Équipe 109

Fais-moi un dessin
Document d'architecture logicielle

Version 1.5

Historique des révisions

Date	Version	Description	Auteur
2020-01-28	1.0	Première rédaction	Florence Cimon-Paquet, Geneviève Laroche, Annie Rochette
2020-02-04	1.1	Ajout des diagrammes des cas d'utilisation, des diagrammes de séquence et du diagramme de déploiement	Florence Cimon-Paquet, Geneviève Laroche, Annie Rochette
2020-02-06	1.2	Ajout de la section 7	Florence Cimon-Paquet
2020-02-07	1.3	Ajout des diagrammes de paquetages	Maxime Bernier
2020-02-07	1.4	Ajout de la section 2	Geneviève Laroche
2020-02-07	1.5	Révision du document d'architecture	Florence Cimon-Paquet

Table des matières

Document d'architecture logicielle

1. Introduction

Ce document présent décrit l'architecture logicielle du logiciel *Fais-moi un dessin.* Tous les diagrammes d'architecture vont devoir respecter le langage UML.

Dans cet artefact, il sera question d'expliquer les objectifs et les contraintes architecturales, de présenter les cas d'utilisation ainsi que la vue logique en expliquant le rôle de chaque paquetage, la vue des processus en démontrant leurs interactions, la vue de déploiement en montrant la configuration des différentes composantes matérielles physiques et, finalement, les caractéristiques de la taille et de la performance qui pourraient avoir un impact sur l'architecture du logiciel.

2. Objectifs et contraintes architecturaux

2.1 Performance

Le serveur devra être en mesure de supporter plusieurs parties simultanément.

2.2 Sécurité

Afin de garder confidentielles les informations personnelles de nos utilisateurs, les mots de passe devront être encryptés.

2.3 Réutilisation

Pour le client lourd, il ne sera pas nécessaire de développer des outils pour dessiner. En effet, l'application réutilisera certains des outils de l'application *PolyPaint*.

2.4 Langage de développement

Le client lourd sera développé en C#/WPF et le client léger en java. Pour le serveur et la base de donnée, afin de respecter l'échéancier, les développeurs utiliseront des technologies qu'ils connaissent déjà, soit Node.js et MongoDB.

3. Vue des cas d'utilisation

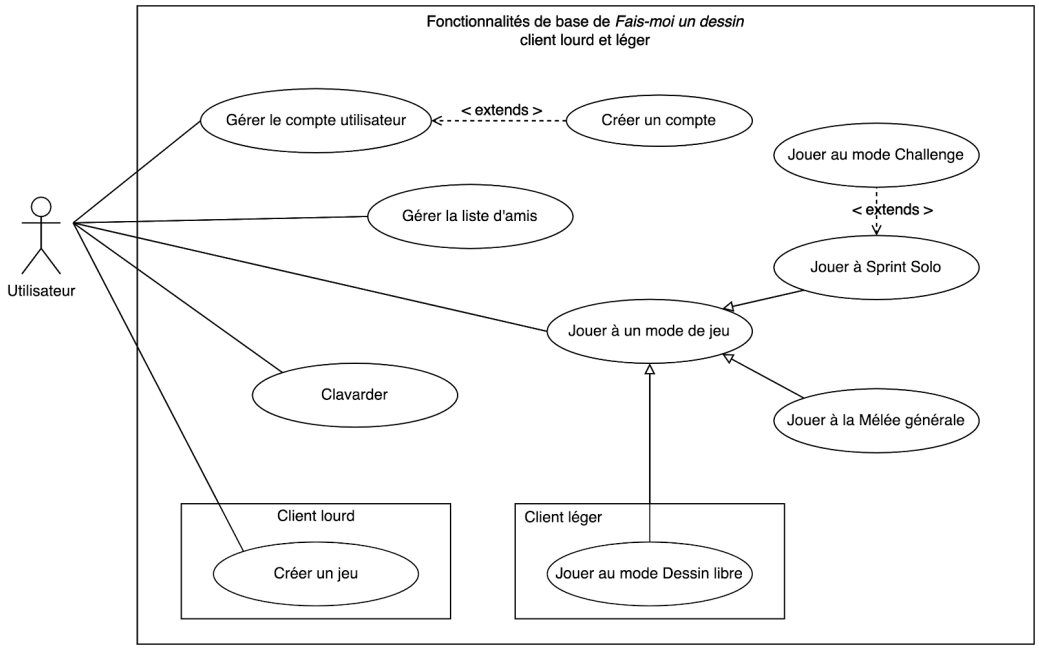

Figure 3.1 Diagramme de cas d'utilisation pour le comportement général du logiciel

