205

Fais-moi un dessin
Document d'architecture logicielle

Version 1.7

Historique des révisions

Date	Version	Description	Auteur
2021-02-03	1.0	Ajout des premiers cas d'utilisation	Guilhem Dubois
2021-02-07	1.1	Ajout des sections 1 et 2	Augustin Bouchard
2021-02-10	1.2	Ajout vue base de données	Félix Dumont
2021-02-12	1.3	Ajout vue des processus	Guilhem Dubois
2021-02-13	1.4	Ajout vue de déploiement et quelques mise-à-jour	Guilhem Dubois
2021-02-17	1.5	Ajout des diagrammes de paquetages et de classe	Julien Desalliers
2021-02-19	1.6	Ajout vue des processus	Mark Weber-Sadler
2021-02-19	1.7	Ajout diagrammes de paquetages du serveur	Guilhem Dubois
2021-04-18	1.8	Modifications des diagrammes en fonction de la correction	Julien Desalliers et Félix Dumont
2021-04-19	1.9	Ajout du diagramme de séquence d'une partie	Julien Desalliers

Table des matières

Document d'architecture logicielle

1. Introduction

Le présent document contient les informations nécessaires pour comprendre l'architecture des logiciels du projet *Fais-moi un dessin*. La première section porte sur les objectifs et contraintes architecturaux. Certains éléments comme la confidentialité, la portabilité, la réutilisation, l'échéancier, les coûts et les outils de développement y seront discutés. La deuxième section porte sur les cas d'utilisation. Les diagrammes de cas d'utilisation ainsi que quelques explications y seront présentés. La troisième section présente la vue logique. Il est possible d'y retrouver différents diagrammes de paquetages afin de comprendre l'organisation de nos logiciels. La quatrième section s'attarde plutôt à la vue des processus. Cette section comporte plusieurs diagrammes de séquences pour comprendre les actions logiques possibles. La cinquième section porte sur la vue de déploiement. Les différentes configurations du matériel physique pourront y être retrouvées. Finalement, la dernière section porte sur la taille et la performance des différents logiciels.

2. Objectifs et contraintes architecturaux

Sécurité:
Contraintes: Une autre personne qui est à côté d'un joueur ne peut pas voir le mot de passe qu'un joueur écrit.
Objectifs: L'application doit toujours cacher les mots de passe dans les champs de textes par défaut. Par contre, le mot de passe n'est pas encrypté pour l'envoyer à la base de données.

Confidentialité:
Contraintes: Le projet requiert que certaines informations, notamment associées à un utilisateur, ne soient pas visibles par tous les utilisateurs.
Objectifs: L'adresse email, le prénom, le nom et mot de passe doit être privée. Il faut donc qu'on gère le principe d'identité et de données accessibles pour chaque utilisateur.

Portabilité:
Contraintes: Nous devons développer 2 applications: une qui doit fonctionner sur Windows 10 (PC) et une autre sur une tablette Samsung Galaxy Tab A 2019 avec Android.
Objectifs: La tablette n'offre pas la même capacité de performance que l'ordinateur, alors nous devons nous assurer que les 2 applications fonctionnent correctement dans leur environnements d'exécution particulier.

Réutilisation:
Contraintes: Nous devons utiliser notre application Polydessin créé lors de projet 2.
Objectifs: L'application Angular existe déjà avec plus de fonctionnalités que nécessaires. Nous devons donc nous baser sur l'application Angular existante afin de recréer certaines de ces fonctionnalités sur Android tout en gardant l'interface et l'utilisation aussi similaire que possible entre les deux types de clients.

Échéancier:
Contraintes: Nous devons remettre un appel d'offres le 19 février et l'application finale le 19 avril. Nous avons donc 16 semaines pour compléter le projet.
Objectifs: Le projet sera divisé en sprints de 2 semaines. Il nous faudra donc 1020 heures-personnes pour compléter le projet, et 170 heures-personnes par sprint.

Coûts:
Contraintes: Les développeurs sont payés 100$/h et le gestionnaire de projet 125$/h.
Objectifs: En plus des autres coûts pour les services d'hébergement, nous estimons un coût total s'élevant à 107 300$ + tx.

Outils de développement:
Contraintes: Les outils de développement doivent nous permettre d'optimiser notre travail pour éviter des conflits de code et permettre un travail en parallèle.
Objectifs: Le projet sera développé en parallèle par 6 personnes. Pour s'assurer d'éviter d'avoir des conflits, l'équipe développe les différents composants du projet avec Git et Gitlab. Puisque le projet sera développé pour Electron, Android et sur la base de données d'Azure, l'équipe aura besoin de trois outils de développement majeurs. Pour electron, l'équipe développera le code HTML, CSS et TypeScript avec Visual Studio. Visual Studio sera aussi utilisé pour le développement du serveur Node.js pour le code JavaScript et TypeScript. L'application Android sera

développée sur Android Studio et la base de données sera développée sur Microsoft Azure.

Langages de développement:

Contraintes: Nous devons réutiliser le projet 2 Polydessin, donc nous utilisons le TypeScript avec Angular, HTML et CSS. De plus, pour l'application Android doit utiliser Java ou Kotlin.
Objectifs: Le client lourd (Electron) sera développé avec TypeScript, HTML, CSS, JavaScript pour le framework Angular. Le client léger sera développé avec Kotlin. Le serveur en TypeScript, JavaScript.

3. Vue des cas d'utilisation

Les bulles de cas d'utilisation en bleu proviennent d'exigences souhaitables.

