Équipe 102

PolyScrabble
Document d'architecture logicielle

Version 1.7

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
2022-11-26	1.5	Modifications des diagrammes de cas d'utilisation	Maximiliano
2022-12-01	1.6	Correction des diagrammes	Équipe 102
2022-12-02	1.7	Mise en page finale	Maximiliano

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.

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 voulons que notre système soit sécurisé et à ce fait, nous avons un système d'authentification et 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 et utiliser des routes protégées avec l’utilisation 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

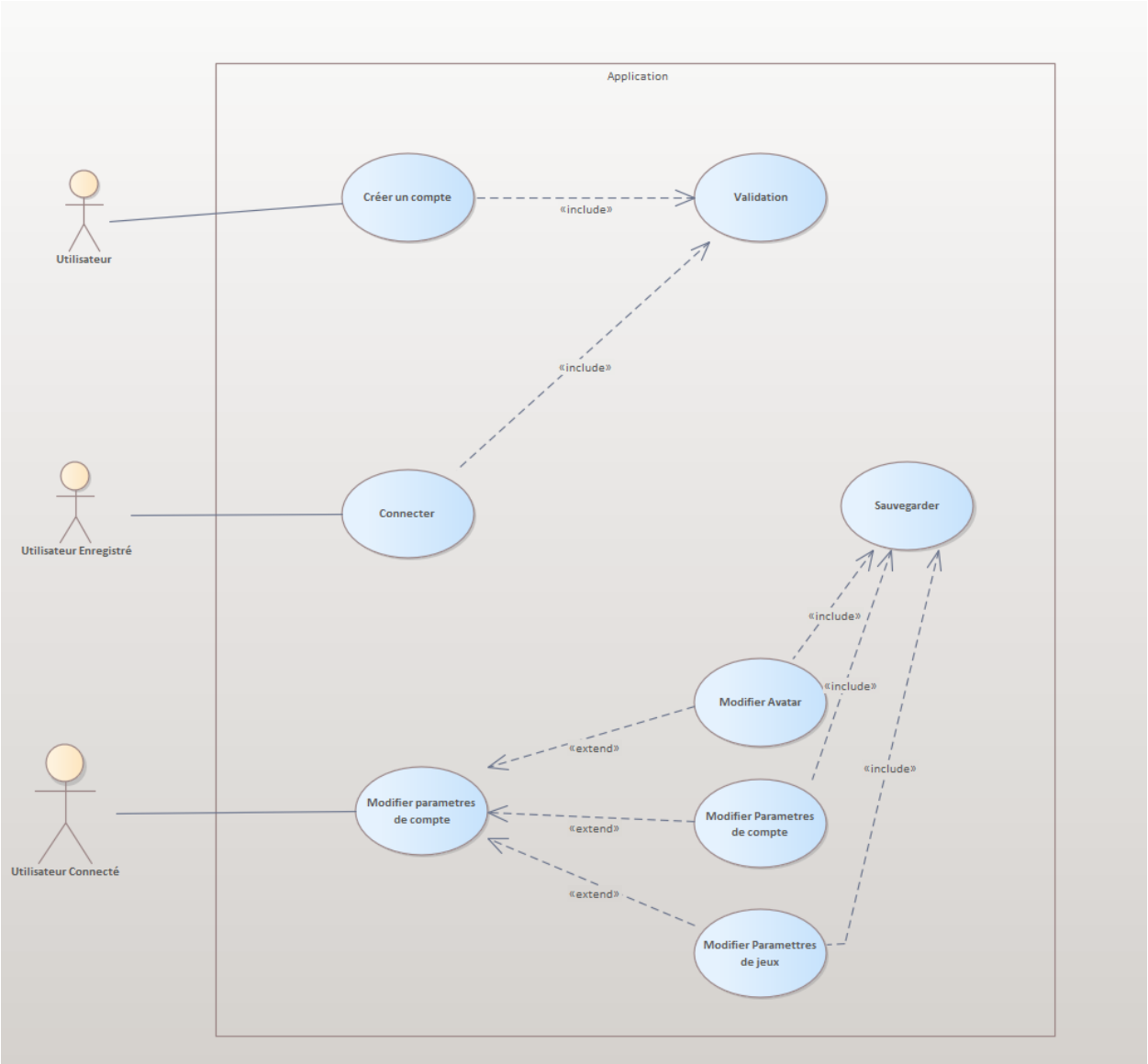
Nous devons pour le client léger développer une application pour la plateforme Android, plus particulièrement, une Samsung Galaxy Tab A 2019. Nous avons choisi d'utiliser le framework Flutter afin de développer notre application.
Notre client léger doit être capable de communiquer avec le serveur et 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

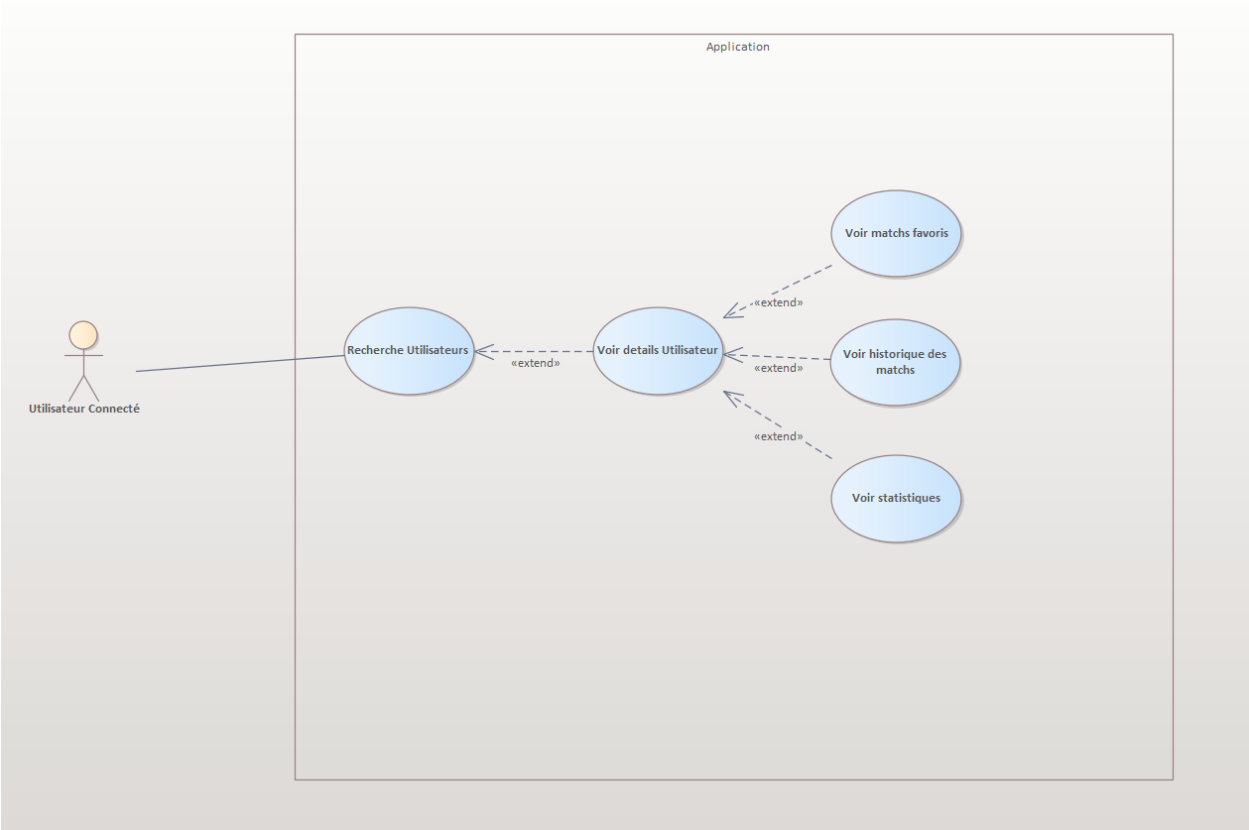
Pour le client lourd, on a choisi le cadriciel ElectronJS ce qui nous permet de réutiliser la plupart du code source de projet 2. Le cadriciel permet de wrapper notre site pour qu'il fonctionne comme exécutable sur n'importe quelle plateforme (Linux, Windows ou MacOs), dans le cadre de ce projet, nous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

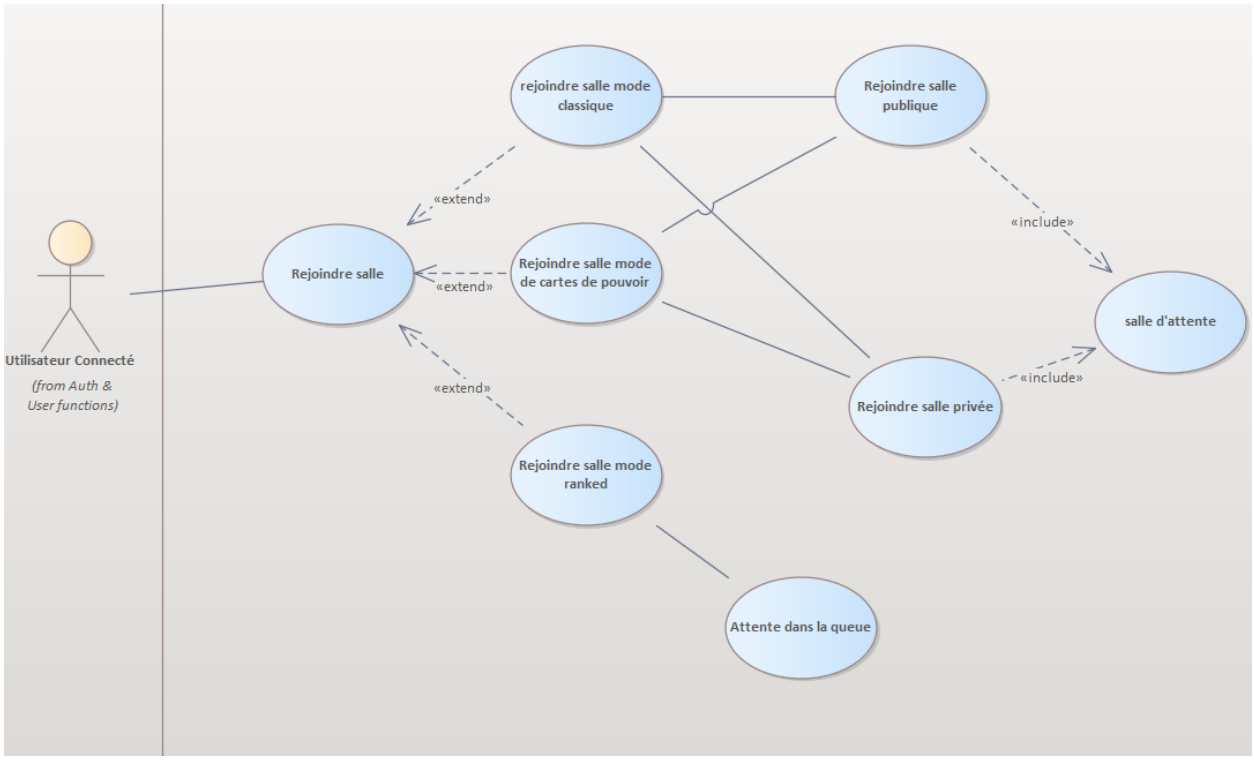
3. Vue des cas d’utilisation



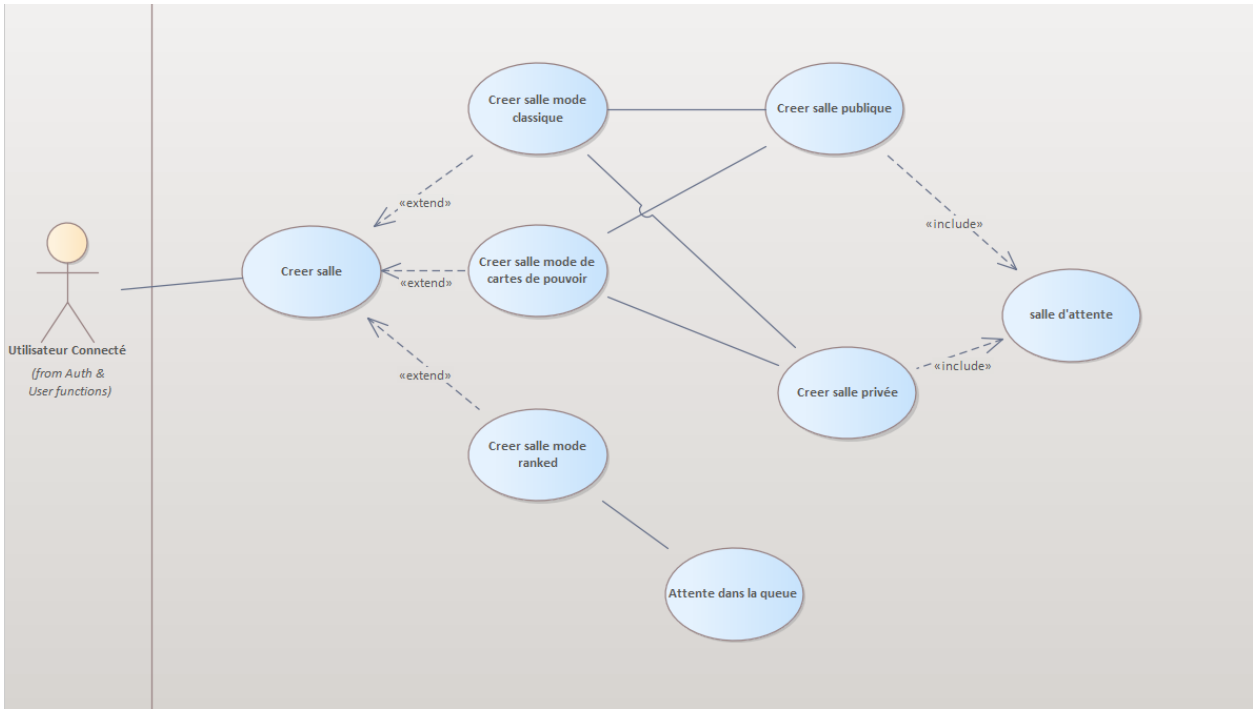
[Figure 1] Cas d’utilisation de l'authentification



