# Équipe 207

# Fais-moi un dessin
# Protocole de communication

**Version 2.0**

# Historique des révisions

| Date | Version | Description | Auteur |
|---|---|---|---|
| 2021-02-03 | 1.0 | Structure de base du document | Marc-Olivier Riopel |
| | 1.1 | Communication client-serveur | Julien Witty |
| 2021-02-07 | 1.2 | Premier jet de la description des paquets | Marc-Olivier Riopel & Samuel Ouvrard |
| 2021-02-13 | 1.3 | Avancement de la description des paquets | Marc-Olivier Riopel & Samuel Ouvrard |
| 2021-02-17 | 1.4 | Modification de requêtes en lien avec la transmission des dessins | Marc-Olivier Riopel |
| 2021-02-18 | 1.5 | Modification de la section 2. | Marc-Alain Tétreault |
| 2021-04-11 | 2.0 | Mise à jour des routes | Marc-Olivier Riopel |

# Table des matières


Introduction **5**

Communication client-serveur **5**

- Technologies utilisées 5
  - Socket.io 5
  - HTTP 6
- Liens entre les technologies utilisées et les fonctionnalités du système 6
  - Socket.io 6
  - HTTP 6

Interfaces nécessaires à la communication client-serveur **7**

- Vec2 7
- Stroke 7
- DrawingEvent 7
- MouseDown 7
- Chat 8
- IncomingMessage 8
- ChatHistory 8
- PrivateStats 9
- Log 9
- GameInfo 9
- BasicUser 9
- AuthInfo 10
- DetailedUser 10
- Player 10
- Lobby 10
- Vote 11
- Drawing 11

Description des packets **12**

- Client vers serveur avec Socket 12
  - Clavardage 12
  - Connexion 12
  - Dessins 12
  - Parties 13
- Serveur vers client avec Socket 14
  - Clavardage 14
  - Dessins 14
  - Parties 15
- Communication client-serveur avec REST API 16
  - Clavardage 16

Liste des canaux de discussion 16
Liste des noms des canaux de discussion d'un utilisateur 16
Création d'un nouveau canal de discussion 17
Supprimer un canal de discussion 17
Historique des canaux de discussion 18
Historique d'un canal de discussion 19
Dessins 20
Envoyer le mot choisi 20
Authentification 21
S'authentifier à l'application 21
Déconnecter un utilisateur de l'application 21
Enregistrer un nouvel utilisateur 22
Demander le temps de la dernière connexion 23
Statistiques 23
Récupérer les statistiques de l'utilisateur 23
Tableau de classement 24
Demander les statistiques des 10 meilleurs joueurs 24
Fil d'actualité 25
Demander les dessins du fil d'actualité 25
Parties 25
Demander les parties publiques 25
Commencer une partie 26
Créer une partie publique 26
Créer une partie privée 27
Rejoindre une partie publique 28
Rejoindre une partie privée 28
Quitter une partie 29
Ajouter un joueur virtuel 29
Retirer un joueur virtuel 30
Envoyer le dessin de la venant de se terminer 30
Votes sur dessin du joueur virtuel 31
Ajouter un vote 31
Création paire mot-image 32
Ajouter une paire mot-image 32

# Protocole de communication

## 1. Introduction

Le but du document de protocole de communication est de décrire les différentes requêtes nécessaires pour bien implémenter les différentes fonctionnalités de l'application. La partie 2 du document tentera de décrire la technologie utilisée dans le cadre du développement de l'application. La partie 3 a pour but de décrire toutes les requêtes aux serveurs jugées nécessaires pour remplir les critères des exigences fonctionnelles établies dans le document SRS.

## 2. Communication client-serveur

### 2.1. Technologies utilisées

L'application *Fais-moi un dessin* est composée d'un client lourd et d'un client léger. Il est important que le serveur soit en mesure de répondre aux requêtes des clients et que le serveur se comporte uniformément, peu importe la plateforme utilisée.

#### *2.1.1. Socket.io*

La communication client-serveur est facilitée par l'utilisation de la librairie *Socket.io*. Construite à partir du protocole WebSocket, Socket.io est une librairie puissante qui facilite les communications entre plusieurs clients et le serveur en temps réel. Cette technologie nous permet d'éviter d'implémenter nous-mêmes les différentes requêtes lors de l'ouverture d'un canal.

