Équipe 101

Protocole de communication

Version 1.3

Historique des révisions

Date	Version	Description	Auteur
2023-10-30	1.0	Section 1 et 2 50%	Mathieu B.
2023-10-31	1.1	Section 2 100%	Mathieu B.
2023-11-01	1.2	Section 3	Mathieu B.
2023-11-04	1.3	Adjustments from feedback	Mathieu B.

Table des matières

Protocole de communication

1. Introduction

Ce document présente le protocole de communication qui sera complètement implémenté dans notre plateforme de jeu interactive à la fin de ce cours. Nous aborderons la communication client-serveur, en se concentrant sur l'utilisation d'HTTP pour gérer les jeux-questionnaires et les matchs. Les acteurs étant les joueurs, les administrateurs et les organisateurs, nous nous baserons sur ces rôles pour donner les exemples concrets des cas d'utilisation. Nous explorerons également le rôle du protocole WebSocket dans la communication en temps réel entre les joueurs, l'organisateur et eux-mêmes. Puis, nous décrirons les interfaces utilisées pour transmettre ces informations essentielles. Ce document inclus des opérations, des flux de données et des structures sous-jacentes de notre système, servant de référence pour les développeurs.

2. Communication client-serveur

Le protocole HTTP est utilisé pour le chargement des jeux-questionnaires lors de leur présentation dans la page d'administration à l'aide d'une requête GET de tous les jeux sur la BD. Ainsi, les noms des jeux et leur de date de dernière modification est affichée pour que l'administrateur ait une meilleure idée de ce qu'il souhaite faire, puis en modifiant le jeu-questionnaire, on a une page avec tous les attributs de celui-ci, et une manière de les modifier. Et donc, à l'initialisation de la page, une requête HTTP est faite pour obtenir le jeu avec son id. Une requête PATCH est faite lorsque qu'un jeu-questionnaire existant est modifié et une requête POST sera faite lorsqu'un jeu non existant est créé et sauvegardé. Finalement, un jeu de la liste supprimé veut dire une requête DELETE pour l'objet jeu.

Ensuite, dans la page de création de partie, une requête POST est faite pour les matchs à sa création, et une requête PATCH est ensuite faite quand la partie est verrouillée ce qui permet à l'organisateur de commencer la partie, contenant tous les joueurs s'étant ajoutés. Si le match n'est pas complété, une requête DELETE devra être faite.

Chaque joueur initialisé au début du match est ajouté dans une salle, c'est de là que leurs actions comme le clavardage, leurs réponses aux questions, leur placement dans la liste des joueurs et leurs statuts seront communiquées à l'aide du protocole WebSocket. L'interaction avec le compte à rebours (mode pause et panique), le ban du chat et la correction de question QRL seront des capacités de l'organisateur à utiliser le WebSocket dans la salle. L'organisateur doit être le premier à s'être joint dans la salle.

Le chargement de l'historique des matchs aura besoin d'une requête HTTP, de type GET, ainsi que la fonction de testage du jeu. La différence est l'utilisation d'un id spécifique pour tester un jeu en particulier, et d'un GET pour tous les matchs pour la liste (dont la visibilité est activée) de l'historique.

3. Description des paquets

3.1 Paquets HTTP

Méthode	**URI**	**Paramètres URI**	**Description**	**Corps de la Requête**	**orps de la Réponse**	**ode(s) de retour**
GET	/editgame	—	écupère tous les jeux-uestionnaires sur la DB	—	Game[]	ıccès: 200 chec: 404
GET	/editgame/:*id*	id: string	Récupère une game (jeu-uestionnaire) par son id	—	*Game*	ıccès: 200 chec: 404

POST	/editgame	—	Envoi un jeu-questionnaire sur la DB	{ "id": "string", "title": "string", "isVisible": true, "lastModification": "string", "duration": 0, description": "string", "questions": ["string"] }	—	uccès: 201 chec: 404
PATCH	/editgame/:id	id: string	Modification d'un jeu-questionnaire	{ "id": "string", "title": "string", "isVisible": true, "lastModification": "string", "duration": 0, description": "string", "questions": ["string"] }	—	uccès: 201 chec: 404
DELETE	/editgame/:id	id: string	Suppression d'un jeu-questionnaire	—	—	uccès: 200 chec: 404
GET	/waitingplayers	—	Récupère tous s matchs de la B (peut ajouter condition de visibilité)	—	Match[]	uccès: 200 chec: 404
GET	aitingplayers/:acc essCode	accessCode:string	Récupère un tch spécifique n fonction de l'id.	—	Match	uccès: 200 chec: 404

POST	/waitingplayers	—	Envoi d'un nouveau match sur la DB	{ accessCode: string; canBeAccessed: boolean; game: Game; players: Player[]; time: number; questionId: string; messages: string[]; creator: string; nomsBannis: string[]; }	—	Succès: 201 Echec: 404
PATCH	/waitingplayers/ :accessCode	accessCode:string	Modification d'un match existant sur la DB	{ accessCode: string; canBeAccessed: boolean; game: Game; players: Player[]; time: number; questionId: string; messages: string[]; creator: string; nomsBannis: string[]; }	—	Succès: 201 Echec: 404
DELETE	/waitingplayers/ : accessCode	accessCode:string	Suppression d'un match de la DB	—	—	Succès: 200 Echec: 404
GET	/history	—	Récupère tous les historiques de partie créé en parallèle avec les matchs.	—	Historic[]	Succès: 200 Echec: 404
POST	/history	—	Ajoute un historique de partie à la liste	{ gameName: string; playDate: Date; players: number; bestPoints : number id : string }	—	Succès: 201 Echec: 404

