Équipe 209

Protocole de communication

Version 1.3

Historique des révisions

Date	Version	Description	Auteur
2023-11-02	1.0	Compléter les interfaces (3.3) et les paquets HTTP (3.1).	Melliz Medina, Vivian
2023-11-03	1.1	Compléter les paquets WebSockets (3.2).	Melliz Medina, Vivian
2023-11-07	1.2	Ajout de la partie 2. Communication client-serveur	Thoelen, Nathan
2023-11-07	1.3	Bonification de la partie 2.	Nadeau, Melody

Table des matières

Protocole de communication

1. Introduction

Pour faciliter la communication entre client-serveur, l'utilisation de protocoles de communication est nécessaire. Dans le contexte de notre projet, nous avons utilisé les protocoles HTTP et WebSocket. Le présent document vise à expliquer les raisons de ce choix ainsi que les avantages de chacun des protocoles. Également, il propose une description des paquets échangés au cours de la communication.

2. Communication client-serveur

Au sein de ce projet, la communication entre le client et le serveur fonctionne autour de deux principaux protocoles : HTTP et WebSockets. L'utilisation du protocole HTTP est relativement restreinte, étant réservée aux échanges d'informations ponctuels nécessitant des requêtes individuelles entre le client et le serveur. Les WebSockets, quant à eux, prennent une place importante pour les interactions, demandant une transmission instantanée de données, dès qu'elles deviennent disponibles.

Dans ce projet, les WebSockets sont employés pour la gestion des joueurs, de l'organisateur, ainsi que pour la zone de discussion au sein des parties impliquant plusieurs participants. Cette approche assure une communication instantanée entre le client et le serveur, garantissant une réactivité optimale et une interaction directe entre les deux entités. Des exemples de l'utilisation des WebSockets pour la gestion des joueurs et de l'organisateur sont la mise à jour du score de chacun et le retrait de ceux ayant quitté la partie. L'utilisation des WebSockets se justifie également dans le contexte des salles d'attente, permettant d'ajouter les joueurs dans des groupes et de les placer dans des espaces d'attente différents pour chaque partie. Ainsi, les participants peuvent visualiser la présence d'autres joueurs dans la salle d'attente, une information qui est diffusée en temps réel par le serveur, sans nécessiter de requête explicite.

Divers autres aspects du projet exploitent le protocole WebSocket, tels que l'envoi de questions et de chronomètres du serveur vers le client, la synchronisation de l'écran de jeu, nécessitant une communication continue entre le serveur et le client, la réinitialisation des chronomètres de jeu et l'affichage en temps réel des joueurs avec leurs scores. En revanche, le protocole HTTP est principalement réservé aux opérations liées à la base de données, notamment la récupération complète des parties, de l'historique de jeu, la recherche d'un jeu par son identifiant et la création d'un nouveau jeu. De plus, il est utilisé pour des opérations telles que la suppression d'une partie en fonction de son identifiant, et la modification d'un jeu.

3. Description des paquets

3.1 Paquets HTTP

Méthode	URI	Paramètres URI	Description	Corps de la Requête	Corps de la Réponse	Code(s) de retour
GET	*/:id*	id: string	Récupérer les informations d'un jeu par son id.	—	Succès: *Game* Échec: error.message	Succès: 200 Échec: 404
GET	/	—	Récupérer les informations de tous les jeux.	—	Succès: *Game[]* Échec: error.message	Succès: 200 Échec: 404
POST	/	—	Ajouter un nouveau jeu.	*Game*	Succès: — Échec: error.message	Succès: 201 Échec: 400
DELETE	/:id	id: string	Supprimer un jeu par son id.	—	Succès: — Échec: error	Succès: 200 Échec: 404
PATCH	/:id	id: string	Modifier un jeu par son id.	*Game*	Succès: — Échec: error.message	Succès: 200 Échec: 404
GET	/	—	Récupérer tous les mots de passe.	—	Succès: Password[] Échec: error.message	Succès: 200 Échec: 404

POST	/:file	file: Express.Multer.File	Importer un fichier .JSON.	Game	Succès: *Game* Échec: error.message	Succès: 201 Échec: 400
POST	/	—	Ajouter une partie dans l'historique de partie	Play	Succès: — Échec: error.message	Succès: 201 Échec: 400
GET	/:id	id: string	Récupérer toutes les parties jouées du même code d'access.	—	Succès: Play[] Échec: error.message	Succès: 200 Échec: 404

3.2 Paquets WebSockets

Événement	**Source**	**Description**	**Données**	**Événements Potentiellement déclenchés**
validate	Client	Demander une validation du nouveau message d'un client.	*string*	wordValidated
wordValidated	Serveur	Envoyer la validation du nouveau message d'un client.	*boolean*	roomMessage
roomMessage	Client	Notifier d'un nouveau message dans le chat de la salle destinée.	*string*	roomMessage
roomMessage	Serveur	Envoyer un nouveau message dans le chat de la salle destinée.	*Message*	—
createPlay	Client	Notifier la création d'un jeu et demander un code d'accès de la salle créée.	*Game*	getAccessCode
getAccessCode	Serveur	Donner le code d'accès de la salle créée.	*string*	—
toggleLock	Client	Demander une modification du verrouillage à bascule de la partie du jeu.	—	—

