Protocole de communication

Version 1.0

Historique des révisions

Date	Version	Description	Auteur
2023-03-20	1.0	Rédaction de la documentation (fin du sprint 2)	Sébastien Roy

Table des matières

Protocole de communication

1. Introduction

Ce document est divisé en 2 parties contenant 1 tableau et 2 tableaux respectivement.

La première partie contient le tableau 2.1 qui indique les décisions de protocole de transmission d'information entre l'application client et le serveur. Ce tableau présente chaque fonctionnalité de notre système, leur protocole de communication et le raisonnement derrière nos décisions de protocole.

La deuxième partie de ce document présente le format des données qui sont échangées entre notre application client et le serveur. Le tableau 3.1 est dédié aux interfaces qui sont échangées par le protocole HTTP, alors que le tableau 3.2 présente les interfaces qui sont échangées par le protocole WebSocket.

2. Communication client-serveur

Tableau 2.1 - Moyens de communication

Sujet	Protocole	Raisonnement
Création de jeux	HTTP	- C'est toujours le client qui initialise les demandes de création de jeu. Le serveur n'aura qu'à répondre aux requêtes du client.
Indices de jeu	WebSocket	- Permet de combiner au WebSockets des Sessions en ajoutant qu'un message. - Perm
Suppression de jeux	WebSocket/ HTTP	- HTTP permet de combiner la création et la suppression de jeux en un seul contrôleur. - Websocket permet de notifier les joueurs en attente que le jeu n'existe plus et ne peut donc plus être joué.
Historique des parties	WebSocket	- Les WebSockets du système de détection de différences permettent d'enregistrer les coordonnés des essais de l'utilisateur et à quels moments elles ont été faites. - Est combinée à la logique de détection des différences et écoute pour une demande de la reprise vidéo après que le message de victoire ai été envoyé au joueur par le serveur..
Détection de différences	WebSocket/ HTTP	- Les Websockets permettent au serveur de notifier tous les joueurs qu'une différence a été trouvée grâce au concept de salle. - Permet au serveur d'envoyer différents messages aux joueurs d'une même session, car ils sont dans la même salle. - HTTP permet de comparer 2 images lors de la création de nouveaux jeux.
Clavardage entre joueurs	WebSocket	- Permet la communication entre joueurs (en passant par le serveur). - Permet de grouper les utilisateurs pouvant communiquer entre eux dans des salles.
Minuterie de jeux	WebSocket	- Permet d'assurer la validité des temps enregistrés dans les tableaux des meilleurs temps (moins de chance de triche) en gardant la minuterie sur le serveur. - Permet de contrôler la mise à jour du temps sur toutes les interfaces utilisateurs à partir du serveur. (le serveur envoie un message à chaque seconde)
Tableaux des meilleurs temps	WebSocket	- Intégré dans notre logique de WebSockets pour la détection de différences. (Lorsqu'on obtient un vainqueur, on enregistre son temps et on le compare aux meilleurs temps) - Mise à jour en temps réel des tableaux des meilleurs résultats pour les clients sur la page de sélection de partie. - Est combiné avec les messages de partie qui sont en WebSocket.
Recherche de parties	WebSocket	- Mise à jour en temps réel de l'interfaces de tous les joueurs cherchant des parties en multijoueur. - Facilite la communication du serveur aux joueurs dans une session grâce au concept de salle.
Messages de partie	WebSocket	- Permet d'initier l'envoie de messages à tous les joueurs actifs à partir du serveur.

3. Description des paquets

Tableau 3.1 - Requêtes HTTP

Route		Méthode	Paramètres	Corps	Code(s) de la réponse	Corps de la réponse
api	/games	POST		name: string, radius: string, files: GameImageInput	200, 400	Game
		GET			200, 500	Game[]
		GET	id: string		200, 404, 500	Game
		DELETE	id: string		200	
		DELETE*			200	
		PATCH*	id: string	gameLeaderboard: [string, number][]	200	
	/games/history	DELETE*	id: string		200	
	/games/constants	PATCH*	id: string		200	
	/images	GET			200	number[]
		GET	id: string		200, 404	StreamableFile
		POST		radius: string, files: GameImageInput	200, 400	Promise<ImageComparisonResult>

* fonctionnalités implémentées au sprint 3

Tableau 3.2.1 - Événements WebSocket

Dossier	Source	Nom d'évènement	Contenu	Contenu du retour
Matchmaking	Client	startMatchMaking	gameId: string	
		someOneWaiting	gameId: string	response: boolean
		roomCreatedForThisGame	gameId: string	response: boolean
		leaveWaitingRoom	gameId: string	
		joinRoom	{ gameId: string, playerName: string }	
		acceptOpponent	opponentName: string	response: boolean
		rejectOpponent	{ gameId: string, playerName: string }	
	Serveur	opponentJoined	opponentName: string	
		opponentLeft		
		acceptOtherPlayer	opponentName: string	
		rejectOtherPlayer	playerName: string	
		roomReachable		
		sessionId	sessionId: number	
		updateRoomView		
Session	Client	getClientId		id: string
		closeSession	sessionId: number	sessionId: number
		startSession	data: StartSessionData	(sessionId: number) ou undefined
		leaveRoom		
		submitCoordinate	data: [sessionId: number, coordinates: Coordinates]	(response: GuessResult) ou undefined
		cheatGetAllDifferences	sessionId: number	(response: Coordinates[][]) ou undefined
		playerLeft	sessionId: number	
		playerName	clientName: string	
		askForClue*	sessionId: number	response: Clue
		askForReplay*	sessionId: number	response: ReplayData

* fonctionnalités implémentées au sprint 3

Tableau 3.2.2 - Événements WebSocket (suite du tableau 3.2.1)

Sujet	Source	Nom d'évènement	Contenu	Contenu du retour
Session	Serveur	playerWon	winnerInfo: WinnerInfo	
		differenceFound	differenceFound: GuessResult	
		opponentLeftGame		
		provideName		
		timerUpdate	time: string	
		newLeaderNotification*	newLeaderBoard: LeaderBoardUpdate	
Chat	Client	giveName	playerName: string	
		messageFromClient	message: Message	
	Serveur	systemMessageFromServer	systemMessage: SystemMessage	
		messageFromServer	newMessage: Message	
		giveClientID	receivedId: string	

* fonctionnalités implémentées au sprint 3