<Équipe 101>

<Erratum>
Document d'architecture logicielle

Version 1.07

Historique des révisions

Date	Version	Description	Auteur
21/09/23	1.01	Cas d'utilisation	Nina Lounici
28/09/23	1.02	Description des paquetages	Nina Lounici
28/09/23	1.03	Vue de processus	Mohamed Reda Rhanmouni
28/09/23	1.04	Ajout de paquetages	Etienne Desclaux
29/09/23	1.05	Taille et performance	Nassour Nassour
29/09/23	1.06	Vue de processus	Amirmasood Dadkhah
29/09/23	1.07	Révision complète du document	Mohamed Reda Rhanmouni, Nina Lounici, Etienne Desclaux, Nassour Nassour, Ghali Chraibi, Amirmasood Dadkhah

Table des matières

1. Introduction 4
2. Objectifs et contraintes architecturaux 4
3. Vue des cas d’utilisation 4
4. Vue logique 9
5. Vue des processus 16
6. Vue de déploiement 21
7. Taille et performance 22

Document d'architecture logicielle

1. Introduction

Ce document présente l'architecture de l'application Erratum. La section « Vue des cas d'utilisation » comprend les diagrammes des principaux cas d'utilisation. La « Vue des processus » présente l'interaction entre les différents processus à l'aide de diagrammes de séquences. La « Vue logique » contient un ensemble de tableaux énumérant les parties importantes du modèle de design ainsi que leurs rôles. Elle comporte par la suite les diagrammes de paquetages qui regroupent les composantes ayant une même responsabilité ainsi que les diagrammes de classes associés. Finalement, la « Vue de déploiement » représente l'organisation physique de l'application et les protocoles utilisés grâce à un diagramme de déploiement.

2. Objectifs et contraintes architecturaux

Nous ne disposons que de deux mois entre la remise de l'appel d'offre ainsi que la remise du produit final. L'échéancier représente donc une réelle contrainte dans notre projet. Dans ce délai, nous ne pouvons pas débuter l'application de zéro. Ainsi, l'application Erratum se basera sur un projet déjà existant : la plateforme de jeu en ligne GiaDifference. Nous allons réutiliser la structure client-serveur préexistante et ajuster l'architecture afin d'intégrer les nouvelles fonctionnalités et nouvelles plateformes. Nous sommes donc contraints d'utiliser les mêmes langages de programmation et continuerons donc avec Angular et TypeScript pour le client lourd. Nous avons également décidé de maintenir une base de données orientée document : nous pensons en effet disposer de trop peu de temps pour changer la structure de la base de données, ou encore du serveur. Le client déjà présent représentera le client lourd seulement, et nous allons ajouter le client léger. Ces clients devront communiquer avec le serveur et la base de données.
L'objectif étant de déployer un serveur qui soit stable et fiable, nous avons eu besoin de choisir une instance pour l'héberger. Nous sommes contraints d'un point de vue financier et avons donc choisi AWS.

3. Vue des cas d'utilisation

En troisième partie, nous présenterons les diagrammes des principaux cas d'utilisation de l'application Erratum.

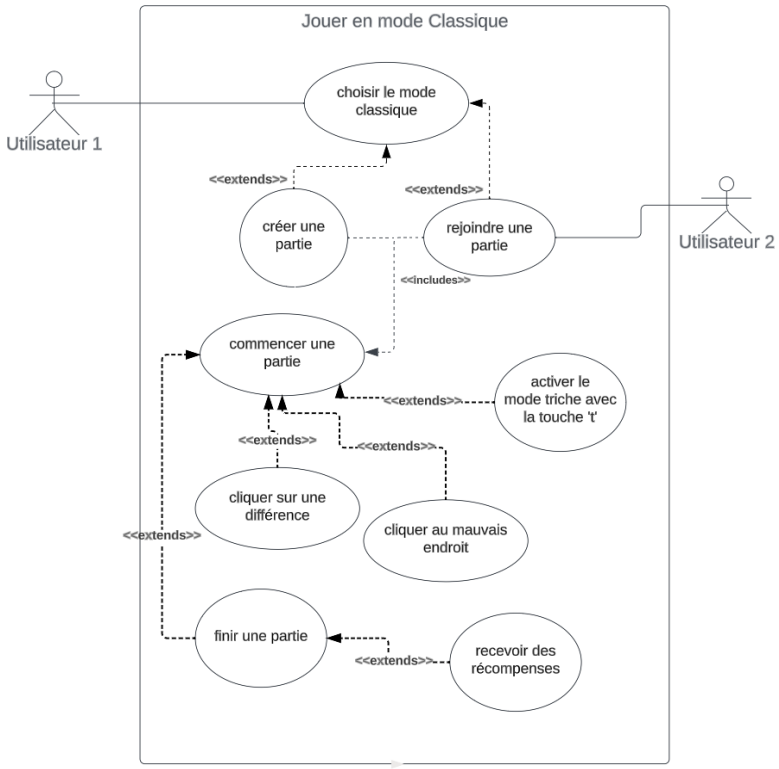

Figure 1 : Diagramme du cas d'utilisation 'Jouer une partie en mode classique'

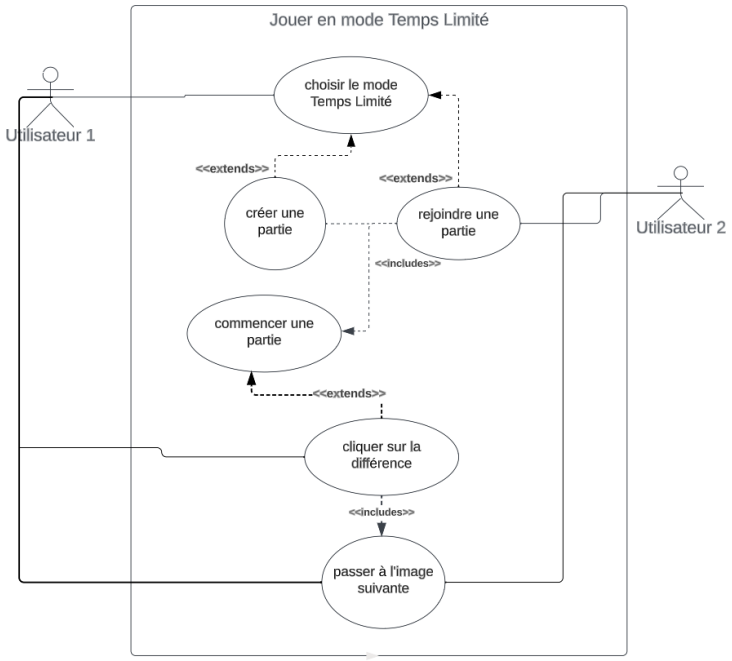

Figure 2 : Diagramme du cas d'utilisation 'Jouer une partie en mode temps limité

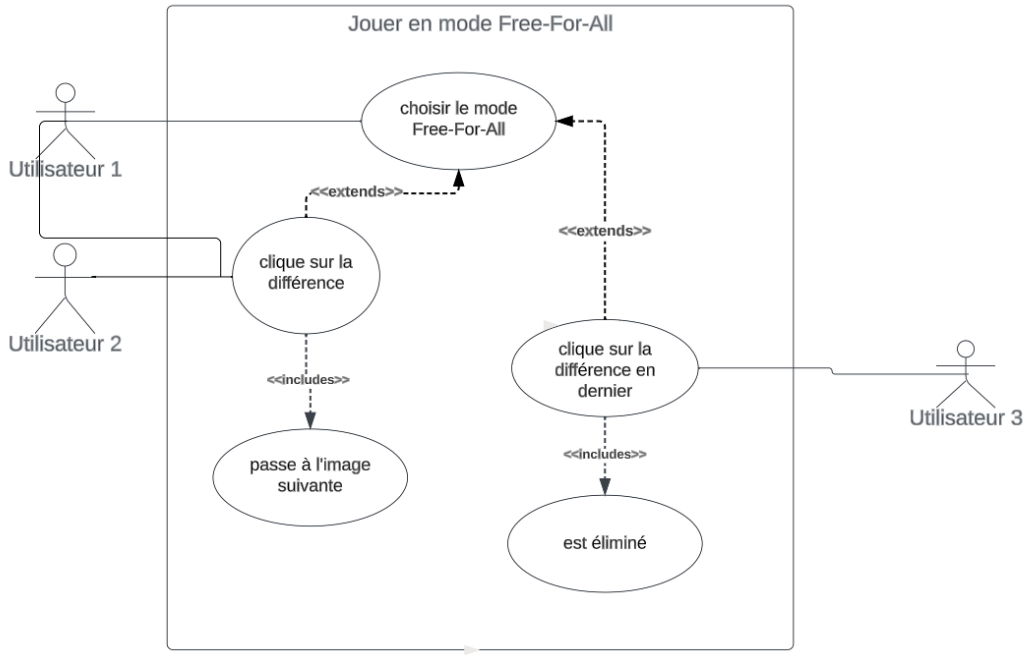

Figure 4 : Diagramme du cas d'utilisation 'Jouer une partie en mode Free-For-All'

