Équipe 11

Projet Intégrateur de Troisième Année
Document d'architecture logicielle

Version 2.0

Historique des révisions

Date	Version	Description	Auteur
2019-01-25	1.0	1 diagramme ajouté (authentification)	Mylène P.
2019-01-26	1.1	Diagramme général + diagramme général édition dessin	Tom G.
2019-02-05	1.2	Correction + Section 2	Mylène P.
2019-02-06	1.3	Ajout des diagrammes de séquence	Olivier
2019-02-07	1.4	Section 7	Maxim Cousineau
2019-02-08	2.0	Revue finale	Mylène, Tom, Olivier, Maxim

Table des matières

Document d'architecture logicielle

1. Introduction

Le document d'architecture logicielle présente la vision d'ensemble de la structure architecturale du projet PolyPaint et se base sur les exigences décrites dans le document SRS. Les diagrammes contenus dans cet artefact respecteront UML.

Tout d'abord, nous décrirons les objectifs et les contraintes architecturales. Ensuite, nous présentons une vue d'ensemble des cas d'utilisation grâce à des diagrammes de cas d'utilisation.

Pour ce qui est de la vue logique, nous allons utiliser des diagrammes de paquetage. Nous mentionnerons également les responsabilités associées à chaque paquetage.

En ce qui concerne la vue des processus, nous illustrons les différentes interactions dans le système à l'aide de diagrammes de séquence.

Quant à la vue de déploiement, elle décrira les configurations du matériel physique.

Finalement, la section Taille et Performance décrit les caractéristiques de taille et de performance pouvant influencer l'architecture et le design du logiciel.

2. Objectifs et contraintes architecturaux

2.1. Outils de développement et langage de développement

Pour le client lourd, le langage utilisé est C#/WPF.

Pour le client léger, le langage utilisé est Java.

Le serveur sera développé en Typescript sur le cadre logiciel de Node.js. Le serveur est hébergé sur Azure.

Le codage sera fait en anglais afin d'être consistant avec le reste du code.

2.2. Sécurité et confidentialité

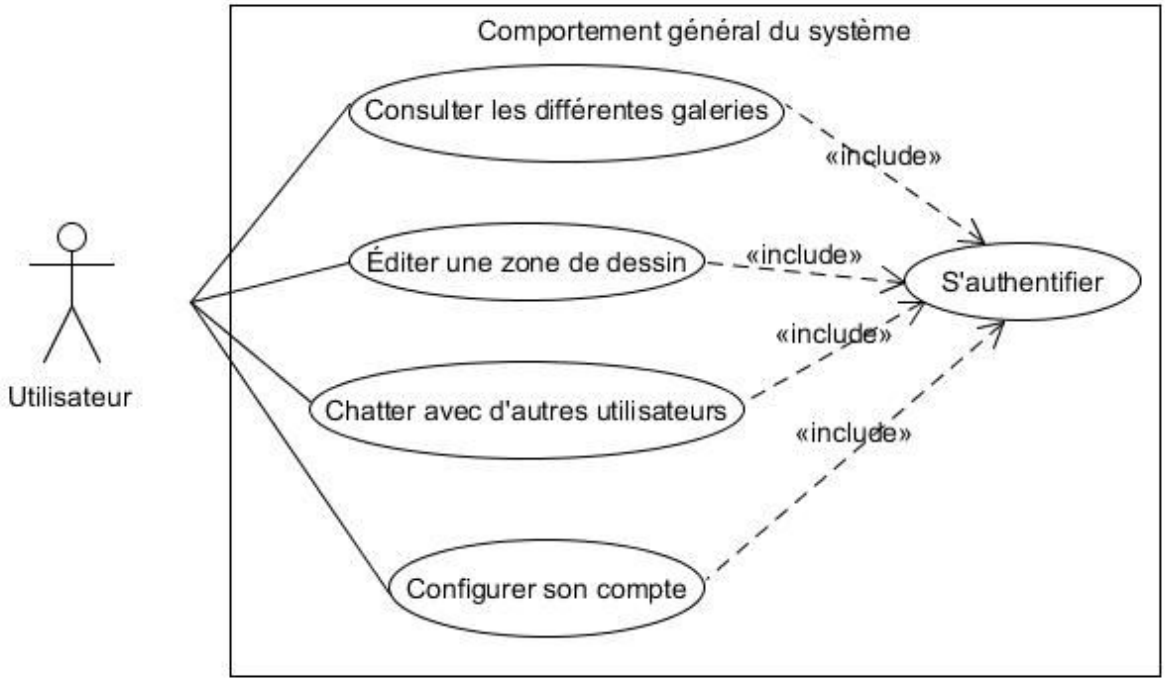
Le serveur doit pouvoir gérer un système de gestion d'utilisateur, en plus de pouvoir permettre à un utilisateur de récupérer un nouveau mot de passe au besoin. Il faut donc établir des protocoles pour sécuriser les informations de connexion entre le serveur, la base de données et les clients.

2.3. Échéancier et coûts

Pour le produit final, on prévoit 8 semaines de travail, où la quatrième semaine sera plus intensive et où la dernière est réservée aux tests. Dans les semaines normales, il y aura 6 développeurs qui travaillent environ 8 heures-personne par semaines. Il faut donc tenir compte des limites de temps disponible pour créer le produit lors de la conception afin de pouvoir le compléter dans les temps.

