

Document d'architecture logicielle

Version 2.1

Historique des révisions

Date	Version	Description	Auteur
2023-03-13 2023-03-15	1.0	Rédaction de la partie 5 du rapport d'architecture logicielle, sur la vue de déploiement du projet. Création du diagramme de déploiement	Sam Younan
2023-03-18	1.1	Ajout des cas d'utilisation	Elouan Guyon
2023-03-18	1.2	Début de l'introduction	Julien Légaré
2023-03-20	1.3	Ajout du diagramme de classes et de paquetages	Julien Légaré
2023-03-21	1.4	Création des diagrammes de cas d'utilisation à partir des cas d'utilisation rédigés par Elouan	Sam Younan
2023-04-18	2.0	Modification des diagrammes	Sam Younan, Julien Légaré, Jasper Lai,
2023-04-19	2.1	Modification des diagrammes de séquence	Chahine Hamitouche

Table des matières

1. Introduction	4
2. Vue des cas d'utilisation	5
3. Vue des processus	8
4. Vue logique	13
5. Vue de déploiement	17

Document d'architecture logicielle

1. Introduction

1.1. Mise en contexte

Dans le cadre du cours LOG2990 - Projet de logiciel d'application Web, nous avons été chargés de créer une version internet du jeu des sept différences. Pour réussir à mener ce projet à jour, nous avons dû commencer par analyser et comprendre la tâche qui nous était demandée grâce au document de vision qui nous était fourni, ainsi qu'avec l'aide des descriptions de tâches présentes sur GitLab, puis ensuite amorcer le développement en nous familiarisant avec le mode de travail en intégration continue, une méthode avec laquelle peu d'entre nous étaient familiers.

1.2. Contenu et organisation du document

Ce document contient les différentes vues qui sont nécessaires à la compréhension générale de notre projet et de son fonctionnement, ainsi que son architecture.

Nous avons commencé par décrire la vue des cas d'utilisation, qui consiste en les différents cas d'utilisations pertinents pour notre projet. Puis, avec l'aide d'un diagramme de séquence, nous avons décrit la vue de processus, qui décrit le système en termes d'interactions entre les différents processus significatifs. Ensuite, nous avons décrit la vue logique, qui présente les parties architecturalement significatives du modèle de design, en utilisant des diagrammes de classe et de de paquetages. Finalement, nous avons décrit la vue de déploiement, qui décrit la configuration de déploiement concret des différentes parties du système en indiquant les nœuds physiques qui exécutent les logiciels, ainsi que leurs interconnexions.

