Équipe 207

Fais-moi un dessin
Protocole de communication

Version 1.5

Historique des révisions

Date	Version	Description	Auteur
2021-02-03	1.0	Structure de base du document	Marc-Olivier Riopel
	1.1		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

Table des matières

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, chronomètre, 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

1.1. Vec2

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

1.2. Stroke

Attribut	Type	Description
points	Vec2[]	Tous les points du trait
color	string	Couleur du trait
width	number	Épaisseur du trait

1.3. Clavardage

Attribut	Type	Description
chatName	string	Nom du canal de discussion
chatID	uuid	Identifiant unique du canal de discussion
users	PublicPlayer[]	Liste des utilisateurs dans le canal de discussion

1.4. Message

Attribut	Type	Description
username	string	Pseudo de l'utilisateur
message	string	Message envoyé
messageType	string	Type de message envoyé
timestamp	Date	Date du message

1.5. PublicStats

Attribut	Type	Description
avatar	svg	Avatar de l'utilisateur
username	string	Pseudo de l'utilisateur
sprintSoloHighScore	number	Meilleur score obtenu en sprint solo
winRatio	number	Ratio des parties gagnées/parties perdues en mode classique
wins	number	Nombre de parties classique gagnées
gamesPlayed	number	Nombre de parties jouées
sprintCoopHighScore	number	Meilleur score obtenu en sprint coop
drawingUpvotes	number	Score total des upvotes - downvotes sur les paires mot-image créées

1.6. PrivateStats

Attribut	Type	Description
avatar	svg	Avatar de l'utilisateur
username	string	Pseudo de l'utilisateur
sprintSoloHighScore	number	Meilleur score obtenu en sprint solo
winRatio	number	Ratio des partie gagné/parties perdues en mode classique
wins	number	Nombre de parties classique gagnées
gamesPlayed	number	Nombre de parties jouées
sprintCoopHighScore	number	Meilleur score obtenu en sprint coop
totalAppTime	number	Durée total du temps passé sur l'application
averageTimePerGame	number	Durée moyenne d'une partie classique

1.7. Log

Attribut	Type	Description
logType	string	Peut être soit un login ou un logout
timestamp	Date	Date du login/logout

1.8. GameInfo

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

1.9. PublicPlayer

Attribut	Type	Description
avatar	svg	Avatar de l'utilisateur
username	string	Pseudo de l'utilisateur

1.10. PrivatePlayer

Attribut	Type	Description
avatar	svg	Avatar de l'utilisateur
username	string	Pseudo de l'utilisateur
name	string	Nom de l'utilisateur
surname	string	Prénom de l'utilisateur

1.11. Game

Attribut	Type	Description
players	PublicPlayer[]	Les joueurs de la partie
gameType	string	Mode de jeu
difficulty	string	Difficulté
gameID	uuid	Identifiant unique de la partie
gameName	string	Nom de la partie
isPrivate	boolean	Accès des joueurs à la partie

1.12. Vote

Attribut	Type	Description
drawingID	uuid	Identifiant unique du dessin
isUpvote	boolean	Indication s'il s'agit d'un vote positif ou négatif

1.13. Drawing

Attribut	Type	Description
drawingID	uuid	Identifiant unique du dessin
image	png	Image du dessin demandé
drawingVotes	number	Score du dessin actuel

4. Description des packets

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

1.14. Client vers serveur avec Socket

1.14.1. Clavardage

Nom de la requête	Description	Attributs
addChatMessage	Ajouter un message dans un canal de discussion	{jsonwebtoken: string, chatID: uuid, message: string}
leaveChatRoom	Quitter un canal de discussion	{jsonwebtoken: string, chatID: uuid}
joinChatRoom	Rejoindre un canal de discussion existant	{jsonwebtoken: string, chatID: uuid}

1.14.2. Dessins

Nom de la requête	Description	Attributs
updateDrawing	Envoie du dessin en cours mis à jour vers le serveur	{jsonwebtoken: string, strokes: Stroke[], gameID: uuid}
guessDrawing	Tenter de deviner un dessin	{jsonwebtoken: string, guessedWord: string, gameID: uuid}

1.15. Serveur vers client avec Socket

1.15.1. Clavardage

Nom de la requête	Description	Attributs
dispatchMessage	Envoyer le message à tous les clients du canal de discussion	{chatID: uuid, message: string}
dispatchInvitation	Envoyer une invitation pour une partie privée dans un canal de discussion	{gameUrl: string, gameID: uuid}
dispatchNewChat	Notifier le client qu'il a été ajouté à un nouveau canal de discussion	{chatID: uuid, chatName: string}

