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

[**1. Introduction**](#_gjdgxs) **5**

[**2. Objectifs et contraintes architecturaux**](#_30j0zll) **5**

[2.1 Serveur](#_g5zikfweqvbc) 5

[2.1.1 L’objectif et l'importance](#_pjjrsh8u708g) 5

[2.1.2 La sécurité et confidentialité des données](#_25rk3ga7gxhi) 5

[2.1.3 La fiabilité](#_xpe3zkc273i1) 5

[2.1.4 L’intégrité](#_so3duje03vqt) 5

[2.2 Le client lourd](#_e4z1g4x3f83z) 6

[2.2.1 L’objectif et l'importance](#_tito952das4y) 6

[2.2.2 La portabilité](#_xssvs0h3u0sd) 6

[2.2.3 Le langage de programmation](#_9tieza3bjqjg) 6

[2.2.4 Robustesse](#_5bvcecif2ngh) 6

[2.3 Client Léger](#_3vbub85nahvj) 6

[2.3.1 L’objectif et l'importance](#_dj9n6thzewxi) 6

[2.2.2 La portabilité](#_bsqiwiaaijds) 6

[2.3.3 Langage de programmation](#_eykzwyhoweuz) 6

[2.3.4 Robustesse](#_4z5l6n37ga2x) 6

[**3. Vue des cas d’utilisation**](#_3znysh7) **7**

[3.1. Diagramme de cas d’utilisation pour le clavardage](#_ag5tis3s0l4u) 7

[3.2. Diagramme de cas d’utilisation pour la gestion de profils](#_ws553bli1px1) 8

[3.3. Diagramme de cas d’utilisation pour l'accessibilité de l’image](#_nks33lcqhurk) 9

[3.4. Diagramme de cas d’utilisation pour l’édition de formes](#_zgs49ep0q16n) 10

[3.5. Diagramme de cas d’utilisation pour la gestion d’amis](#_36ng0yfzd4n) 11

[3.6. Diagramme de cas d’utilisation pour le site web](#_no5i6l3vplox) 12

[3.7. Diagramme de cas d’utilisation pour Galerie - j’aime et commentaires](#_bhya4auus25b) 13

[**4. Vue logique**](#_4rp135j7a8d5) **14**

[**5. Vue des processus**](#_tyjcwt) **18**

[5.1. Diagramme séquence système pour l’authentification](#_2vrgf4nuq4eu) 18

[5.2. Diagramme séquence système pour la création de canal](#_55oecdijvfoh) 19

[5.3. Diagramme séquence système pour l’ajout d’amis](#_bdapb7ir4w7v) 20

[5.4. Diagramme séquence système pour l’attribution de droits de gestion de compte](#_ofgqdvkgia32) 21

[5.5. Diagramme séquence système pour la gestion de profils](#_rktemail4dmb) 22

[**6. Vue de déploiement**](#_3dy6vkm) **23**

[**7. Taille et performance**](#_1t3h5sf) **23**

[7.1 Client léger](#_2jkrjdkj8imn) 23

[7.2 Client lourd](#_8vu8e59g0be3) 23

[7.3 Serveur](#_798gy4cdf4q4) 24

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 RDS pour la base de données relationnelle PostgreSQL. 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

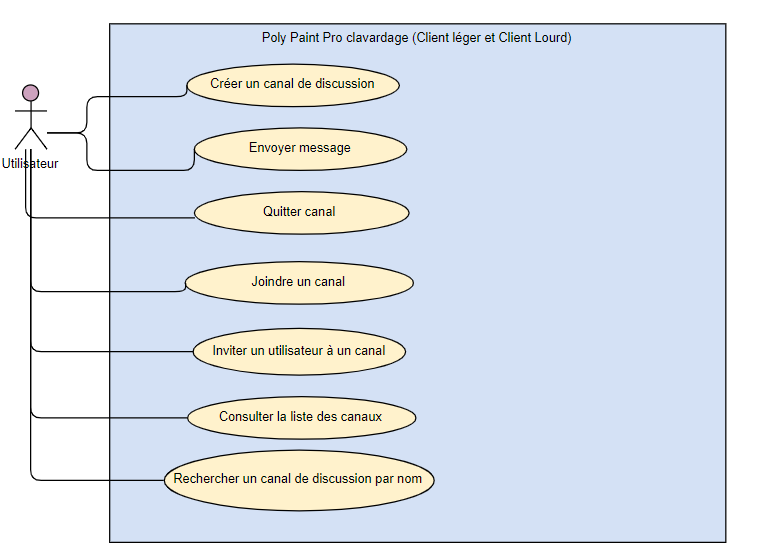
Le client léger est développé avec le langage Swift en utilisant la plateforme Xcode. Le développement sur iOS est principalement basé sur le patron modèle vue contrôleur (MVC). Ce patron de conception est incontournable dans le développement des applications mobiles puisqu’il isole ses différentes composantes 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.