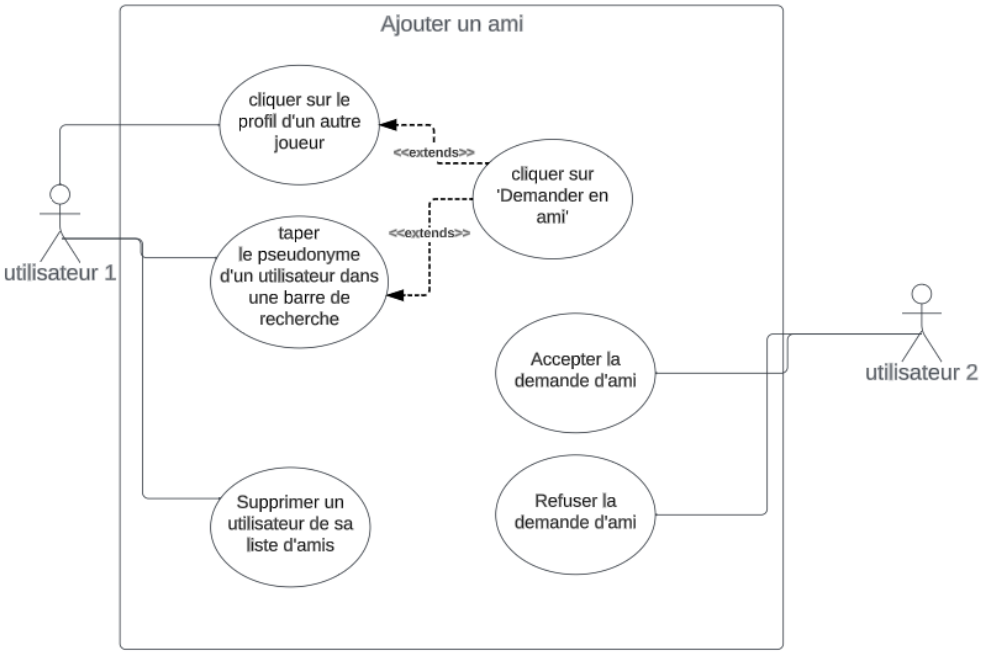

Figure 5 : Diagramme du cas d'utilisation 'Ajouter un ami'

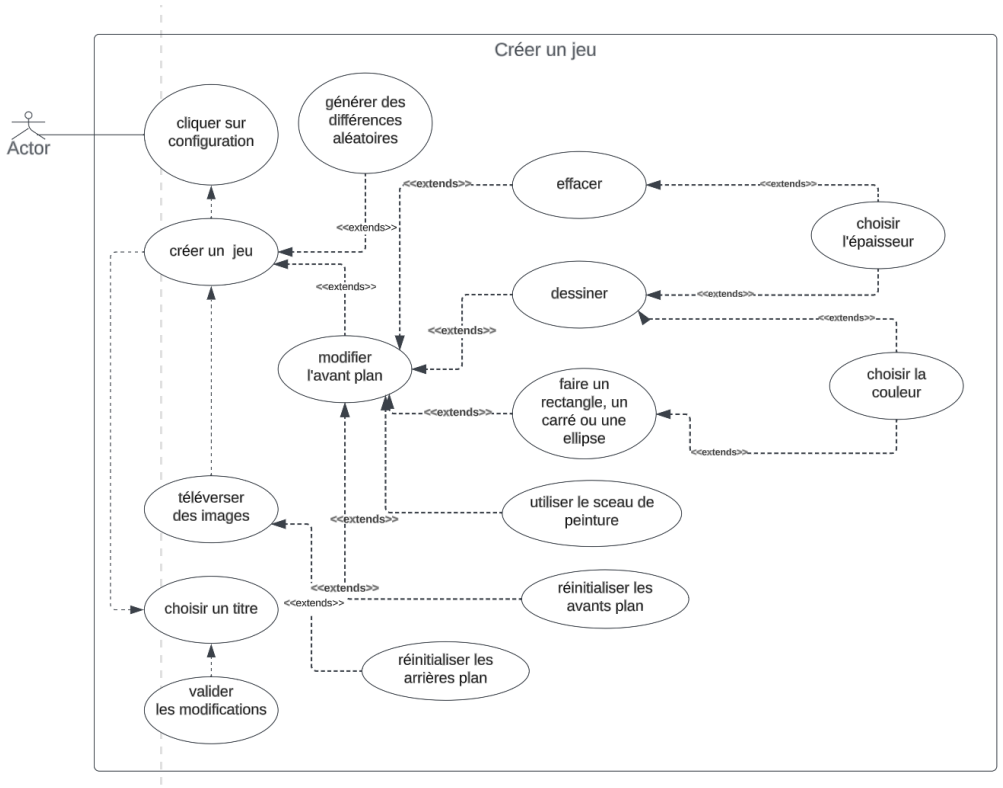

Figure 6 : Diagramme du cas d'utilisation 'Créer un jeu'

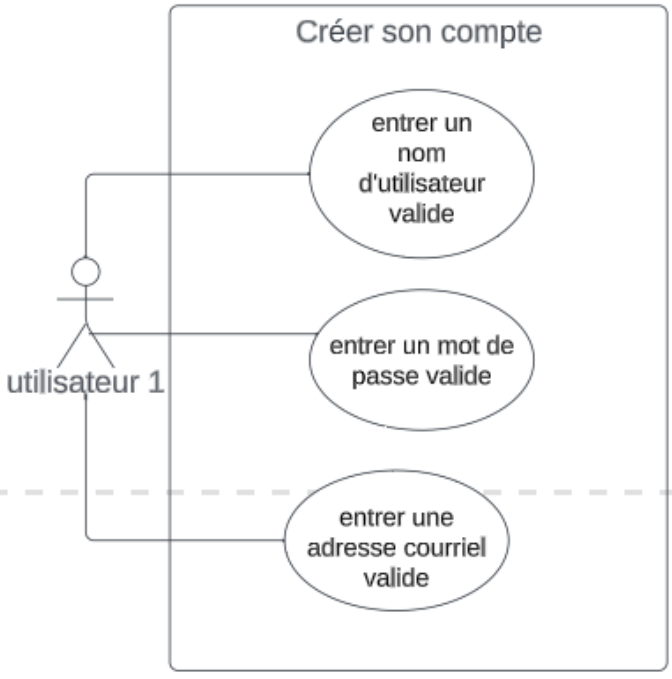

Figure 7 : Diagramme du cas d'utilisation 'Créer son compte'

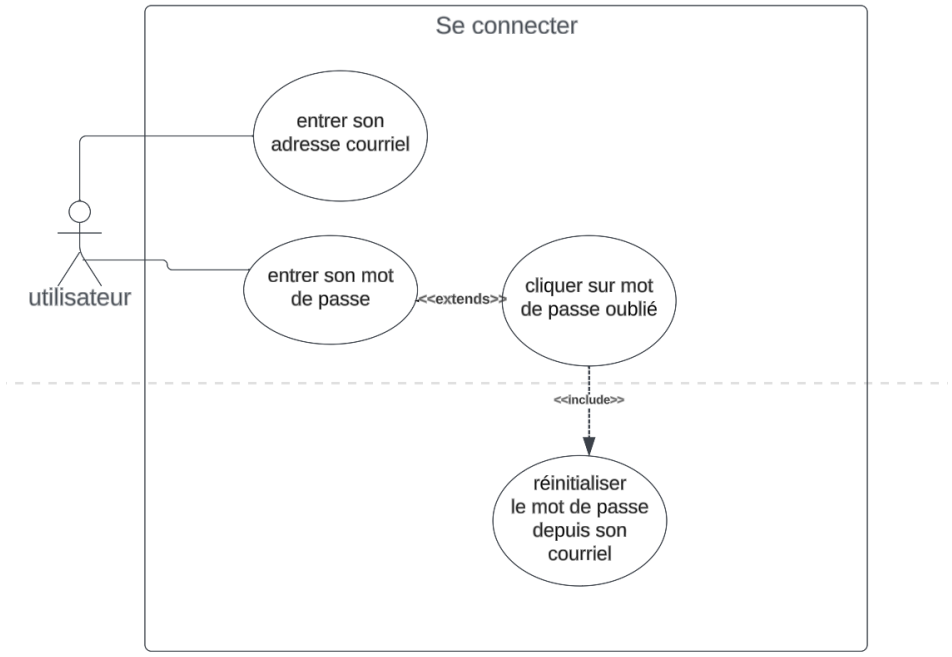

Figure 8: Diagramme du cas d'utilisation 'Se connecter'

