Équipe 106

Projet «Fais moi Dessin»
Document d'architecture logicielle

Version 1.0

Historique des révisions

Date	Version	Description	Auteur
2020-09-20	1.0	Première version	Gloria Aubierge Sohou
2020-09-21	1.0	Ajouter les diagrammes de cas d'utilisation	Sophie Dorval
2020-09-22	1.0	Ajouter la section des déploiements	Kevin Nguyen
2020-09-22	1.0	Ajouter la section introduction	Youbo Wang
2020-09-25	1.0	Finalisation de la vue des processus	Gloria Aubierge Sohou
2020-10-01	1.0	Rédaction des objectifs architecturaux	Sophie Dorval
2020-10-01	1.0	Révision et correction des fautes d'orthographes	Tous
2020-10-02	1.0	Révision des contraintes architecturaux	Gloria Aubierge Sohou
2020-10-02	1.0	Révision des contraintes architecturaux	Tous

Table des matières

Document d'architecture logicielle

1. Introduction

1.1. Objet

Ce document présente une description de l'architecture logicielle de haut niveau du système à développer. Les différents choix de conception de l'équipe 106 y sont explicités à travers diverses vues logicielles et des diagrammes architecturaux.

1.2 Portée

Ce document destiné aux ingénieurs participant à l'implémentation du projet «Fais moi un Dessin».

1.3. Définitions acronymes et abréviations

L'annexe A à la fin de ce document définit tous les mots, acronymes et abréviations utilisés et nécessaires à la compréhension de l'architecture logicielle.

1.4 Références

Ce document se base en partie sur le document de vision et le complément pédagogique fournis par l'autorité contractante puis sur le document de Spécification des Requis du Système (SRS).

1.5. Vue d'ensemble

Tout d'abord, ce présent document débute par une description des objectifs et contraintes architecturaux du projet. Ensuite, chacune des vue logicielles est justifiée et illustrée avec l'aide de diagrammes. Celles-ci sont, dans l'ordre, la vue des cas d'utilisation, la vue logique, la vue des processus et la vue de déploiement. Enfin, les facteurs de taille et performance du logiciel risquant influencer la conception sont présentées.

2. Objectifs et contraintes architecturaux

2.1 Objectifs

2.1.1. Sécurité et confidentialité

Le serveur doit garantir la sécurité des données communiquées entre les différents clients. Les informations sur la base de données, entre autres le mot de passe, doivent être inaccessibles aux utilisateurs. De plus, un nom d'utilisateur doit être unique.

2.1.2. Robustesse

Le serveur doit pouvoir gérer la communication avec jusqu'à 40 clients, car le nombre maximal de joueurs est de 40 comme spécifié dans les contraintes générales du SRS. La communication entre les clients et le serveur doit être constante et stable. Il ne doit pas avoir d'inconsistance dans les données. De plus, il doit être robuste face aux différents scénarios d'utilisations. Il doit être aussi synchronisé en temps réel lors d'une partie de jeu.

2.1.3. Réutilisabilité et testabilité

Le code du système doit être modulaire et structuré en services pour faciliter la réutilisabilité. Le couplage entre les les différentes modules doivent être évitées. Cela permettra aussi de tester plus facilement les fonctionnalités du système.

2.2. Contraintes

2.2.1. Portabilité

Le logiciel “Fai-moi un dessin” doit pouvoir être utilisé sur les ordinateurs ayant le système d’exploitation Windows. Il doit pouvoir être utilisé sur tablette Android de 10.1 pouces ayant une version d’Android 9.0 Pie et ayant une mémoire de 2 GB de RAM plus 32 GB.

2.2.2. Outils et langages de développement

Le serveur doit utiliser Node.js et comme langage de requêtes, GraphQL. De plus le serveur doit être hébergé sur Heroku. Le serveur doit intégrer les données dans la base de données utilisant PostgreSQL.

Le client lourd doit être développé en Typescript avec le cadre logiciel Electron. Le client lourd doit utiliser Apollo Client pour gérer la communication avec le serveur. Le client léger doit être développé dans le langage Kotlin sur Android Studio. Tout comme le client léger, il doit utiliser Apollo Client pour gérer la communication entre le serveur et le client lourd.

2.2.3. Échéancier

La remise finale du projet doit se faire au plus tard le 02 Décembre 2020.

