PolyPaint™

Document d'architecture logicielle

Version 1.3

Historique des révisions

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Auteur** |
| 2018-01-29 | 1.0 | Première ébauche et début des sections 1 et 2 | Équipe 11 |
| 2018-01-07 | 1.1 | Diagramme de séquence/cas d’utilisation/déploiement | Équipe 11 |
| 2018-02-09 | 1.2 | Révisions finales pour la première remise | Équipe 11 |
| 2018-04-10 | 1.3 | Correction suite à la première remise | Équipe 11 |

Table des matières

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

[**2. Objectifs et contraintes architecturaux**](#_30j0zll) **4**

[**3. Vue des cas d’utilisation**](#_uywiccxur4t9) **4**

[**4. Vue logique**](#_2et92p0) **8**

[**5. Vue des processus**](#_xzxtrzdqudud) **11**

[**6. Vue de déploiement**](#_3dy6vkm) **13**

[**7. Taille et performance**](#_b3g23o5bhn3v) **14**

Document d'architecture logicielle

# 1. Introduction

Le présent document définit les objectifs et contraintes architecturaux de notre logiciel. Les orientations choisies à ce qui a trait à l’architecture logicielle et plus spécifiquement les vues des cas d’utilisation, de logique, des processus ainsi que de déploiement pour le projet PolyPaint y sont aussi représentées. Finalement, une description des caractéristiques de taille et performance pouvant avoir un impact sur l’architecture logicielle est présentée.

# 2. Objectifs et contraintes architecturaux

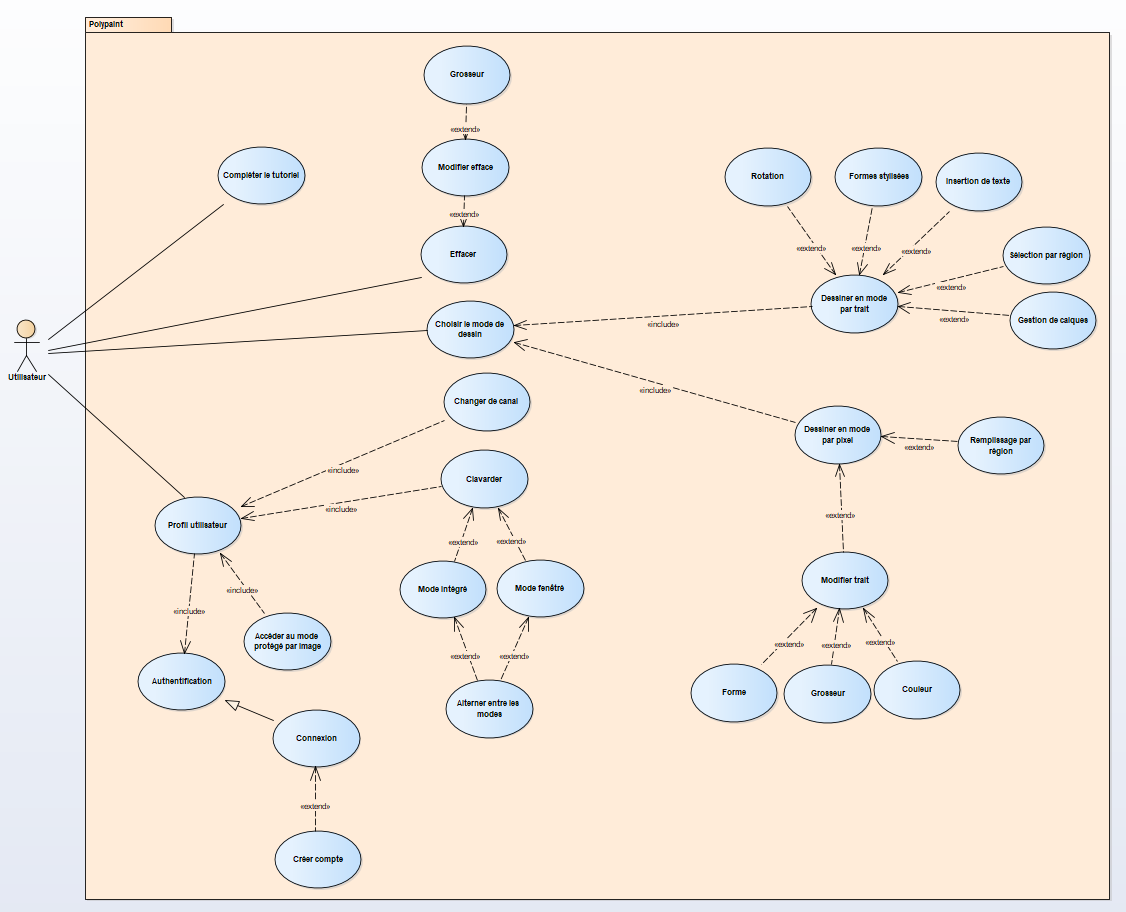
L’architecture doit être conçue de manière à être compatible tout aussi bien sur un ordinateur Windows qu’un iPad mini. Le code doit être réalisé de manière à favoriser la réutilisabilité le plus possible. Le code réutilisé doit être libre de droits. Il doit être en anglais afin de pouvoir être compris par la plupart des personnes dans le futur. L’architecture doit être conçue de manière à assurer une protection des données des utilisateurs d’un accès non autorisé.

