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

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:
L'application doit toujours cacher les mots de passe par défaut.

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

Portabilité:
L'application doit fonctionner aussi bien sur Windows 10 (PC) que sur une tablette Samsung Galaxy Tab A 2019 avec Android. La tablette n'offre pas la même capacité de performance que l'ordinateur, alors nous devons nous assurer que l'application fonctionne correctement dans ces deux environnements d'exécution.

Réutilisation:
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:
Le projet sera divisé en sprints de 2 semaines. Au moment la réponse à l'appel d'offres, le deuxième sprint sera déjà terminé.

Sprint	Exigences/Artéfacts	Temps	Date
1	• SRS • Liste d'exigences • Nettoyage du code Angular • Migration vers Electron • Recherches sur les architectures possibles • Création de l'application Android • Création de la base de données • Création du serveur	32 heures 32 heures 24 heures 8 heures 32 heures 4 heures 8 heures 8 heures	2 février 2021
2	• Architecture du logiciel • Plan de projet • Protocole de communication • Gestion des connexions • Clavardage - communication entre deux utilisateurs	16 heures 16 heures 16 heures 48 heures 48 heures	16 février 2021
Remise	Réponse à l'appel d'offres		19 février 2021

3	• Clavardage - Intégration • Clavardage - Canaux de discussion • Clavardage - Censure • Profil utilisateur et historique • Outils de dessin Android • Personnalité des joueurs virtuels • Création d'une paire mot-image - Assistée 1	4 heures 40 heures 8 heures 16 heures 16 heures 24 heures 16 heures	2 mars 2021
4	• Rédaction du plan de tests logiciel • Tutoriel - non interactif • Construction d'un jeu - Indices • Construction d'un jeu - Classique • Leaderboard • Pouces	16 heures 8 heures 4 heures 48 heures 16 heures 16 heures	16 mars 2021
5	• Effets visuels et sonores • Construction d'un jeu - Sprint Solo • Construction d'un jeu - Sprint Coop • Création d'une paire mot-image - Assistée 2	4 heures 32 heures 16 heures 24 heures	30 mars 2021
6	• Construction d'un jeu - Battle Royal • Révision des artéfacts • Rédaction des résultats de tests	32 heures 32 heures 16 heures	13 avril 2021
Remise	• Révision du projet final	32 heures	19 avril 2021
Total		712 heures	

Chaque sprint implique 4 heures de rencontre par personne, donc 24 heures-personne
Chaque sprint implique 2 heures de gestion de projet par le gestionnaire
Légende:

- Exigences essentielles
- Exigences souhaitables (sujet à changement) (voir ci-dessous pour les choix)

Exigences souhaitables	Temps
• Tutoriel interactif • Construction d'un jeu - Mode spectateur • Connexion Google • Liste d'amis • Authentification en deux étapes	16 heures 8 heures 8 heures 32 heures 16 heures

Coûts:

L'équipe se base sur un budget ventilé en fonction des lots de travail et des taux suivants :

- développeur : 100$/h ;
- gestionnaire de projet : 125$/h.

Le gestionnaire rédige tous les documents, il s'occupe des planifications et il fait partie des rencontres bi-hebdomadaires de l'équipe.
Gestionnaire: 176 heures-personne de rédaction + 36 heures-personne de gestion = 212 heures-personne
Développeurs: 536 heures personne de développement + 120 heures-personne de rencontre = 656 heures-personne
Les membres de l'équipe ont tous déjà le matériel nécessaire pour avancer le projet.
La base de données azure: 67$

Selon les estimations préparées dans l'échéancier du projet, le total est estimé à 92 167$ + tx.

Outils de développement:

Le projet sera développé en parallèle par 6 personnes. Pour s'assurer d'éviter d'avoir des conflits, l'équipe développera 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:

Le client-lourd (Electron) sera développé avec Typescript, HTML, CSS, javascript pour la framework Angular.
Le client léger sera développé avec Kotlin.
Le serveur en TypeScript, javascript.
La base de données est développée en SQL.

