204

Document d'architecture logicielle

Version 1.5

Historique des révisions

Date	Version	Description	Auteur
2023-03-17	1.1	Complétion de la partie vue logique	Équipe 204
2023-03-18	1.2	Complétion de la partie vue des cas d'utilisation	Équipe 204
2023-03-18	1.3	Complétion de la partie vue des processus	Équipe 204
2023-03-20	1.4	Complétion de la partie vue de déploiement	Équipe 204
2023-03-20	1.5	Complétion de l'introduction	Équipe 204

Table des matières

1. Introduction 4
2. Vue des cas d'utilisation 5
3. Vue des processus 12
4. Vue logique 14
5. Vue de déploiement 17

Document d'architecture logicielle

1. Introduction

Ce document décrit nos décisions de l'architecture logicielle du projet en fournissant différentes vues selon la notation UML pour faciliter la compréhension et la gestion du système. On commence avec les diagrammes de cas d'utilisation qui illustrent les fonctions générales développées lors du sprint 3 ainsi que la portée du système. Par la suite, on détaille la vue des processus à l'aide d'un diagramme de séquence. Le but de cette partie est de montrer comment nos composantes interagissent dans le temps dans les processus implémentés lors de ce dernier sprint. La quatrième section modélise les éléments et mécanismes principaux du système à l'aide de diagrammes de paquetages et de diagrammes de classes. Dans cette section, on montre l'organisation des composants ainsi que les dépendances entre ces composants. On montre également l'organisation des modules en sous-systèmes. La cinquième et dernière section, la vue de déploiement, décrit les ressources matérielles ainsi que la répartition du logiciel dans ces ressources.

