PFA

Spécification des exigences logicielles

Informatique Deuxième Année

Partage d'écran en réseau "local"

Élèves :
Antoine Pringalle (apringalle@enseirb-matmeca.fr)
Benjamin Upton (bupton@enseirb-matmeca.fr)
Jalal Izekki (jizekki@enseirb-matmeca.fr)
Saad Margoum (smargoum@enseirb-matmeca.fr)
Souhail Nadir (snadir@enseirb-matmeca.fr)
Saad Zoubairi (szoubairi@enseirb-matmeca.fr)
Théo Lelasseux (tlelasseux@enseirb-matmeca.fr)
Zaid Zerrad (zzerrad@enseirb-matmeca.fr)

Enseignant :
David Renault
(renault@labri.fr)

18 / 01 / 2021

1 Introduction et motivation

L'idée de ce projet consiste à utiliser la transmission de flux vidéo pour mettre en place un système de partage d'écran sur un réseau local, comme par exemple sur le réseau de l'ENSEIRB-Matmeca. Un tel système permettrait de discuter facilement entre tandems d'une équipe, ou entre personnes d'un groupe de TD. L'une des applications possibles consisterait à ce qu'un enseignant puisse consulter facilement un ensemble d'écrans correspondant à la liste des personnes présentes dans une salle de TD.

Puisque la plupart des cours ont lieu désormais à distance à cause de la crise sanitaire, le projet proposé a évolué d'un outil qui ne marche qu'en réseau local, en un outil qui marche sur internet. De plus, il ne doit nécessiter que très peu de débit internet pour fonctionner, afin que les TDs puissent être suivis par tous, et ce même avec une mauvaise connexion internet.

1.1 Objectif et portée du document

L'objectif du document de spécification d'exigences logicielles est de décrire le comportement du système par une vision globale de ce dernier d'abord, et ensuite détailler les différentes exigences fonctionnelles. Ce document décrit aussi les exigences non fonctionnelles et les facteurs supplémentaires nécessaires à la description complète et compréhensible des besoins pour le projet.

1.2 Définitions, acronymes et abréviations

Dans la suite du document, le professeur désigne l'organisateur d'une réunion virtuelle. C'est celui qui programme et lance la réunion. L'élève ou l'étudiant sont les participants de la réunion, ils possèdent à priori moins de droits de modération sur la réunion dans un premier temps. Le terme utilisateur regroupe les deux entités précédentes, ce sont toutes les personnes susceptibles de se servir de cet outil.

1.3 Perspectives du projet

Ce projet s'inscrit dans un cadre pédagogique permettant de faciliter l'organisation de TDs, et qui permettra aussi de palier à plusieurs difficultés qui se posent notamment avec l'enseignement à distance. L'outil sujet du projet permettra à un groupe (d'étudiants et d'un professeur) de partager leur écran et leur son éventuellement pendant une séance de TD.

Cet outil diffère des outils déjà présents par un paramétrage plus affiné sur la qualité de sortie vidéo notamment. En effet, pour un partage de code, il n'est pas nécessaire de fournir un flux vidéo de 30 fps, 1 fps ou moins sont suffisants. Ce paramétrage a pour but de limiter le transfert de données coûteuses en terme de débit internet.

Nous ne savons pas encore s'il est possible de changer ces paramètres pendant la conférence ou s'il doivent être définis avant l'appel. Ce projet a donc pour but de tester ce qui est possible ou non de faire sur un outil de communication.

2 Description générale

2.1 Fonctions du produit

L'objectif du projet consisterait à mettre en place un serveur permettant de faire dialoguer et partager des écrans facilement à des groupes de personnes. Ainsi, on s'intéresse à donner la possibilité de passer des appels entre encadrants de TP et élèves, dans lesquels tout le monde peut partager son écran. Un des objectifs principaux est de réaliser ce partage à moindre coût réseau.

2.2 Caractéristiques des utilisateurs

Les utilisateurs seront des étudiants et des professeurs de l'ENSEIRB-Matmeca.