\

2. Vue des cas d'utilisation

Diagramme 1 : Jouer une partie en mode temps limité

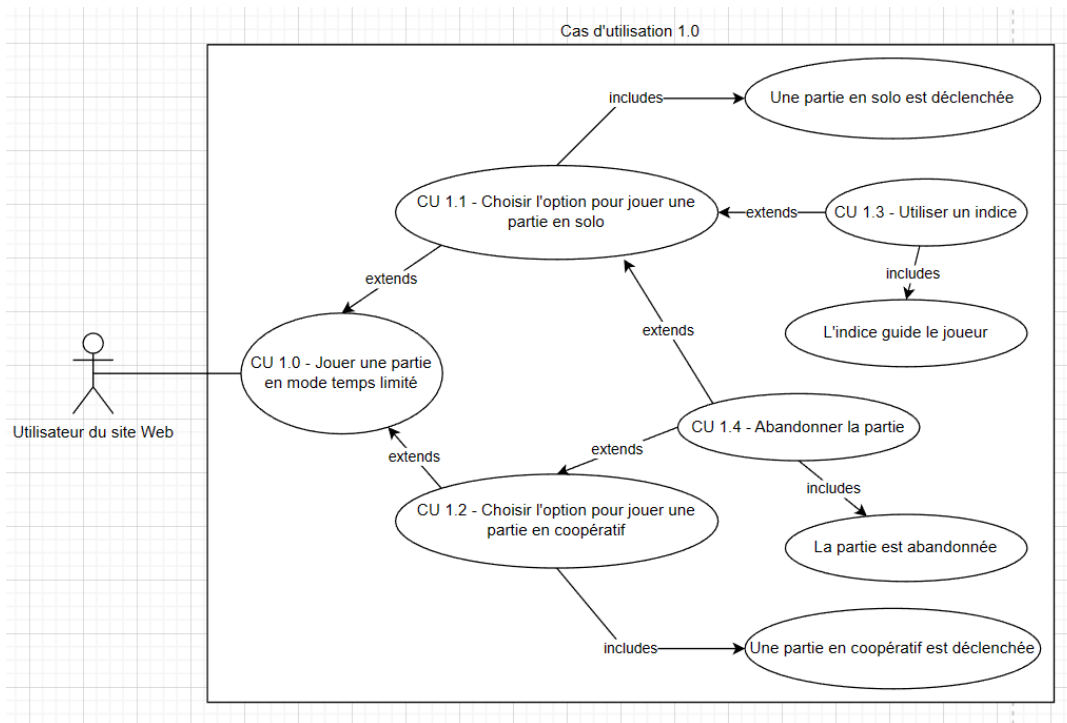


Diagramme 2 : Configurer le jeu

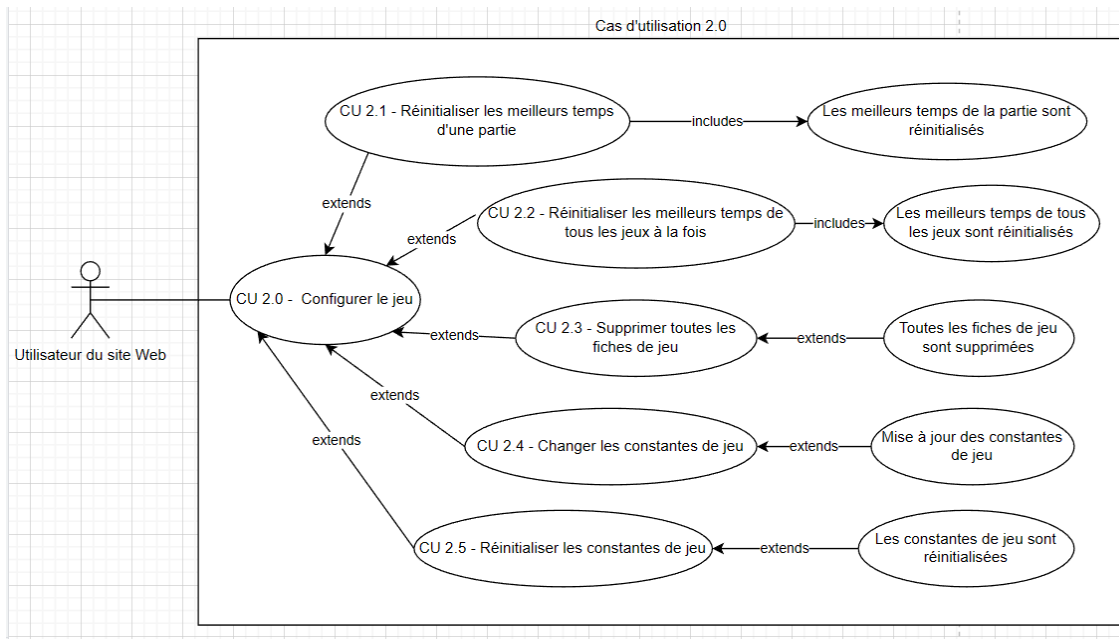


