



RÉPUBLIQUE DU BÉNIN  
MINISTÈRE DE L'ENSEIGNEMENT SUPÉRIEUR  
ET DE LA RECHERCHE SCIENTIFIQUE  
UNIVERSITÉ D'ABOMEY-CALAVI  
INSTITUT DE FORMATION ET DE  
RECHERCHE EN INFORMATIQUE  
BP 526 Cotonou Tel : +229 55 02 80 70  
<http://www.ifri-uac.bj> Courriel : [contact@ifri.uac.bj](mailto:contact@ifri.uac.bj)



# MÉMOIRE

pour l'obtention du

Diplôme de Licence en Informatique

Option : Génie Logiciel

Présenté par :

Hans Bignon. K. TOGNON

## Conception d'un SaaS pour les cours en ligne en milieu universitaire

Sous la supervision :

Ing. Miranda GNONLONFOUN

Membres du jury :

Année Académique : 2021-2022

# Sommaire

Dédicace	ii
Remerciements	iii
Résumé	iv
Abstract	v
Liste des acronymes	viii
Glossaire	xi
Introduction	2
1 Revue de littérature	4
2 Matériels et méthodes	11
3 Résultats et Discussion	24
Conclusion	36
Bibliographie	37
Webographie	38

# Dédicace

A *Ma famille*, pour le soutien dont vous avez toujours fait montre.

# Remerciements

Je tiens à exprimer mes sincères remerciements à toutes les personnes qui ont contribué de près ou de loin à la réalisation de ce mémoire.

Tout d'abord, je remercie Professeur Eugène C. EZIN, directeur de l'institut, ainsi que tous les cadres de l'institut, pour m'avoir offert un environnement propice à l'apprentissage et au développement de mes compétences.

Je suis également reconnaissant envers mes parents, pour leur soutien indéfectible tout au long de mon parcours universitaire.

Je souhaite exprimer ma profonde gratitude à Mme. Miranda GNONLONFOUN, mon maître de mémoire, pour son encadrement, ses conseils éclairés et son soutien durant tout le processus de rédaction de ce mémoire. Ses remarques et suggestions ont été très précieuses et ont contribué à en améliorer significativement la qualité.

Je voudrais aussi remercier M. Chukwudi Nwachukwu, pour m'avoir enseigné bien des choses, sans lesquelles ce projet n'aurait su aboutir.

Enfin, je tiens à remercier toutes les personnes que je ne peux pas mentionner individuellement ici. Votre soutien, vos encouragements et vos conseils ont été d'une valeur inestimable.

# Résumé

L'enseignement supérieur est confronté à de nouvelles contraintes ; en témoigne la pandémie de COVID-19 qui a pertubé le déroulement de cours présentiel. Les technologies de communication en temps réel offrent une alternative prometteuse, qu'il convient d'exploiter pour répondre à ces défis. Afin de favoriser l'utilisation de ces outils de communication, notre projet propose StudX, un prototype d'application permettant la tenue de classes virtuelles. Le développement de cette plateforme a nécessité l'utilisation de techniques modernes de modélisation logicielle, de technologies de pointe et de ressources variées. StudX offre une solution fiable pour répondre aux besoins des universités en matière d'enseignement à distance, grâce à des fonctionnalités de communication en temps réel basées sur WebRTC. Ce projet est une contribution importante à l'exploitation des technologies de communication en temps réel dans le domaine de l'éducation et offre des perspectives d'extension pour répondre à de nouveaux besoins.

**Mots clés :** StudX, WebRTC, cours en ligne, universités, communication en temps réel

# Abstract

Higher education is facing new constraints, such as the inability to attend face-to-face classes due to various factors such as pandemics. Real-time communication technologies offer a promising alternative that must be exploited to meet these challenges. In order to promote the use of these communication tools, our project proposes StudX, a prototype application for virtual classes. The development of this platform required the use of modern software modeling techniques, state-of-the-art technologies and various resources. StudX provides a reliable solution to the distance learning needs of universities through WebRTC-based real-time communication capabilities. This project is an important contribution to the exploitation of real-time communication technologies in the field of education and has the potential to be extended to meet new needs.