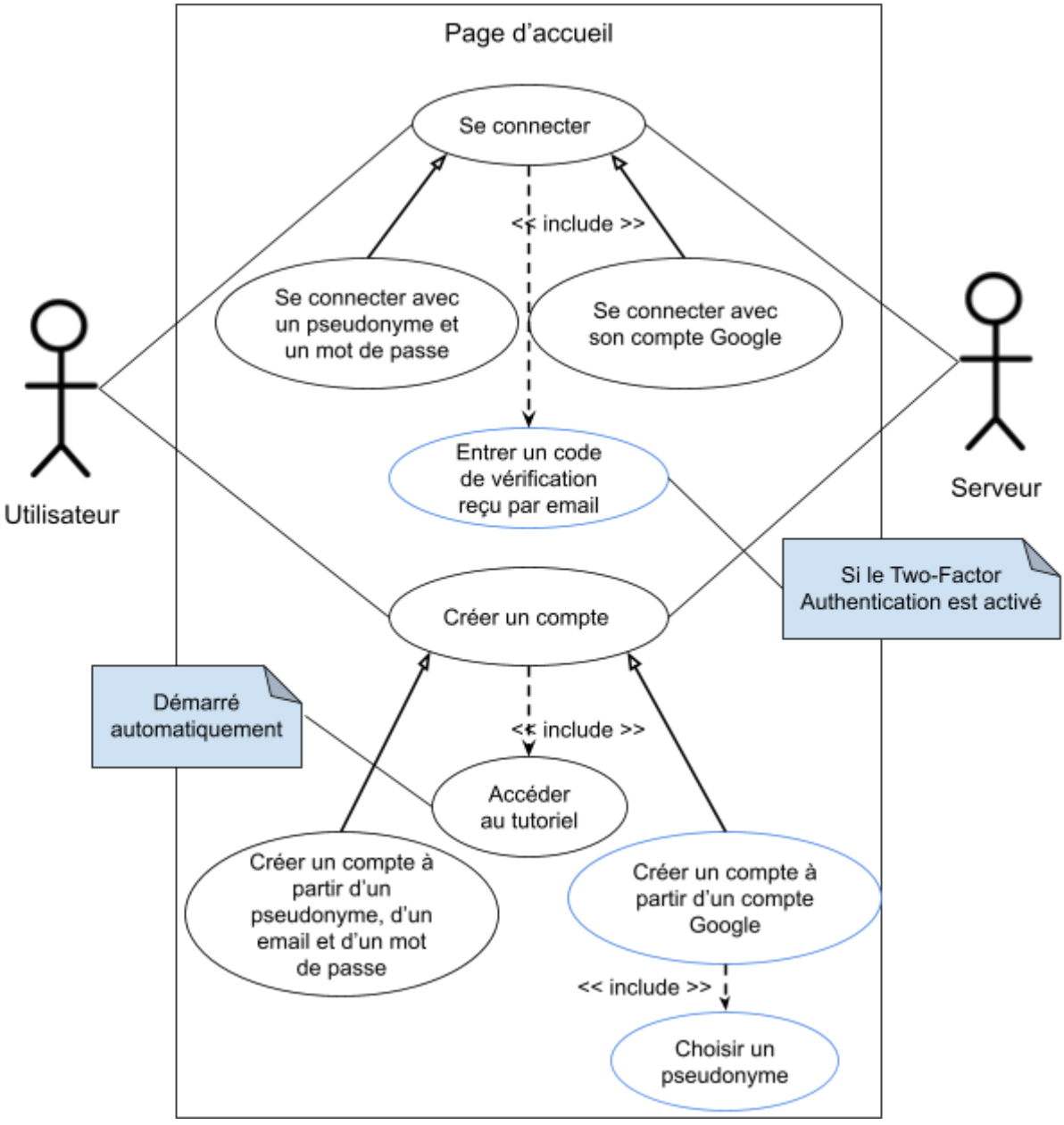

Figure 1: diagramme des cas d'utilisation de la page d'accueil

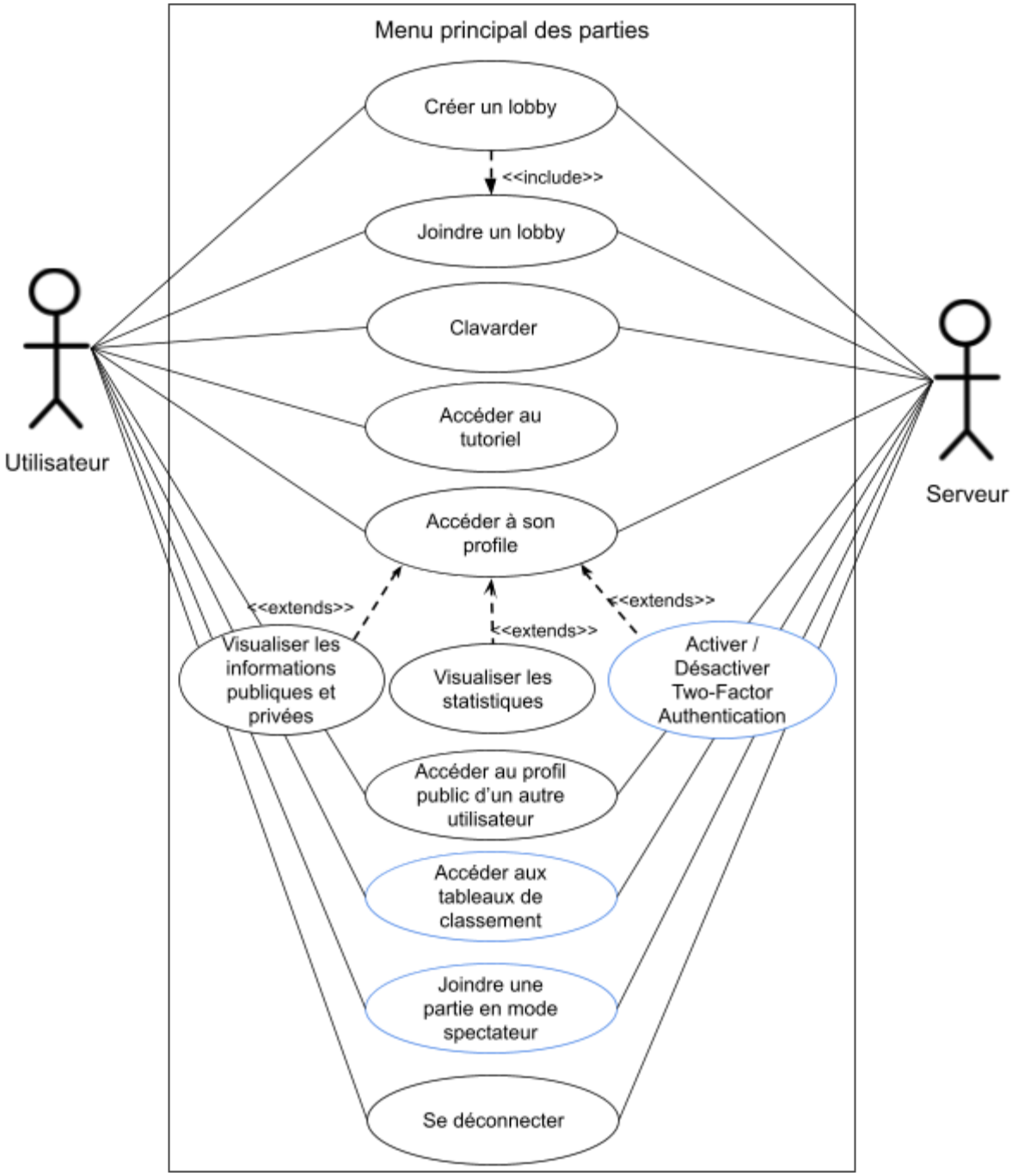

Figure 2: diagramme des cas d'utilisation du menu principal

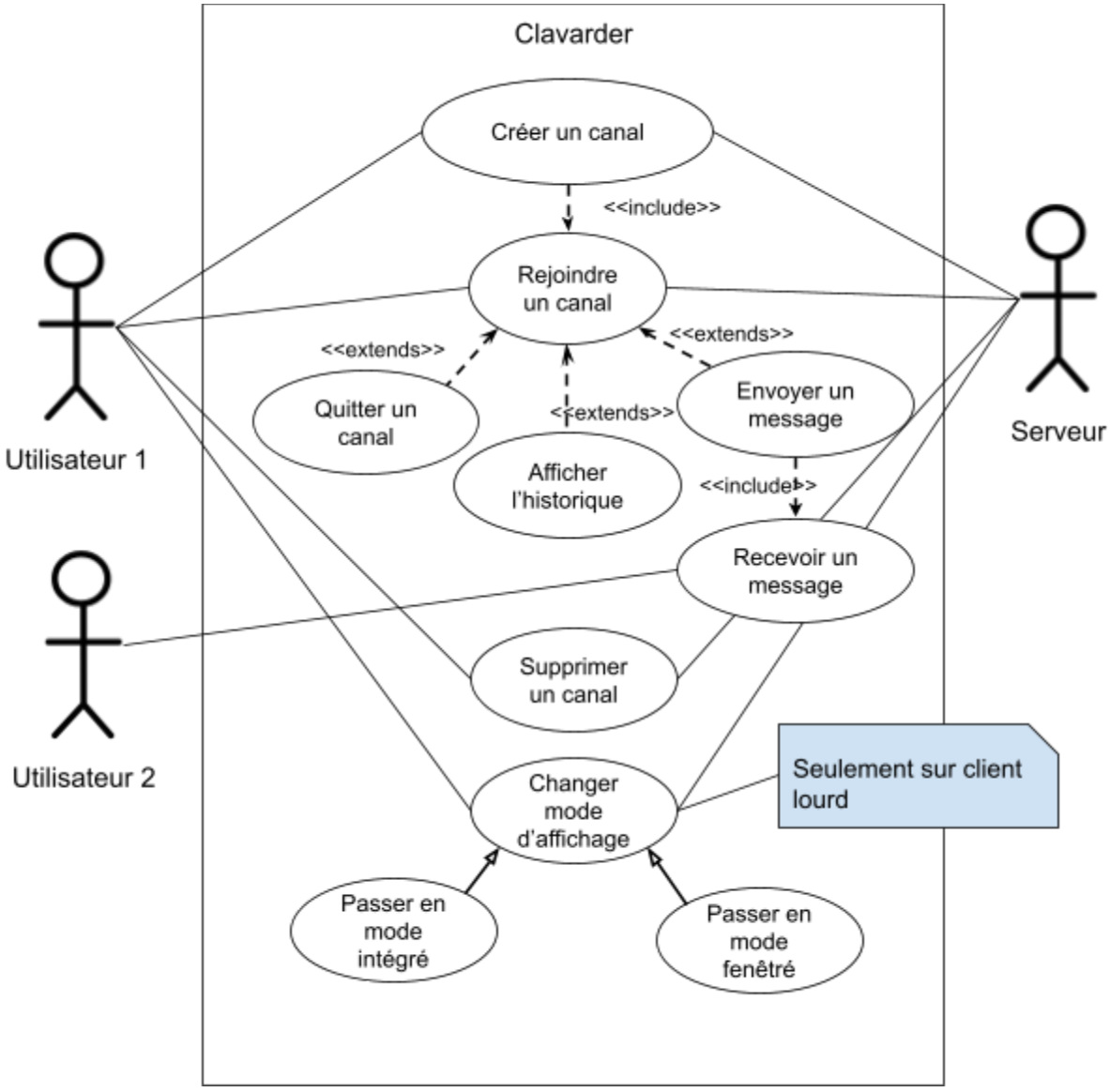

Figure 3: diagramme des cas du clavardage

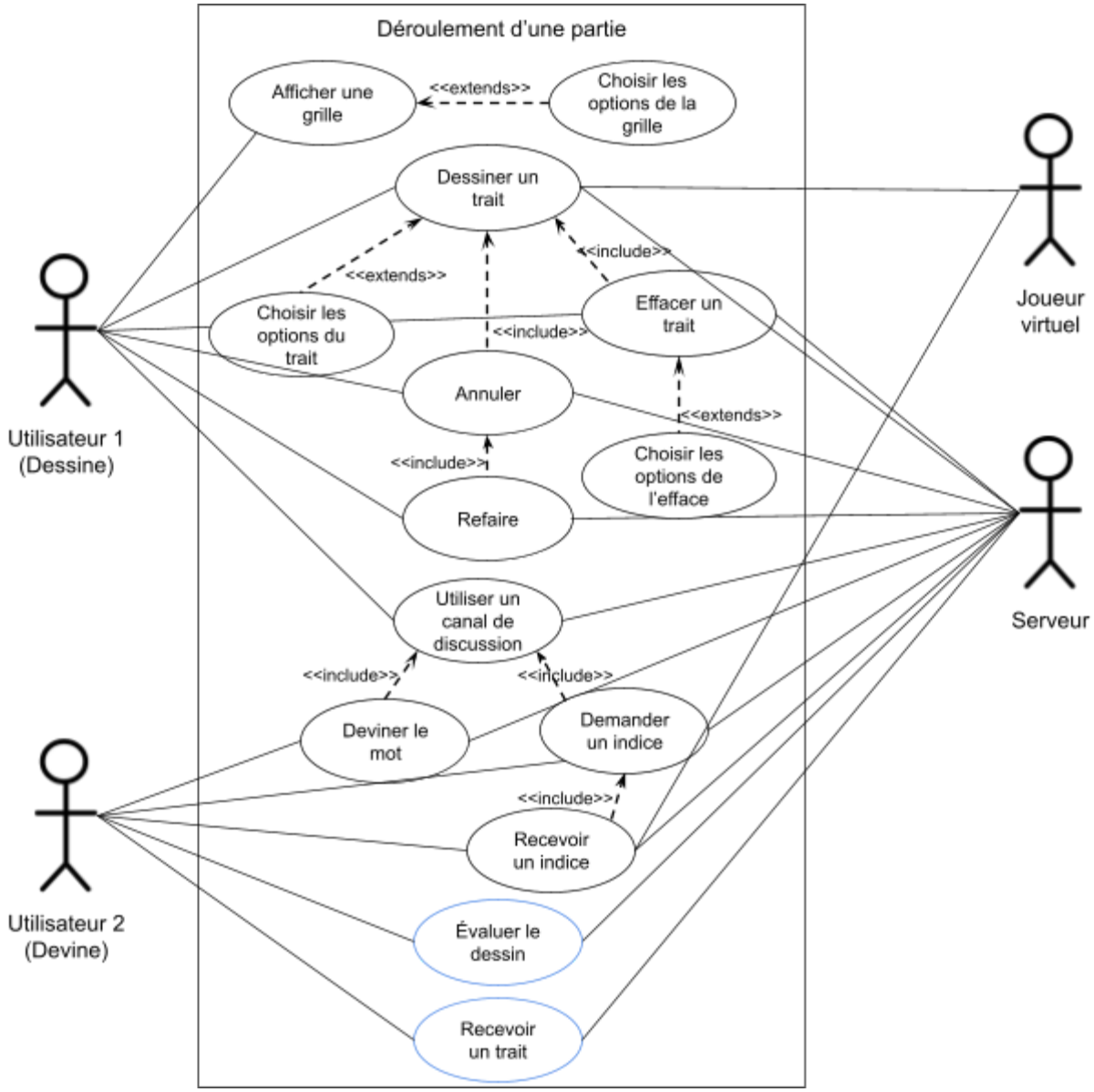

Figure 4: diagramme des cas d'utilisation lors du déroulement d'une partie

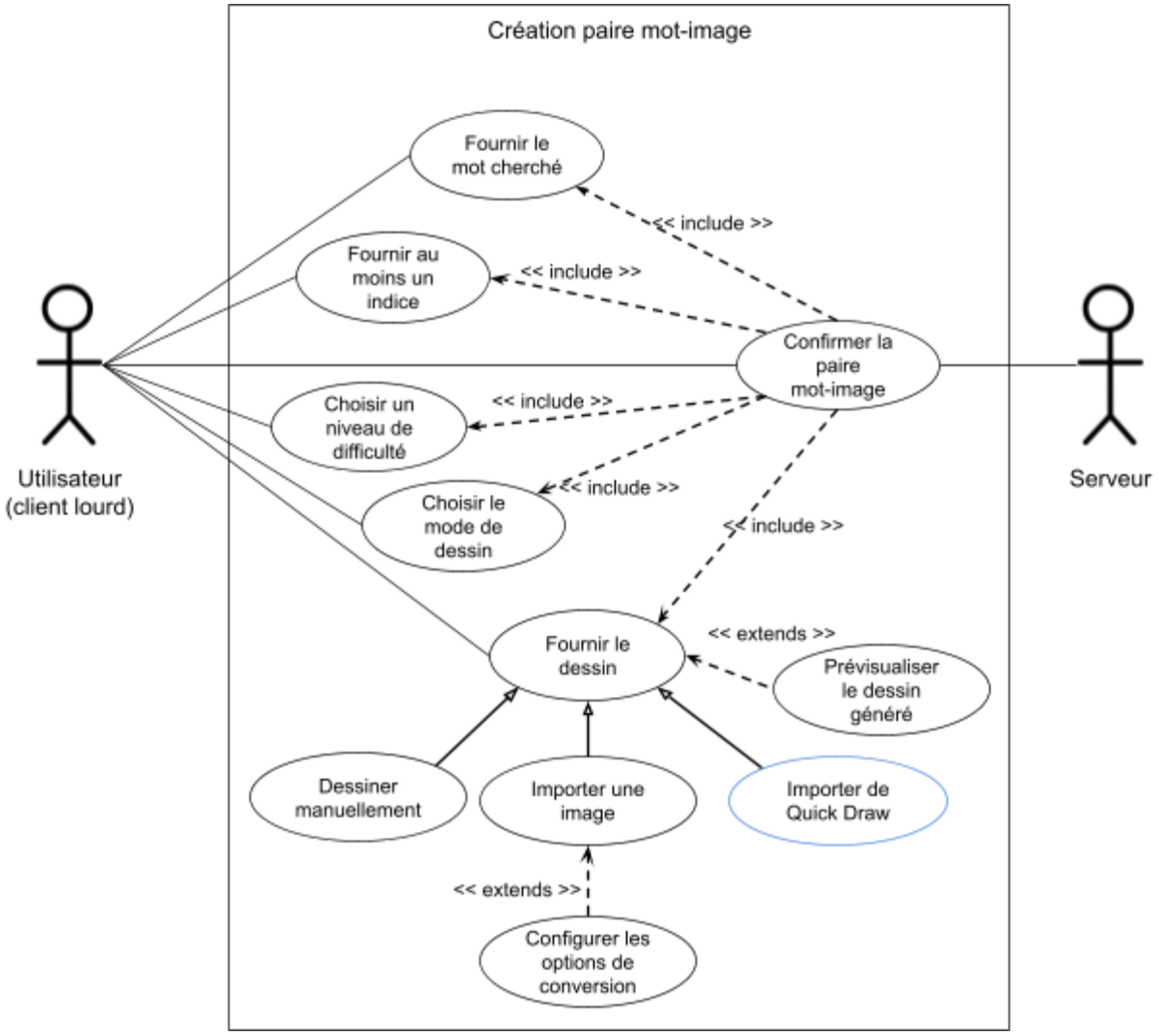

Figure 5: diagramme des cas d'utilisation de la création d'une paire mot-image

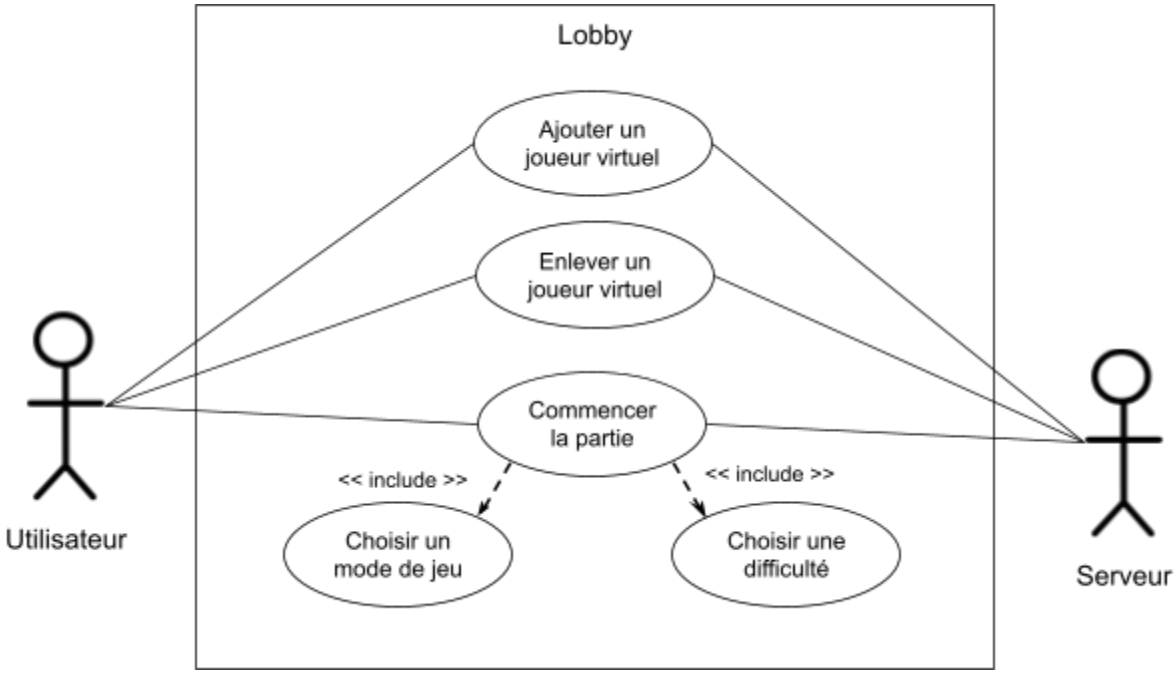

Figure 6: diagramme des cas d'utilisation dans un lobby

4. Vue logique

Fais-moi un dessin
Ce paquetage représente l'ensemble du projet. Il est le diagramme de paquetage du plus haut niveau. Il contient donc les 3 plus grands paquetages de notre projet qui sont le serveur, le client léger et le client lourd. Sa responsabilité principale est d'offrir aux utilisateurs plusieurs modes de jeux multiplateformes par l'entremise d'un serveur.

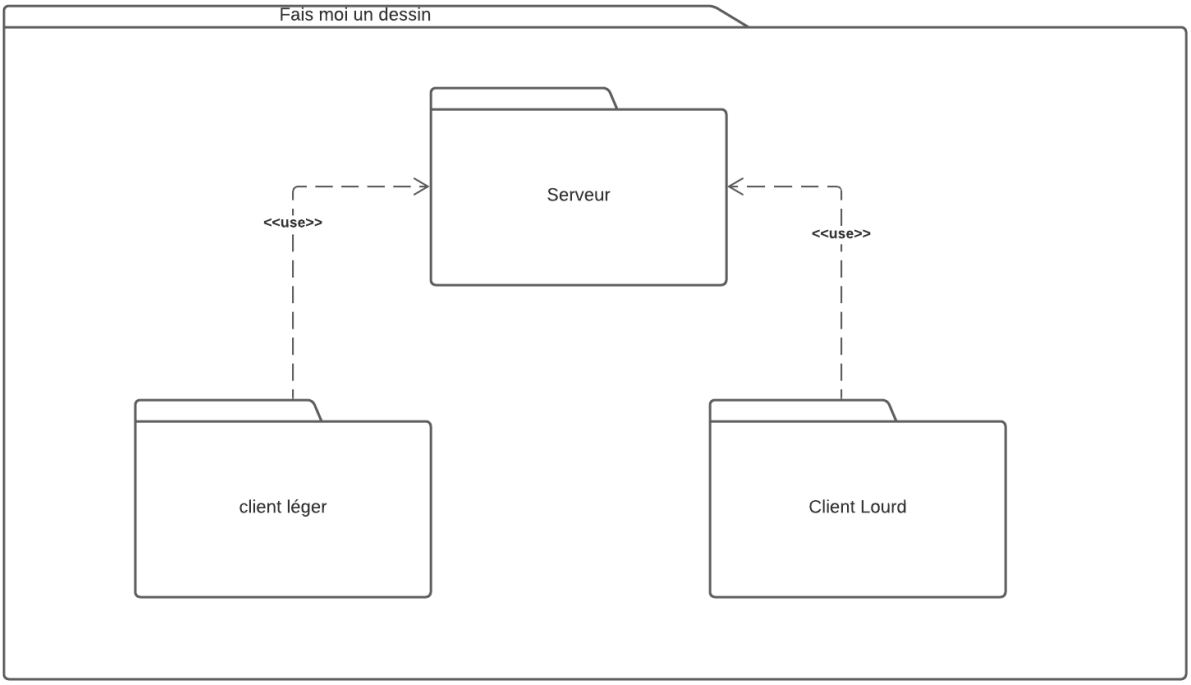

Figure 7: Diagramme de paquetage du projet Fais moi un dessin

Client léger:

Client léger
Ce paquetage représente l'ensemble du client léger qui est l'application Android. Ce paquetage a beaucoup de responsabilités. En général, il doit permettre de rejoindre des lobbys, chatter avec d'autres joueurs et jouer aux différents modes de jeux en entrant en communication avec le serveur par l'entremise de socket ou de requêtes Http.

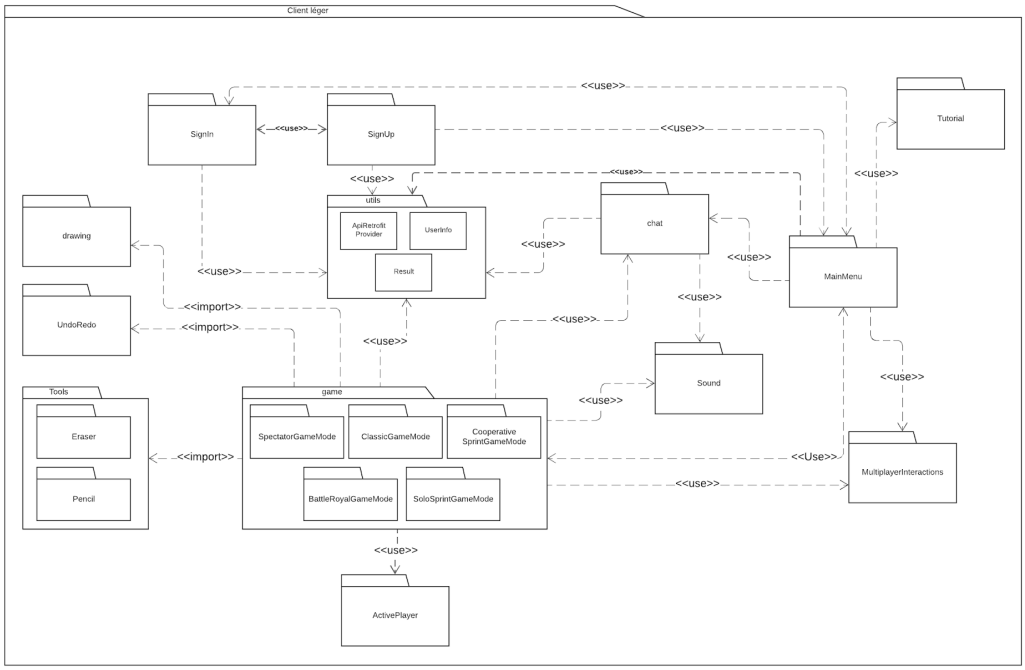

Figure 8: Diagramme de paquetage du client léger

SignIn
Ce paquetage est le point d'entrée principal de l'application. En effet, son but principal est de permettre aux utilisateurs de se connecter en effectuant une requête POST au serveur. Il doit, par la suite, rediriger l'utilisateur.

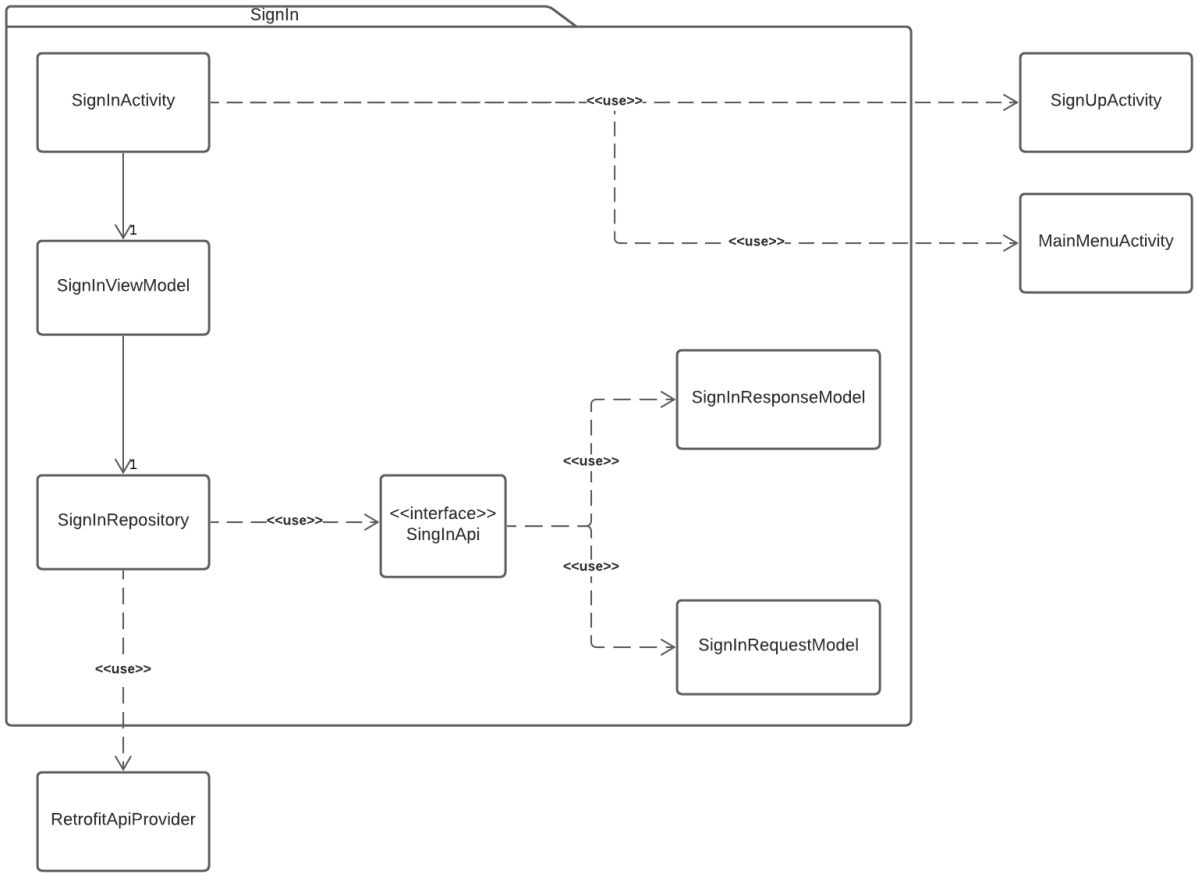

Figure 9: Diagramme de classe de SignIn sur le client léger

SignUp
Ce paquetage est un des points d'entrée pour accéder à l'application. Son but principal est de permettre aux utilisateurs de se créer un compte. Il doit aussi permettre à l'utilisateur de se rendre au menu principal de l'application s'il se crée un compte.

