Équipe 4

Poly Paint Pro
Document d'architecture logicielle

Version 2.0

Historique des révisions

Date	Version	Description	Auteur
2018-09-06	1.0	La première rédaction du document architecture	Mohamed Laziz Taouali
2018-09-18	1.1	La première rédaction de l'introduction	Mohamed Laziz Taouali
2018-09-27	1.2	Correction du document	Pascal et Audrey
2018-09-28	1.3	Ajout des nouveaux diagrammes de use case et les diagrammes de séquence système	Ayman
2018-09-28	2.0	Dernière vérification et mise en page	Sébastien

Table des matières

Document d'architecture logicielle

1. Introduction

Le document d'architecture logicielle spécifie notre vision sur les différents éléments architecturaux du système Poly Paint Pro. Ces éléments sont basés sur les exigences provenant du document de spécification des requis du système (SRS) et de la liste des exigences.

Le présent document comprend les objectifs et les contraintes ayant un impact architectural. Plus spécifiquement, la sécurité, la fiabilité et la portabilité seront présentés. Par la suite se trouvent les diagrammes de cas d'utilisation les plus pertinents. Ceux-ci permettent de présenter les actions principales faites par les acteurs et le système. Ensuite, il y a la vue logique qui contient le diagramme de classes et de paquetage et la vue des processus contenant le diagramme de séquence. Finalement, le document se termine par la vue de déploiement et la description des caractéristiques de tailles et performances pouvant avoir un impact sur l'architecture et le design du logiciel.

2. Objectifs et contraintes architecturaux

Il y a quelques défis à relever en ce qui concerne nos objectifs architecturaux, le principal étant la cohésion entre les différents modules de notre application, c'est-à-dire le client lourd, le client léger et le serveur.

2.1 Serveur

2.1.1 L'objectif et l'importance

Le serveur a plusieurs responsabilités. La première est d'assurer la bonne communication entre les différents clients (lourds et légers). La deuxième est le stockage des différentes données du système pour les partager dans l'ensemble du système selon les requêtes envoyées. La troisième est le stockage des différents fichiers du site web et les images sous différentes extensions.

2.1.2 La sécurité et confidentialité des données

Le serveur garantit la sécurité des données communiquées entre les différents clients. Les informations sur la base de données ne seront pas cryptées, mais nous utilisons Amazon Web Services pour entreposer nos données. Ainsi, seuls les services faisant partie du Virtual Private Cloud (VPC) pourront accéder à nos données, ce qui assure une sécurité minimale pour les utilisateurs. Au niveau du site web, l'équipe de développement utilisera le protocole HTTP, donc les informations ne seront pas cryptées.

2.1.3 La fiabilité

Le serveur permet l'accès à différents clients en tout temps. L'offre gratuite d'AWS inclut 750 heures par mois d'utilisation d'instances t2.micro du service Amazon EC2, donc il satisfait notre besoin de 24 h * 31 j / mois = 744 h / mois. De plus, ce type d'instance est très robuste et résiste aux nombreuses requêtes de différents clients. Plusieurs collaborations de dessins et des canaux de discussions pourront donc travailler ensemble sans se soucier de la fiabilité de notre système. Ce service s'exécute au sein de l'infrastructure de réseau et des centres de données Amazon qui sont disponibles à 99.99 %.

2.1.4 L'intégrité

Amazon EC2 s'intègre à la plupart des services AWS, comme Amazon Simple Storage Service (Amazon S3) que nous utiliserons pour stocker les fichiers de notre site web et Amazon DynamoDB pour la base de données non relationnelle. Cette dernière fournit une solution sécurisée et complète, servant au calcul, au traitement des requêtes et au stockage dans l'infonuagique.

2.2 Le client lourd

2.2.1 L'objectif et l'importance

Notre client lourd doit permettre aux utilisateurs d'utiliser les fonctions de base du logiciel telles que l'ajout des formes de base (rectangle, ellipse, triangle), des formes de diagrammes (diagramme de classe et de cas d'utilisation), de relier les formes entre elles et d'ajouter des formes de connexion (lignes et flèches). Le logiciel permet une édition collaborative ainsi que des canaux de discussion pour faciliter la communication.

2.2.2 La portabilité

Le client lourd est développé en C# avec le cadriciel Windows Presentation Foundation (WPF). Il peut être utilisé sur les ordinateurs ayant le système d'exploitation Windows qui répondent aux caractéristiques décrites dans la section 7.2.

2.2.3 Le langage de programmation

Le langage de programmation pour le client lourd est imposé. L'utilisation du C# avec le cadriciel WPF est obligatoire. WPF est un cadriciel permettant de réaliser des applications graphiques qui réagissent à des événements (clic sur un bouton, redimensionnement de la fenêtre, saisie de texte, etc.).

2.2.4 Robustesse

Le client lourd doit pouvoir communiquer avec le serveur et la base de données sur AWS. La communication entre le client lourd et le serveur doit être constante, stable et il ne devrait pas y avoir de différence de données entre les deux.

2.3 Client Léger

2.3.1 L'objectif et l'importance

Notre client lourd doit permettre aux utilisateurs d'utiliser les fonctions de base du logiciel telles que l'ajout des formes de base (rectangle, ellipse, triangle), des formes de diagrammes (diagramme de classe et de cas d'utilisation), de relier les formes entre elles et d'ajouter des formes de connexion (lignes et flèches). Le logiciel permet une édition collaborative ainsi que des canaux de discussion pour faciliter la communication.

2.2.2 La portabilité

Le client lourd est développé dans le langage Swift. Il peut être utilisé sur les tablettes iPad Mini 4 utilisant le système d'exploitation iOS.

2.3.3 Langage de programmation

Le client léger est développé avec le langage Swift en utilisant la plateforme Xcode. Le développement sur iOS est principalement basé sur le patron modèle vue contrôleur (MVC). Ce patron de conception est incontournable dans le développement des applications mobiles puisqu'il isole ses différentes composantes et évite un couplement trop serré entre les classes.

2.3.4 Robustesse

L'application finale doit être robuste face aux différents scénarios d'utilisations selon les exigences déterminées. L'application doit être adaptée à la synchronisation temps réel pour le mode de collaboration en ligne.

