Équipe 4

Poly Paint Pro
Protocole de communication

Version 2.0

Historique des révisions

Date	Version	Description	Auteur
2018-09-25	1.1	Écriture de l'introduction, du protocole de communication et début de le l'explication des paquets	Pascal
2018-09-26	1.2	Écriture de l'explication des paquets	Pascal et Sébastien
2018-09-27	1.3	Redéfinition des paquets de type "Object" pour une documentation plus complète des paquets	Pascal, Sébastien, Audrey et Éric
2018-09-28	1.4	Ajout de la section 4 pour rendre le tableau des paquets plus clair	Pascal
2018-09-28	2.0	Révision et mise en forme du document en vue de la remise.	Audrey, Sébastien

Table des matières

Protocole de communication

1. Introduction

Ce document détaille extensivement le moyen de communication entre le client lourd, le client léger et le serveur. Son objectif est de décrire quelles actions une entité peut effectuer sur une autre.

Dans la section 2, nous allons décrire quelles technologies nous allons utiliser dans le cadre du projet afin de réaliser ces communications. Par la suite, nous allons décrire le contenu des paquets utilisés pour chaque message réussi.

2. Communication client-serveur

Afin d'établir la communication entre le client et le serveur, nous allons utiliser des sockets, avec la librairie socket.io pour le système de communication et l'édition des images. Cette librairie est un logiciel libre et est disponible en trois versions pour les trois langages du projet, c'est-à-dire le client lourd programmé sur Windows en C#, le client léger programmé sur IOS en Swift, le site web codé en TypeScript avec le cadriciel Angular et le serveur également en TypeScript sur Node.js. Pour les autres fonctionnalités comme le téléchargement des images et l'authentification, nous allons utiliser l'API REST afin de faire nos requêtes HTTP.

Le serveur sera déployé sur une machine virtuelle EC2 d'Amazon Web Services (AWS) et les clients seront sur un ordinateur pour le client lourd et sur une tablette pour le client léger. De plus, nous allons utiliser S3 d'AWS afin de stocker les images pour notre projet. Finalement DynamoDB et RDS vont stocker les données de notre projet. Pour réaliser cette connexion, nous aurons besoin de ports d'accès ainsi que d'une adresse IP pour le serveur et pour le client. La communication se fera donc par TCP/IP.

3. Description des paquets

Les paquets devront respecter le protocole de communication TCP/IP. Ils ne pourront donc pas dépasser la taille de maximale imposée par ce protocole. De plus l'encodage doit être little endian pour toutes les plateformes afin que la communication entre clients soit uniforme. Voici un exemple de paquet générique:

Description	Type de message	Données

3.1. Communication client vers serveur avec sockets

3.1.1. Système de conversation

Description	**Nom de la requête**	**Données**
Entrer dans un chat room	enter-chat-room	{userId: String, chatRoomId: String, token: String[]}
Quitter un canal	quit-chat-room	{userId: String, chatRoomId: String}
Envoyer un message	send-message	{userId: String, chatRoomId: String, message: String[]}

3.1.2. Édition des images

Description	Nom de la requête	Données
Modification de la forme	modify-image	{userId: String, imageId String, shape: shapeObject[1]}
Choix de l'image	pick-image	{userId: String, imageId: String}
Ajouter une forme	add-shape	{userId: String, imageId String, shapeType: String}
Retirer une forme	remove-shape	{userId: String, imageId String, shapeType: String}
Sauvegarder l'image	save-image	{userId: String, imageId String}
Revenir à une version précédente	previous-version	{userId: String, imageId String, version: String}
Créer une version	new-version	{userId: String, imageId String, version: String}

3.2. Communication Client vers Serveur avec la REST API

3.2.1. Galerie d'image

Description	Nom de la requête	Données
Demander la liste des images	list-image	{userId: String, imageList: String[]}
Demander un thumbnail	thumbnail-image	{userId: String, thumbnailList: String[]}
Demander les commentaires	view-comment-image	{userId: String, imageId: String, message: String[]}
Envoyer un nouveau commentaire	send-comment-image	{userId: String, imageId: String, message: String}

