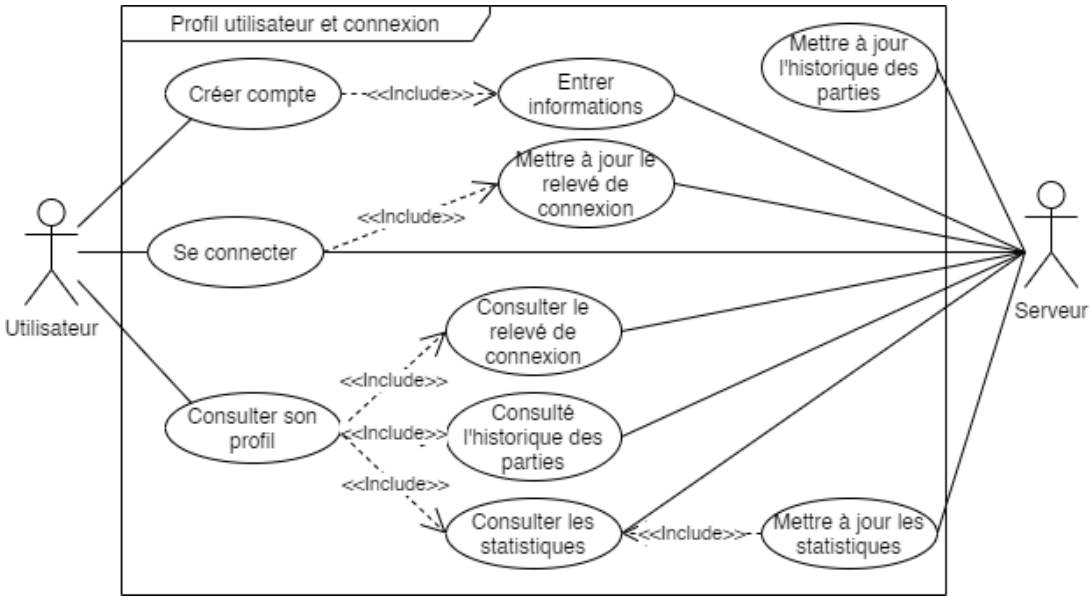

Figure 2. Diagramme de cas d'utilisation du profil utilisateur

3.3 Cas d'utilisation de la sélection d'un mode de jeu

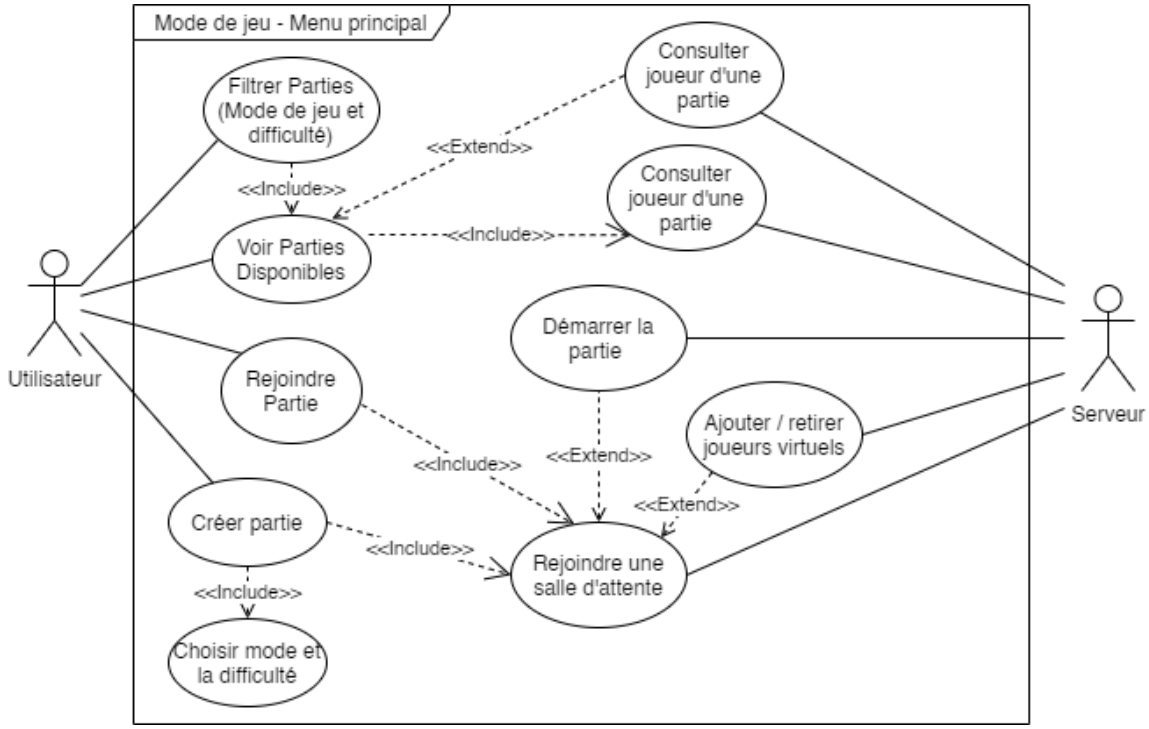

Figure 3. Diagramme de cas d'utilisation pour la sélection de modes de jeu

3.4 Cas d'utilisation du mode de jeu classique

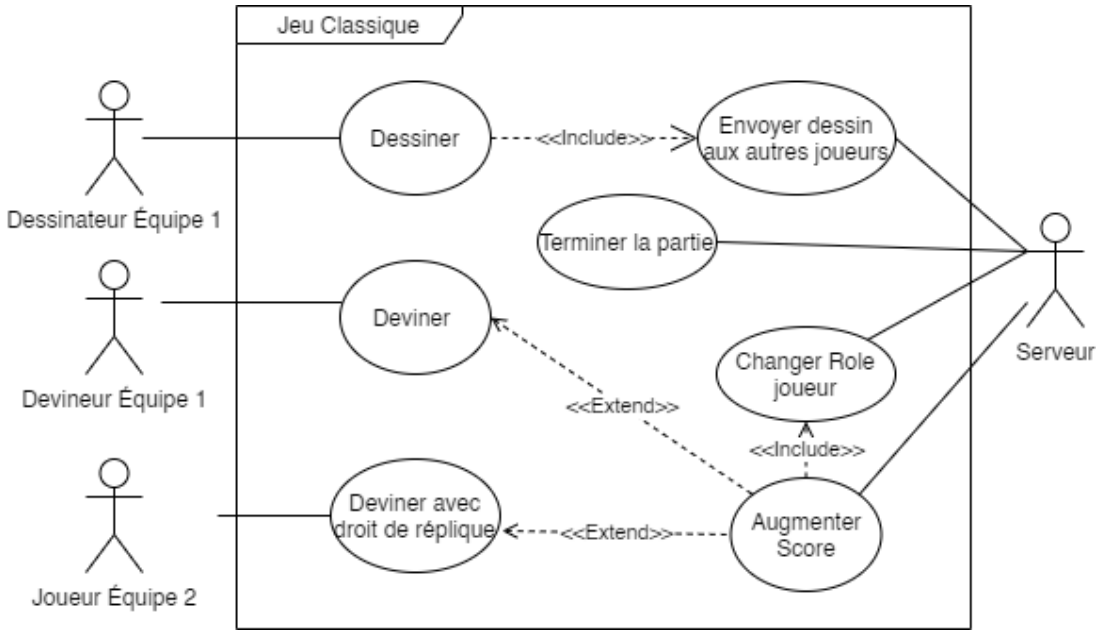

Figure 4. Diagramme de cas d'utilisation pour le mode de jeu classique

3.5 Cas d'utilisation du mode de jeu sprint solo et sprint coop

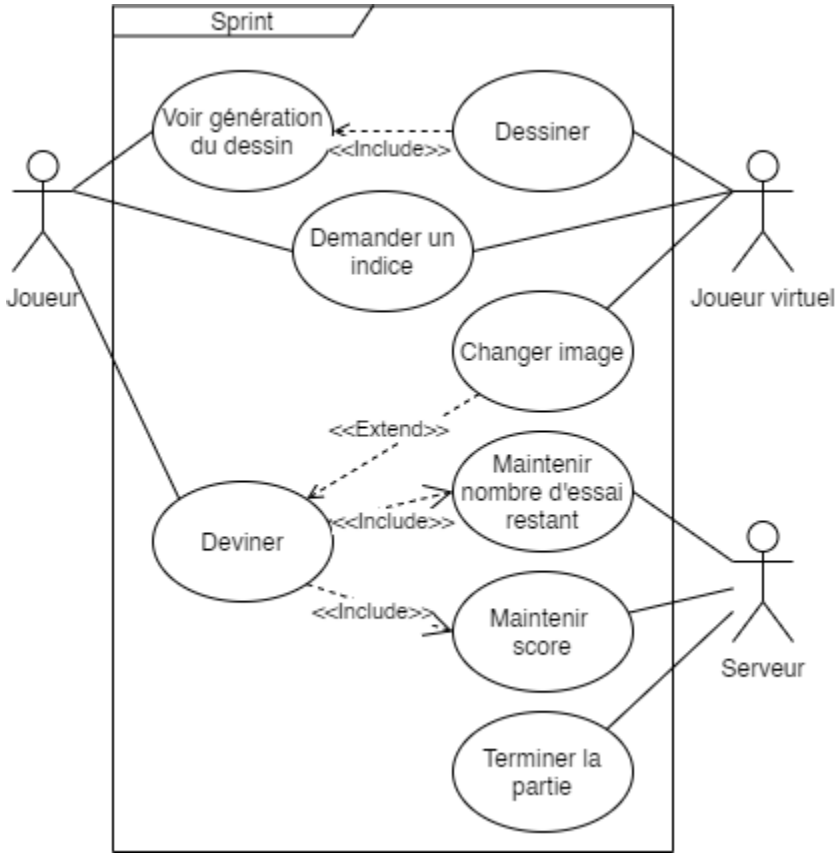

Figure 5. Diagramme de cas d'utilisation du mode de jeu sprint solo et sprint coop