2. Vue des cas d'utilisation

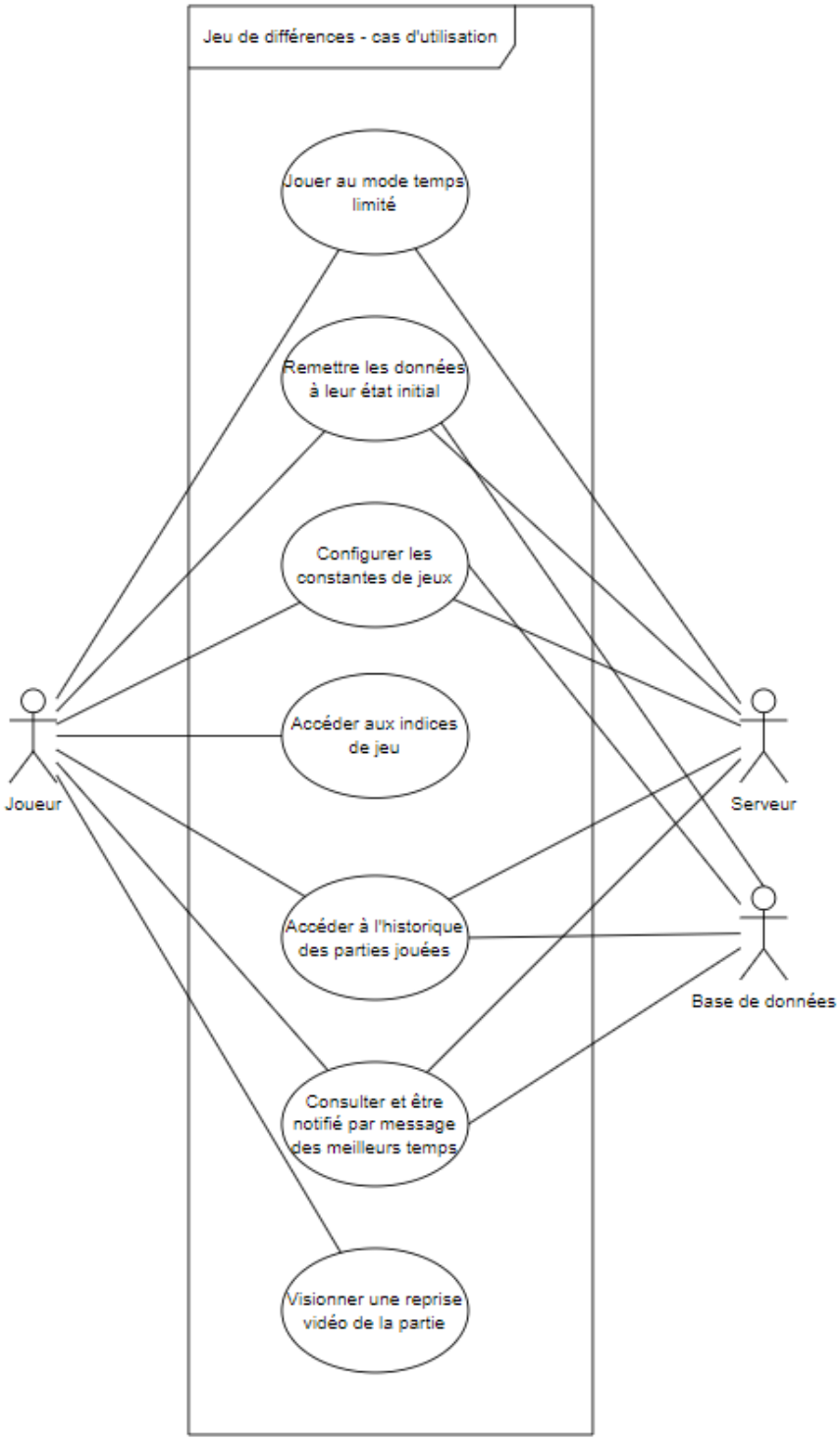


Figure 1: Schéma du cas d'utilisation du jeu de différences

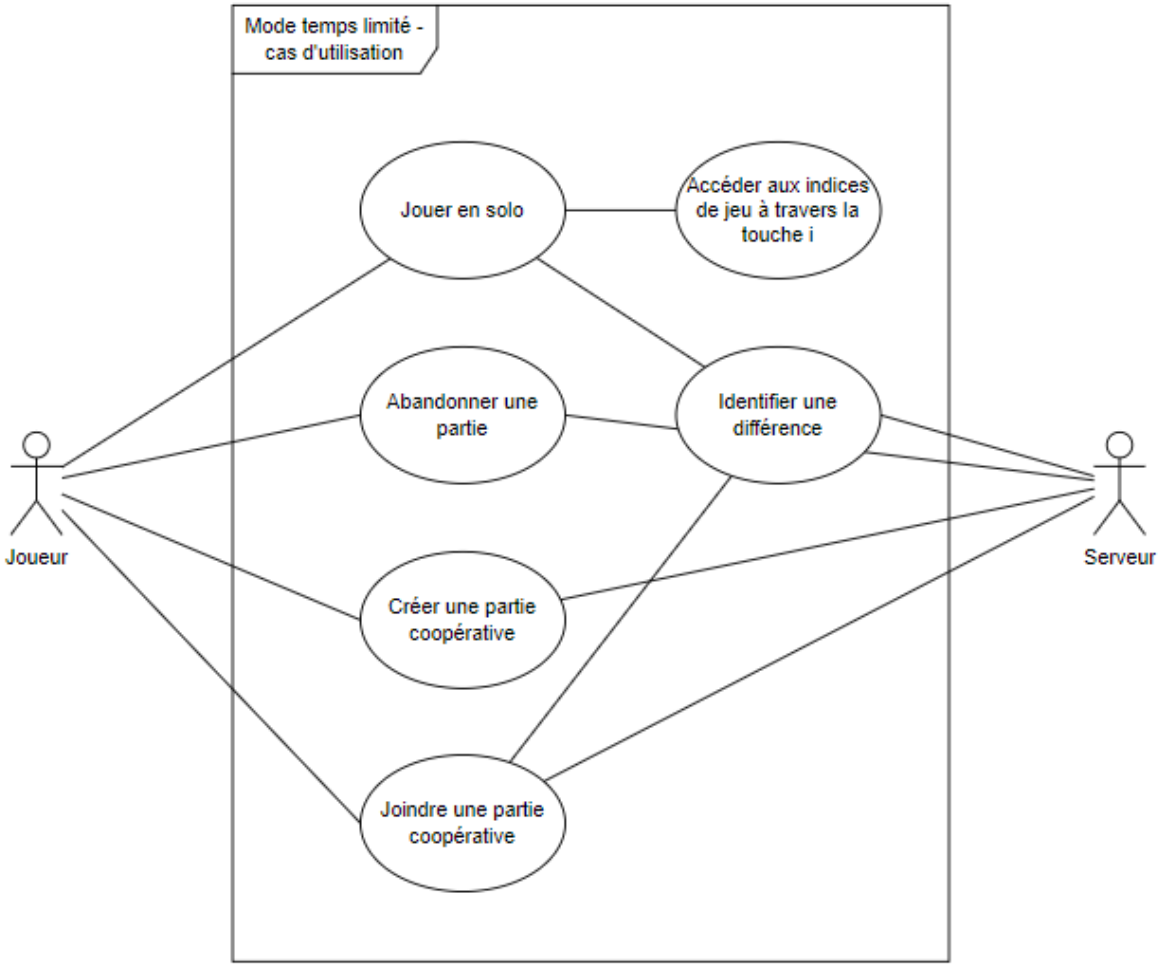


Figure 2: Schéma du cas d'utilisation du jeu en mode temps limité