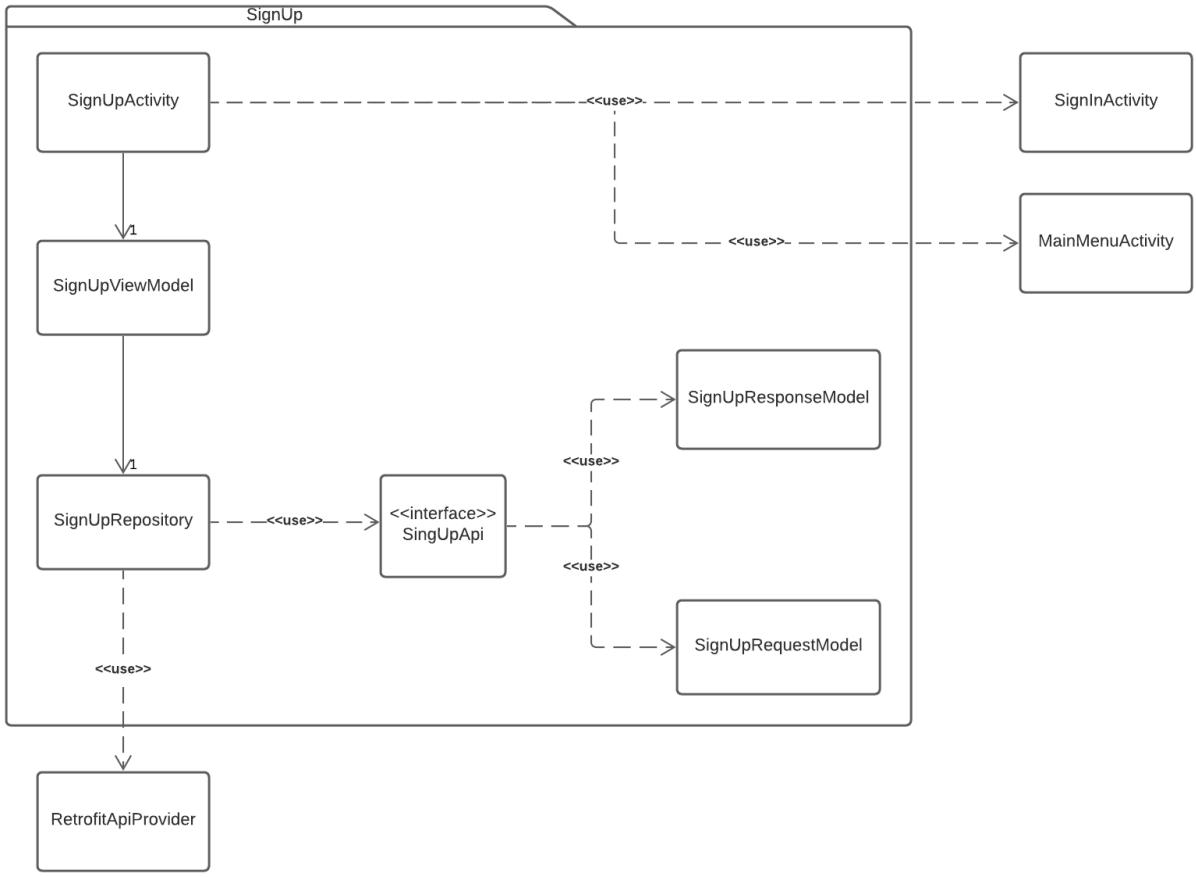

Figure 10: Diagramme de classe de SignUp sur le client léger

UndoRedo
Ce paquetage permet d'annuler et de refaire des actions posées par l'utilisateur sur son dessin. Son principal but est de stocker en format de commande les différentes actions de l'utilisateur sur son dessin et au besoin de les annuler ou les refaire. Il doit aussi transmettre ses commandes au serveur afin de d'annuler et refaire des commandes sur les autres clients connectés à la partie.

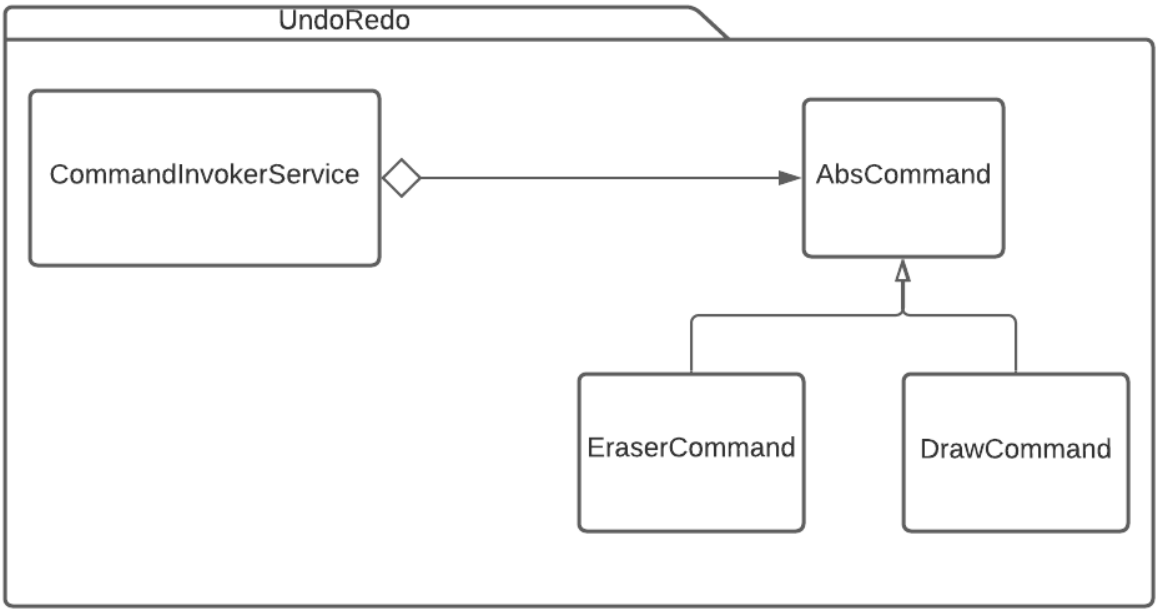

Figure 11: Diagramme de classe de UndoRedo sur le client léger

Drawing
Ce paquetage représente l'endroit ou un dessin sera effectué durant la partie. Sa responsabilité principale sera d'ajouter de permettre de dessiner des "path" et aussi d'en recevoir des autres joueurs lors d'une partie.

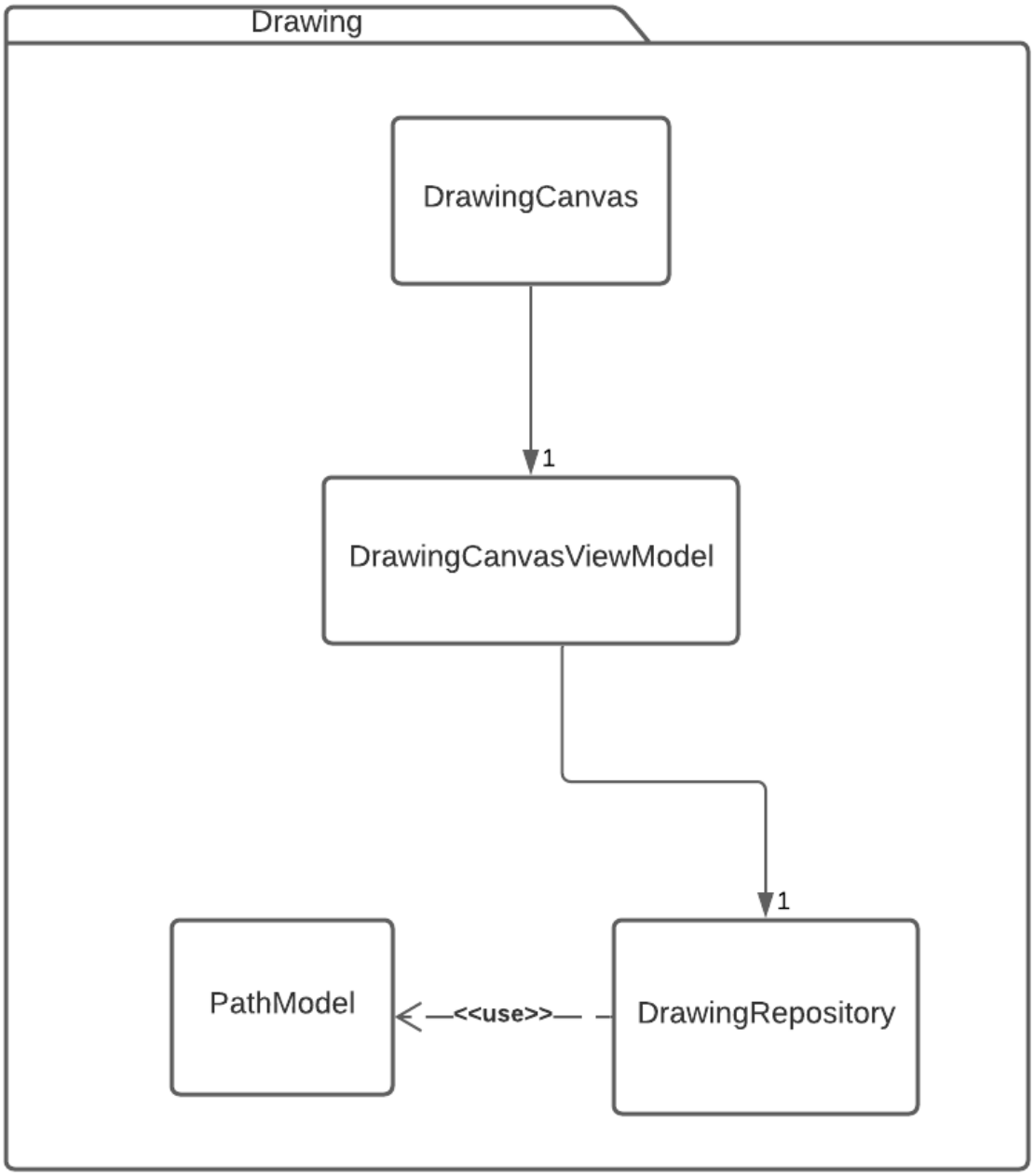

Figure 12: Diagramme de classe de Drawing sur le client léger

MainMenu
Ce paquetage représente le menu principal de l'application. Principalement, ses tâches seront de rediriger l'utilisateur vers le choix qu'il aura effectué dans le menu principal. Il pourra se déconnecter, créer un lobby, rejoindre un lobby, regarder son profil, regarder sa liste d'amis, regarder le profil de d'autres joueurs et effectuer le tutoriel.

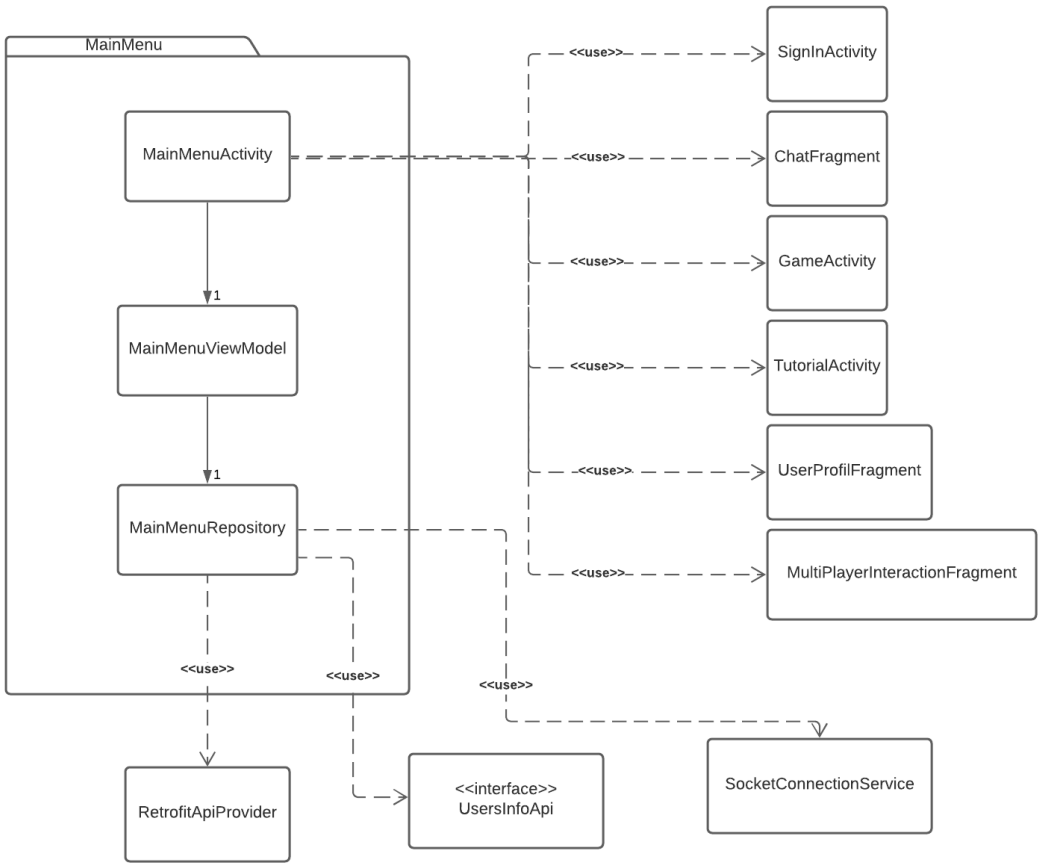

Figure 13: Diagramme de classe de MainMenu sur le client léger

Tutoriel
Ce paquetage représente le tutoriel de l'application qui montrera à l'utilisateur les différentes options qui s'offrent à lui. retrouve dans l'application, cependant l'utilisateur devra effectuer les actions demandé par le tutoriel et sera forcé de faire ce qui est demandé puisque les autres actions, normalement possible, seront bloquées.