WebSocket est un protocole de communication basé sur une connexion TCP. WebSocket est particulièrement utile lorsqu'il est nécessaire d'acheminer et transmettre de l'information en temps réel. Lors des échanges en temps réel, le port TCP reste ouvert en tout temps. La latence de ce protocole de communication est négligeable et est très adaptée pour l'application actuelle.

Comme il a déjà été mentionné, la librairie Socket.io permet de faire abstraction de la complexité du protocole de communication WebSocket. Cette librairie offre plusieurs avantages. Premièrement, la librairie s'adapte automatiquement aux différentes mises à jour du protocole de communication. Socket.io va être particulièrement utile pour diffuser de l'information à plusieurs sockets en même temps.

La librairie Socket.io possède 2 volets: un côté client et un côté serveur. Le côté serveur de la libraire a 4 types d'instances principales ayant des objectifs différents. Premièrement, l'objet Server permet de faire des opérations de bases comme l'écoute d'un port (listen) et lier à des moteurs spécifiques (bind) pour améliorer les performances de connexion. Deuxièmement, les namespaces restreignent la portée de certains sockets. Troisièmement, l'instance Socket est probablement la plus importante de la partie serveur. Chaque Socket appartient à un Namespace. Les instances Sockets ont la possibilité d'émettre certains signaux suite à la réception d'un événement. Il existe plusieurs types d'événements comme "*Broadcast*", "*Volatile*" et "*Disconnect*". La dernière instance de la partie serveur est l'instance Client qui représente une nouvelle connexion au serveur.

L'API client possède 2 types d'instances primaires pertinentes. L'instance socket est très importante pour la communication avec le serveur. L'instance Manager permet de gérer la logique de connexion entre les différents sockets.

### *2.1.2. HTTP*

Tout dépendant des requêtes du client, il sera parfois nécessaire de retrouver certaines données qui vont être entreposées dans la base de données de l'application. La base de données collectera des dessins à afficher sur le fil d'actualités, les mots de passe, etc. MongoDB est régie sous le principe NoSQL qui permet d'avoir une plus grande flexibilité dans nos données. Il est possible de faire les requêtes à la base de données via la REST API. La REST API permet d'effectuer plusieurs types de requêtes HTTP à la base de données. Les données voyagent en format JSON.

En résumé, le client va faire des requêtes au serveur. Le serveur va faire des requêtes à la base de données si nécessaire pour ensuite retourner les résultats au client.

Pour des fins de sécurités, il a été décidé d'utiliser les JSON Web Token pour la gestion des authentifications. Les utilisateurs vont ainsi recevoir un token après que leur authentification ait été confirmée pour qu'ils aient accès au reste de l'application. Ce token est généré selon une clé secrète inconnue des clients et selon le nom d'utilisateur des clients. Cette manière de procéder nous permet de toujours avoir accès au nom d'utilisateur du client lorsque celui-ci nous envoie une requête avec son token.

## 2.2. Liens entre les technologies utilisées et les fonctionnalités du système

### *2.2.1. Socket.io*

Socket.io sera utilisé pour les communications en temps réel. Les fonctionnalités du jeu nécessitant un affichage simultané pour tous les clients utiliseront Socket.io (affichage du dessin en temps réel, le chronomètre, l'affichage des indices et affichage des tentatives). Également, Socket.io sera utilisé pour permettre le clavardage en temps réel entre les utilisateurs connectés.

### *2.2.2. HTTP*

HTTP sera utilisé pour tout transfert d'information entre le client et le serveur ne nécessitant pas d'être en temps réel. Dans certains cas, HTTP sera utilisé conjointement avec Socket.io pour la présentation des données en temps réel ainsi que la persistance de celle-ci. C'est le cas pour le clavardage (communication temps réel et conservation de l'historique) ainsi que pour les parties classiques et sprint coop (le jeu se déroule en temps réel et les informations sur une partie sont conservées).

