Équipe 103

Document d'architecture logicielle

Version 1.4

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

Table des matières

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.

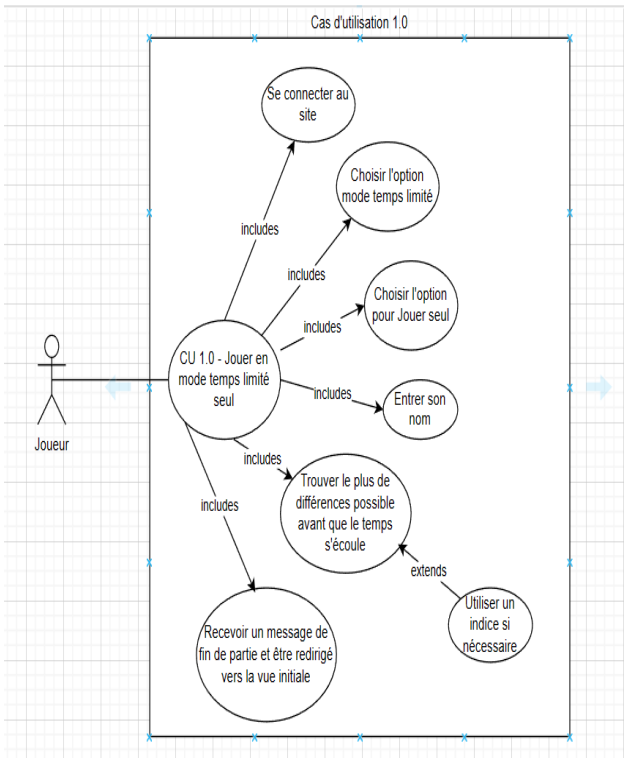
1.2. Contenu et organisation du document

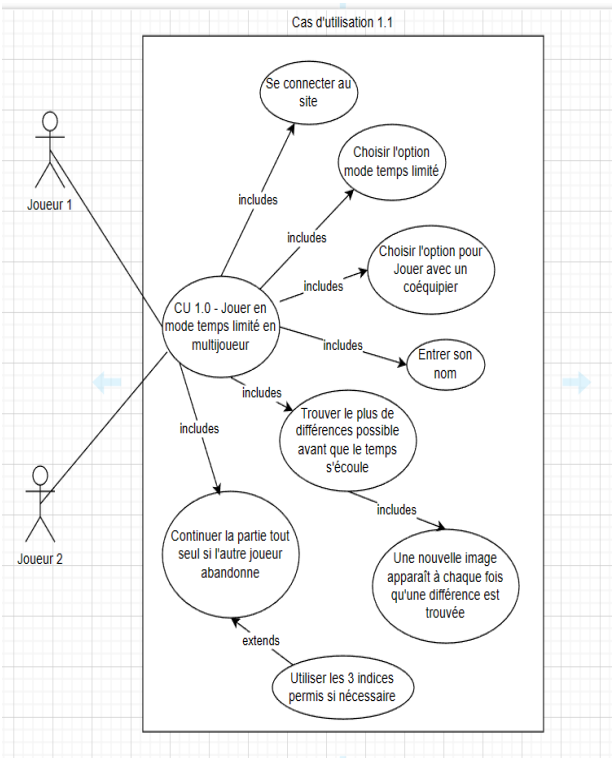
Ce document contient les différentes vues qui sont nécessaires à la compréhension générale de notre projet et de son fonctionnement, ainsi que son architecture.

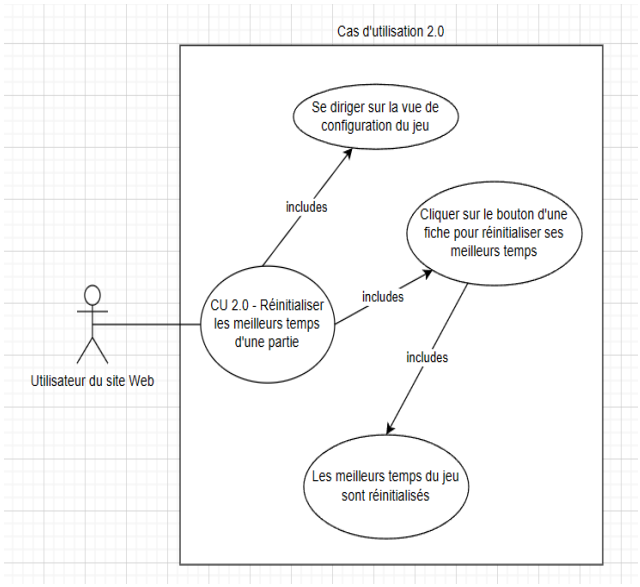
Nous avons commencé par décrire la vue des cas d'utilisation, qui consiste en les différents cas d'utilisations pertinents pour notre projet. Puis, avec l'aide d'un diagramme de séquence, nous avons décrit la vue de processus, qui décrit le système en termes d'interactions entre les différents processus 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.