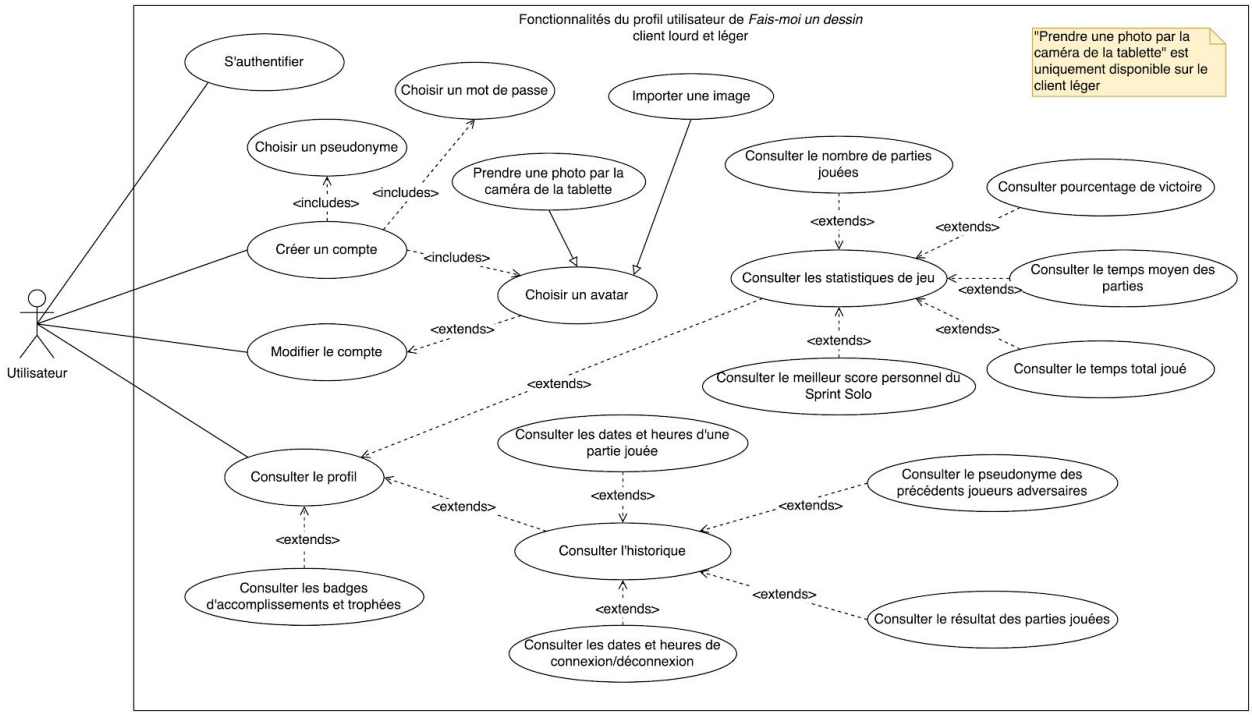

Figure 3.2 Diagramme de cas d'utilisation pour le comportement du profil utilisateur du logiciel

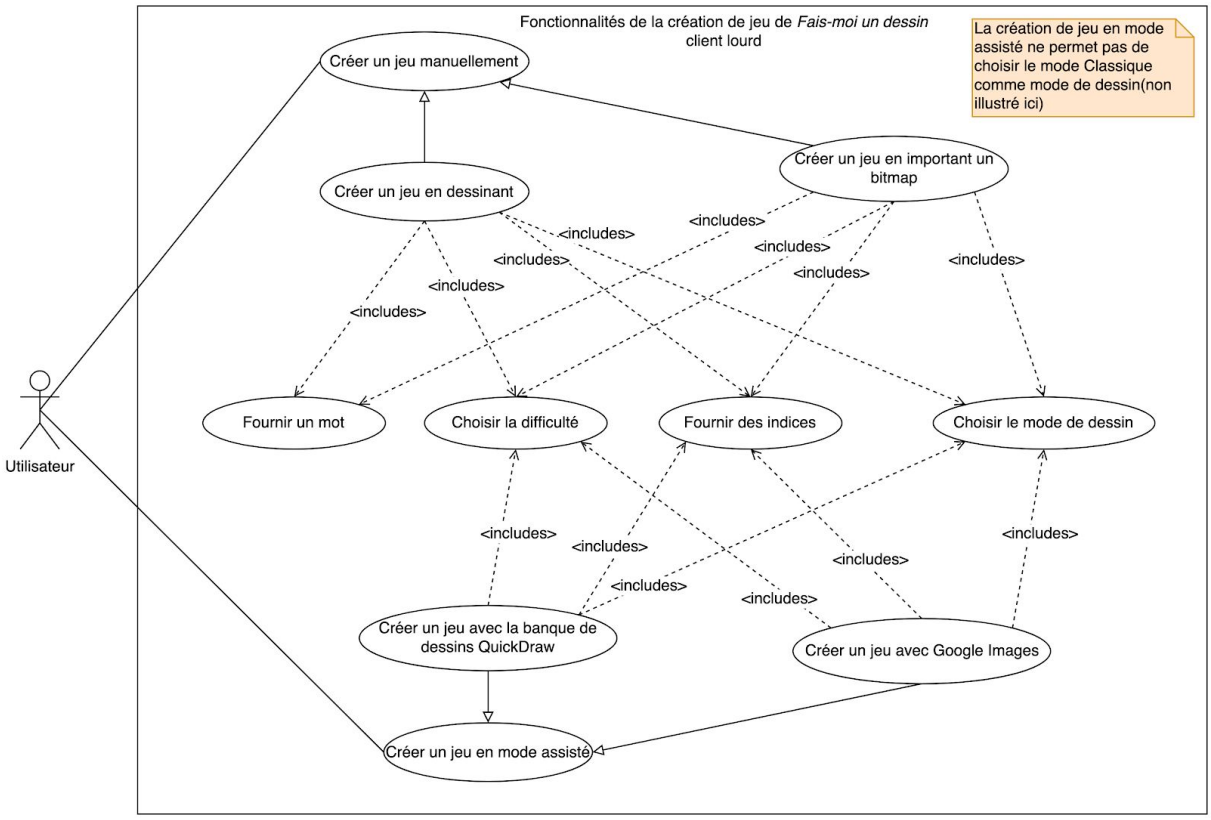

Figure 3.3 Diagramme de cas d'utilisation pour le comportement de création d'un jeu du logiciel

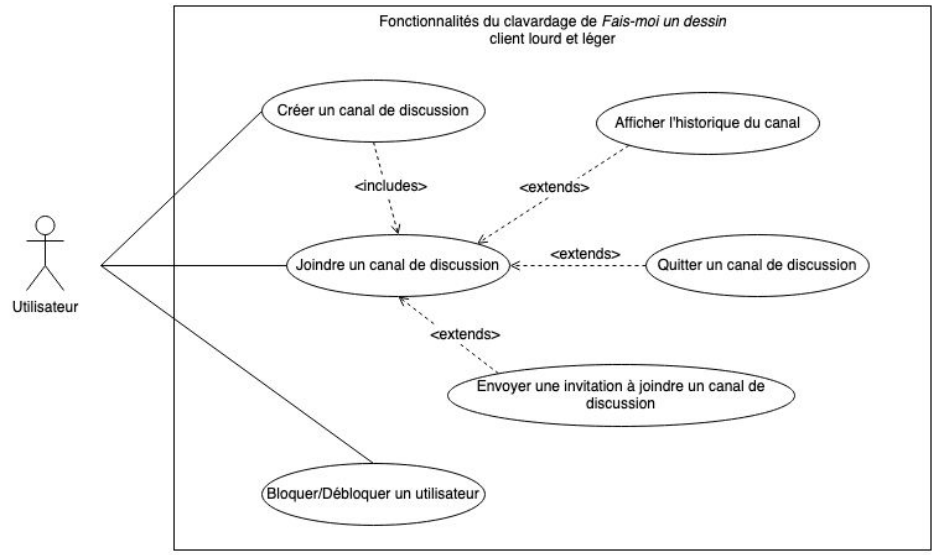

Figure 3.4 Diagramme de cas d'utilisation pour le comportement du clavardage du logiciel

