Institut Universitaire de Technologie,
Aix-Marseille Université

RAPPORT DE STAGE
Diplôme Universitaire de Technologie
Spécialité Réseaux et Télécommunications

Développement d'une application Web de visioconférence

Damien JEANNE

Laboratoire Interdisciplinaire des Sciences du Numérique

Responsable entreprise : Olivier GLADIN

Responsable académique : Delphine ROUSSEAU

2021

Table des matières

1 Introduction

J'ai effectué mon stage au sein du Laboratoire Interdisciplinaire des Sciences du Numérique (**LISN),** au cours duquel j'ai développé une application web de visioconférence ayant pour but de faciliter les discussions informelles entre les différents sites du laboratoire.

Ma mission a été de recueillir des informations sur les besoins de notre client dans notre cas Michel BEAUDOIN-LAFON, enseignant-chercheur de l'équipe Ex-Situ*, afin de pouvoir choisir le matériel approprié et commencer le développement de l'application en fonction des besoins exprimés. Une fois ces informations recueillies, il nous a fallu imaginer les différents scénarios de visioconférence afin de pouvoir établir des séquences de fonctionnement, visant à nous guider lors de la création de l'application. Ce stage en développement m'a permis d'acquérir plus de maîtrise en développement informatique et d'aborder le réseau sous un autre angle autre que l'administration et la mise en place.

Après la présentation du laboratoire et ses différents secteurs de recherche, puis l'équipe que j'ai intégré, je détaillerai la démarche du projet et ce de la récolte d'information jusqu'à la réalisation du projet.

2 Présentations

2.1 Présentation du laboratoire

Le LISN a vu le jour grâce à la collaboration de 16 équipes de recherche de deux laboratoires, le Laboratoire d'Informatique pour la Mécanique et les Sciences de l'Ingénieur **(LIMSI)** et le Laboratoire de Recherche en Informatique **(LRI)** ainsi que les services de soutien et support à la recherche, soit 380 personnes. Il est dirigé par Sophie Rosset et Johanne Cohen, toutes deux Directrices de Recherches au CNRS.

Le laboratoire unifié se compose de 5 départements, chacun spécialisé dans un domaine :

- Département Algorithmes, Apprentissage et Calcul (AAC) :

Ce département s'intéresse à l'algorithmie et le calcul, ils touchent à tous les aspects théoriques et pratiques, logiciels et matériels de l'informatique. L'intelligence artificielle et l'apprentissage s'intéressent à la conception automatique d'algorithmes et de processus de calcul, guidée par les données, l'expert, l'utilisateur et/ou l'environnement.

- Département Science des données (SD) :

Le département Science des Données rassemble quatre équipes d'expertises complémentaires, couvrant la modélisation, la collection, la gestion, l'analyse et la construction de données et de connaissances. Des traces numériques de l'ensemble des activités humaines sont aujourd'hui disponibles dans tous les domaines, de la santé à l'éducation, de la finance et des assurances à la production industrielle, des sciences expérimentales aux sciences humaines.

- Département Mécanique Énergétique (ME) :

Les activités de recherche du département Mécanique-Énergétique recouvrent un spectre large, allant du numérique à l'expérimental, des échelles nanométriques aux échelles géophysiques, des analyses fondamentales aux études applicatives, et comprennent de larges interfaces avec la physique, les mathématiques appliquées et l'informatique.

- Département Interaction avec l'Humain (IaH) :

Ce département s'intéresse aux interactions entre l'Homme et les machines sous toutes ses formes en alliant différentes disciplines tel que l'informatique le traitement du signal ainsi que la psychologie cognitive et sociale dans le but de concevoir de nouveaux moyens d'interaction entre l'Homme et les machines omniprésentes dans nos quotidiens.

- Département Sciences et Technologies des Langues (STL) :

Le département Sciences et Technologies des Langues étudie des questions fondamentales relatives aux systèmes linguistiques, en exploitant de larges corpus collectés, annotés et enrichis de manière non-supervisée ou semi-supervisée, exploités par des modèles d'apprentissage statistique. En retour, ils permettent d'étudier le fonctionnement des langues. Enfin, le département développe les grandes applications du traitement des langues : reconnaissance vocale, traduction automatique, recherche d'information, agents conversationnels etc.

