Document d'architecture logicielle

Version 2.0

Historique des révisions

Date	Version	Description	Auteur
2023-03-19	1.0	Diagramme de cas d'utilisation: CU-1.0, CU-2.0, CU-3.0, CU-4.0	Debanjan Bhaumick
2023-03-20	1.1	Diagrammes des cas d'utilisation : Jeu des différences, CU-1.0, CU-2.0, CU-3.0, CU-4.0	Gerty Marcy Sima
2023-03-20	1.0	Ajout des diagrammes de paquetages clients et du diagramme de déploiement	Benjamin Leveille
2023-03-21	1.0	Diagramme de paquetages serveur	Asimina Koutsos
2023-03-21	1.0	Diagrammes des cas d'utilisation : CU-5.0, CU-6.0, CU-7..0	Gerty Marcy Sima
2023-03-21	1.1	Diagrammes de séquence 1,2, 3 et 4	Gerty Marcy Sima
2023-03-21	1.0	Introduction	Asimina Koutsos et Gerty Marcy Sima
2023-03-21	1.0	Diagramme de séquence 5	Asimina Koutsos
2023-04-19	2.0	Diagramme de cas d'utilisation: CU-2.0, CU-3.0, CU-4.0, CU-5.0 (en incluant le mode triche et la reprise vidéo avec ses différentes possibilités)	Gerty Marcy Sima
2023-04-19	2.1	Diagramme de cas d'utilisation: Mettre les relations «extends» dans le bon sens	Gerty Marcy Sima
2023-04-20	2.0	Diagramme de séquence sur l'utilisation des indices plus développée	Gerty Marcy Sima
2023-04-20	2.0	Diagramme de séquence sur la reprise vidéo modifiée et plus détaillée	Gerty Marcy Sima
2023-04-20	2.0	Diagramme de séquence 7 : meilleurs temps et message de partie global	Gerty Marcy Sima
2023-04-20	2.0	Corrections des diagrammes de paquetages et déploiement	Benjamin Léveillé

Table des matières

Document d'architecture logicielle

1. Introduction

La conception de notre jeu des différences implique plusieurs paramètres à prendre en compte dans la structure architecturale comme le comportement attendu des joueurs, la communication entre le client et le serveur, etc. Ce document présente l'architecture du jeu à travers quatre vues: cas d'utilisation, processus, logique et déploiement.

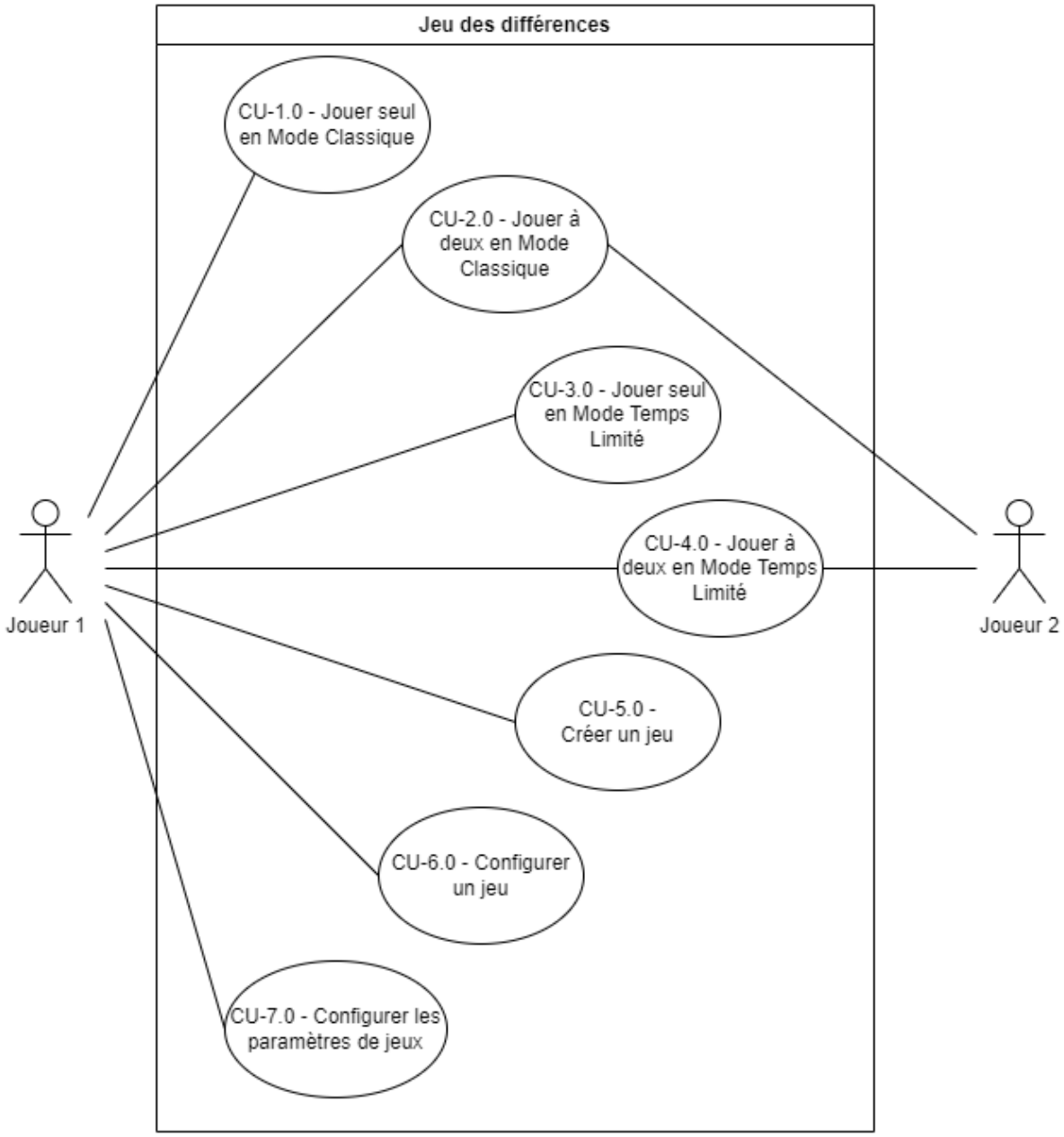
En ce qui concerne les cas d'utilisations, un diagramme global des différents cas est exposé, ainsi que des diagrammes plus détaillés pour la description de chaque cas principal.

Les processus du sprint 3 sont présentés dans des diagrammes de séquences.

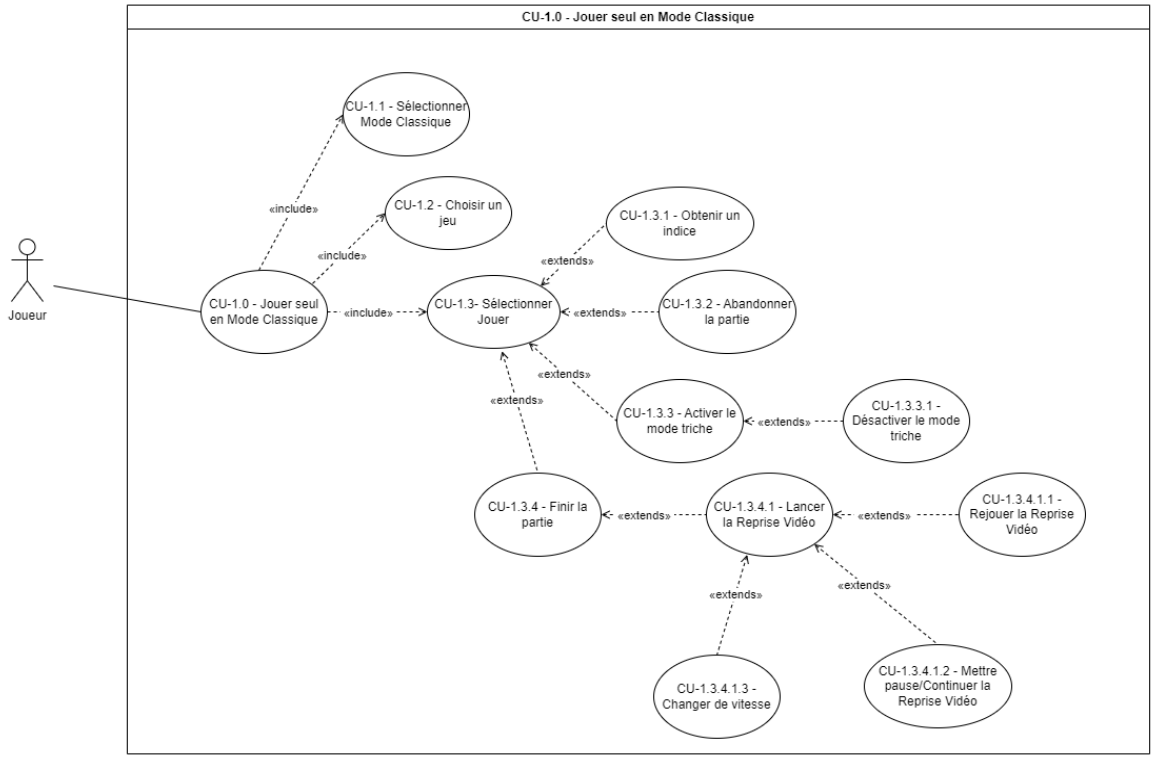
La vue logique, quant à elle, présente les parties principales de l'architecture qui sont divisées en 3 sections, soit les parties reliées au client, au serveur et à la base de données. Ces différentes parties sont présentées avec des diagrammes de paquetages et de classes.