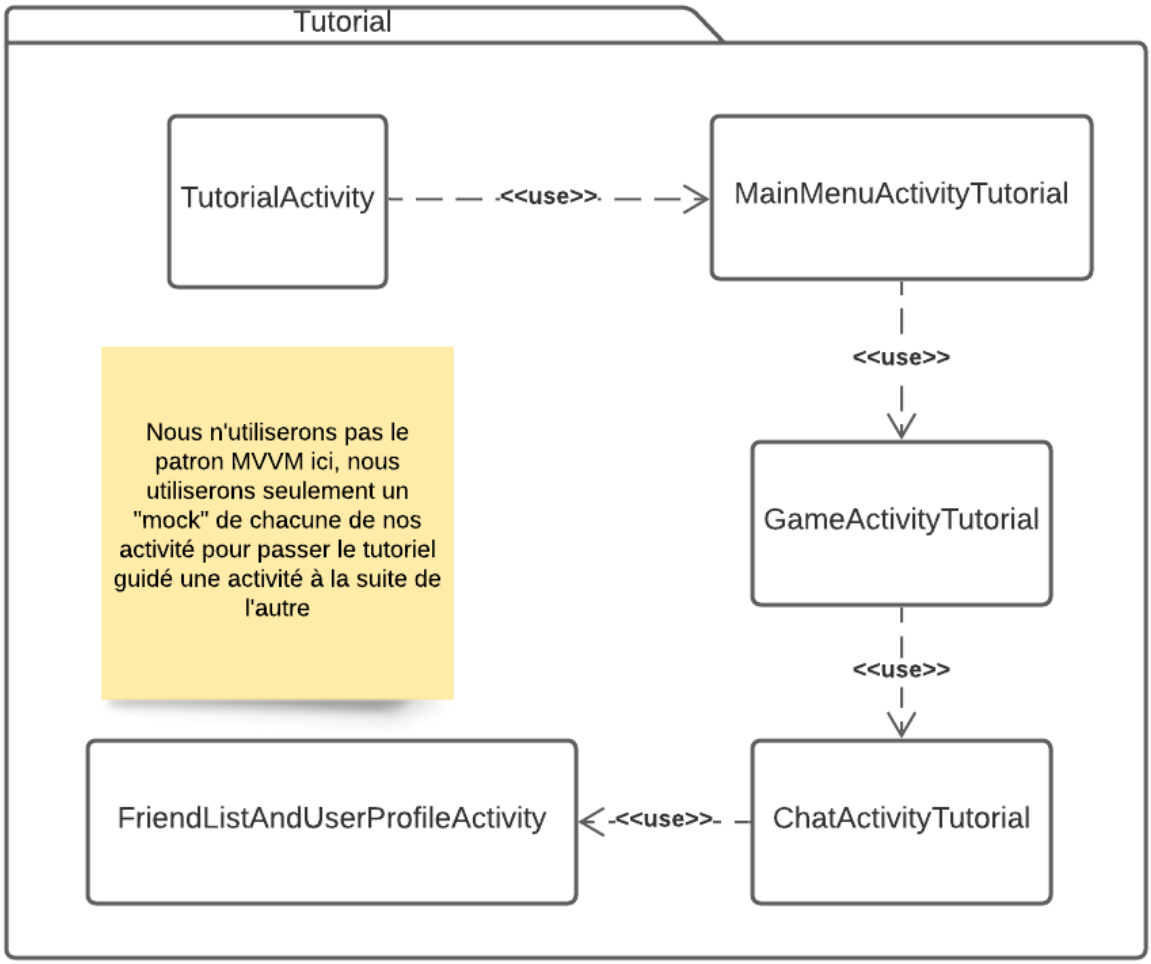

Figure 14: Diagramme de classe de Tutorial sur le client léger

Game
Ce paquetage touche tout ce qui est en lien avec les différents modes de jeux, À l'intérieur de ce paquetage, on retrouve 5 paquets qui sont les 5 modes de jeux disponibles. Son but principal est de contrôler les activités lors d'une partie et de rester en connexion continue avec le serveur à l'aide d'un socket. Il devrait contrôler le temps, les modifications sur le dessin, les tentatives de réponses, le chat, etc.

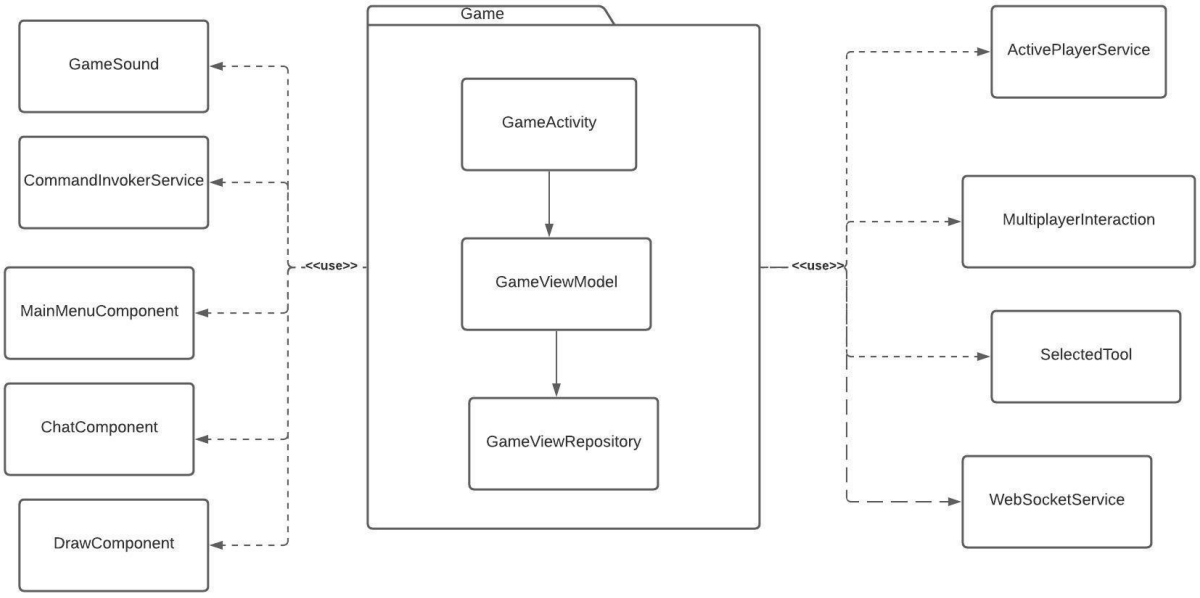

Figure 15: Diagramme de classe de Game sur le client léger

Chat
Ce paquetage représente tout ce qui est en lien avec la discussion avec d'autres joueurs. Ses tâches principales sont de recevoir des messages provenant d'autres clients et de les afficher, distinguer les messages d'un chat room d'un autre et de communiquer des tentatives de réponses durant une partie.

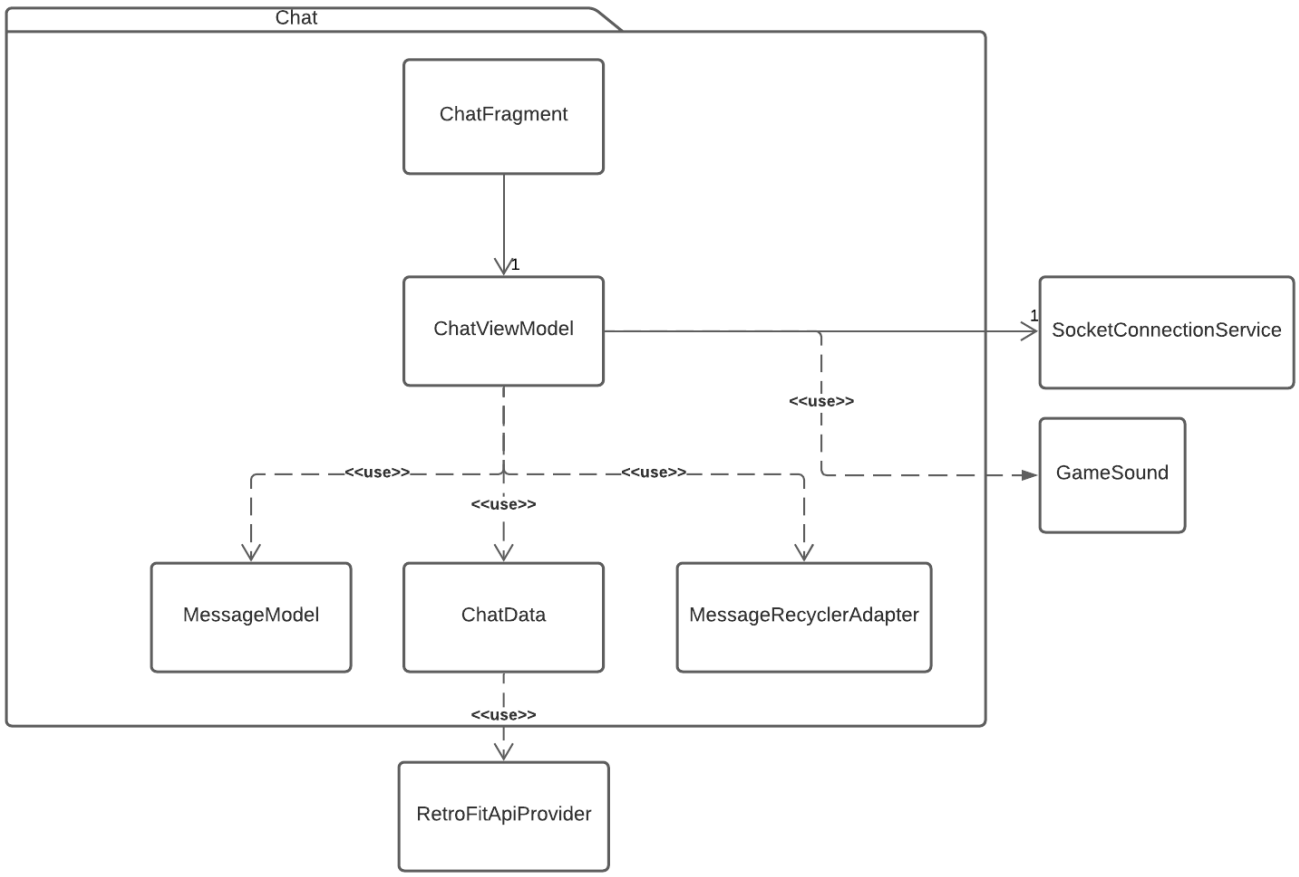

Figure 16: Diagramme de classe de Chat sur le client léger

Tools
Ce paquetage représente les outils disponibles à l'utilisateur lorsqu'il veut effectuer un dessin. Le but principal de ce paquetage est de fournir le comportement désiré lors d'un touché à l'écran en fonction de l'outil choisi.

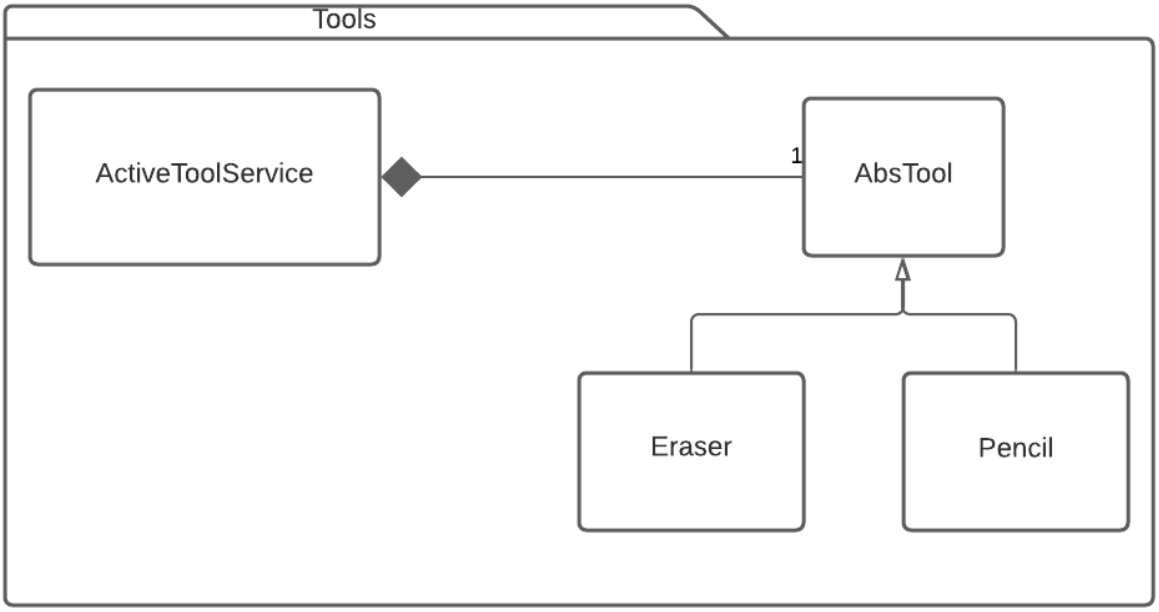

Figure 17: Diagramme de classe de Tools sur le client léger

Sound
Ce paquetage s'occupe des différents sons que l'on retrouve dans l'application. Son but principal est de faire jouer le son désiré lorsqu'un événement particulier est observé, par exemple un nouveau message.

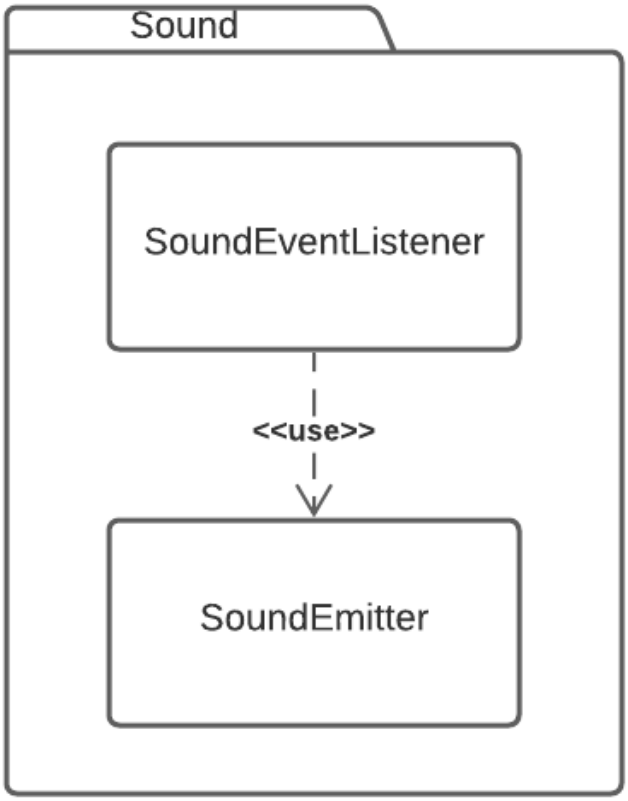

Figure 18: Diagramme de classe de Sound sur le client léger

MultiplayerInteractions
Ce paquetage contrôle l'ensemble des interactions entre les joueurs durant le jeu. Son but principal est d'envoyer des pouces rouge ou vert aux autres joueurs pour noter leur dessins. Un autre de ses buts est de permettre aux joueurs virtuels d'interagir avec les autres joueurs en fonction des statistiques des autres joueurs.

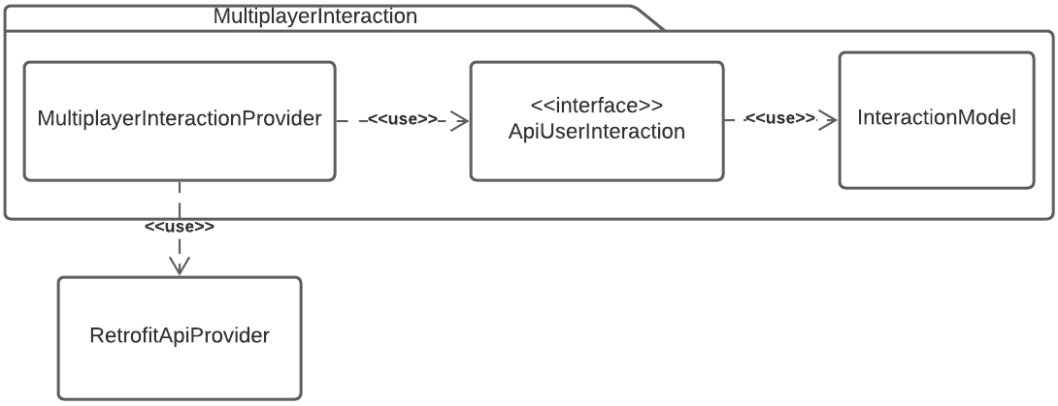

Figure 19: Diagramme de classe de MultiplayerInteraction sur le client léger

Client lourd:

Client lourd
Ce paquetage représente l'ensemble du client lourd qui est l'application Electron. Ce paquetage a beaucoup de responsabilités. En général, il doit permettre de rejoindre des lobbys, chatter avec d'autres joueurs, créer des paires mots images et jouer aux différents modes de jeux en entrant en communication avec le serveur par l'entremise de socket ou de requêtes Http.

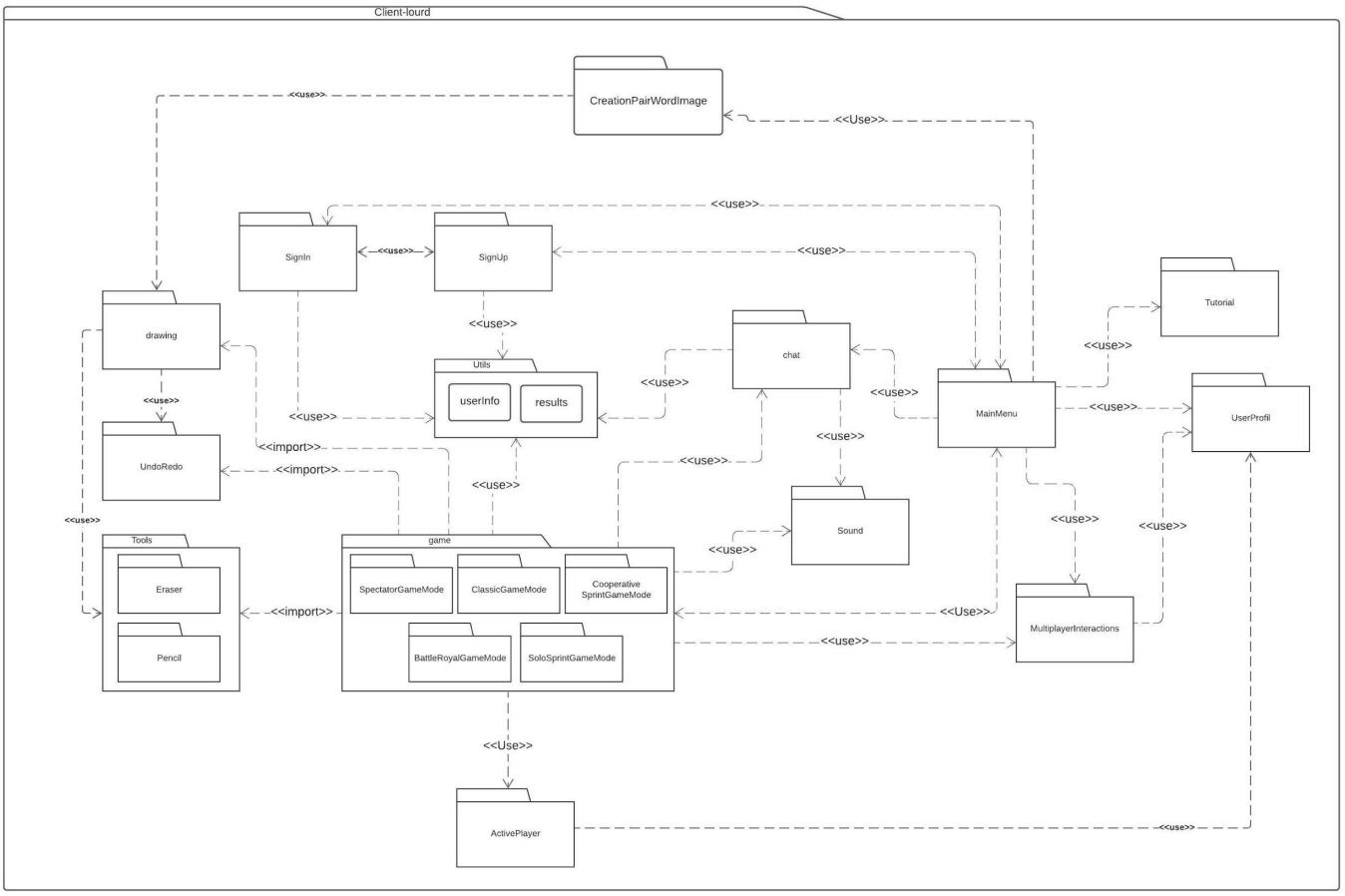

Figure 20: Diagramme de paquetage du client lourd

SignIn
Ce paquetage est le point d'entrée principale de l'application. En effet, son but principal est de permettre aux utilisateurs de se connecter en effectuant une requête POST au serveur. Il doit, par la suite, rediriger l'utilisateur vers le menu principal.