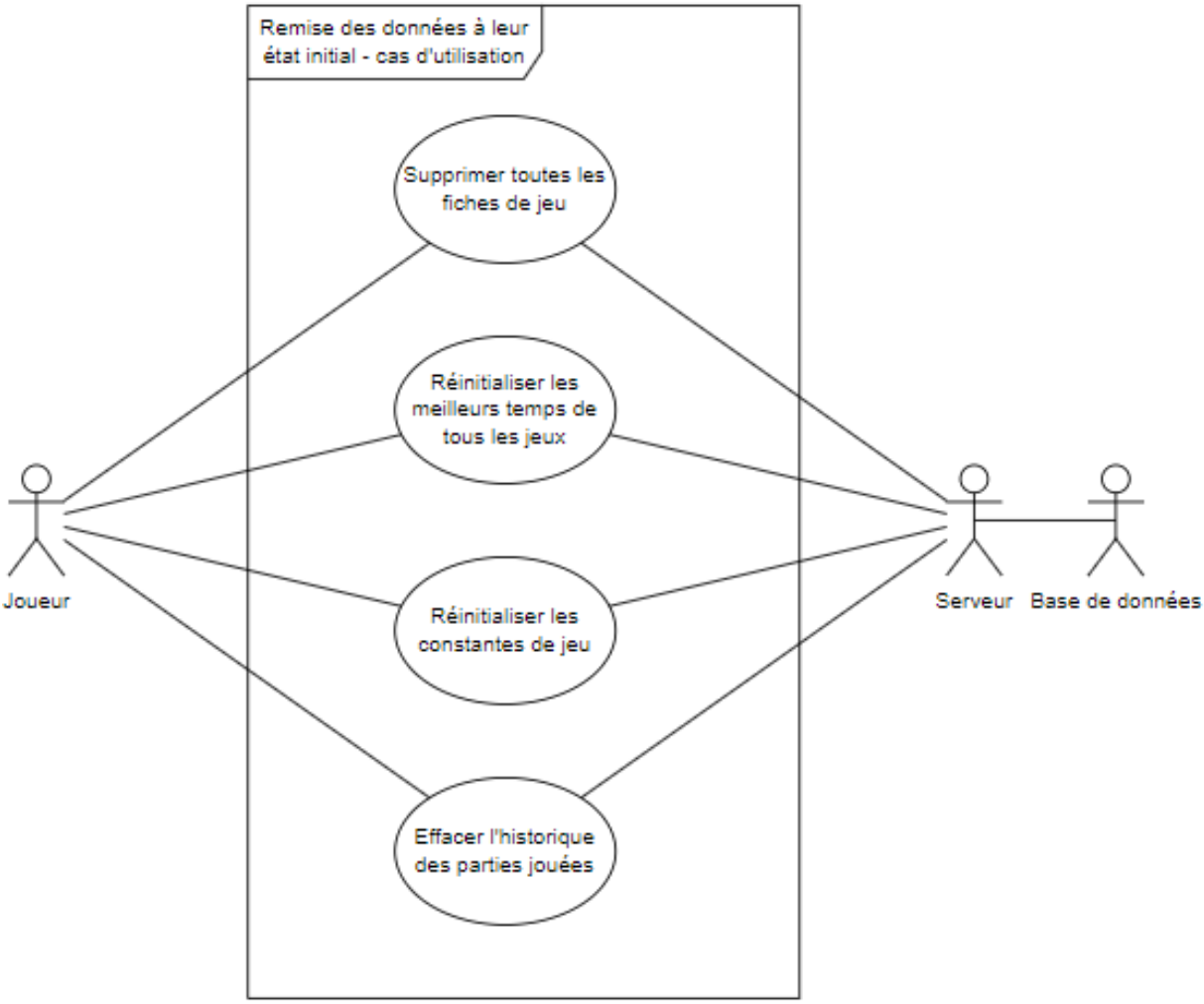


Figure 3: Schéma du cas d'utilisation de remise des donnés à leur état initial

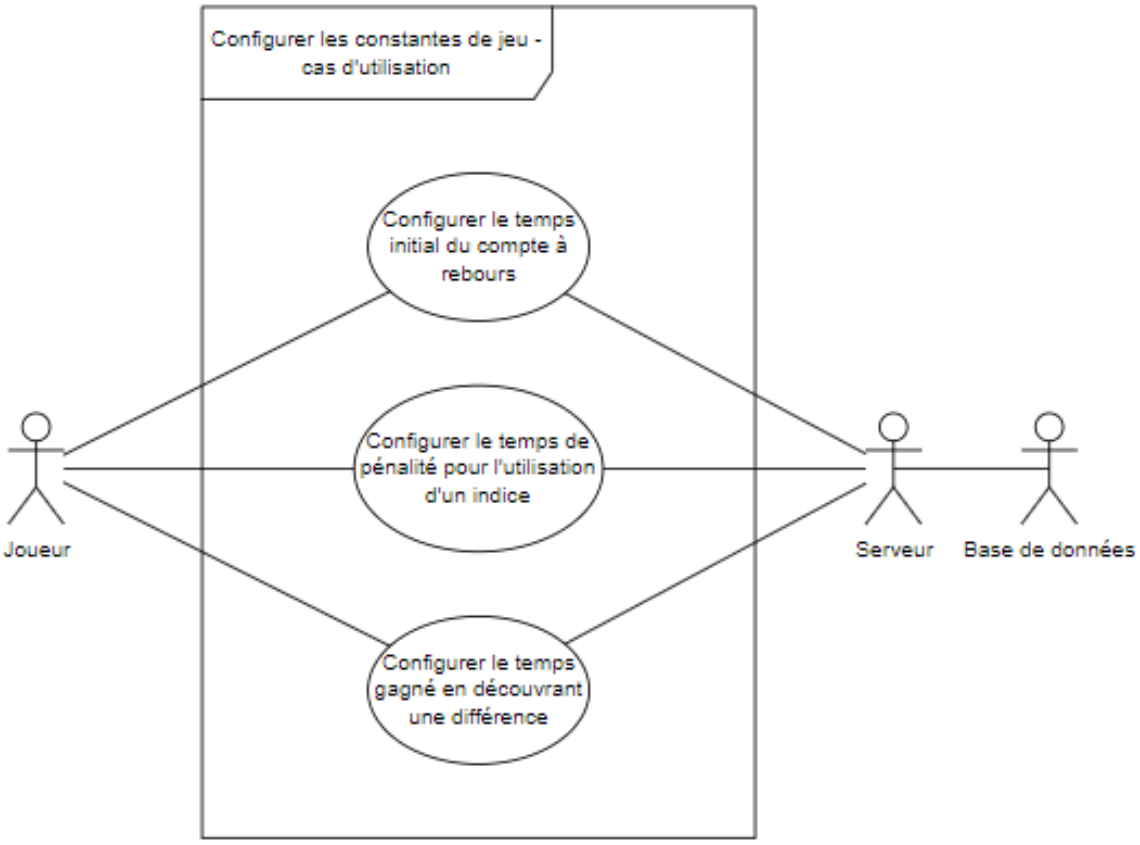


Figure 4: Schéma du cas d'utilisation de configuration des constantes de jeu

