210

Protocole de communication

Version 2.0

Historique des révisions

Date	Version	Description	Auteur
2023-03-09	0.1	Disponibilité initiale	Jean-Christophe Pelletier
2023-03-18	0.2	Ajout majeur de contenu	Benjamin Dancause
2023-03-20	0.3	Ajout majeur de contenu	Jean-Christophe Pelletier
2023-03-02	0.4	Ajout de la section WebSocket au complet	Jérémy Langevin
2023-04-20	2.0		

Table des matières

Protocole de communication

1. Introduction

Ce document vise à fournir une description détaillée du protocole de communication qui sera utilisé entre les clients et le serveur. Une description complète de toutes les fonctionnalités du protocole de communication, y compris les messages, les en-têtes et les données qui seront échangées sera présente dans ce document.

2. Communication client-serveur

Il fut décidé que par défaut, le protocole de communication utilisé est HTTP lorsqu'il n'est pas nécessaire au serveur d'envoyer plus d'information qu'une réponse à une requête.

Le suivant tableau indique les fonctionnalités et le protocole qu'ils utilisent

Protocole	Fonctionnalité	Justification
HTTP	Peupler le carrousel de cartes de jeu	défaut
HTTP	Enregistrement d'une nouvelle partie crée	défaut
HTTP	Détection d'erreur à la position du clique	défaut
HTTP	Recevoir les informations d'une partie	défaut
HTTP	Mode triche	défaut
HTTP	Créer une nouvelle partie	défaut
HTTP	Effacer une partie	défaut

3. Description des paquets

Pour cette section, les adresses seront écrites avec l'omission de l'adresse du serveur pour des raisons de concision et de lisibilité. Ainsi, GET /X équivaudrait à GET adresse_serveur/api/X.

3.1 Protocole HTTP

3.1.1 Peupler le carrousel de cartes de jeu

GET /games/all

Paramètres: aucun

Corps: aucun

Réponse: Code 200. La réponse contient une liste de toutes les cartes de jeu. Les données des cartes sont décrites par des objets avec les caractéristiques suivantes:

Clé	Type de la valeur	Description
name	string	nom de la partie
image	string	chemin de l'image
difficulty	boolean	si la partie est difficile

3.1.2. Enregistrement d'une nouvelle partie

POST /games/images

Paramètres: aucun

Corps: Le contenu du corps est un objets avec les caractéristique suivantes:

Clé	Type de la valeur	Description
name	string	nom de la partie
originalIamge	string	image original en base64
modifiableImage	string	image modifié en base64
difficulty	boolean	si la partie est difficile
count	number	numéro de la différence
differences	Coords[][]	coordonnée du pixel

Où l'objets Coords est:

Clé	Type de la valeur	Description
x	number	position horizontal du pixel
y	number	position vertical du pixel

Réponse: Code 201. Aucune valeur dans le corps de la réponse

GET /games/names

Paramètre: aucun

Corps: aucun

Réponse: Code 200. La réponse contient un liste de valeur de type string, il s'agit de tous les noms de partie.

3.1.3. Recevoir les informations d'une partie

POST /games/gameByName

Paramètre: aucun

Corps: Le contenu du corps est un objet avec comme seul attribut "name" une string qui représente le nom de la partie.

Réponse: Code 200. La réponse est un objets avec les caractéristique suivantes:

Clé	Type de valeur	Description
name	string	nom de la partie
images	string[]	deux images en base64
count	number	nombre de différence
difficulty	boolean	si la partie est difficile

POST /gaming/diffAmount

Paramètre: aucun

Corps: Le contenu du corps est un objet avec comme seul attribut "name" une string qui représente le nom de la partie.

Réponse: Code 200. La réponse contient un nombre correspondant au nombre de différences dans la partie.

GET /games/${name}

Paramètre: Le paramètre name est une string qui correspond à un nom de partie

Corps: Aucun

Réponse: Code 200. La réponse contient un boolean qui indique si la partie est disponible.

3.1.4. Mode triche

POST /gaming/findAll

Paramètre: aucun

Corps: Le contenu du corps est un objet avec comme seul attribut "name" une string qui représente le nom de la partie.

Réponse: Code 200. La réponse est un objets avec les caractéristique suivantes:

Clé	Type de valeur	Description
id	number	l'identifiant de la partie
count	number	nombre de difference
differences	Coords	positions des pixels de

		différence

3.1.5 Créer une nouvelle partie

POST /games/images

Paramètre: Aucun

Corps: Le contenu du corps est un objet avec les caractéristique suivantes:

Clé	Type de la valeur	Description
name	string	Nom de la partie
originalImage	string	Image original en base 64
modifiableImage	string	Image modifié en base 64
difficulty	boolean	Difficulté de la nouvelle partie
count	number	Nombre de différences
differences	Coords[][]	Arrays avec les positions x et y de toutes les différences

3.1.6 Effacer une partie

DELETE /games/${name}

Paramètre: Le paramètre name est une string qui indique la partie à effacer.