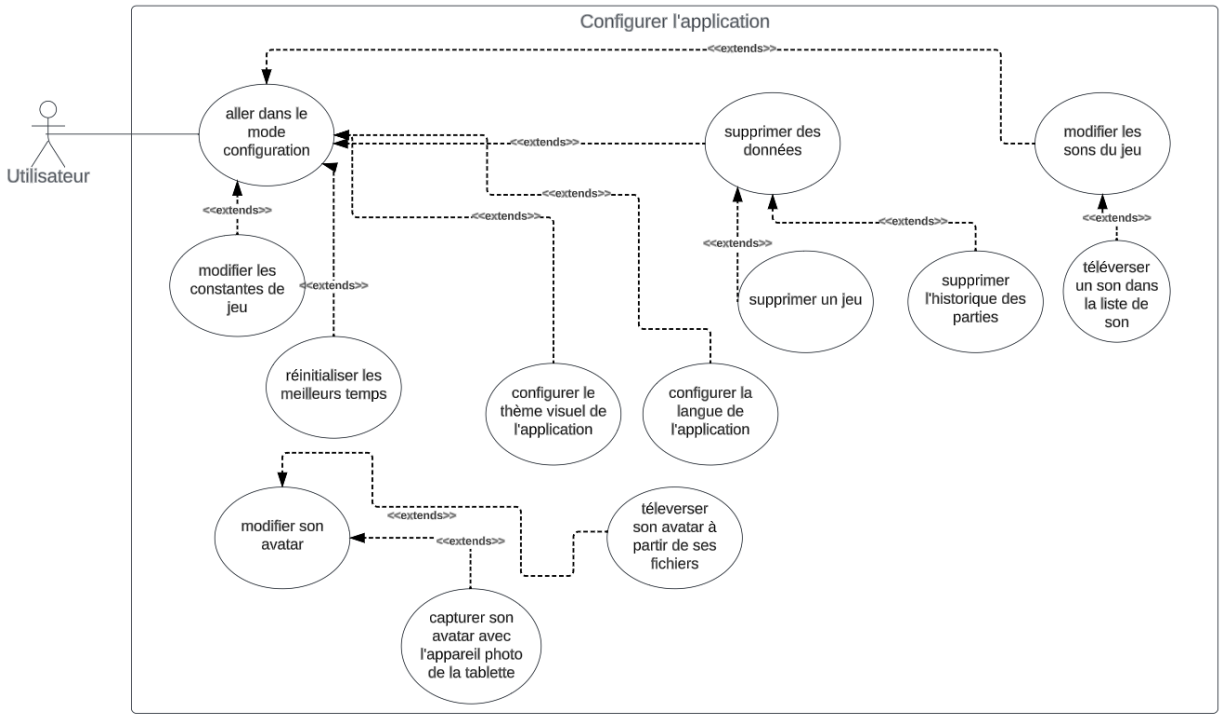

Figure 9 : Diagramme du cas d'utilisation 'Configurer l'application'

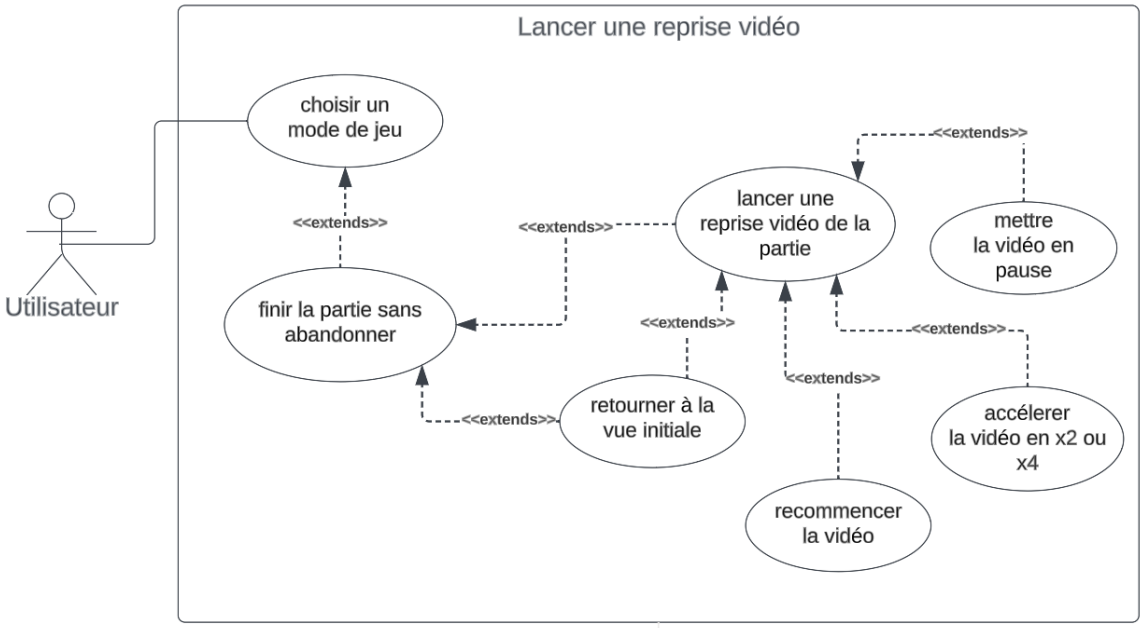

Figure 10 : Diagramme du cas d'utilisation 'Lancer une reprise vidéo'

4. Vue logique

Les tableaux suivants comprennent une description des paquetages significatifs à l'architecture de notre projet. Ils sont suivis de diagrammes de paquetages et de classes.

ClientLourd
Ce paquetage englobe les composantes de la vue et les services permettant de mettre à jour cette vue et d'interagir avec le serveur par des requêtes http et des sockets.

GUI Components
Ce paquetage contient les éléments de la vue et permet à l'utilisateur d'interagir avec le système. Il s'assure de formater et de mettre à jour dans l'affichage les informations qui lui sont fournies par le serveur.

Drawing
Ce paquetage englobe les services permettant à l'utilisateur de dessiner (rectangle, carré, ellipse, pipette…) et d'appliquer des modifications aux canevas dans la page de création.

Creating
Ce paquetage englobe les services nécessaires à la page de création permettant la gestion, l'affichage, l'extraction des images et l'ajout d'une fiche de jeu au site web. Ces services permettent donc plus précisément la vérification de l'image de fond importée par le joueur et l'envoi des images au serveur.

Playing (Client)
Ce paquetage comprend les services permettant le bon déroulement d'une partie, c'est-à-dire le clignotement des différences, les notifications sonores, les messages d'erreur et les reprises vidéo.

Communication
Ce paquetage rend possible la communication du client vers le serveur via http ou socket.

Server
Ce paquetage englobe les controllers et services permettant de traiter les requêtes du client via http ou socket, de gérer le système de fichiers, de communiquer avec la base de données et d'appliquer de la logique côté serveur.

Controllers
Ce paquetage permet de recevoir les requêtes http du client, de communiquer l'information au bon service à des fins de traitement et de renvoyer une réponse au client. Cela inclut les contrôleurs responsables des jeux, des images et des paramètres.

Differences
Ce paquetage permet la détection des différences entre 2 images fournies par le client.

Playing (Serveur)
Ce paquetage contient les services et managers permettant le bon fonctionnement de la création et du déroulement d'une partie de jeu. Cela comprend le traitement des requêtes du client via socket, la gestion du chronomètre de partie et la validation des différences trouvées en cours de partie.

Storage
Ce paquetage gère la conservation de toutes les données de l'application. Il communique avec la base de données MongoDB pour gérer les informations des fiches de jeu, les profils utilisateurs, les amis de chaque profil, les historiques de parties, les paramètres globaux, les meilleurs scores et gère la conservation des images dans un système de fichiers.