3. Vue des cas d’utilisation

3.1. Diagramme de cas d’utilisation pour le clavardage

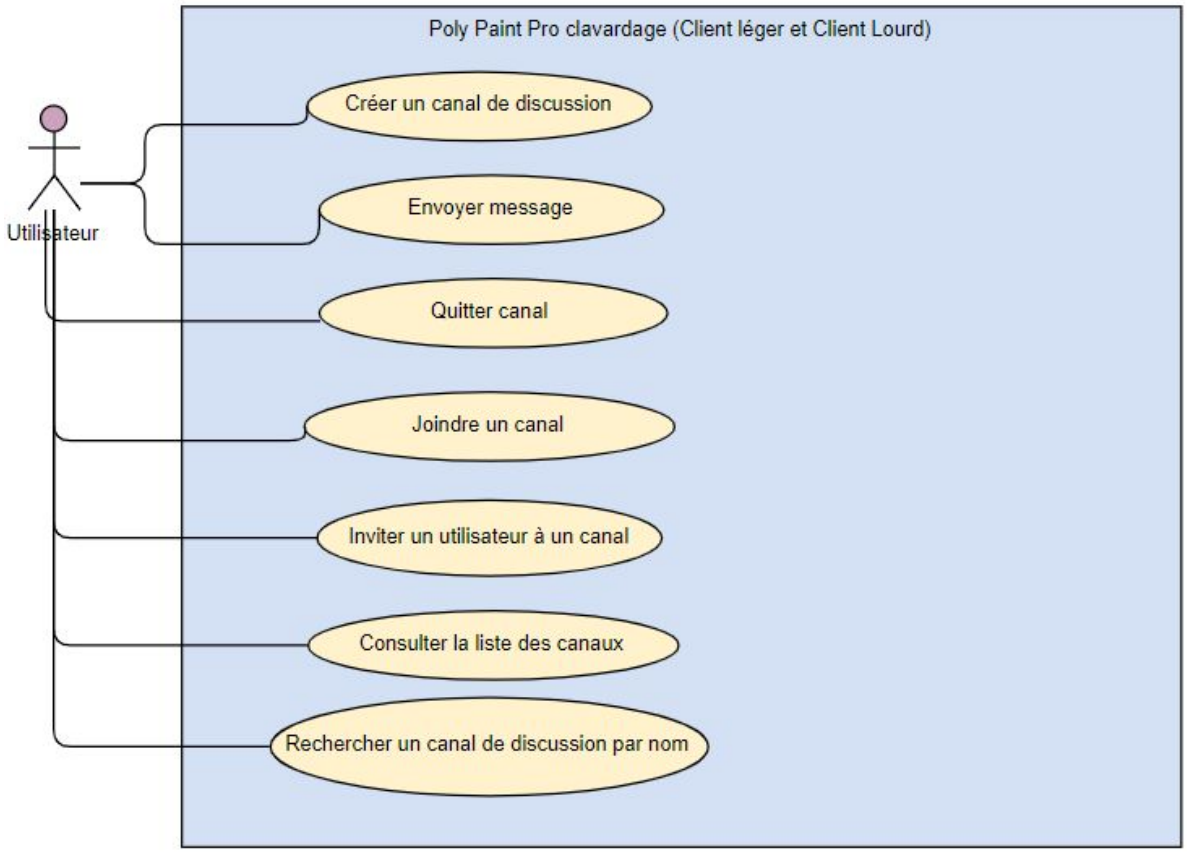

Figure 1: Diagramme de cas d’utilisation pour le clavardage

3.2. Diagramme de cas d’utilisation pour la gestion de profils

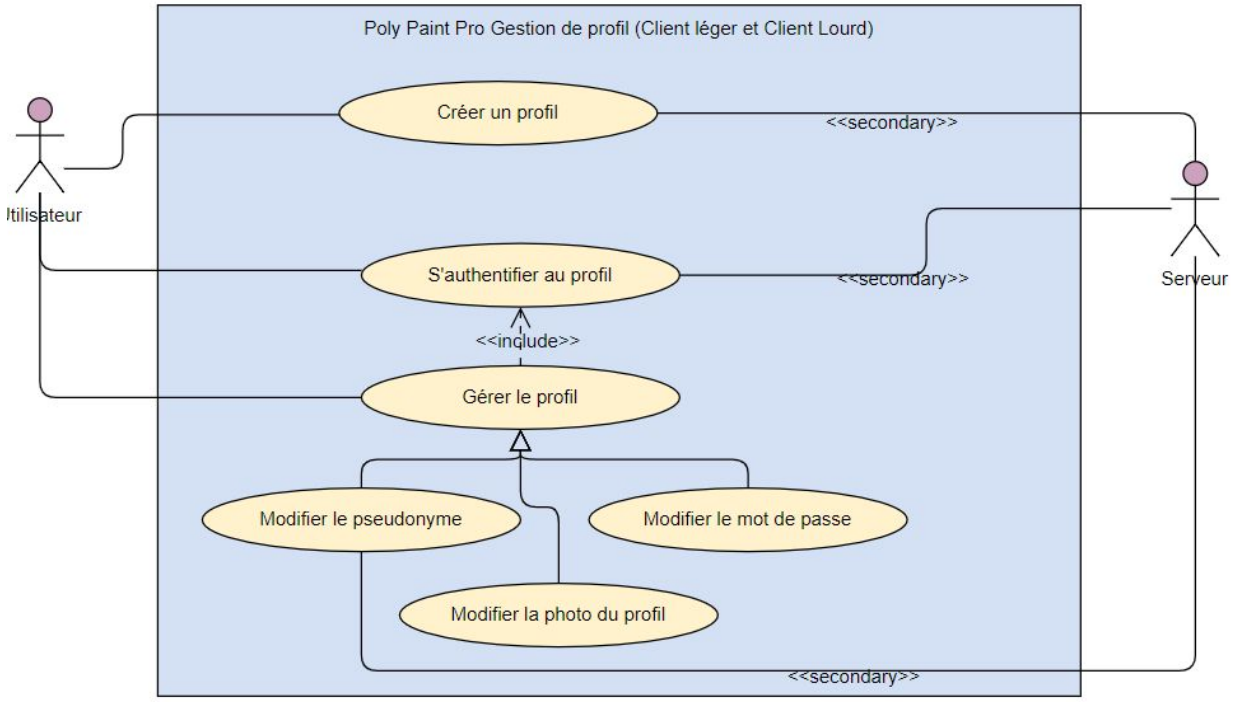

Figure 2: Diagramme de cas d’utilisation pour la gestion de profils

3.3. Diagramme de cas d’utilisation pour l'accessibilité de l’image

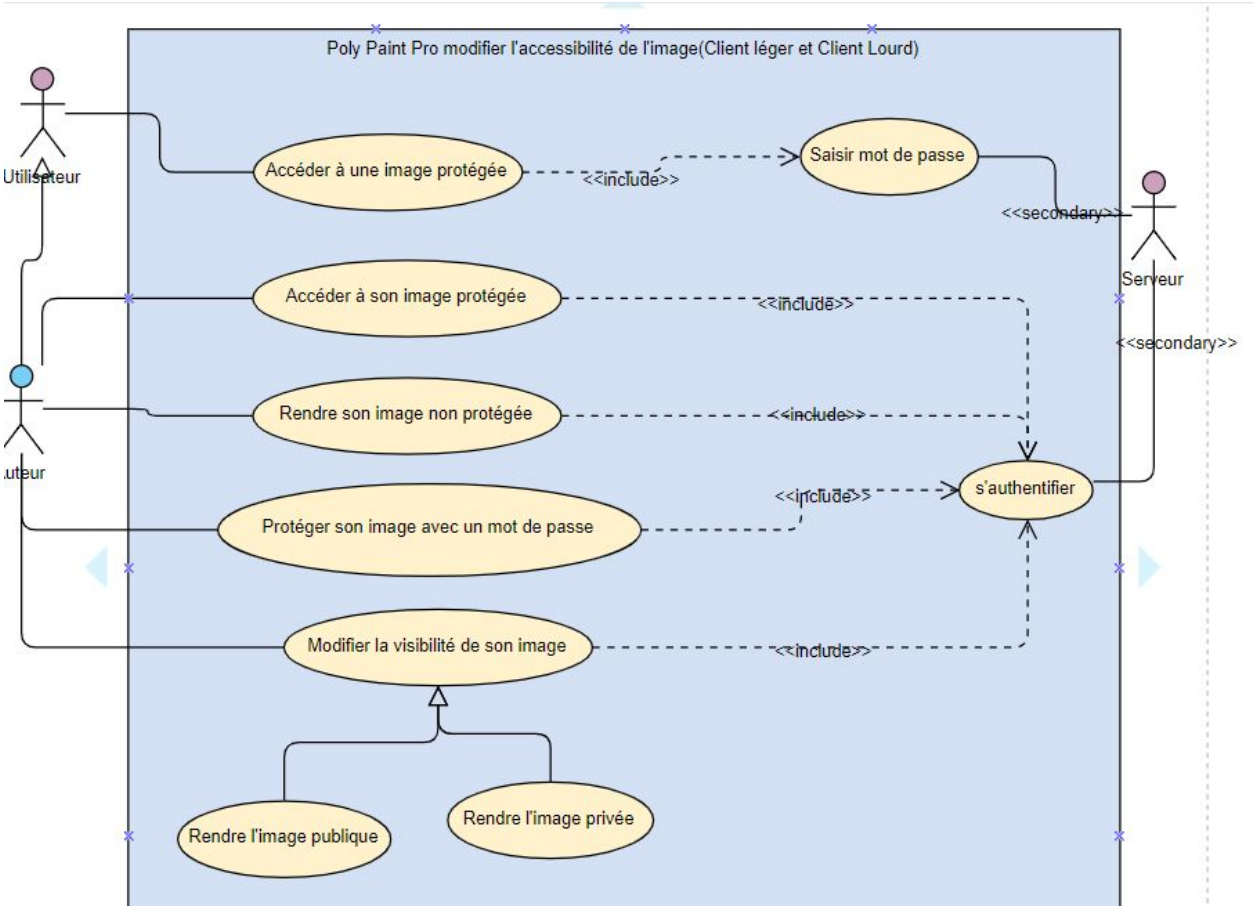