Corps: Aucun

Réponse: Code 200. Le corps de la réponse est vide.

POST /games/delete/games

Paramètre: aucun

Corps: Aucun

Réponse: Code 200.

3.1.7 Manipulation des Meilleurs temps

GET /best-times/all

Paramètre: Aucun

Corps: Aucun

Réponse: Code 200. Le corps de la réponse est un objet contenant tous les meilleurs temps pour toutes les parties.

GET /best-times/${gameTitle}/${gameMode}

Paramètre: gameTitle, qui représente le nom de la carte de jeu et gameMode, le mode de jeu de la partie, soit solo ou 1v1.

Corps: Aucun

Réponse: Code 200. Le corps de la réponse est un objet contenant tous les meilleurs temps pour la partie désirée.

POST /best-times/${name}

Paramètre: name, qui représente le nom de la carte pour laquelle on veut mettre à jour les meilleurs temps et time, qui est le nouveau temps obtenu par le joueur ainsi que son nom et s'il jouait en solo ou non.

Corps: time, qui contient l'information pour mettre à jour les meilleurs temps

Réponse: Code 201. La réponse nous indique que le temps a bien été mis à jour.

POST /best-times/reset/${name}

Paramètre: name, qui représente le nom de la carte pour laquelle on veut mettre à neuf les meilleurs temps

Corps: Aucun

Réponse: Aucun code de retour. Il s'agit plutôt d'une opération 'subscribe' afin d'avoir le résultat de retour.

POST /best-times/reset/

Paramètre: Aucun

Corps: Aucun

Réponse: Aucun code de retour. Il s'agit plutôt d'une opération 'subscribe' afin d'avoir le résultat de retour.

3.1.8 Manipulation de l'historique de jeu

GET /history/${name}

Paramètre: name, qui représente le nom de la carte pour laquelle on veut aller chercher l'historique

Corps: Aucun

Réponse: Code 200. Un objet contenant l'historique des parties pour une carte donnée.

GET /history/all

Paramètre: Aucun

Corps: Aucun

Réponse: Code 200. Un objet contenant l'historique des parties pour toutes les cartes jouées

DELETE /history

Paramètre: Aucun

Corps: Aucun

Réponse: Code 200. Confirmation de la suppression de l'historique

PUT /history

Paramètre: Objet de type GameHistoryInfo, contenant le nom du jeu, le gagnant et le perdant de la partie, la durée de la partie, ainsi que plusieurs booléens déterminant s'il y a eu un abandon, s'il la partie a été jouée en solo et un dernier indiquant si la partie était en mode temps limité.

Corps: Le paramètre en question est ce qui forme le corps.

Réponse: Code 201. Confirmation de l'ajout de l'objet dans la base de données.

3.1.9 Manipulation des constantes de jeu

GET /gameConfig/constants/countdownTime

Paramètre: Aucun

Corps: Aucun

Réponse: Code 200. Un nombre contenant le temps de départ pour le mode temps limité

GET /gameConfig/constants/penaltyTime

Paramètre: Aucun

Corps: Aucun

Réponse: Code 200. Un nombre contenant le temps appliqué en pénalité pour l'utilisation d'un indice

GET /gameConfig/constants/timeGained

Paramètre: Aucun

Corps: Aucun

Réponse: Code 200. Un nombre contenant le temps gagné lors de la découverte d'une différence en temps limité.

POST /gameConfig/constants/countdownTime
Paramètre: time, qui est la valeur de temps à ajuster dans les constantes
Corps: Objet contenant le nouveau temps à ajuster dans les constantes
Réponse: Code 204. Réponse de succès sans objet

POST /gameConfig/constants/penaltyTime
Paramètre: time, qui est la valeur de temps à ajuster dans les constantes
Corps: Objet contenant le nouveau temps à ajuster dans les constantes
Réponse: Code 204. Réponse de succès sans objet

POST /gameConfig/constants/timeGained
Paramètre: time, qui est la valeur de temps à ajuster dans les constantes
Corps: Objet contenant le nouveau temps à ajuster dans les constantes
Réponse: Code 204. Réponse de succès sans objet

3.2 WebSockets

Préface: Voici la signification des attributs utilisés dans les événements WebSockets:

