T6

Projet intégrateur LOG3900
Document d'architecture logicielle

Version 1.5.4

Historique des révisions

Date	Version	Description	Auteur
2019-01-27	1.0	Première rédaction des sections 1 et 2	Freddy Sossa
2019-02-02	1.1	Rédaction de la section 7	Hakim Kasimi
2019-02-04	1.2	Rédaction de la section 3 et 5	Freddy Sossa, Amal Metsahel
2019-02-06	1.3	Rédaction de la section 6	Kelvin Tran
2019-02-07	1.4.0	Rédaction de la vue logique pour le client lourd	Zouhair Chiguer
2019-02-07	1.4.1	Révision de la partie 5	Amal Metsahel
2019-02-08	1.5.1	Rédaction de la vue logique et ajout du diagramme de paquetage pour le client léger	Freddy Sossa
2019-02-08	1.5.2	Création du diagramme de paquetage du serveur et première esquisse de la vue logique du serveur	Kelvin Tran
2019-02-08	1.5.3	Révision et complétion de la section 4.3.	Kelvin Tran
2019-02-08	1.5.4	Révision finale du document	Amal Metsahel

Table des matières

Document d'architecture logicielle

1. Introduction

Le présent document présente une description de l'Architecture logicielle de haut niveau de l'application *PolyPaint* Pro. Elle expose plus particulièrement les objectifs et contraintes de l'architecture, une vue des cas d'utilisation les plus pertinents, une vue logique du système, un de déploiement et fait également ressortir les aspects taille et performance du système à développer.

2. Objectifs et contraintes architecturaux

L'objectif principal est de développer une application *PolyPaint* qui permet la création et l'édition d'images en incluant un mode collaboratif en réseau tout en permettant aux utilisateurs de communiquer par un canal de discussion. Un tel système pour des raisons de performances optimales nécessite une architecture client-serveur qui sépare le client du serveur afin de gérer efficacement les traitements. Il faut aussi s'assurer de la sécurité du système en effectuant un cryptage des données sensibles. Le mode collaboratif doit assurer à l'utilisateur un délai de latence aussi faible que possible. Il faut donc une technologie qui permet une transmission en temps réel pour maintenir une bonne expérience de l'utilisateur.

Il est à noter ici qu'on développe pour deux systèmes différents, notamment le client lourd (Windows) et le client léger (Android). Pour ce qui est du client lourd, on part d'une architecture déjà existante. La contrainte étant ici qu'on doive plus ou moins conserver cette architecture afin de garder une certaine cohérence.

3. Vue des cas d'utilisation

Il s'agit ici d'illustrer des différents cas d'utilisation du système. Ces cas d'utilisation donnent une vue assez globale du comportement et des interactions entre l'utilisateur et le système. Nous présentons seulement les cas d'utilisations les plus pertinents, qui font ressortir les fonctionnalités principales de l'application.

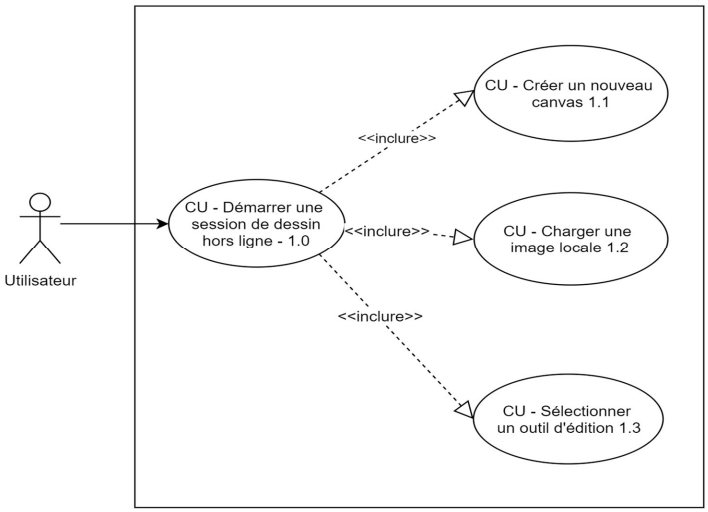

Figure 1 : Cas d'utilisation 1.0 - Démarrer une session de dessin hors ligne (Client lourd et léger)