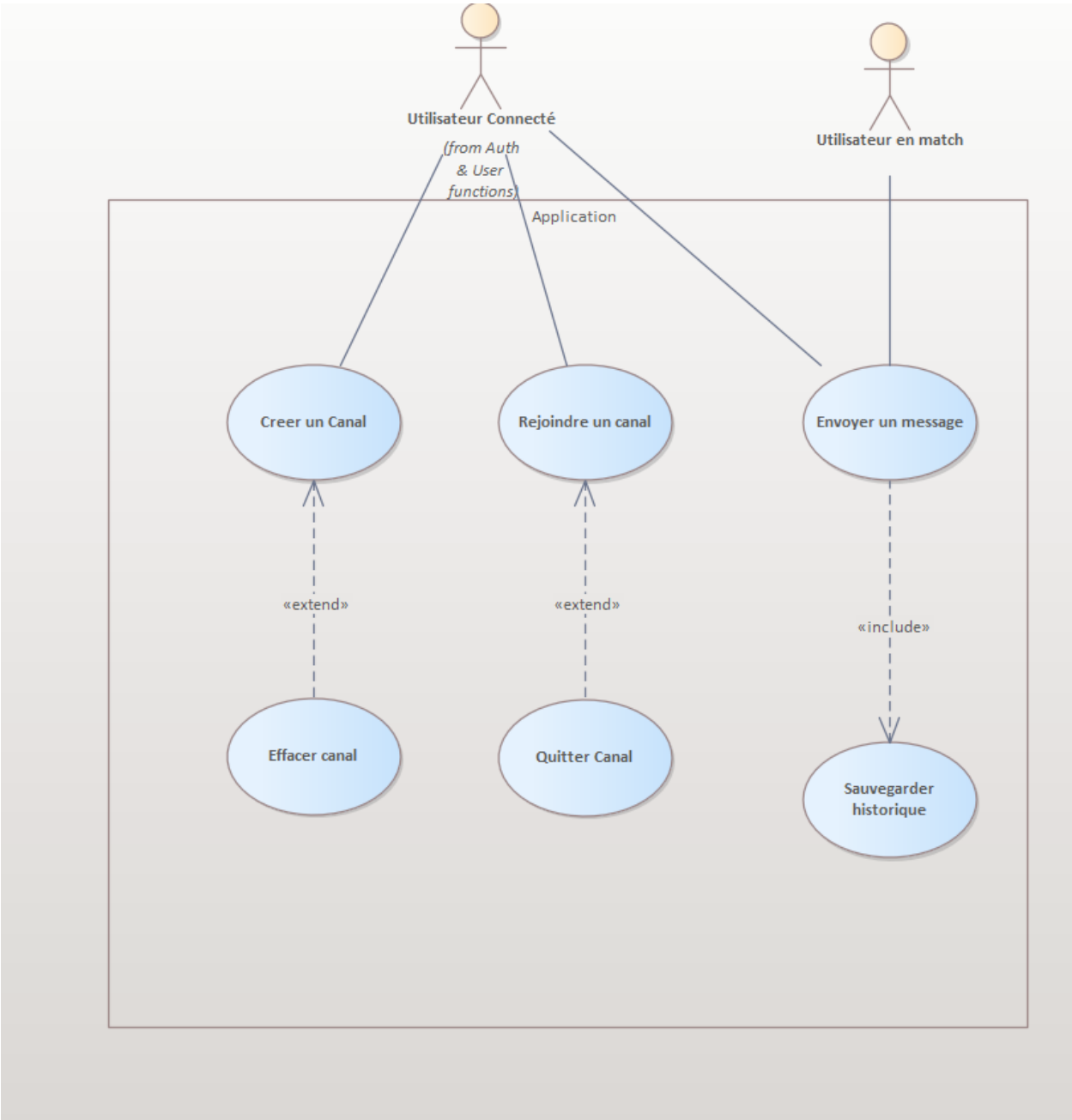
[Figure 2] Cas d'utilisation de la recherche des autres utilisateurs



