101

Document d'architecture logicielle

Version 2.9

Historique des révisions

Date	Version	Description	Auteur
2023-03-12	1.0	Rédaction de l'introduction	Edgar Kappauf
2023-03-16	1.0	Rédaction du premier jet des cas d'utilisation	Sulayman Hosna
2023-03-18	1.0	Début des diagrammes de séquences	Sulayman Hosna
2023-03-19	1.1	Rédaction de la vue logique	Edgar Kappauf et Jeremy Ear
2023-03-20	1.2	Suite des diagrammes de séquences	Sulayman Hosna, Mathieu Prévost, Zakaria Zair
2023-03-21	1.3	Fin des diagrammes de séquences	Mathieu Prévost, Zakaria Zair
2023-03-21	1.4	Révision des cas d'utilisation	Edgar Kappauf et Zakaria Zair
2023-03-21	1.5	Rédaction de la vue logique	Edgar Kappauf et Jeremy Ear
2023-03-21	1.6	Réalisation du diagramme de déploiement	Edgar Kappauf et Mathieu Prévost
2023-03-21	1.7	Révision complète du document	Zakaria Zair
2023-03-24	1.8	Correction des idées architecturales	Abderrahim Zebiri
2023-04-10	2.0	Rectification des erreurs sur les diagrammes de cas d'utilisation	Sulayman Hosna
2023-04-10	2.1	Rectification des erreurs sur la vue logique pour la partie client	Edgar Kappauf
2023-04-12	2.2	Rectification du diagramme de séquence Jouer une partie en temps limité (CU 8.0)	Sulayman Hosna
2023-04-13	2.3	Rectification des erreurs sur la vue logique pour la partie serveur	Jeremy Ear
2023-04-14	2.4	Rectifications Diagrammes de séquences : Voir l'historique d'une partie (CU 2.0)	Sulayman Hosna
2023-04-16	2.5	Ajout Diagramme de séquences : Réinitialiser les données (CU 5.0)	Sulayman Hosna
2023-04-17	2.6	Rectification du diagramme de déploiement	Edgar Kappauf et Sulayman Hosna
2023-04-17	2.7	Ajout des diagrammes de classes pour la vue logique	Edgar Kappauf et Jeremy Ear
2023-04-20	2.8	Révision finale	Edgar Kappauf, Sulayman Hosna, Zakaria Zair, Mathieu Prévost et Jeremy Ear
2023-04-20	2.9	Correction finale des diagrammes de paquetages	Jeremy Ear, Mathieu Prévost, Abderrahim Zebiri et Edgar Kappauf

Table des matières

1. Introduction **4**

2. Vue des cas d'utilisation **5**

3. Vue des processus **9**

4. Vue logique **13**

5. Vue de déploiement **18**

Document d'architecture logicielle

1. Introduction

Le document présente une vue globale de l'architecture logicielle de notre application web : un jeu des sept différences. Le document est organisé en 4 sections principales : la vue des cas d'utilisation, la vue des processus, la vue logique et la vue de déploiement. Les sections seront majoritairement composées de diagrammes UML tels que vus dans le cours LOG1410.

Premièrement, dans la première section, nous présenterons les cas d'utilisation du modèle de cas d'utilisation, en mettant l'accent sur les fonctionnalités du Sprint 3 et certaines fonctionnalités du sprint 1 et 2.

Puis dans la deuxième section, nous présenterons les processus du système qui mérite un diagramme de séquence pour mieux comprendre certains cas d'utilisation qui sont plus complexes.

Ensuite, dans la troisième section, nous présenterons les parties significatives au niveau architectural de notre système grâce à des diagrammes de paquetages.

Enfin, dans la dernière section, nous décrirons la vue de déploiement des différentes parties du système. C'est-à-dire les nœuds physiques, les interconnexions et les protocoles de communication utilisés sous la forme d'un diagramme de déploiement.

2. Vue des cas d'utilisation

Les diagrammes de cas d'utilisation permettent de mieux décrire les interactions entre les acteurs et le système pour chaque fonctionnalité. Le premier diagramme de cas d'utilisation présente le format étendu avec les Acteurs (les deux joueurs) et les cas d'utilisation générales qui seront plus détaillés dans les quatres diagramme de cas d'utilisation détaillés qui vont suivre le premier diagramme.

Note: L'ensemble des cas d'utilisation qui n'ont pas de numéro sont des cas d'utilisation qui proviennent du Sprint 1 ou 2.