Clavardage
Ce paquetage permet de gérer tout ce qui est relié à la messagerie de l'application des des canaux de messagerie. La communication est faite principalement par sockets à un serveur dédié de messagerie mais des requêtes http sont aussi faites pour récupérer l'historique d'un certain canal.

Chat Vocal
Ce paquetage permet de gérer tout ce qui est relié au chat vocal de l'application. Un serveur est dédié au chat vocal. La communication sera faite avec un protocole en temps réel.

Authentification
Ce paquetage permet de gérer tout ce qui est relié à l'authentification. Un serveur est dédié à la gestion de l'authentification. Ce serveur communique avec notre serveur principal qui contient la logique ainsi que les clients directement. Les communications sont faites par sockets et par http. Le serveur d'authentification contient aussi les ids uniques de chaque utilisateur.

ClientLéger
Ce paquetage englobe les composantes de la vue et les services permettant de mettre à jour cette vue et d'interagir avec le serveur par des requêtes http et des sockets pour le mobile

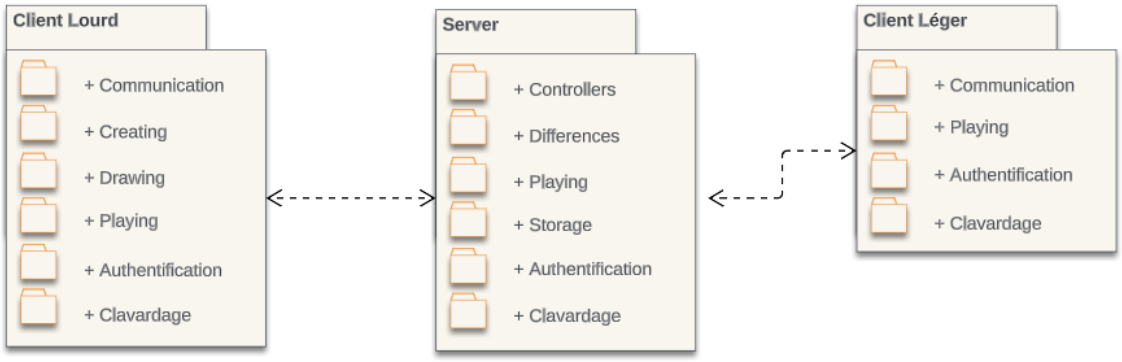

Figure 11 : Diagramme de paquetages du projet complet

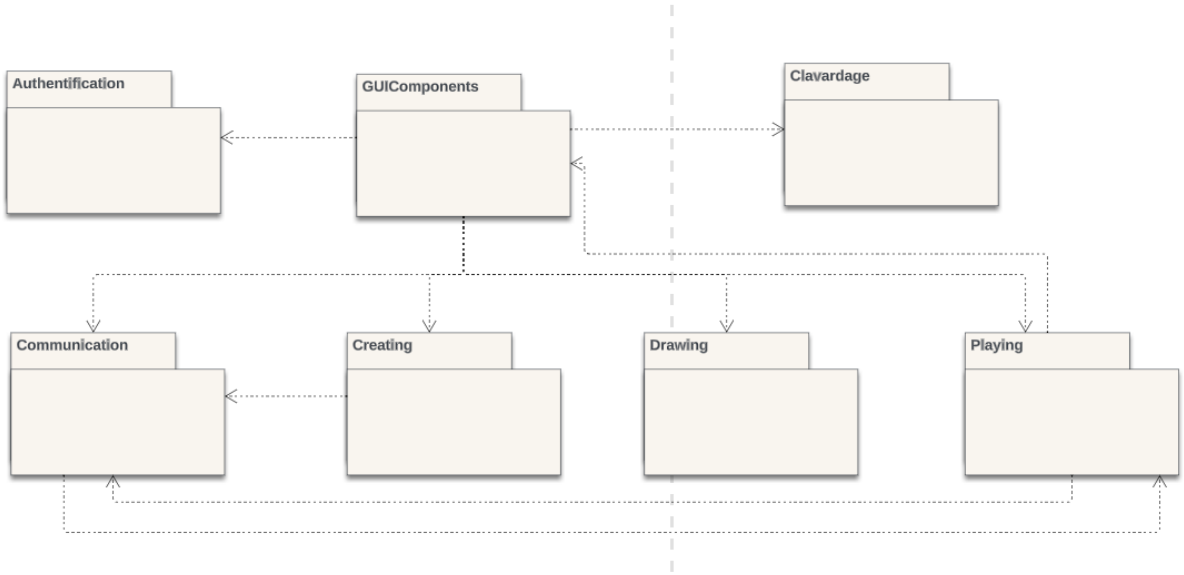

Figure 12 : Diagramme de paquetages du Client Lourd

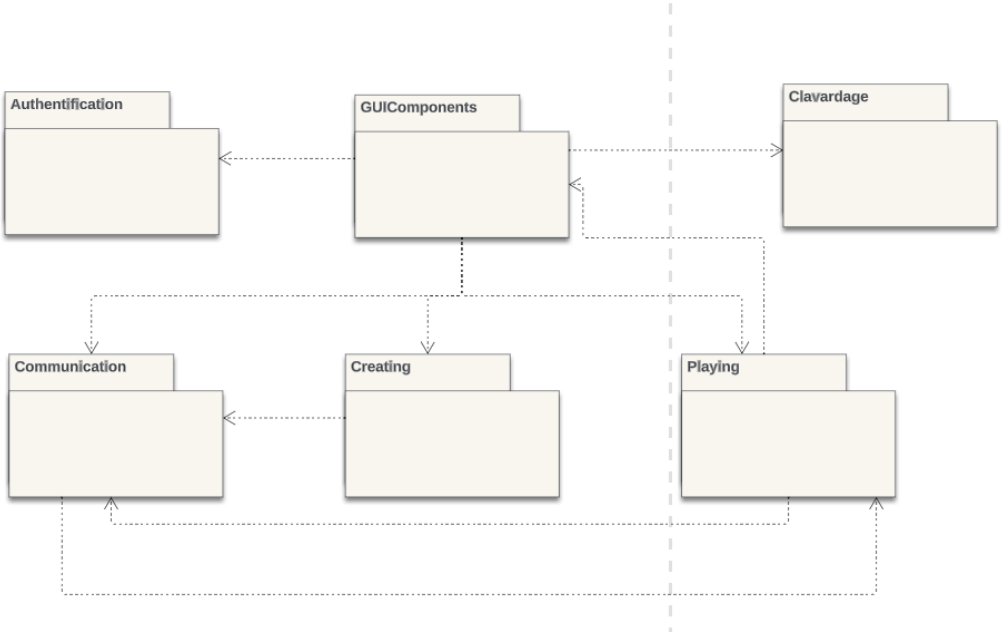

Figure 13 : Diagramme de paquetages du Client Léger

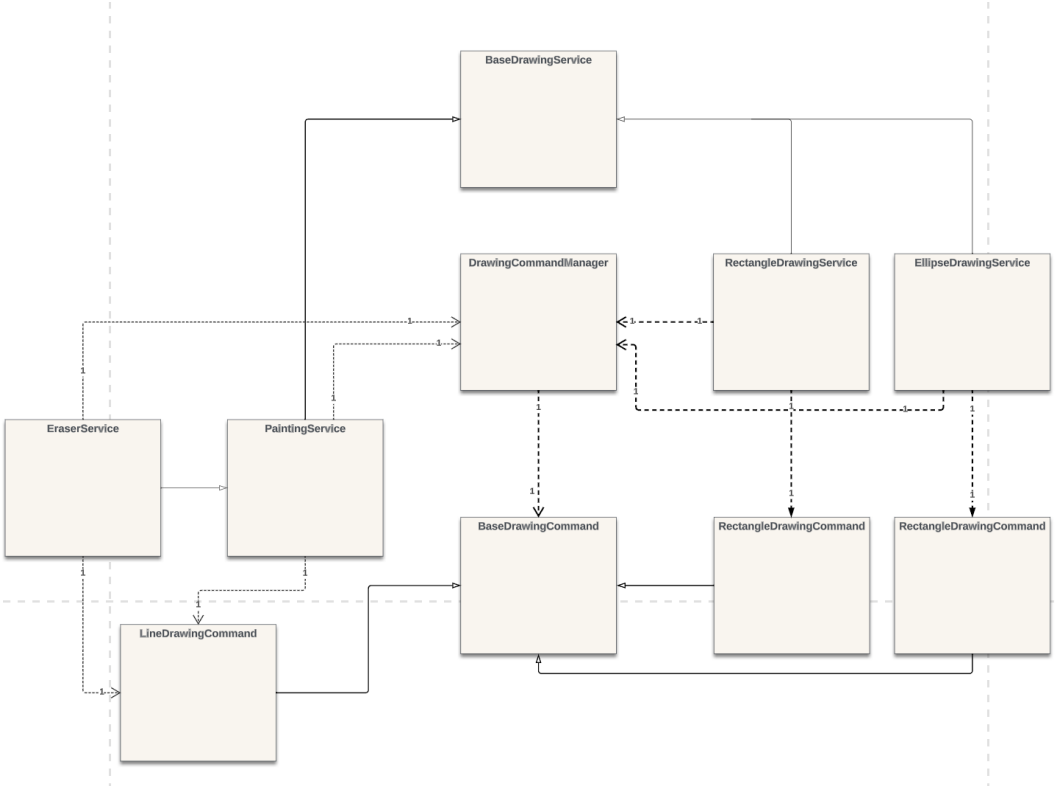

Figure 14 : Diagramme de classes pour le paquetage Drawing

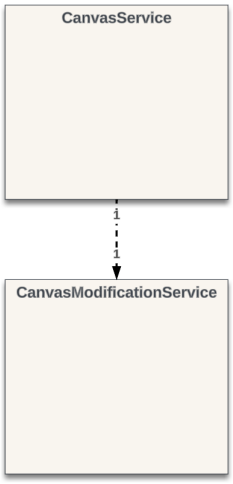

Figure 15 : Diagramme de classes pour le paquetage Creating

Figure 16 : Diagramme de classes pour le paquetage Playing côté client

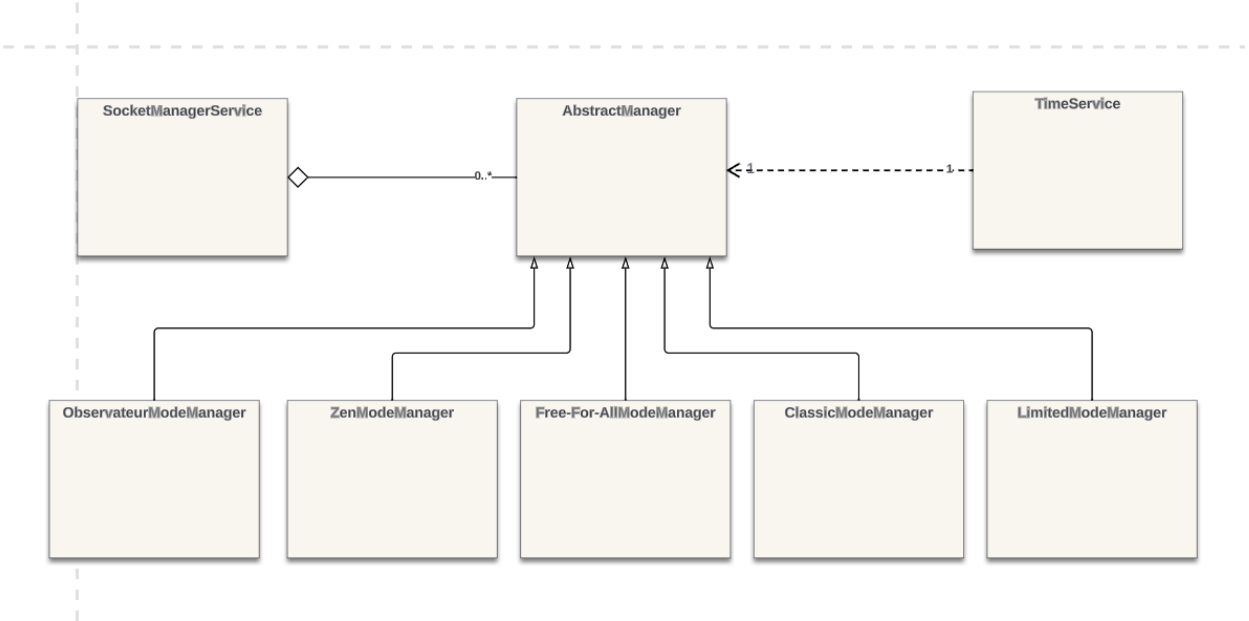

Figure 17 : Diagramme de classes pour le paquetage Playing côté server

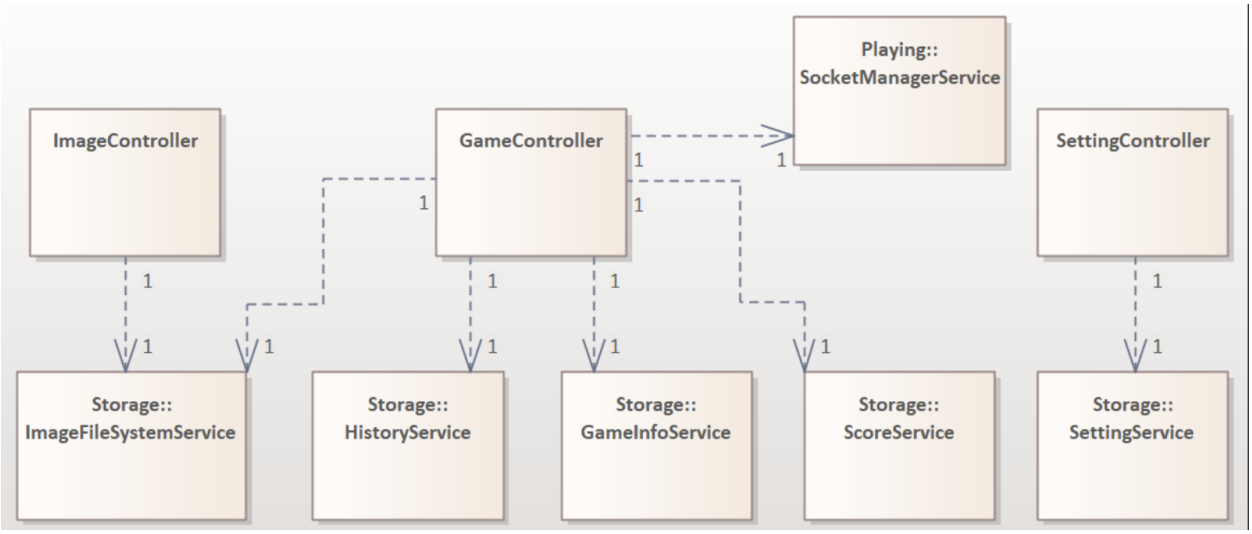

Figure 18 : Diagramme de classes pour le paquetage Controllers

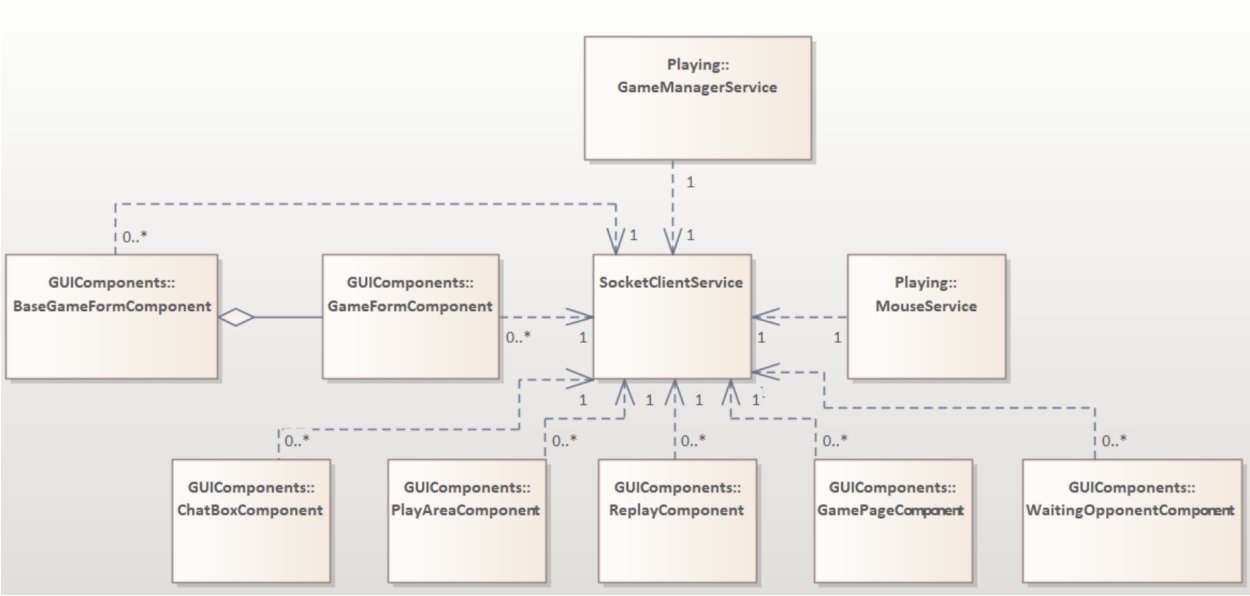

Figure 19 : Diagramme des classes utilisant la classe SocketClientService du paquetage Communication