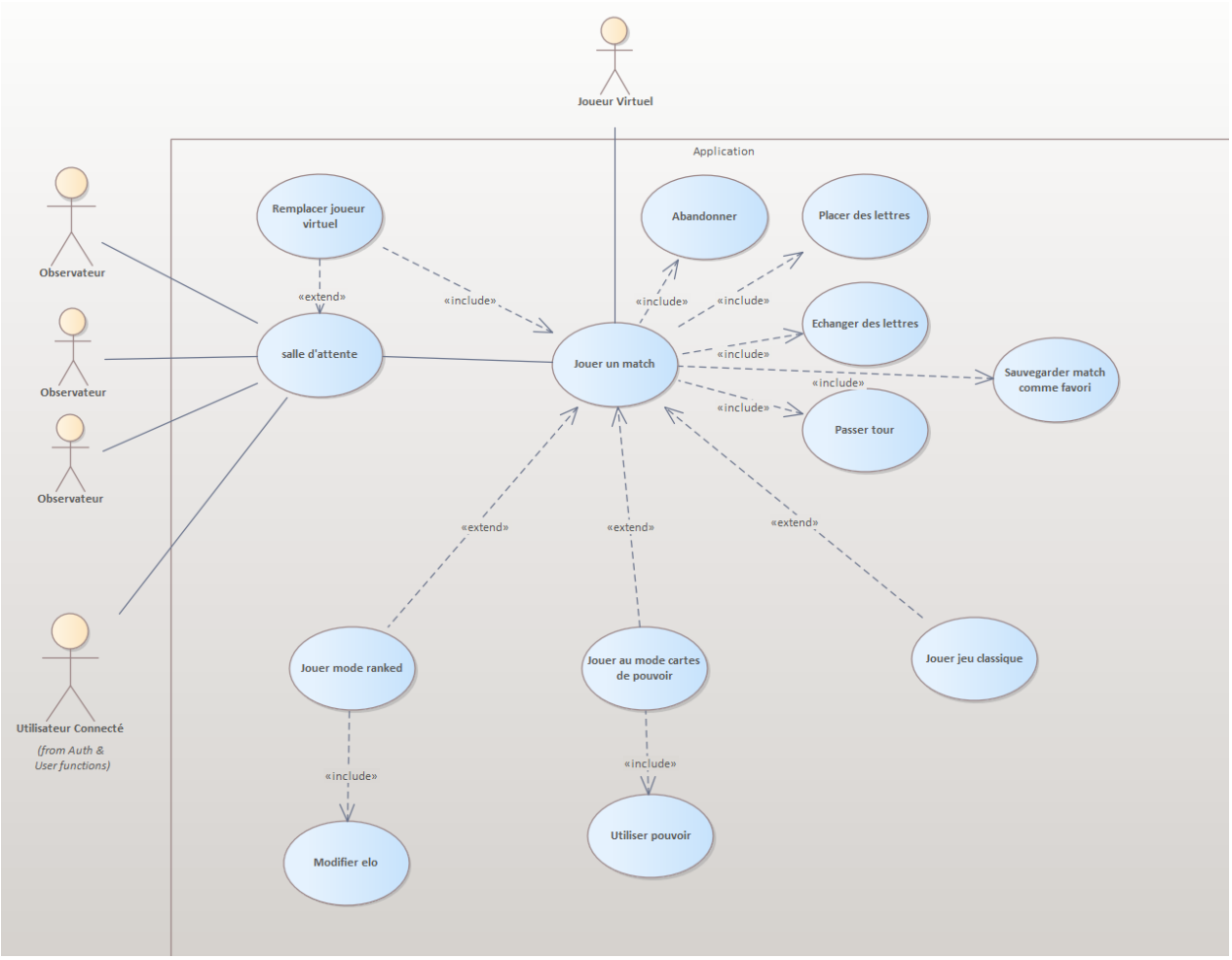
[Figure 3] Rejoindre une salle



[Figure 4] Créer une salle

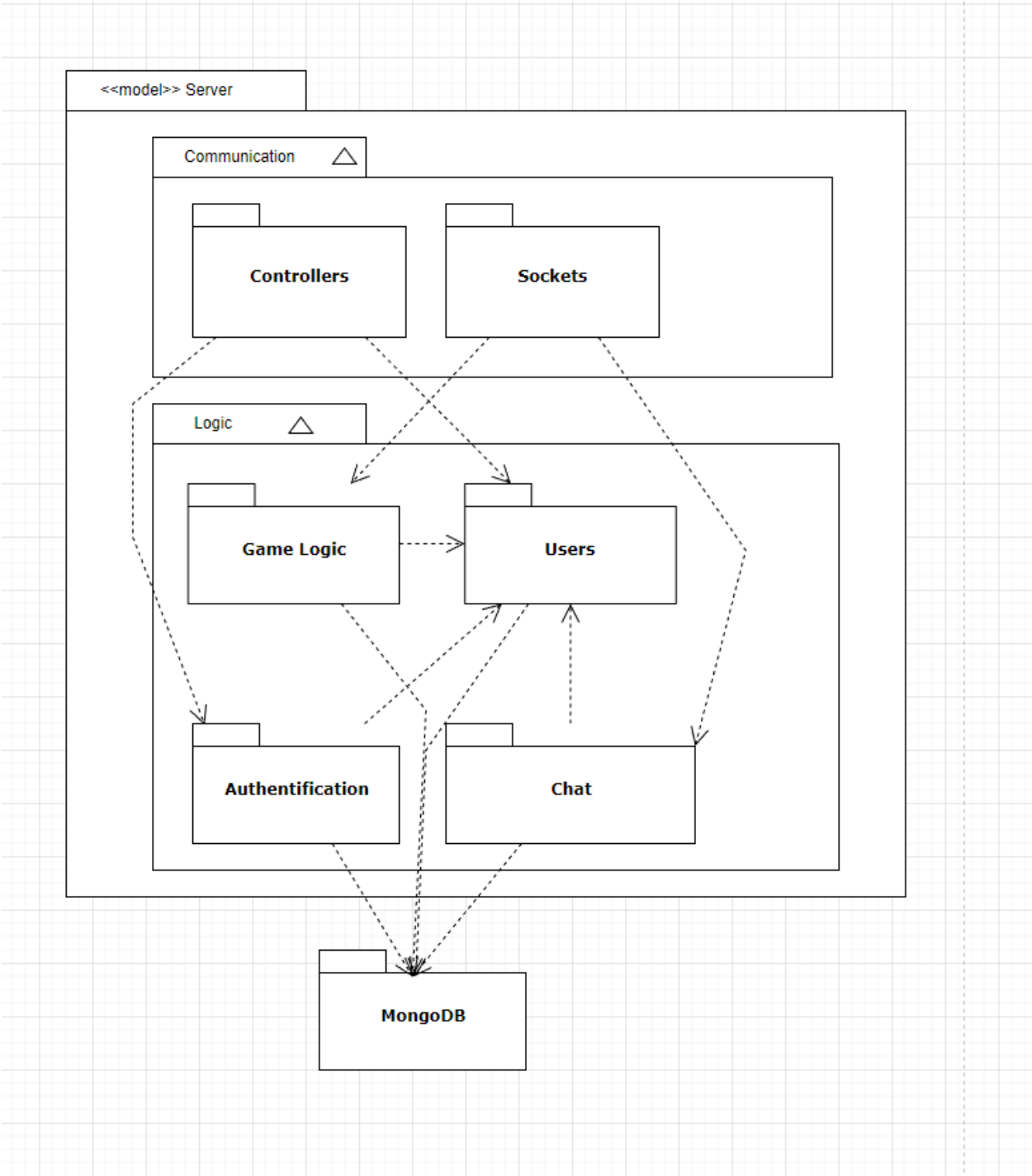


[Figure 5] Clavardage



[Figure 6] Jouer un match

4. Vue logique

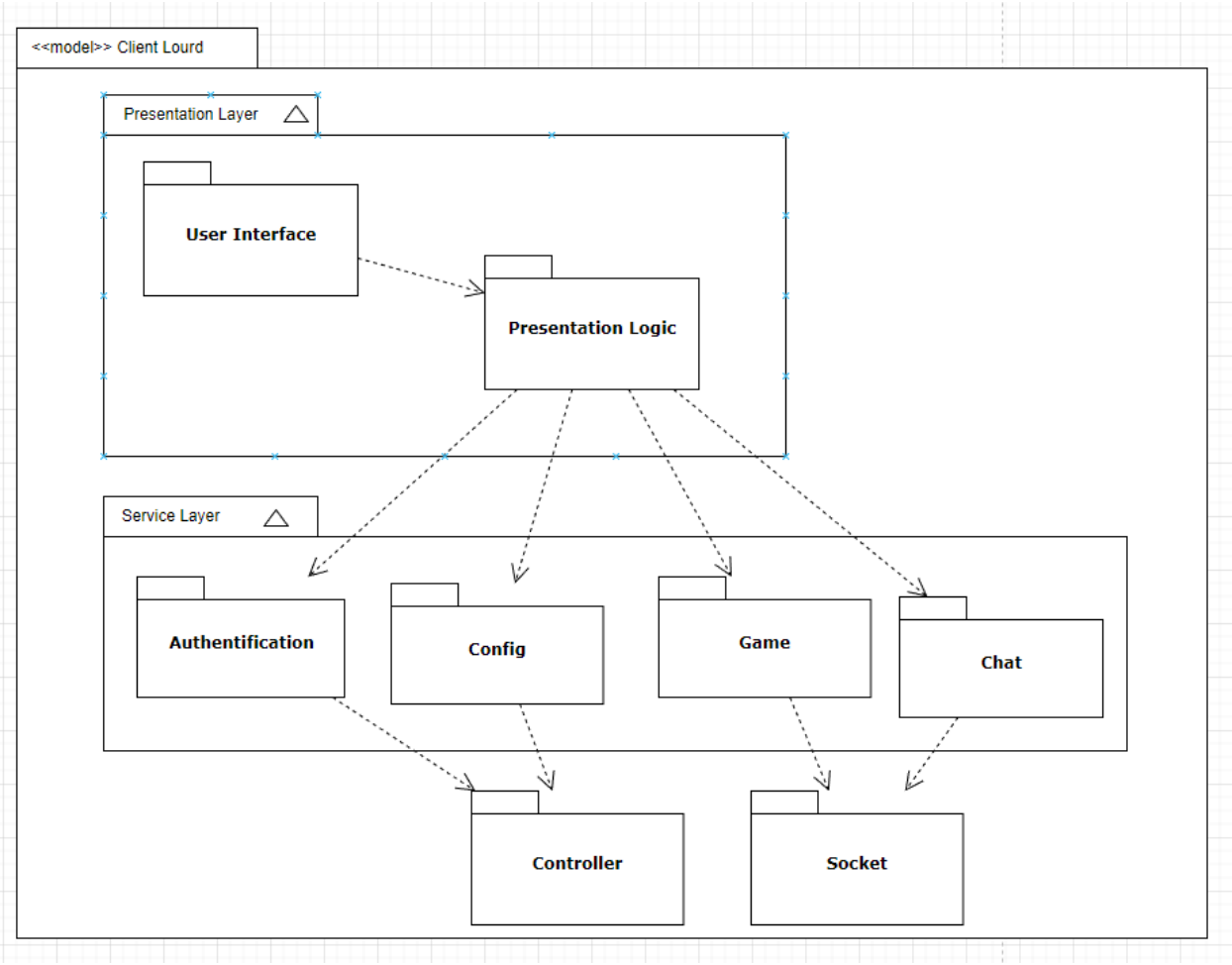


[Figure 7] Diagramme de paquetage pour le serveur

Communication
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.

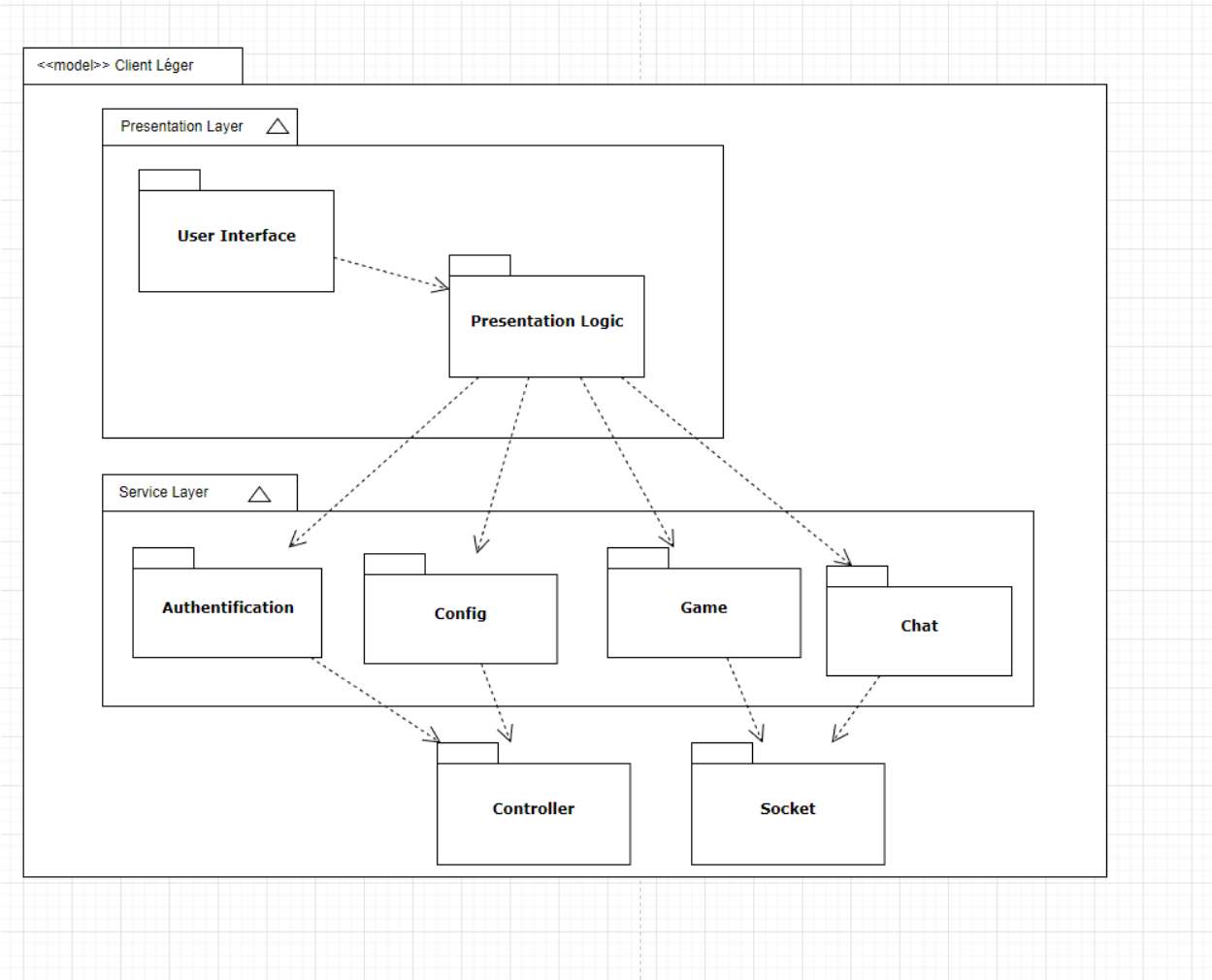


[Figure 8] Diagramme de paquetage pour le client lourd

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 de façon http avec le serveur.



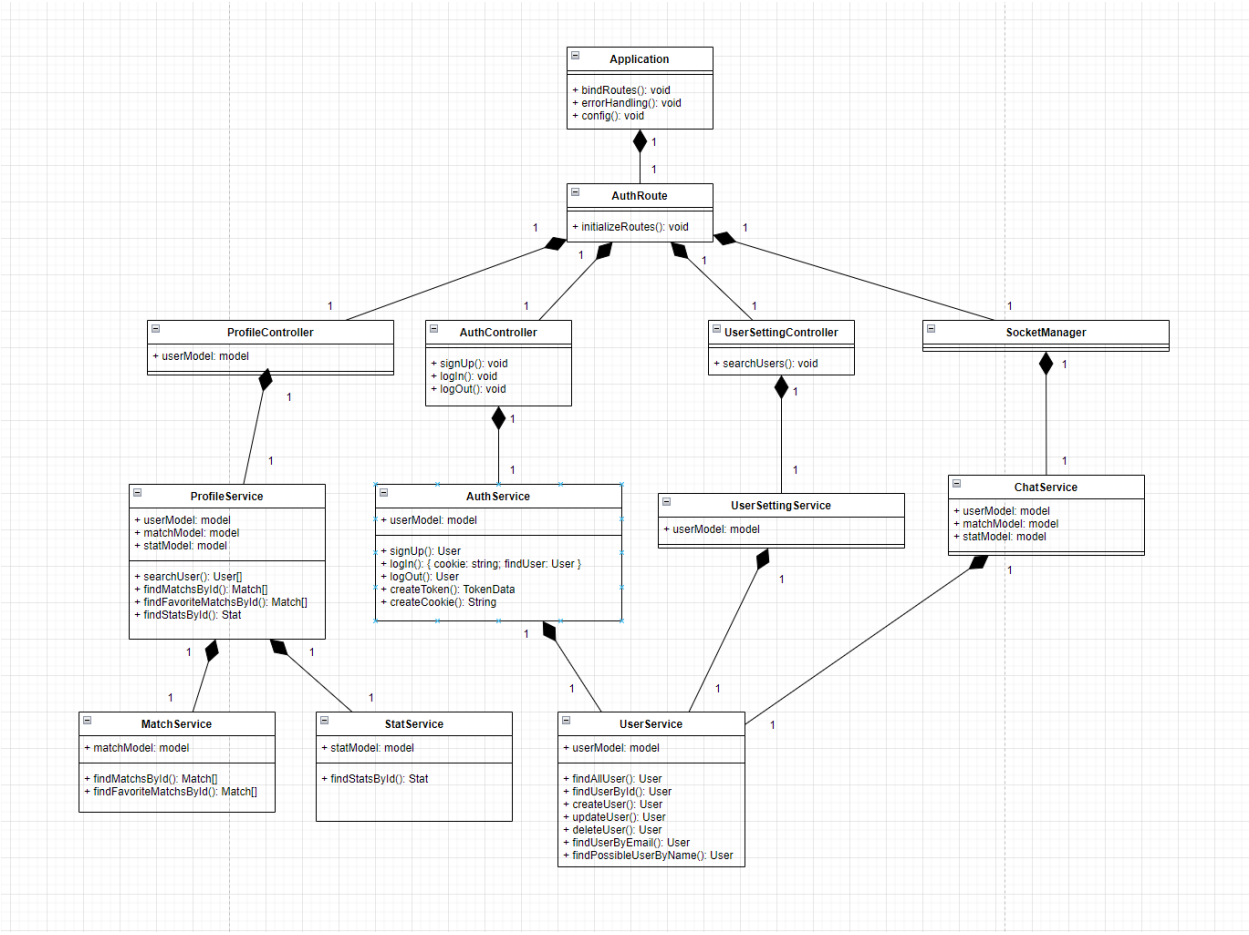
[Figure 9] Diagramme de paquetage pour le client léger

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.