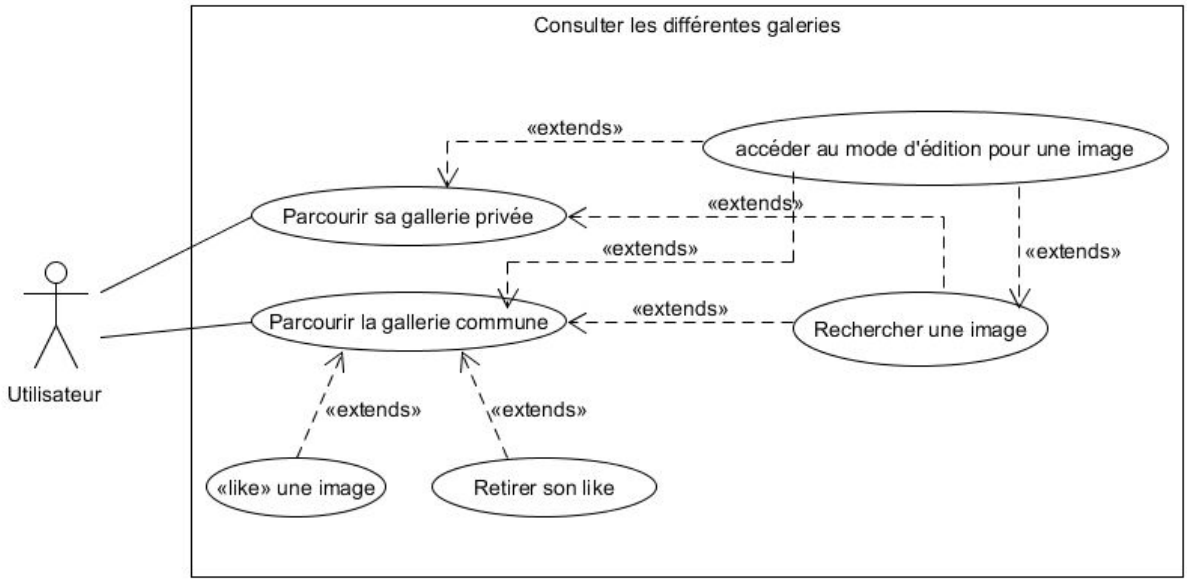
3. Vue des cas d'utilisation

3.1. Diagramme général

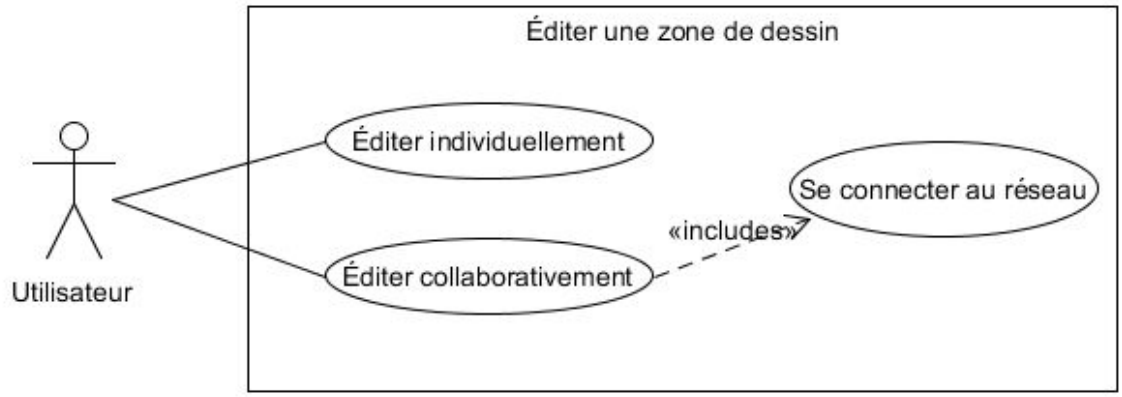


