

Protocole de communication

Version 1.0

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

Table des matières

1. Introduction	4
2. Communication client-serveur	4
3. Description des paquets	4
3.1 Protocole HTTP	5
3.1.1 Peupler le carrousel de cartes de jeu	5
3.1.2. Enregistrement d'une nouvelle partie	5
3.1.3. Détection d'erreur à la position du clique	6
3.1.4. Recevoir les informations d'une partie	6
3.1.5. Mode triche	7
3.1.6 Créer une nouvelle partie	7
3.1.7 Effacer une partie	8
3.2 WebSockets	8

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 créée	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: 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
originallamge	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: aucune valeur dans le corps de la réponse

GET /games/names

Paramètre: aucun

Corps: aucun

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

3.1.3. Détection d'erreur à la position du clique

POST /gaming/find

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
mousePosition	Coords	coordonnée du clique

Réponse: La réponse est un objets avec les caractéristique suivantes:

Clé	Type de la valeur	Description
IsDifference	boolean	s' il s'agit d'une différence
differenceNumber	number	le numéro de la différence
coords	Coords[]	positions des pixels de différence

3.1.4. 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: 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: 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: La réponse contient un boolean qui indique si la partie est disponible.

3.1.5. 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: 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

3.1.6 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
originallImage	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

3.1.7 Effacer une partie

DELETE /games/\${name}

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

Corps: Aucun

Réponse: Le corps de la réponse est vide.

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 auquel 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

Nom d'événement	Source	Contenu
'send-player-error'	Client (SocketService)	name: string
'send-player-success'	Client (SocketService)	name: string
'on-victory-sequence'	Client (SocketService)	player1: boolean
'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	'/game-selection'

Nom d'événement	Source	Contenu
	(ClassicModeGateway)	Morceau d'URL pour pouvoir rediriger le joueur à la bonne page
'player-left'	Serveur (ClassicModeGateway)	Aucun
'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