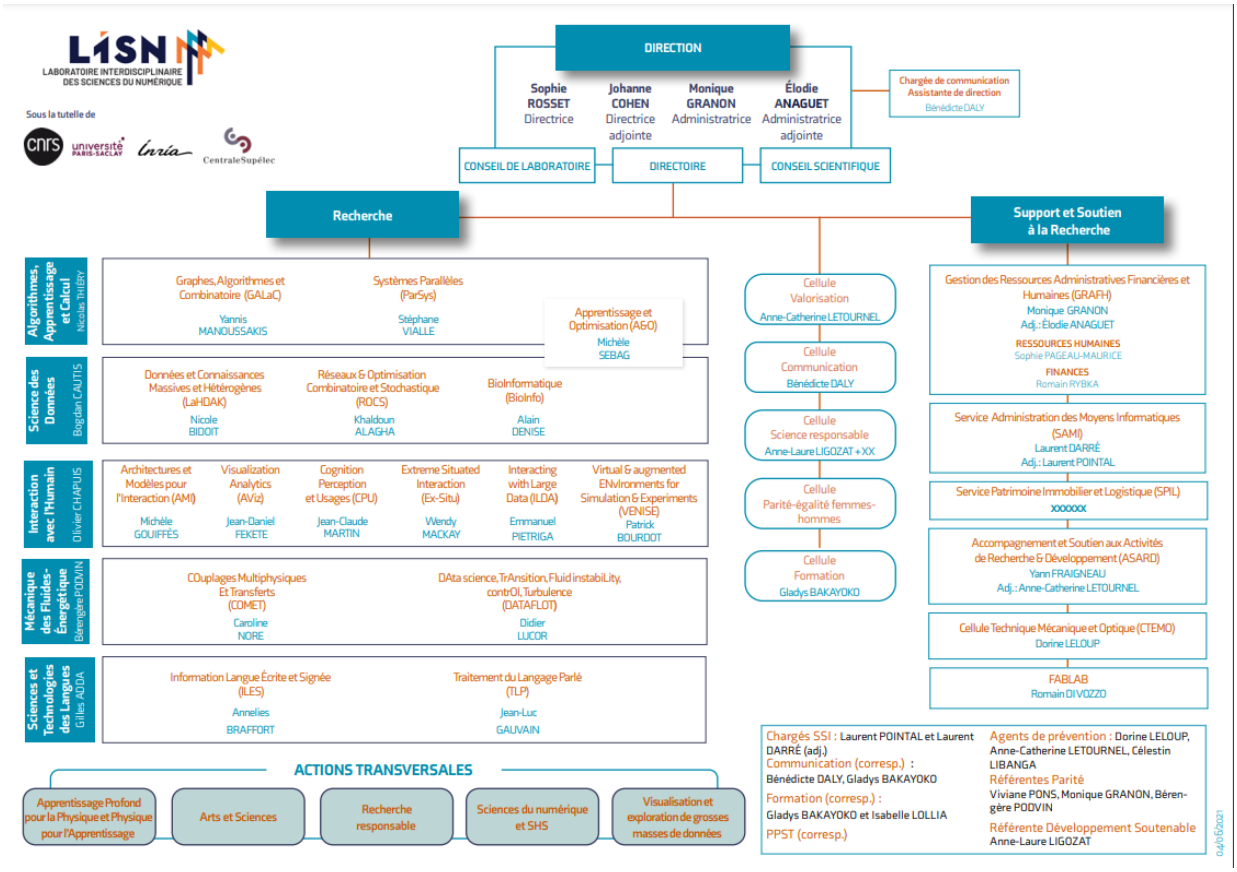

Figure 1 : Organigramme général du laboratoire

L'organigramme complet du laboratoire est disponible en annexe 1 : Organigramme complet

2.2 Présentation de l'équipe au sein du laboratoire

Au cours de mon stage j'ai intégré le département Interaction avec l'Humain (**IaH**), au sein de l'équipe Extreme Situated Interaction (**Ex-Situ**). Cette équipe a pour but de créer de nouvelle interface entre les hommes et les machines. De nos jours, nous avons les écrans tactiles qui commencent à se répandre sur tous nos équipements, cependant nous utilisons beaucoup plus le clavier et la souris, inventés en 1963 et toujours principalement utilisés de nos jours. L'équipe utilise donc des technologies naissantes telles que des casques de réalité augmentée et des caméras infrarouges afin de créer les interfaces de demain.

Les plateformes de travail de l'équipe sont les murs d'écrans des projets WILD* et WILDER*, ses murs d'écrans sont accompagnés de plusieurs caméras infra-rouge afin de pouvoir traquer des mouvements en utilisant des petits marqueurs réactifs aux ondes infrarouges. Ce système permet de créer différents types d'interface allant du pointeur infrarouge aux systèmes de spatialisations du son par exemple.

2.3 Objectif du stage

Ce projet consiste en l'installation de deux murs d'écrans placés dans des lieux de vie différents. Le but premier serait de permettre la communication de ces deux sites de manière fluide et sans difficultés. Ces deux écrans seront accompagnés d'un équipement qui assure la vision et l'écoute mutuelle de chaque participant. Bien que ce type de solution existe déjà notre objectif serait à terme d'avoir une solution évolutive à laquelle nous pourrons y ajouter des fonctionnalités en fonction des besoins des utilisateurs.
Les possibilités d'utilisation de ce système d'appel vidéo sont variées mais celui-ci va principalement être axé sur les interactions sociales entre les différentes personnes qui travaillent sur ces sites.
Son placement dans des lieux de vie et non dans des zones de travail n'est pas anodin. En effet, l'appel vidéo n'étant pas apprécié à cause de son utilisation quotidienne spécifiquement pour le travail. Avec cette méthode et cette décision de placement, l'appel deviendra plus chaleureux. Il sera moins contraignant pour les personnes qui l'utilisent. De ce fait, l'interface devra être aussi simple que concise pour assurer une utilisation complète et agréable de la part de la majorité des personnes présentes sur les différents sites.

3 Développement de l'application

3.1 Création des scénarios

Comme la durée du stage était très courte pour développer une application nous avons choisis de ne pas créer de cahier des charges et ne pas suivre un « cycle en V » (voir figure 2), car celui-ci tend généralement à évoluer tout au long de l'avancée du projet. Hebdomadairement nous avions donc une réunion avec Michel Beaudouin-Lafon afin de lui faire part de l'avancée du projet et qu'il nous fasse part de ses nouvelles exigences. Cela nous a permis de ne pas perdre de temps à développer des fonctions ne correspondant plus au besoin du client.

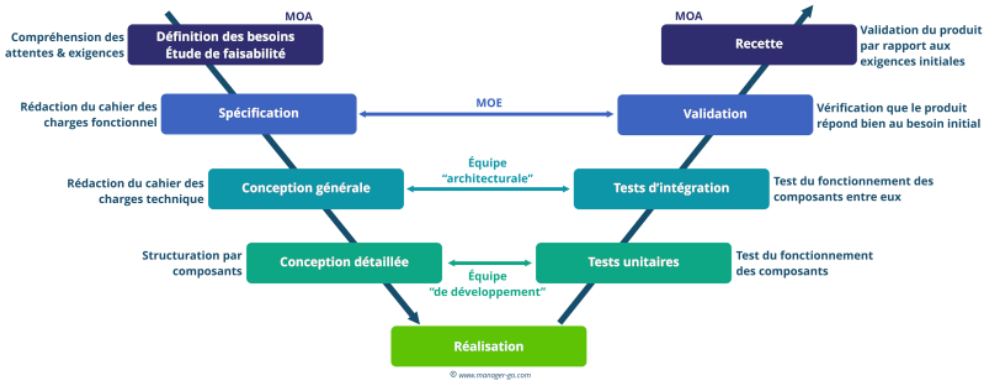

Figure 2 : Cycle en V

Les différents scénarios imaginés sont disponibles en Annexe 2 : Liste scénarios

3.2 Choix du matériel

Le choix du matériel a été crucial dans le projet, car des solutions de visioconférences existaient déjà. Notre but était donc de choisir du matériel offrant la meilleure expérience utilisateur tout en restant le plus simple possible.

