Équipe 102

ColorImage
Document d'architecture logicielle

Version 2.0

Historique des révisions

Date	Version	Description	Auteur
2021-09-30	1.0	Rédaction initiale	Équipe 102
2021-12-05	2.0	Révision finale	Équipe 102

Table des matières

Document d'architecture logicielle

1. Introduction

Dans ce document, on présente l'architecture logicielle de ColorImage. Tout d'abord, on commencera par présenter les objectifs et contraintes architecturaux qui possèdent un impact architectural sur notre logiciel. On met l'accent sur le serveur, le client lourd et le client léger. Par la suite, la section 3 sera composée de diagrammes de cas d'utilisation pour montrer les différents types de cas que notre application devra prendre en compte lors de sa conception. Ensuite, la section 4 portera sur la vue logique de ColorImage à l'aide de diagrammes de paquetages de haut niveau et des diagrammes de classes. Puis, la section 5 sera sur la section de la vue des processus où on utilisera des diagrammes de séquences pour illustrer les différentes interactions entre les objets. À la section 6, elle sera décrite à l'aide de diagrammes de déploiement qui montrent la configuration des différentes composantes matérielles et virtuelles. Finalement, à la section 7, on présentera une description de la taille et de la performance de notre architecture logicielle.

2. Objectifs et contraintes architecturaux

L'objectif du projet est de développer une application de dessin collaboratif à partir du logiciel *PolyDessin*. C'est-à-dire, il nous faudra concevoir une application pour permettre à plusieurs utilisateurs d'éditer en temps réel des dessins. Le logiciel doit également être fonctionnel sur un ordinateur (client lourd) et sur une tablette Android (client léger).

Notre architecture est contrainte par la structure du projet original. Effectivement, puisque le projet consiste à retravailler le logiciel *Polydessin*, il nous faudra utiliser ce logiciel, possédant déjà sa propre architecture, et lui apporter les changements nécessaires pour répondre aux exigences. Alors, notre architecture est contrainte par la structure du projet original.

Nous sommes également contraints par deux échéanciers. Le premier étant la remise d'un prototype pour le 1er octobre. La deuxième échéance est le 6 décembre pour la remise du produit final. Il faudra alors gérer notre temps adéquatement pour bien construire l'architecture de notre logiciel.

Le serveur occupe une place importante dans la structure de notre système. En effet, le serveur s'occupera de gérer les données des utilisateurs pour les clients lourd et léger ainsi que de permettre la communication entre les deux types de clients. D'ailleurs, étant donné que le logiciel doit pouvoir accueillir plusieurs utilisateurs en même temps, il nous faudra assurer que la communication soit fluide et que la vue soit cohérente pour tous les utilisateurs. De plus, pour avoir une couche de sécurité, nous avons ajouté un système d'authentification pour utiliser le logiciel. Le serveur doit posséder une bonne architecture afin de répondre à toutes ses exigences de manière efficace.

Finalement, la dernière contrainte pour ce projet est que le client lourd et le client léger ne soient pas codés avec le même langage de développement. Donc, pour le client léger, il nous faudra reproduire l'application qui a été implémentée en Typescript à l'aide du cadriciel Angular en Kotlin afin que le logiciel puisse fonctionner sur un appareil Android. Ainsi, l'architecture de notre projet est fortement influencée par ces technologies.

Nous avons comme objectif architectural de réaliser une architecture robuste, structurée et modulable afin de favoriser l'intégration facile des fonctionnalités.

En plus des différents points mentionnés ci-haut, le serveur sera en grande partie géré par des services singleton qui gèrent la logique de transformation et de gestion de données backend du serveur. Ces services garderont en mémoire l'état actuel des données pour s'assurer que les temps de latence des requêtes achevés au serveur sont réduits. La partie interface sera habillée d'un décorateur REST API pour recevoir des requêtes du client lourd et léger et répondre à ceux-ci selon des codes de réponses définies et structurées. De plus, on utilise aussi le patron commande pour gérer la fonctionnalité défaire-refaire.

Quant au client léger et au client lourd, nous allons utiliser un modèle MVVM (*Model-View-ViewModel*) dans lequel on sépare la couche de vue et la couche de logique. Le *ViewModel* contiendra la grande majorité de la logique de l'application. La couche de vue rassemble tous les éléments visuels. Cette dernière se liera avec des données contenues dans le *ViewModel* afin de pouvoir les afficher. Quant au *Model*, c'est l'ensemble des classes qui permet de structurer les données.