**Keywords:** StudX, WebRTC, online classes, universities, real time communication

# Liste des Figures

1.1	Protocoles employés par la technologie WebRTC . . . . .	6
1.2	Page d'accueil de Google Classrooms . . . . .	7
1.3	Utilisation de Google Meet . . . . .	8
1.4	Offres de souscription à la suite Google pour éducation . . . . .	8
1.5	Interface de l'application Zoom . . . . .	9
1.6	Interface de l'application Zoom . . . . .	9
1.7	Démonstration des capacités de Moodle . . . . .	10
1.8	Démonstration de BigBlueButton . . . . .	10
2.1	Diagramme de cas d'utilisation du prototype StudX . . . . .	12
2.2	Diagramme de séquence pour le cas d'utilisation "Ajouter un événement" . . . . .	13
2.3	Diagramme de séquence pour le cas d'utilisation "Rejoindre une réunion" . . . . .	14
2.4	Diagramme de classe . . . . .	15
2.5	Architecture du système du prototype StudX . . . . .	15
2.6	Example de configuration du proxy Caddy . . . . .	16
2.7	Logo du framework Django . . . . .	17
2.8	Logo du framework Vue.js . . . . .	18
2.9	Logo de PostgreSQL . . . . .	18
2.10	Logo de Celery . . . . .	19
2.11	Logo de Redis . . . . .	19
2.12	Logo de TipTap et version actuelle . . . . .	19
2.13	Interface par défaut de whiteboard . . . . .	20
3.1	Page d'authentification de StudX . . . . .	24
3.2	Calendrier des planifications . . . . .	25
3.3	Calendrier des planifications . . . . .	25
3.4	Détails d'un événement . . . . .	26
3.5	Grille des participants et boutons de contrôle. . . . .	27
3.6	Messagerie instantanée intégrée à StudX . . . . .	28
3.7	Partage d'écran. . . . .	29
3.8	Tableau virtuel . . . . .	30
3.9	Outil de note synchronisé . . . . .	30
3.10	Liste des participants . . . . .	31
3.11	Processus de déconnexion. . . . .	32
3.12	Tutoriel interactif d'introduction à StudX . . . . .	33
3.13	Mode sombre . . . . .	33
3.14	Paramètres d'une organisation. . . . .	34

Liste des figures

# Liste des Tables

1.1	Classification des types de formations en ligne . . . . .	5
2.1	Liste non exhaustive des outils employés dans la réalisation de <b>StudX</b> . . . . .	22
	Liste des tableaux	

# Liste des acronymes

**API :**

Application Programming Interface [17](#)

**CSS :**

Cascading StyleSheet [17](#)

**DTLS :**

Datagram Transport Layer Security : [DTLS](#), [6](#)

**EDI :**

Environnement de développement intégré [22](#)

**HTML :**

HyperText Markup Language [17](#)

**HTTP :**

HyperText Transfer Protocol [17](#), [18](#), [20](#)

**ICE :**

Interactive Connectivity Establishment : [ICE](#), [6](#)

**IP :**

Internet Protocol : [IP](#), [xi](#), [1](#), [6](#)

**MOOC :**

Massive Open Online Courses [5](#)

**NAT :**

Network Address Translation : [NAT](#), [1](#), [6](#)

**OMG :**

Object Management Group [11](#)

**P2P :**

Peer To Peer : [P2P](#), [xi](#), [5](#)

**PWA :**

Progressive Web Apps [17](#)

**REST :**

Representational State Transfer [17](#)

**RTP :**

Real-time Transport Protocol : [RTP](#), [1](#), [6](#)

**SCTP :**

Stream Control Transmission Protocol : [SCTP](#), [6](#)

**SDP :**

Session Description Protocol : [SDP](#), [6](#)

**SFU :**

Selective Forward Unit : [SFU](#), [20](#)

**SMOC :**

Synchronous Massive Online Courses [5](#)

**SPOC :**

Small Private Online Courses [5](#)

**SRTP :**

Secure Real-Time Transport Protocol : [SRTP](#), [6](#)

**SSOC :**

Synchronous Small Online Courses [5](#)

**STUN :**

Session Traversal Utilities for NAT : [STUN](#), [6](#)

**TLS :**

Transport Layer Security [20](#)

**TURN :**

Traversal Using Relays around NAT : [TURN](#), [6](#)

**UDP :**

User Datagram Protocol : [UDP](#), [1](#)

**UML :**

Unified Modeling Language [11](#)

**WebRTC :**

Web Real-Time Communication [5](#), [6](#), [20](#)

# Glossaire

**DTLS :** Protocole qui fournit une sécurisation des échanges basés sur des protocoles en mode datagramme. [viii](#)

**ICE :** Protocole utilisé dans les réseaux informatiques pour trouver des moyens permettant à deux ordinateurs de se parler aussi directement que possible dans le cadre d'un réseau P2P. [viii](#)