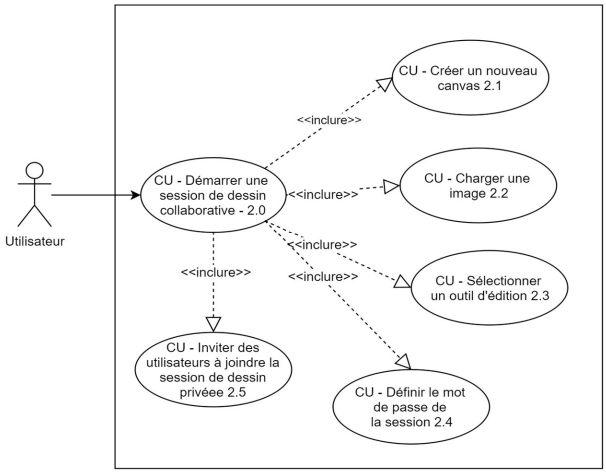

Figure 2 : Cas d'utilisation 2.0 - Démarrer une session collaborative de dessin (Client lourd et léger)

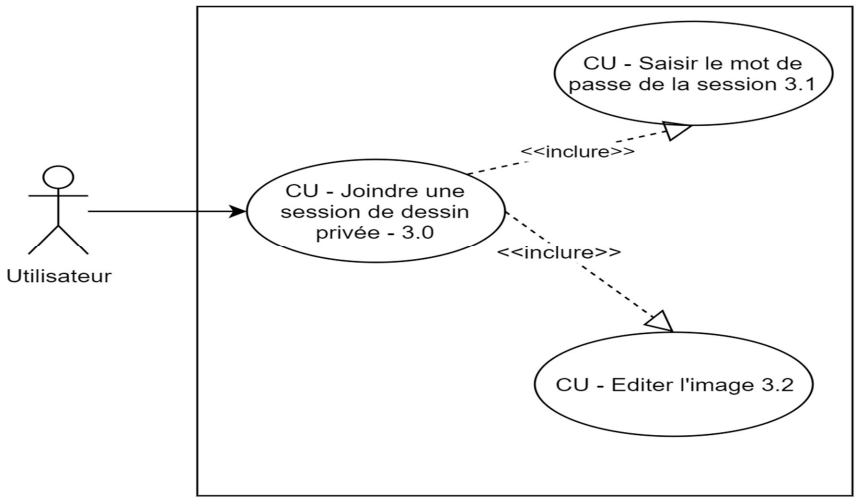

Figure 3 ; Cas d'utilisation 3.0 - Joindre une session de dessin privée (Client lourd et léger)

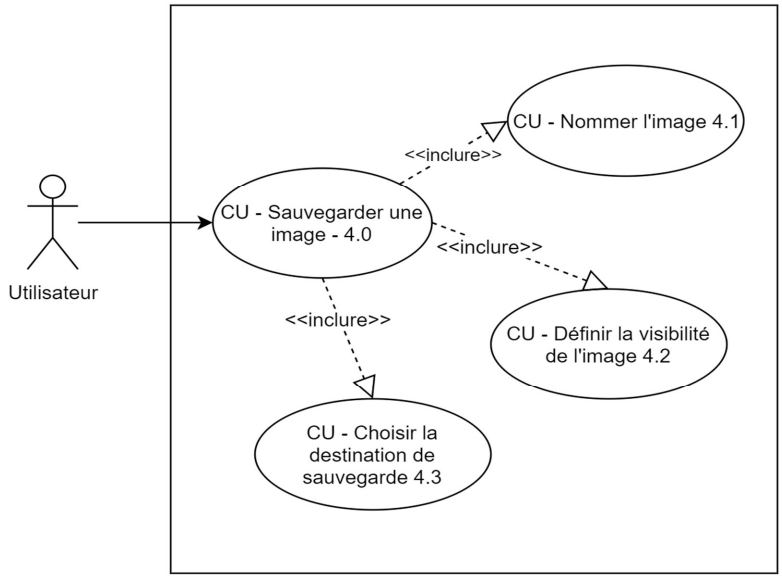

Figure 4 : Cas d'utilisation 4.0 - Sauvegarder une image (Client lourd)