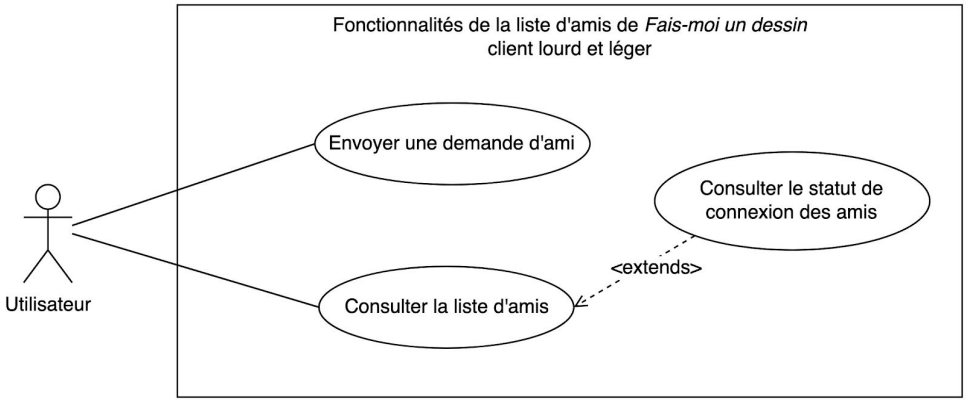

Figure 3.5 Diagramme de cas d'utilisation pour le comportement de la liste d'amis du logiciel

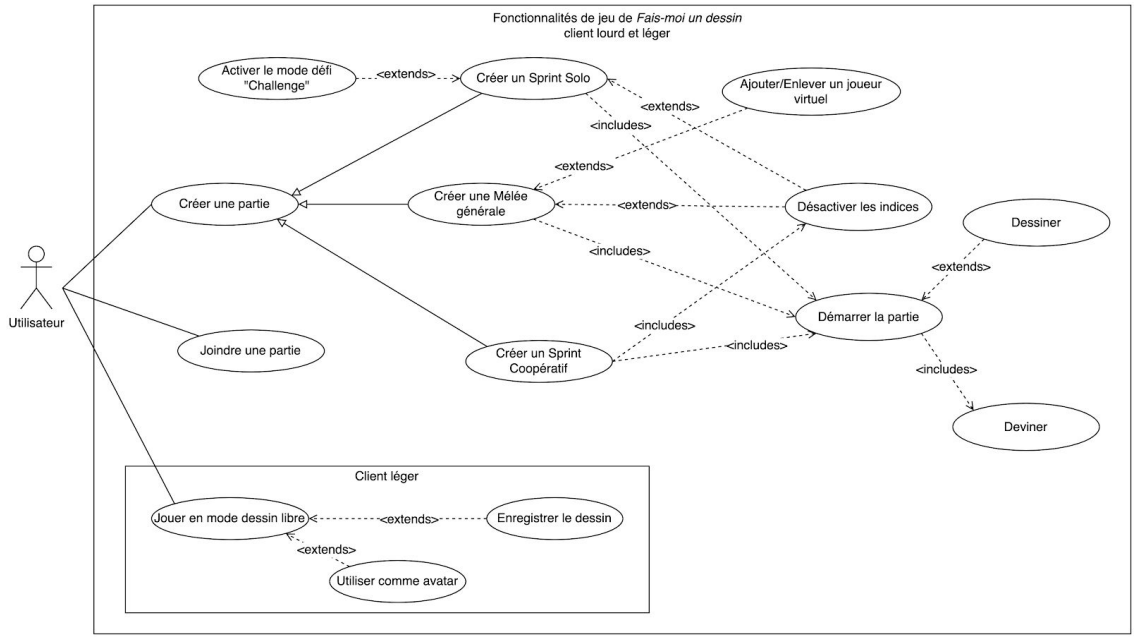

Figure 3.6 Diagramme de cas d'utilisation pour le comportement d'une partie ou d'un jeu du logiciel

