101

Protocole de communication

Version 1.3

Historique des révisions

Date	Version	Description	Auteur
2023-03-18	1.0	Ajout des features utilisés avec leur protocoles associés (sprint 1 et 2)	Zakaria Zair
2023-03-19	1.1	Modification des features avec leur protocoles associées et ajout de ceux du (sprint 3)	Zakaria Zair
2023-03-20	1.2	Ajout des descriptions des paquets (sprint 1 et 2)	Zakaria Zair
2023-03-20	1.2	Modification des descriptions des paquets (sprint 1 et 2)	Zakaria Zair
2023-03-20	1.3	Ajout des descriptions des paquets (sprint 3)	Zakaria Zair
2023-03-21	1.3	Révision du document	Edgar Kappauf

Table des matières

Protocole de communication

1. Introduction

Ce document décrit l'implémentation des fonctionnalités d'un projet de communication client-serveur. Il présente les protocoles de communication utilisés, ainsi que la description des paquets échangés entre les clients et le serveur. Les protocoles de communication utilisés dans ce projet sont HTTP et WebSocket, chacun ayant des fonctions spécifiques pour permettre une communication efficace et rapide. Les paquets échangés sont décrits en détail, y compris les informations transmises et leurs formats pour chaque protocole.

2. Communication client-serveur

Les protocoles utilisés pour la communication client-serveur du projet sont WebSocket et HTTP. Le protocole HTTP est utilisé pour les fonctionnalités primordiales qui ne demandent pas de réponses actives du site web tandis que le protocole WebSocket est utilisé pour les fonctionnalités qui demandent des requêtes et des réponses rapides.

Fonctionnalités	**Protocole utilisé**	**Raison d'utilisation**
Obtenir les fiches de jeu	HTTP	Obtenir les informations sur les différents jeux disponibles
Créer un jeu	HTTP	Permettre la création de nouveaux jeux pour les utilisateurs
Obtenir les constantes de jeu	HTTP	Obtenir les constantes de jeu
Vérifier si un jeu existe	HTTP	Vérifier si un jeu existe déjà parmi ceux créés avant de le créer
Supprimer un jeu	HTTP et WS	*Supprimer un jeu et mettre à jour en temps réel les fiches de jeu et la base de données*
Démarrer une partie de jeu en solo	WS	Démarrer une partie de jeu en mode solo
Démarrer une partie de jeu en 1v1	WS	Démarrer une partie de jeu en mode 1 contre 1 (avec toutes les étapes préalables en attente)
Rejoindre une partie de jeu en 1v1	WS	Rejoindre une partie de jeu en mode 1 contre 1 (avec toutes les étapes préalables en attente)
Mise à jour des meilleurs temps	HTTP	Mettre à jour les meilleurs temps pour un jeu donné
Mise à jour de l'historique des parties	HTTP/WS	Mettre à jour l'historique des parties pour un joueur donné. *L'ajout d'une partie jouée se fait à la fin d'une partie, ce qui se fait plus facilement par WS*

Utilisation des indices	WS	Permettre aux joueurs de voir des indices pour les aider à trouver les différences
Mise à jour des constantes de jeu	HTTP	Mettre à jour les constantes de jeu
Remise à défaut des données globales du site web	HTTP	Réinitialiser toutes les données globales du site web

3. Description des paquets

Protocole HTTP :

Méthode	Route	Paramètre(s) (requête)	Type du corps (requête/réponse)	Informations du paquet (réponse)	Code(s) de retour (réponse)
loadGameCarrousel (Get)	http://localhost:3000/api/games/carousel/:index	Index du carrousel	number / CarouselPaginator	Carrousel de fiches de jeu	200
loadConfigConstants (Get)	http://localhost:3000/api/games/constants		/ GameConfigConst	Constantes de jeu	200
loadGameById (Get)	http://localhost:3000/api/games/:id	Index du jeu	string / Game	Jeu	200
postGame (Post)	http://localhost:3000/api/games/	Jeu crée	CreateGameDto /		201
deleteGameById (Delete)	http://localhost:3000/api/games/:id	Index du jeu à supprimer	string /		200
verifyIfGameExists (Get)	http://localhost:3000/api/games/	Nom du jeu à comparer	Query(string) / boolean	Si le jeu existe ou non (boolean)	200
getBestTimes (Get)	http://localhost:3000/api/games/bestTimes:id	Index de la fiche de jeu	number / CarouselPaginator	Carrousel de fiches de jeu	200
modifyGameConstants (Patch)	http://localhost:3000/api/games/constants/:id	Index de la constante + nouvelle valeur de la constante	Query(string) + number / GameConfigConst	Constantes de jeu	200

