Équipe 210

Draw It Up!
Protocole de communication

Version 1.6

Historique des révisions

Date	Version	Description	Auteur
2021-02-12	1.0	Écriture de l'introduction	Marc-André Filion
2021-02-13	1.1	Écriture des sections "communication client-serveur" et "description des paquets"	Benjamin Saint-Cyr, Marc-André Filion
2021-02-14	1.2	Ajout de la section "communication serveur-base de données"	Benjamin Saint-Cyr
2021-02-17	1.3	Ajout de complément d'information pour la section "communication client-serveur"	Marc-André Filion
2021-02-18	1.4	Dernières modifications et révision finale en équipe avant la remise.	Équipe 210
2021-03-19	1.5	Mise à jours de l'information	Benjamin Saint-Cyr
2021-04-19	1.6	Révision finale de l'équipe avant la remise	Équipe 210

Table des matières

Protocole de communication

1. Introduction

Le projet *Fais-moi un dessin* étant un jeu multijoueur en ligne, des technologies de communication adéquates sont requises pour son développement. Le présent document fournit les renseignements décrivant d'abord le moyen de communication pour le réseau client-serveur, le contenu des paquets utilisés pour le transfert de données, puis comment le serveur va communiquer avec la base de données.

2. Communication client-serveur

Afin d'établir une connexion entre les différents clients et le serveur, les Websockets seront d'abord utilisés afin de former des canaux de communication bidirectionnels sous Transmission Control Protocol (TCP). Les clients pourront notamment recevoir de cette façon des mises à jour concernant l'état du serveur sans besoin d'envoyer de requêtes. Pour chaque client connecté, trois connexions Websockets seront établies, une pour le clavardage, une pour la transmission de données permettant l'affichage des dessins en continu et une autre pour établir une session de connexion pour les utilisateurs. Le serveur pourra être rejoint au nom de domaine *drawitup.ddns.net* au port 80, soit HTTP pour la règle entrante.

Pour toutes les autres fonctionnalités, nous utilisons des appels avec REST API, celles-ci ne nécessitant pas les bénéfices des Websockets. La section 3.2 porte sur les requêtes HTTP et décrit quelles sont les fonctionnalités touchées.

3. Description des paquets

Notre protocole de communication transférera des paquets sous forme de chaînes de caractères (*string*) contenant des données en format JSON.

Requêtes (voir page suivante):

3.1 Interfaces connues

User

Attribut	**Valeur**	**Description**
email	string	Courriel de l'utilisateur
pseudo	string	Pseudonyme de l'utilisateur
lastName	string	Prénom de l'utilisateur
firstName	string	Nom de famille de l'utilisateur
password	string	Mot de passe de l'utilisateur

UserPublicInformation

Attribut	**Valeur**	**Description**
pseudo	string	Le pseudonym unique de l'utilisateur
avatar	string	Chemin vers l'avatar choisi par l'utilisateur

UserPrivateInformation

Attribut	**Valeur**	**Description**
lastName	string	Nom de famille de l'utilisateur
firstName	string	Prénom de l'utilisateur
password	string	Mot de passe encrypté de l'utilisateur

UserStatistics

Attribut	**Valeur**	**Description**
games	number	Nombre de parties jouées
winRatio	number	Ratio victoire/défaites
wins	number	Nombre de victoires
losts	number	Nombre de défaites
averageTime	string	Temps moyen d'une partie
totalTime	string	Temps total des parties

BasicGameHistory

Attribut	**Valeur**	**Description**
start	Date	Début de la partie
end	Date	Fin de la partie

ClassicGameHistory

Attribut	**Valeur**	**Description**
basicGameHistory	BasicGameHistory	N/A
players	string[]	Liste des joueurs dans une partie
result	map[string]bool	Associe chaque joueur à un booléen qui dicte une victoire ou défaite

SoloGameHistory

Attribut	**Valeur**	**Description**
basicGameHistory	BasicGameHistory	N/A
result	number	Score de l'utilisateur durant la partie