4. Vue logique

4.1 Vue de haut niveau

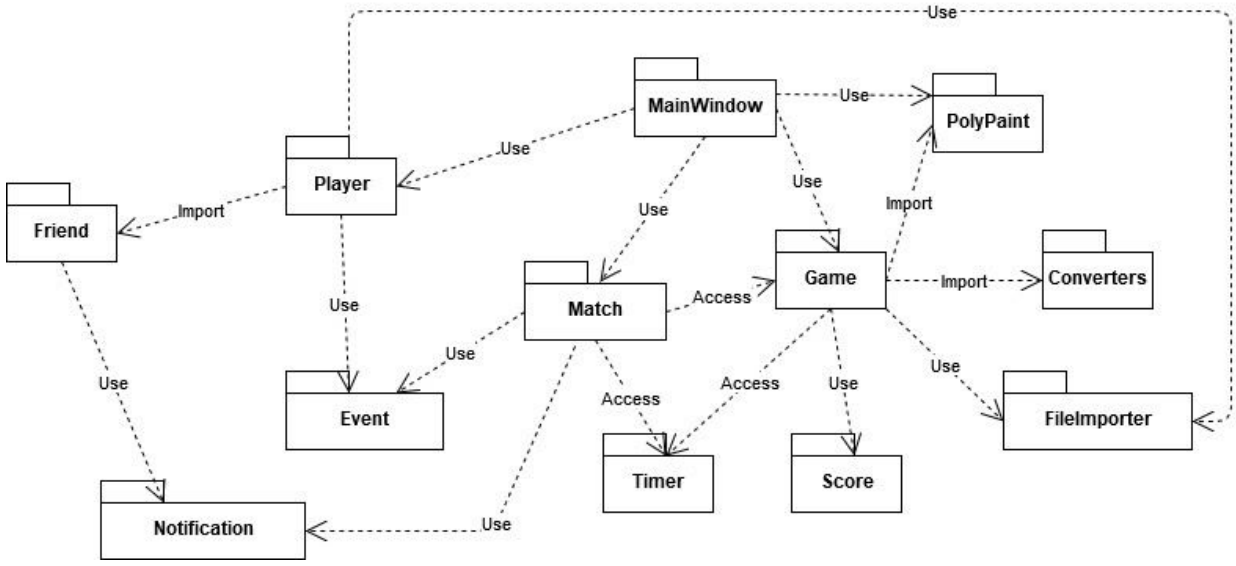

Figure 4.1: Diagramme de paquetage de haut niveau

Match
Ce paquetage a pour objectif de gérer tout ce qui concerne les parties. Il utilise le paquetage *Event* afin d'enregistrer les résultats obtenus. Il délègue au paquetage *Game* la responsabilité de gérer chaque jeu. Il utilise le paquetage *Notification* dans le cas qu'un défi est lancé. Il délègue au paquetage *Timer* la responsabilité de déterminer la fin de la partie en mode solo.
Match
Cette classe a pour responsabilité de conserver l'état d'une partie.
MatchFactory
Cette classe a pour responsabilité de créer une partie.
MatchManager
Cette classe a pour responsabilité de gérer le déroulement d'une partie (changement de rôles, changement de jeu, détermination de la fin de partie).
MatchFiller
Cette classe a pour responsabilité de s'assurer que la partie contient les joueurs requis et d'ajouter les joueurs virtuels s'il y a lieu.
MatchExporter
Cette classe a pour responsabilité d'enregistrer dans la base de données une partie.
MatchImporter
Cette classe a pour responsabilité de charger une partie à partir de la base de données.
ChallengeThrower
Cette classe a pour responsabilité de gérer le lancement d'un défi.

Tableau 4.1: Détails du paquetage Match

Game
Ce paquetage a pour objectif de gérer tout ce qui concerne les jeux. Il délègue au paquetage *Timer* la responsabilité de déterminer la fin d'un jeu. Il importe le paquetage *PolyPaint* afin de permettre au dessinateur de dessiner. Il importe le paquetage *Converters* pour s'assurer que les fichiers d'image soient du bon format et pour faire le pont avec *PolyPaint*. Il utilise le paquetage *FileUploader* pour permettre l'import de fichier non généré par *PolyPaint*. Il délègue aux paquetages *Score* et *Timer* la gestion du score et du temps respectivement.
Game
Cette classe a pour responsabilité de conserver l'état d'un jeu.
GameFactory
Cette classe a pour responsabilité de créer un jeu.

GameManager
Cette classe a pour responsabilité de gérer le déroulement d'un jeu.
GameExporter
Cette classe a pour responsabilité d'enregistrer dans la base de données un jeu.
GameImporter
Cette classe a pour responsabilité de charger un jeu à partir de la base de données.
AnswerValidator
Cette classe a pour responsabilité de traiter les réponses des joueurs.

Tableau 4.2: Détails du paquetage Game

Timer
Ce paquetage a pour objectif de suivre l'évolution du temps.
BasicTimer
Cette classe a pour responsabilité de permettre les fonctions de base (démarrer, arrêter, reset) du chronomètre.
SoloTimer
Cette classe a pour responsabilité de s'assurer que du temps est ajouté au chronomètre lorsque l'utilisateur donne une bonne réponse dans le mode solo.

Tableau 4.3: Détails du paquetage Timer

Score
Ce paquetage a pour objectif de calculer le score d'un jeu.
SoloScore
Cette classe permet de calculer le score lors d'un jeu en partie solo.
FreeForAllScore
Cette classe permet de calculer le score lors d'un jeu en partie mêlée générale.

Tableau 4.4: Détails du paquetage Score

Converters
Ce paquetage a pour objectif de convertir des éléments en un type supportée par le jeu.
ImageConverter
Cette classe a pour responsabilité de convertir des images (bmp, jpg) en SVG.
StrokeConverter
Cette classe a pour responsabilité de convertir les traits faits avec *PolyPaint* en Stroke.

Tableau 4.5: Détails du paquetage Converters

FileImporter
Ce paquetage a pour objectif de permettre l'accès à des fichiers situés sur l'ordinateur et QuickDraw.
LocalFileImporter
Cette classe a pour responsabilité de permettre l'accès aux images sur l'ordinateur.
QuickDrawImporter
Cette classe a pour responsabilité de permettre l'accès aux images de QuickDraw.

Tableau 4.6: Détails du paquetage FileImporter

Player
Ce paquetage a pour objectif de gérer tout ce qui a trait au joueur. Il utilise le paquetage *Event* afin de mémoriser ses connections. Il délègue sa gestion de la liste d'amis au paquetage *Friend*.
Authentification
Cette classe a pour responsabilité d'authentifier le joueur.
Encryption
Cette classe a pour responsabilité d'encrypter les mots de passe.
Player
Cette classe a pour responsabilité de conserver l'état du joueur.
PlayerFactory
Cette classe a pour responsabilité de créer un joueur.

Tableau 4.7: Détails du paquetage Player

Friend
Ce paquetage a pour objectif de gérer la liste d'amis. Il utilise le paquetage *Notification* pour envoyer la demande d'amitié.
FriendManager
Cette classe a pour responsabilité de permettre l'ajout et le retrait d'un ami.
FriendStatus
Cette classe a pour responsabilité de déterminer le statut de connection d'un ami.

Tableau 4.8: Détails du paquetage Friend

Event
Ce paquetage a pour objectif d'enregistrer les événements d'intérêt sur la base de données.
ConnectionEvent
Cette classe a pour responsabilité d'enregistrer sur la base de données les connections et déconnections des joueurs.
MatchEvent
Cette classe a pour responsabilité d'enregistrer le résultat d'une partie.

Tableau 4.9: Détails du paquetage Event

Notification
Ce paquetage a pour objectif de transmettre des notifications d'un joueur ou du serveur à un autre joueur.
FriendRequest
Cette classe a pour responsabilité de notifier un joueur d'une demande d'ami.
ChallengeRequest
Cette classe a pour responsabilité de notifier un joueur d'un défi à relever.