3.2. Diagrammes des clients

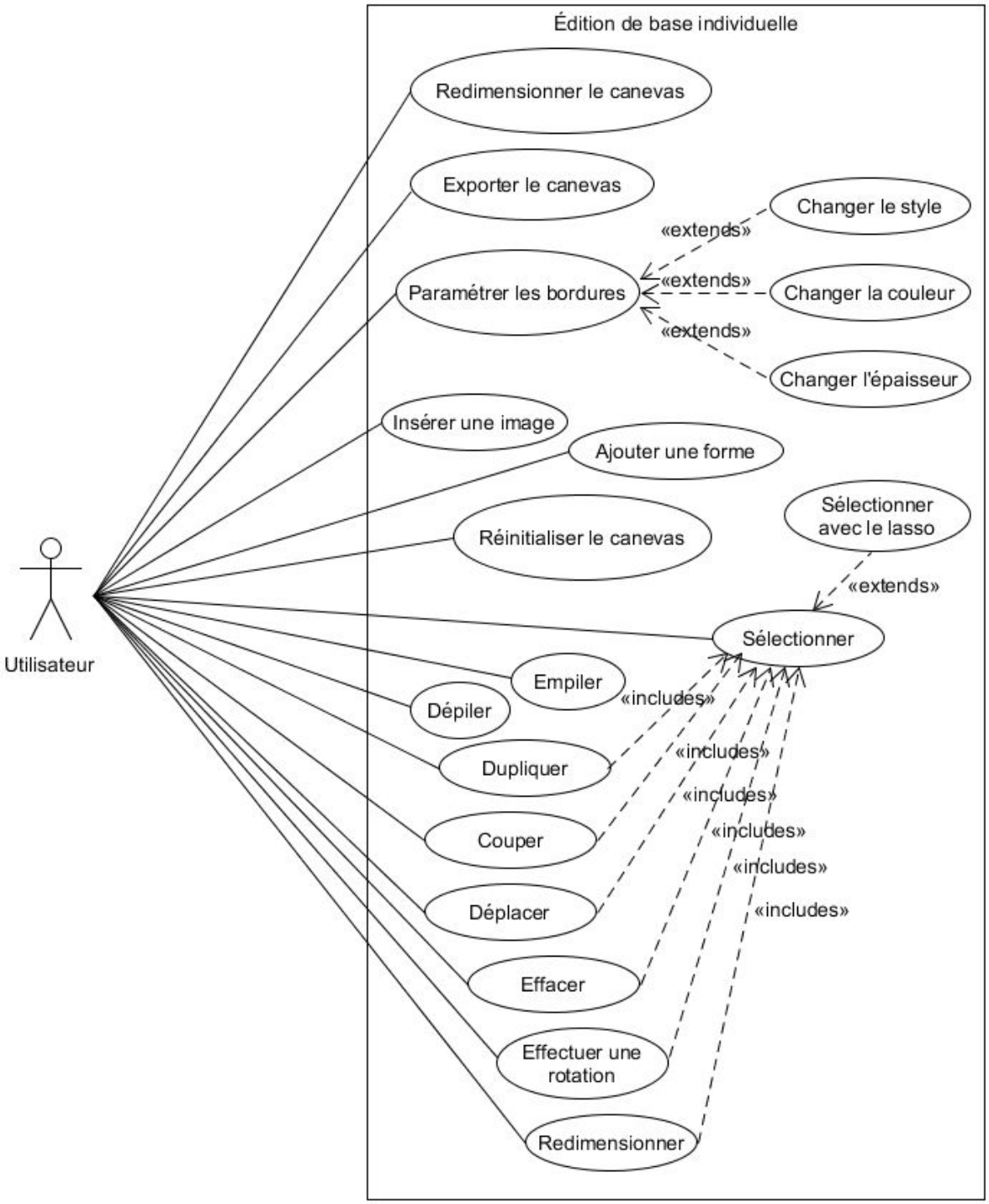
3.2.1. Consulter les différentes galeries