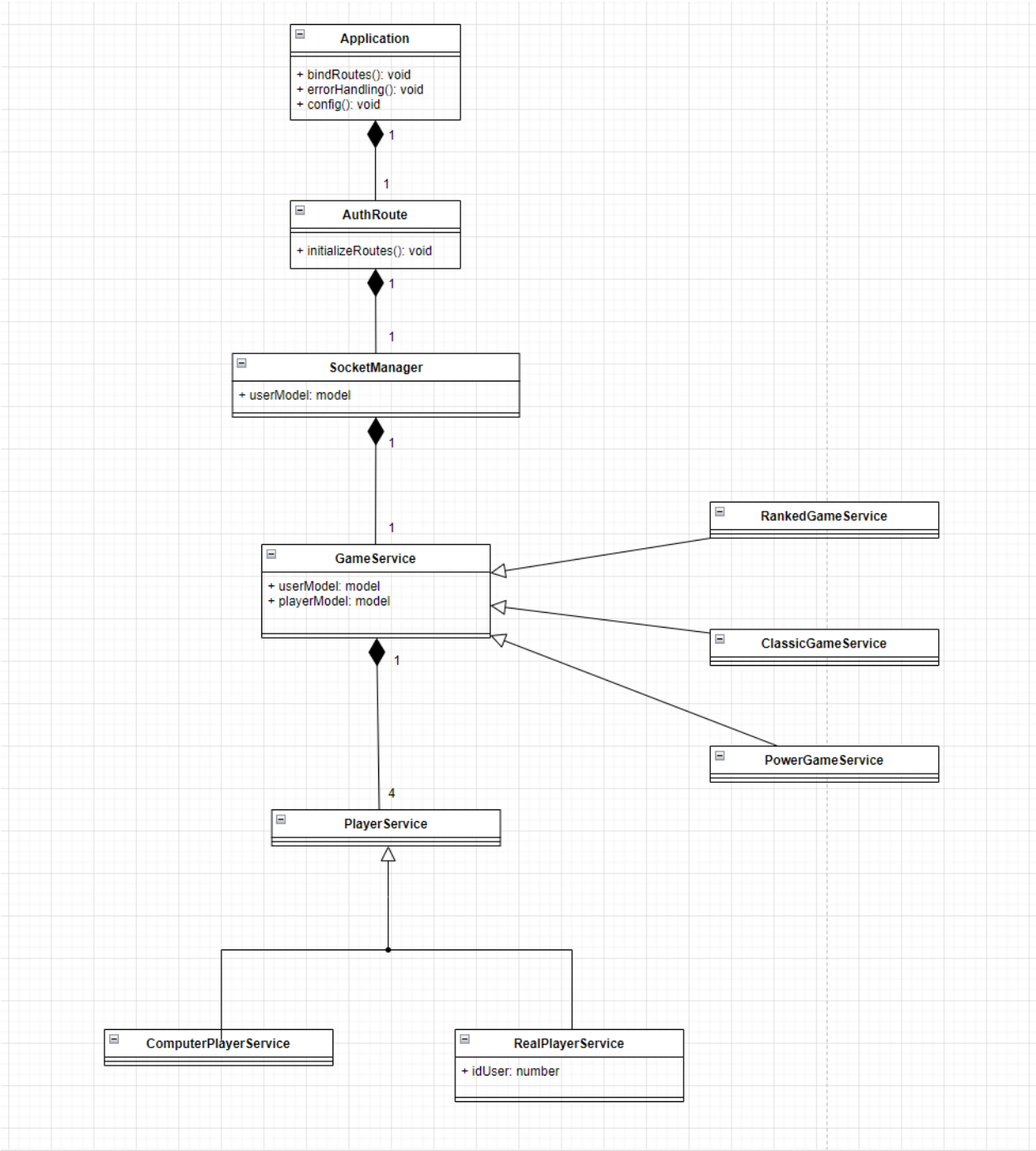
[Figure 10] Diagramme de Classe pour l'Authentification, la gestion des profiles et le chat dans le serveur

AuthController
Cette classe s'occupe de gérer les requêtes pour l'authentification du côté serveur.

ProfileController
Cette classe s'occupe de gérer les requêtes pour la recherche d'utilisateur et de la transmission d'information

nécessaire pour afficher le profil des utilisateurs du côté serveur.

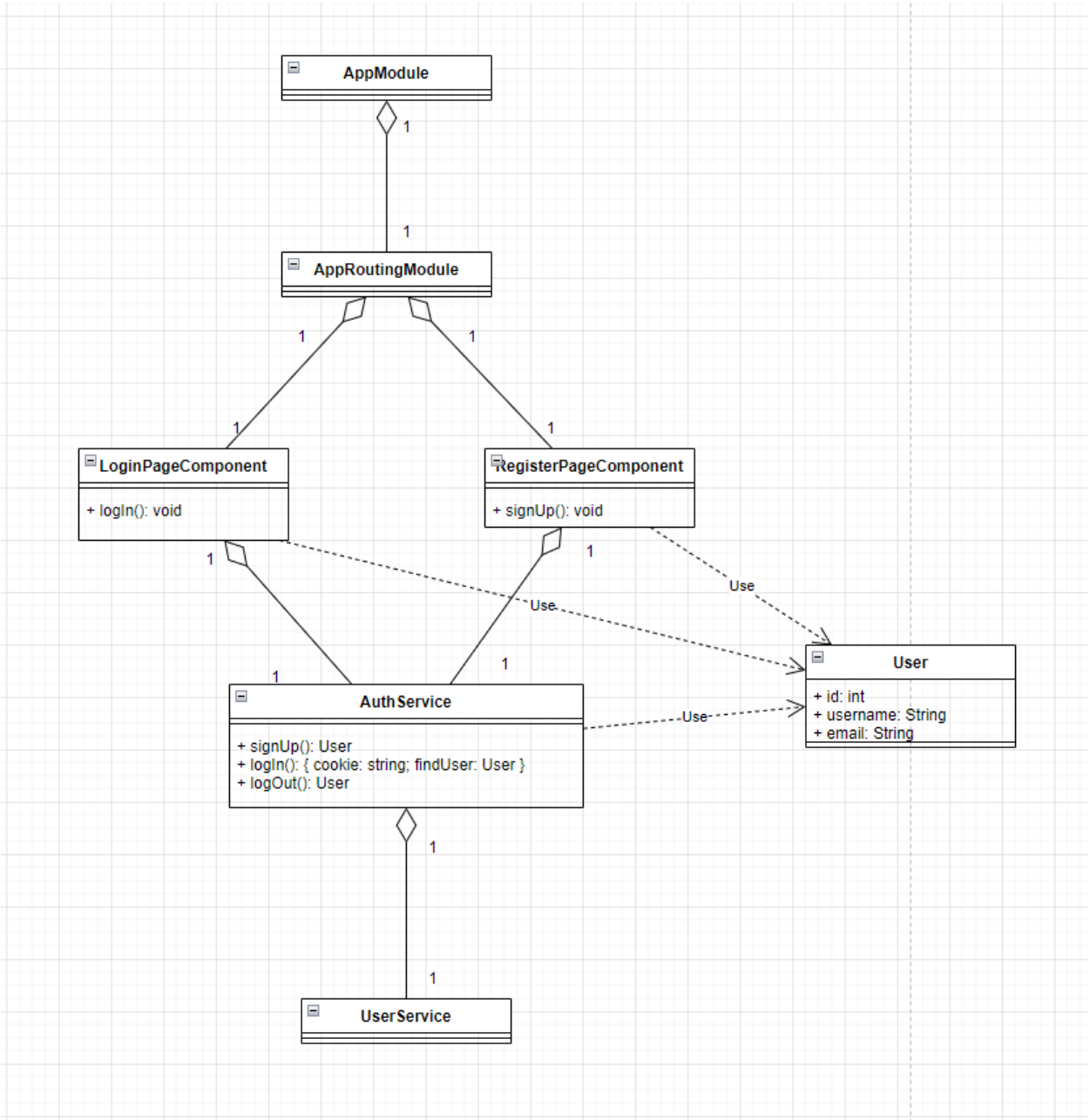
ChatService
Cette classe s’occupe de gérer le chat du côté serveur.



[Figure 11] Diagramme de Classe pour la partie dans le server

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

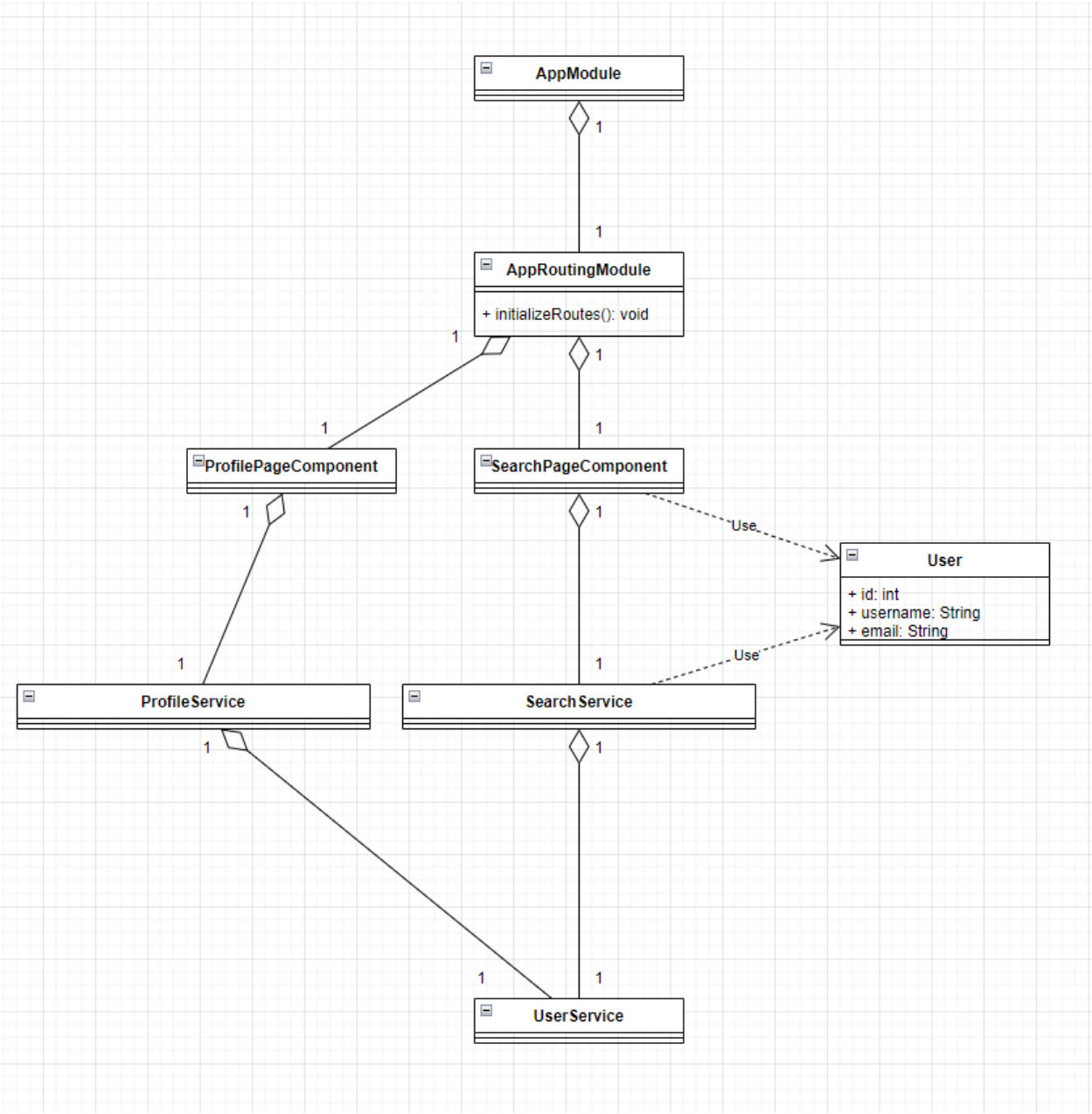
les différents clients.



Changer Controller à UserService

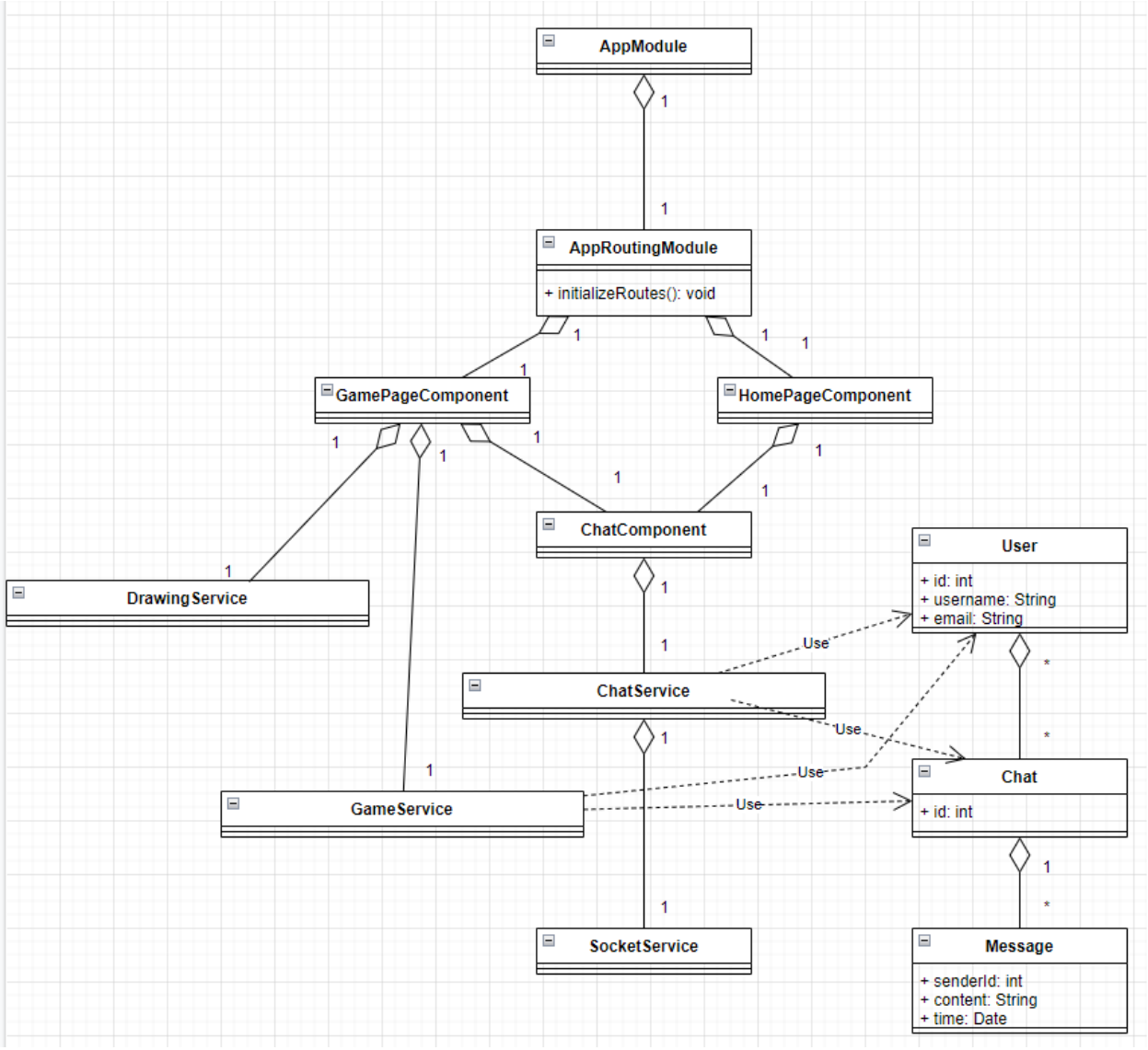
[Figure 12] Diagramme de Classe pour l'Authentification sur le client lourd