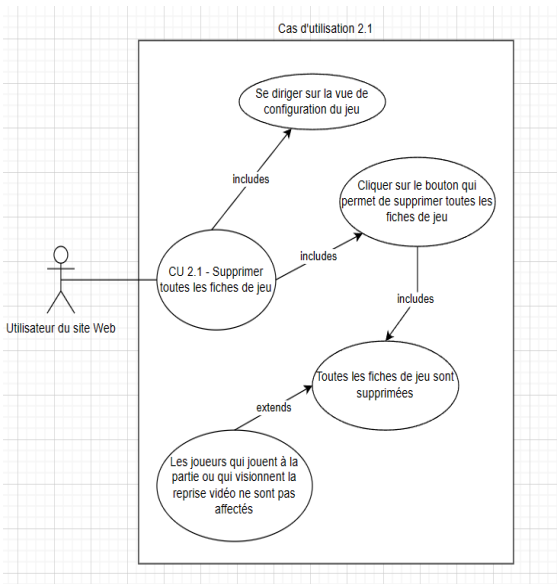
\

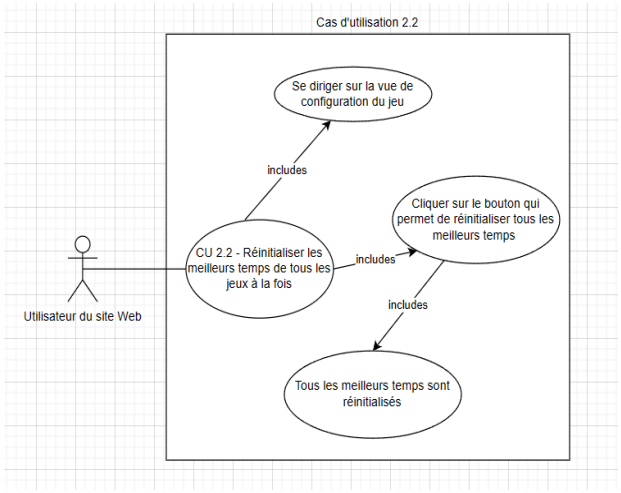
2. Vue des cas d’utilisation

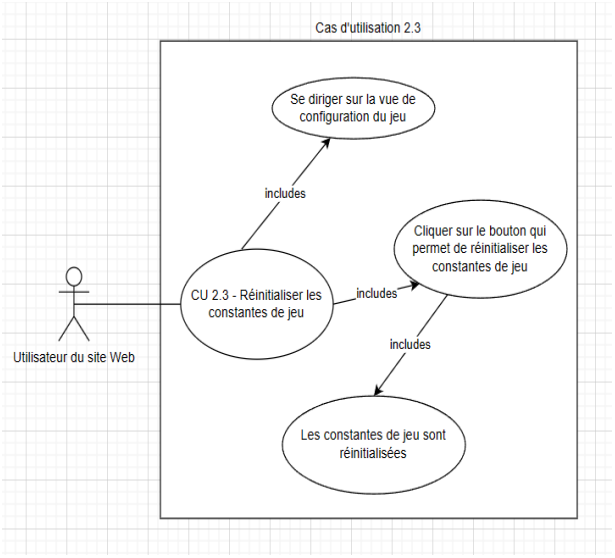


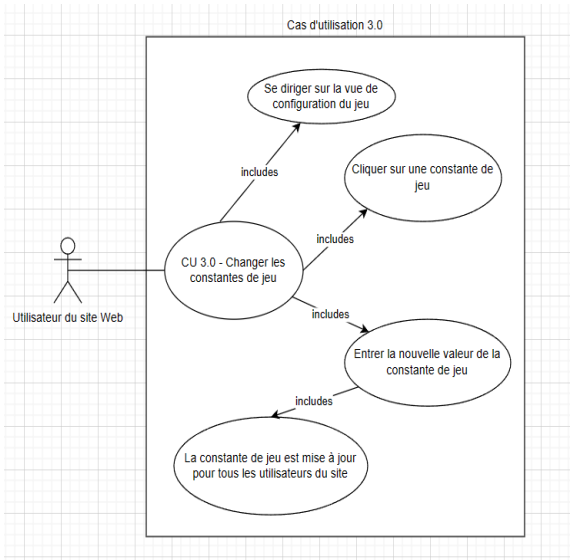


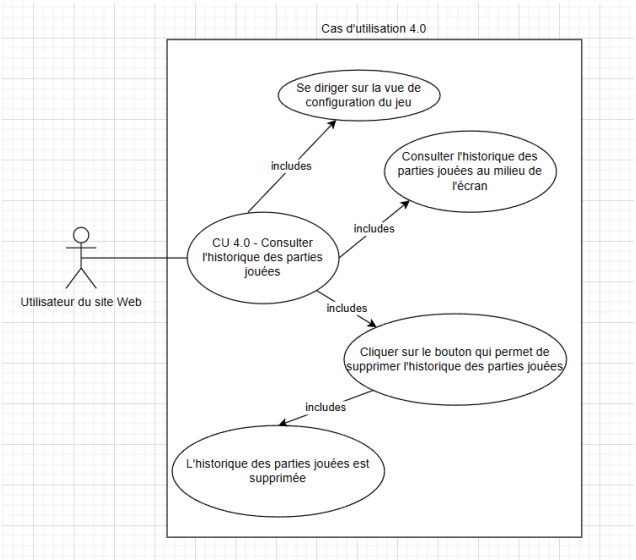

Cas d'utilisation 2.0
Se diriger sur la vue de configuration du jeu
includes
Cliquer sur le bouton d'une fiche pour réinitialiser ses meilleurs temps
includes
CU 2.0 - Réinitialiser les meilleurs temps d'une partie
includes
Utilisateur du site Web
Les meilleurs temps du jeu sont réinitialisés

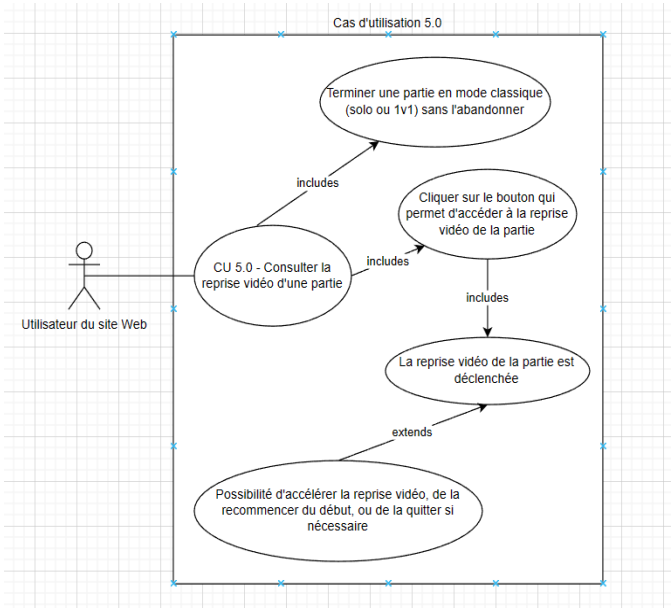

Cas d'utilisation 2.1
Se diriger sur la vue de configuration du jeu
includes
Cliquer sur le bouton qui permet de supprimer toutes les fiches de jeu
includes
CU 2.1 - Supprimer toutes les fiches de jeu
includes
Utilisateur du site Web
Toutes les fiches de jeu sont supprimées
extends
Les joueurs qui jouent à la partie ou qui visionnent la reprise vidéo ne sont pas affectés


Cas d'utilisation 2.2
Se diriger sur la vue de configuration du jeu
includes
Cliquer sur le bouton qui permet de réinitialiser tous les meilleurs temps
includes
CU 2.2 - Réinitialiser les meilleurs temps de tous les jeux à la fois
includes
Utilisateur du site Web
Tous les meilleurs temps sont réinitialisés