Une vue globale du déploiement est exposée dans un diagramme de déploiement.

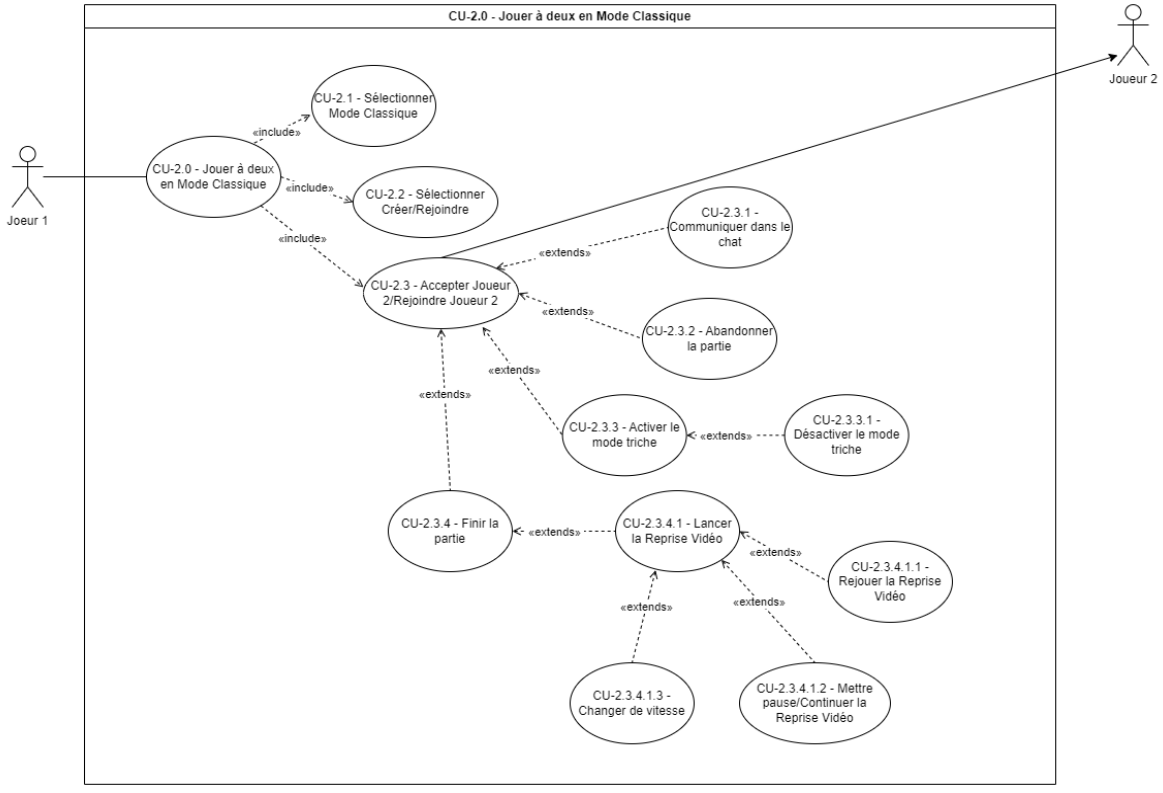
2. Vue des cas d'utilisation



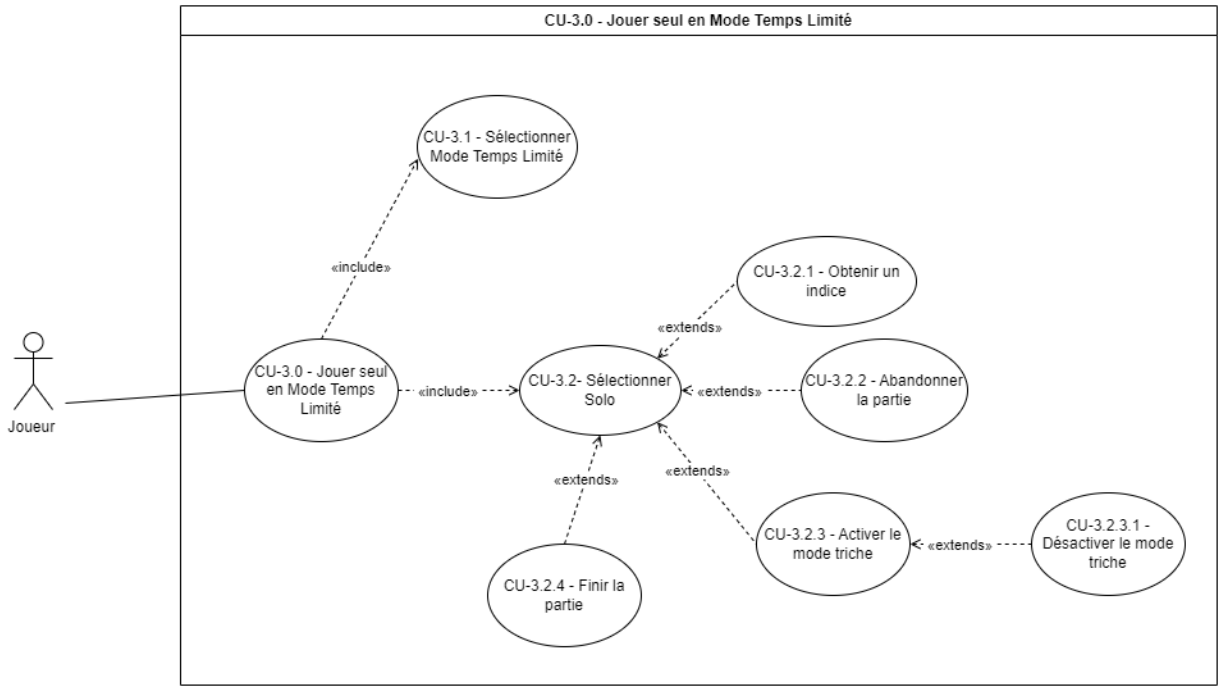
Description de CU - 1.0



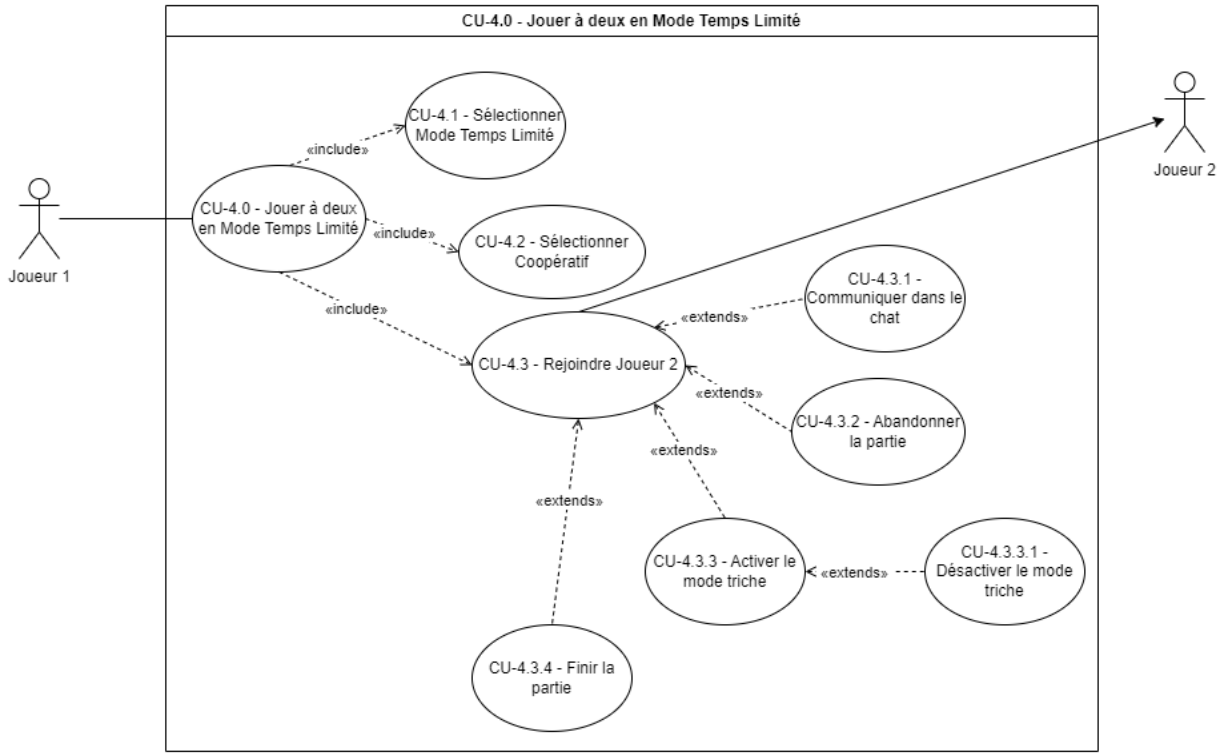
Description de CU - 2.0



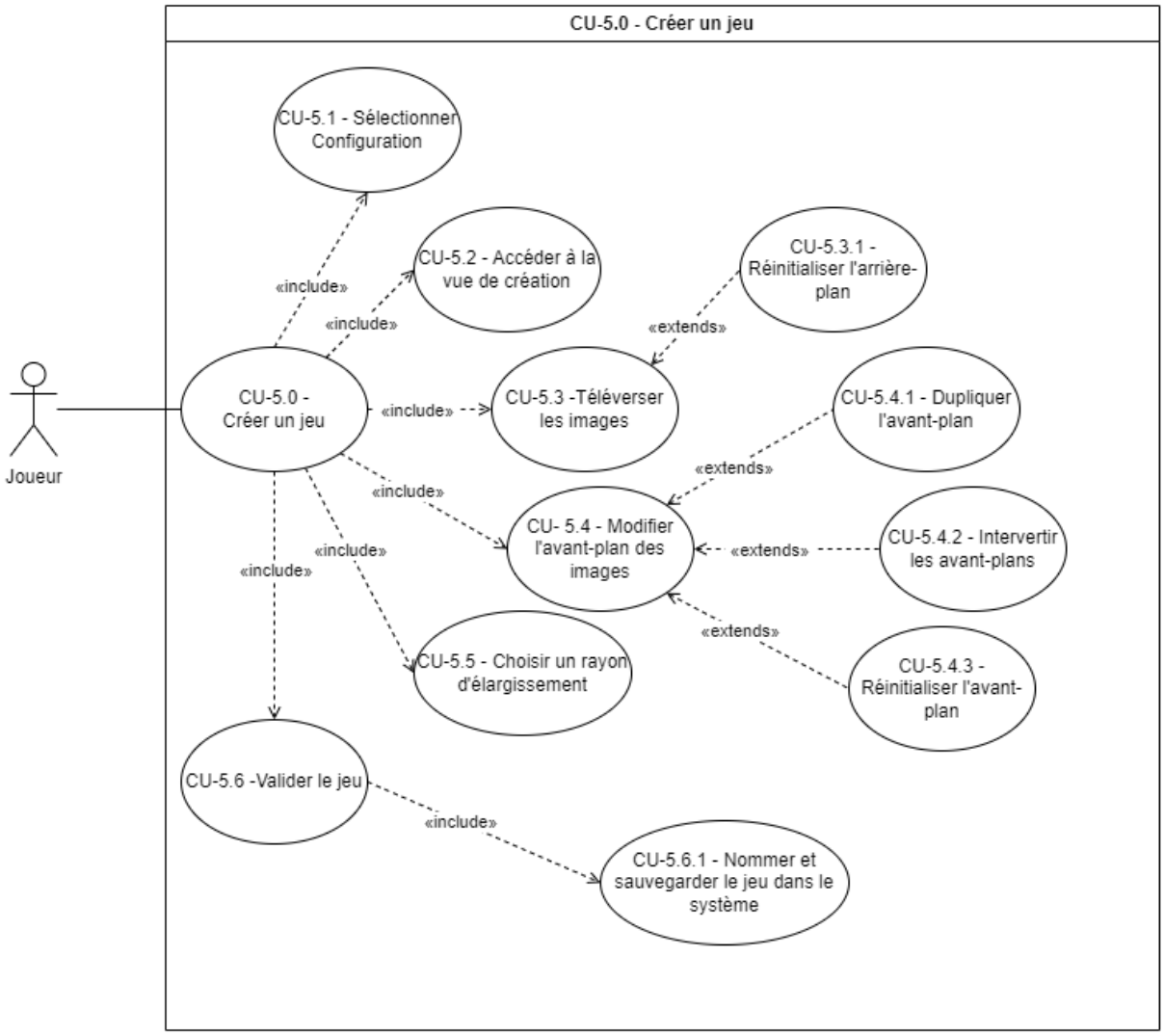