Figure 3:Diagramme de cas d’utilisation pour l'accessibilité de l’image

3.4. Diagramme de cas d'utilisation pour l'édition de formes

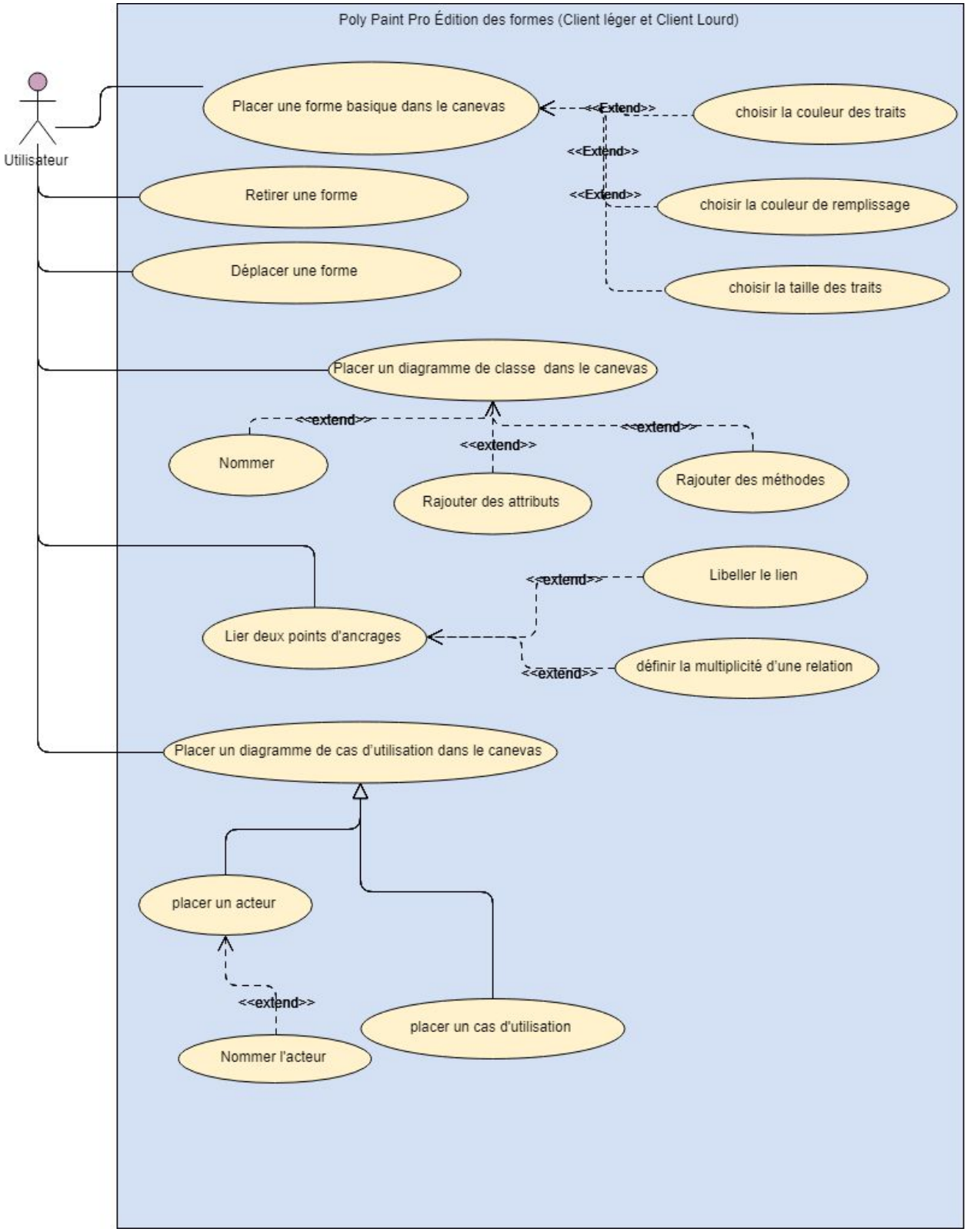

Figure 4: Diagramme de cas d'utilisation pour l'édition de formes

3.5. Diagramme de cas d'utilisation pour la gestion d'amis

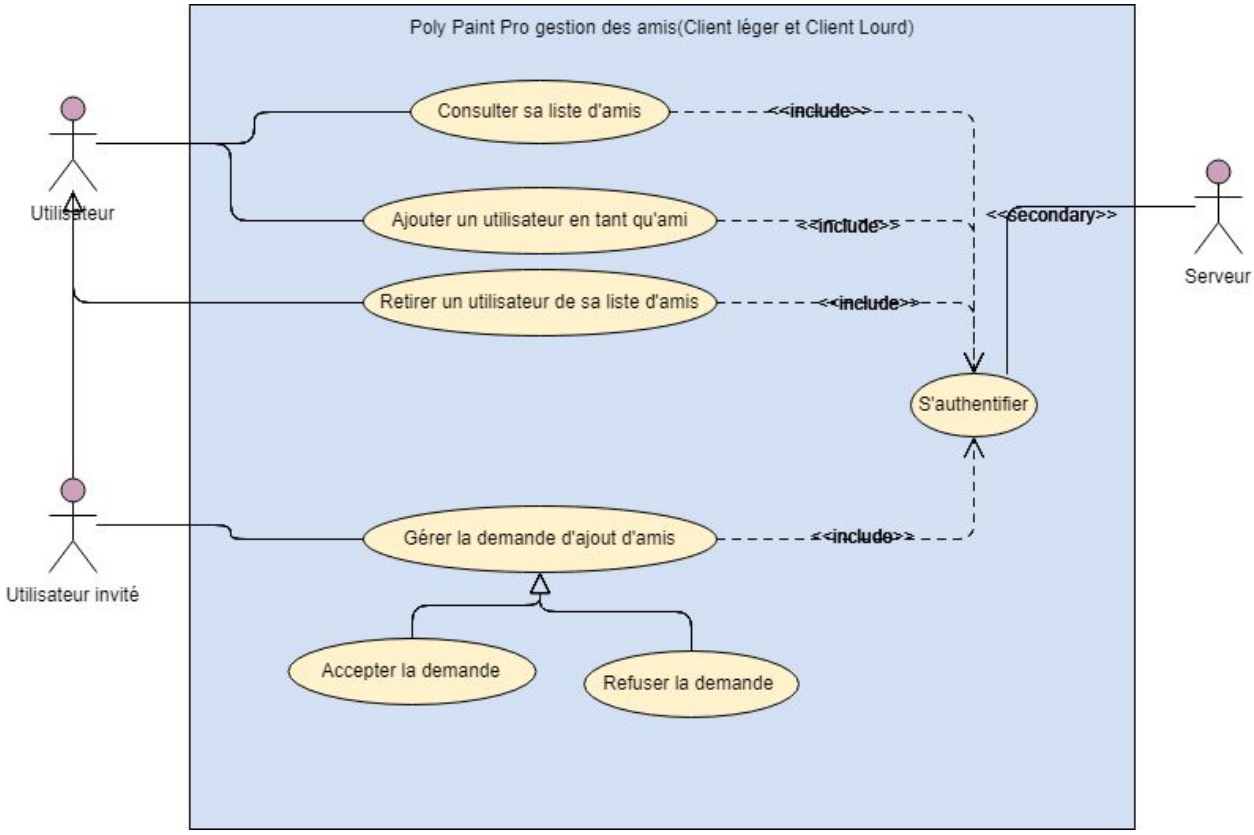

Figure 5: Diagramme de cas d'utilisation pour la gestion d'amis