Figure 1 : Diagramme des cas d'utilisation généraux

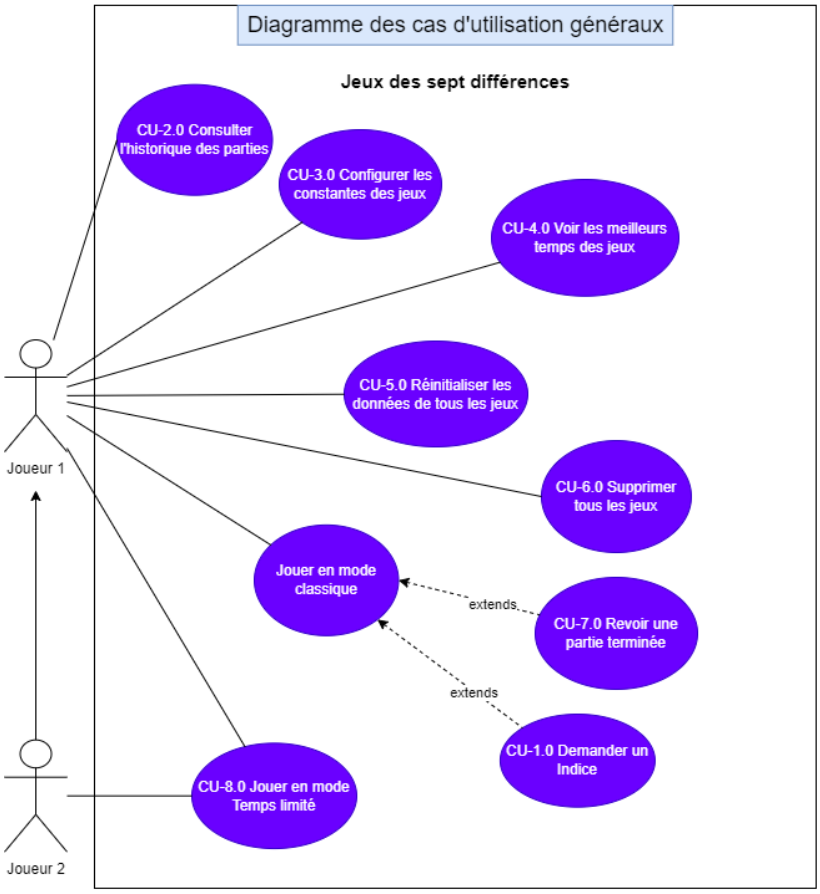


Figure 2 : Diagramme de cas d'utilisation CU-1.0 et 2.0 détaillés

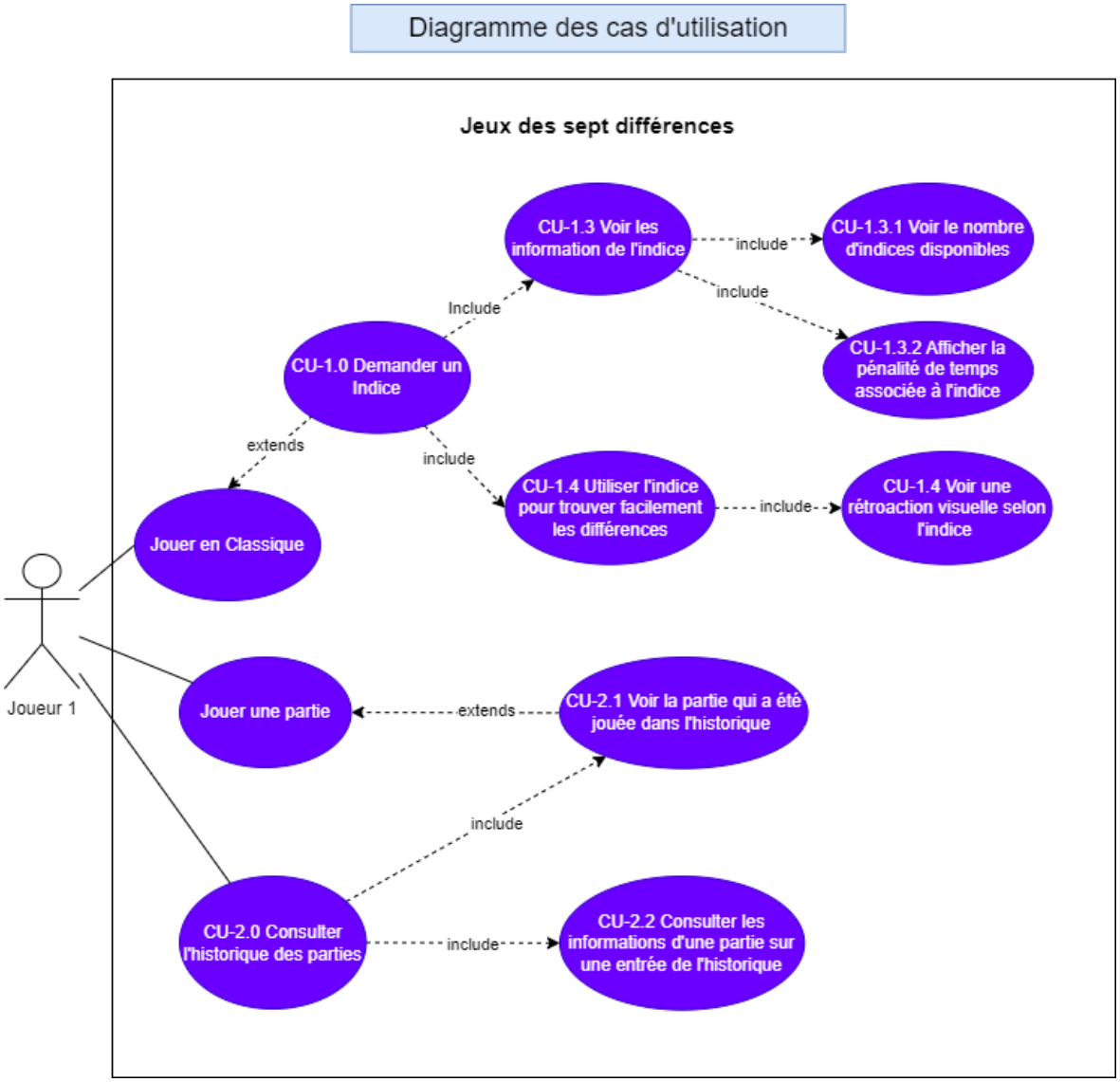


Figure 3 : Diagramme de cas d'utilisation CU-3.0, 4.0, 5.0 et 6.0 détaillés

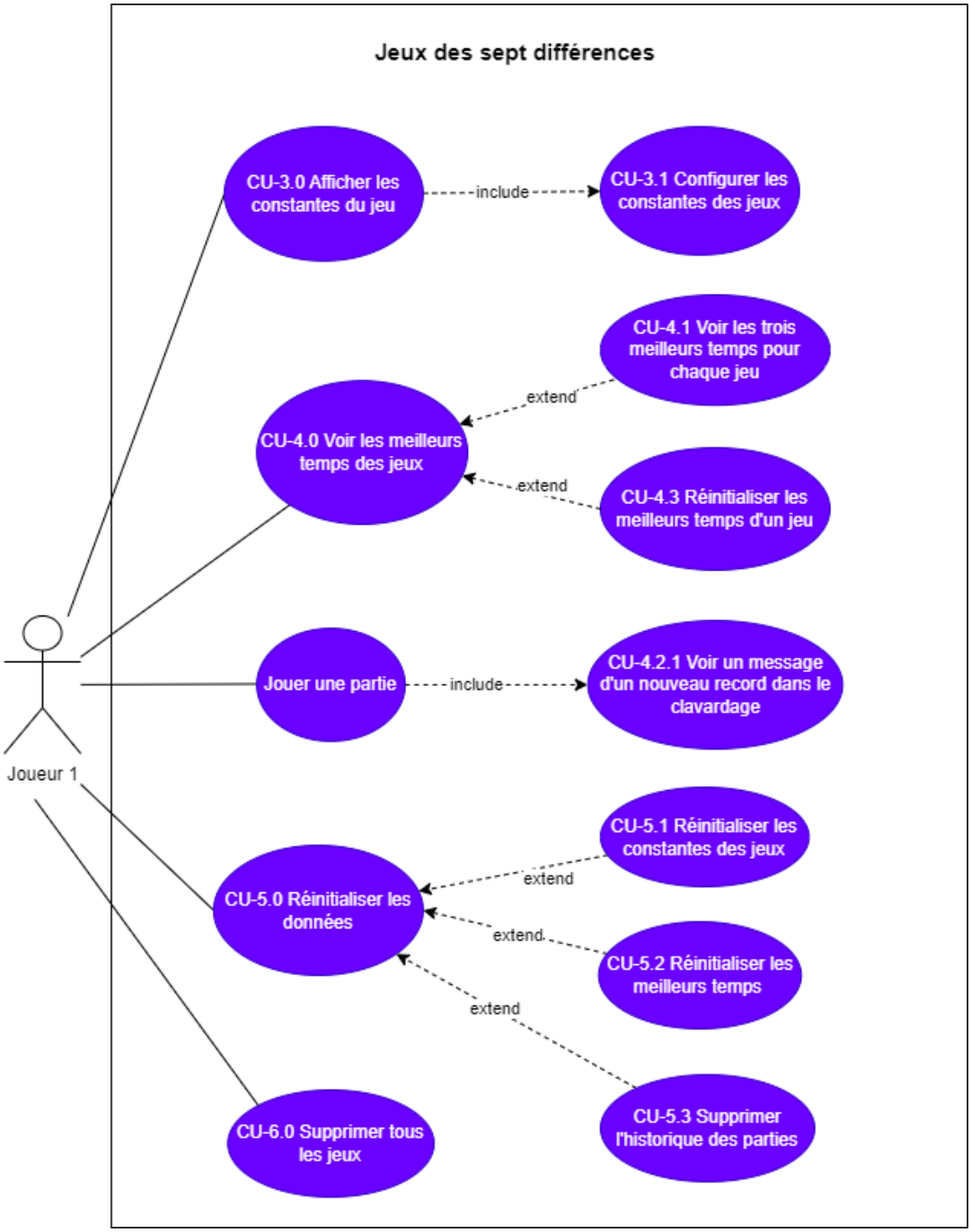


Figure 4 : Diagramme de cas d'utilisation CU-7.0 détaillés

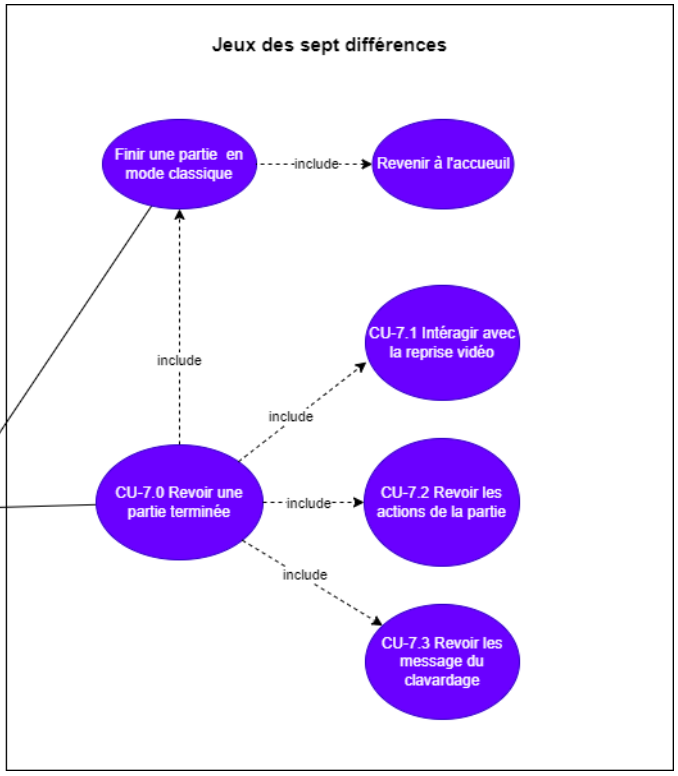
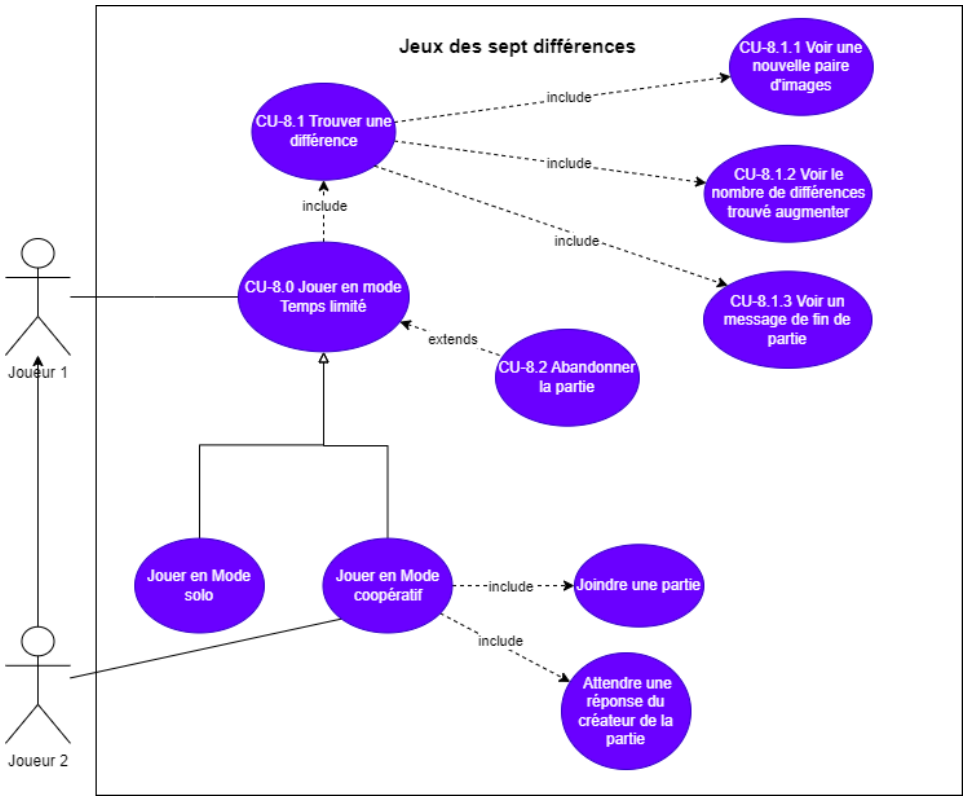


Figure 5 : Diagramme de cas d'utilisation CU-8.0 détaillés



3. Vue des processus

Cette section permet de décrire les interactions entre les différents processus du système. L'accent est mis sur les fonctionnalités importantes du Sprint 3, en identifiant les principaux processus. Nous avons choisi de présenter des diagrammes de séquences pour les points suivants :

- Demander un indice (CU 1.0)
- Voir l'historique des parties (CU 2.0)
- Changer les constantes (CU 3.0)
- Mets à jour les meilleurs temps (CU 4.0)
- Réinitialiser les données (CU 5.0)
- Jouer une partie en temps limité (CU 8.0)
- Regarder la reprise vidéo d'une partie (CU 7.0)

Ces fonctionnalités nécessitent un diagramme de séquence pour mieux montrer les interactions entre les acteurs et le système selon un ordre chronologique .

Figure 6 : Diagramme de séquence : Demander un indice (CU 1.0)

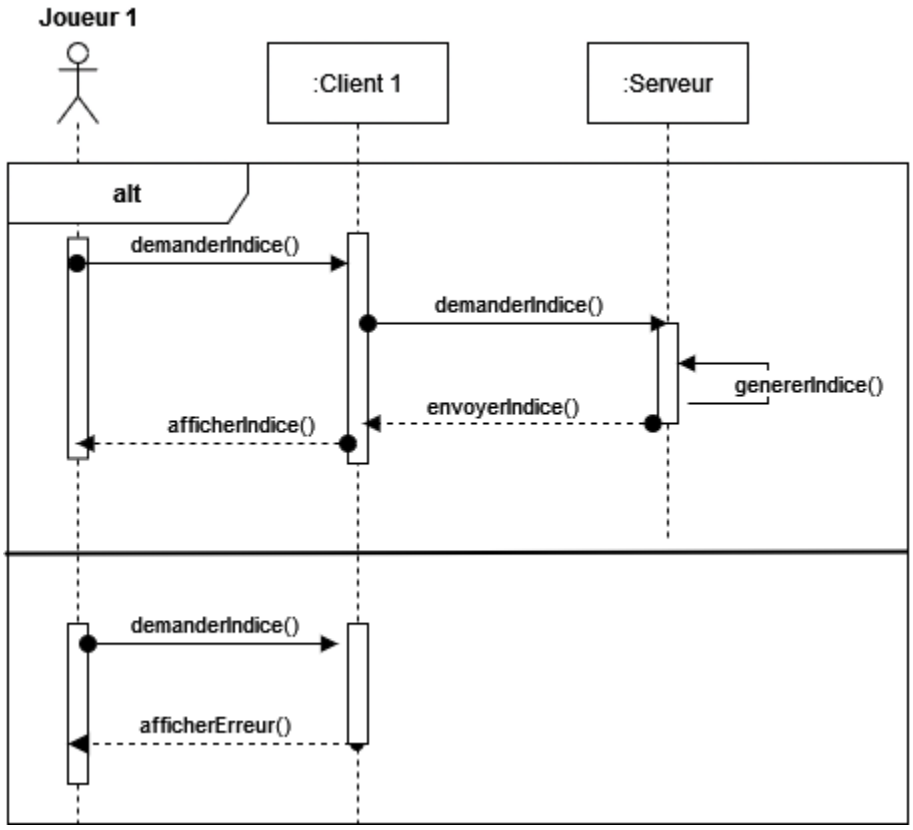


Figure 7 : Diagramme de séquence : Voir l'historique des parties (CU 2.0)