Tableau 4.10: Détails du paquetage Notification

4.2 Vue détaillée de l'interface usager

La patron de conception Modèle - Vue - Présentation est utilisé pour tous les paquetage. Ceux-ci ne seront pas présentés en détail afin d'alléger le document.

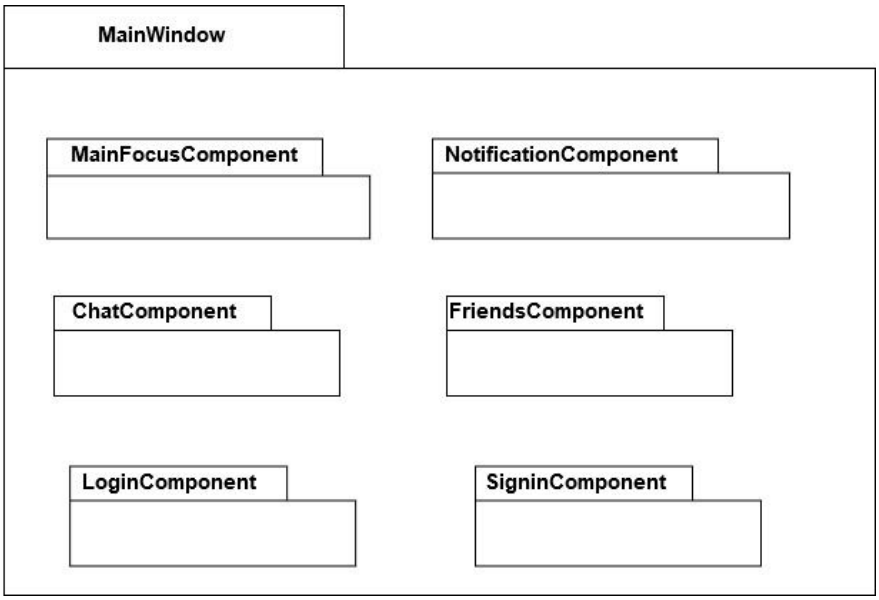

Figure 4.2: Représentation du paquetage MainWindow

MainWindow
Ce paquetage constitue la base de la vue de l'application.
MainFocusComponent
Ce paquetage a pour objectif d'afficher l'activité principale du joueur.
NotificationComponent
Ce paquetage a pour objectif d'afficher les notifications adressées au joueur.
ChatComponent
Ce paquetage a pour objectif d'afficher le clavardage.
FriendsComponent
Ce paquetage a pour objectif d'afficher les amis du joueur.
LoginComponent
Ce paquetage a pour objectif d'afficher la page d'authentification.
SigninComponent
Ce paquetage a pour objectif d'afficher la page d'inscription.

Tableau 4.11: Détail du paquetage *MainWindow*

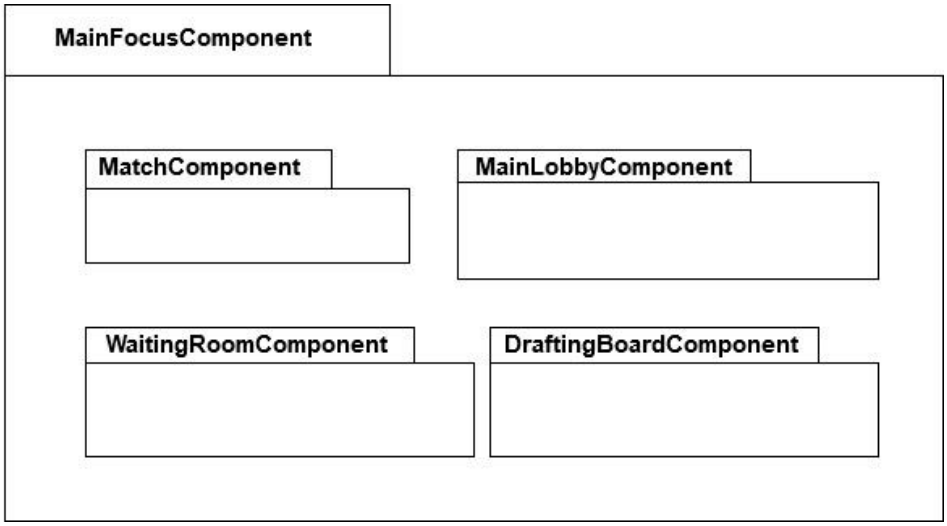

Figure 4.3: Représentation du paquetage *MainFocusComponent*

MainWindow
Ce paquetage constitue la base de la vue de l'application.
MatchComponent
Ce paquetage a pour objectif d'afficher la partie en cours.
MainLobbyComponent
Ce paquetage a pour objectif d'afficher le menu principal.
WaitingRoomComponent
Ce paquetage a pour objectif d'afficher la salle d'attente d'un jeu.
DraftingBoardComponent
Ce paquetage a pour objectif d'afficher l'application *PolyPaint* en mode création ou en mode jeu.