3.6. Diagramme de cas d'utilisation pour le site web

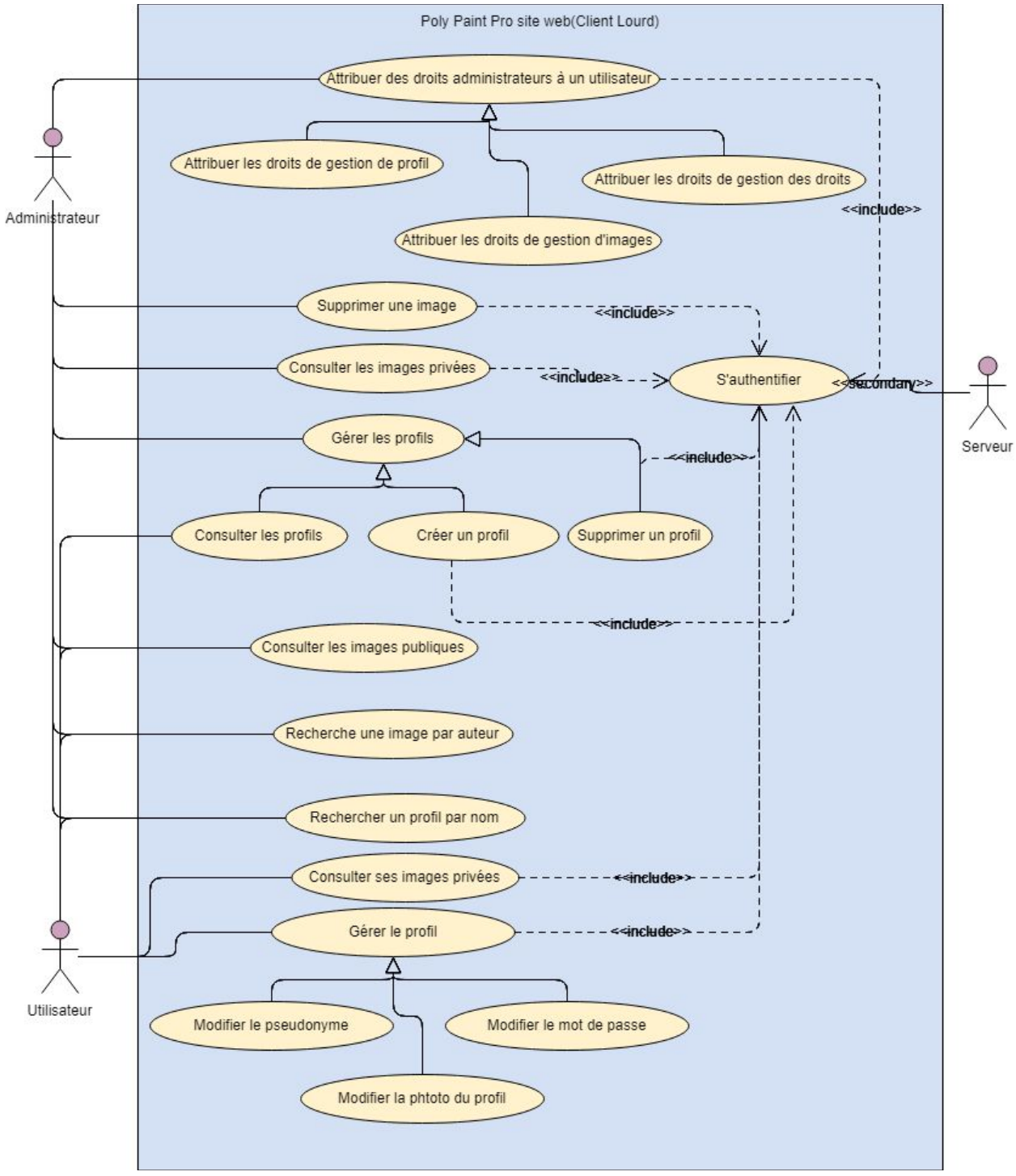

Figure 6: Diagramme de cas d'utilisation pour le site web

3.7. Diagramme de cas d’utilisation pour Galerie - j’aime et commentaires

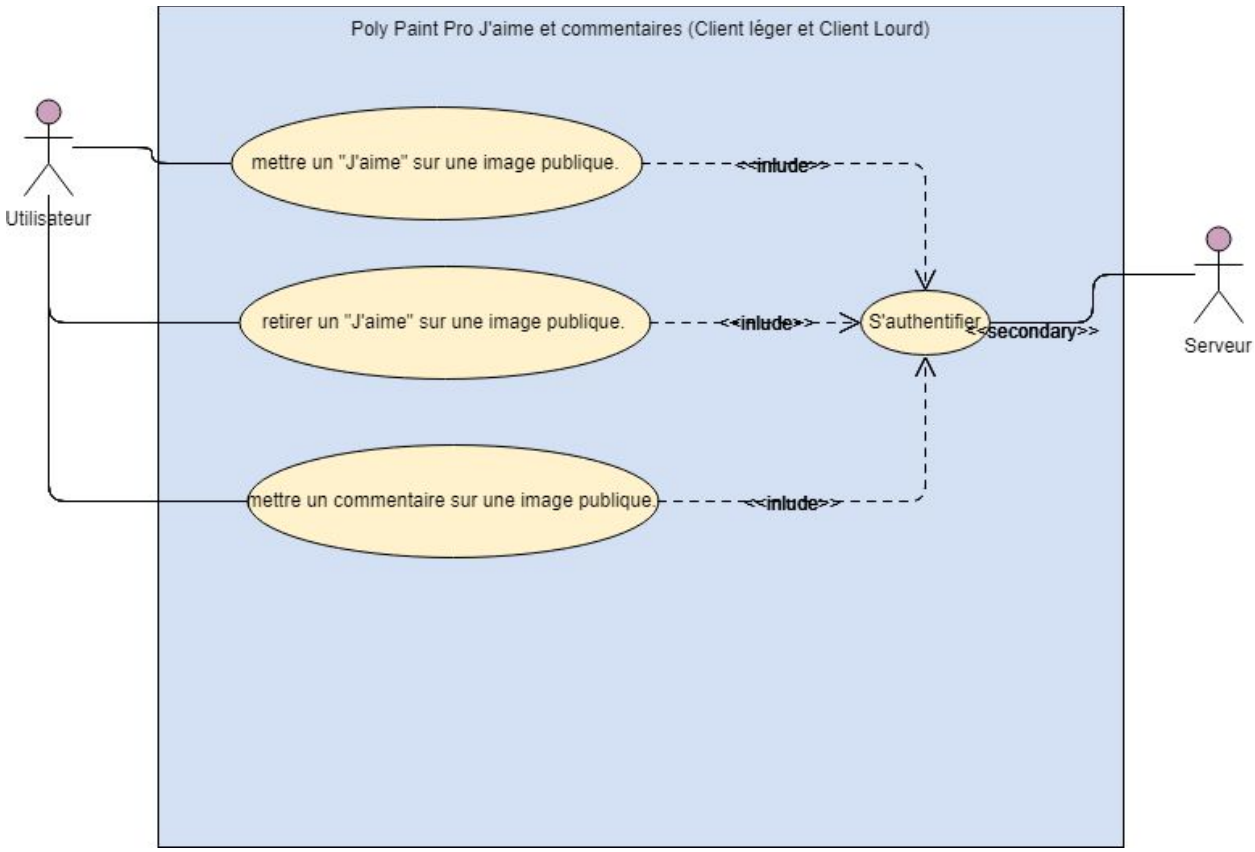

Figure 7: Diagramme de cas d’utilisation pour Galerie - j’aime et commentaires

4. Vue logique

Ci-dessous se trouvent les paquetages pour l'application Poly Paint Pro ainsi que la description de chacun.