# 3. Interfaces nécessaires à la communication client-serveur

## 3.1. Vec2

| Attribut | Type | Description |
|---|---|---|
| x | number | Coordonnée en x du point |
| y | number | Coordonnée en y du point |

## 3.2. Stroke

| Attribut | Type | Description |
|---|---|---|
| path | Vec2[] | Tous les points du trait |
| color | string | Couleur du trait |
| lineWidth | number | Épaisseur du trait |

## 3.3. DrawingEvent

| Attribut | Type | Description |
|---|---|---|
| eventType | number | Type d'événement |
| event | Object | L’événement en soi |
| gameId | string | Identifiant de la partie |

## 3.4. MouseDown

| Attribut | Type | Description |
|---|---|---|
| lineColor | string | Couleur sélectionnée |
| lineWidth | number | Épaisseur du trait |
| coords | Vec2 | Coordonnées de l'événement |

### 3.5. Chat

| Attribut | Type | Description |
|---|---|---|
| chatName | string | Nom du canal de discussion |
| chatId | string | Identifiant unique du canal de discussion |

### 3.6. IncomingMessage

| Attribut | Type | Description |
|---|---|---|
| token | string | Jeton de l'utilisateur |
| text | string | Message envoyé |
| chatId | string | Identifiant du canal de discussion |

### 3.7. ChatHistory

| Attribut | Type | Description |
|---|---|---|
| messages | Message[] | Liste des messages envoyées dans un canal de discussion |
| chatName | string | Nom du canal de discussion |
| chatId | uuid | Identifiant unique du canal de discussion |

### 3.8. PrivateStats

| Attribut | Type | Description |
|---|---|---|
| avatar | svg | Avatar de l'utilisateur |
| username | string | Pseudo de l'utilisateur |
| bestSoloScore | number | Meilleur score obtenu en sprint solo |
| classicWinRatio | number | Ratio des parties gagnées/parties perdues en mode classique |
| gamesPlayed | number | Nombre de parties jouées |
| bestCoopScore | number | Meilleur score obtenu en sprint coop |
| timePlayed | number | Durée totale du temps passé à jouer |
| meanGameTime | number | Durée moyenne d'une partie classique |

### 3.9. Log

| Attribut | Type | Description |
|---|---|---|
| isLogin | boolean | isLogin peut être soit une connexion ou une déconnexion |
| timestamp | Date | Date du login/logout |

### 3.10. GameInfo

| Attribut | Type | Description |
|---|---|---|
| gameType | string | Mode de jeu |
| timestamp | Date | Date de la partie |
| players | PublicPlayer[] | Les joueurs de la partie |

### 3.11. BasicUser

| Attribut | Type | Description |
|---|---|---|
| avatar | number | Avatar de l'utilisateur |
| username | string | Pseudo de l'utilisateur |

### 3.12. AuthInfo

| Attribut | Type | Description |
|---|---|---|
| username | string | Pseudo de l'utilisateur |
| password | string | Mot de passe crypté de l'utilisateur |

### 3.13. DetailedUser

| Attribut | Type | Description |
|---|---|---|
| avatar | number | Avatar de l'utilisateur |
| username | string | Pseudo de l'utilisateur |
| name | string | Nom de l'utilisateur |
| surname | string | Prénom de l'utilisateur |
| password | string | Mot de passe de l'utilisateur |

### 3.14. Player

| Attribut | Type | Description |
|---|---|---|
| avatar | number | Avatar de l'utilisateur |
| username | string | Pseudo de l'utilisateur |
| isVirtual | boolean | Type de joueur |
| socketId | string | Identifiant socket de l'utilisateur |

### 3.15. Lobby

| Attribut | Type | Description |
|---|---|---|
| gameType | number | Mode de jeu |
| difficulty | number | Difficulté |
| id | string | Identifiant unique de la partie |
| gameName | string | Nom de la partie |
| isPrivate | boolean | Accès des joueurs à la partie |

### 3.16. Vote

| Attribut | Type | Description |
|---|---|---|
| drawingId | string | Identifiant unique du dessin |
| isUpvote | boolean | Indication s'il s'agit d'un vote positif ou négatif |

### 3.17. Drawing

| Attribut | Type | Description |
|---|---|---|
| drawingId | string | Identifiant unique du dessin |
| drawingVotes | number | Score du dessin actuel |
| strokes | Stroke[] | Liste des traits du dessin |
| hints | string[] | Liste des indices du dessin |
| difficulty | number | Difficulté du dessin |
| drawingName | string | Nom du dessin |

# 4. Description des packets

La section ci-dessous comporte la description des paquets avec Socket.io.