UserHistory

Attribut	**Valeur**	**Description**
creationDate	Date	La date quand l'utilisateur a créé son compte
connections	Date[]	Les dates de connexion
deconnections	Date[]	Les dates de déconnexion
gameHistory	DeadGameHistory	Toutes les parties qui ont déjà été jouées
chatChannels	number[]	ID des salons de clavardage

DeadGameHistory

Attribut	**Valeur**	**Description**
Classic	ClassicGameHistory[]	Liste des parties classiques que l'utilisateur a jouée
Solo	SoloGameHistory[]	Liste de parties solos que l'utilisateur a jouées

UserProfile

Attribut	**Valeur**	**Description**
email	string	Courriel de l'utilisateur
public	UserPublicInformation	L'information publique de l'utilisateur
private	UserPrivateInformation	L'information privée de l'utilisateur
statistics	UserStatistics	Les statistiques de l'utilisateur
history	UserHistory	L'historique des parties de l'utilisateur
trophies	string[]	Les trophées que l'utilisateur a gagnés

Message

Attribut	**Valeur**	**Description**
mail	string	Courriel de l'utilisateur qui a envoyé le message
username	string	Pseudonyme de l'utilisateur qui a envoyé le message
message	string	Le texte du message
sendDate	string	Date quand le serveur a distribué le message

GameInfo

Attribut	**Valeur**	**Description**
gameType	string	Type de partie
Difficulty	string	Difficulté de la partie
HumanPlayers	number	Nombre de joueurs humains
VirtualPlayers	number	Nombre de joueurs virtuels
GameID	number	ID de la partie
ChatID	number	ID du salon de clavardage associé à la partie
State	string	Indique si la partie est en cours, en attente ou terminée
WordToGuess	string	Le mot à deviner présentement dans la partie
Teams	TeamGame[]	Les équipes de la partie
Drawer	string	Quel utilisateur dessine

TeamGame

Attribut	**Valeur**	**Description**
Players	string[]	Courriels des joueurs des membres de l'équipe
IsCurrentlyPlaying	boolean	Indique si l'équipe tient le crayon ou non
Score	number	Nombre de dessins devinés par l'équipe

WordImage

Attribut	**Valeur**	**Description**
word	string	Le mot à deviner
hints	string[]	Indices sur le mot à deviner
image	Stroke[]	Tous les traits du dessin dans l'ordre à dessiner
difficulty	Difficulty	La difficulté

Stroke

Attribut	**Valeur**	**Description**
pathData	Vec2[]	Les points pour reconstruire le dessin sur le canvas
color	string	La couleur du trait
size	number	La taille du trait
opacity	number	L'opacité du trait

Vec2

Attribut	**Valeur**	**Description**
x	number	La coordonnée X pour le canvas
y	number	La coordonnée Y pour le canvas

Difficulty

Attribut	**Valeur**	**Description**
Easy	'Facile'	Indique une difficulté associée à un dessin ou une partie
Medium	'Moyen'	Indique une difficulté associée à un dessin ou une partie
Hard	'Difficile'	Indique une difficulté associée à un dessin ou une partie

Notifications

Tous les types de notifications que le serveur peut envoyer aux clients.

Attribut	**Valeur**	**Description**
StartGame	"startGame"	Quand une partie que le joueur a rejoint commence
StartTurn	"startTurn"	Quand un tour commence
CorrectAnswer	"correctAnswer"	Quand la réponse de l'utilisateur est correcte
WrongAnswer	"wrongAnswer"	Quand la réponse est mauvaise
EndTurn	"endTurn"	Quand la manche finit
EndGame	"endGame"	Quand la partie finit
Disconnected	"disconnected"	Quand quelqu'un quitte en cours de partie
GameLobbyStateUpdate	"gameLobbyStateUpdate"	Quand quelqu'un rejoint ou quitte un salon de partie
YouGuess	"youGuess"	Quand c'est au tour de l'utilisateur de deviner
YouDraw	"youDraw"	Quand c'est au tour de l'utilisateur de dessiner
NotYourTurn	"notYourTurn"	Quand ce n'est pas le tour de l'utilisateur
RightOfReply	"rightOfReply"	Quand il y a droit de réplique
YouWon	"youWon"	Quand l'utilisateur gagne
YouLost	"youLost"	Quand l'utilisateur perd