**IP :** Famille de protocoles de communication de réseaux informatiques conçus pour être utilisés sur Internet. [viii](#)

**NAT :** Méthode de conversion d'un espace d'adresses IP en un autre en modifiant les informations relatives aux adresses de réseau dans l'en-tête IP des paquets pendant qu'ils transitent par un dispositif d'acheminement du trafic. [vii](#)

**P2P :** Modèle d'échange en réseau où chaque entité est à la fois client et serveur. [ix](#)

**RTP :** Protocole réseau qui décrit comment transmettre divers médias (audio, vidéo) d'un point de terminaison à un autre en temps réel. [ix](#)

**SCTP :** Protocole de transmission fiable des messages de télécommunication à travers des réseaux IP. [ix](#)

**SDP :** Protocole de communication de description de paramètres d'initialisation d'une session de diffusion en flux. [ix](#)

**SFU :** Serveur qui relaie le trafic multimédia, où chaque homologue s'y connecte pour récupérer les flux de média. [ix](#)

**SRTP :** Protocole employant le cryptage et l'authentification pour minimiser le risque d'attaques par déni de service et les failles de sécurité du protocole [RTP](#). [ix](#)

**STUN :** Protocole client-serveur permettant à un client [UDP](#) situé derrière un routeur [NAT](#) de découvrir son adresse IP publique ainsi que le type de routeur [NAT](#) derrière lequel il est. [ix](#)

**TURN :** Protocole qui aide à traverser les traducteurs d'adresses réseau ou les pare-feu pour les applications multimédias. [ix](#)

**UDP :** Protocole de transmission de données sans connexion (sous forme de datagrammes) entre deux entités. [ix](#)

# Introduction Générale

L'utilisation des technologies de l'information et de la communication dans l'enseignement supérieur est effective. Cependant, face aux défis d'ordre logistique et financier, il faut adopter des solutions efficaces et minimales, pour la tenue de cours en ligne.

## Contexte

Les temps changent et les méthodes d'enseignement également. Dans le cadre de la mondialisation, on assiste à l'emploi du numérique, avec des méthodes de plus en plus créatives et collaboratives, pour une meilleure éducation. Cela reste valable dans le milieu de l'enseignement supérieur. Il va sans dire que des facteurs comme la pandémie de COVID-19 ou encore, l'indisponibilité de cadres de cours adéquats, rendent plus urgent le besoin de transitionner vers des salles de classe virtuelle, pour répondre aux besoins. De ce fait, les technologies de l'information et de la communication constituent un atout décisif dans le succès de cette transition.

## Problématique

L'expansion des cours en ligne est désormais un fait. Cela requiert une organisation logistique accrue et un investissement financier pour les entités universitaires. Toutefois, l'on note l'emploi de solutions génériques, qui rendent difficile, voire impossible l'émulation d'un environnement de classe. Pour peu qu'elles soient conformes aux exigences, c'est alors le prix qui peut poser problème. C'est la raison d'être de notre projet, qui vise la mise en place d'une application, pour répondre au besoin d'interactivité lors des cours en ligne, et éliminer les barrières d'ordre logistique et financier, imposées par les solutions génériques.

## Objectifs

Le principal objectif est la conception de StudX, un prototype d'application SaaS, permettant la tenue de cours en ligne. En termes de fonctionnalités et buts, il s'agira notamment de pouvoir :

- organiser les différentes classes, filières ou promotions des entités en sections bien définies ;

- définir le calendrier des cours à tenir;
- organiser des sessions d'audio-conférence pour le déroulement des cours;
- mettre en place des fonctionnalités telles le partage d'écran et bien d'autre pour émuler un tant soit peu, un environnement de classe présentiel;
- minimiser les coûts requis dans le cadre de la mise en oeuvre d'une solution de classe virtuelle.

## Organisation du document

Le présent document renferme trois chapitres. Dans le premier chapitre, nous ferons une revue de littérature sur le sujet et présenterons les généralités sur quelques notions essentielles. Le second chapitre relate les méthodes employées pour la conception de notre solution ainsi que les outils et matériels utilisés à cette fin. Le dernier chapitre sera consacré à la présentation des résultats obtenus, des interfaces conçues ainsi que des potentielles insuffisances liées à la solution que nous avons développée.