3. Vue des cas d'utilisation

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

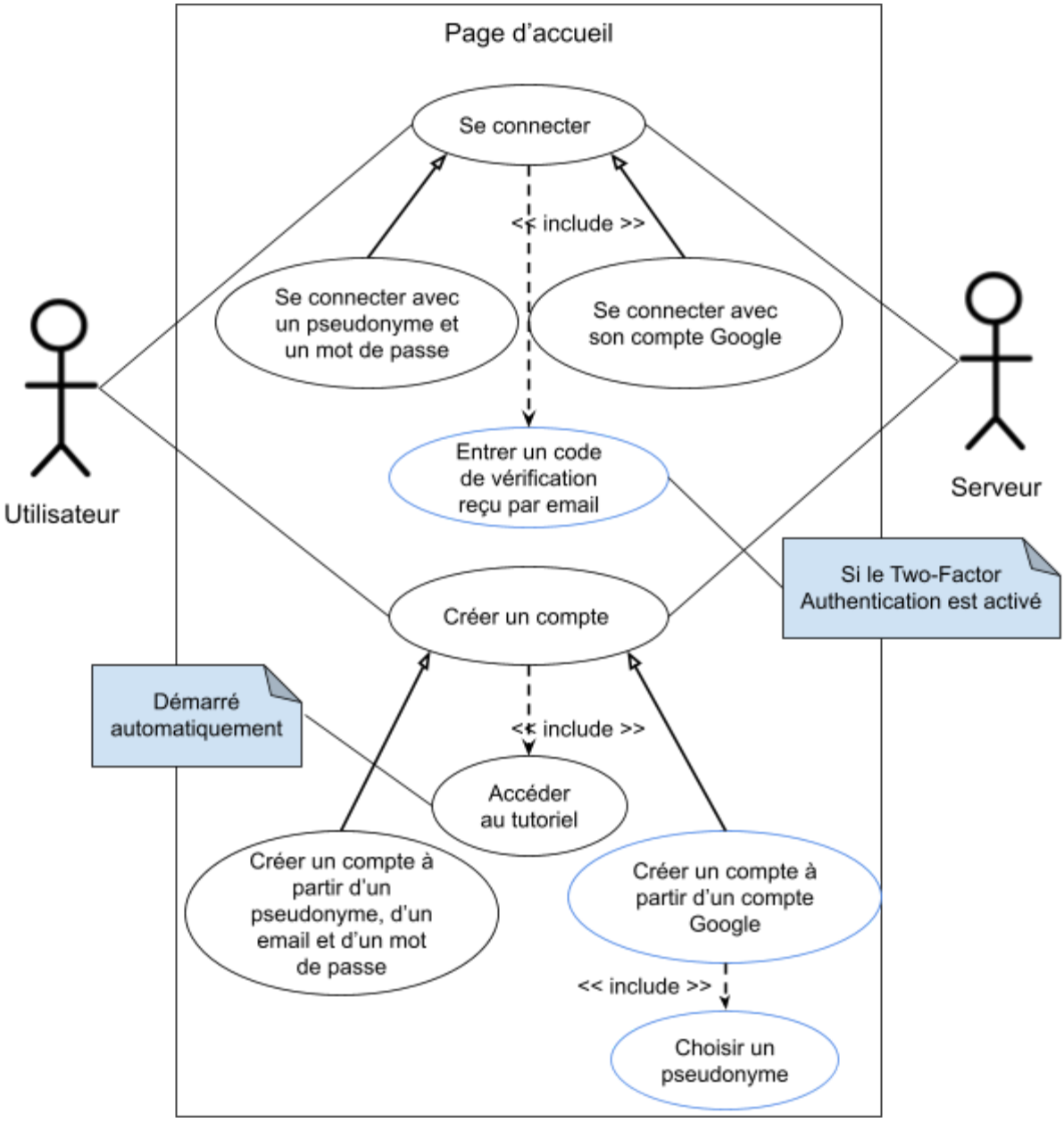


Figure 1: diagramme des cas d'utilisation de la page d'accueil

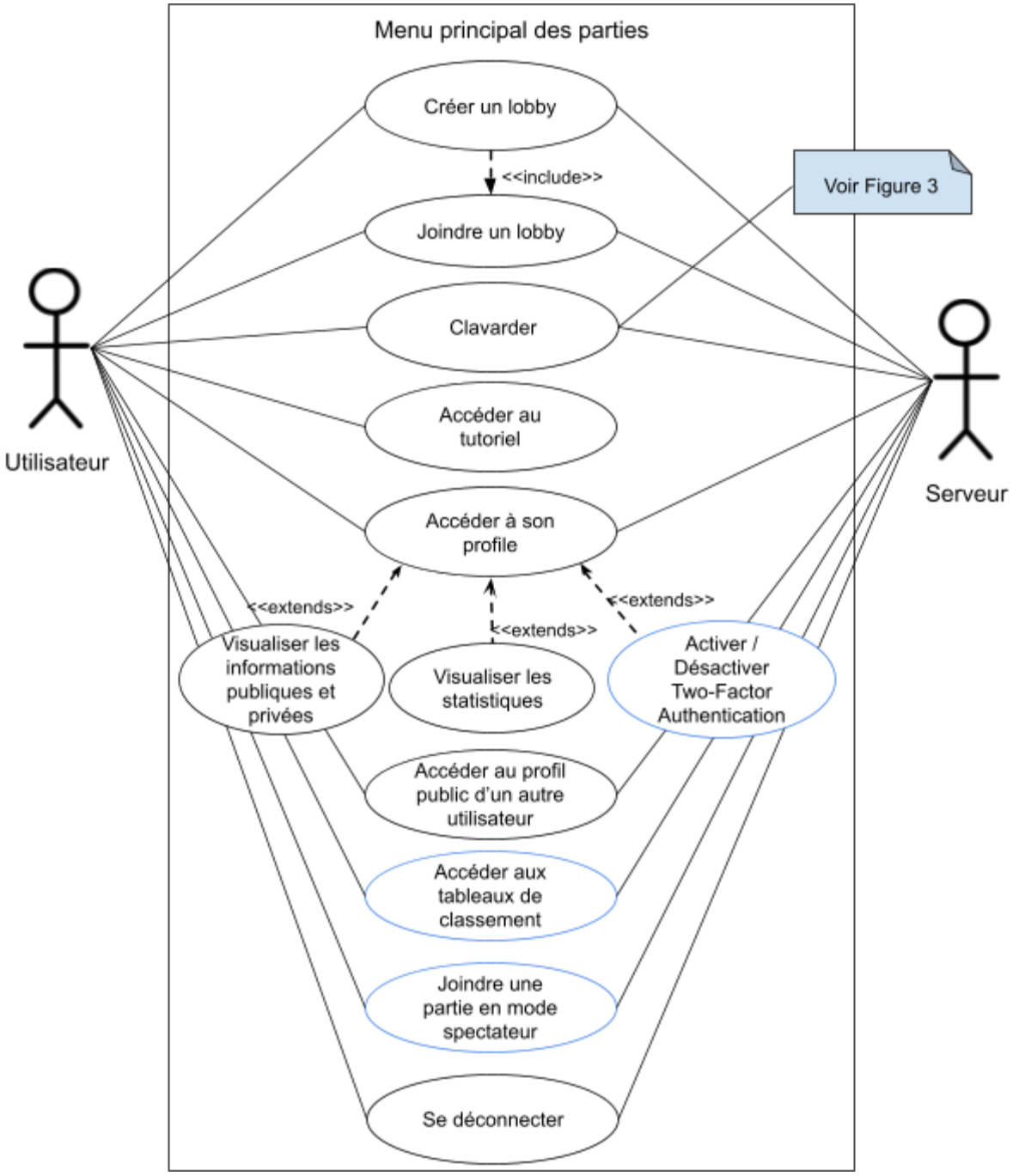


Figure 2: diagramme des cas d'utilisation du menu principal

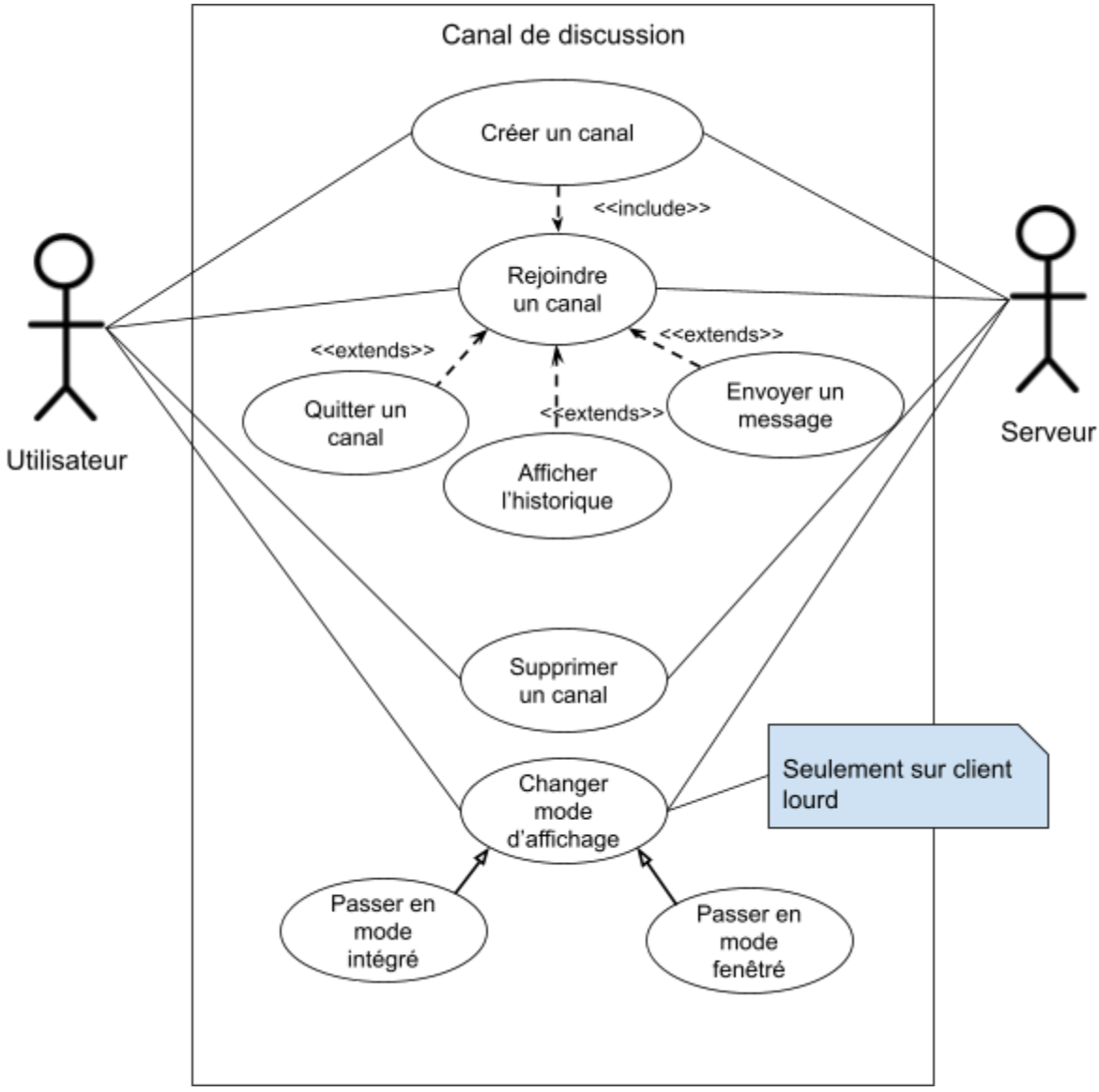


Figure 3: diagramme des cas d'utilisation des canaux de discussion

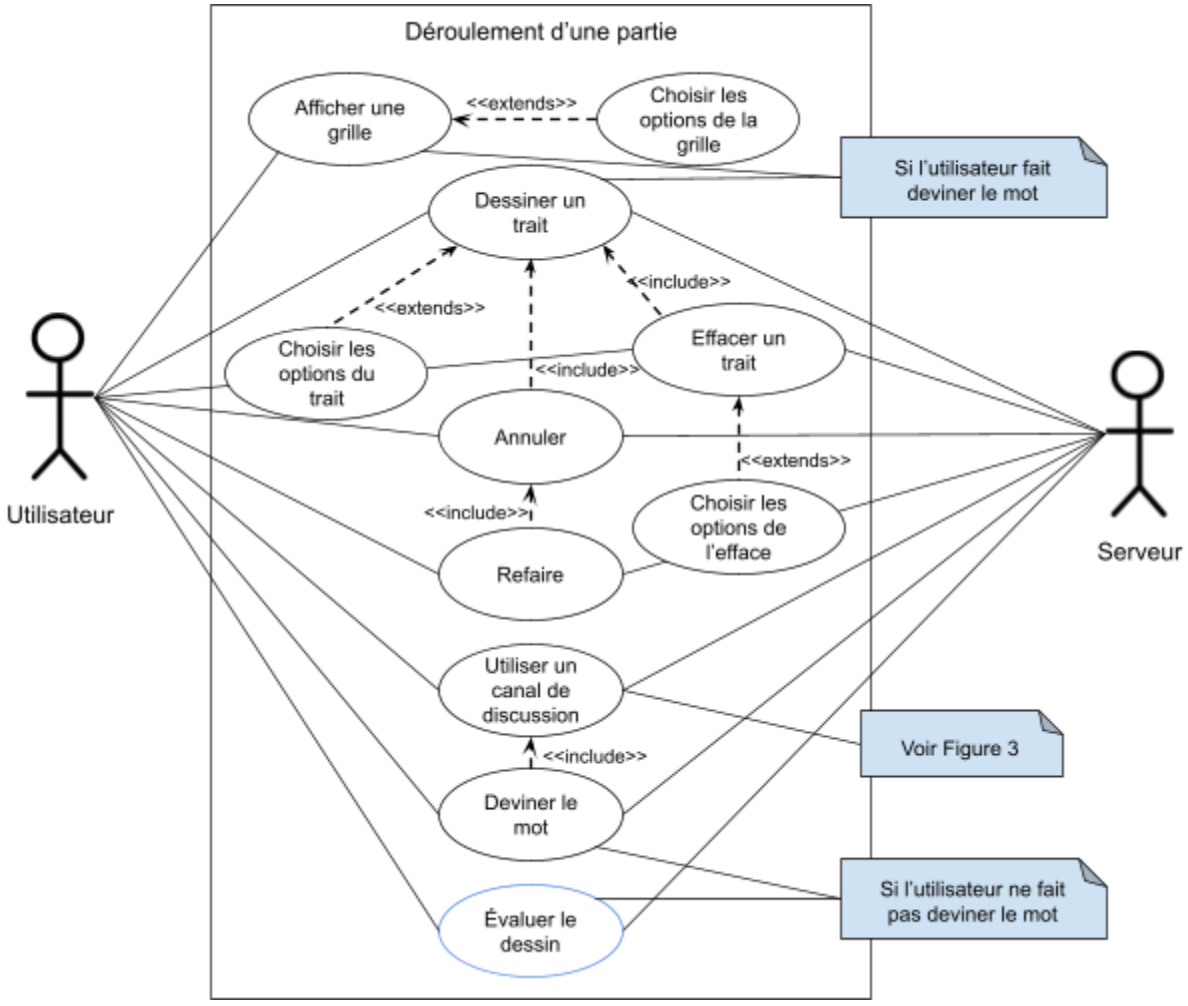


Figure 4: diagramme des cas d'utilisation lors du déroulement d'une partie

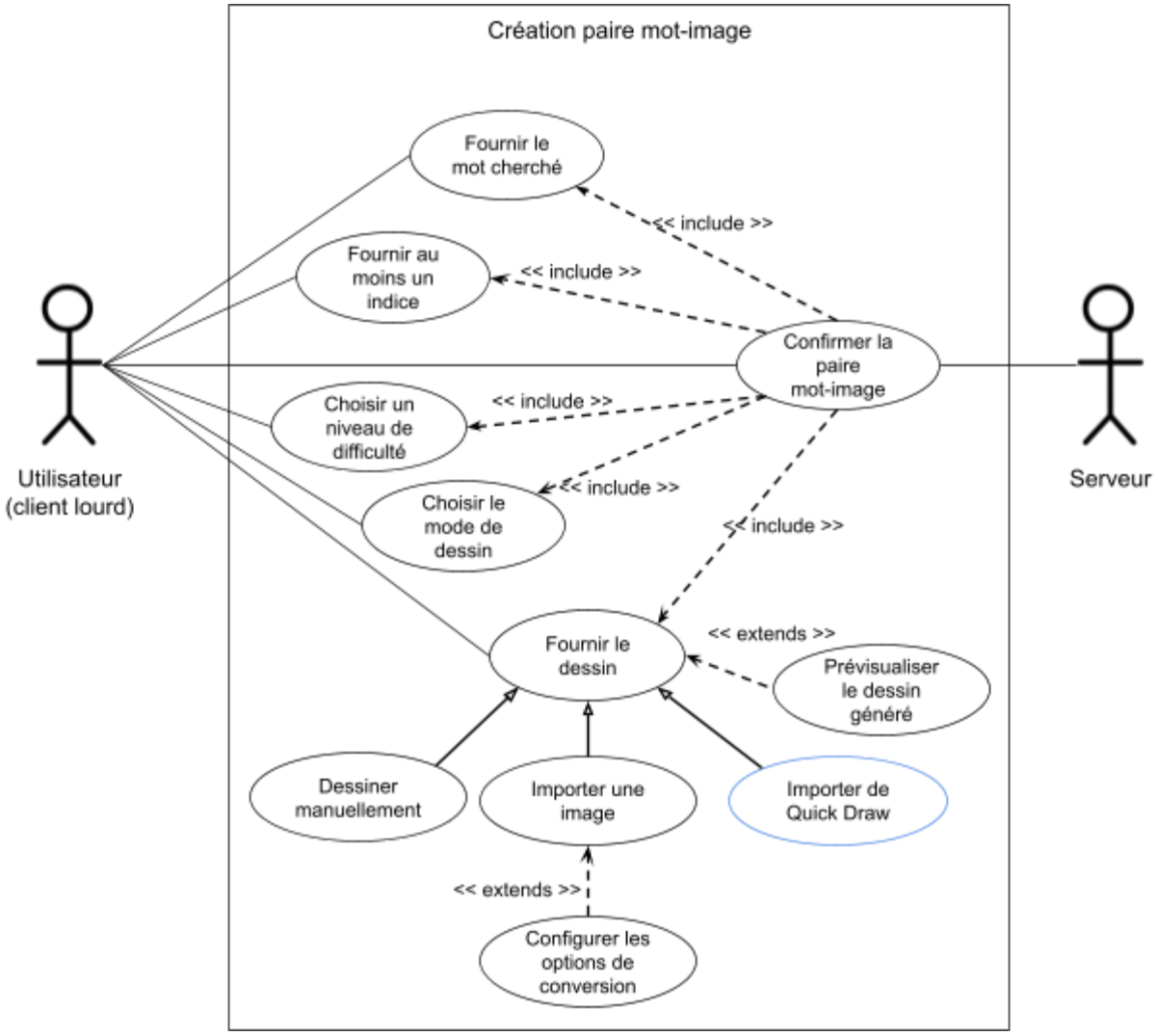


Figure 5: diagramme des cas d'utilisation de la création d'une paire mot-image

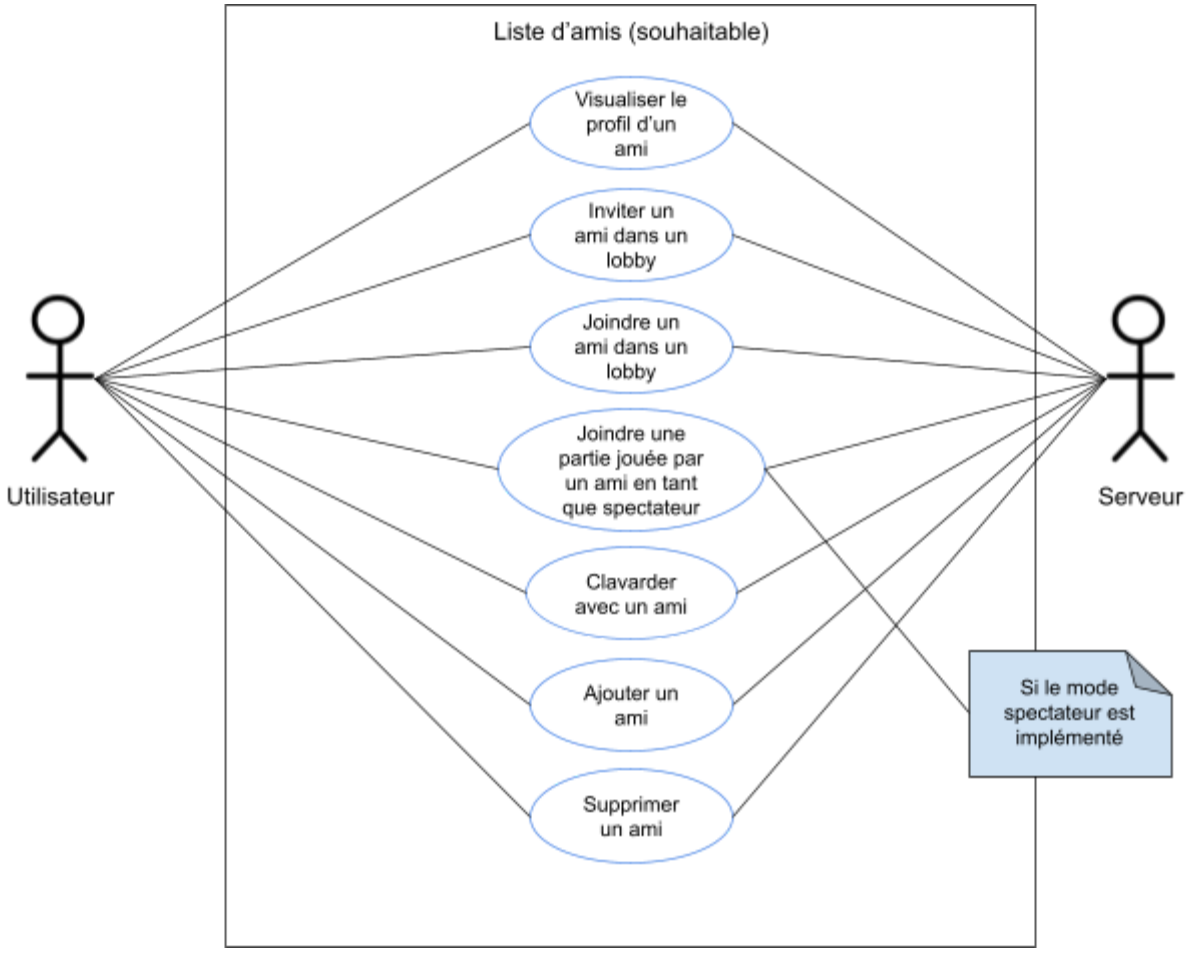


Figure 6: diagramme des cas d'utilisation de la liste d'amis

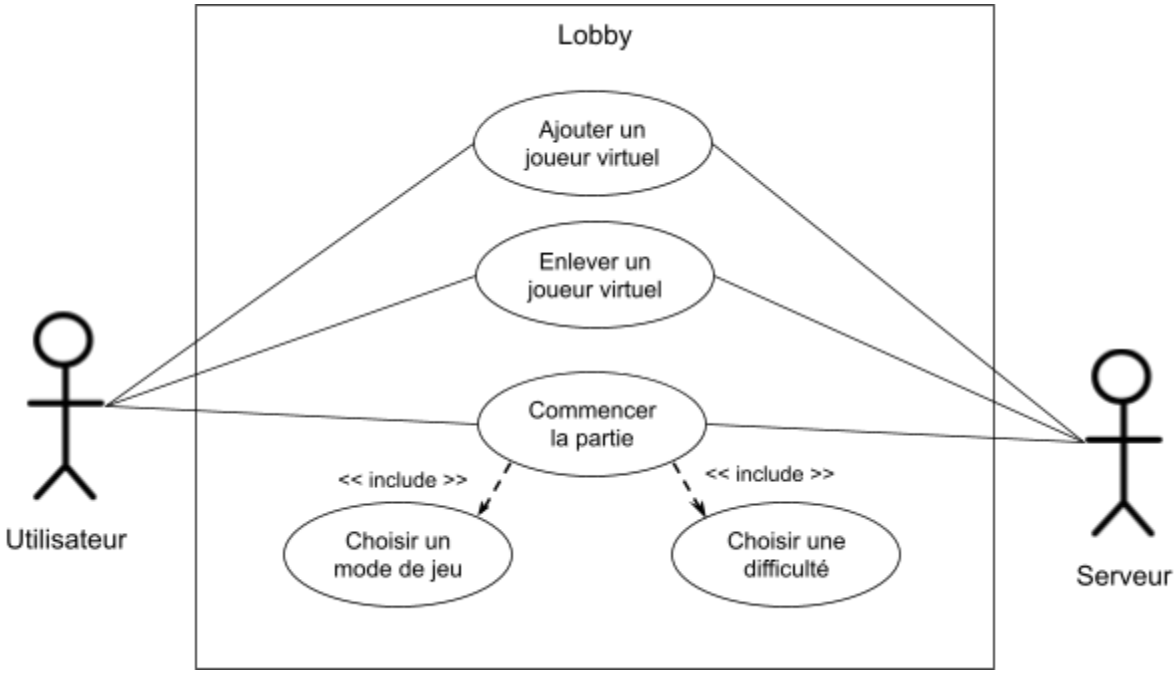
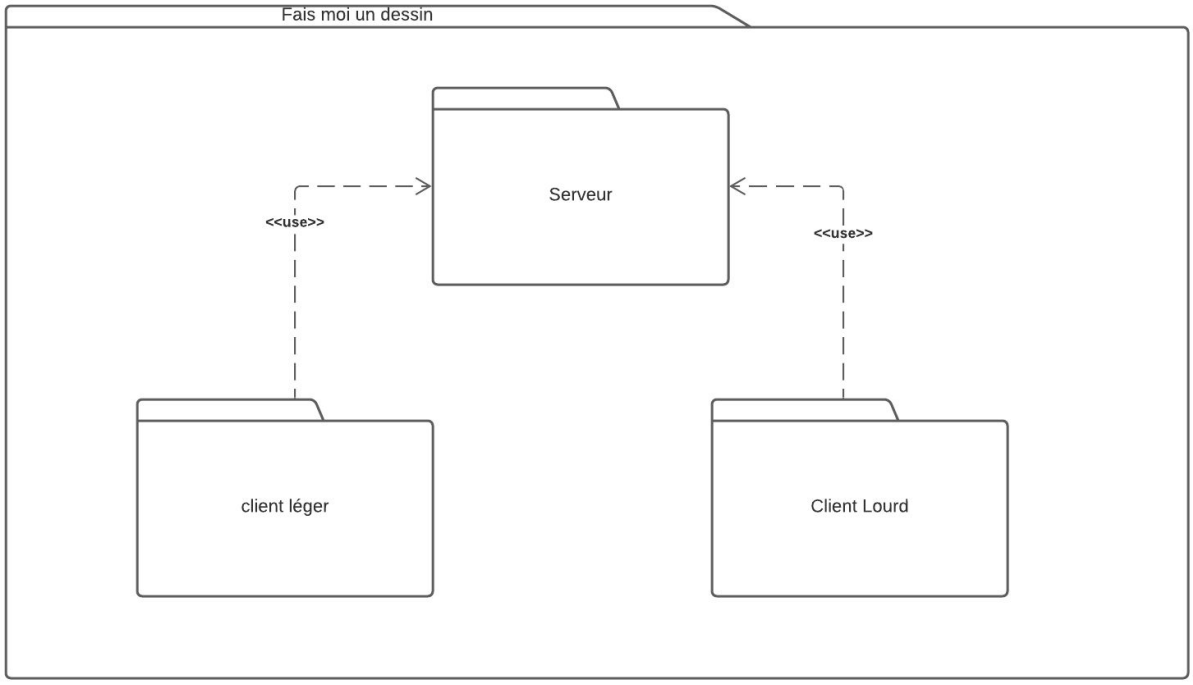