Diagramme 3: Contrôler la reprise vidéo d'une partie

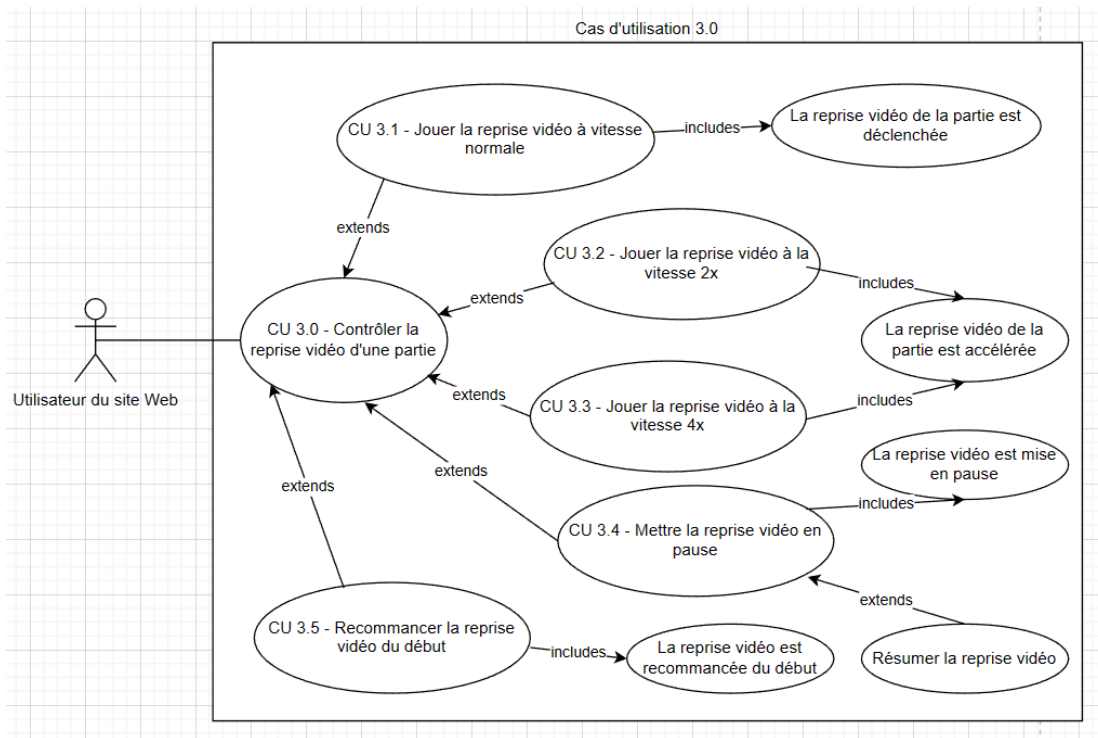
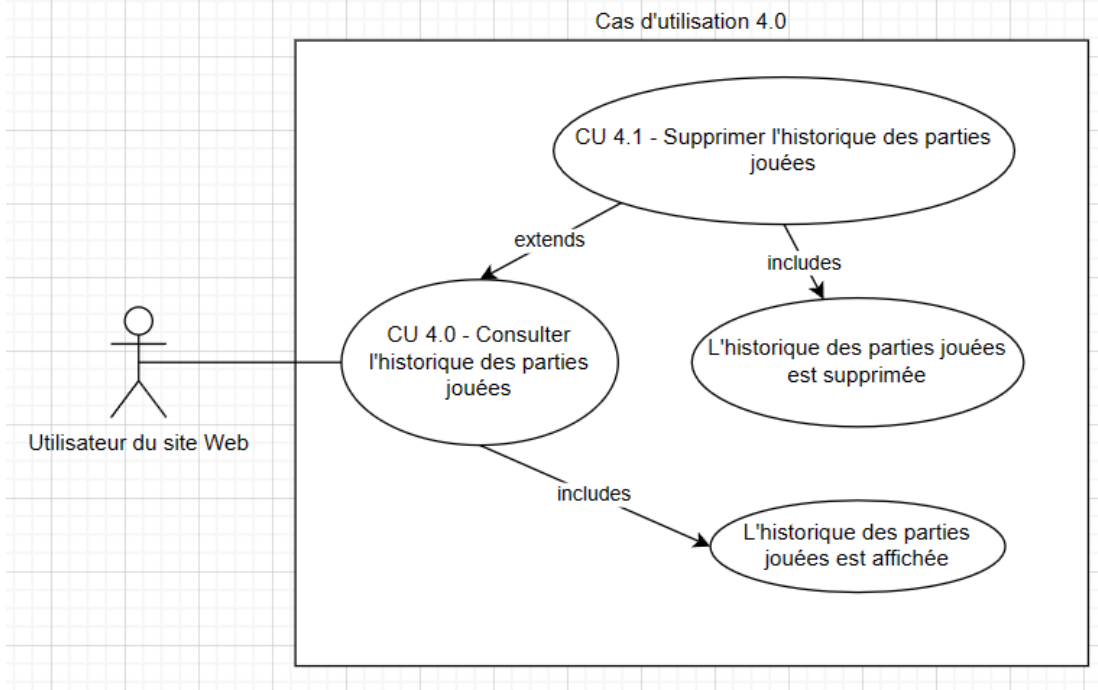
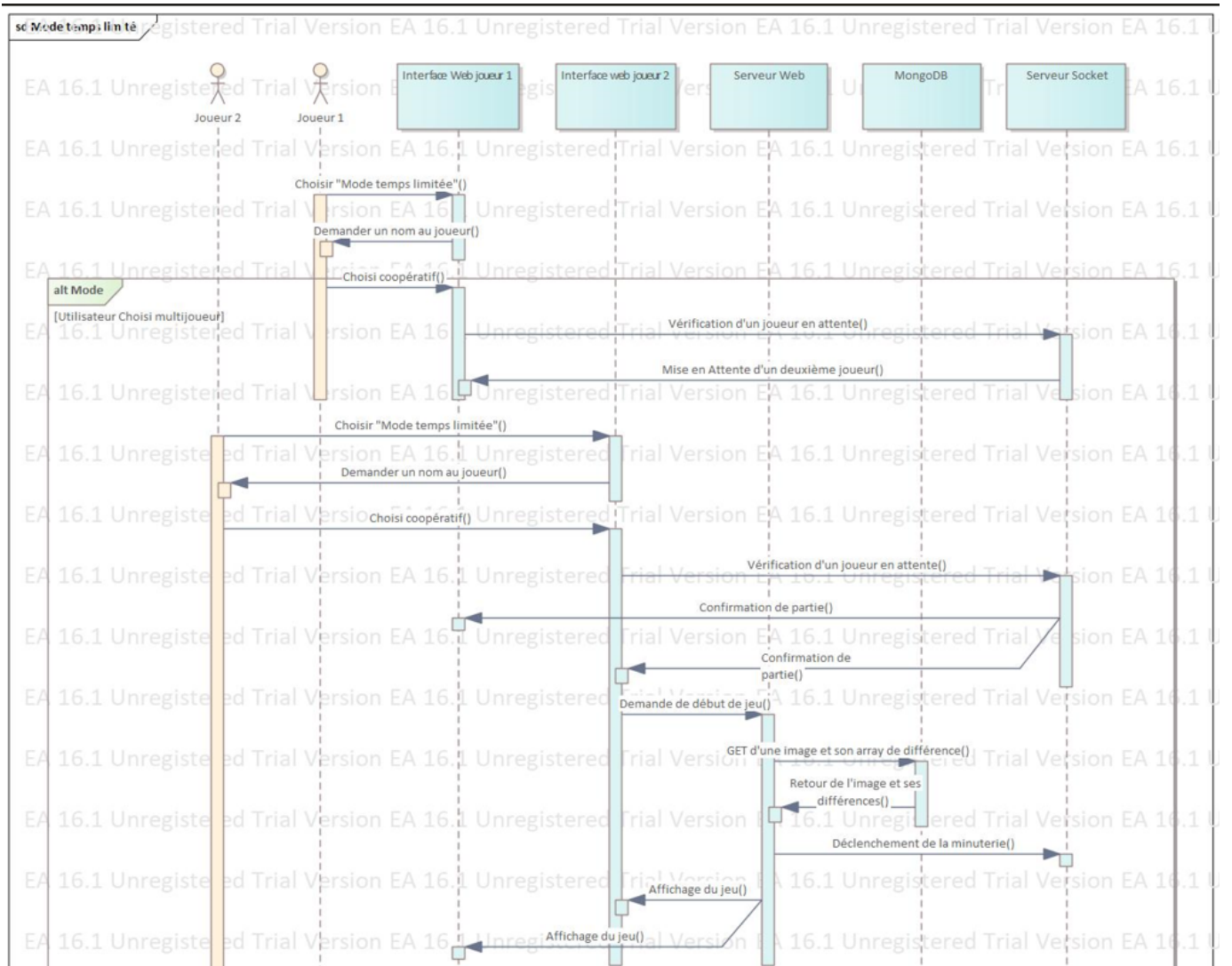