## 4.1. Client vers serveur avec Socket

### *4.1.1. Clavardage*

| Nom de la requête | Description | Attributs |
|---|---|---|
| message | Ajouter un message dans un canal de discussion | {text: string, chatId: string, token: string} |
| leaveChatRoom | Quitter un canal de discussion | {token: string, chatId: string} |
| joinChatRoom | Rejoindre un canal de discussion existant | {toekn: string, chatId: string} |

### *4.1.2. Connexion*

| Nom de la requête | Description | Attributs |
|---|---|---|
| disconnect | Déconnecter un utilisateur de l'application | {token: string} |

### *4.1.3. Dessins*

| Nom de la requête | Description | Attributs |
|---|---|---|
| drawingEvent | Envoie du dessin en cours mis à jour vers les clients | {event: DrawingEvent, token: string} |
| guessDrawing | Tenter de deviner un dessin | {token: string, guess: string, gameId: string} |

*4.1.4. Parties*

| Nom de la requête | Description | Attributs |
|---|---|---|
| listenLobby | Écouter les changements d'un lobby en temps réel | {lobbyId: string, token: string} |
| joinLobby | Rejoindre un lobby | {lobbyId: string, token: string} |
| leaveLobby | Quitter un lobby | {lobbyId: string, token: string} |
| leaveGame | Quitter une partie | {gameId: string, token: string} |
| hintRequest | Demander un indice pour un dessin | {gameId: string, token: string} |
| drawingSuggestions | Demander une liste de suggestions de mots à dessiner | {gameId: string, token: string} |

## 4.2. Serveur vers client avec Socket

*4.2.1. Clavardage*

| Nom de la requête | Description | Attributs |
|---|---|---|
| message | Envoyer le message à tous les clients du canal de discussion | {user: string, text: string, timestamp: number, avatar: number, textcolor: string, chatId: string} |
| joinChatRoomCallback | Notifier le client qu'il a été ajouté à un nouveau canal de discussion | {} |
| leaveChatRoomCallback | Notifier le client qu'il a quitté un canal de discussion | {} |

*4.2.2. Dessins*

| Nom de la requête | Description | Attributs |
|---|---|---|
| guessCallback | Envoyer un message qui indique si le joueur a bien deviné | {isCorrectGuess: boolean, guessingPlayer: string} |
| drawingEvent | Envoie du dessin en cours mis à jour vers les clients | {event: DrawingEvent} |

*4.2.3. Parties*

| Nom de la requête | Description | Attributs |
|---|---|---|
| dispatchTeams | Notifier les joueurs de la mise à jour de l'équipe | {players: Player[]} |
| newRound | Notifier les joueurs du prochain joueur qui doit dessiner | {newDrawingPlayer: string} |
| drawingName | Notifier le joueur qui dessine du dessin à tracer | {drawingName: string} |
| gameStart | Notifier les joueurs que la partie commence avec le premier joueur qui dessine | {player: string, teams: Map<string, Player[]>} |
| score | Notifier les joueurs du score de la partie | {score: number[2]} |
| guessesLeft | Notifier les joueurs du nombre d'essais restant | {guessesLeft: number} |
| endGame | Notifier les joueurs de la fin de la partie et du score final | {finalScore: number[2]} |
| timer | Notifier les joueurs du temps restant pour trouver la réponse | {timer: number} |
| userDisconnect | Notifier les joueurs que quelqu'un s'est déconnecté pendant la partie | {username: string} |
| hintError | Notifier les joueurs qu'il n'y a plus d'indice disponible pour ce dessin | {message: string} |
| transitionTimer | Chronomètre de transition entre les rondes | {timer: number, state: number} |
| maxScore | Notifier les joueurs qu'ils ont atteint le score maximal | {} |
| gameTimer | Chronomètre de la durée de la partie totale | {timer: number} |
| drawingSuggestions | Envoi de suggestions de mots à dessiner | {drawingNames: string[]} |

## 4.3. Communication client-serveur avec REST API

### *4.3.1. Clavardage*

4.3.1.1. Liste des canaux de discussion

| *Attribut* | *Valeur* | *Description* |
|---|---|---|
| **Requête** | | |
| URL | /api/chat/list | Demander la liste de tous les canaux de discussion |
| Type de requête | GET | |
| **En-tête de la requête** | | |
| authorization | jsonwebtoken: string | |
| **Corps de la réponse** | | |
| chat | Chat[] | |
| **Statuts de réponse** | | |
| Si réponse valide | 200 | Ok |
| Si autorisation échoue | 401 | Unauthorized |
| Si mauvaise requête | 400 | Bad request |