3. Vue des cas d'utilisation

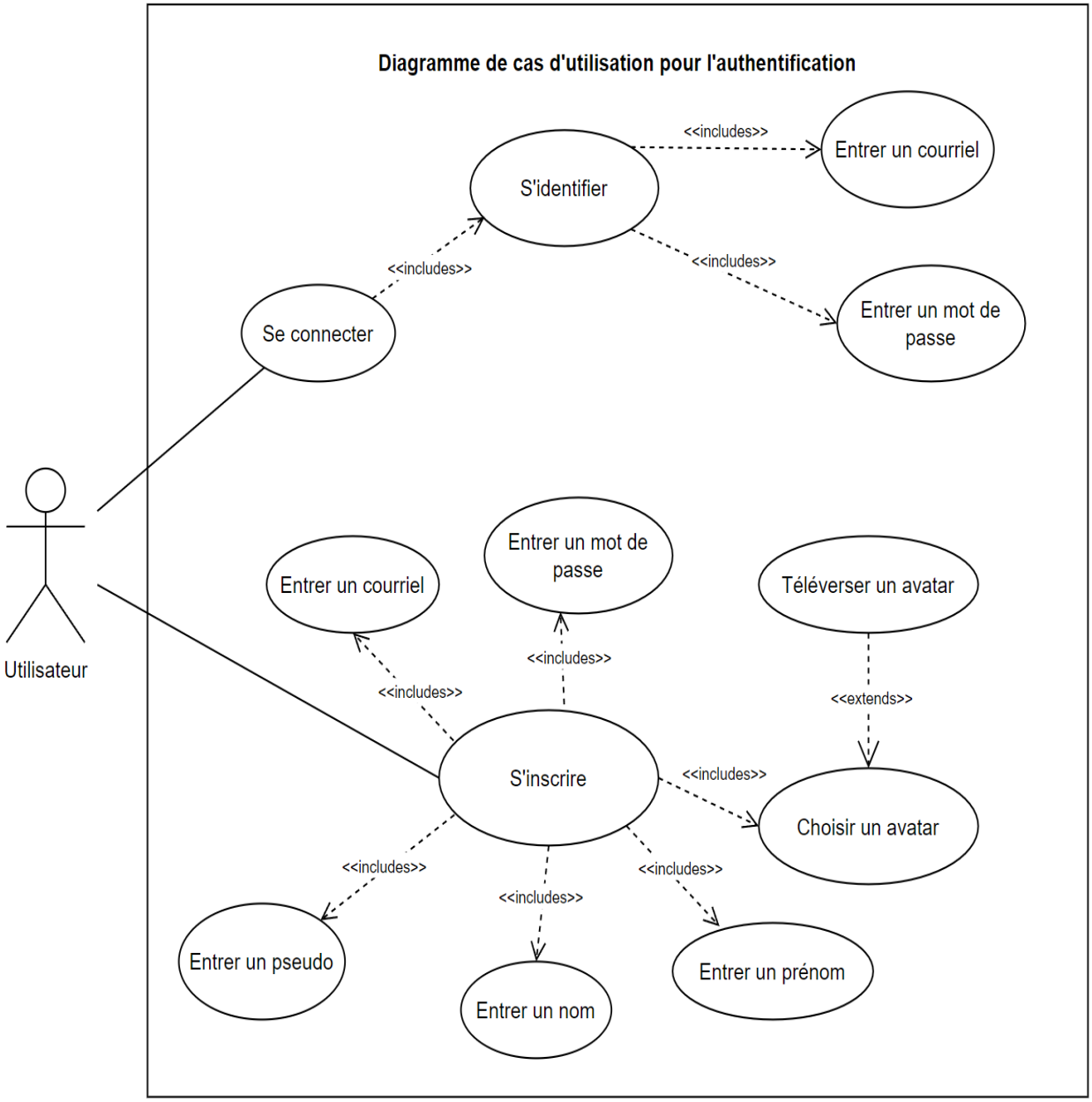

Figure 1: Diagramme de cas d'utilisation pour l'authentification

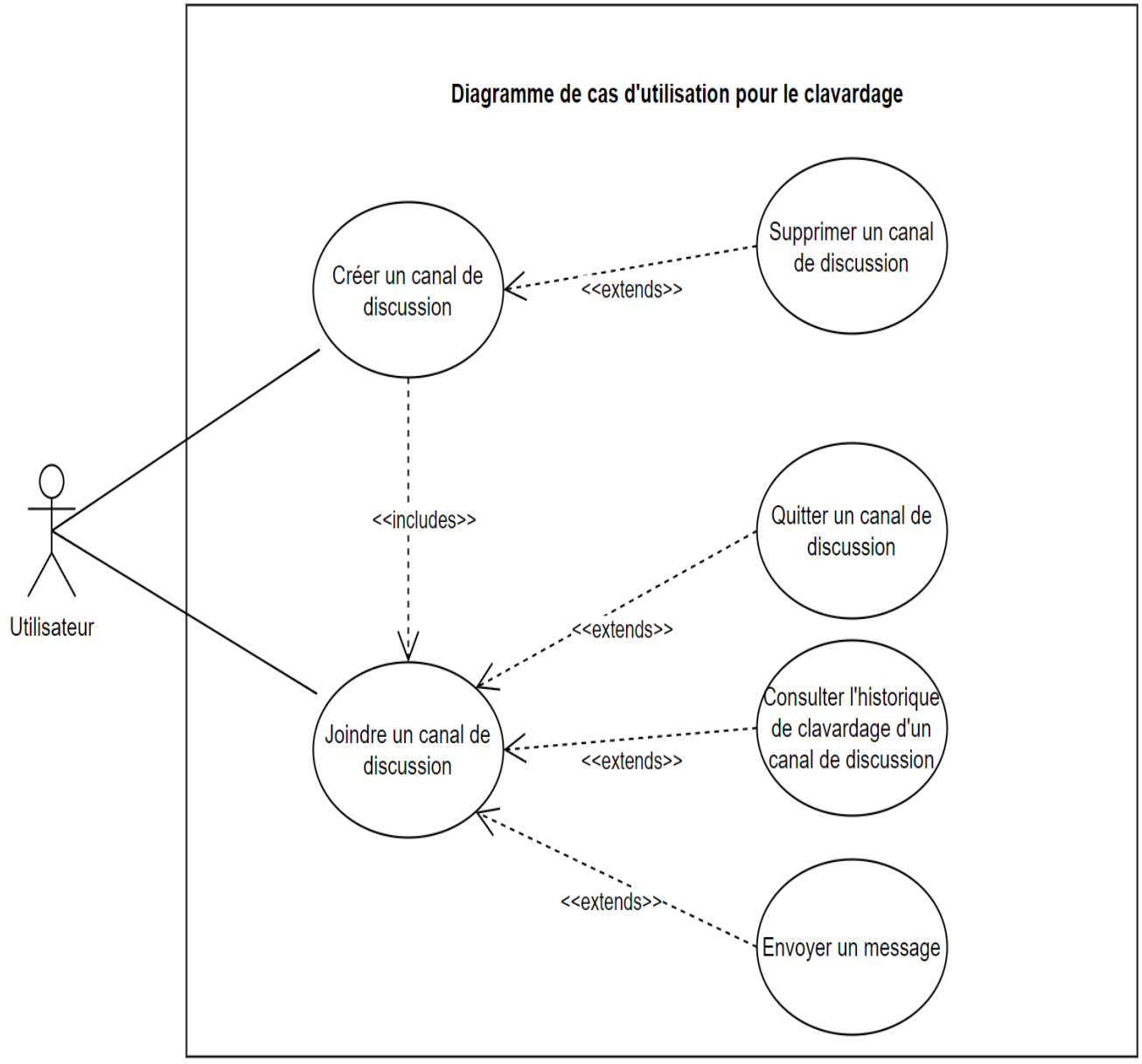

Figure 2 : Diagramme de cas d'utilisation pour le clavardage

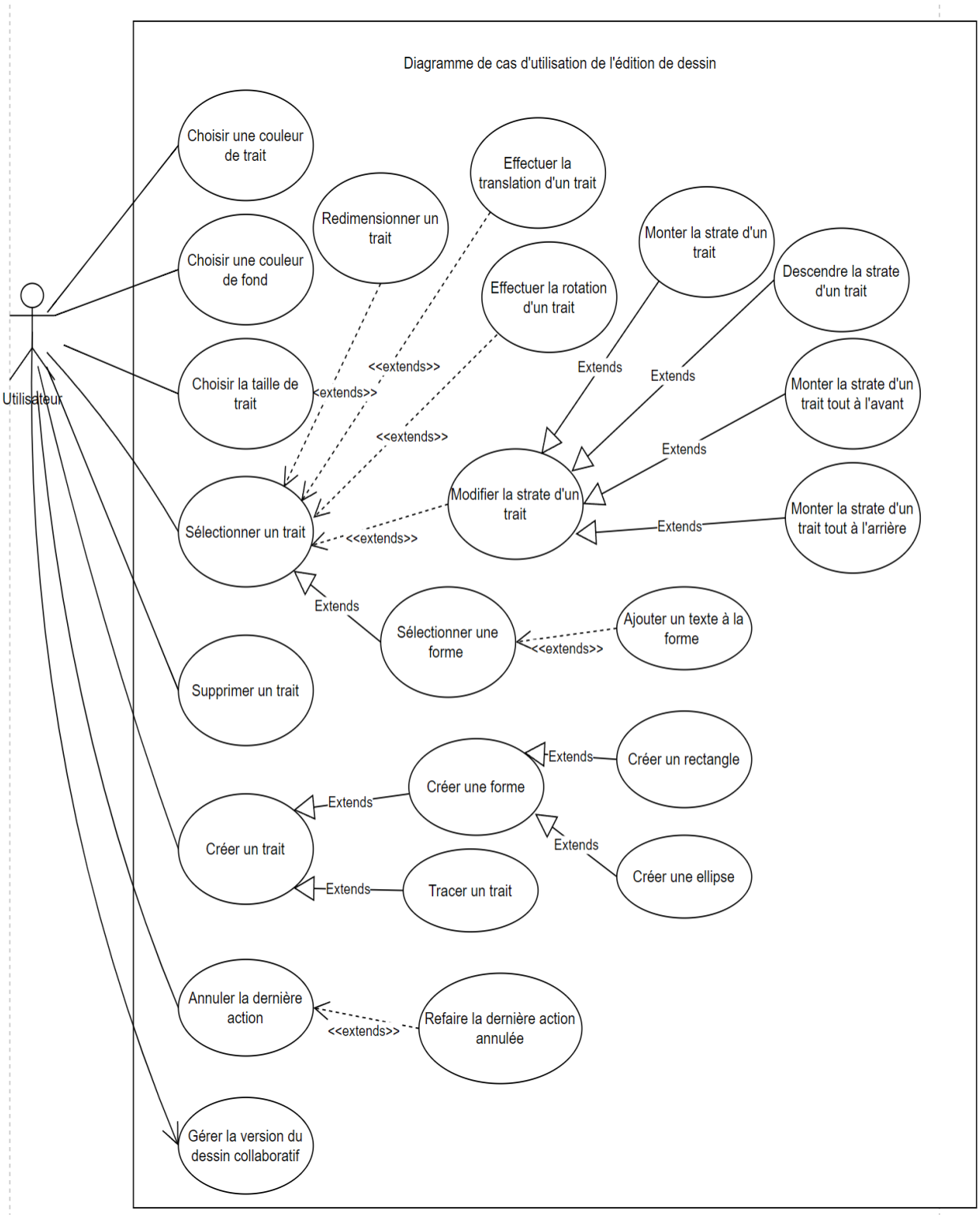

Figure 3: Diagramme de cas d'utilisation de l'édition de dessin

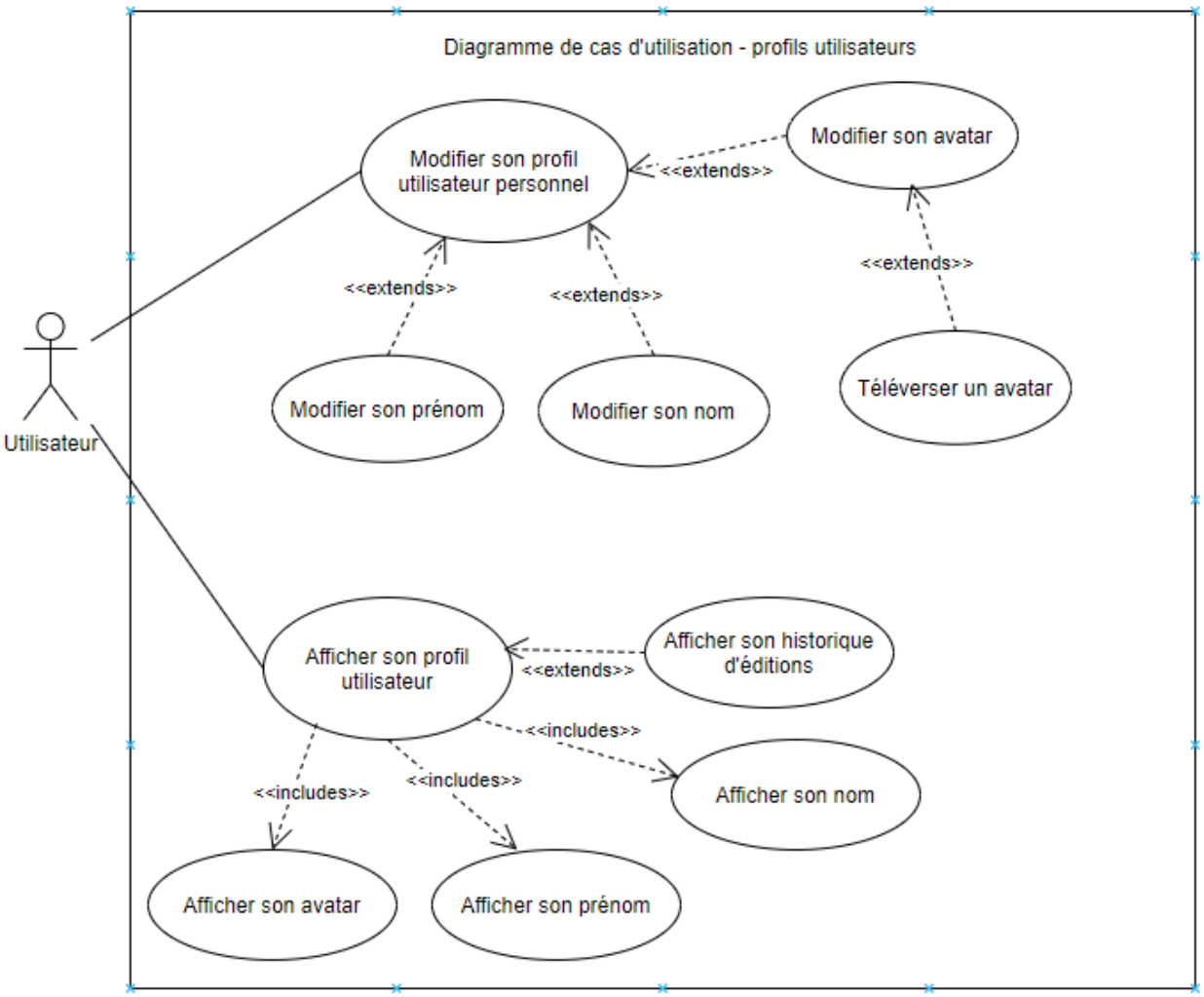

Figure 4: Diagramme de cas d'utilisation des profils utilisateurs

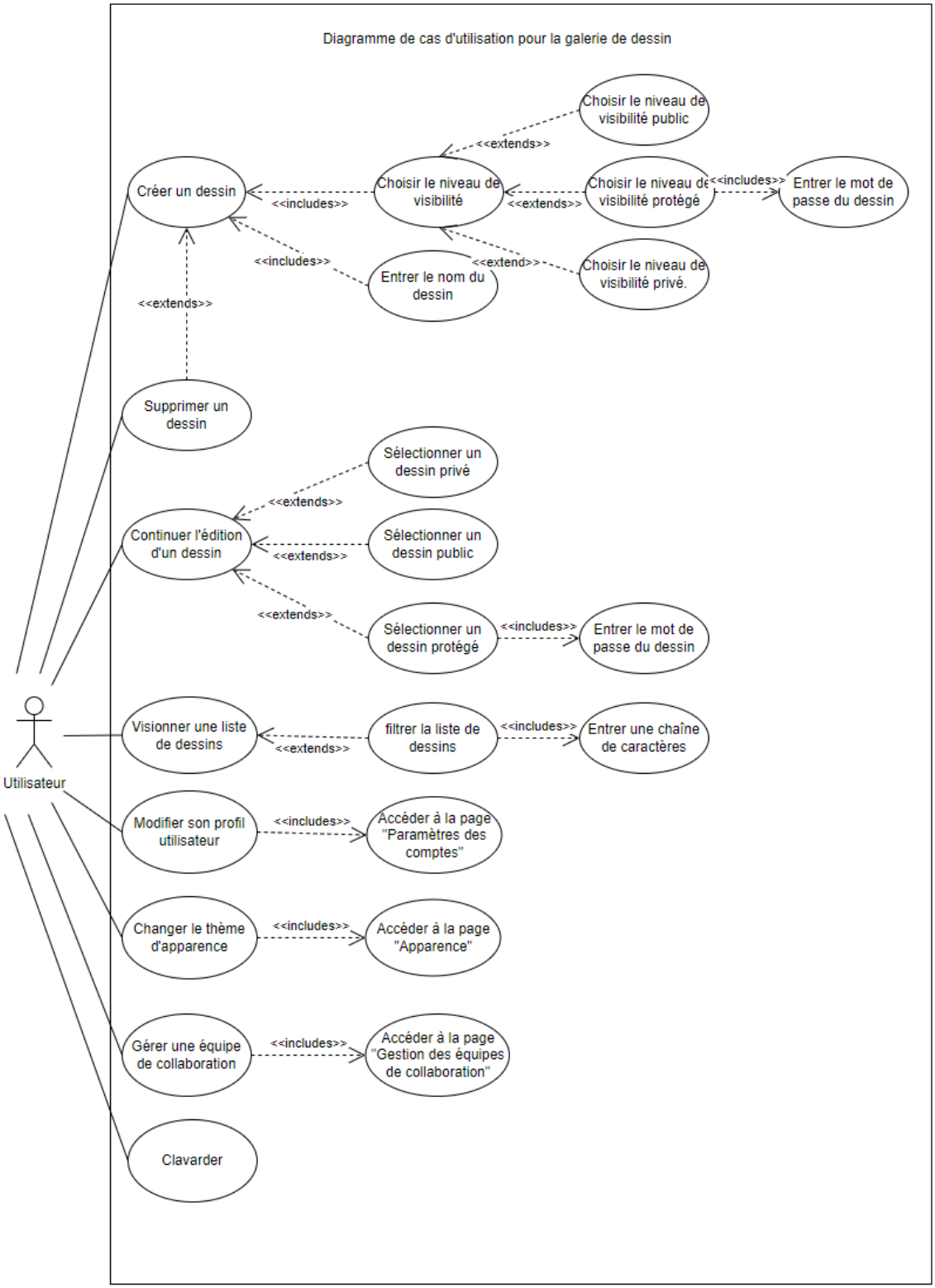

Figure 5: Diagramme de cas d'utilisation pour la galerie de dessin

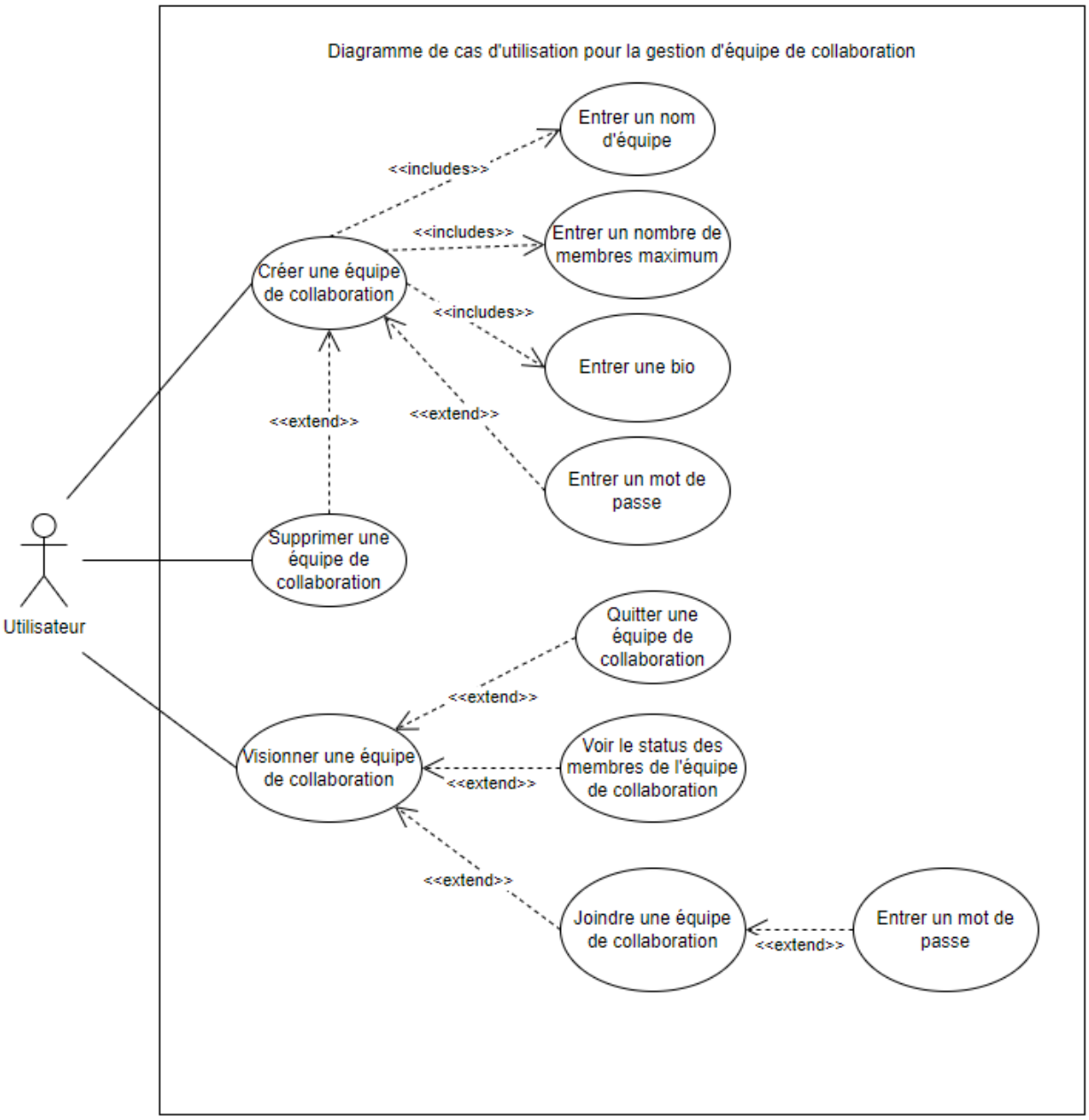

Figure 6: Diagramme de cas d'utilisation pour la gestion d'équipe de collaboration