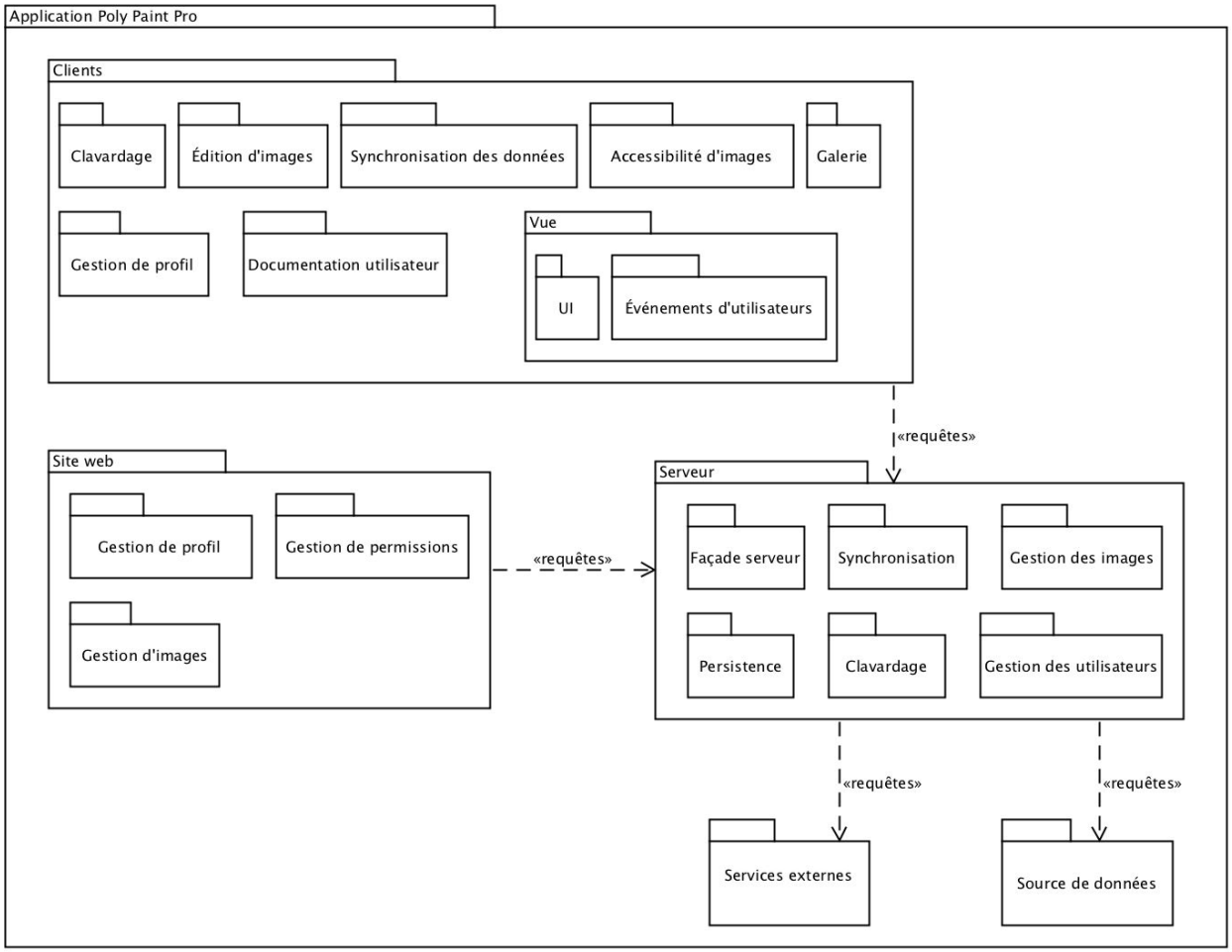

Figure 8 : Diagramme de paquetage pour le client lourd et léger (Poly Paint Pro)

Les paquetages suivants sont les paquetages dans le paquetage « Clients ». Ce paquetage représente les paquetages du client lourd et du client léger.

Clavardage
Ce paquetage regroupe l'ensemble des classes qui s'occupent des discussions entre les utilisateurs et des canaux de communications.

Édition d'images
Ce paquetage regroupe l'ensemble des classes qui s'occupent de l'édition des images et des formes.

Synchronisation des données
Ce paquetage regroupe les classes qui gèrent la synchronisation de données entre le client, le disque (persistance locale) et le serveur.

Accessibilité d'images
Ce paquetage regroupe l'accès aux images selon leur protection (public, privé, protégé).

Galerie
Ce paquetage regroupe les classes qui font la gestion de la galerie des images et des avis des utilisateurs.

Gestion de profil
Ce paquetage regroupe les classes qui font la gestion des profils des utilisateurs (création de comptes, authentification, gestion de mot de passe, photo de profil) et la gestion des amis.

Documentation utilisateur
Ce paquetage regroupe les classes qui gèrent des documents d'aide à l'utilisateur soit le tutoriel et les rubriques d'aides présents dans chacune des interfaces.

UI
Ce paquetage regroupe les classes qui représentent tous les éléments graphiques des interfaces.

Événements d'utilisateurs
Ce paquetage regroupe les classes qui gèrent tous les événements déclenchés par l'utilisateur soit des cliques, des entrées à partir du clavier et des interactions tactiles à partir de la tablette.

Les paquetages suivants sont ceux pour le site web.

Gestion de profil
Ce paquetage regroupe la gestion des profils des utilisateurs (création de comptes, gestion de mot de passe, photo de profil) .

Gestion des permissions
Ce paquetage regroupe les classes qui font la gestion des permissions des images et des utilisateurs (fait par l'administrateur).

Gestion d'images
Ce paquetage regroupe les classes qui gèrent les images, la galerie et les liens de partage d'images.

Les paquetages suivants sont ceux pour le serveur.

Façade serveur
Ce paquetage est un regroupement de classes qui gèrent les requêtes reçues et envoyées par les clients et le site web.

Synchronisation
Ce paquetage regroupe les classes qui s'occupent de la synchronisation des données entre les clients et les services de persistances (paquetage de persistance).

Persistance
Ce paquetage regroupe les classes qui s'occupent des requêtes vers les bases de données, des caches et les services stockage de données externes (ex. Amazon S3).

Gestion des images
Ce paquetage regroupe les classes qui gèrent l'accès aux images, à l'historique des images et à leurs versions.

Gestion des utilisateurs
Ce paquetage regroupe les classes qui gèrent l'authentification, la gestion de comptes et les permissions des utilisateurs.

Clavardage
Ce paquetage regroupe les classes qui gèrent la réception de messages et de la diffusion de messages vers les différents canaux.

Les paquetages suivants représentent les services Amazon Web Services (AWS) qui sont externes à notre application. Ceci comprend entre autres les bases de données ainsi que le système de stockage.

Services externes
Ce paquetage correspond aux services externes gérés par AWS.

Sources de données
Ce paquetage représente les bases de données de l'application.

La figure suivante représente le diagramme de classes des formes et éléments qui seront présents dans le canevas.