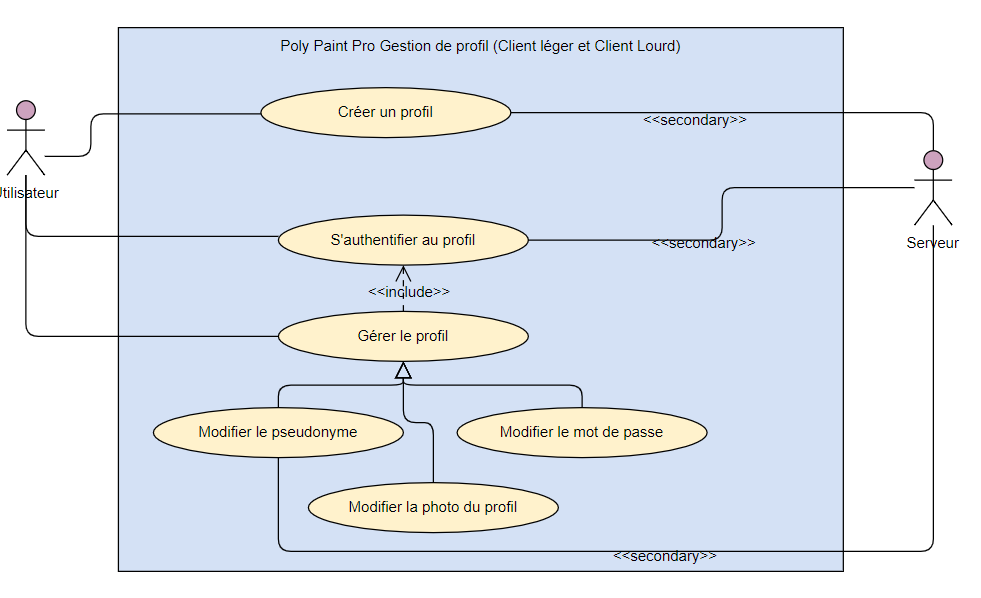
# 3. Vue des cas d’utilisation

## 3.1. Diagramme de cas d’utilisation pour le clavardage



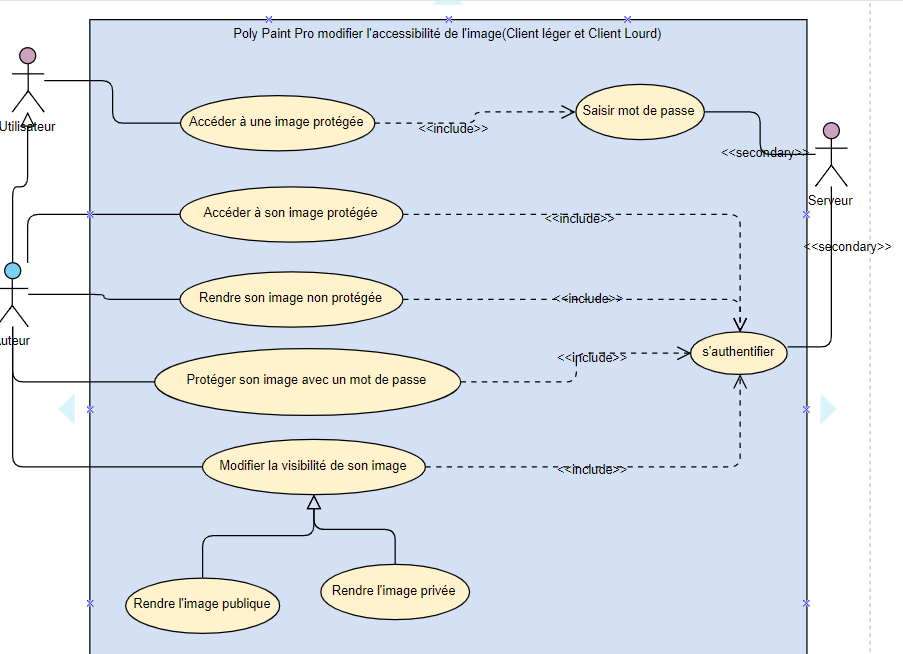
*Figure 1: Diagramme de cas d’utilisation pour le clavardage*

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



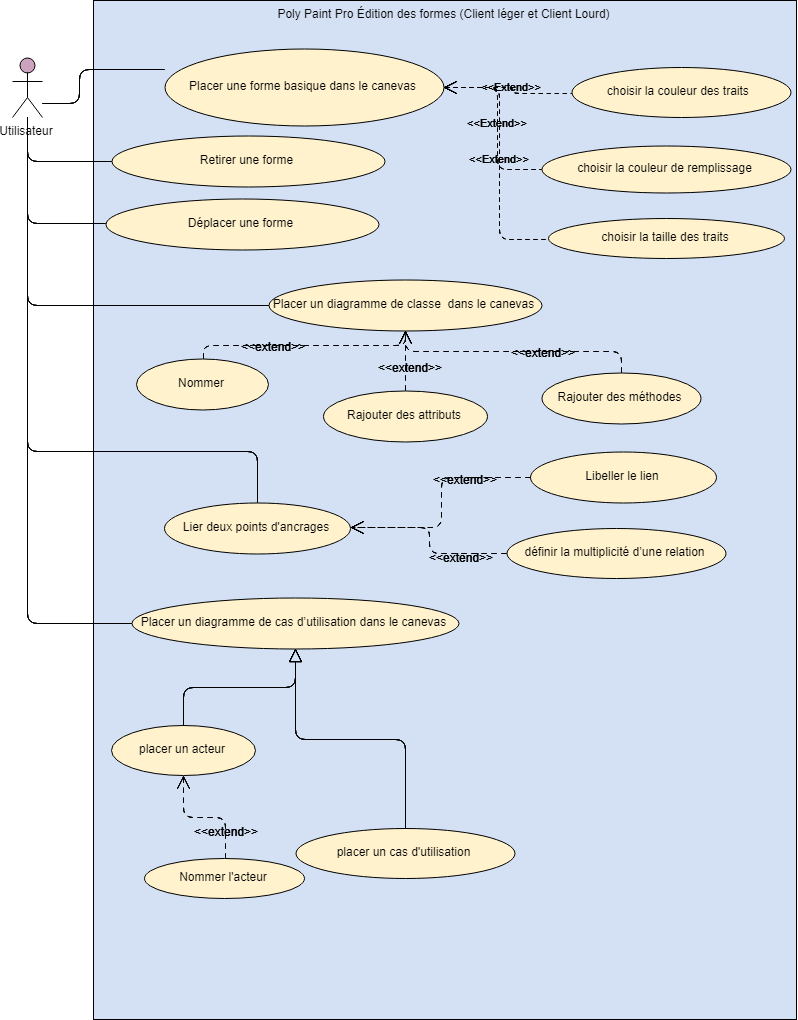
*Figure 2: Diagramme de cas d’utilisation pour la gestion de profils*