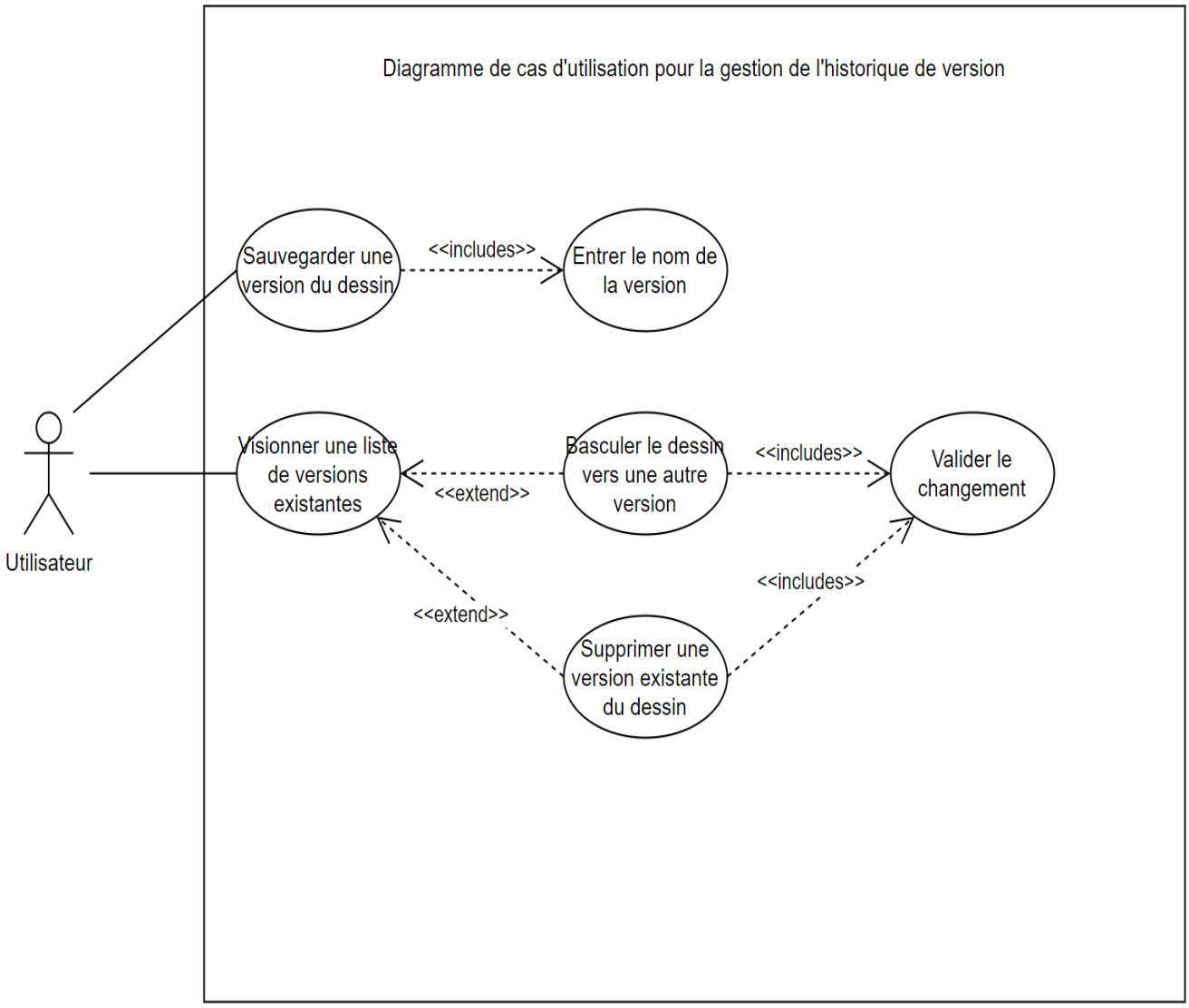

Figure 7: Diagramme de cas d'utilisation pour la gestion de l'historique de version

4. Vue logique

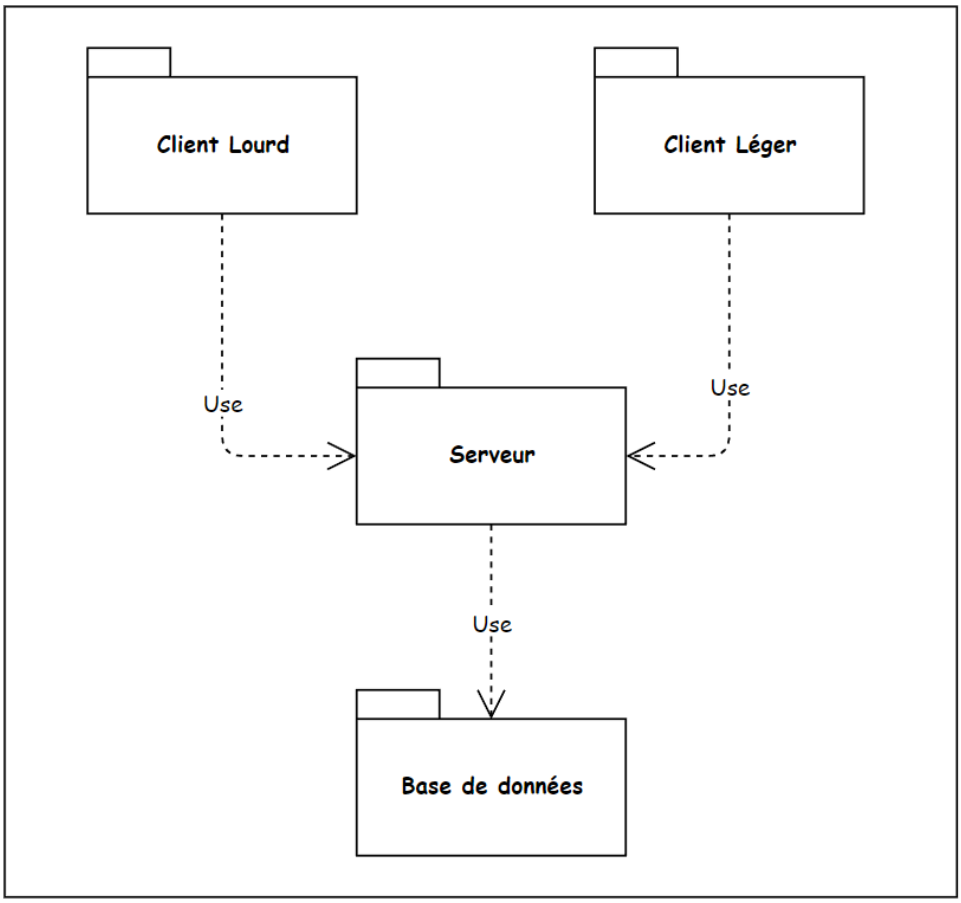

Figure 8: Diagramme de paquetage global du système

Client Lourd
Ce paquetage contient les paquetages du client lourd.

Client Léger
Ce paquetage contient les paquetages du client léger.

Serveur
Ce paquetage contient les paquetages du serveur.

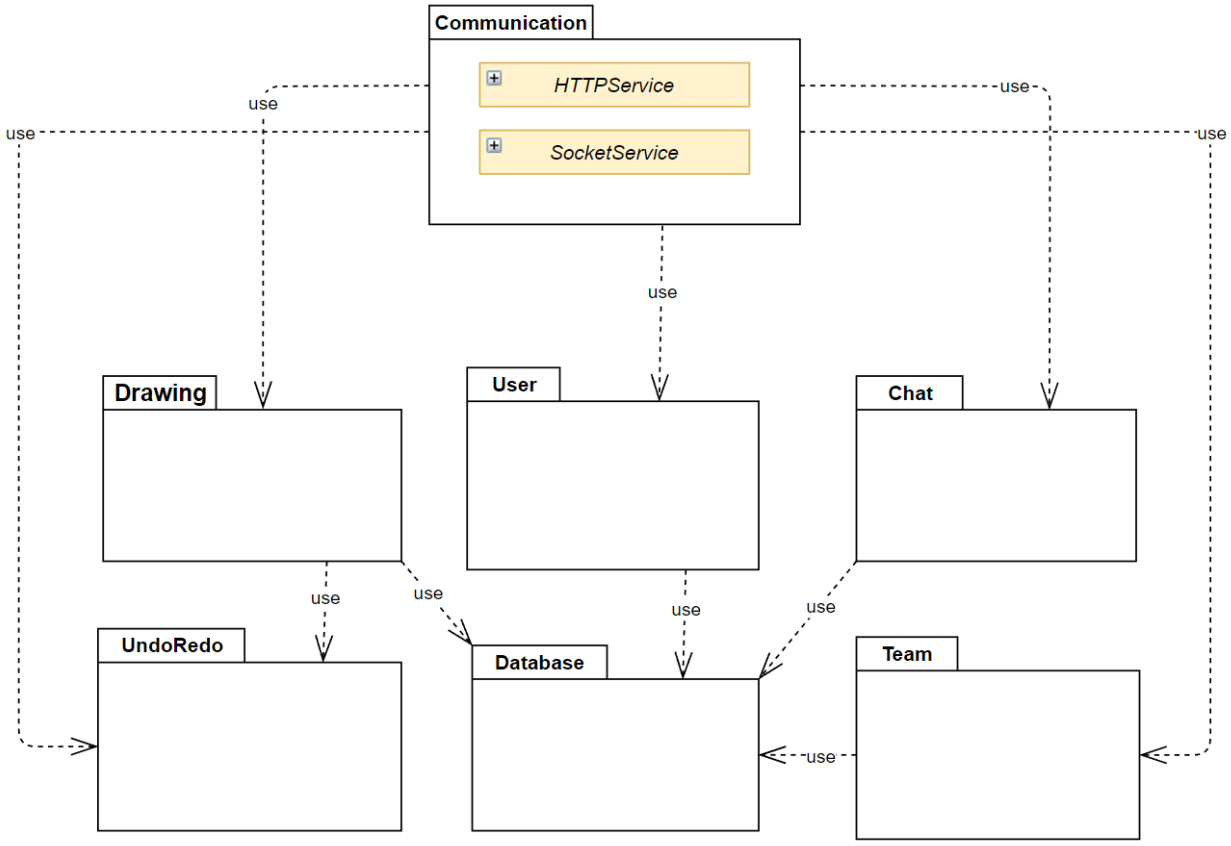

Figure 9: Diagramme de paquetage pour le serveur

Communication
Ce paquetage contient les services permettant d'établir une communication entre le client et le serveur.

Drawing
Ce paquetage contient les classes et services qui s'occupent du dessin collaboratif.

User
Ce paquetage contient les classes et services qui s'occupent de la gestion d'un utilisateur.

Chat
Ce paquetage contient les classes et services qui s'occupent de la gestion du clavardage.

UndoRedo
Ce paquetage contient les classes et services qui s'occupent de la

fonctionnalité permettant de défaire et de refaire.

Database
Ce paquetage contient les services permettant de faire une requête à la base de données.

Team
Ce paquetage contient les classes et services qui s'occupent de la gestion d'une équipe de collaboration.