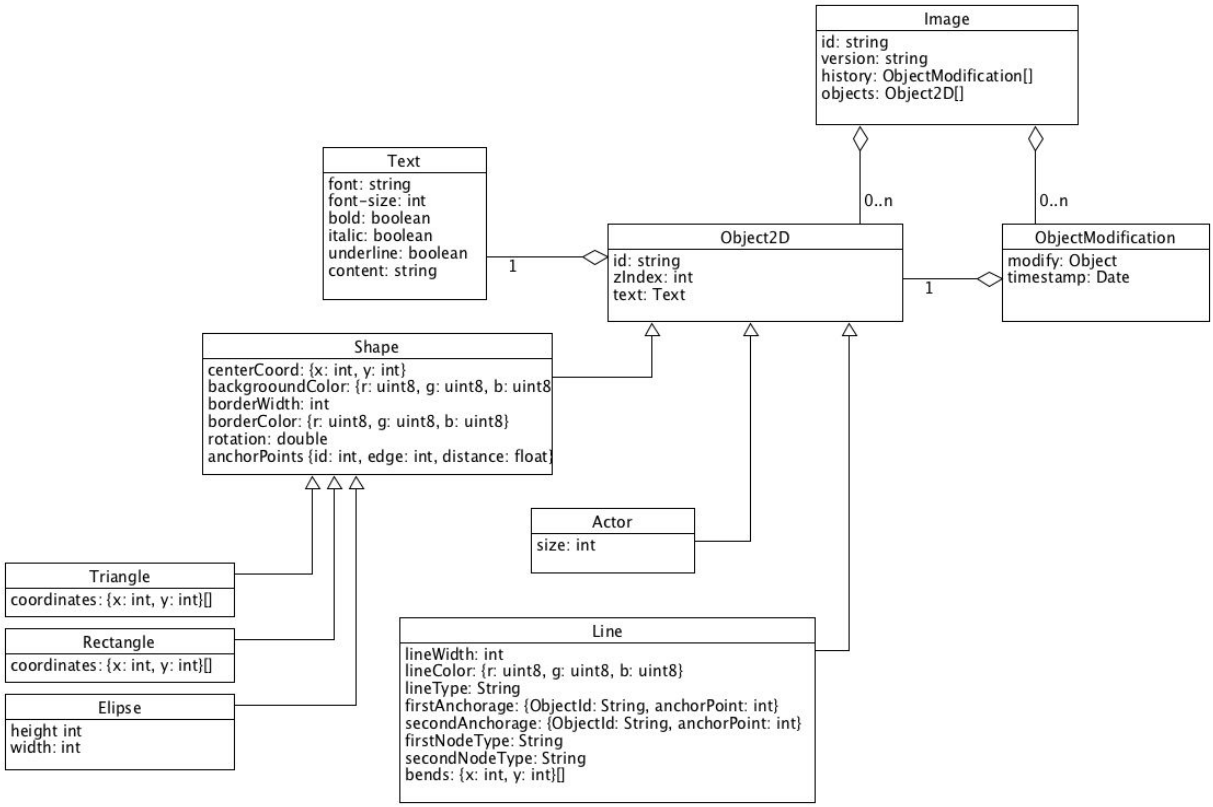

Figure 9: Diagramme de classes des formes

5. Vue des processus

5.1. Diagramme séquence système pour l'authentification

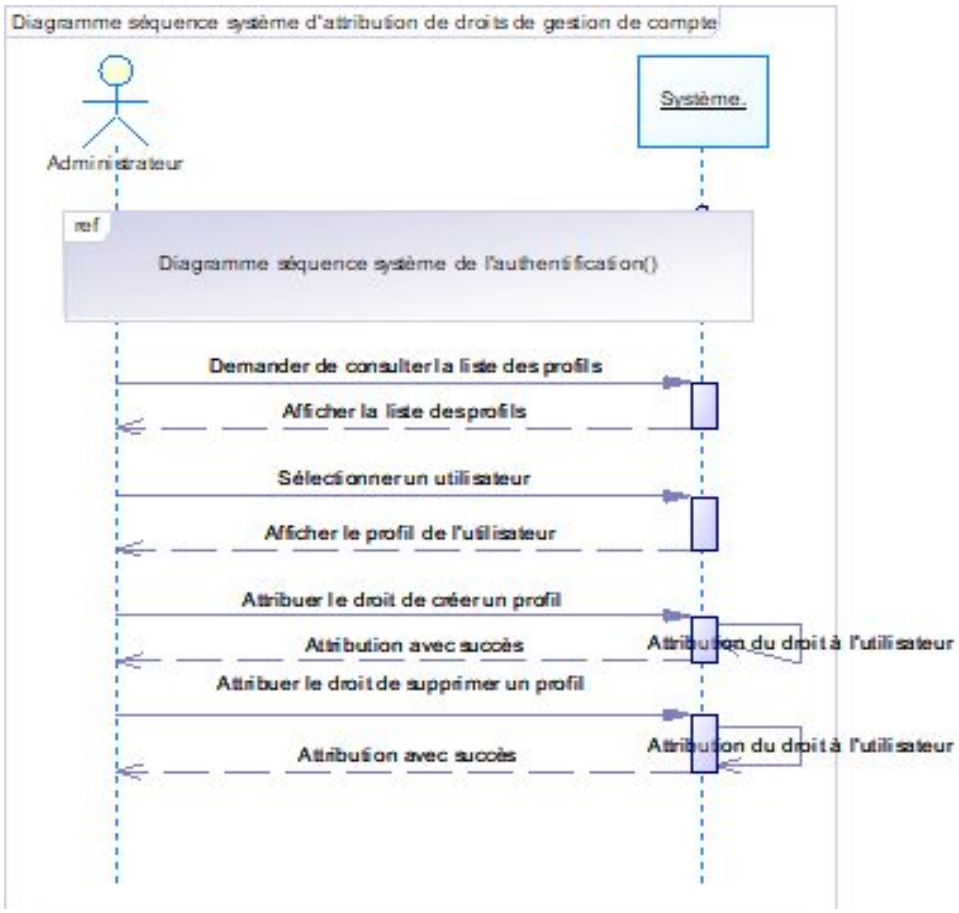

Figure 10: Diagramme séquence système pour l'authentification

5.2. Diagramme séquence système pour la création de canaux

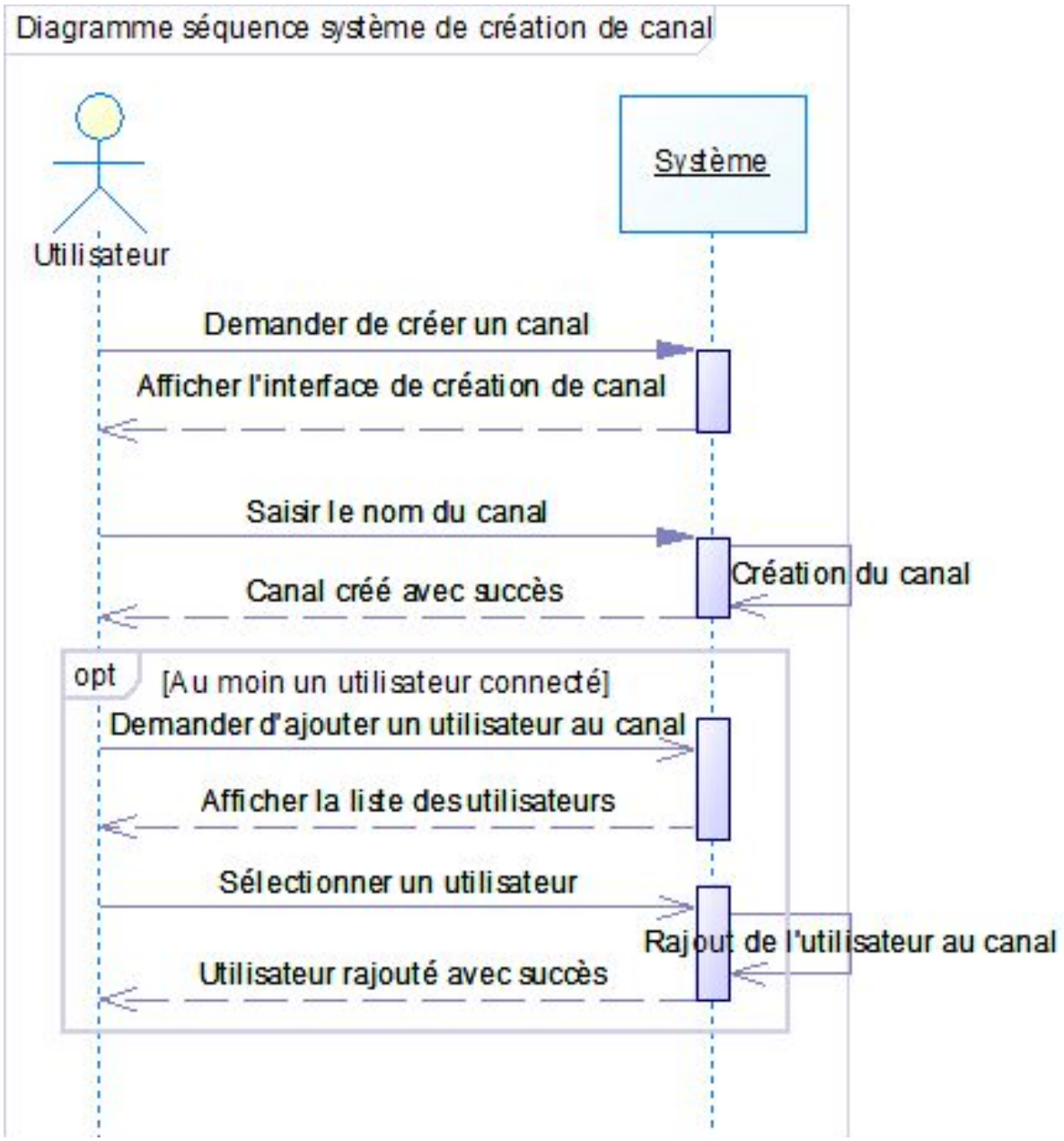