Register

Attribut	**Valeur**	**Description**
Email	string	Le courriel de l'utilisateur qui sera utilisé pour son profil
Pseudo	string	Le pseudonyme de l'utilisateur qui sera utilisé pour son profil
LastName	string	Le nom de famille de l'utilisateur qui sera utilisé pour son profil
FirstName	string	Le prénom de l'utilisateur qui sera utilisé pour son profil
Password	string	Le mot de passe de l'utilisateur qui sera utilisé pour son profil
Avatar	string	L'avatar de l'utilisateur qui sera utilisé pour son profil

Login

Attribut	**Valeur**	**Description**
Email	string	Le courriel d'un utilisateur qu'il utilisera pour se connecter à son compte
Password	string	Le mot de passe d'un utilisateur qu'il utilisera pour se connecter à son compte

3.2 Requêtes HTTP

Requêtes de compte

Type de requête	**URL**	**Description**	**Paramètres**	**Réponse [code status http]**
POST	/user/login	Se connecter à son compte	"email": string "password": string	[200] (si le mot de passe est valide) [401] (si le mot de passe n'est pas valide)
POST	/user/register	Le système enregistre le compte de l'utilisateur dans la base de données	User	[200] mongoDB user ID (si l'utilisateur est enregistré dans le serveur) [400] (si ● le email est déjà enregistré ● le pseudo contient de la profanité) [503] (si la base de données n'est pas accessible) [507] (si la base de données est pleine)

Requêtes d'information

Type de requête	**URL**	**Description**	**Paramètres**	**Réponse [code status http]**
GET	/user/:email	Donne l'information de profile sur l'utilisateur	email: string	[200] UserProfile [500] "L'utilisateur est introuvable"
GET	info/chats	Retourne toute l'information sur tous les chats		[200] ChatInfo[]
GET	/info/chat/:chatID	Donne l'information sur un chat particulier	chatID: number	ChatInfo
GET	/info/games	Retourne l'information sur toute les parties active sur le serveur		[200] GameInfo[]
GET	/info/game/:gameID	Retourne l'information spécifique pour une partie active sur le serveur	gameID: number	[200] GameInfo [400] "ID mal écrit" [400] "Cette partie n'existe pas"

Requêtes pour le classement

Type de requête	**URL**	**Description**	**Paramètres**	**Réponse [code status http]**
GET	/leaderboard/game	Donne le classement des joueurs selon le nombre de partie jouée		[200] "_id":string, "email": string, "statistics": {"games": number}
GET	/leaderboard/win-rate	Donne le classement des joueurs selon le ratio de victoire		[200] "_id":string, "email": string, "statistics": {"winratio": number}
GET	/leaderboard/average-time	Donne le classement des joueurs selon le plus grand temps jouée en moyenne		[200] "_id":string, "email": string, "statistics": {"averageTime": number}
GET	/leaderboard/most-time	Donne le classement des joueurs selon le plus grand temps jouée au total		[200] "_id":string, "email": string, "statistics": {"totalTime": number}