Le but de l’architecture est d’utiliser une architecture simple afin de respecter l'échéancier et les coûts mais qui respectent les contraintes précédentes et qui permettra de mener le reste du projet à bien.

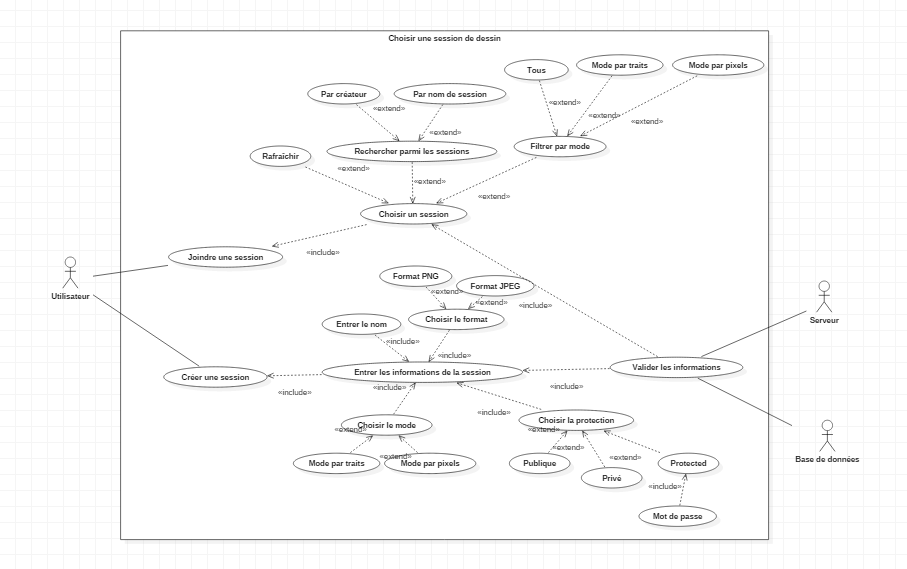
# 3. Vue des cas d’utilisation

Cette section présente les diagrammes de cas d’utilisation jugés les plus pertinents pour le projet.