Figure 11: Diagramme séquence système pour la création de canaux

5.3. Diagramme séquence système pour l'ajout d'amis

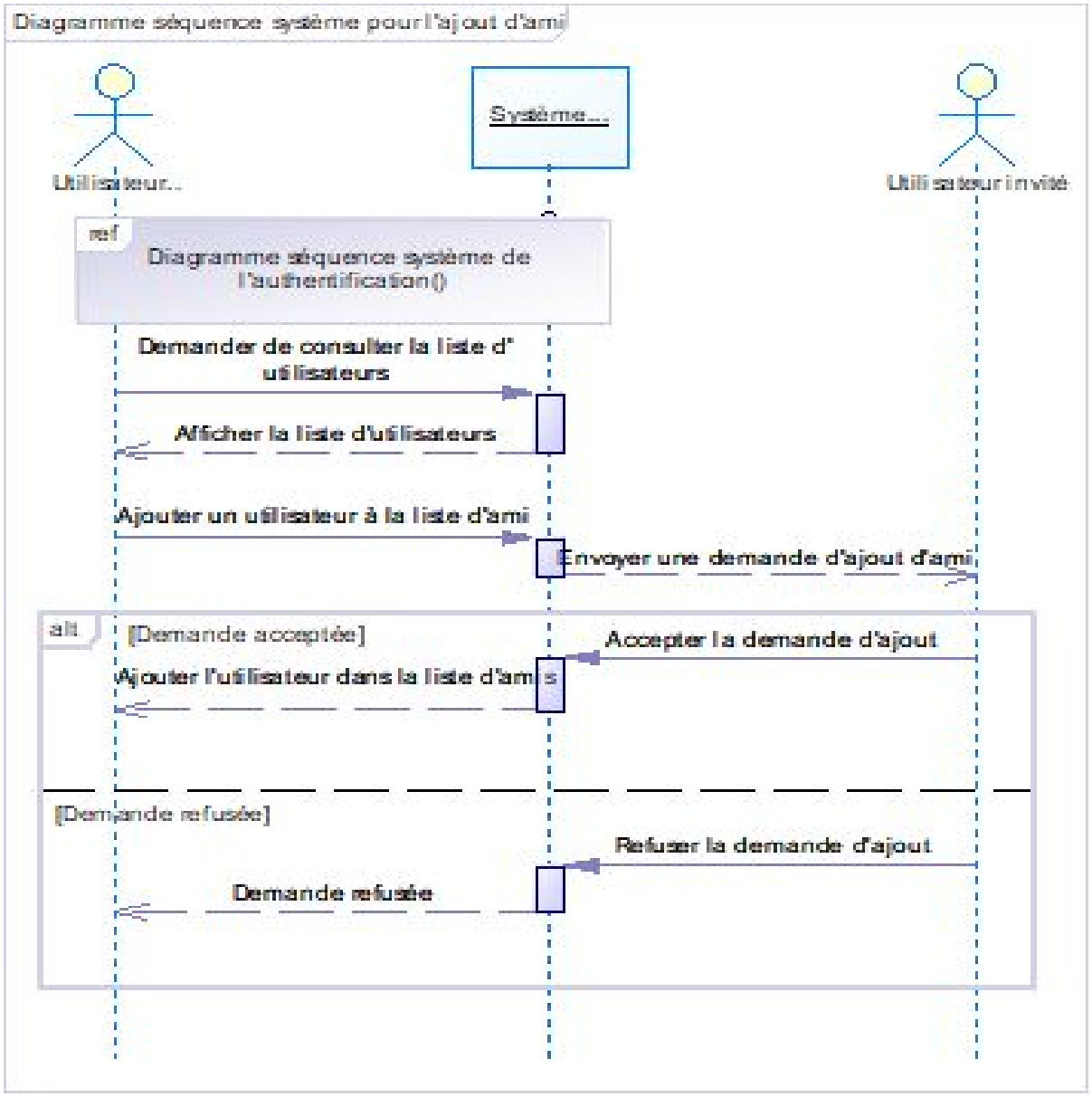

Figure 12: Diagramme séquence système pour l'ajout d'amis

5.4. Diagramme séquence système pour l'attribution de droits de gestion de compte

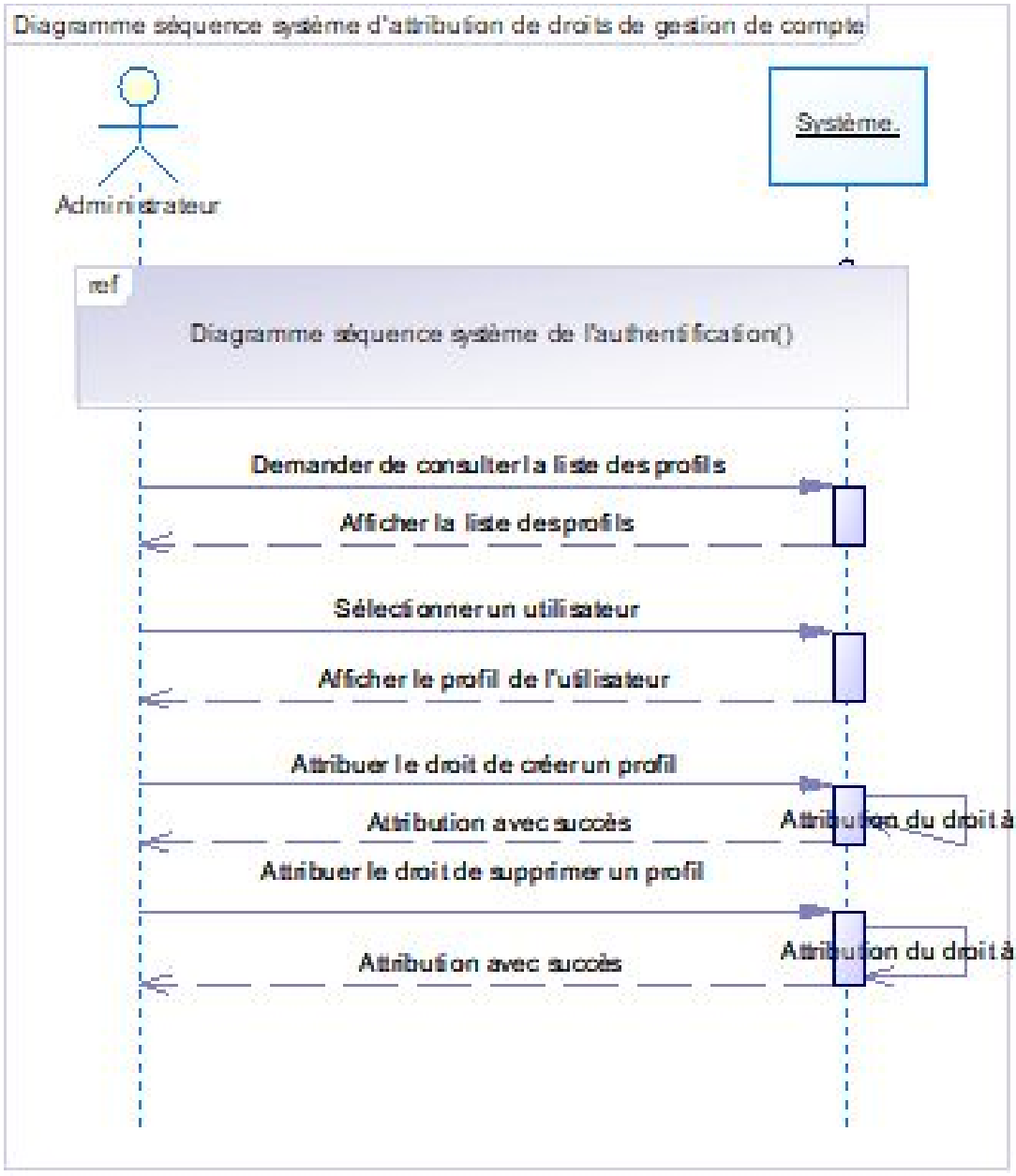

Figure 13: Diagramme séquence système pour l'attribution de droits de gestion de compte