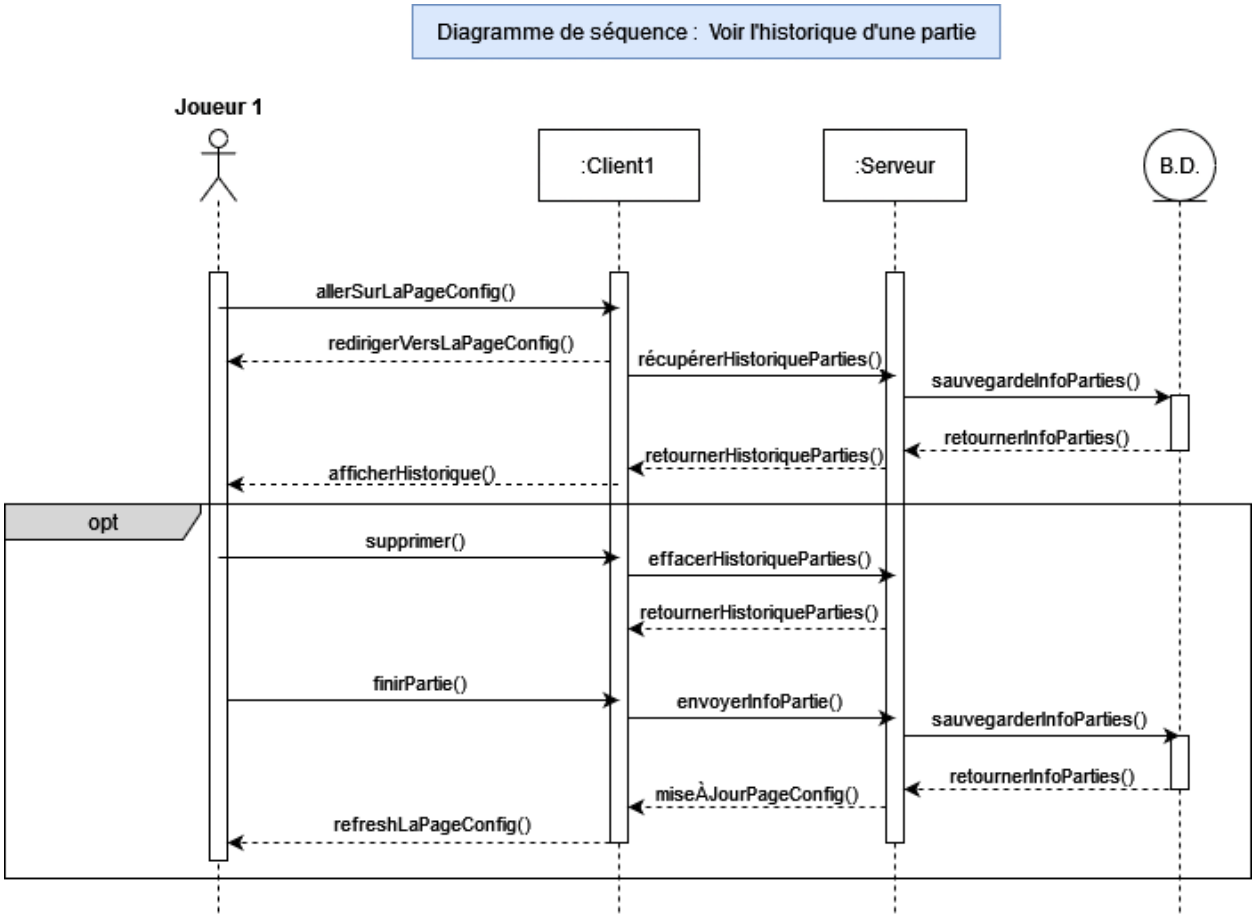


Figure 8 : Diagrammes de séquences : Changer les constantes (CU 3.0)

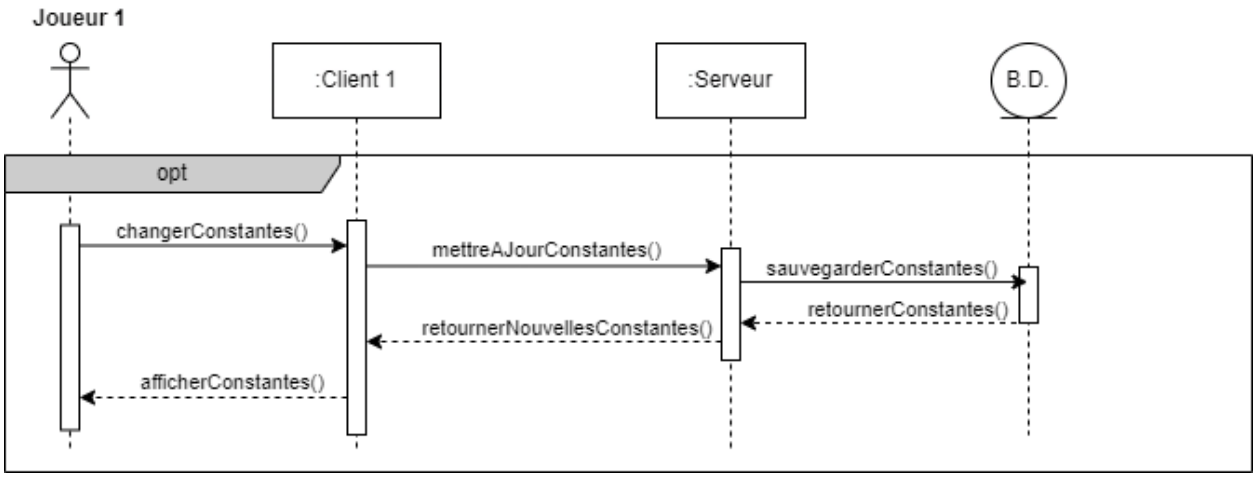


Figure 9 : Diagrammes de séquences : Mettre à jour les meilleurs temps (CU 4.0)

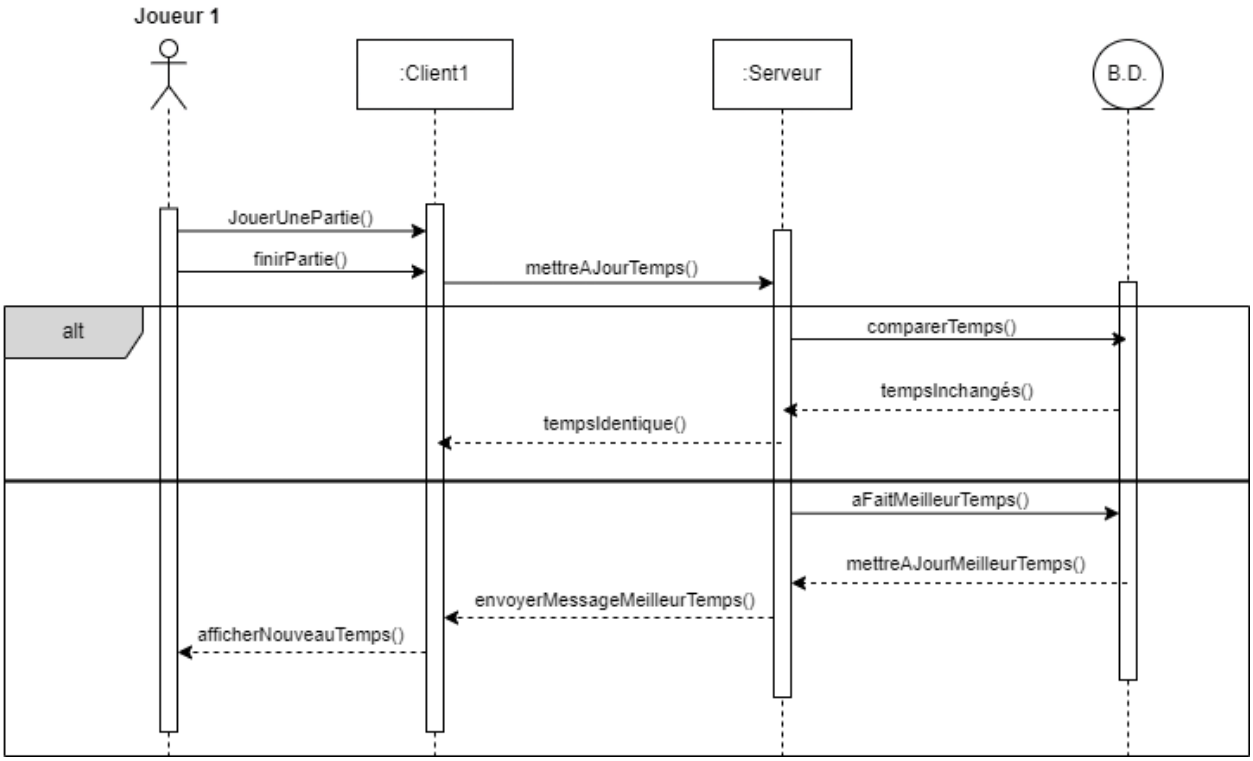


Figure 10 : Diagrammes de séquences : Réinitialiser les données (CU 5.0)

