Équipe 102

PolyScrabble
Document d'architecture logicielle

Version 1.4

Historique des révisions

Date	Version	Description	Auteur
2022-09-19	1.0	Rédaction initial du document	Équipe 102
2022-09-21	1.1	Ajouts des diagrammes	Stéphane, Maximiliano
2022-09-26	1.2	Correction, ajout et mise à jour de diagrammes (séquences, paquetages, déploiement)	Stéphane
2022-09-26	1.3	Ajouts des diagrammes de paquetages et de classes	Corentin, Mohamed
2022-09-28	1.4	Derniers ajouts, retraits, ajustements, corrections au diagrammes. Mise en page du document	Équipe 102

Table des matières

Document d'architecture logicielle

1. Introduction

Ce document d’architecture logicielle contient tous nos choix et apportent une justification à nos choix de notre architecture logicielle. Le but de ce document est de montrer les différents éléments de notre architecture a un haut niveau et en décrivant les différentes séquences possibles. Le document suivant seras une série de de diagrammes afin de montrer visuellement nos choix.

On compte différents types de diagrammes, notamment les cas d’utilisation, la vue logique, la vue du déploiement. Nous allons commencer par décrire nos objectifs et nos contraintes en termes d’architecture, suivi par tous nos diagrammes, et conclure sur la taille et la performance du système et de ses implications sur nos choix d’architecture.

2. Objectifs et contraintes architecturaux

2.1 Serveur

Le but du serveur est de s’assurer le bon déroulement du jeu et de bien gérer la communication client-serveur, peu importe qu’il soit le client lourd ou léger. Dû aux exigences de l’avatar, il doit aussi s’assurer du stockage des données sur la base de données en communiquant avec elle.

Nous avons un système d'authentification et à ce fait il est impératif de garder la confidentialité des utilisateurs, ce qui signifie ne pas stocker les mots de passe en clair sur la base de données, utiliser des routes protégées à travers des tokens, cookies. Le serveur doit toujours faire une vérification de son bord de toute requête même si celle-ci est déjà faite par le client.

Le serveur est central à l’utilisation de notre plateforme, a ce fait il est impératif qu’il n’y ait le moins de pannes possible, notre instance EC2 sur AWS doit être toujours en marche avec un downtime de maximum une heure par semaine.

A propos de nos choix technologiques, nous avons repris la base de code de projet 2, qui est donc NodeJS avec Express dans le langage de TypeScript. Par principe, il peut rouler sur n’importe quel système d’exploitation ayant une installation de NodeJs, mais le notre roulera sur une instance Linux sur AWS. En ce qui concerne notre base de données, on utilise MongoDB.

2.2 Client Léger

Nous devons pour le client léger développer une application pour la plateforme Android, plus particulièrement, une Samsung Galaxy Tab A 2019. Nous avons choisi d’utiliser le framework Flutter afin de développer notre application.

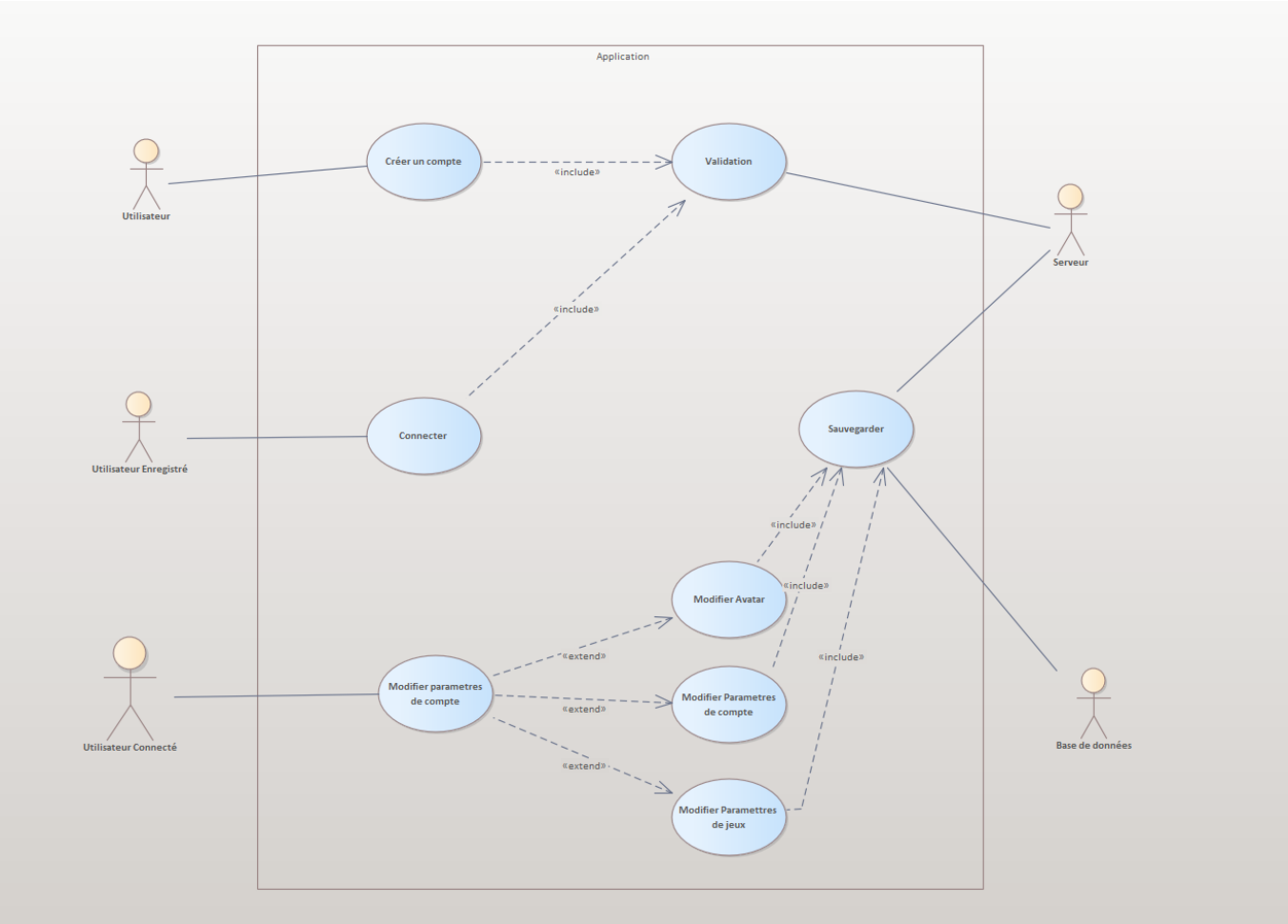
Notre client léger doit être capable de communiquer avec le serveur et ainsi avec d'autres clients léger et des clients lourds, il ne doit pas remarquer les différences entre les deux plateformes.

2.3 Client Lourd

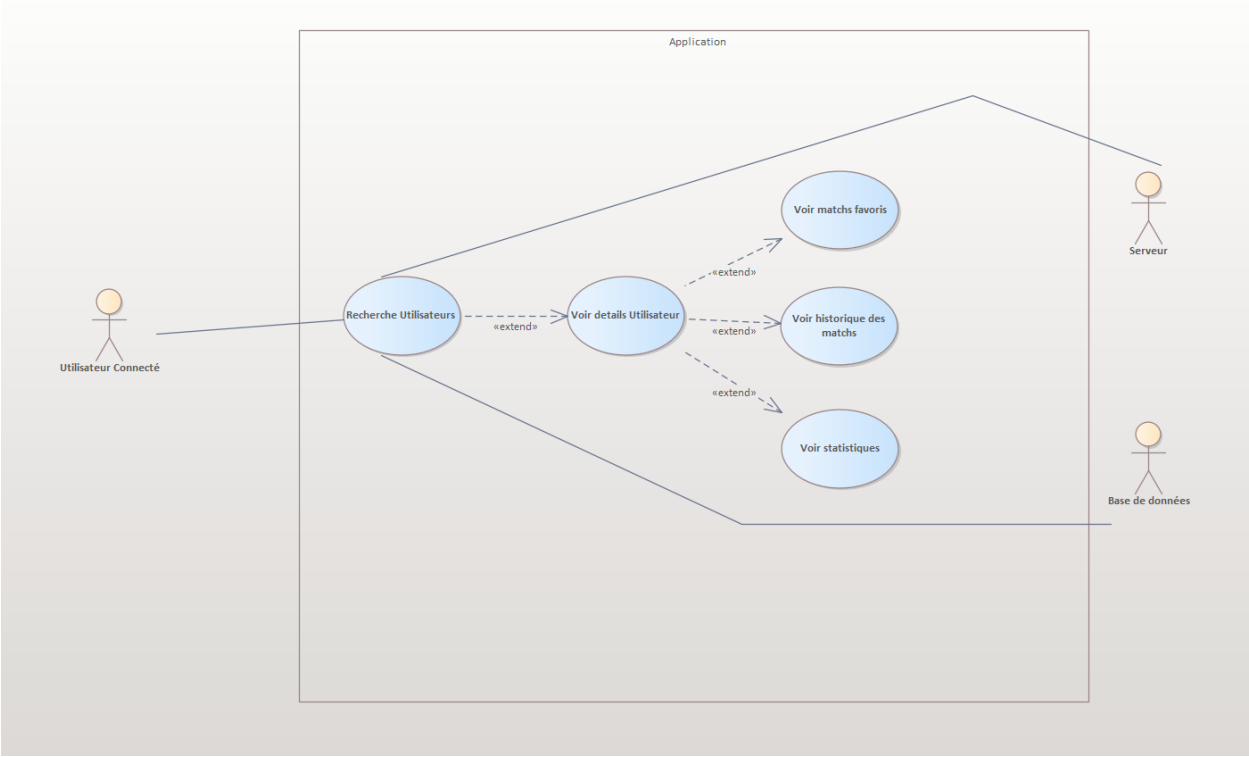
Pour le client lourd, on utilise le cadriciel ElectronJS ce qui nous permet de réutiliser la plupart de code source du projet 2. Le cadriciel permet de wrapper notre site pour qu'il fonctionne comme exécutable sur n'importe quelle plateforme (Linux, Windows ou MacOs), dans le cadre de ce projet, nous allons développer spécifiquement la plateforme de Windows .

Pareil que le client léger, notre client lourd doit être capable de communiquer avec le serveur, ainsi que d'autres clients lourds et des clients légers.