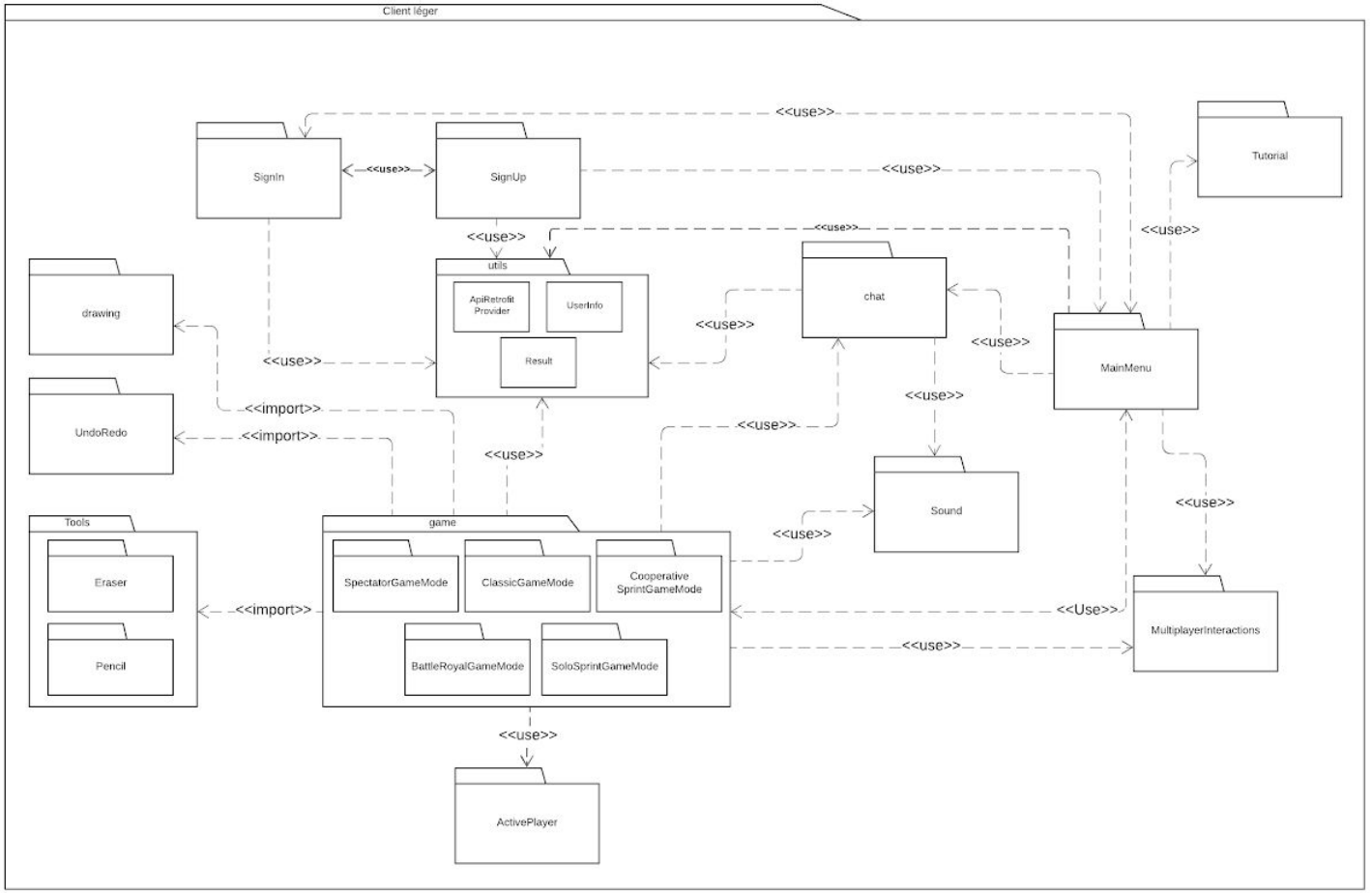
Figure 7: diagramme des cas d'utilisation dans un lobby

4. Vue logique

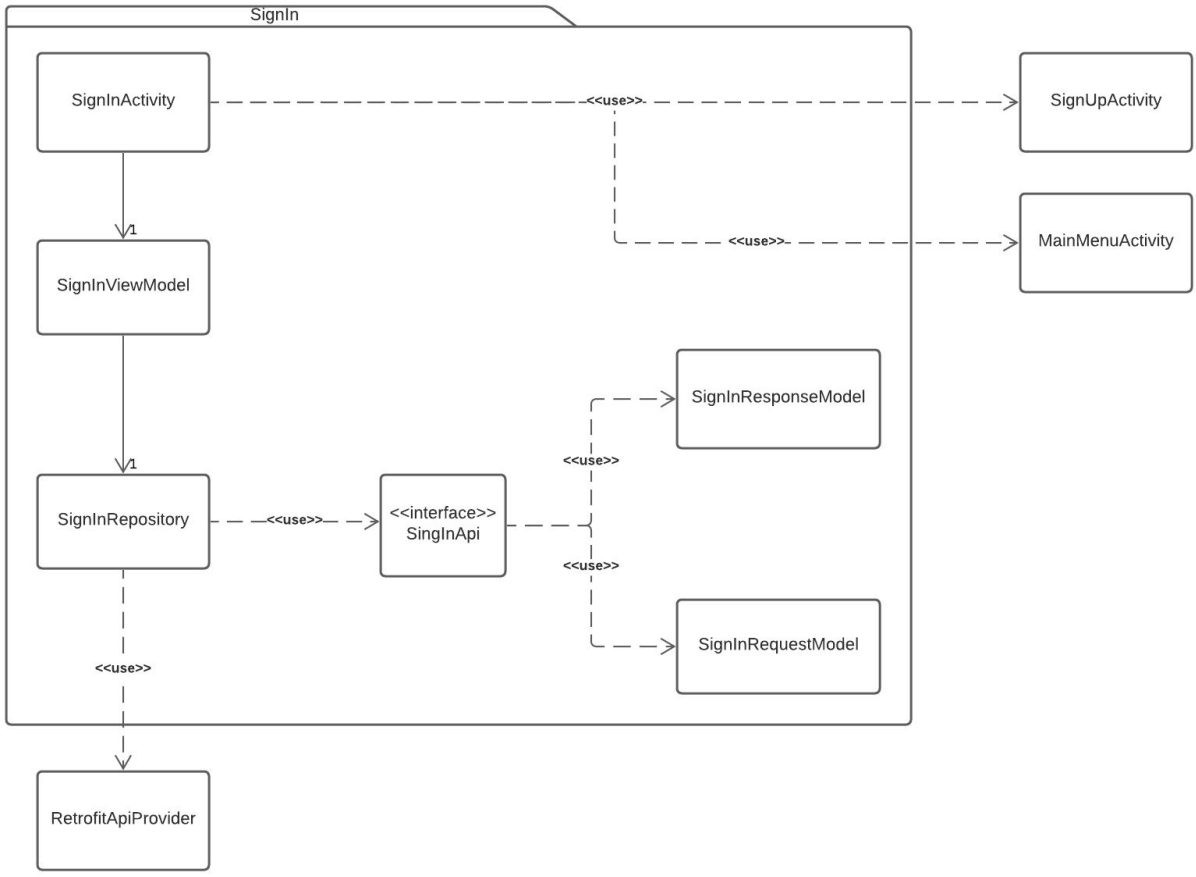
Fais-moi un dessin
Ce paquetage représente l'ensemble du projet. Il est le diagramme de paquetage du plus haut niveau. Il contient donc les 3 plus grands paquetages de notre projet qui sont le serveur, le client léger et le client lourd. Sa responsabilité principale est d'offrir aux utilisateurs plusieurs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



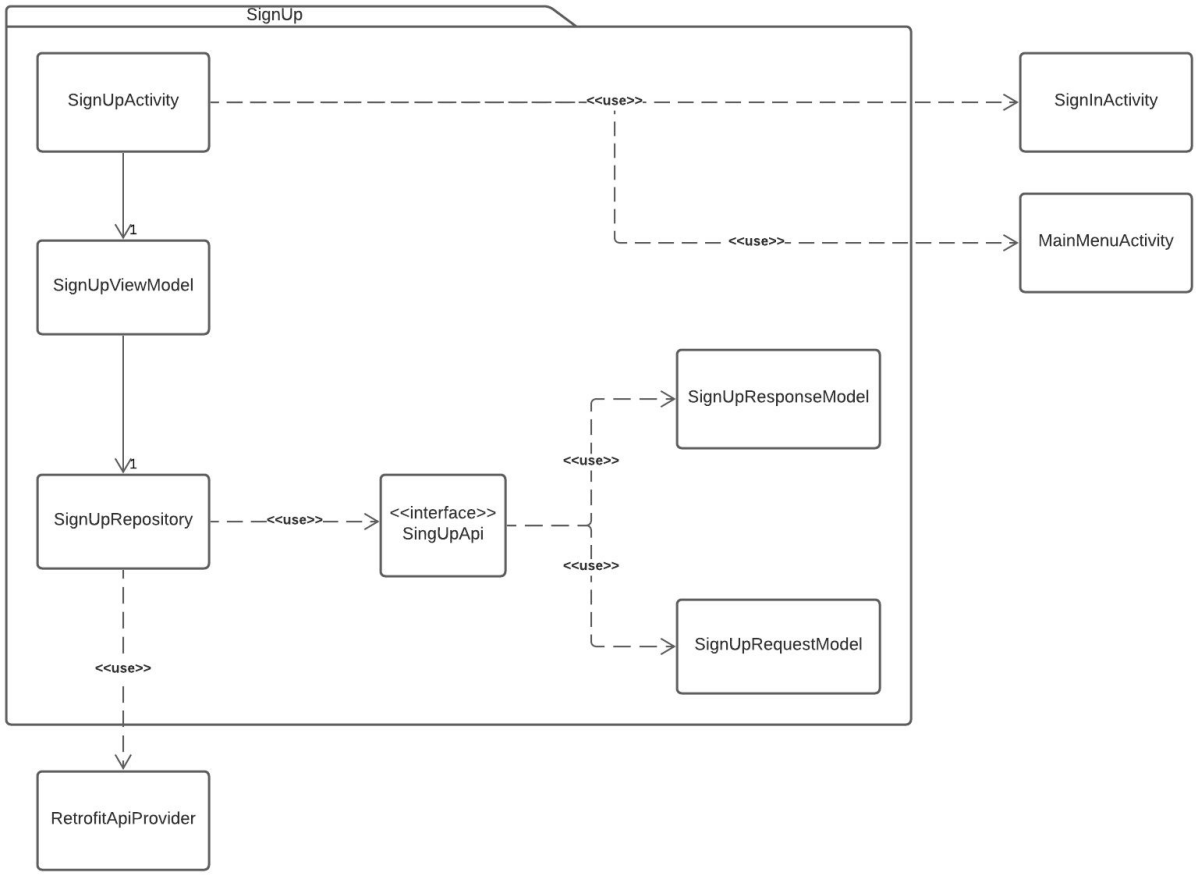
Client léger
Ce paquetage représente l'ensemble du client léger qui est l'application Android. Ce paquetage a beaucoup de responsabilités. En général, il doit permettre de rejoindre des lobbys, chatter avec d'autres joueurs et jouer aux différents modes de jeux en entrant en communication avec le serveur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



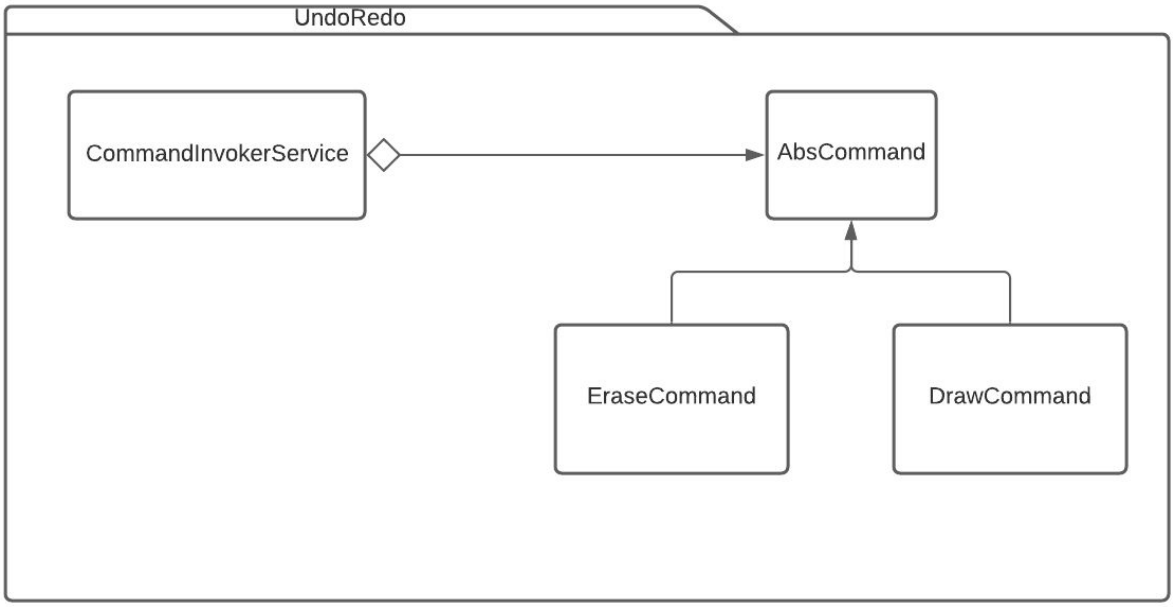
SignIn
Ce paquetage est le point d'entrée principal de l'application. En effet, son but principal est de permettre aux utilisateurs de se connecter en effectuant une requête POST au serveur. Il doit, par la suite, rediriger l'utilisateur.



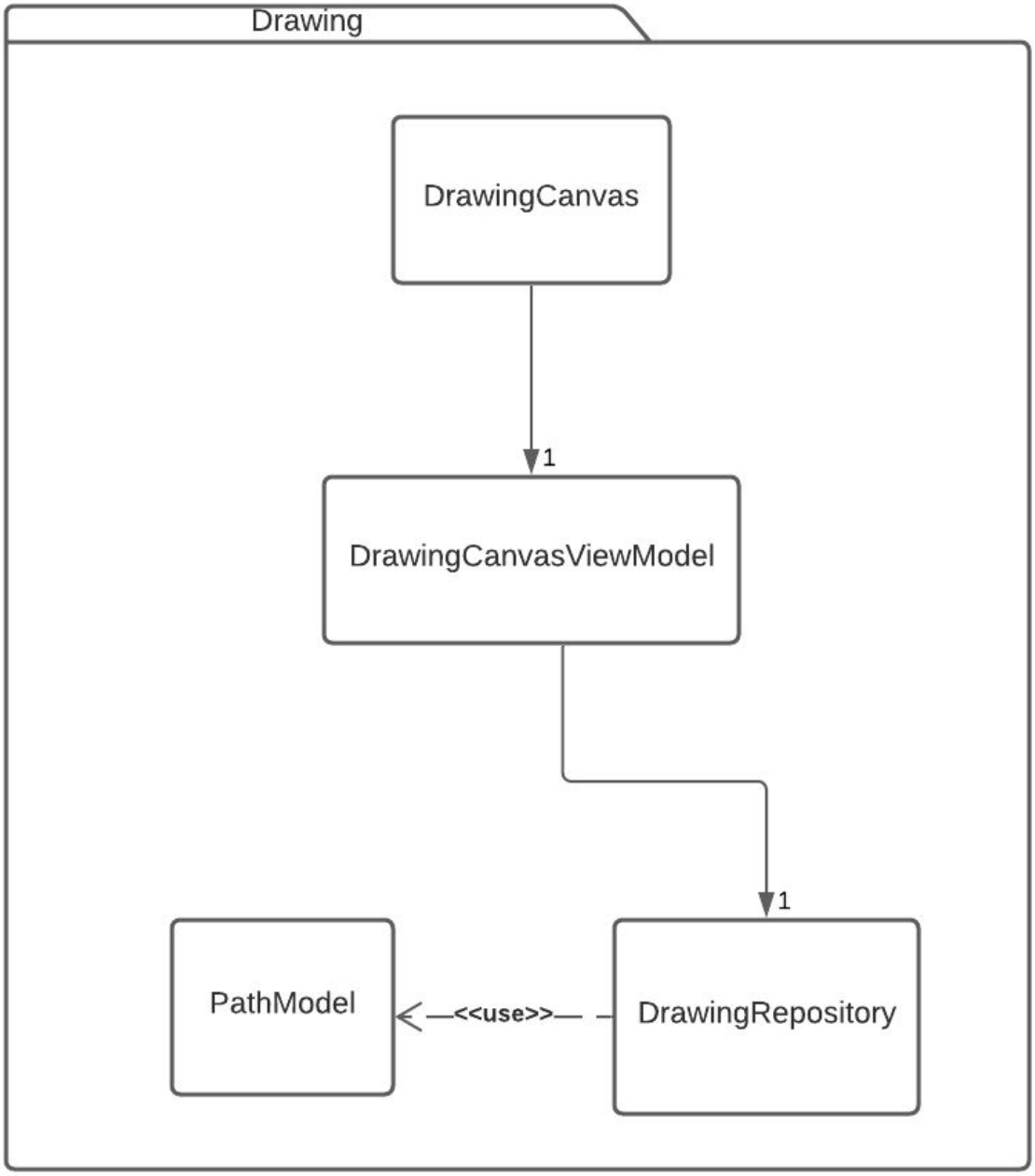
SignUp
Ce paquetage est un des points d'entrée pour accéder à l'application. Son but principal est de permettre aux utilisateurs de se créer un compte. Il doit aussi permettre à l'utilisateur de se rendre au menu principal de l'application s'il se crée un compte.