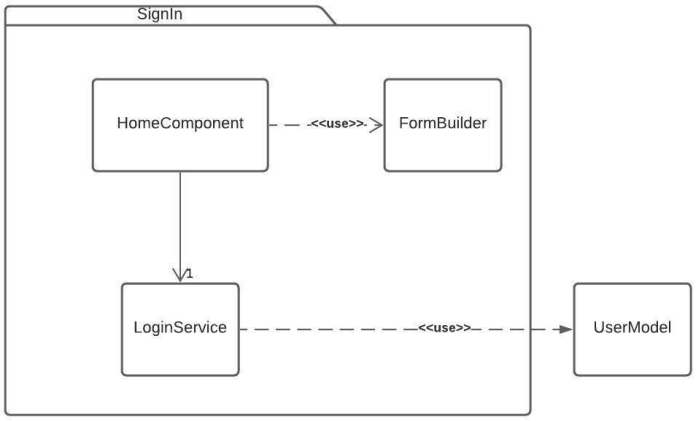

Figure 21: Diagramme de classe de SignIn sur le client lourd

SignUp
Ce paquetage est un des points d'entrée pour accéder à l'application. Son but principal est de permettre aux utilisateurs de se créer un compte. Il doit aussi permettre à l'utilisateur de se rendre au menu principal de l'application s'il se créer un compte.

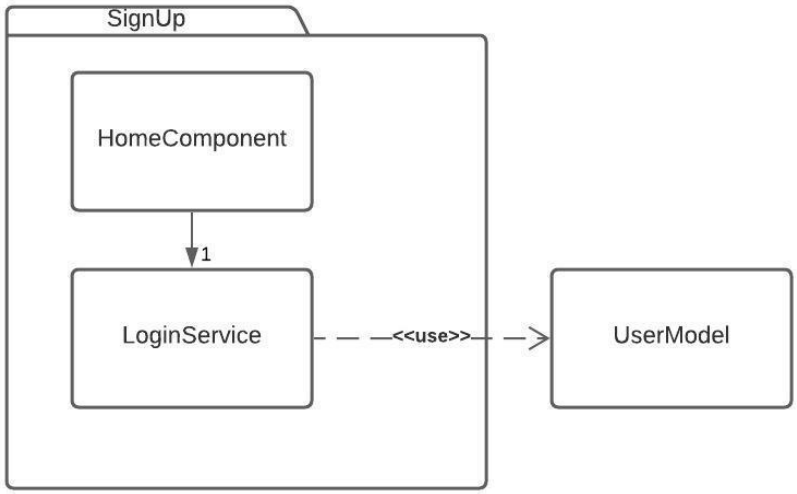

Figure 22: Diagramme de classe de SignUp sur le client lourd

UndoRedo
Ce paquetage permet d'annuler et de refaire des actions posées par l'utilisateur sur son dessin. Son principal but est de stocker en format de commande les différentes actions de l'utilisateur sur son dessin et au besoin de les annuler ou les refaire. Il doit aussi transmettre ses commandes au serveur afin de d'annuler et refaire des commandes sur les autres clients connectés à la partie.

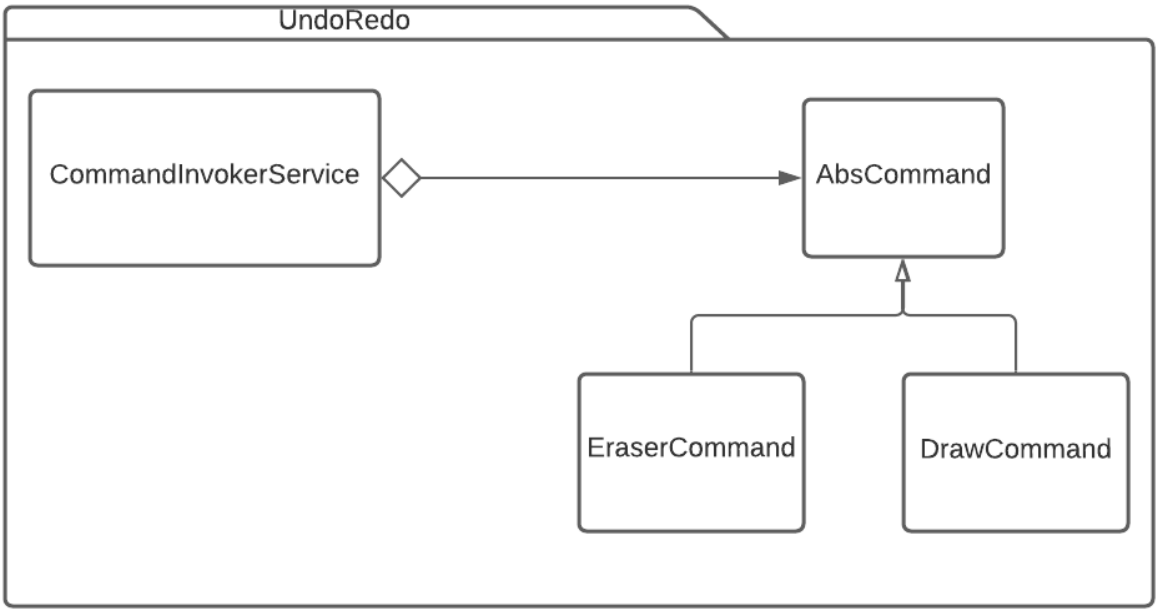

Figure 22: Diagramme de classe de undoRedo sur le client lourd

Drawing
Ce paquetage représente l'endroit ou un dessin sera effectué durant la partie. Sa responsabilité principale sera d'ajouter de permettre de dessiner des "path" et aussi d'en recevoir des autres joueurs lors d'une partie.

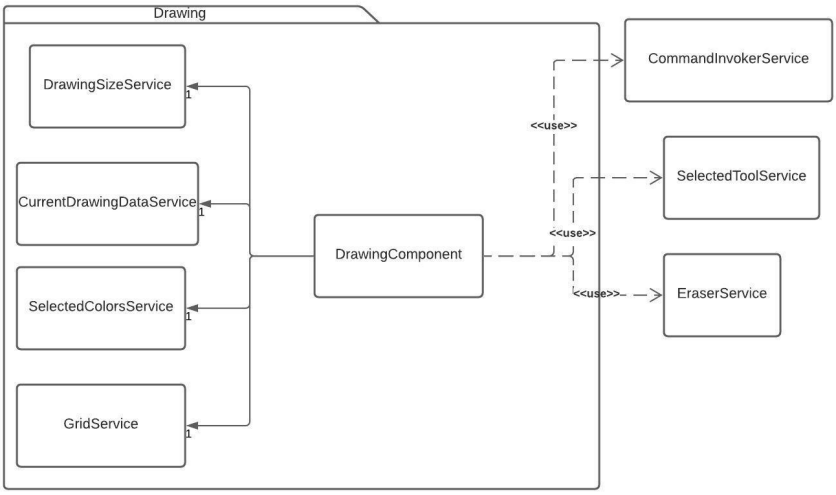

Figure 23: Diagramme de classe de drawing sur le client lourd

MainMenu
Ce paquetage représente le menu principal de l'application. Principalement, ses tâches seront de rediriger l'utilisateur vers le choix qu'il aura effectué dans le menu principal. Il pourra se déconnecter, créer un lobby, rejoindre un lobby, regarder son profil, regarder sa liste d'amis, regarder le profil de d'autres joueurs et effectuer le tutoriel.

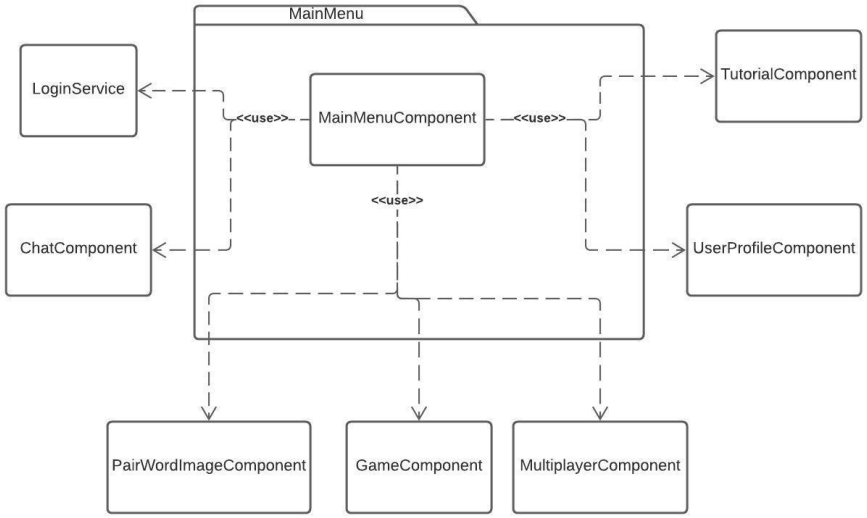

Figure 24: Diagramme de classe de mainMenu sur le client lourd

Tutorial
Ce paquetage représente le tutoriel de l'application qui montrera à l'utilisateur les différentes options qui s'offrent à lui. Ce paquetage est un ensemble d'activités ayant les mêmes écrans que l'on retrouve dans l'application, cependant l'utilisateur devra effectuer les actions demandées par le tutoriel et sera forcé de faire ce qui est demandé puisque les autres actions, normalement possible, seront bloquées.

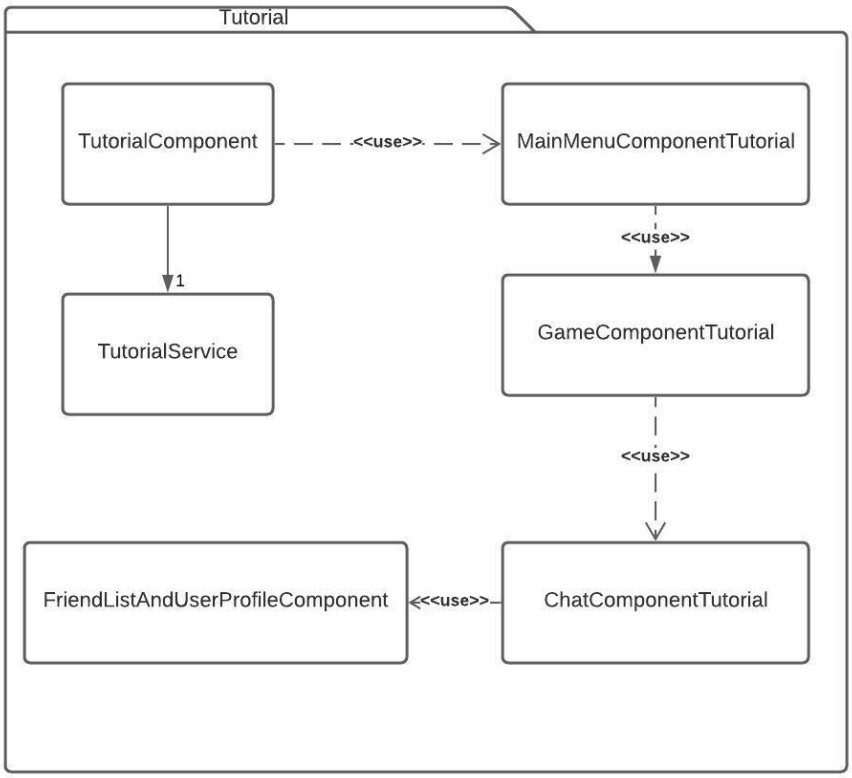

Figure 25 : Diagramme de classe de Tutorial sur le client lourd

Games
Ce paquetage touche tout ce qui est en lien avec les différents modes de jeux, À l'intérieur de ce paquetage, on retrouve 5 paquets qui sont les 5 modes de jeux disponibles. Son but principal est de contrôler les activités lors d'une partie et de rester en connexion continue avec le serveur à l'aide d'un socket. Il devrait contrôler le temps, les modifications sur le dessin, les tentatives de réponses, le chat, etc. Le paquetage GameMode s'applique pour les différent mode de jeu, soit classique, battle royal, coop et solo.

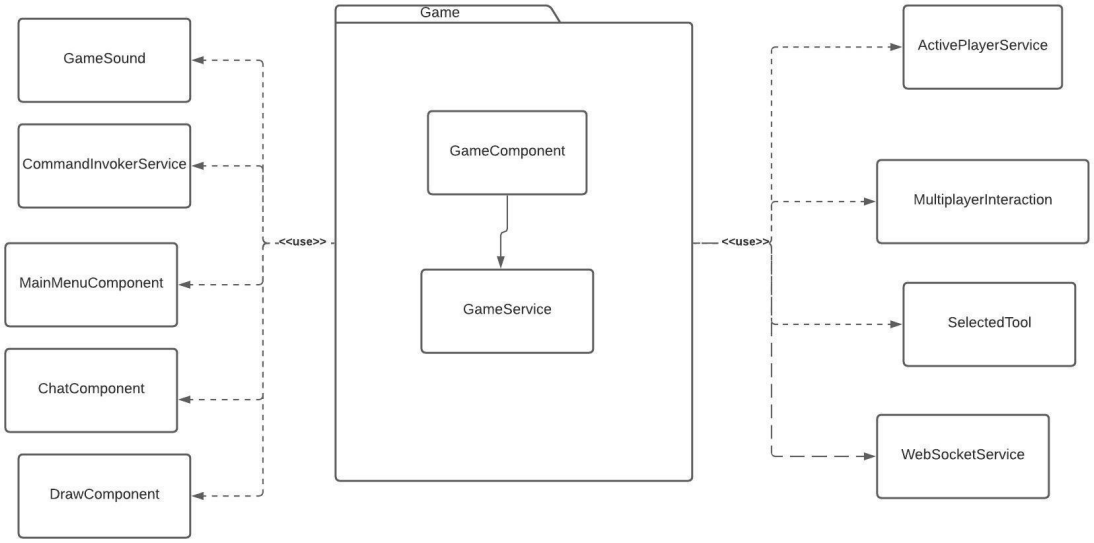

Figure 26: Diagramme de classe de game sur le client lourd

Chat
Ce paquetage représente tout ce qui est en lien avec la discussion avec d'autres joueurs. Ses tâches principales sont de recevoir des messages provenant d'autres clients et de les afficher, distinguer les messages d'un chat room d'un autre et de communiquer des tentatives de réponses durant une partie.

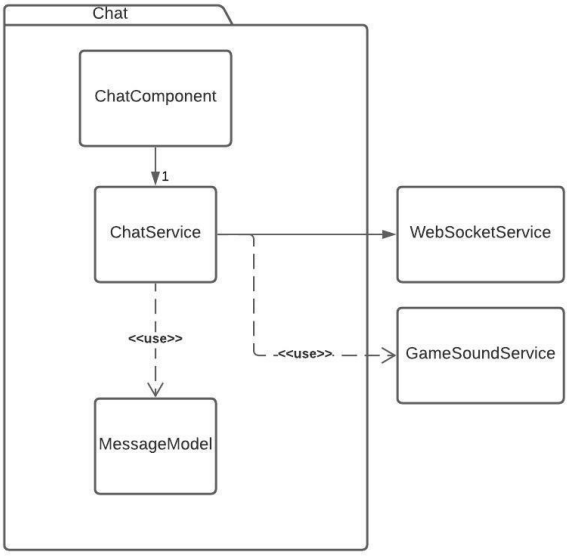

Figure 27: Diagramme de classe de chat sur le client lourd

Tools
Ce paquetage représente les outils disponibles à l'utilisateur lorsqu'il veut effectuer un dessin. Le but principal de ce paquetage est de fournir le comportement désiré lors d'un touché à l'écran en fonction de l'outil choisi.