Tableau 4.12: Détail du paquetage *MainFocusComponent*

5. Vue des processus

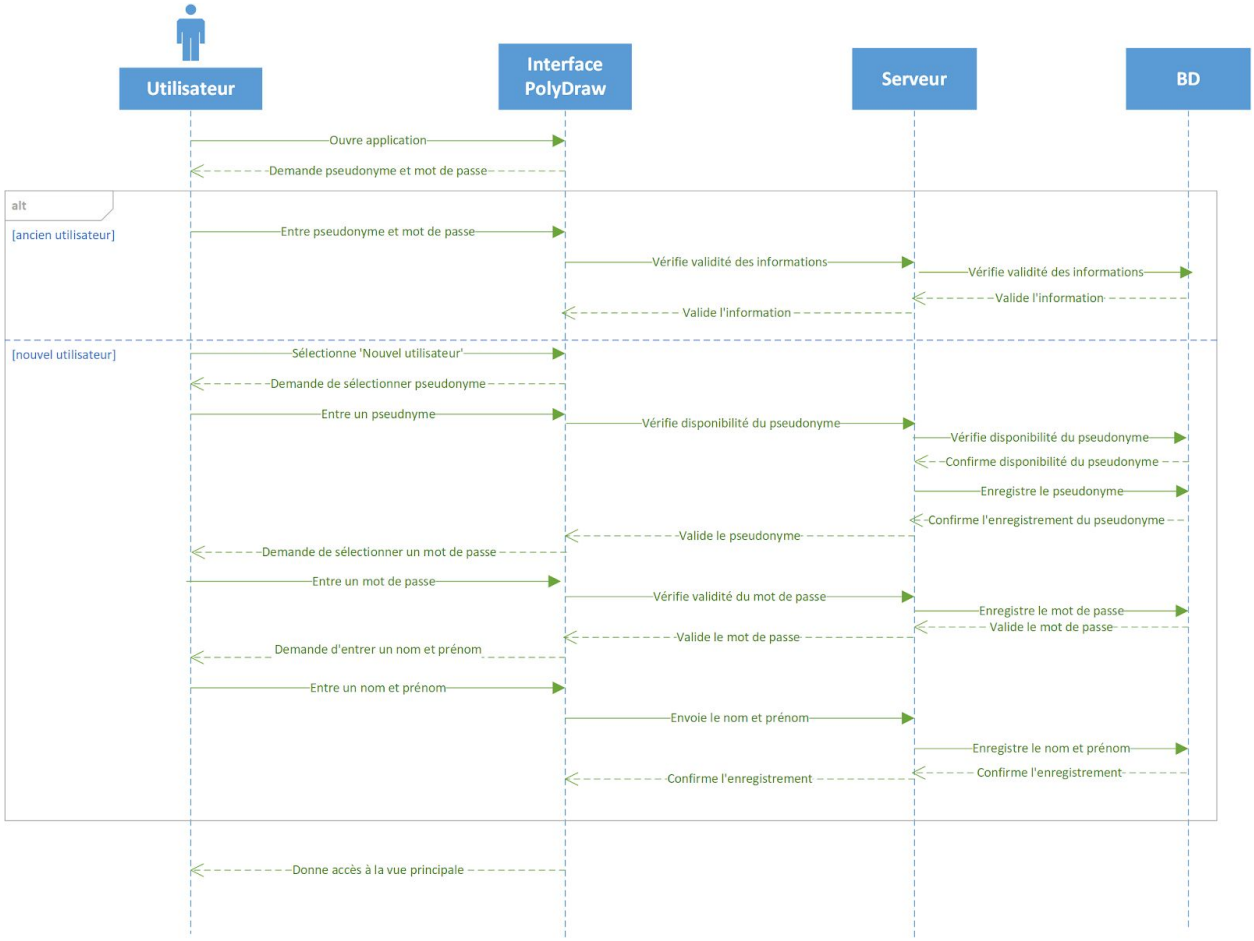

Figure 5.1 Diagramme de séquence pour l'authentification au système

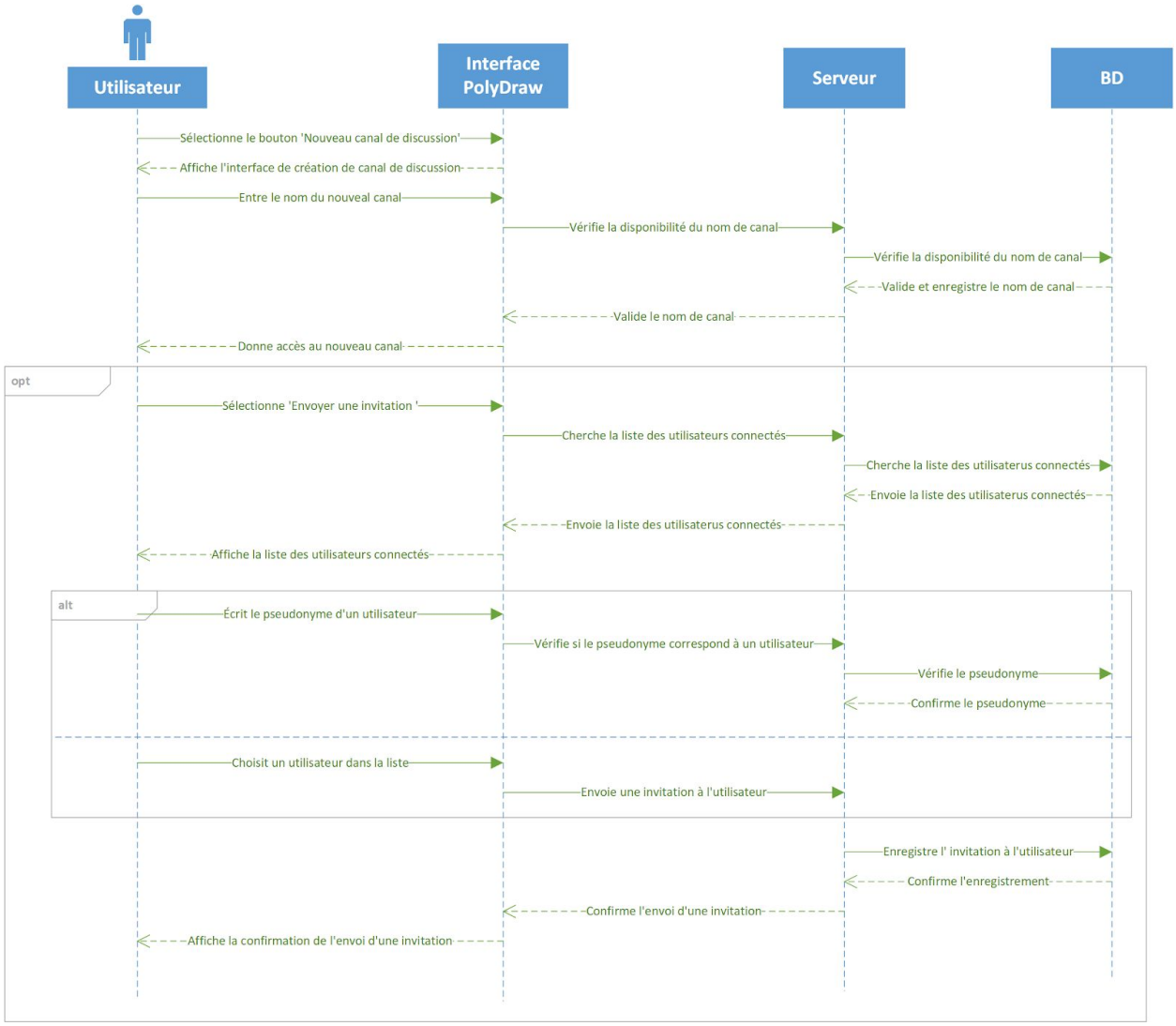

Figure 5.2 Diagramme de séquence pour la création d'un canal de discussion

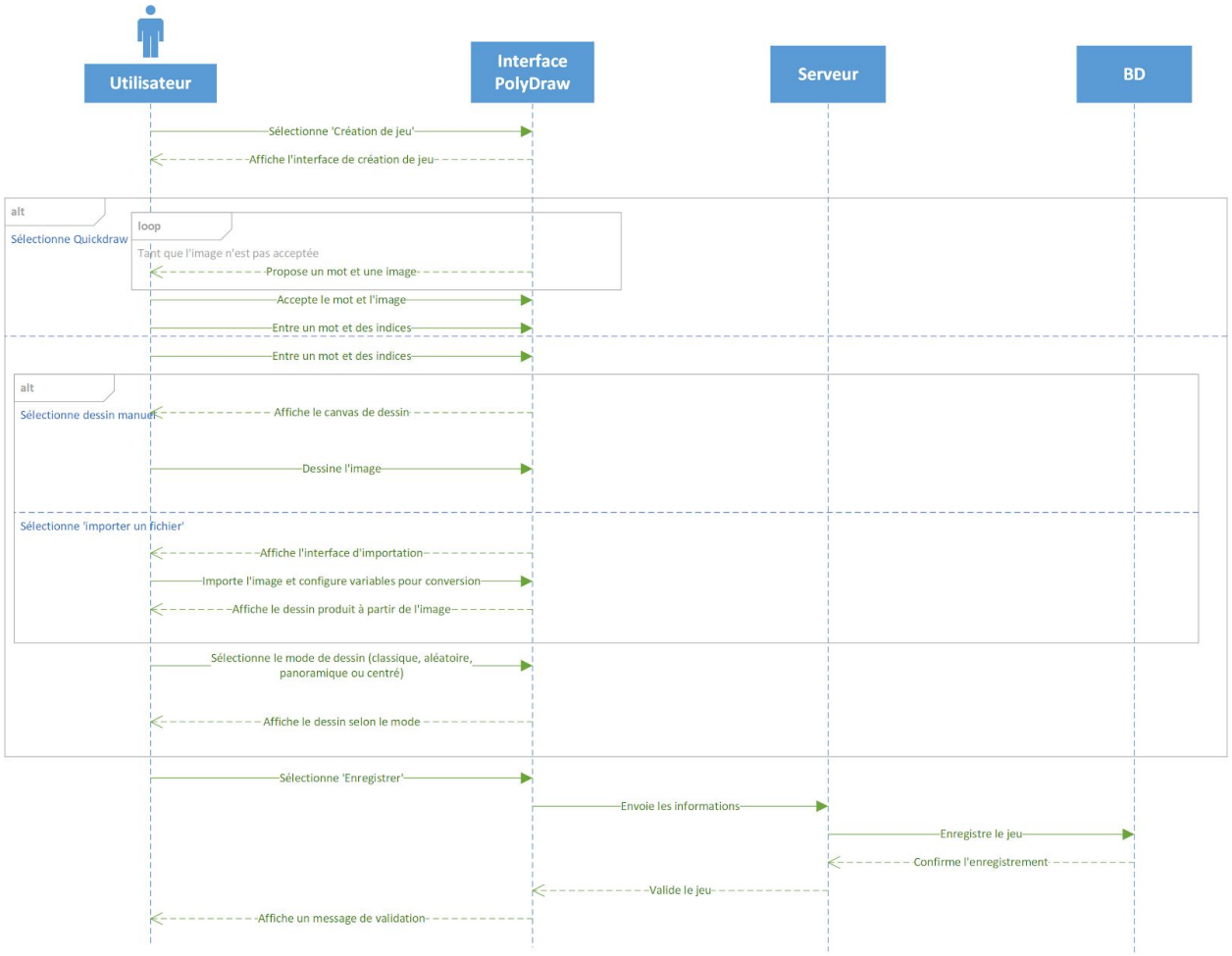

Figure 5.3 Diagramme de séquence pour la création d'un jeu

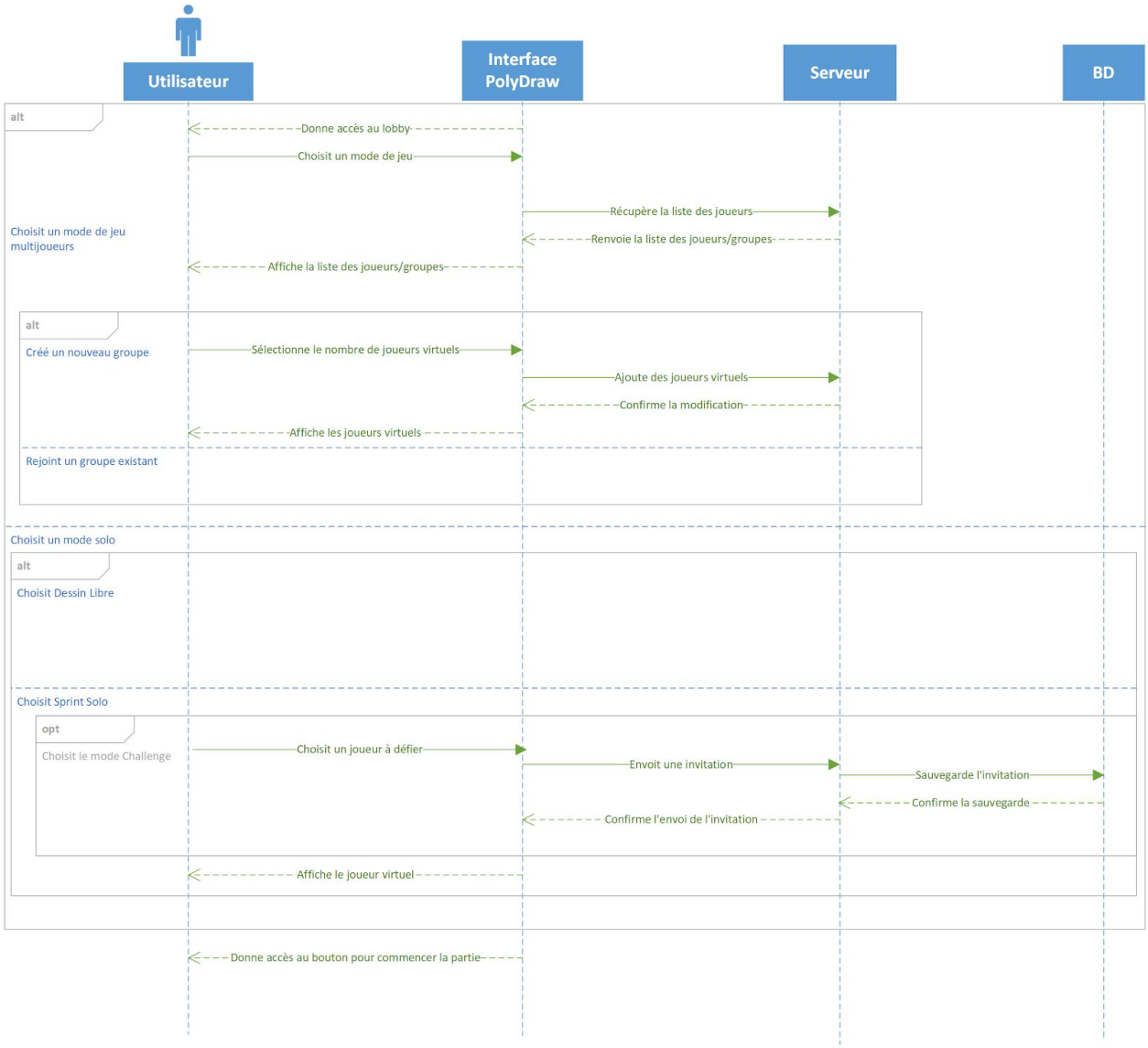

Figure 5.4 Diagramme de séquence pour le lancement d'une nouvelle partie

6. Vue de déploiement

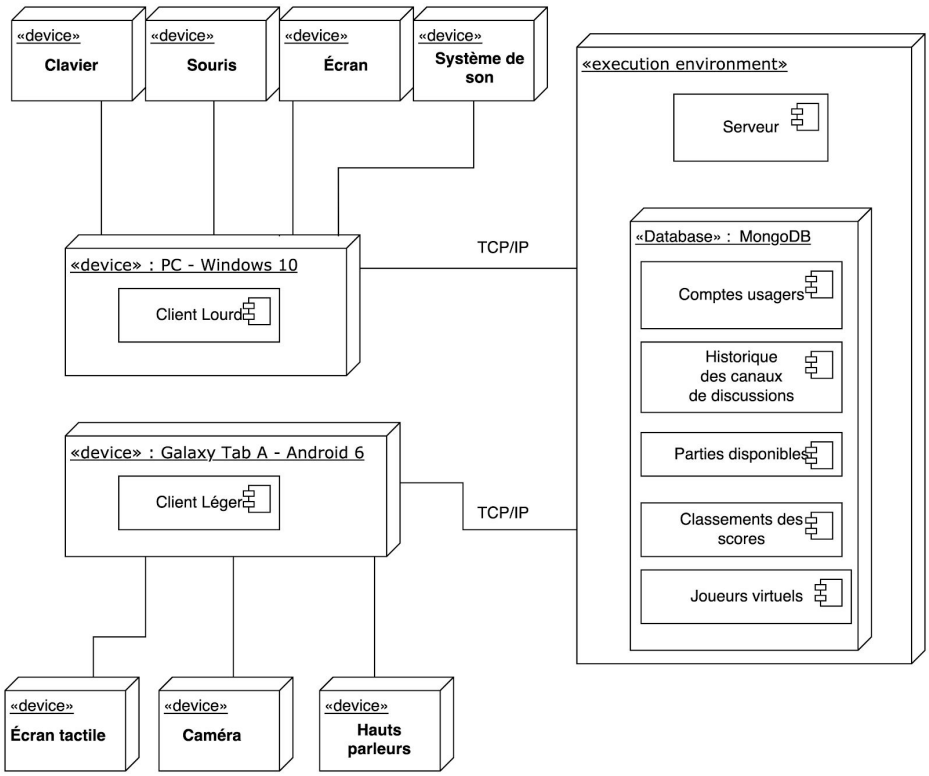

Figure 6.1 Diagramme de déploiement

7. Taille et performance