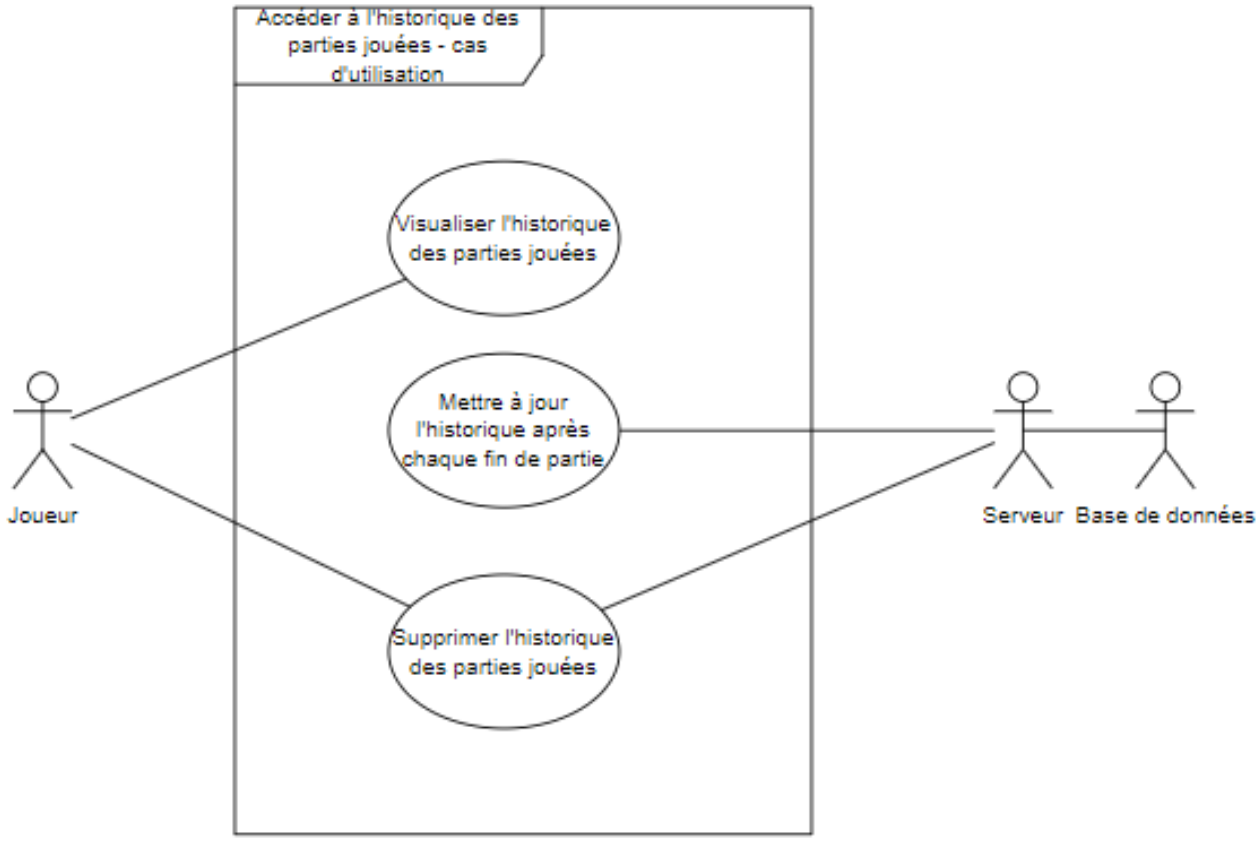


Figure 5: Schéma du cas d'utilisation de l'historique des parties jouées

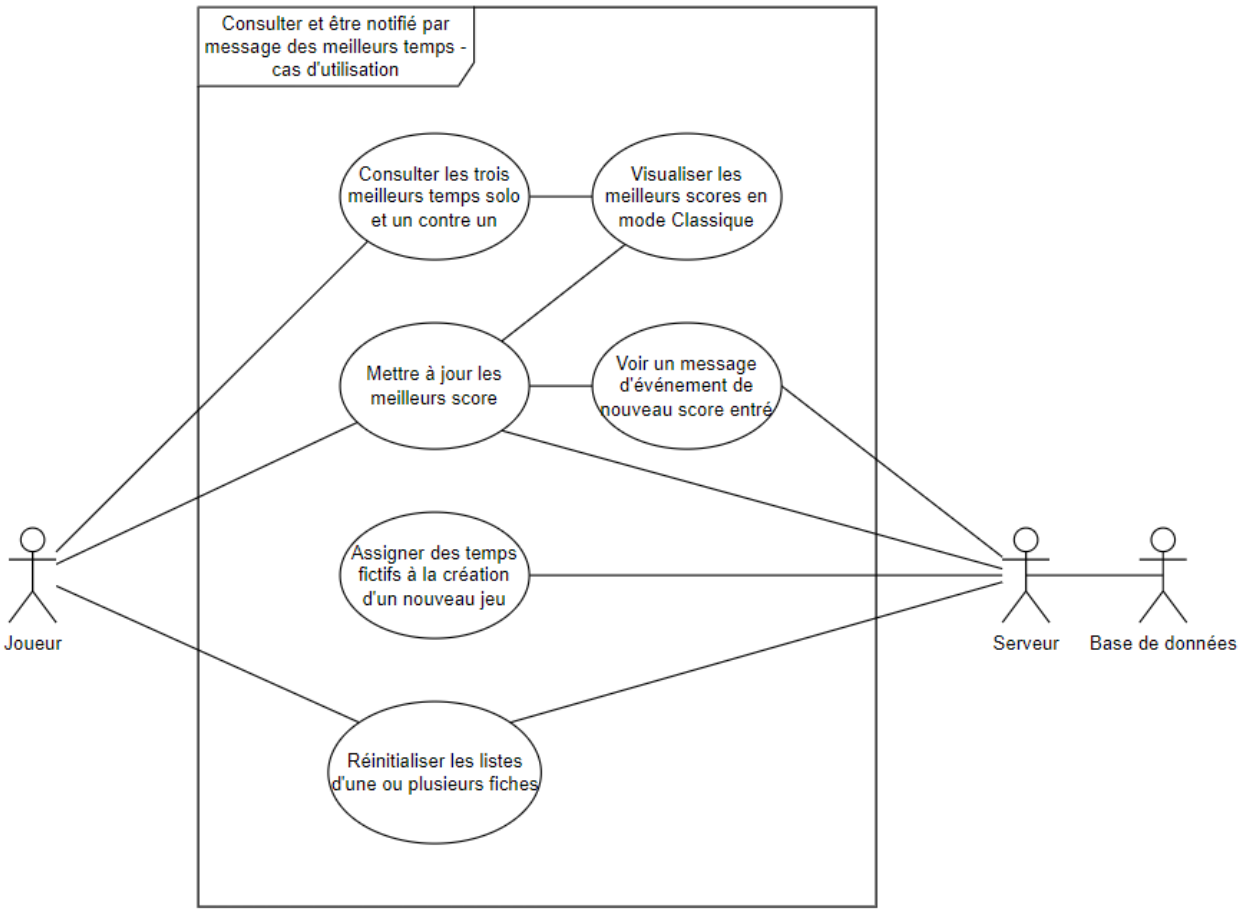


Figure 6: Schéma du cas d'utilisation de consultation et être notification par message des meilleurs temps