3. Vue des cas d’utilisation

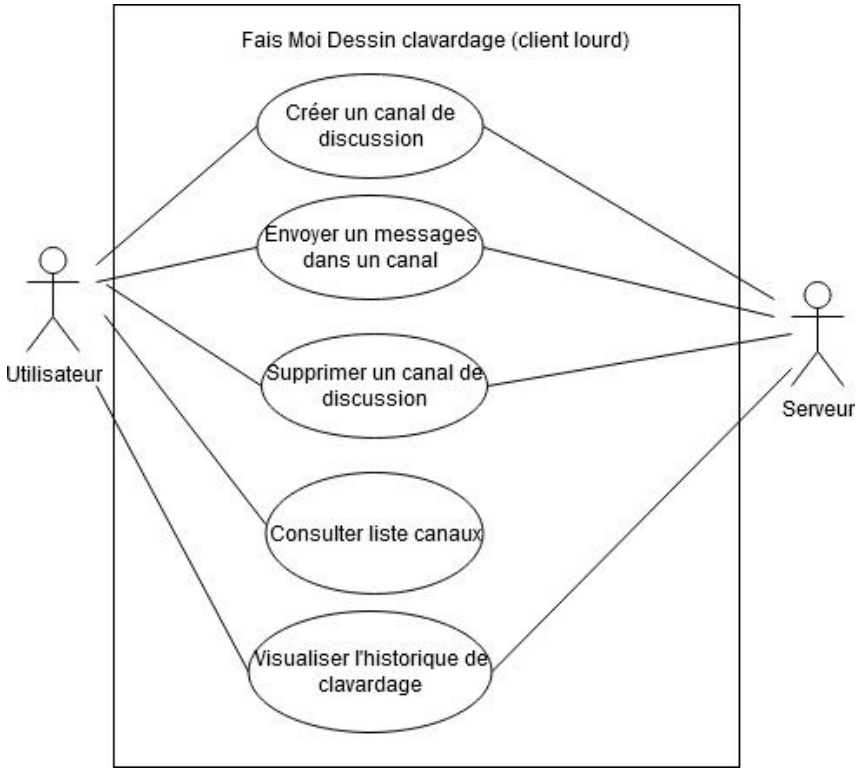

Figure 1. Diagramme de cas d’utilisation pour le clavardage du client lourd

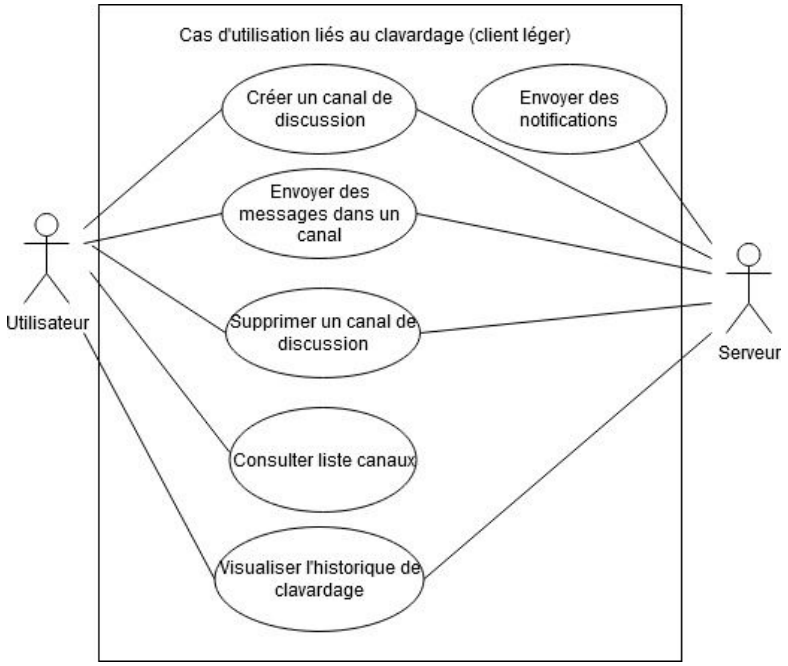

Figure 2. Diagramme de cas d'utilisation pour le clavardage du client léger

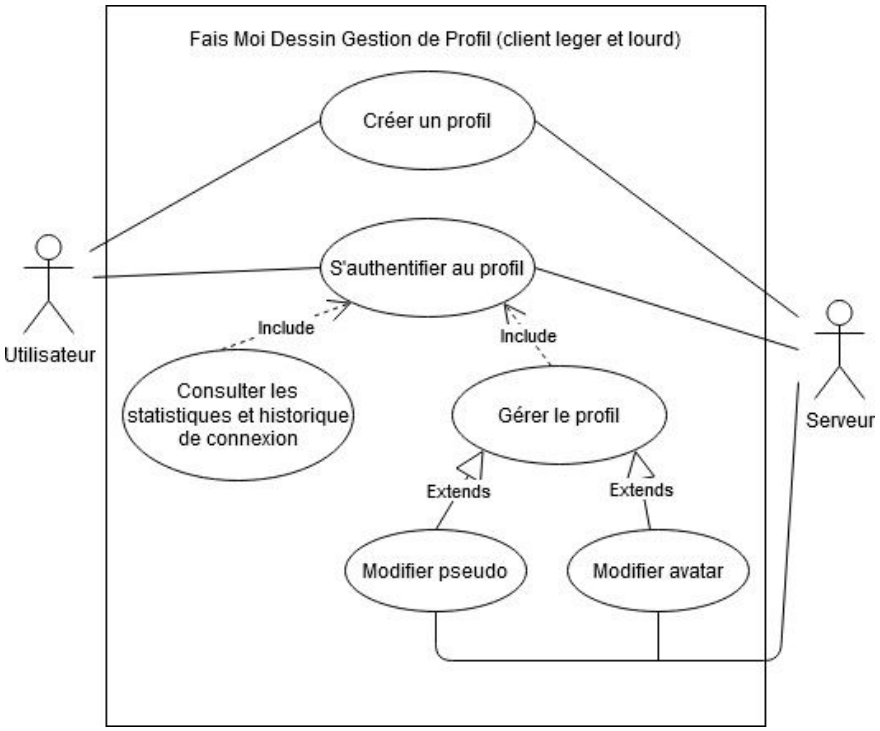

Figure 3: Diagramme de cas d'utilisation pour la gestion de profils