3.2.1 Capture vidéo

Pour les caméras une des demandes était qu'elles soient Full-HD et pilotable afin que les utilisateurs puissent cadrer et orienter la caméra à leur guise. Notre choix s'est donc porté sur deux modèles de caméra PTZ* de chez Panasonic. La Panasonic AW-HE40 (figure 3), une caméra contrôlable Full HD* et la Panasonic AW-UE70(figure 4), une caméra contrôlable aussi mais cette fois ci avec qualité allant jusqu'à la 4K*, ce qui nous permettrai d'avoir un bien meilleur rendu vidéo dans le cas où l'on aurait des écrans capables d'afficher cette résolution. À ce stade nous n'avions pas encore fait notre choix sur les caméras à utiliser car les écrans n'étaient pas encore livrés et selon leurs résolutions cela nous limiterait dans le choix de la caméra.

Figure 3 : Panasonic AW-HE40

Figure 4 : Panasonic AW-UE70

3.2.2 Restitution vidéo

Les écrans nous ont été fournis par une banque qui ne les utilisait plus, nous avons donc reçu 6 écrans NEC Multisync v462n, malheureusement ces écrans n'ont qu'une résolution allant jusqu'au Full HD, cela nous limite donc dans le choix des caméras. Nous avons donc choisis les caméras Panasonic AW-HE40 car elles avaient la même résolution que nos écrans. Si nous avions eu la possibilité d'avoir 8 écrans nous aurions pu monter les écran sde sorte à atteindre une résolution allant à la [] et ainsi pouvoir utilisé les caméras 4k dans de meilleures conditions

Figure 5 : NEC Multisync v462n

3.2.3 Capture audio

Pour la capture audio nous avons choisis des micros de chez Audio Technica, le modèle PRO 44 (voir figure 5). Ce microphone de surface offre une excellente capture du son. Cependant afin de l'utiliser nous avons besoin d'une carte son externe avec des entrées XLR, au laboratoire des cartes RME Fireface UFX II (voir figure 6) étaient déjà disponibles. Bien que surdimensionné par rapport au cadre notre projet, elles conviendront parfaitement à l'utilisation que nous en ferons. De plus elles offrent des possibilités d'améliorations futures, telle qu'une spatialisation du son dans le cas où l'on décide d'utiliser plusieurs entrée XLR à différents angle.

Figure 6 : Microphone Audio Technica PRO 44

Figure 7 : Carte son RME Fireface UFX II

3.2.4 Restitution audio

La qualité du son est un paramètre principal lors de visioconférence, d'autant plus que la vidéo. Lors de mes recherches je suis tombé sur des enceintes ayant un rayon de projection du son directionnelle, ceci nous permettra de placer l'utilisateur dans une bulle sonore afin qu'il puisse entendre sans être entendu. Cependant ce matériel n'étant pas commun il nous est très compliqué de s'en procurer et de tester leurs efficacités sur la courte période du stage. Nous avons plutôt choisi une simple barre de son. Ce sera un point à améliorer par la suite afin d'améliorer l'expérience des utilisateurs.

3.2.5 Interface Homme Machine

Notre objectif ici était d'avoir l'interface la plus simple et accessible possible, nous nous sommes donc orientés vers 2 solutions, l'utilisation de cadre tactiles autour de nos écrans où utiliser de petits boîtiers programmables afin de créer une interface. Des boitiers de chez Elgato, les Elgato Stream Decks (voir figure 7) étaient déjà disponibles au laboratoire. Je me suis donc penché sur ses boîtiers, ils sont utilisés normalement par les streamers* afin d'implémenter une fonction à chaque touche et donc avoir une sorte de petite régie lors de leurs diffusions. Dans notre cas nous allons assigner des actions tels que raccrocher, appeler et contrôler la caméra aux différentes touches.

Figure 8 : Elgato stream deck

3.3 Outils de développement pour l'application

L'application sera basée sur du développement Web, nous utiliserons donc des outils basés sur le langage Javascript afin de créer les fonctionnalités de l'application ainsi que du HTML* et CSS* afin de créer la page de l'application.

3.3.1 Typescript :

Typescript est un langage de programmation libre et open source développé par Microsoft qui a pour but d'améliorer et de sécuriser la production de code JavaScript. Il s'agit d'un sur-ensemble syntaxique strict de JavaScript, il permet de typer les variables afin qu'elle ne puisse pas changer de type au cours du code car ceci entraine généralement des erreurs à l'execution. Si le cas se présente une erreur sera relevée et indiquée au développeur lors de l'édition afin qu'il puisse la corriger. Cependant ce langage n'est pas exécutable en tant que tel, il faut donc retransformer ce code en javascript, pour cela on utilise un « transpileur », un outil qui permet de traduire du code typescript en code javascript comme le ferait un outil de traduction linguistique.

3.3.2 Node.js

Node.js est une plateforme de développement Javascript. Ce n'est pas un serveur, ce n'est pas un framework*, c'est juste le langage Javascript avec des bibliothèques permettant de réaliser des actions comme écrire sur la sortie standard, ouvrir/fermer des connexions réseau ou encore créer un fichier. On l'utilisera pour créer des serveurs afin de créer les différentes connexions entre les composants logiciels et matériel.

3.3.3 Vue.js

Vue.js est un framework Javascript libre de droit permettant de créer des applications web monopage. Il se base sur un système de composant que l'on développe indépendamment, le framework se charge ensuite de créer notre page web avec les différents composants créés (voir figure 8). Un composant peut contenir du code HTML sous forme de template* et des scripts afin d'y ajouter des fonctionnalités. De plus, un composant peut lui-même contenir des composants cela nous permet de réutiliser certains template sur la même page simplement.