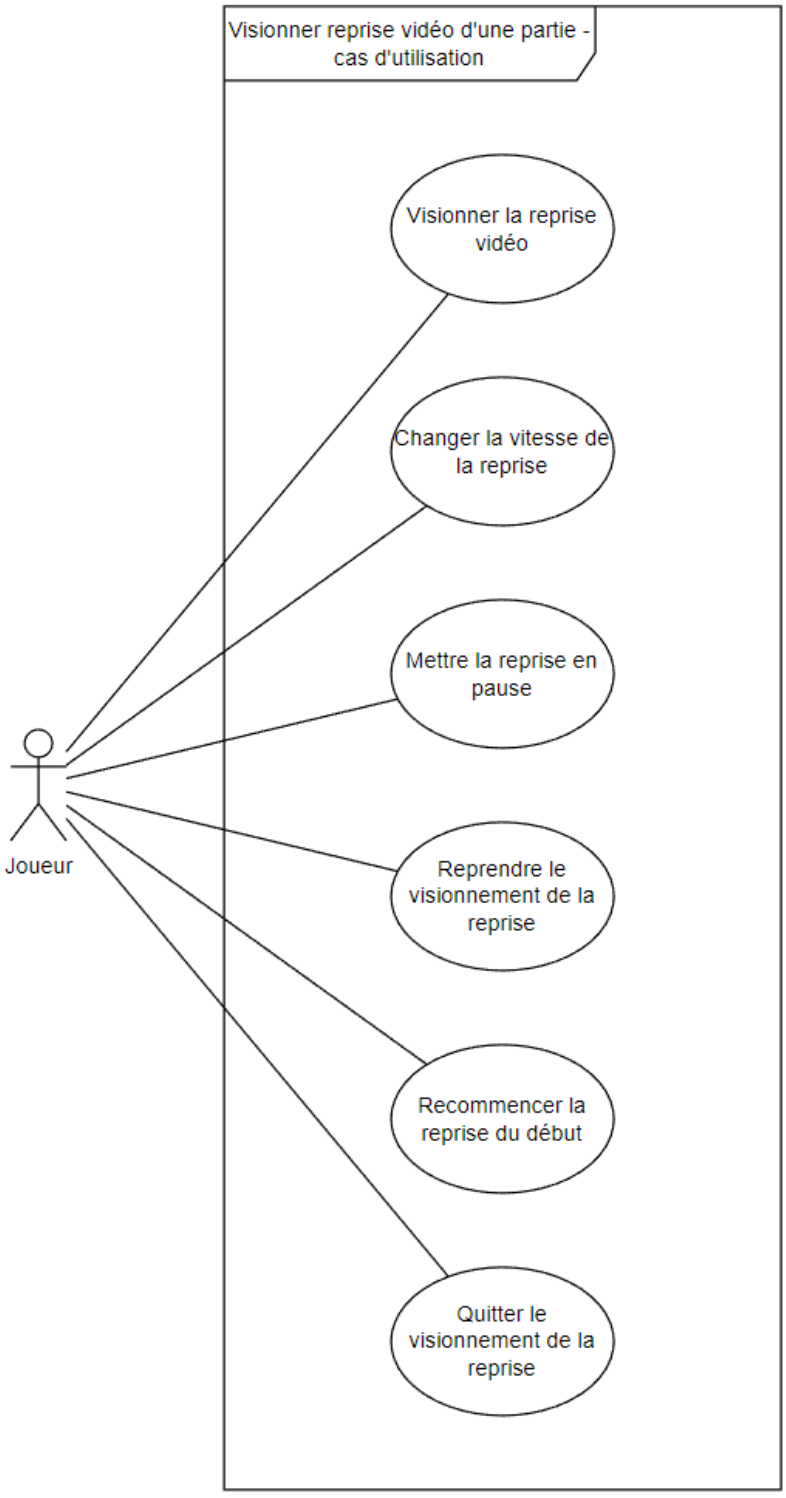
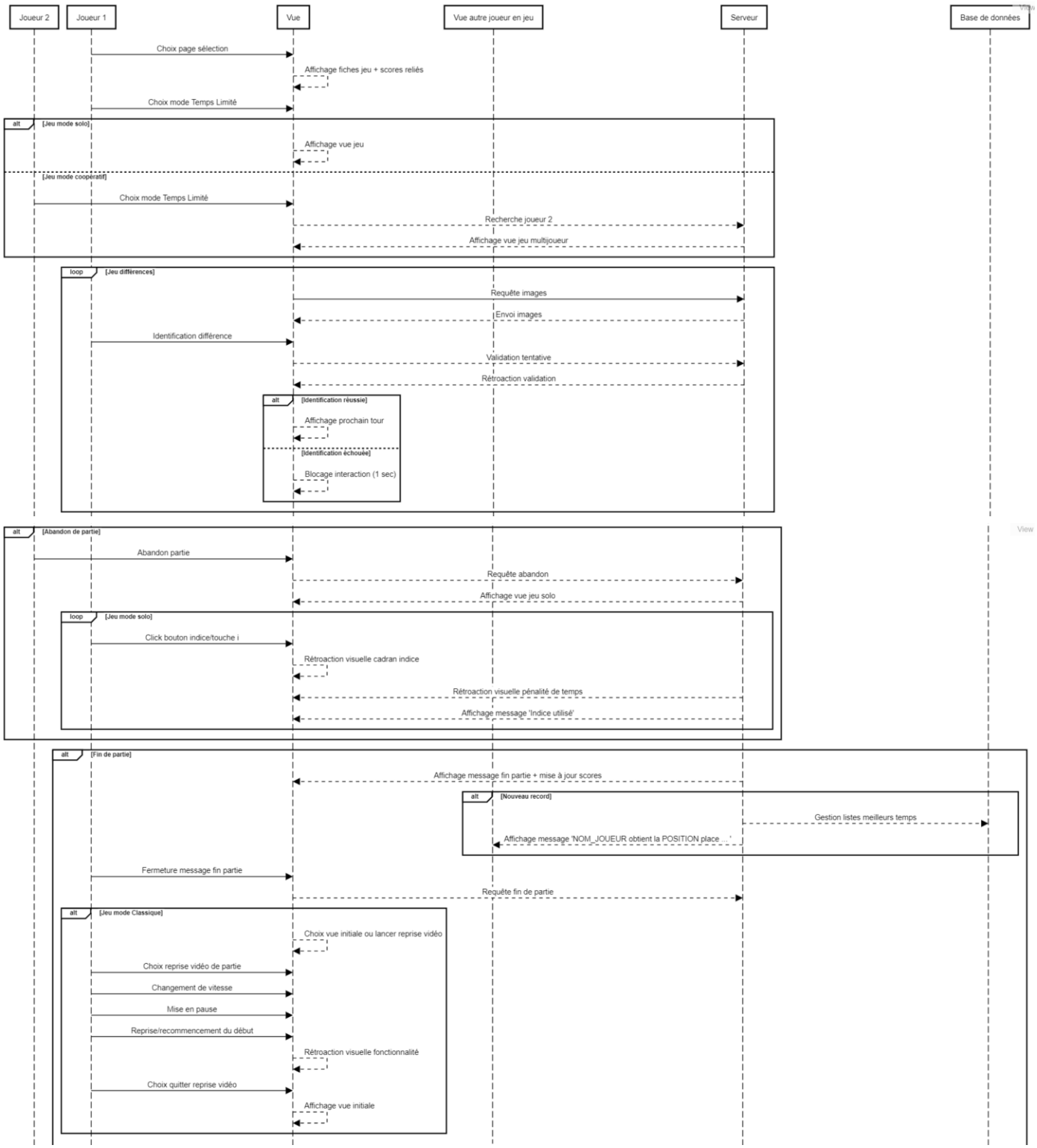