3 Contraintes de conception

3.1 Être programmé en Ruby

Nous devons faciliter l'adaptation de notre application avec la forge de l'école. Ainsi, le serveur doit être codé en Ruby.

3.2 Utiliser du WebRTC

4 Exigences spécifiques

4.1 Fonctionnalités

4.1.1 Partager son écran

- **Partager à une seule personne ou à un groupe.** Notre outil doit permettre à un étudiant de partager son écran à un groupe, qui serait une salle de classe virtuelle. Un étudiant doit pouvoir choisir entre partager son écran à une salle ou bien seulement à une personne lors d'un appel privé.
- **Pouvoir faire des sous-groupes.** Dans notre utilisation, il serait intéressant de pouvoir diviser une salle de classe en plusieurs petits groupes pour des travaux de groupe en TD. Notre salle doit donc pouvoir se subdiviser et chaque étudiant doit pouvoir joindre la salle qu'il veut.

4.1.2 Paramétrer l'appel

- **Paramétrer les FPS.** C'est l'objectif premier de ce travail, nous devons pouvoir régler la fréquence des images envoyées et reçues pour que l'appel soit le plus économe en données échangées si besoin (si un étudiant a une connexion faible, il doit pouvoir suivre une conférence et partager lui aussi son écran).
- **Paramétrer l'écran partagé.** Nous devons permettre à l'utilisateur d'activer ou déactiver son partage d'écran. Le partage d'écran est initialement désactivé pour l'utilisateur, c'est à lui de l'activer et choisir les fenêtres à partager.
- **Paramétrer les micros.** Nous devons permettre à l'utilisateur d'activer ou déactiver son micro.
- **Paramétrer l'interface** (nombre d'écrans, agencement, choisir quel écran on regarde...). L'utilisateur doit pouvoir sélectionner l'utilisateur dont il souhaite afficher l'écran. Il peut

aussi choisir d'afficher une grille de tous les écrans. Initialement, les écrans qui s'affichent en premier sont de ceux qui sont en train de parler.

— **Réduction de bruit** (noise cancellation). Un paramètre qui réduirait le bruit ambiant autour d'un étudiant pour lui permettre de travailler avec d'autres sans les gêner (ou sans qu'il ait à se mute à chaque fois qu'il ne parle pas).

— **Choisir ses périphériques d'entrée et de sortie pendant l'appel** (pas besoin de déco-reco). L'utilisateur doit pouvoir choisir son entrée audio (micro), son entrée vidéo (quel écran ou quelle fenêtre) et sa sortie audio (haut parleur, écouteurs ou bien casque) sans avoir à quitter ou actualiser la page.

4.1.3 Messagerie

— **Envoyer des messages au groupe de la réunion ou en privé.** Autrement que de parler vocalement, les utilisateurs doivent pouvoir échanger par message, soit sur une conversation de la salle, soit en privé à un autre utilisateur.

— **Pouvoir mentionner d'autres utilisateurs.** Il serait intéressant pour les étudiants de pouvoir mentionner (*@Équipe1* ou *@Paul Dupont*) d'autres étudiants pour leur dire de rejoindre leur sous-groupe vocal dans le cadre d'un TD ou bien pour faire venir le professeur afin qu'il réponde à une question.

— **Réagir aux messages.** Cette fonctionnalité permettrait de rendre un chat écrit plus vivant en autorisant les participants à réagir à un message posté par un emoji.

— **Répondre à un message.** Il est également très utile de pouvoir répondre à un message pour clarifier une conversation avec plusieurs interlocuteurs.

— **Pouvoir envoyer des messages en Markdown.** Avoir la possibilité d'envoyer des messages qui commençant par un une suite de symboles qui reste à définir pourrait permettre d'activer un affichage Markdown sur le chat vocal.

