102

Protocole de communication

Version 2.0

Historique des révisions

Date	Version	Description	Auteur
2023-03-19	1.0	Entièreté du document	Antoine G. Soldati
2023-04-17	2.0	Correction pour le sprint 3	Antoine G. Soldati

Table des matières

Protocole de communication

1. Introduction

Ce document présente la documentation de la communication client-serveur de notre application. Les protocoles HTTP et WebSocket sont utilisés pour faire fonctionner le logiciel. Dans la prochaine section, seront décrits les cas d'utilisations pour chaque protocole. Cette section sert à décrire dans quel cas le protocole de communication est utilisé et pour accomplir quelle tâche. La dernière section de ce document explique les différentes méthodes HTTP utilisées, les paramètres, le corps et les messages de réponse. Cette information est nécessaire pour pouvoir savoir comment utiliser l'API de notre serveur. Cette énumère également les différents événements websockets utilisés pour faire communiquer le serveur et le client avec des sockets. Il est indiqué la source de l'événement, le contenu passé en paramètre et le nom de l'événement.

2. Communication client-serveur

HTTP

- Utilisé pour demander toutes les cartes au serveur, afin de remplir la vue de sélection et de configuration.
- Utilisé pour demander une carte en particulier au serveur, afin de commencer le jeu.
- Utilisé pour sauver une carte sur le serveur
- Utilisé pour avoir les constantes de jeu
- Utilisé pour effacer une carte sur le serveur.
- Utilisé pour effacer toutes les cartes sur le serveur
- Utilisé pour demander l'image de différences d'un jeu en cours de création
- Utilisé quand un joueur souhaite remettre les statistiques d'une carte à l'état initial
- Utilisé quand un joueur souhaite remettre les statistiques de toutes les cartes à l'état initial
- Utilisé pour réinitialiser l'historique
- Utilisé pour avoir l'historique des parties
- Utilisé quand un joueur change les constantes de jeu
- Utilisé pour avoir l'historique des parties jouées et des meilleurs temps
- Utilisé pour avoir la reprise vidéo de la partie
- Utilisé pour avoir les images des cartes

WEBSOCKET

- Utilisé pour gêrer le lobby. Un événement est envoyé quand un joueur créé un lobby, pour informer les autres clients et changer le bouton 'créer' à 'joindre'. Un événement est envoyé quand un joueur demande de rejoindre un lobby et quand il abandonne sa demande pour rejoindre. Un événement est aussi envoyé quand un joueur abandonne sa création de lobby.
- Des événements sont envoyés quand un client se connecte et se déconnecte du serveur. Cela est utile pour pouvoir mettre à jour l'état des boutons de jeu multijoueurs et de signaler au serveur que un jeu est finit puisqu'un joueur a quitté.
- Un événement est également envoyé au serveur pour signifier qu'un joueur a rejoint une partie en solo ou en multijoueur. Le serveur va pouvoir créer la partie ou créé/mettre à jour le lobby.
- Un événement de click est envoyé quand un client clique sur une image de jeu et le serveur renvoit un événement signifiant une réussite ou une erreur.
- Le serveur peut également envoyer un événement de victoire si le click a fait en sorte que la partie est gagné ou si un autre joueur se déconnecte/abandonne le jeu.
- Des événements sont envoyés quand un joueur créé ou efface une carte pour mettre à jour la vue de sélection et de configuration.
- Un événement est envoyé quand un joueur abandonne le jeu à partir du bouton dans le jeu. L'autre joueur gagne automatiquement.
- Un événement est envoyé au serveur et au client quand un message est entré dans le chatbox et quand un événement de partie ou global est envoyé par le serveur
- Un événement est envoyé quand une partie se finit sans qu'un joueur abandonne pour permettre de mettre à jour les statistiques de la carte.