Figure 7: Schéma du cas d'utilisation de visionnement de la reprise vidéo d'une partie

3. Vue des processus



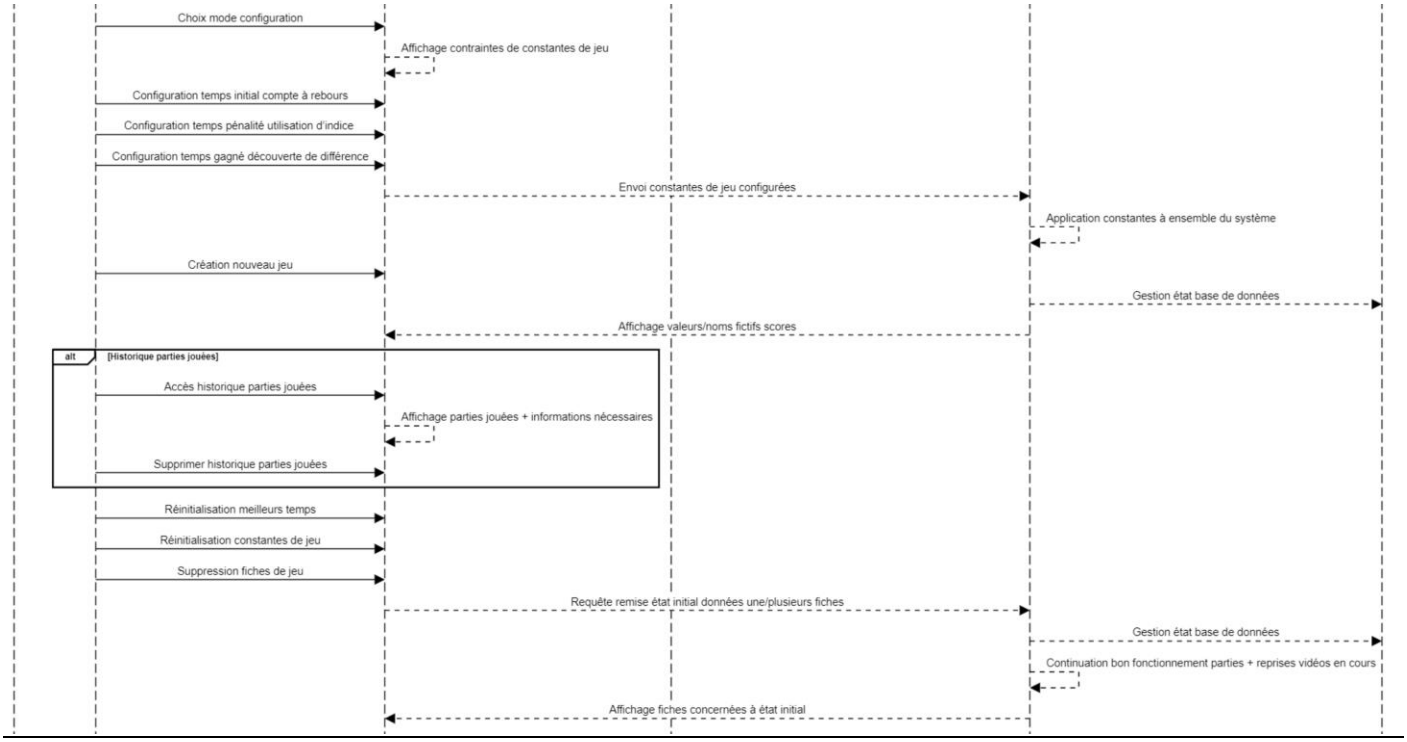


Figure 8: Schéma de séquence de la vue des processus

4. Vue logique

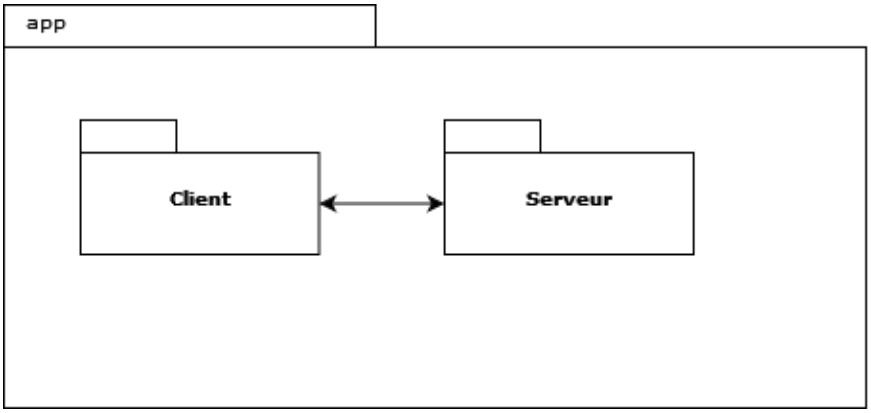


Figure 9: Schéma de paquetage de l'app

Client
Le client crée les différentes pages de notre application web et permet la navigation entre ces dernières. Il est responsable de la création des interfaces graphiques et du développement front-end. Il est également responsable de la gestion des interactions avec l'utilisateur à travers des boutons, des entrées d'images, des entrées de texte ou des clics et mouvements de souris. Le client fait le lien entre ces interactions et la logique d'arrière-plan placée dans des services. Enfin, il s'assure de générer des requêtes HTTP correctes puis les communiquer au serveur. Il est aussi responsable de traiter la réponse du serveur.