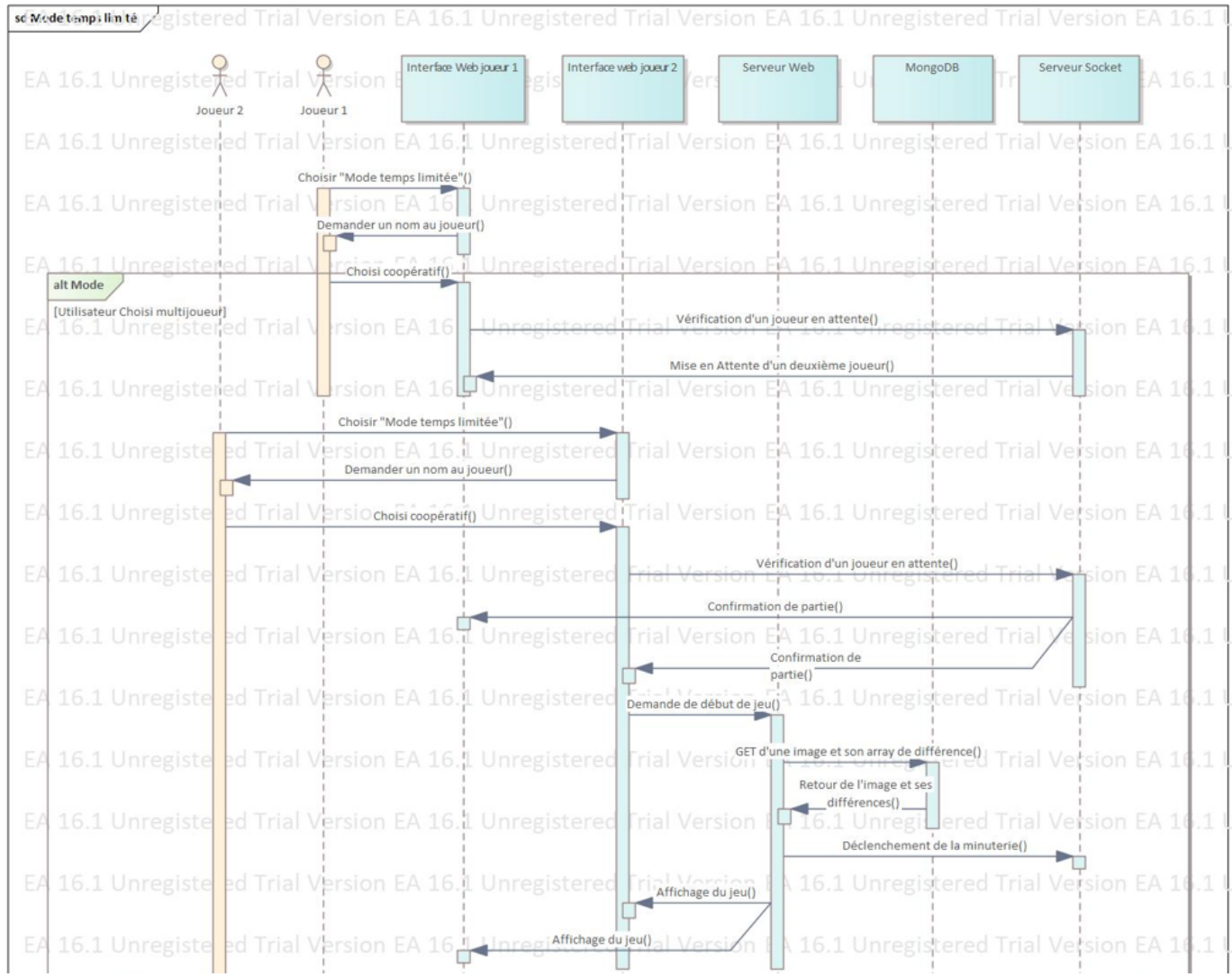

Cas d'utilisation 2.3
Se diriger sur la vue de configuration du jeu
includes
Cliquer sur le bouton qui permet de réinitialiser les constantes de jeu
includes
CU 2.3 - Réinitialiser les constantes de jeu
includes
Utilisateur du site Web
Les constantes de jeu sont réinitialisées