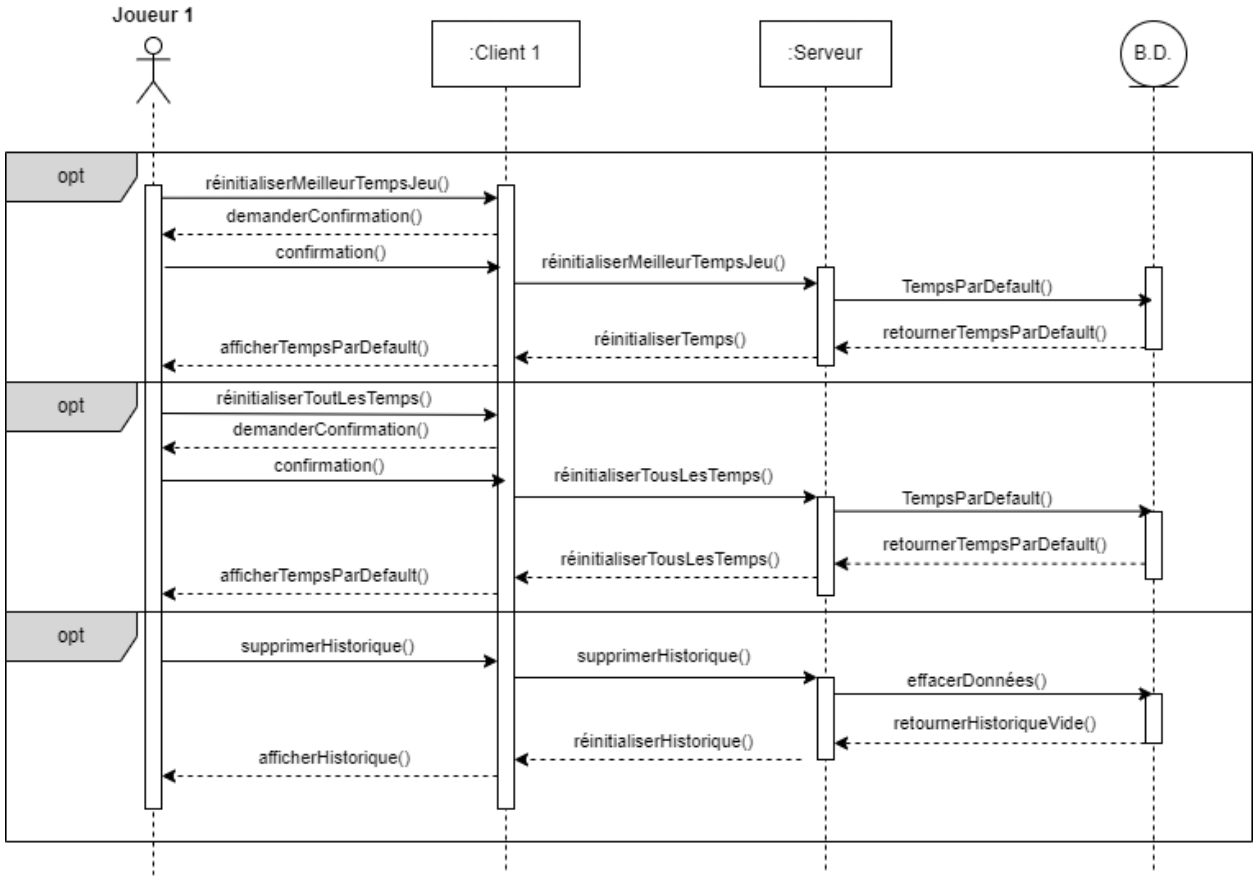


Figure 11 : Diagrammes de séquences : Jouer une partie en temps limité - Connexion (CU 8.0)

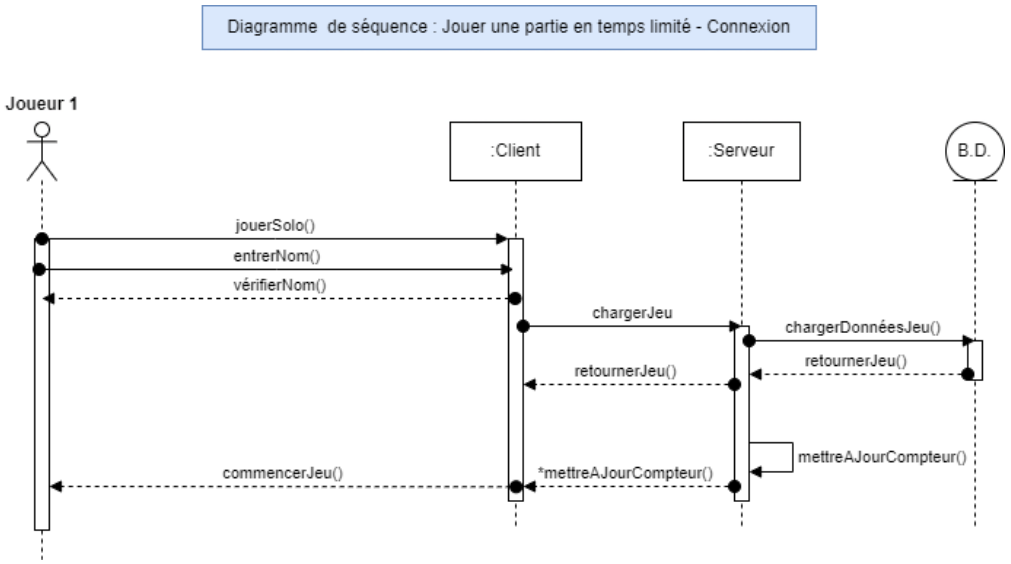
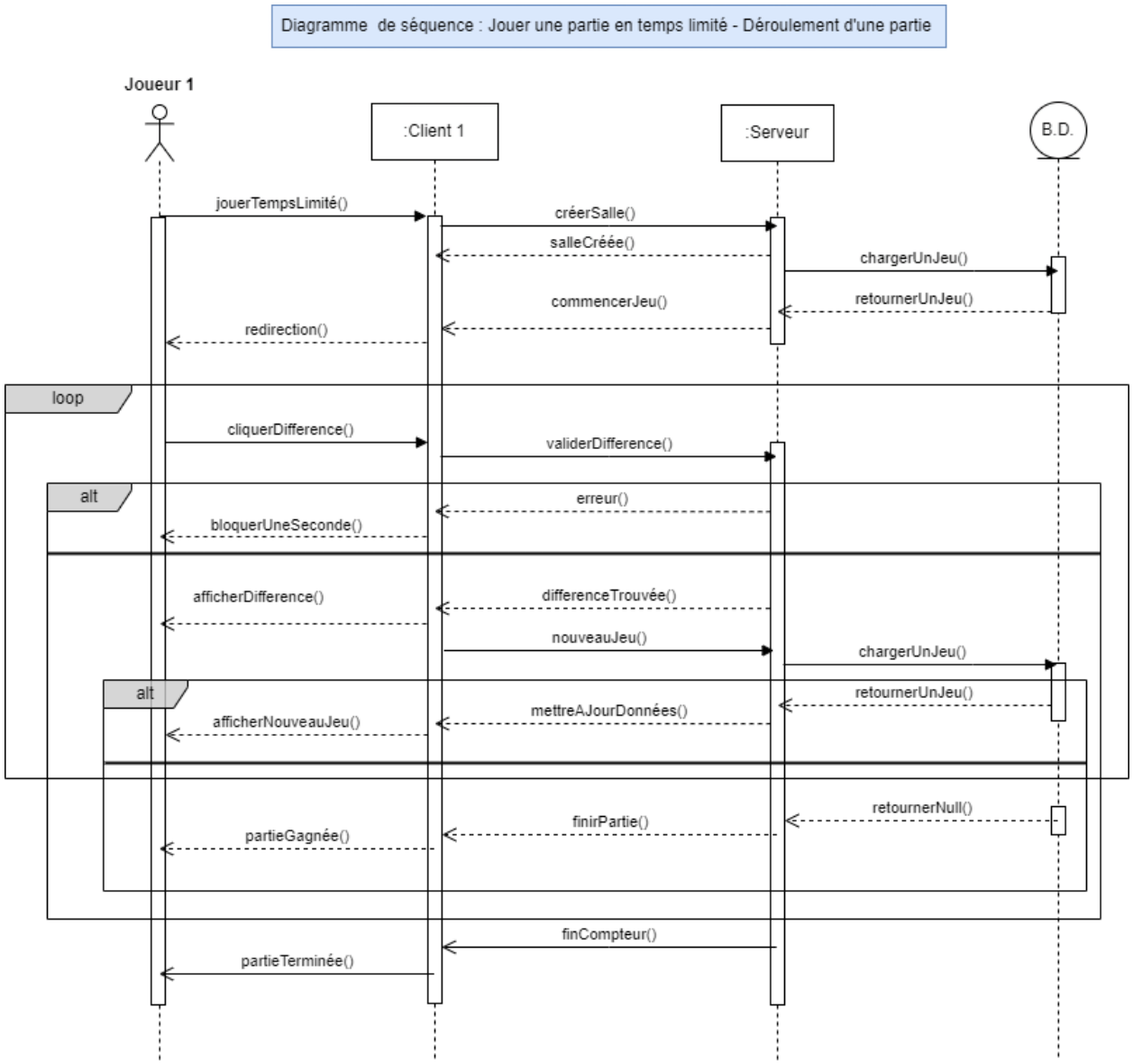


Figure 12.1 : Diagrammes de séquences : Jouer une partie en temps limité - Déroulement d'une partie (CU 8.0)



Note sur le diagramme de séquence : Jouer une partie en temps limité spécificités du mode Coopératif :

- Ce mode décrit uniquement les spécificités liées au mode coopératif

Figure 12.2 : Diagrammes de séquences : Jouer une partie en temps limité - Spécificités du Mode Coopératif (CU 8.0)