— **Possibilité d'envoyer des commandes dans la conversation pour effectuer certaines actions.** Certaines de ces commandes nécessiteront des droits de modérateur pour être exécutées. Parmi les commandes possibles :
  — Création d'un sondage rapide (Ex. `/poll A B C D`).
  — Envoi d'un message à un sous-groupe en particulier (Ex. `/msg @equipe1 "Revenez dans le salon principal"`).
  — Couper le micro de tous les étudiants (Ex. `/muteall`).

— **Envoi d'images.** Pouvoir copier/coller des images dans la messagerie, ou en importer depuis son ordinateur peut permettre aux étudiants de partager des captures d'écran. Cette option devrait être modérée pour limiter l'envoi de fichiers trop volumineux.

— **Envoi de fichiers.** Plutôt que d'envoyer le code avec un copier coller, on devrait pouvoir directement joindre des fichiers à la classe, au groupe ou au sous-groupe. Cette option devrait être modérée pour limiter l'envoi de fichiers trop volumineux.

4.1.4 Afficher des statistiques

— **Montrer les fps.** Il est intéressant de voir à quelle fréquence d'images le partage d'écran est fait au cours d'un appel. Cette option peut être affichée dans un des coins de l'écran et pourquoi pas désactivable dans un menu si elle ne nous intéresse pas.

— **Afficher les noms des utilisateurs.** Il est intéressant de voir la liste des utilisateurs

présents. Ainsi, afficher les identifiants des utilisateurs permettrait de faciliter la discussion.

— **Montrer le nombre de gens connectés.** Dans une salle de classe il est important de savoir rapidement pour l'utilisateur combien d'élèves sont connectés et possiblement déconnectés. L'affichage de ce paramètre peut être disponible pour tous cependant.

— **Montrer la qualité de connexion de chaque participant.** Cette statistique permet de voir quels sont les élèves qui ont une connexion internet instable, et de régler le paramètre des fps afin de permettre à tous de suivre le cours convenablement.

— **Montrer qui est l'encadrant / administrateur du TP.** Cette distinction peut se faire par un petit symbole à côté de son nom ou bien un cadre d'une couleur particulière autour de son écran. Cela faciliterait la recherche de cet encadrant pour sélectionner rapidement son écran quand il souhaite expliquer une notion utile au cours.

4.2 Product backlog

Un **Product backlog** est une liste des fonctionnalités priorisées qui sont à développer ou améliorer dans le cadre d'un projet informatique. Notre product backlog est représenté dans cette partie par une liste de user-stories organisées de la plus importante vers la moins importante.

— En tant qu'utilisateur, je veux partager mon écran afin de pouvoir expliquer certaines choses aux autres utilisateurs.

— En tant qu'utilisateur, je veux échanger par audio afin de faciliter la communication.

— En tant qu'utilisateur, je veux pouvoir envoyer des messages dans la conversation publique/privée afin de prendre des notes et communiquer en cas de présence de problèmes de micro.

— En tant qu'utilisateur, je veux pouvoir augmenter/diminuer mes fps quand je veux lors du partage d'écran, afin d'économiser la consommation des ressources.

— En tant qu'utilisateur, je veux pouvoir activer et désactiver le partage d'écran quand je veux, afin de mieux gérer la session. En tant que prof, je veux pouvoir activer et désactiver le micro de n'importe qui quand je veux, afin de mieux gérer les droits.

— En tant qu'utilisateur, je veux pouvoir voir les statistiques (fps, débit de connexion, informations sur la session) de chaque personne sur l'interface. Et ce, afin de détecter les problèmes de connexions.

— En tant qu'utilisateur, je veux pouvoir disposer d'une fonctionnalité de réduction de bruit, afin de filtrer les bruits indésirables.

— En tant qu'utilisateur, je veux pouvoir paramétrer l'interface (choisir quel écran regarder ou les afficher tous en grille). Et ce, afin de mieux suivre les explications.

— En tant qu'utilisateur, je veux pouvoir formater mes messages sous forme Markdown, afin de rendre la conversation plus lisible et simple à parcourir.

— En tant qu'utilisateur, je veux pouvoir réagir aux messages.

— En tant qu'utilisateur, je veux pouvoir envoyer des fichiers ou des images dans le chat, afin d'échanger des fichiers ou des images du code.