Description de CU - 3.0



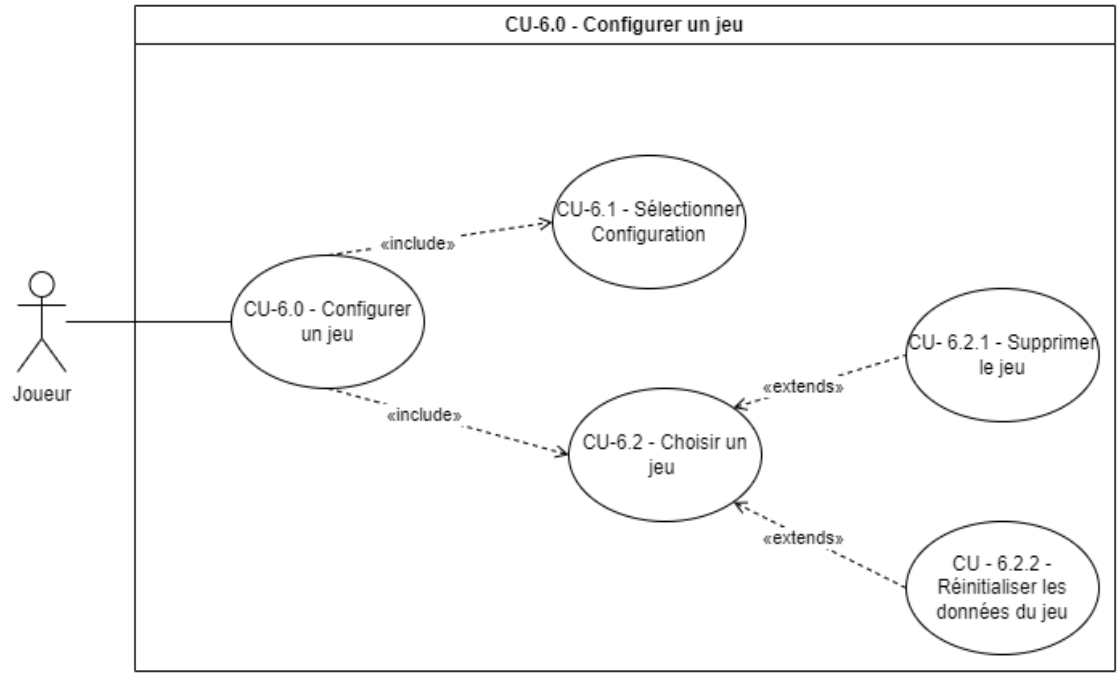
Description de CU - 4.0



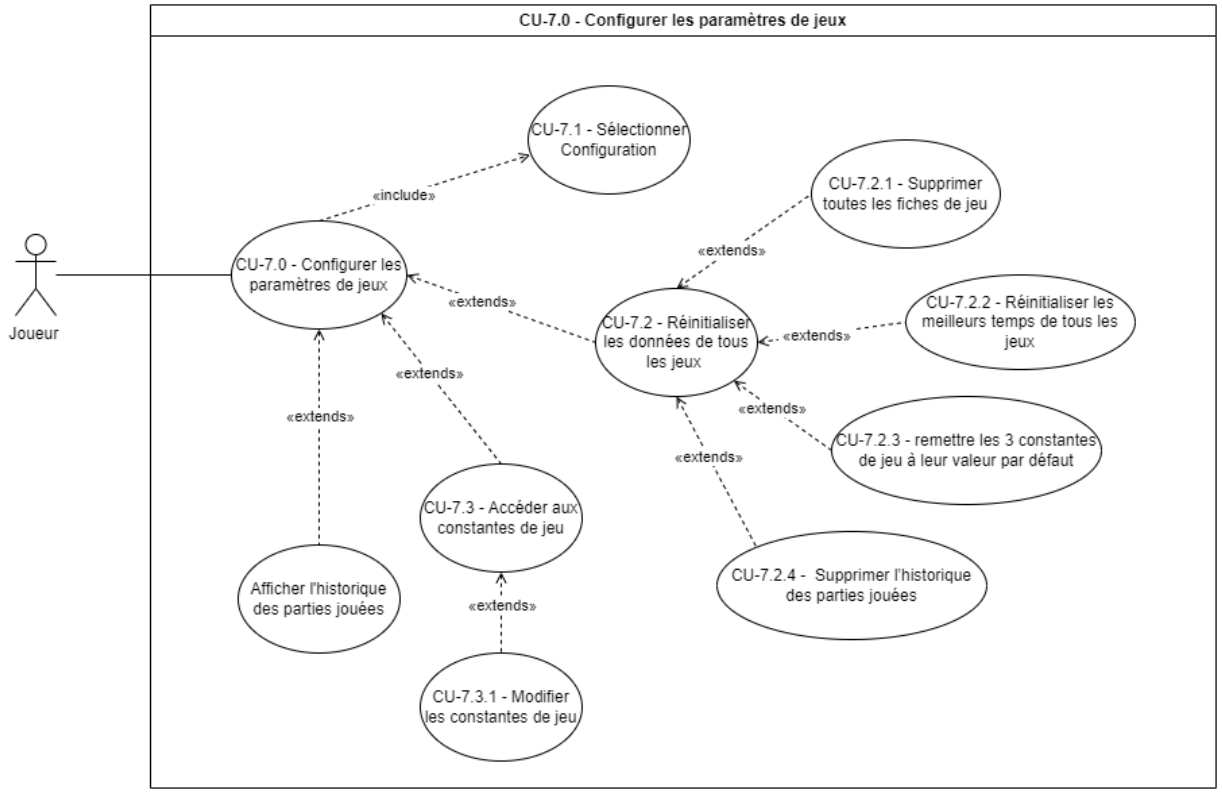
Description de CU - 5.0



Description de CU - 6.0



Description de CU - 7.0



3. Vue des processus

Diagramme de séquence 1: une partie Solo (Classique) avec demande d'indice.