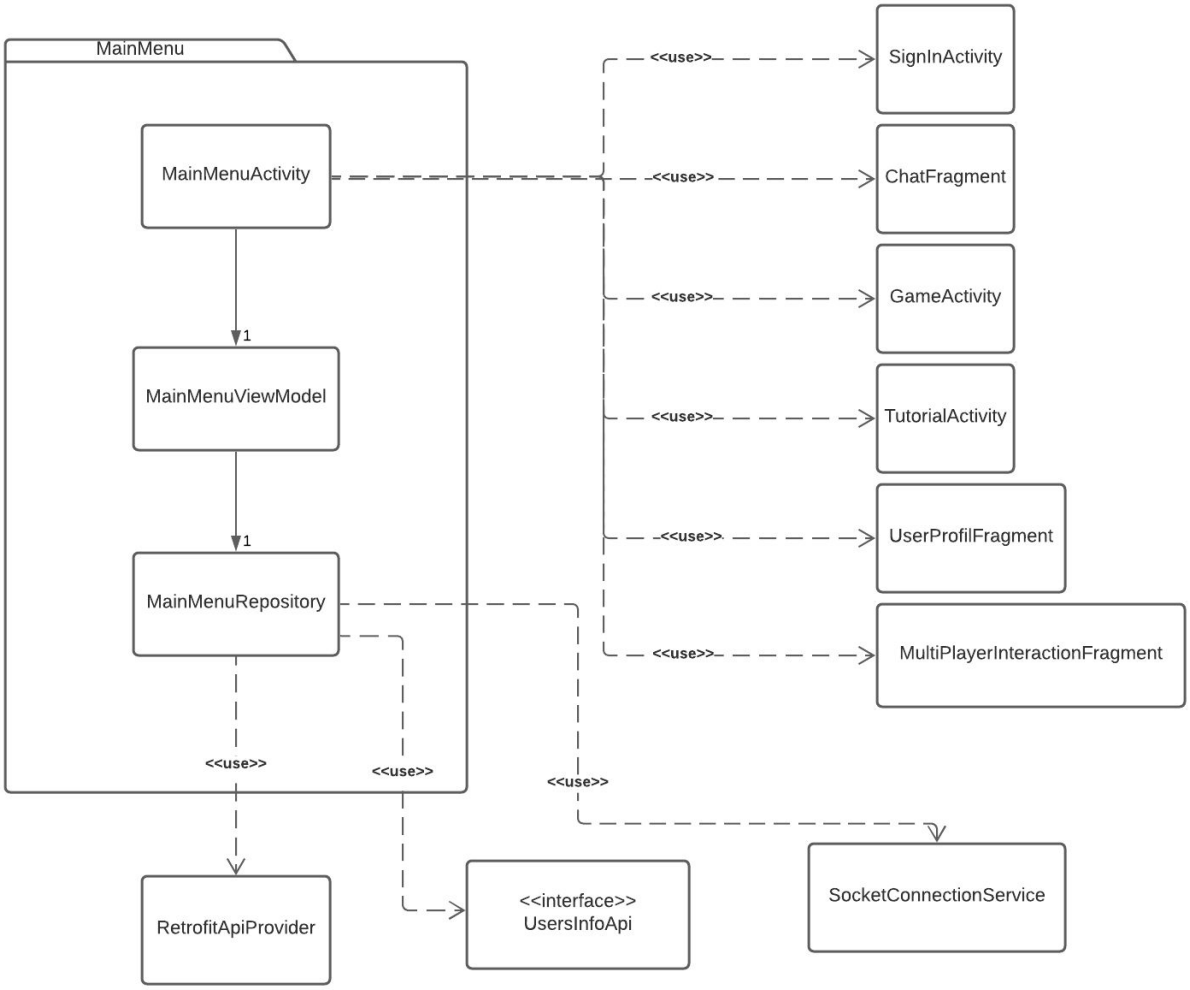
UndoRedo
Ce paquetage permet d'annuler et de refaire des actions posées par l'utilisateur sur son dessin. Son principal but est de stocker en format de commande les différentes actions de l'utilisateur sur son dessin et au besoin de les annuler ou les refaire. Il doit aussi transmettre ses commandes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



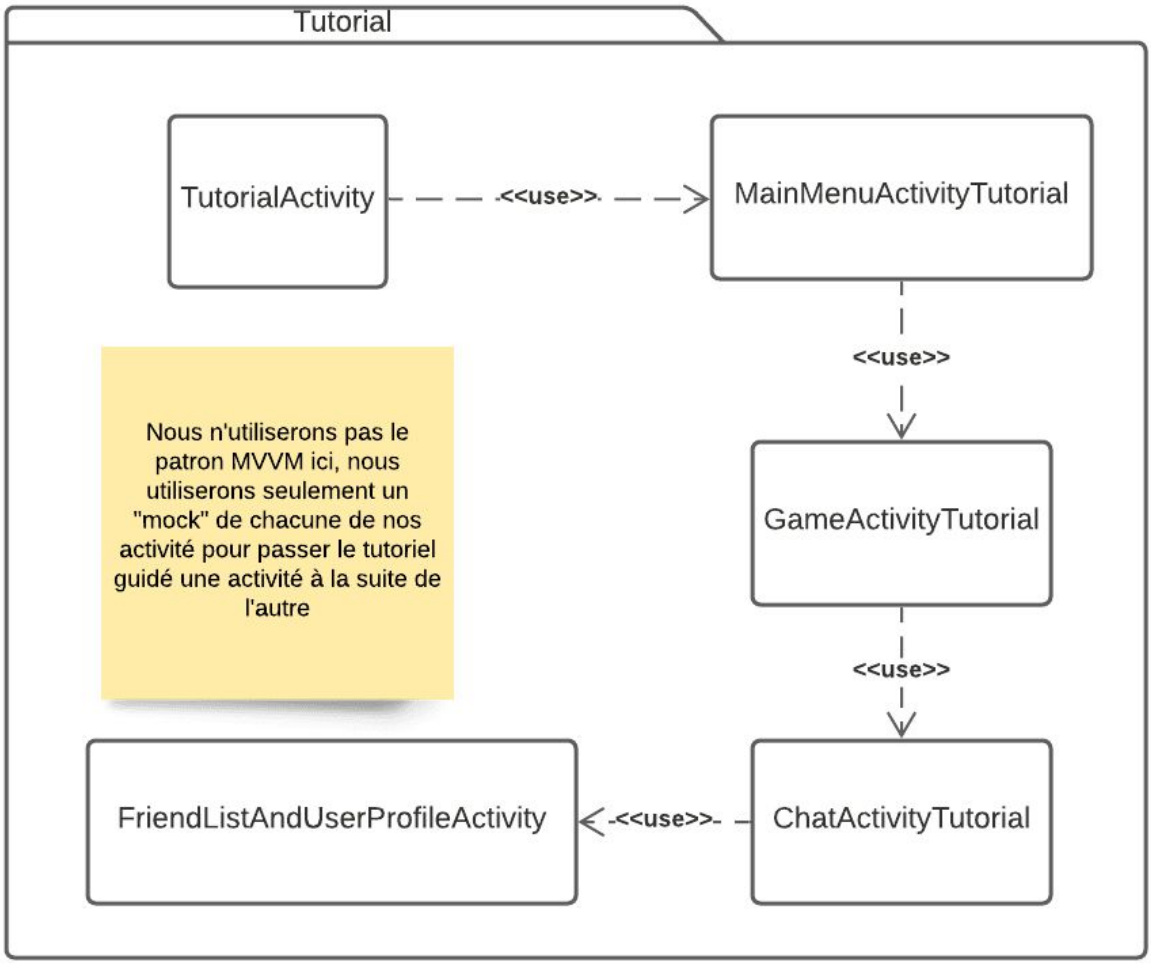
Drawing
Ce paquetage représente l'endroit ou un dessin sera effectué durant la partie. Sa responsabilité principale sera d'ajouter de permettre de dessiner des "path" et aussi d'en recevoir des autres joueurs lors d'une partie.



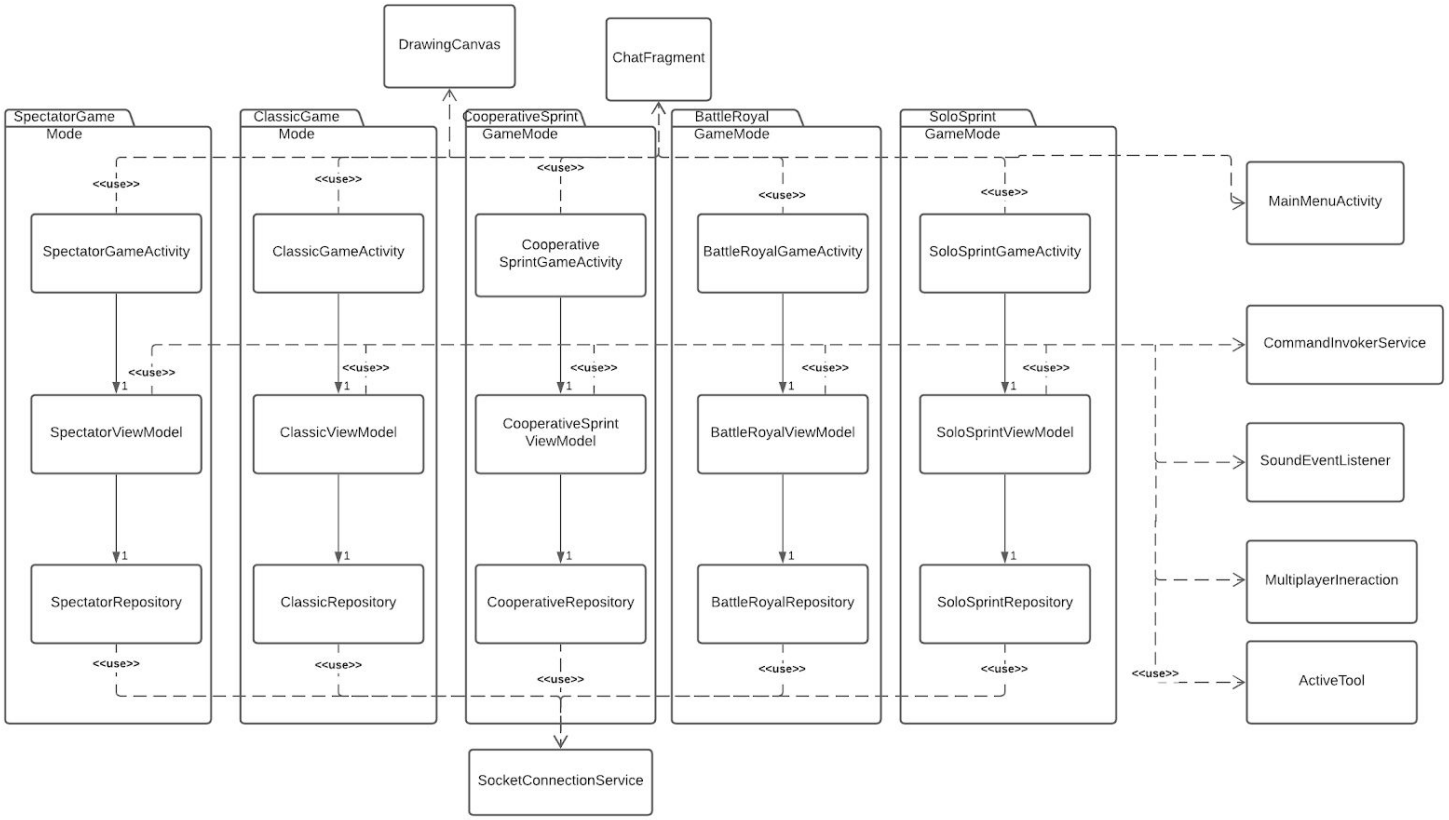
MainMenu
Ce paquetage représente le menu principal de l'application. Principalement, ses tâches seront de rediriger l'utilisateur vers le choix qu'il aura effectué dans le menu principal. Il pourra se déconnecter, créer un lobby, rejoindre un lobby, regarder son profil, regarder sa liste d'amis, regarder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



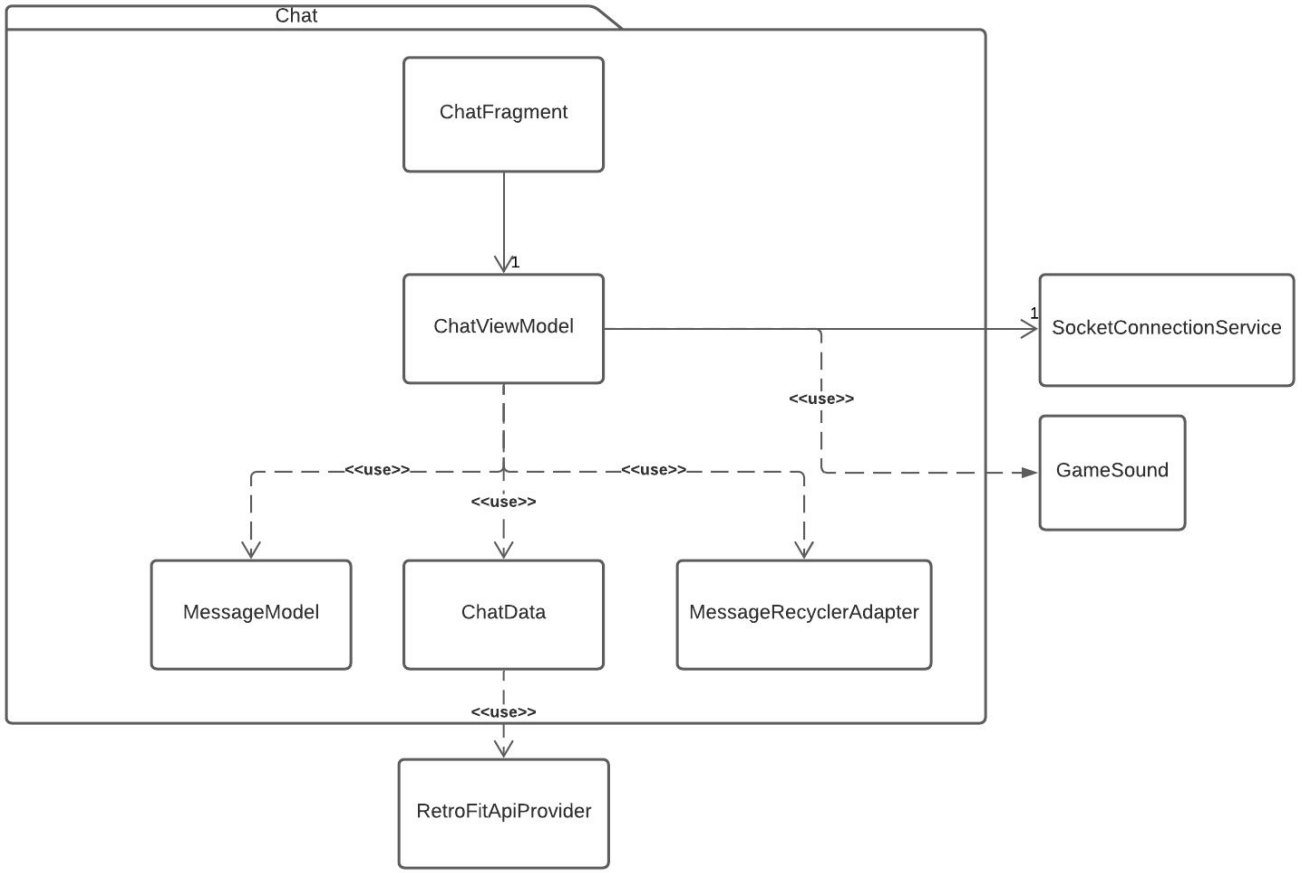
Tutoriel
Ce paquetage représente le tutoriel de l'application qui montrera à l'utilisateur les différentes options qui s'offrent à lui. Ce paquetage est un ensemble d'activités ayant les mêmes écrans que l'on retrouve dans l'application, cependant l'utilisateur devra effectuer les actions demandé par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