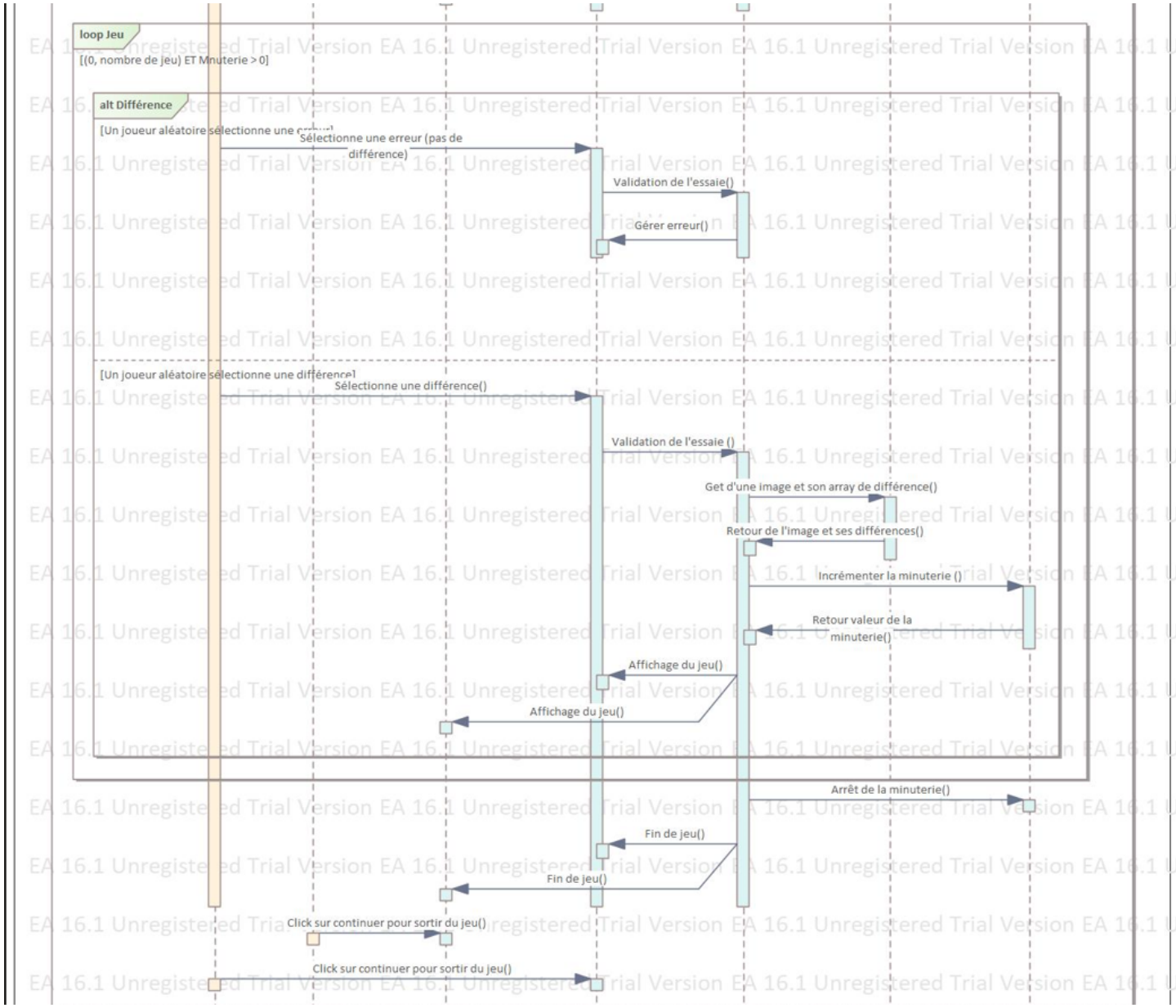

Cas d'utilisation 3.0
Se diriger sur la vue de configuration du jeu
Cliquer sur une constante de jeu
includes
includes
CU 3.0 - Changer les constantes de jeu
includes
Utilisateur du site Web
Entrer la nouvelle valeur de la constante de jeu
includes
La constante de jeu est mise à jour pour tous les utilisateurs du site

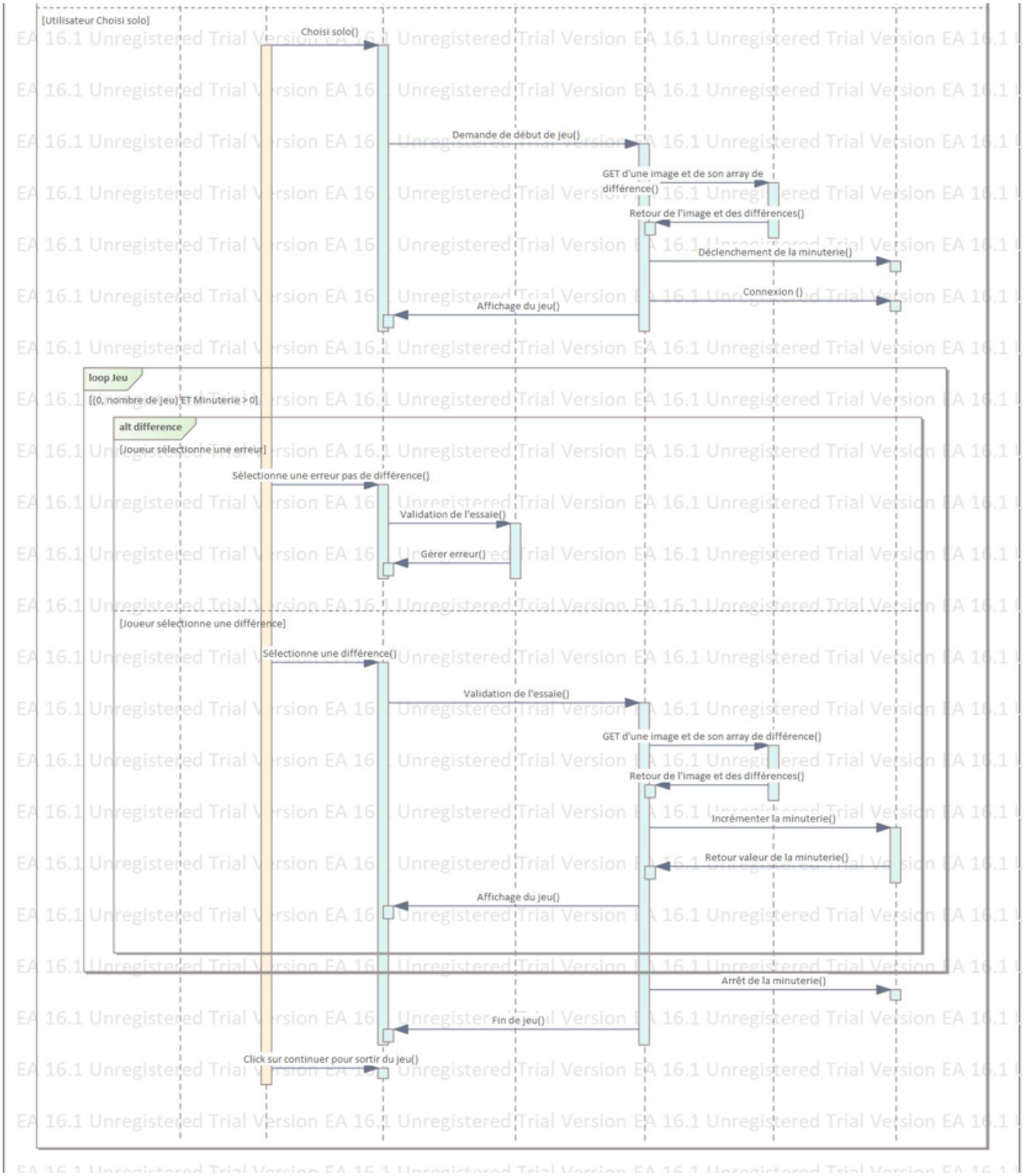

Cas d'utilisation 4.0
Se diriger sur la vue de configuration du jeu
Consulter l'historique des parties jouées au milieu de l'écran
includes
includes
CU 4.0 - Consulter l'historique des parties jouées
Utilisateur du site Web
includes
Cliquer sur le bouton qui permet de supprimer l'historique des parties jouées
includes
L'historique des parties jouées est supprimée