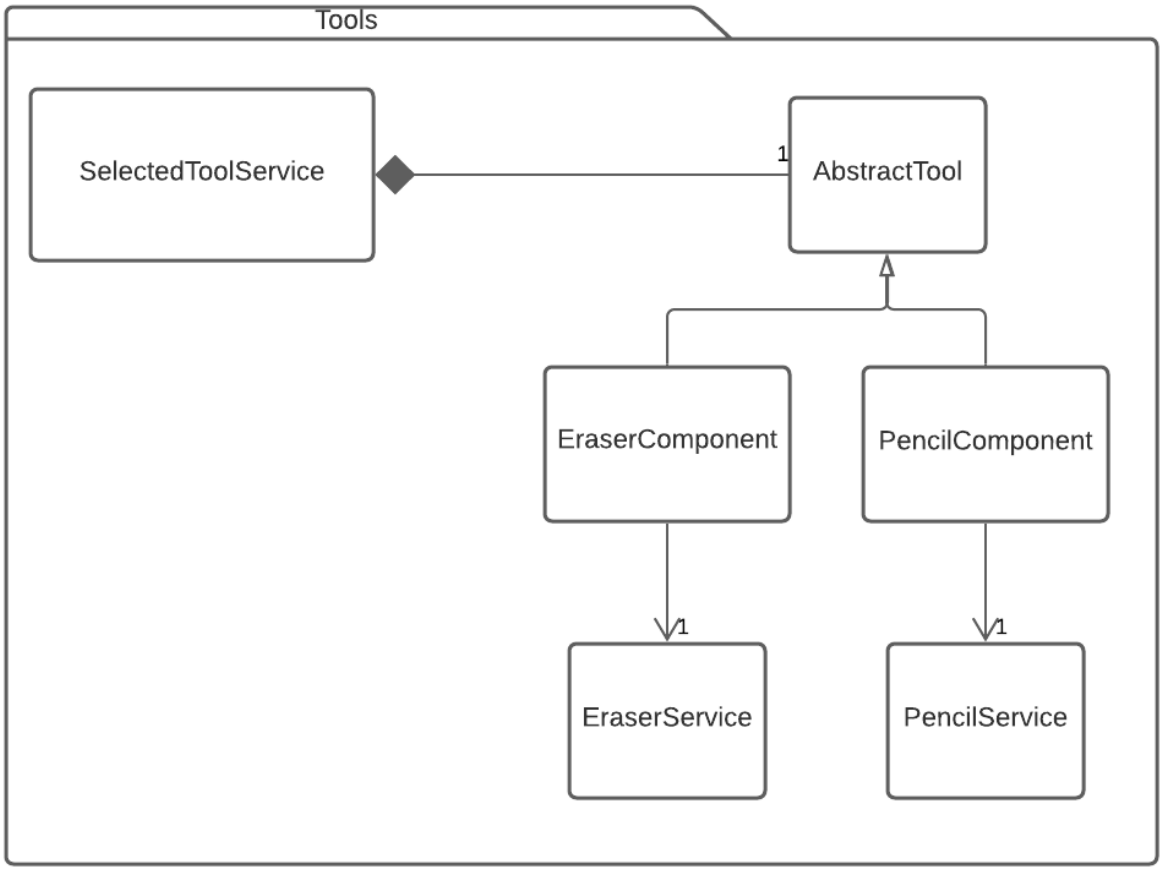

Figure 28: Diagramme de classe de tools sur le client lourd

Sound
Ce paquetage s'occupe des différents sons que l'on retrouve dans l'application. Son but principal est de faire jouer le son désiré lorsqu'un événement particulier est observé, par exemple un nouveau message.

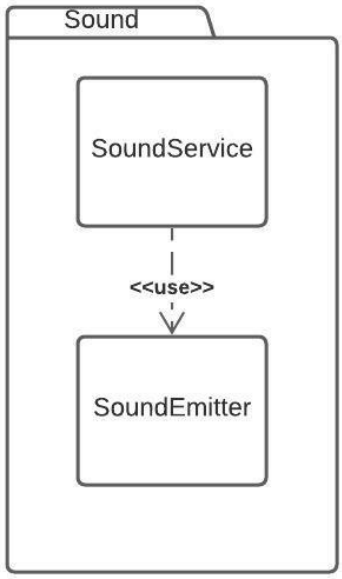

Figure 29: Diagramme de classe de sound sur le client lourd

MultiplayerInteraction
Ce paquetage contrôle l'ensemble des interactions entre les joueurs durant le jeu. Son but principal est d'envoyer des pouces rouge ou vert aux autres joueurs pour noter leur dessins. Un autre de ses buts est de permettre aux joueurs virtuels d'interagir avec les autres joueurs en fonction des statistiques des autres joueurs.

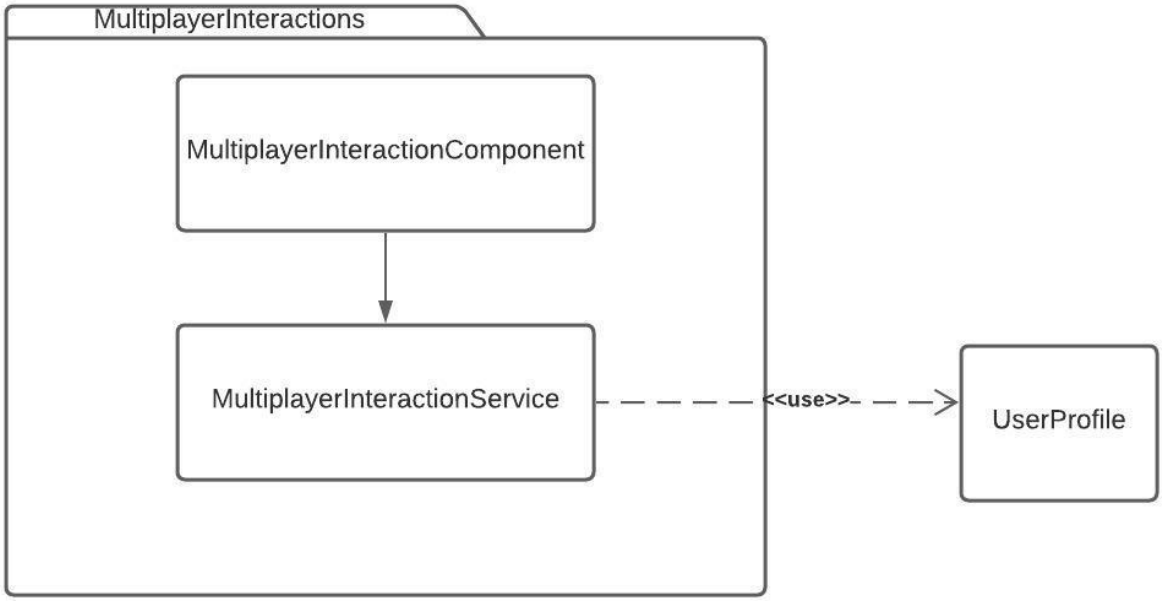

Figure 30: Diagramme de classe de MultiplayerInteraction sur le client lourd

PairWordImage
Ce paquetage s'occupe de la création des paires mot-image sur le client lourd uniquement. Son but principal est de permettre aux utilisateurs du client lourds de lié un mot à deviner avec un dessin qu'un joueur virtuel pourra ensuite recréer lors d'une partie.

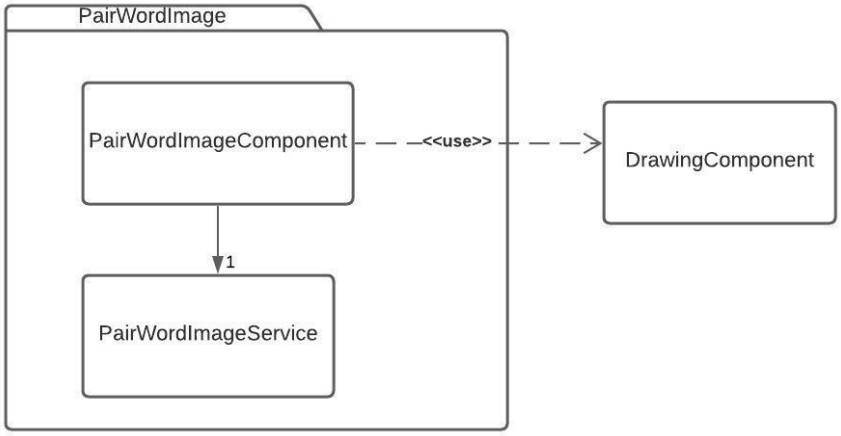

Figure 31: Diagramme de classe de pairWordImage sur le client lourd

Serveur:

Serveur
Ce paquetage représente l'ensemble du serveur communiquant avec les clients. Ce paquetage contient donc les fonctionnalités essentielles afin que les clients communiquent entre eux ainsi que la connexion à la base de données afin de sauvegarder des informations provenant des clients et de leur envoyer lorsque nécessaire.

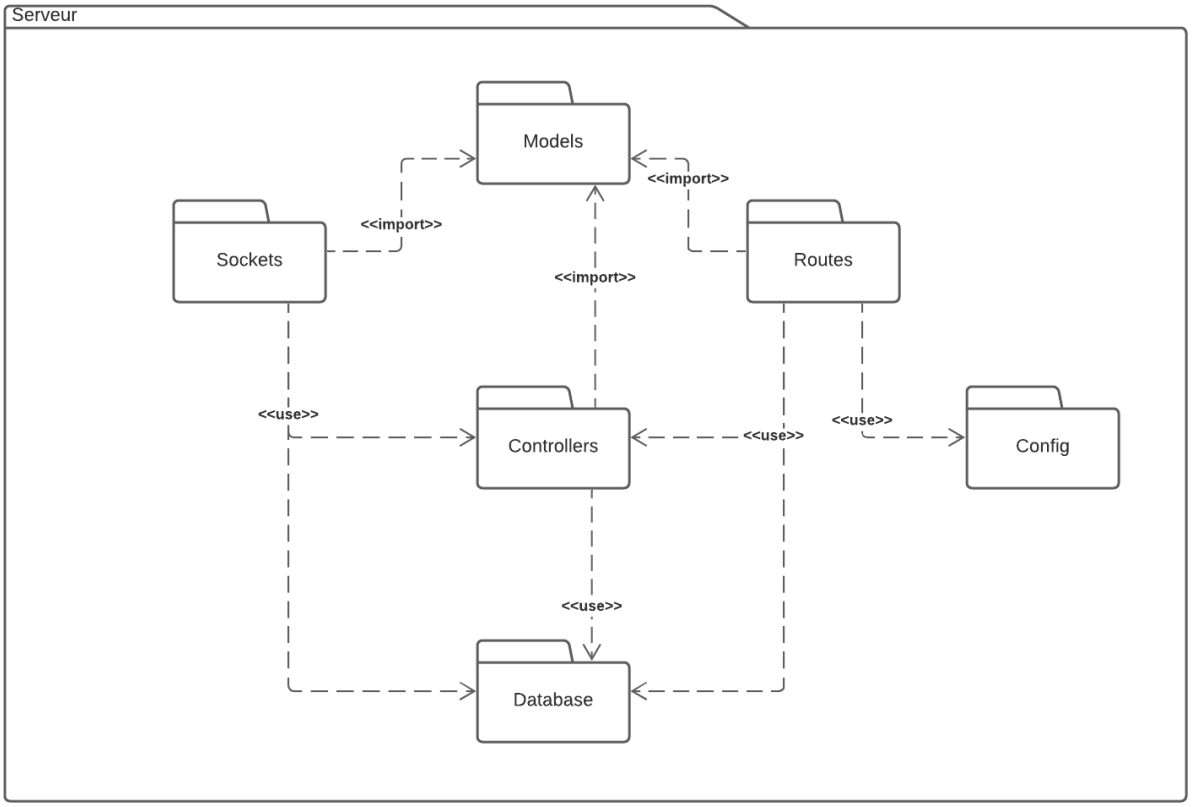

Figure 32: Diagramme de paquetage du serveur

Routes
Ce paquetage contient toutes les routes disponibles par requêtes HTTP (GET, POST, etc.) Chaque classe correspond à un endpoint. De plus, chaque classe doit rester le plus simple possible en ne s'occupant que de recevoir une requête, d'appeler la méthode correspondante dans le paquetage Controllers, puis de retourner la réponse.

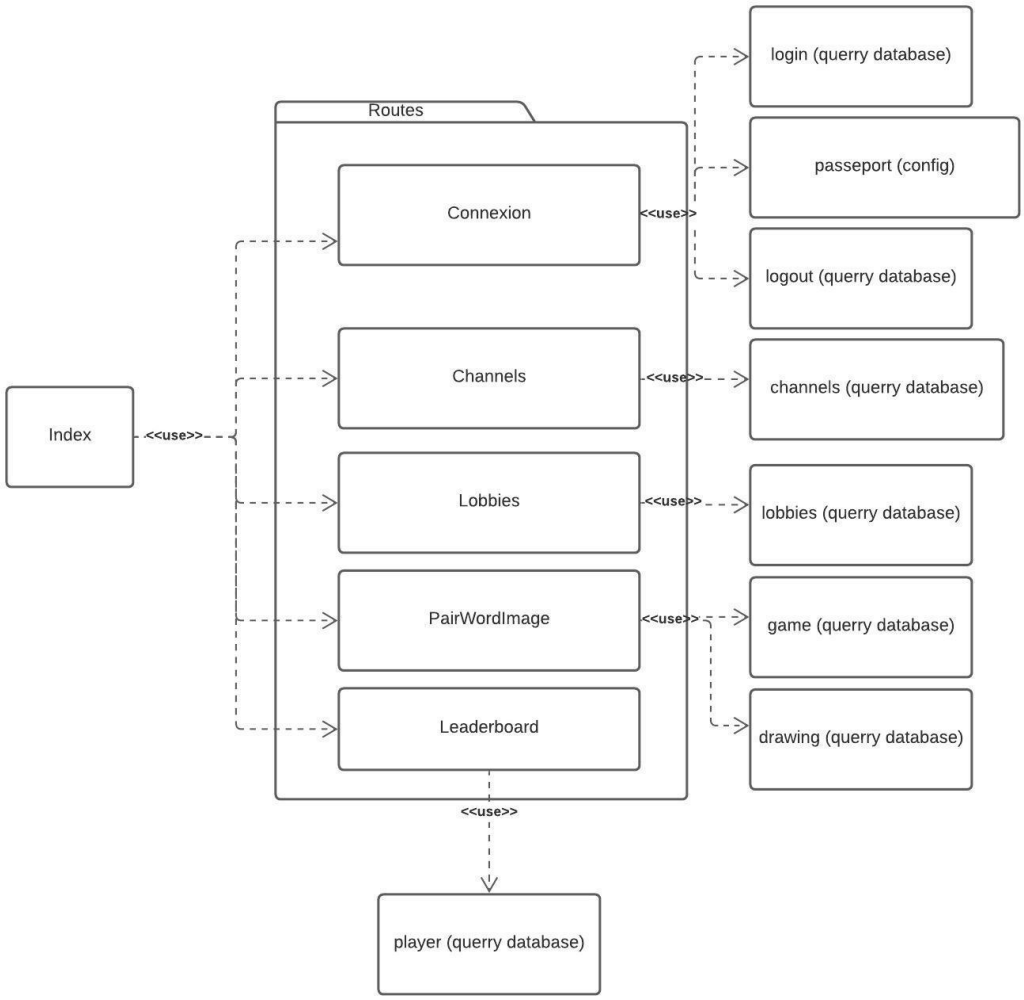

Figure 33: Diagramme de classe du routage pour le serveur

Sockets
Ce paquetage contient tous les événements disponibles par WebSockets. Chaque classe correspond à une fonctionnalité et peut donc avoir plusieurs *listeners* de WebSockets selon la fonctionnalité associée.

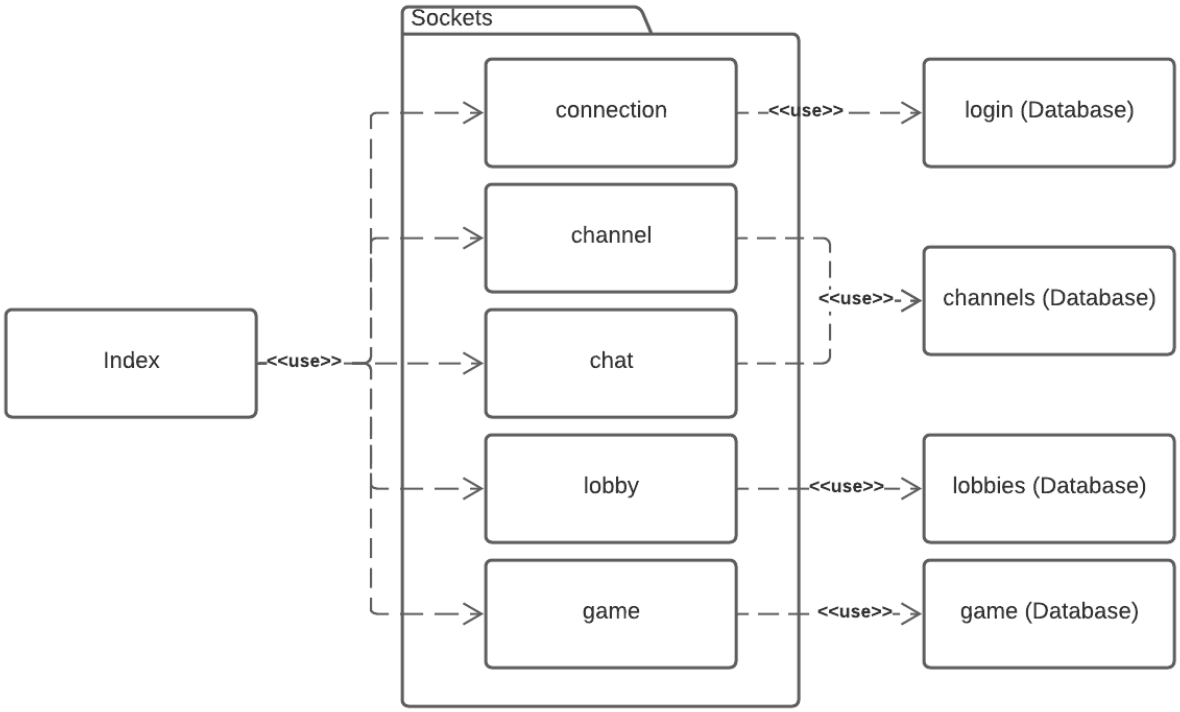

Figure 34: Diagramme de classe des sockets pour le serveur

Models
Ce paquetage contient des représentations d'entités utiles pour transférer des informations avec les clients.