- Quand un joueur demande un indice de jeu, on événement est envoyé au serveur pour signaler un indice, le serveur
- Quand un joueur finit une partie, un événement est envoyé pour mettre à jour l'historique des parties jouées et des meilleurs temps.
- Quand la partie prend fin, sans abandon, un événement est envoyé pour signaler une fin de partie avec la statistique du joueur gagnant
- Un événement est envoyé à tous les clients quand un joueur change les constantes de jeu
- Un événement est envoyé à tous les clients quand le mode limité est disponible ou pas
- Un événement est reçu et envoyé quand un joueur demande un indice et que le serveur lui répond
- Un événement est envoyé au joueur dans un jeu en 1v1 quand l'autre joueur quitte le jeu
- Un événement est envoyé à tous les clients quand l'historique est mis à jour
- Un événement est envoyé en temps limité pour signifier une fin de partie

3. Description des paquets

Adresse du serveur (api) : http://ec2-15-223-122-215.ca-central-1.compute.amazonaws.com:3000/api
Adresse du serveur (websocket) : http://ec2-15-223-122-215.ca-central-1.compute.amazonaws.com:3000

HTTP

Interface pour la communication :

Message {status: string, body: string}

- status : Le status de la requête
- body : Le contenu de la requête

Interface pour la communication de cartes sur le réseau:

CardIO {firstImage: string, secondImage: string, metadata: Card}

- firstImage : L'image originale de type base64
- secondImage : L'image modifiée de type base64
- metadata : L'information de la carte

Interface pour l'information de carte :

```
Card  {
   enlargementRadius: number;
   differences: Coordinate[][];
   title: string;
   stats: CardStats;
   difficultyLevel: string;
   id: string;
}
```

- enlargementRadius : Le rayon d'élargissement de la carte
- differences : Les différences de la carte sous forme de listes de coordonnées
- title : Le titre de la carte
- stats : Les statistiques de la carte
- difficultyLevel : Le niveau de difficulté de la carte
- id : L'identifiant unique de la carte

Interface pour les coordonnées : Coordinate { x: number; y: number; }
Interface pour les stats:

CardStats { classical: FirstModeStats; }

- classical : Les statistiques du mode classique

FirstModeStats { solo: SecondModeStats[]; versus: SecondModeStats[]; }

- solo : Les statistiques pour le mode solo
- versus : Les statistiques pour le mode versus

SecondModeStats { name: string; score: number; }

- name : Le nom du joueur
- score : Le temps en secondes de sa partie

Interface pour les constantes: GameConstants{initial: number, penalty: number, gain: number}

- Chaque constante est en secondes

Interface pour l'historique :

GameHistory {dateStarted: string, timeStarted: string, timeLength: string, gameType: string, firstPlayer: string, secondPlayer: string, winnerSocketId: string, surrender: boolean, firstPlayerSocketId: string, secondPlayerSocketId: string}

- dateStarted: La date de début de la partie
- timeStarted: Le temps de début de la partie
- timeLength: La durée de la partie
- gameType: Le type de la partie (classique ou Temps limité)
- firstPlayer: Le nom du premier joueur
- secondPlayer: Le nom du second joueur
- winnerSocketId: L'id du socket du joueur gagnant
- surrender: true si un joueur a surrender, false sinon
- firstPlayerSocketId: Le socketId du premier joueur
- secondPlayerSocketId: Le socketId du second joueur

Tout message d'erreur de la forme (message_error) signifie que le serveur va spécifier le message d'erreur dans la requête.

Cards:

POST /card
Description: Ajouter une carte au serveur
Corps: {status: 'card', body: CardIO as string}
Réponses:

- Code 200 : Message {status: 'OK', body: '(card_id)'}
- Code 400 : Message {status: 'BAD REQUEST', body: 'Request badly formulated. Information missing'}
- Code 500: Message {status: 'INTERNAL SERVER ERROR', body: '(message_error)'}

DELETE /card/stats/{id}
Description: Réinitialiser les meilleurs temps
Paramètre: Id de la carte en string
Réponses:

- Code 204 (réussite)
- Code 404: Message {status: 'NOT FOUND', body: 'No card was found with this id'}
- Code 500: Message {status: 'INTERNAL SERVER ERROR', body: '(message_error)'}

DELETE /card/stats
Description: Réinitialiser les meilleurs temps de tous les jeux
Réponses:

- Code 204 (réussite)
- Code 500: Message {status: 'INTERNAL SERVER ERROR', body: '(message_error)'}

POST /card/constants
Description: Réinitialiser les constantes de jeux pour une carte
Corps: {status: 'constants', body: '(information_constants as Constants)'}
Réponses:

- Code 204 (réussite)
- Code 400 : Message {status: 'BAD REQUEST', body: 'Request badly formulated. Information missing'}
- Code 500: Message {status: 'INTERNAL SERVER ERROR', body: '(message_error)'}

GET /card/constants
Description: Avoir les constantes de jeu
Réponses:
- Code 200 : Message {status: ‘OK’, body: ‘constants as GameConstants as string’}
- Code 500: Message {status: ‘INTERNAL SERVER ERROR’, body: ‘(message_error)’}

GET /card/history
Description: Avoir l’historique de jeu
Réponses:
- Code 200 : Message {status: ‘OK’, body: ‘history as GameHistory as string’}
- Code 500: Message {status: ‘INTERNAL SERVER ERROR’, body: ‘(message_error)’}

DELETE /card/history
Description: Réinitialiser l’historique de toutes les cartes
Réponses:
- Code 204 (réussite)
- Code 500: Message {status: ‘INTERNAL SERVER ERROR’, body: ‘(message_error)’}

DELETE /card
Description: Effacer toutes les cartes
Réponses:
- Code 204 (réussite)
- Code 500: Message {status: ‘INTERNAL SERVER ERROR’, body: ‘Couldn't delete all cards’}

GET /card/{id}
Description: Avoir l’information de la carte
Paramètre: Id de la carte en string
Réponses:
- Code 204 (réussite)
- Code 404: Message {status: ‘NOT FOUND’, body: ‘No card was found with this id’}

DELETE /card/{id}
Description: Effacer une carte
Paramètre: Id de la carte en string
Réponses:
- Code 204 (réussite)
- Code 404: Message {status: ‘NOT FOUND’, body: ‘Couldn't delete Card with id (card_id as string) ’}

POST /card/difference-image
Description: Avoir l’image de différences
Corps: {status: ‘card’, body: CardIO as string}
Réponses:
- Code 200 : Message {status: ‘OK’, body: ‘(image_difference as CardIO)’}
- Code 400 : Message {status: ‘BAD REQUEST’, body: ‘Request badly formulated. Information missing’}
- Code 500: Message {status: ‘INTERNAL SERVER ERROR’, body: ‘(message_error as string)’}

GET /card
Description: Avoir toutes les cartes sur le serveur avec leur titre, image originale, statistiques, historique des parties jouées et meileurs temps
Réponses:
- Code 200 Message {status: ‘OK’, body: ‘(cards as Card[])’}
- Code 404: Message {status: ‘NOT FOUND’, body: ‘No card was found with this id’}
- Code 500 : Message {status: ‘INTERNAL SERVER ERROR’, body: ‘(message_error as string)’}

Images:

GET /image/{id}
Description: Avoir l'image associé à l'id
Paramètre: Id de la carte en string
Réponses:

- Code 200 : Fichier de type File
- Code 404: Message {status: 'NOT FOUND', body: 'Image not found with this id'}

WEBSOCKET

Interface pour un message de chat : ChatEntry {message: string; timestamp: string; type: ChatEntryType; }

- timestamp : Temps d'envoi du message
- type : Type du message

Enum ChatEntryType {USER = 0, EVENT = 1, GLOBAL = 2, SELF = 3, OPPONENT = 4}

Interface pour information de lobby : LobbyIO {

```
  cardId: string;
  firstMode?: string;
  secondMode?: string;
  firstPlayerName?: string;
  secondPlayerName?: string;
  firstPlayerId?: string;
  secondPlayerId?: string;
}
```

- cardId : L'id unique de la carte
- firstMode : Le premier mode de la carte (classique ou Temps limité)
- secondMode: Le deuxième mode de la carte (solo ou versus)
- firstPlayerName : Le nom du premier joueur
- secondPlayerName : Le nom du second joueur
- firstPlayerId : L'id du premier joueur
- secondPlayerId : L'id du second joueur