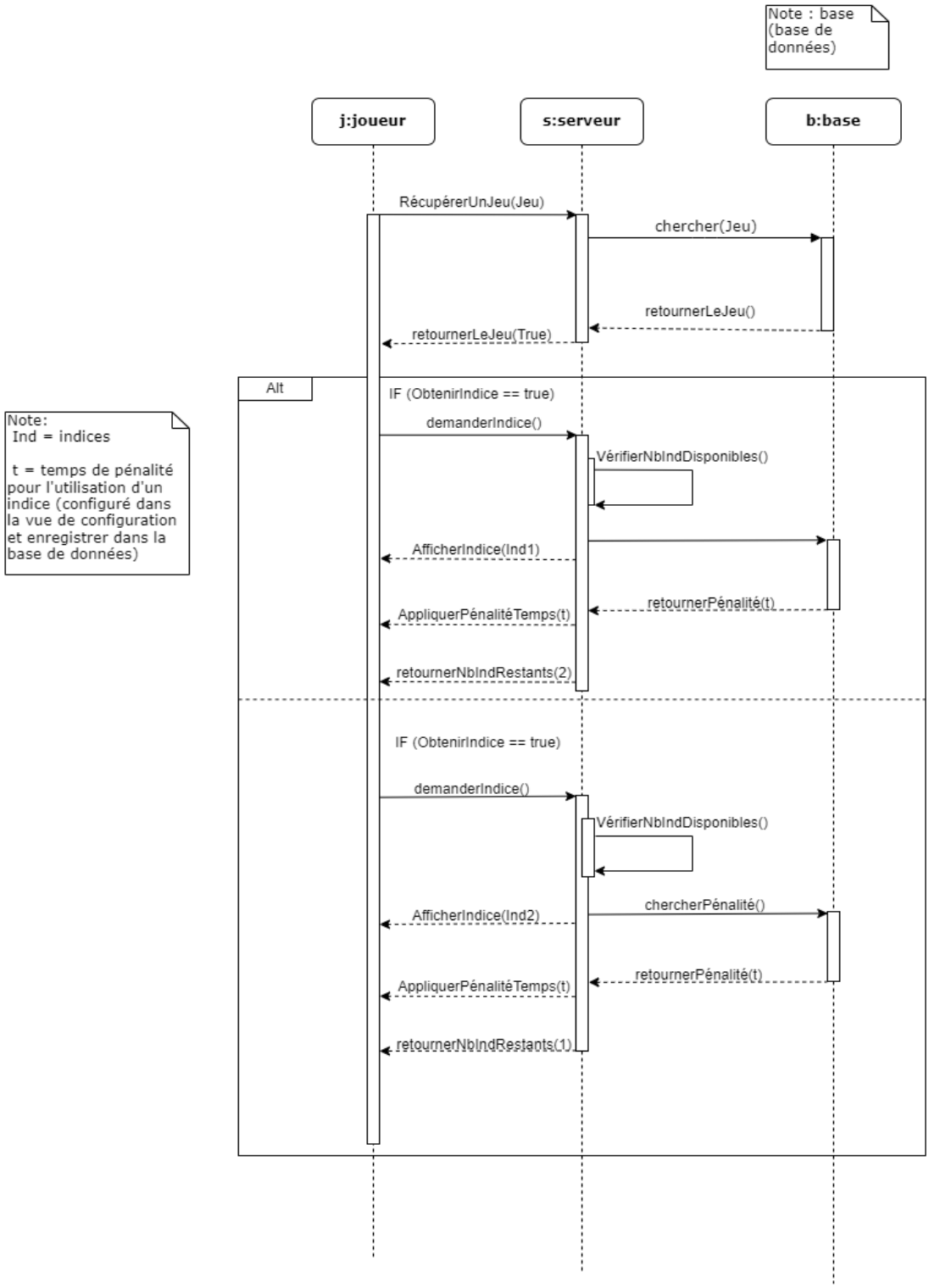


Diagramme 2: Une partie Solo (Classique) avec reprise Vidéo et Mode triche

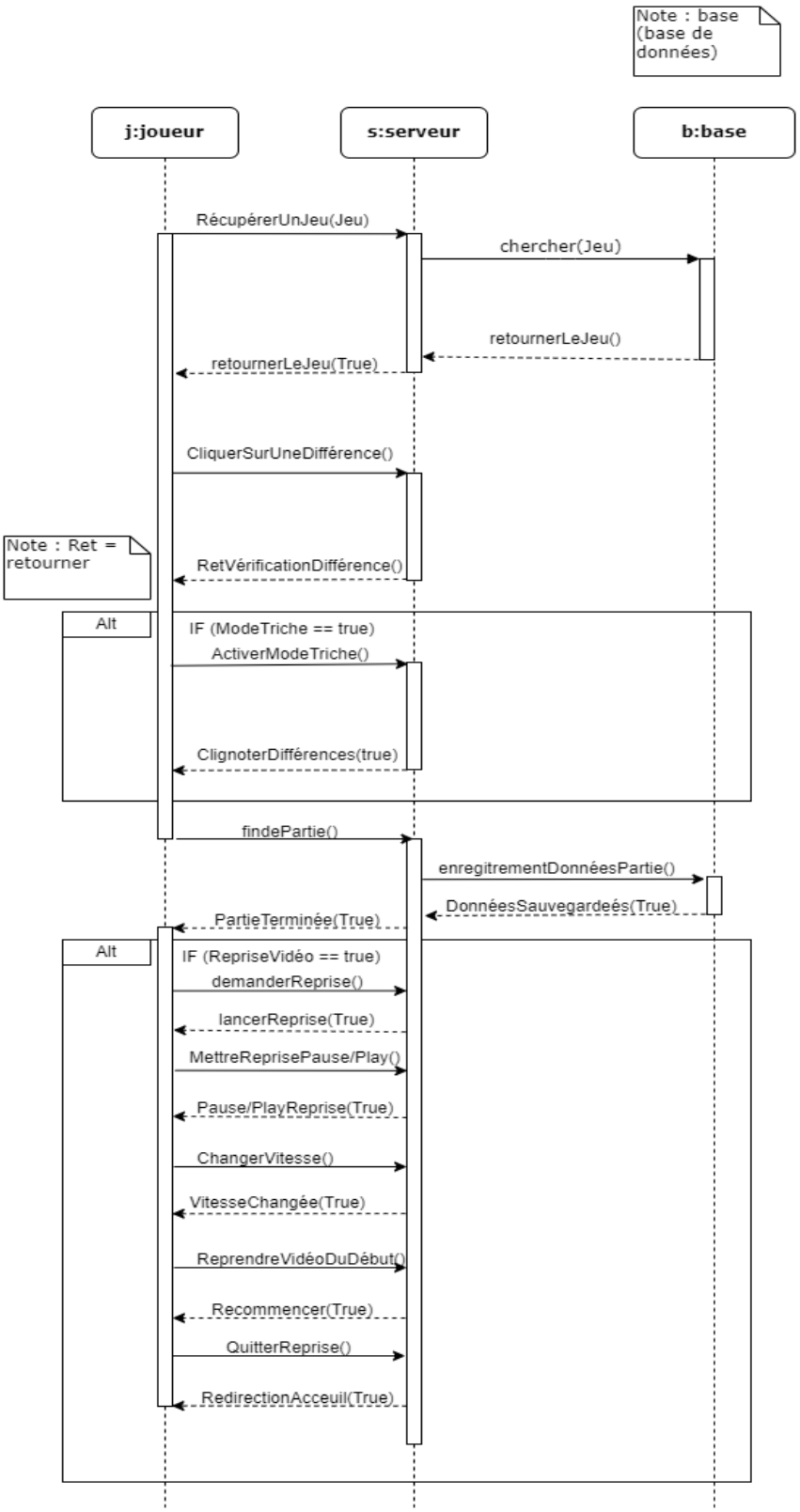


Diagramme de séquence 3: Configurer un jeu