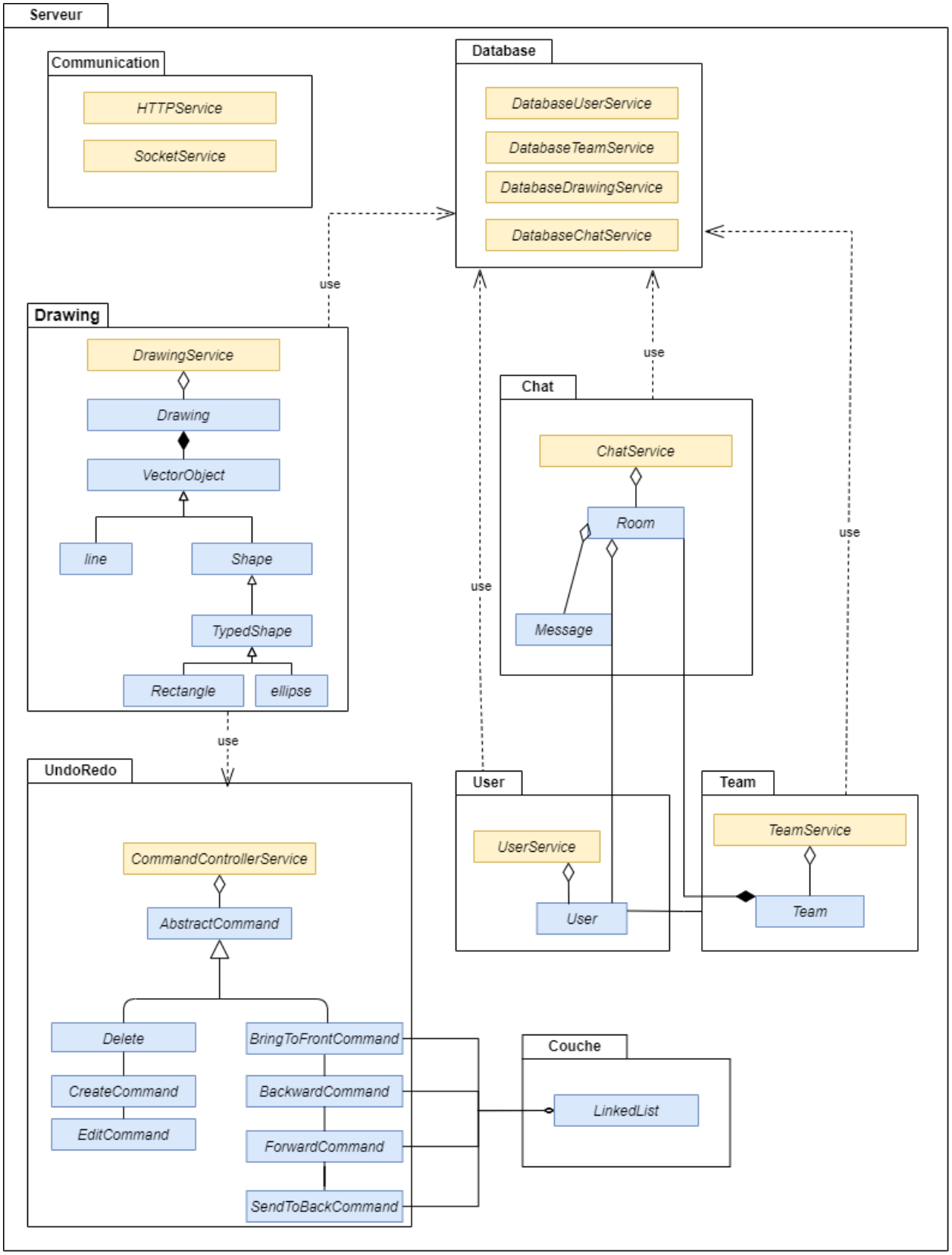

Figure 10: Diagramme de classe de chaque paquet du serveur

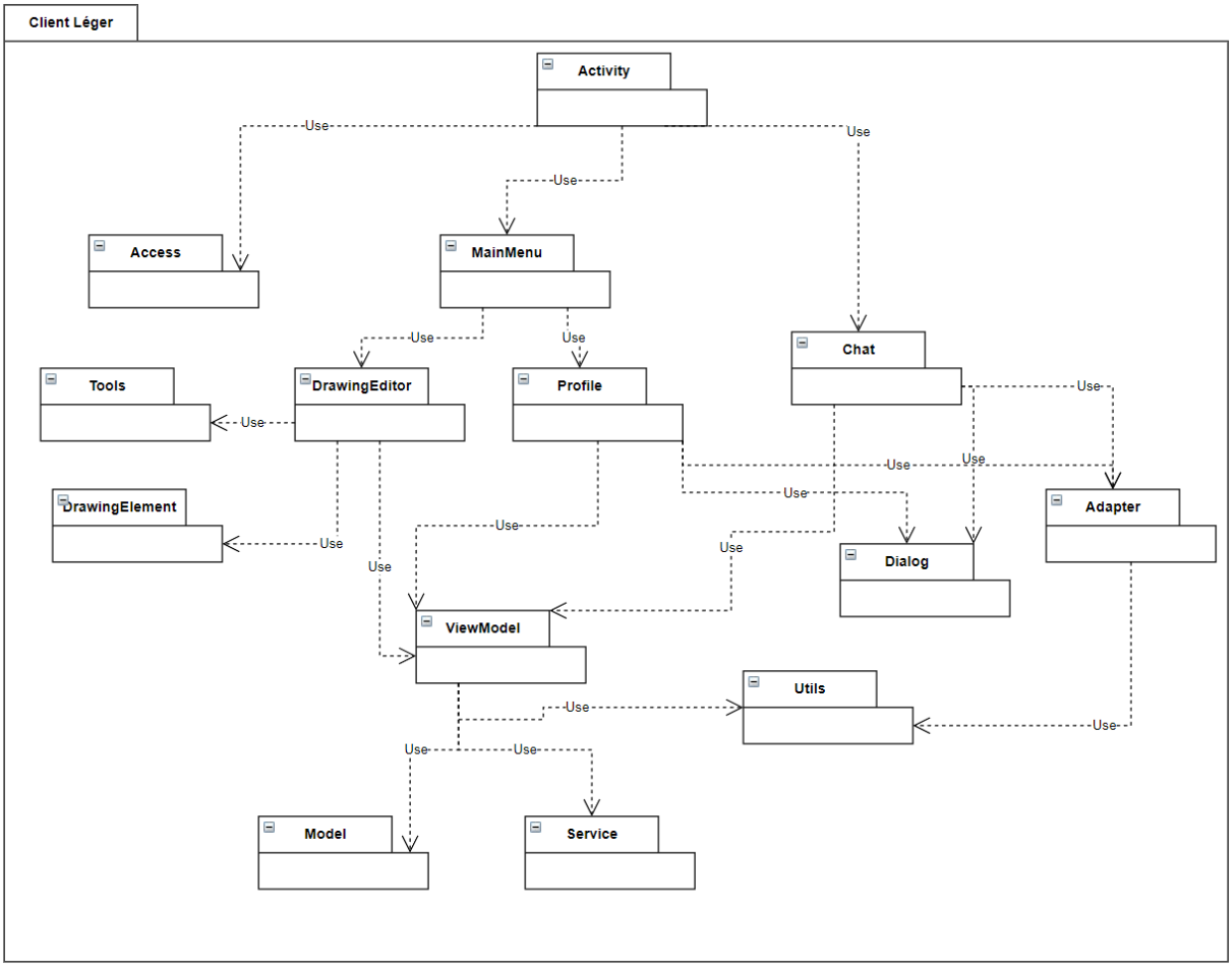

Figure 11: Diagramme de paquetage pour le client léger

Activity
Ce paquetage contient toutes les activités qui enveloppent les interfaces du client léger.

Access
Tous les éléments concernant la connexion et la page d'inscription

MainMenu
Ce paquetage contient toutes les vues du menu principal.

Chat
Ce paquetage contient toutes les vues du clavardage.

Profile
Ce paquetage contient toutes les vues de la page de profil..

DrawingEditor

Ce paquetage contient toutes les vues de l'édition de dessins.

Adapter
Ce paquetage contient tous les « Adapters ».

Dialog
Ce paquetage contient toutes les composantes « Dialogs» utilisées.

DrawingElement
Ce paquetage contient les classes afin de tracer les différents éléments d'un dessin.

Utils
Ce paquetage contient les fonctions et constantes qui sont fréquemment utilisées.

Service
Ce paquetage contient les classes qui fournissent un service tel que la communication avec le serveur.

ViewModel
Ce paquetage contient les classes qui s'occupent de la logique des vues.

Model
Ce paquetage contient les modèles utilisés pour sauvegarder les messages provenant du serveur.

Tools
Ce paquetage contient la logique de tous les outils de dessin.

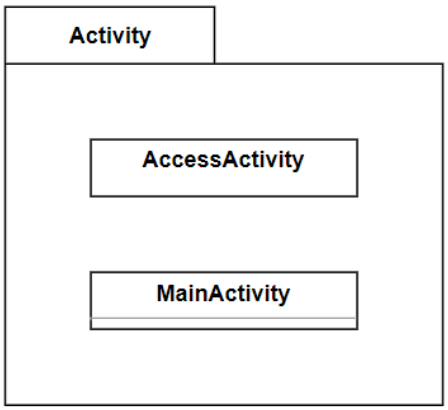

Figure 12: Diagramme de classe du paquetage « Activity »

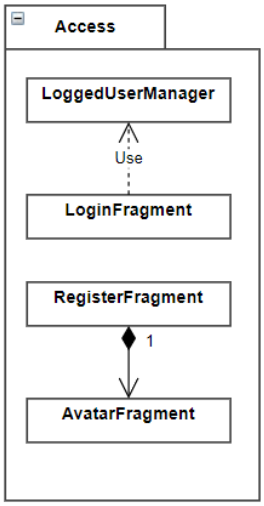

Figure 13: Diagramme de classe du paquetage « Access»

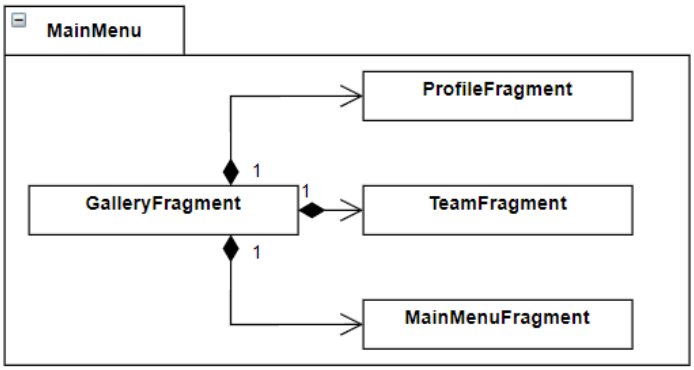

Figure 14: Diagramme de classe du paquetage « MainMenu »

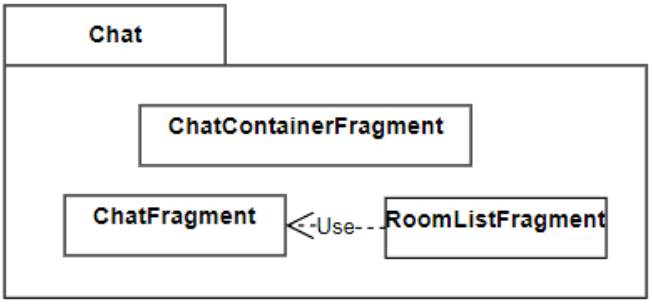

Figure 15: Diagramme de classe du paquetage « Chat»

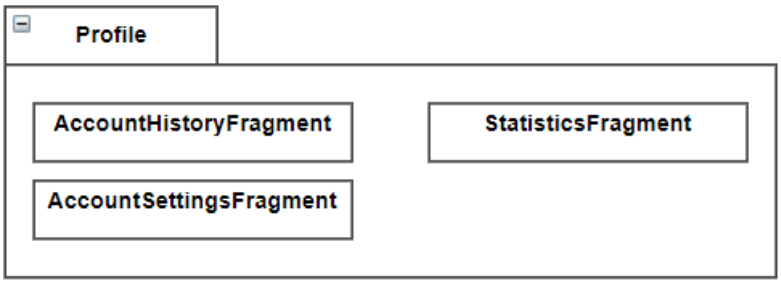

Figure 16: Diagramme de classe du paquetage « Profile»

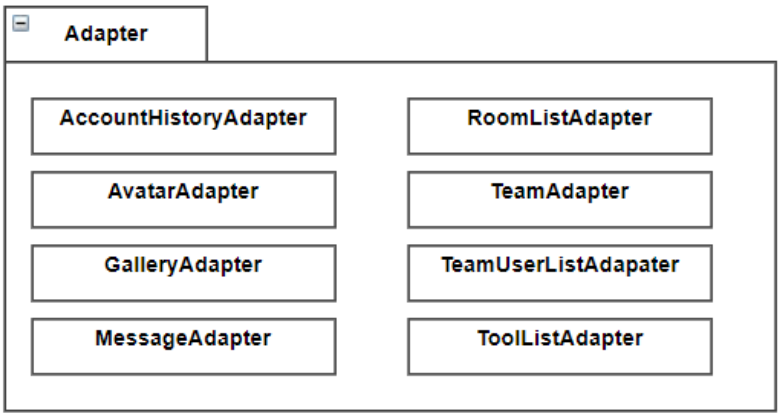

Figure 17: Diagramme de classe du paquetage « Adapter»

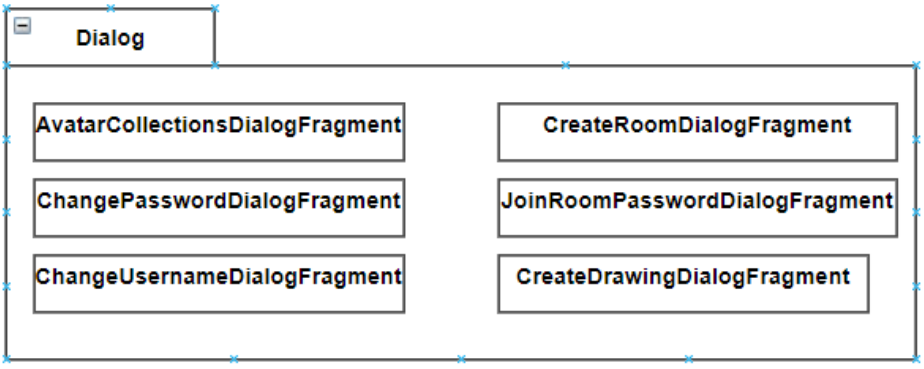

Figure 18: Diagramme de classe du paquetage « Dialog»

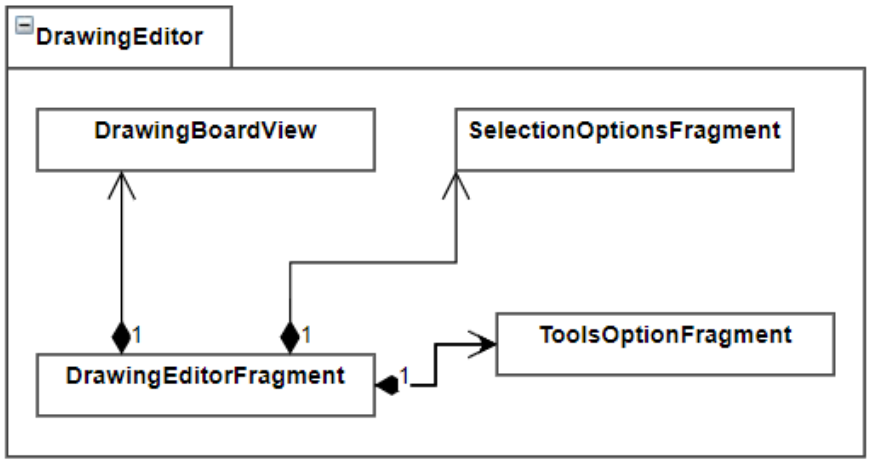

Figure 19: Diagramme de classe du paquetage « DrawingEditor»

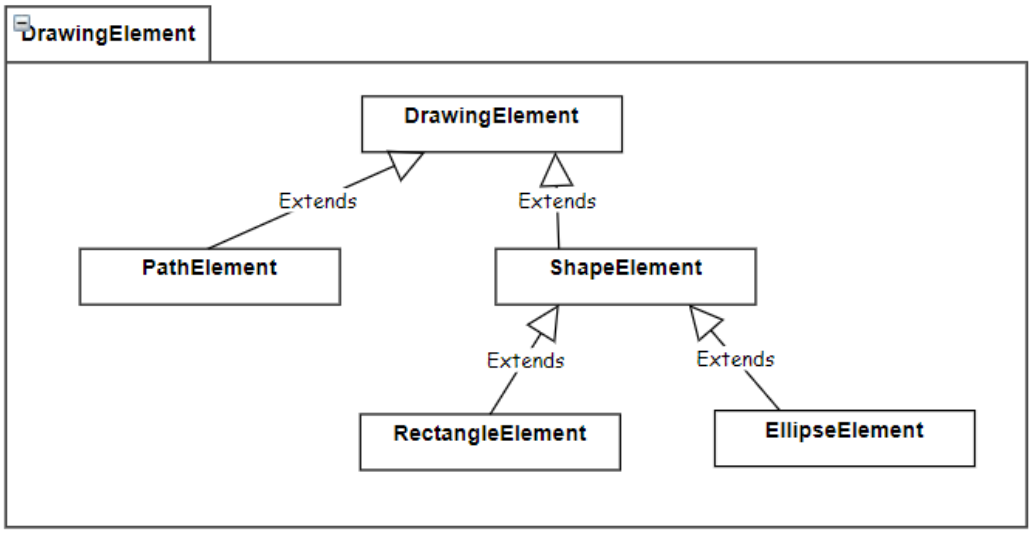

Figure 20: Diagramme de classe du paquetage « DrawingElement»