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



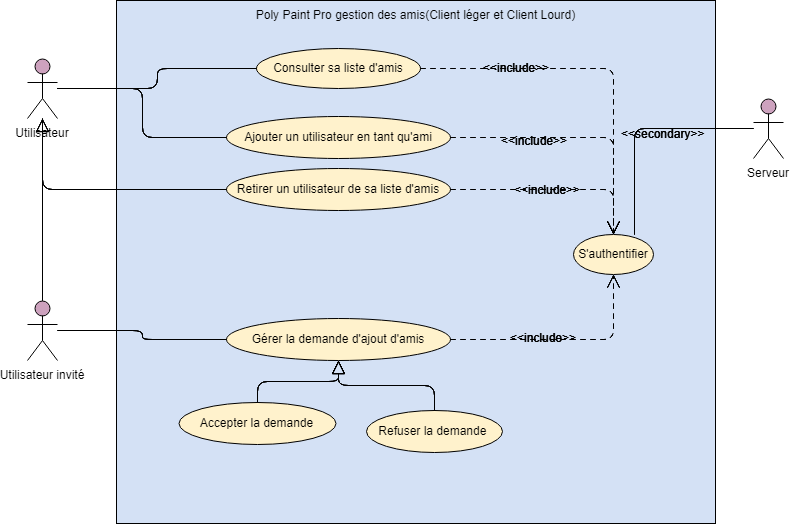
*Figure 3:Diagramme de cas d’utilisation pour l'accessibilité de l’image*

## 3.4. Diagramme de cas d’utilisation pour l’édition de formes



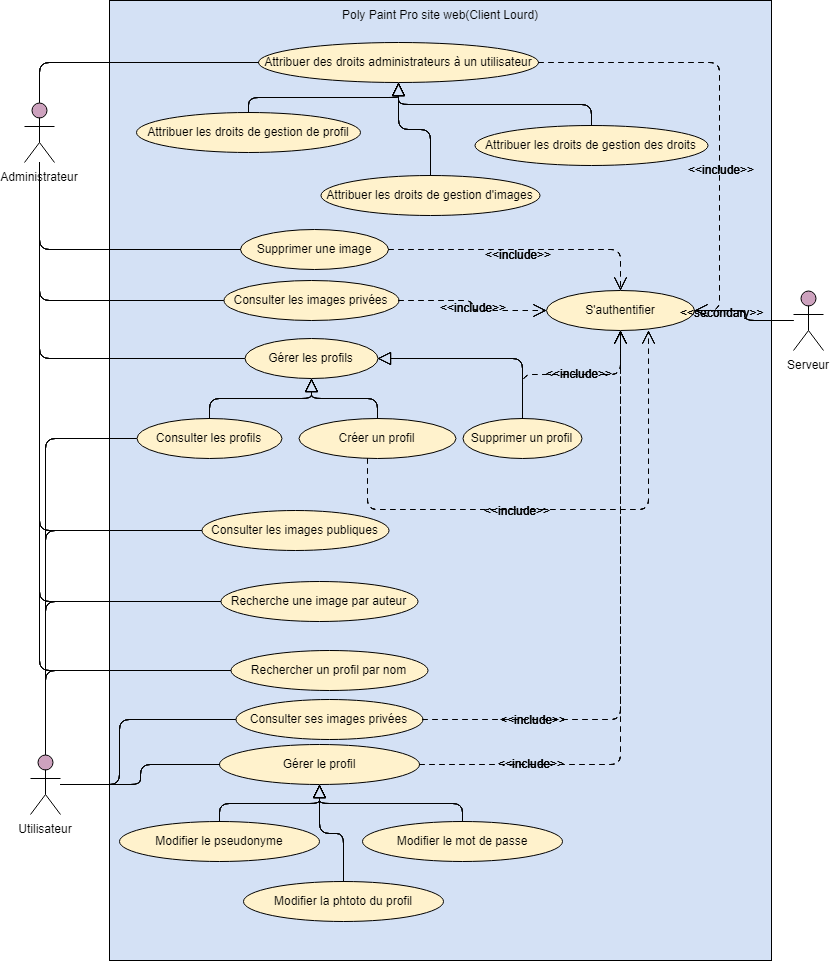
*Figure 4: Diagramme de cas d’utilisation pour l’édition de formes*