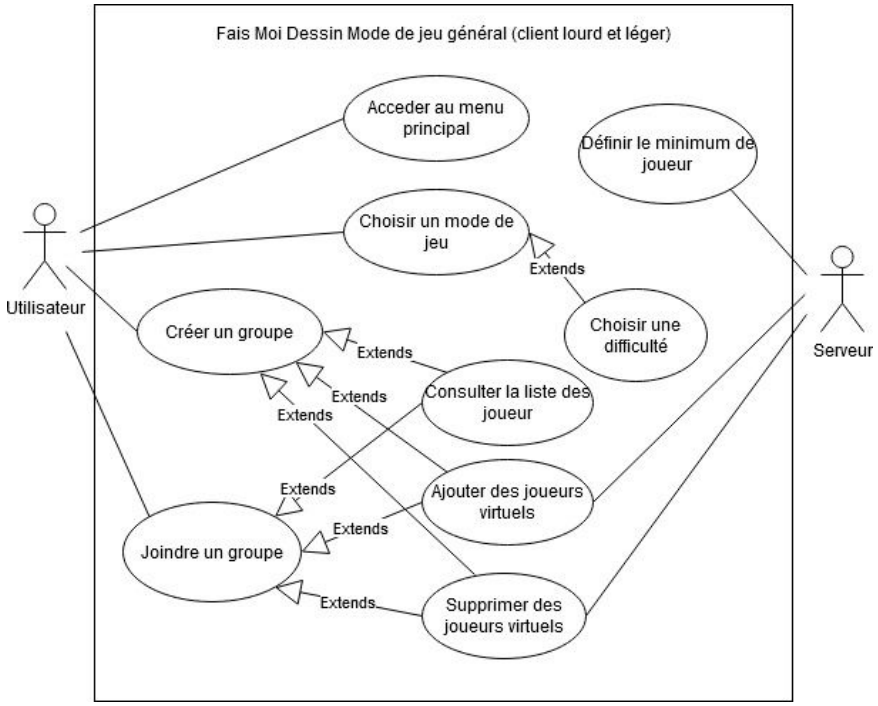

Figure 4:Diagramme de cas d'utilisation pour la création de jeu

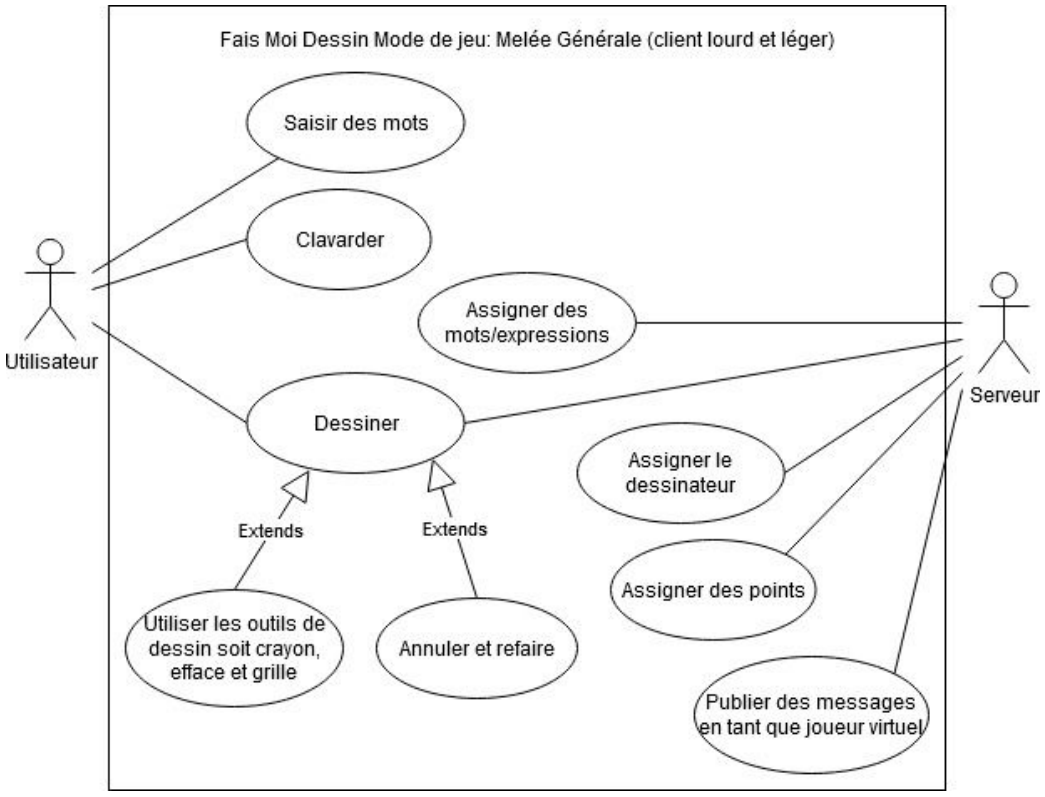

Figure 5: Diagramme de cas d'utilisation pour le mode de jeu mêlée générale

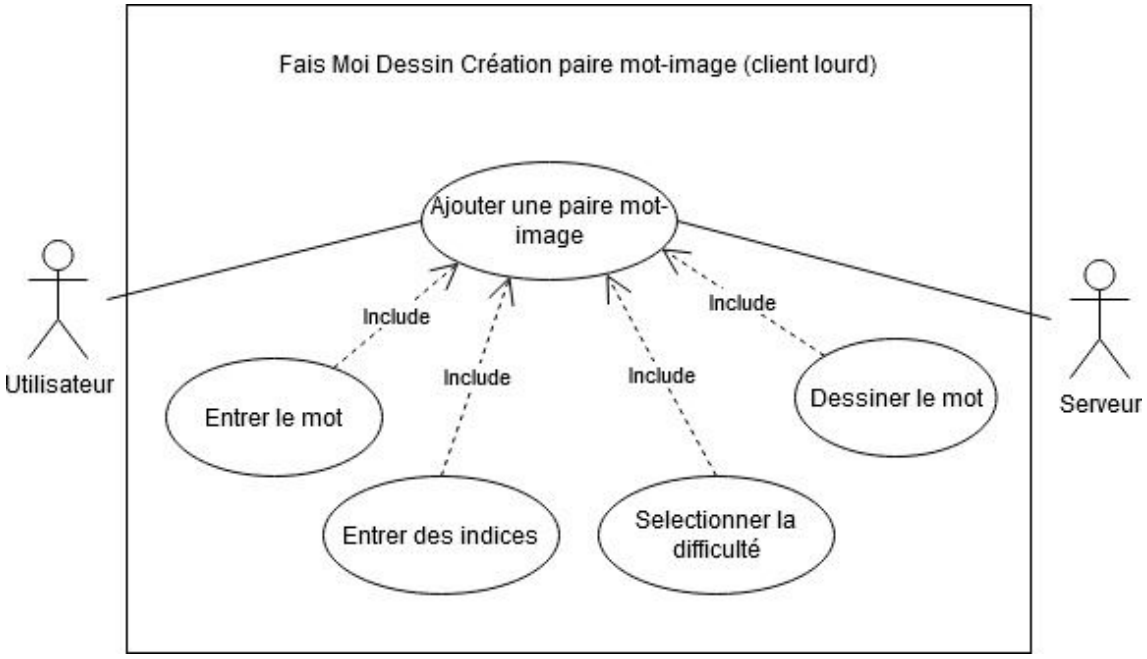

Figure 6: Diagramme de cas d'utilisation pour la création d'une paire mot-image pour client lourd