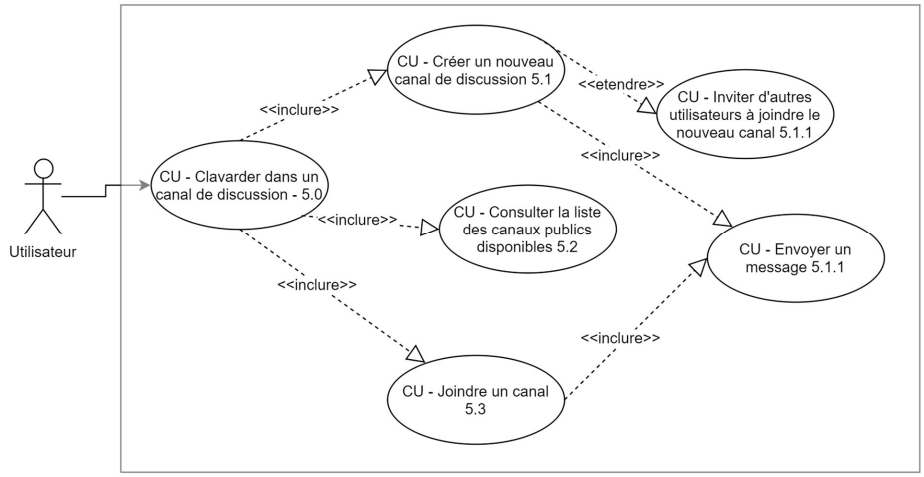

Figure 5 : Cas d'utilisation 5.0 - Clavarder dans un canal de discussion

4. Vue logique

4.1 Client Lourd

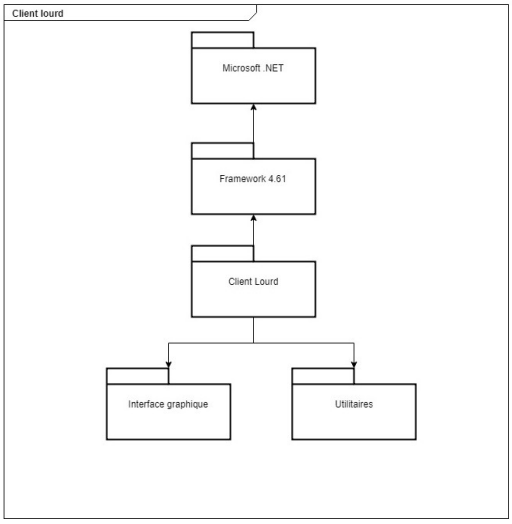

Figure 6 : Diagramme de paquetage Client lourd

Microsoft .NET	
Description:	C'est le paquetage qui contient tout le nécessaire pour faire fonctionner le framework qui en dépend.
Classes incluses:	----------
Relations:	----------
Sous-paquetages:	-Framework 4.61 **Microsoft**

Framework 4.61	
Description:	C'est le paquetage qui définit les limites du compilateur C#. Il s'agit du langage de programmation qui s'occupe de la communication du serveur.
Classes incluses:	- System.Windows - System.Threading - System.Net.Sockets - System.ComponentModel
Relations:	----------
Sous-paquetages:	-Client Lourd

Client lourd	
Description:	Ce paquetage englobe l'application client qui est déployée sur un PC. Il échange des informations avec le serveur. Il implémente l'interface utilisateur ainsi que le modèle de l'application client lourd.
Classes incluses:	----------
Relations:	-Serveur : Échange de messages et d'information entre le client et le serveur
Sous-paquetages:	- Interface Graphique - Utilitaires

Interface Graphique	
Description:	Ce paquetage englobe l'application client qui est déployée sur un PC. Il échange des informations avec le serveur. Il implémente l'interface utilisateur ainsi que le modèle de l'application client lourd.
Classes incluses:	- FenetreDessin - ChatWindow - LoginWindow - VueModele - ChatBubbleControl
Relations:	----------
Sous-paquetages:	----------

Utilitaires	
Description:	Ce paquetage englobe l'application client qui est déployée sur un PC. Il échange des informations avec le serveur. Il implémente l'interface utilisateur ainsi que le modèle de l'application client lourd.
Classes incluses:	- ChatManager - CommunicationManager - RelayCommand - Convertisseurs - Editeur - ProfilManager - FileManager
Relations:	----------
Sous-paquetages:	----------