— En tant que professeur, je veux pouvoir créer des sous-groupes, afin d'organiser un travail de groupe.

4.3 Spécification des cas d'utilisation

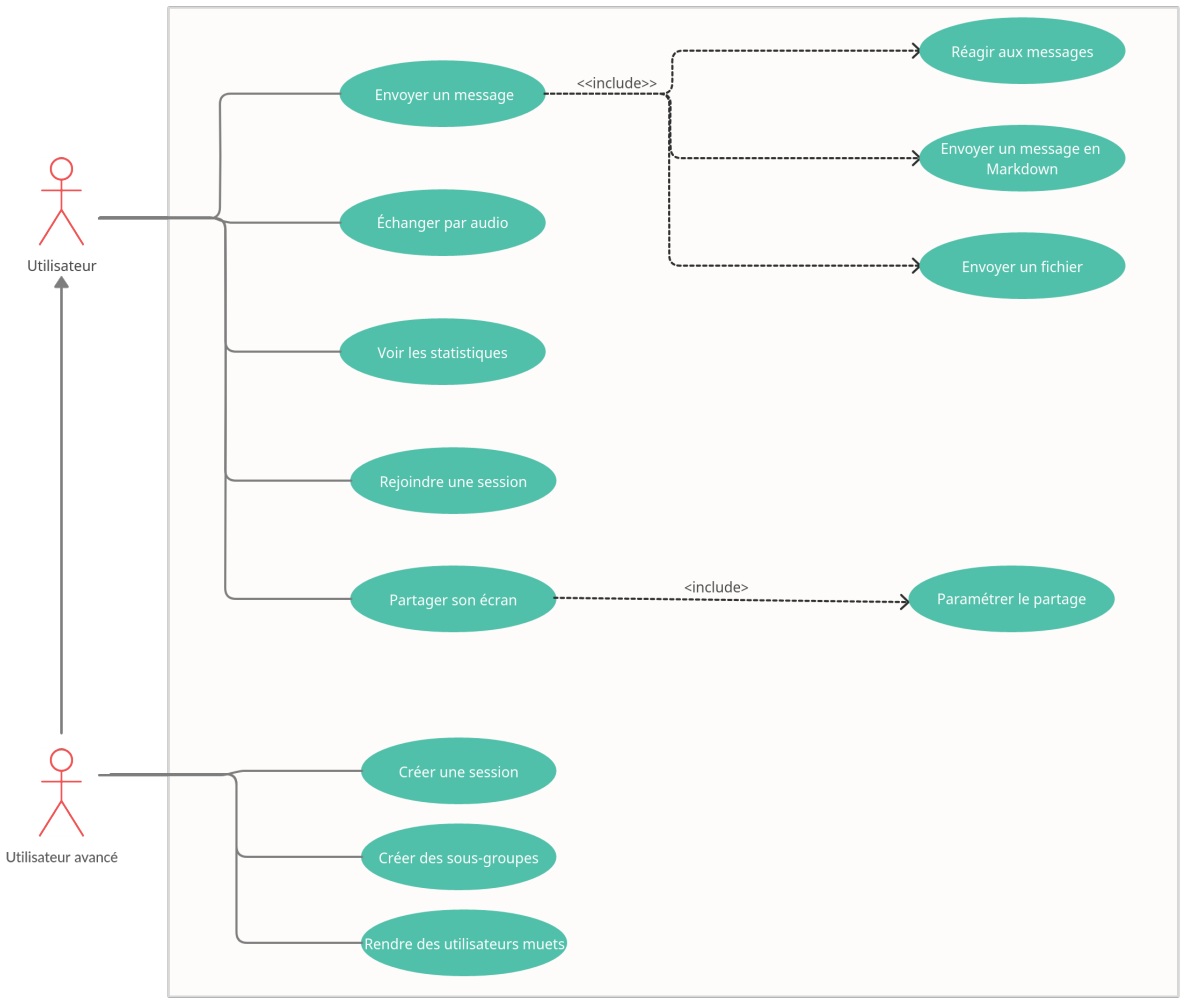

FIGURE 1 – Diagramme des cas d'utilisation

4.4 Exigences supplémentaires

4.4.1 Fonctionnel avec une connexion faible

L'application doit permettre à des étudiants ayant un débit de connexion très limité de suivre un cours. De plus, si l'application tourne sur les serveurs de l'école, il est intéressant qu'elle soit la moins gourmande possible. Ainsi, tout doit fonctionner avec la connexion la plus faible possible.