4. Vue logique

4.1 Diagramme de paquet, vue de haut.

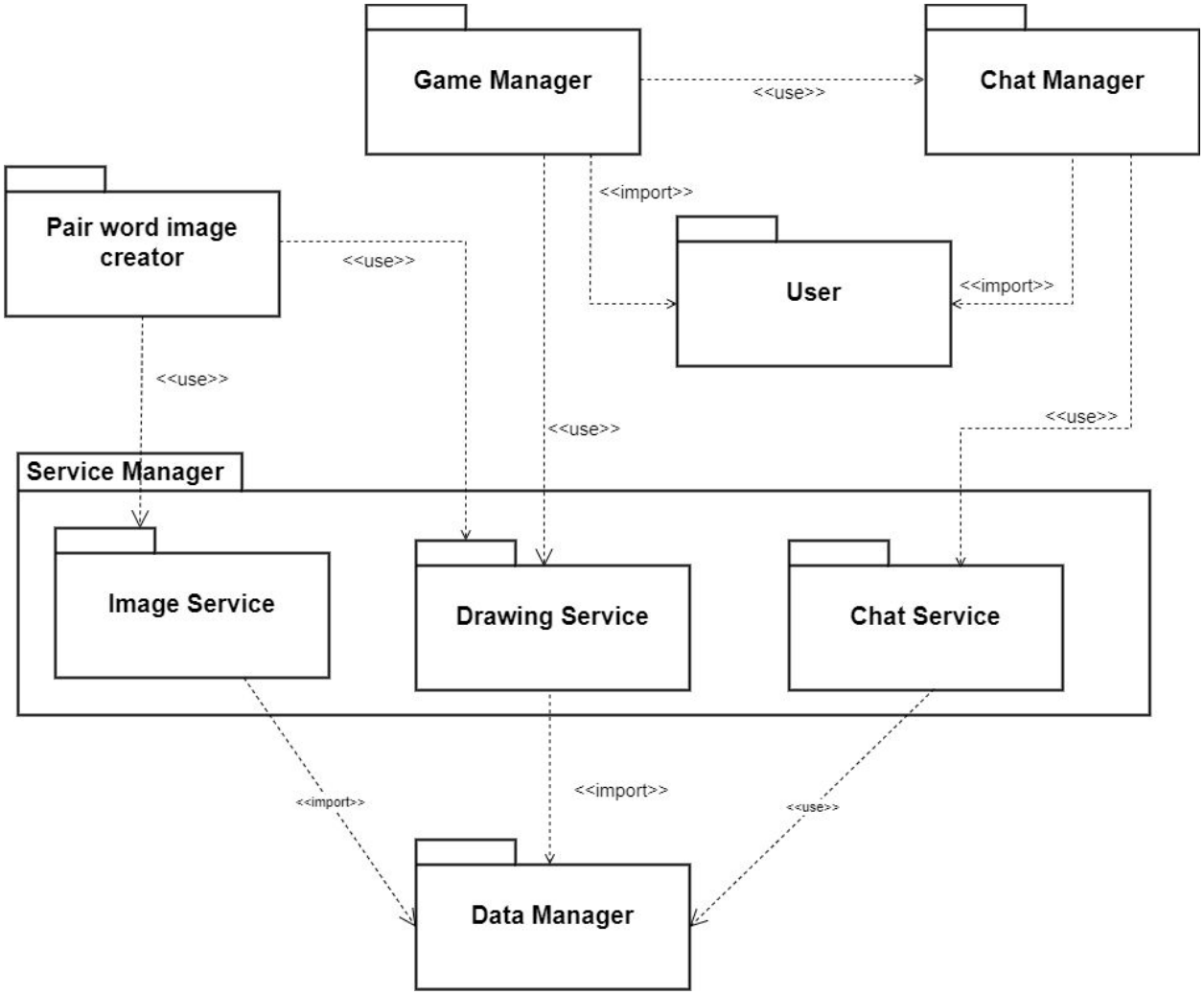

Figure 7: Diagramme de paquet, vue de haut.