4.3.1.2. Liste des noms des canaux de discussion d'un utilisateur

| *Attribut* | *Valeur* | *Description* |
|---|---|---|
| **Requête** | | |
| URL | /api/chat/joined | Demander la liste de noms de tous les canaux de discussion rejoints |
| Type de requête | GET | |
| **En-tête de la requête** | | |
| authorization | jsonwebtoken: string | |
| **Corps de la réponse** | | |
| chat | Chat[] | |
| **Statuts de réponse** | | |
| Si réponse valide | 200 | Ok |
| Si autorisation échoue | 401 | Unauthorized |
| Si mauvaise requête | 400 | Bad request |

4.3.1.3. Création d’un nouveau canal de discussion

| *Attribut* | *Valeur* | *Description* |
|---|---|---|
| **Requête** | | |
| URL | /api/chat/create | Créer un nouveau canal de discussion |
| Type de requête | POST | |
| **En-tête de la requête** | | |
| authorization | jsonwebtoken: string | |
| **Corps de la requête** | | |
| chatName | string | |
| **Corps de la réponse** | | |
| chatId | string | |
| **Statuts de réponse** | | |
| Si réponse valide | 200 | Ok |
| Si autorisation échoue | 401 | Unauthorized |
| Si mauvaise requête | 400 | Bad request |

4.3.1.4. Supprimer un canal de discussion

| *Attribut* | *Valeur* | *Description* |
|---|---|---|
| **Requête** | | |
| URL | /api/chat/delete/:chatID | Supprimer un canal de discussion |
| Type de requête | DELETE | |
| **Paramètres de l'url** | | |
| chatId | string | |
| **En-tête de la requête** | | |
| authorization | jsonwebtoken: string | |
| **Statuts de réponse** | | |
| Si réponse valide | 200 | Ok |
| Si autorisation échoue | 401 | Unauthorized |
| Si canal non vide | 405 | Ce clavardage n’est pas vide |
| Si mauvaise requête | 400 | Bad request |

4.3.1.5. Historique des canaux de discussion

| *Attribut* | *Valeur* | *Description* |
|---|---|---|
| **Requête** | | |
| URL | /api/chat/all/history/ | Demander l'historique des canaux de discussion dont fait partie un utilisateur |
| Type de requête | GET | |
| **En-tête de la requête** | | |
| authorization | jsonwebtoken: string | |
| **Corps de la réponse** | | |
| chats | ChatHistory[] | Liste des historiques des canaux de discussion |
| **Statuts de réponse** | | |
| Si réponse valide | 200 | Ok |
| Si autorisation échoue | 401 | Unauthorized |
| Si mauvaise requête | 400 | bad request |

4.3.1.6. Historique d’un canal de discussion

| *Attribut* | *Valeur* | *Description* |
|---|---|---|
| **Requête** | | |
| URL | /api/chat/history/ | Demander l’historique d’un canal de discussion |
| Type de requête | GET | |
| **En-tête de la requête** | | |
| authorization | jsonwebtoken: string | |
| **Corps de la requête** | | |
| chatId | string | |
| drawingName | string | |
| **Corps de la réponse** | | |
| chats | ChatHistory | Liste de l’historique du canal de discussion |
| **Status de réponse** | | |
| Si réponse valide | 200 | Ok |
| Si autorisation échoue | 401 | Unauthorized |
| Si mauvaise requête | 400 | bad request |

### *4.3.2. Dessins*

4.3.2.1. Envoyer le mot choisi

| *Attribut* | *Valeur* | *Description* |
|---|---|---|
| **Requête** | | |
| URL | /api/games/word/selection | Envoyer le mot choisi pour le dessin |
| Type de requête | POST | |
| **En-tête de la requête** | | |
| authorization | jsonwebtoken: string | |
| **Corps de la requête** | | |
| gameId | string | |
| drawingName | string | |
| **Statuts de réponse** | | |
| Si réponse valide | 200 | Ok |
| Si autorisation échoue | 401 | Unauthorized |
| Si mauvaise requête | 400 | bad request |