4.4.2 Capacité

Nous voulons assurer des conférences avec un nombre maximale de personnes dans la mesure du possible. Et ensuite voir quelles sont les réelles limites et où elles vont se trouver (du côté utilisateur, du côté serveur ou bien directement les applications WebRTC).

5 Architecture

Un client principal et les autres en étoile autour de lui, ils communiquent par messages. Il y a deux types de messages : ceux que les utilisateurs écrivent pour communiquer dans le chat et

ceux que les clients s'envoient pour que l'appel se passe bien et qui sont décrits en figure 2.

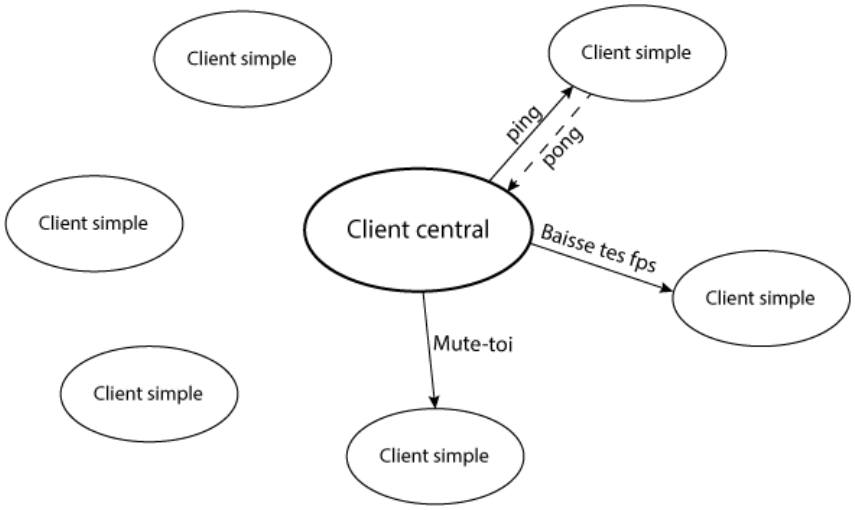

FIGURE 2 – Architecture des communications entre les clients

6 Plan de tests de validation

Scénario de test 1 : Vérifier la fonctionnalité audio

— Vérifier le bon fonctionnement de l'appel si un des utilisateurs ne possède pas de micro. L'utilisateur "Utilisateur sans micro" lance un appel et le message "micro non détecté" s'affiche sur son écran. L'appel se déroule normalement pour les autres utilisateurs possédant un micro.

— Vérifier que lors d'un appel chaque interlocuteur peut parler et entendre. "Utilisateur 1" et "Utilisateur 2" lancent un appel et communiquent en audio.

— Vérifier que le son n'est pas entendu si le micro de l'interlocuteur est coupé. "Utilisateur muet" lance un appel et coupe son micro. Les autres utilisateurs n'entendent pas "Utilisateur muet".

Scénario de test 2 : Vérifier la fonctionnalité chat

— Vérifier qu'un message envoyé par un utilisateur est bien reçu par les autres utilisateurs dans l'appel. "Utilisateur du chat" envoie le message "hello" sur le chat d'un appel et tous les autres utilisateurs reçoivent ce message.

— Vérifier que l'envoi des images, fichiers et textes avec caractères spéciaux se fait correctement. "Utilisateur du chat" envoie une image et le message "HLlÕ " sur le chat ainsi qu'un fichier texte "test.txt" vide. On vérifie que tous les éléments envoyés sont bien reçus.