Plusieurs caractéristiques de taille et de performance peuvent impacter l'architecture et le design du logiciel. En effet, les deux types de clients posséderont de légères différences puisque le client lourd doit être en mesure de créer un jeu. Cette fonctionnalité n'est pas présente sur le client léger. Ceci pourrait donc avoir un impact sur la performance lorsque l'utilisateur va téléverser une image au serveur. Cependant, cela ne devrait pas avoir un impact sur la performance du logiciel lorsque l'utilisateur joue une partie. L'expérience utilisateur ne devrait pas être affectée lorsque l'utilisateur joue une partie soit sur le client lourd soit sur le client léger. Aucun délai notable devrait être constaté lors de la partie jouée. Plus précisément, le délai devrait être sous les 200 millisecondes. Aucun délai notable devrait être constaté dans la messagerie lorsque l'utilisateur envoie un message. Le délai devrait être sous les 500 millisecondes. Le temps d'authentification devrait être au maximum de 2 secondes.

De plus, les informations stockées sur la base de données ne devraient pas être trop volumineuses et seront appelées que lorsque l'utilisateur désir voir son profil. L'application ne devrait pas utiliser plus de 200 Mo pour le client léger une fois installée sur la tablette. Aussi, sur le client léger, il y aura la possibilité de télécharger un dessin dans le mode "dessin libre". Le fichier ne devrait pas dépasser la taille de 2 Mo.