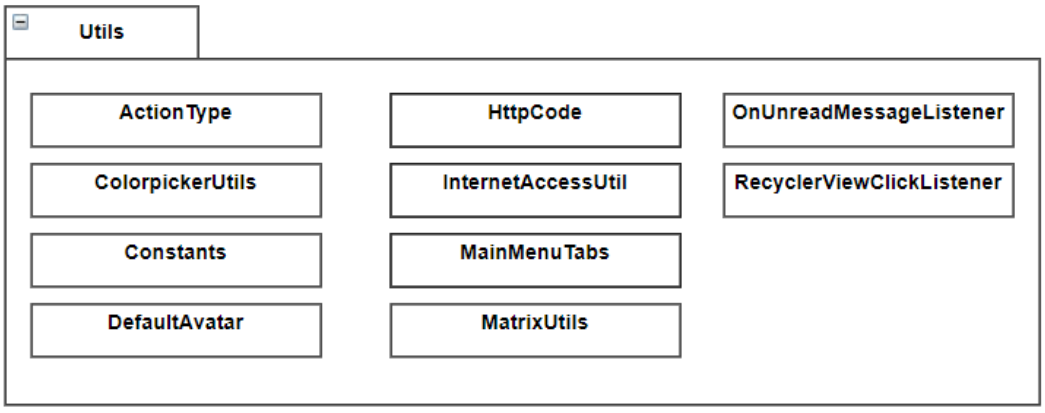

Figure 21: Diagramme de classe du paquetage « Utils»

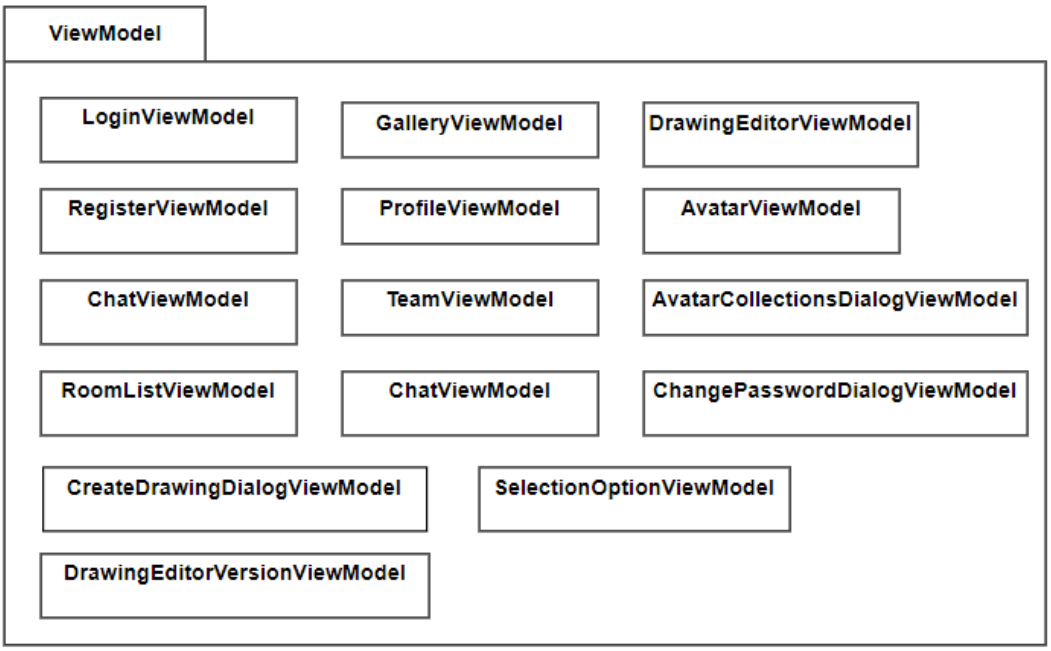

Figure 22: Diagramme de classe du paquetage « ViewModel »

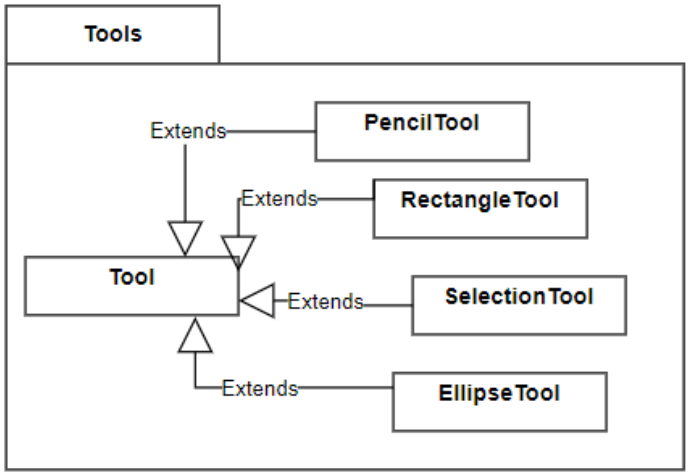

Figure 23: Diagramme de classe du paquetage « Tools »

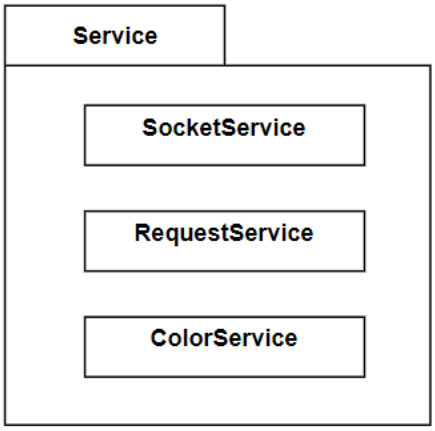

Figure 24: Diagramme de classe du paquetage « Service »

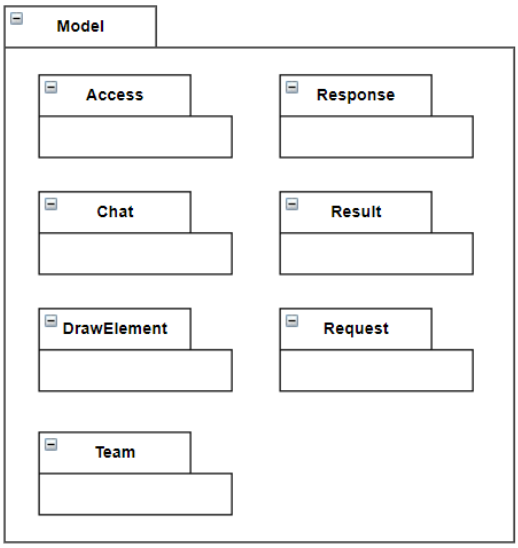

Figure 25: Diagramme de paquetage pour le paquetage « Model »

Access
Ce paquetage contient les modèles utilisés pour les messages provenant du serveur concernant la connexion et l'inscription.

Chat
Ce paquetage contient les modèles utilisés pour les messages provenant du serveur concernant la connexion et l'inscription.

DrawElement
Ce paquetage contient les modèles utilisés par le serveur pour transmettre les informations des différents éléments d'un dessin.

Team

Ce paquetage contient les modèles utilisés par le serveur pour transmettre les informations des équipes de collaborations.

Response
Ce paquetage contient les modèles utilisés par le serveur pour nous transmettre des données.

Result
Ce paquetage contient tous les informations transmises à l'aide de la classe « LiveData »

Request
Ce paquetage contient les modèles utilisés qu'on transmet au serveur lorsqu'on envoie une requête.

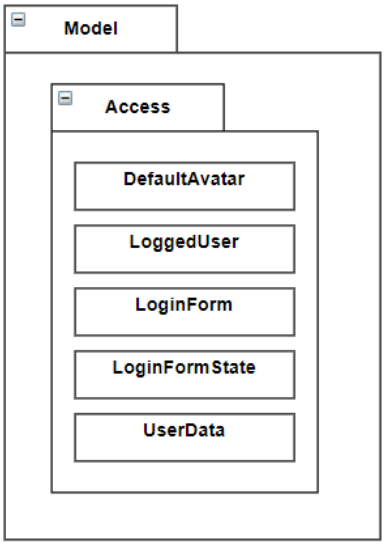

Figure 26: Diagramme de classe du paquetage « Access» contenu dans paquetage « Model»

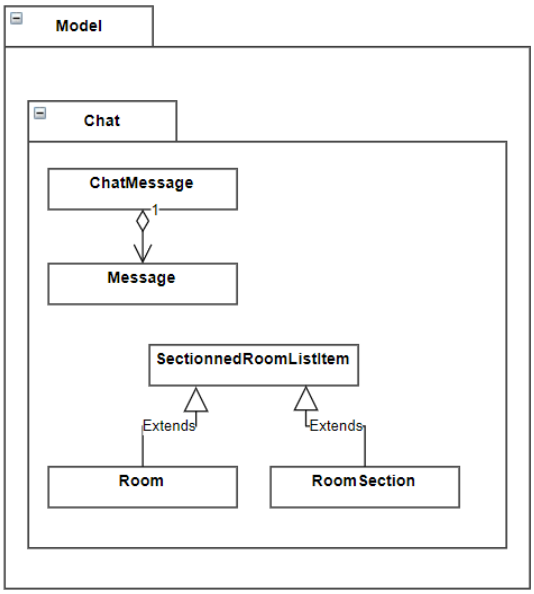

Figure 27: Diagramme de classe du paquetage « Chat » contenu dans paquetage « Model»

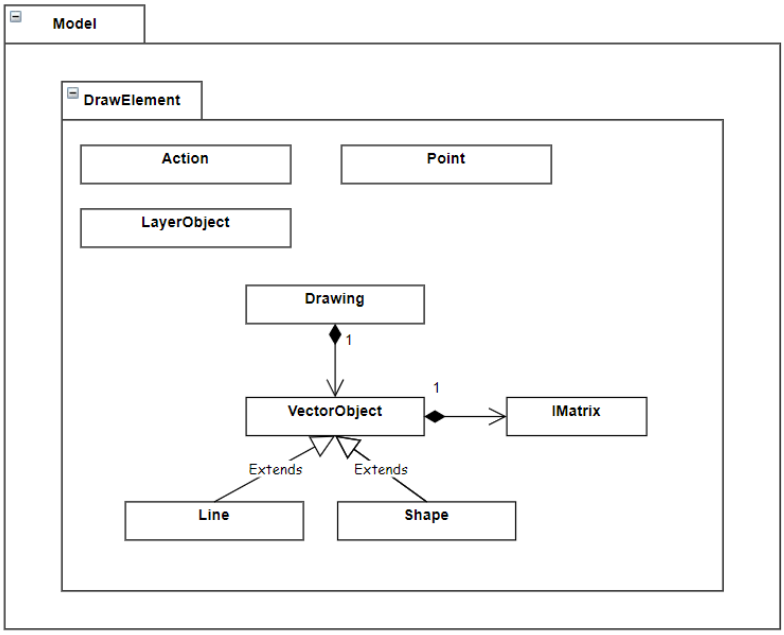

Figure 28: Diagramme de classe du paquetage « DrawElement » contenu dans paquetage « Model»

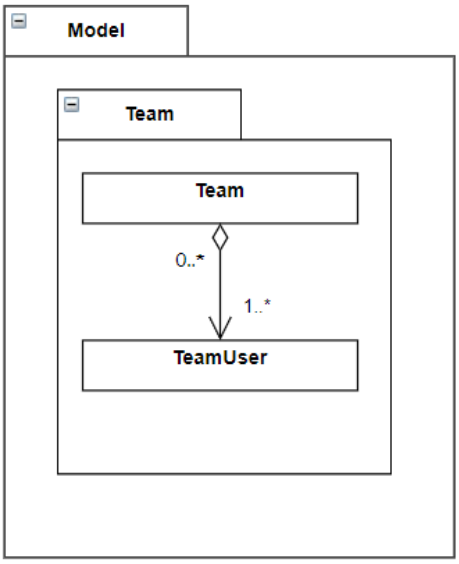

Figure 29: Diagramme de classe du paquetage « Team » contenu dans paquetage « Model»

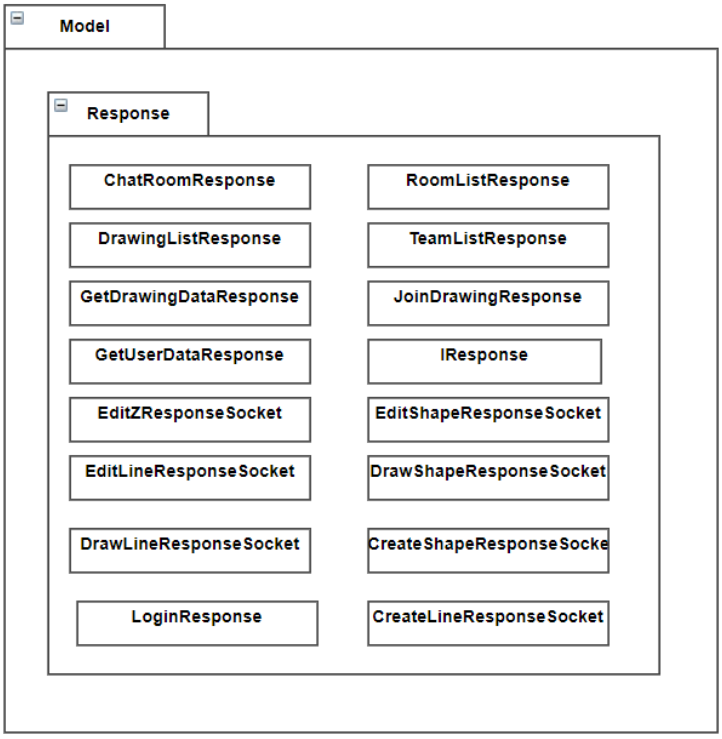

Figure 30: Diagramme de classe du paquetage « Response » contenu dans paquetage « Model»