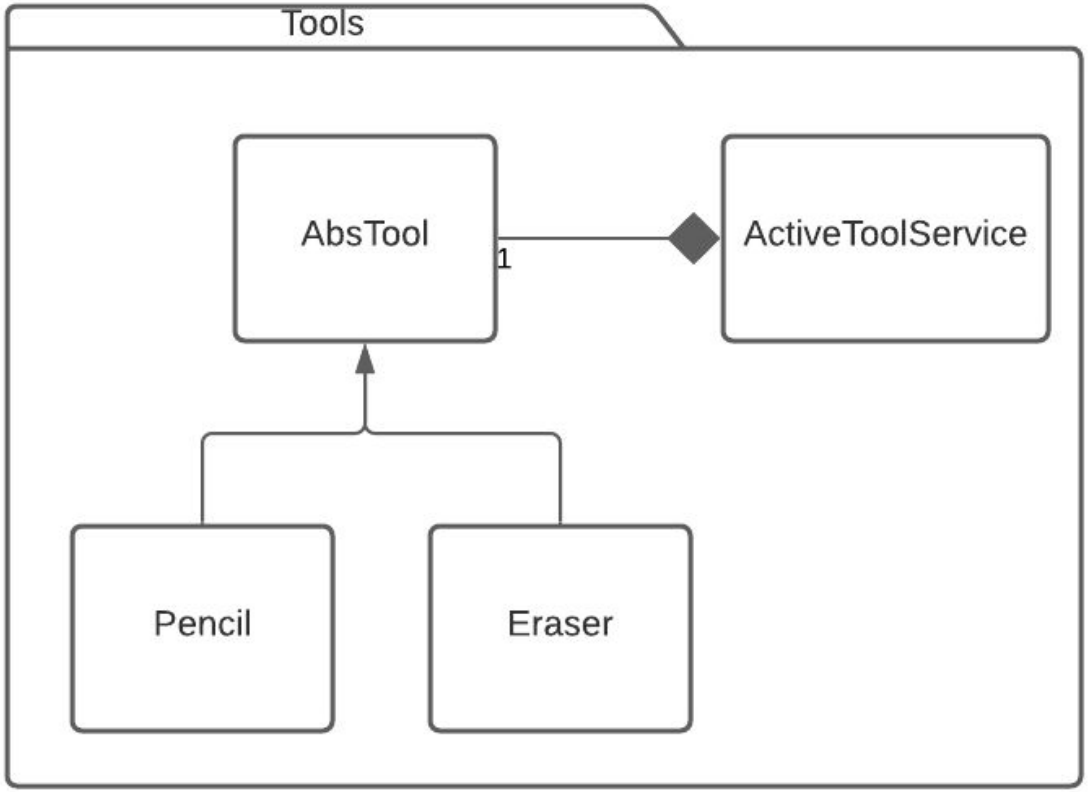
Game
Ce paquetage touche tout ce qui est en lien avec les différents modes de jeux, À l'intérieur de ce paquetage, on retrouve 5 paquets qui sont les 5 modes de jeux disponibles. Son but principal est de contrôler les activités lors d'une partie et de rester en connexion continue avec le serveur à l'aide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



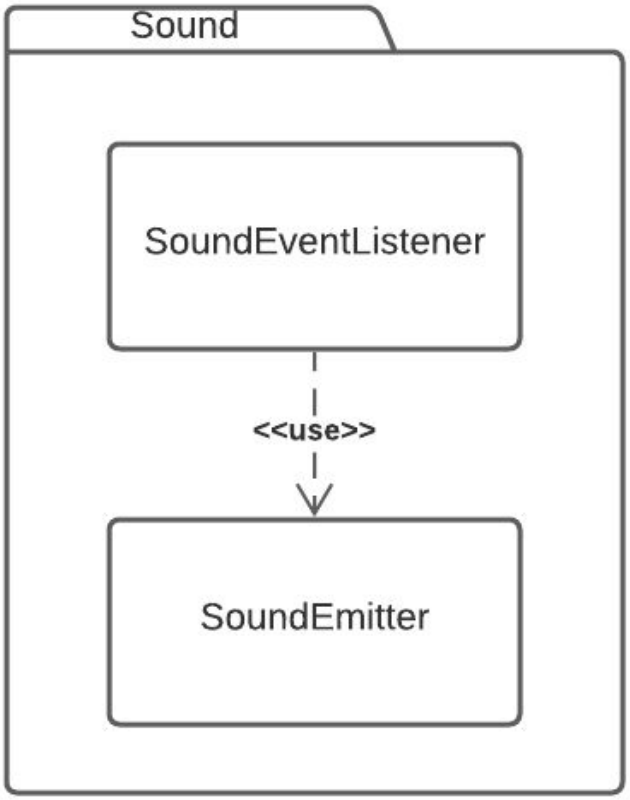
Chat
Ce paquetage représente tout ce qui est en lien avec la discussion avec d'autres joueurs. Ses tâches principales sont de recevoir des messages provenant d'autres clients et de les afficher, distinguer les messages d'un chat room d'un autre et de communiquer des tentatives de réponses durant une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



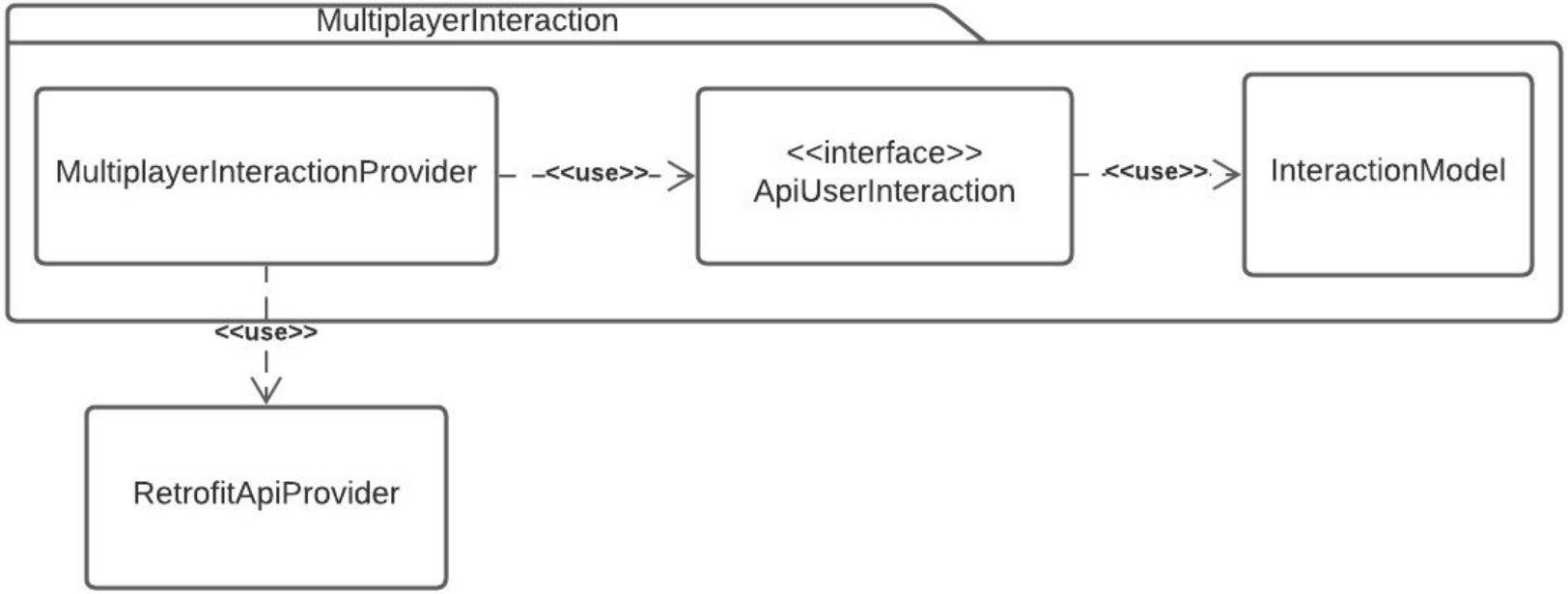
Tools
Ce paquetage représente les outils disponibles à l'utilisateur lorsqu'il veut effectuer un dessin. Le but principal de ce paquetage est de fournir le comportement désiré lors d'un touché à l'écran en fonction de l'outil choisi.



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.

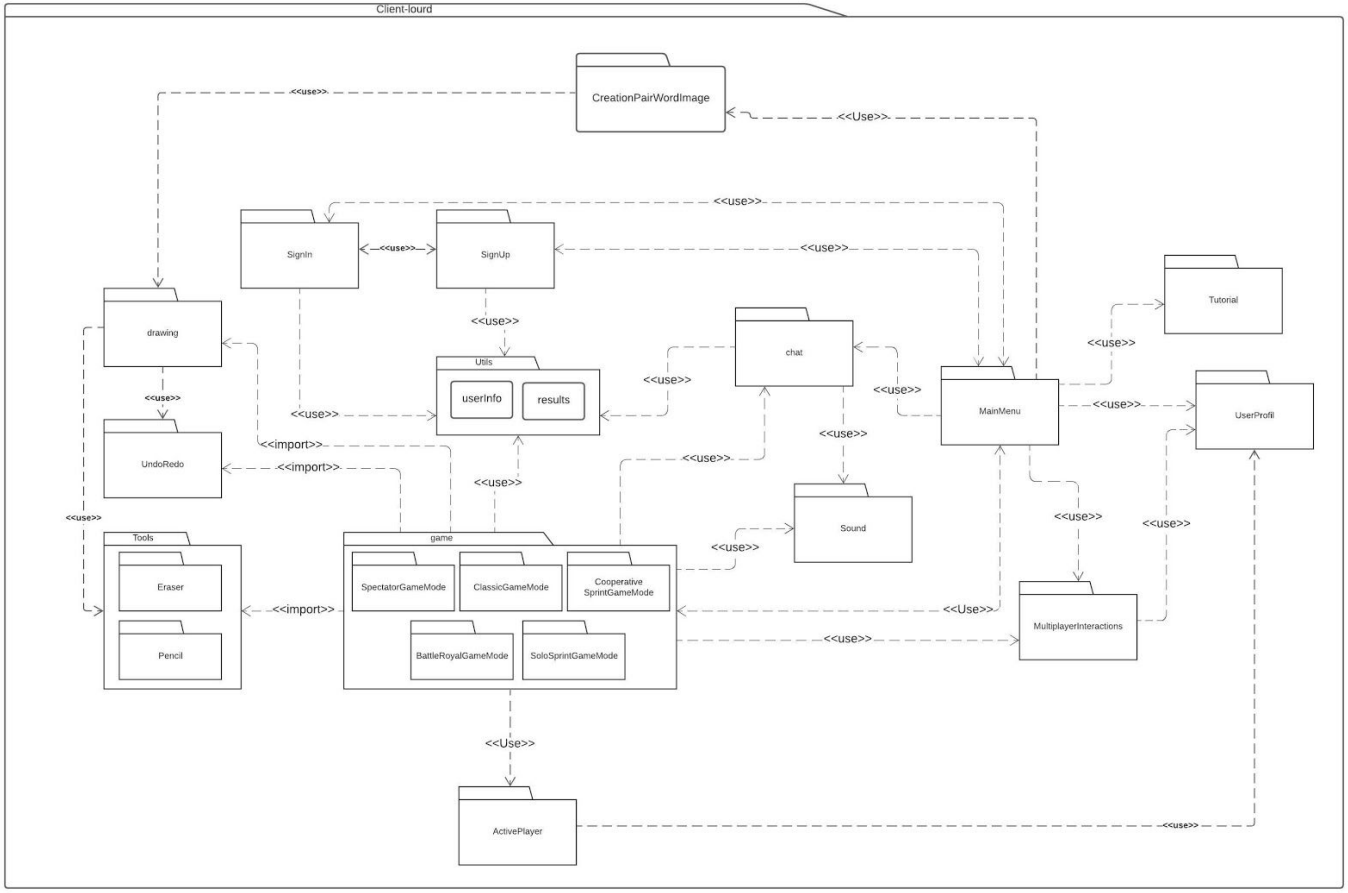


MultiplayerInteractions
Ce paquetage contrôle l'ensemble des interactions entre les joueurs durant le jeu. Son but principal est d'envoyer des pouces rouge ou vert aux autres joueurs pour noter leur dessins. Un autre de ses buts est de permettre aux joueurs virtuels d'interagir avec les autres joueurs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

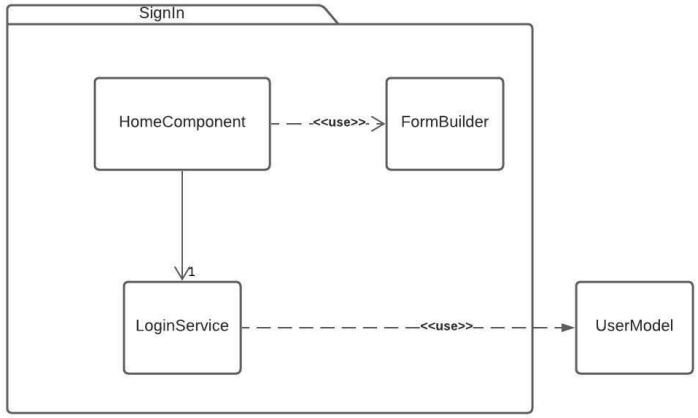


Client lourd:

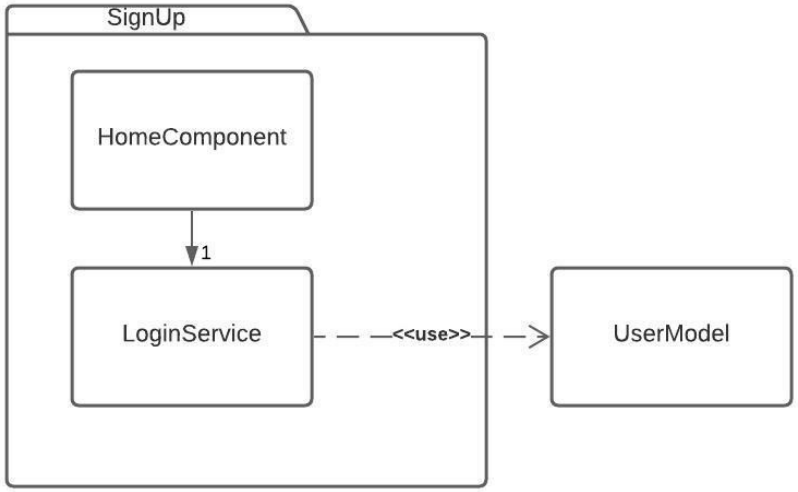
Client lourd
Ce paquetage représente l'ensemble du client lourd qui est l'application Electron. Ce paquetage a beaucoup de responsabilités. En général, il doit permettre de rejoindre des lobbys, chatter avec d'autres joueurs, créer des paires mots images et jouer aux différents modes de jeux en entrant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



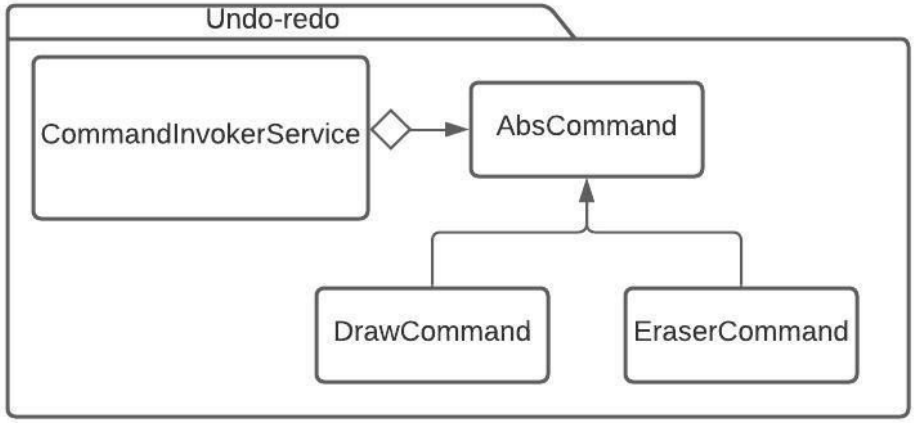
SignIn
Ce paquetage est le point d'entrée principale de l'application. En effet, son but principal est de permettre aux utilisateurs de se connecter en effectuant une requête POST au serveur. Il doit, par la suite, rediriger l'utilisateur vers le menu principal.