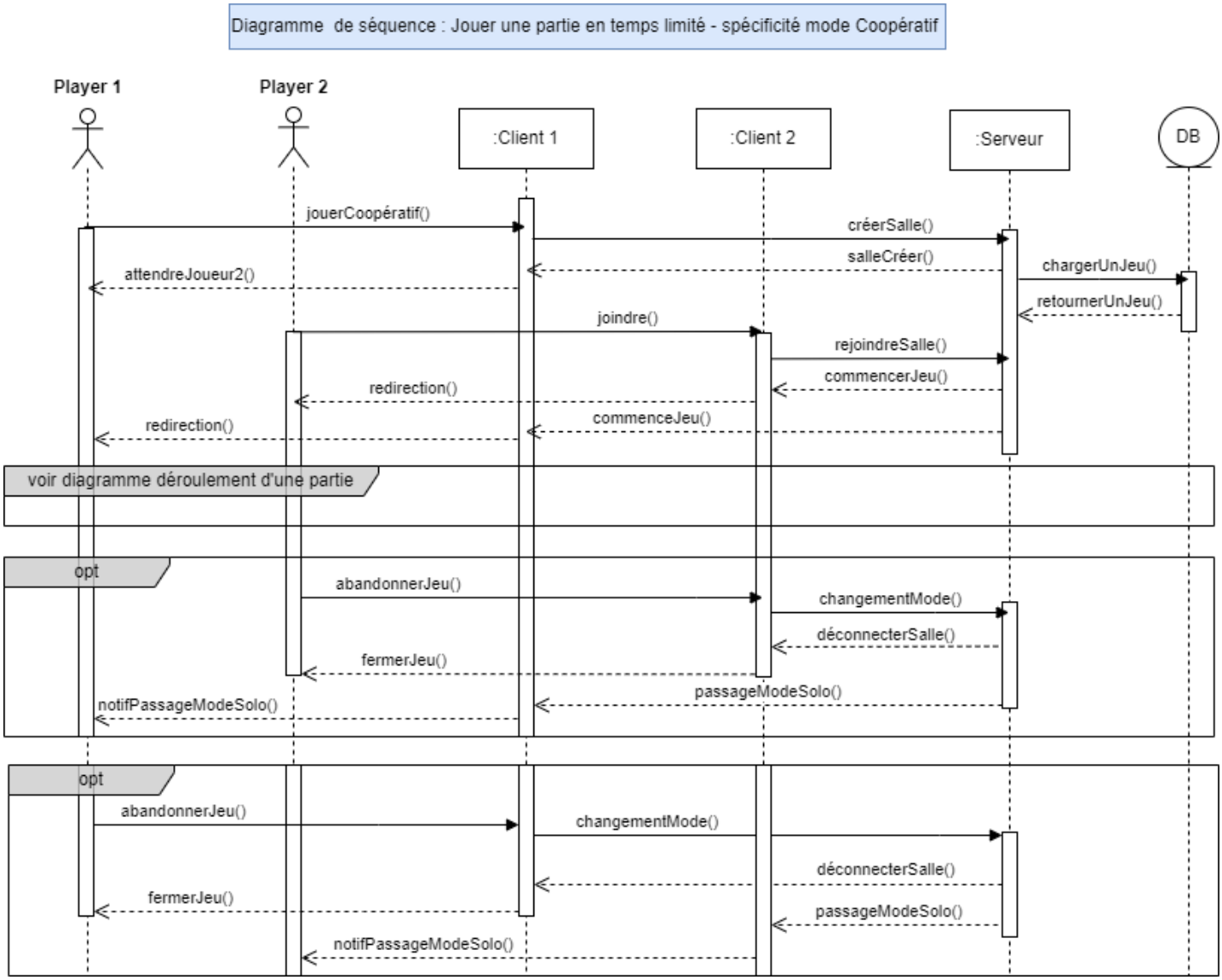
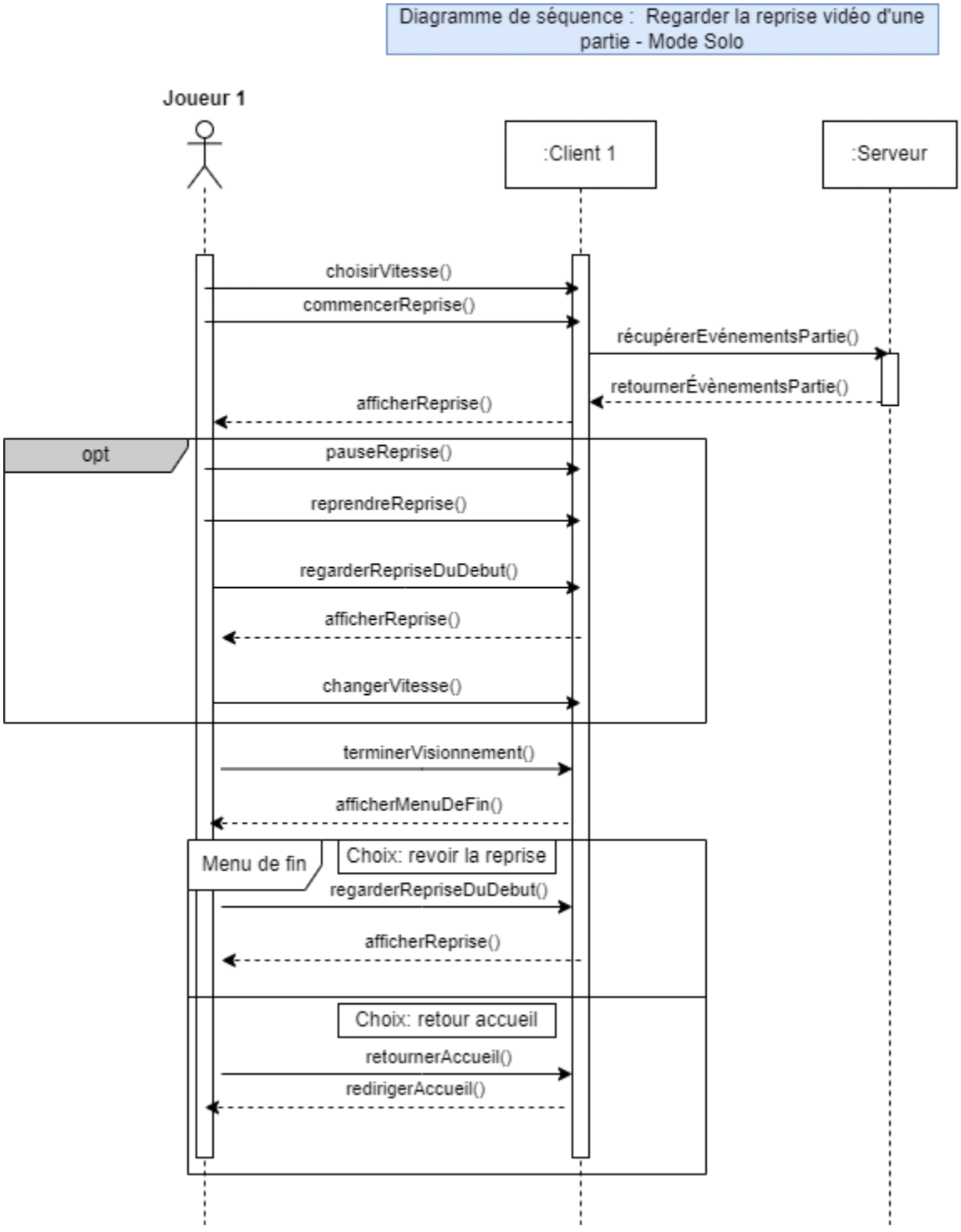


Figure 13 : Diagramme de séquence : Regarder la reprise vidéo d'une partie (CU 7.0)



4. Vue logique

Cette section permet de comprendre la manière dont les différents éléments du système interagissent et collaborent pour effectuer les différentes fonctionnalités demandées.

Notre vue logique est représentée sous forme de diagrammes des classes et de paquetages.

- Les diagrammes de classe modélisent les classes et leurs relations dans notre système, en montrant comment les classes sont organisées, quelles sont leurs propriétés et quelles sont leurs relations (par exemple, l'héritage, l'agrégation, l'association, etc.).
- Les diagrammes de paquetage montrent l'organisation logique et hiérarchique des éléments d'un système en paquets, et comment ces paquets sont organisés et interconnectés. Cela facilite la compréhension de l'architecture de notre système.

<Client>
Ce paquetage représente la vue logique de notre client. On peut voir qu'il est composé de plusieurs vues : la vue de sélection de partie, la vue d'accueil, la vue de jeu un contre un et solo, la vue de jeu en temps limité, la vue de configuration de jeu et enfin la vue de création de jeu.

<Vue de sélection de partie>
Ce paquetage représente la vue de sélection de partie : c'est-à-dire la logique pour créer une partie un contre un ou solo avec des services pour gérer les salles d'attente et le mode classique. Avec aussi des components pour la disposition de la vue.

<Créer et joindre une partie un contre un>
Ce paquetage représente la gestion d'une partie un contre un : c'est-à-dire la logique des salles en multijoueur et la logique de liaison entre le serveur et le client avec la gestion des événements de A à Z d'une partie.

<Vue d'accueil>
Le paquetage contient la vue qui permet d'aller aux diverses sections du site lorsqu'on se connecte ainsi que les informations générales du site.

<Vue de jeu >
Ce paquetage représente la vue de jeu un contre un et solo : c'est-à-dire les logiques de reprise vidéo (avec un service et un component dédiés), du jeu et du canvas (avec ImageService) et du jeu en temps limité (en mode solo ou un contre un) géré par un service dédié (LimitedModeService).

<Logique des canvas>
Ce paquetage représente la logique de tout ce qui concerne les canvas : c'est-à-dire la logique des canvas lors d'une partie jouée en mode solo ou en mode un contre un.

<Logique de reprise vidéo>
Ce paquetage représente la logique de reprise vidéo : c'est-à-dire la gestion de la vitesse de la reprise, de la possibilité de mettre en pause la reprise, de recommencer la reprise et le fait de ne pas pouvoir interagir avec la partie (comme si on était un joueur).

<Logique du jeu>
Ce paquetage représente la logique du jeu : c'est-à-dire la gestion des sons (lorsqu'on trouve une différence et lorsqu'on se trompe), la gestion de ce que l'on affiche à l'utilisateur lorsqu'il joue, la logique du mode de jeu : "temps limité" et enfin la gestion complète des événements d'une partie, incluant la logique de communication entre le serveur et le client.

<Logique des indices >
Ce paquetage représente la logique des indices : c'est-à-dire que tout est géré par un service nommé HintService qui sera chargé de déterminer les trois indices.

<Vue de configuration de jeu>
Ce paquetage représente la vue de configuration de jeu : c-à-d les logiques pour sélectionner un jeu ou un mode de jeu, pour créer un jeu et pour réinitialiser les constantes d'un jeu et supprimer les informations des jeux (historiques, jeux ou meilleurs temps).

<Vue de création de jeu>
Ce paquetage représente la vue de création de jeu : c'est-à-dire les logiques de gestion d'images, de gestion des dessins, de gestion des différences, de gestion des images et de la gestion des canvas.

<Gestion des canvas>
Ce paquetage représente la gestion de tout ce qui concerne les canvas : c'est-à-dire les actions que l'on peut appliquer aux canvas avec les boutons qui permettent d'autres actions (ou les mêmes comme CTRL + Z par exemple).

<Gestion des images>
Ce paquetage représente la gestion de tout ce qui concerne les images : c'est-à-dire la validation de l'image qu'on téléverse (si elle respecte les contraintes sur la taille, le format et le type) et toute la logique qui permet de mettre une image dans un canvas, transformer l'image en tableau de pixels ou encore dessiner les différences.

<Gestion des dessins>
Ce paquetage représente la gestion de tout ce qui concerne les dessins : c'est-à-dire les actions qu'on peut effectuer sur les canvas (comme dessiner un trait, dessiner un rectangle, dessiner en maintenant sa souris, changer la couleur ou encore changer la taille de la gomme par exemple) et les actions qu'on peut faire sur un avant-plan (comme en un effacer un, en dupliquer un ou encore le remplacer avec celui d'à côté par exemple).

<Gestion des différences>
Ce paquetage représente la gestion de tout ce qui concerne les différences : c'est-à-dire vérifier si le nombre de différences est suffisant, générer des paquets de différences, élargir les différences ou encore établir le niveau de difficulté du jeu créé.