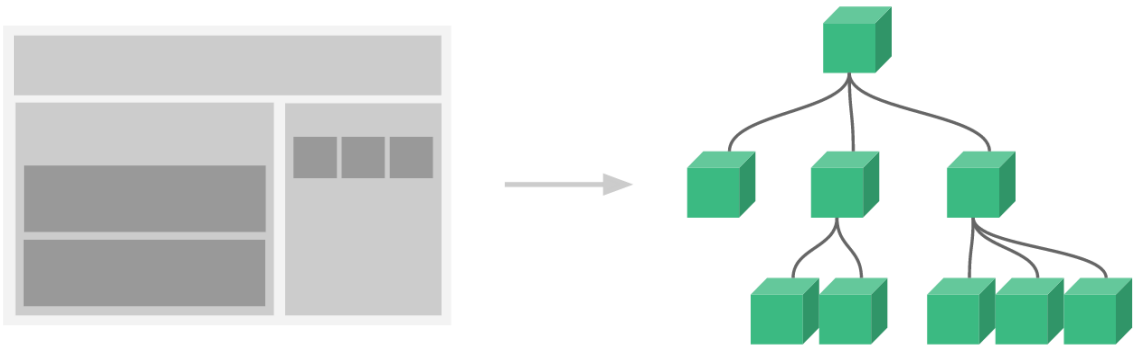

Figure 8 : Fonctionnement du système de composants dans Vue.js

3.4 Création des différents composants de l'application :

3.4.1 Création de la machine à état finit avec : Xstate

Avant de créer la machine, il nous a fallu définir les différents états logiques où la machine pourrait se trouver. Nous nous sommes inspirés du langage SysML* afin de créer un diagramme d'état de la machine. En fonction de l'état de la machine, on pourra désactiver ou activer des fonctionnalités de l'application en fonction des actions de l'utilisateur. Pour cela, nous utilisons Xstate, une librairie Javascript / Typescript permettant de créer une machine à état fini (**appelé FSM* par la suite**) dont le passage d'état en état se fait par l'envoie d'événements prédéfinis dans le code.

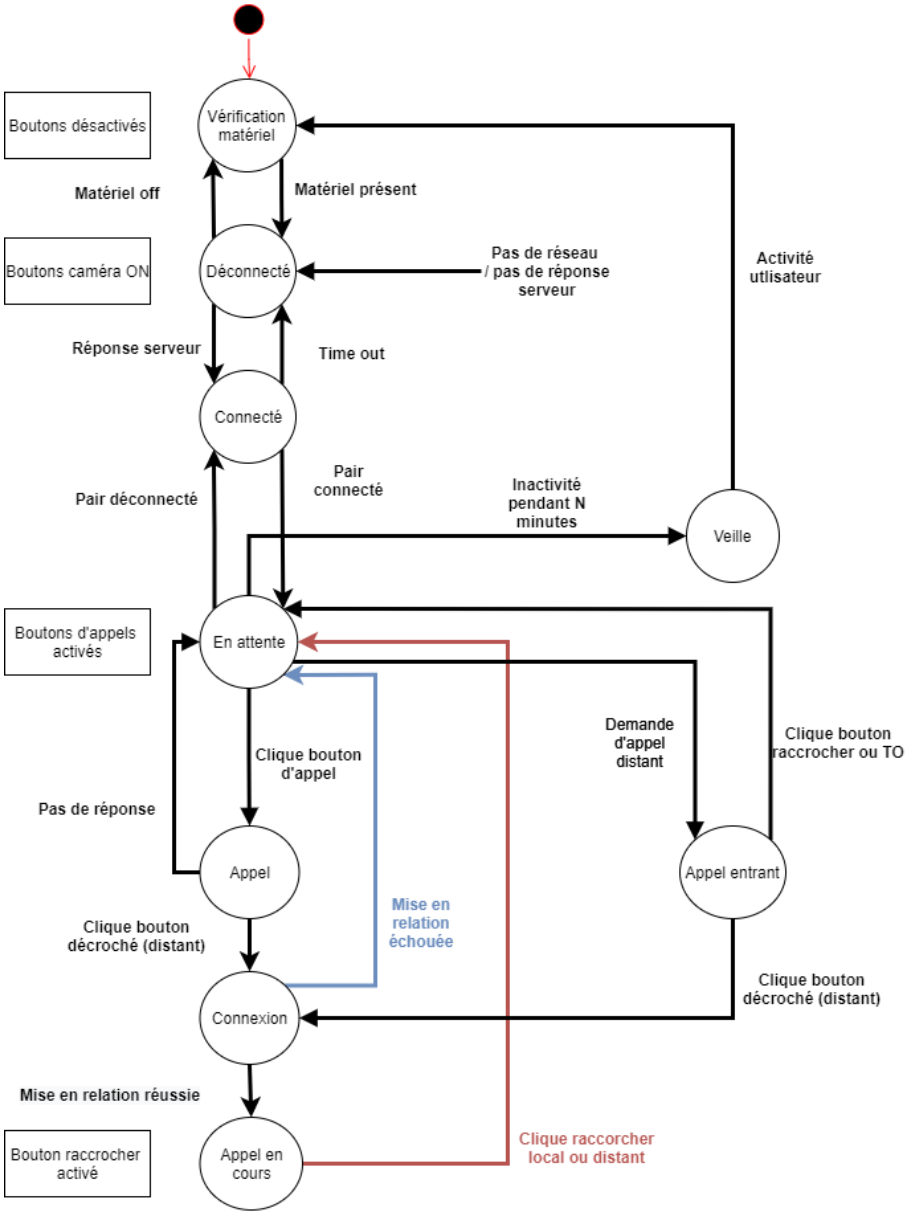

Figure 9 : Machine à états finis

3.4.2 Mise en relation des machines distantes : Socket.io & WebRTC

La connexion entre les deux applications se fera en Peer to Peer* (**appelé P2P par la suite**), c'est-à-dire une connexion directe entre les deux utilisateurs. Cependant cette connexion ne s'établits pas n'importe quand et de n'importe quelle manière, et ce pour des raisons évidentes de sécurité. De nos jours les réseaux ont deux types d'adresse IP, les adresses privées où se connectent nos ordinateurs et les adresses publiques rendant le réseau visible sur internet. Les routeurs se trouver à la frontière du réseau publique et privée (voir figure 9) et se chargent de transformer nos adresses privées en adresses publiques via de la translation d'adresse (**appelé NAT* par la suite**).