# Revue de littérature

## Introduction

Le concept de la formation à distance ne date pas d'hier. Dans ce chapitre, nous ferons une revue des origines de cette méthode d'enseignement. S'en suivra une analyse des techniques modernes de communication en temps réel et des solutions existantes qui permettent de dispenser des cours à distance.

### 1.1 Formation à distance

L'encyclopédie Wikipedia définit la formation à distance comme une forme d'enseignement où l'enseignant et l'étudiant sont séparés dans le temps et/ou par l'espace [11].

#### 1.1.1 Origines

Les premiers essais de formation à distance remontent à bien avant l'ère moderne. En effet, déjà en 1728, Caleb Phillips, un professeur, recherchaient des étudiants désirant acquérir des compétences en sténographie, auxquels il dispensait les cours par courrier.

Au sens moderne, le premier cours d'enseignement à distance, est attribué à Isaac Pitman, toujours en rapport avec la sténographie. Un nouvel élément qui apparaît dans le cas actuel, c'est la rétroaction des étudiants, qui devaient envoyer leurs transcriptions par la poste, pour correction. Ce mode de fonctionnement fut rendu possible par l'uniformisation des tarifs postaux en Angleterre. Plusieurs institutions telles que Oxford et l'université de Londres ont également expérimenté l'enseignement à distance.

Aujourd'hui, l'expansion d'Internet et du World Wide Web, permettent la mise en œuvre des moyens toujours plus sophistiqués, pour dispenser les cours à distance.

#### 1.1.2 Internet et formations en ligne

L'avènement des nouvelles technologies de l'information et de la communication a donné lieu à la mise en place des formations en ligne, une forme évoluée de formation à distance [17]. En 2020, par

exemple, sous l'impulsion de la pandémie alors en cours, plusieurs universités dont celle d'Abomey-Calavi, effectuent la transition partielle ou totale vers les classes virtuelles [10].

On distingue deux environnements d'apprentissage. D'une part, les environnements asynchrones, qui offrent une totale liberté à l'apprenant quant à la gestion de son temps. L'enseignant et lui sont séparés littéralement par le temps et la distance. Ainsi, il peut consulter les ressources au moment qui lui convient le mieux. Cela permet une assimilation plus facile, étant donné que chaque apprenant peut s'adapter en fonction de ses besoins spécifiques. Toutefois, il est possible que l'apprenant se retrouve isolé et ne fasse finalement aucun progrès, faute de support.

D'autre part, un environnement synchrone essaie d'émuler une classe présentiel, à la seule différence que les participants sont physiquement distants. Avec des outils de messagerie instantanée et/ou de visioconférence, les apprenants peuvent interagir avec leurs pairs ainsi que le ou les enseignants.

En termes de classification des diverses formes de cours en ligne, Andreas Kaplan, auteur du livre *Contemporary Issues in Social Media Marketing* [5], propose une approche simplifiée basée sur les facteurs comme le temps, la distance et le nombre d'apprenants. Le tableau 1.1 en fait un récapitulatif.

Type de cours	Nombre d'apprenants	Type d'environnement
Massive Open Online Courses (MOOC)	illimité(en théorie)	asynchrone
Synchronous Massive Online Courses (SMOC)	illimité(en théorie)	synchrone
Small Private Online Courses (SPOC)	limité	asynchrone
Synchronous Small Online Courses (SSOC)	limité	synchrone

TABLE 1.1 : Classification des types de formations en ligne

Notons que notre étude s'intéresse notamment aux environnements d'apprentissage synchrones, avec le support d'un grand nombre d'apprenants (SMOCs).

## 1.2 Communication en temps réel

Les outils de communications en temps réel désignent une catégorie de logiciels qui garantissent le traitement et la transmission instantanée, ou avec un délai fortement négligeable, de l'information. Parmi les protocoles permettant ce type de communication, le plus en vogue reste [Web Real-Time Communication \(WebRTC\)](#).

[WebRTC](#) est un protocole open source de transmission P2P, qui assure la transmission de média (audio, vidéo) et de données brutes, presque sans latence (moins d'une seconde), le tout dans un contexte hautement sécurisé. Il s'agit en réalité, d'une collection de protocoles datant des années 2000. Pour établir une connexion, il faut quatre étapes à savoir la signalisation, la connexion proprement dite, la sécurisation puis la communication.