resetGameConstants (Delete)	http://localhost:3000/api/games/constants		/ GameConfigConst	Constantes de jeu	200
modifyGameHistory (Patch)	http://localhost:3000/api/games/gameHistory/:id	Index de la partie à supprimer	number / GameHistory	Historique des parties jouées	200
getGameHistory (Get)	http://localhost:3000/api/games/gameHistory		/ GameHistory	Historique des parties jouées	200
deleteGameHistory (Delete)	http://localhost:3000/api/games/allGameHistory		/ GameHistory	Historique des parties jouées	200
resetBestTime (Patch)	http://localhost:3000/api/games/resetBestTime/:id	Index de la fiche de jeu	number / CarouselPaginator	Carrousel de fiches de jeu	200
deleteAllGame (Delete)	http://localhost:3000/api/games/deleteAllGames		/ CarouselPaginator	Carrousel de fiches de jeu	200
resetBestTimes (Patch)	http://localhost:3000/api/games/allBestTimes		/ CarouselPaginator	Carrousel de fiches de jeu	200
getReplayData (Get)	http://localhost:3000/api/games/replay/:roomId		/ Game	Jeu	200

Protocole WebSocket :

Nom de l'évent	Source	Contenu
CreateSoloGame	Client	gameId: string et playerName: string
StartGameByRoomId	Client	roomId: string et playerName: string
CreateOneVsOneRoom	Client	gameId: string
RemoveDiff	Client	coords: Coordinate[]
CheckStatus	Client	*Vide*
UpdateRoomOneVsOneAvailability	Client	gameId: string
CheckRoomOneVsOneAvailability	Client	gameId: string

UpdateWaitingPlayerNameList	Client	gameId: string
RefusePlayer	Client	gameId: string et playerName: string
AcceptPlayer	Client	gameId: string, roomId: string et playerNameCreator: string
CancelJoining	Client	roomId: string et playerName: string
AbandonGame	Client	*Vide*
LocalMessage	Client	tag: MessageTag.received et message: textMessage : ChatMessage
DeleteGameCard	Client	gameId: string
CreateSoloGame	Serveur	clientGame: ClientSideGame
GameStarted	Serveur	clientGame: ClientSideGame, players: Players et cheatDifferences: Coordinate[]
RemoveDiff	Serveur	differencesData: Differencese et playerId: string et cheatDifferences: Coordinate[]
TimerStarted	Serveur	timer: number
EndGame	Serveur	endGameMessage: string
LocalMessage	Serveur	receivedMessage: ChatMessage
RoomSoloCreated	Serveur	roomId: string
RoomOneVsOneAvailable	Serveur	availabilityData: RoomAvailability
OneVsOneRoomDeleted	Serveur	availabilityData: RoomAvailability
UpdateWaitingPlayerNameList	Serveur	waitingPlayerNameList: WaitingPlayerNameList
PlayerNameTaken	Serveur	playerNameAvailability: PlayerNameAvailability
RoomOneVsOneCreated	Serveur	roomId: string
UndoCreation	Serveur	gameId: string
PlayerAccepted	Serveur	acceptedPlayer: AcceptedPlayer
GameCardDeleted	Serveur	gameId: string

HintUsed	Client	roomId: string
HintUsed	Serveur	coords: Coordinate[] et nHintUsed: number
CreateCoopRoom	Client	gameId: string
LimitedTimerStarted	Serveur	timer: number
AbandonLimitedGame	Client	*Vide*
LimitedEndGame	Serveur	limitedEndGameMessage: string
GlobalMessage	Client	tag: MessageTag.received et message: textMessage : ChatMessage