4.2 Client Léger

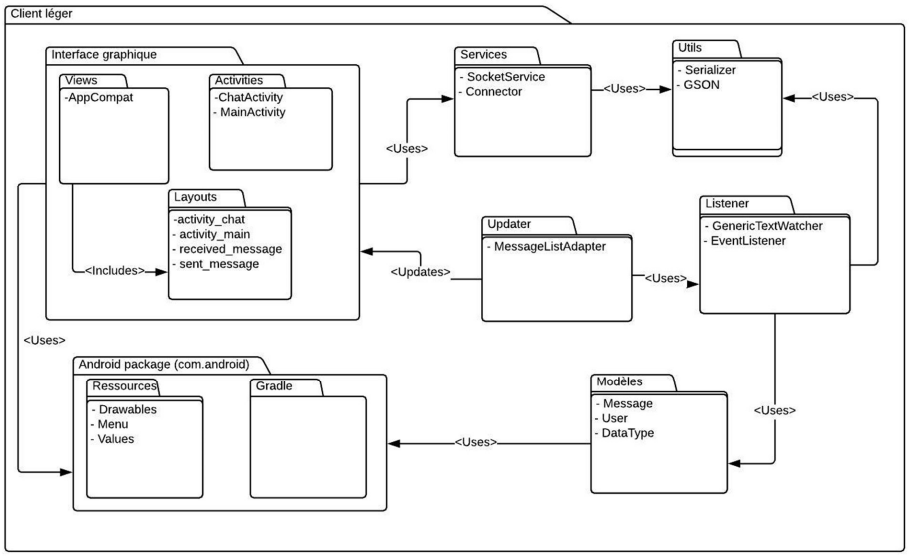

Figure 7 : Diagramme de paquetage Client Lourd

Interface Graphique	
Description:	C'est le paquetage qui contient toutes les classes, les activités et tout ce qui gère la vue. Il assure l'interaction entre l'application et l'utilisateur
Sous-paquetages:	- Views - Activities - Layouts

Views	
Description:	C'est le sous paquetage qui définit la vue de l'utilisateur. Elle très liée pour ne pas dire confondue au sous paquetage des layouts.
Classes incluses:	- AppCompat

Layouts	
Description:	C'est le sous paquetage qui définit la vue de l'utilisateur. Elle très liée pour ne pas dire confondue au sous paquetage des layouts.
Classes incluses:	- activity_chat - activity_main - activity_message - sent_message

Activities	
Description:	C'est le sous paquetage qui représente à la fois la vue et les activités qui y sont effectuées en arrière plan.
Classes incluses:	- ChatActivity - MainActivity

Services	
Description:	Ce paquetage regroupe l'ensemble des différents services utilisés pour effectuer des traitements.
Classes incluses:	- SocketService - Connector
Relations	Server : Il assure également la communication avec le serveur.

Utils	
Description:	C'est le paquetage qui regroupe l'ensemble des éléments provenants de librairies extérieures ou des classes écrites pour servir d'utilitaire ou effectuer des traitements internes pour alléger le code.
Classes incluses:	- GSON - Serializer

Listener	
Description:	C'est le paquetage qui est connecté aux modèles et qui prévient l'Updater des changements effectués sur le modèle.
Classes incluses:	- GenericTextWatcher - EventListener

Modèles	
Description:	Ce paquetage regroupe l'ensemble des différents éléments traités. Il représente des éléments concrets tels que les messages, les utilisateurs, les types réponses du serveur.
Classes incluses:	- Message - User - DataType

Updater	
Description:	C'est le sous paquetage qui gère la mise à jour des éléments de la vue. Il poste est connecté au Listener qui le prévient des changements sur le modèle et fait constater ses changements à l'utilisateur sur la vue
Classes incluses:	- MessageListAdapter

Android Package(com.android)	
Description:	C'est le paquetage général qui nous donne accès aux ressources du langage et à toutes les librairies connexes qui interviennent dans les traitements.

Ressources	
Description:	Ce sous paquetage regroupe l'ensemble des ressources utilisées par l'application. Ce sont des fichiers xml qui définissent des formes, des valeurs, des couleurs, etc.
Classes incluses:	- Drawable - Menu - Values

Gradle	
Description:	C'est le sous paquetage d'Android qui gère le build et la configuration en arrière-plan de l'application.

