PAINseau

Protocole de communication

Version 1.2

Historique des révisions

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Auteur** |
| 2021-02-07 | 1.0 | Rédaction initiale du protocole de communication | Équipe 107 |
| 2021-02-16 | 1.1 | Première révision en équipe du protocole de communication | Équipe 107 |
| 2021-02-18 | 1.2 | Version finale pour la réponse à l’appel d’offre | Équipe 107 |
|  |  |  |  |
|  |  |  |  |

Table des matières

[1. Introduction 2](#_Toc64625507)

[2. Communication client-serveur 2](#_Toc64625508)

[3. Description des paquets 2](#_Toc64625509)

[Communication REST API 2](#_Toc64625510)

[Communication des sockets 4](#_Toc64625511)

Protocole de communication

# 1. Introduction

Ce document a pour objectif de présenter les détails entourant la communication client-serveur afin de dresser une idée générale des flux de communication. En d’autres mots, ce présent document justifiera d’abord l’utilisation des technologies utilisées pour la communication client-serveur, tout en précisant les parties spécifiques de notre application qui y sont touchées. Une description des types de données transférées s’y retrouvera également. La section suivante contiendra ensuite une description du contenu des différents types de paquets utilisés au sein de notre protocole.

# 2. Communication client-serveur

Un premier type de communication entre le client et le serveur est les requêtes HTTP. Ces requêtes serviront principalement à la gestion des utilisateurs. Ainsi, elles permettront d’envoyer les informations nécessaires au serveur à partir du client pour enregistrer un nouvel utilisateur, pour se connecter, pour récupérer ou envoyer de l’information du profil d’un utilisateur, etc. Cette communication est cependant unidirectionnelle et ne permet pas au serveur de communiquer au client. Le serveur sera hébergé sur la plateforme Heroku afin d'être accessible à travers Internet et donc on pourra réaliser des requêtes au serveur hébergé à travers n'importe quelle machine. Le code du côté serveur sera écrit en Typescript pour gérer les requêtes HTTP. Un port sera automatiquement assigné par Heroku pour gérer les requêtes. Une base de données MongoDB sera utilisée pour sauvegarder les informations. Cette dernière sera hébergée sur Atlas afin d'être accessible sur Internet.

Un deuxième type de communication utilisé pour notre application est celui des sockets. Ces derniers serviront principalement pour l’envoi, la réception et la gestion des messages en temps réel. Les sockets sont initialisés dans les clients et le serveur. La librairie de socket.io sera utilisée pour implémenter cette partie de code du côté serveur ainsi que dans les deux clients. Heroku assignera des ports automatiquement pour gérer les émissions des sockets.

Lors de la connexion d’un utilisateur, il n’est pas nécessaire de fournir une adresse IP puisqu’une connexion automatique est effectuée.

# 3. Description des paquets

## Communication REST API

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Description | Méthode | Requête | Paramètres | Données retournées |
| Ajouter un usager | POST | /database/add-user | {user: IUser} | {confirmation: string} |
| Recevoir la liste d’usagers | GET | /database/users | N/A | {users: User[]} |
| Supprimer un utilisateur | DELETE | /database/delete-user | {user: IUser} | {confirmation: string} |
| Recevoir informations à propos d’un usager | GET | /database/user | {user: IUser} | {user: IUser} |
| Modifier un utilisateur | POST | /database/modify-user | {user: IUser} | {confirmation: string} |
| Connexion d’un usager | POST | /database/login | {user: IUser, password: string} | {confirmation: string} |
| Déconnexion d’un usager | DELETE | /database/logout | {user: IUser, password: string} | {confirmation: string} |
| Recevoir la liste des usagers connectés | GET | /database/connected | N/A | {users: IUser[]} |
| Ajouter un dessin | POST | /database/add-drawing | {drawing: IDrawing, user: IUser} | {confirmation: string} |
| Recevoir la liste de dessins | GET | /database/drawings | {user: IUser} | {drawings: Drawing[]} |
| Ajouter un dessin dans la banque de paire mot-image | POST | /database/add-wordImage | {pair: IWordImage} | {confirmation: string} |
| Recevoir la liste de dessins de la banque paire mot-image | GET | /database/wordImages | N/A | {drawings: IWordImage[]} |
| Recevoir les informations d’un canal de discussion | GET | /database/chatRoom | {chatroom: IChatroom} | {messages: IMessage[]} |
| Supprimer des informations d’un canal de discussion | DELETE | /database/delete-chatRoom | {chatroom: IChatroom} | {confirmation: string} |
| Ajouter les informations d’un canal de discussion | POST | /database/add-chatRoom | {messages: IMessage[]} | {confirmation: string} |
| Ajouter un utilisateur à un canal de discussion | POST | /database/add-user-chatRoom | {chatroom: IChatroom, user: IUser} | {confirmation: string} |
| Retirer un utilisateur d’un canal de discussion | DELETE | /database/delete-user-chatRoom | {chatroom: IChatroom, user: IUser} | {confirmation: string} |
| Recevoir la liste d’utilisateurs connectés à un canal de discussion | GET | /database/users-chatRoom | {chatroom: IChatroom} | {users: IUser[]} |

## 

## Communication des sockets

|  |  |  |
| --- | --- | --- |
| Description | Événement envoyé | Données |
| Connexion d’un utilisateur à un canal de clavardage | user-joined | {user: IUser, chatroom: IChatRoom} |
| Déconnexion d’un utilisateur un canal de clavardage | logout | {user: IUser, chatroom: IChatRoom} |
| Échange de messages dans un canal de clavardage | send-message | {message: IMessage} |
| Commencement d’une partie de jeu | new game | {game; IGame} |
| Échange de traits de dessin | stroke | {drawing: IDrawing} |
| Signalisation du début d’un tour | new round | {round: IRound} |
| Signalisation de la fin d’un tour | end round | {round: IRound} |
| Validation des essais lors d’un tour | game chat | {guesses: IGuess[]} |
| Fin d’une partie de jeu | end game | {score: int, gameID: string} |

**Interfaces**

|  |  |
| --- | --- |
| IChatroom | {msgHistory: Message[], name: string, users: User[] } |
| IDrawing | {id: string, strokes: Stroke[]} |
| IGame | {mode: string, gameID: string, players: User[], rounds: IRound[], score: int} |
| IGuess | {message: Message[], guessCount: int, validAnswer: bool} |
| IMessage | {author: IUser, timestamp: time, message: string, chatRoomID: string} |
| IRound | {timeElapsed: string, roundID: int, validWord: string} |
| IStroke | {color: string, svg: SVG, order: int, width: int, positions: int[]} |
| IUser | {username: string, password: string, avatar:png, title: string, border: png, experience: int, level: int} |
| IWordImage | {word: string, drawing: IDrawing} |