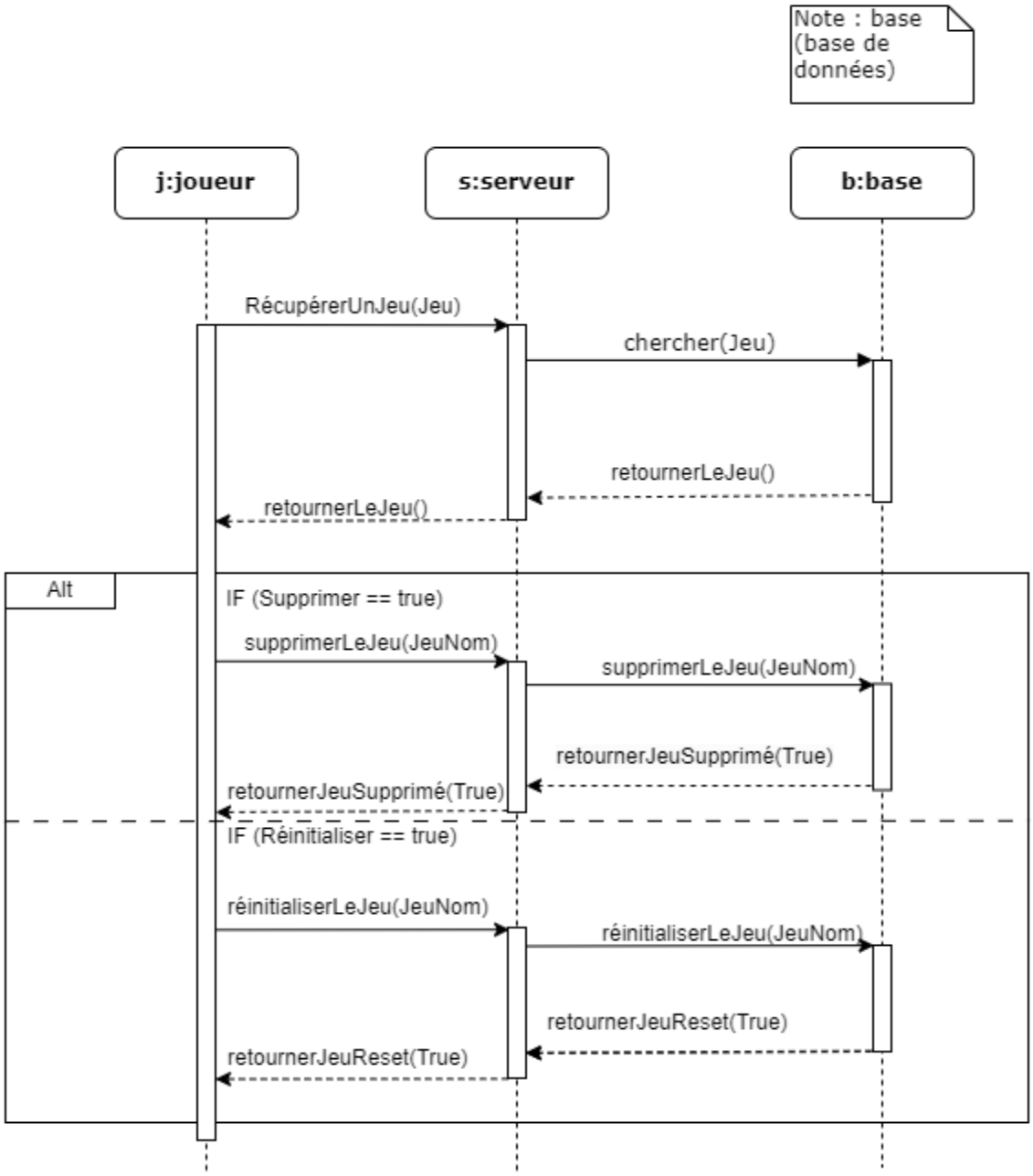


Diagramme de séquence 4: Message dans une partie un contre un ou coopérative

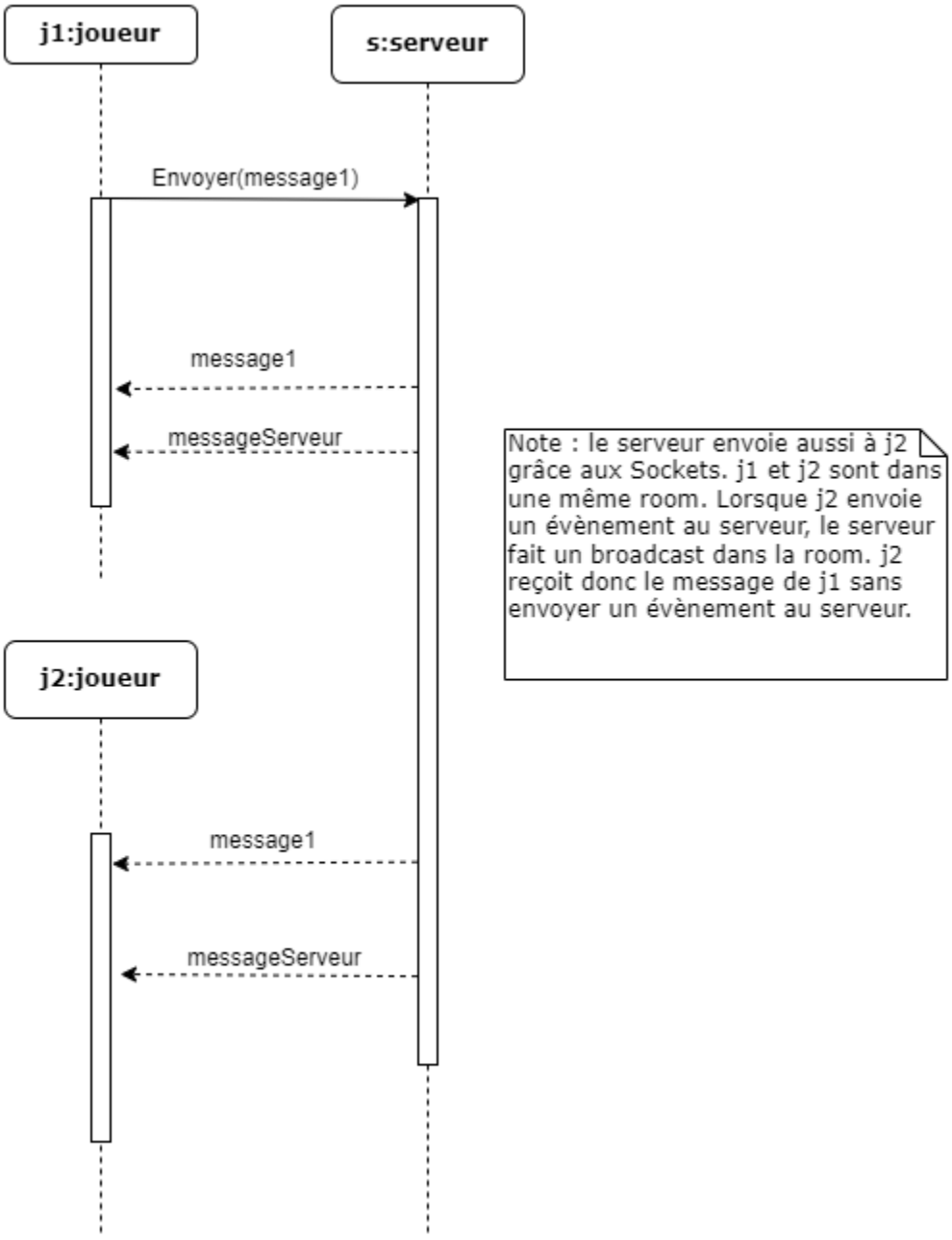


Diagramme de séquence 5: Jouer solo en Mode Temps Limité