### *4.3.3. Authentification*

4.3.3.1. S’authentifier à l’application

| *Attribut* | *Valeur* | *Description* |
|---|---|---|
| **Requête** | | |
| URL | /api/authenticate/login | Authentification à l’application |
| Type de requête | POST | |
| **Corps de la requête** | | |
| username | string | |
| password | string | |
| **Corps de la réponse** | | |
| jsonwebtoken | string | |
| avatar | number | |
| **Statuts de réponse** | | |
| Si réponse valide | 200 | Ok |
| Si mauvais username ou mot de passe | 404 | Not found |
| Si mauvaise requête | 400 | Bad request |

4.3.3.2. Déconnecter un utilisateur de l’application

| *Attribut* | *Valeur* | *Description* |
|---|---|---|
| **Requête** | | |
| URL | /api/authenticate/logout | Déconnexion de l’application |
| Type de requête | POST | |
| **En-tête de la requête** | | |
| authorization | jsonwebtoken: string | |
| **Statuts de réponse** | | |
| Si réponse valide | 200 | Ok |
| Si mauvaise requête | 400 | Bad request |

4.3.3.3. Enregistrer un nouvel utilisateur

| *Attribut* | *Valeur* | *Description* |
|---|---|---|
| **Requête** | | |
| URL | /api/authenticate/register | Enregistrement pour un nouvel utilisateur |
| Type de requête | POST | |
| **Corps de la requête** | | |
| username | string | |
| password | string | |
| name | string | |
| surname | string | |
| avatar | number | |
| **Corps de la réponse** | | |
| token | string | |
| avatar | number | |
| **Statuts de réponse** | | |
| Si réponse valide | 200 | Ok |
| Si username déjà pris | 409 | Conflict |
| Si mauvaise requête | 400 | Bad request |

4.3.3.4. Demander le temps de la dernière connexion

| *Attribut* | *Valeur* | *Description* |
|---|---|---|
| **Requête** | | |
| URL | /api/authenticate/last/logout | Demander sa dernière déconnexion |
| Type de requête | GET | |
| **En-tête de la requête** | | |
| authorization | jsonwebtoken: string | |
| **Corps de la réponse** | | |
| lastLogout | number | |
| **Statuts de réponse** | | |
| Si réponse valide | 200 | Ok |
| Si pas d'historique | 404 | Not found |
| Si mauvaise requête | 400 | Bad request |

### *4.3.4.Statistiques*

4.3.4.1. Récupérer les statistiques de l'utilisateur

| *Attribut* | *Valeur* | *Description* |
|---|---|---|
| **Requête** | | |
| URL | /api/stats/private | Demander ses propres statistiques de jeu |
| Type de requête | GET | |
| **En-tête de la requête** | | |
| authorization | jsonwebtoken: string | |
| **Corps de la réponse** | | |
| privateStats | PrivateStats | |
| name | string | |
| surname | string | |
| logs | Log[] | |
| games | Game[] | |
| **Statuts de réponse** | | |
| Si réponse valide | 200 | Ok |
| Si autorisation échoue | 401 | Unauthorized |
| Si mauvaise requête | 400 | Bad request |

### *4.3.5. Tableau de classement*

4.3.5.1. Demander les statistiques des 10 meilleurs joueurs

| *Attribut* | *Valeur* | *Description* |
|---|---|---|
| **Requête** | | |
| URL | /api/stats/leaderboard/:category | Demander les statistiques des 10 meilleurs joueurs pour une catégorie spécifique |
| Type de requête | GET | |
| **Paramètres de l'URL** | | |
| category | string | |
| **En-tête de la requête** | | |
| authorization | jsonwebtoken: string | |
| **Corps de la réponse** | | |
| publicStats | PublicStats[] | |
| **Statuts de réponse** | | |
| Si réponse valide | 200 | Ok |
| Si autorisation échoue | 401 | Unauthorized |
| Si mauvaise requête | 400 | Bad request |

### *4.3.6. Fil d'actualité*

4.3.6.1. Demander les dessins du fil d'actualité

| *Attribut* | *Valeur* | *Description* |
|---|---|---|
| **Requête** | | |
| URL | /api/drawing/feed/info | Demander les URL des dessins du fil d'images |
| Type de requête | GET | |
| **En-tête de la requête** | | |
| authorization | jsonwebtoken: string | |
| **Corps de la réponse** | | |
| drawings | string[] | |
| **Statuts de réponse** | | |
| Si réponse valide | 200 | Ok |
| Si autorisation échoue | 401 | Unauthorized |
| Si mauvaise requête | 400 | Bad request |