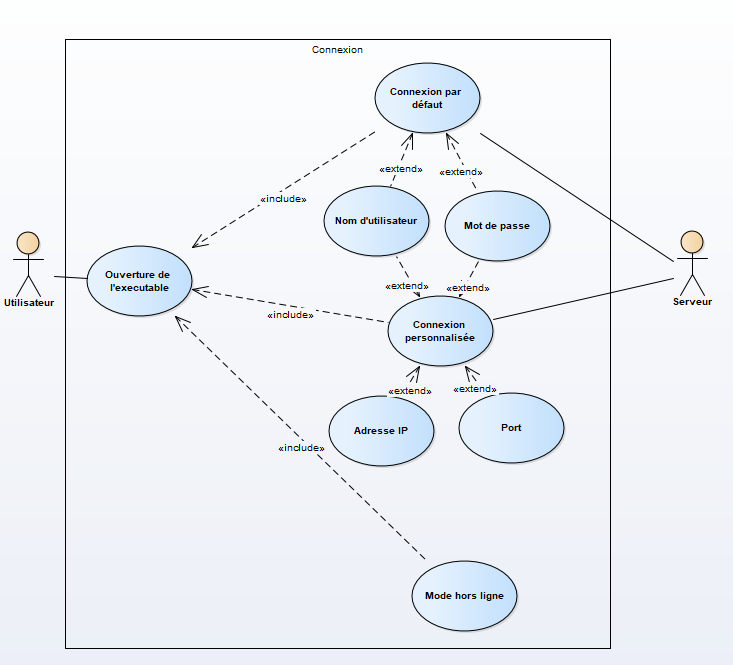
Cas d’utilisation - PolyPaint générale



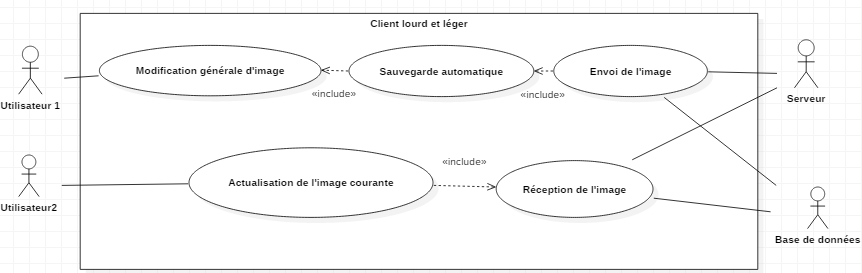
Cas d’utilisation - Choisir une session de dessin



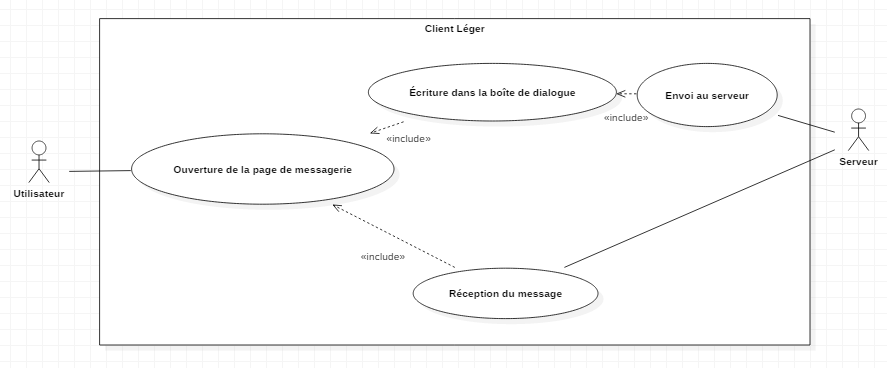
Cas d’utilisation - Connexion



Cas d’utilisation - Collaboration simultanée



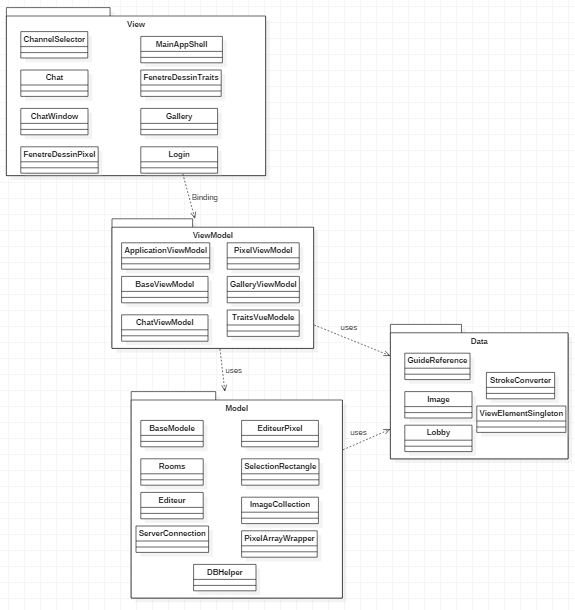
Cas d’utilisation - Messagerie client léger

****

# 4. Vue logique

Cette section illustre les différents paquets qui seront utilisés pour réaliser le projet ainsi que le diagramme de classes du logiciel PolyPaint.