## 3.5. Diagramme de cas d’utilisation pour la gestion d’amis



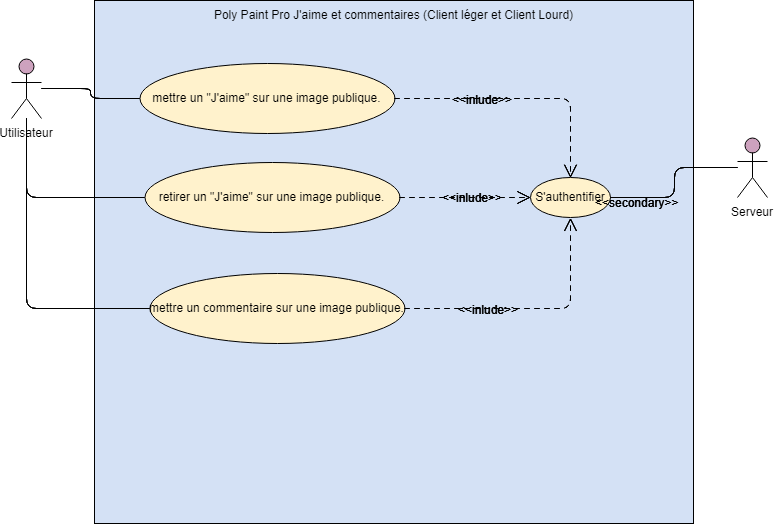
*Figure 5: Diagramme de cas d’utilisation pour la gestion d’amis*

## 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

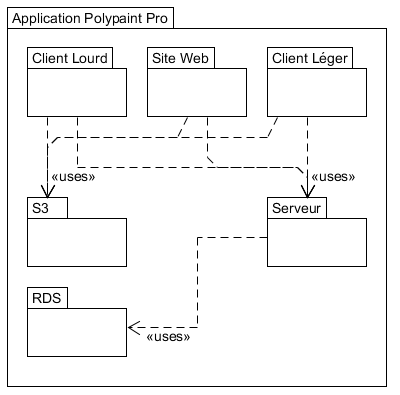


*Figure 7: Diagramme de cas d’utilisation pour Galerie - j’aime et commentaires*

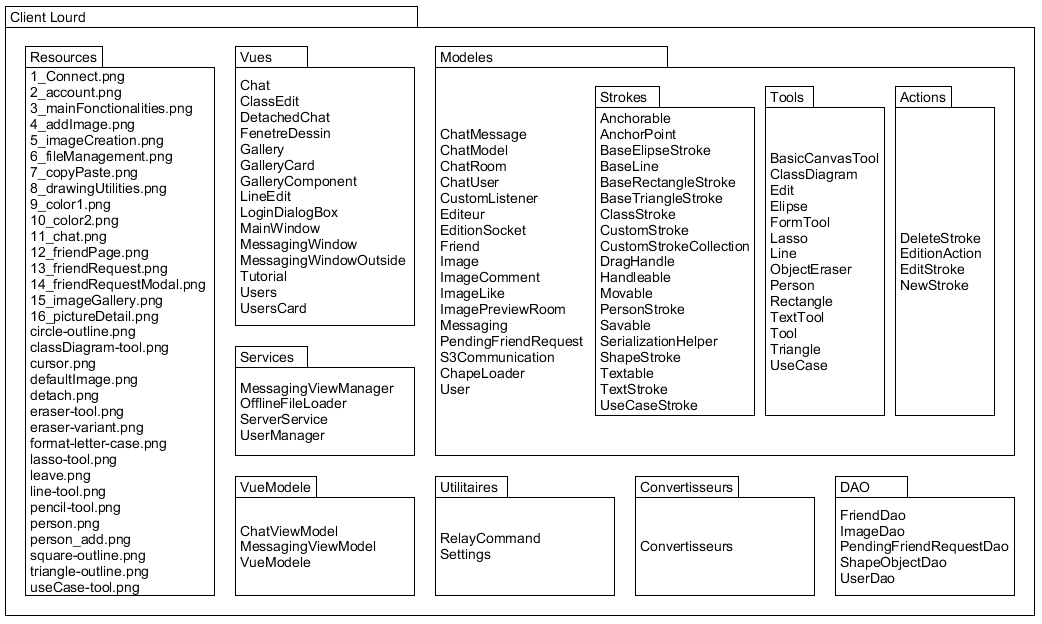
# 

# 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 l’ensemble des divisions de Polypaint Pro*



*Figure 9: Diagramme de paquetage pour le client lourd*

Les paquetages suivants sont les paquetages dans le paquetage « Client Lourd».

|  |
| --- |
| **Ressources** |
| Ce paquetage regroupe toutes les images et les icônes utilisées par l’application. |

|  |
| --- |
| **Vues** |
| Ce paquetage regroupe l’ensemble des fenêtres, pages et sections. Les vues XAML et leurs fichiers associés s’y retrouvent. |

|  |
| --- |
| **Services** |
| Ce paquetage regroupe les classes qui offrent des services statiques à l’ensemble du projet. |

|  |
| --- |
| **VueModele** |
| Ce paquetage regroupe les classes VueModele de l’architecture MVVM. |

|  |
| --- |
| **Modeles** |
| Ce paquetage regroupe les classes qui gèrent la logique de l’application. |

|  |
| --- |
| **Strokes** |
| Ce paquetage regroupe l’ensemble des classes qui représentent les différentes formes et objets pouvant être mis dans le projet (rectangle, ligne, acteur, etc.). |