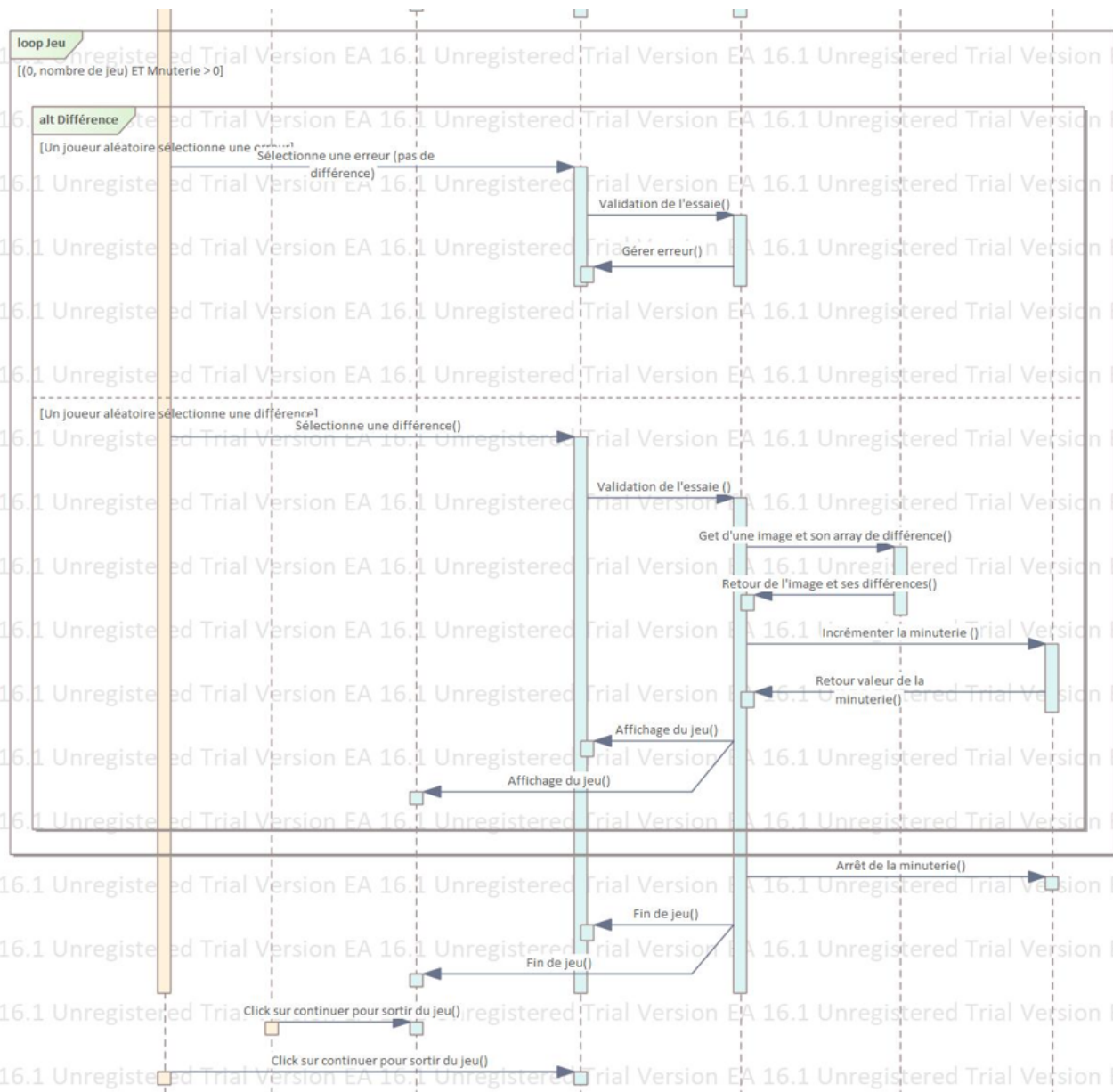
Diagramme 4 : Consulter l'historique des parties jouées