5.5. Diagramme séquence système pour la gestion de profils

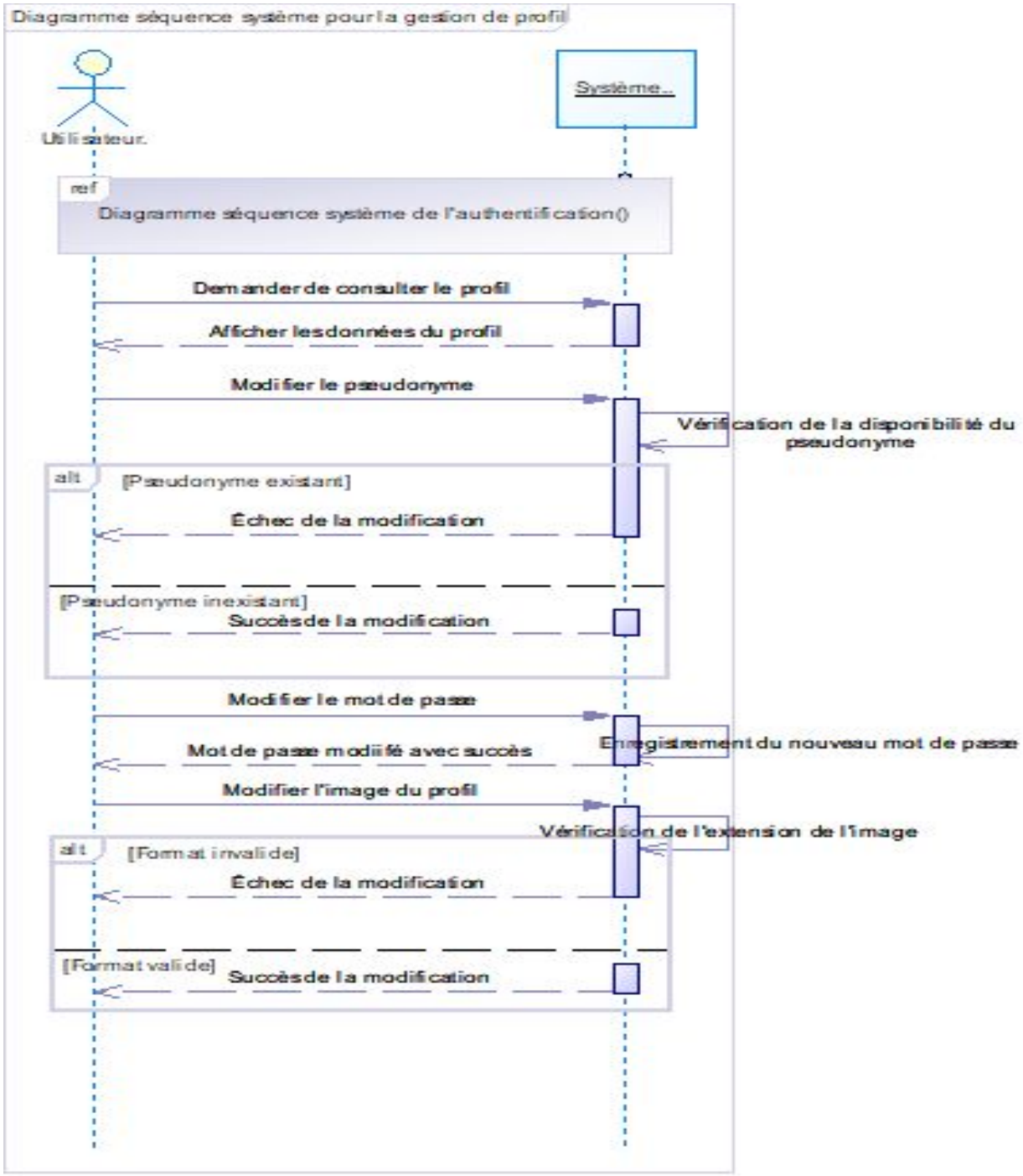

Figure 14: Diagramme séquence système pour la gestion de profils

6. Vue de déploiement

Ci-dessous est présenté le schéma de déploiement du projet qui présente les noeuds physiques et leurs interconnexions.

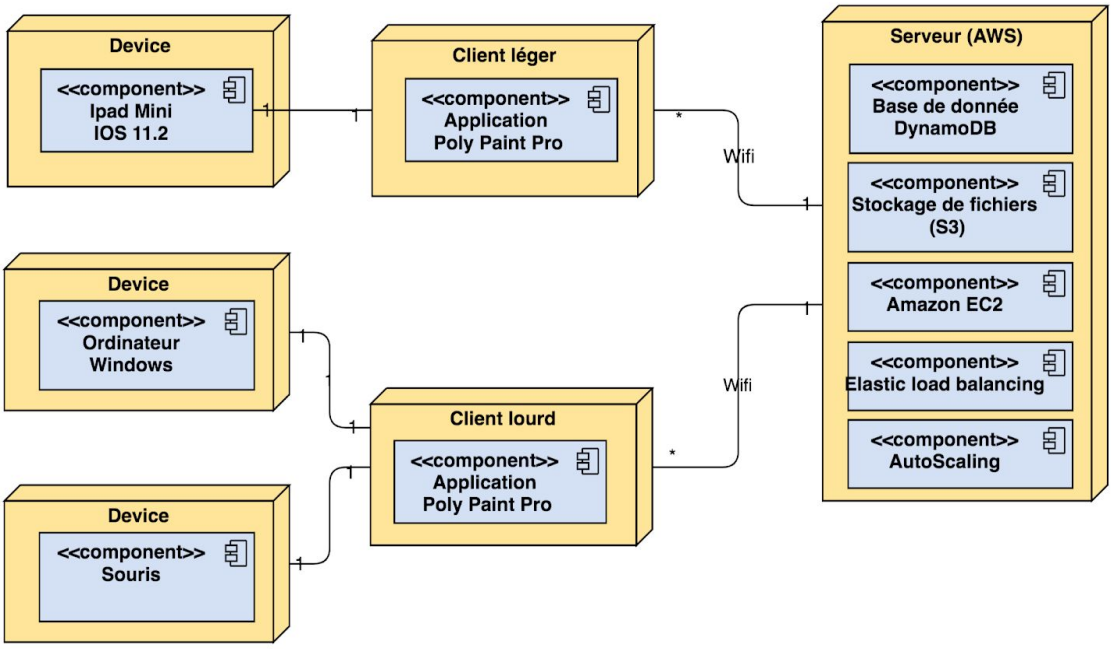

Figure XXX: Schéma de déploiement de notre système [https://online.visual-paradigm.com]

7. Taille et performance

7.1 Client léger

Pour le client léger, l'équipe 4 a choisi de développer l'application Poly Paint Pro sur la plateforme iOS. Plusieurs caractéristiques de fluidité et d'ergonomie doivent être respectées.

Le site d'Apple spécifie les performances et les spécificités techniques de notre tablette[1] . La taille de notre projet ne devra pas dépasser la mémoire maximale de l'iPad mini 4 qui est de 128Go et la quantité de mémoire vive de l'iPad mini 4 (2Go). Ces spécifications devront être suffisantes pour faire rouler l'application de manière fluide.

7.2 Client lourd

Puisque le client lourd sera utilisé sur un ordinateur ayant le système d'exploitation Windows 10, l'ordinateur doit avoir minimalement les spécifications suivantes. Le processeur doit avoir une vitesse de 1 GHz, une RAM de 1Go pour un système de 32 bits ou d'un RAM de 2Go pour un système de 64 bits. De plus, le disque dur doit être de 16Go pour un système de 32 bits ou de 20Go pour un système de 64 bits et la carte graphique doit être compatible avec DirectX 9 ou supérieur et avec un modèle de pilote d'affichage Windows (WDDM) 1.0. Finalement, l'écran doit avoir une résolution minimale de 800x600 pixels.[2]

[1] Site web d'apple: https://www.apple.com/ca/fr/ipad-mini-4/specs/
[2] Site de microsoft: https://www.microsoft.com/en-ca/windows/windows-10-specifications

7.3 Serveur

Le serveur se compose de plusieurs services essentiels d'Amazon Web Services (AWS). Premièrement, nous allons utiliser le service S3 d'AWS pour stocker les fichiers du site web, ce qui comprend entre autres les images sauvegardées. Afin de sauvegarder les données des utilisateurs, nous allons utiliser MongoDB sur le service DynamoDB d'AWS. Nous allons héberger notre serveur sur une machine virtuelle EC2 et nous allons gérer nos requêtes avec le service Elastic Load Balancing, deux services offerts par AWS. Le cadriciel utilisé pour notre serveur sera Node.js.

Puisque nous utilisons les services AWS, nous n'avons pas à nous soucier de la performance et de l'évolutivité de notre matériel. Toutefois, nous devons prendre en compte que si nous utilisons ces services trop extensivement, nous allons devoir payer en conséquence.