5. Vue des processus

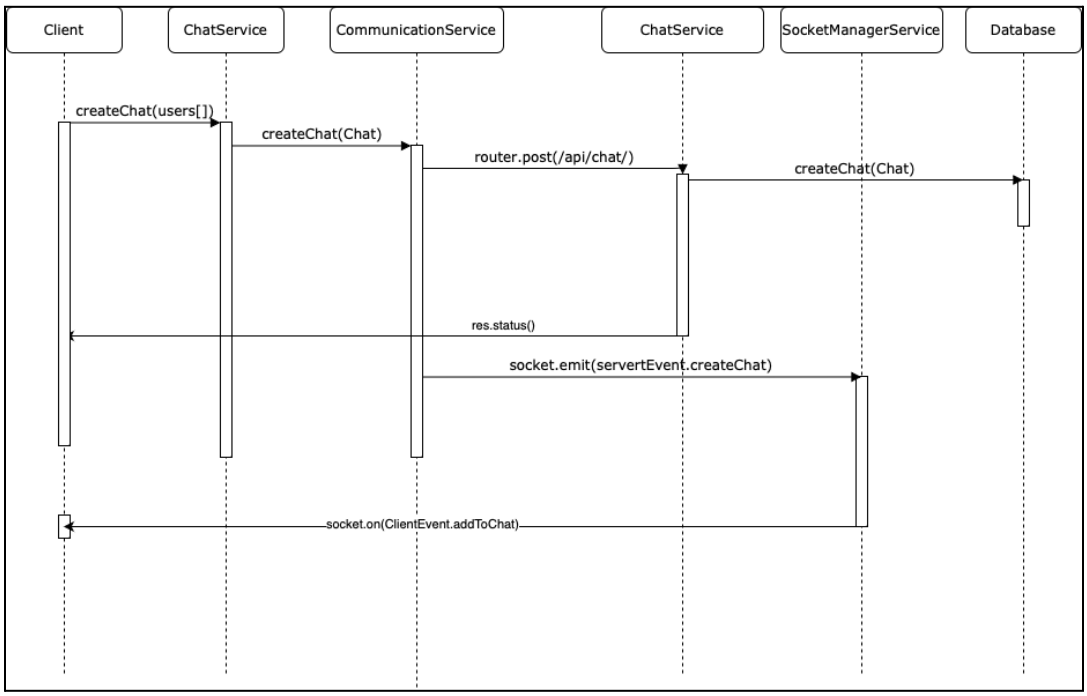

Figure 20. Diagramme de séquence pour le processus de création d'un canal de clavardage

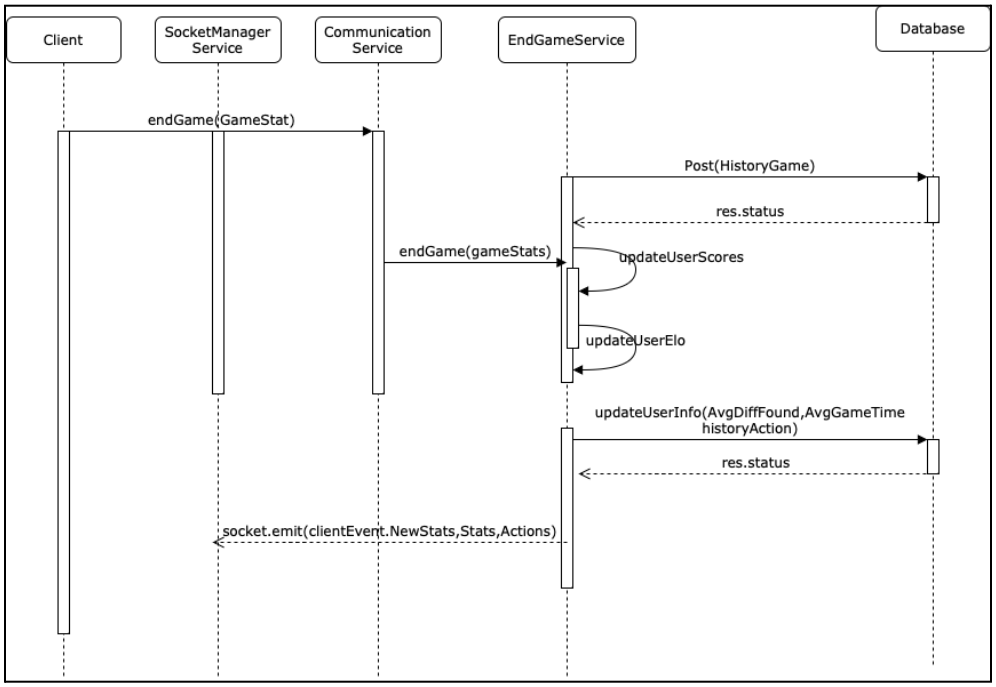

Figure 21. Diagramme de séquence pour le processus de la mise à jour des statistiques et de l'historique des actions de l'utilisateur

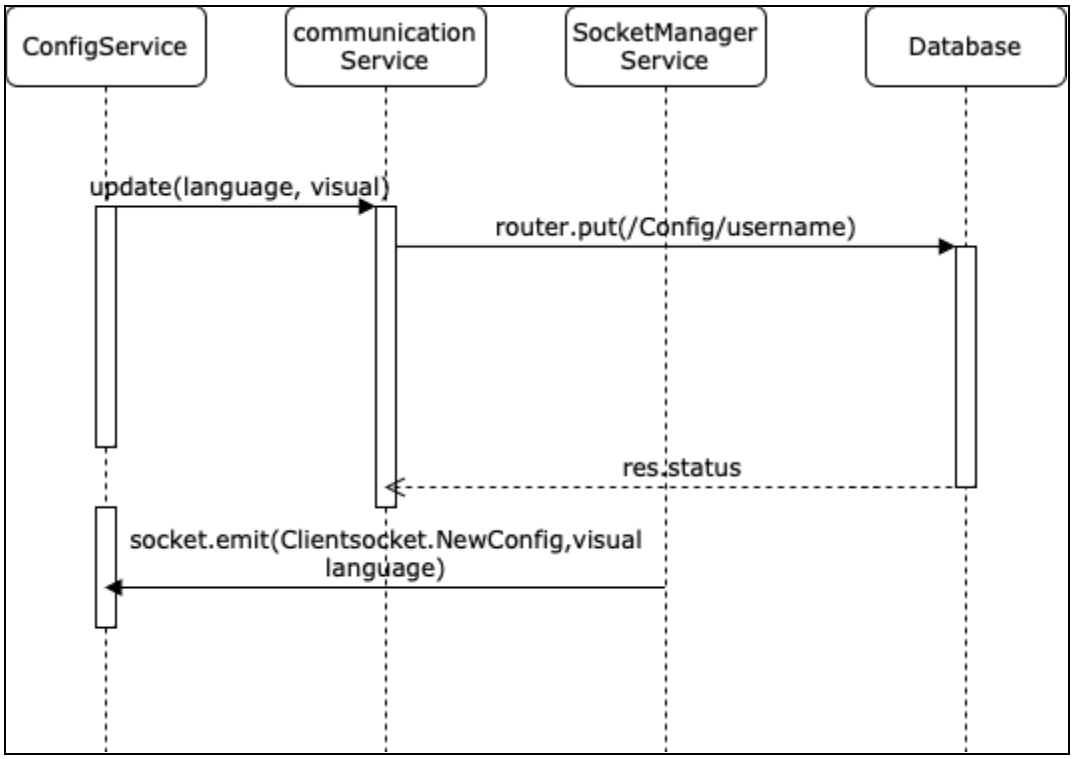

Figure 22. Diagramme de séquence pour le processus de configuration du thème visuel et de la langue et pour l'avatar