3.6 Cas d'utilisation de la création paire mot-image

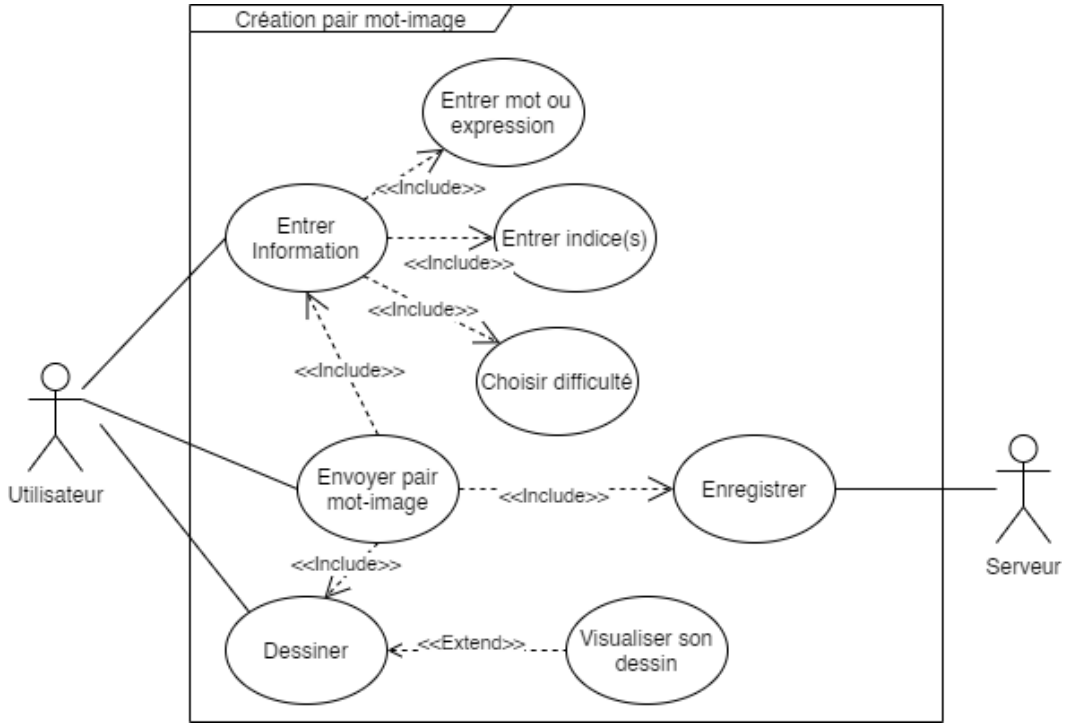

Figure 6. Diagramme de cas d'utilisation pour la création d'une paire mot-image

3.7 Cas d'utilisation du tableau de classement

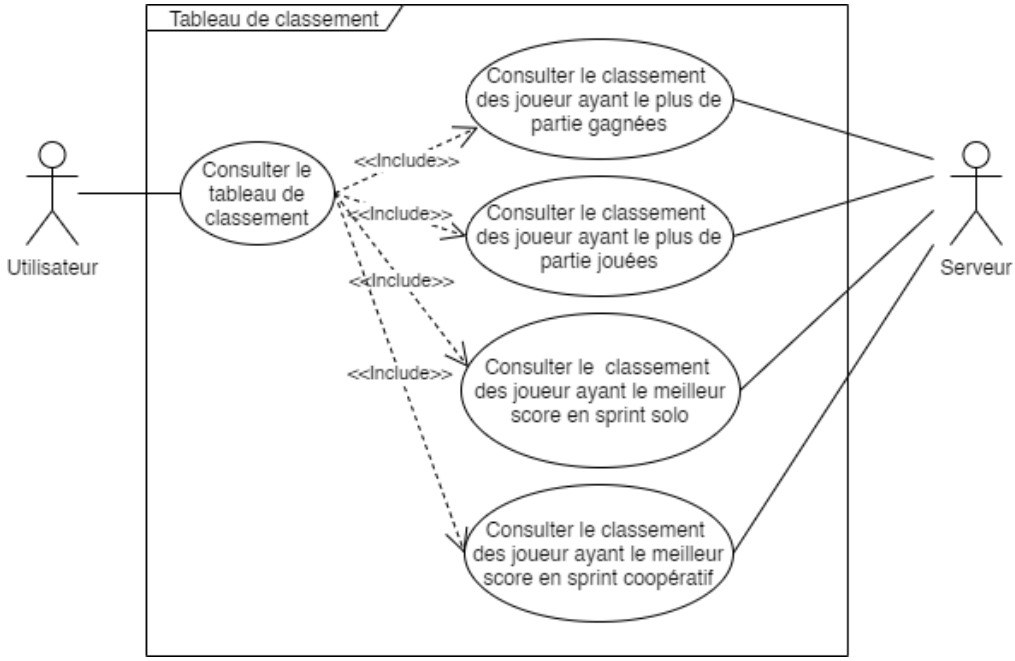

Figure 7. Diagramme de cas d'utilisation pour la consultation du tableau de classement

3.8 Cas d'utilisation de l'envoi du dessin après une partie classique

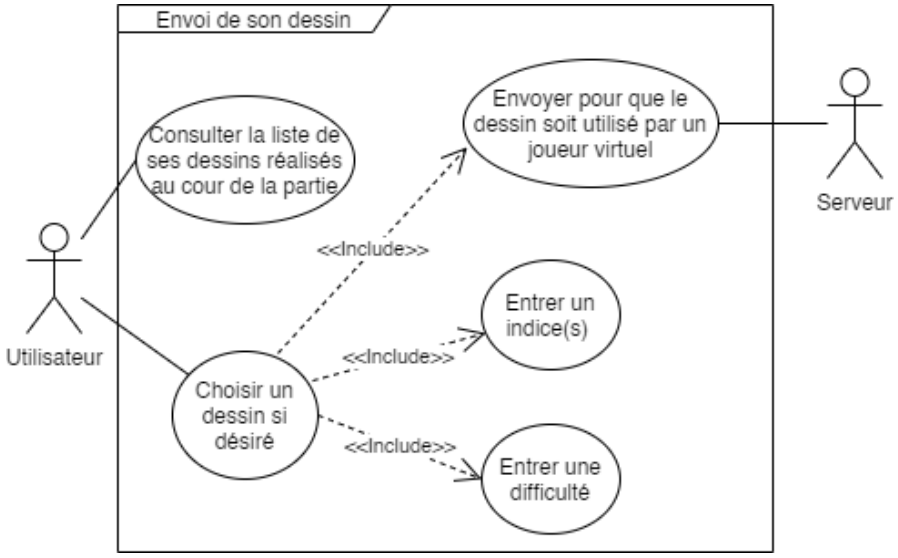

Figure 8. Diagramme de cas d'utilisation pour l'envoi de dessins après une partie