GET	/history/ :id	id : string	Récupère un historique en particulier pour un match.	—	Historic	uccès: 200 chec: 404
DELETE	//history/ :id	id : string	Suppression historique en particulier pour un match.	—	—	uccès: 200 chec: 404

3.2 Paquets WebSockets

Événement	Source	Description	Données	Événements Potentiellement déclenchés
ndChronoValues	Client	nvoi de la valeur chrono vers tous les joueurs	*TimeDTO*	—
setReady	Client	nvoi un signal si le joueur est prêt au début du match	*ReadyDTO*	playerReady
setPanic	Client	Commence le mode panique	*TimeDTO*	panic
onPanic	Server	Envoit (toggle) du mode anique pour tous les joueurs	*TimeDTO*	togglePanic
goPanic	Server	Reçoit et met en place l'événement de panique ou l'arrête	*TimeDTO*	togglePanic
ceiveChronoValues	Server	Reçoit le chrono du client rsqu'un message est envoyé	*TimeDTO*	chronoReception
sendChat	Client	Envoi du message sur le component de clavardage	*MessageDTO*	chat
receiveChat	Server	Reçoit le message du client	*MessageDTO*	chat

joinRoom	Client	Client se join à une salle	*match.id*	joinRoom
onJoinRoom	Server	Le server reçoit la requête et met le client dans la salle	*client.id*	ServerRoom
andleConnection	Server	Augmente le nombre de clients sur la salle en les connectant	*users*	users
andleDisconnect	Server	Réduit le nombre de clients sur la salle en les déconnectant	*users*	users
orgQuitGame	Client	Les joueurs se font donner l'ordre de quitter	*match.accessCode*	orgQuitGame
ndToResultView	Client	Donne l'ordre de changer de page pour tous ceux dans la salle	*match.accessCode*	resultView
oToResultView	Server	Reçoit le signal de changer de page	*match.accessCode*	goToResultView
onQuitGame	Server	L'ordre de quitter est envoyé à la salle	*match.accessCode*	rderedToQuitTheGame
deredToQuitTheGame	Server	Reçoit le signal de quitter la partie	*match.accessCode*	rderedToQuitTheGame
nNextQuestion	Server	Envoit le signal de changer de question	*question.id*	goToNextQuestion
rgNextQuestion	Client	Demande de changer de question pour la salle	*question.id*	nextQuestion
yerNextQuestion	Server	Reçoit la nouvelle question	*question.id*	goToNextQuestion
skValidateQuestions	Server	Envoit la validation des questions à la salle	*SelectionDTO OU AnswerDTO*	validateQuestions
gValidateQuestions	Server	Reçoit la validation d'une question pour chaque joueur selon la bonne réponse du jeu-questionnaire ou selon l'organisateur	*SelectionDTO OU AnswerDTO*	validateQuestions
banChatter	Client	Fait demande de ban du chat pour un joueur	*BanDTO*	ban

onBanChatter	Server	Envoit une demande de ban du chat au joueur	*BanDTO*	banned
chatterBanned	Server	Joueur reçoit le ban	*BanDTO*	banned
setStatus	Client	Joueur update son statut en achevant une action (type, confirme question, déconnecte, etc.)	*StatusDTO*	status
OnStatusUpdate	Server	Envoit le statut mis à jour du joueur pour la question	*StatusDTO*	updatedStatus
getStatus	Server	Reçoit les statuts de tous les joueurs	*StatusDTO*	updatedStatus
setPoints	Client	Joueur accumule des points à la fin de la question	*PointsDTO*	points
OnPointsUpdate	Server	nvoit les points mis à jour du joueur pour la question	*PointsDTO*	updatedPoints
getPoints	Server	Reçoit les points de tous les ueurs et la liste est remise en ordre	*PointsDTO*	updatedPoints

3.3 Interfaces

Nom	Description	Structure

Player	Information sur un joueur individuel	Player {
		points: number;
		status: string;
		name: string;
		selection: number[];
		}

Match	formations sur un match passé ou présent	{
		accessCode: string;
		canBeAccessed: boolean;
		game: Game;
		players: Player[];
		time: number;
		questionId: string;
		messages: string[];
		creator: string;
		nomsBannis: string[];
		}

Choice	nformation sur le choix d'une question QCM	{
		text: string;
		isCorrect: boolean;
		}

Question	formation sur la question QCM ou QRL d'une jeu-questionnaire	{
		id: string;
		type: string;
		text: string;
		points: number;
		choices: Choice[];
		answer: string;
		}

Game	Information sur le jeu-questionnaire	{
		id: string;
		title: string;
		isVisible: boolean;
		lastModification: Date;
		duration: number;
		description: string;
		questions: Question[];
		}

Historic	Information sur les parties jouées	{
		gameName: string;
		playDate: Date;
		players: number;
		bestPoints : number;
		id: string;
		}

ReadyDTO	Signale qu'un joueur est prêt	{
		name: string,
		}

PlayerListDTO	formations sur une mise à jour des choix un joueur, sur le statut du joueur envoyée et son score, mis à jour au courant du match.	{
		name: string;
		role: string;
		socketId: string;
		isFirstAmount: number;
		score: number;
		selection: number[];
		status: string;
		}

AnswerDTO	Note pour une question particulière et un joueur particulier	{
		name: string,
		grade: number,
		}

TimeDTO	Information sur la mise à jour du temps	{
		matchid: string,
		time: number,
		}

MessageDTO	Information sur le message entré dans le clavardage	{
		name: string,
		text: string,
		time : Date,
		}

BanDTO	Information sur le ban d'un joueur par l'organisateur	{
		name: string,
		}