Requêtes sur les parties et les chats

Type de requête	**URL**	**Description**	**Paramètres**	**Réponse [code status http]**
POST	/new/game	Crée une nouvelle partie dans le serveur	"type" : string "difficulty" : string	[200] gameID: number [500] (Si le serveur n'arrive pas à créer la partie)
POST	/new/chat	Crée un nouveau chat avec un nom et des nouveau membres	"name" : string "members" : string[]	[200] chatID: number [500] (Si le serveur n'arrive pas à créer le chat)
POST	/chat/rename	Renomme le chat spécifié	"name": string "chatID": number	[200] (si tout se passe bien) [400] "Ce chat n'existe pas" [500] (Si le serveur n'arrive pas a enregistrer dans la base de donnée)
POST	/chat/new/member	Ajoute un nouveau membre dans le chat spécifié	"email": string "chatID": number	[200] (si tout se passe bien) [500] (Si le serveur n'arrive pas a enregistrer dans la base de donnée)
GET	/chat/delete/:id	Efface un chat du serveur	id: number	[200] (si tout se passe bien) [400] "Ce chat n'existe pas"
GET	/info/user/chats/:email	Retourne tous les chats dont l'utilisateur est membre	email: string	[200] {chatId: number, name: string}[]
POST	/game/newPair	Enregistre une nouvelle paire mot-image	WordImage	[200] (si tout se passe bien) [400] "Un des mots est trop vulgaire" [500] (Si le serveur n'arrive pas à enregistrer dans la base de données)
GET	/game/start/:gameID	Commence une partie classique	gameID: number	[200] (si tout se passe bien) [400] "Cette partie n'existe pas"

GET	/game/lobbyUpdate/:gameID	Notifie à tous les clients qu'il y a eu un changement (connexion/déconnexion) dans le salon de jeu	gameID: number	[200] (si tout se passe bien) [400] "Cette partie n'existe pas"
GET	/game/next-turn/:gameID	Demande au serveur de continuer la partie et notifie tous les clients dans la partie qu'il faut continuer	gameID: number	[200] (si tout se passe bien) [400] "Cette partie n'existe pas"

Requêtes sur mot de passe

Type de requête	**URL**	**Description**	**Paramètres**	**Réponse [code status http]**
POST	/recovery/start	Commence la récupération de mot de passe en envoyant un courriel à l'adresse courriel de l'utilisateur	"email":string	[200] (si tout se passe bien) [400] "L'utilisateur n'existe pas"
POST	/recovery/confirm	Confirme le code de récupération reçu par courriel	"Email": string "Code": number	[200] (si le code est correct) [401] (si le code n'est pas correct)
POST	/recovery/change	Change le mot de passe de l'utilisateur	"email": string "password": string	[200] [400] "L'utilisateur n'existe pas" [500] (Si le serveur n'arrive pas à enregistrer dans la base de données)

Requête sur les trophées

Type de requête	**URL**	**Description**	**Paramètres**	**Réponse [code status http]**
POST	/trophy/detect	Détecte si l'utilisateur a un nouveau trophée. Si oui, l'utilisateur est notifié et il reçoit un trophée dans la base de données	"email": string	[200]
POST	/trophy/give	Pour les trophées qui sont indétectables par le serveur, le client peut se donner un trophée s'il n'existe pas déjà. Le client sera alors notifié et il recevra un trophée dans la base de données	"email": string "trophy": string	[200]

3.3 Requêtes Websocket

URL	Description	Paramètre de connection	Réponse de connection	Message	Réponse
/chat/join/:roomID	Canal de chat selon le ID	roomID: number	[200] (connection) [400] "Le chat n'existe pas"	Message	Message
/game/connect/:gameID/:email	Permet de rejoindre une partie.	gameID: number email: string	[200] (connection) [400] "La partie n'existe pas" [400] "L'utilisateur n'existe pas"	Drawing	Drawing
/session/:email	Permet de notifier l'utilisateur des évènements de sa connection et permet au serveur de savoir quand l'utilisateur est connecté	email: string	[200] (connection) [400] "L'utilisateur n'existe pas"		Notifications

4. Communication serveur-base de données

Notre serveur communique avec une base de données MongoDB qui sera sur une instance EC2 de AWS. Ils se connectent en utilisant l'URI de la base de données et se communiqueront en utilisant des messages encodés en BSON (binary JSON).