Diagramme de paquetages - Logiciel Polypaint

**

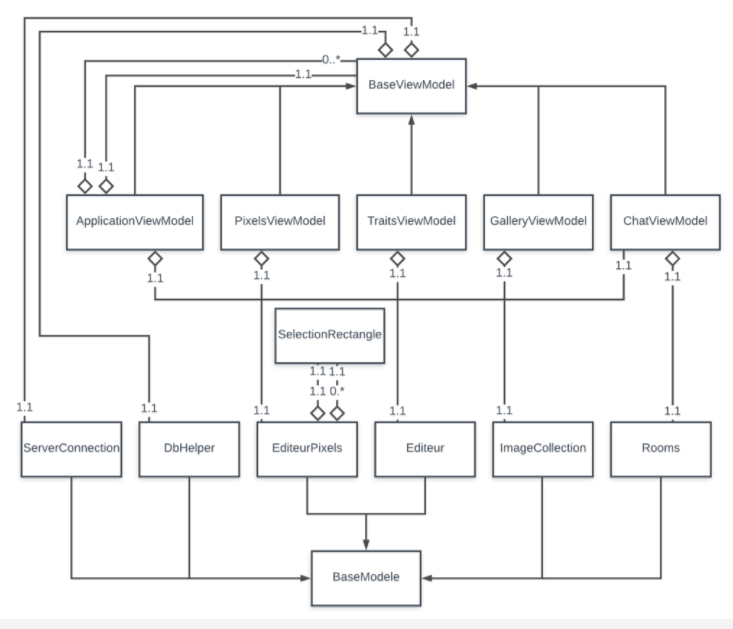
|  |
| --- |
| **View** |
| **ChannelSelector** |
| *Cette classe est responsable du côté présentation des différents canaux pouvant être joint ainsi que les canaux déjà joints* |
| **Chat** |
| *Cette classe est responsable de la présentation de la fenêtre de chat en mode intégré* |
| **ChatWindow** |
| *Cette classe est responsable de la présentation de la fenêtre de chat en mode fenêtré* |
| **FenetreDessinPixels** |
| *Cette classe se charge de la présentation de la fenêtre de dessin en mode par pixels* |
| **FenetreDessinTraits** |
| *Cette classe se charge de la présentation de la fenêtre de dessin en mode par traits* |
| **Gallery** |
| *Cette classe se charge de la présentation de la galerie d’images* |
| **Login** |
| *Cette classe se charge de la présentation de la fenêtre de connexion à PolyPaint* |
| **MainAppShell** |
| *Cette classe est responsable de la présentation de la fenêtre de base de l’application* |

|  |
| --- |
| **ViewModel** |
| **ApplicationViewModel** |
| *Cette classe est responsable de la logique de la vue pour le logiciel Polypaint* |
| **BaseViewModel** |
| *Cette classe abstraite représente la base pour la logique de la vue.* |
| **Chat** |
| *Cette classe est responsable de la logique de la vue pour la partie clavardage* |
| **GalleryViewModel** |
| *Cette classe se charge de la la logique de la vue pour la galerie du logiciel PolyPaint* |
| **PixelViewModel** |
| *Cette classe est responsable de la logique de la vue pour la partie mode par pixels* |
| **TraitsVueModele** |
| *Cette classe est responsable de la logique de la vue pour la partie mode par traits* |

|  |
| --- |
| **Model** |
| **BaseModele** |
| *Cette classe abstraite représente un modèle de base.* |
| **DBHelper** |
| *Cette classe est responsable du modèle de des données qui communiquent avec la base de données* |
| **Editeur** |
| *Cette classe est responsable du modèle des données pour le mode par traits* |
| **EditeurPixel** |
| *Cette classe est responsable du modèle des données pour le mode par pixels* |
| **ImageCollection** |
| *Cette classe est responsable du modèle des données pour les images dans la galerie d’images* |
| **PixelArrayWrapper** |
| *Cette classe est responsable du modèle des données pour convertir les pixels en tableaux d’octets qui seront envoyés dans la base de données* |
| **Rooms** |
| *Cette classe est responsable du modèle des données pour les différents lobbies de dessin* |
| **SelectionRectangle** |
| *Cette classe est responsable du modèle de des données pour les opérations de sélection sur le mode par pixels* |
| **ServerConnection** |
| *Cette classe est responsable du modèle de des données pour la connexion sur le serveur* |

|  |
| --- |
| **Data** |
| **GuideReference** |
| *Références GUID.* |
| **Image** |
| *Objet Image qui contient toutes les propriétés d’une image.* |
| **Lobby** |
| *Object Lobby qui contient toutes les propriétés d’un lobby (session).* |
| **StrokeConverter** |
| *Permet de convertir les Strokes-byte[] pour l’envoit et la réception du léger.* |
| **ViewElementSingleton** |
| *Permet de gérer les éléments singleton dans l’application.* |

Diagramme de classes - Logiciel Polypaint



# 5. Vue des processus

Tous les différents outils de dessin possible sont regroupés dans les messages DRAW\_OFFLINE() et DRAW\_ONLINE() afin d’alléger la lecture du diagramme ci-dessous.