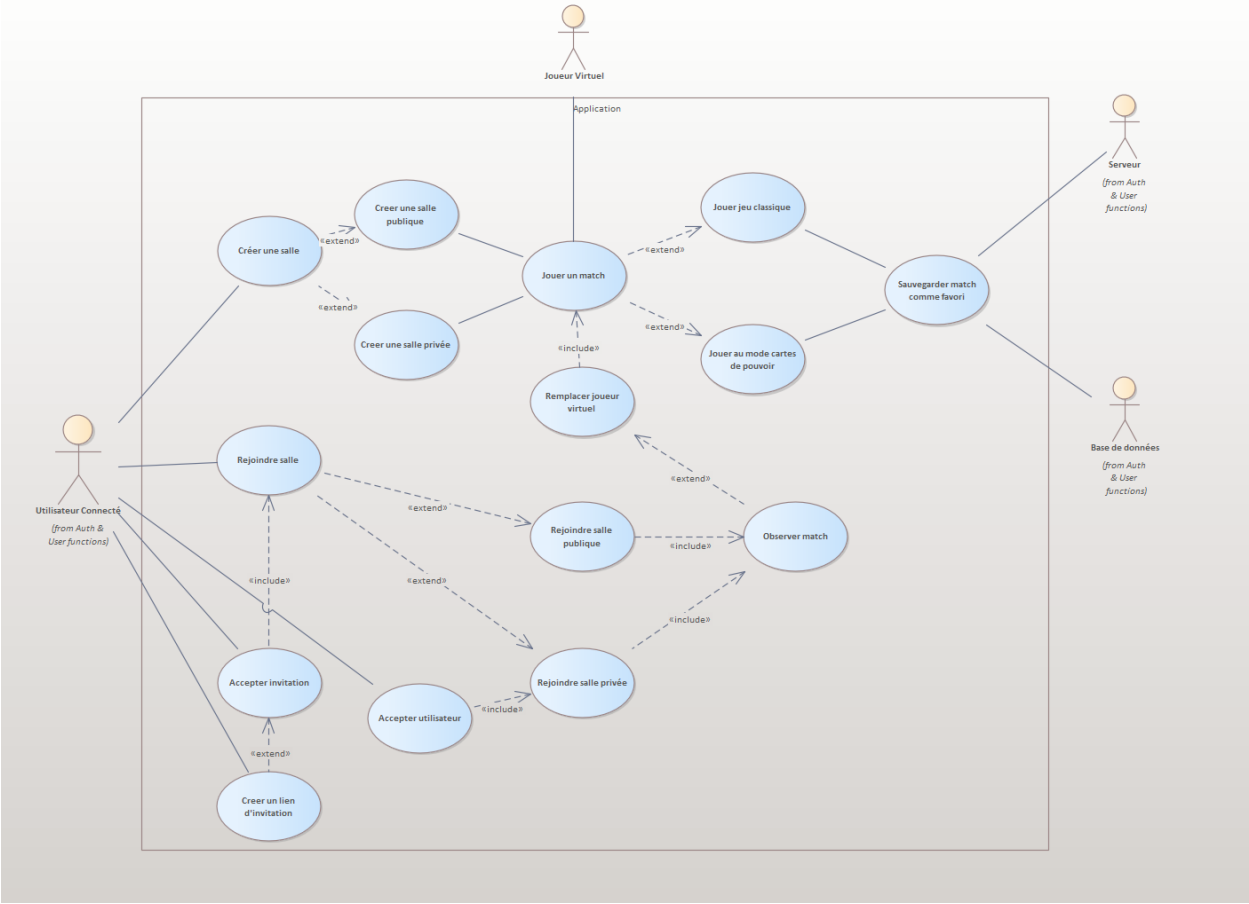
3. Vue des cas d'utilisation



[Figure 1] Cas d'utilisation de l'authentification

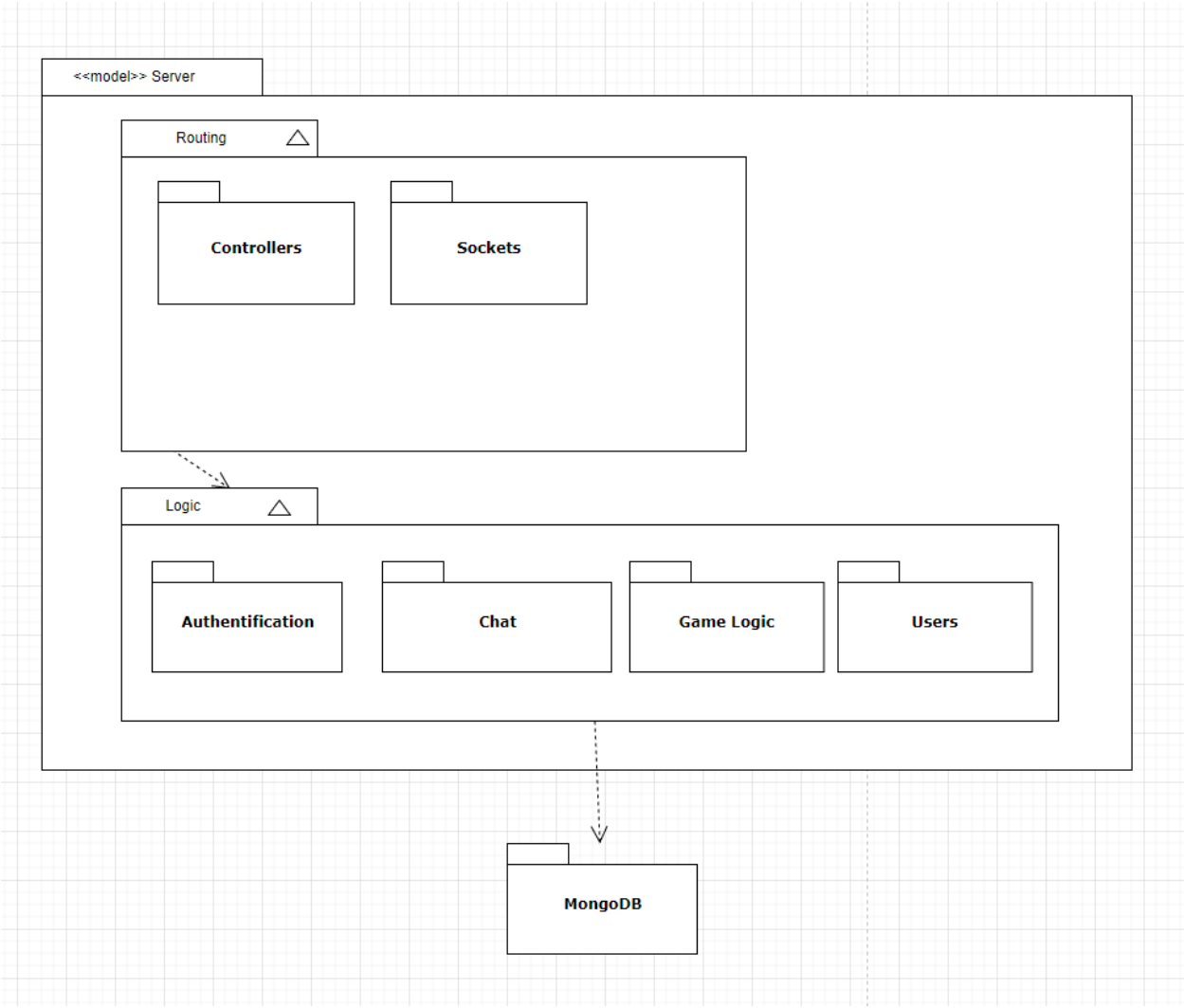


[Figure 2] Cas d'utilisation de la recherche des autres utilisateurs



[Figure 3] Cas d'utilisation du jeu PolyScrabble

4. Vue logique

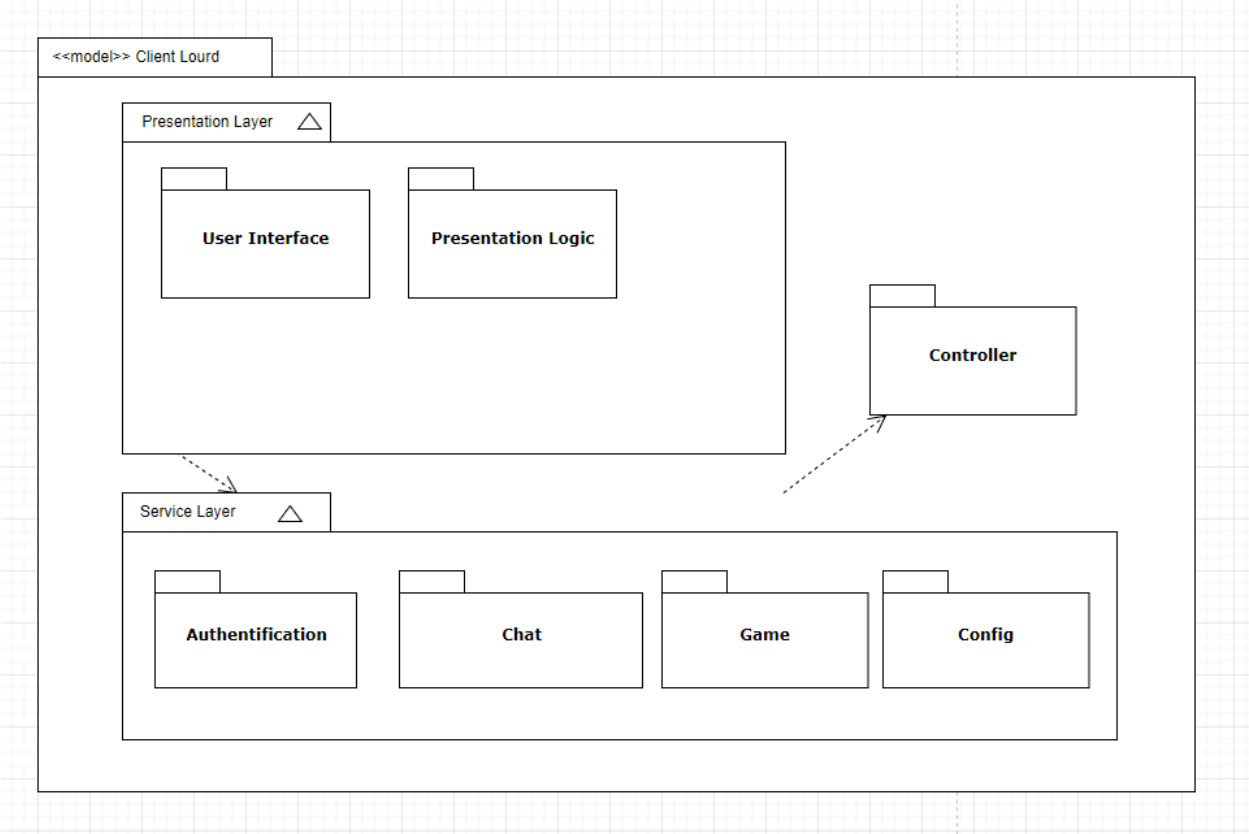


[Figure 4] Diagramme de packetage pour le seveur

Routing
Le package Routing contient toutes les classes qui s'occupent du routing et des sockets.

Logic
Le package Logic s'occupe de toute la logique nécessaire pour l'authentification, le chat, le game logic et les users. Il s'occupe aussi de communiquer avec MongoDB..

MongoDB
MongoDB correspond à la base de données MongoDB.

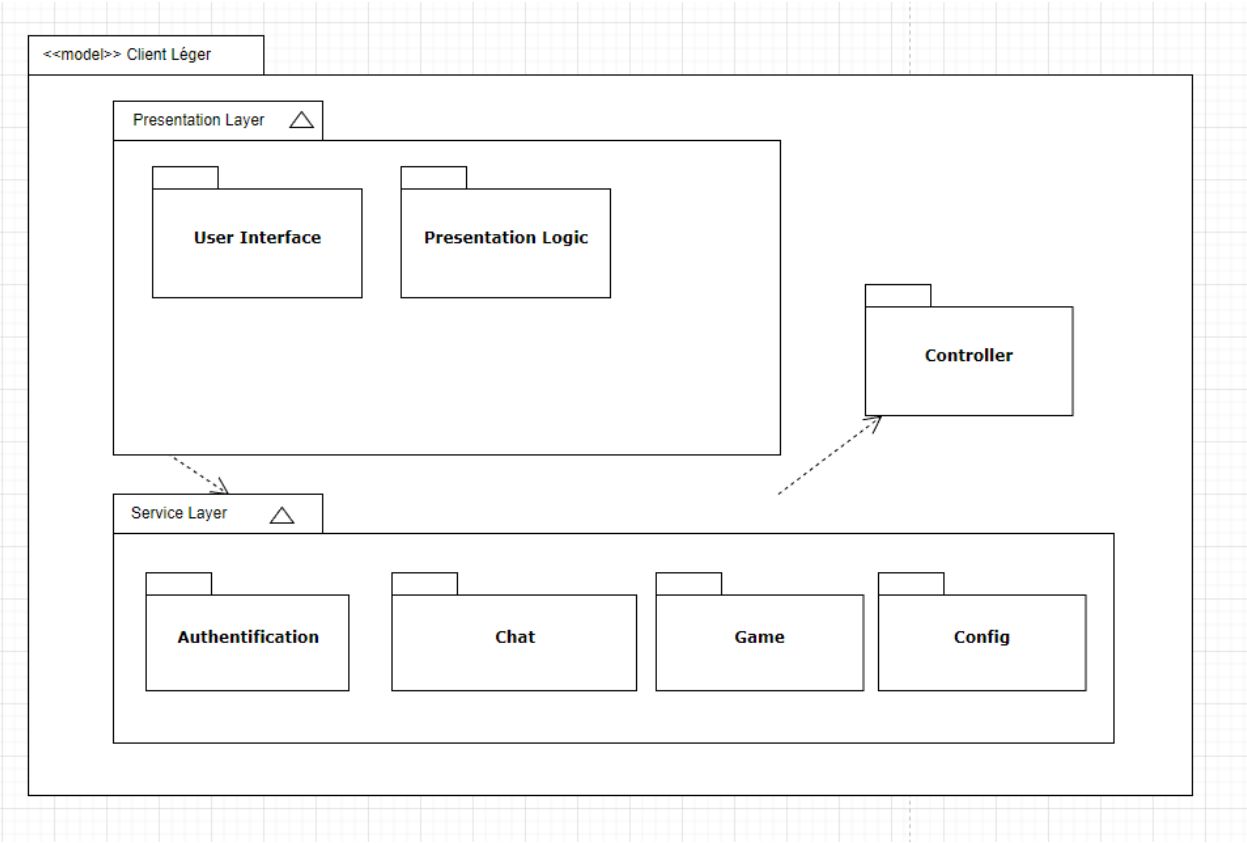


[Figure 5] Diagramme de packetage pour le client lourd

Presentation Layer
Le “presentation layer” s’occupe de la présentation des données avec tous les “components”.

Service Layer
Le “service layer” regroupe tous les services qui vont servir les components.

Controller
Ce package va regrouper tous les services qui vont interagir avec le serveur.