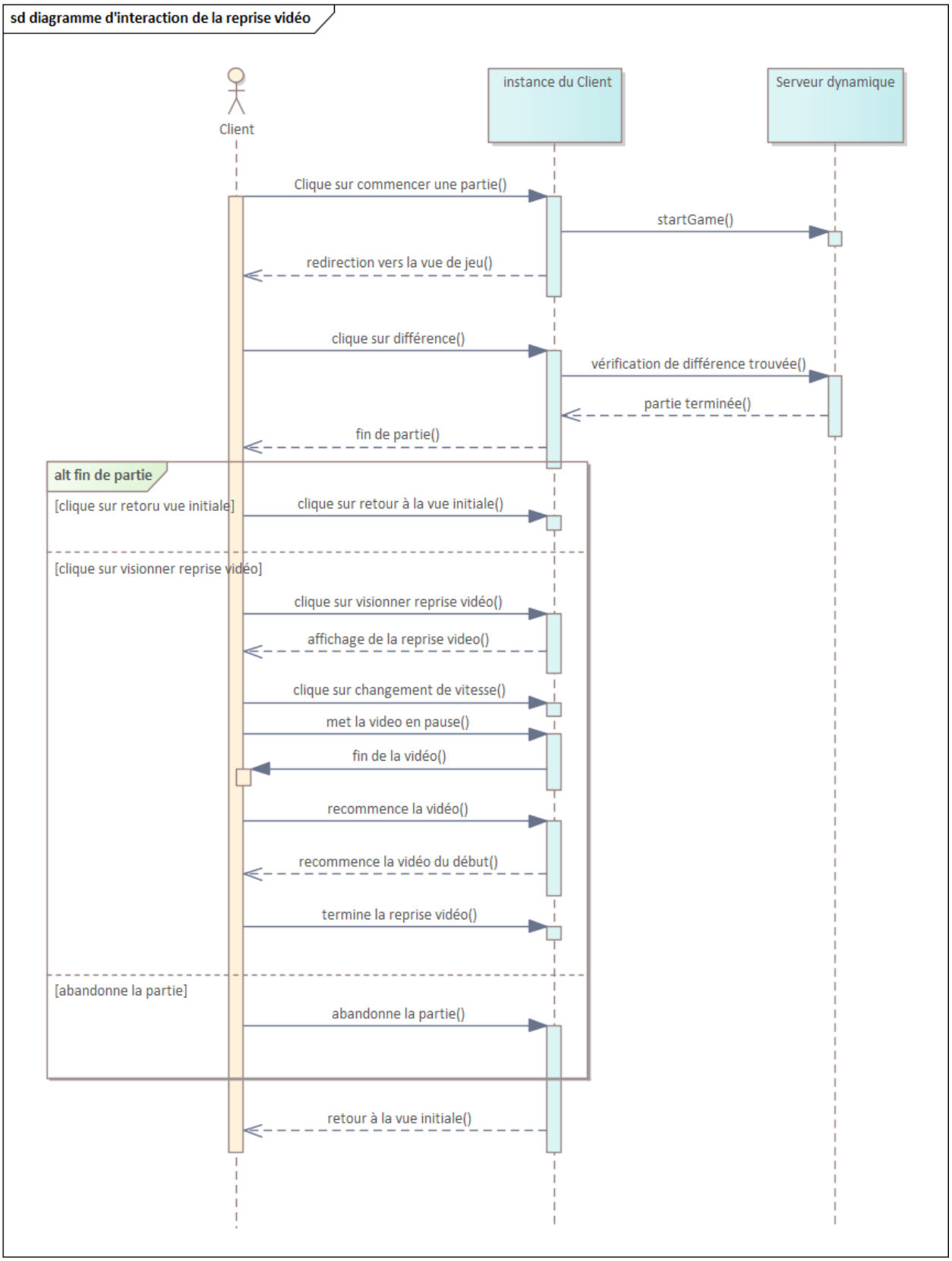

Cas d'utilisation 5.0
Terminer une partie en mode classique (solo ou 1v1) sans l'abandonner
includes
Cliquer sur le bouton qui permet d'accéder à la reprise vidéo de la partie
includes
CU 5.0 - Consulter la reprise vidéo d'une partie
includes
Utilisateur du site Web
La reprise vidéo de la partie est déclenchée
extends
Possibilité d'accélérer la reprise vidéo, de la recommencer du début, ou de la quitter si nécessaire