Diagramme de séquence - Dessiner

|  |
| --- |
|  |

Diagramme de séquence - Connexion

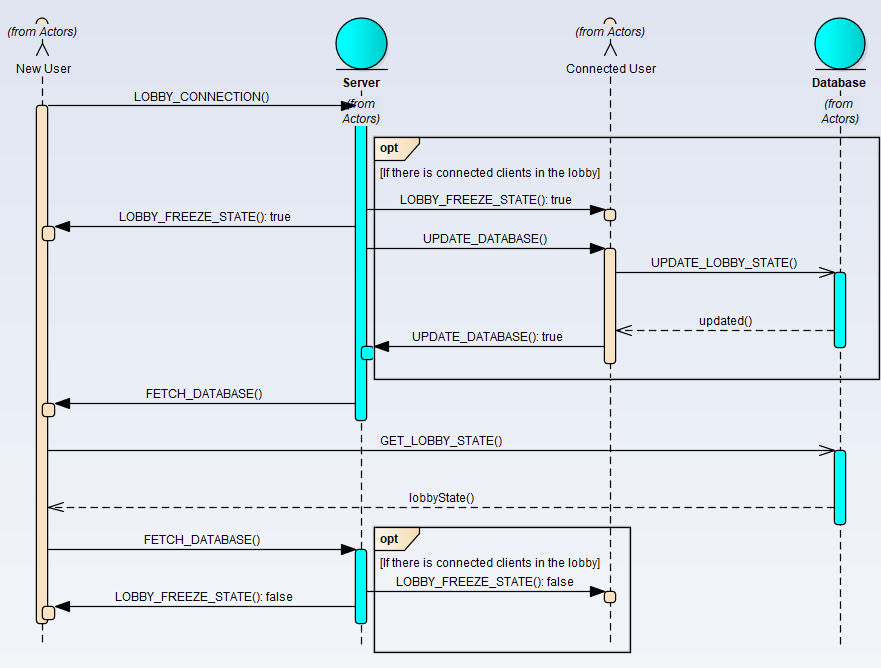
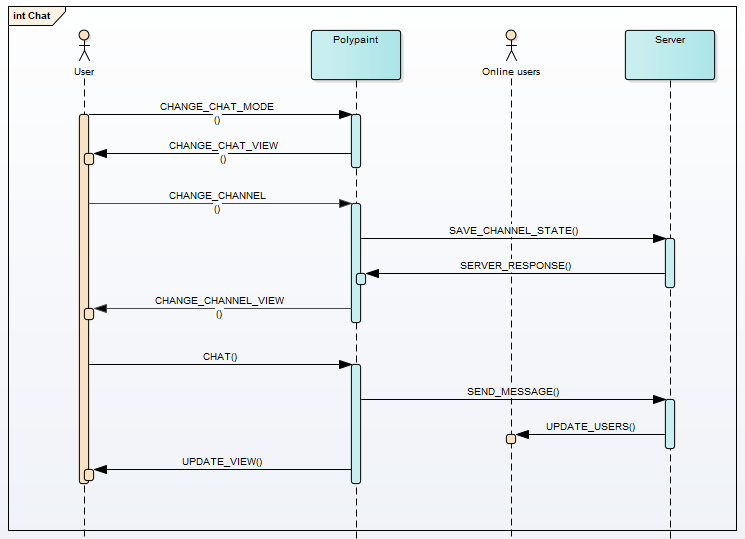
**

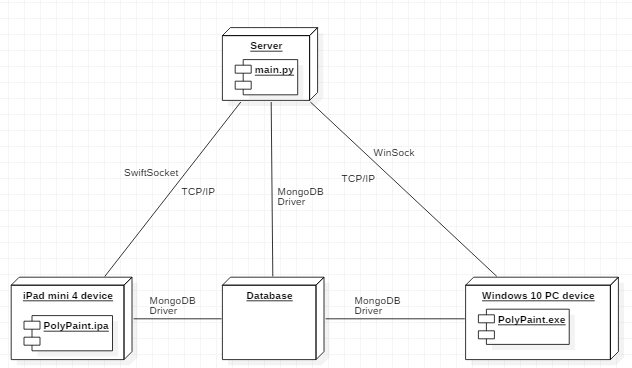
Diagramme de séquence - Clavardage

**

# 6. Vue de déploiement

Le diagramme de déploiement illustre les connexions entre les différents équipements utilisés par le logiciel PolyPaint.

Diagramme de déploiement - Logiciel PolyPaint

**

# 7. Taille et performance

L’architecture logicielle a été développée de manière à respecter les contraintes de taille et performance décrites dans le SRS tant au niveau de la mémoire utilisée qu’au niveau de la latence entre les utilisateurs. Les composantes du logiciel devront être développées de façon à permettre une expérience de dessin avec une latence la plus minimale possible et devront aussi être conçues de manière à utiliser le moins de mémoire et espace sur les périphériques. Ce dernier point est encore plus important pour le client léger ayant des contraintes de mémoire plus strictes qu’un poste avec Windows 10. L’architecture du serveur a aussi été développée de manière à supporter au moins cinq utilisateurs avec une latence minimale lors de la communication dans un canal de chat ou d’une modification d’un canevas dans un lobby de dessin.