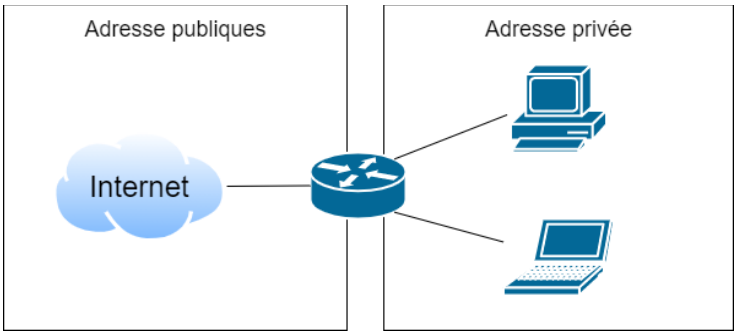

Figure 10 : Routeur frontière entre adresses publiques et privées

Serveur STUN :

Notre premier problème a été de trouver l'adresse IP publique de chaque machine, pour cela nous allons utiliser un serveur STUN (voir figure 10) qui nous fournira les adresses publiques de chaque machine. Cependant parfois le serveur STUN ne suffit pas à pouvoir se connecter au pair distant car il se peut que la sécurité du réseau soit plus accrue, il nous faudra donc passer par un intermédiaire.

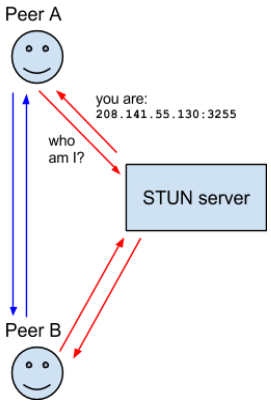

Figure 11 : Illustration serveur STUN

Serveur STUN + serveur TURN :

Lorsqu'un pare-feu rend impossible l'échange d'information direct entre les deux pairs, on peut passer par un serveur qui servira d'intermédiaire entre les deux pairs, c'est un serveur TURN* (voir figure 11). Cette solution est à utiliser seulement dans les cas où on ne peut pas faire sans, car ses serveurs ajoutent de la latence lors de la communication en limitant la bande passante et en fonction des performances réseaux du serveur lui-même.

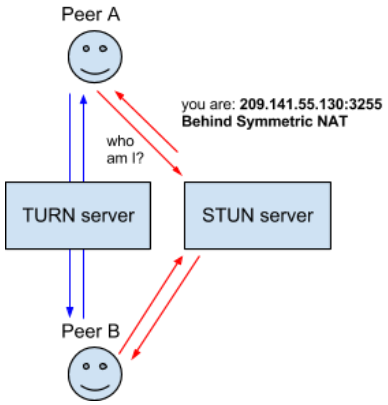

Figure 12 : Serveur TURN faisant l'intermédiaire

Processus de mise en relation :

Dans notre cas, l'utilisation d'un serveur STUN suffit à pouvoir mettre en relation les deux pairs. Une fois ses adresses publiques obtenues nous utiliserons WebRTC, une interface de programmation Javascript qui nous permettra de créer des offres et des réponses d'offre de connexions pour les connexions P2P. Cependant nous n'avons aucun moyen d'échanger ses messages, nous allons donc avoir besoin de passer par un serveur externe afin d'échanger ses informations. Il ne servira plus par la suite car l'échange des vidéos et du son se fera d'un Pair à l'autre. On pourra y connecter un socket* afin de pouvoir échanger des messages avec le pair distant, en passant par cet intermédiaire. Attention il ne faut pas le confondre avec le serveur TURN car il ne sera pas destiné à envoyer des flux vidéo ou audio.

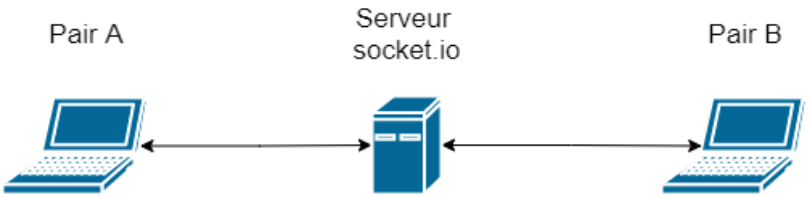

Figure 13 : Passerelle socket.io

Les messages échangés sont des offres contenant des informations SDP*, ce sont des informations concernant la résolution, les formats, codecs et cryptages des différents médias de l'utilisateur. Ses offres ont pour but de mettre d'accord les deux pairs, afin qu'ils puissent comprendre les données qu'ils s'envoient. Ce processus d'échange de message doit suivre un certain ordre, afin que la connexion s'établisse correctement. Le processus est décrit dans la figure 13 ci-dessous.