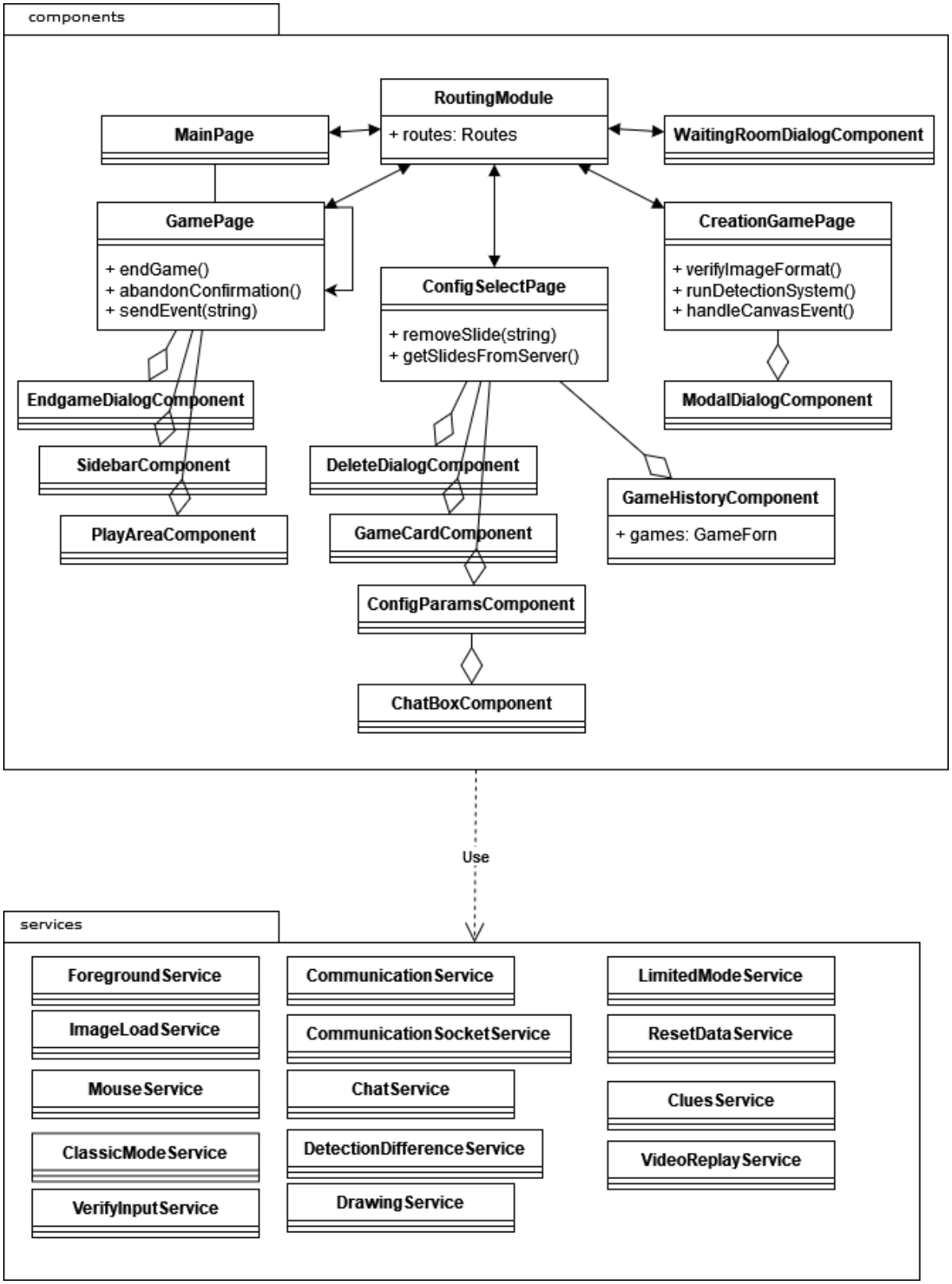


Figure 10: Schéma du paquetage du client

Serveur
Le serveur a pour rôle de communiquer avec le client les différentes informations concernant les jeux et les joueurs. Il contient différentes fonctionnalités puisqu'il est responsable de communiquer avec le client avec deux protocoles différents, Web Socket (socket IO) et HTTP. On a donc un contrôleur qui s'occupe des requêtes HTTP, principalement pour la création, récupération et suppression de jeux. Les deux Gateways communique avec socket IO au client des information comme les messages utilisateurs, ou principalement des informations de connexion aux parties en ligne. Ces deux composants du serveur utilisent chacun leur service attribué pour effectuer la logique de ClassicMode d'un côté et pour communiquer les données des jeux avec la base de données pour le contrôleur.