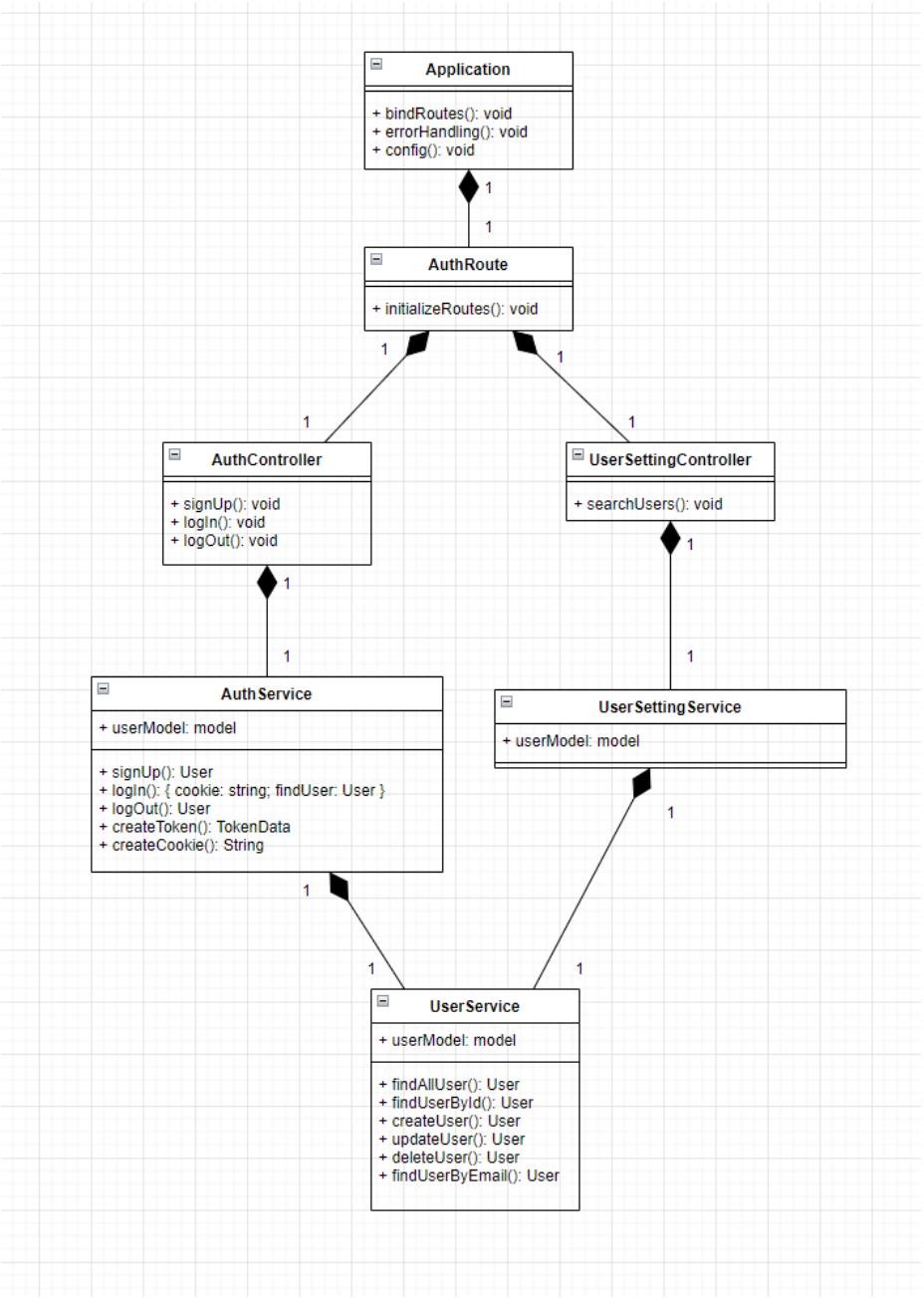


[Figure 6] Diagramme de packetage pour le client léger

Presentation Layer
Le “presentation layer” s’occupe de la présentation des données avec tous les widgets.

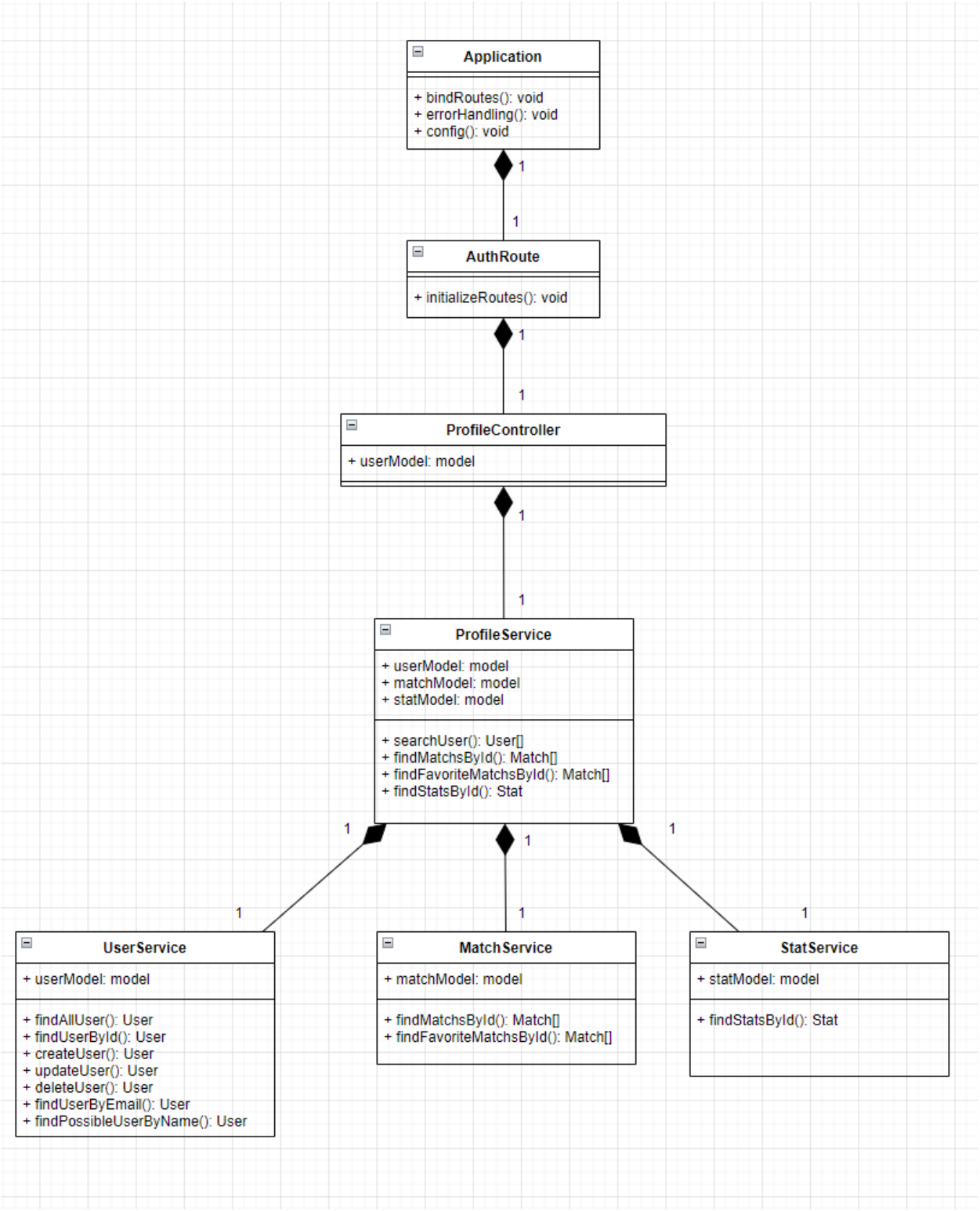
Service Layer
Le “service layer” regroupe toutes les classes qui vont servir les widgets.

Controller
Ce package va regrouper toutes les classes qui vont interagir avec le serveur.



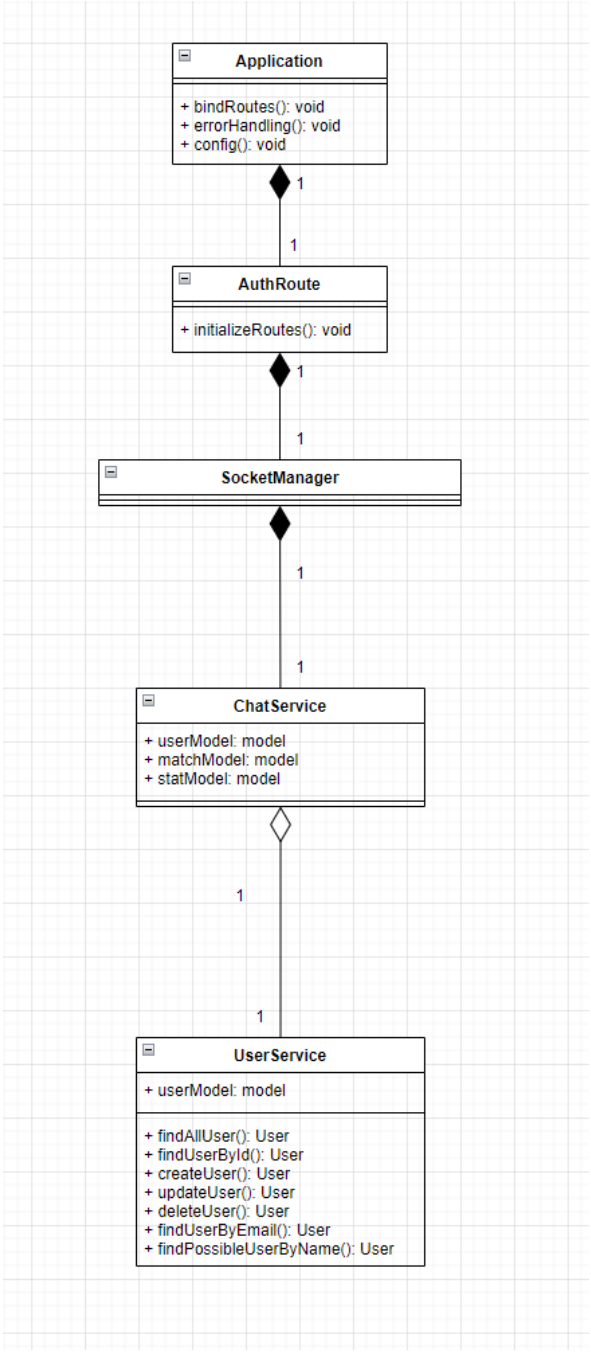
[Figure 7] Diagramme de Classe pour l'Authentification

Authentification
Ce package s'occupe de l'authentification des utilisateurs dans le serveur.



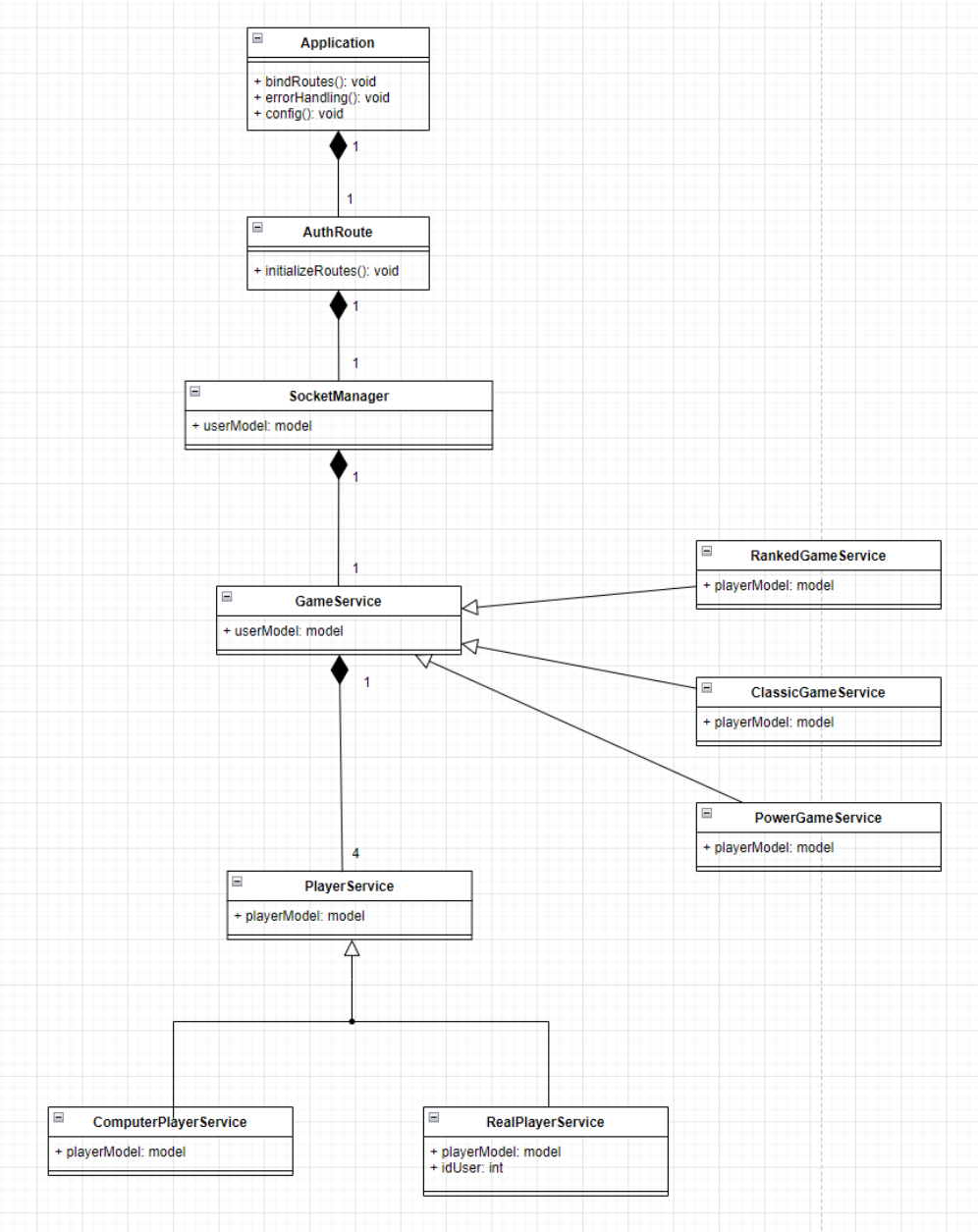
[Figure 8] Diagramme de Classe pour le profile

Profile
Ce package s'occupe de la recherche d'utilisateur et du profil des utilisateurs du côté serveur.