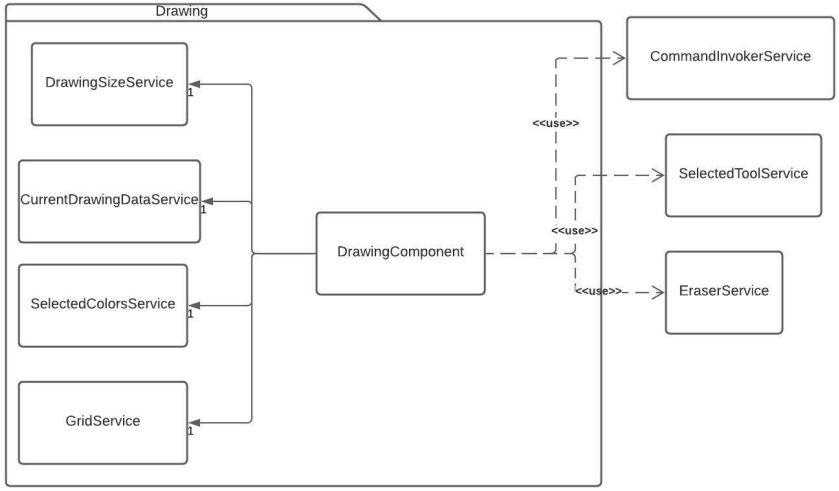
SignUp
Ce paquetage est un des points d'entrée pour accéder à l'application. Son but principal est de permettre aux utilisateurs de se créer un compte. Il doit aussi permettre à l'utilisateur de se rendre au menu principal de l'application s'il se créer un compte.



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.

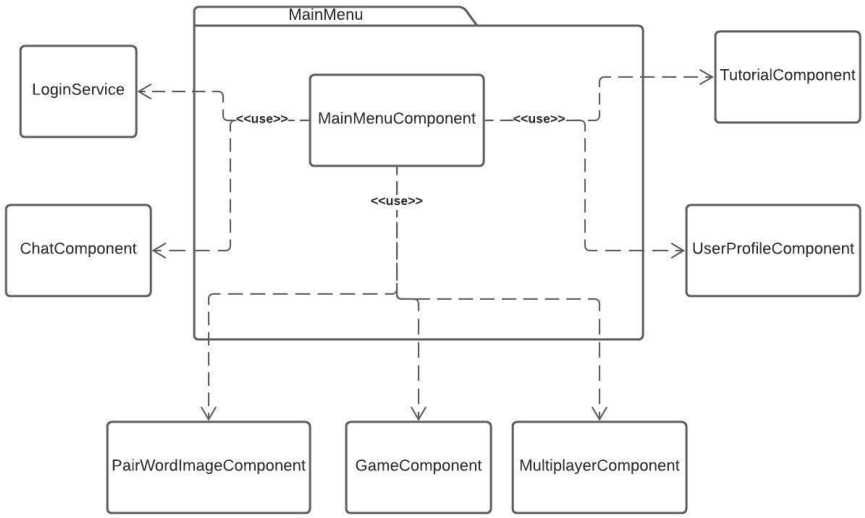


Drawing
Ce paquetage représente l'endroit ou un dessin sera effectué durant la partie. Sa responsabilité principale sera d'ajouter de permettre de dessiner des "path" et aussi d'en recevoir des autres joueurs lors d'une partie.

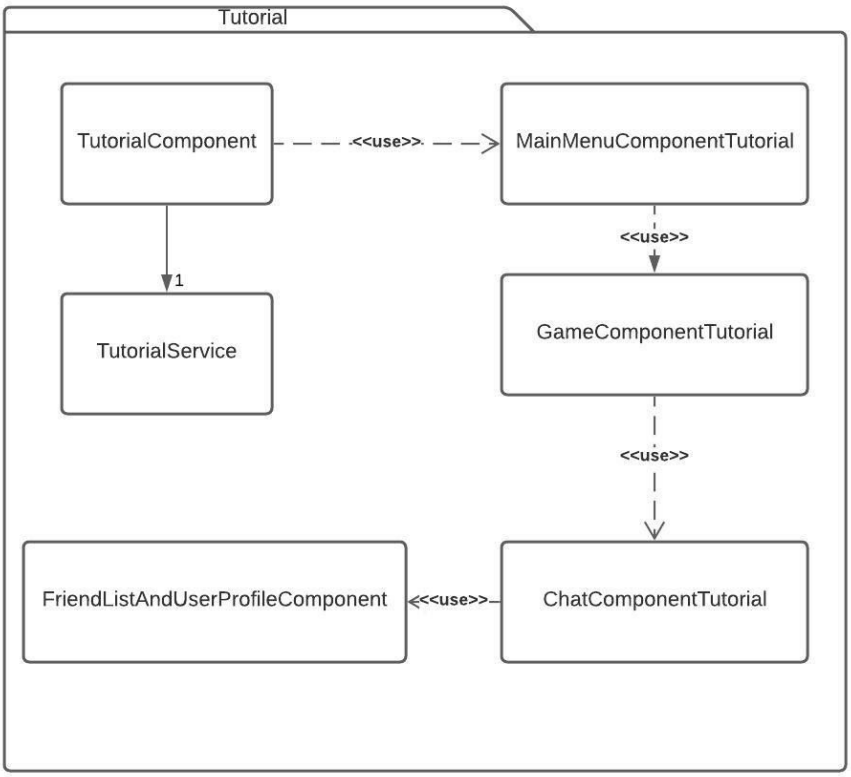

Drawing
DrawingSizeService
CurrentDrawingDataService
SelectedColorsService
GridService
DrawingComponent
CommandInvokerService
SelectedToolService
EraserService
<<use>>
<<use>>
<<use>>
1
1
1
1

=

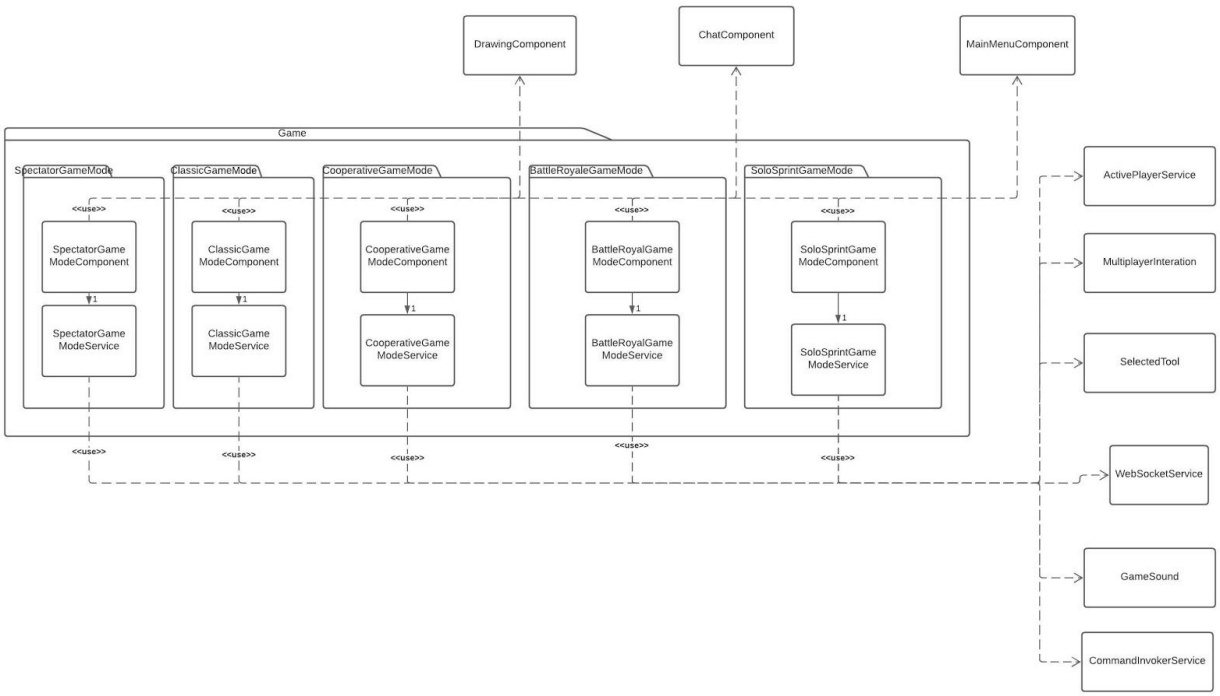
MainMenu
Ce paquetage représente le menu principal de l'application. Principalement, ses tâches seront de rediriger l'utilisateur vers le choix qu'il aura effectué dans le menu principal. Il pourra se déconnecter, créer un lobby, rejoindre un lobby, regarder son profil, regarder sa liste d'amis, regarder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



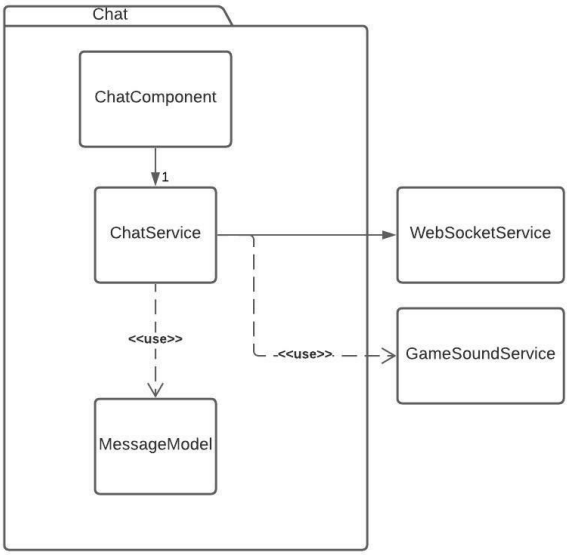
Tutorial
Ce paquetage représente le tutoriel de l'application qui montrera à l'utilisateur les différentes options qui s'offrent à lui. Ce paquetage est un ensemble d'activités ayant les mêmes écrans que l'on retrouve dans l'application, cependant l'utilisateur devra effectuer les actions demandées par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



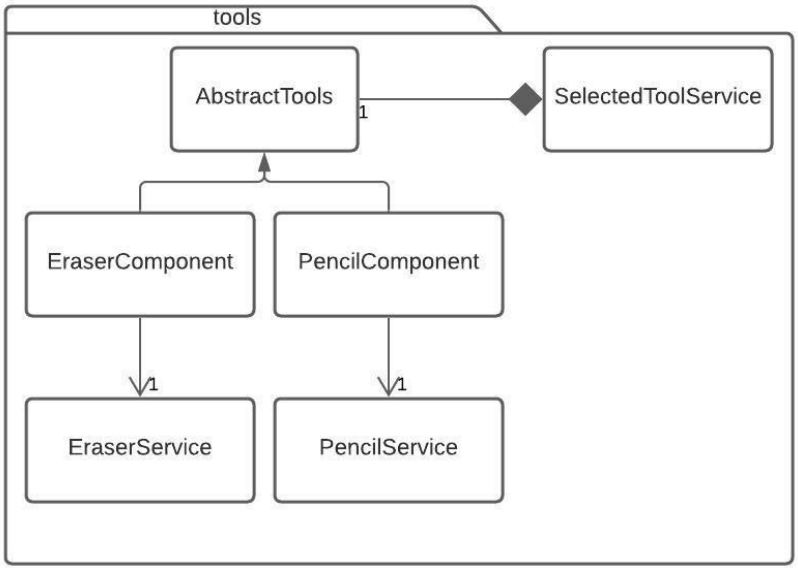
Games
Ce paquetage touche tout ce qui est en lien avec les différents modes de jeux, À l'intérieur de ce paquetage, on retrouve 5 paquets qui sont les 5 modes de jeux disponibles. Son but principal est de contrôler les activités lors d'une partie et de rester en connexion continue avec le serveur à l'aide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



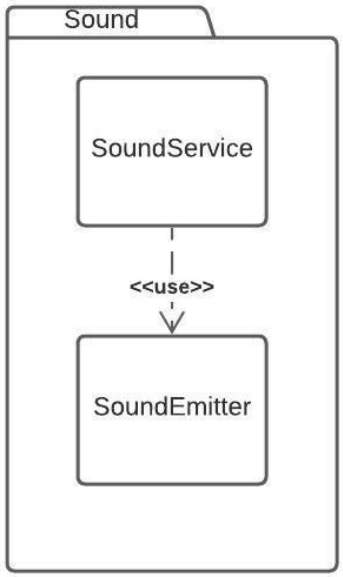
Chat
Ce paquetage représente tout ce qui est en lien avec la discussion avec d'autres joueurs. Ses tâches principales sont de recevoir des messages provenant d'autres clients et de les afficher, distinguer les messages d'un chat room d'un autre et de communiquer des tentatives de réponses durant une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



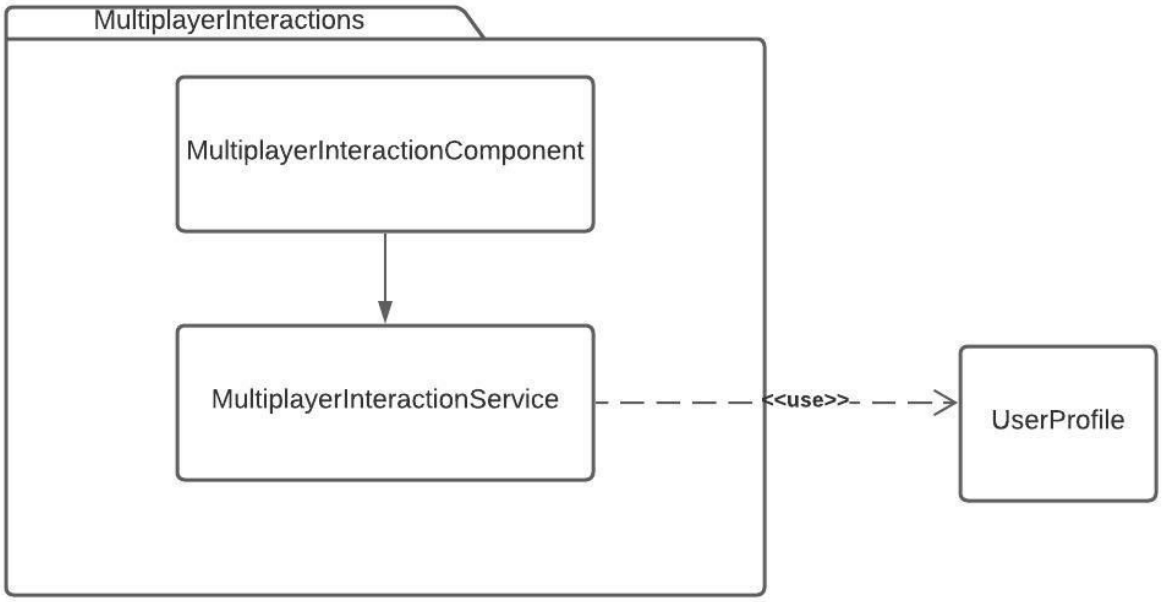
Tools
Ce paquetage représente les outils disponibles à l'utilisateur lorsqu'il veut effectuer un dessin. Le but principal de ce paquetage est de fournir le comportement désiré lors d'un touché à l'écran en fonction de l'outil choisi.