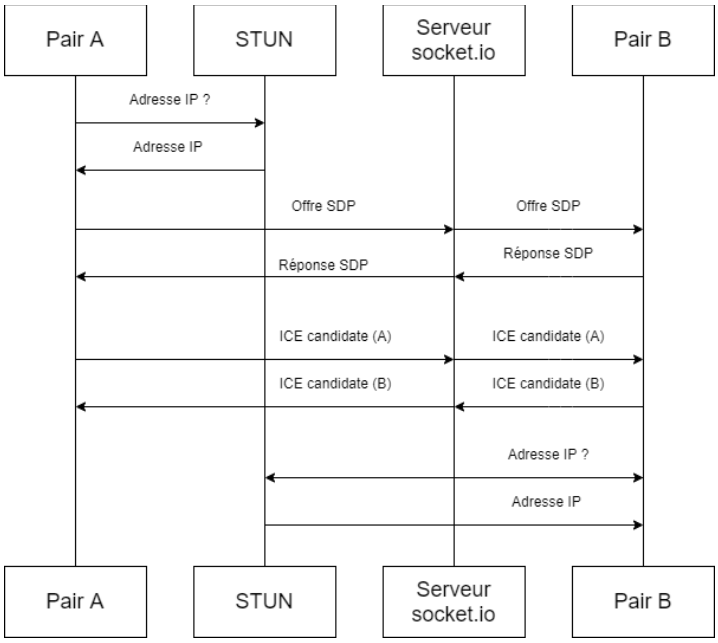

Figure 14 : Processus de mise en relation P2P

3.4.3 Programmation de l'Elgato Stream Deck :

Pour l'interface de contrôle de l'application nous avons choisis d'utiliser des Elgato Stream Deck, car nous avons fait des recherches et il se trouve qu'ils sont personnalisables tant au niveau de l'apparence qu'au niveau des fonctionnalités qu'on peut lui attribuer. Une fois le deck branché au PC nous pouvons lancer un serveur sur ce même PC qui sera à l'écoute des évènements relatifs au deck, tel que l'appuie d'une touche ou son relâchement. Une fois la récupération des touches faite on peut commencer à y assigner des actions tel que l'orientation de la caméra, raccrocher, appeler, et toutes les autres actions. Ses actions seront envoyées à l'application par le biais d'un serveur socket.io dans la même idée que dans le cas des échanges des offres SDP.

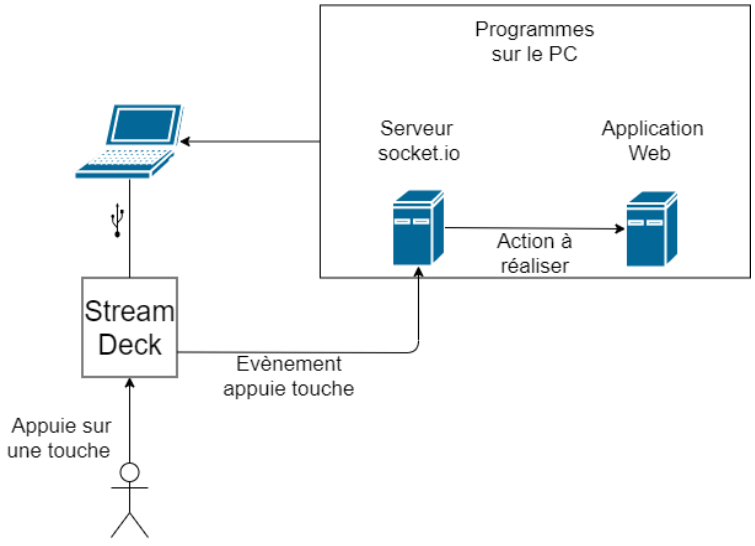

Figure 15 : Envoie d'évènements du Stream deck à l'application web

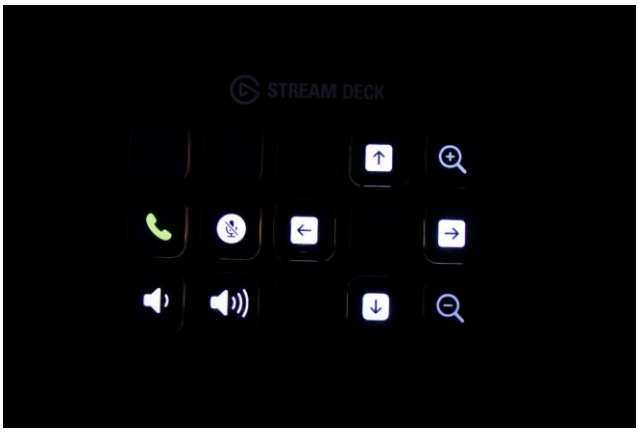

Figure 16 : Interface finale du stream deck

Certaines touches du stream deck auront des fonctions fixes comme les touches de déplacement de la caméra, puis d'autres verront leurs fonctionnalités changer en fonction de l'état de la machine comme la touche d'appel ou la touche permettant de couper / réactiver son micro. Pour cela nous utilisons l'état de la FSM afin de définir l'apparence et la fonctionnalité de la touche.

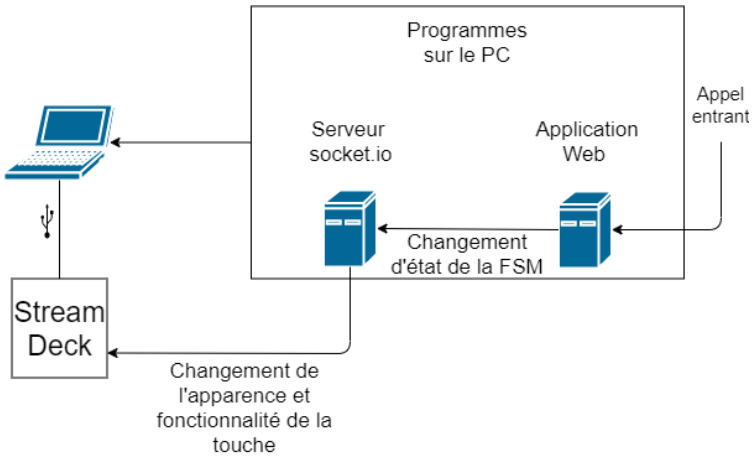

Figure 17 : Changement des fonctionnalités du stream deck

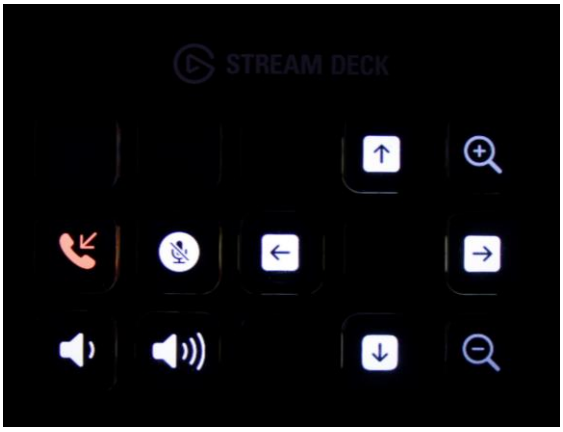

Figure 18 : Touche en mode raccrocher

3.4.4 Déplacement de la caméra :

Les caméras que nous avons choisis d'utiliser intègre en elles des serveurs web nous permettant de les contrôler à distance. En envoyant à ce serveur des requêtes http avec en options les paramètres de mouvement, nous pourrons la contrôler. Ces requêtes sont basées sur 3 paramètres principaux :