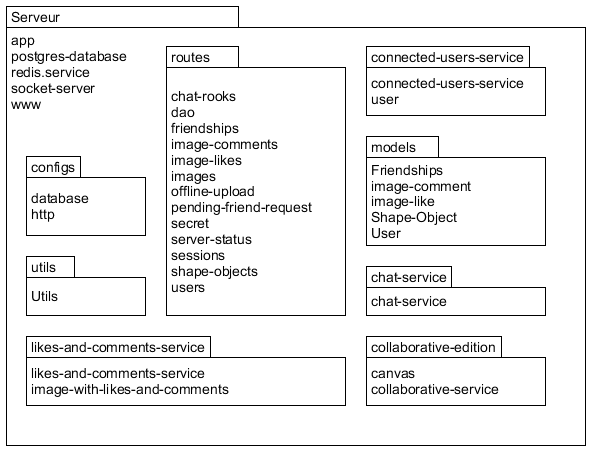
|  |
| --- |
| **Tools** |
| Ce paquetage regroupe les classes qui représentent les différents outils pouvant être utilisés dans l’édition d’image (lasso, rectangle, classe, etc.). |

|  |
| --- |
| **Actions** |
| Ce paquetage regroupe les classes représentant une action d’édition. Ces classes sont utilisées pour les boutons “précédents” et “refaire”. |

|  |
| --- |
| **Utilitaires** |
| Ce paquetage regroupe les paramètres de l’application. |

|  |
| --- |
| **Convertisseurs** |
| Ce paquetage regroupe les convertisseurs pour les liaisons de données. |

|  |
| --- |
| **Dao** |
| Ce paquetage regroupe les classes offrant une interface avec le serveur. |



*Figure 10: Diagramme de paquetage pour le serveur*

Les paquetages suivants sont les paquetages dans le paquetage «Serveur».

|  |
| --- |
| **Serveur** |
| Ce paquetage contient l’ensemble de l’application et les classes de base du projet. |

|  |
| --- |
| **configs** |
| Ce paquetage regroupe les fichiers de configuration avec les paramètres du serveur. |

|  |
| --- |
| **utils** |
| Ce paquetage regroupe des fichiers utilitaires génériques utilisables par tout le serveur. |

|  |
| --- |
| **routes** |
| Ce paquetage regroupe l’ensemble des méthodes exécutées lorsqu’une requête est faite au serveur. |

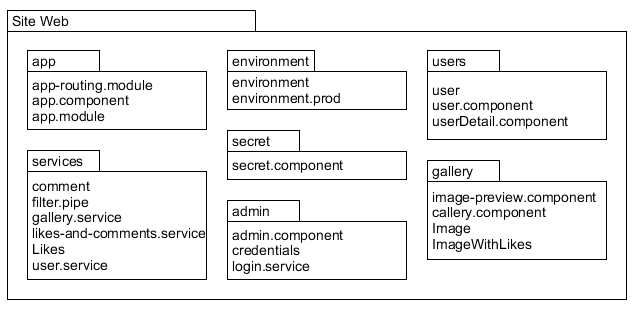
|  |
| --- |
| **likes-and-comments-service** |
| Ce paquetage contient le service utilisable pour les "j’aime" et commentaires. |

|  |
| --- |
| **connected-users-service** |
| Ce paquetage contient le service qui gère les connexions et les sockets des utilisateurs. |

|  |
| --- |
| **models** |
| Ce paquetage regroupe les fonctions appelables par les routes et qui font les requêtes à la base de données et le filtrage des résultats. |

|  |
| --- |
| **chat-service** |
| Ce paquetage contient le service qui gère les différents canaux de communication ainsi que la réception et la transmission des images. |

|  |
| --- |
| **collaborative-edition** |
| Ce paquetage contient le service qui gère le mode d’édition de tous les images et leurs éditeurs. |



*Figure 11: Diagramme de paquetage pour le client lourd*

Les paquetages suivants sont les paquetages dans le paquetage «Site Web».

|  |
| --- |
| **app** |
| Ce paquetage regroupe la base du site web. Ce paquetage est le conteneur de tous les autres composants du site web. |

|  |
| --- |
| **services** |
| Ce paquetage regroupe les services qui gèrent l’état de l’application. |

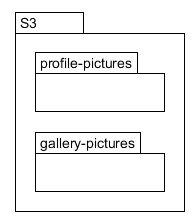
|  |
| --- |
| **environment** |
| Ce paquetage regroupe les classes qui gèrent le fonctionnement de l'application avant et après le déploiement. |

|  |
| --- |
| **secret** |
| Ce paquetage regroupe le composant qui affiche une image privée partagée par URL. |

|  |
| --- |
| **admin** |
| Ce paquetage regroupe les classes qui gèrent le mode administrateur. |

|  |
| --- |
| **users** |
| Ce paquetage contient le composant de l’application qui affiche la liste d’utilisateurs. |

|  |
| --- |
| **gallery** |
| Ce paquetage contient le composant de l’application qui affiche la liste des images.. |

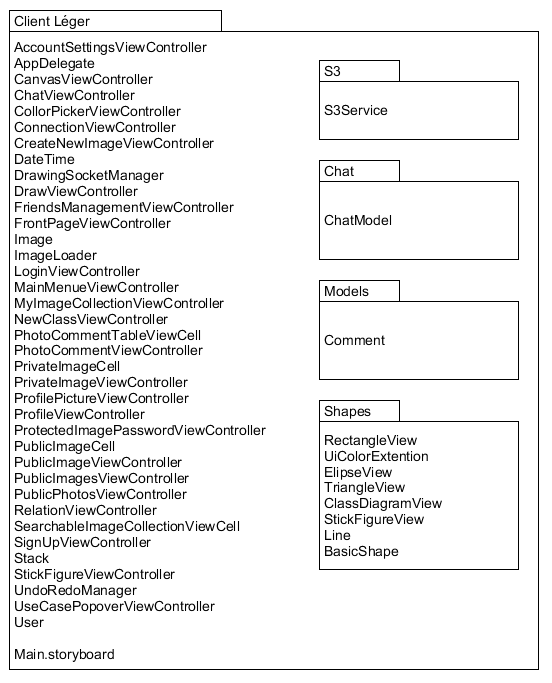


*Figure 12: Diagramme de paquetage pour S3*

Les paquetages suivants sont les paquetages dans le paquetage «S3».