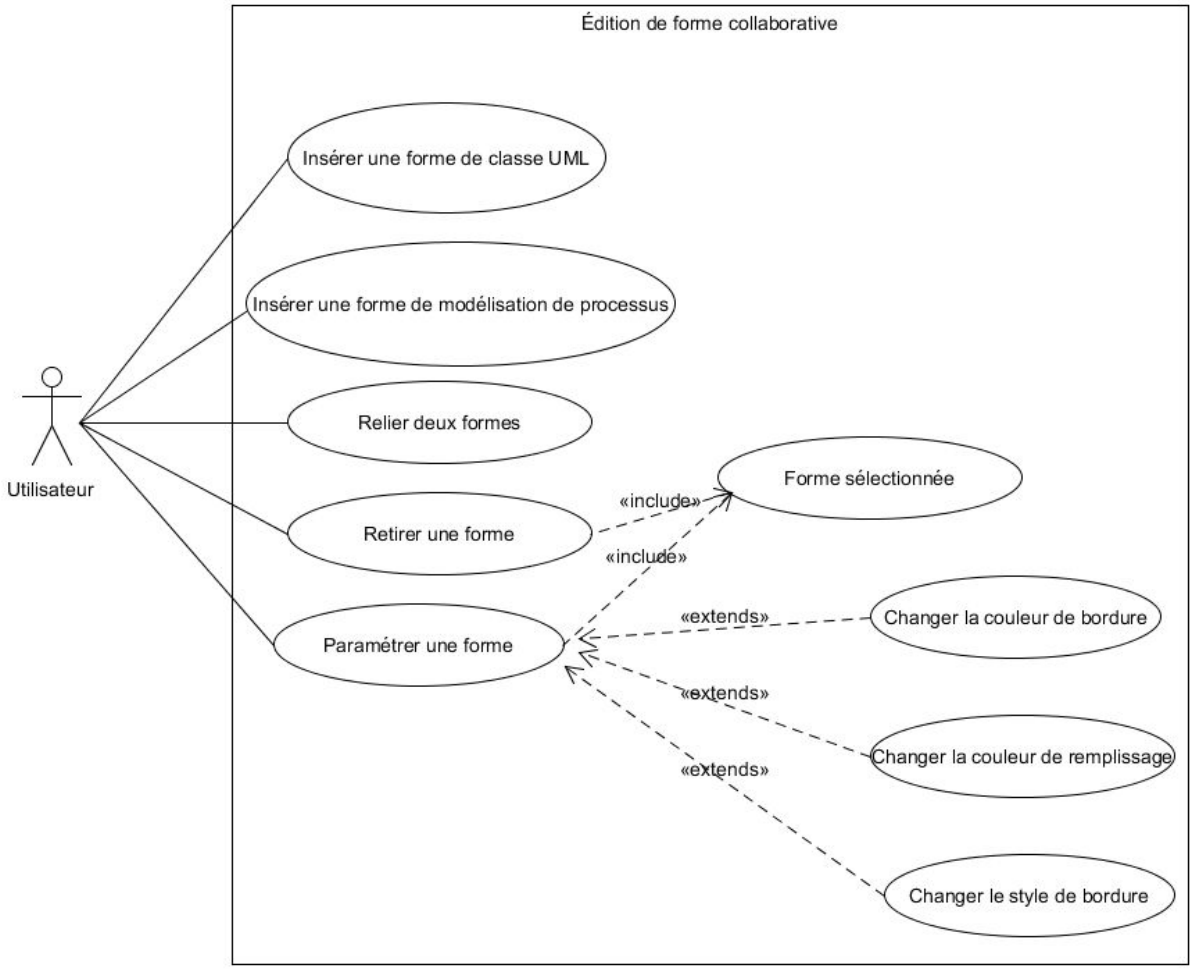
3.2.2. Éditer une zone de dessin



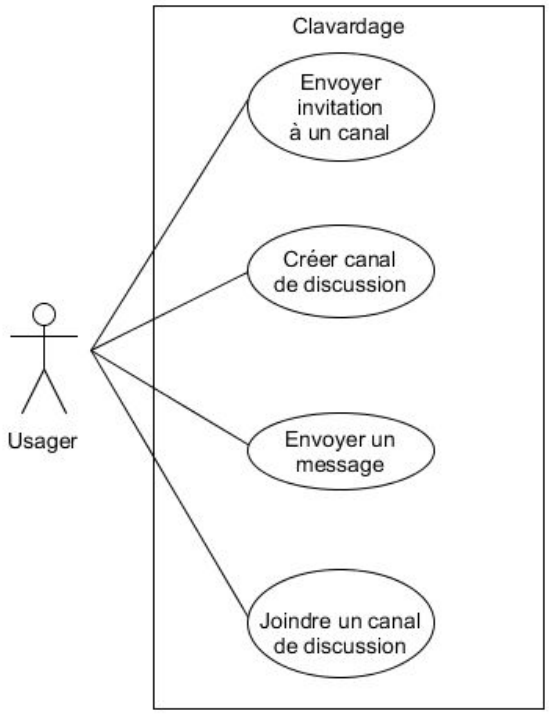
3.2.2.1. Édition de base



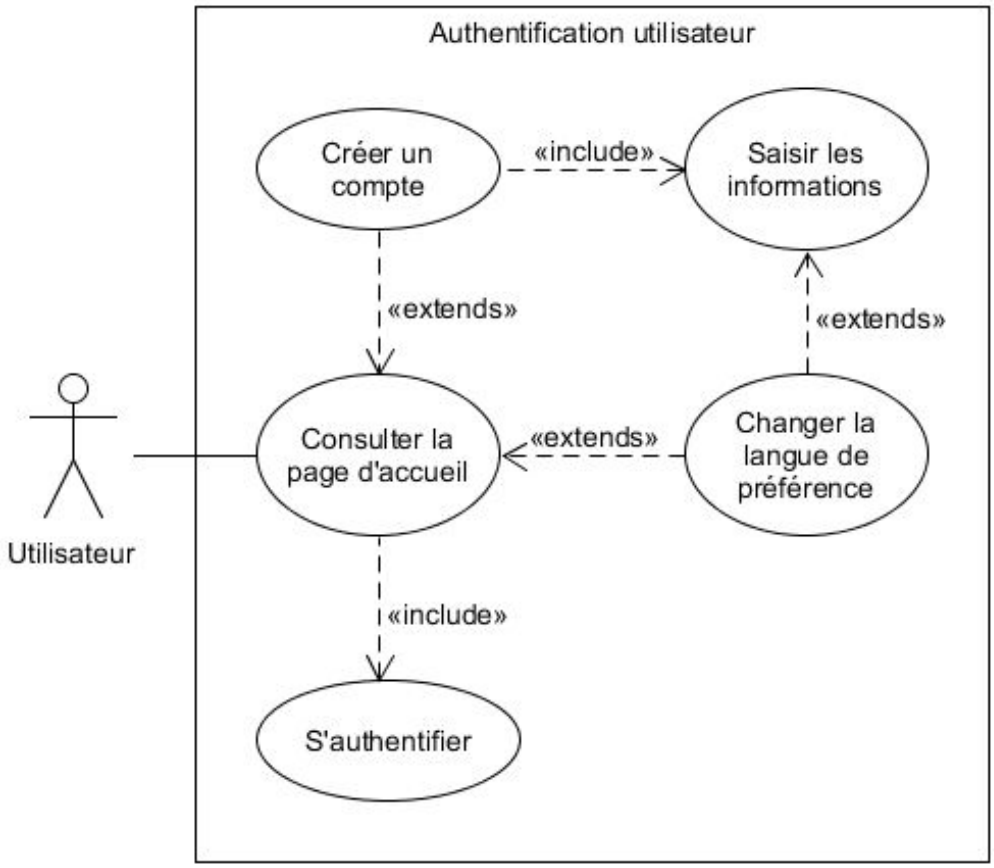
3.2.2.2. Édition de formes



3.2.3. Clavarder avec d’autres utilisateurs



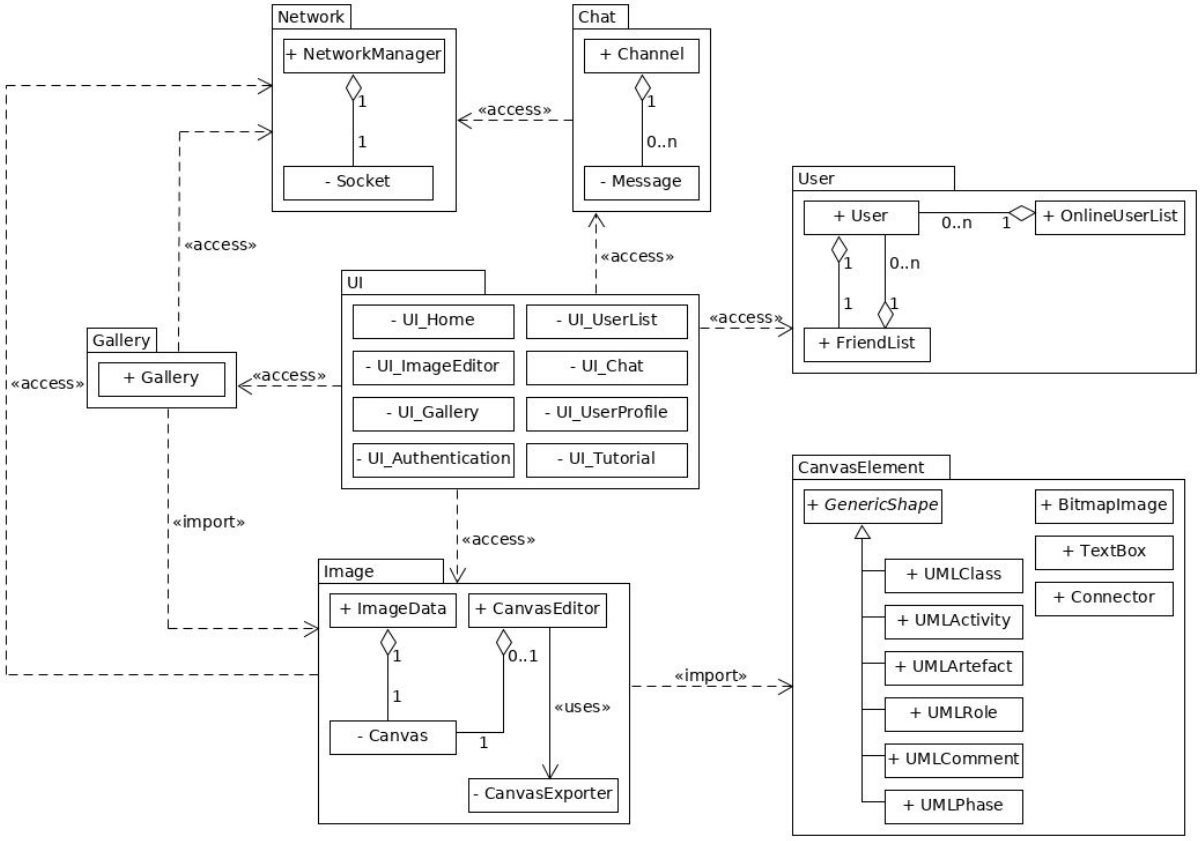
3.2.4. Diagramme de cas d'utilisation pour s'authentifier



4. Vue logique

Cette section présente un diagramme de paquetages pour le client et un diagramme de paquetages pour le serveur, et inclut une description pour chaque paquetage.

4.1. Diagramme de paquetage pour le client



UI
Ce paquetage gère l'affichage de l'application et capte les entrées de l'utilisateur. Chaque classe représente une page de l'application. Les classes de ce paquetage ne devront pas contenir de logique d'application.

Network
Gère les communications avec le serveur par l'intermède de la technologie socket et de requêtes HTTP. Les fonctionnalités qui nécessitent une communication au serveur feront usage de la classe NetworkManager pour assurer cette communication.

Image
Gère les données d'images, incluant l'édition d'image et l'exportation du canvas.

