Poly Paint Pro

Protocole de communication

Version 3.0

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 |
| 2018-11-27 | 3.0 | Mise à jour du document pour la remise finale | Pascal |

Table des matières

[**1. Introduction**](#_30j0zll) **4**

[**2. Communication client-serveur**](#_1fob9te) **4**

[**3. Description des paquets**](#_3znysh7) **4**

[3.1. Communication client vers serveur avec sockets](#_mb0gr8n64o4v) 4

[3.1.1. Système de conversation](#_cbfa1olahfp6) 4

[3.1.2. Édition des images](#_dh58tjeiumcf) 5

[3.2. Communication Client vers Serveur avec la REST API](#_ql4lhfh8on65) 5

[3.2.1. Galerie d’image](#_ny6frgsnip0n) 5

[3.2.2. Canaux de discutions](#_qookw6ygc2y9) 6

[3.2.3. Administration des usagers](#_wnbzolbswxfx) 6

[3.2.4. Authentification](#_64fu96uxmbms) 7

[3.3. Communication Serveur vers Client avec sockets](#_jd7l6uy5zmc0) 7

[3.3.1. Système de conversation](#_4xcb2t8x4eyj) 7

[3.1.2. Édition des images](#_efsn9ag271ln) 8

[3.4. Communication Serveur vers Client avec REST](#_rt2h4bne8r8j) 8

[**4. Détail des objets dans les paquets**](#_3b4kwi1h41ji) **8**

[4.1. shapeObject:](#_g0a2t1gznsdz) 8

[4.2. imageObject:](#_t0mkjpn73fzs) 9

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 RDS va 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 | joinRoom | {user: User, room: String} | | | |
| Quitter un canal | leaveRoom | {user: User, room: String} | | | |
| Envoyer un message | message | {user: User, room: String, message: String} | | | |
| Inviter une nouvelle personne au chat | addToRoom | {room: String, username: String, user; User} | | | |

### 

### 3.1.2. Édition des images

|  |  |  |
| --- | --- | --- |
| **Description** | **Nom de la requête** | **Données** |
| Choix de l’image | joinImage | {user: User, imageId: String} |
| Partir de l’édition | leaveImage | {user: User} |
| Ajouter une forme | addStroke | {user: User, stringifiedStroke: String} |
| Retirer une forme | removeStroke | {user: User, strokeId: String} |
| Moditifer une forme | editStroke | {user: User, stringifiedStroke: String} |
| Ajouter une protection à l’image | requestProtection | {user: User, strokeIds: String} |
| Retirer une protection à l’image | removeProtection | {userId: String} |
| Effacer l’image | clearCanvas | {userId: 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 | get  /image | {imageList: Image[]} | | | |
| Demander une image | get  /image/:id | {image: Image} | | | |
| Demander les images d’un usager | get  /imageByOwnerId/:id | {userId: String, imageList: Image[]} | | | |
| Demander tous les images sauf les miennes | get  /imagesPublicExceptMine/:id | {userId: String, imageList: Image[]} | | | |
| Ajouter une image | post  /image | {image: Image} | | | |
| Modifier une image | put  /image/:id | {imageId: String, image: Image} | | | |
| Detruire une image | delete  /image/:id | {imageId: String} | | | |
| Checher les j’aime de tous les images | get  /imageLikes | {imageIds: String[], userIds: String[]} | | | |
| Checher les j’aime de une image | get  /imageLikes/:imageId | {imageId: String, userId: String} | | | |
| Ajouter un j’aime | post  /imageLikes | {imageId: String, userId: String} | | | |
| Retirer un j’aime | delete  /imageLikes/:imageId/:userId | {imageId: String, userId: String} | | | |
| Ajouter un commentaire | post  /imageComments | {imageId: String, message: String} | | | |
| Rechercher les commentaires d’une image | get  /imageComments/:imageId | {imageId: String} | | | |
| Detruire un commentaire | Delete  /imageComments/:imageId/:userId | {imageId: String, userId: String} | | | |