4.2 Diagramme de paquet, détail 1.

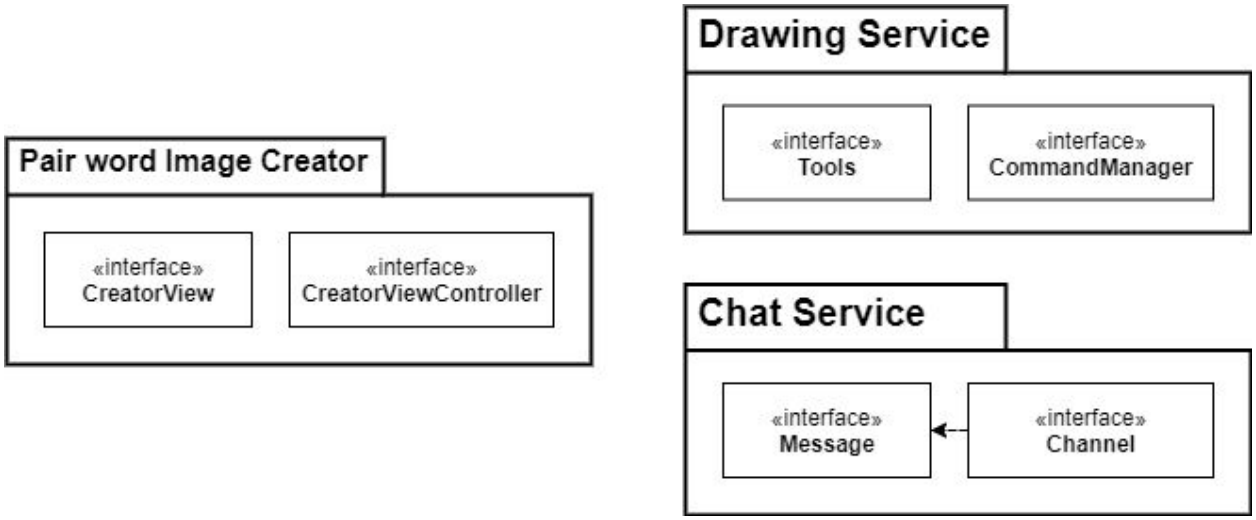

Figure 8: Diagramme de paquet, détail 1

4.3 Diagramme de paquet, détail 2.

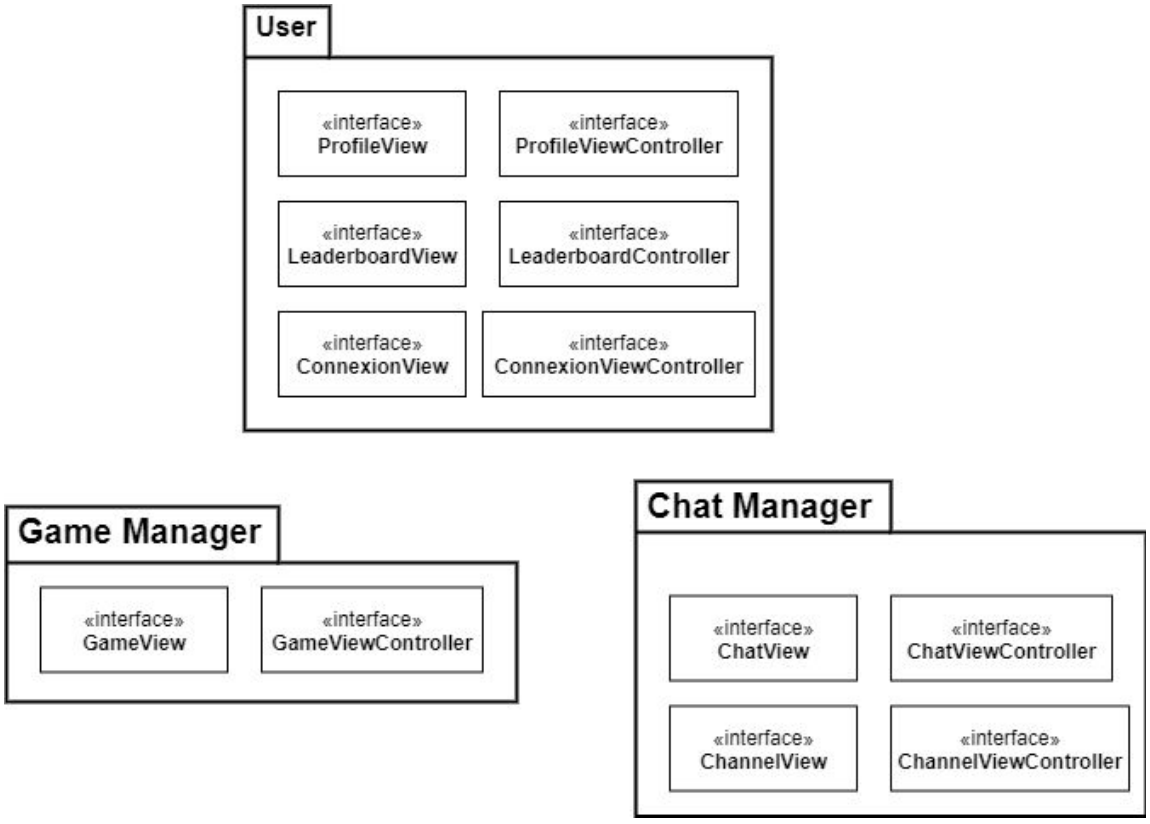

Figure 9: Diagramme de paquet, détail 2.

4.2 Diagramme de paquet, détail 3.

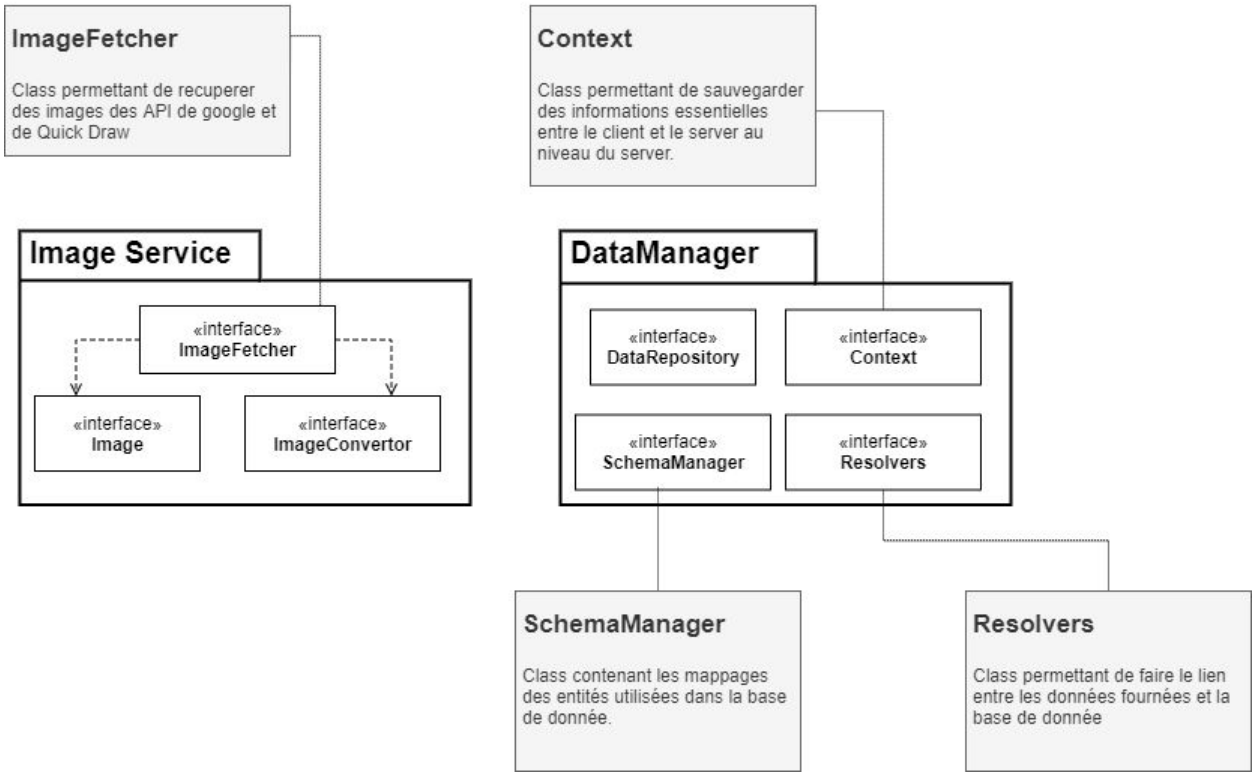

Figure 10: Diagramme de paquet, détail 3.

<Game Manager>
Ce paquetage gère les différentes facettes de chaque mode de jeu

<Chat Manager>
Ce paquetage gère la présentation du chat

<User>
Ce paquetage contient toutes les interfaces liées au user

<Image Service>
Ce paquetage gère la l'importation, et la transformation des images depuis un API ou en local

<DrawingService>
Ce paquetage gère la création d'une image

<ChatService>
Ce paquetage fait la gestion des messages et channels.