Scénario de test 3 : Vérifier la fonctionnalité vidéo

— Vérifier que le partage d'écran concerne bien la fenêtre choisie par l'utilisateur. L'utilisateur "Screenshare" partage un terminal. Le terminal partagé et uniquement ce terminal est bien visible par tous les utilisateurs de l'appel.

— Vérifier que le paramètrage du fps est fonctionnel. L'utilisateur "Screenshare" choisit le paramètre 10 pour le fps de son partage d'écran. On vérifie que le fps est bien de 10 en sortie.

— Vérifier que la fenêtre partagée peut être visionnée par tous les autres utilisateurs dans l'appel ayant le droit. On ajoute les utilisateurs "Utilisateur avec droit" et "Utilisateur sans droit" à l'appel. On vérifie que seul "Utilisateur avec droit" peut visionner l'écran

partagé par l'utilisateur "screenshare".

N°	Action	Attendu
1	Lancer un appel	L'appel se lance
2	Partager son écran	L'interlocuteur voit notre écran
3	L'interlocuteur partage son écran	On voit son écran
4	Parler	L'interlocuteur nous entend
5	L'interlocuteur parle	On l'entend
6	On écrit un message dans le chat	L'interlocuteur le reçoit
7	L'interlocuteur écrit un message dans le chat	On le reçoit
8	Envoyer un fichier dans le chat	L'interlocuteur le reçoit
9	Envoyer une image dans le chat	L'interlocuteur la reçoit
10	Rejoindre un sous-groupe	sous-groupe rejoint
11	Modifier la valeur du fps	fps modifié
12	Choisir la fenêtre à partager	L'interlocuteur voit la fenêtre choisie