- gameMaster: Désigne le joueur ayant créé un lobby
- joiningPlayer: Désigne le joueur qui rejoint et complète un lobby
- name: Désigne le nom qu'un joueur sur son client
- gameTitle: le nom d'une partie
- roomId: l'identifiant de la salle auquelle deux joueurs sont connectés
- player1: Désigne le booléen qui décrit si un joueur est le gameMaster
- response: Désigne un ClickResponse contenant les informations reliées à une différence spécifique
- message: Désigne le contenu d'un message envoyé par un joueur sur le chat

Nom d'événement	Source	Contenu
'solo-game'	Client (SocketService)	Aucun
'one-vs-one-game'	Client (SocketService)	Aucun
'handle-lobby'	Client (SocketService)	gameMaster: string gameTitle: string
'start-OneVsOne'	Client (SocketService)	Aucun
'reject-player'	Client (SocketService)	gameMaster: string gameTitle: string
'close-lobby'	Client (SocketService)	gameTitle: string
'leave-lobby'	Client (SocketService)	gameMaster: name gameTitle: gameTitle
'reset-lobby'	Client (SocketService)	roomId: string
'get-gameTitle'	Client (SocketService)	roomId: string
'delete-game'	Client (SocketService)	gameTitle: string
'get-OneVsOne-info'	Client (SocketService)	gameTitle: string
'init-OneVsOne-components'	Client (SocketService)	player1: boolean
'send-difference-found'	Client (SocketService)	response: ClickResponse
'leave-game'	Client (SocketService)	Aucun
'send-player-message'	Client (SocketService)	name: string message: string
'send-player-error'	Client (SocketService)	name: string
'send-player-success'	Client (SocketService)	name: string
'on-victory-sequence'	Client (SocketService)	player1: boolean

Nom d’événement	Source	Contenu
‘reset-timer’	Client (SocketService)	roomId: string
‘increment-counter’	Client (SocketService)	player1: boolean
‘reset-counter’	Client (SocketService)	player1: boolean
‘redirect-to-game’	Serveur (ClassicModeGateway)	‘/gameOnevsOne’ Morceau d’URL pour pouvoir rediriger les joueurs à la bonne page
‘player-quit-game’	Serveur (ClassicModeGateway)	Aucun
‘game-deleted’	Serveur (ClassicModeGateway)	gameTitle: string
‘awaiting-lobby’	Serveur (ClassicModeGateway)	lobby.gameTitle: string
‘lobby-created’	Serveur (ClassicModeGateway)	completeGameInfo: { gameMaster: string joiningPlayer: string gameTitle: string roomId: string }
‘completed-lobby’	Serveur (ClassicModeGateway)	lobby.gameTitle: string
‘counter-update’	Serveur (ClassicModeGateway)	counter: number player1: boolean
‘update-difference’	Serveur (ClassicModeGateway)	response: ClickResponse
‘incoming-player-message’	Serveur (ClassicModeGateway)	messageInfo: { name: string message: string }
‘player-error’	Serveur (ClassicModeGateway)	name: string
‘player-success’	Serveur (ClassicModeGateway)	name: string
‘rejection’	Serveur (ClassicModeGateway)	‘/game-selection’ Morceau d’URL pour pouvoir rediriger le joueur à la bonne page
‘player-left’	Serveur (ClassicModeGateway)	Aucun

Nom d’événement	Source	Contenu
‘player-info’	Serveur (ClassicModeGateway)	gameplayInfo: { gameTitle: string roomId: string player1: boolean}
‘lobby-closed’	Serveur (ClassicModeGateway)	‘/game-selection’ Morceau d’URL pour pouvoir rediriger les joueurs à la bonne page
‘send-victorious-player’	Serveur (ClassicModeGateway)	player1: boolean
‘timer’	Serveur (ClassicModeGateway)	time: number
‘leave-limited-time’	Client (SocketService)	Aucun
‘send-new-record’	Client (SocketService)	name: string position: string title: string mode: string
‘initialize-game’	Client (SocketService)	gameTitles: string[]
‘verify-position’	Client (SocketService)	mousePosition: Coords
‘delete-room-game-info’	Client (SocketService)	Aucun
‘add-to-timer’	Client (SocketService)	Aucun
‘remove-to-timer’	Client (SocketService)	Aucun
‘switch-game’	Client (SocketService)	Aucun
‘switch-images	Serveur (ClassicModeGateway)	images: string[] title: string
‘click-response’	Serveur (ClassicModeGateway)	ClickReponse: { isDifference: boolean differenceNumber: number coords: Coords[][]