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 13] Diagramme de Classe pour le profile sur le client lourd

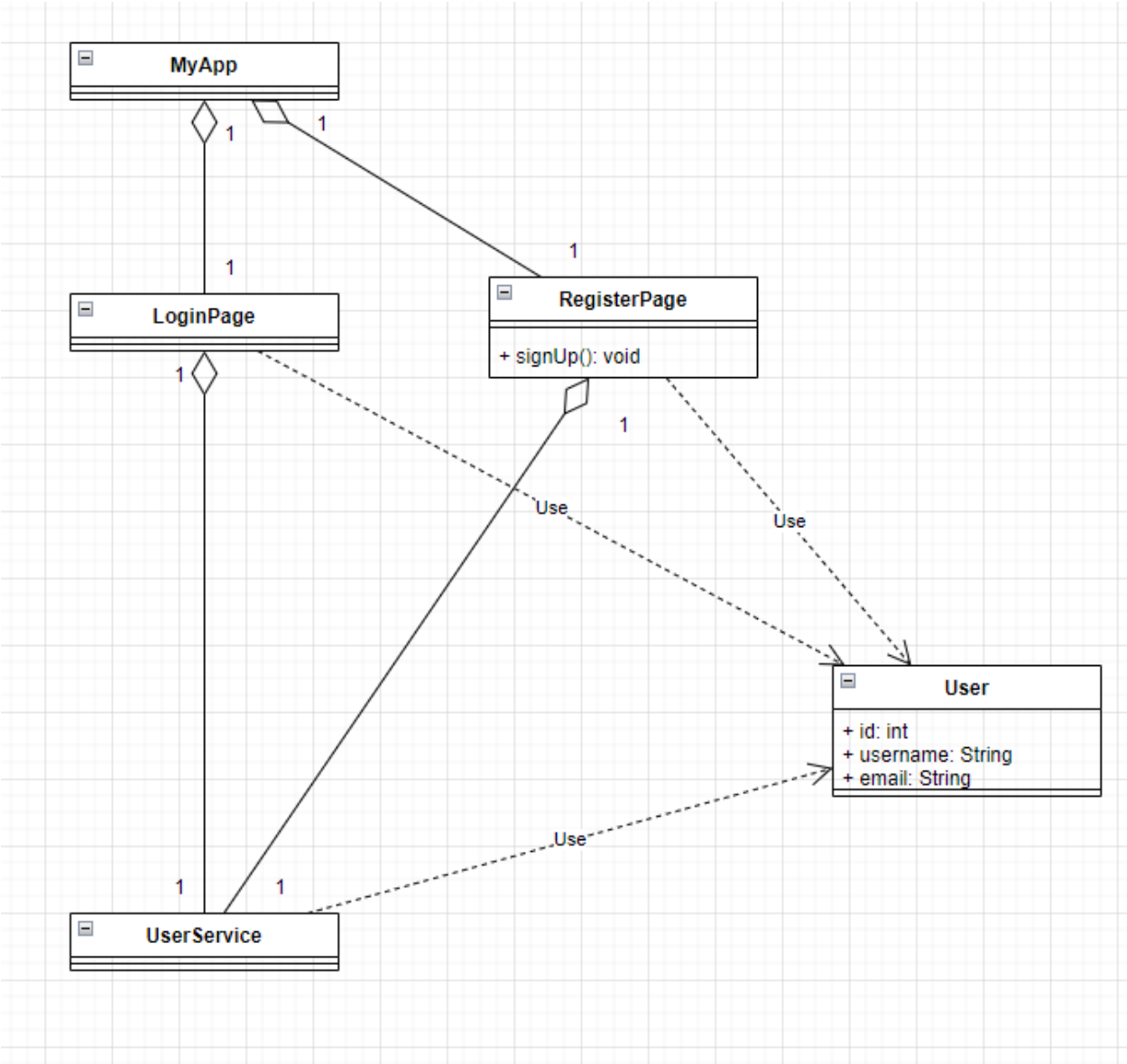
Profile
Ce package s'occupe de d'afficher les différentes pages et la communication avec le serveur pour tout ce qui touche à la recherche d'utilisateurs et des profils utilisateurs sur le client lourd.



[Figure 14] Diagramme de Classe pour le chat et la partie sur le client lourd

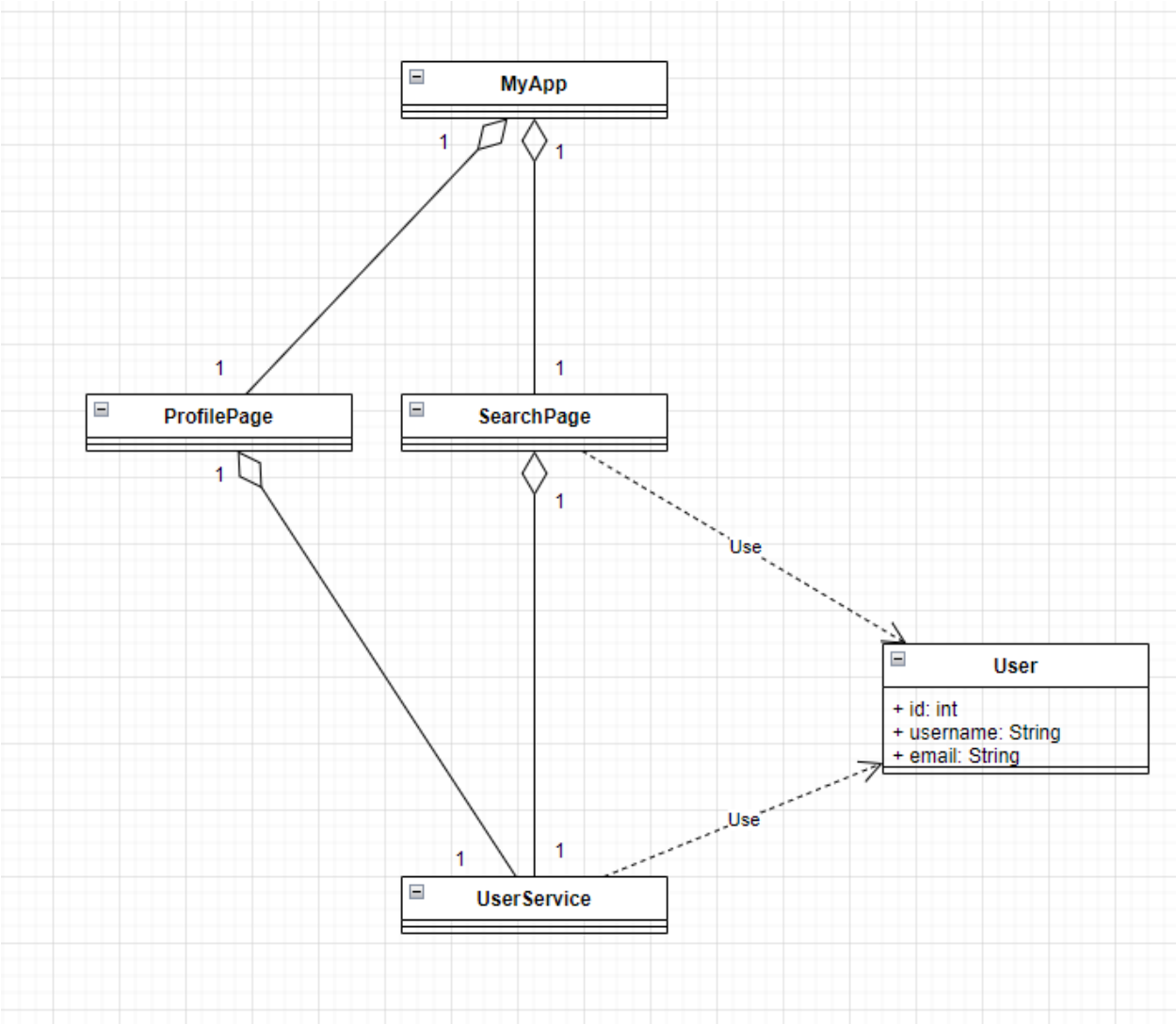
ChatComponent
Cette classe s'occupe d'afficher la messagerie sur le client lourd.

GamePageComponent
Cette classe s'occupe d'afficher le plateau de jeux et les actions possibles sur le client lourd.



[Figure 15] Diagramme de Classe pour l'Authentification sur le client léger

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.

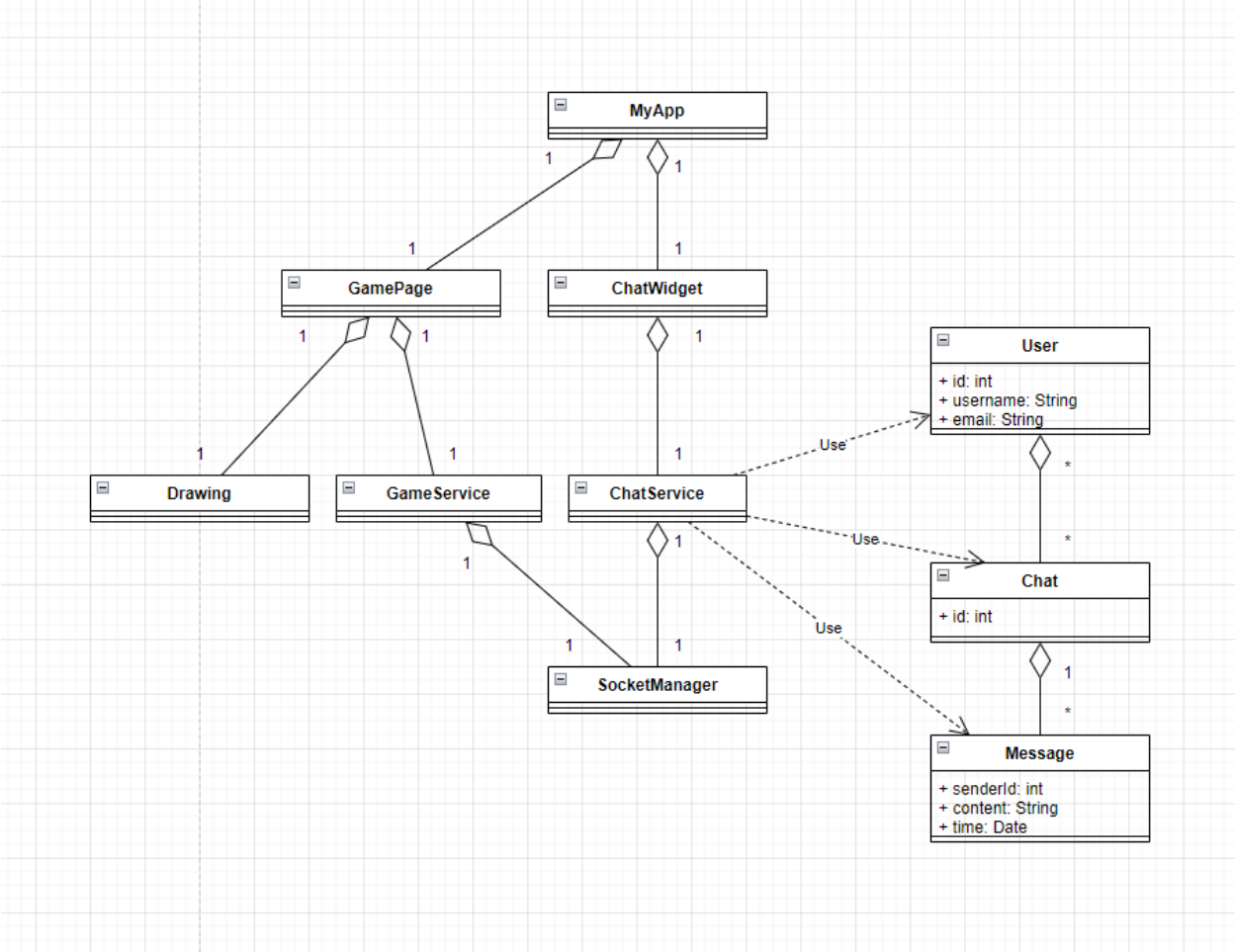


[Figure 16] Diagramme de Classe pour le profile sur le client léger

ProfilePage
Cette page s'occupe d'afficher les différentes information des des utilisateurs sur le client léger.