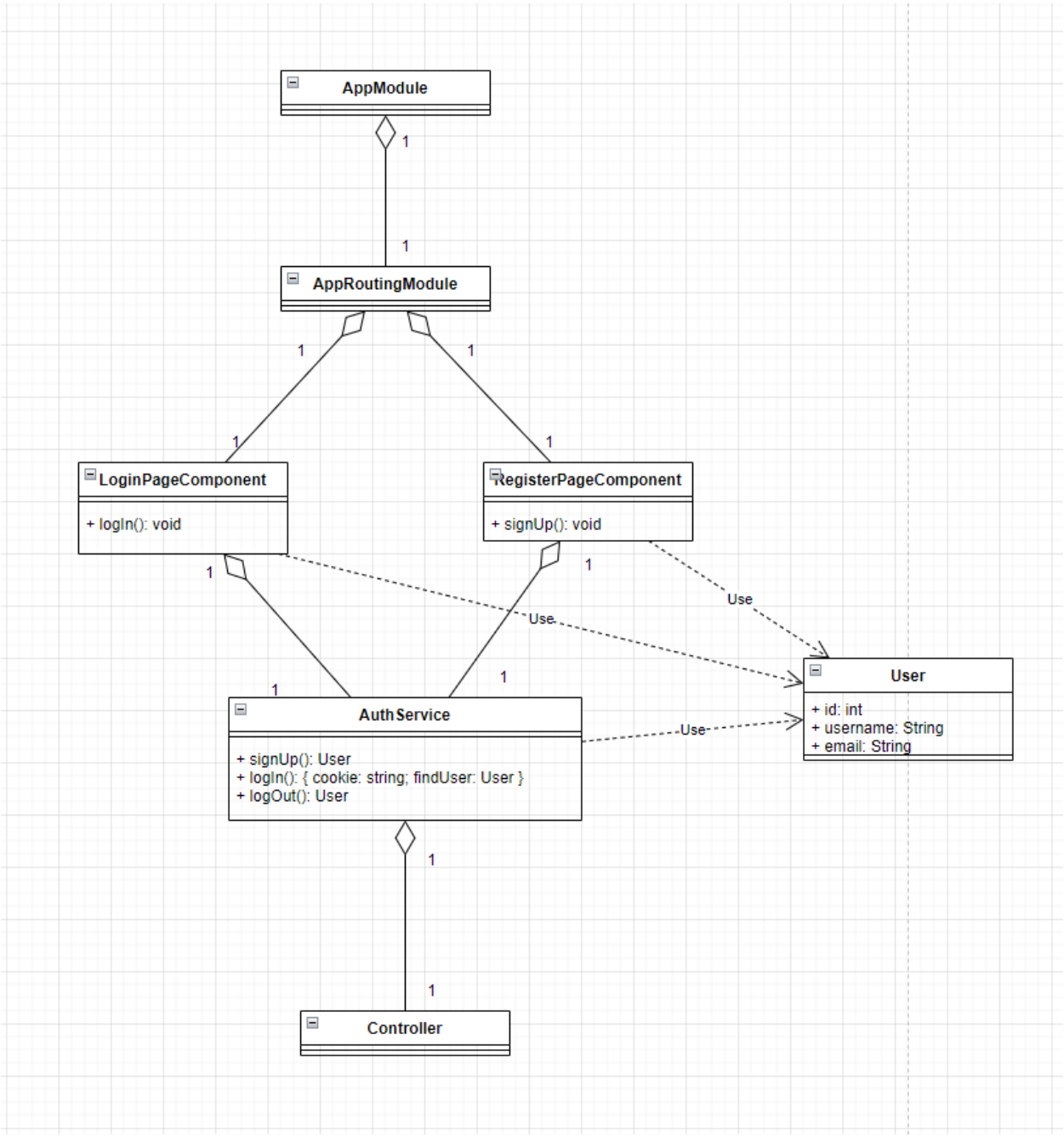
[Figure 9] Diagramme de Classe pour le Chat

Chat
Ce package s'occupe de la communication entre les joueurs et de garder l'historique de cette communication du côté serveur.



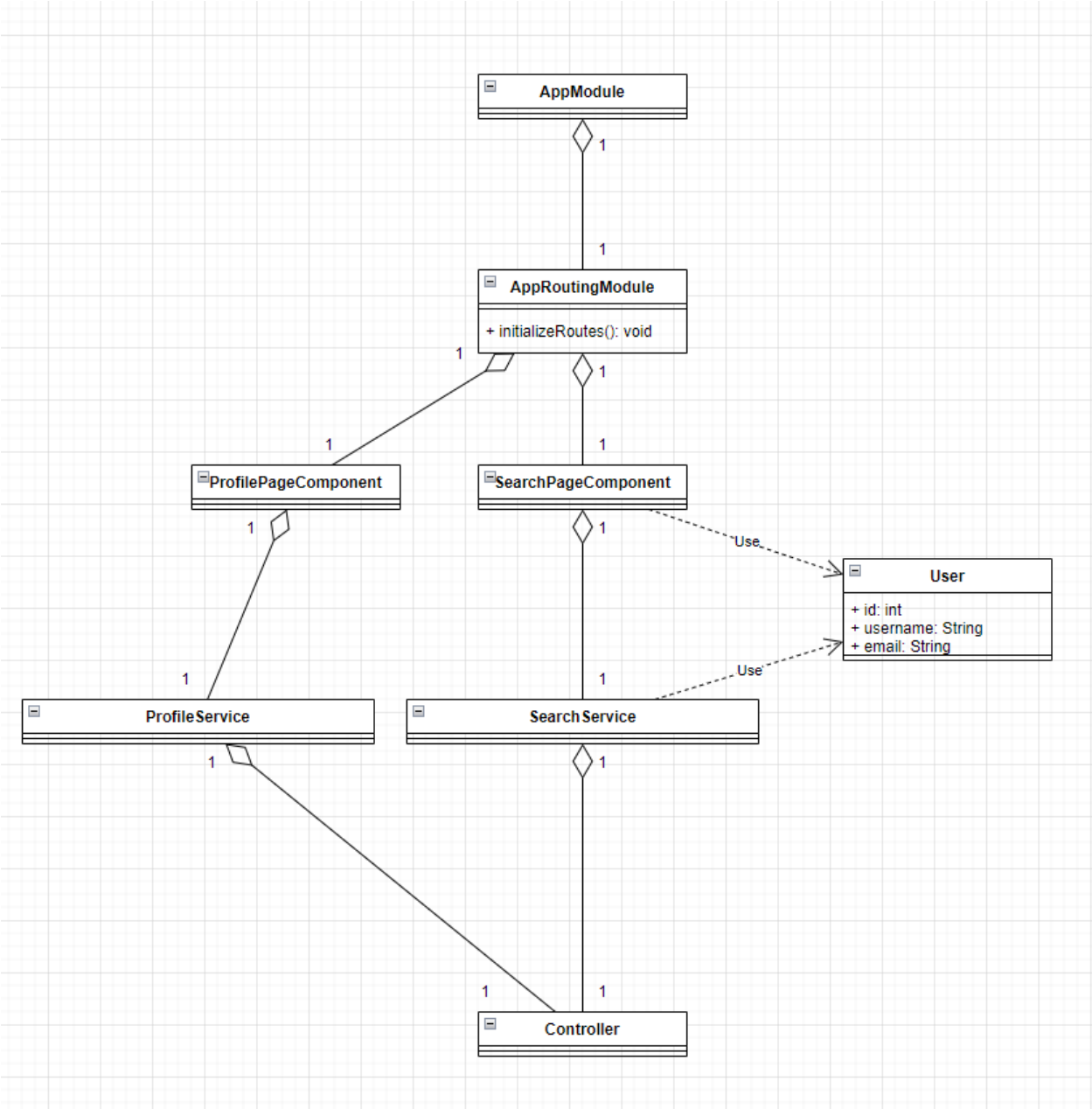
[Figure 10] Diagramme de Classe pour la game

Game
Ce package s'occupe de toute la logique du jeu de scrabble et des sockets utilisés pour communiquer avec les différents clients.



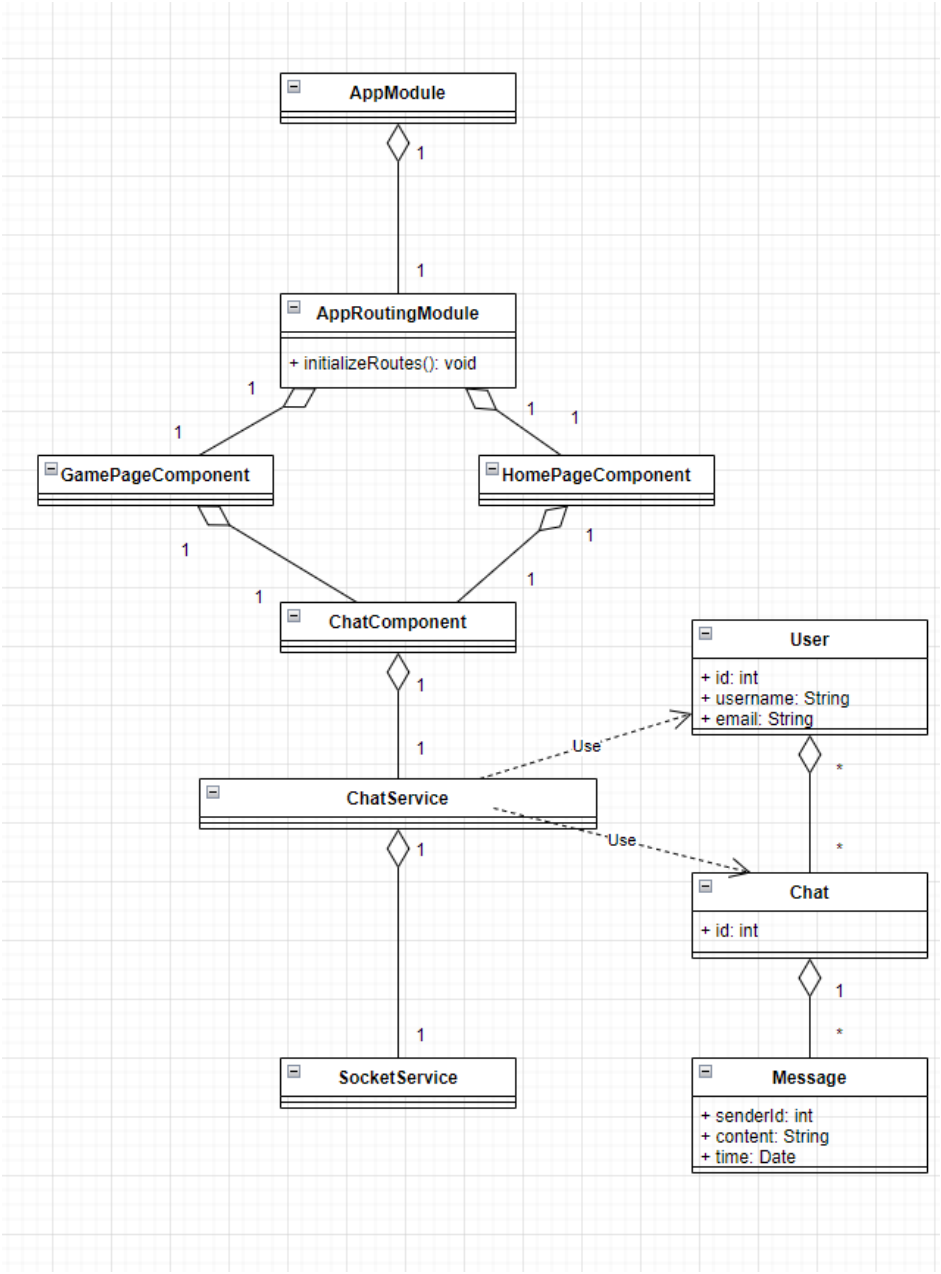
[Figure 11] Diagramme de Classe pour l'Authentification

Authentification
Ce package s'occupe de d'afficher les différentes pages et la communication avec le serveur pour tout ce qui touche à l'authentification des utilisateurs sur le client lourd.



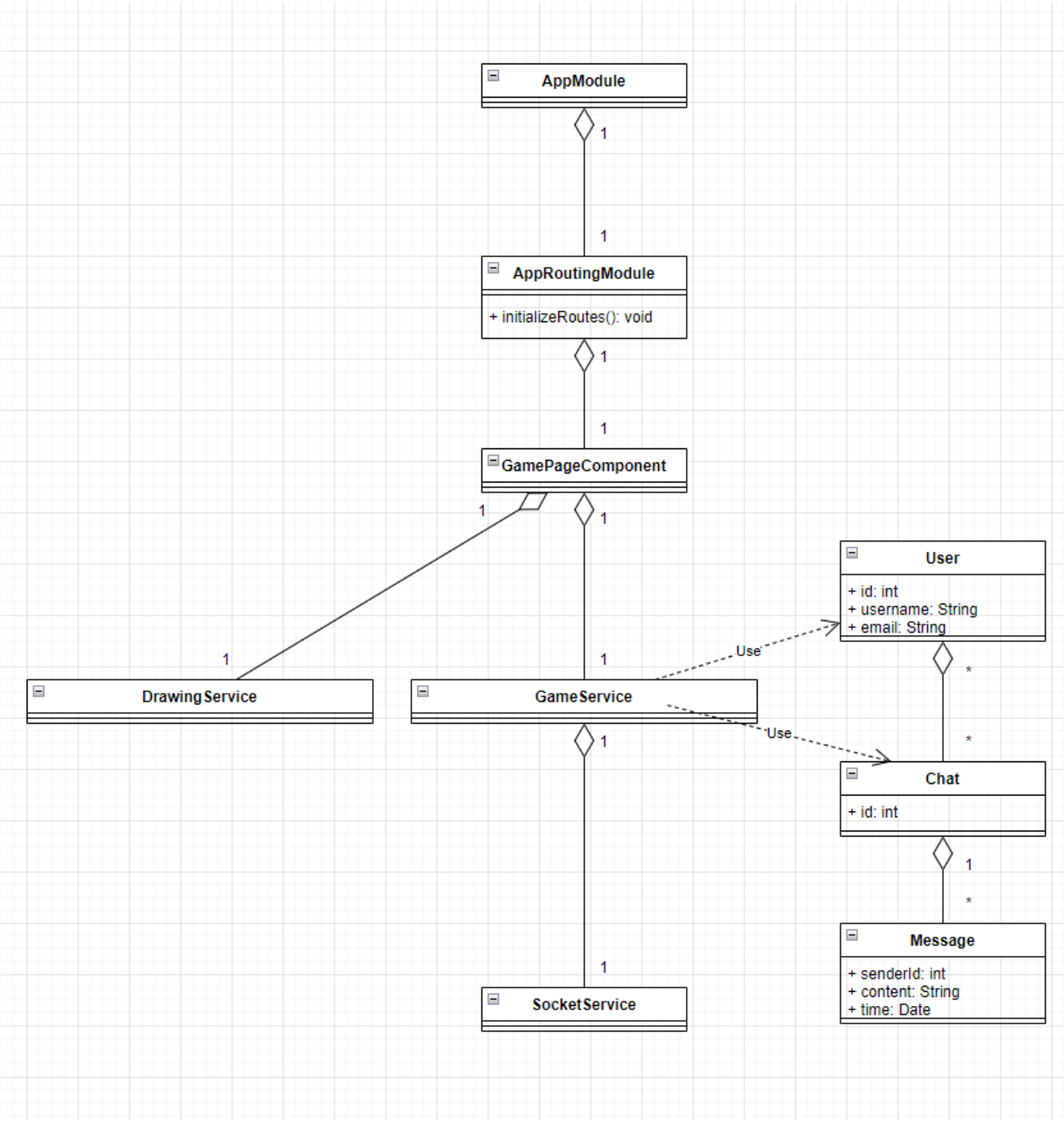
[Figure 12] Diagramme de Classe pour le profile

Profile
Ce package s'occupe de d'afficher les différentes pages et la communication avec le serveur pour tout ce qui touche à la recherche d'utilisateurs et des profils utilisateurs sur le client lourd.