<Data Manager>
Ce paquetage s'occupe de la persistance et de la distribution des données.

<Pair word image creator>
Ce paquetage de la présentation d'une création de pair mot image.

5. Vue des processus

Les diagrammes d'interaction suivants présentent les interactions entre les différents processus significatifs. **(Voir section 8 pour des liens de meilleures qualités vers les diagrammes).**

5.1 Authentification

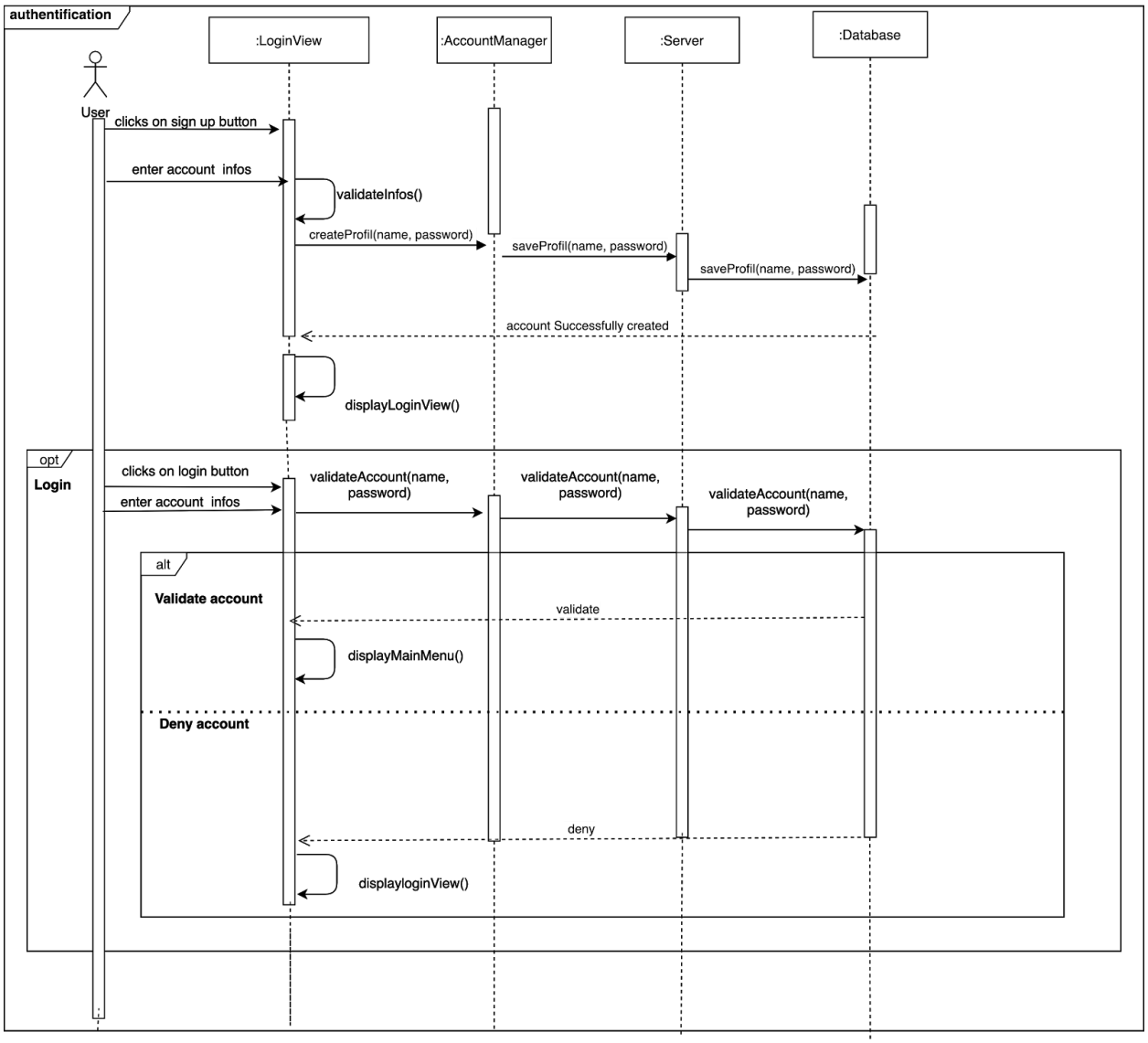

Figure 11: Authentification

5.2 Créer une paire mot-image

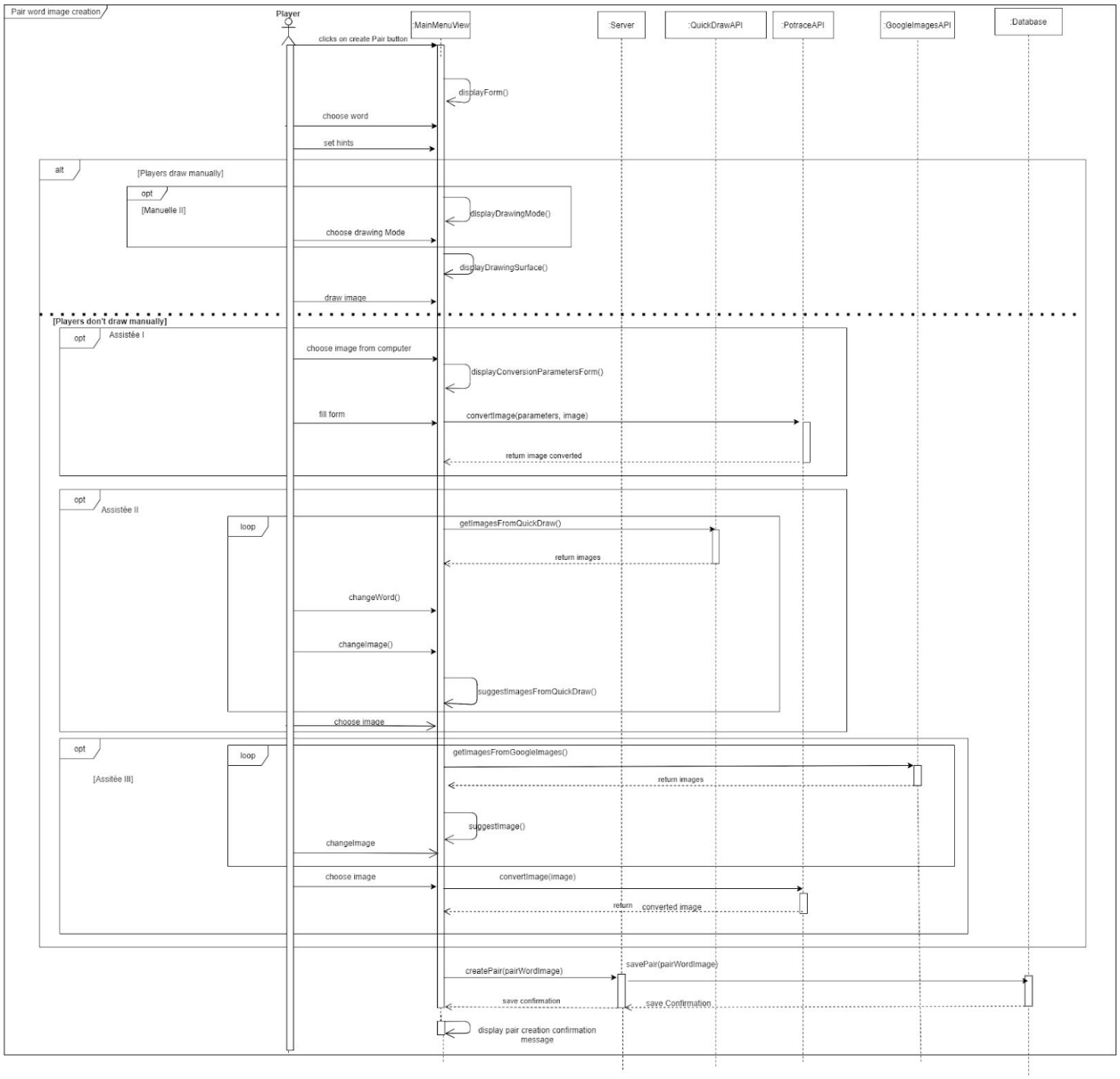

Figure 12: Paire-mot image

5.3 Une partie de jeu

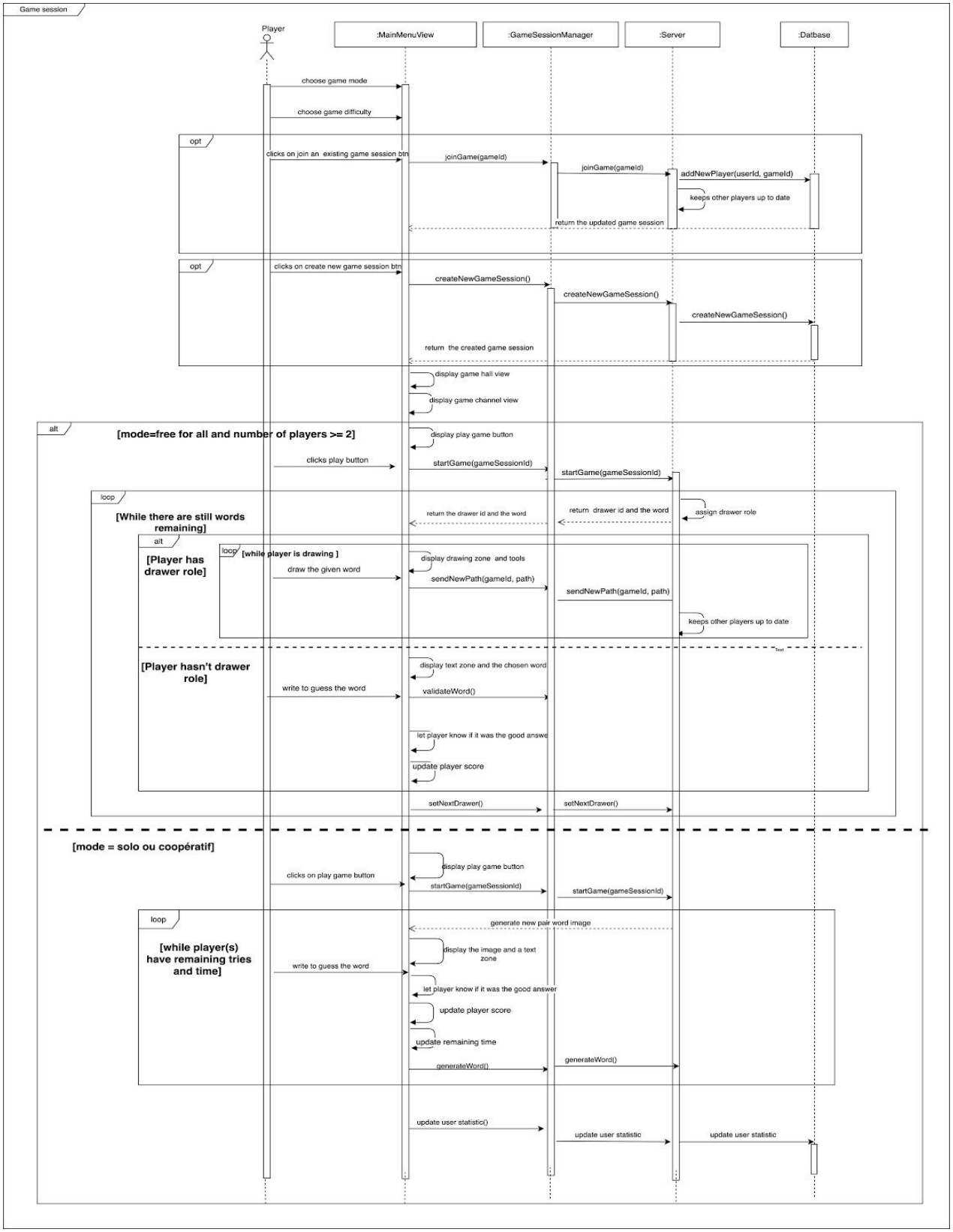

Figure 13: Partie de jeu

6. Vue de déploiement

Le système est divisé en trois composantes. Le client lourd, le client léger et le serveur sont déployés sur des plateformes différentes. Le serveur est automatiquement déployé sur la plateforme cloud de Heroku. Notre pipeline du serveur sur github va directement mettre la version la plus récente sur le serveur pour s'assurer que la version utilisée soit la plus récente. Pour le client lourd qui est une application Electron roulant sur un ordinateur Windows, l'utilisateur peut télécharger l'application via un lien sur le serveur. La dernière version du client léger sera déployée sur App Center et téléchargeable sur les tablettes Android. Les clients communiquent directement avec le serveur en utilisant des requêtes HTTP ainsi que des Websocket utilisant le protocole de communication TCP/IP.