1.15.2. Dessins

Nom de la requête	Description	Attributs
dispatchDrawing	Envoie du dessin en cours mis à jour vers les clients	{strokes: Stroke[], gameID: uuid}
dispatchCorrectGuess	Envoyer un message qui indique que le joueur a bien deviné	{gameID: uuid, username: string}
dispatchWrongGuess	Envoyer un message qui indique que le joueur a mal deviné	{gameID: uuid, username: string, guess: string}

1.15.3. Fil d'actualité

Nom de la requête	Description	Attributs
dispatchNewFeaturedDrawing	Envoyer un dessin du fil d'actualité	{image: string}

1.15.4. Liens d'amitié

Nom de la requête	Description	Attributs
dispatchFriendInvite	Notifier le client lors d'une nouvelle demande d'ami	{username: string}

1.15.5. Parties

Nom de la requête	Description	Attributs
dispatchGameWinner	Notifier les joueurs d'une partie du vainqueur de celle-ci	{username: string}
dispatchNewPlayer	Notifier les joueurs que quelqu'un vient de rejoindre la partie	{username: string, avatar: svg}

1.16. Communication client-serveur avec REST API

1.16.1. Clavardage

1.16.1.1. 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		
auth	jsonwebtoken: string	
Corps de la réponse		
chat	Chat[]	
Status de réponse		
Si réponse valide	200	Ok
Si autorisation échoue	401	Unauthorized
Si mauvaise requête	400	Bad request

1.16.1.2. Création d'un nouveau canal de discussion

Attribut	*Valeur*	*Description*
Requête		
URL	/api/chat/create	Créer un nouveau canal de discussion. On peut ajouter des amis au canal en ajoutant leur username dans la requête
Type de requête	POST	
En-tête de la requête		
auth	jsonwebtoken: string	
Corps de la requête		
chatName	string	
usernames	string[]	
Corps de la réponse		
chatID	uuid	
Status de réponse		
Si réponse valide	200	Ok
Si autorisation échoue	401	Unauthorized
Si chatName déjà pris	409	Conflict
Si mauvaise requête	400	Bad request

1.16.1.3. Supprimer un canal de discussion

Attribut	*Valeur*	*Description*
Requête		
URL	/api/chat/:chatID	Supprimer un canal de discussion
Type de requête	DELETE	
Paramètres du url		
chatID	uuid	
En-tête de la requête		
auth	jsonwebtoken: string	
Status 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

1.16.1.4. Historique d'un canal de discussion

Attribut	*Valeur*	*Description*
Requête		
URL	/api/chat/history/:chatID	Demander l'historique d'un canal de discussion
Type de requête	GET	
Paramètres du url		
chatID	uuid	
En-tête de la requête		
auth	jsonwebtoken: string	
Corps de la réponse		
messages	Message[]	
Status de réponse		
Si réponse valide	200	Ok
Si autorisation échoue	401	Unauthorized
Si mauvaise requête	400	bad request

1.16.2. Dessins

1.16.2.1. Demander un indice pour le dessin

Attribut	*Valeur*	*Description*
Requête		
URL	/api/drawing/hint/:drawingID/:hintID	Demander un indice pour le dessin
Type de requête	GET	
Paramètres du url		
drawingID	uuid	
hintNumber	number	
En-tête de la requête		
auth	jsonwebtoken: string	
Corps de la réponse		
hint	string	
Status de réponse		
Si réponse valide	200	Ok
Si autorisation échoue	401	Unauthorized
Si indice pas trouvé	406	Il n’y plus d’indice disponible
Si mauvaise requête	400	Bad request

1.16.2.2. Demander des suggestions de mot

Attribut	*Valeur*	*Description*
Requête		
URL	/api/drawing/suggestions/:gameID	Demander des suggestions de mot pour le dessin
Type de requête	GET	
Paramètres du url		
gameID	uuid	
En-tête de la requête		
auth	jsonwebtoken: string	
Corps de la réponse		
wordSuggestions	string[]	
Status de réponse		
Si réponse valide	200	Ok
Si autorisation échoue	401	Unauthorized
Si mauvaise requête	400	bad request

1.16.2.3. Envoyer le mot choisi

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

1.16.2.4. Récupérer des dessins

Attribut	*Valeur*	*Description*
Requête		
URL	/api/drawing/:gameID	Récupérer la liste des dessins utilisés par le joueur virtuel dans la partie
Type de requête	GET	
En-tête de la requête		
auth	jsonwebtoken: string	
Corps de la requête		
gameID	uuid	
Corps de la réponse		
drawings	Drawing[]	
Status de réponse		
Si réponse valide	200	Ok
Si autorisation échoue	401	Unauthorized
Si mauvaise requête	400	bad request