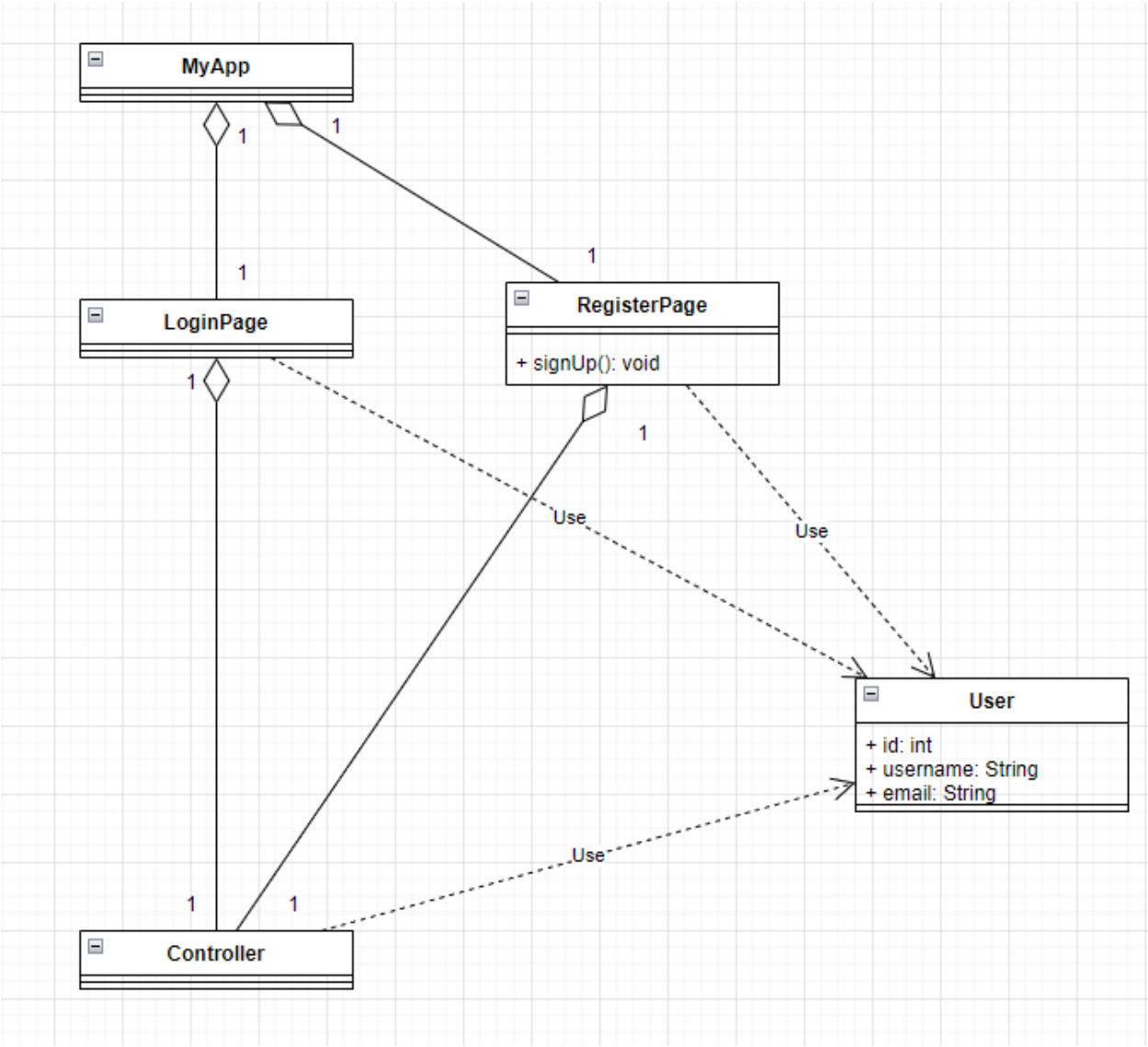
[Figure 13] Diagramme de Classe pour le chat

Chat
Ce package s'occupe d'afficher la messagerie et de communiquer avec le serveur au sujet de la messagerie sur le client lourd.



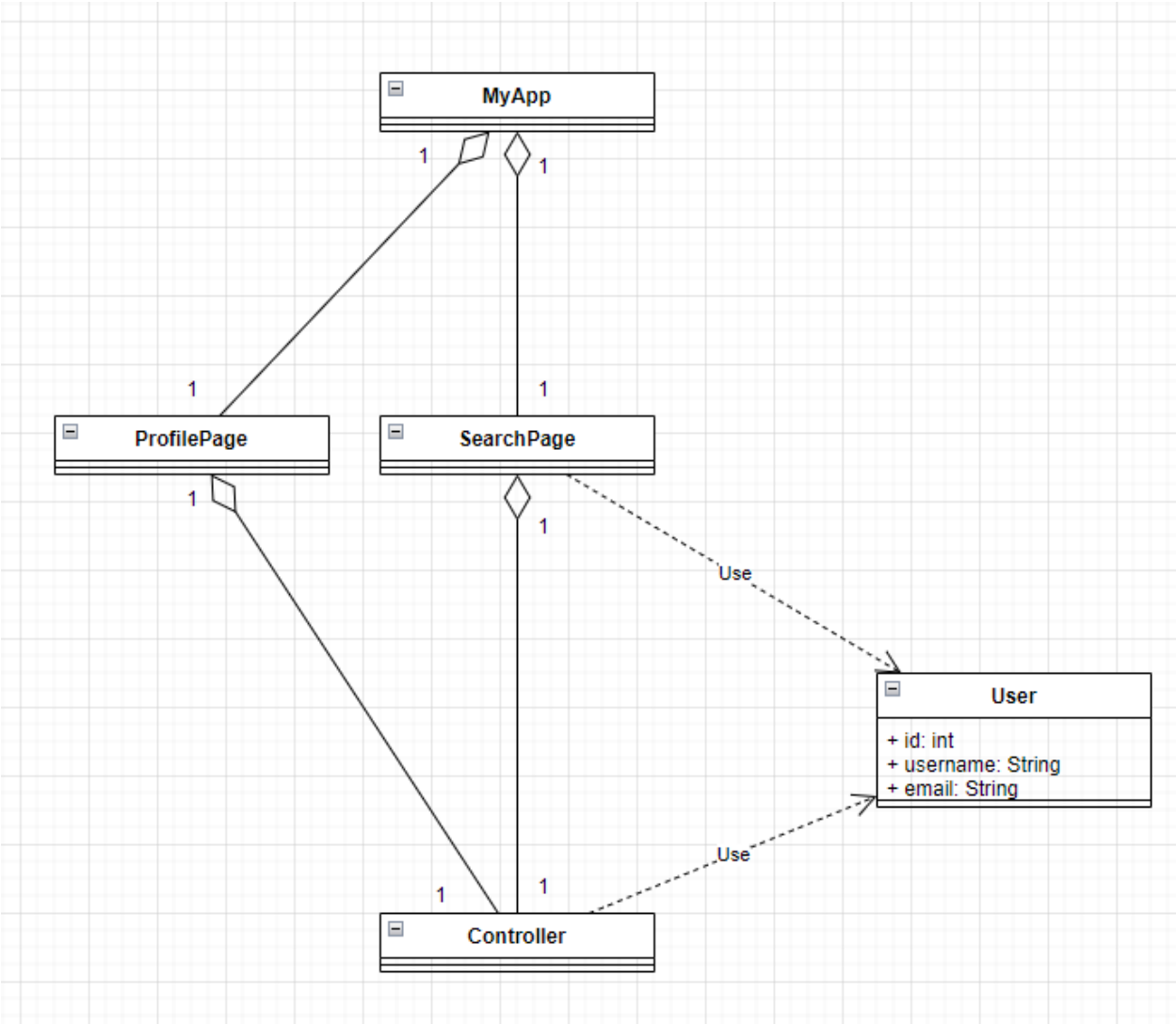
[Figure 14] Diagramme de Classe pour la game

Game
Ce package s'occupe de d'afficher le plateau et de la communication avec le serveur lors des parties sur le client lourd.



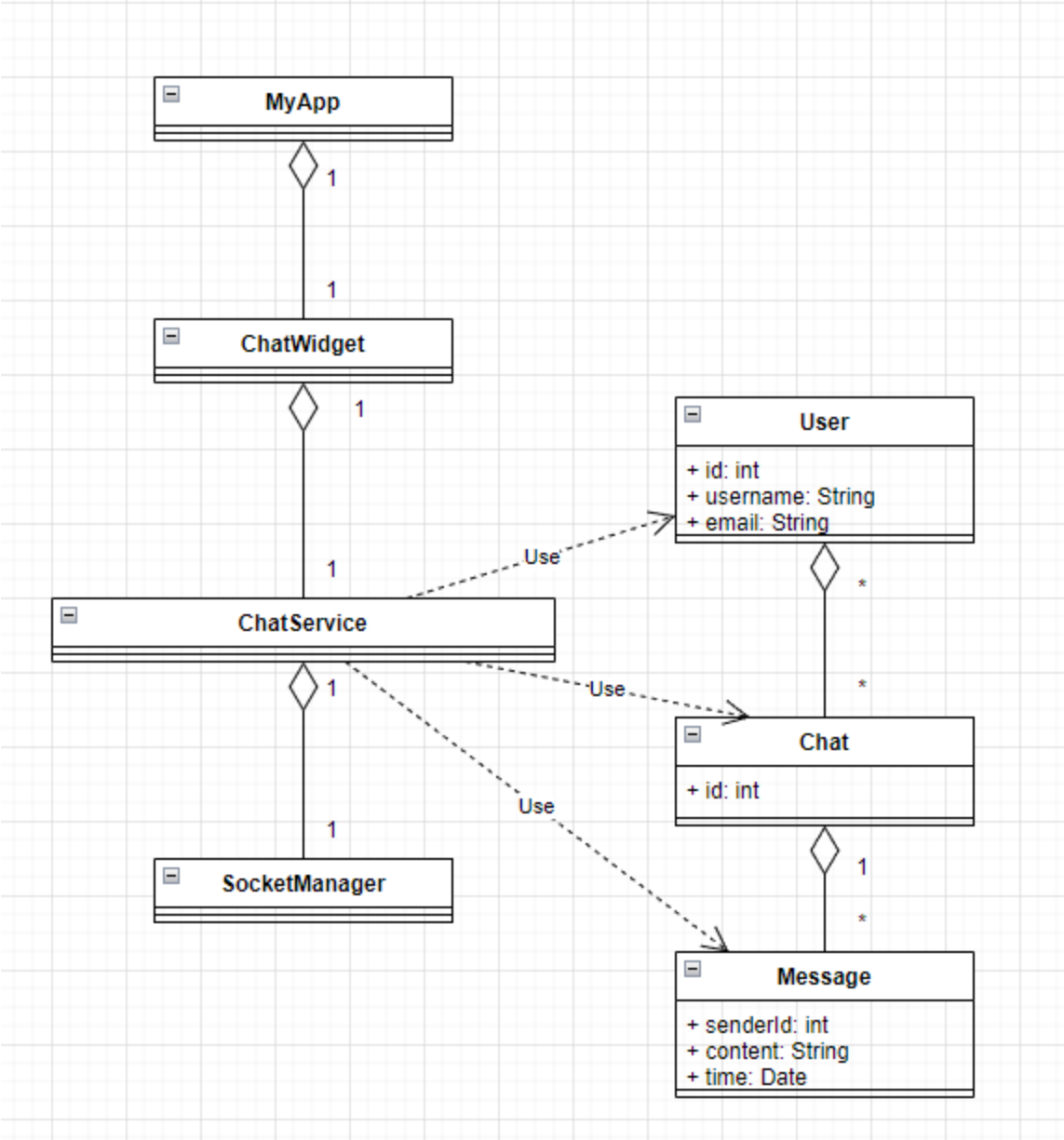
[Figure 15] Diagramme de Classe pour l'Authentification

Authentification
Ce package s'occupe de d'afficher les différentes pages pour s'authentifier et de la communication avec le serveur pour tout ce qui touche à l'authentification des utilisateurs sur le client léger.