4.3 Serveur

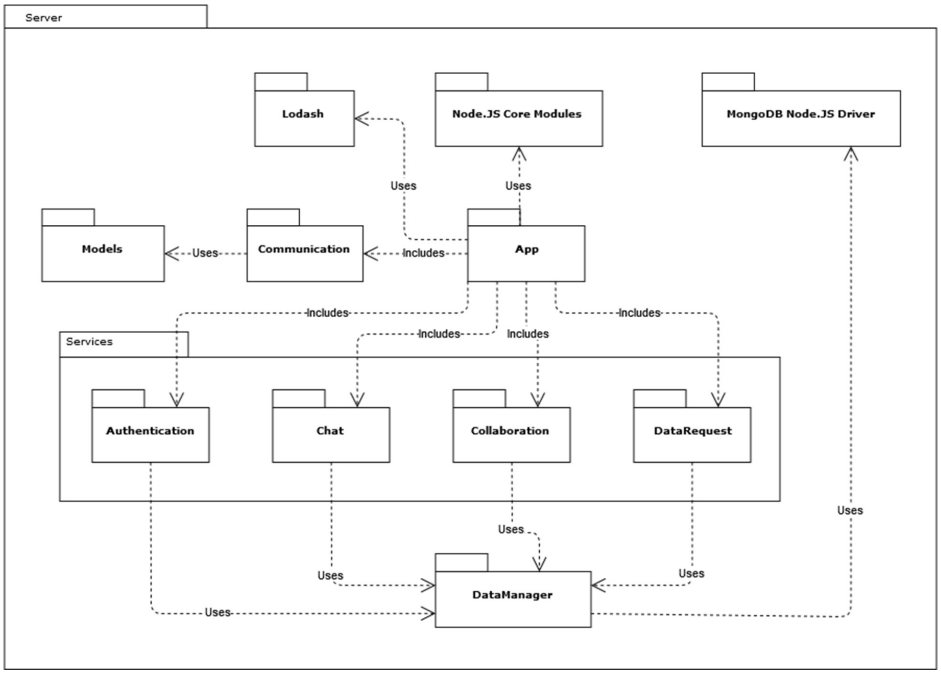

Figure 8 : Diagramme de paquetage du serveur

Node.JS Core Modules	
Description:	Ce paquetage inclut tous les modules essentiels déjà intégrés à Node.JS.
Classes incluses:	----------
Relations:	----------
Sous-paquetages:	- net - os - tls - readline - buffer - fs - events - stream - path - util … (les autres ne sont pas pertinents)

Lodash	
Description:	Ce paquetage représente le module lodash qui permet la manipulation facile et performante d'objets et de collections
Classes incluses:	----------
Relations:	----------
Sous-paquetages:	----------

MongoDB Node.JS Driver	
Description:	Ce paquetage représente l'ensemble des modules qui permettent d'exécuter des opérations sur la base de données MongoDB
Classes incluses:	----------
Relations:	----------
Sous-paquetages:	----------

App	
Description:	Ce paquetage contient l'application de base du serveur. Il est responsable de faire rouler le serveur.
Classes incluses:	- app - Server
Relations:	- Communication: Le serveur utilise ce paquetage qui permet à l'application de communiquer avec les clients. - Node.JS Core Modules: Différents modules intégrés offerts par Node.JS pour accéder à certaines interfaces de la machine et pour créer un serveur - Services: Le serveur offre différents services accessibles: - Authentication - Chat - Collaboration - DataRequest
Sous-paquetages:	----------

Communication	
Description:	Ce paquetage permet à l'application de communiquer avec les clients.
Classes incluses:	- Network - DataSerializer - PacketHandler - SocketHandler - Client - ClientManager
Relations:	- Models: Les modèles de données (interfaces) sont utilisés pour la sérialisation et la désérialisation des données.
Sous-paquetages:	----------

Models	
Description:	Ce paquetage contient tous les modèles de données utilisés pour la communication entre le serveur et le client.
Classes incluses:	- IData - IAuthRequest - IMessage - IEvent - IError - IOperation - IOperationCreate - IOperationDelete - IOperationModify - IOperationSelect - IOperationReset - IRequest

	- IResponse
Relations:	Communication: Paquetage qui utilise les modèles
Sous-paquetages:	----------

Services	
Description:	Ce paquetage contient tous les services offerts par le serveur.
Classes incluses:	- ServiceManager - Service
Relations:	----------
Sous-paquetages:	- Authentification - Chat - Collaboration - DataRequest