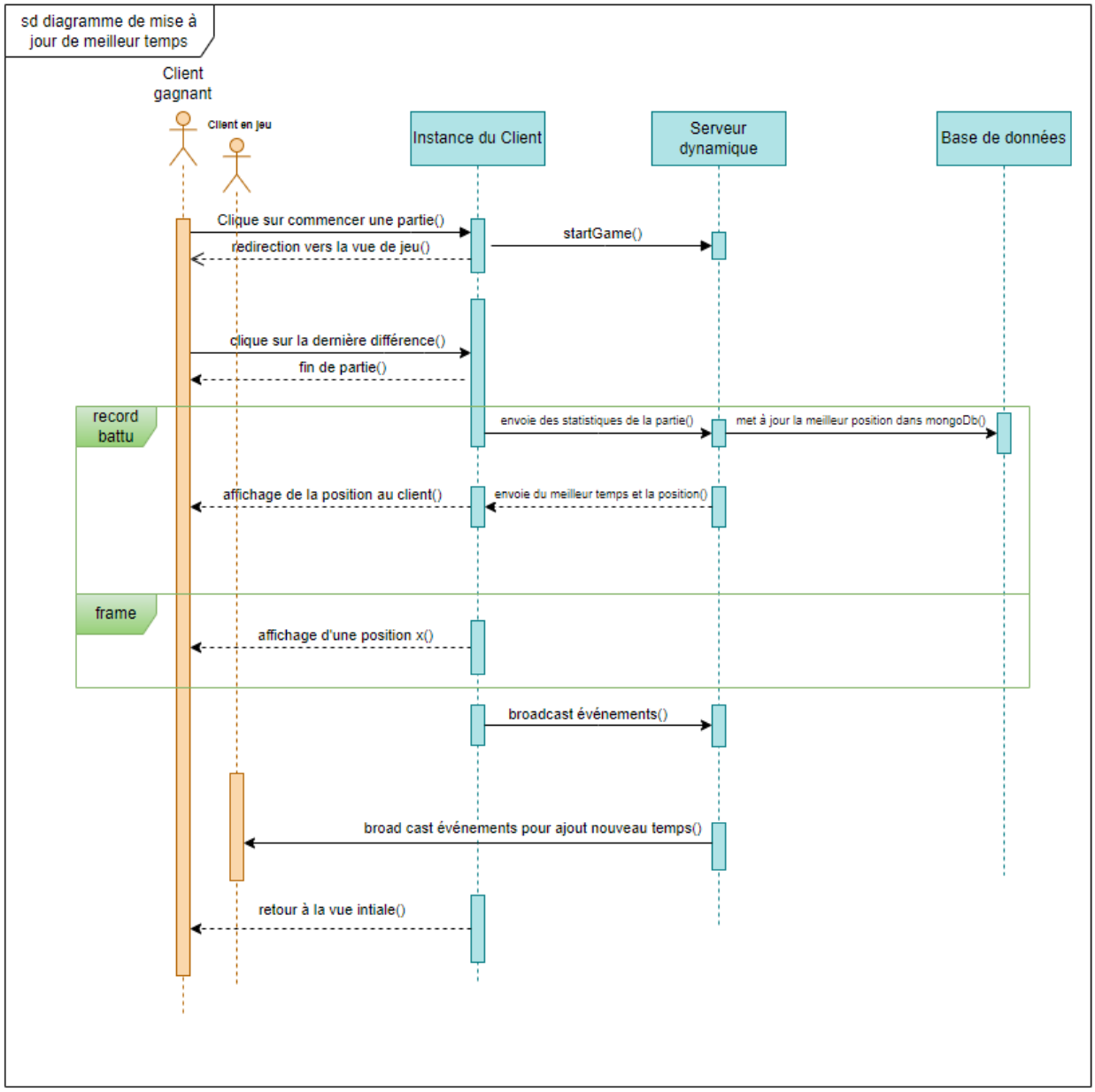
3. Vue des processus



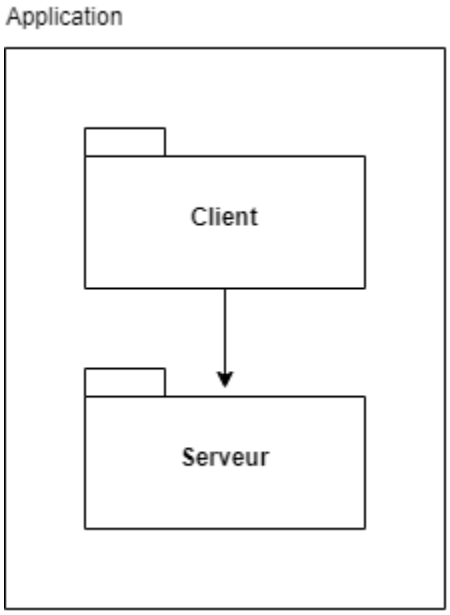

loop Jeu
[(0, nombre de jeu) ET Mnuterie > 0]
alt Différence
[Un joueur aléatoire sélectionne une erreur]
Sélectionne une erreur (pas de différence)
Validation de l'essaie()
Gérer erreur()
[Un joueur aléatoire sélectionne une différence]
Sélectionne une différence()
Validation de l'essaie ()
Get d'une image et son array de différence()
Retour de l'image et ses différences()
Incrémenter la minuterie ()
Retour valeur de la minuterie()
Affichage du jeu()
Affichage du jeu()
Arrêt de la minuterie()
Fin de jeu()
Fin de jeu()
Click sur continuer pour sortir du jeu()
Click sur continuer pour sortir du jeu()


[Utilisateur Choisi solo]
Choisi solo()
Demande de début de jeu()
GET d'une image et de son array de différence()
Retour de l'image et des différences()
Déclenchement de la minuterie()
Connexion ()
Affichage du jeu()
loop Jeu
[(0, nombre de jeu) ET Minuterie > 0]
alt difference
[Joueur sélectionne une erreur]
Sélectionne une erreur pas de différence()
Validation de l'essaie()
Gérer erreur()
[Joueur sélectionne une différence]
Sélectionne une différence()
Validation de l'essaie()
GET d'une image et de son array de différence()
Retour de l'image et des différences()
Incrémenter la minuterie()
Retour valeur de la minuterie()
Affichage du jeu()
Arrêt de la minuterie()
Fin de jeu()
Click sur continuer pour sortir du jeu()


sd diagramme d'interaction de la reprise vidéo
Client
instance du Client
Serveur dynamique
Clique sur commencer une partie()
startGame()
redirection vers la vue de jeu()
clique sur différence()
vérification de différence trouvée()
partie terminée()
fin de partie()
alt fin de partie
[clique sur retoru vue initiale]
clique sur retour à la vue initiale()
[clique sur visionner reprise vidéo]
clique sur visionner reprise vidéo()
affichage de la reprise video()
clique sur changement de vitesse()
met la video en pause()
fin de la vidéo()
recommence la vidéo()
recommence la vidéo du début()
termine la reprise vidéo()
[abandonne la partie]
abandonne la partie()
retour à la vue initiale()

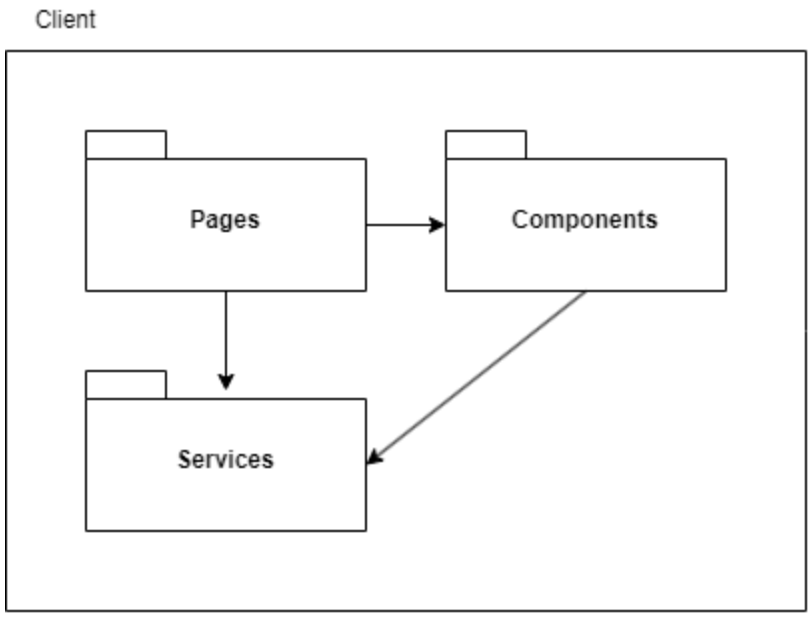

sd diagramme de mise à jour de meilleur temps
Client gagnant
Client en jeu
Instance du Client
Serveur dynamique
Base de données
Clique sur commencer une partie()
redirection vers la vue de jeu()
startGame()
clique sur la dernière différence()
fin de partie()
record battu
envoie des statistiques de la partie()
met à jour la meilleur position dans mongoDb()
affichage de la position au client()
envoie du meilleur temps et la position()
frame
affichage d'une position x()
broadcast événements()
broad cast événements pour ajout nouveau temps()
retour à la vue intiale()