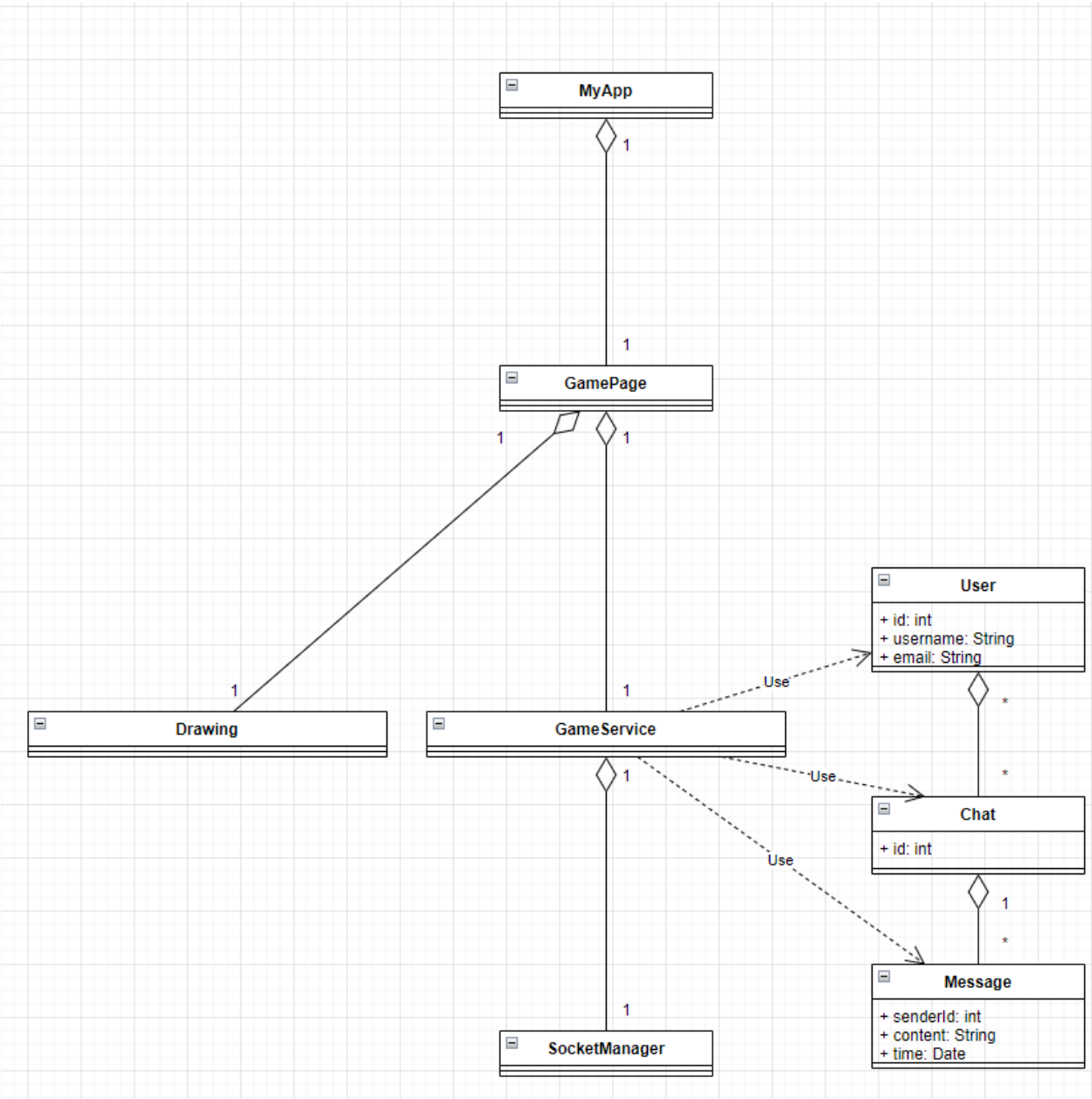
[Figure 16] Diagramme de Classe pour le profile

Profile
Ce package s'occupe de d'afficher les différentes pages pour s'authentifier et de la communication avec le serveur pour tout ce qui touche à l'authentification des utilisateurs sur le client léger.



[Figure 17] Diagramme de Classe pour le chat

Chat
Ce package s'occupe d'afficher la messagerie et de communiquer avec le serveur lorsque l'utilisateur est dans la messagerie du client léger.

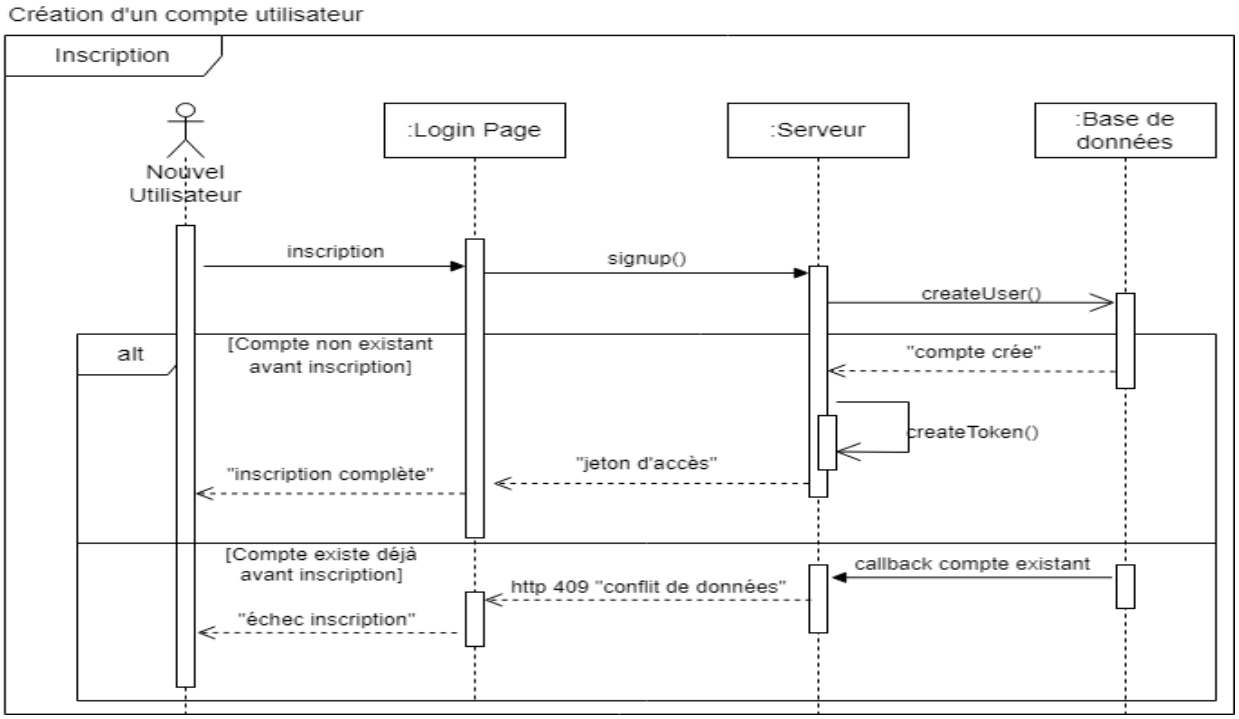


[Figure 18] Diagramme de Classe pour la game

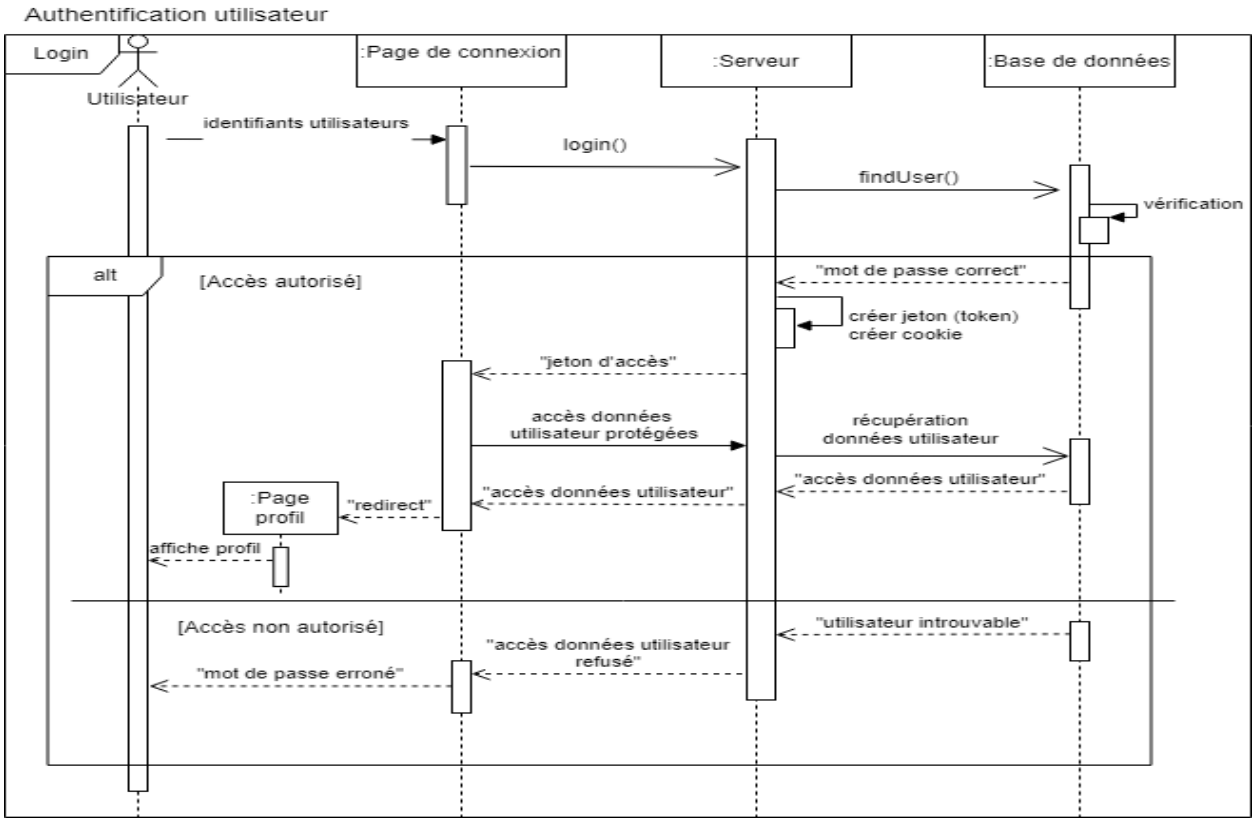
Game
Ce package s'occupe de d'afficher le plateau et de la communication avec le serveur lors des parties sur le client léger.