|  |
| --- |
| **profile-pictures** |
| Ce paquetage représente le “bucket” S3 qui contient toutes les images de profil des utilisateurs. |

|  |
| --- |
| **gallery-pictures** |
| Ce paquetage représente le “bucket” S3 qui contient toutes les images qui ont été dessinées par les utilisateurs. |



*Figure 13: Diagramme de paquetage pour S3*

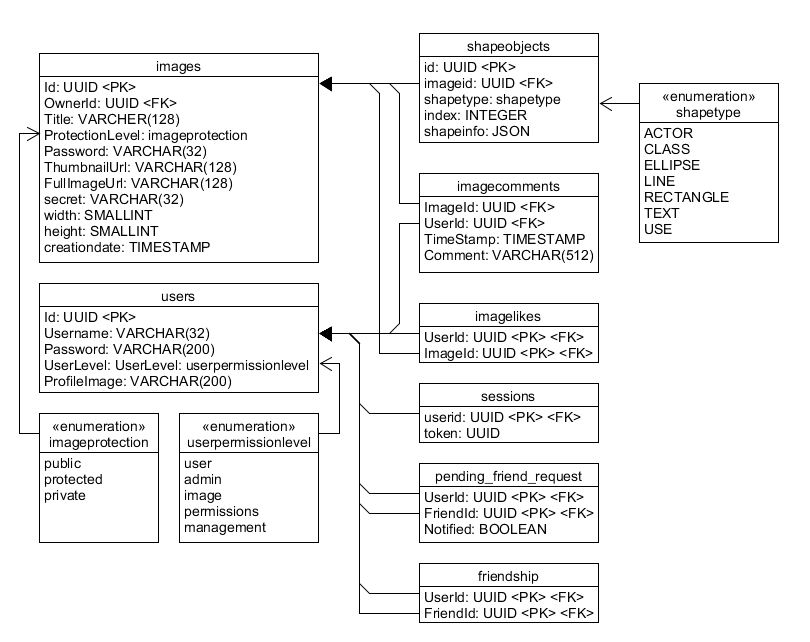
|  |
| --- |
| **Client Leger** |
| Ce paquetage contient les classes qui gèrent tout le projet les VueModele |

|  |
| --- |
| **S3** |
| Ce paquetage contient le service qui permet la connexion à S3. |

|  |
| --- |
| **Chat** |
| Ce paquetage contient les classes qui gèrent le chat et les messages. |

|  |
| --- |
| **Models** |
| Ce paquetage contient la logique des commentaires. |

|  |
| --- |
| **Shapes** |
| Ce paquetage contient les différentes formes pouvant être ajoutées au dessin. |



*Figure 14: Modèle de relation de la base de données relationnelle*

Les descriptions suivantes sont les différentes tables de la base de données relationnelle.

|  |
| --- |
| **images** |
| Cette table contient l’ensemble des images créées par les utilisateurs ainsi que leurs paramètres. |

|  |
| --- |
| **users** |
| Cette table contient l’ensemble des utilisateurs de l’application ainsi que leur niveau de permission. |

|  |
| --- |
| **shapeobject** |
| Cette table contient l’ensemble des objets placés sur les images. |

|  |
| --- |
| **imagecomments** |
| Cette table contient l’ensemble des commentaires laissés à une image. |

|  |
| --- |
| **imagelikes** |
| Cette table contient l’ensemble des "j’aime" laissés à une image. |

|  |
| --- |
| **sessions** |
| Cette table contient l’ensemble des sessions ouvertes par les utilisateurs. |

|  |
| --- |
| **pending\_friend\_request** |
| Cette table contient l’ensemble des requêtes d’amis pas encore acceptées. |

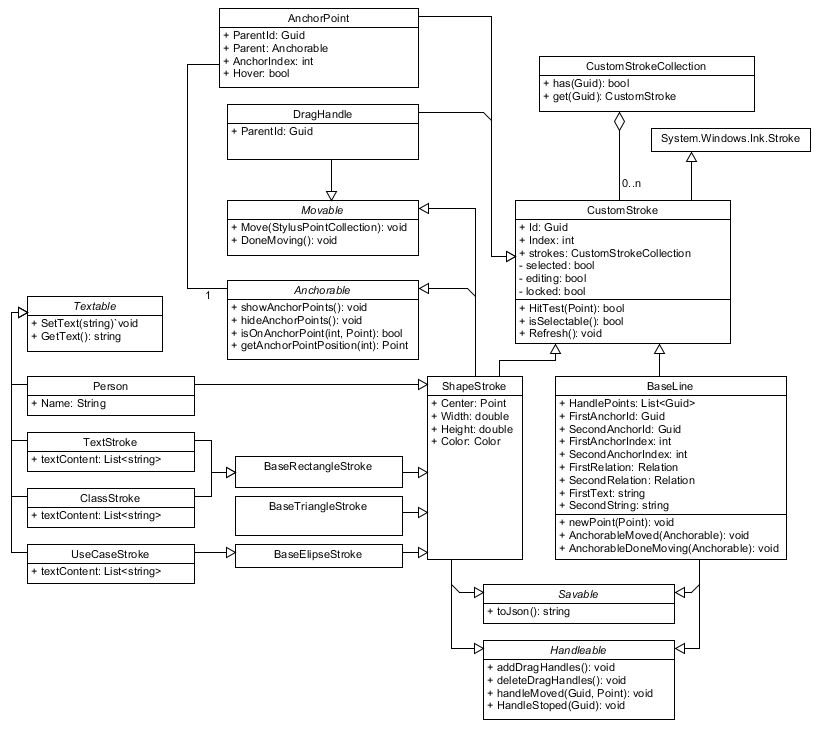
|  |
| --- |
| **friendship** |
| Cette table contient l’ensemble des liens d’amitié entre deux utilisateurs. |