- Pan → Tourner la caméra : ce paramètre varie de -16 à 16 selon si nous désirons aller à gauche ou à droite ;
- Tilt → Monter / descendre la caméra : ce paramètre varie de -16 à 16 selon si nous désirons aller en bas ou en haut ;
- Zoom → Zoom / dézoome de la caméra : ce paramètre varie de -4 à 4 selon si nous désirons dézoomer / zoomer ;

Ses paramètres sont des valeurs relatives à la position de la caméra, c'est-à-dire que si pan=4 la caméra tournera de 4 unités vers la droite à partir de sa position actuelle.

Afin d'envoyer ses requêtes nous utiliserons un objet Javascript, l'objet XMLHttpRequest, il nous permet de communiquer avec des serveurs et d'y récupérer tout type de donnée via des requêtes http. Ici nous utiliserons seulement sa capacité à pouvoir envoyer des requêtes car ce que nous récupèrerons de la caméra est son flux vidéo via le réseau local du laboratoire.
Pour créer la requête http nous utiliserons une boucle qui sera à l'écoute d'évènements des appuis de certaines touches du clavier. Ses évènements se déclenchent à l'appui de n'importe quelle touche, il nous a donc fallu un moyen de trier les touches nous intéressant. Dans notre cas les flèches directionnelles, correspondant aux directions de la caméra, ainsi que les touches « + » et « - » pour zoomer et dézoomer. Nous avons donc créé une liste avec ses touches puis à chaque appuie nous vérifions si la touche pressée nous concerne et si elle nous concerne augmenter le paramètre correspondant dans la requête. Les algorigrammes de ses deux fonctions sont présents à la figure n ci-dessous.

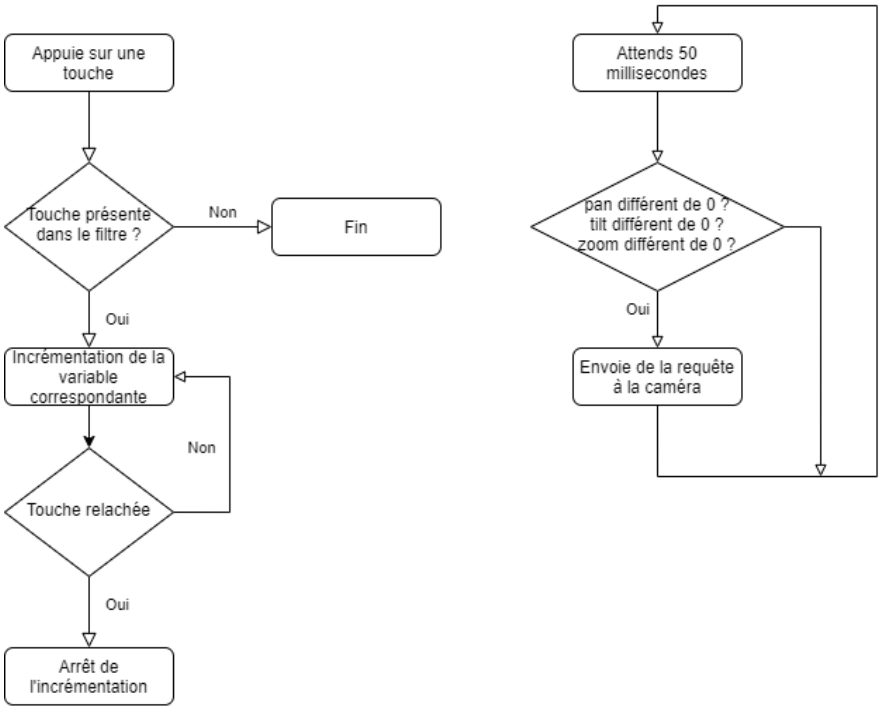

Figure 19 : Algorithme du contrôle de la caméra.

Ci-dessous se trouver un schéma illustrant le détail des connexions et des échanges de données lorsque l'utilisateur appuie sur la touche correspondant à la monté de la caméra.