### *4.3.7. Parties*

4.3.7.1. Demander les parties publiques

| *Attribut* | *Valeur* | *Description* |
|---|---|---|
| **Requête** | | |
| URL | /api/games/list | Demander la liste des parties publiques en attente |
| Type de requête | GET | |
| **En-tête de la requête** | | |
| authorization | jsonwebtoken: string | |
| **Corps de la réponse** | | |
| games | Game[] | |
| **Statuts de réponse** | | |
| Si réponse valide | 200 | Ok |
| Si autorisation échoue | 401 | Unauthorized |
| Si mauvaise requête | 400 | Bad request |

4.3.7.2. Commencer une partie

| *Attribut* | *Valeur* | *Description* |
|---|---|---|
| **Requête** | | |
| URL | /api/games/start | Débuter une partie |
| Type de requête | POST | |
| **En-tête de la requête** | | |
| authorization | jsonwebtoken: string | |
| **Corps de la requête** | | |
| lobbyId | string | |
| **Statuts de réponse** | | |
| Si réponse valide | 200 | Ok |
| Si autorisation échoue | 401 | Unauthorized |
| Si mauvaise requête | 400 | Bad request |
| Si pas assez de jouers | 406 | Not acceptable |

4.3.7.3. Créer une partie publique

| *Attribut* | *Valeur* | *Description* |
|---|---|---|
| **Requête** | | |
| URL | /api/games/create/public | Créer une nouvelle partie publique. |
| Type de requête | POST | |
| **En-tête de la requête** | | |
| authorization | jsonwebtoken: string | |
| **Corps de la requête** | | |
| gameType | string | |
| difficulty | string | |
| gameName | string | |
| **Corps de la réponse** | | |
| lobbyId | string | |
| **Statuts de réponse** | | |
| Si réponse valide | 200 | Ok |
| Si autorisation échoue | 401 | Unauthorized |
| Si mauvaise requête | 400 | Bad request |

4.3.7.4. Créer une partie privée

| *Attribut* | *Valeur* | *Description* |
|---|---|---|
| **Requête** | | |
| URL | /api/games/create/privée | Créer une nouvelle partie privée. |
| Type de requête | POST | |
| **En-tête de la requête** | | |
| authorization | jsonwebtoken: string | |
| **Corps de la requête** | | |
| gameType | string | |
| difficulty | string | |
| gameName | string | |
| **Corps de la réponse** | | |
| lobbyId | string | |
| lobbyInviteId | string | |
| **Status de réponse** | | |
| Si réponse valide | 200 | Ok |
| Si autorisation échoue | 401 | Unauthorized |
| Si mauvaise requête | 400 | Bad request |

4.3.7.5. Rejoindre une partie publique

| *Attribut* | *Valeur* | *Description* |
|---|---|---|
| **Requête** | | |
| URL | /api/games/join/public | Rejoindre une nouvelle partie publique |
| Type de requête | POST | |
| **En-tête de la requête** | | |
| authorization | jsonwebtoken: string | |
| **Corps de la requête** | | |
| lobbyId | string | |
| **Statuts de réponse** | | |
| Si réponse valide | 200 | Ok |
| Si autorisation échoue | 401 | Unauthorized |
| Si lobbyId n’existe pas | 404 | Lobby does not exist |
| Si partie pleine ou déjà en cours | 406 | Lobby full or game started |

4.3.7.6. Rejoindre une partie privée

| *Attribut* | *Valeur* | *Description* |
|---|---|---|
| **Requête** | | |
| URL | /api/games/join/private | Rejoindre une nouvelle partie privée |
| Type de requête | POST | |
| **En-tête de la requête** | | |
| authorization | jsonwebtoken: string | |
| **Corps de la requête** | | |
| lobbyInviteId | string | |
| **Corps de la réponse** | | |
| lobbyId | string | |
| **Statuts de réponse** | | |
| Si réponse valide | 200 | Ok |
| Si autorisation échoue | 401 | Unauthorized |
| Si lobbyId n’existe pas | 404 | Lobby does not exist |
| Si partie pleine ou déjà en cours | 406 | Lobby full or game started |

