Équipe 208

Protocole de communication

Version 2.0

Historique des révisions

Date	Version	Description	Auteur
2023-04-20	1.2	Paquet HTTP + Paquet WS	Hamza et Lounes
2023-04-20	1.2	Communication client-serveur	Melvis
2023-04-20	1.2	Communication client-serveur	Yanis
2023-04-20	2.0	Rédaction de l'introduction, et description des paquets HTTP/WebSocket	Daniel
2023-04-20	2.2	Paquet WS, description des paquets HTTP/WebSocket	Renel

Table des matières

1. Introduction

Dans ce rapport de protocole de communication, nous allons nous pencher sur deux aspects clés du projet. Tout d'abord, nous allons examiner les différents protocoles de communication qui sont utilisés entre les clients et le serveur pour des fonctionnalités spécifiques. Ensuite, nous allons nous intéresser au contenu des différents types de paquets qui sont utilisés dans vos protocoles de communication. Ces deux aspects sont essentiels pour comprendre comment les clients et serveurs communiquent entre eux.

2. Communication client-serveur

Dans cette section du rapport, nous présenterons brièvement les différentes raisons pour lesquelles les protocoles de communication utilisés dans le projet sont nécessaires. De plus, WebSocket a plus été utilisé que HTTP lors de l'implémentation des fonctionnalités dans le cadre du projet.

2.1. WebSocket

- WebSocket est utilisé pour la messagerie afin que les joueurs puissent s'échanger des messages et aussi pour les messages d'événements local et global (chat.gateway), il permet une communication beaucoup plus rapide que HTTP et admet une communication bidirectionnelle entre le serveur et le client, ce qui facilite le partage de données.

- WebSocket est utilisé pour le timer afin d'afficher le temps qu'une partie s'est écoulée ou bien sous forme de compte à rebours pour le mode Temps Limité.

- WebSocket est utilisé à la fois pour les parties de jeu en mode solo et un contre un, car nous exploitons les rooms de SocketIO pour placer les joueurs dans une partie de jeu donnée. Ainsi, nous pouvons aisément transmettre de l'information uniquement aux joueurs présents dans la room, si souhaité, comme les différences trouvées par l'adversaire, les messages des joueurs, etc.

- WebSocket est utilisé pour mettre en attente les joueurs qui souhaitent créer/rejoindre un jeu.

- WebSocket est utilisé pour enregistrer et reprendre les actions qui ont eu lieu durant la partie pour la reprise vidéo de la partie

- WebSocket est utilisé pour les fiches de jeu (game-card-handler.gateway)

2.2. HTTP

- HTTP est utilisé pour le chargement de l'historique des parties jouées.

- HTTP est utilisé pour la logique du jeu (game.controller), surtout dans le but de charger les jeux de la partie et dans la création de ceux-ci. Par ailleurs, ceci nous permettra d'avoir accès à la liste des différences dans les jeux créés et donc d'implémenter notre logique de détection et validation de différences à partir de WebSocket.

- HTTP est utilisé pour sauvegarder le meilleur temps des parties de jeu et réinitialiser les meilleurs temps d'un jeu (game-record.controller), avec l'aide de notre base de données MongoDB.

- HTTP est utilisé pour gérer l'obtention des indices des parties en solo pour les modes de jeux Classique et Temps Limité (game-record.controller).

- HTTP est utilisé pour configurer les constantes de jeu qui sont hébergées sur notre base de données MongoDB.

3. Description des paquets http et WebSocket

Dans cette section, vous trouverez une présentation des différents types de paquets utilisés dans les protocoles de communication. Pour HTTP, un tableau est fourni pour chaque type de requête HTTP, qui inclut la méthode, l'endpoint, le body, le content-type, le code de retour et les informations de réponse. Pour les bodies qui ne contiennent rien, des " {} " sont utilisés. Et enfin, pour WS, un tableau est fourni pour chaque événement, qui inclut le nom de l'événement, sa source, sa destination, son contenu et le type d'événement.

3.1. Details des requêtes http vers le « Endpoint » (/api/gameRecord/)

Méthode	Endpoint	Params	Body	Content-type	Code	Réponse
GET	/api/gameRecord		{}	application/json; charset=utf-8	200	[Array d'object GameRecord incluant le nom des jeux et les records – temps et noms de joueurs]
GET	/api/gameRecord		{}	application/json; charset=utf-8	404	NOT_FOUND
DELETE	/api/gameRecord		{}	application/json; charset=utf-8	200	Suppression réussie des enregistrements de tous les jeux
DELETE	/api/gameRecord/		{}	application/json; charset=utf-8	204	NO_CONTENT
POST	/api/gameRecord/create		{ "gameName": string, "typeGame": string, "time": string, "playerName": string, "dateStart": string, "keyServer": string" }	application/json; charset=utf-8	200	
POST	/api/gameRecord/create		{}	application/json; charset=utf-8	404	NOT_FOUND
DELETE	/api/gameRecord/{gameName}	gameName	{}	application/json; charset=utf-8	200	Le jeu a été supprimé avec succès
DELETE	/api/gameRecord/{gameName}	gameName	{}	application/json; charset=utf-8	204	NO_CONTENT