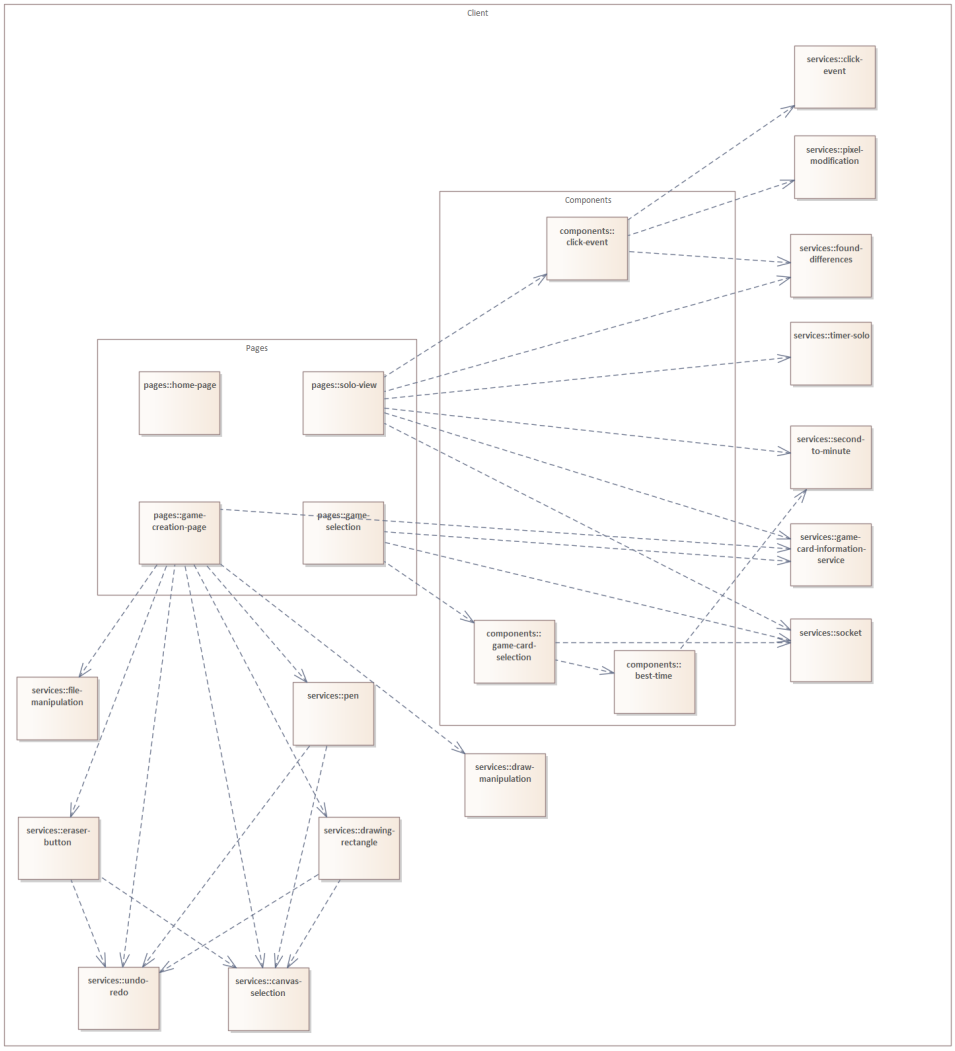
4. Vue logique



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.