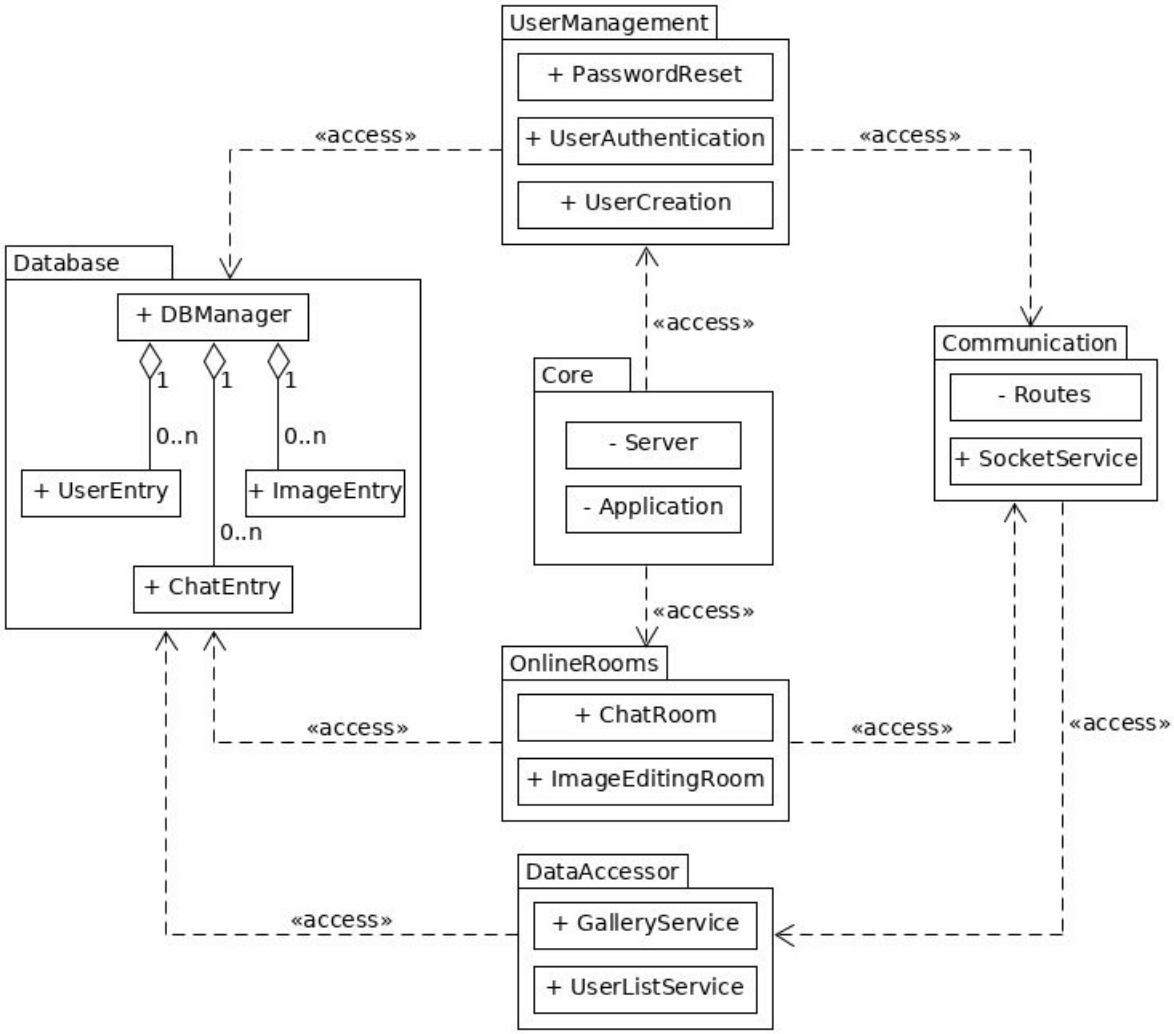
Gallery
Gère la galerie publique et les galeries privées, ainsi que la recherche et le tri de celles-ci.

CanvasElement
Contient les différents types d'éléments d'image pouvant être affichées sur le canvas, soit les formes, les connecteurs et les éléments de texte flottant.

User
Gère les données de l'utilisateur ainsi que les listes d'utilisateurs.

Chat
Gère les fonctionnalités de clavardage.

4.2. Diagramme de paquetage pour le serveur



Core
Assure le roulement de l'application.

Database
Gère les interactions du serveur avec la base de données.

OnlineRooms
Gère les communications directes entre les usagers lors du clavardage et de l'édition en mode collaboratif.

UserManagement
Gère les comptes utilisateurs.

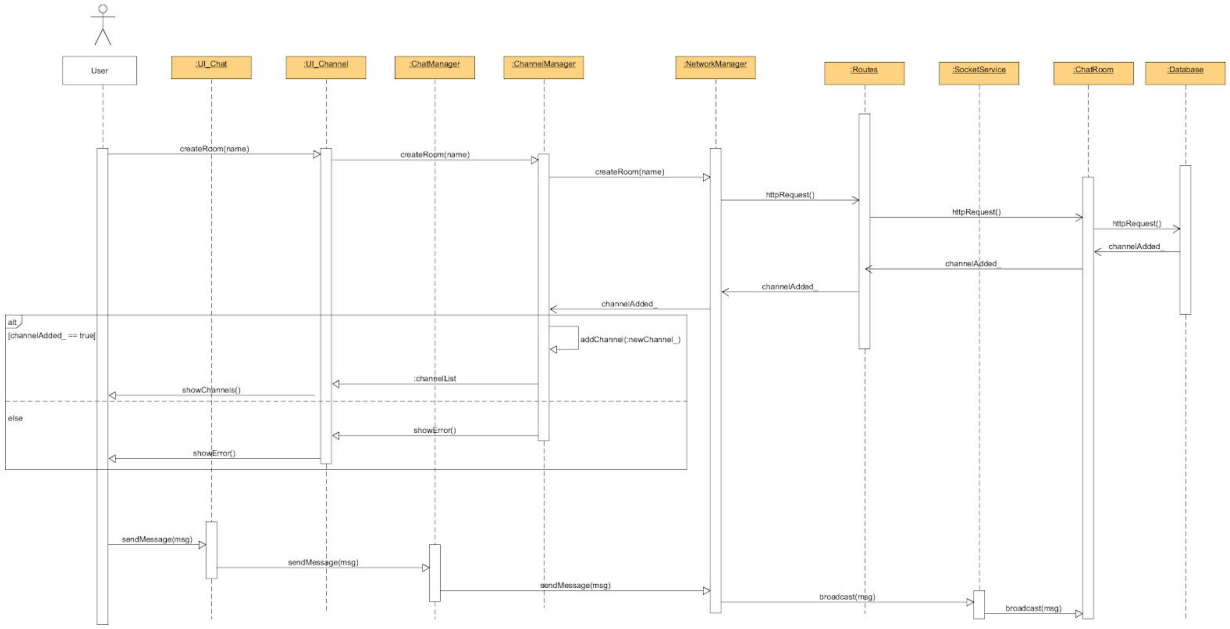
Communication
Assure la communication avec les clients en servant d'interface entre le serveur et l'internet.