validateAccessCode	Client	Demander une validation du code d'accès et du nom de l'utilisateur d'un client pour joindre une partie d'un jeu.	string[]	sameNameError playerJoined getPlayersNames playLocked bannedPlayer accessCodeInvalid
sameNameError	Serveur	Notifier une erreur dans le nom de l'utilisateur inséré qui est que le nom est déjà utilisé par un autre utilisateur.	—	—
playerJoined	Serveur	Envoyer la validation du code d'accès et du nom de l'utilisateur pour joindre une partie d'un jeu.	—	—
getPlayers	Serveur	Donner la liste des joueurs de la salle créée.	*Player[]*	—
getPlayersNames	Serveur	Donner la liste des noms des joueurs de la salle créée.	*string[]*	—
playLocked	Serveur	Notifier le verrouillage de la salle de jeu à joindre.	—	—
bannedPlayer	Serveur	Notifier l'interdiction d'accès à un joueur banni.	—	—
accessCodeInvalid	Serveur	Notifier un code d'accès invalide.	—	—
removePlayer	Client	Demander d'enlèver un joueur qui est entrée dans la page d'attente des joueurs.	*string*	kickPlayer getPlayers getBannedPlayers
kickPlayer	Serveur	Notifier au joueur qu'il a été banni de la partie du jeu.	—	—
getBannedPlayers	Serveur	Donner la liste des joueurs bannis de la salle créée.	*string[]*	—
startPlay	Client	Demander de commencer la partie du jeu.	—	countdown countdownOrganiser getGame getPlayers questionEnd startError
countdown	Serveur	Notifier le compte à rebours avant de commencer la partie du jeu aux joueurs.	number	—
getGame	Serveur	Donner les informations du jeu de la partie du jeu.	*Game*	—
startError	Serveur	Notifier l'interdiction de commencer la partie du jeu dû au verrouillage du jeu ou à l'absence de joueurs dans la partie du jeu.	—	—
nextQuestion	Client	Demander de passer à la prochaine question du jeu.	—	IntervalOfThreeSeconds

				tIntervalOfThreeSeconds Organiser startNextQuestion getTime questionEnd
startNextQuestion	Serveur	Notifier le début de la prochaine question du jeu.	—	updateScore nextHistogramData
updateScore	Client	Demander de mettre à jour le nombre de points des joueurs.	number	getPlayers
nextHistogramData	Client	Demander les informations de la prochaine question à l'histogramme.	number	getSelectedAnswers
getSelectedAnswers	Serveur	Donner la liste des réponses sélectionnées durant la partie du joueur.	Answer[]	—
getTime	Serveur	Donner la durée de la question présente du jeu.	number	—
questionEnd	Serveur	Demander si c'est la dernière question.	—	updateScore
quitPlay	Client	Notifier que le joueur est parti de la salle d'attente du jeu.	—	getPlayersNames
quitPlayerFromList	Client	Notifier que le joueur est parti du jeu durant la partie du jeu.	—	getPlayers
getSelectedOptions	Client	Donner les choix sélectionnés durant la partie du joueur.	string[]	getSelectedAnswers
deletePlay	Client	Demander de supprimer la partie du jeu.	—	playDelected
playdeleted	Serveur	Notifier que la partie du jeu a été supprimé.	—	—
endPlay	Client	Demander de terminer la partie du jeu.	—	playEnded showResults
playEnded	Serveur	Notifier que la partie du jeu est terminé.	—	—
showResults	Serveur	Donner les résultats de la partie du jeu en liste des joueurs.	Player[]	—
goodAnswersConfirmed	Client	Notifier la confirmation de la réponse correcte ou des réponses correctes du joueur.	—	bonusReceived
bonusReceived	Serveur	Notifier la réception d'un bonus au joueur.	—	—
badAnswersConfirmed	Client	Notifier la confirmation de la réponse incorrecte ou des réponses incorrectes du joueur.	—	—
setIntervalOfThreeSeconds	Serveur	Notifier le compte à rebours avant de commencer la prochaine question aux joueurs.	number	—

etIntervalOfThreeSecondsOrganiser	Serveur	Notifier le compte à rebours avant de commencer la prochaine queastion a l'organisateur.	number	—
pauseTimer	Client	L'organisateur met le chronometre en pause	—	—
replayTimer	Client	L'organisateur met le chronometre en route	—	—
panicTimer	Serveur	Le serveur met le timer en mode panic	number	playSound
playSound	Serveur	Le serveur envoi un signal pour faire jouer un son	—	—

3.3 Interfaces

Nom	Description	Structure

Answer	Informations sur la réponse des questions pour l'histogramme de la vue de jeu de l'organisateur.	{
		choice: Choice,
		count: number,
		}

Choice	Informations sur l'option d'une question.	{
		text: string,
		isCorrect: boolean,
		}

Game	Informations du jeu.	{
		id: string,
		title: string,
		description: string,
		duration: number,
		lastModification: Date,
		isVisible: boolean,
		questions: Question[],
		filename?: string
		}

Message	Informations d'un message du chat.	{
		sender: string,
		text: string,
		time: Date,
		}

Password	Information du mot de passe pour l'administrateur.	{
		password: string;
		}

Play	Informations du jeu.	{
		accessCode: string,
		isUnlocked: boolean,
		game: Game,
		PlayersList: Player[],
		currentAnswers: Answer[],
		bannedNames: string[],
		correctAnswersPlayers: string[]
		}

Player	Informations d’un joueur d’un jeu.	{
		id: string,
		name: string,
		points: number,
		isInPlay: boolean,
		numberOfBonuses: number,
		isOrganiser: boolean,
		isAnswerConfirmed: boolean,
		answers: string[]
		}

Question	Informations d’une question d’un jeu.	{
		type: string,
		text: string,
		points: number,
		choices: Choice[]
		}