101

Document d'architecture logicielle

Version 1.3

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	1.1	Suite des diagrammes de séquences	Sulayman Hosna, Mathieu Prévost, Zakaria Zair
2023-03-21	1.2	Fin des diagrammes de séquences	Mathieu Prévost, Zakaria Zair
2023-03-21	1.3	Révision des cas d'utilisation	Edgar Kappauf et Zakaria Zair
2023-03-21	1.3	Rédaction de la vue logique	Edgar Kappauf et Jeremy Ear
2023-03-21	1.3	Réalisation du diagramme de déploiement	Edgar Kappauf et Mathieu Prévost
2023-03-21	1.3	Révision complète du document	Zakaria Zair

Table des matières

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

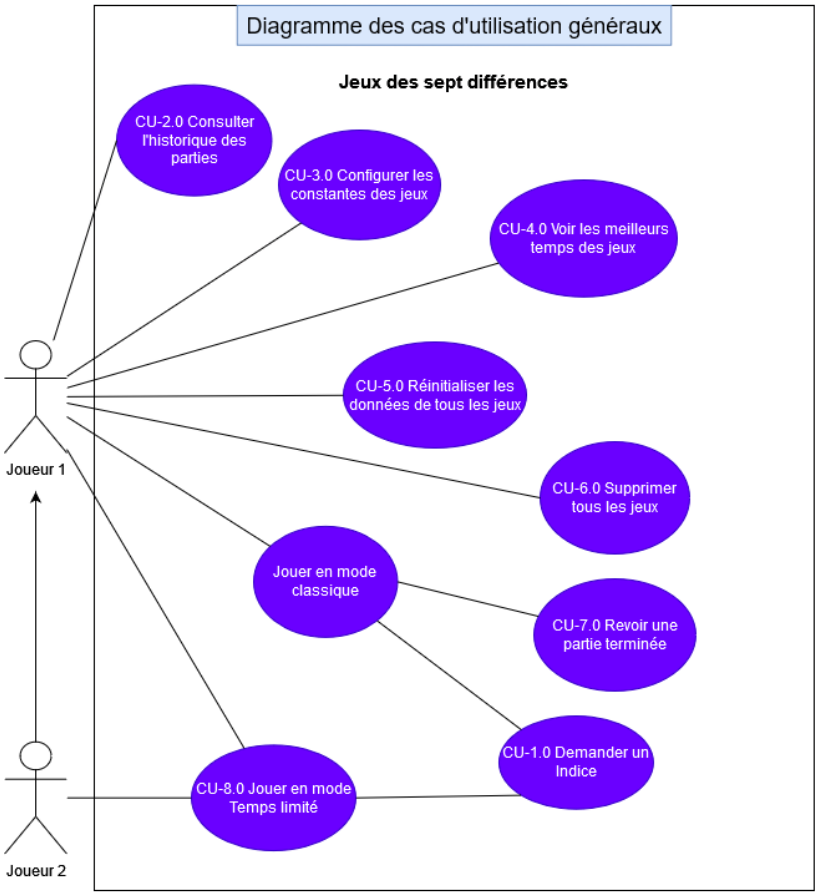


Figure 2 : Diagramme de cas d'utilisation CU-1.0 et 2.0 détaillés

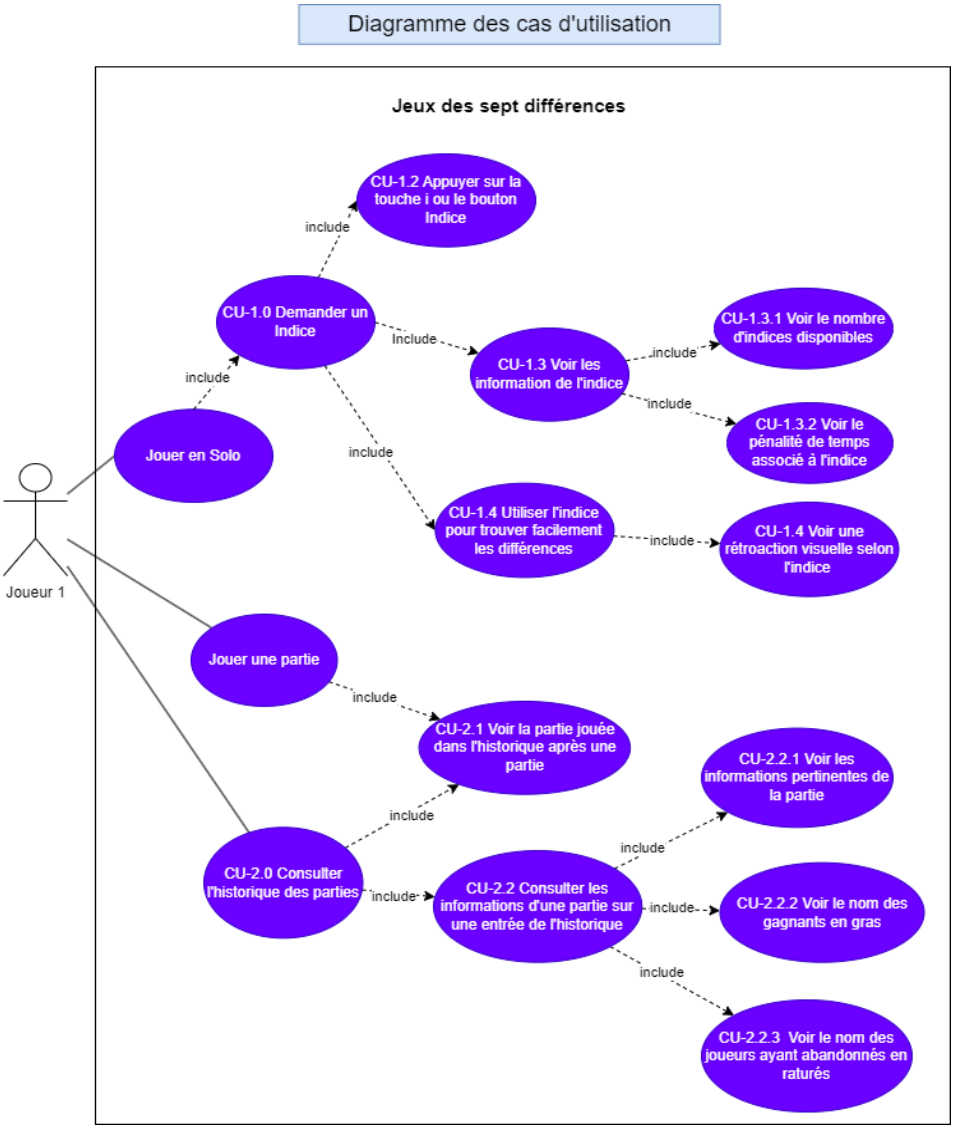


Figure 3 : Diagramme de cas d'utilisation CU-3.0, 4.0, 5.0 et 6.0 détaillés