DataAccessor
Récupère des données de la base de données.

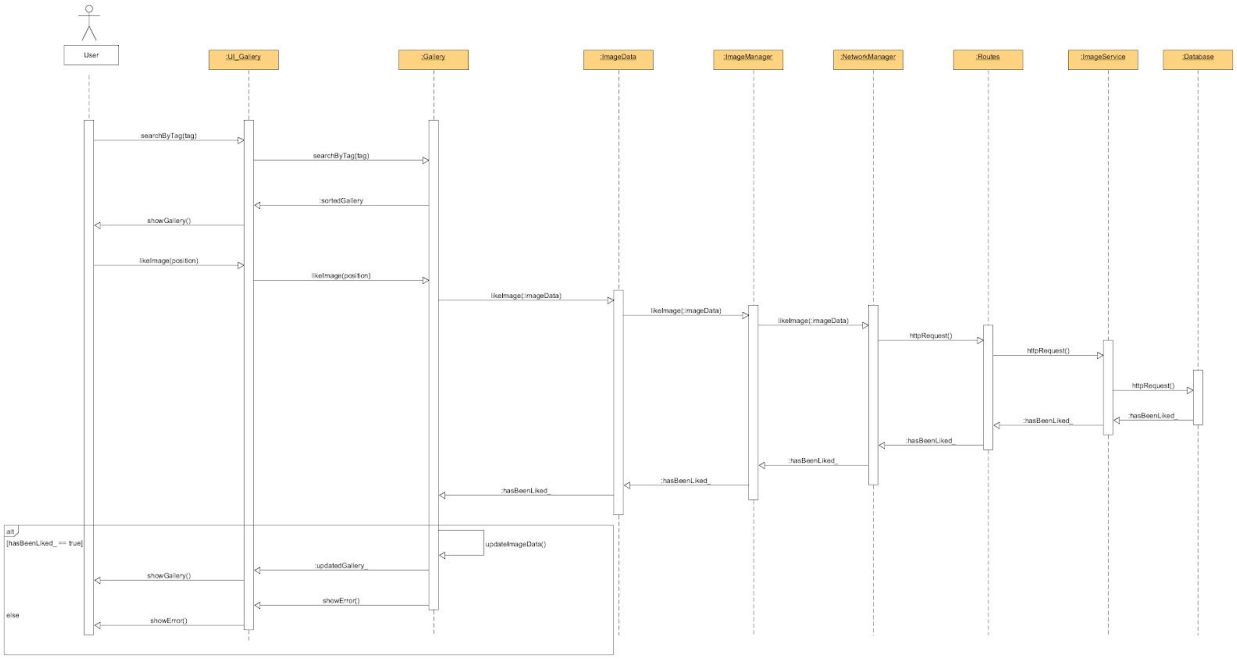
5. Vue des processus

Trois des interactions du système dans le logiciel sont décrites sous la forme de diagrammes de séquence qui sont présents ci-dessous:

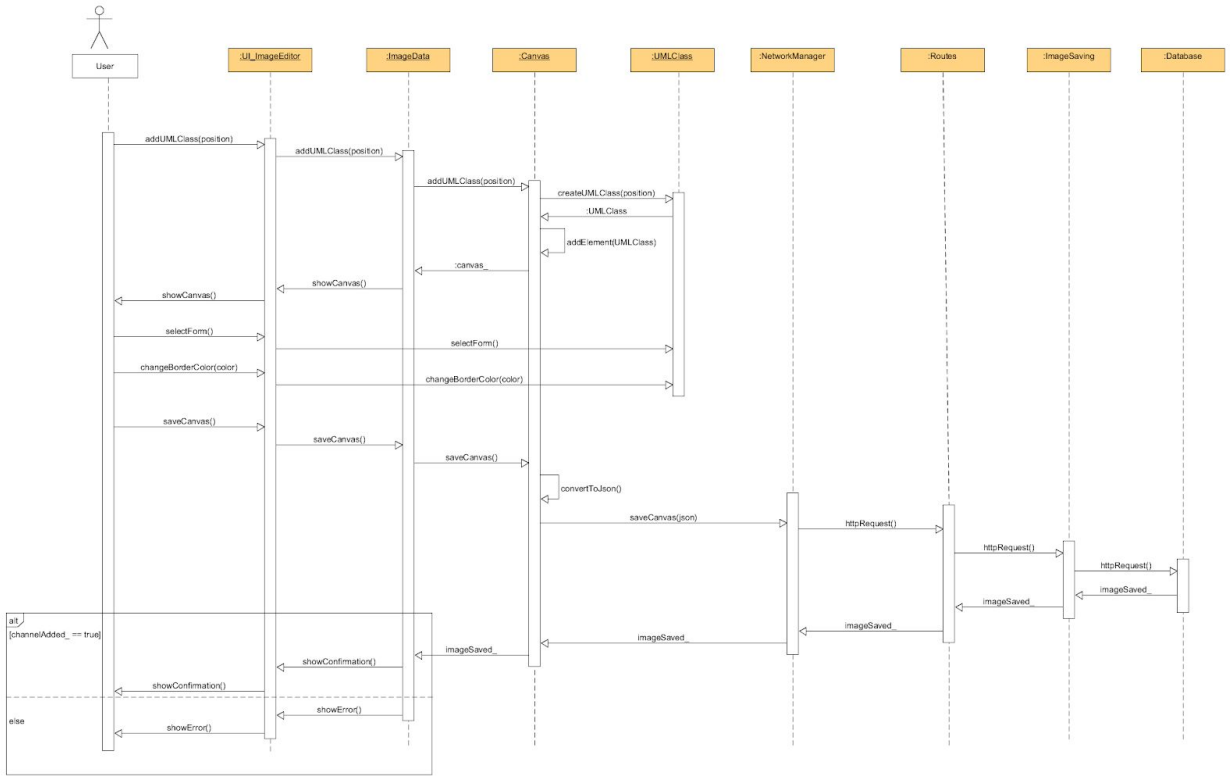
5.1 Diagramme de séquence de la création d'un canal de communication et envoi d'un message