[1] Voire section 4 pour les détails de ce paquet

Ajouter un ‘J’aime’	add-like-image	{userId: String, imageId: String, like: Boolean}
Retirer un ‘J’aime’	remove-like-image	{userId: String, imageId: String, like: Boolean}
Créer une nouvelle image	create-new-image	{userId: String}
Demander un lien de partage	ask-shared-link	{userId: String, lien: String}
Changer le nom d’une image	change-name-image	{userId: String, imageId: String, imageName: String}
Protéger une image personnelle	protect-image	{userId: String, imageId: String, protected: Boolean}

3.2.2. Canaux de discutions

Description	Nom de la requête	Données
Créer un nouveau canal de discussion	new-chat-room	{ name: String }
Inviter une nouvelle personne au chat	invite-chat-room	{invitedUserId: String, chatRoomId: String}
Afficher les chats	get-chat-rooms	{ chatRoomIds: String[]}

3.2.3. Administration des usagers

Description	Nom de la requête	Données
Demander la liste des usagers	list-users	{userIds: String[]}
Changer le type d’un usager	change-type-user	{userId: String, userType: String}
Changer une permission individuelle d’un usager	change-permission-user	{userId: String, userPermission: String}

Ajouter un usager	create-user	{name: String, password: String}
Supprimer un usager	remove-user	{userId: String}
Demander les images privées d'un usager	view-private-image	{userId: String, imageIds: String[]}
Supprimer les images d'un usager	remove-image	{userId: String, imageId: String}

3.2.4. Authentification

Description	Nom de la requête	Données
Connexion	connection-server	{name: String, password: String, addresse: String, Port: Int}
Déconnexion	deconnection-server	{userId: String}
Création de compte	create-account	{name: String, password: String}
Modification du compte	modify-account	{newName: String, password: String}
Obtenir information du compte	view-account	{name: String}

3.3. Communication Serveur vers Client avec sockets

3.3.1. Système de conversation

Description	Nom de la requête	Données
Envoie de message	send-message	{chatRoomId: String, message: String}
Envoie de notification	send-notification	{chatRoomId: String, notificationType: String}

3.1.2. Édition des images

Description	Nom de la requête	Données
Envoi de modifications	send-modification	{imageId: String, shape: shapeObject[2]}
Appliquer une modification	apply-modification	{imageId: String, shape: shapeObject[3]}
Sauvegarde automatique	save-modification	{imageId: String}

3.4. Communication Serveur vers Client avec REST

Description	Nom de la requête	Données
Confirmation	confirm	{confirmationType: String}
Envoi d'image	send-image	{image: imageObject[4]}

4. Détail des objets dans les paquets

4.1. shapeObject:

Voici la description du shapeObject utilisé pour envoyer des formes (format JSON):

```
{shapeType: String,
requestType: String,
imageId: String,
objectId: String,
zIndex: Int,
shapeSpecificInfo: {
       coordinates: [{x:int, y:int}],
       boderWidth: Int,
       boderColor: {r:int, g:int, b:int},
       interiorColor: {r:int, g:int, b:int},
       rotation: Double,
       centerCoordinates: {x:int, y:int},
       text:[{font: String,fontSize: String,
       bold: Bool,
       italic: Bool, underline:bool, content: String}],
       nTextSections: Int
       }
}
```

[2] Voire section 4 pour les détails de ce paquet
[3] Voire section 4 pour les détails de ce paquet
[4] Voire section 4 pour les détails de ce paquet

4.2. imageObject:

Voici la description du imageObject utilisé pour envoyer des images (format JSON):

```
{imageId: String,
userId: String,
VersionId: String,
shapes: [shape: Object[5]],
imageType: String
}
```

[5] Voire section 4 pour les détails de ce paquet