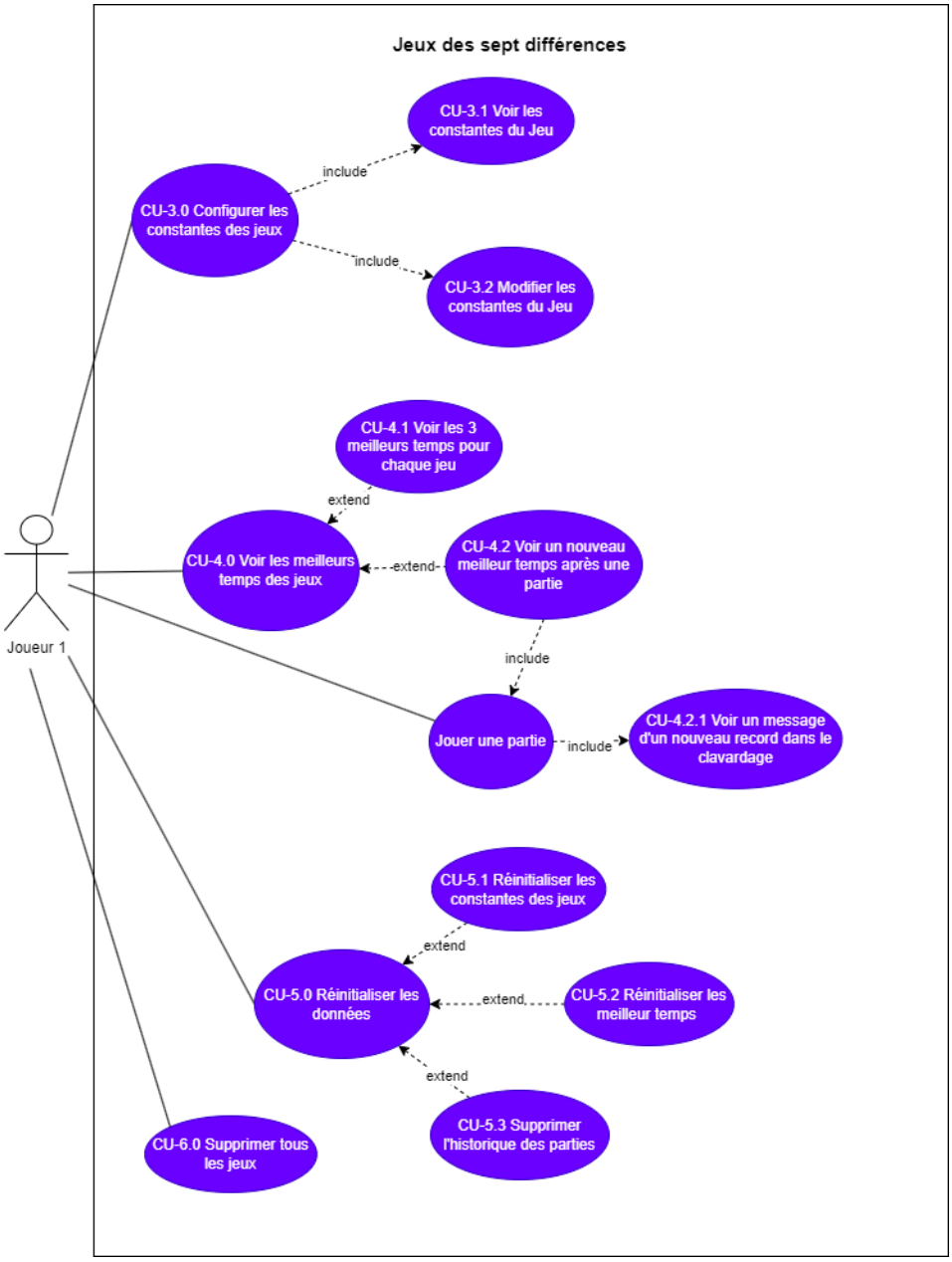


Figure 4 : Diagramme de cas d'utilisation CU-7.0 détaillés

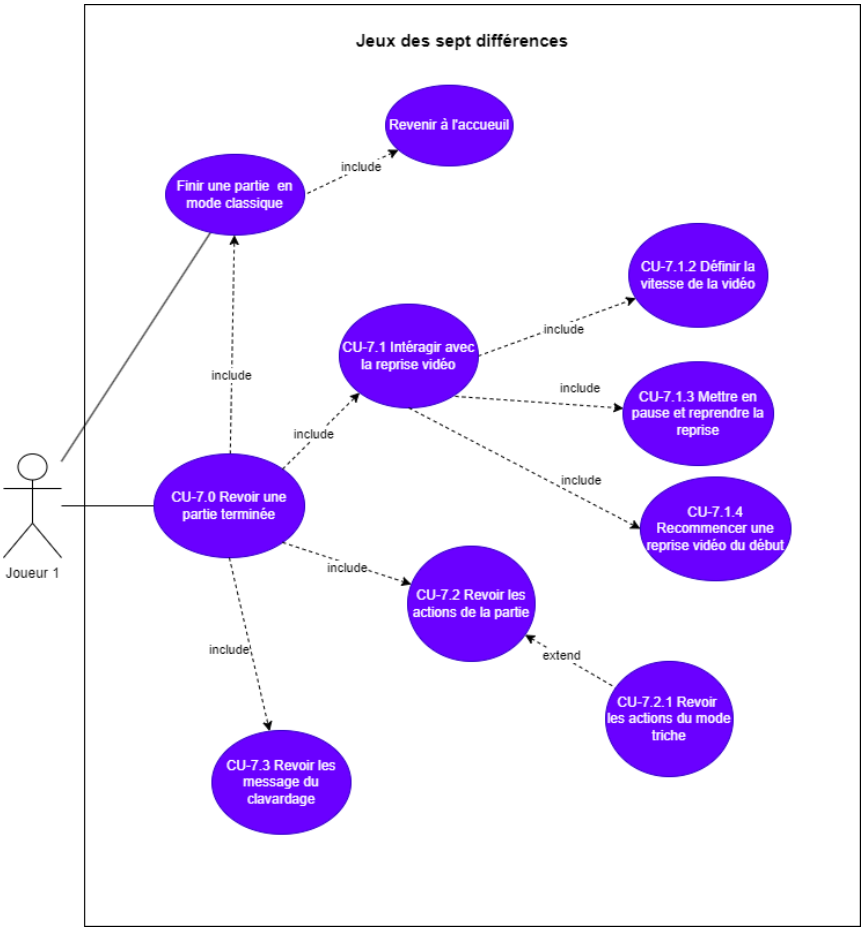
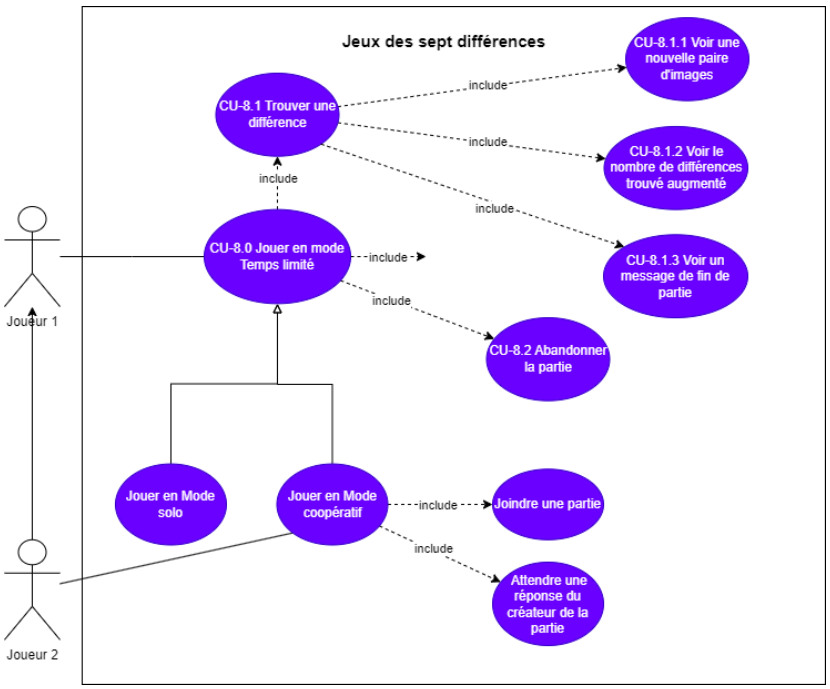


Figure 5 : Diagramme de cas d'utilisation CU-8.0 détaillés



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)
- 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 .

Diagramme de séquence : Demander un indice (CU 1.0)