3. Vue des processus

Diagramme 5 : Jouer en temps limité





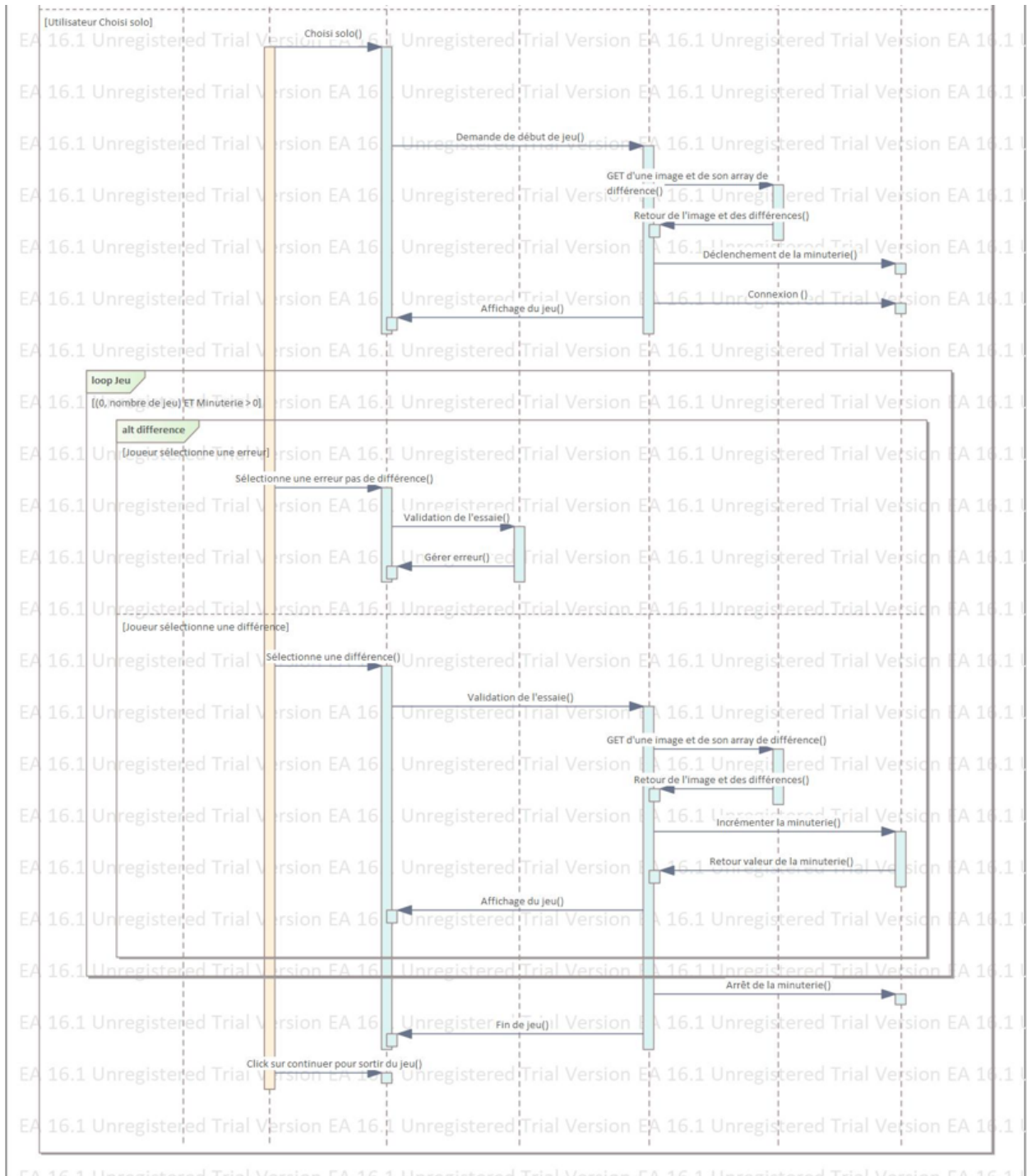


Diagramme 6 : Faire une reprise vidéo

sd diagramme d'interaction de la reprise vidéo

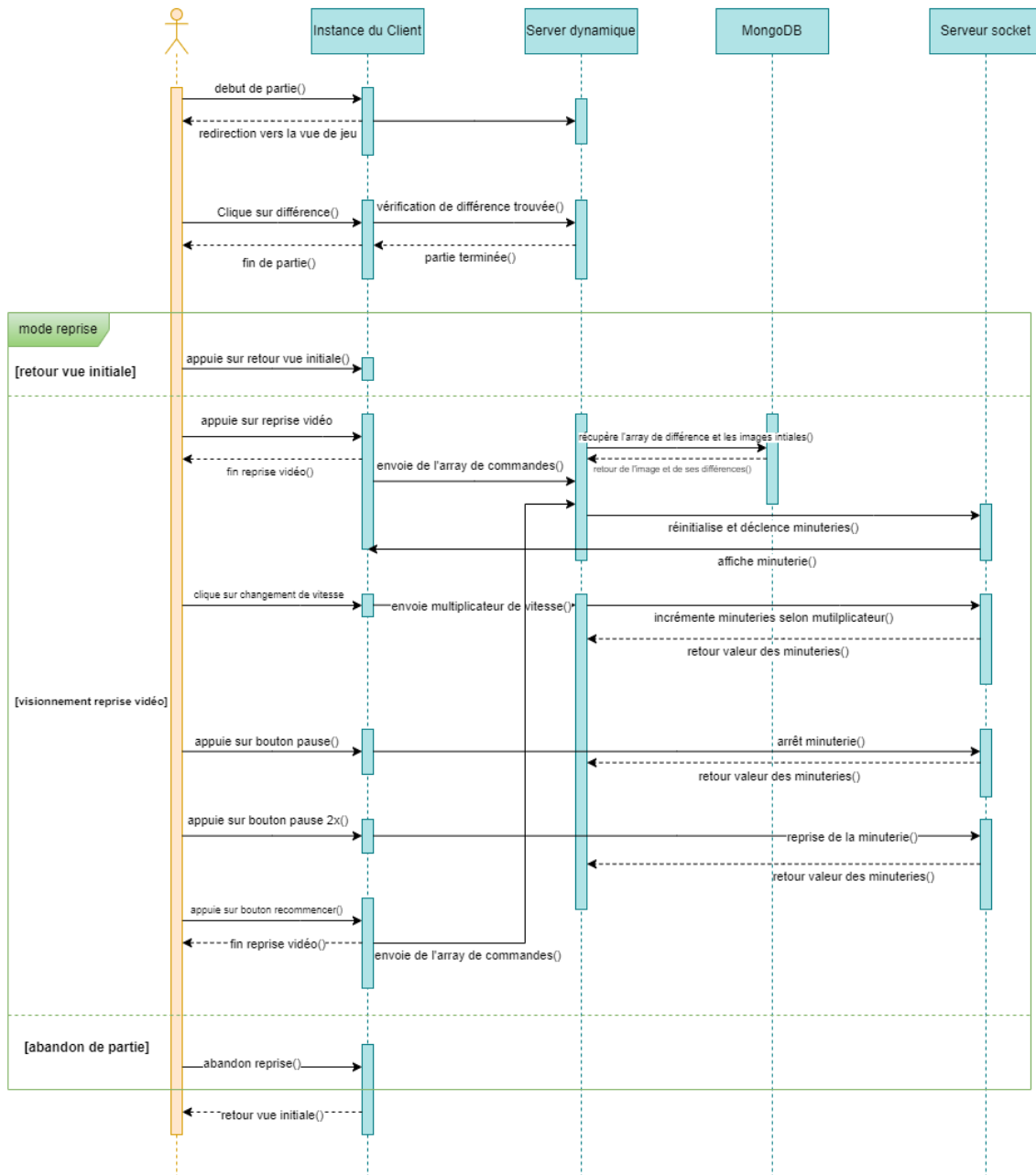
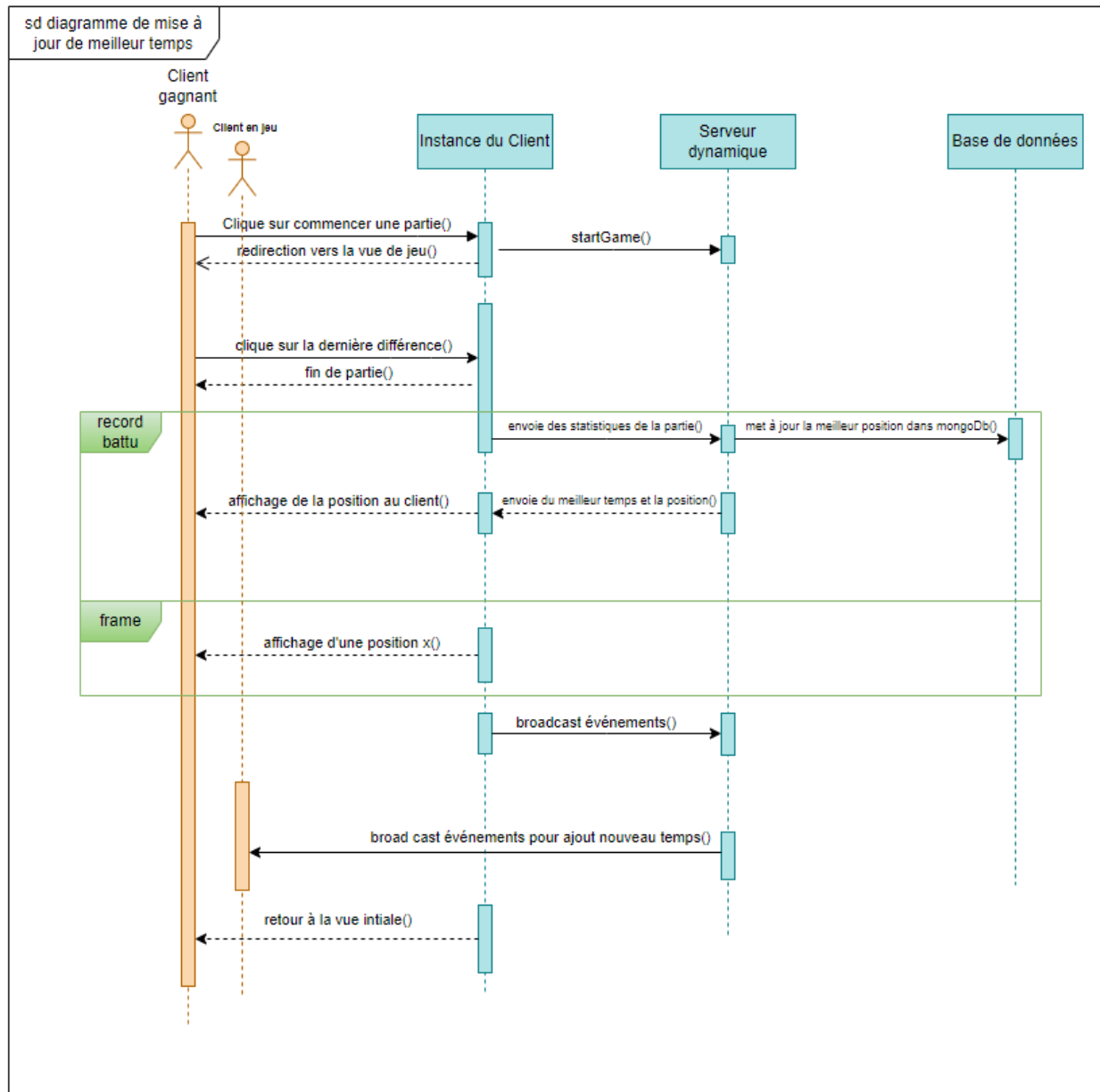
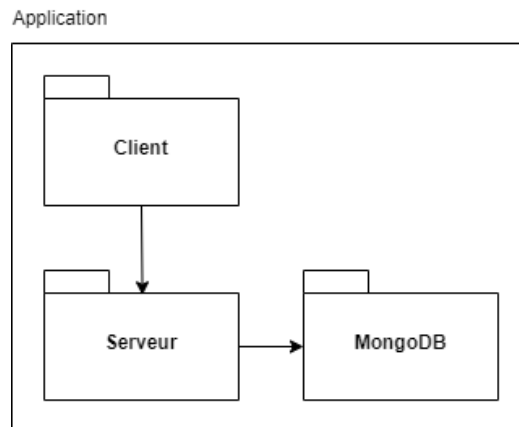