1.16.4.Authentification

1.16.4.1. 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	
Status de réponse		
Si réponse valide	200	Ok
Si mauvais username ou mot de passe	404	Not found
Si mauvaise requête	400	Bad request

1.16.4.2. Déconnecter un utilisateur de l'application

Attribut	*Valeur*	*Description*
Requête		
URL	/api/authenticate/logout	Déconnexion de l'application
Type de requête	DELETE	
En-tête de la requête		
auth	jsonwebtoken: string	
Status de réponse		
Si réponse valide	200	Ok
Si mauvaise requête	400	Bad request

1.16.4.3. 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	svg	
Corps de la réponse		
jsonwebtoken	string	
Status de réponse		
Si réponse valide	200	Ok
Si username déjà pris	409	Conflict
Si mauvaise requête	400	Bad request

1.16.5. Statistiques

1.16.5.1. Demander les statistiques d'un autre joueur

Attribut	*Valeur*	*Description*
Requête		
URL	/api/stats/public/:username	Demander les statistiques de jeu d'un autre joueur
Type de requête	GET	
Paramètres du url		
username	string	
En-tête de la requête		
auth	jsonwebtoken: string	
Corps de la réponse		
publicStats	PublicStats	
Status de réponse		
Si réponse valide	200	Ok
Si autorisation échoue	401	Unauthorized
Si mauvaise requête	400	Bad request

1.16.5.2. 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		
auth	jsonwebtoken: string	
Corps de la réponse		
privateStats	PrivateStats	
Status de réponse		
Si réponse valide	200	Ok
Si autorisation échoue	401	Unauthorized
Si mauvaise requête	400	Bad request

1.16.5.3. Demander son historique de connexion

Attribut	*Valeur*	*Description*
Requête		
URL	/api/stats/logs	Demander son historique de connexion
Type de requête	GET	
En-tête de la requête		
auth	jsonwebtoken: string	
Corps de la réponse		
logs	Log[]	
Status de réponse		
Si réponse valide	200	Ok
Si autorisation échoue	401	Unauthorized
Si mauvaise requête	400	Bad request

1.16.5.4. Demander son historique de parties

Attribut	*Valeur*	*Description*
Requête		
URL	/api/stats/games	Demander son historique de parties
Type de requête	GET	
En-tête de la requête		
auth	jsonwebtoken: string	
Corps de la réponse		
gameInfo	GameInfo[]	
Status de réponse		
Si réponse valide	200	Ok
Si autorisation échoue	401	Unauthorized
Si mauvaise requête	400	Bad request

1.16.6. Informations des utilisateurs

1.16.6.1. Demander les informations de base d'un autre joueur

Attribut	*Valeur*	*Description*
Requête		
URL	/api/user/public/info/:username	Demander les informations de base d'un autre joueur
Type de requête	GET	
Paramètres du url		
username	string	
En-tête de la requête		
auth	jsonwebtoken: string	
Corps de la réponse		
publicPlayer	PublicPlayer	
Status de réponse		
Si réponse valide	200	Ok
Si autorisation échoue	401	Unauthorized
Si mauvaise requête	400	Bad request

1.16.6.2. Demander ses informations de base

Attribut	*Valeur*	*Description*
Requête		
URL	/api/user/private/info	Demander ses propres informations de base
Type de requête	GET	
En-tête de la requête		
auth	jsonwebtoken: string	
Corps de la réponse		
privatePlayer	PrivatePlayer	
Status de réponse		
Si réponse valide	200	Ok
Si autorisation échoue	401	Unauthorized
Si mauvaise requête	400	Bad request

1.16.6.3. Modifier ses informations de base

Attribut	*Valeur*	*Description*
Requête		
URL	/api/user/private/info	Modifier les informations de base sur le profil de l'utilisateur
Type de requête	PATCH	
En-tête de la requête		
auth	jsonwebtoken: string	
Corps de la requête		
privatePlayer	PrivatePlayer	
Status de réponse		
Si réponse valide	200	Ok
Si autorisation échoue	401	Unauthorized
Si mauvaise requête	400	Bad request

1.16.6.4. Mettre son mot de passe à jour

Attribut	*Valeur*	*Description*
Requête		
URL	/api/user/private/password	Mettre le mot de passe à jour
Type de requête	PATCH	
En-tête de la requête		
auth	jsonwebtoken: string	
Corps de la requête		
password	string	
Status de réponse		
Si réponse valide	200	Ok
Si autorisation échoue	401	Unauthorized
Si mauvaise requête	400	Bad request