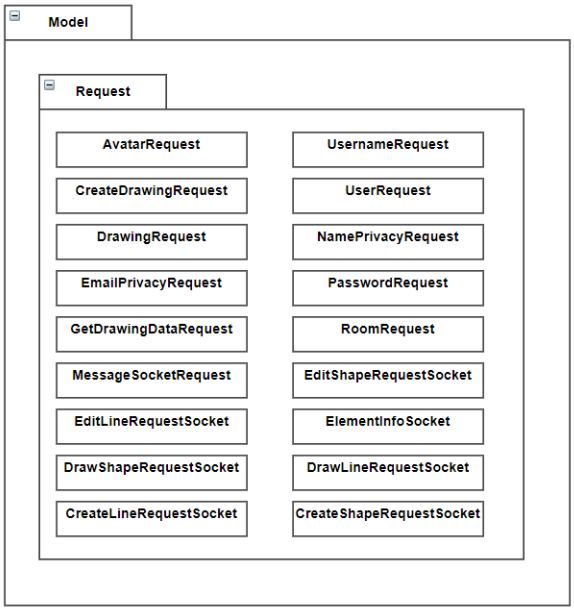

Figure 31: Diagramme de classe du paquetage « Request» contenu dans paquetage « Model»

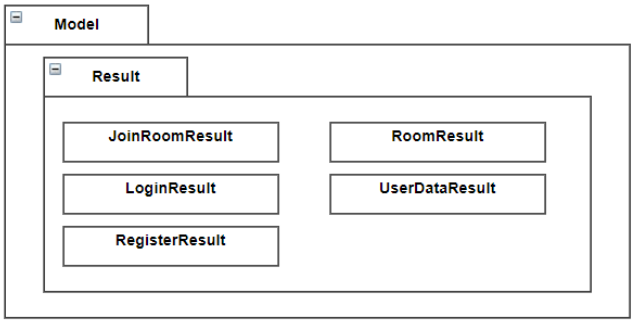

Figure 32: Diagramme de classe du paquetage « Result» contenu dans paquetage « Model»

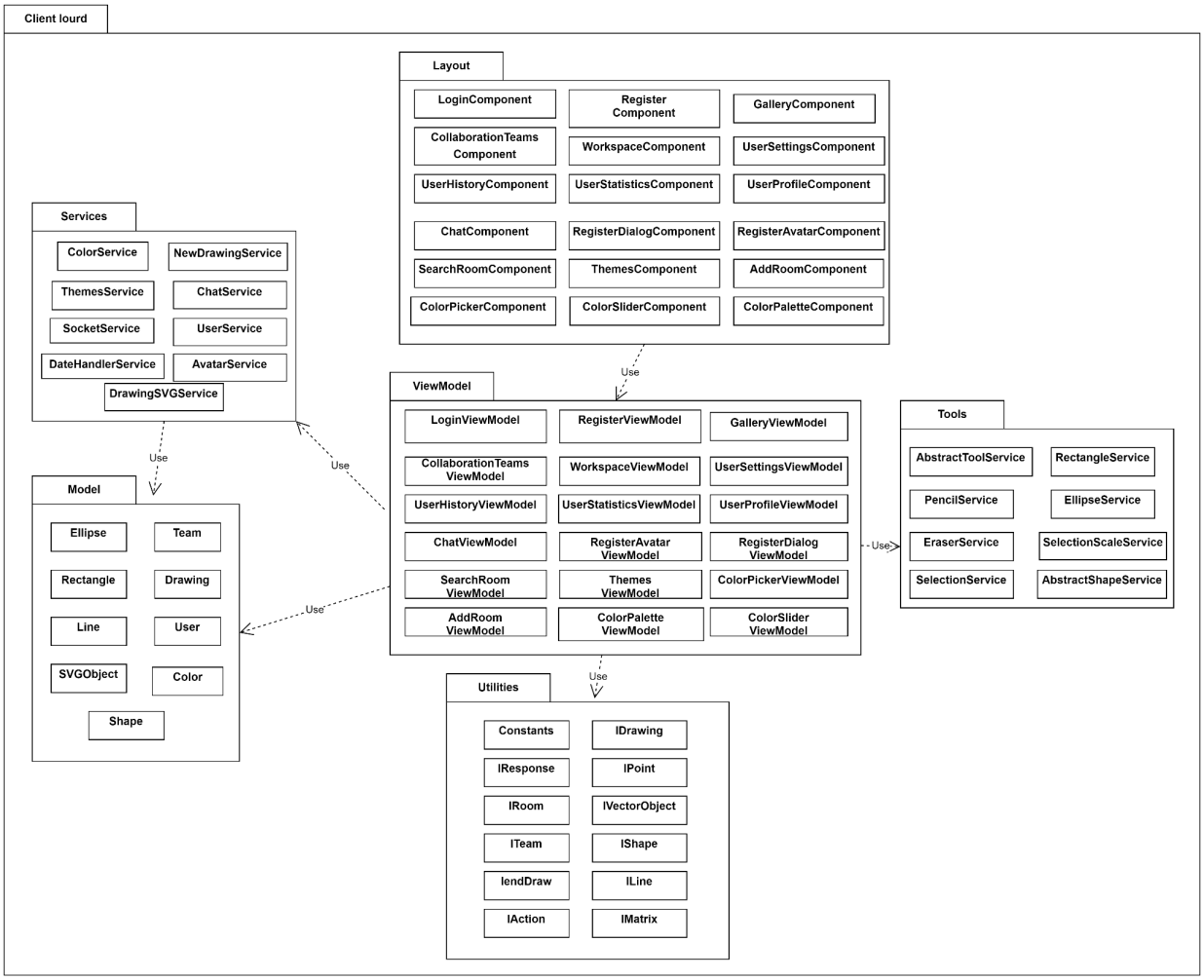

Figure 33: Diagramme de paquetage pour le client lourd

Layout
Ce paquetage contient tous les éléments de vue du client lourd.

ViewModel
Ce paquetage contient les classes qui s'occupent de la logique des vues.

Service
Ce paquetage contient les classes qui fournissent un service tel que le changement de thèmes ou la possibilité de chatter dans l'application.

Model
Ce paquetage contient les modèles utilisés pour sauvegarder les messages provenant du serveur et les dessins provenant de chaque utilisateur.

Tools
Ce paquetage contient la logique de tous les outils de dessin au client lourd.

Utilities
Ce paquetage contient la logique des interfaces et des constantes utiles pour client lourd.

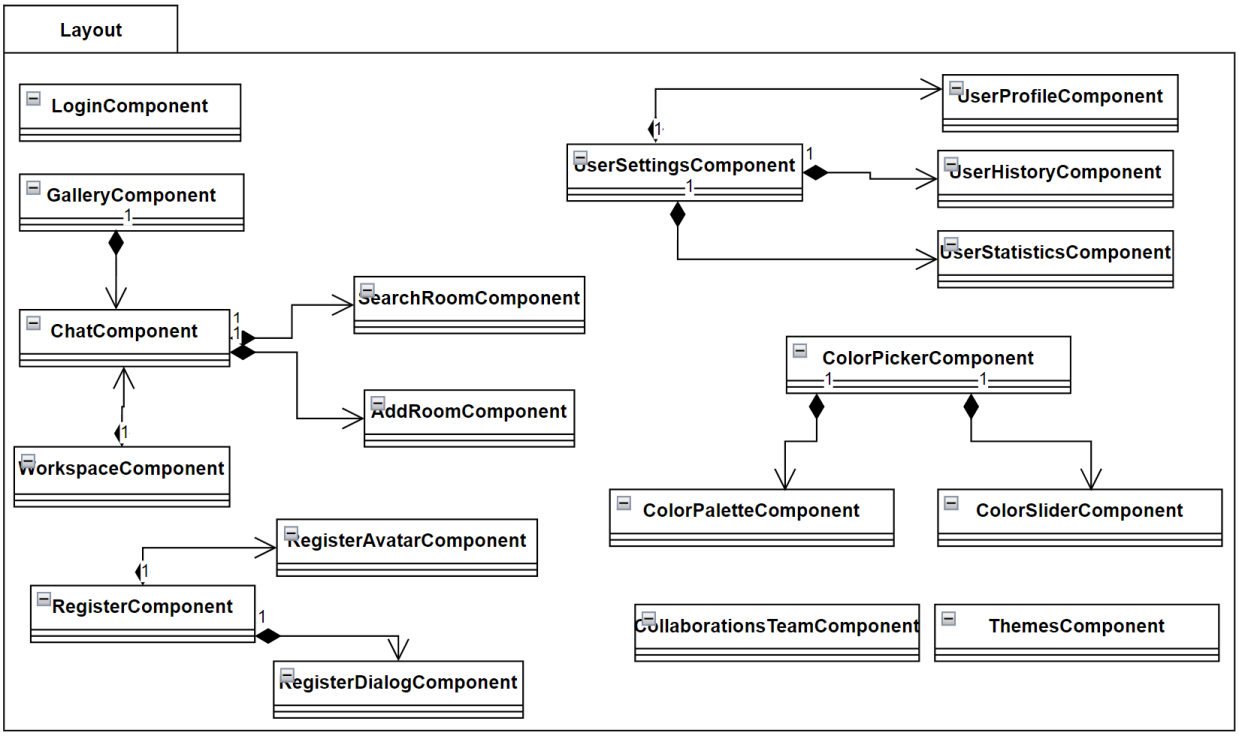

Figure 34: Diagramme de classe du paquetage « Layout »

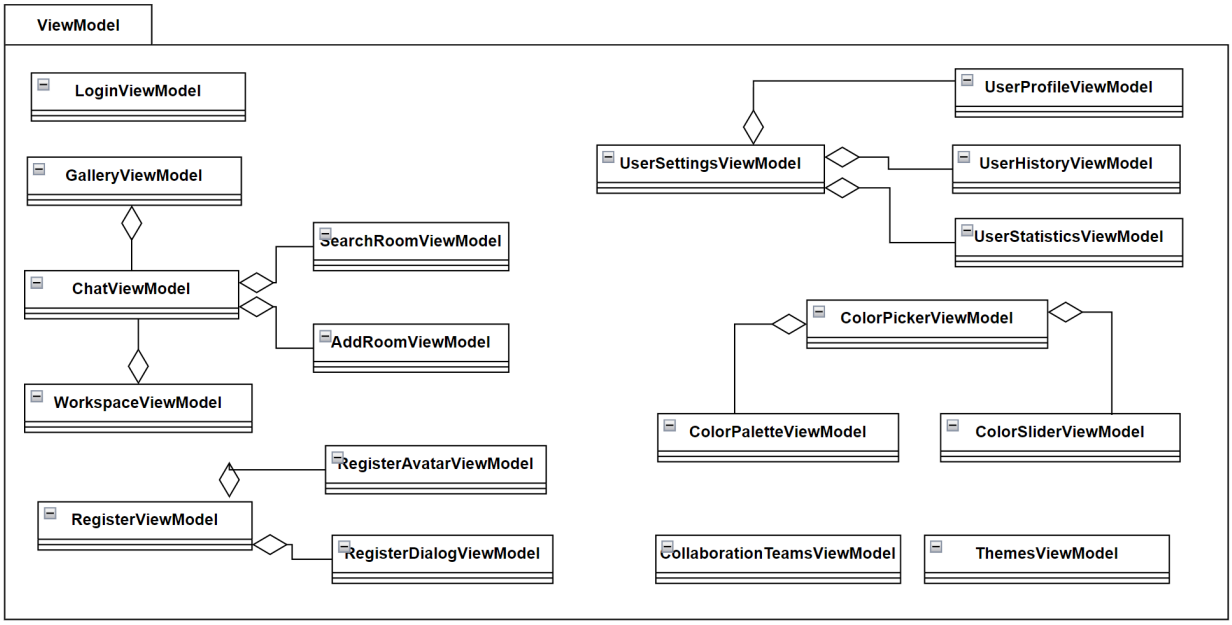

Figure 35: Diagramme de classe du paquetage « ViewModel »

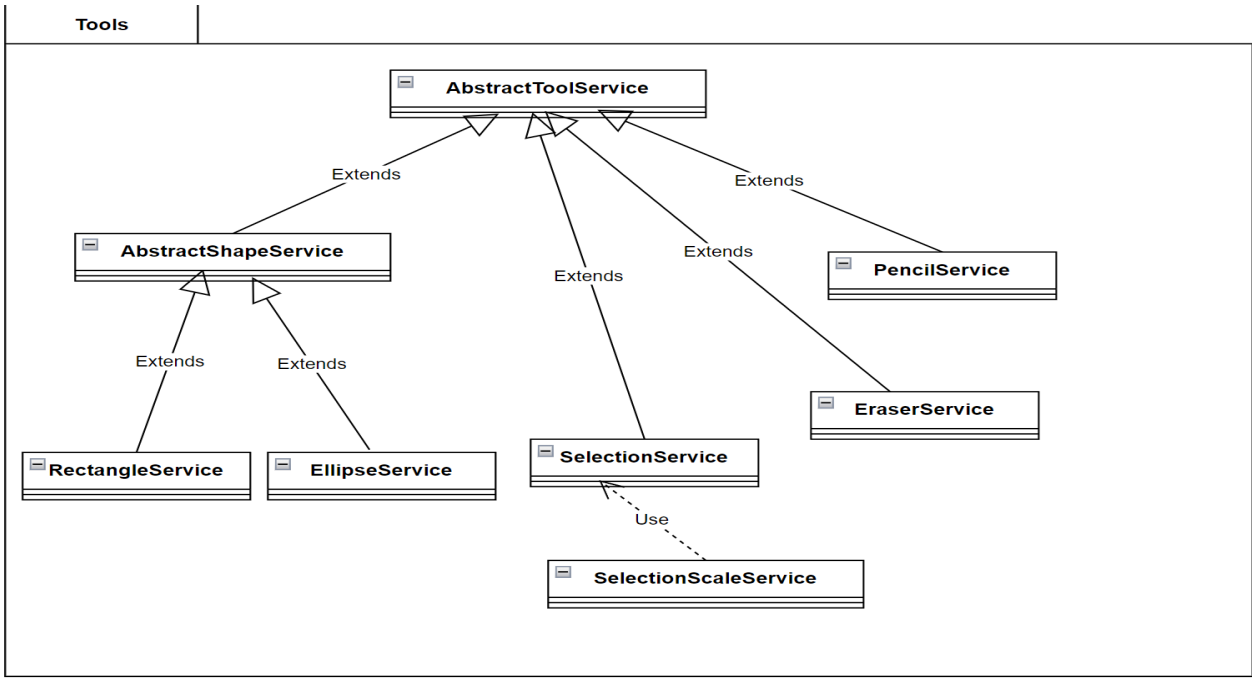

Figure 36: Diagramme de classe du paquetage « Tools »