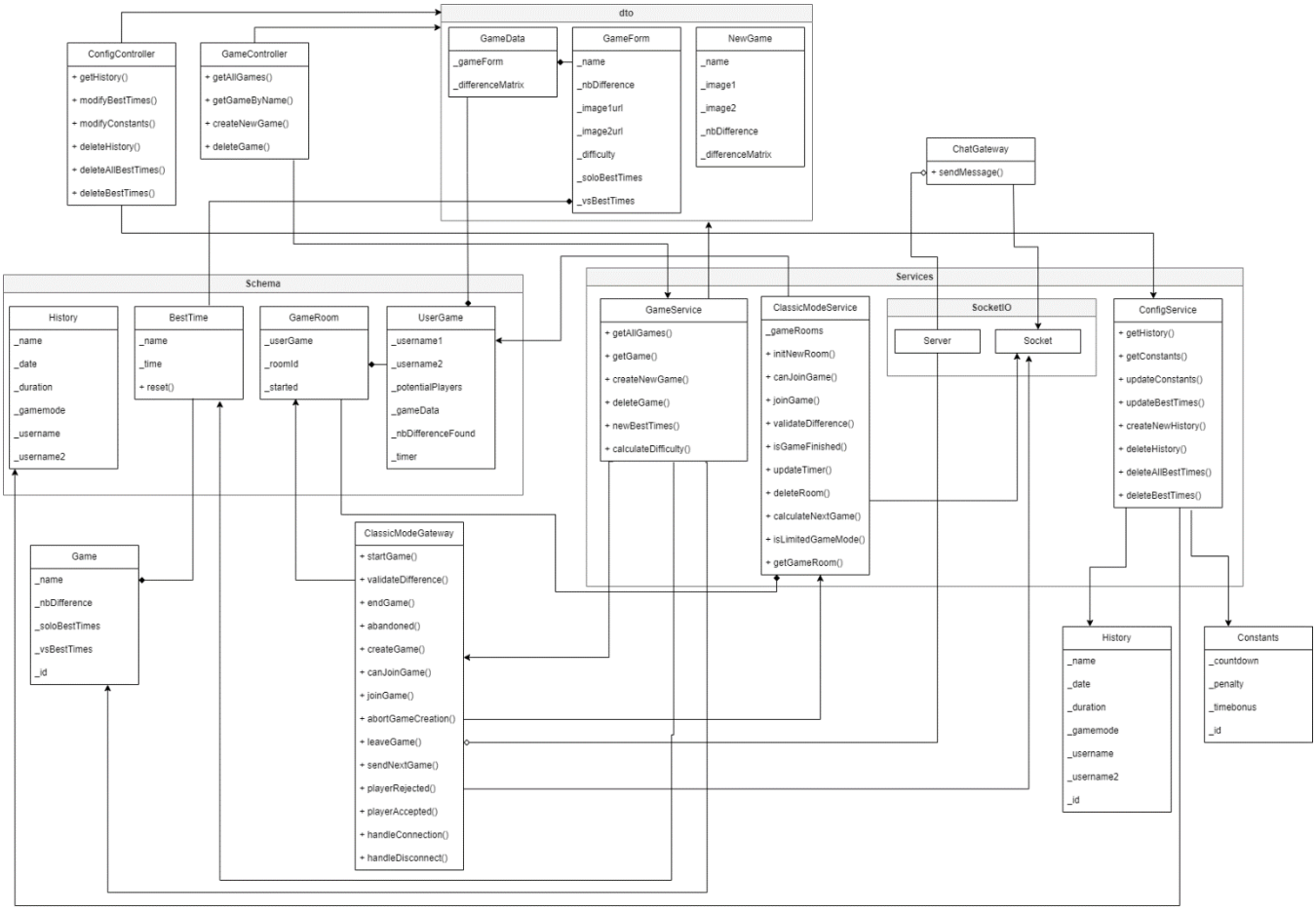


Figure 11: Schéma du paquetage du serveur

5. Vue de déploiement

Lorsque tout le projet est déployé, on peut retrouver trois principaux nœuds :

- Le client, désignant le site web statique, hébergé sur le service *Pages* de **GitLab.**
- Le serveur, hébergé sur une machine virtuelle dans le cloud **AWS.**
- L'utilisateur, qui communique avec le serveur et le serveur Pages de GitLab.

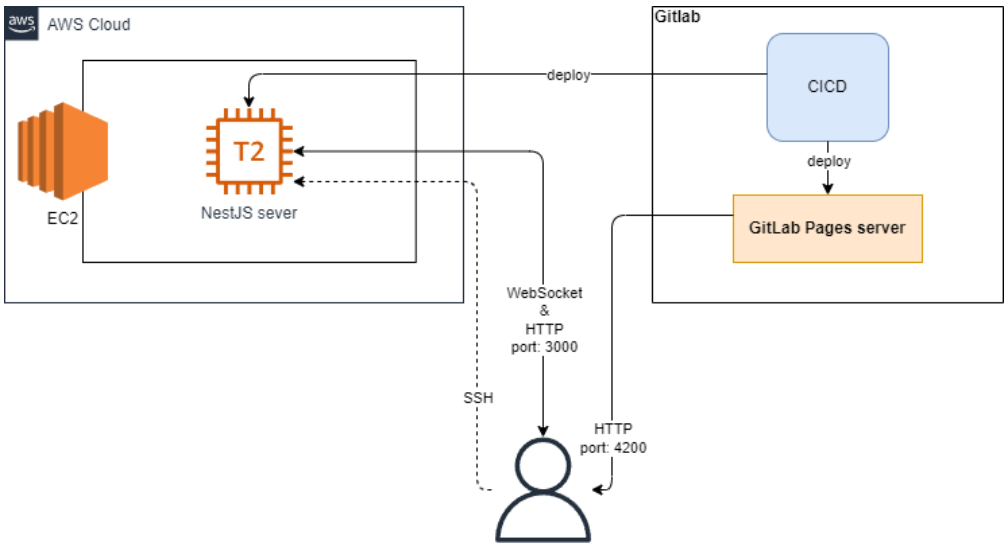


Figure 12: Schéma de l'environnement déployé

Dans l'ordre des choses, lorsque toute notre infrastructure est prête à être déployée, il nous suffit de créer un tag deploy_vX.X pour que le CI/CD configuré dans le projet, détecte une nouvelle version du projet déployable. C'est donc à ce moment, en lançant les pipelines manuellement, que "client" va être déployé sur Pages, "server" va être déployé sur notre instance EC2 t2.mini (linux). Le déploiement du serveur se fait via une connexion SSH a l'instance, pour lancer l'installation du projet et de ses dépendances et lancer en tâche de fond le serveur NestJS.