|  |
| --- |
| **shapetype** |
| Cette énumération représente les différents types de formes dans un dessin. |

|  |
| --- |
| **imageprotection** |
| Cette énumération représente les différents niveaux de protection d’une image. |

|  |
| --- |
| **userpermissionlevel** |
| Cette énumération représente les différents niveaux de permission d’un utilisateur. |

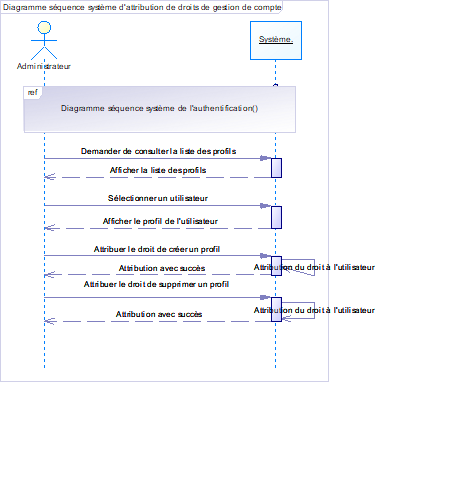
La figure suivante représente le diagramme de classes des formes et éléments qui seront présents dans le canevas. Ce dernier est contenu dans le paquetage Strokes du client lourd.



*Figure 15: Diagramme de classes des formes*

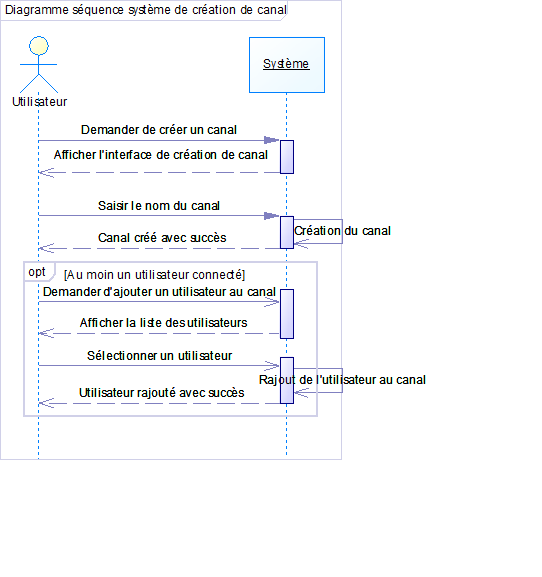
# 5. Vue des processus

## 5.1. Diagramme séquence système pour l’authentification



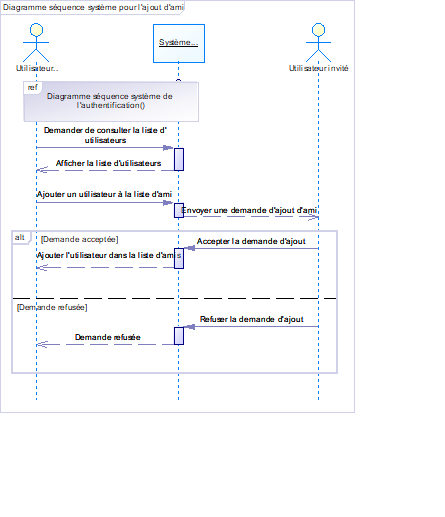
*Figure 16: Diagramme séquence système pour l’authentification*

## 5.2. Diagramme séquence système pour la création de canaux



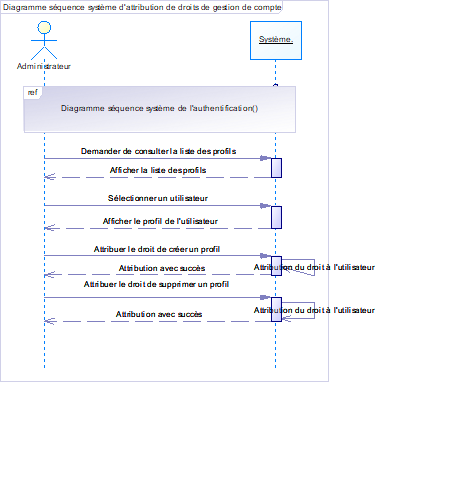
*Figure 17: Diagramme séquence système pour la création de canaux*

## 5.3. Diagramme séquence système pour l’ajout d’amis



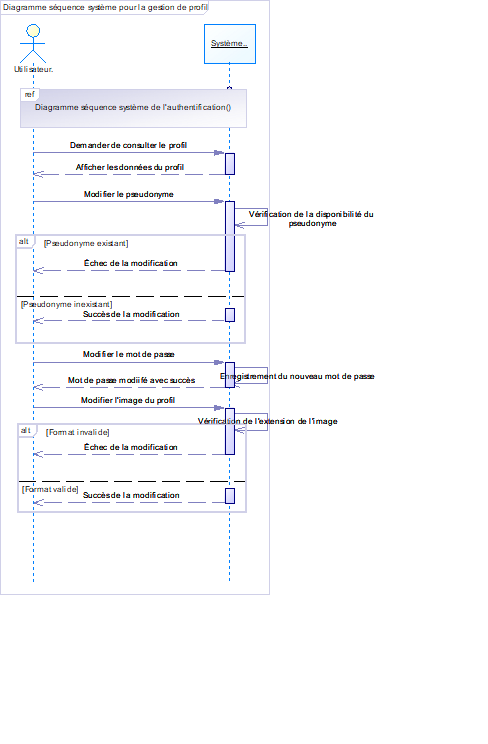
*Figure 18: Diagramme séquence système pour l’ajout d’amis*