5. Vue des processus

Processus no. 1: Création du compte de l'utilisateur

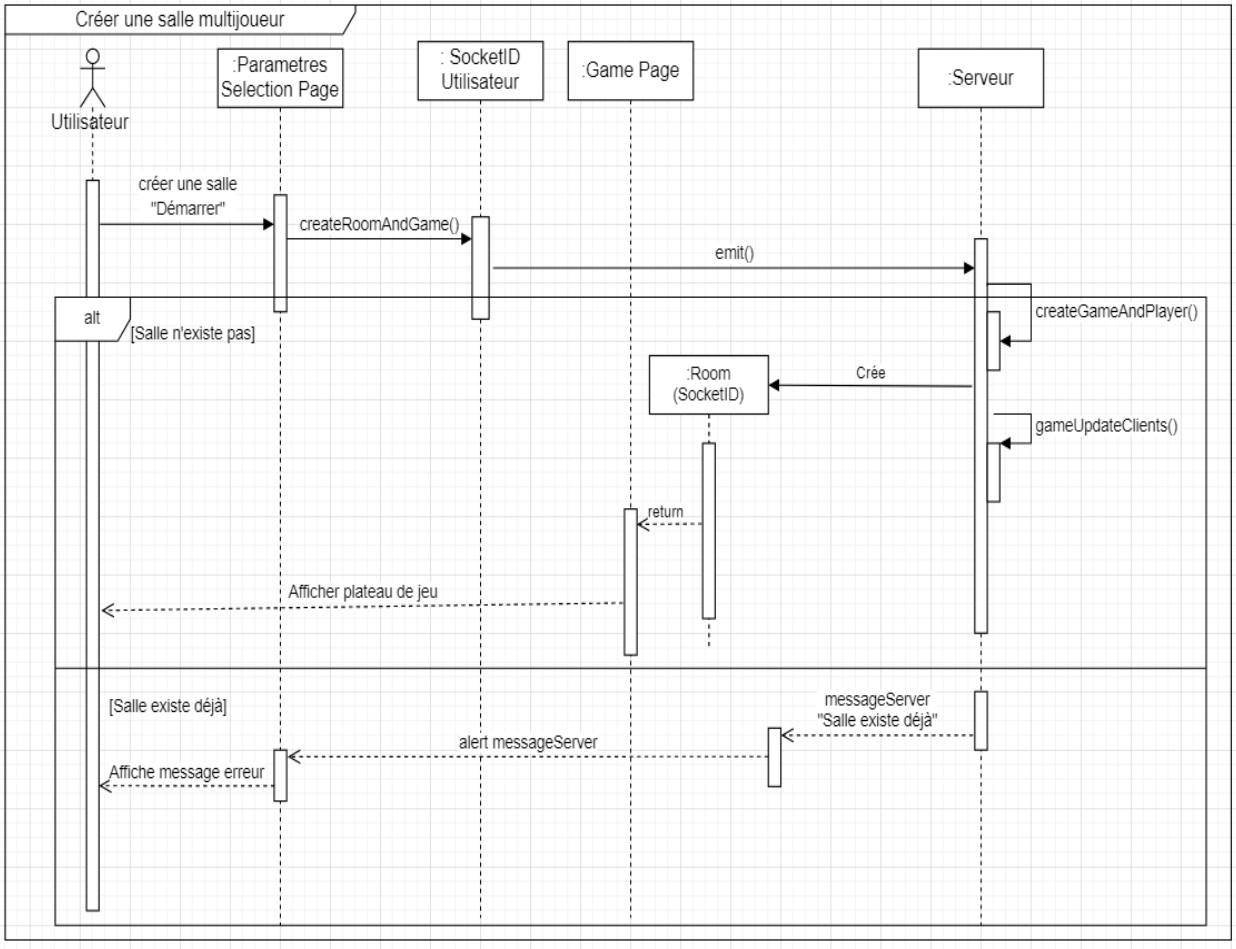


Processus no. 2: Authentification de l'utilisateur

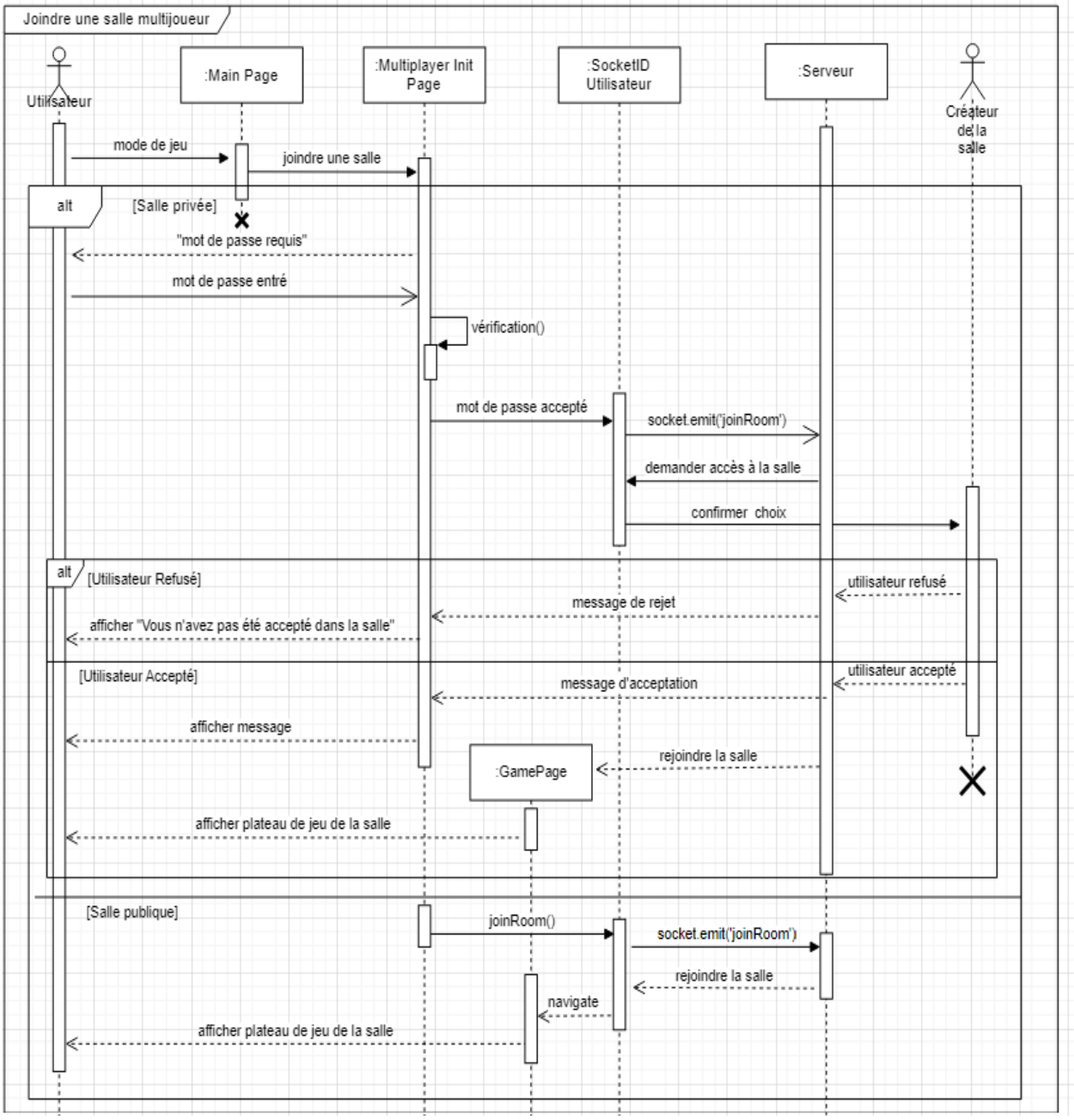


Pour le reste des processus, nous allons assumer que l'utilisateur est déjà connecté à son profil:

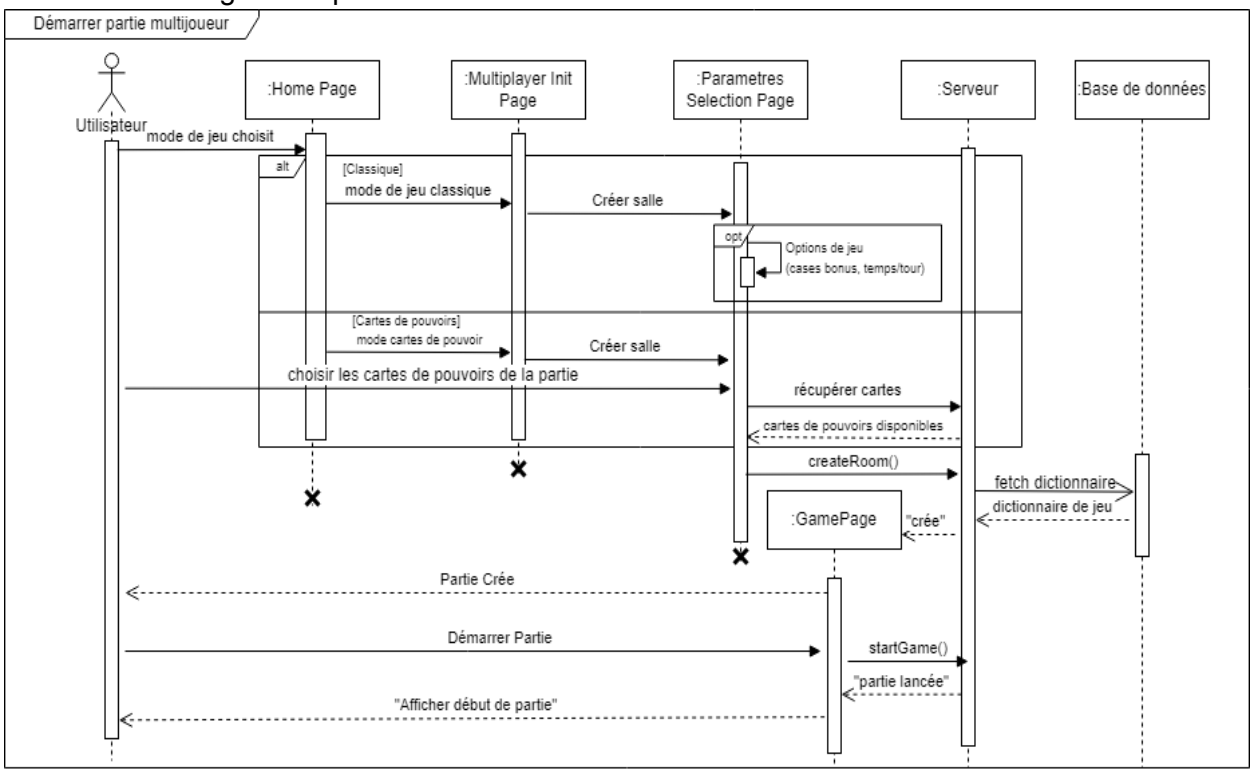
Processus no. 3: Création d'une salle multijoueur



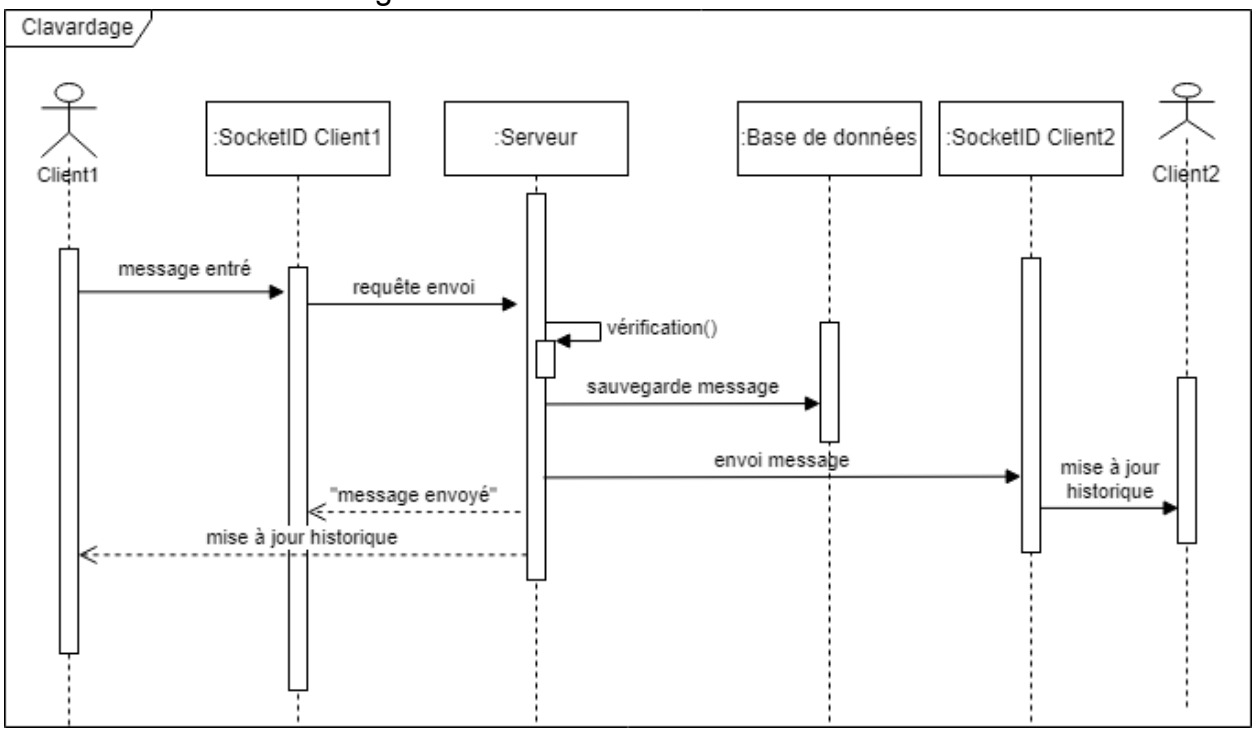
Processus no. 4: Joindre une salle multijoueur (privée ou publique)



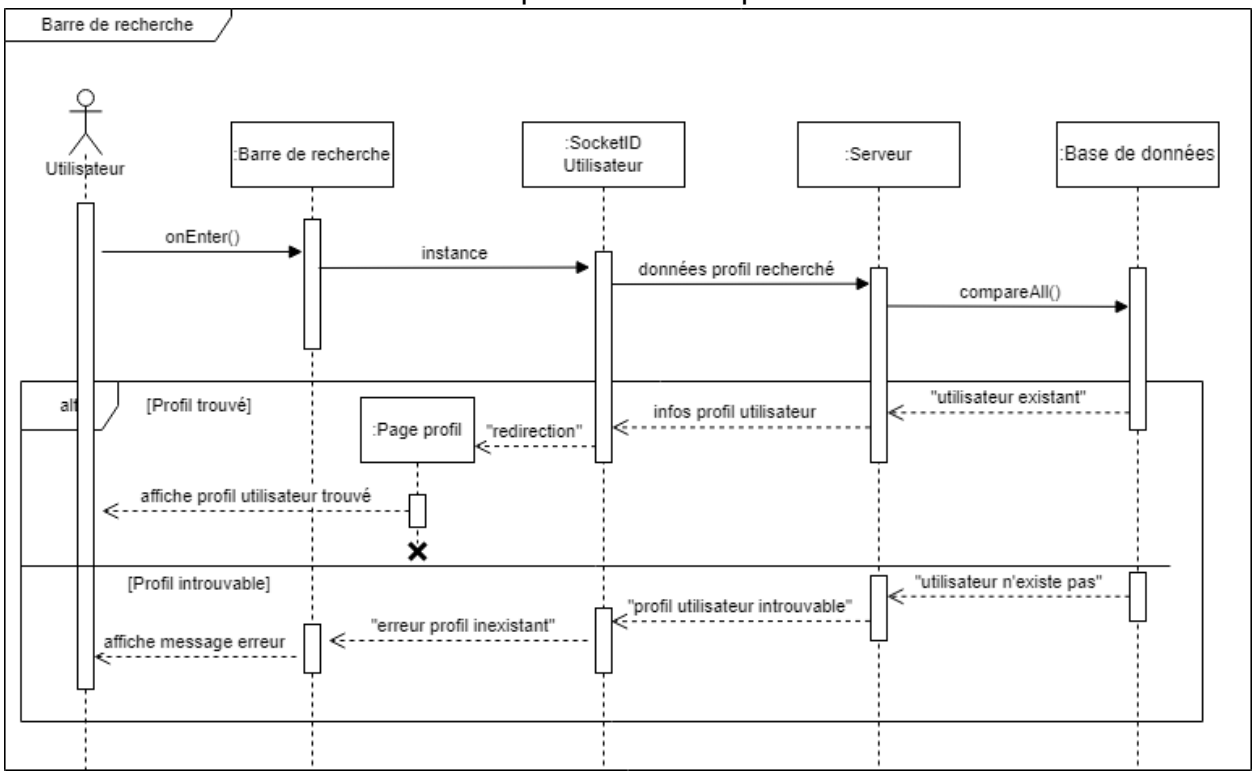
Processus no. 5: Démarrer une partie multijoueur.
Dans ce cas précis, nous allons assumer qu'au moins un autre joueur réel a rejoint la salle avant le démarrage de la partie.