Figure 14 : Diagramme de paquetages du client

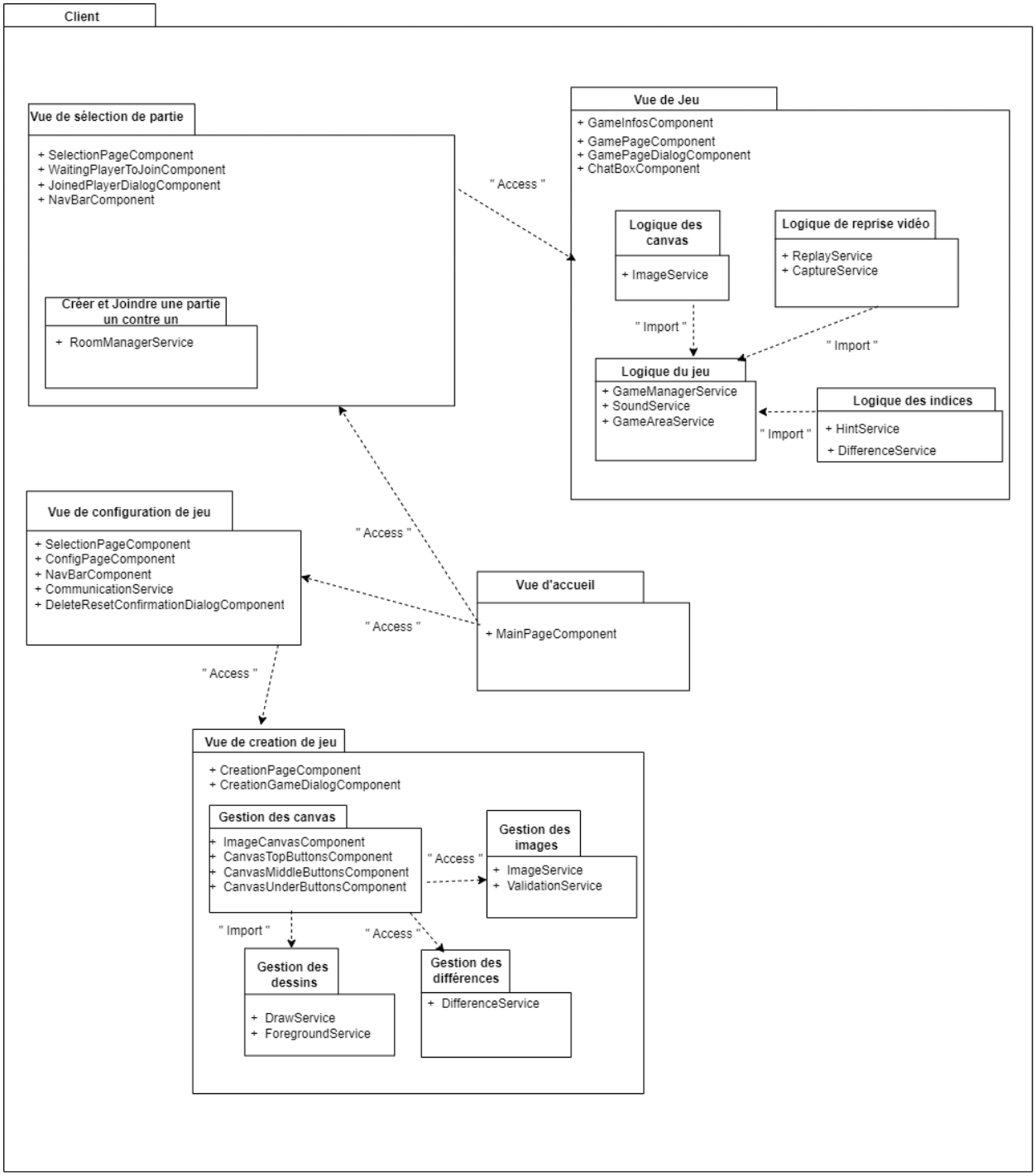


Figure 15 : Diagramme de classes pour la vue de séléction de partie

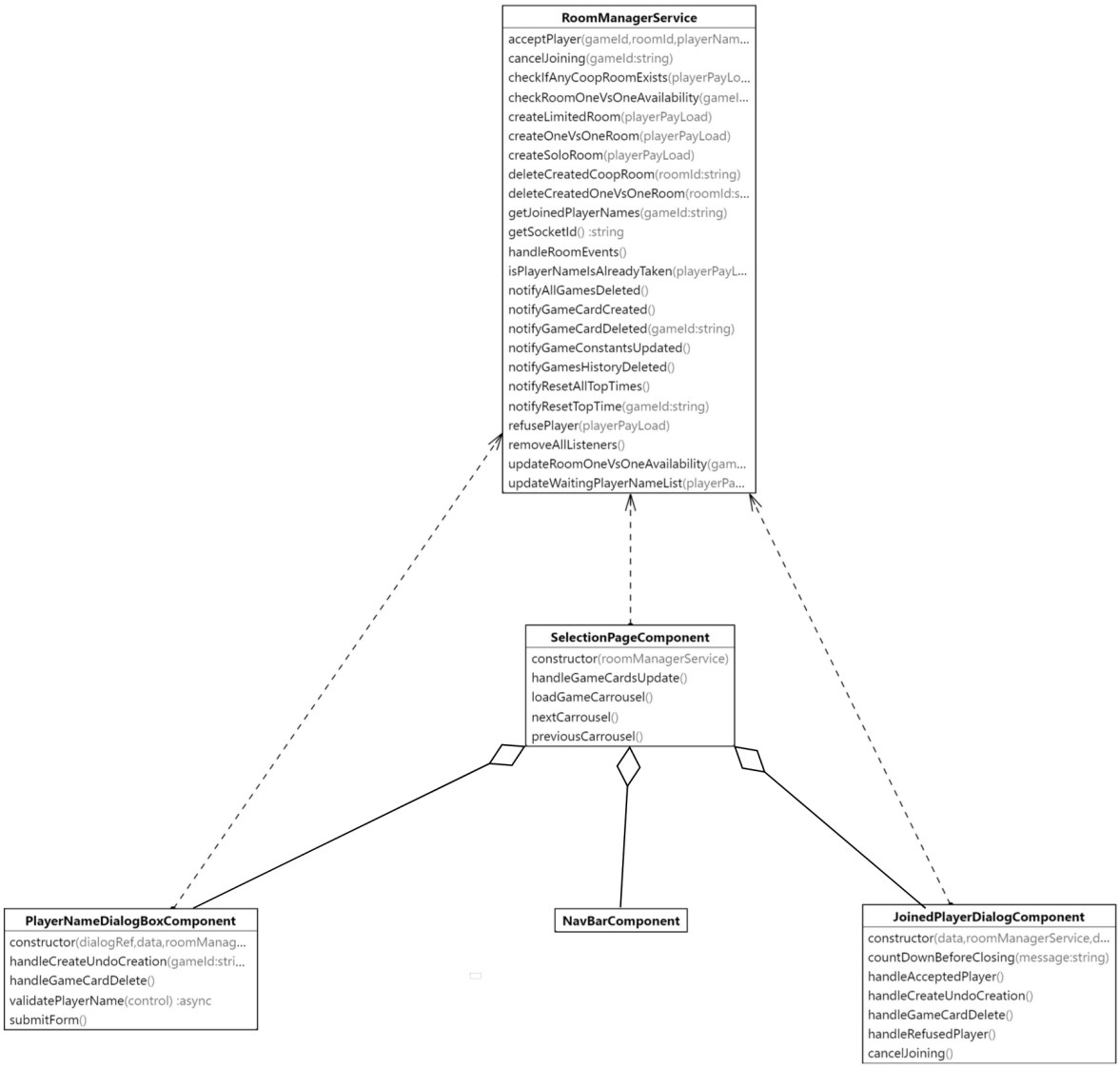


Figure 16 : Diagramme de classes pour la vue du jeu

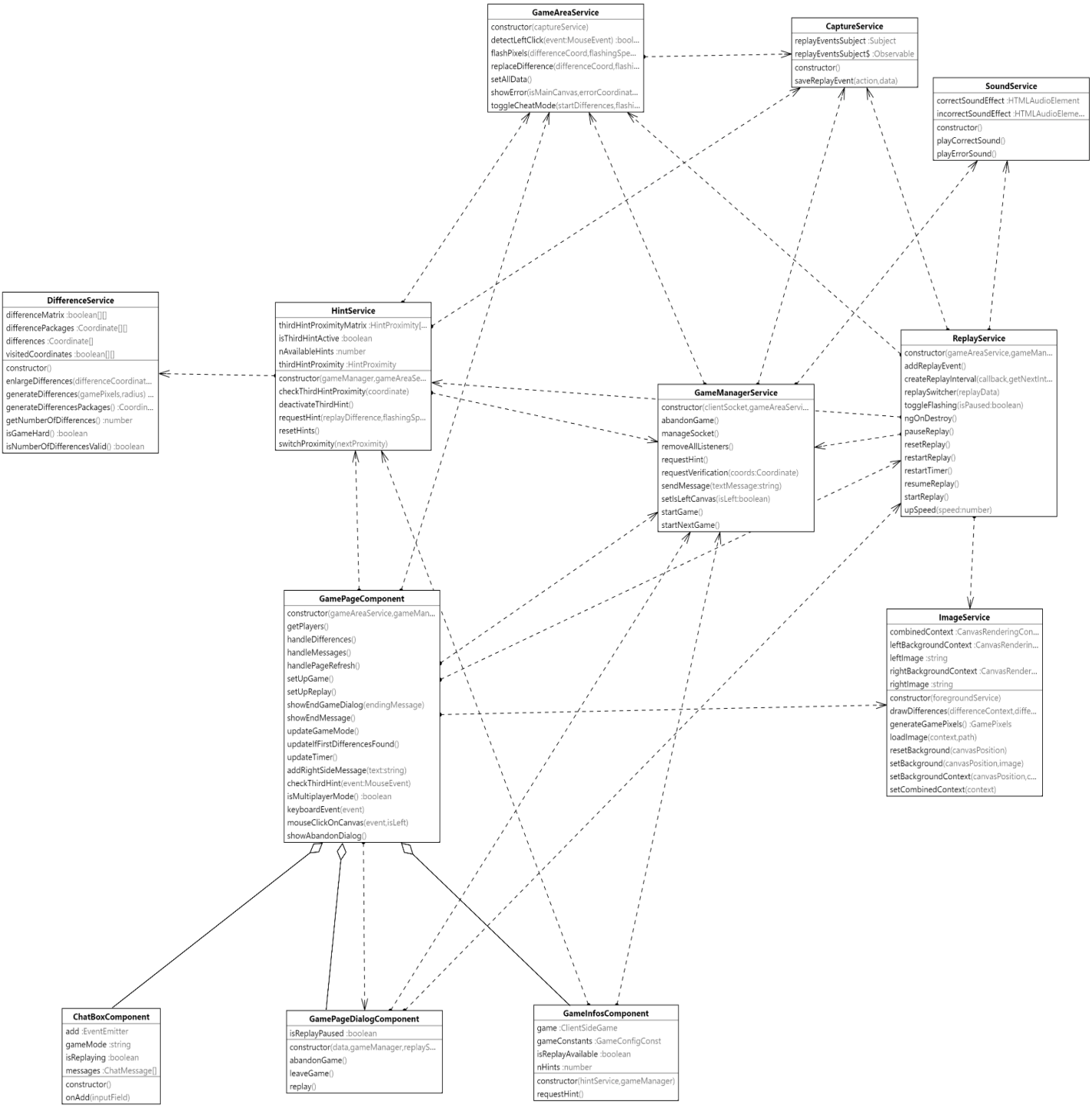


Figure 17 : Diagrammes de classes pour la vue de configuration du jeu

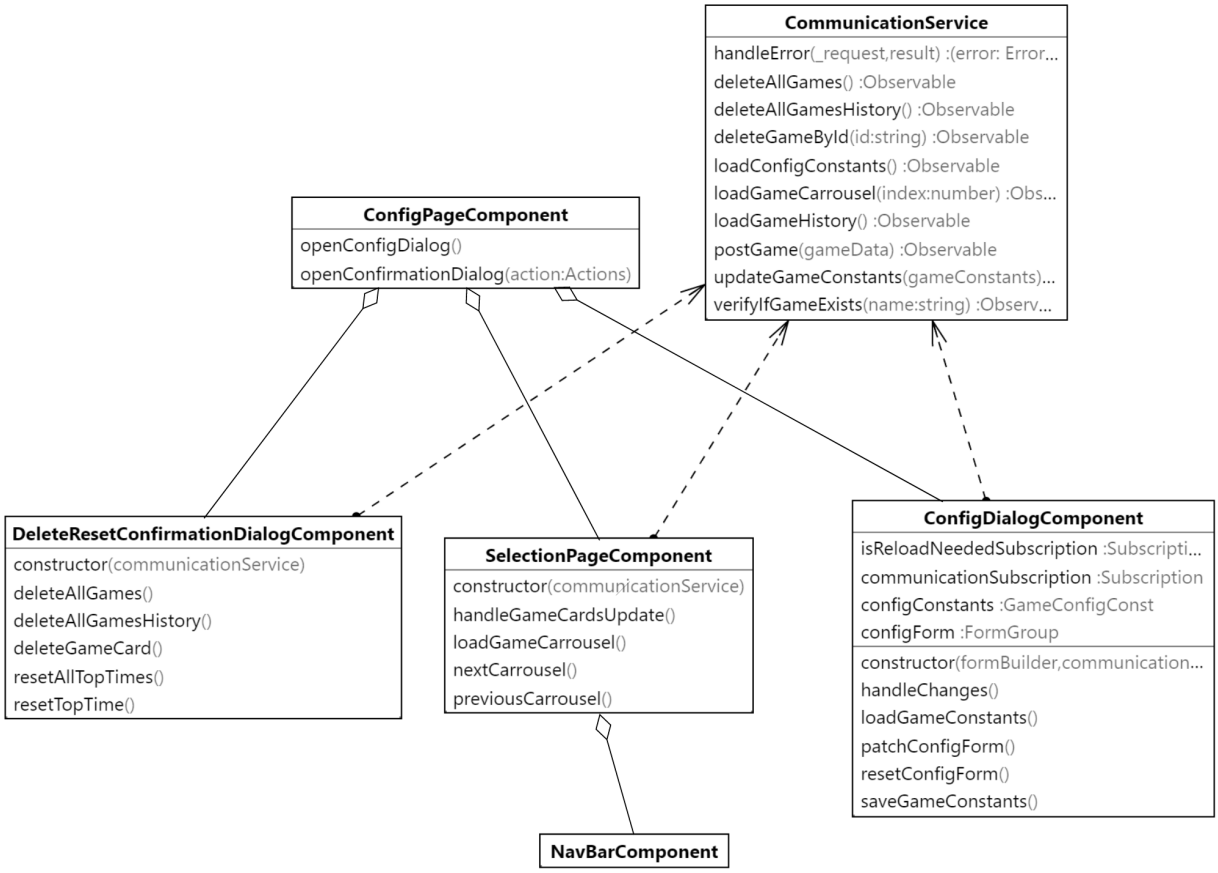
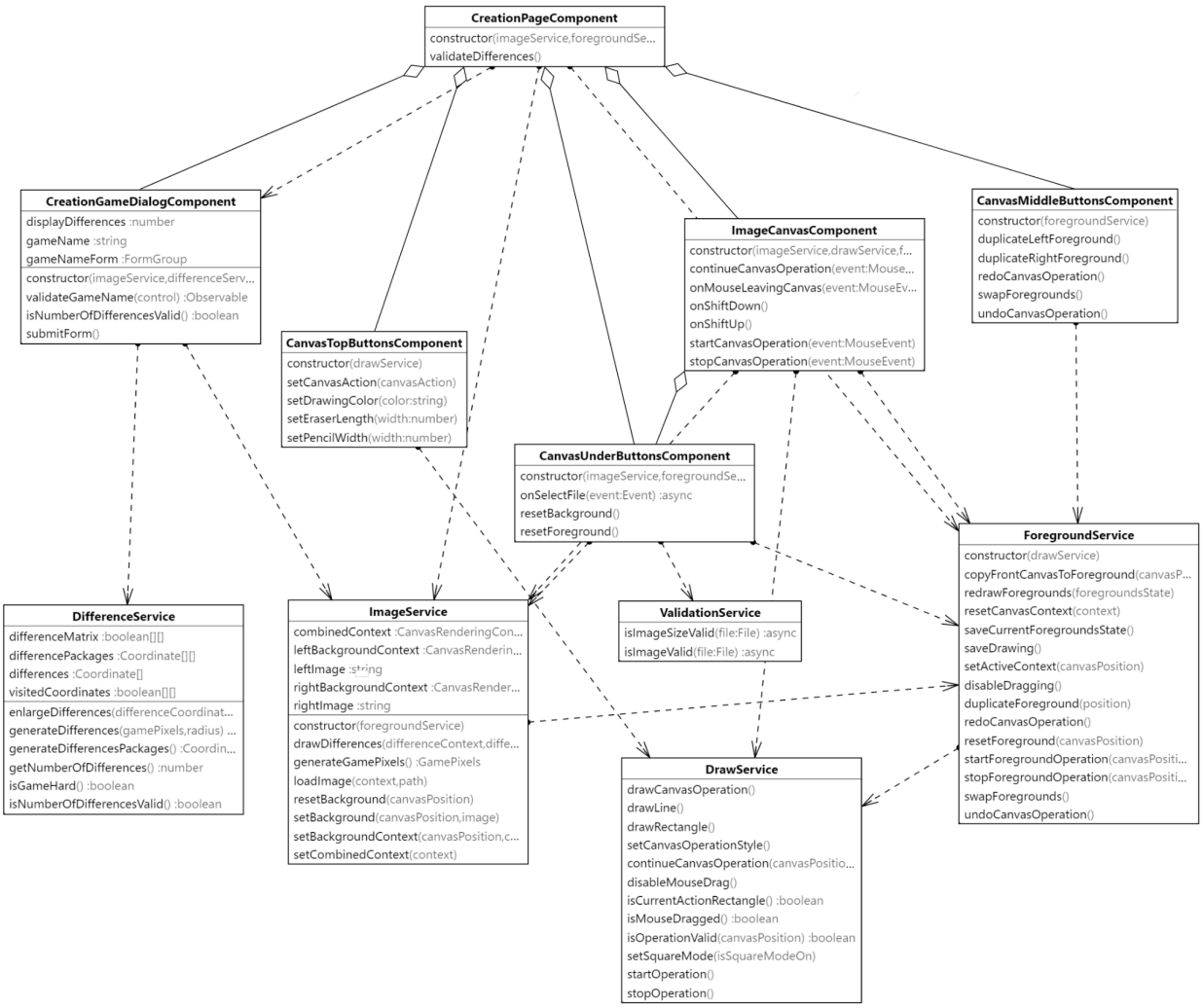


Figure 18 : Diagrammes de classes pour la vue de création de jeu



<Serveur>
Ce paquetage représente la vue logique de notre serveur. Effectivement, ce dernier est décomposé en plusieurs couches pour bien répartir les tâches de chaque service.

<Base de données>
Ce paquetage contient DataBaseService, étant le seul à s'occuper de fournir les fonctionnalités nécessaires pour gérer les opérations de la base de données pour les jeux et cartes de jeux. La majorité des méthodes incluses dans ce service interagissent avec MongoDB via l'utilisation du modèle de données et de Mongoose.

<Logique de jeu classique>
Ce paquetage contient le paquetage de la gestion des listes de jeu ainsi que le ClassicModeService qui implémente la logique pour gérer les parties en solo et un contre un. Elle contient des méthodes pour créer des salles de jeux, pour mettre à jour le chronomètre de partie, pour extraire l'identifiant de la salle en fonction du socket, pour vérifier les coordonnées que le client envoie, etc.

<Logique de jeu temps limité>