Authentication	
Description:	Ce paquetage contient toutes les classes qui permettent de valider une authentification d'utilisateur.
Classes incluses:	- Authenticator - PasswordEncryptor - UserManager
Relations:	- DataManager: Le paquetage Authentication doit utiliser le DataManager pour pouvoir extraire des données de la base de données et d'en rajouter.
Sous-paquetages:	----------

Chat	
Description:	Ce paquetage englobe toutes les classes qui permettent de gérer le service de chat.
Classes incluses:	- Chat - ChatManager - MessageManager - Message
Relations:	- DataManager: Le paquetage Chat doit utiliser le DataManager pour pouvoir sauvegarder les messages sur la base de données.
Sous-paquetages:	----------

Collaboration	
Description:	Ce paquetage englobe tous les classes qui forment le service permettant la collaboration de dessin entre plusieurs clients.
Classes incluses:	- Session - SessionManager - CanvasLoader - CanvasUpdater

	- OperationManager - Operation - OperationQueue - Synchronizer - SelectionManager
Relations:	- DataManager: Le paquetage Collaboration doit utiliser le DataManager pour pouvoir manipuler les données concernant les canevas sur la base de données.
Sous-paquetages:	----------

DataRequest	
Description:	Ce paquetage contient toutes les classes qui gèrent la demande et l'envoi de données face aux demandes reçues.
Classes incluses:	- RequestHandler - RequestParser - Request - RequestManager - ResponseFactory
Relations:	- DataManager: Le paquetage DataRequest doit utiliser le DataManager pour pouvoir aller chercher des données sur la base de données.
Sous-paquetages:	----------

DataManager	
Description:	Ce paquetage contient les classes qui gèrent l'accès à la base de données et qui construisent les requêtes mongodb pour le MongoDB Node.JS driver.
Classes incluses:	- QueryBuilder - DataManager - Database
Relations:	- MongoDB Node.JS Driver: Le paquetage DataManager utilise le MongoDB Node.JS Driver pour envoyer des requêtes CRUD.
Sous-paquetages:	----------

5. Vue des processus

La communication au sein de ce logiciel se fait entre deux processus: le serveur et le client. Le serveur a un thread responsable de l'écoute des connexions émises par les clients. Chaque client a un thread qui écoute les messages envoyés par le serveur. Si une connexion est réussie entre les deux processus, le serveur crée un thread qui interprète les messages envoyés par le client connecté et qui les envoie aux autres clients connectés aussi.