1.16.7. Tableau de classement

1.16.7.1. Demander les statistiques des 10 meilleurs joueurs

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

1.16.8.Fil d'actualité

1.16.8.1. Demander les dessins du fil d'actualité

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

1.16.9. Parties

1.16.9.1. 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		
auth	jsonwebtoken: string	
Corps de la réponse		
games	Game[]	
Status de réponse		
Si réponse valide	200	Ok
Si autorisation échoue	401	Unauthorized
Si mauvaise requête	400	Bad request

1.16.9.2. Créer une partie

Attribut	*Valeur*	*Description*
Requête		
URL	/api/games/create	Créer une nouvelle partie. On peut inviter des amis en même temps au besoin.
Type de requête	POST	
En-tête de la requête		
auth	jsonwebtoken: string	
Corps de la requête		
gameType	string	
difficulty	string	
gameName	string	
isPrivate	boolean	
usernames	string[]	
Corps de la réponse		
gameID	uuid	
Status de réponse		
Si réponse valide	200	Ok
Si autorisation échoue	401	Unauthorized
Si mauvaise requête	400	Bad request

1.16.9.3. Rejoindre une partie

Attribut	*Valeur*	*Description*
Requête		
URL	/api/games/join	Rejoindre une nouvelle partie
Type de requête	POST	
En-tête de la requête		
auth	jsonwebtoken: string	
Corps de la requête		
gameID	uuid	
Status 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

1.16.9.4. 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		
auth	jsonwebtoken: string	
Corps de la requête		
gameID	uuid	
Status 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

1.16.10. Votes sur dessin du joueur virtuel

1.16.10.1. 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		
auth	jsonwebtoken: string	
Corps de la requête		
votes	Vote[]	
gameID	uuid	
Status de réponse		
Si réponse valide	200	Ok
Si autorisation échoue	401	Unauthorized
Si mauvaise requête	400	Bad request

1.16.11.Création paire mot-image

1.16.11.1. 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		
auth	jsonwebtoken: string	
Corps de la requête		
drawing	Stroke[]	liste des traits qui forment le dessin
drawingName	string	Le mot qui représente le dessin
difficulty	string	
hints	string[]	
Status de réponse		
Si réponse valide	200	Ok
Si autorisation échoue	401	Unauthorized
Si mauvaise requête	400	Bad request

1.16.12.Liens d'amitié

1.16.12.1. Envoyer une demande d'amitié

Attribut	*Valeur*	*Description*
Requête		
URL	/api/friends/request	Envoyer une demande d'amitié à un joueur
Type de requête	POST	
En-tête de la requête		
auth	jsonwebtoken: string	
Corps de la requête		
username	string	
Status de réponse		
Si réponse valide	200	Ok
Si autorisation échoue	401	Unauthorized
Si mauvaise requête	400	Bad request
Si username n'existe pas	406	Ce pseudonyme n'existe pas

1.16.12.2. Supprimer un ami

Attribut	*Valeur*	*Description*
Requête		
URL	/api/friends/remove/:username	Supprimer un ami
Type de requête	DELETE	
Corps du url		
username	string	
En-tête de la requête		
auth	jsonwebtoken: string	
Status de réponse		
Si réponse valide	200	Ok
Si autorisation échoue	401	Unauthorized
Si mauvaise requête	400	Bad request
Si username n'existe pas dans liste amis	406	Vous n'avez pas d'ami avec ce pseudonyme

1.16.12.3. Demander sa liste d’amis

Attribut	*Valeur*	*Description*
Requête		
URL	/api/friends/list	Demander sa liste d’amis
Type de requête	GET	
En-tête de la requête		
auth	jsonwebtoken: string	
Corps de la réponse		
publicPlayer	PublicPlayer	
Status de réponse		
Si réponse valide	200	Ok
Si autorisation échoue	401	Unauthorized
Si mauvaise requête	400	Bad request

1.16.12.4. Répondre à une demande d’amitié

Attribut	*Valeur*	*Description*
Requête		
URL	/api/friends/response	Envoyer la réponse d’une demande d’amitié
Type de requête	POST	
En-tête de la requête		
auth	jsonwebtoken: string	
Corps de la requête		
username	string	
isAccepted	boolean	
Status de réponse		
Si réponse valide	200	Ok
Si autorisation échoue	401	Unauthorized
Si mauvaise requête	400	Bad request