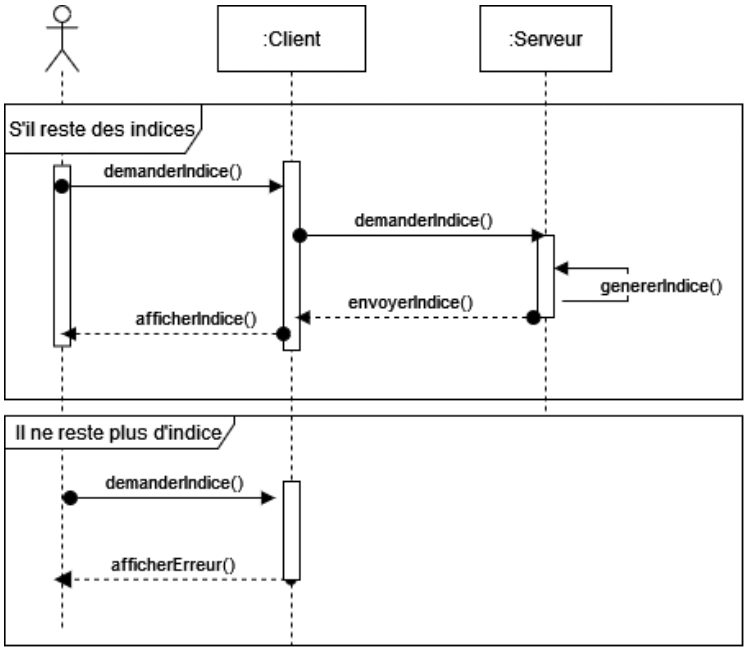
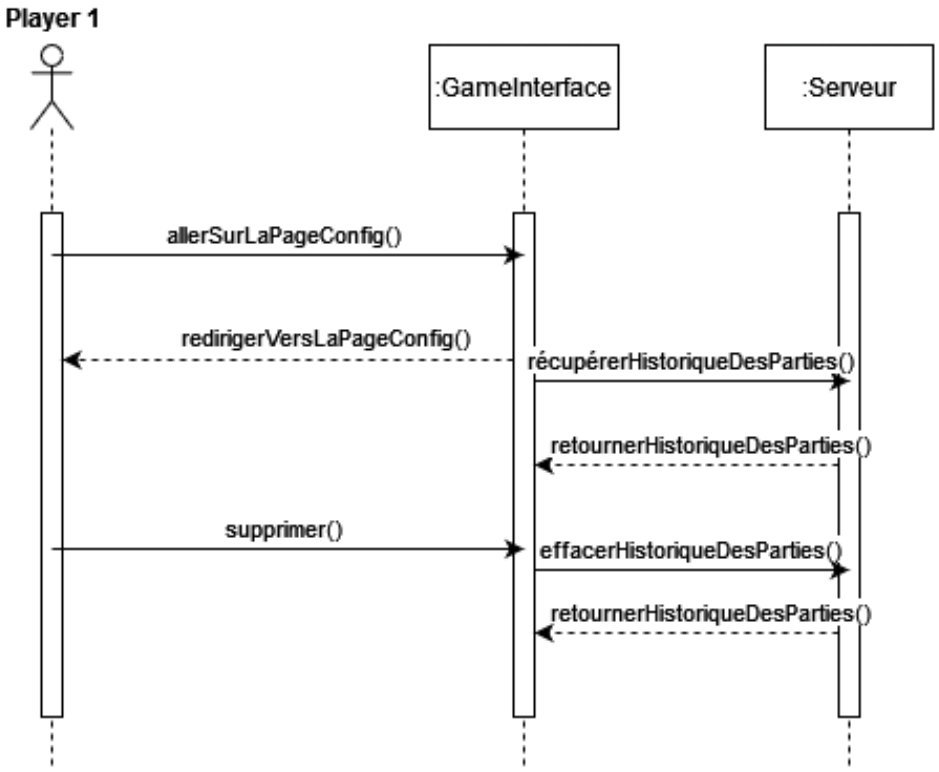


Diagramme de séquence : Voir l'historique des parties (CU 2.0)



Diagrammes de séquences : Jouer une partie en temps limité (CU 8.0)