5.2 Diagramme de séquence sur "Effectuer une recherche selon un tag pour ensuite mettre un "like" sur une image"

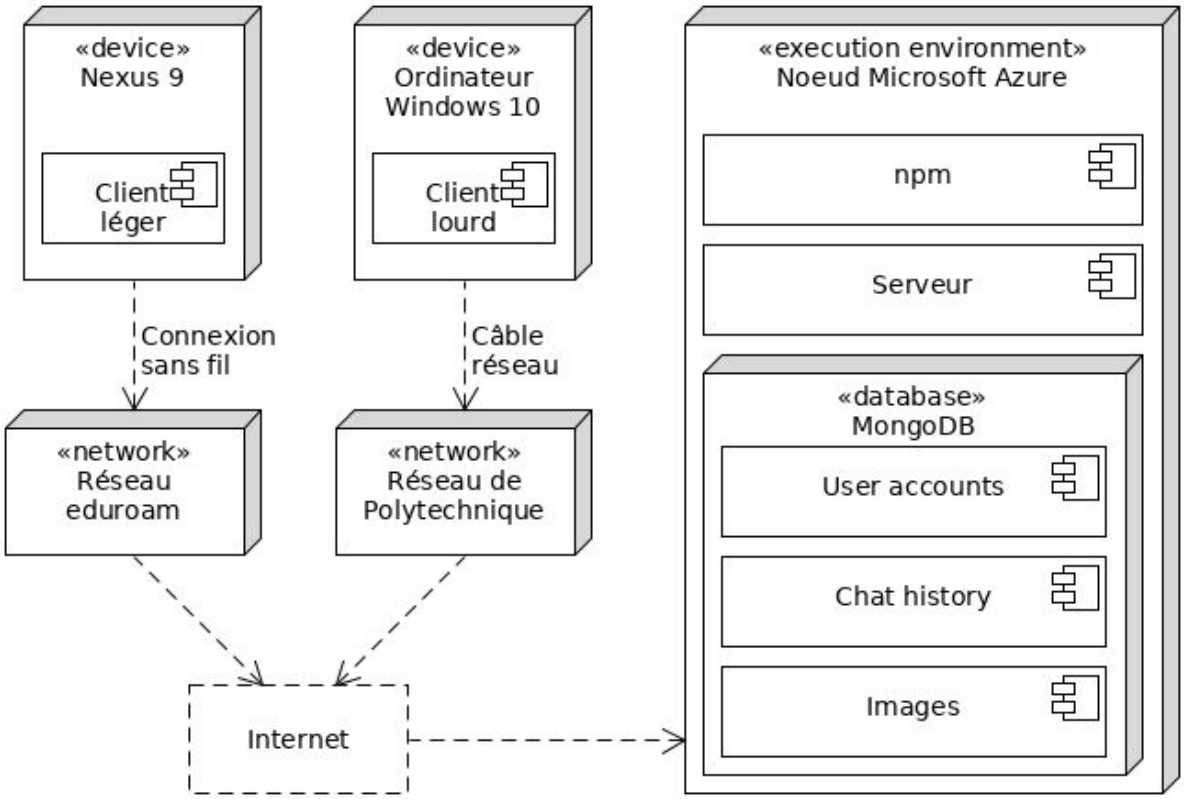


5.3 Diagramme de séquence sur “Ajouter une forme UML, la sélectionner, modifier la couleur de bordure, puis sauvegarder le canevas”



6. Vue de déploiement

Diagramme de déploiement



7. Taille et performance

Les limites de mémoire vive utilisée est de 100 Mo pour le client léger et 200 Mo pour le client lourd. Il faut donc prendre soin que le nombre de classes instanciées à un moment donné soit suffisamment bas pour ne pas dépasser cette limite.
Les sessions collaboratives doivent se faire avec une latence d'au plus 25 ms. La communication entre les clients et le serveur devra être optimisée de manière à ne pas dépasser cette limite sur une connexion normale.
Le chargement et la sauvegarde d'images doivent se faire en moins de 5 secondes, ce qui nécessite une conversion efficace entre les objets côté client, utilisés pour représenter les éléments d'image, et la représentation côté serveur, qui sera sauvegardée dans la base de données.