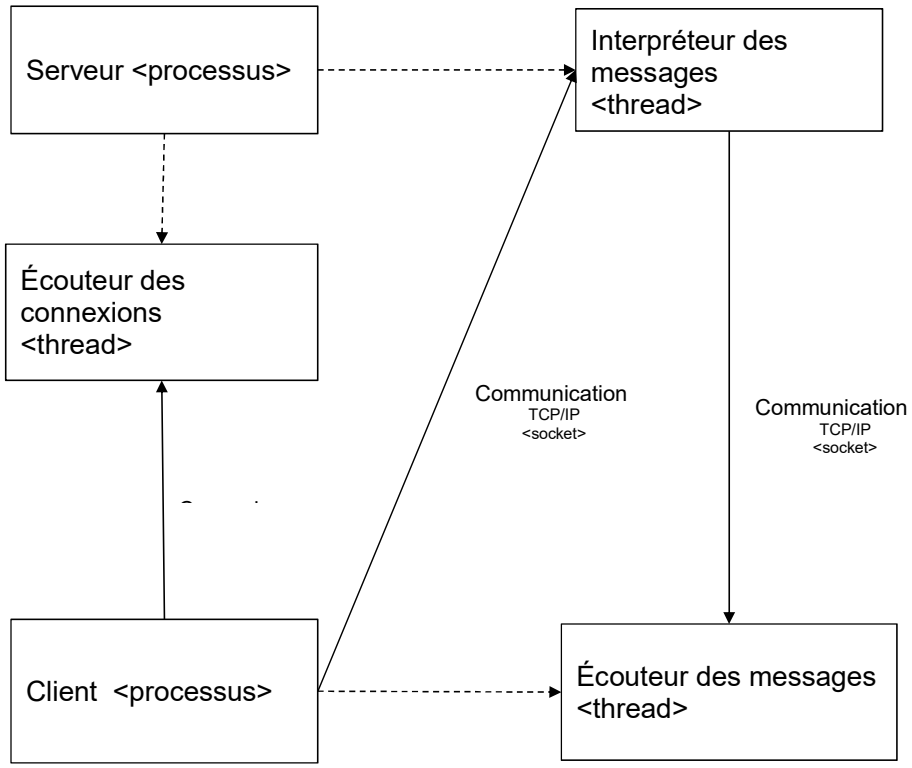

Figure 8 : Diagramme des processus

Commented [1]: À changer

6. Vue de déploiement

Il existe deux types de nœuds physiques sur lesquels les logiciels clients s'exécutent. Le client léger est exécuté sur une tablette roulant sur Android 7.0 et le client lourd et exécuté sur un ordinateur roulant sur Windows 10. Chacun des clients communique avec le serveur par communication TCP/IP. Cette communication est faite utilisant notre protocole de communication définie dans le document de protocole de communication. En plus de communiquer avec les clients, la machine sur laquelle le serveur Node.js roule communique également avec un serveur hébergeant la base de données pour pouvoir envoyer des requêtes et recevoir des données. Cette communication est faite également à l'aide du protocole TCP/IP.

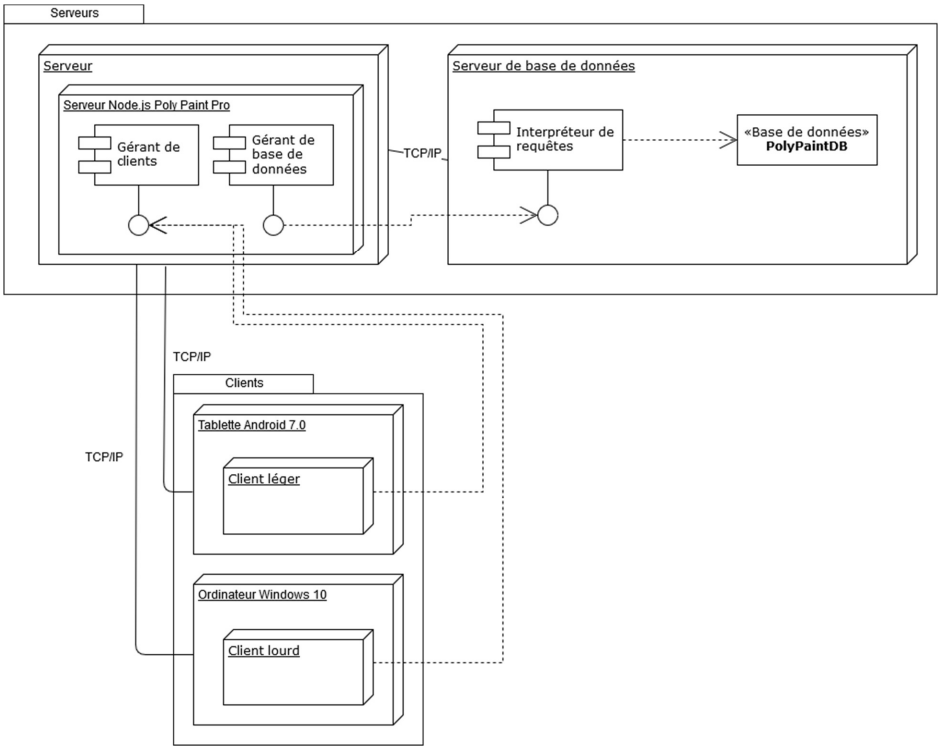

Figure 9 : Diagramme de déploiement

Commented [2]: À changer

7. Taille et performance

En ce qui concerne la taille et la performance, nous avons élaboré une architecture logicielle qui respecte les contraintes énoncées du SRS. Les composantes que nous avons présentées ci-dessus devront prendre en compte ces contraintes pour obtenir une application qui répond aux demandes du client et afin d'obtenir une expérience d'utilisation fluide. En ce qui concerne la taille, nous avons pris en compte que l'espace requis par le client lourd ne doit pas dépasser les 250 Mo et 100 Mo pour le client léger. Pour ce qui est de la performance, un temps de réponse maximal de 500 ms lors des communications avec le serveur et les autres clients et une utilisation de mémoire qui ne dépasse pas 250 Mo pour le client lourd et 150 Mo pour le client léger.