3.9 Cas d'utilisation des parties privées

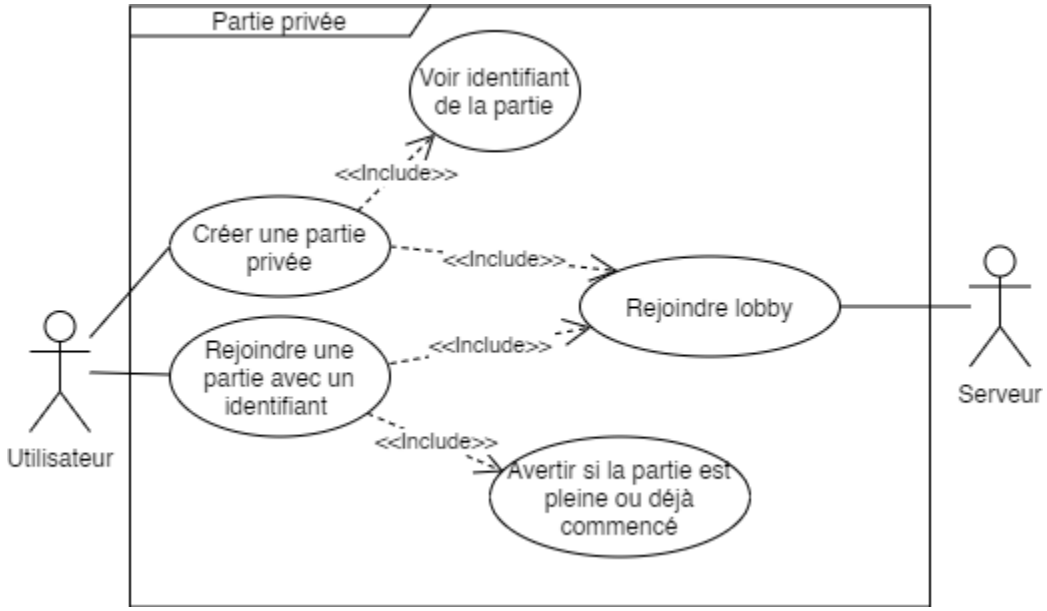

Figure 9. Diagramme de cas d'utilisation pour les parties privées

3.10 Cas d'utilisation du fil d'image des parties jouées

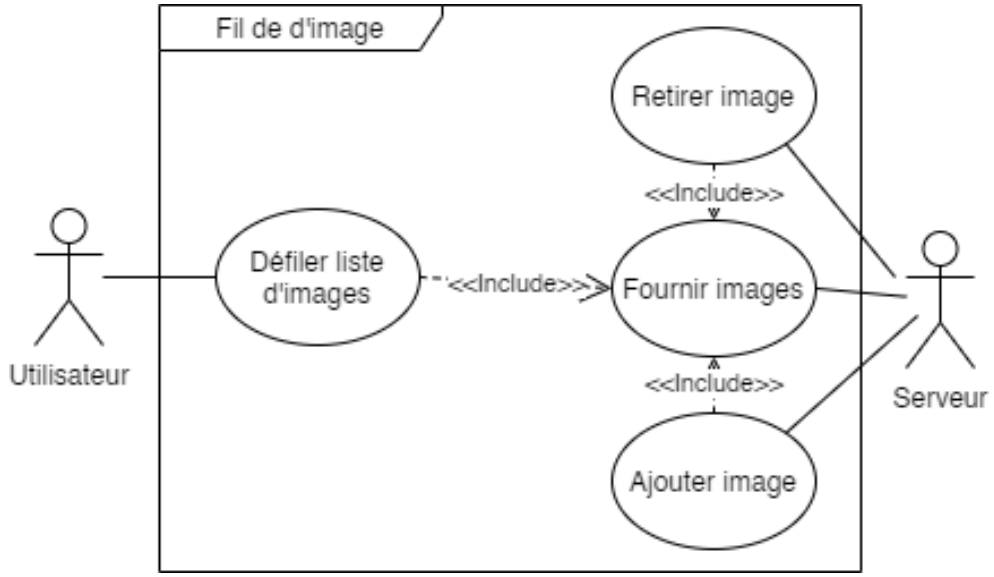

Figure 10. Diagramme de cas d'utilisation pour le fil d'images

3.11 Cas d'utilisation du vote des dessin du joueur virtuel

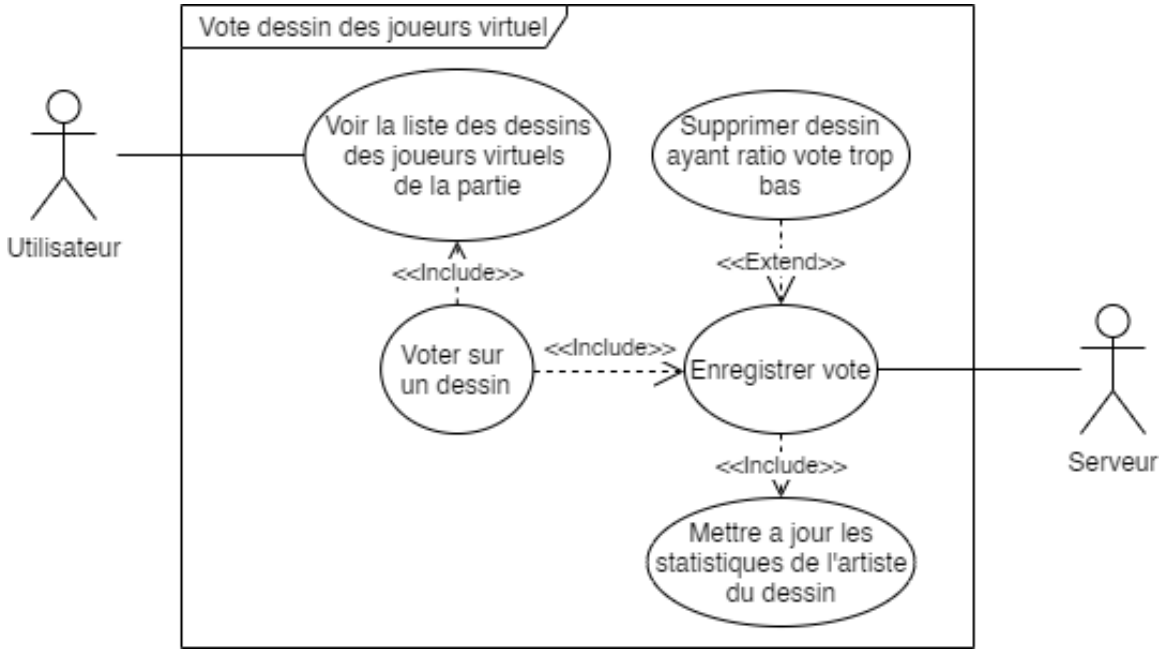

Figure 11. Diagramme de cas d'utilisation pour les votes de dessins

4. Vue logique

4.1 Serveur

4.1.1 Diagramme de paquetages

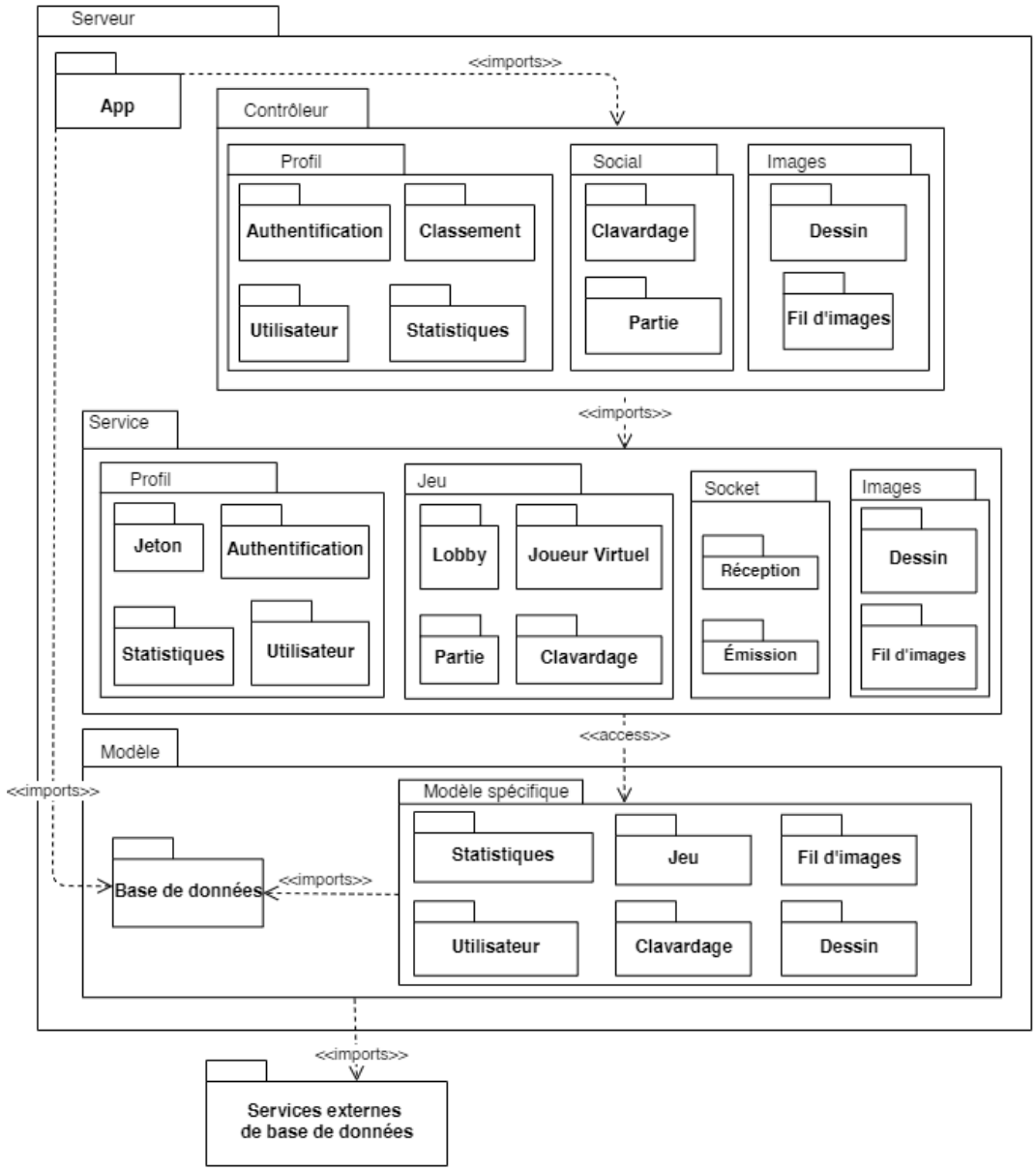

Figure 12. Diagramme de paquetages pour le serveur

4.1.1.1 App

Il s'agit du point d'entrée de l'application serveur. Ce paquetage contient les fichiers tels que *www.ts*, server.ts et app.ts. C'est à ce niveau que nous initialisons la communication par sockets et que nous redirigeons les requêtes HTTP vers les bons contrôleurs. C'est aussi à ce niveau que nous initialisons la communication avec la base de données.

4.1.1.2 Contrôleur

Les contrôleurs sont chargés de gérer les routes du REST API et des Sockets provenant du client lourd et du mobile. Il n'y a que très peu de logique dans les paquetages des contrôleurs puisque celle-ci est déléguée au paquetage de Service. Les contrôleurs ont donc la responsabilité d'appeler les bons services pour faire le traitement des données qu'ils reçoivent des clients. Les tableaux ci-dessous décrivent les paquetages principaux pour les contrôleurs:

Tableau 1. Description des paquetages de la couche contrôleur du serveur

Nom du paquetage	Description
Profil	Ce paquetage regroupe les contrôleurs responsables de la gestion des routes du REST API en lien avec la création et l'authentification des utilisateurs ainsi que leur statistique et leur classement.
Social	Ce paquetage regroupe les contrôleurs responsables de la gestion des routes du REST API en lien avec les activités sociales de l'application soit les canaux de clavardage et les parties.
Images	Ce paquetage regroupe les contrôleurs responsables de la gestion des routes du REST API en lien avec la gestion des images soit la création de dessins et le fil d'image.

4.1.1.3 Service

Ce paquetage contient toute la logique de l'application serveur. Les modules de ce paquetage ont la responsabilité de faire le traitement des données reçu de l'application client et de la base de données avant de retourner une réponse au client. Ces modules peuvent donc utiliser les modules du paquetage de niveau modèle pour interagir avec la base de données. Les tableaux ci-dessous décrivent les paquetages principaux pour les services:

Tableau 2. Description des paquetages de la couche service du serveur

Nom du paquetage	Description
Profil	Ce paquetage regroupe les services responsables de la logique relative à la création et l'authentification des utilisateurs ainsi que le maintien de leurs statistiques.
Jeu	Ce paquetage regroupe les services responsables de la logique relative à la gestion des lobbies, des parties et des canaux de discussions ainsi que la gestion des différents joueurs virtuels présents dans les parties.
Socket	Ce paquetage regroupe les services responsables de la logique relative à la réception des événements socket et leur acheminement vers les services appropriés ainsi que la logique qui permet l'émission d'événement socket à partir de divers services .
Images	Ce paquetage regroupe les services responsables de la logique qui gère la réception

	de paires mot-images ainsi que le maintien à jour du fil d'image.

4.1.1.4 Modèle

Ce paquetage contient tous les modules qui devront interagir avec les différentes collections (tables) de notre base de données. Ces modules ne contiennent que la logique nécessaire pour interagir avec la base de données. Le traitement des informations reçues devra se faire au niveau des services qui utilisent les modèles pour obtenir ces informations. Les tableaux ci-dessous décrivent les paquetages principaux pour les modèles:

Tableau 3. Description des paquetages de la couche modèle du serveur

Nom du paquetage	**Description**
Base de données	Ce paquetage regroupe les modèles responsables de la gestion de la base de données comme telle. Par exemple, la connexion du serveur avec celle-ci. Tous les autres modèles utilisent ce paquetage pour avoir accès à la BD.
Informations utilisateurs	Ce paquetage regroupe les modèles responsables de la communication avec la BD pour ce qui est relatif aux identifiants de connexion des utilisateurs .
Utilisateur	Ce paquetage regroupe les modèles responsables de la communication avec la BD pour ce qui est relatif aux informations propres aux utilisateurs.
Jeu	Ce paquetage regroupe les modèles responsables de la communication avec la BD pour ce qui est relatif aux informations propres aux parties.
Clavardage	Ce paquetage regroupe les modèles responsables de la communication avec la BD pour ce qui est relatif aux canaux de discussion des utilisateurs.
Dessin	Ce paquetage regroupe les modèles responsables de la communication avec la BD pour ce qui est relatif aux informations des dessins.
Fil d'images	Ce paquetage regroupe les modèles responsables de la communication avec la BD pour ce qui est relatif au fil d'actualité.