SearchPage
Cette page permet aux utilisateurs de rechercher d'autres

utilisateurs sur le client léger.



[Figure 17] Diagramme de Classe pour le chat et la partie sur le client léger

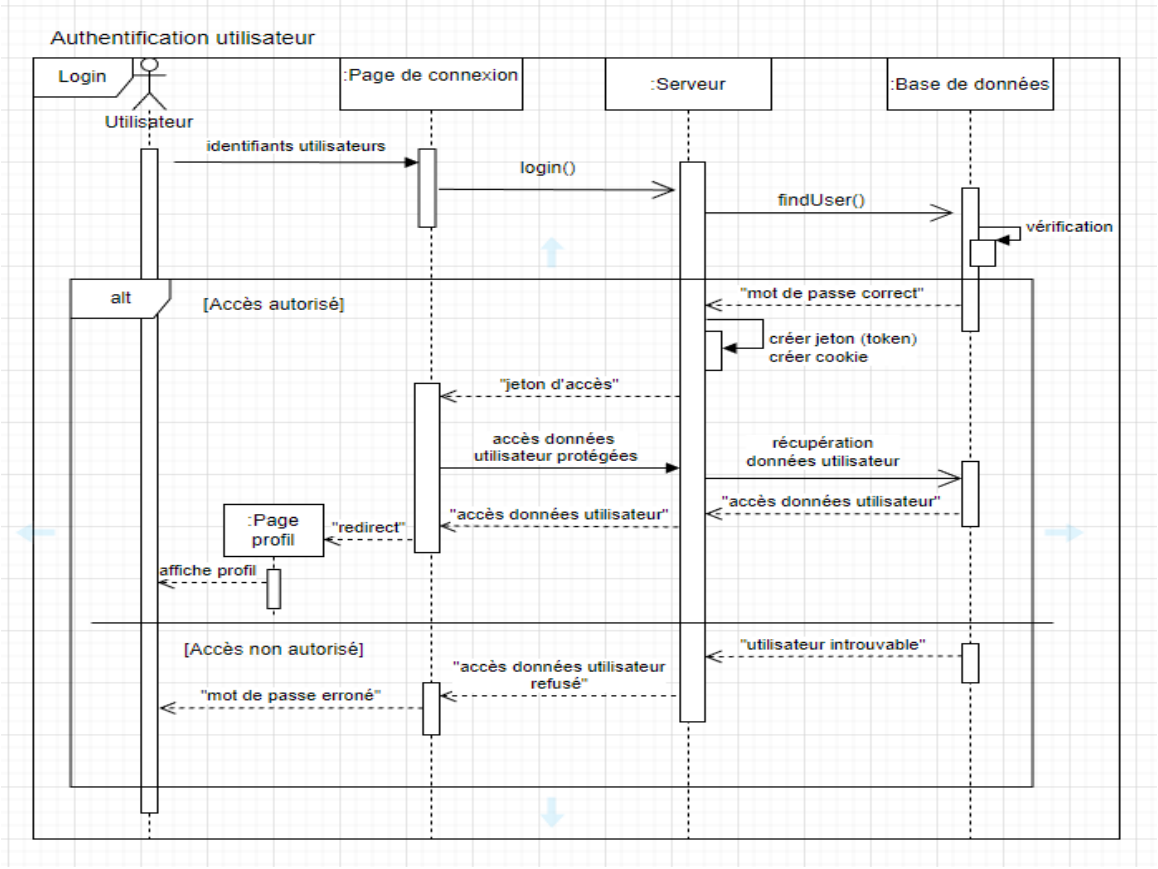
ChatWidget
Ce « widget » s'occupe d'afficher la messagerie sur le client léger.

GamePage

Ce « widget » s’occupe d’afficher le plateau de jeux et les actions possibles sur le client léger.

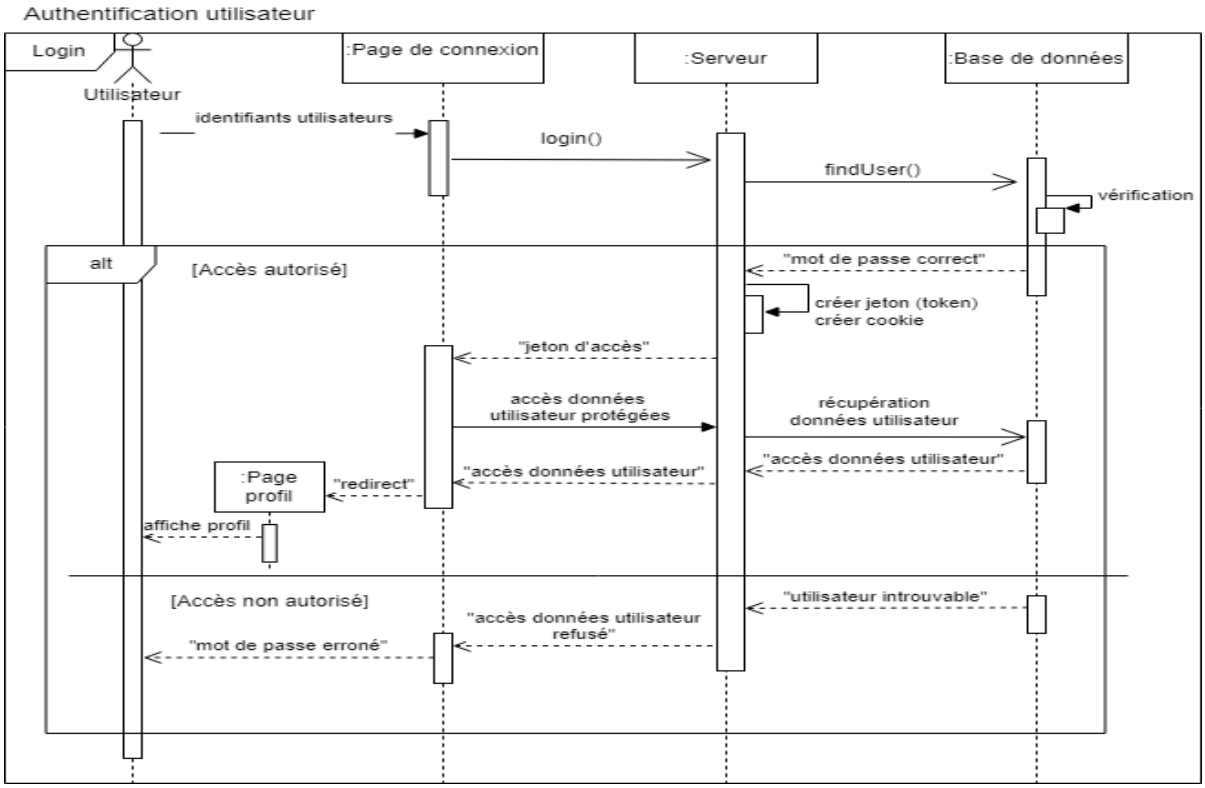
5. Vue des processus

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



[Figure 18] Création du compte de l'utilisateur

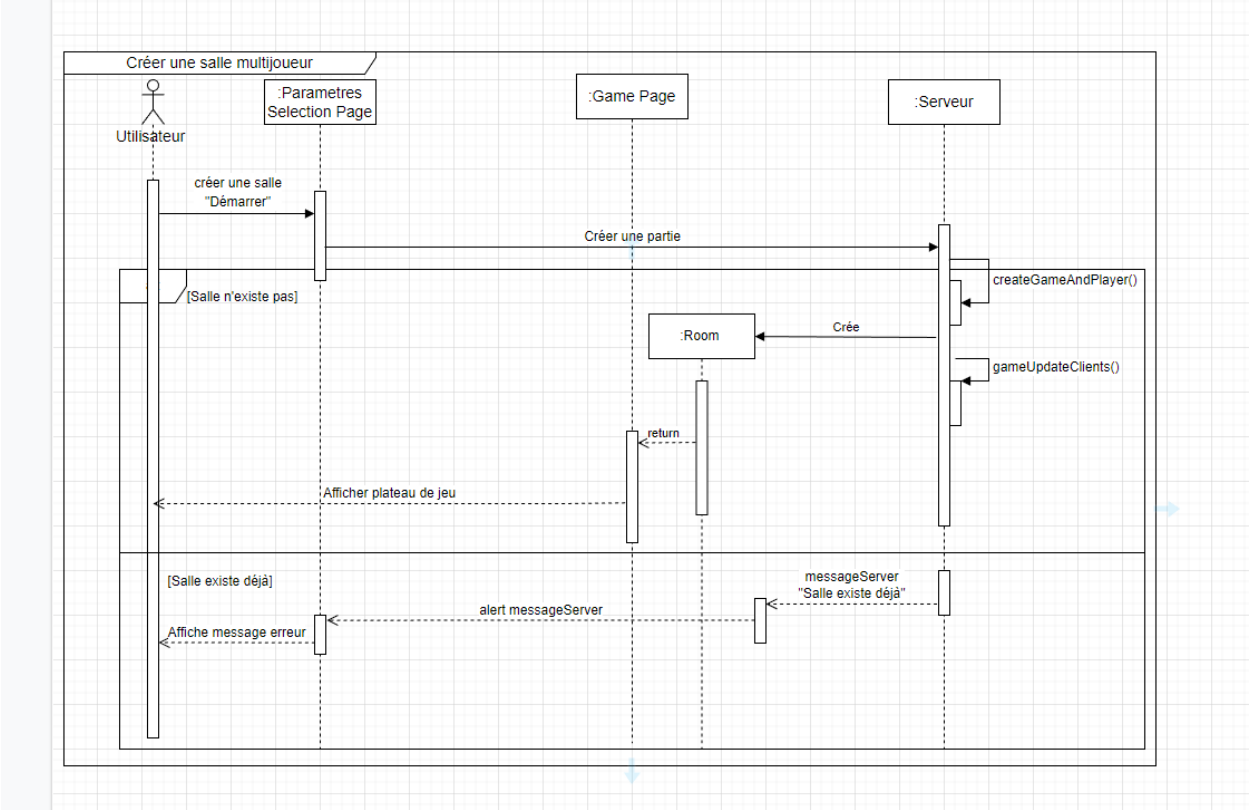
Processus no. 2: Authentification de l'utilisateur