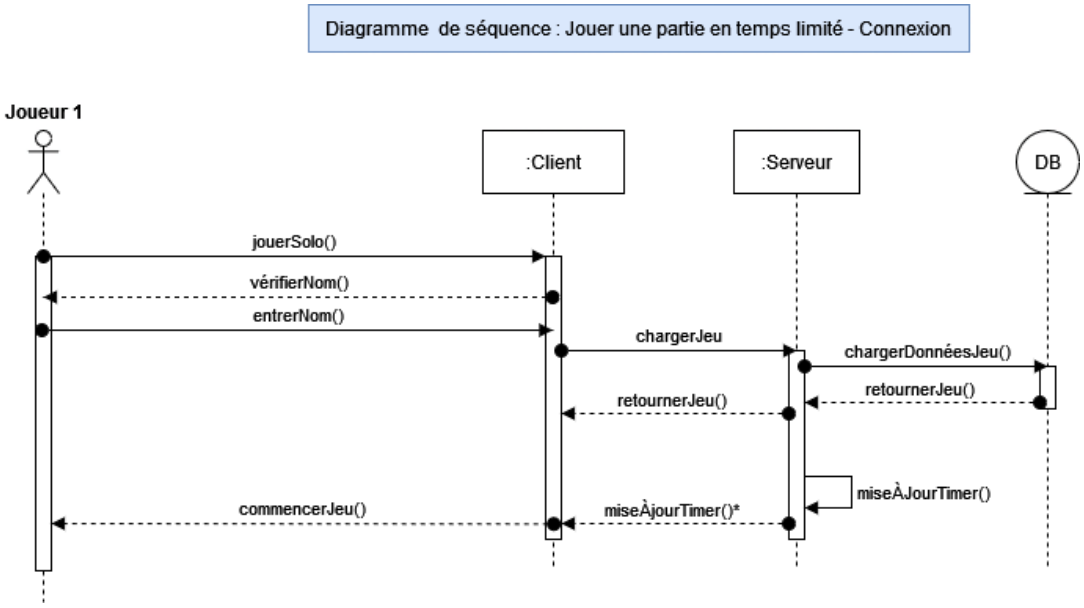


Diagramme de séquence : Jouer une partie en temps limité

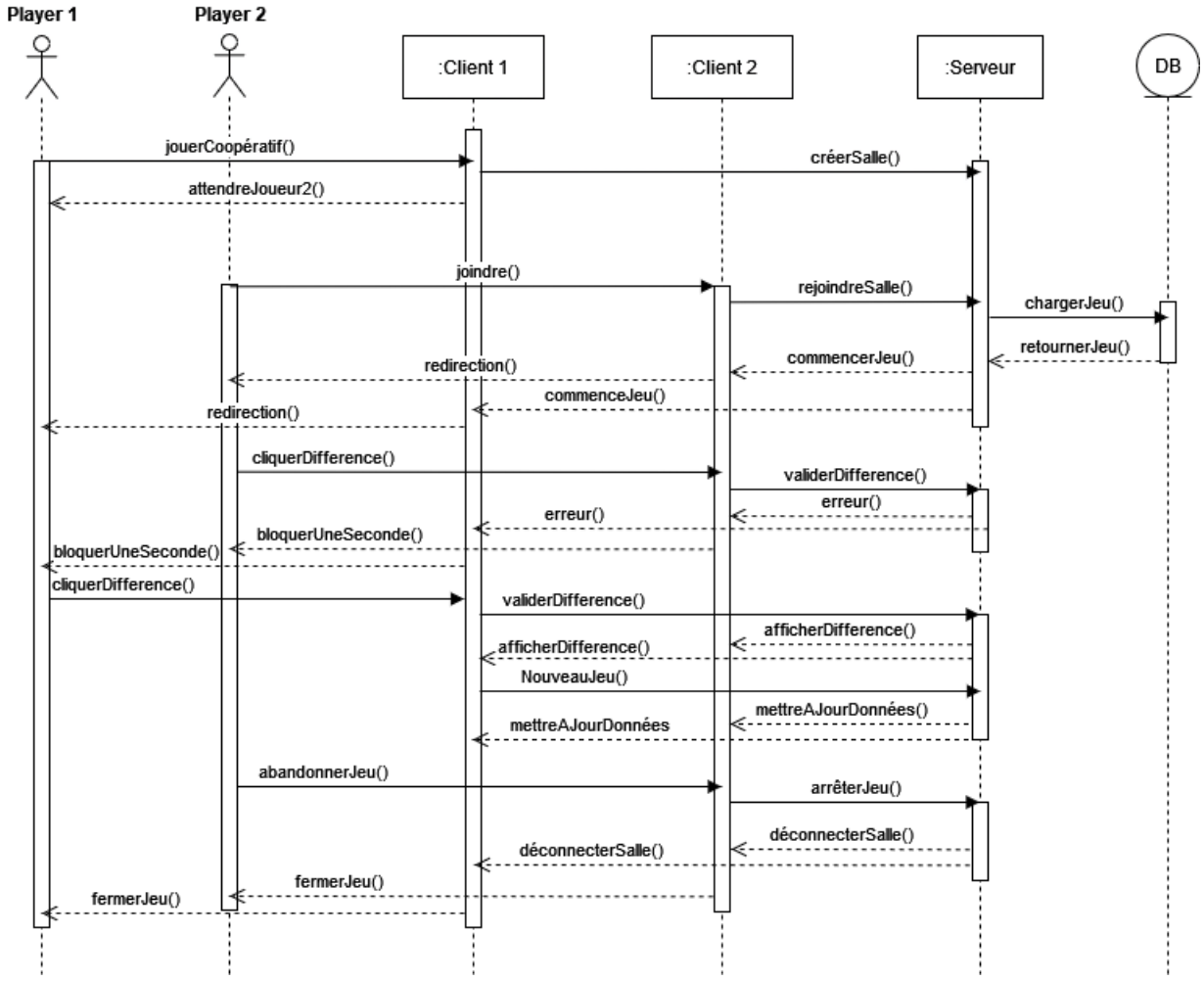
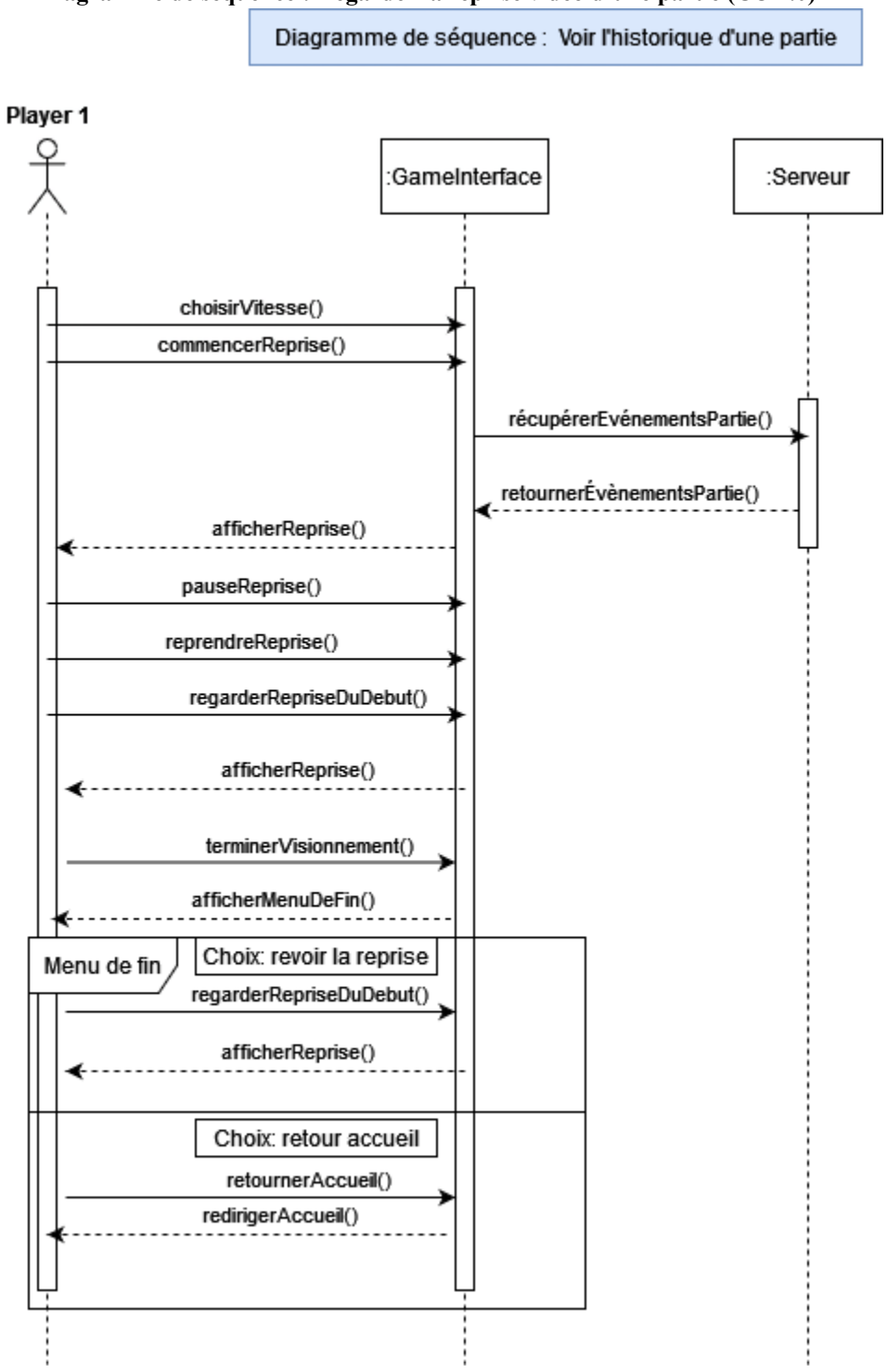