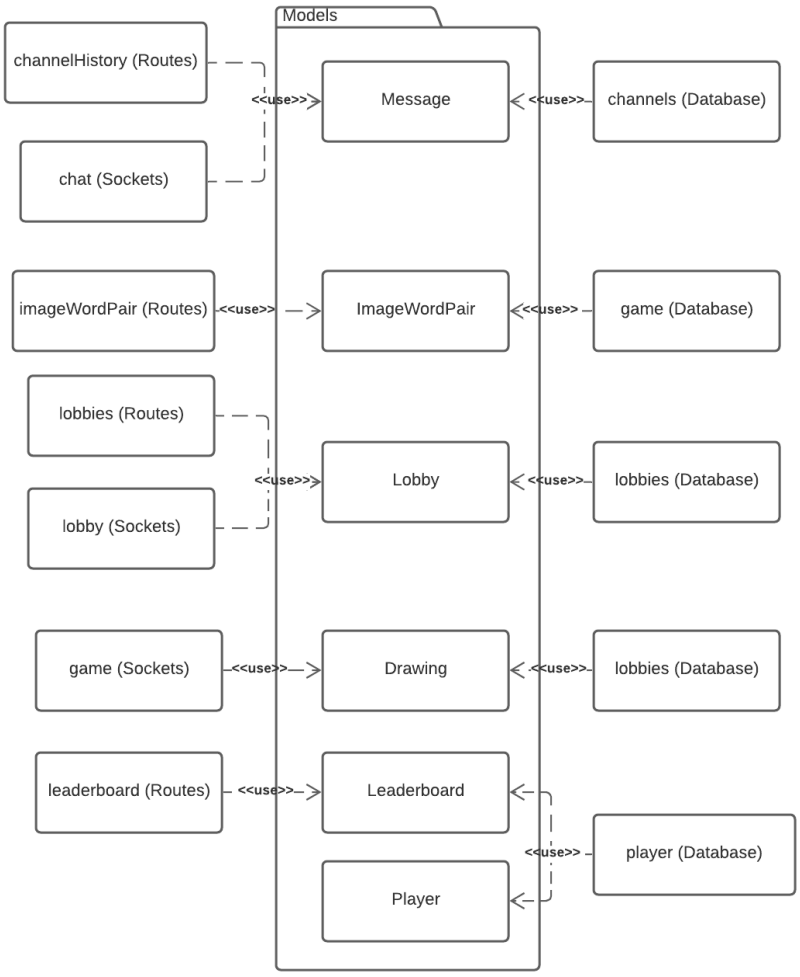

Figure 35: Diagramme de classe des models pour le serveur

Controllers
Ce paquetage contient la logique entre les données reçues d'un client et les données mises dans la base de données. Il est surtout utile dans les cas où un travail assez important est nécessaire, ce qui permet d'alléger le travail fait dans *Routes* (par exemple la conversion d'une image bitmap à un dessin svg).

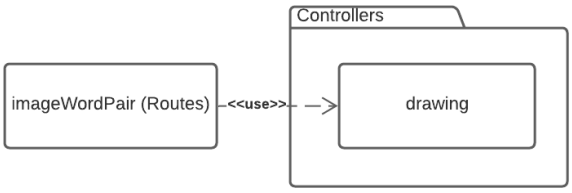

Figure 36: Diagramme de classe des controllers pour le serveur

Config
Ce paquetage contient les informations nécessaires à la configuration du serveur. Il contient aussi la configuration du service d'authentification Passport.js

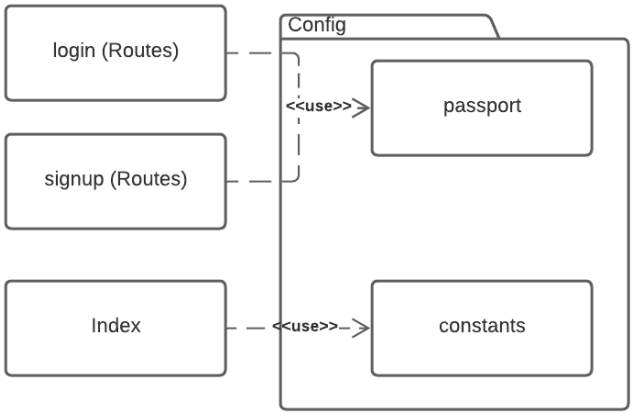

Figure 37: Diagramme de classe de la config pour le serveur

Database
Ce paquetage contient tous les appels à la base de données.

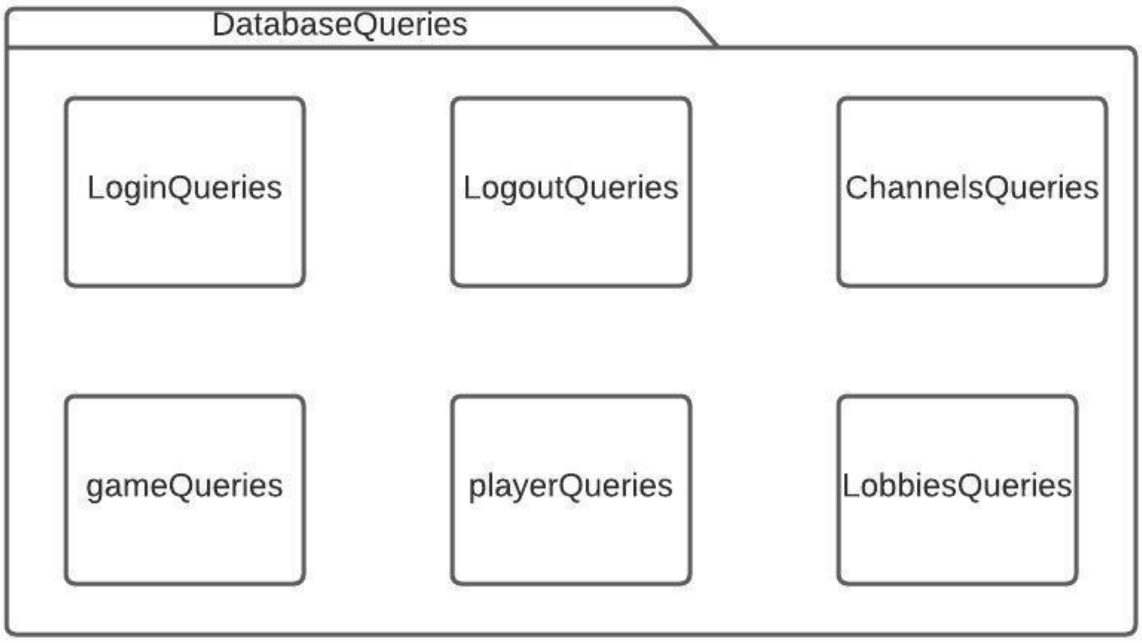

Figure 38: Diagramme de classe de la database pour le serveur

5. Vue des processus

Chaque client a notamment des composants et des services. L'utilisateur interagit avec l'interface des composants qui font des appels aux services. Par la suite, ces services communiquent avec le serveur via des requêtes HTTP ou l'utilisation de socket. Si nécessaire, le serveur peut communiquer avec la base de données par des requêtes SQL. Les données sont ensuite retournées au client.