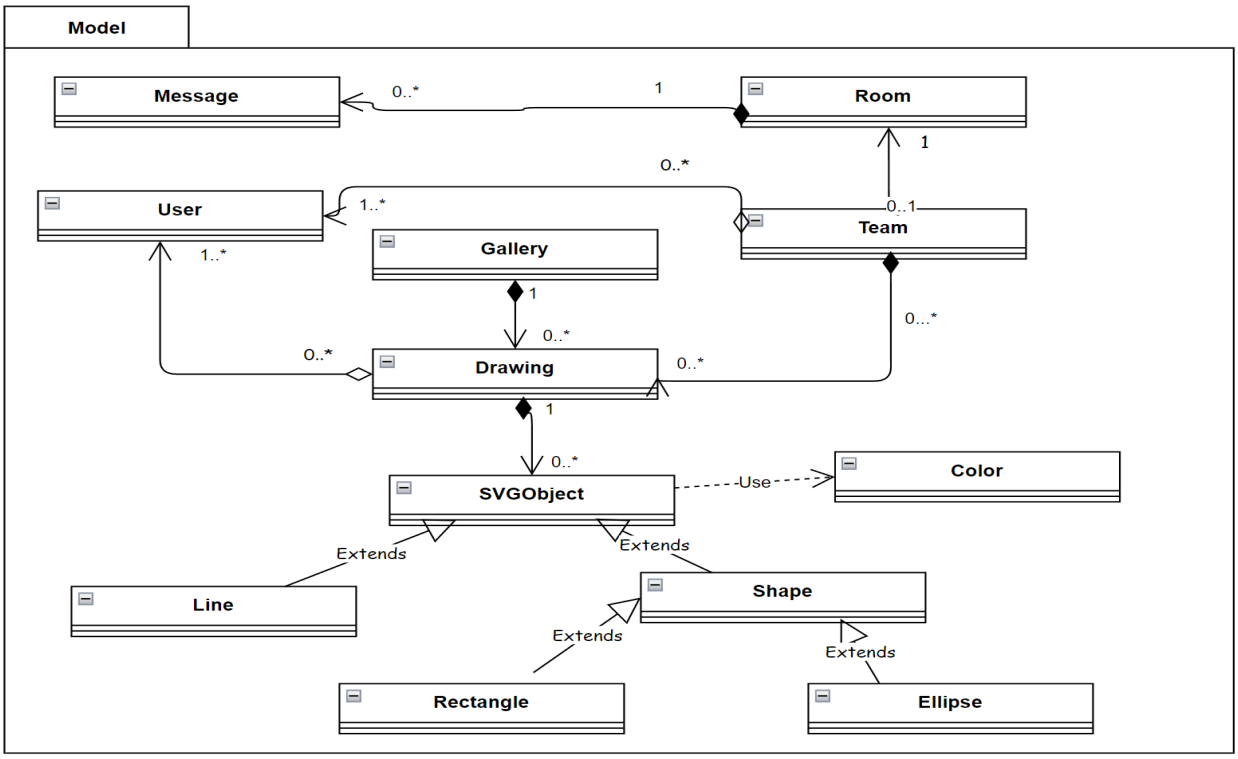

Figure 37: Diagramme de classe du paquetage « Model »

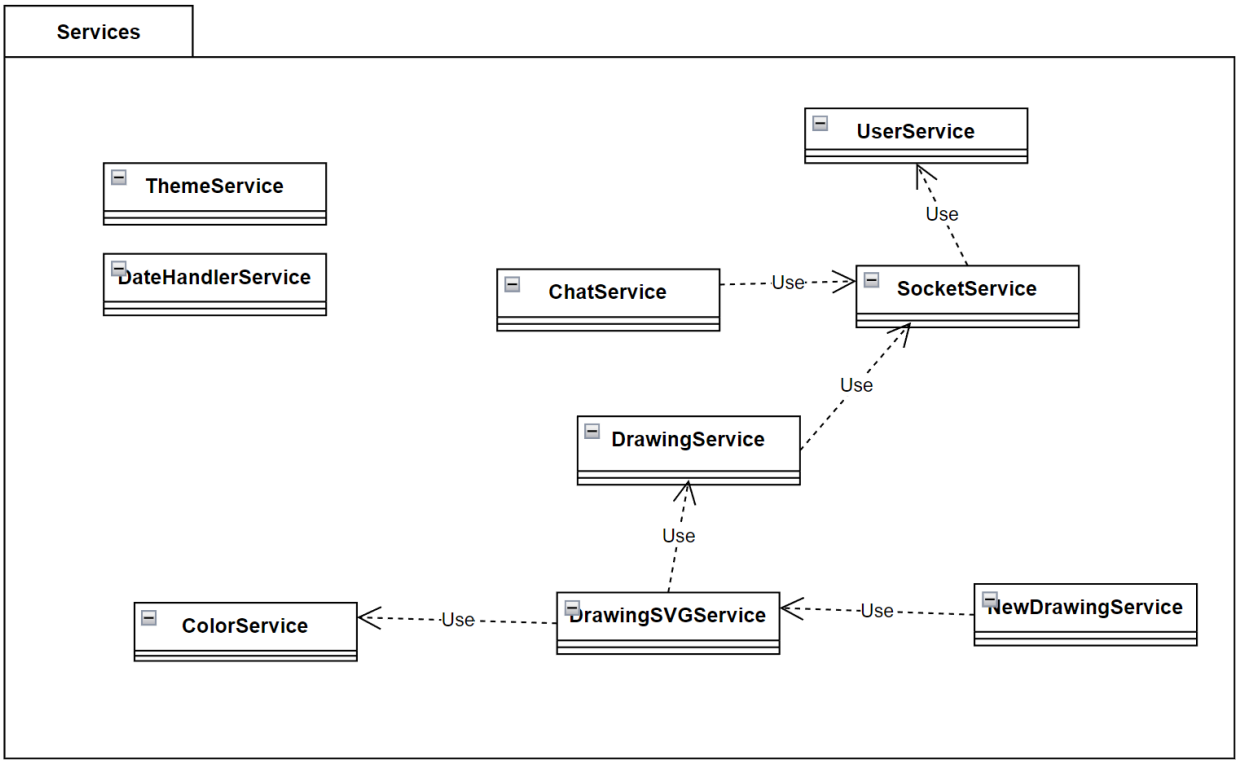

Figure 38: Diagramme de classe du paquetage « Services »

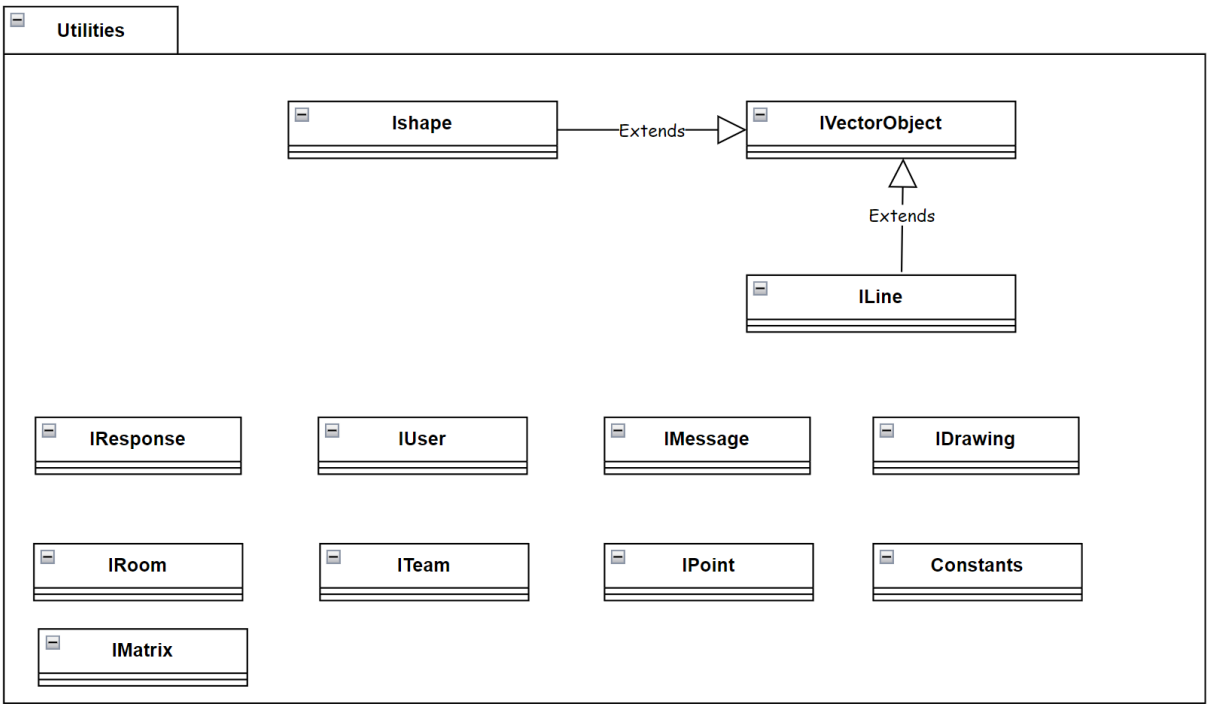

Figure 39: Diagramme de classe du paquetage « Utilities »