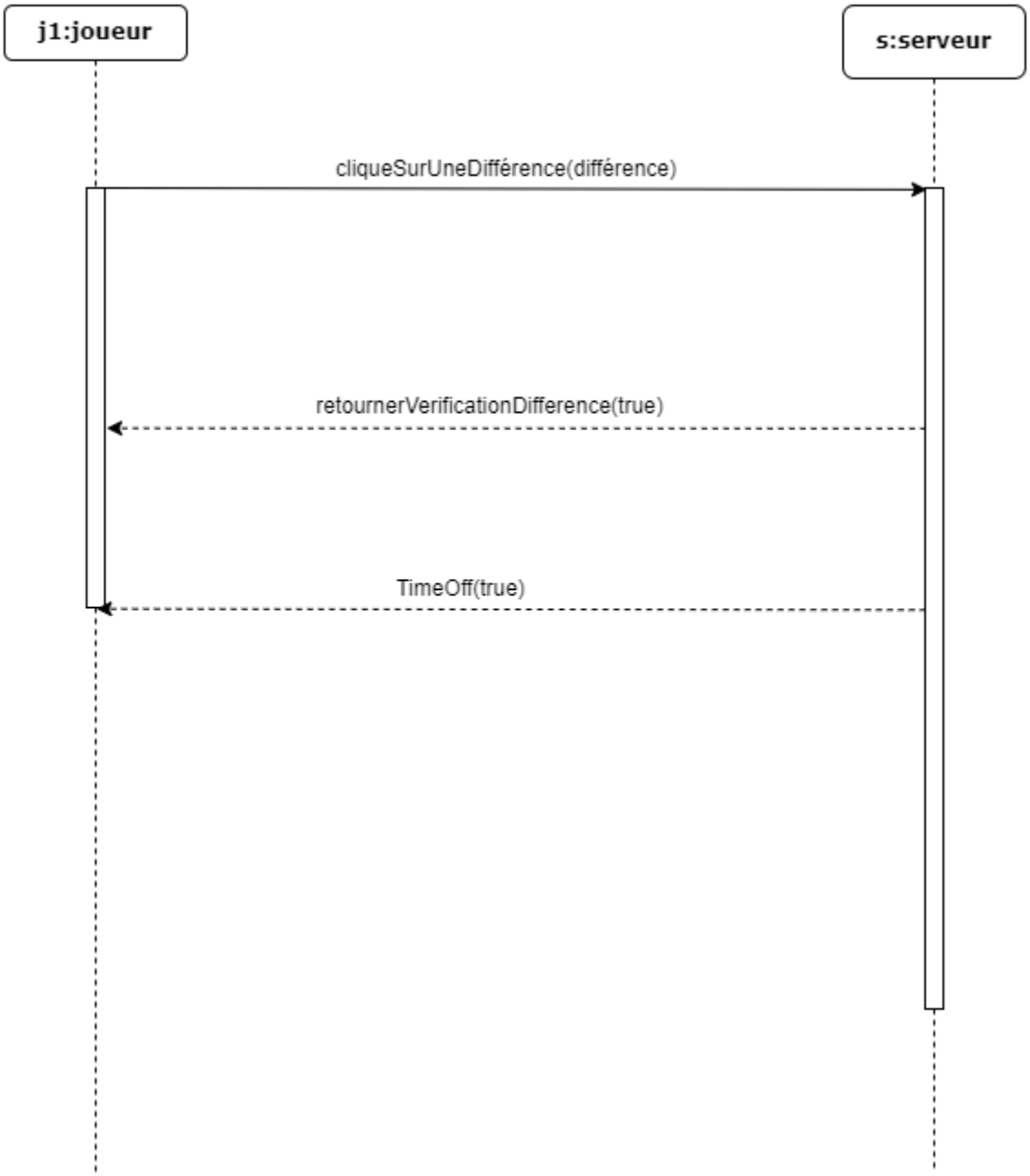


Diagramme de séquence 6: configuration de tous les jeux

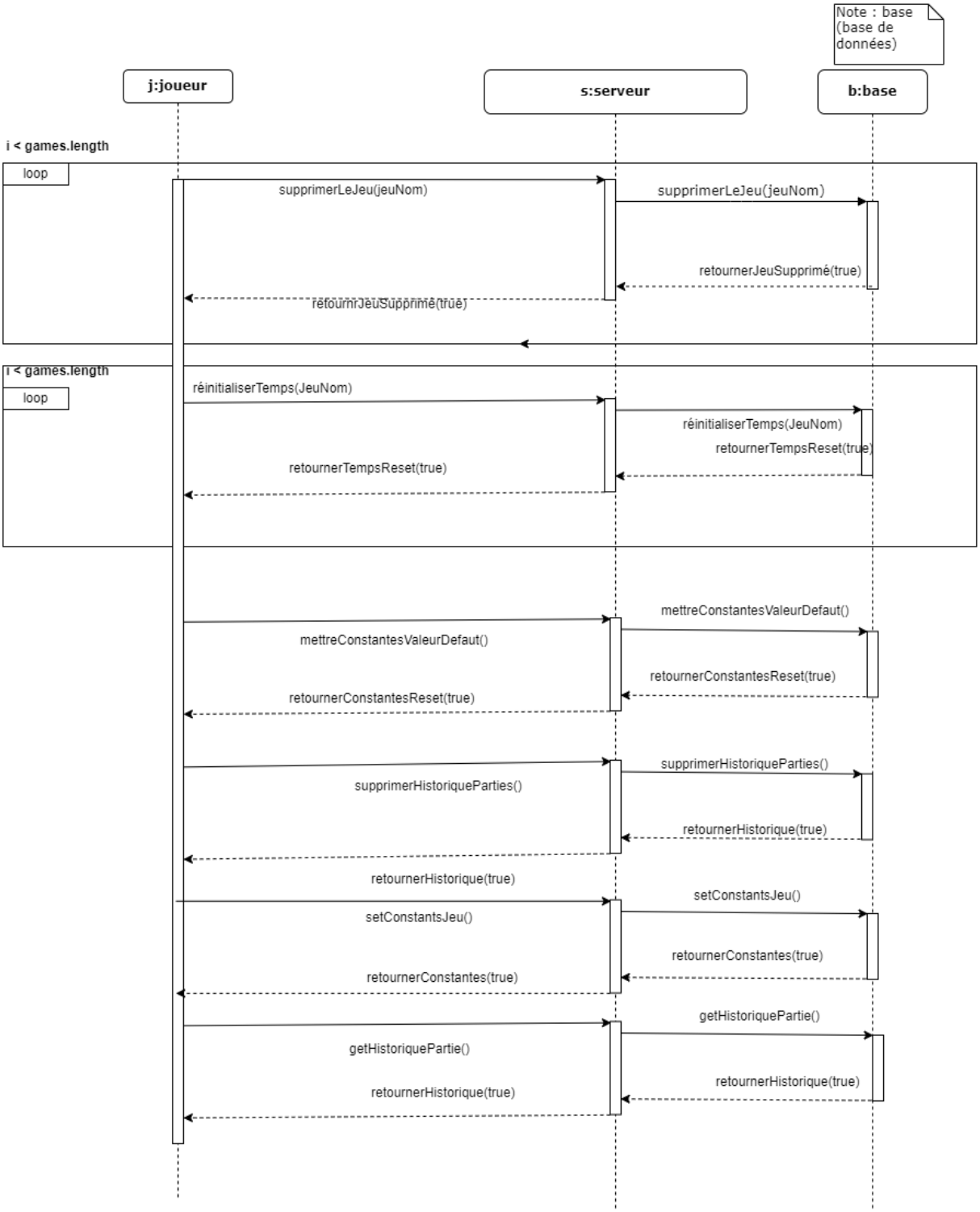
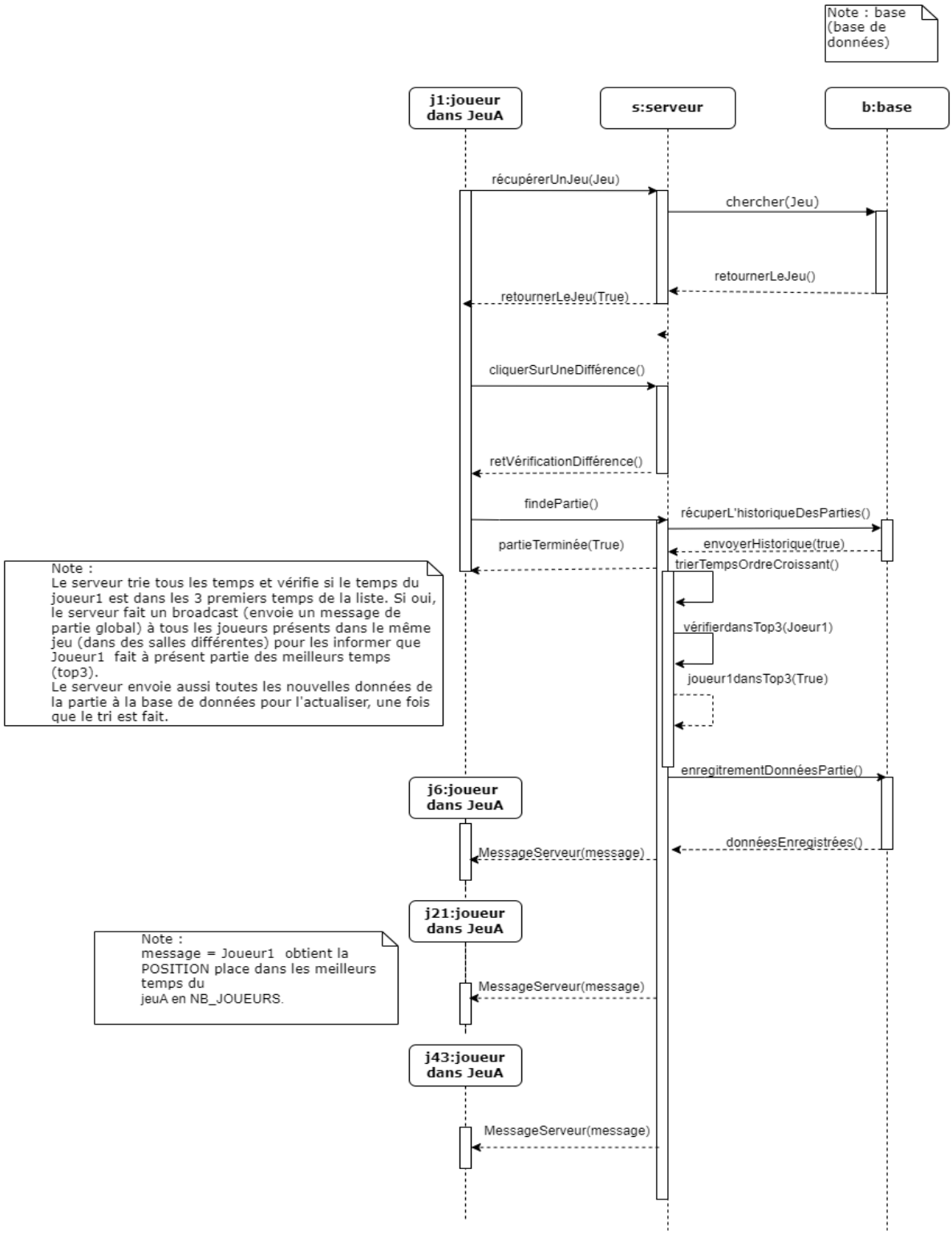
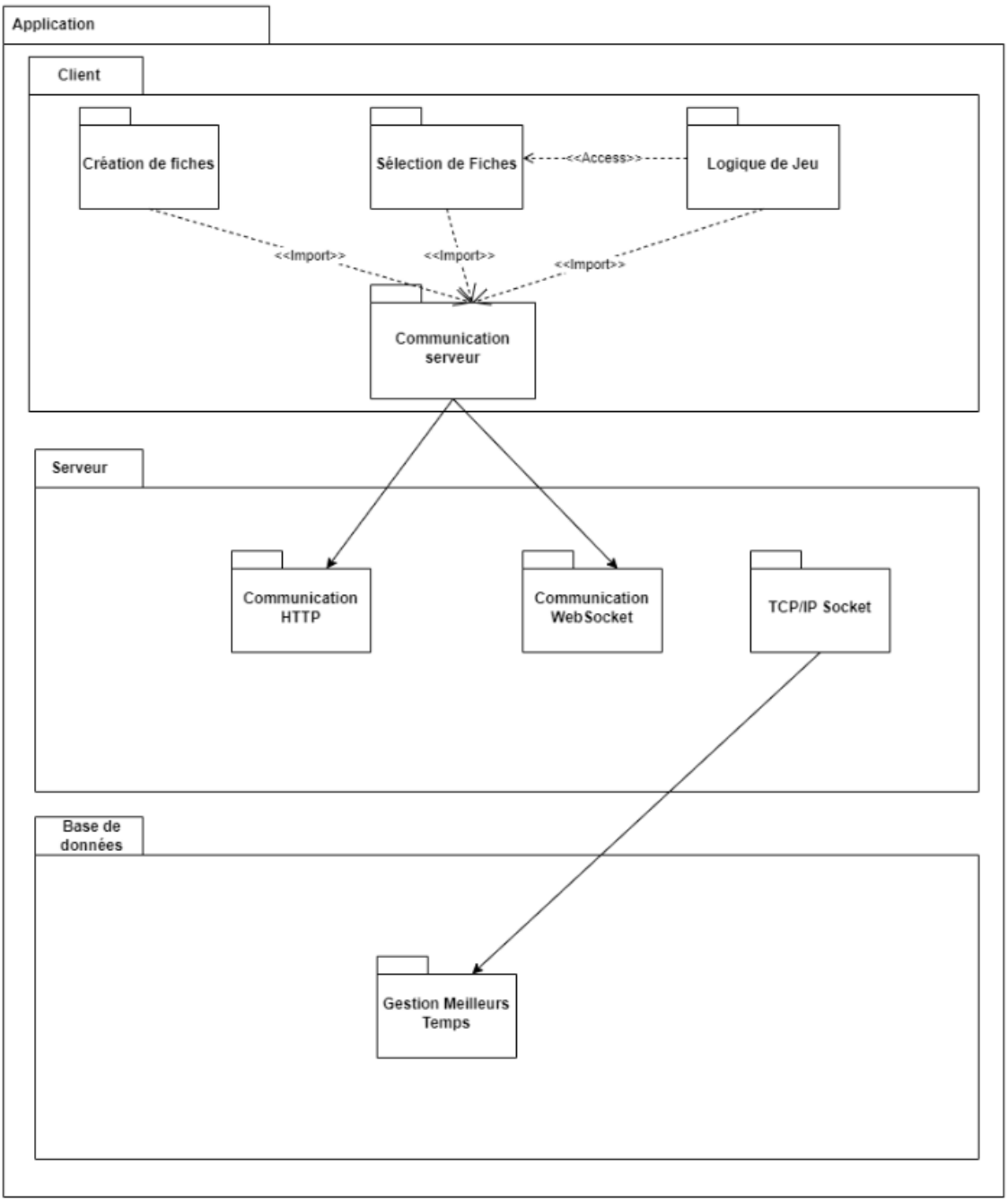