4.3.7.7. Quitter une partie

| *Attribut* | *Valeur* | *Description* |
|---|---|---|
| **Requête** | | |
| URL | /api/games/leave | Quitter une partie |
| Type de requête | DELETE | |
| **En-tête de la requête** | | |
| authorization | jsonwebtoken: string | |
| **Params de la requête** | | |
| lobbyId | string | |
| **Statuts de réponse** | | |
| Si réponse valide | 200 | Ok |
| Si autorisation échoue | 401 | Unauthorized |
| Si lobbyId n'existe pas | 404 | Lobby does not exist |
| Si le joueur n'est pas dans la partie | 406 | le joueur n'est pas dans la partie |

4.3.7.8. Ajouter un joueur virtuel

| *Attribut* | *Valeur* | *Description* |
|---|---|---|
| **Requête** | | |
| URL | /api/games/add/virtual/player | Ajouter un joueur virtuel à la partie |
| Type de requête | POST | |
| **En-tête de la requête** | | |
| authorization | jsonwebtoken: string | |
| **Corps de la requête** | | |
| lobbyId | string | |
| teamNumber | number | |
| **Statuts de réponse** | | |
| Si réponse valide | 200 | Ok |
| Si autorisation échoue | 401 | Unauthorized |
| Si mauvaise requête | 400 | Bad request |
| Si partie pleine ou déjà en cours | 406 | Partie pleine ou déjà en cours |

4.3.7.9. Retirer un joueur virtuel

| *Attribut* | *Valeur* | *Description* |
|---|---|---|
| **Requête** | | |
| URL | /api/games/remove/virtual/player | Enlever un joueur virtuel à la partie |
| Type de requête | DELETE | |
| **En-tête de la requête** | | |
| authorization | jsonwebtoken: string | |
| **Params de la requête** | | |
| lobbyID | string | |
| teamNumber | number | |
| username | string | |
| **Status de réponse** | | |
| Si réponse valide | 200 | Ok |
| Si autorisation échoue | 401 | Unauthorized |
| Si mauvaise requête | 400 | Bad request |
| Si joueur n'est plus dans la partie | 404 | Not Found |

4.3.7.10. Envoyer le dessin de la venant de se terminer

| *Attribut* | *Valeur* | *Description* |
|---|---|---|
| **Requête** | | |
| URL | /api/games/upload | Envoyer le dessin que l'utilisateur vient de terminer au serveur |
| Type de requête | POST | |
| **En-tête de la requête** | | |
| authorization | jsonwebtoken: string | |
| **Corps de la requête** | | |
| imageUrl | string | |
| gameId | string | |
| **Statuts de réponse** | | |
| Si réponse valide | 200 | Ok |
| Si autorisation échoue | 401 | Unauthorized |
| Si mauvaise requête | 400 | Bad request |

### *4.3.8. Votes sur dessin du joueur virtuel*

4.3.8.1. Ajouter un vote

| *Attribut* | *Valeur* | *Description* |
|---|---|---|
| **Requête** | | |
| URL | /api/drawing/vote | Ajouter un vote à un dessin |
| Type de requête | PATCH | |
| **En-tête de la requête** | | |
| authorization | jsonwebtoken: string | |
| **Corps de la requête** | | |
| isUpvote | boolean | |
| drawingId | string | |
| **Statuts de réponse** | | |
| Si réponse valide | 200 | Ok |
| Si autorisation échoue | 401 | Unauthorized |
| Si mauvaise requête | 400 | Bad request |

### *4.3.9.Création paire mot-image*

4.3.9.1. Ajouter une paire mot-image

| *Attribut* | *Valeur* | *Description* |
|---|---|---|
| **Requête** | | |
| URL | /api/drawing/create | Ajouter une paire mot-image à la banque d'images |
| Type de requête | POST | |
| **En-tête de la requête** | | |
| authorization | jsonwebtoken: string | |
| **Corps de la requête** | | |
| strokes | Stroke[] | liste des traits qui forment le dessin |
| drawingName | string | Le mot qui représente le dessin |
| difficulty | string | |
| hints | string[] | |
| **Statuts de réponse** | | |
| Si réponse valide | 200 | Ok |
| Si autorisation échoue | 401 | Unauthorized |
| Si mauvaise requête | 400 | Bad request |