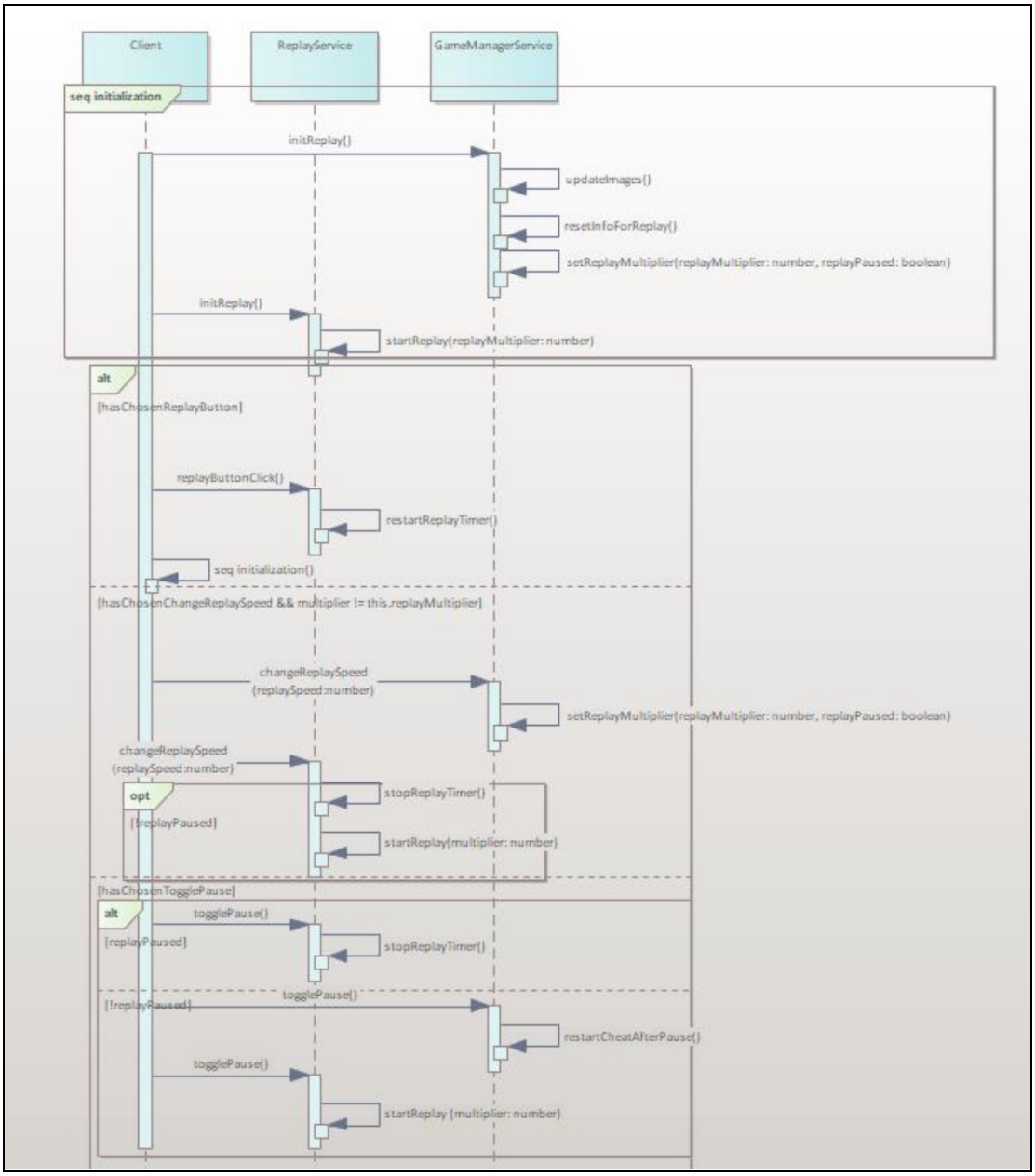

Figure 23. Diagramme de séquence pour le processus de reprise vidéo

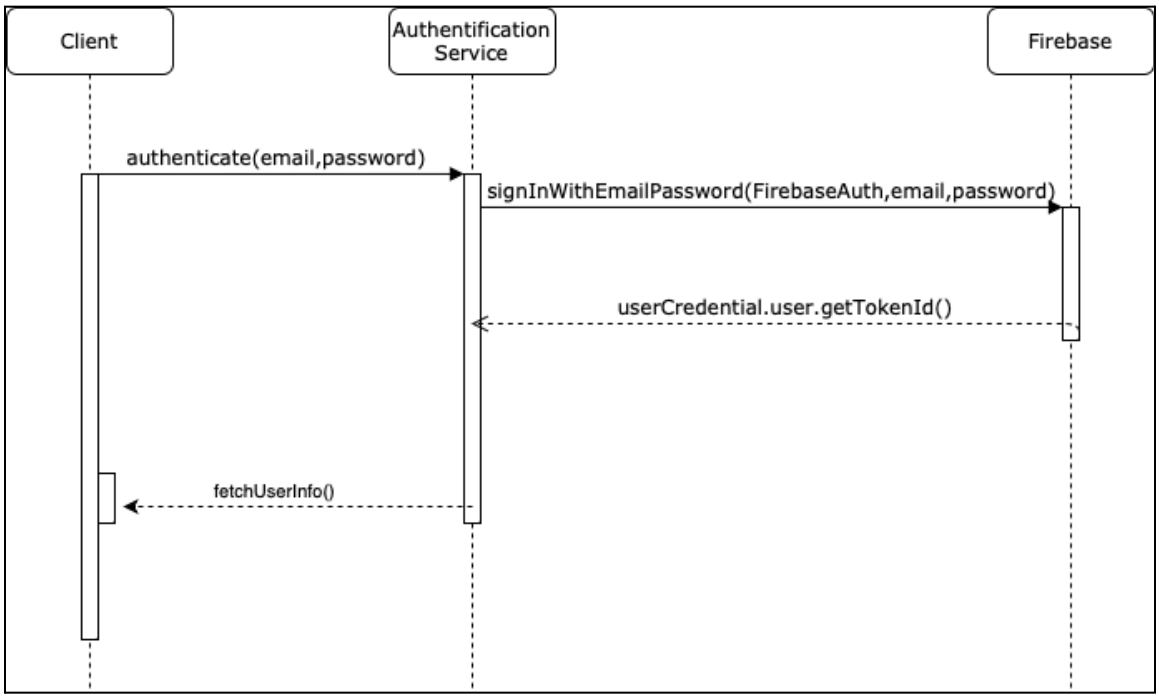

Figure 24. Diagramme de séquence pour le processus d'authentification

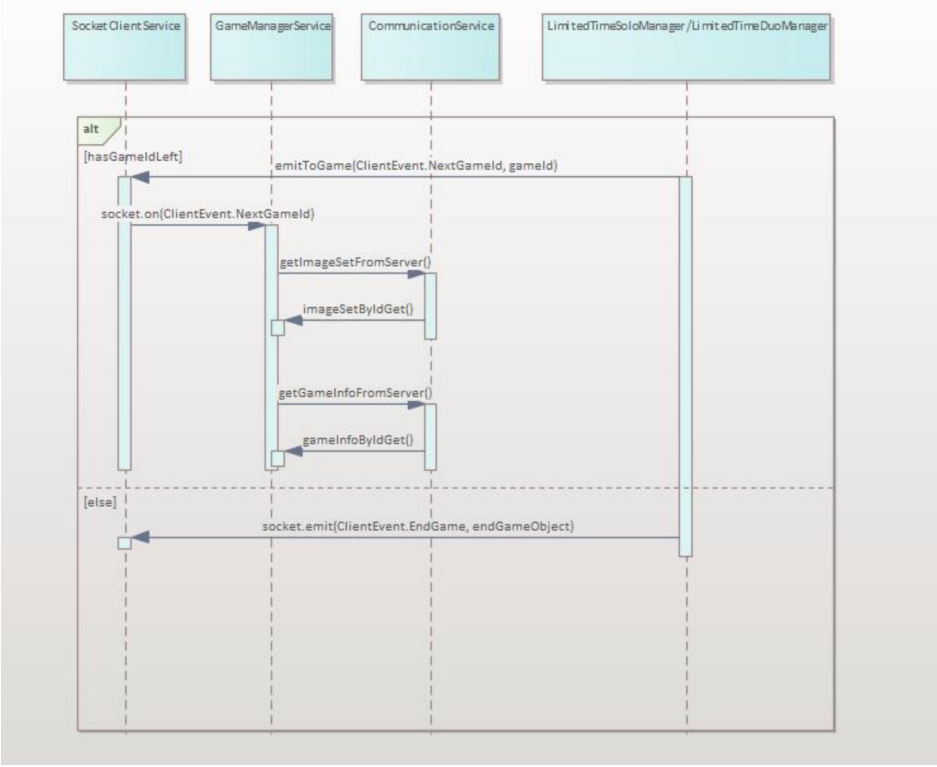

Figure 25. Diagramme de séquence pour le processus de l'obtention d'une image en mode temps limité, mode zen et mode free for all