Diagramme de séquence 7: meilleurs temps et message global de partie



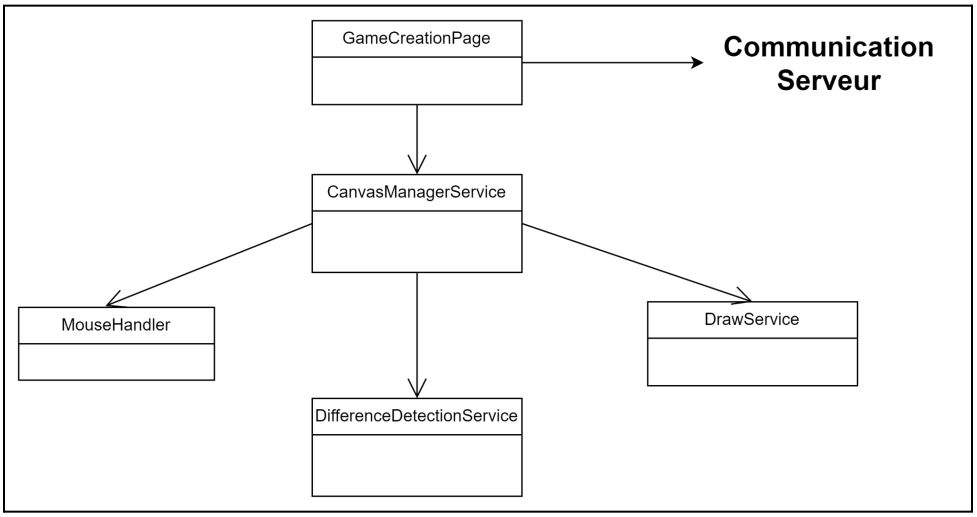
4. Vue logique

Application
Représente le paquetage global de notre application. Divisée en troisparties, le client, le serveur et la base de données.



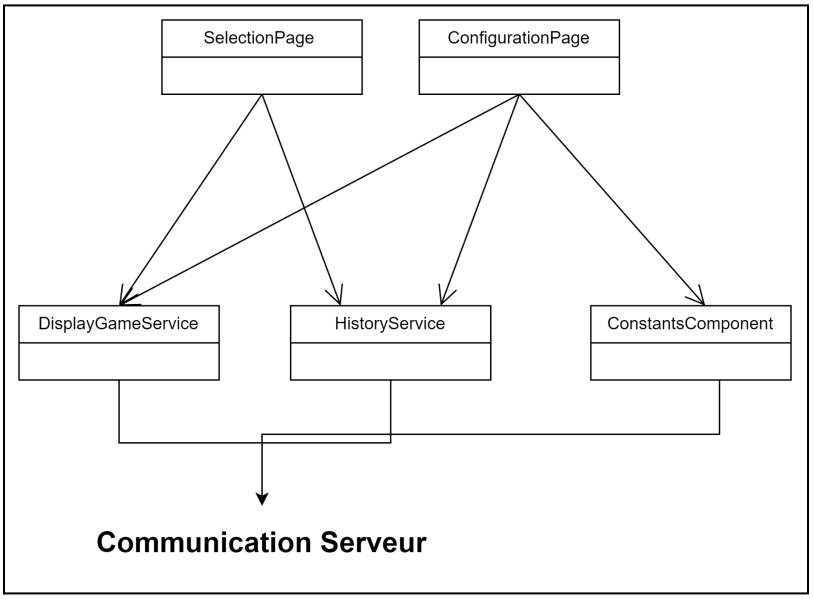
Paquet Création de Fiches

Regroupe les Components et services servant à la création de fiches. MouseHandler gère les positions et états de la souris lors de clics sur les canevas. DrawService s'occuppe de faire les dessins sur les canevas. DifferenceDetectionService s'occupe de faire les calculs nécessaires pour trouver les différences entre deux images. CanvasManager s'occupe de gèrer les évènement reçus de la vue et de mettre à jour celle-ci.