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



*Figure 19: Diagramme séquence système pour l’attribution de droits de gestion de compte*

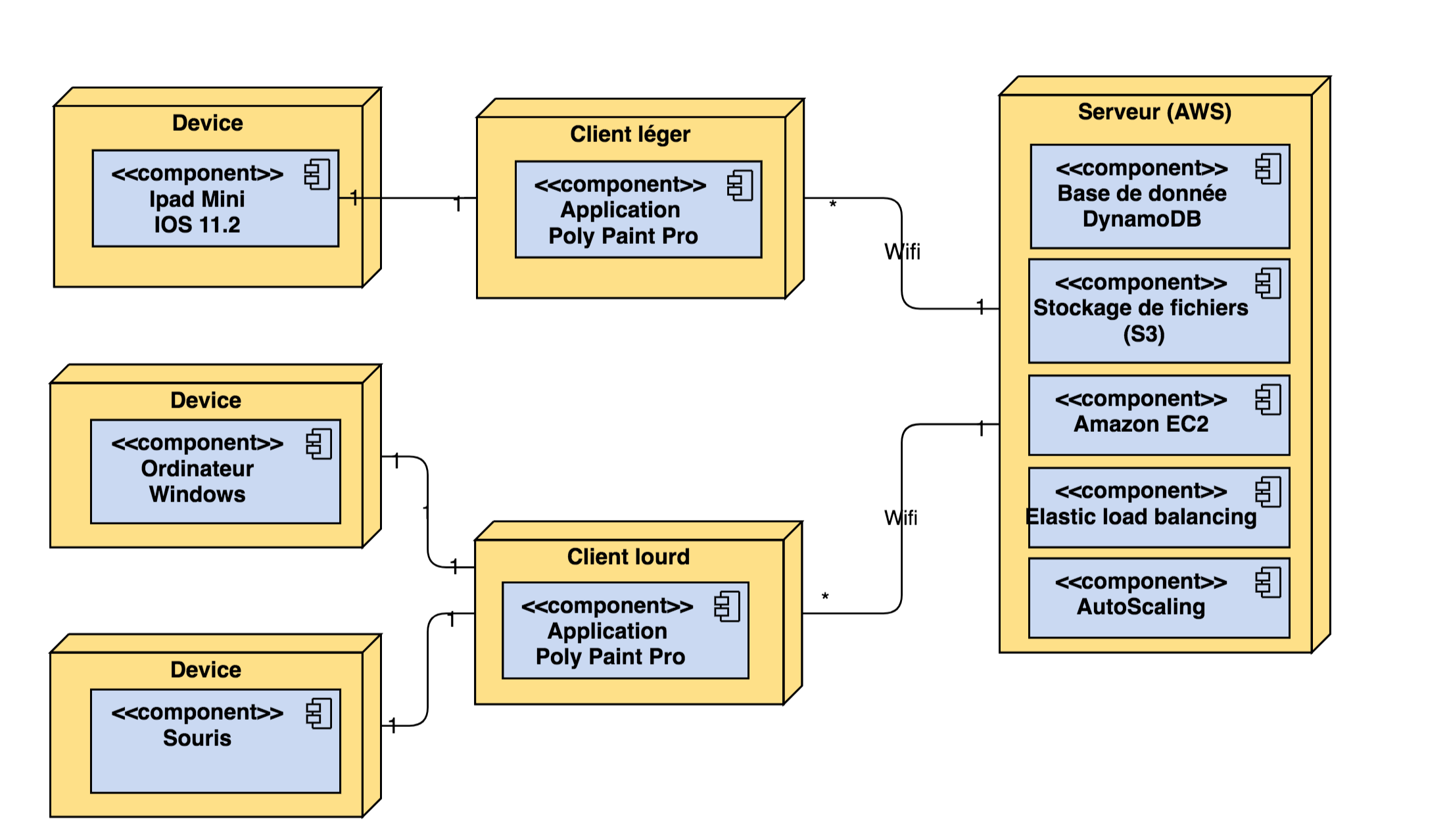
## 5.5. Diagramme séquence système pour la gestion de profils



*Figure 20: 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 21: Schéma de déploiement de notre système [*[*https://online.visual-paradigm.com*](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]](#footnote-0) . La taille de notre projet ne devra pas dépasser une quantité raisonnable de 50Mo. Au moins 50Mo de RAM est aussi recommandé pour 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 requises.[[2]](#footnote-1) De plus, pour rouler l’application, 150 Mo de RAM doivent être disponibles, ainsi que 50 Mo d’espace disque pour l’application. Une quantité de mémoire supplémentaire sera requise pour sauvegarde les images en mode hors-ligne.

## 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 Postgres sur le service RDS 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.

1. Site web d’apple: <https://www.apple.com/ca/fr/ipad-mini-4/specs/> [↑](#footnote-ref-0)
2. Site web d’apple: <https://www.apple.com/ca/fr/ipad-mini-4/specs/> [↑](#footnote-ref-1)