Ce paquetage contient le paquetage de la gestion des listes de jeu ainsi que le LimitedTimeModeService qui implémente la logique pour gérer une partie en mode temps limité. En effet, le tout pourrait être implémenté dans un service générique qu'on appellerait GameModeService. Cependant, cela alourdira et placera beaucoup trop de responsabilité pour ce service.

<Logique des messages>
Ce paquetage contient le paquetage de la gestion des messages. Nous avons un MessageManagerService qui s'occupe de gérer la génération des chaînes de caractères pour les messages locaux et messages de fin de partie.

<Logique de l'historique des parties>
Ce paquetage contient le paquetage de la logique de l'historique de partie et qui contient le service HistoryService qui prend en note les informations nécessaires après une fin de partie et qui envoie ces données au client pour qu'il l'affiche dans la vue de configuration.

<Gestion des listes des joueurs>
Ce paquetage contient le paquetage de la gestion des listes de joueurs et contient PlayerListsManagerService qui maintient une liste de tous les joueurs qui attendent de rejoindre une partie en attente. Il fournit des méthodes pour mettre à jour cette liste, pour vérifier si un nom de joueur est disponible, pour refuser un joueur et pour obtenir un joueur accepté. De plus, il fournit des méthodes pour annuler l'inscription des joueurs et pour mettre à jour les temps record des joueurs en fonction des nouvelles performances.

<Logique commune>
Ce paquetage contient la gestion des salles de jeu. Le service RoomsManagerService a des fonctions pour créer une salle de jeu, pour obtenir une salle de jeu par id, pour obtenir l'id d'une salle de jeu à partir d'un socket, pour obtenir une salle de jeu par identifiant de joueur, etc.