Diagramme de séquence : Regarder la reprise vidéo d'une partie (CU 7.0)



4. Vue logique

<Client>
Ce diagramme de 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.

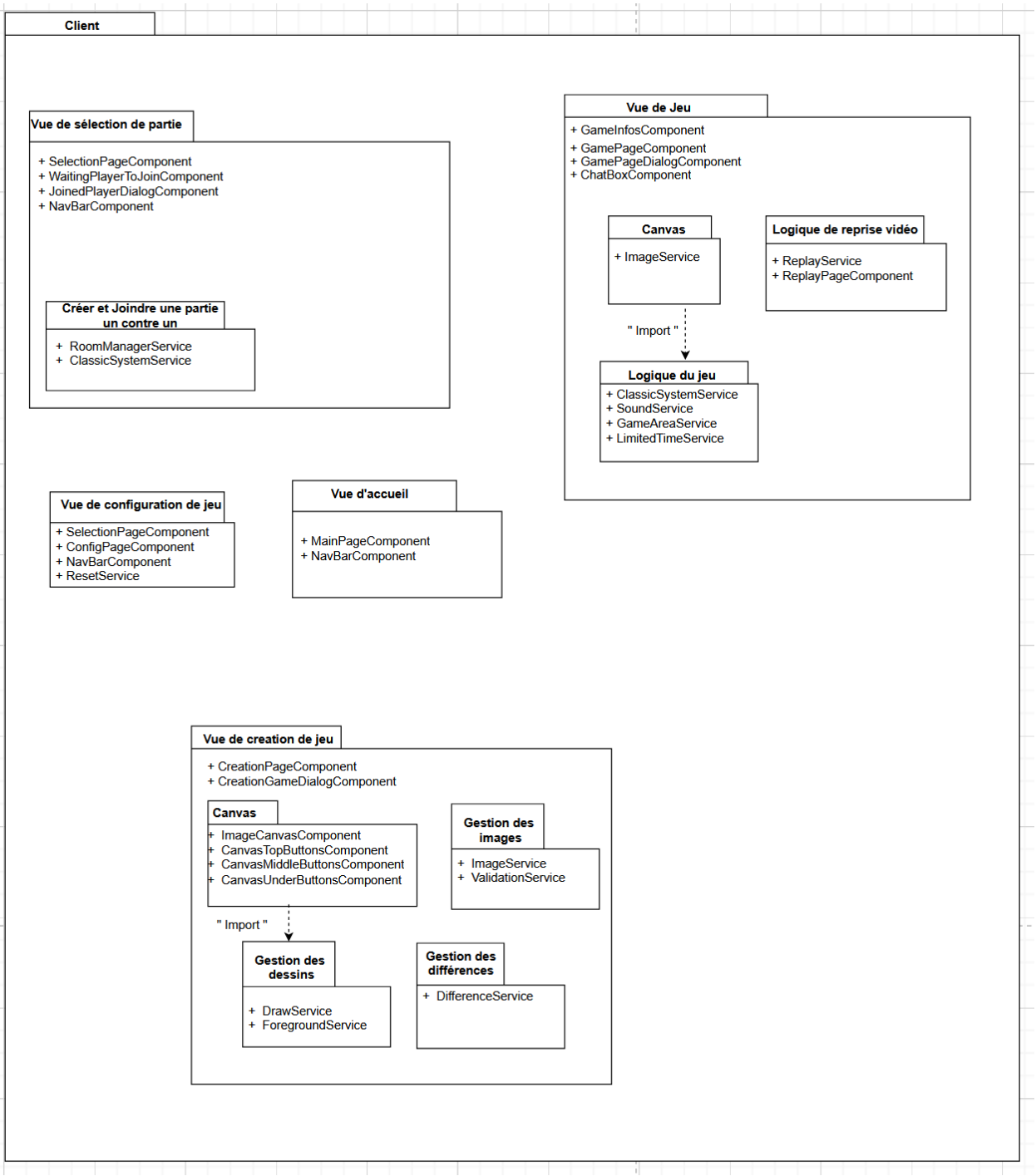
<Vue de sélection de partie>
Ce diagramme de paquetage représente la vue de sélection de partie : c'est-à-dire la logique pour créer une partie un contre un ou solo avec des services pour gérer les salles d'attente et le mode classique. Avec aussi des components pour la disposition de la vue.