7 Planning prévisionnel : Diagramme de Gantt

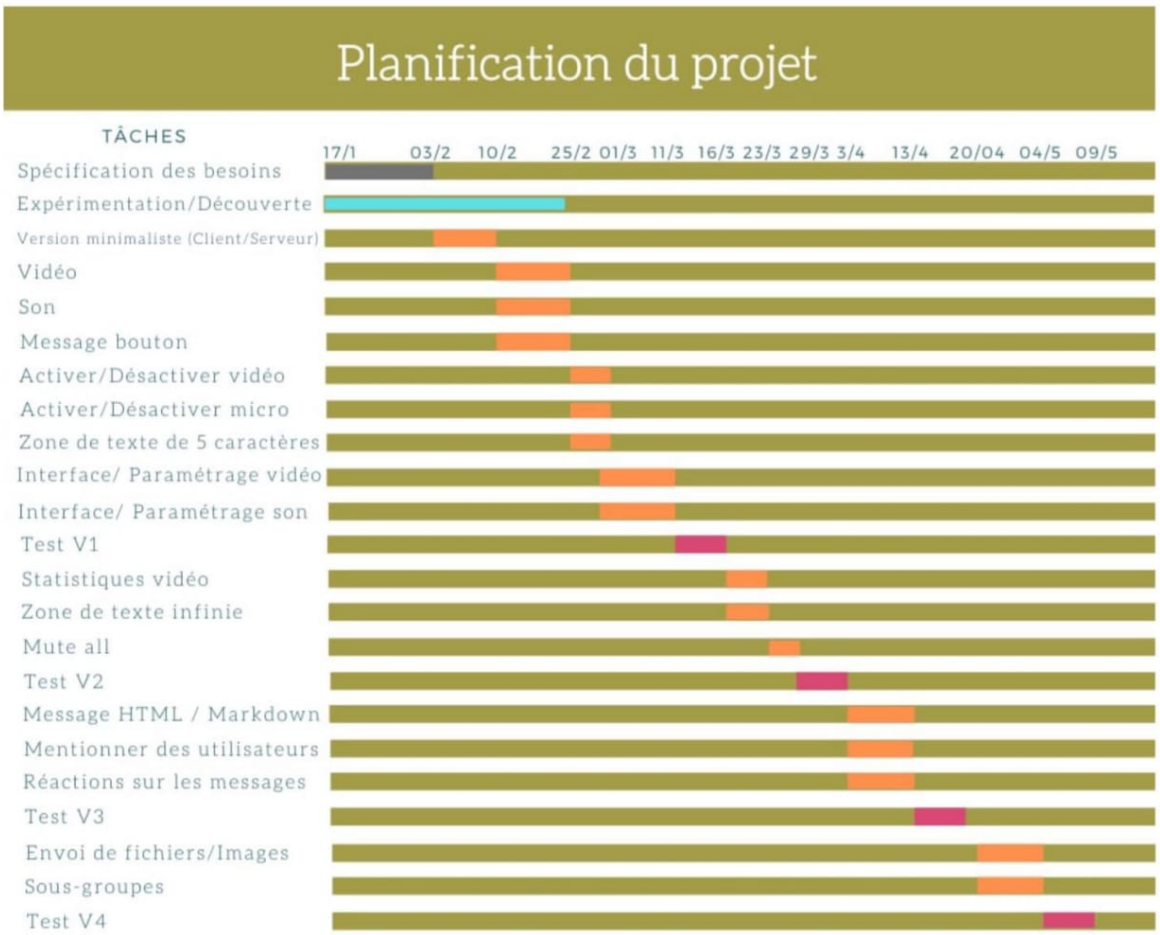

FIGURE 3 – Planification du projet

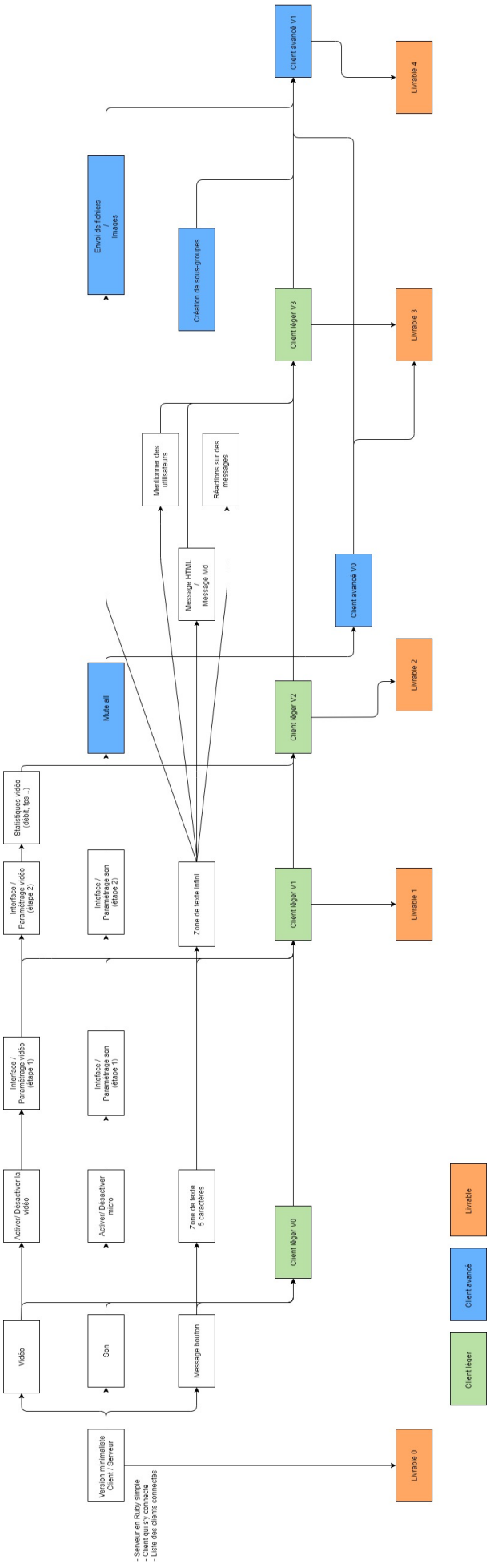

FIGURE 4 – Dépendances des tâches