Diagramme 7 : Mettre à jour le meilleur temps



4. Vue logique



Client

S'occupe de toute la vue de l'application, ainsi que de toute la logique qui doit s'effectuer directement sur le client et fait les appels nécessaires au serveur.

Serveur

S'occupe de toute la logique opérationnelle qui n'a pas besoin d'être effectuée sur le client, s'occupe de faire les appels à la base de données pour aller chercher l'information et permet de fournir cette information au client.

MongoDB

Représente l'endroit où nous stockons nos données. Tout cela est fait à partir du serveur. Nous n'avons pas de diagramme plus précis pour ce paquetage, car il s'agit ici uniquement d'une illustration pour montrer où se situe notre base de données.

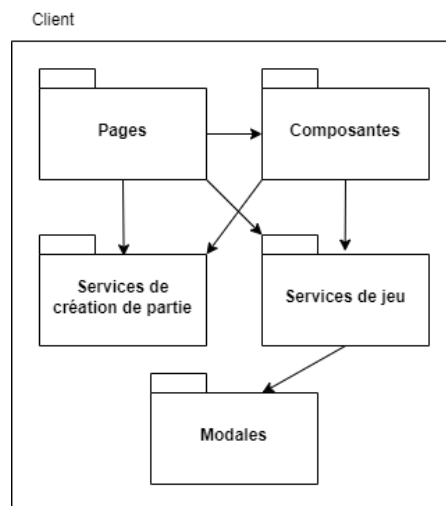
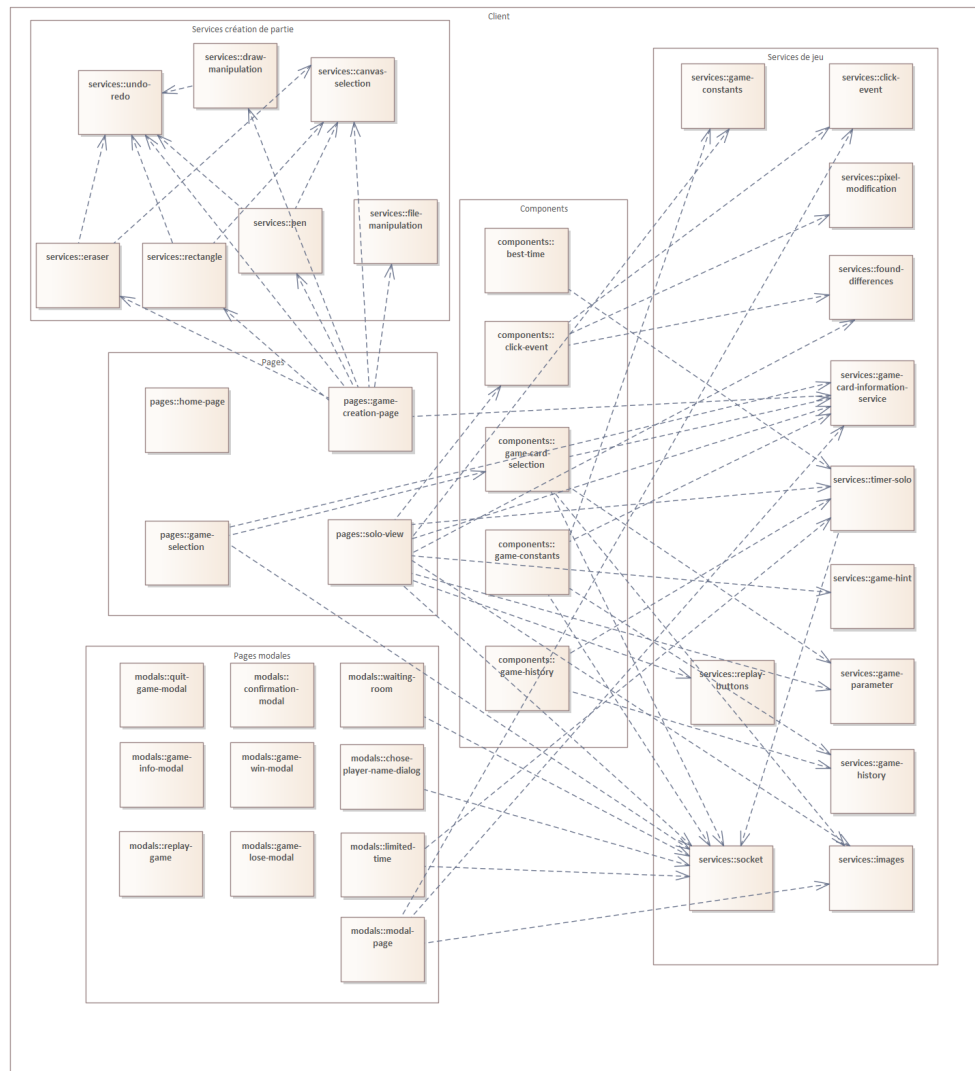