4.1.2 Diagramme de classes

4.1.2.1 Jeux

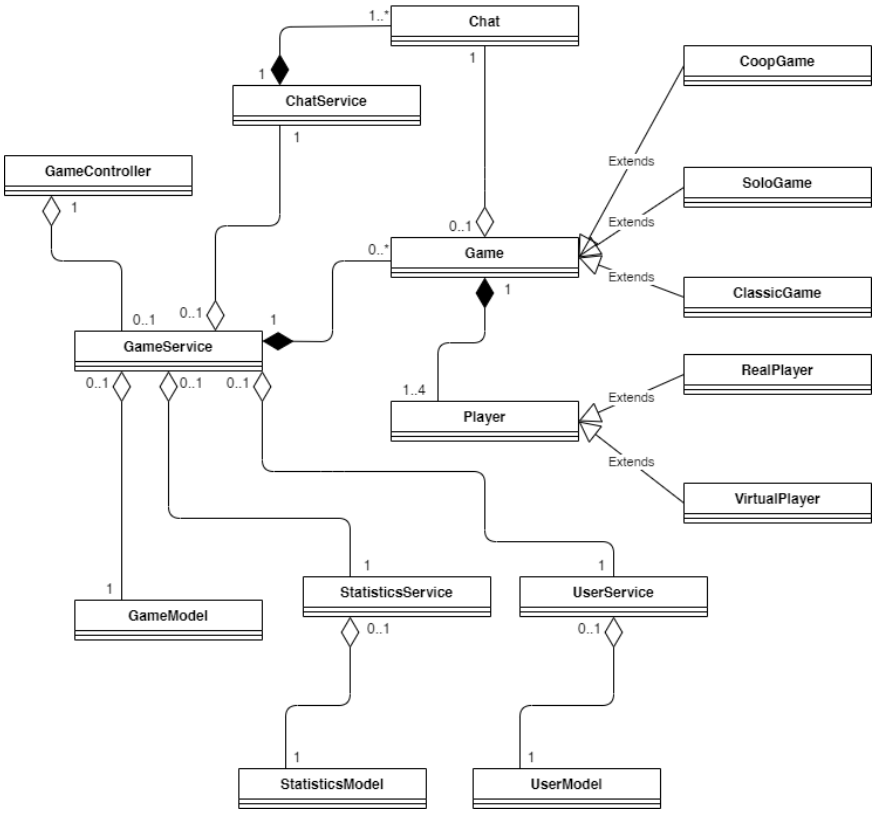

Figure 13. Diagramme de classes pour les jeux

4.1.2.2 Clavardage

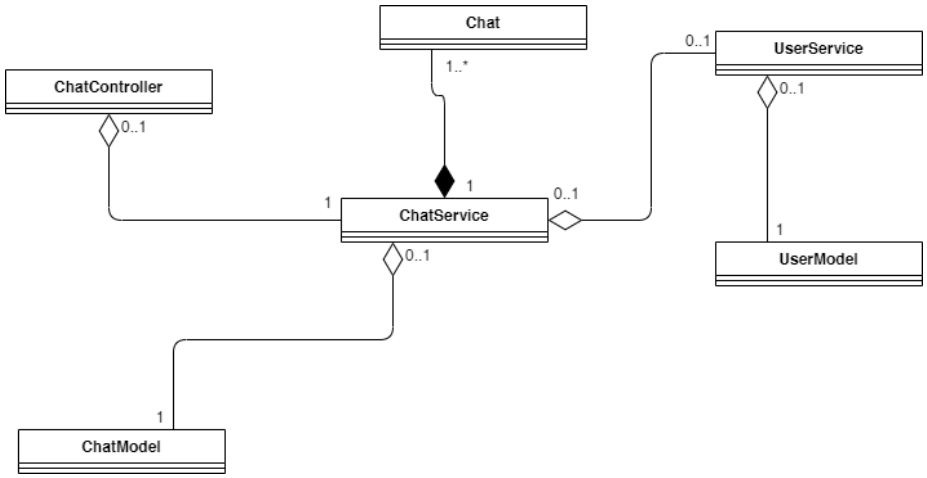

Figure 14. Diagramme de classes pour le clavardage

4.1.2.3 Authentification

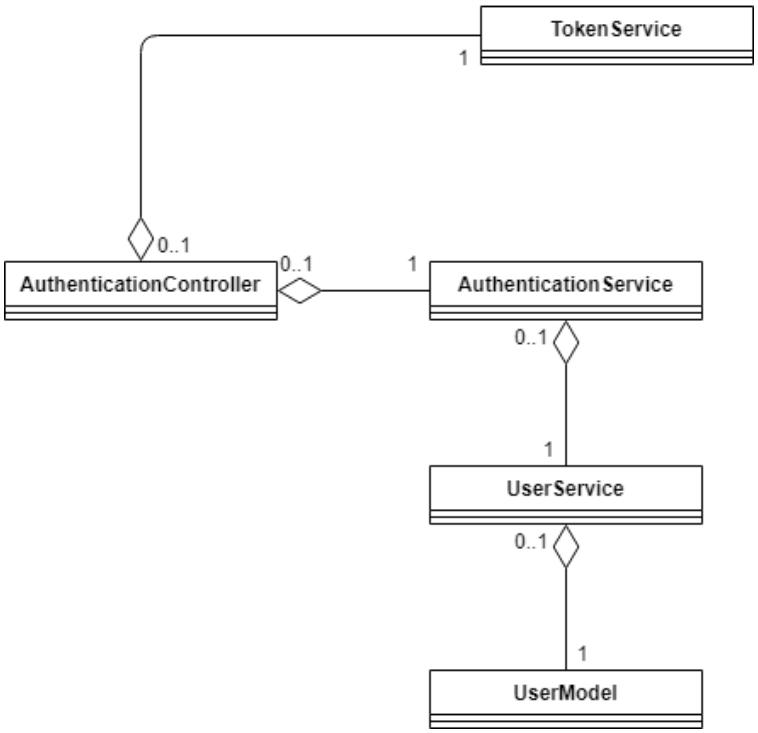

Figure 15. Diagramme de classes pour l'authentification

4.1.2.4 Statistiques

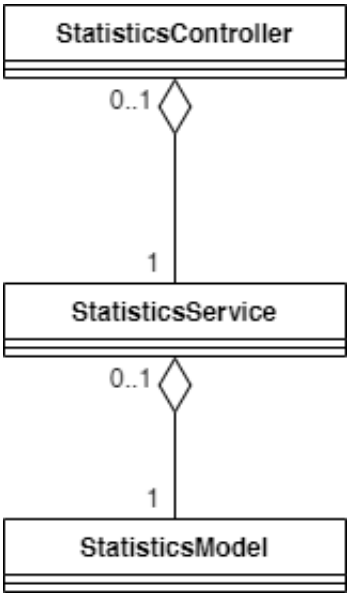

Figure 16. Diagramme de classes pour les statistiques

4.1.2.5 Utilisateur

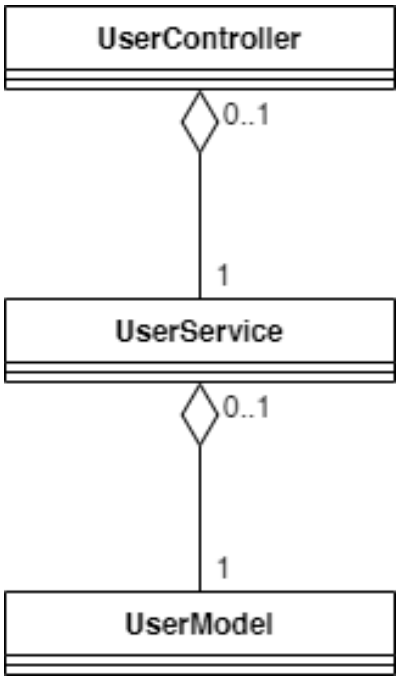

Figure 17. Diagramme de classes pour l'utilisateur

4.1.2.6 Joueur virtuel

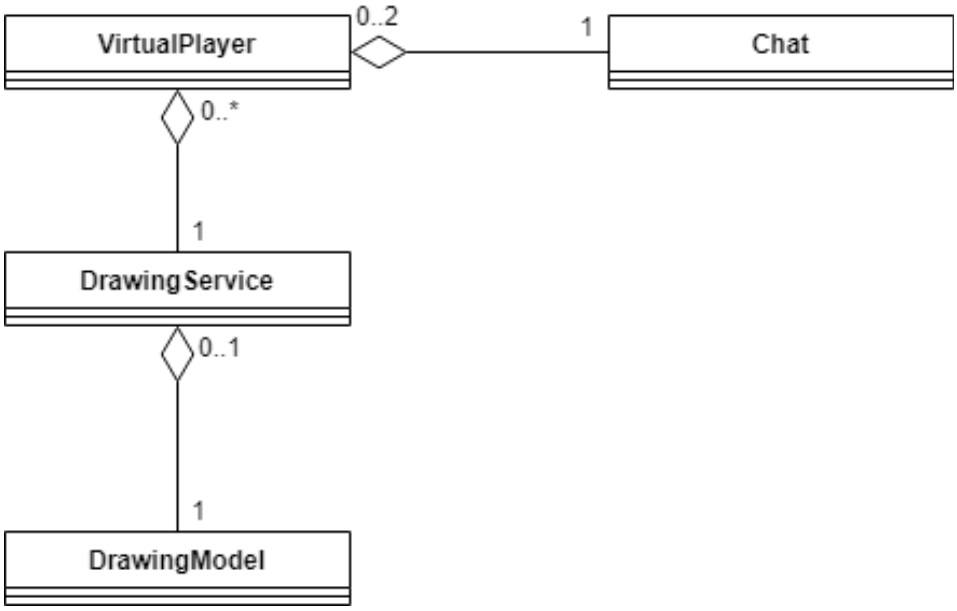

Figure 18. Diagramme de classes pour le joueur virtuel

4.2 Client Léger

Le client léger est bâti selon une architecture MVVM. Cette architecture est composée de trois couches: vue, vue modèle et modèle. La vue s'occupe de regrouper les classes d'interface utilisateur. La couche vue modèle permet d'adapter les données du modèle pour la vue. La vue observe les changements de la couche vue modèle pour adapter les données qu'elle affiche. La couche vue modèle s'occupe de communiquer au modèle les changements à apporter aux données. Le modèle avertit la couche vue modèle si des données ont changé. Le diagramme à la figure 22 indique les principaux paquetages que l'on peut retrouver dans les différentes couches.

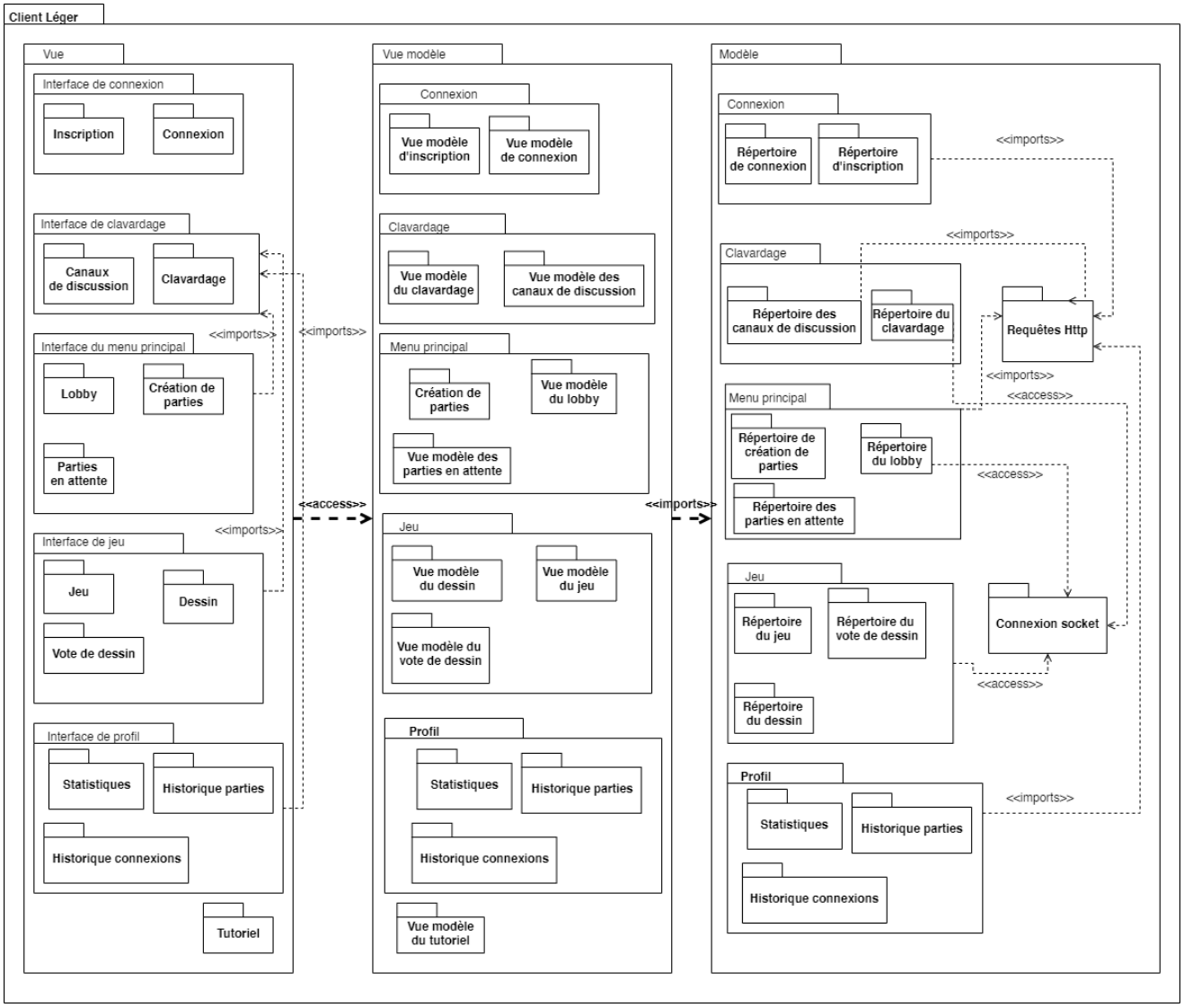

Figure 19. Diagramme de paquetages pour le client léger

4.2.1 Vue

Tableau 4. Description des paquetages de la couche vue du client léger

Nom du paquetage	Description
Interface de connexion	Ce paquetage regroupe les classes d'interface utilisateur qui s'occupent de la connexion et de l'inscription de l'utilisateur.
Interface de clavardage	Ce paquetage regroupe les classes d'interface utilisateur qui s'occupent des canaux de discussion et du clavardage.
Interface du menu principal	Ce paquetage regroupe les classes d'interface utilisateur qui s'occupent de présenter le menu principal qui contient les parties en attente la création de parties et la salle d'attente que l'utilisateur a rejoint.
Interface de jeu	Ce paquetage regroupe les classes d'interface utilisateur qui s'occupent de présenter l'interface de jeu ainsi que de gérer les dessins reçus du serveur et le vote sur les dessins des joueurs virtuels.
Interface de profil	Ce paquetage regroupe les classes d'interface utilisateur qui s'occupent de présenter le profil de l'utilisateur qui comprend les statistiques ainsi que l'historique de parties et de connexions.
Interface de tutoriel	Ce paquetage regroupe les classes d'interface utilisateur qui s'occupent de présenter le tutoriel.

4.2.3 Vue modèle

Tableau 5. Description des paquetages de la couche vue modèle du client léger

Nom du paquetage	Description
Connexion	Ce paquetage regroupe les classes qui préparent les données provenant du modèle pour l'interface de connexion.
Clavardage	Ce paquetage regroupe les classes qui préparent les données provenant du modèle pour le clavardage et les différents canaux de discussion.
Menu principal	Ce paquetage regroupe les classes qui préparent les données provenant du modèle pour le lobby ainsi que la création de parties et les parties en attente.
Jeu	Ce paquetage regroupe les classes qui préparent les données provenant du modèle pour les modes de jeu incluant les dessins, le vote sur les dessins et les autres éléments du jeu.

Profil	Ce paquetage regroupe les classes qui préparent les données provenant du modèle pour les statistiques ainsi que l'historique des parties et des connexions de l'utilisateur.
Vue modèle du tutoriel	Ce paquetage regroupe les classes qui préparent les données provenant du modèle du tutoriel

4.2.3 Modèle

Tableau 6. Description des paquetages de la couche modèle du client léger

Nom du paquetage	Description
Connexion	Ce paquetage regroupe les classes répertoires qui s'occupent de conserver et d'obtenir les données provenant du serveur pour la connexion et l'inscription de l'utilisateur.
Clavardage	Ce paquetage regroupe les classes répertoires qui s'occupent de conserver et d'obtenir les données provenant du serveur pour les canaux de discussion et le clavardage.
Menu principal	Ce paquetage regroupe les classes répertoires qui s'occupent de conserver et d'obtenir les données provenant du serveur pour le lobby rejoint par l'utilisateur ainsi que pour la création de parties les parties en attentes.
Jeu	Ce paquetage regroupe les classes répertoire qui s'occupent de conserver et d'obtenir les données provenant du serveur pour le jeu ainsi que les dessins et les votes sur certains de ces dessins.
Profil	Ce paquetage regroupe les classes qui s'occupent de conserver les données et d'obtenir les données provenant du serveur pour le profil personnel d'un utilisateur contenant ses statistiques, son historique de connexions et son historique de parties jouées.