3.2.1. Details des requêtes http vers le « Endpoint » (/api/game/)

Méthode	Endpoint	Params	Body	Content-type	Code	Réponse
GET	/api/game		{}	application/json; charset=utf-8	200	[Array d'object Game incluant le nom des jeux et les deux objets images]
GET	/api/game		{}	application/json; charset=utf-8	404	NOT_FOUND
DELETE	/api/game		{}	application/json; charset=utf-8	200	Suppression réussie de tous les jeux
DELETE	/api/game		{}	application/json; charset=utf-8	400	BAD_REQUEST
GET	/api/game/constant		{}	application/json; charset=utf-8	200	Objet de constantes incluant le temps initial, le temps de pénalité et le temps de bonus
GET	/api/game/constant		{}	application/json; charset=utf-8	404	NOT_FOUND

PUT	/api/game/constant		{
			"timeInit": number,
			"timePen": number,
			"timeBonus": number,
			}
				application/json; charset=utf-8	200	

3.2.2. Details des requêtes http vers le « Endpoint » (/api/game/)

Méthode	Endpoint	Params	Body	Content-type	Code	Réponse

PUT	/api/game/constant		{
			"timeInit": number,
			"timePen": number,
			"timeBonus": number,
			}
				application/json; charset=utf-8	404	NOT_FOUND

GET	/api/game/count		{}	application/json; charset=utf-8	200	Count du jeu
GET	/api/game/count		{}	application/json; charset=utf-8	404	NOT_FOUND
GET	/api/game/validate/{name}	name	{}	application/json; charset=utf-8	404	NOT_FOUND
GET	/api/game/validate/{name}	name	{}	application/json; charset=utf-8	200	Le nom du jeu a été validé avec succès
GET	/api/game/{id}	id	{}	application/json; charset=utf-8	200	Le jeu avec ses attributs
GET	/api/game/{id}	id	{}	application/json; charset=utf-8	404	NOT_FOUND
DELETE	/api/game/{id}	id	{}	application/json; charset=utf-8	200	Jeu a été supprimé avec succès
DELETE	/api/game/{id}	id	{}	application/json; charset=utf-8	204	NO_CONTENT
POST	/api/game/create		{}	application/json; charset=utf-8	200	Le jeu a été ajouté avec succès
POST	/api/game/create		{}	application/json; charset=utf-8	409	CONFLICT

3.3. Details des requêtes http vers le « Endpoint » (/api/gamingHistory)

Méthode	Endpoint	Params	Body	Content-type	Code	Réponse
GET	/api/gamingHistory		{}	application/json; charset=utf-8	200	[Array d'object GamingHistory incluant le nom du jeu, le nom du ou des joueur, le mode de jeu, la durée du jeu et la date et heure du début du jeu]
GET	/api/gamingHistory		{}	application/json; charset=utf-8	404	NOT_FOUND
DELETE	/api/gamingHistory		{}	application/json; charset=utf-8	200	Suppression réussie des historiques de tous les jeux
DELETE	/api/gamingHistory		{}	application/json; charset=utf-8	204	NO_CONTENT
POST	/api/gamingHistory		{}	application/json; charset=utf-8	200	Créer avec succès
POST	/api/gamingHistory		{}	application/json; charset=utf-8	409	CONFLICT

3.4.1. Details du protocole WebSocket

Nom de l'événement	Source	Destination	Contenu	Type d'événement
serverTime	Server	Client	Un array du temps écoulé de tous les jeux en cours	Broadcast à tous les clients
hello	Server	Client	id du joueur en string	Broadcast à tous les clients
sendRoomName	Server	Client	Un array contenant le mode de jeu classique et le nom de la room	Broadcast à tout le monde
nbrDifference	Server	Client	Un number indiquant le nombre de jeu	Broadcast au client de la room
nbrDiffLeft	Server	Client	Le nombre du jeu de la room	Broadcast au client de la room
diffFound	Server	Client	Un array de set de number des différences non trouvées	Broadcast à tout le monde
findDifference-return	Server	Client	Un objet contenant le nom de joueur	Broadcast tau client de la room
error	Server	Client		Broadcast à tout le monde
feedbackDifference	Server	Client	Un objet contenant le message	Broadcast au client de la room
giveup-return	Server	Client	Un objet contenant le nom du joueur qui a abandonné	Broadcast au client de la room
gameEnded	Server	Client	Un array contenant un boolean true et le nom du joueur	Broadcast au client de la room
teamateDisconnected	Server	Client	Un boolean true	Broadcast au client de la room
timeLimitStatus	Server	Client	Un boolean false	Broadcast au client de la room

3.4.2. Details du protocole WebSocket

Nom de l'événement	Source	Destination	Contenu	Type d'événement
getRandomGame	Server	Client	Un objet game incluant le nom du jeu, le niveau de difficulté, la liste des différences et le contenu des images originaux et modifiés	Broadcast au client de la room
Message-return	Serveur	Client	Un objet contenant le message et le nom du joueur	Broadcast à tout le monde sauf le client qui l'a déclenché
UpdateStatus	Serveur	Client	Un array d'objet incluant le nom du jeu et le statut du jeu qui peut être 0 ou 1 s'il y a un joueur en attente	Broadcast à tout le monde
globalEvent	Serveur	Client	1. Objet réponse 2. Objet réponse incluant le sous type d'événement qui peut être : - feedbackOnJoin - feedbackOnLeave - feedbackOnStart - feedbackOnAccept - feedbackOnWait - feedbackOnWaitLonger - gameUnavailable - feedbackOnReject - et les objects correspondant : - le nom de l'opposant - un message indiquant qui a quitté - un objet indiquant l'id du jeu, le nom du jeu, le nom du créateur, le nom de l'opposant et le mode de jeu - le nom de l'opposant	Broadcast au client de la room

			- le nom du créateur - le nom du créateur - un message indiquant que le jeu a été supprimé par l'administrateur - un message indiquant que le joueur a été refusé par le créateur	