Processus no. 6: Clavardage entre les deux clients sur un serveur

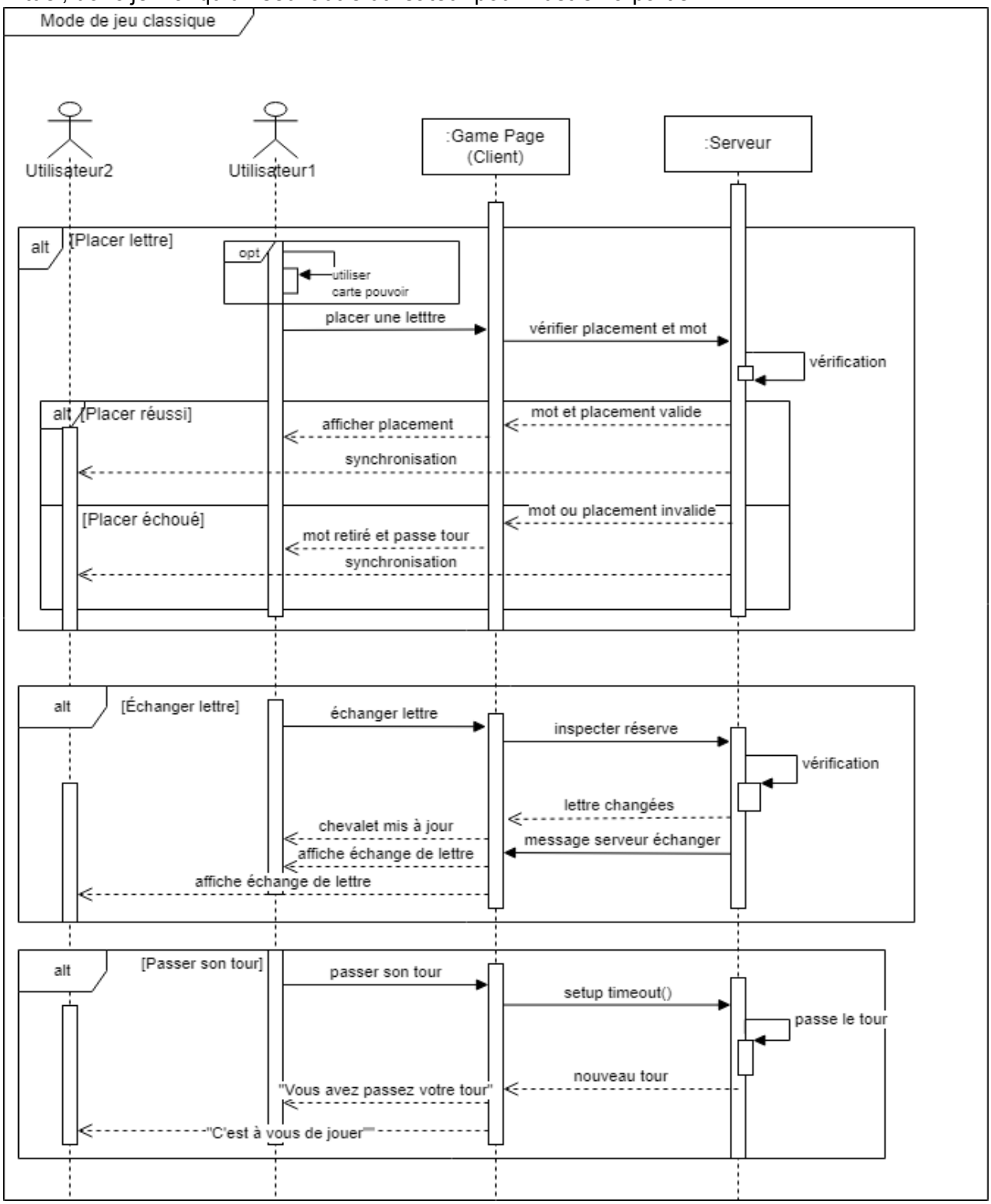


Processus no.7: Barre de recherche pour afficher le profil d’autres utilisateurs

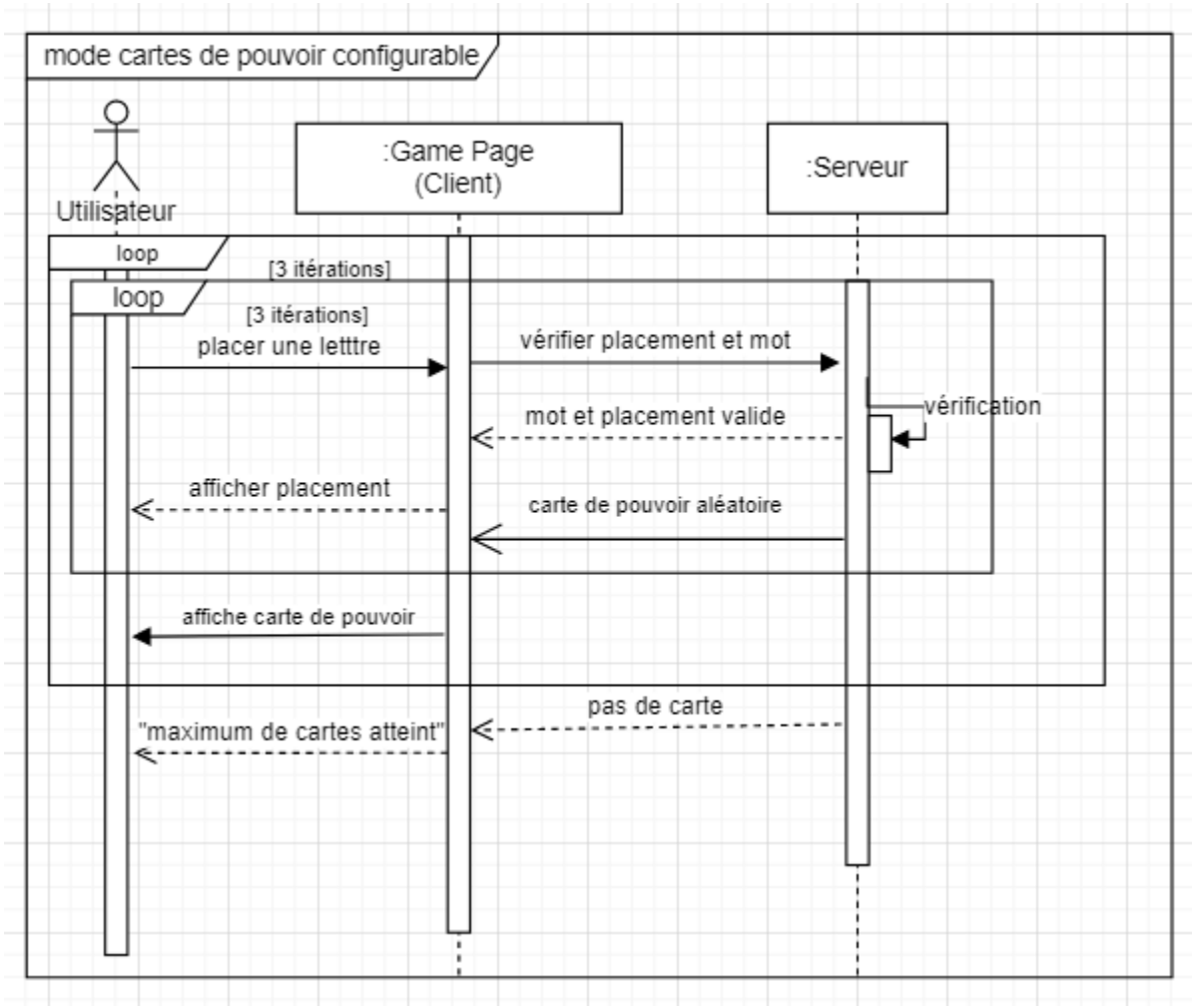


Processus no. 8: Partie mode classique à 4 joueurs

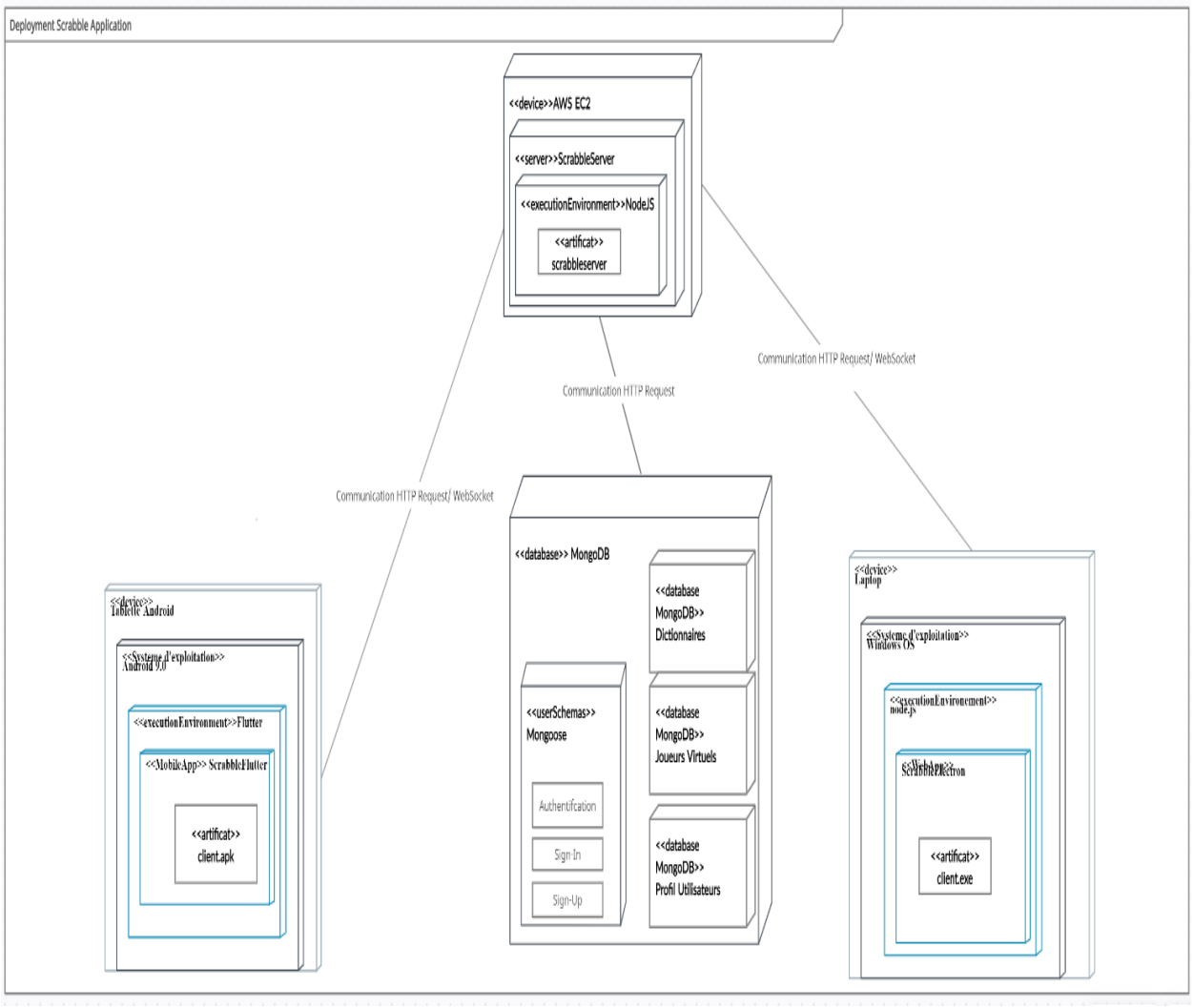
Ici, il n'y a aucune différence entre un joueur réel et un joueur virtuel. Nous décrivons les séquences de jeu une fois. À ce stade, il n'y a pas de différence entre un utilisateur réel et virtuel, donc je n'ai qu'un seul autre utilisateur pour illustrer la partie.



Processus no. 9: Mode de jeu cartes de pouvoir
Séquence d'acquisition des cartes au cours de la partie



6. Vue de déploiement



[Figure] Diagramme de déploiement pour notre serveur, base de données et deux clients

7. Taille et performance

7.1 Serveur

Notre instance de EC2 est de type t2.micro, avec un 3.3 GHz Intel Xeon, 1GB de RAM et un stockage de 20 GB.

7.2 Client Léger

Le client léger sera un apk de taille d'environ 30 Mb, avec une projection de 250 Mo de mémoire vive nécessaire pour rouler l'application.

7.3 Client Lourd

Le client lourd, dû à nature d'être bundle avec une instance de chrome, sera assez large d'une taille prévue d'autour 150 MB, compatible avec Windowes 10 et 11, avec 300 MB d'utilisation de mémoire vive nécessaire.

7.4 Base de Donne

Notre base de données sera sur une instance de MongoDB Atlas, avec un cluster de taille de 512 MB.