4.2.4 Diagrammes de classes

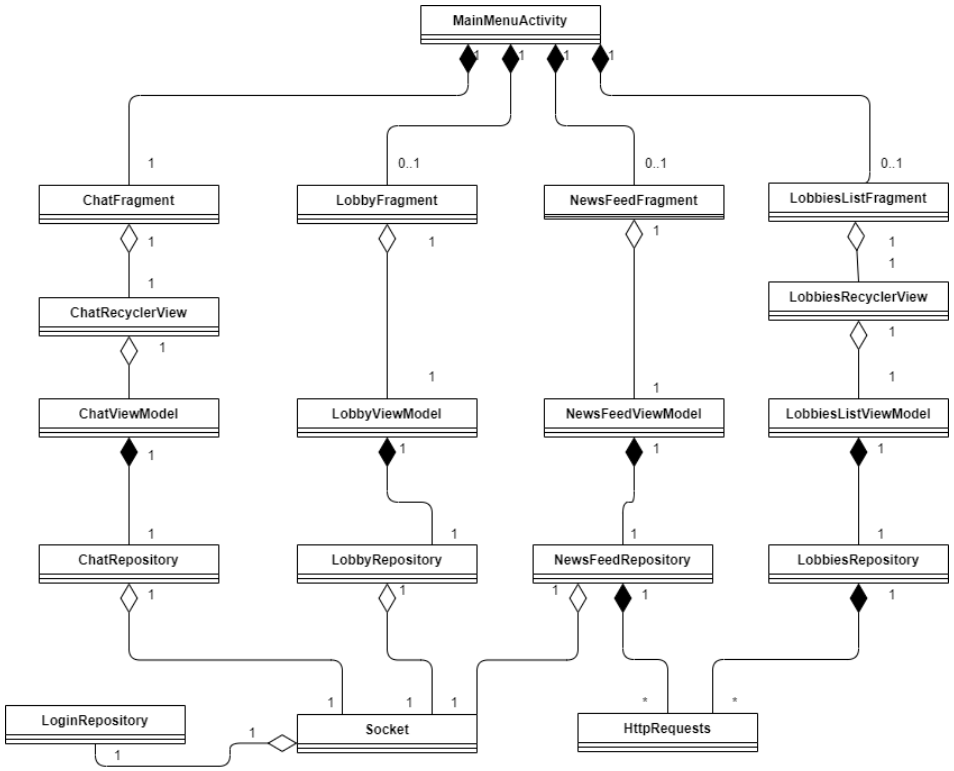

Figure 20. Diagramme de classes pour les différentes composantes du menu principal du client léger

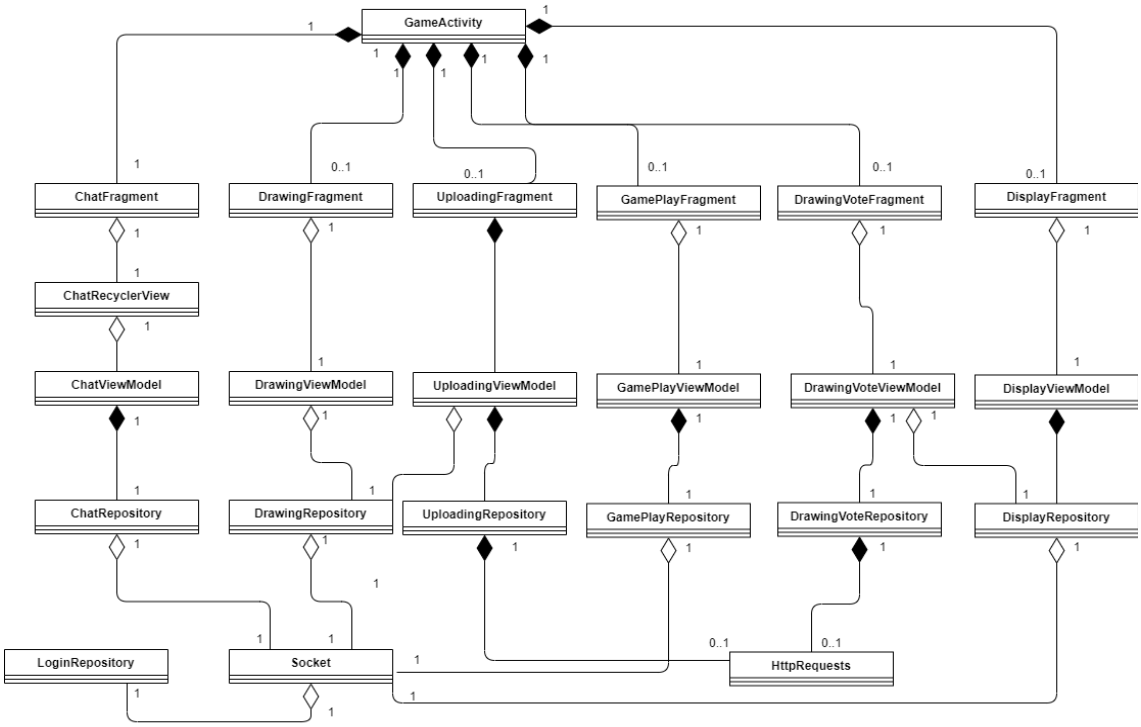

Figure 21. Diagramme de classes pour les différentes composantes de la portion jeu pour le client léger

4.3 Client lourd

Le client lourd est composé de “components” et de “services”. Les “components” regroupent des fichiers Typescript, CSS et HTML qui sont référencés dans le metadata du “component”. Cette séparation permet une réutilisation facile de certains blocs d’éléments visuels en plus de fournir une structure de fichier rigide. Pour le traitement des données et la communication entre “components”, nous utilisons les “services”, dont chaque instance est unique. Le diagramme suivant représente la séparation de la logique du client lourd en “components” et en “services”.

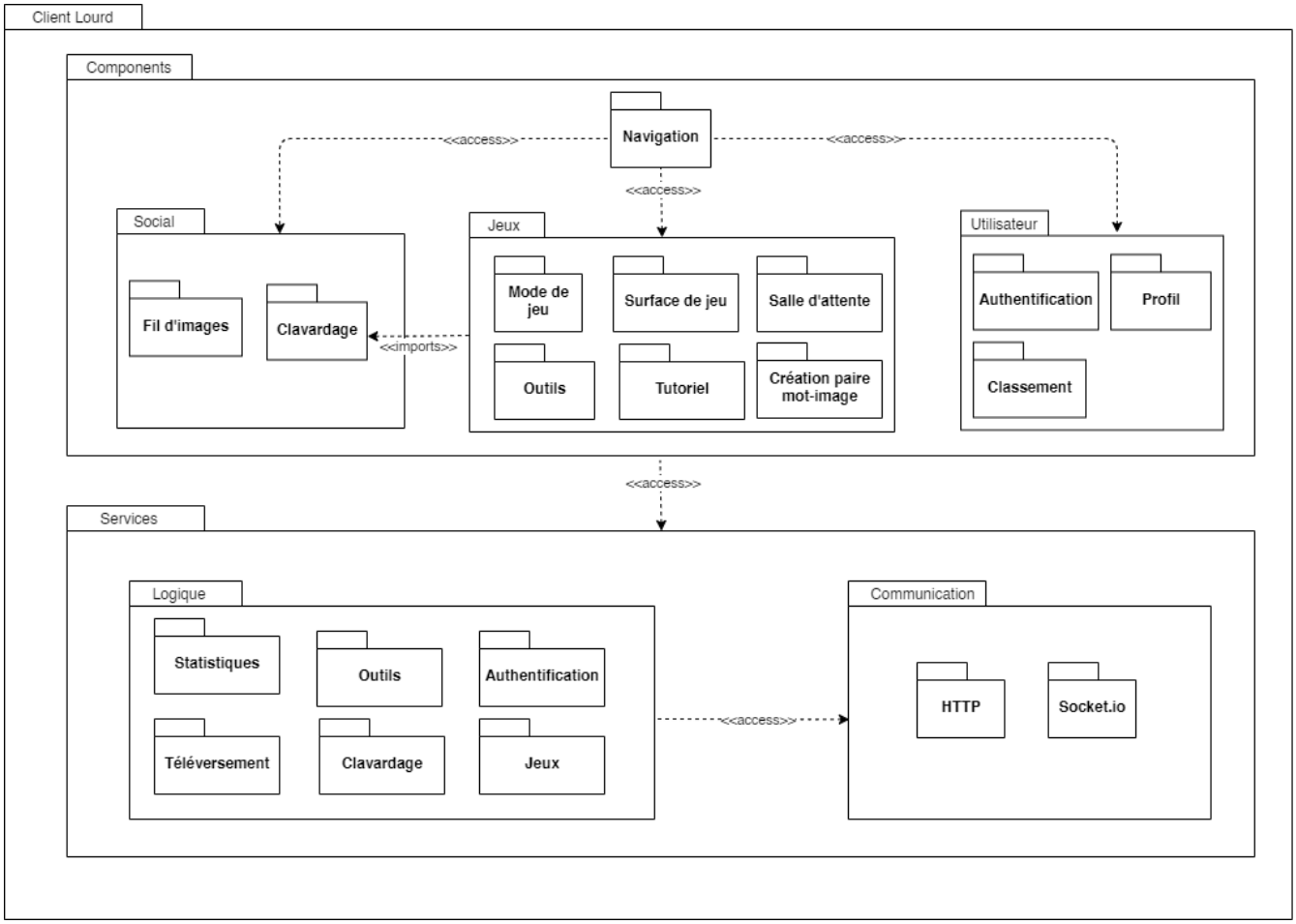

Figure 22. Diagramme de paquetages du client lourd

4.3.1 Component

Tableau 7. Description des paquetages de la couche component du client lourd

Nom du paquetage	Description
Clavardage	Ce paquetage regroupe les composants responsables de la logique du clavardage utilisé dans toute l'application.
Navigation	Ce paquetage regroupe les composants responsables de la navigation entre les différentes composantes de l'application.
Surface de jeu	Ce paquetage regroupe les composants englobants les différentes fonctionnalités utilisées dans le cadre du jeu.
Mode de jeu	Ce paquetage regroupe les composants responsables des différents modes de jeu dans leur entièreté.
Salle d'attente	Ce paquetage regroupe les composants responsables de la salle d'attente d'une partie, incluant l'affichage des parties en cours visible du menu principal.
Tutoriel	Ce paquetage regroupe les composants responsables du tutoriel.
Authentification	Ce paquetage regroupe les composants responsables du mécanisme d'authentification.
Profil	Ce paquetage regroupe les composants responsables de la gestion du profil d'un utilisateur.
Classement	Ce paquetage regroupe les composants responsables de la gestion des tableaux de classement
Fil d'images	Ce paquetage regroupe les composants responsables du fil d'images de la page principale.
Création paire mot-image	Ce paquetage regroupe les composants responsables du mécanisme d'ajout d'une paire mot-image à la banque utilisée par les joueurs virtuels.

4.3.2 Services

Tableau 8. Description des paquetages de la couche service du client lourd

Nom du paquetage	Description
Statistiques	Ce paquetage regroupe les services responsables de la gestion des informations utilisées pour le profil utilisateur et le système de classement.
Outils	Ce paquetage regroupe les services responsables du fonctionnement des outils utilisés pour dessiner sur le canevas.
Authentification	Ce paquetage regroupe les services chargés de traiter les requêtes en lien avec l'authentification.
Téléversement	Ce paquetage regroupe les services responsables du téléversement d'un dessin vers le serveur.
Clavardage	Ce paquetage regroupe les services chargés de gérer les données utilisées par le système de clavardage.
Jeux	Ce paquetage regroupe les services responsables des fonctionnalités pour les modes de jeu qui ne seront pas gérées dans les composants.
HTTP	Ce paquetage regroupe les services responsables des requêtes HTTP.
Socket.io	Ce paquetage regroupe les services responsables des communications utilisant socket.io.