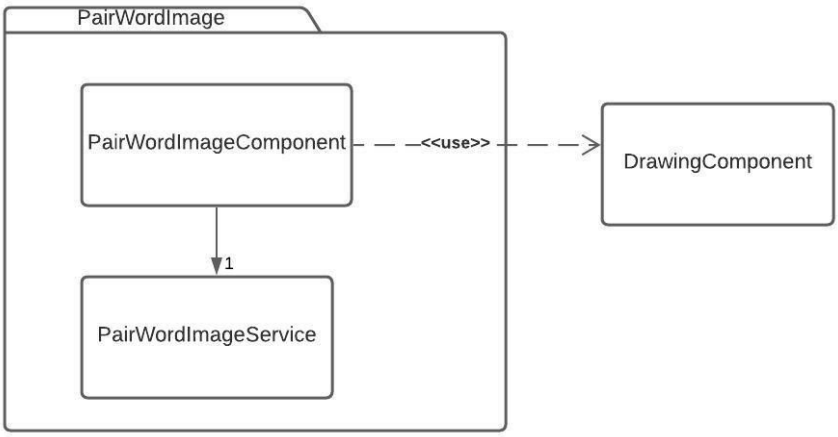
Sound
Ce paquetage s'occupe des différents sons que l'on retrouve dans l'application. Son but principal est de faire jouer le son désiré lorsqu'un événement particulier est observé, par exemple un nouveau message.



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.

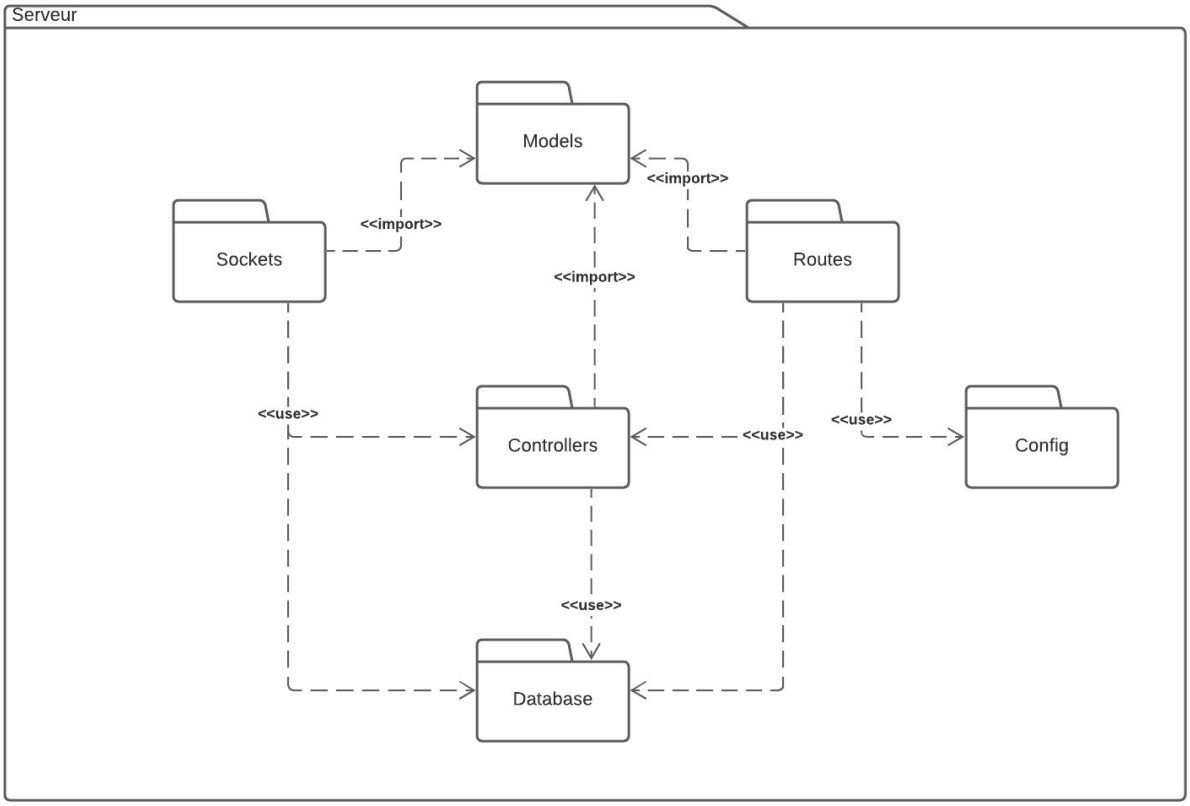


PairWordImage
Ce paquetage s'occupe de la création des paires mot-image sur le client lourd uniquement. Son but principal est de permettre aux utilisateurs du client lourds de lié un mot à deviner avec un dessin qu'un joueur virtuel pourra ensuite recréer lors d'une partie.

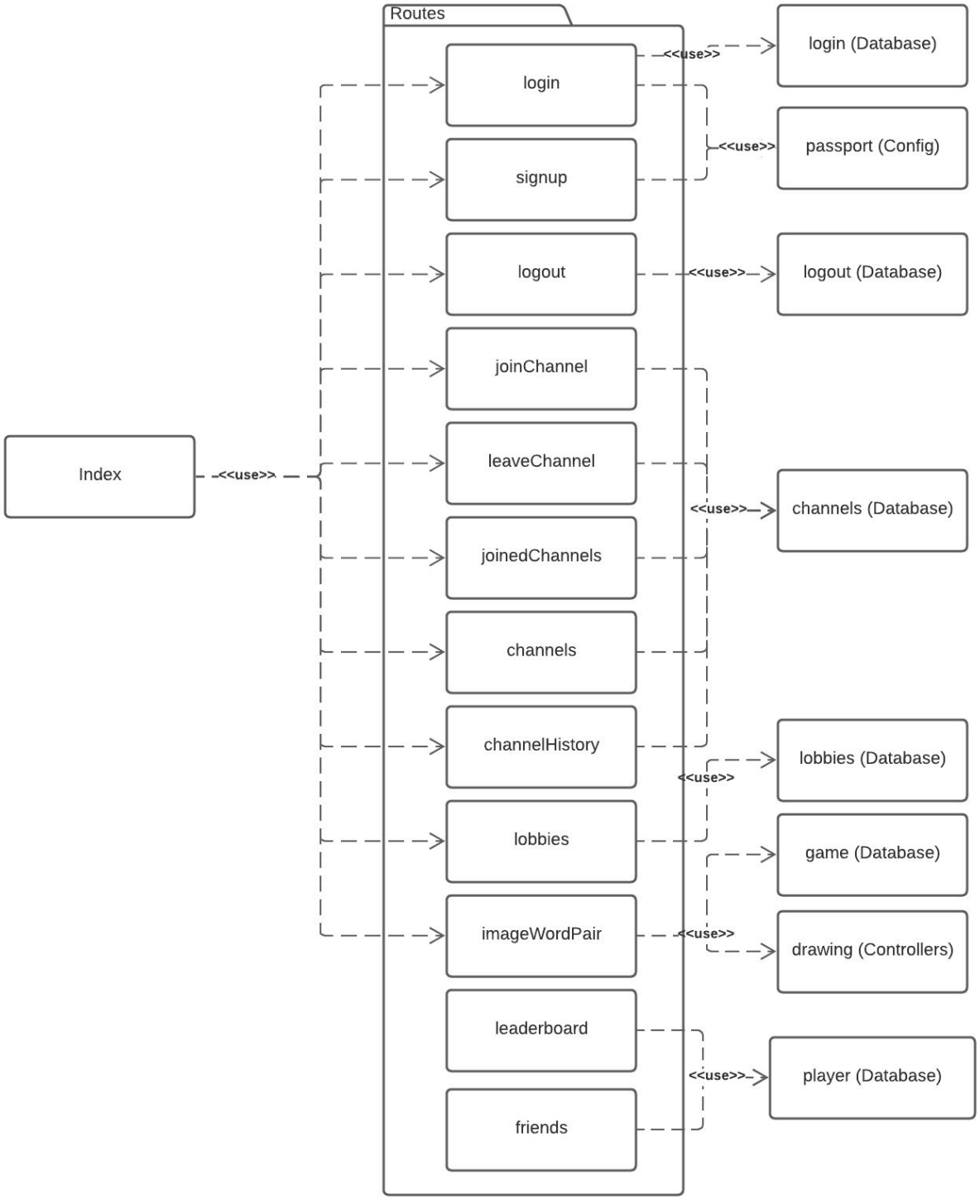


Serveur:

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.

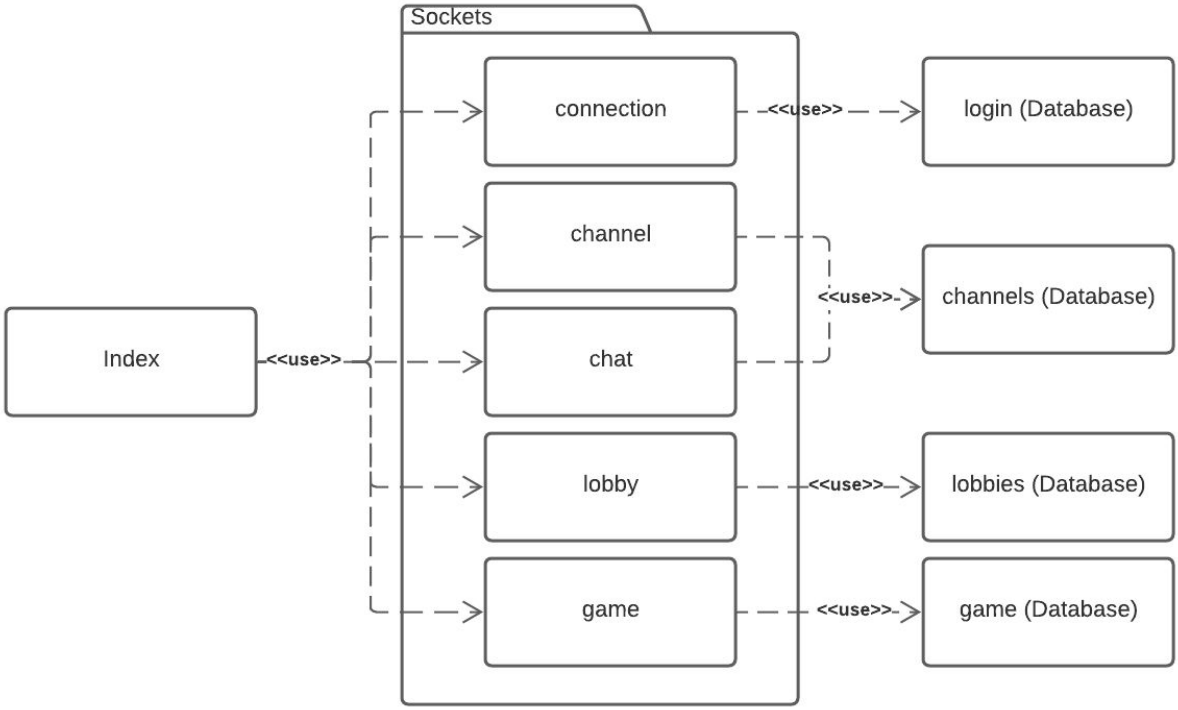


Routes
Ce paquetage contient toutes les routes disponibles par requêtes HTTP (GET, POST, etc.) Chaque classe correspond à un endpoint. De plus, chaque classe doit rester le plus simple possible en ne s'occupant que de recevoir une requête, d'appeler la méthode correspondante dans le paquetage Controllers,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

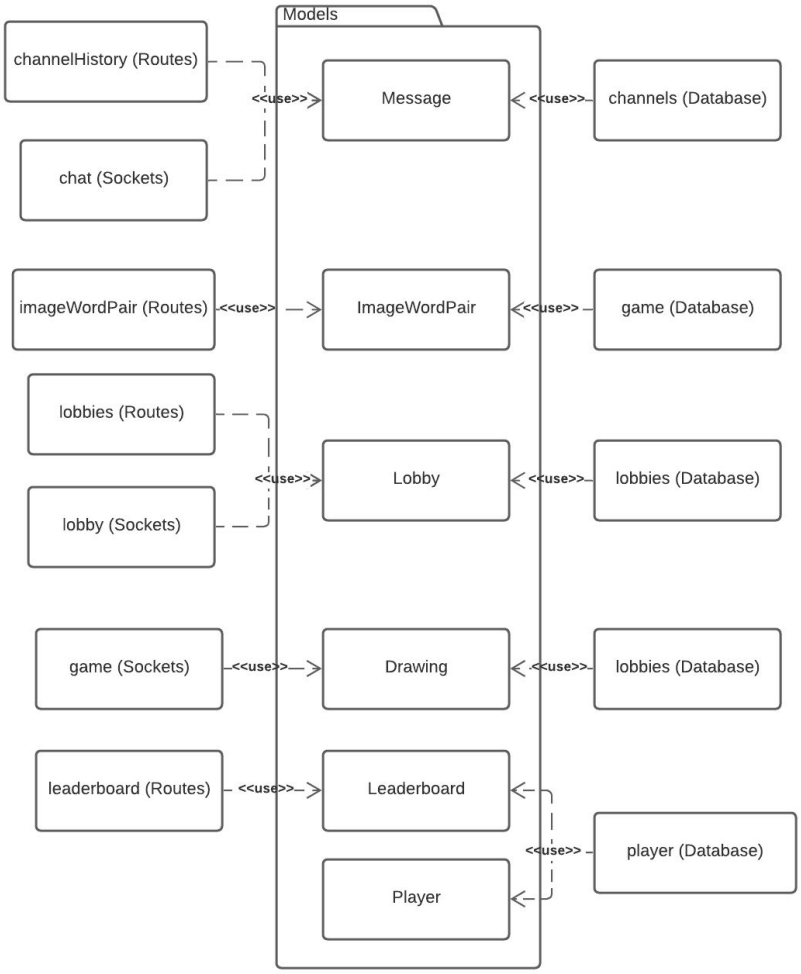


Sockets
Ce paquetage contient tous les événements disponibles par WebSockets. Chaque classe correspond à une fonctionnalité et peut donc avoir plusieurs

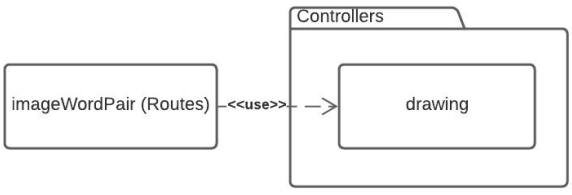
listeners de WebSockets selon la fonctionnalité associée.