### 3.2.2. Canaux de discutions

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Description** | **Nom de la requête** | **Données** | | | |
| Rechercher tous les salles de conversations | get  /chatRooms | - | | | |
| Rechercher tous les usagers d’une salle de conversation | get  /chatRooms/:id | {chatRoomId: String} | | | |
| Rechercher les usagers | get  /conntectedUsers | {users: User[]} | | | |

### 3.2.3. Administration des usagers

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Description** | **Nom de la requête** | **Données** | | | |
| Demander la liste des usagers | get  /users | {users: User[]} | | | |
| Changer le type d’un usager | put  /users/:id | {userId: String, user: User} | | | |
| Ajouter un usager | post  /users | {user: User} | | | |
| Supprimer un usager | delete  /users/:id | {userId: String} | | | |
| Demander un usager specifique | get  /users/:id | {userId: String} | | | |
| Supprimer un usager | delete  /users/:id | {userId: String} | | | |

### 3.2.4. Authentification

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Description** | **Nom de la requête** | **Données** | | | |
| Connexion | post  /sessions | {username: String, password: String} | | | |
| Déconnexion | delete  /sessions/:id | {userId: String} | | | |
| Obtenir les usagers authentifiés | get  /sessions | {name: String} | | | |

### 3.2.5. Gestion des amis

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Description** | **Nom de la requête** | **Données** | | | |
| Rechercher amitiés d’un usager | get  /friendship/:id | {name: String, password: String, addresse: String, Port: Int} | | | |
| Rechercher usagers qui ne sont pas ami avec un certain usager | get  /userExecptFriends/:id | {userId: String} | | | |
| Ajouter un lien d’ami | post  /friendship/:id | {userId: String, friendId: String} | | | |
| Retirer un lien d’amitie | delete  /friendship/:id | {userId: String, friendId: String} | | | |
| Rechercher tous les relations d’ami en attente | get  /pendingFriendRequest | {pendingRequests: PendingFriendRequest[]} | | | |
| Recherche les relations d’ami en attente d’un usager | get  /pendingFriendRequest/:id | {pendingRequests: PendingFriendRequest[]} | | | |
| Recherche les relations d’ami envoyés par un usager | get  /pendingFriendRequestByRequesterId/:id | {pendingRequests: PendingFriendRequest[]} | | | |
| Retirer un relation d’ami en attente | delete  /pendingFriendRequest/:id | {pendingRequest: PendingFriendRequest} | | | |

## 

## 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[[1]](#footnote-0)} | | | |
| Appliquer une modification | apply-modification | {imageId: String, shape: shapeObject[[2]](#footnote-1)} | | | |
| 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[[3]](#footnote-2)} |

# 4. Détail des objets dans les paquets

## 4.1. shapeObject:

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

{

id: String,

imageId: String,

shapeType: String,

index: int,

shapeInfo: {ShapeInfo} (strignified)

}

## 4.2. image:

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

{

imageId: String,

userId: String,

title: String,

protectionLevel: String,

password: String

thumbnailUrl: String,

fullImageUrl: String,

secret: String,

width: int,

height: int,

creationdate: Date

}

## 4.3. shapeInfo:

**4.3.1: Pour les lignes:**

{

Points: [{x: double, y: double}]

FirstAnchorId: String

FristAnchorIndex: int,

SecondAnchorId: String,

SecondAnchorIndex: int,

FirstEndLabel: String,

FirstEndRelation: String,

SecondEndLabel: String,

SecondEndRelation: String,

}

**4.3.2: Pour les autres formes:**

{

Center: {x: double, y: double},

Width: int,

Height:int,

Color: String

}

1. Voire section 4 pour les détails de ce paquet [↑](#footnote-ref-0)
2. Voire section 4 pour les détails de ce paquet [↑](#footnote-ref-1)
3. Voire section 4 pour les détails de ce paquet [↑](#footnote-ref-2)