4.3.3 Diagramme de classes des principales fonctionnalités du client lourd

4.3.3.1 Authentification

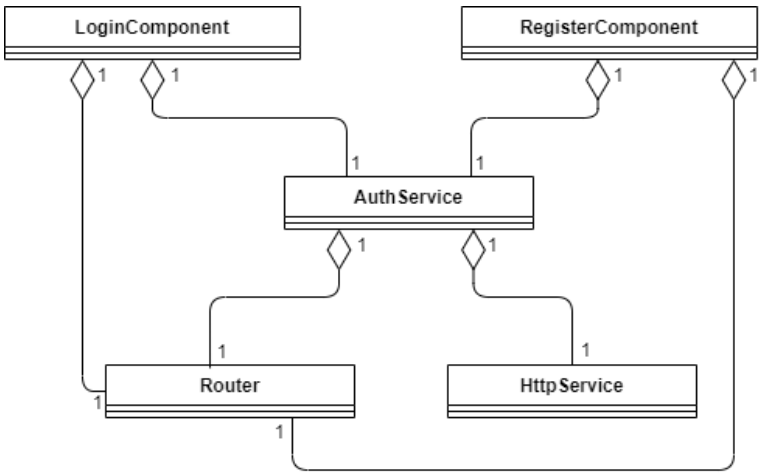

Figure 23. Diagramme de classes du système d'authentification

4.3.3.2 Dessiner

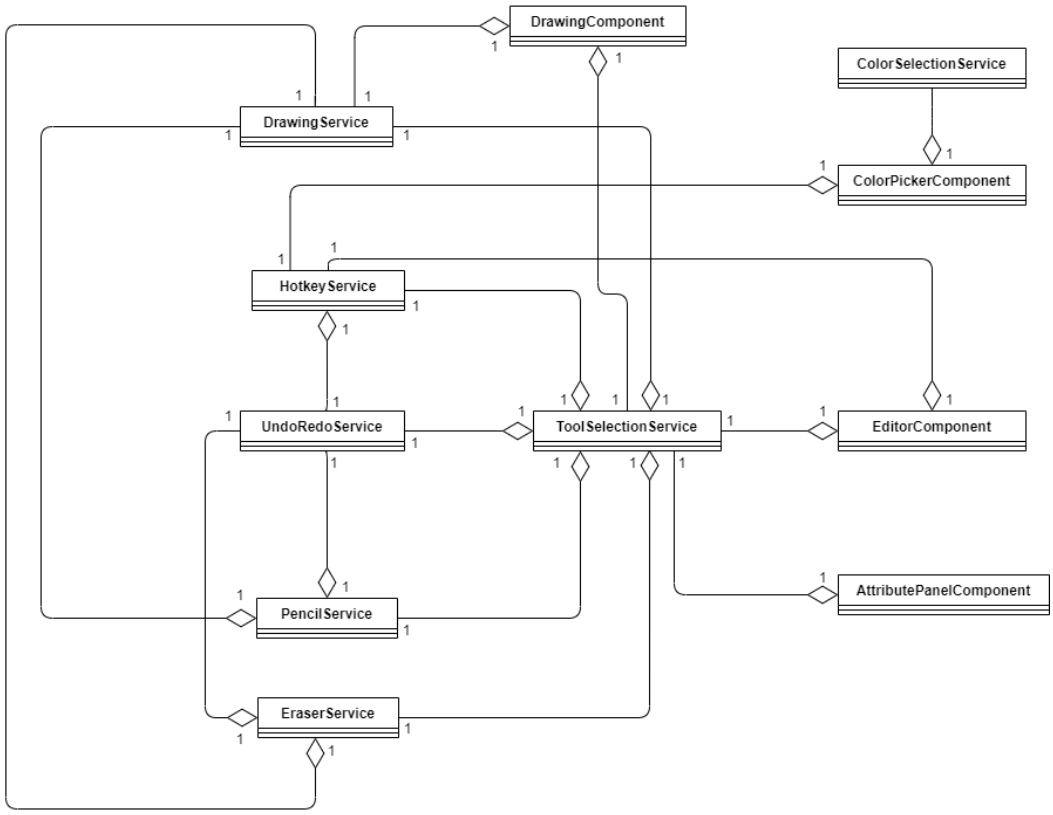

Figure 24. Diagramme de classes pour le système de dessin

4.3.3.3 Clavardage

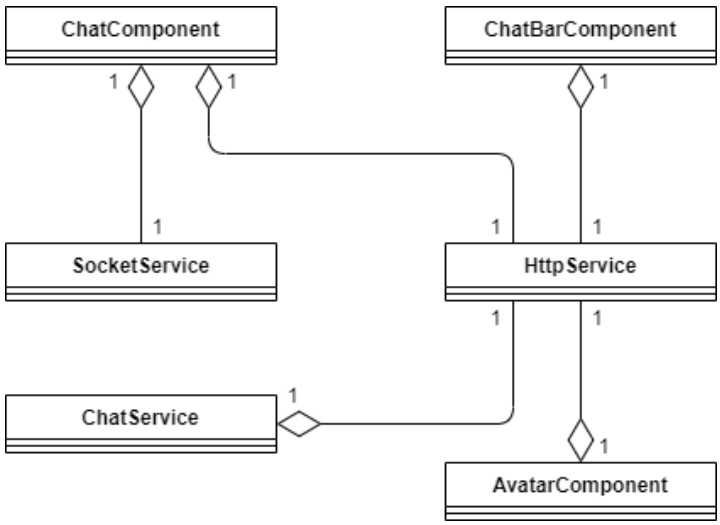

Figure 25. Diagramme de classes pour le système de clavardage

4.3.3.4 Mécanisme de jeu

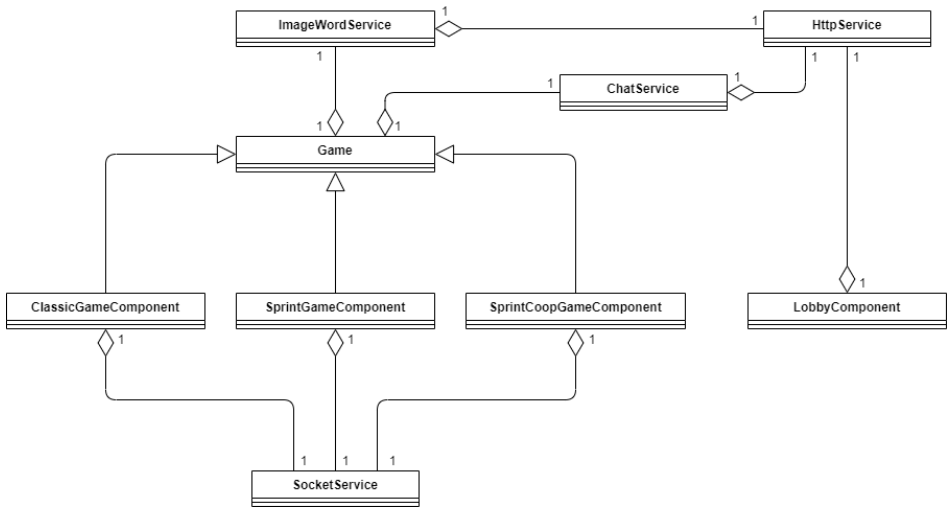

Figure 26. Diagramme de classes pour le système de jeu

5. Vue des processus

5.1 Connexion

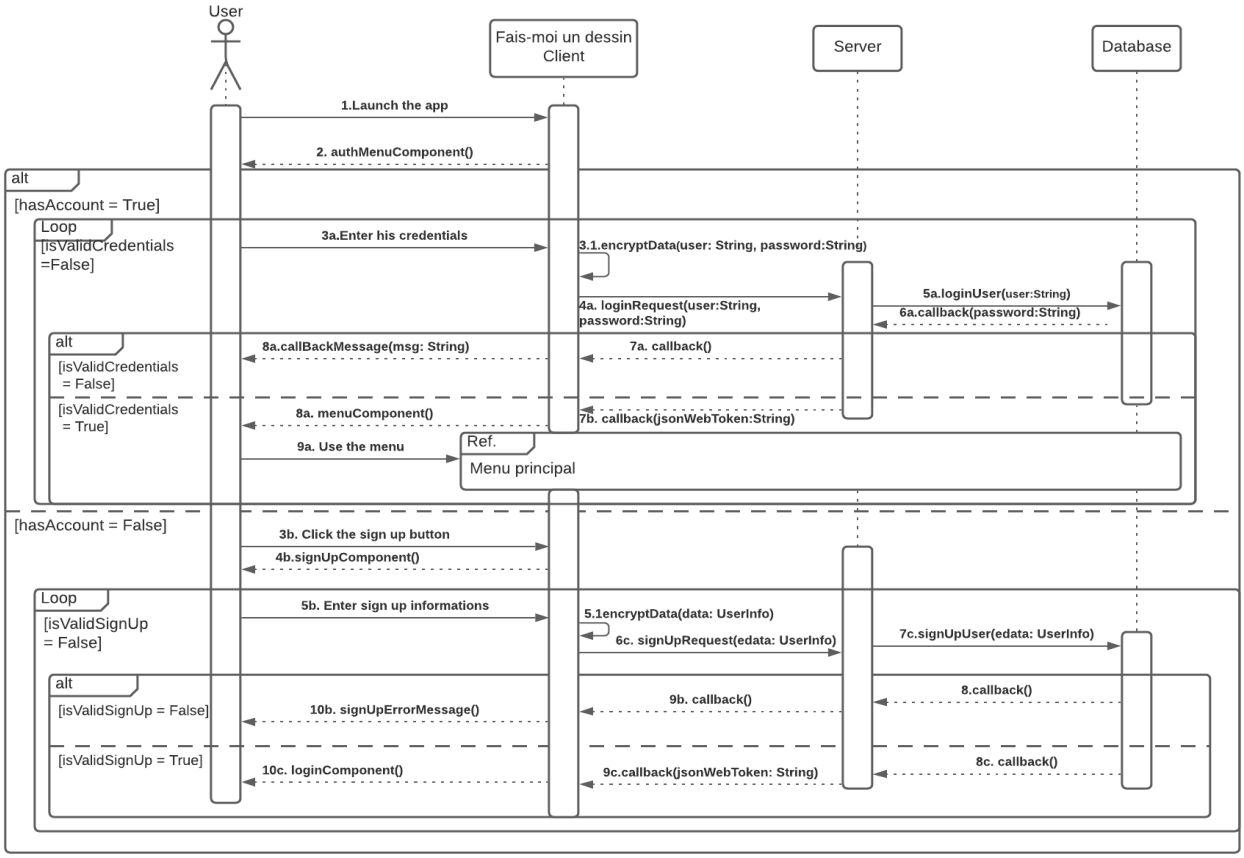

Figure 27. Diagramme de séquence du système de connexion

5.2 Création d'une partie

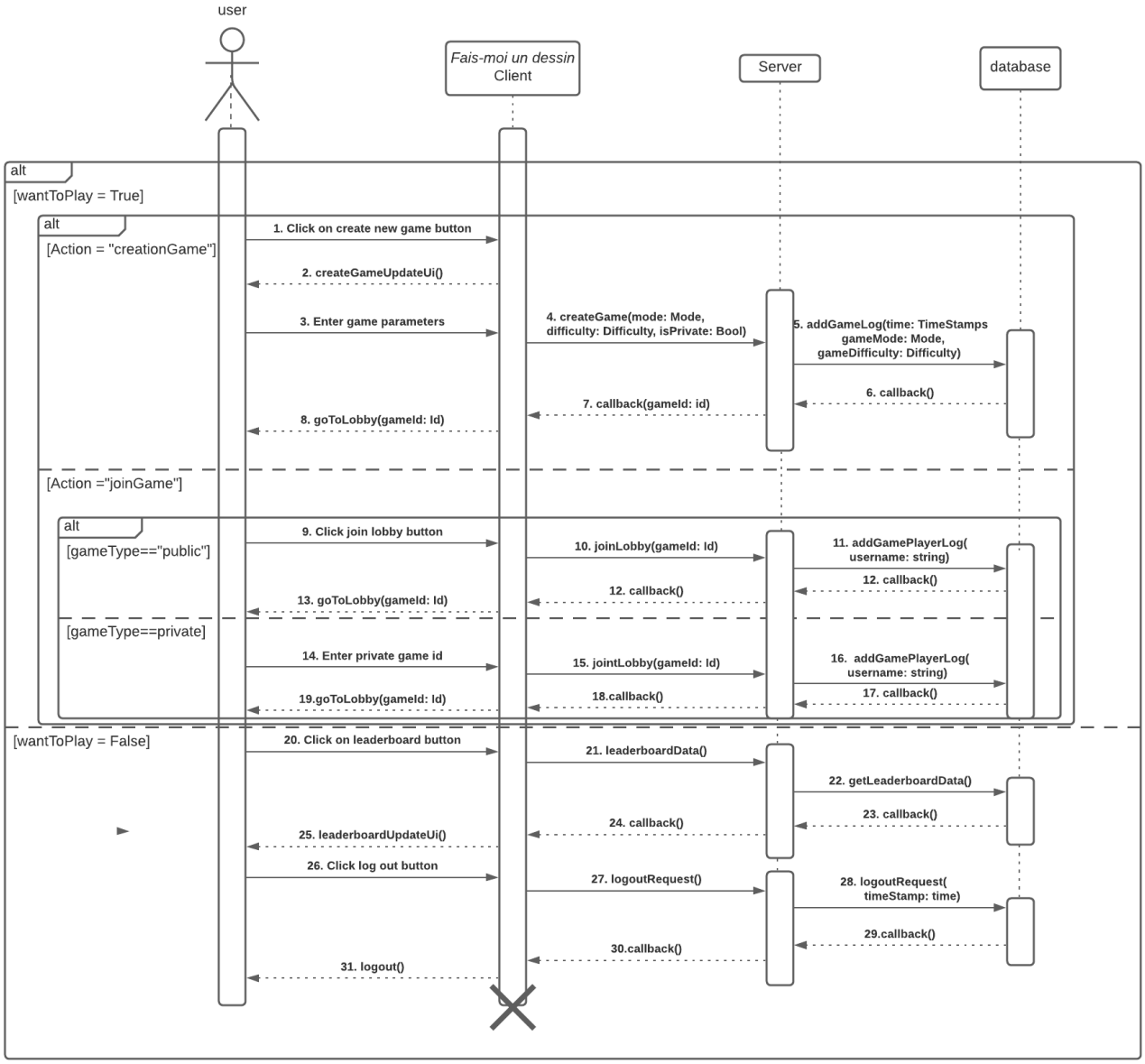

Figure 28. Diagramme de séquence pour la création d'une partie

5.3 Partie sprint solo et coopérative

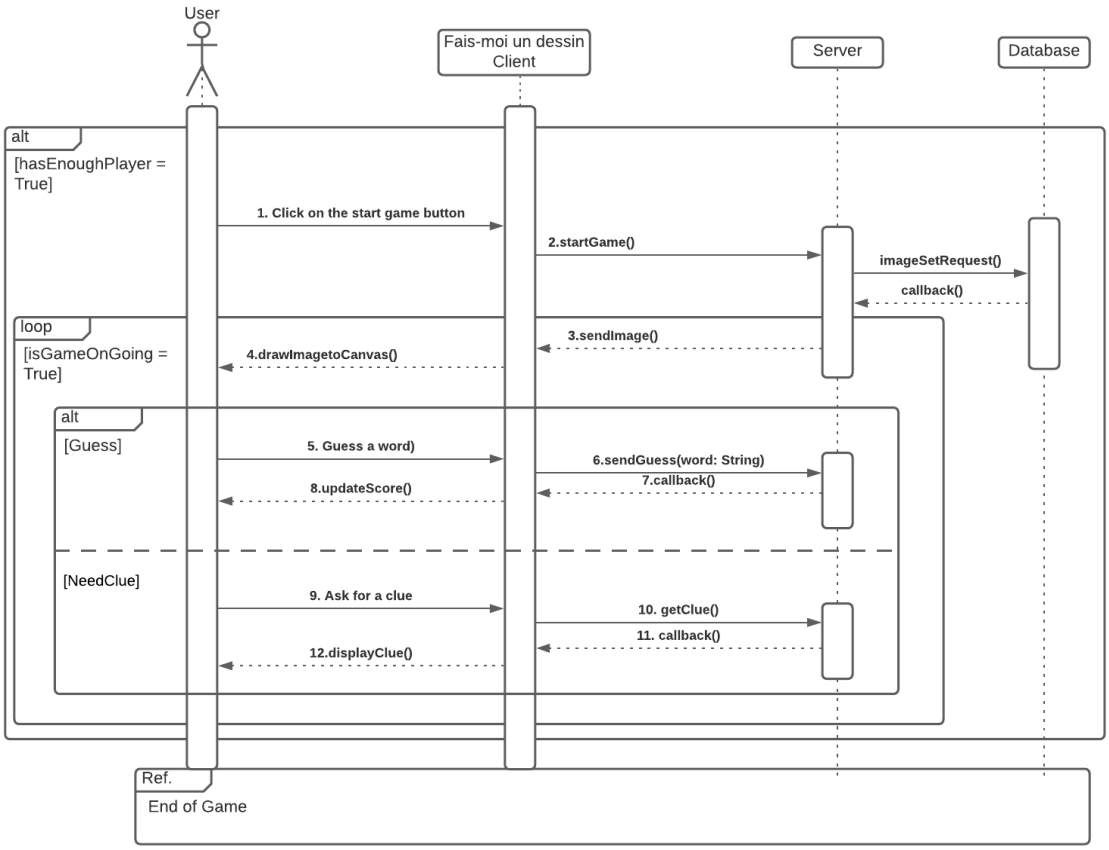

Figure 29. Diagramme de séquence pour les modes de jeu sprint solo et coopératif

5.4 Partie mode classique

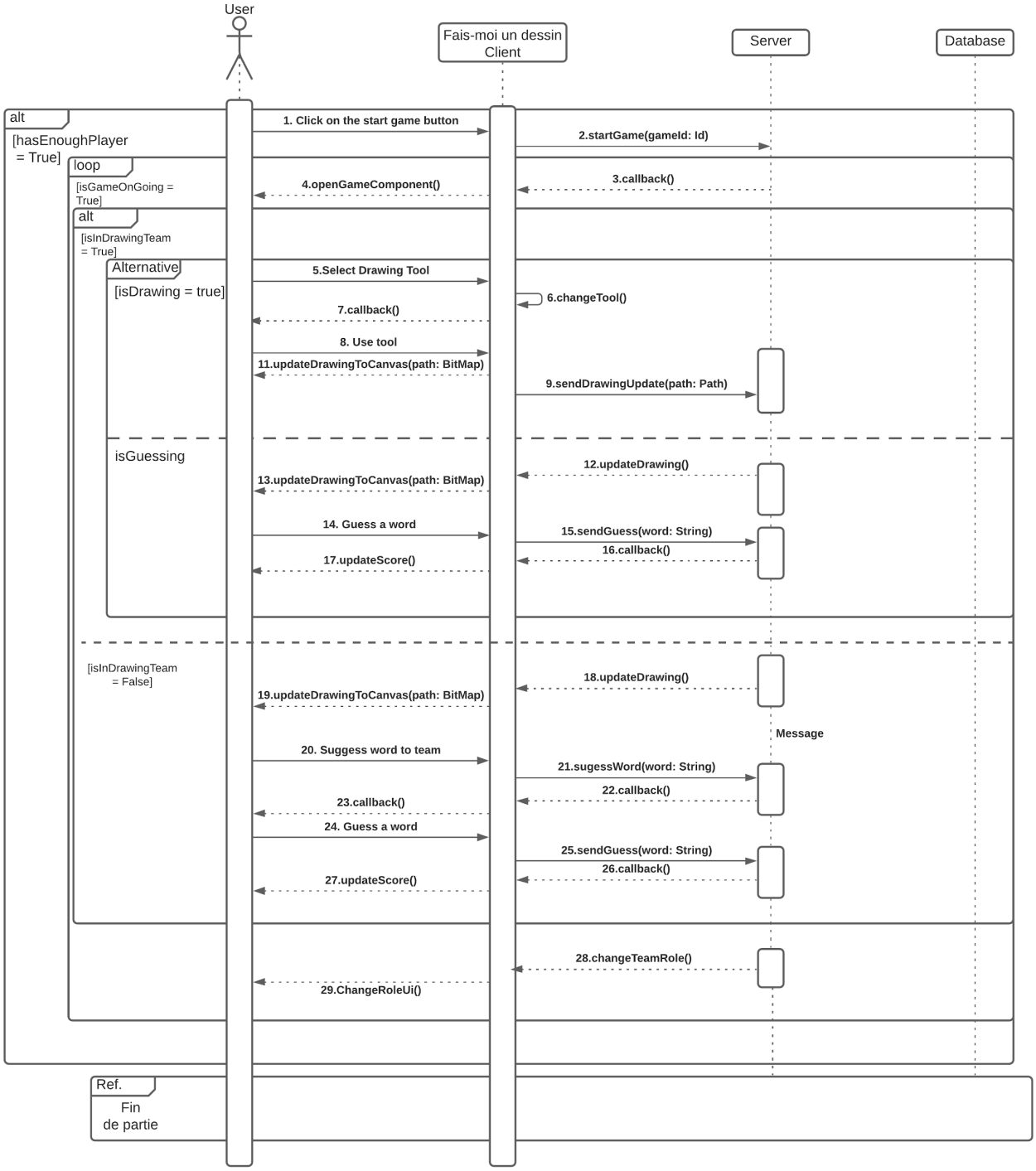

Figure 30. Diagramme de séquences pour le mode de jeu classique

5.5 Fin de partie

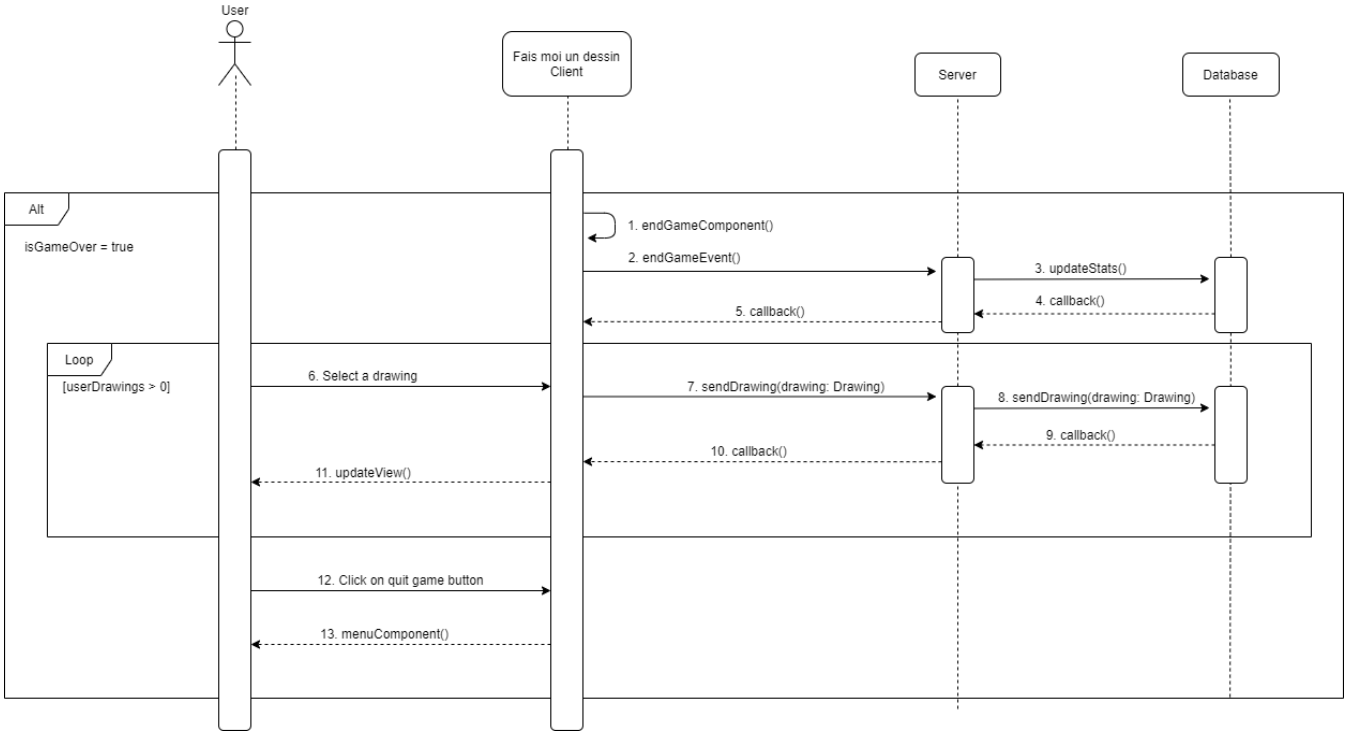

Figure 31. Diagramme de séquence pour la fin d'une partie