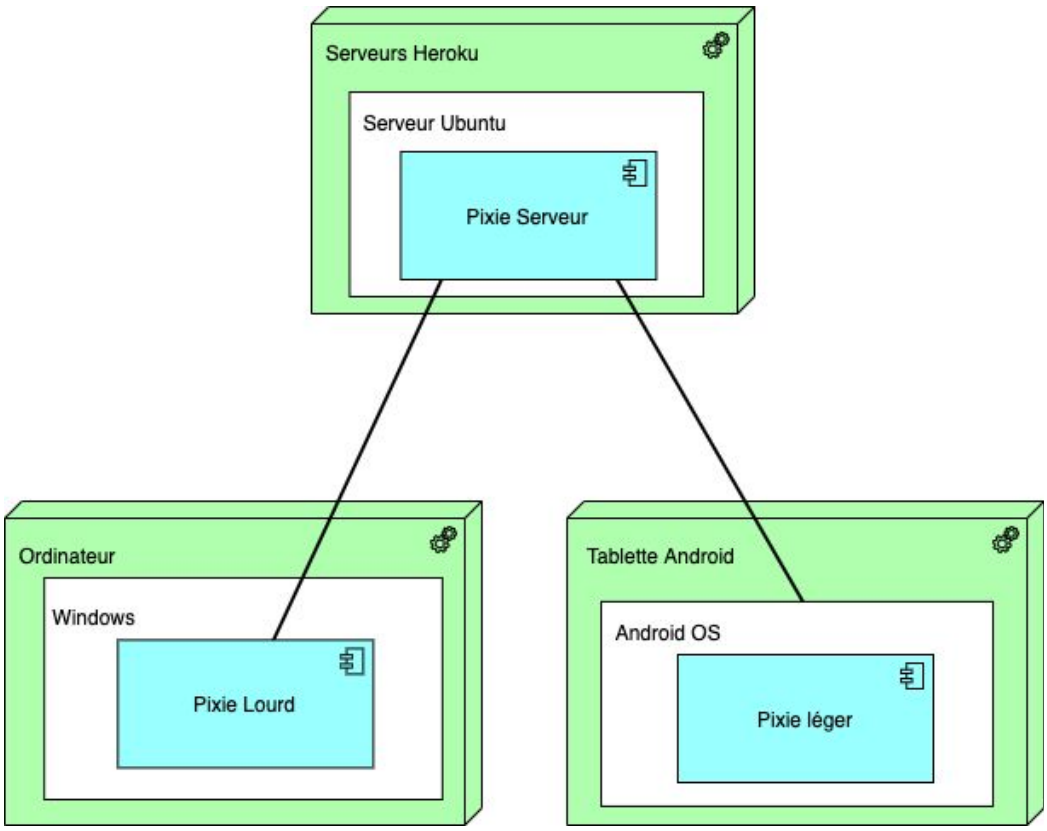

Figure 14:Diagramme de Déploiement

7. Taille et performance

L'architecture et le design du logiciel peuvent être impactés par la mémoire limitée du côté client léger ainsi que le délais dans la transmission de données entre les utilisateurs. Il faut donc développer cette application de sorte que les utilisateurs ne remarquent pas le délai de transmission de données et ce même s'il y a plusieurs utilisateurs connectés et qui communiquent dans la zone de clavardage. Il faut également limiter l'utilisation de la mémoire ainsi que l'espace sur les deux clients, mais surtout sur le client léger. L'architecture du serveur doit faciliter la communication entre les deux clients lors d'une communication entre les différents utilisateurs et l'envoie du dessin après une modification.

Notre architecture permet de répondre aux besoins de taille et performance nommés ci-dessus, car il permet de connecter plusieurs utilisateurs en même temps et facilite la communication entre eux grâce à notre protocole de communication. Les fonctions exigeantes en taille et performance comme garder en mémoire tous les utilisateurs, permette plusieurs jeux en même temps est délégué au serveur au lieu du client. Ceci permet au clients d'être plus léger et performant sur le système local des utilisateurs.

8. Annexes

8.1.Glossaire

Client Lourd : Le logiciel qui propose des fonctionnalités complexes avec un traitement autonome, ciblant un environnement windows dans ce projet.
Client Léger : Le logiciel qui n'a presque pas de logique d'application, ciblant un environnement Android dans ce projet.
Heroku**:** Plateforme infonuagique utilisé par le serveur de ce projet.
GraphQL: Un langage de requêtes et un environnement d'exécution.
Node.js: Une plateforme logicielle libre en JavaScript orientée vers les applications réseau événementielles.

8.2. Liens publics vers les diagrammes de séquences

- Figure 11:
  https://viewer.diagrams.net/?highlight=0000ff&edit=_blank&layers=1&nav=1&title=Login#Uhttps%3A%2F%2Fdrive.google.com%2Fuc%3Fid%3D10_TeslIilMwn-FYSIB-V70i-cBmbfFm-%26export%3Ddownload
- Figure 12:

  https://viewer.diagrams.net/?highlight=0000ff&edit=_blank&layers=1&nav=1&title=Paire%20Mot%20image#Uhttps%3A%2F%2Fdrive.google.com%2Fuc%3Fid%3D1TVaGl2cU78GonYTUw8Uw32AmavSQFxVo%26export%3Ddownload

- Figure 13:

  https://viewer.diagrams.net/?highlight=0000ff&edit=_blank&layers=1&nav=1&title=GameSession#Uhttps%3A%2F%2Fdrive.google.com%2Fuc%3Fid%3D19AcQHkp4N9wI7n9Nep3X3ibxLOz-Ehf-%26export%3Ddownload