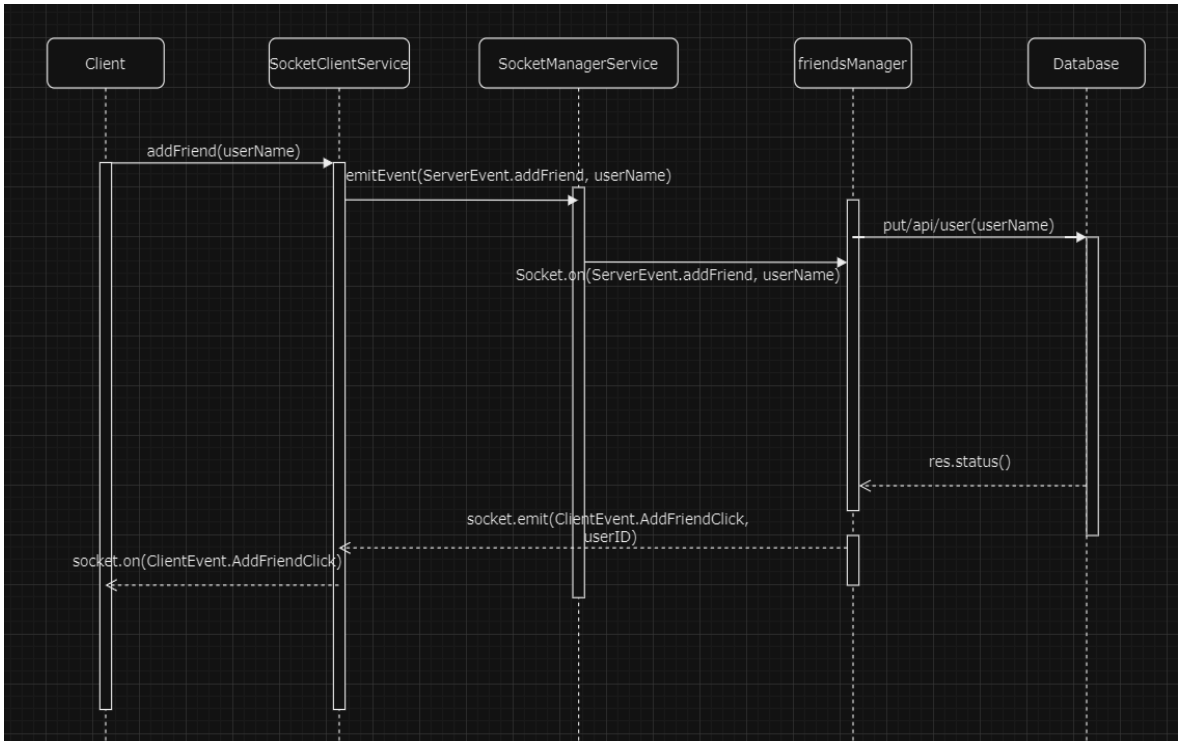

Figure 26. Diagramme de séquence pour le processus de demande d'amis

6. Vue de déploiement

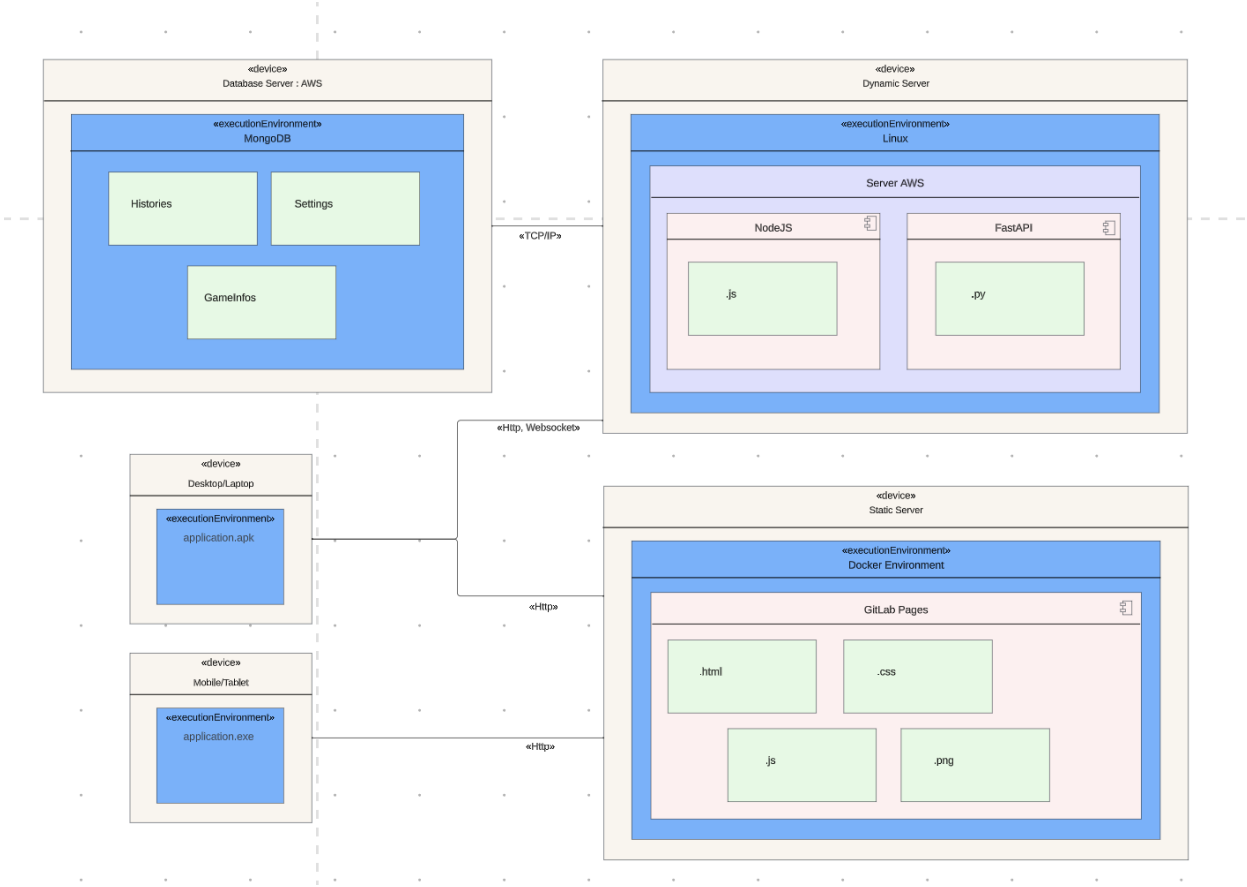

Figure 27. Diagramme de déploiement

7. Taille et performance

Taille du code source: Une taille trop grande de notre code source peut rendre notre logiciel très difficile à maintenir pour le futur. Plus d'effort sera demandé pour contribuer à son évolution. Afin de garder une garder notre logiciel maniable même si la taille de celui-ci sera importante, nous nous engageons à garder une limite de 200 lignes de code par fichier. Cela favorise l'utilisation de divers services, rendant ainsi notre code plus maintenable et lisible.

Taille de la mémoire: Étant donné que nous créons deux applications (client lourd et léger) qui doivent être téléchargées dans des tablettes et ordinateurs à taille de mémoire limitée, nous nous engageons à limiter la taille de chaque application à un maximum de 1 Go.

Vitesse d'exécution: Puisque notre application est un jeu en ligne avec un système de chat intégré, la latence doit être minimale. Elle doit réagir en temps réel aux événements afin de garantir un jeu équitable envers tous les utilisateurs.

Utilisation des ressources: Nous nous engageons à écrire des applications qui minimisent la charge sur le CPU en utilisant des algorithmes peu coûteux. Nous nous engageons aussi à éviter les fuites de mémoires et bien utiliser les ressources de la RAM en libérant la mémoire dynamiquement. Une bonne gestion de la batterie est aussi importante, il faut alors s'assurer de désactiver les processus qui ne sont pas en utilisation courante.