Diagramme 8 : Interactions au niveau du Client



Pages

S'occupe de l'affichage et de la logique simple des différentes pages de l'application

Components

S'occupe de l'affichage et de la logique simple des sous-éléments présents dans les pages. Par exemple, les canvas dans la page de jeu sont des composants.

Services de création de partie

S'occupe de la logique plus complexe des différents composants, permet le transfert de valeur entre ces derniers, et permet d'effectuer de la logique simple plus générale qui doit réutiliser à plusieurs endroits. Ceux-ci sont

utilisés pour tout ce qui touche à la création d'une partie.

Services de jeu

S'occupe de la logique plus complexe des différents composants, permet le transfert de valeur entre ces derniers, et permet d'effectuer de la logique simple plus générale qui doit réutiliser à plusieurs endroits. Ceux-ci sont utilisés pour tout ce qui touche directement au jeu, par exemples les indices, les meilleurs temps ou l'historique de parties.

Pages modales

S'occupe de l'affichage des différentes modales contenues dans l'application

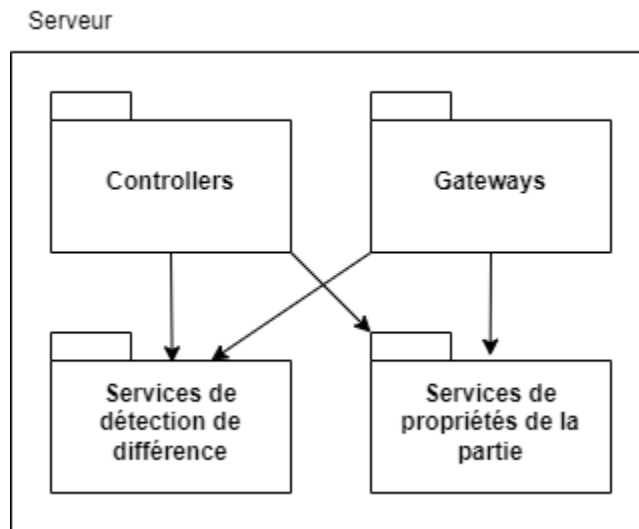
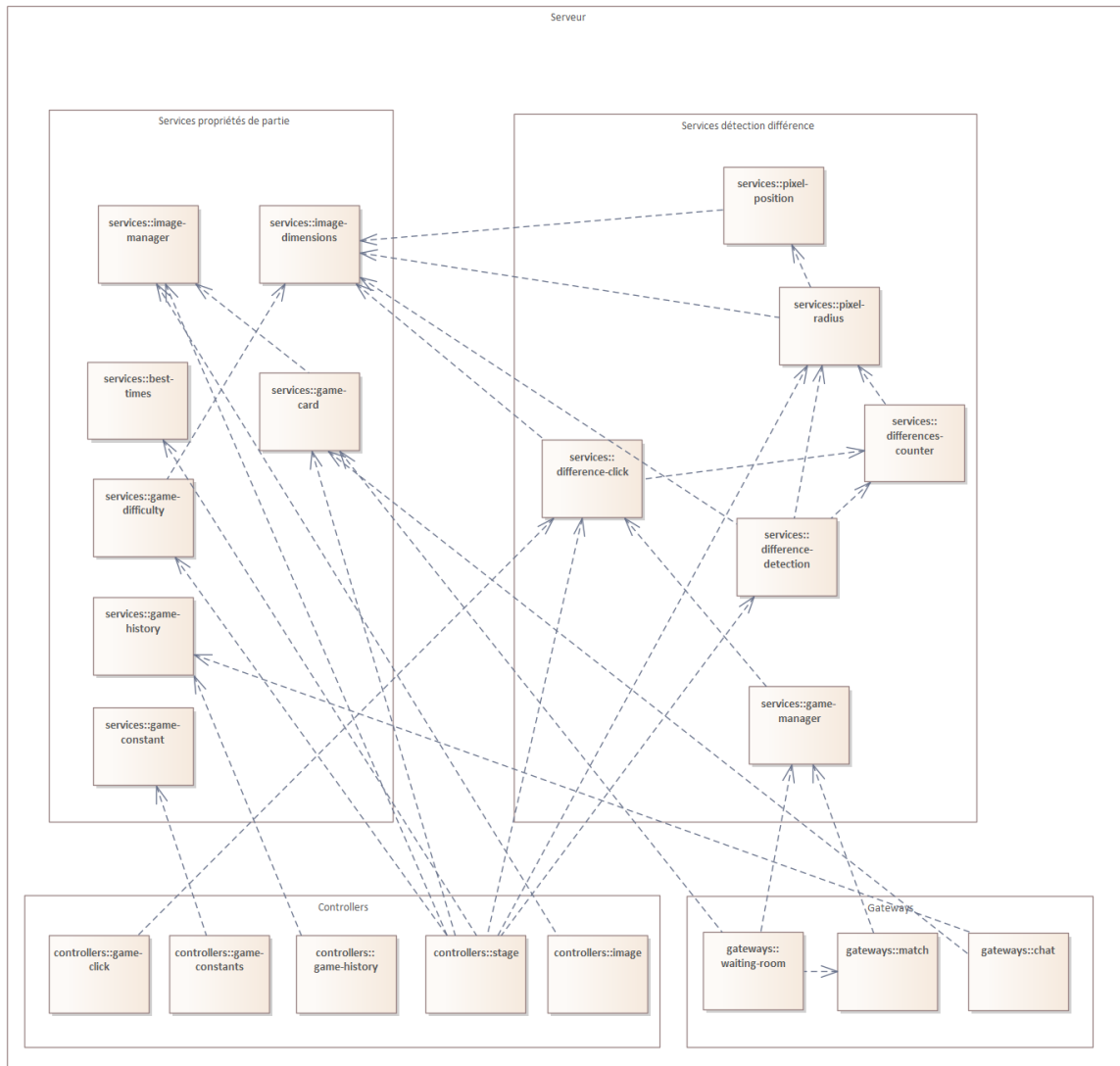