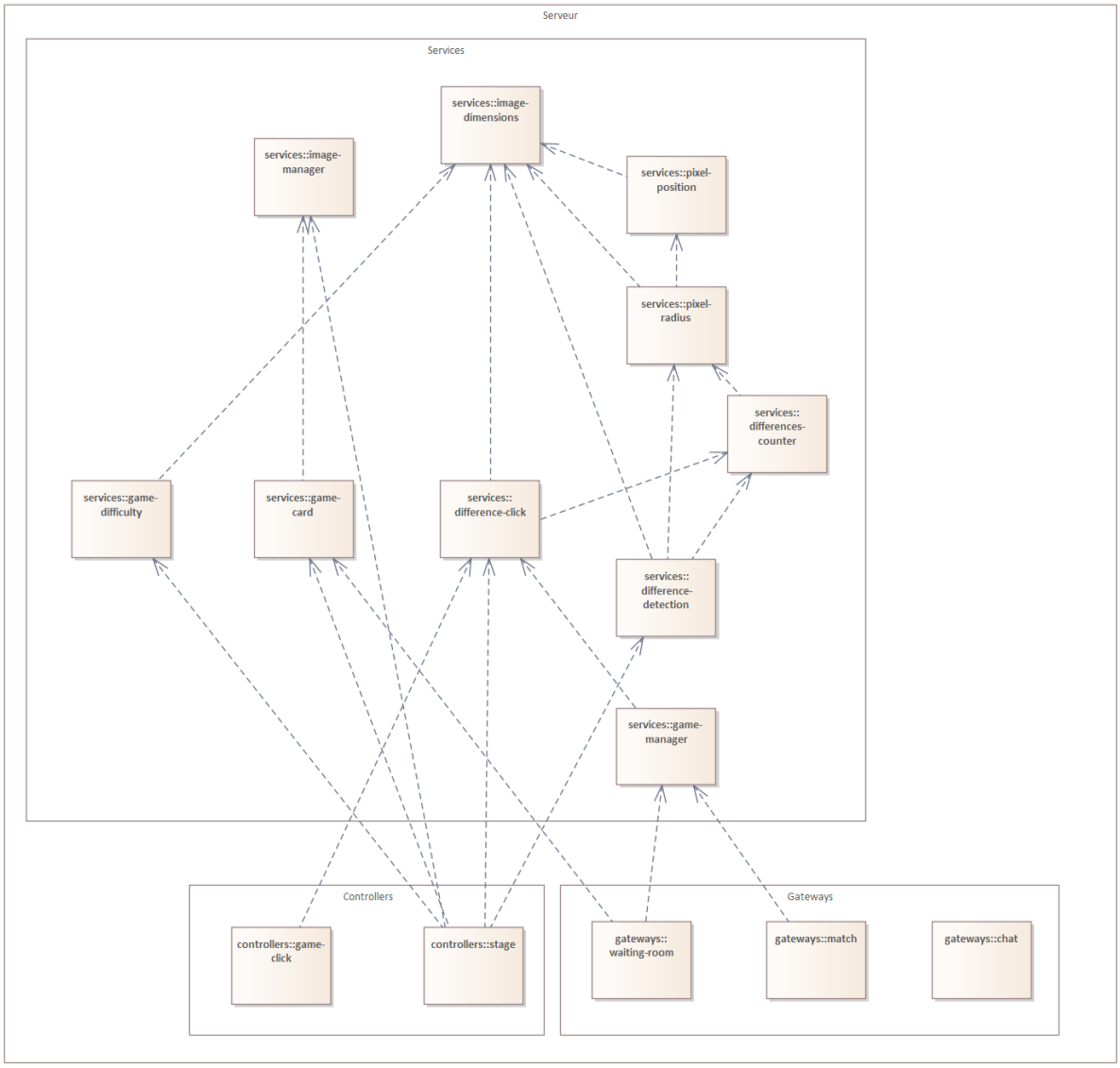

Client
Pages
Components
Services

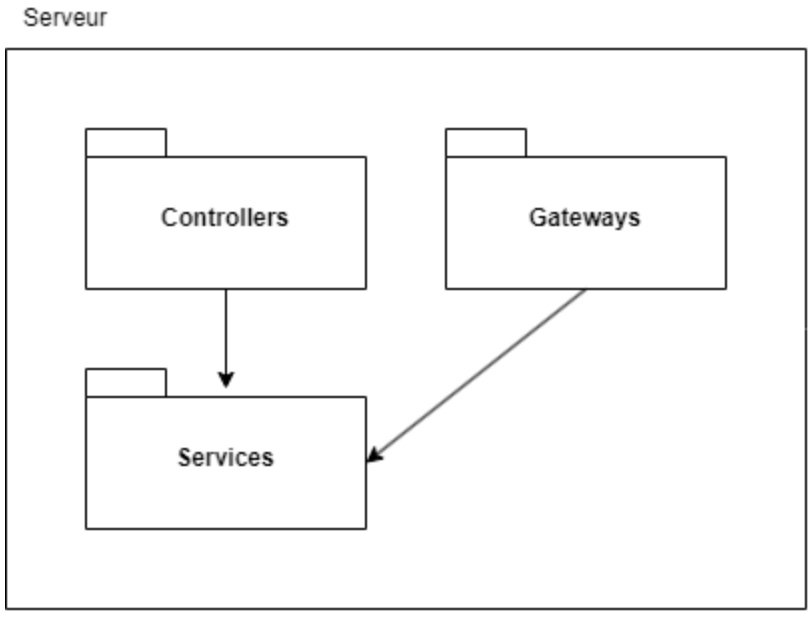

Client
Components
Pages
services::click-event
services::pixel-modification
components::click-event
services::found-differences
services::timer-solo
pages::home-page
pages::solo-view
services::second-to-minute
pages::game-creation-page
pages::game-selection
services::game-card-information-service
components::game-card-selection
components::best-time
services::socket
services::file-manipulation
services::pen
services::draw-manipulation
services::eraser-button
services::drawing-rectangle
services::undo-redo
services::canvas-selection

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
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. Certains services servent aussi à envoyer des requêtes au serveur.


Serveur
Services
services::image-dimensions
services::image-manager
services::pixel-position
services::pixel-radius
services:: differences-counter
services::game-difficulty
services::game-card
services:: difference-click
services:: difference-detection
services::game-manager
Controllers
controllers::game-click
controllers::stage
Gateways
gateways:: waiting-room
gateways::match
gateways::chat


Serveur
Controllers
Gateways
Services

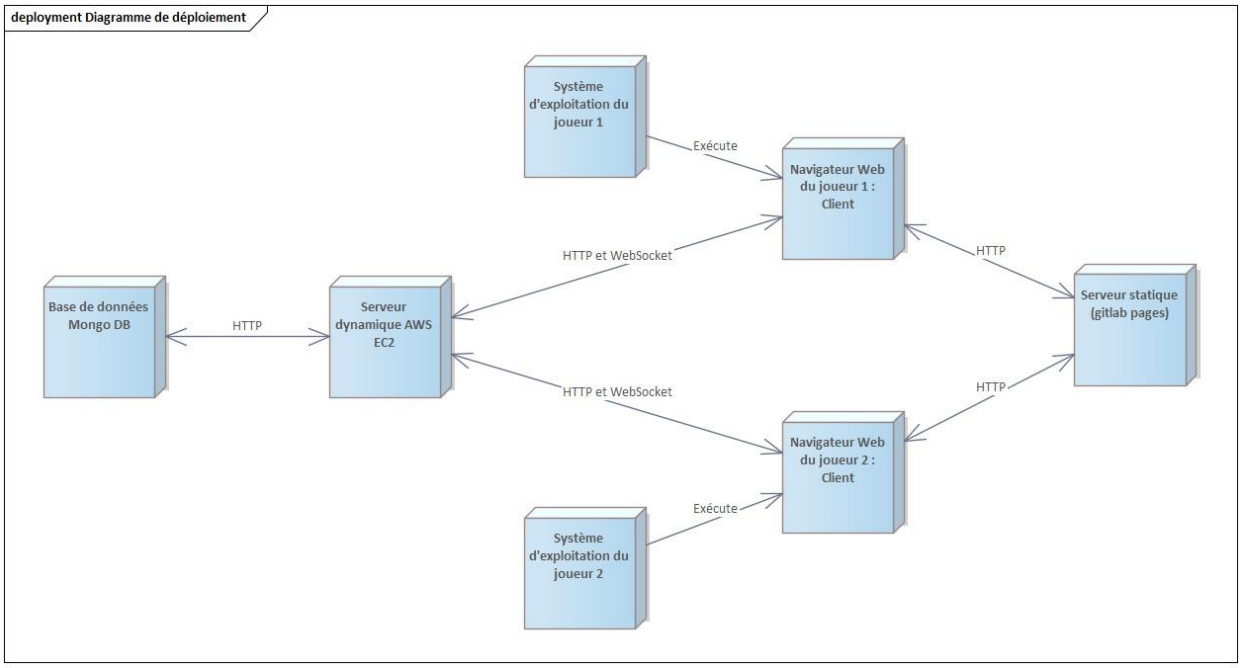
Controllers
Permet de recevoir les différents appels HTTP venant du client, et d'exécuter la logique présente dans les services en conséquence

Gateways
Initialise les événements possibles qu'un socket peut envoyer. Chaque subscribe message est un événement séparer 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
Permet de gérer la logique du côté serveur

5. Vue de déploiement

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



La vue de déploiement qui suit présente la configuration des différentes parties de notre système. Elle illustre les divers nœuds physiques impliqués dans le système, qui permettent d'exécuter l'ensemble du programme, ainsi que les interactions entre ces divers nœuds.

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, Internet Explorer ou un autre navigateur. 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 sur GitLab pages que le code du programme de l’application sera exécuté. GitLab pages permettra ainsi 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 joue le rôle de 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 l’API 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, l’API WebSocket 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. L’API WebSocket permet également à deux joueurs qui se trouvent dans la même salle de s’envoyer et de 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.