5.6 Clavardage

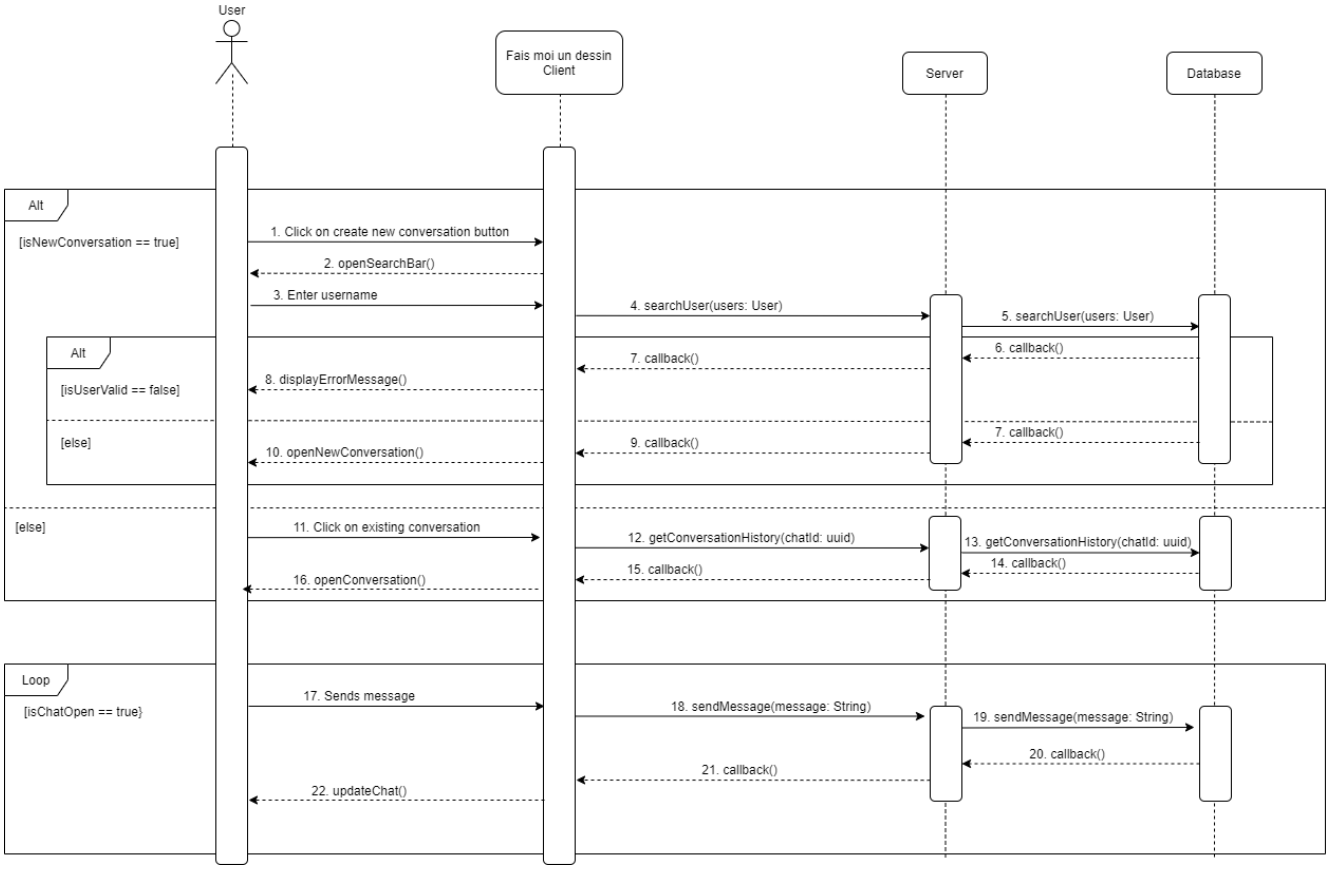

Figure 32. Diagramme de séquence pour le système de clavardage

5.7 Tutoriel

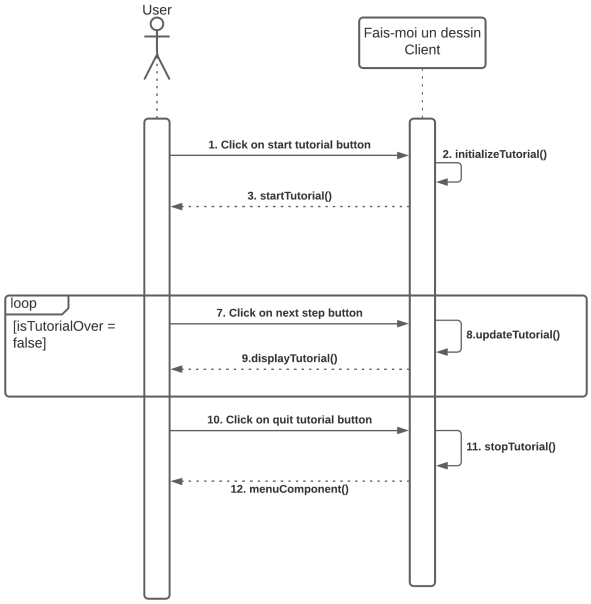

Figure 33. Diagramme de séquence pour le tutoriel

5.8 Requêtes au serveur

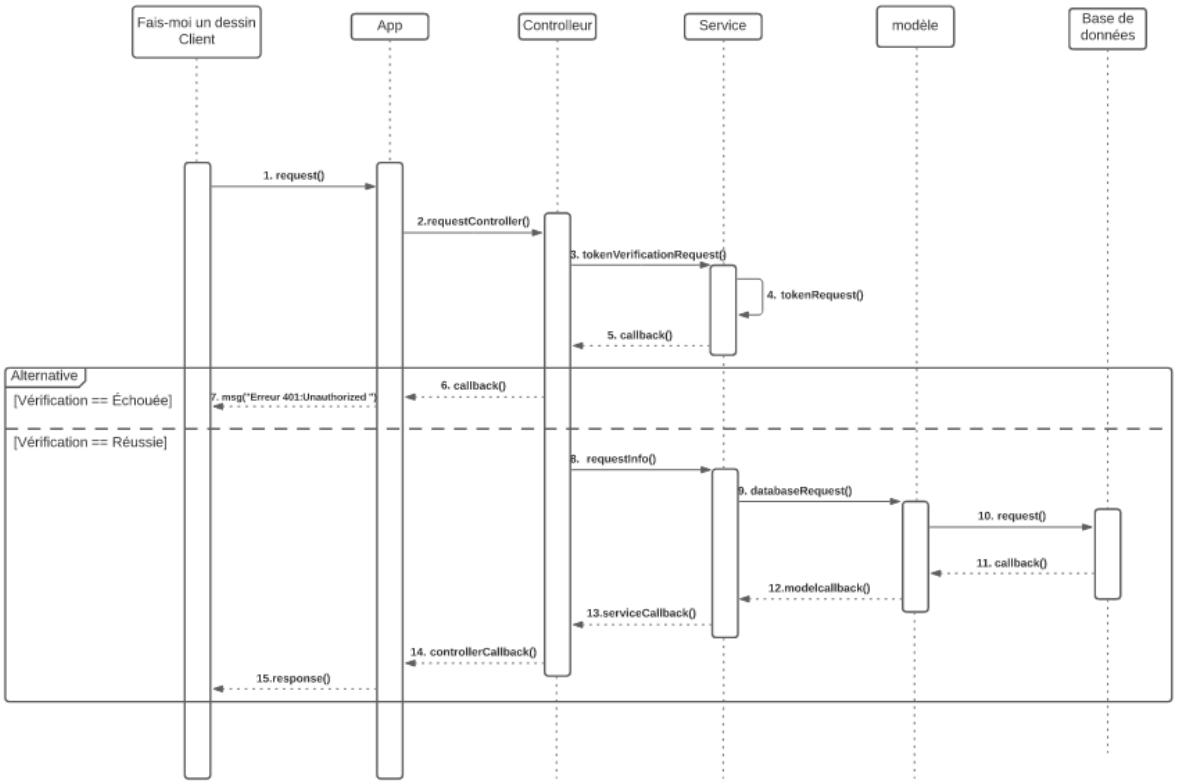

Figure 34. Diagramme de séquence pour le mécanisme de requêtes au serveur

6. Vue de déploiement

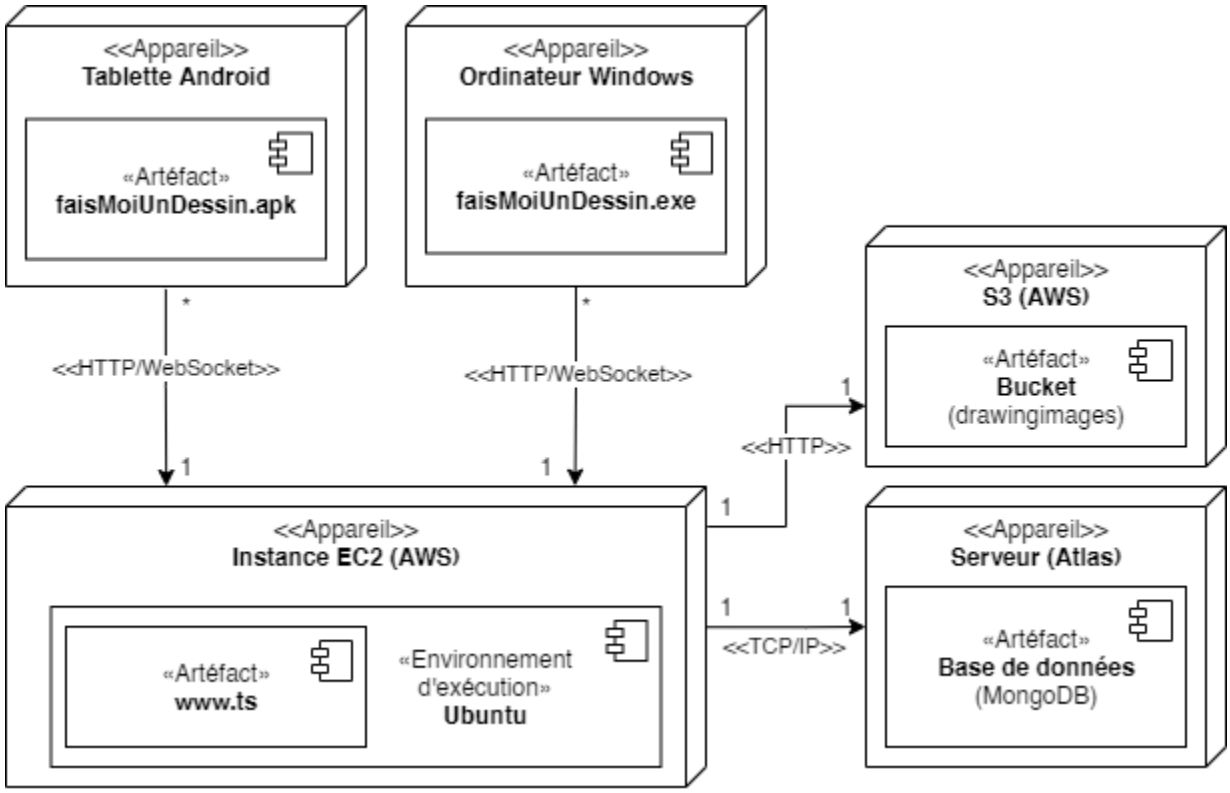

Figure 35. Diagramme de déploiement du système

7. Taille et performance

Pour la taille et la performance de l'application, plusieurs éléments sont à prendre en considération. Le serveur, le client lourd ainsi que le client léger fonctionnent sur des appareils ayant des capacités très différentes. Le serveur, construit à l'aide de la plateforme Node.js, sera hébergé par Amazon Web Service. En utilisant Amazon Web Service, nous n'avons pas à ajuster le serveur en fonction de la mémoire requise ou de l'intensité des calculs, car il est possible d'obtenir ces éléments à la demande. Par contre, utiliser beaucoup de mémoire ou utiliser des algorithmes intensifs en calcul à un coût monétaire. Il faudra donc tenter de les limiter.

Pour le client lourd, l'application doit fonctionner sur un ordinateur avec 8 Go de mémoire vive et un processeur i5. Nous souhaitons limiter l'utilisation de la RAM à 300 Mo et que l'utilisation du processeur ne dépasse pas 10% pour s'assurer que l'expérience utilisateur soit plaisante et que l'utilisateur puisse continuer d'utiliser d'autres applications en même temps. De plus, l'application ne devra pas dépasser 500 Mo d'espace disque.

Pour le client léger, l'application doit fonctionner sur une tablette Android possédant 2 Go de mémoire vive, 32 Go de mémoire interne ainsi qu'un processeur Exynos 7904A. Le client léger doit donc fonctionner sur un appareil moins performant que le client lourd. La mémoire vive est particulièrement limitée sur l'appareil et il sera important de limiter son utilisation à 100Mo.

Pour le serveur, l'application doit fonctionner sur une instance d'Amazon EC2. Le serveur doit donc rouler sur une instance t2.small Linux. Les images des dessins seront stockées sur Amazon S3 et ne doivent pas dépasser 5Go de mémoire tous ensemble. De plus, pour que les différents modes de jeu soient optimaux, il serait préférable que la latence entre les communications entre le serveur et le client demeure en dessous de 200 millisecondes.