<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 diagramme de 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 (LimitedTimeService).

<Vue de configuration de jeu>
Ce diagramme de 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 diagramme de 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.



<Serveur>
Ce diagramme de paquetage représente la vue logique de notre serveur. Effectivement, ce dernier est décomposé en plusieurs couches pour bien répartir les tâches de chaque service. Nous avions la gestion des données et la gestion de jeu qui englobe plusieurs couches incluant la logique de jeu en mode classique, la logique de jeu en mode temps limité, la gestion des listes de jeu, la logique des messages, la logique de l'historique des parties et la logique des indices.

<Gestion des données>

Ce diagramme de 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.

<Gestion de jeu>

Ce diagramme englobe tout ce qui est en lien avec la logique du jeu de différence, c'est-à-dire celle des messages de parties, celle du jeu classique, celle du jeu avec temps limité, celle des indices et celle de l'historique des parties. De plus, dans ce paquetage nous avions un contrôleur et un gateway.

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.)

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.

<Logique de jeu classique>

Ce diagramme 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.

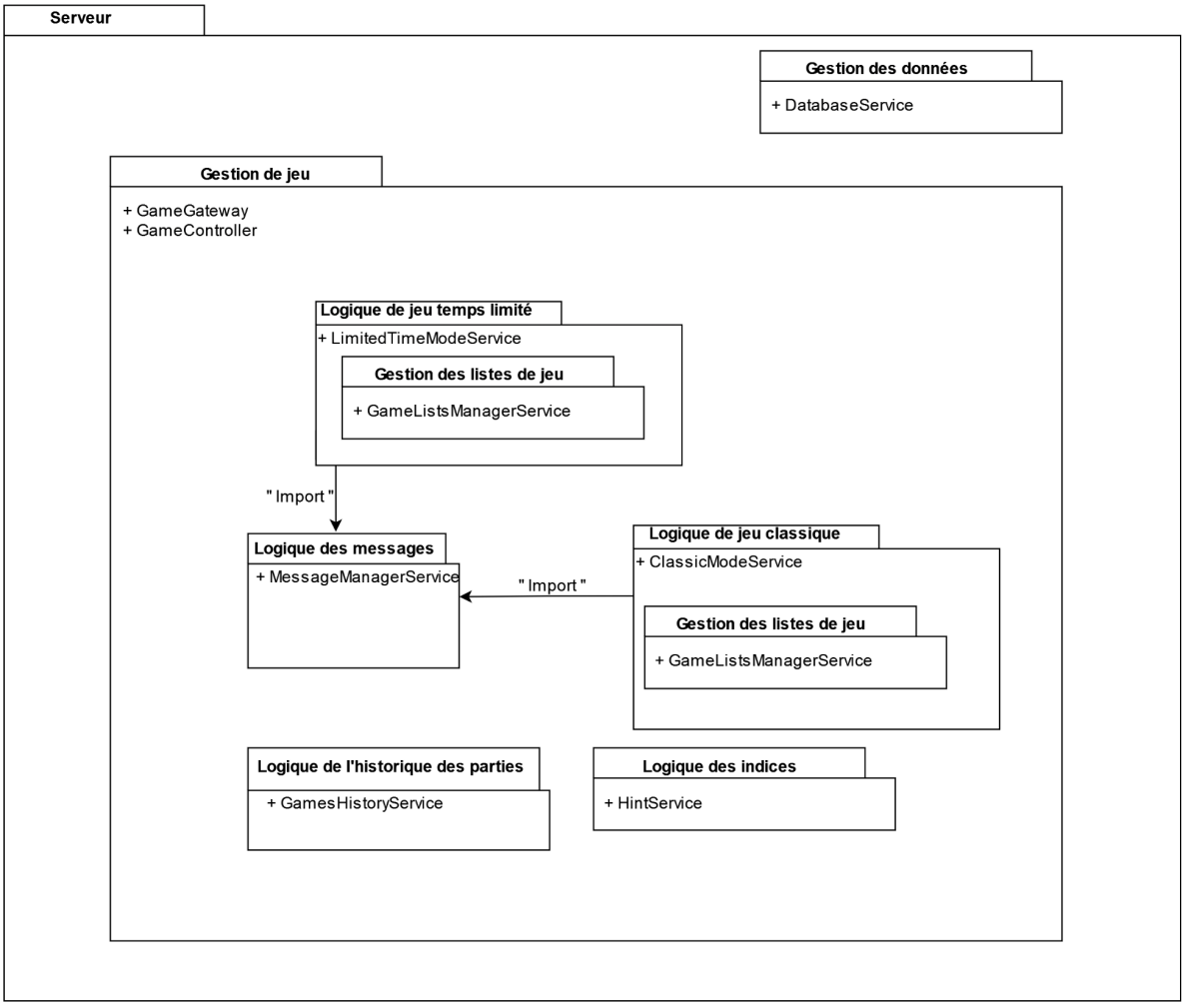
<Logique de jeu temps limité>
Ce diagramme contient le paquetage de la gestion des listes de jeu ainsi que le LimitedTimeModeService qui implémente la logique pour gérer une partie en mode temps limité. En effet, le tout pourrait être implémenté dans un service générique qu'on appellerait GameModeService.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