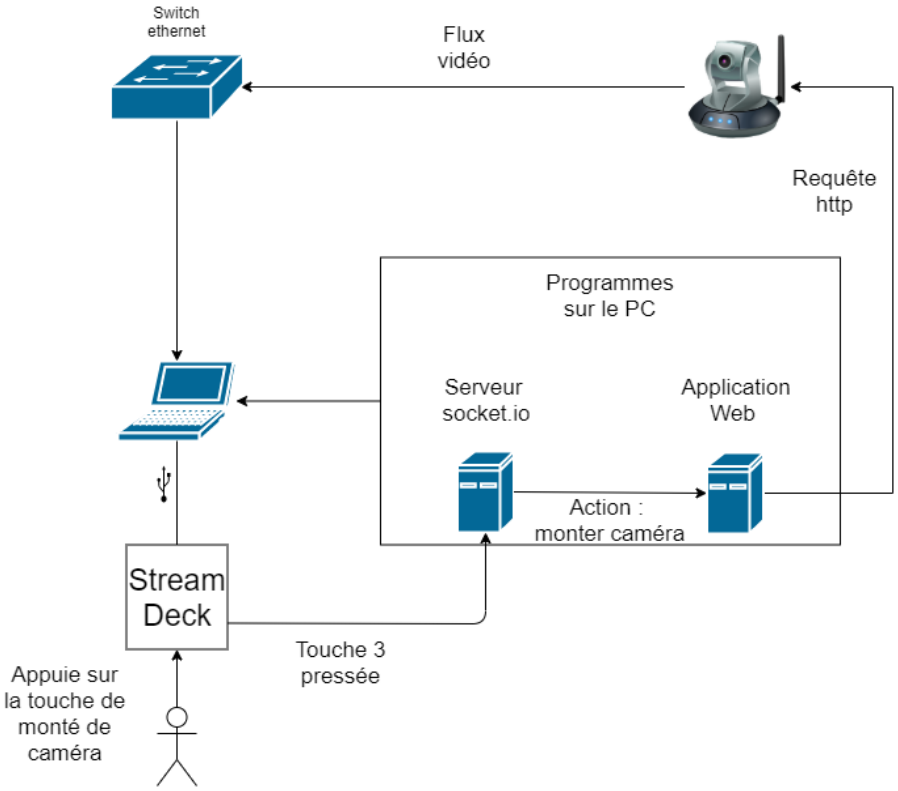

Figure 20 : Poste de vidéoconférence

4 Conclusion

Le mardi 22 juin nous avons pu faire une démonstration entre les deux murs d'écran WILD et WILDER à Michel BEAUDOIN-LAFON, lors que laquelle il nous a fait part des améliorations à apporter à l'application, par exemple au niveau du retour d'information. Il nous a notifié l'absence d'information à l'écran sur l'état de la machine. Mis à part l'affichage dynamique des touches sur le stream deck pouvant nous révéler des informations concernant l'état de la machine et le cadrant de la vidéo distante, il manque nous manque ses informations d'état. Cette démonstration a été réalisée sur des prototypes, les périphériques utilisés sur la figure 17 ne sont pas définitifs, la qualité atteinte lors de cette n'est pas optimale et ne représente donc pas exactement ce que nous souhaitions réaliser.

Figure 21 : Prototype du système de visioconférence.

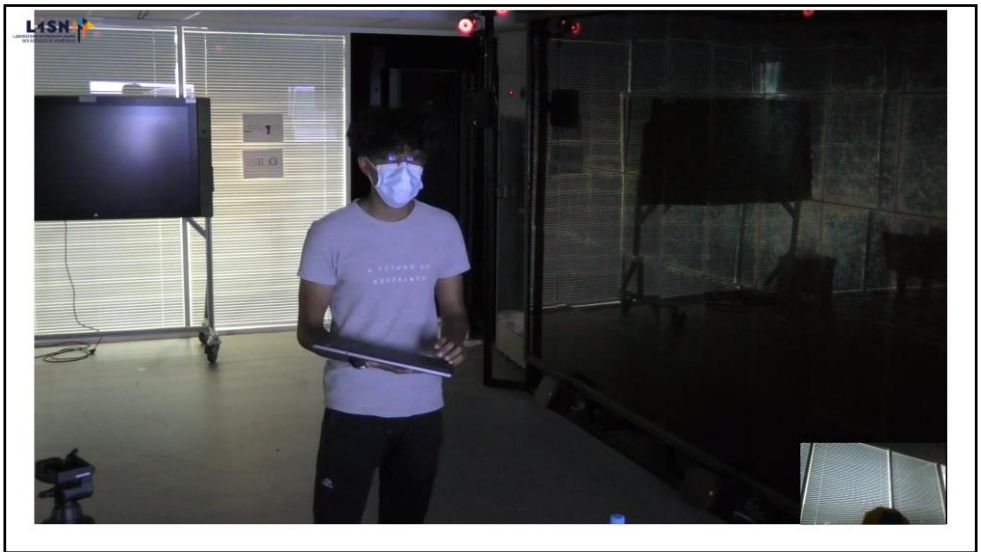

Figure 22 : Interface de l'application finale

Finalement nous avons pensé à laisser ce système de démonstration entre les deux salles de WILD et WILDER, car la communication entre ses deux salles peut s'avérer utile lors d'expérience. Avant d'ajouter ce dispositif dans des lieux de vie, une amélioration notable à amener serait l'automatisation du lancement de l'application, cela rendrait le système plus robuste aux erreurs et rendrait l'expérience utilisateur d'autant plus agréable.

5 Remerciements

Premièrement je tiens particulièrement à remercier mon maitre de stage M. Olivier GLADIN et M. Michel BEAUDOUIN-LAFON pour leur confiance et leur temps pendant toute la durée de mon stage, dont les conditions étaient difficiles.

Je tiens aussi à remercier pour leur accueil chaleur, toutes les personnes que j'ai rencontrées de près ou de loin durant mon court passage au laboratoire.

6 Glossaire

DUT, Diplôme Universitaire de Technologie

Ex-Situ, Extreme Situated Intercation, nom de l'équipe intégrée

WILD, Wall-sized Interaction with Large Datasets, mur d'écran très haute résolution avec suivi de mouvement

WILDER, Mur d'écrans très haute résolution multitactile avec de suivi de mouvements

PTZ, de l'anglais Pan Tilt Zoom, action de monter, desendre, zoom et rotation de la caméra

Full HD, résolution d'image, 1920 x 1080 pixels

4k, résolution d'image, 3840 x 2160 pixels

Streamer, personne partageant du contenu vidéoludique en direct sur des platforms en ligne

HTML, langage de programmation web servant à créer des pages web statiques

CSS, langage de programmation permettant de styler les pages web HTML

Framework, environnement de développement

Template, terme informatique définissant un modèle / exemple d'une page web

SysML, System Modeling Language, langage de modélisation de système

FSM, Final State Machine, machine à états finis

Peer to Peer / P2P, connexion directe entre deux utilisateurs via internet

NAT, Network Address Translation, transformation d'adresse réseau

STUN, Session Traversal Utilities for NAT (acronyme dans un acronyme oui…)

TURN, Transveral Using Relay around NAT

Socket, anglissime correspondant à une connexion à un serveur

SDP, Session Description Protocol, protocole permettant de définir les médias d'un utilisateur

7 Sitographie

https://developer.mozilla.org/fr/docs/Web/API/WebRTC_API/Connectivity
https://xstate.js.org/docs/
https://vuejs.org/
https://webrtc.org/
https://socket.io/
https://www.lisn.upsaclay.fr/
http://digiscope.fr/fr/platforms/wild
http://digiscope.fr/fr/platforms/wilder
https://images.cnrs.fr/recherche?direct-query=wilder