[Figure 19] Authentification de l'utilisateur

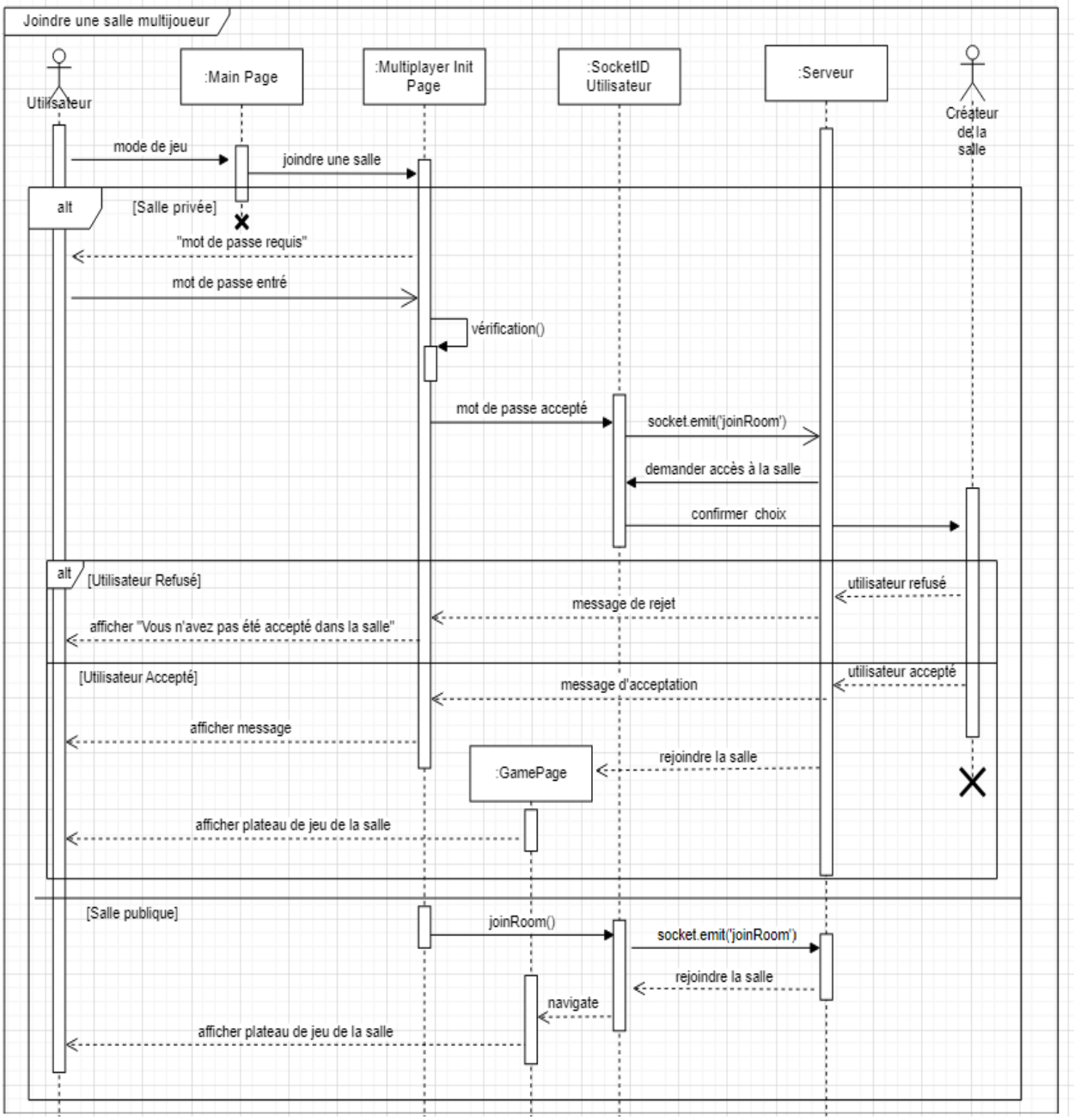
Pour le reste des processus, nous allons assumer que l'utilisateur est déjà connecté à son profil:

Processus no. 3: Création d’une salle multijoueur



[Figure 20] Création d’une salle multijoueur

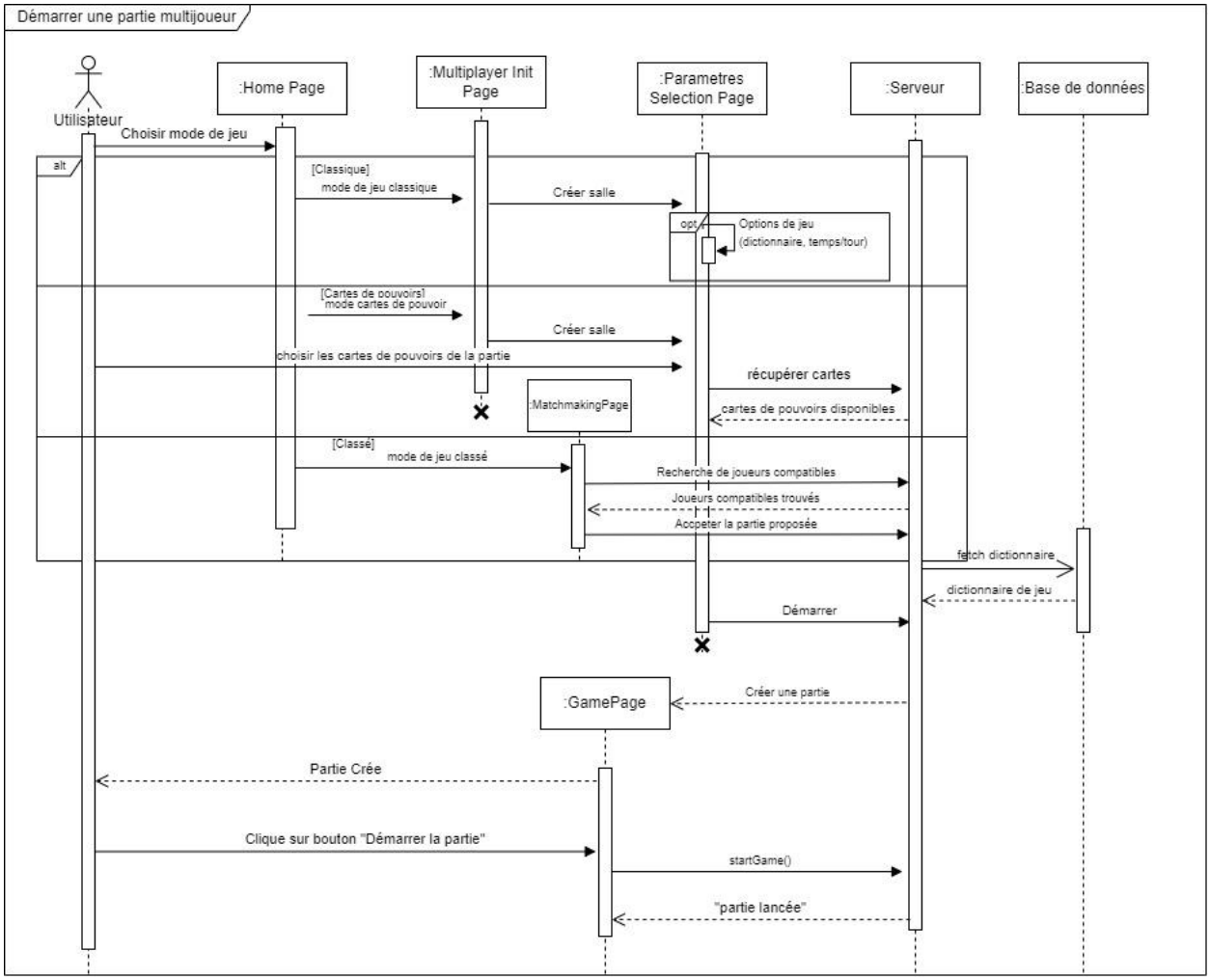
Processus no. 4: Joindre une salle multijoueur (privée ou publique)



[Figure 21] Joindre une salle multijoueur

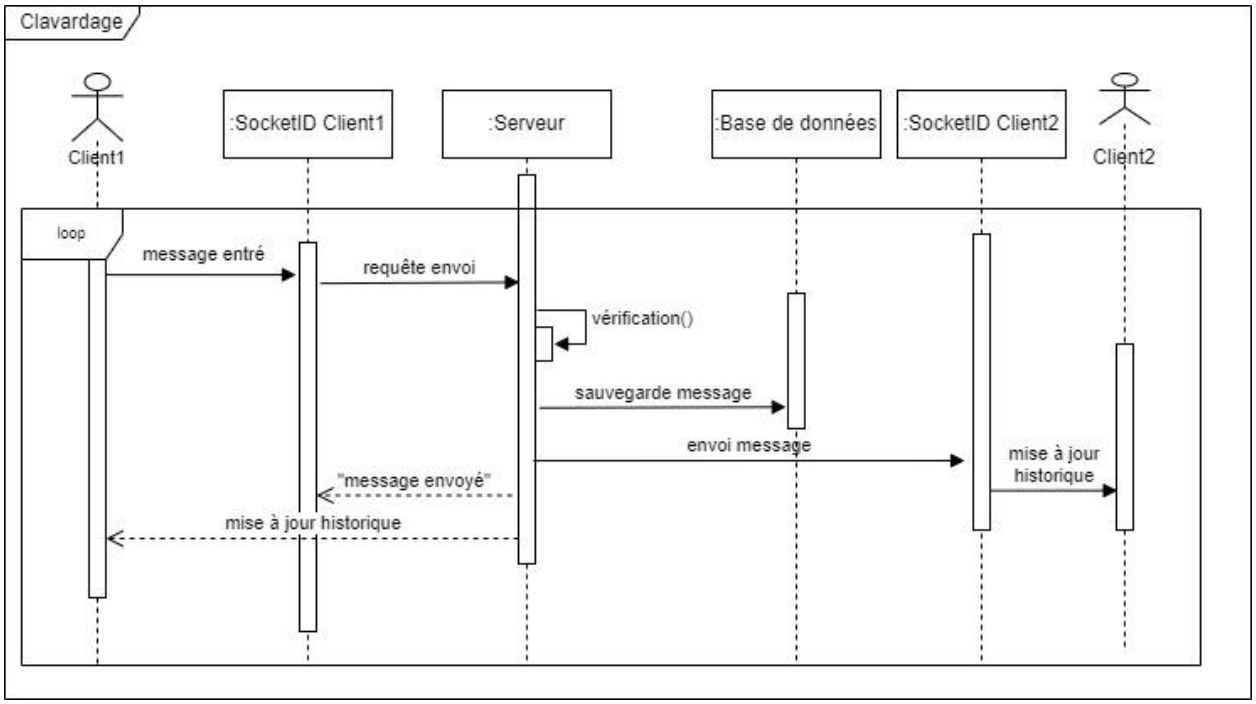
Processus no. 5: Démarrer une partie multijoueur.

Dans ce cas précis, nous allons assumer qu'au moins un autre joueur réel a rejoint la salle avant le démarrage de la partie.



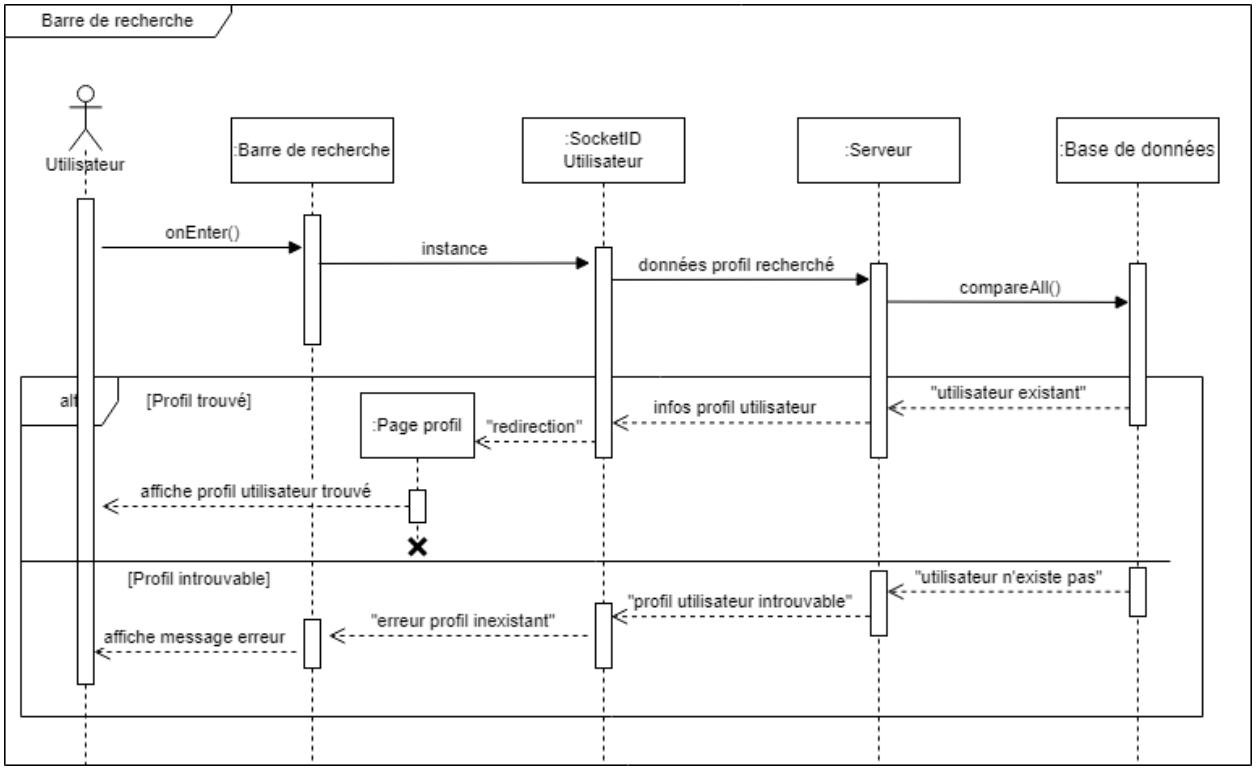
[Figure 22] Démarrer une partie multijoueur