Interface pour succès de click : SuccessClick{socketId: string; differences: Coordinate[];}

- differences : Les coordonnées de la différence trouvée

Interface pour une statistique de joueur : PlayerStats {

```
  cardId: string;
  firstMode: string;
  secondMode: string;
  playerName: string;
  score: number;
  position: number;
}
```

- firstMode : Le premier mode de la carte (classique ou Temps limité)
- secondMode: Le deuxième mode de la carte (solo ou versus)
- score: Le temps en secondes
- position: La position dans le leaderboard

Connexion:

Connection d'un client
Source: Client
Événement : 'connection'
Contenu : Vide

Déconnexion d'un client
Source: Client
Événement : 'disconnect'
Contenu : Vide

Abandonner le jeu
Source: Client
Événement : 'surrender'
Contenu : Vide

Jeu:

Click sur une image
Source: Client
Événement : 'handleClick'
Contenu : La position du click de type Coordinate en string

La position du click indique un succès
Source: Serveur
Événement : 'success'
Contenu : L'information du succès de type SuccessClick en string

La position du click indique une erreur
Source: Serveur
Événement : 'error'
Contenu : La position du click de type Coordinate en string

Envoie d'un message pour le chatbox
Source: Client et serveur
Événement : 'message'
Contenu : Le message en question de type ChatEntry en string

La partie est finie
Source: Serveur
Événement : 'gameEnded'
Contenu : La statistique du joueur gagnant de type PlayerStats

Un joueur a gagné la partie
Source: Serveur
Événement : 'winner'
Contenu : Le id du socket du joueur gagnant en string

Un joueur demande un indice pour la partie
Source: Serveur et client
Événement : 'clue'
Contenu : La différence de type Coordinate[] en string

Un joueur quitte la partie
Source: Serveur
Événement : 'playerQuit'
Contenu : Rien

La partie en temps limité est terminée
Source: Serveur
Événement : ‘endGame’
Contenu : Rien

Lobby:

Ajout d’un joueur au jeu
Source: Client
Événement : ‘addPlayer’
Contenu : L’information du Lobby de type LobbyIO en string

Le créateur du lobby a abandonné son lobby
Source: Client et Serveur
Événement : ‘createAborted’
Contenu : L’information du lobby de type LobbyIO en string

Le créateur accepte la demande d’un joueur qui veut rejoindre le jeu
Source: Client
Événement : ‘joinRequestAccepted’
Contenu : L’information du lobby de type LobbyIO en string

Le joueur qui rejoint a abandonné le lobby
Source: Client et Serveur
Événement : ‘joinRequestAborted’
Contenu : L’information du lobby de type LobbyIO en string

Le joueur qui rejoint a été rejeté
Source: Client et Serveur
Événement : ‘joinRequestRejected’
Contenu : L’information du lobby de type LobbyIO en string

Le joueur qui rejoint a abandonné le lobby
Source: Client et Serveur
Événement : joinRequestAborted
Contenu : L’information du lobby de type LobbyIO en string

Gestion des cartes:

Une carte a été créé
Source: Serveur
Événement : ‘created’
Contenu : L’information du lobby de type LobbyIO en string

Une carte a été effacée
Source: Serveur
Événement : ‘cardDeleted’
Contenu : Le id de la carte en question en string

Les statistiques d’une carte doivent être mises à jour
Source: Serveur
Événement : ‘statsChanged’
Contenu : L’information de la carte de type Card en string

L’historique d’une partie doit être mis à jour
Source: Serveur
Événement : ‘historyChanged’
Contenu : L’information de la partie de type Card en string

Les constantes doivent être mises à jour
Source: Serveur
Événement : ‘constantsChanged’
Contenu : Les constantes de type GameConstants en string

L’état du lobby en temps limité doit être mis à jour
Source: Serveur
Événement : ‘limitedModeEnable’
Contenu : true si le mode temps limité est disponible, false sinon (en string). Le mode temps limité est disponible s’il y a des cartes