<Logique des messages>
Ce diagramme 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 diagramme contient le paquetage de la logique de l'historique de partie et qui contient le service GameHistoryService 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.

<Logique des indices>
Ce diagramme contient le paquetage de la logique des indices et à l'intérieur s'y trouve un service HintService sera chargé de déterminer les trois indices et de les envoyer au client.

<Gestion des listes de jeu>
Ce diagramme 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.


Serveur
Gestion des données
+ DatabaseService
Gestion de jeu
+ GameGateway
+ GameController
Logique de jeu temps limité
+ LimitedTimeModeService
Gestion des listes de jeu
+ GameListsManagerService
" Import "
Logique des messages
+ MessageManagerService
" Import "
Logique de jeu classique
+ ClassicModeService
Gestion des listes de jeu
+ GameListsManagerService
Logique de l'historique des parties
+ GamesHistoryService
Logique des indices
+ HintService

5. Vue de déploiement

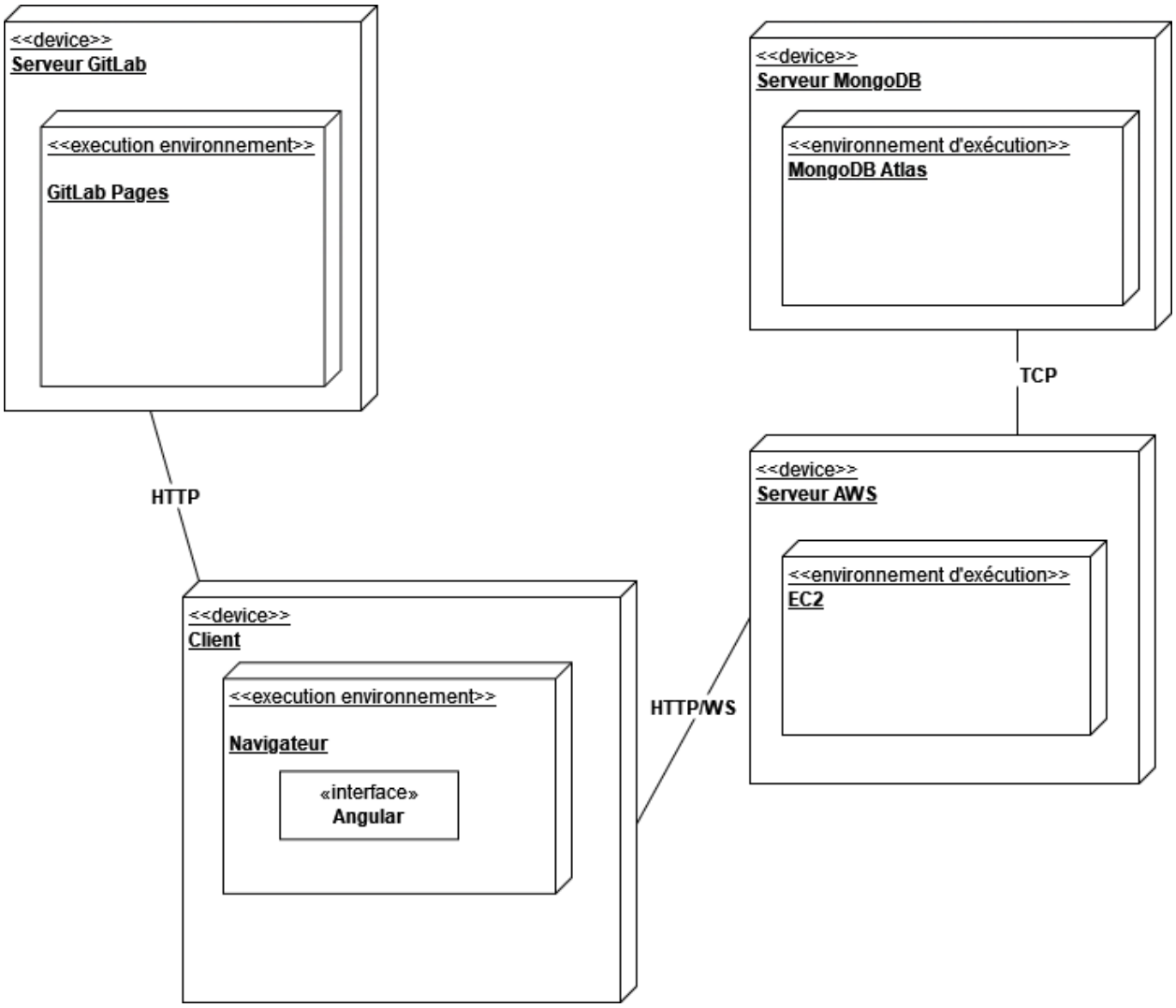


Diagramme de Déploiement