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

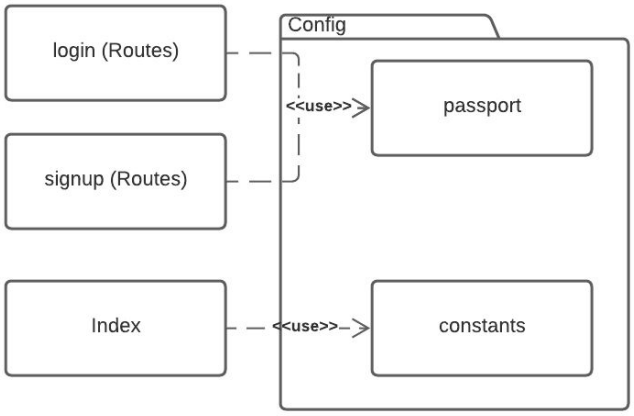


Controllers
Ce paquetage contient la logique entre les données reçues d'un client et les données mises dans la base de données. Il est surtout utile dans les cas où un travail assez important est nécessaire, ce qui permet d'alléger le travail fait dans *Routes* (par exemple la conversion d'une image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

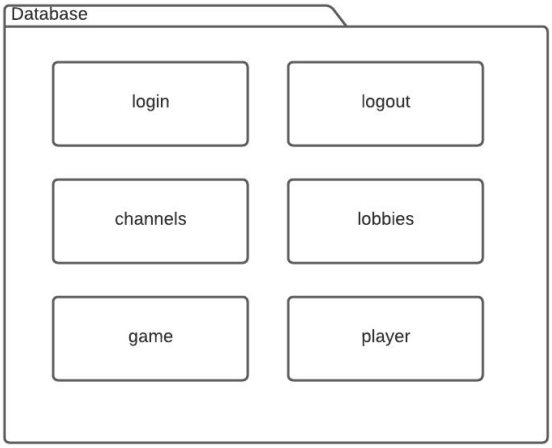


Config
Ce paquetage contient les informations nécessaires à la configuration du

serveur. Il contient aussi la configuration du service d'authentification Passport.js

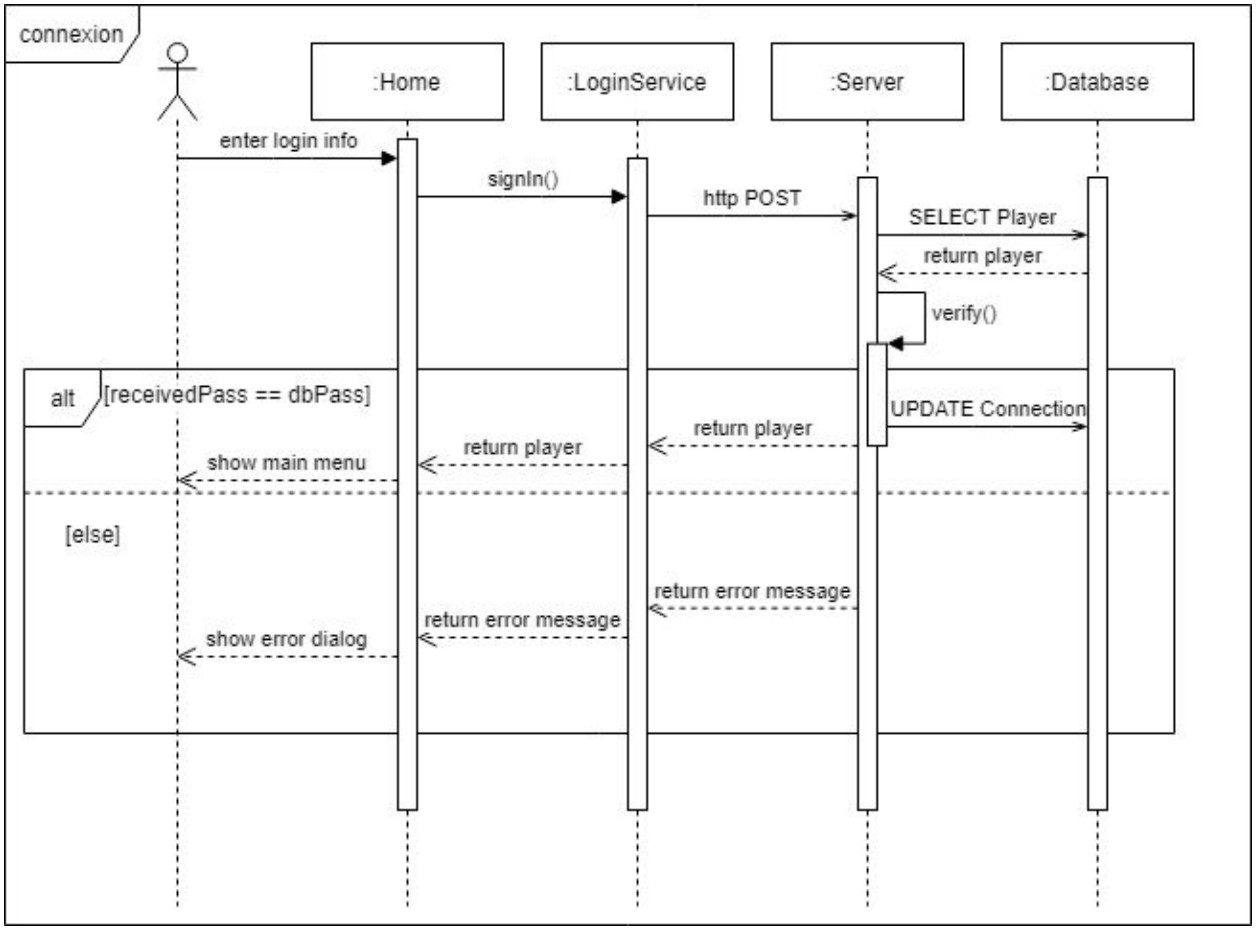


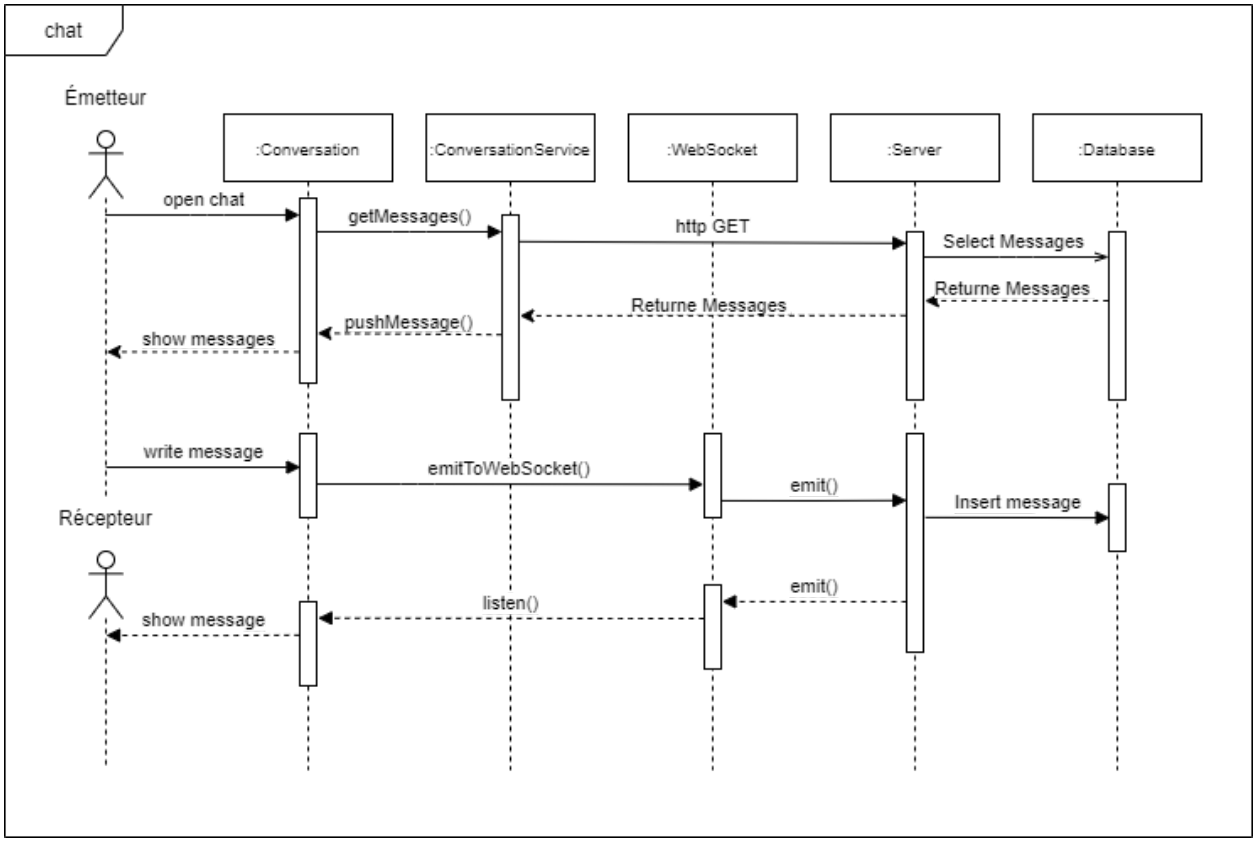
Database
Ce paquetage contient tous les appels à la base de données.

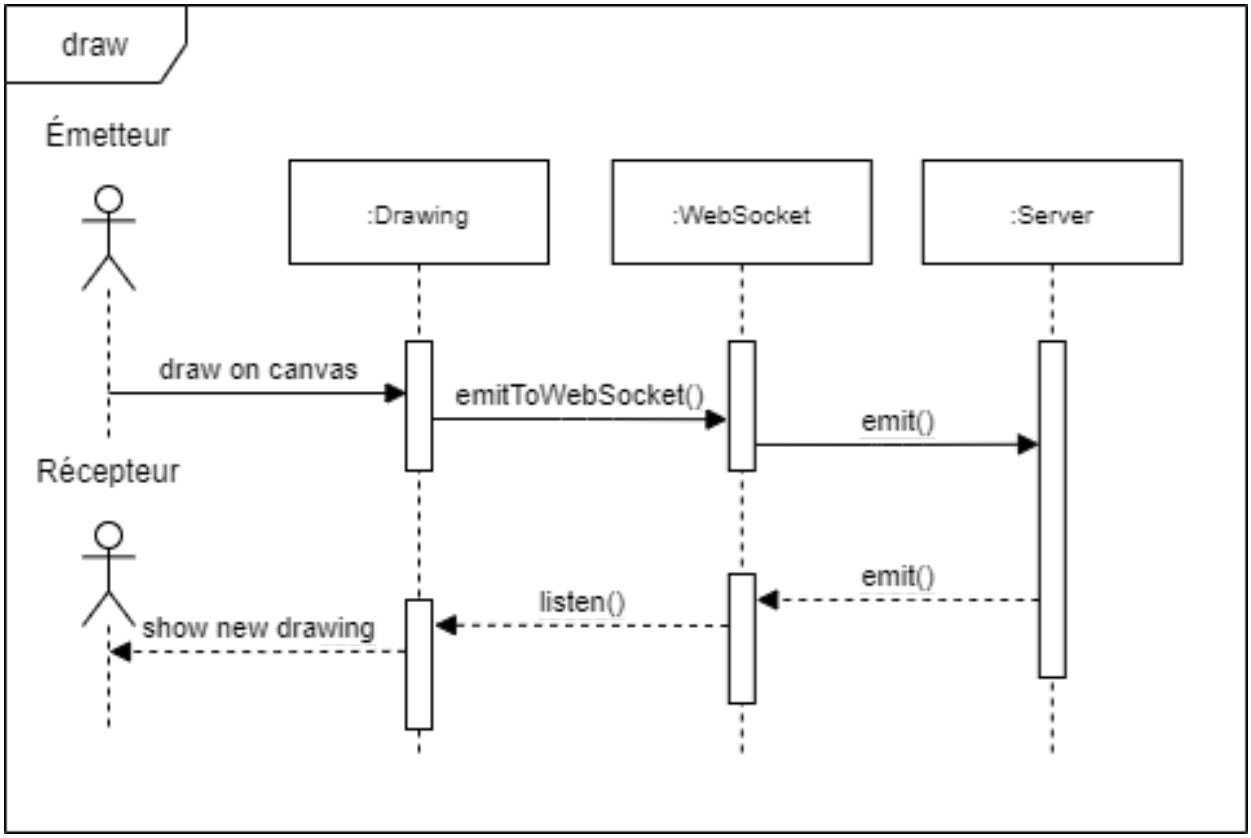


5. Vue des processus

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.

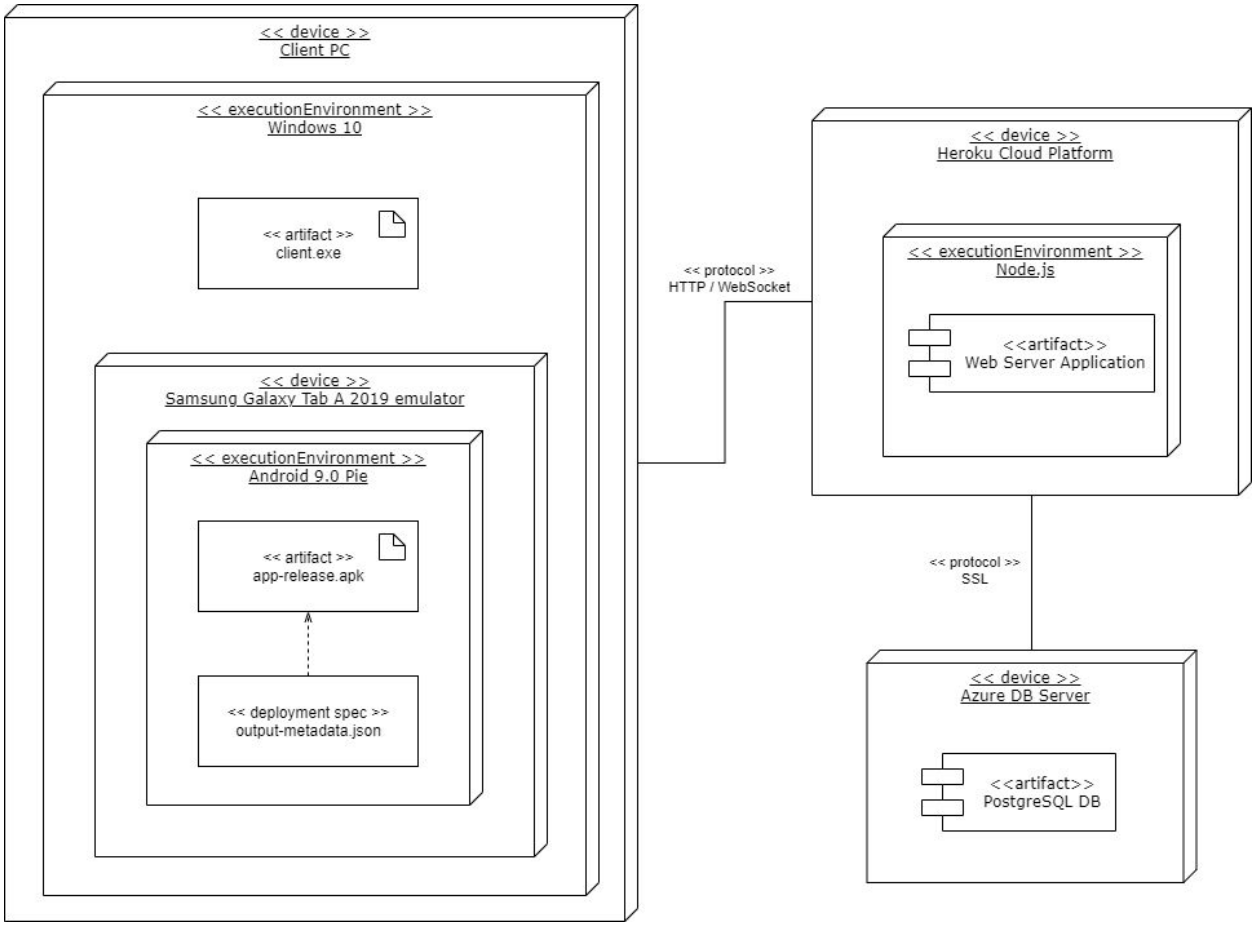

connexion
:Home
:LoginService
:Server
:Database
enter login info
signIn()
http POST
SELECT Player
return player
verify()
alt
[receivedPass == dbPass]
UPDATE Connection
return player
return player
show main menu
[else]
return error message
return error message
show error dialog


chat
Émetteur
:Conversation
:ConversationService
:WebSocket
:Server
:Database
open chat
getMessages()
http GET
Select Messages
Returne Messages
Returne Messages
pushMessage()
show messages
write message
emitToWebSocket()
emit()
Insert message
Récepteur
emit()
listen()
show message


draw
Émetteur
:Drawing
:WebSocket
:Server
draw on canvas
emitToWebSocket()
emit()
Récepteur
emit()
listen()
show new drawing

6. Vue de déploiement

Le système est composé de trois parties: le client lourd, le client léger et le serveur.
Le client léger est représenté par un fichier .apk déployé sur un émulateur de tablette Samsung Galaxy Tab A 2019 qui roule Android 9.0 Pie. Cet émulateur est lui-même déployé sur le système d'opération Windows 10 du PC du client. Le client lourd est d'ailleurs déployé sur ce même système, mais à partir d'un fichier .exe généré par Électron.
Le serveur comporte un serveur web déployé sur la plateforme Node.js et hébergé sur Heroku. Un serveur de Azure est aussi utilisé pour déployer un serveur de base de données de type PostgreSQL.
Le client lourd et le client léger communiquent avec le serveur par des requêtes HTTP et par connexion WebSocket. Les deux types de clients utilisent donc la connexion internet du PC pour communiquer avec le serveur sur Heroku à l'aide de son adresse IP.



7. Taille et performance

L'exécutable du logiciel ne doit pas être trop lourd. Nous viserons une taille de 300 Mb maximum.

Le client lourd doit être en mesure d'offrir une expérience sans latence à l'utilisateur. L'utilisateur doit être capable de s'authentifier dans un temps acceptable. Le délai devrait être de maximum 5 secondes une fois la requête envoyée. 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 complètement sans latence. L'utilisateur devrait être capable de clavarder en voyant le dessin s'afficher sans latence.

Le client-léger doit offrir une expérience similaire au client-lourd.

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.

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.