<Gestion des listes de jeux>
Ce paquetage contient le paquetage de la gestion des listes de jeu et contient GameListsManagerService qui gère une liste de jeux de manière paginée afin de les afficher sous forme de carrousel. Elle contient une méthode pour construire un objet GameCard, pour ajouter un jeu au carrousel, pour créer un nouveau carrousel à partir d'un tableau de jeu et pour retourner le tableau de jeu actuellement emmagasiné dans le carrousel.

<BD Contrôleur>
Ce paquetage contient le service GameService qui est une classe qui fournit des méthodes pour interagir avec la base de données de jeux. Il est injectable et dépend de DatabaseService pour les opérations de lecture et d'écriture de données.

<HTTP API>
GameController Cette classe est un contrôleur qui gère les requêtes HTTP liées à la manipulation des jeux. De plus, ses méthodes contiennent des décorateurs @Get @Post et @Delete qui indiquent à NestJs la façon de montrer ces méthodes en tant que points de terminaison HTTP pour les requêtes GET, POST et DELETE.

<Gateway API>
GameGateway C'est essentiellement les points d'extrémité pour la communication entre les clients et le serveur. En effet, elle contient plusieurs méthodes décorées par @SubscribeMessage() pour gérer différents types d'événements de jeu (ex: GameEvents.CreateSoloGame, GameEvents.StartGameByRoomId, etc.)

Figure 19 : Diagramme de paquetages du serveur

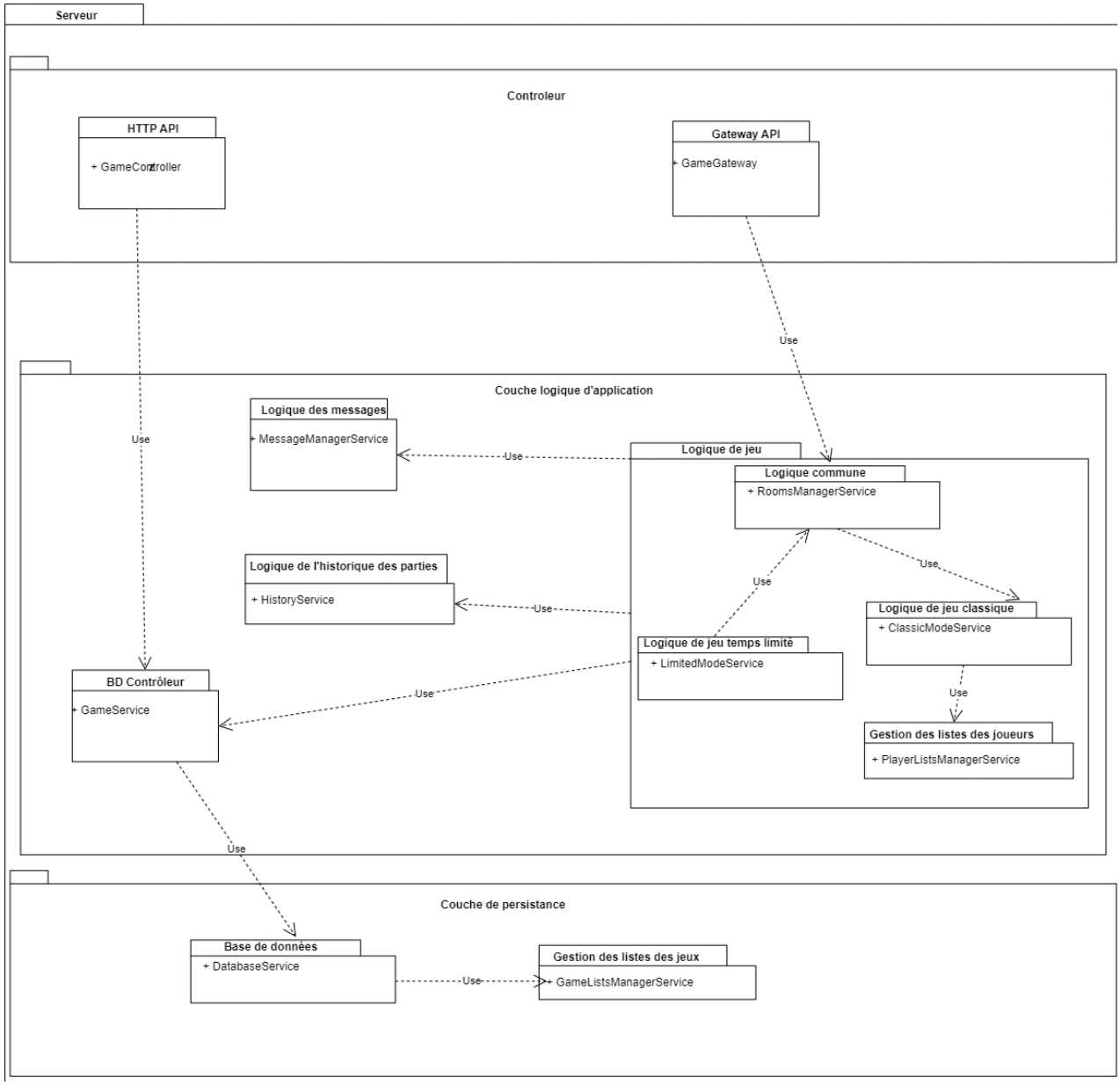


Figure 20 : Diagramme de classes de la gestion de jeu

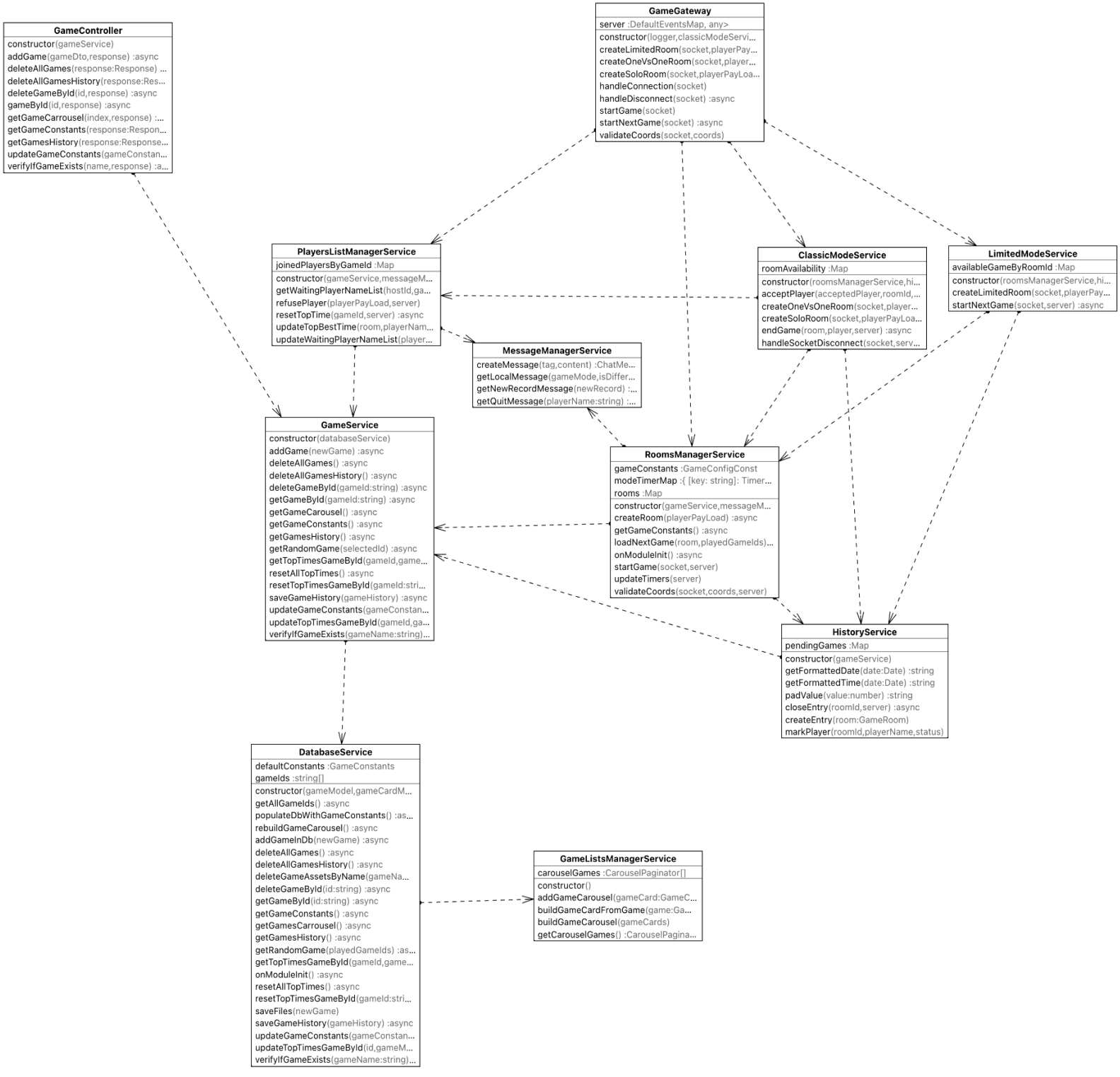
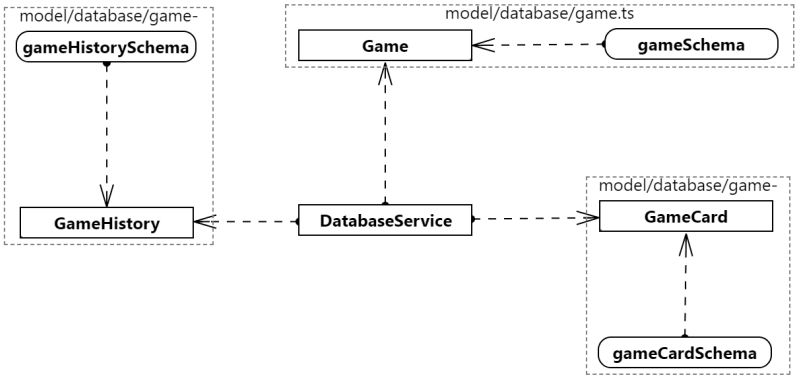


Figure 21 : Diagramme de classes d'objet de transfert de données (DTO) et définition de schéma



5. Vue de déploiement

Cette section permet de décrire le déploiement physique des informations générées par le système sur des composants matériels. Les boîtes en trois dimensions, appelées nœuds, représentent les composants du système, qu'ils soient logiciels ou matériels. Les lignes entre les nœuds indiquent les relations et les petites formes à l'intérieur des boîtes représentent les artefacts logiciels qui sont déployés. Notre diagramme permet de :

- Montrer quels éléments logiciels sont déployés par quels éléments matériels.
- Illustrer le traitement d'exécution du point de vue matériel.
- Visualiser la topologie du système matériel.

Figure 22 : Diagramme de Déploiement