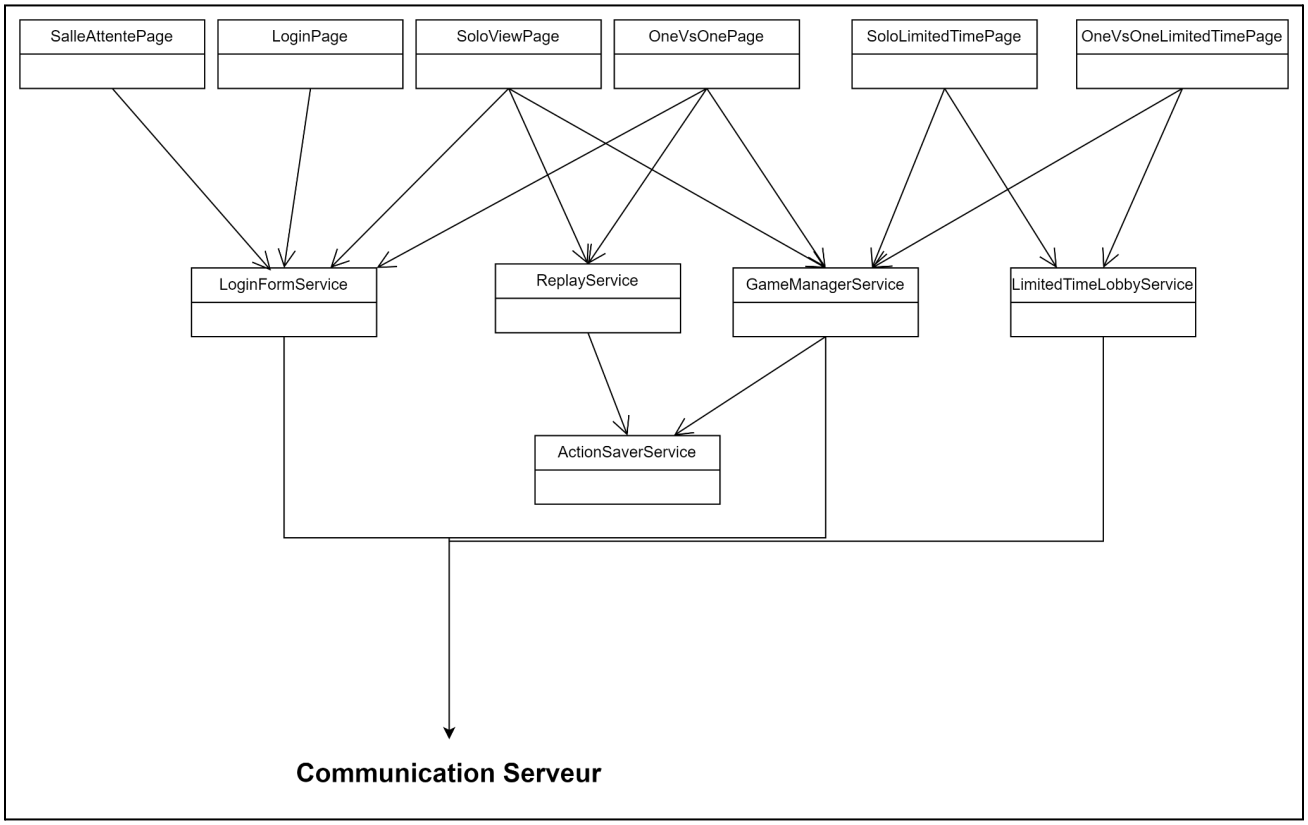


Paquet Sélection de fiches

Regroupe les Components et services servant à l'affichage des listes de jeu. DisplayGame Service s'occupe de l'affichage des grilles de jeu. HistoryService s'occupe de gérer les historiques et les meilleurs temps. ConstantsComponent gère les paramètres de jeu.



Paquet Logique de jeu
Regroupe les Components et services servant à gérer le déroulement complet d'une partie de jeu. LoginFormService permet de gérer les noms d'utilisateurs, id de parties, de connecter les joueurs,etc. LimitedTimeLobbyService fait la même chose pour les parties temps limité. GameManager permet de gérer les actions des joueurs sur la vue, ainsi que la logique de jeu.. ActionSaverService sert à enregistrer les actions faites par l'utilisateur, et ReplayService permet de gérér la reprise des parties classiques.



Paquet Communication Serveur

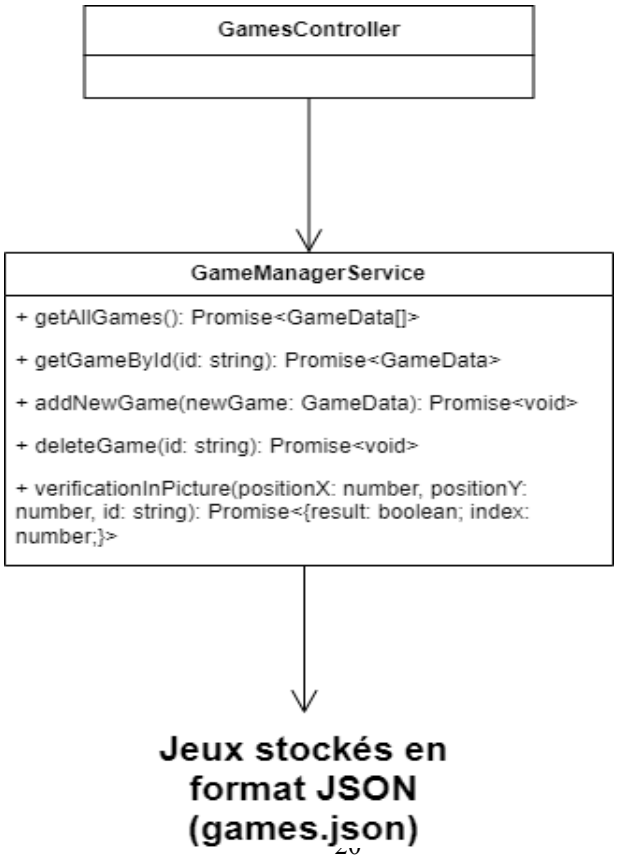
Regroupe les Services qui s'occupent de faire les communications au serveur. Communication Service s'occupe d'envoyer les requêtes HTTP au serveur, alors que Socket Client Service gère les communications faites par Socket.io

CommunicationService
+ getGameById(string): GameData
+ addNewGame(GameData): void
+ deleteGame(string): void

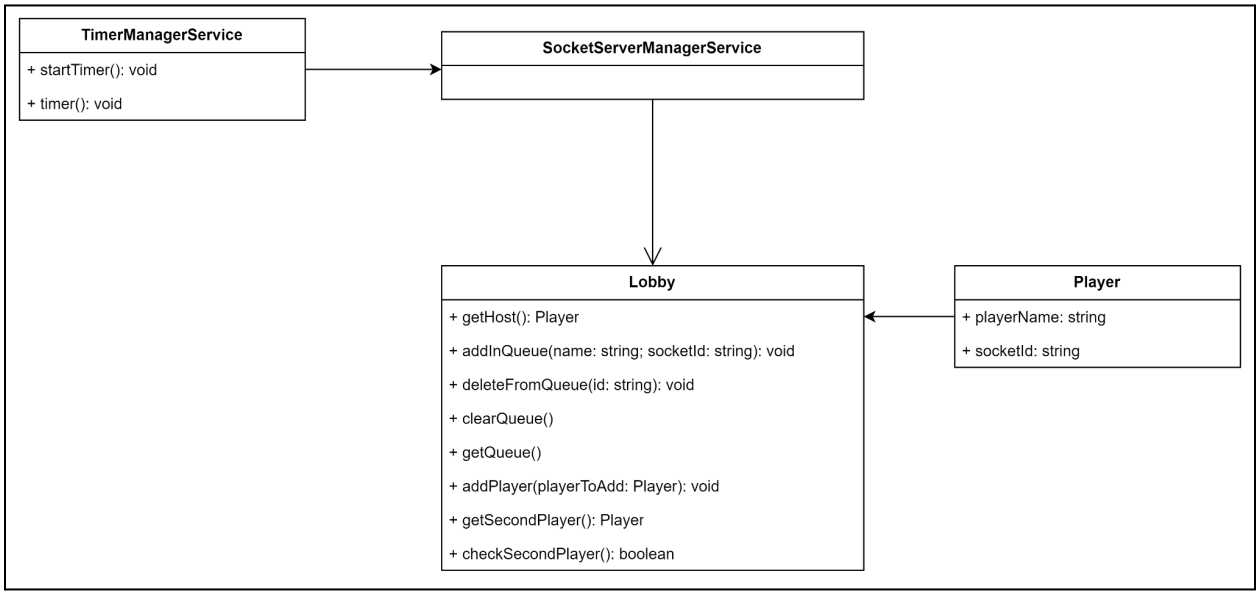
SocketClientService
+ connect(): void
+ disconnect():void
+ on(string, action)
+ send(string, data)

Paquet Communication HTTP

Regroupe les Services qui s'occupent de recevoir les requêtes HTTP du client et exécuter l'action voulue. GamesController s'occupent de recevoir les requêtes (get, post, delete) selon la route spécifiée et utilise les méthodes de GameManagerService. GameManagerService définit les méthodes qui interagissent avec les jeux stockés dans le fichier games.json



Paquet Communication WebSockets
Regroupe les Services qui s'occupent de gérer les événements WS transmis entre le client et le serveur. Le Service SocketServerManager s'occupe de recevoir les événements du client et d'en émettre vers le client. La Classe Lobby gère la queue de file d'attente pour être acceptée dans une partie créée. La classe Player définit la structure des joueurs se trouvant dans la queue. Le Service TimerManager gère le timer qui apparaît sur la vue 1v1 des 2 joueurs.



5. Vue de déploiement

Diagramme de déploiement du système