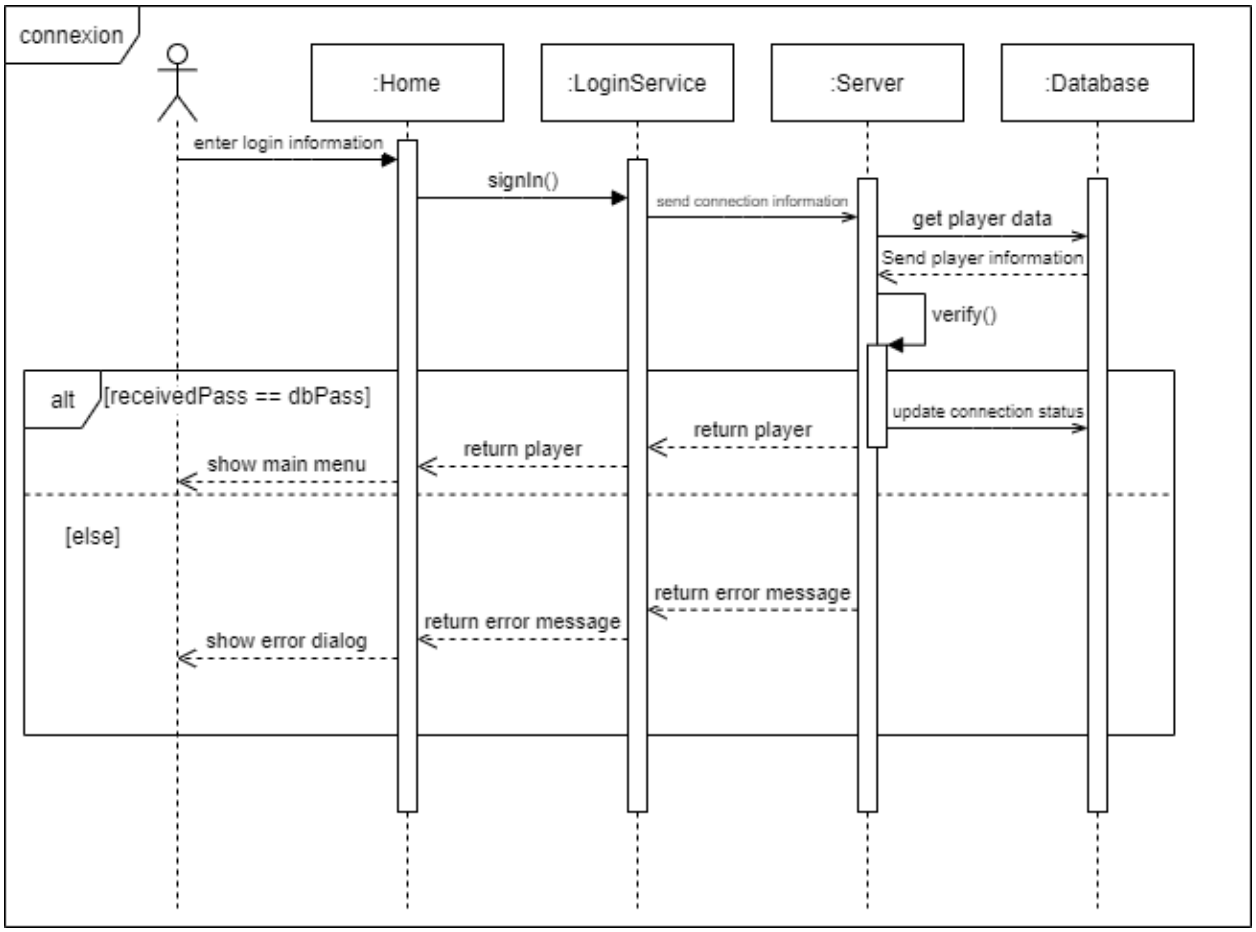

Figure 39: Diagramme de séquence de la connexion pour le serveur

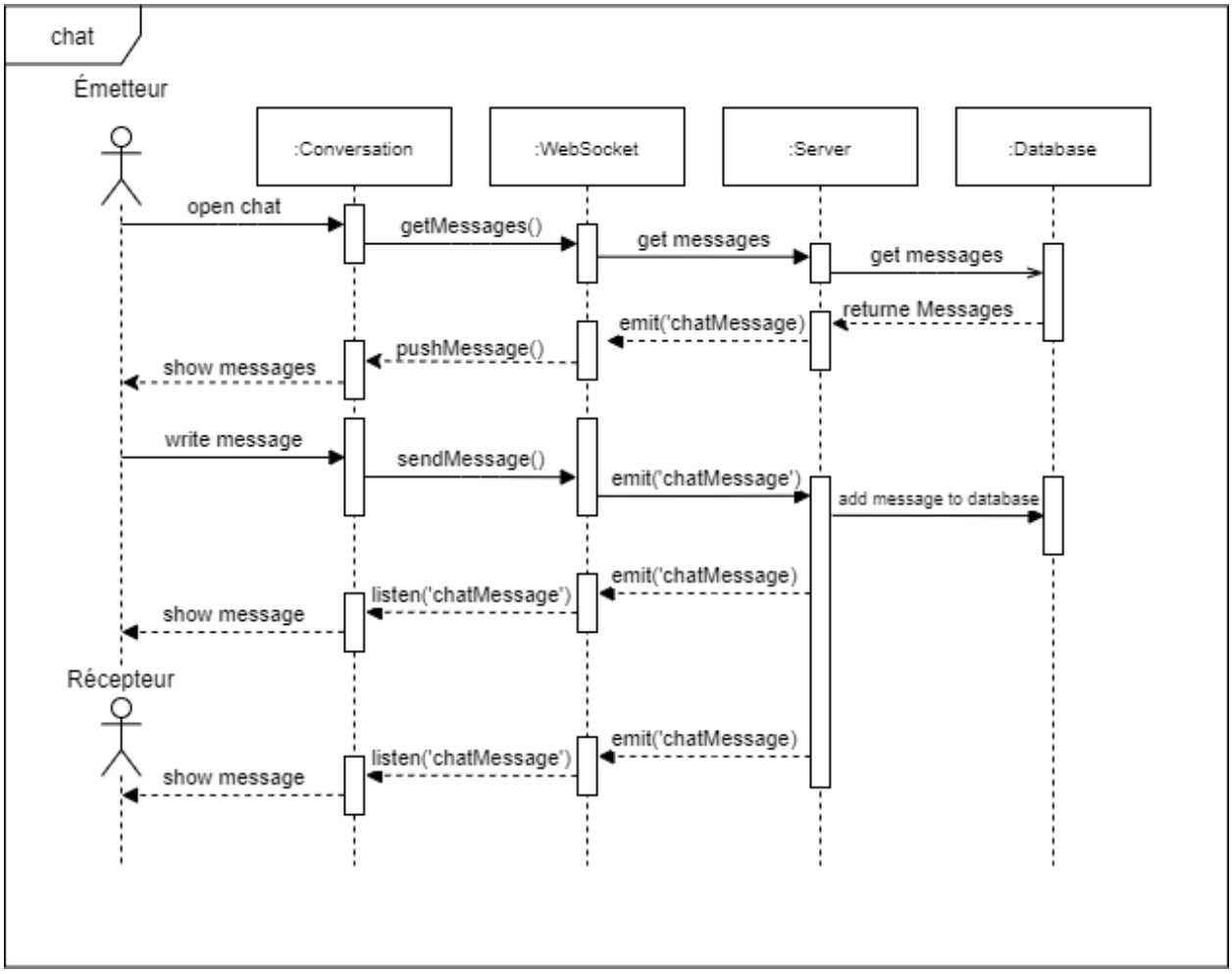

Figure 40: Diagramme de séquence du chat pour le serveur

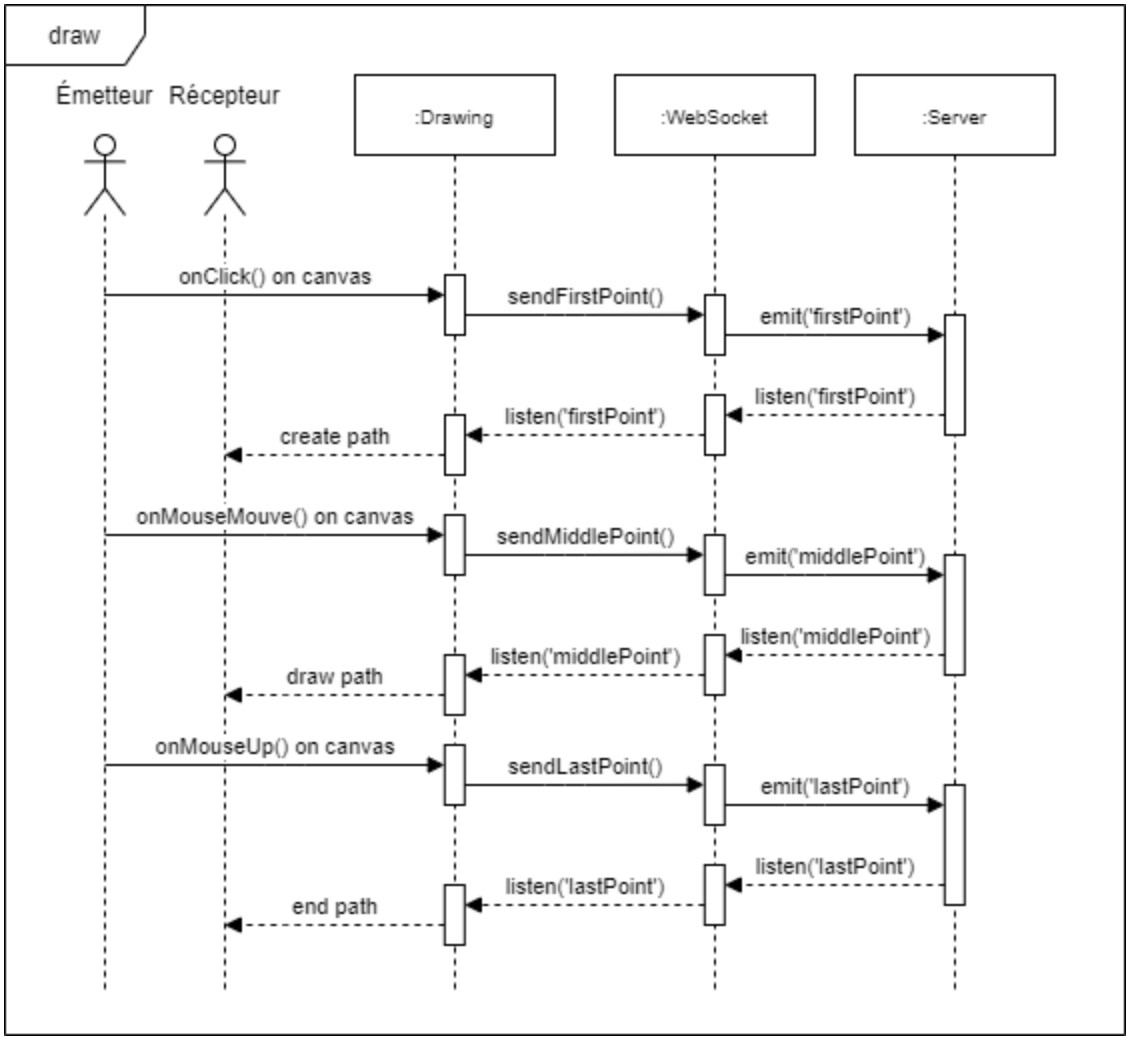

Figure 41: Diagramme de séquence de draw pour le serveur

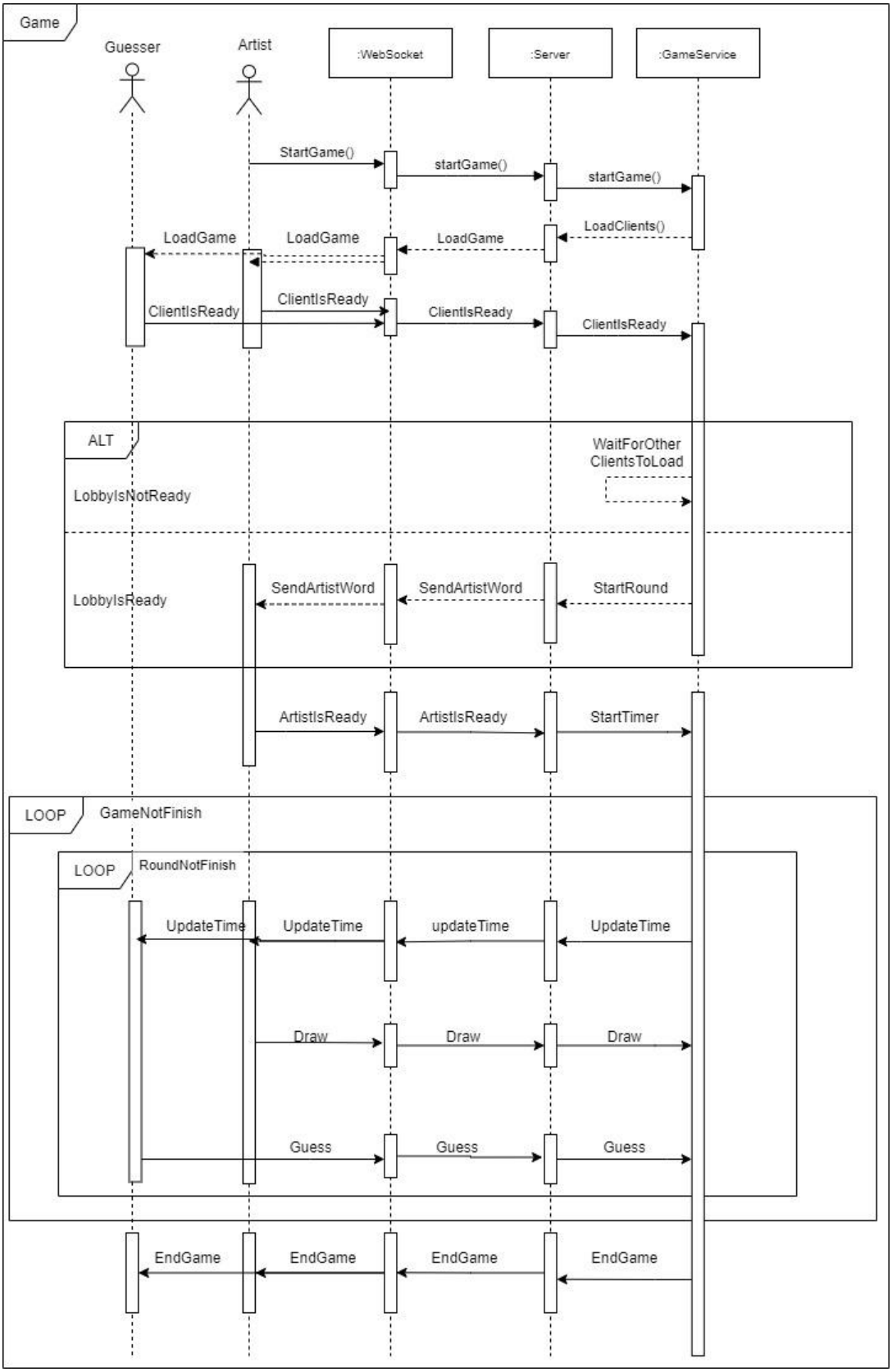

Figure 42: Diagramme de séquence du déroulement d'une partie

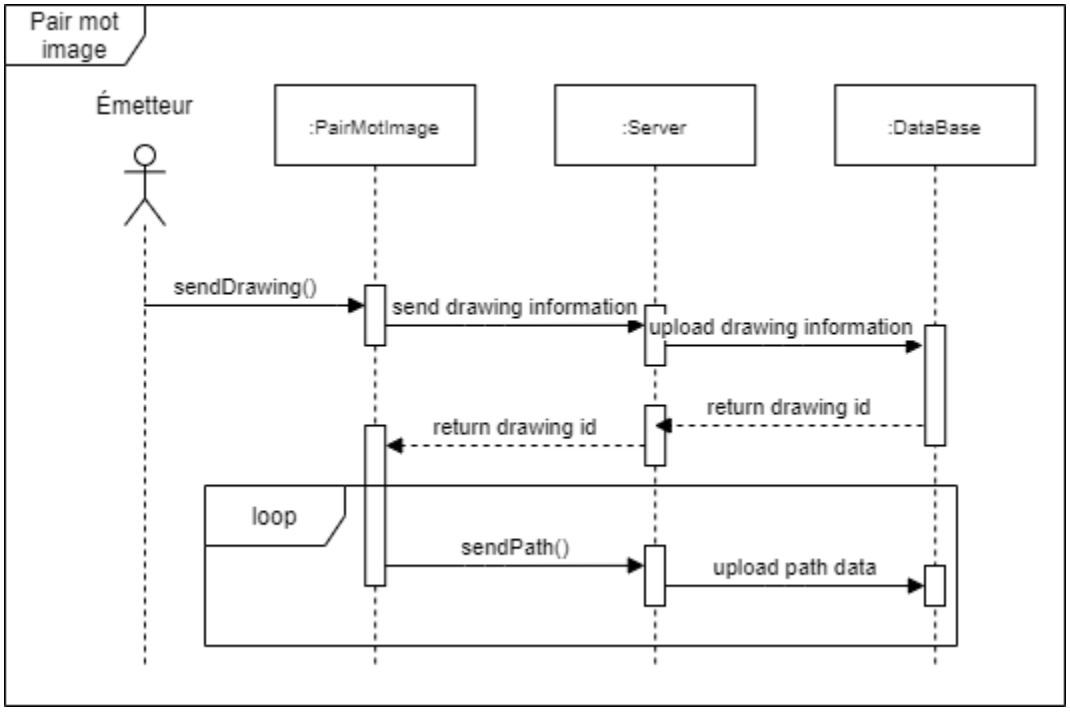

Figure 43: Diagramme de séquence démontrant le déroulement d'envoie d'un dessin de pair mot image

6. Vue de déploiement

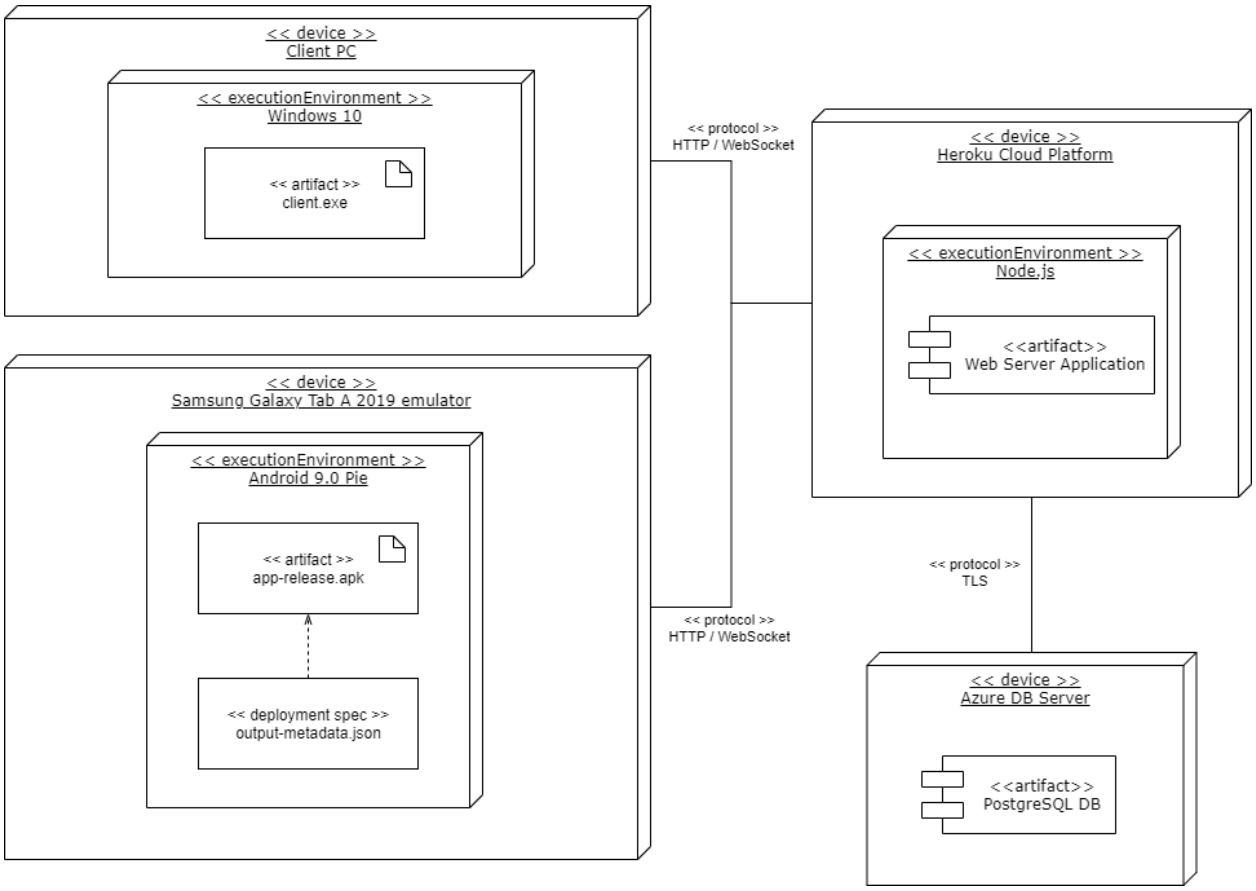

Figure 44: Diagramme de déploiement pour fais moi un dessin

7. Taille et performance

L'exécutable du logiciel ne doit pas être trop lourd. Nous viserons une taille de 300 MB maximum pour notre client lourd.

Le client lourd doit être en mesure d'offrir une expérience avec une fluidité d'affichage pour l'utilisateur. L'utilisateur doit être capable de s'authentifier dans un temps acceptable. Le délai des requêtes doit être rapide avec une latence inférieure à 500 ms. La connexion à un lobby, l'envoi et la réception des messages devraient suivre la même logique. Pour ce qui est du dessin, l'affichage de ce dernier devrait s'exécuter avec une latence d'affichage minimale pour l'utilisateur avec une latence inférieure à 100 ms. L'utilisateur devrait être capable de clavarder en voyant le dessin s'afficher avec une fluidité. Le client lourd utilise 100 MB de RAM.

Le client léger doit offrir une expérience similaire au client lourd en termes de fluidité d'affichage des dessins et des délais des requêtes. Le client léger a une taille de 134 MB et utilise 156 MB de RAM lorsqu'il roule dans l'émulateur d'Android Studio.

Pour ce faire, il faut s'assurer que la communication au serveur est suffisamment rapide et fiable. Mettre notre serveur sur Heroku nous garantit une bonne disponibilité du système et une performance plus que suffisante pour nos besoins. De plus, déployer la base de données sur un serveur Azure permet d'assurer une bonne vitesse de communication entre l'application serveur et la base de données. Pour la communication entre les clients et le serveur, le protocole WebSocket est utilisé pour les requêtes nécessitant des mises à jour en temps réel (tel que le dessin pendant les parties et les canaux de communication) puisqu'il est très performant dans ces situations. La taille du serveur est de 59 MB avec les nodes modules et utilise 10 MB de RAM .

8. Vue base de donnée

Le diagramme UML de notre base de données représente l'architecture des tables créées. Nous utilisons une base de données Postgres sur un serveur Azure.

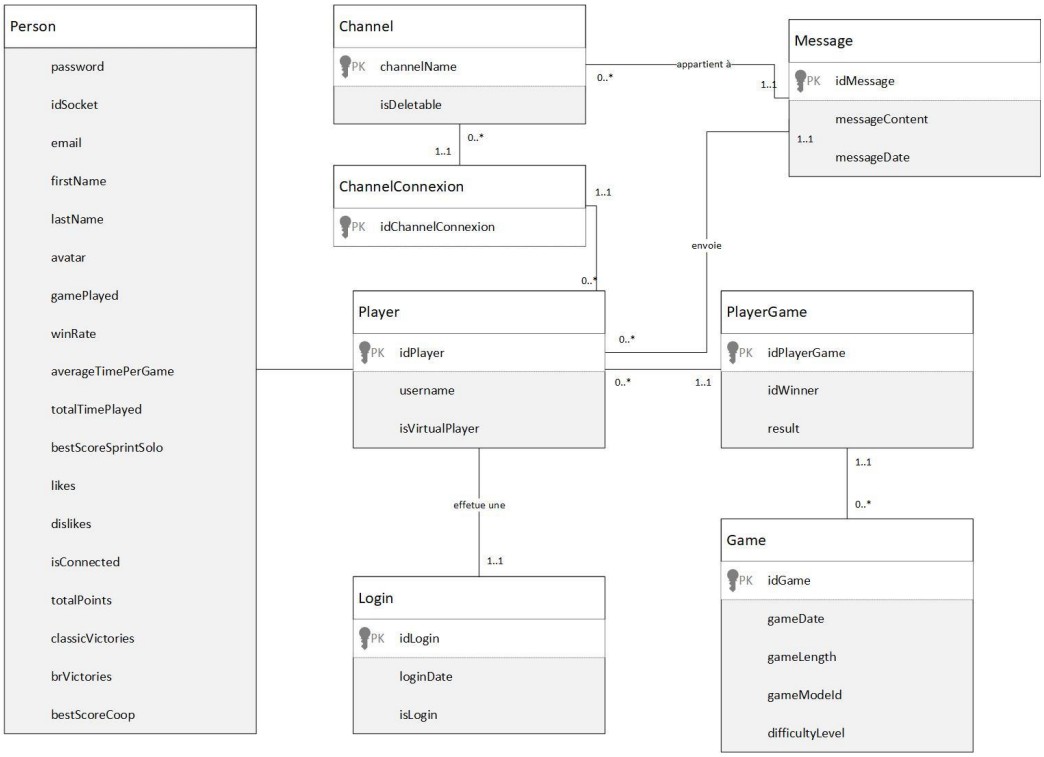

Figure 45: Diagramme entité relation joueurs et parties

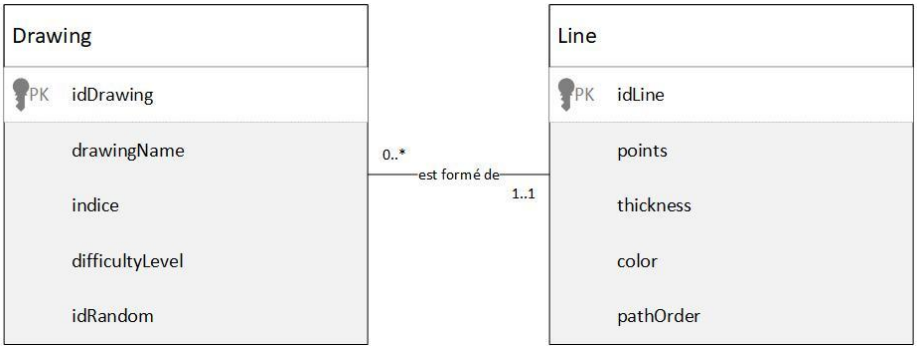

Figure 46: Diagramme entité relation paire mot-image