5. Vue des processus

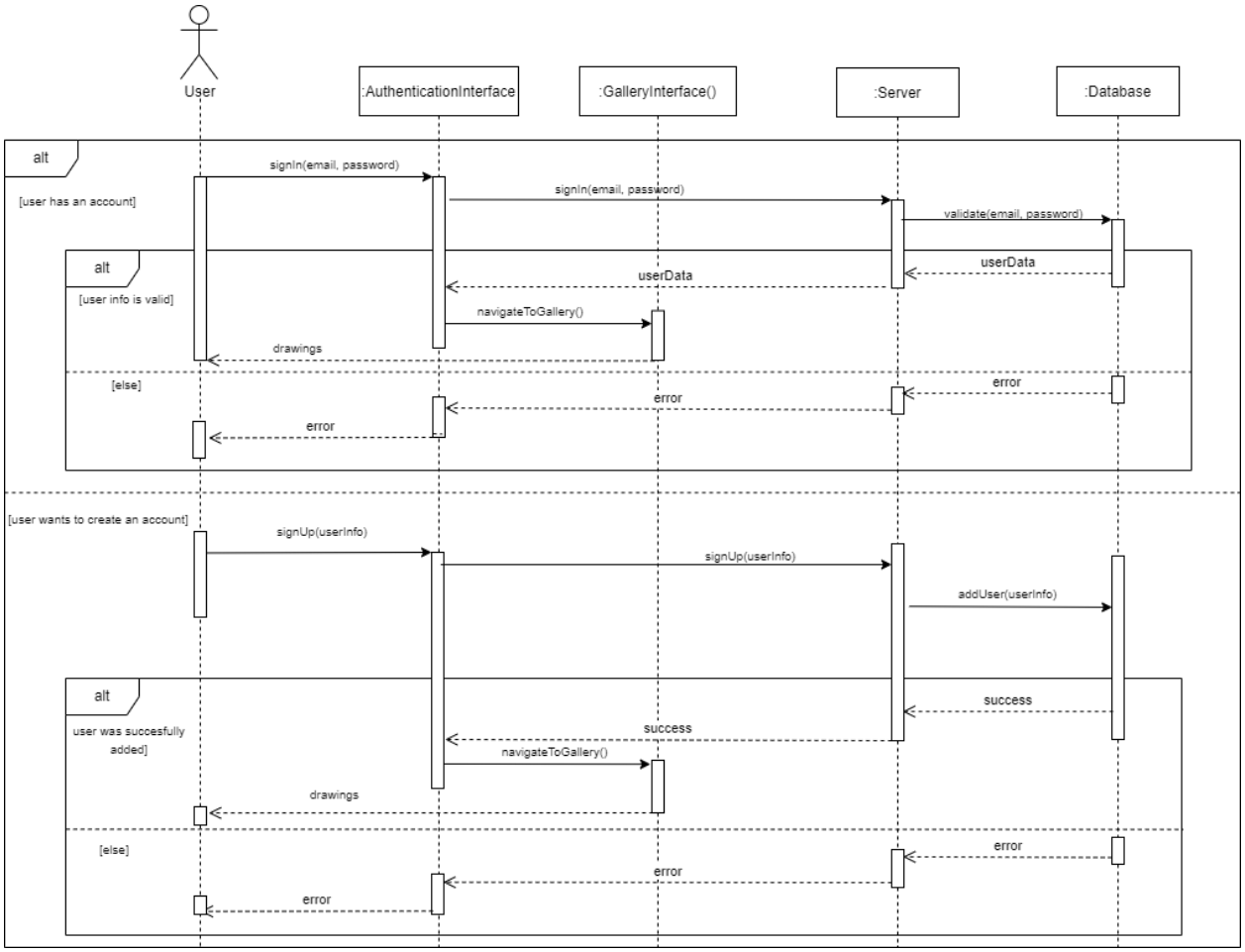

Figure 40: Diagramme de séquence pour l'authentification de l'utilisateur

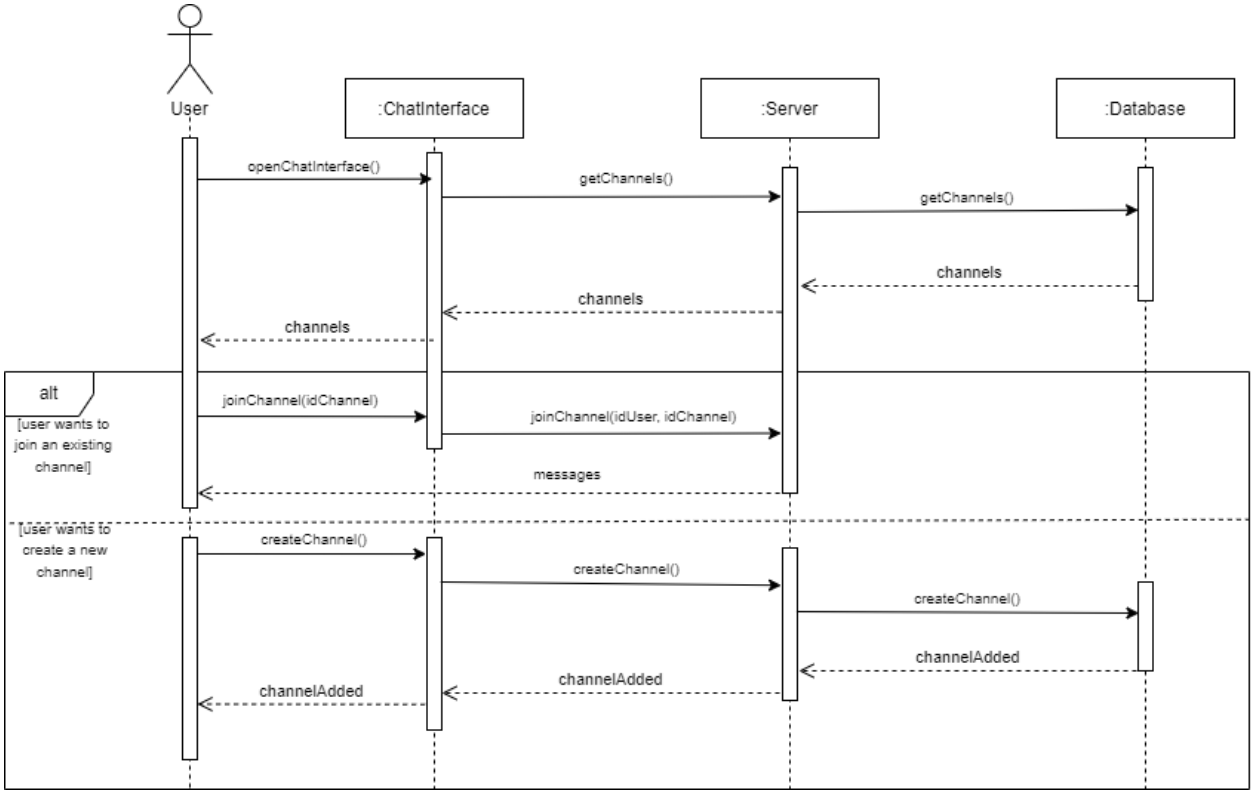

Figure 41: Diagramme de séquence pour joindre et créer un canal de messagerie

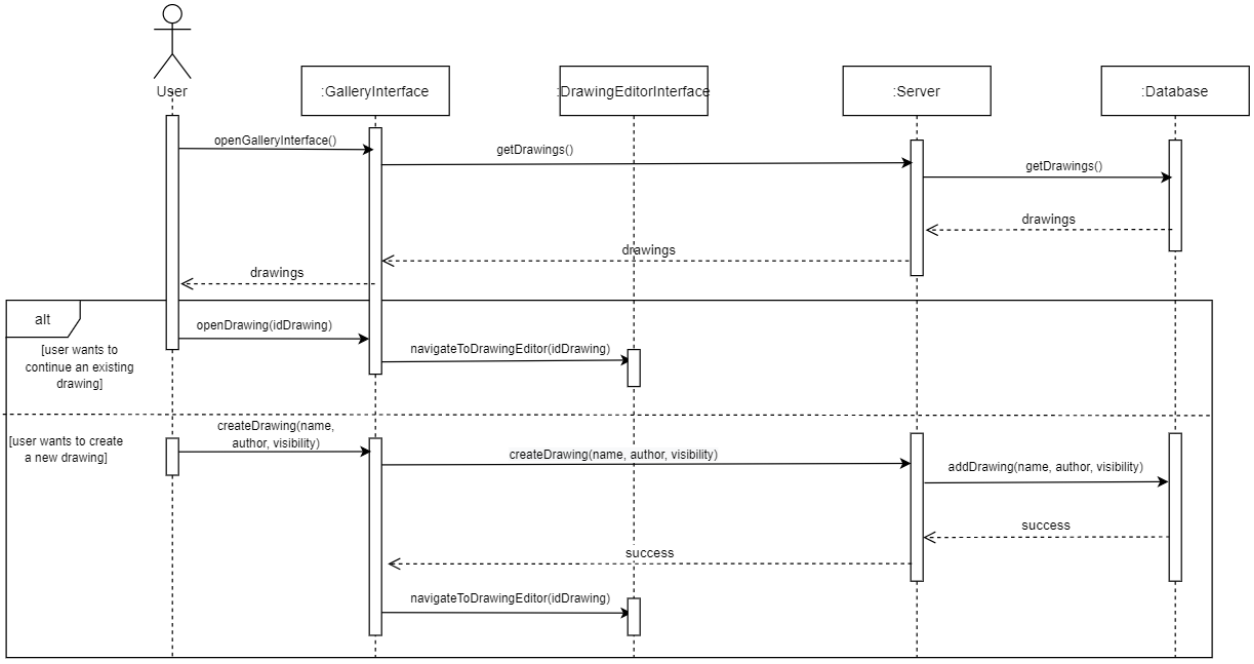

Figure 42: Diagramme de séquence pour créer et continuer un dessin

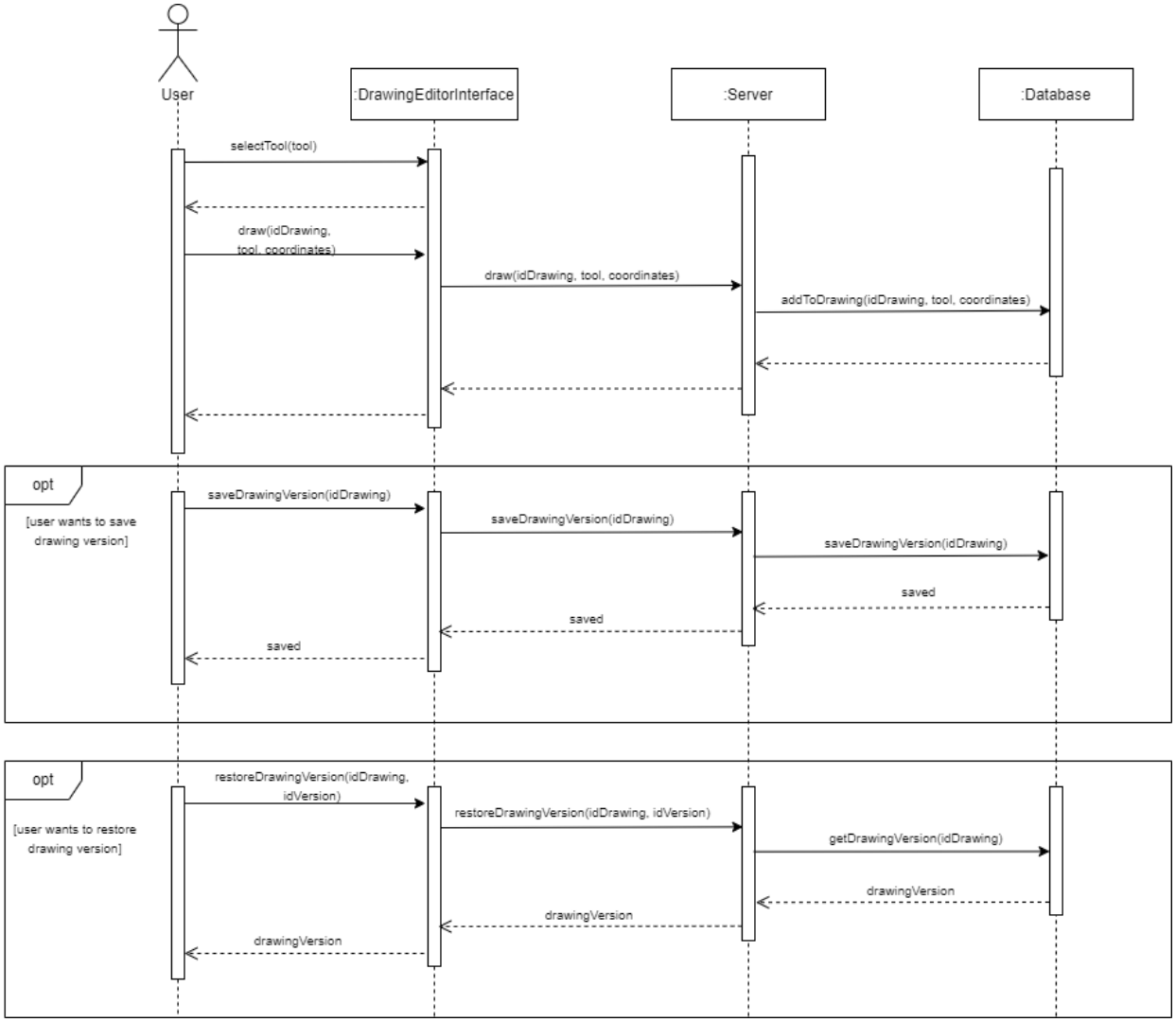

Figure 43: Diagramme de séquence pour dessiner et utiliser l'historique de dessin

6. Vue de déploiement

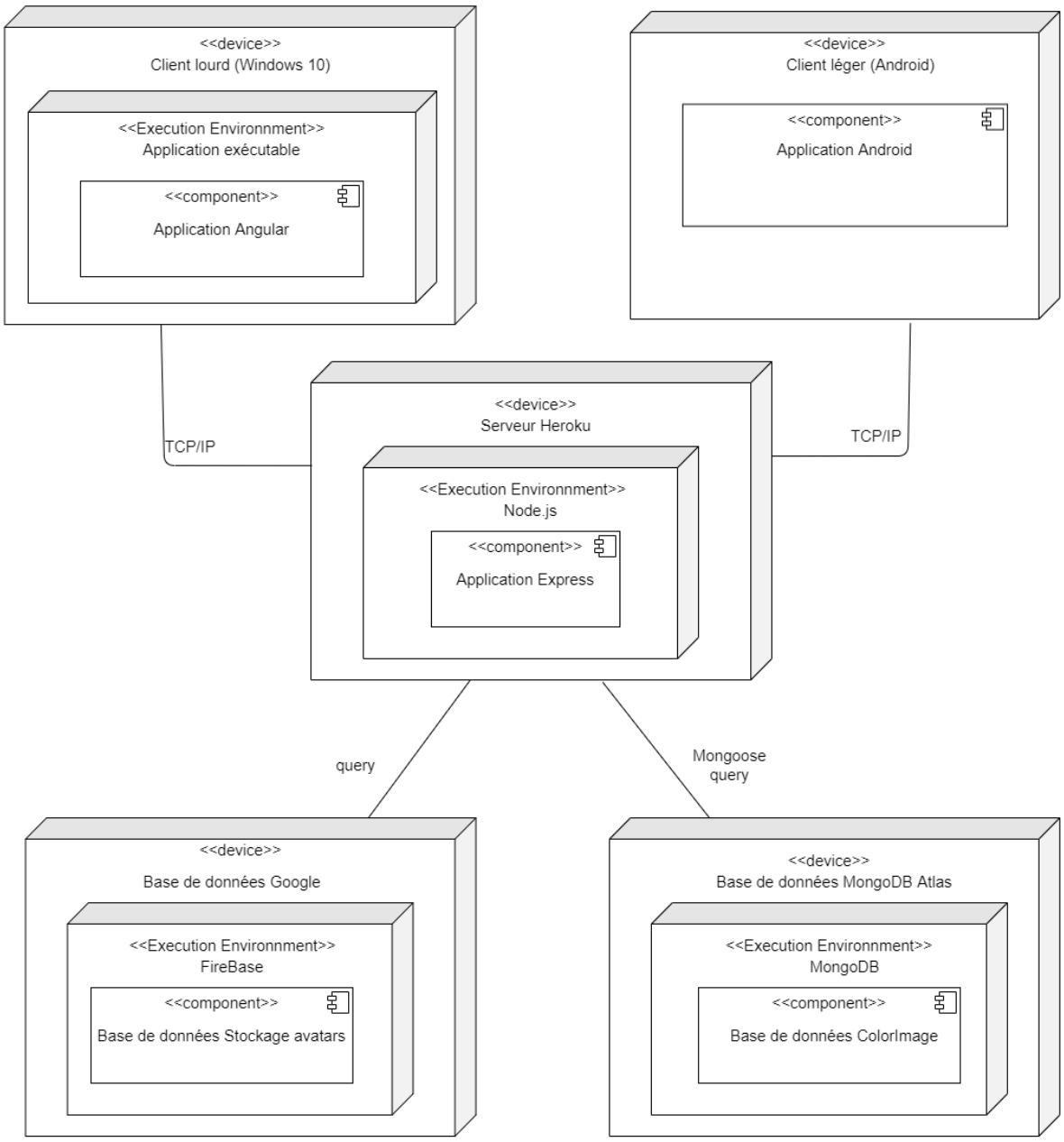

Figure 44: Diagramme de déploiement du système

7. Taille et performance

Afin que l'architecture puisse satisfaire l'utilisateur, plusieurs contraintes de taille et de performance devront être prises en considérant. Premièrement, le client léger impactera fortement l'architecture du projet, car la capacité de mémoire, la performance et la durée de vie de la tablette Samsung Galaxy Tab A est limitée. En effet, avec 2 Go de mémoire vive, on se donne comme objectif de ne pas aller au-delà de 200 Mo. De plus, le système ne devrait pas prendre plus de 250 Mo d'espace mémoire sur l'appareil. De cette manière, on s'assure que le système ne nuit pas à l'expérience utilisateur en s'appropriant trop de ressources sur l'appareil.

Concernant le client lourd, on a évidemment une performance et un espace mémoire plus élevés. Donc, le système pourra occuper un maximum de 500 Mo sur le disque dur et pourra consumer jusqu'à 300 Mo de mémoire vive.

Par ailleurs, étant donné que le projet est un logiciel de dessin collaboratif, on veut s'assurer que les données restent consistent entre les utilisateurs. À cette fin, notre serveur doit avoir un délai maximum de 50 ms pour retourner une réponse. Finalement, notre base de données sera hébergée par MongoDB et on vise un temps de réponse inférieur à 25 ms. Grâce à cela, des fonctionnalités telles que la messagerie et l'édition de dessin seront quasi instantanées.

Bref, notre architecture prend en considération ces différents aspects pour satisfaire l'utilisateur. Ce dernier, ne devrait pas avoir une expérience utilisation négative en raison de la performance du système.