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



[Figure 23] Clavardage entre les deux clients sur un serveur

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

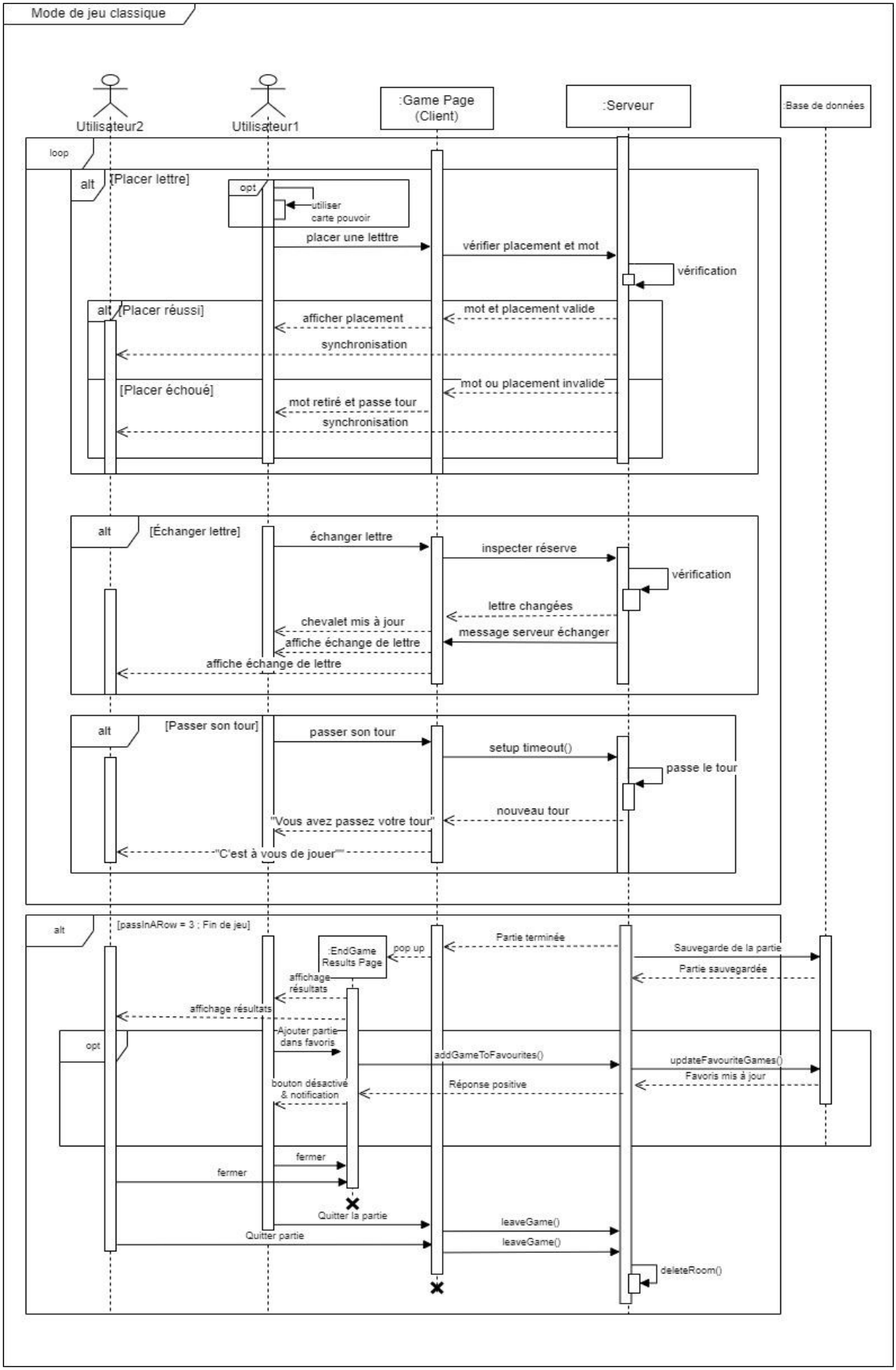


[Figure 24] Barre de recherche pour afficher le profil d'autres utilisateurs

Processus no. 8: Partie mode classique à 4 joueurs

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.

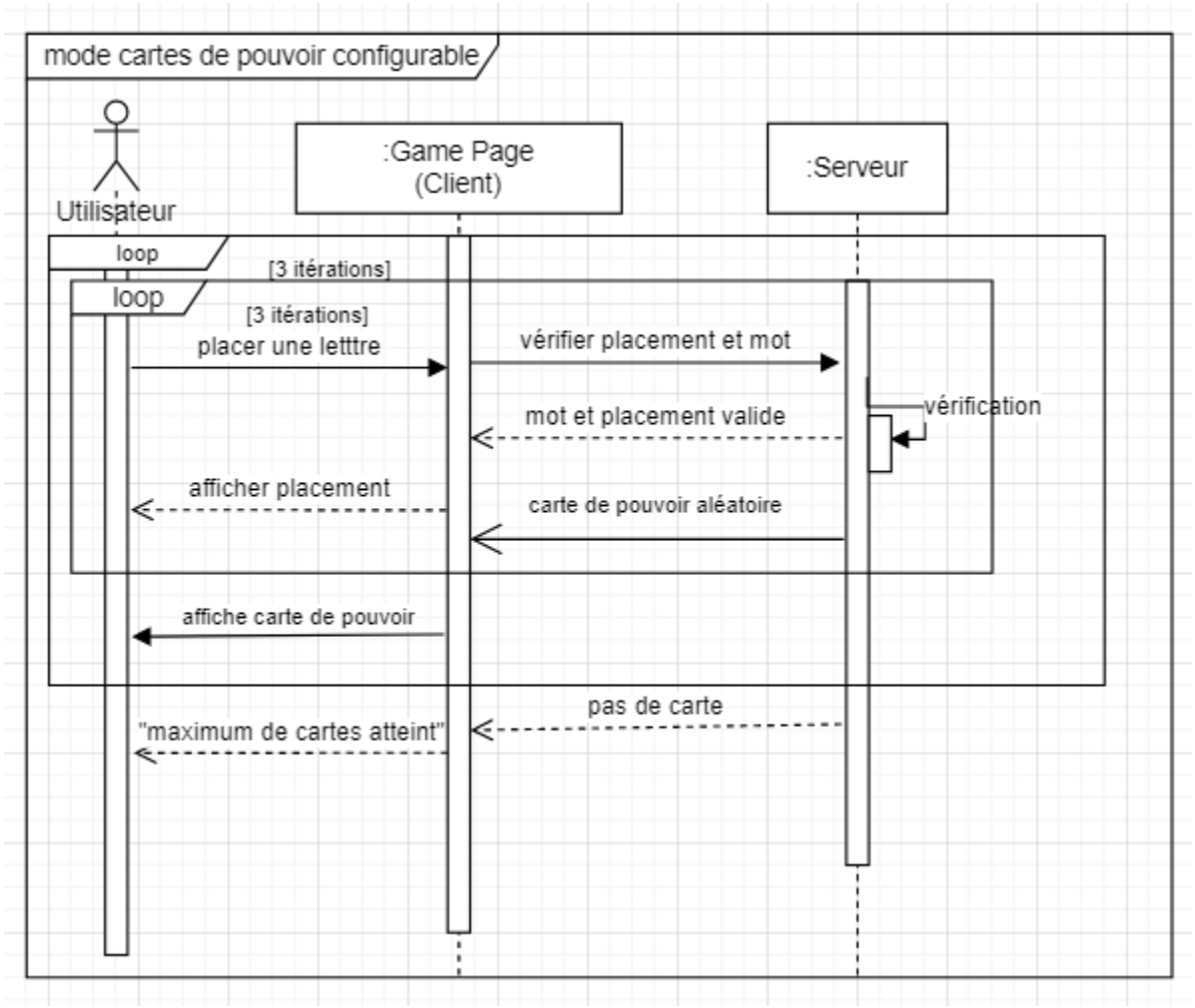
passInARow désigne la variable qui compte le nombre de fois où tous les joueurs ont passé leur tour 7 la suite.


Mode de jeu classique
Utilisateur2
Utilisateur1
:Game Page (Client)
:Serveur
:Base de données
loop
alt [Placer lettre]
opt
utiliser carte pouvoir
placer une letttre
vérifier placement et mot
vérification
alt [Placer réussi]
afficher placement
mot et placement valide
synchronisation
[Placer échoué]
mot ou placement invalide
mot retiré et passe tour
synchronisation
alt [Échanger lettre]
échanger lettre
inspecter réserve
vérification
lettre changées
chevalet mis à jour
message serveur échanger
affiche échange de lettre
affiche échange de lettre
alt [Passer son tour]
passer son tour
setup timeout()
passe le tour
nouveau tour
"Vous avez passez votre tour"
"C'est à vous de jouer""
alt [passInARow = 3 ; Fin de jeu]
:EndGame Results Page
pop up
Partie terminée
Sauvegarde de la partie
Partie sauvegardée
affichage résultats
affichage résultats
opt
Ajouter partie dans favoris
addGameToFavourites()
updateFavouriteGames()
Favoris mis à jour
bouton désactivé & notification
Réponse positive
fermer
fermer
Quitter la partie
leaveGame()
Quitter partie
leaveGame()
deleteRoom()

[Figure 25] Partie mode classique à 4 joueurs

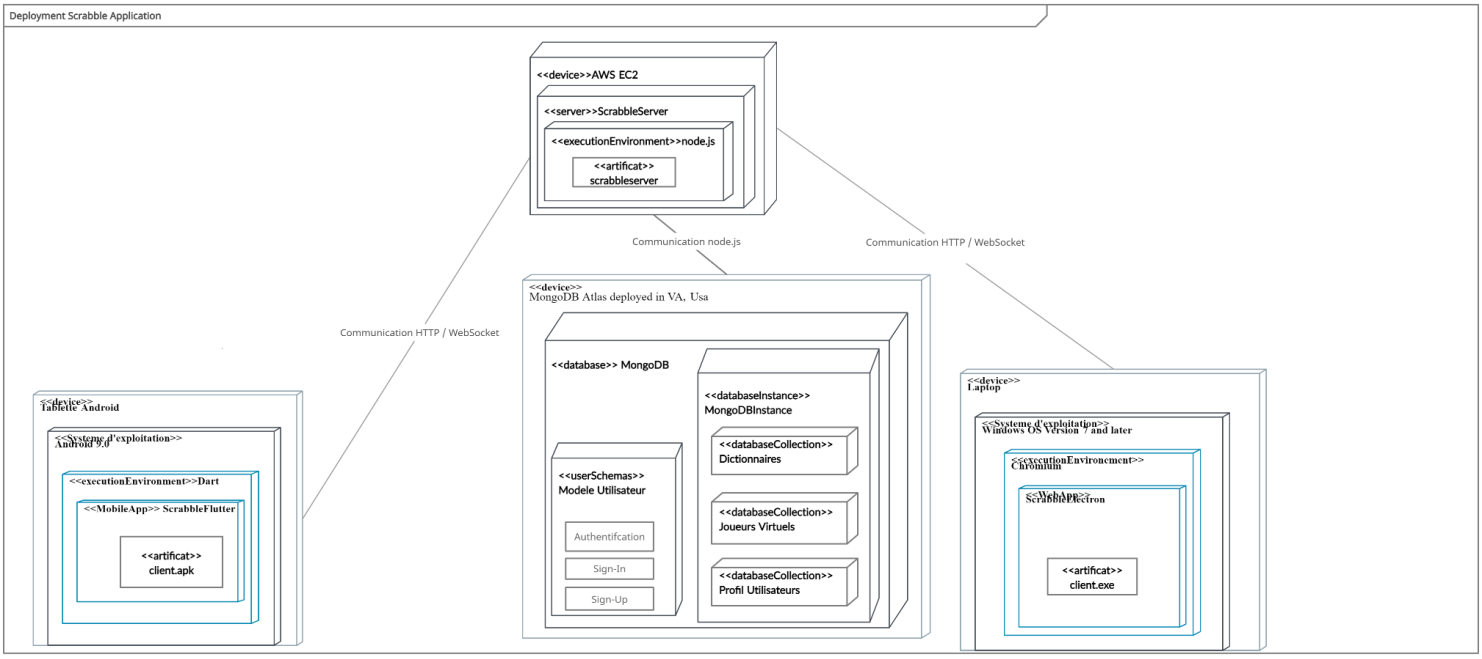
Processus no. 9: Mode de jeu cartes de pouvoir

Séquence d’acquisition des cartes au cours de la partie



[Figure 26] Mode de jeu cartes de pouvoir

6. Vue de déploiement



[Figure 27] 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.
Notre serveur utilise en moyenne 5% de RAM lorsqu'il roule et utilise 0.3% du CPU.

7.2 Client Léger

Le client léger sera un apk de taille d'environ 30 Mo, 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 Mo, compatible avec Windows 10 et 11, avec 300 Mo d’utilisation de mémoire vive nécessaire.

7.4 Base de Données

Il y a une latence de communication entre le réseau et la base de données de l’ordre de 200 ms, avec un cluster de taille de 512 Mo.