Diagramme 9 : Interactions au niveau du Serveur



Controllers

Permet de recevoir les différents appels HTTP et des appels aux Sockets venant du client, et d'exécuter la logique présente dans les services en conséquence

Gateways

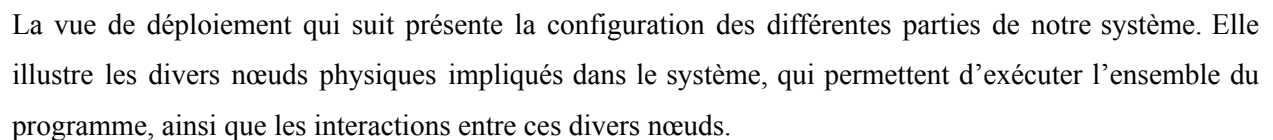
Initialise les événements possibles qu'un socket peut envoyer. Chaque subscribe message est un événement séparé qu'un socket quelconque peut envoyer au serveur et la logique qui suit dans la fonction en dessous représente la logique que le serveur implémente

Services de détection de différence

Permet de gérer la logique du côté serveur en ce qui concerne la détection d'une différence lors d'un clic, et la logique associée.

Permet de gérer la logique du côté serveur en ce qui concerne les propriétés d'une partie, que ce soit de la modification des images, à l'enregistrement des données dans la base de données.

Diagramme 10 : Diagramme de déploiement du projet d'application Web de l'équipe 103



D'abord, le système d'exploitation du premier joueur permet à celui-ci d'utiliser son navigateur Web. Le système d'exploitation du joueur peut être un système Windows, un système Linux, un système Mac ou un système d'une autre origine. Le navigateur utilisé par le joueur peut être Google Chrome, Microsoft Edge ou un autre navigateur permettant l'exécution du code. Par la suite, chaque joueur peut utiliser son navigateur Web pour se connecter au serveur statique du site Web, qui est déployé sur GitLab pages. Cette connexion est établie à travers une communication HTTP. C'est GitLab pages qui permettra au code du programme de l'application d'être exécuté par le navigateur. Plus précisément, GitLab pages permettra de déployer les pages du site Web et de les rendre accessibles au client à travers son navigateur Web.

Par la suite, chaque joueur est connecté à travers son navigateur Web à une même machine virtuelle EC2 fournie par Amazon Web Services (AWS). Cette machine virtuelle fournit le serveur dynamique de AWS, permettant au client de pouvoir chercher les détails concernant chaque partie, puis de se connecter à une salle où celui-ci pourra jouer une partie en mode classique un contre un, contre un autre joueur. La connexion entre le client et le serveur dynamique AWS est établie grâce à une communication HTTP, ainsi que le protocole WebSocket. La communication HTTP est utilisée pour envoyer les données d'un jeu au client, telles que les images du jeu, le nombre de différences, les pixels d'une différence, les meilleurs temps, etc. De son côté, SocketIO permet au client de se connecter à une salle, où celui-ci pourra soit attendre l'arrivée d'un autre joueur, ou bien se joindre à un autre joueur, dans le but de participer à une partie en mode classique un contre un. SocketIO permet également à deux joueurs qui se trouvent dans la même salle de s'envoyer et de recevoir des messages, qui seront visibles par les deux joueurs dans la salle.

Enfin, les informations sur chaque jeu ainsi que les différences contenues dans chaque partie sont stockées dans une base de données qui est hébergée sur MongoDB. En utilisant le protocole HTTP, le client envoie des requêtes au serveur dynamique AWS pour récupérer les informations d'une partie. Le serveur dynamique lui retourne alors une réponse contenant un code de retour correspondant au résultat de la requête, accompagné des informations que le client a demandées s'il n'y a pas d'erreurs. S'il y a une erreur dans la requête du client, alors ce dernier ne recevra qu'un code d'erreur du serveur, sans recevoir les données demandées. C'est à travers ce protocole de communication que le client pourra avoir accès aux données d'une partie de jeu Spot the Differences de l'équipe 103.