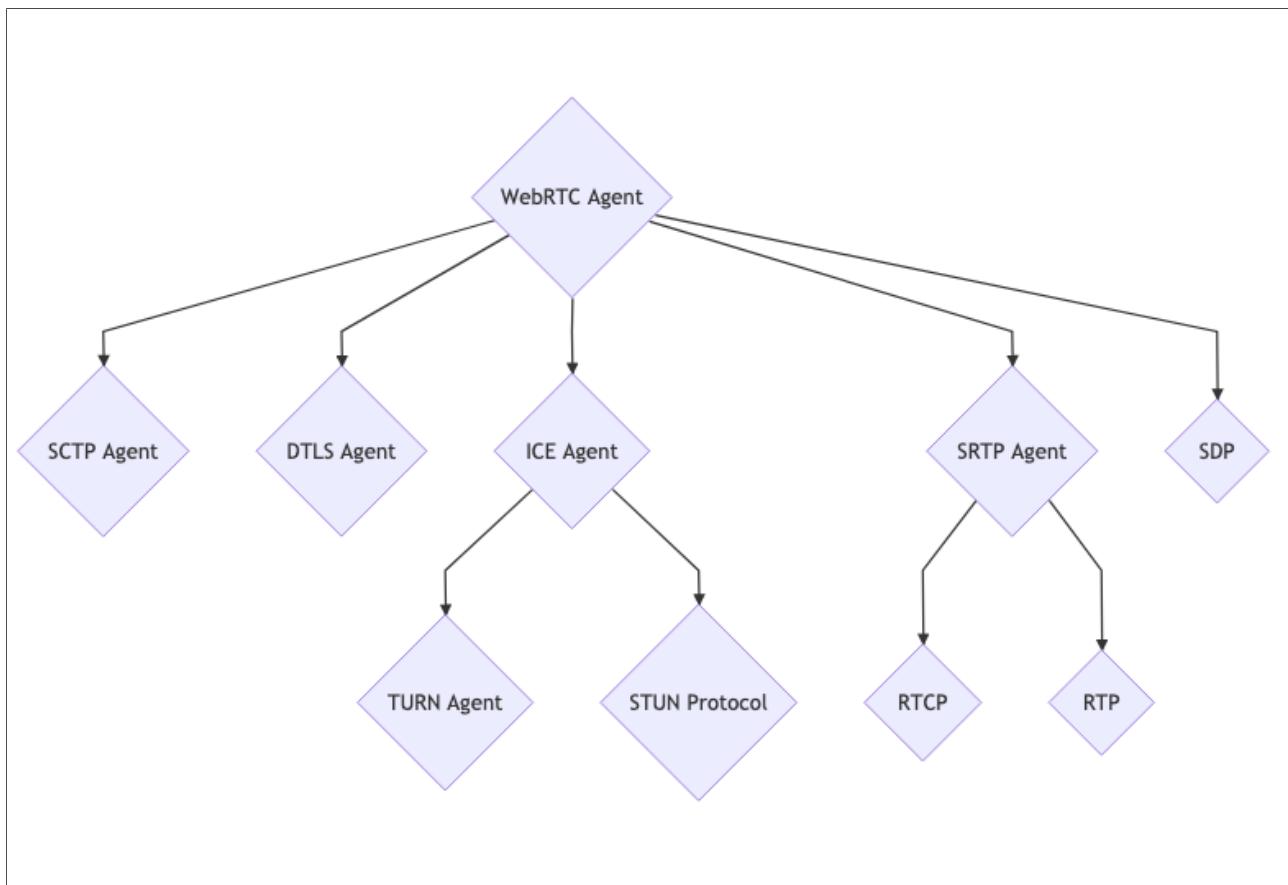


FIGURE 1.1 : Protocoles employés par la technologie WebRTC

La signalisation désigne le processus initial de mise en relation des pairs. Sans ce processus, une machine quelconque n'a aucune idée de qui voudrait bien la contacter. Pour ce faire, le protocole est utilisé et permet la transmission d'informations capitales comme :

- l'adresse et le port de chaque agent **WebRTC** (plusieurs variantes, en réalité)
- les codecs multimédia supportés
- d'autres valeurs comme des certificats de sécurité nécessaires à la mise en place de la connexion et la sécurisation.

A la connexion, les agents **WebRTC** établissent un lien direct entre eux, sans intermédiaire. Face à la multitude de possibilités de connexion (couples constitués de l'adresse et du port), le protocole permet de choisir le meilleur candidat, en faisant recours au serveur et parfois, à un serveur. Le serveur permet la retransmission des données lorsqu'il est impossible pour un agent **WebRTC** d'établir un lien direct avec un autre agent en raison de la configuration réseau (et les types de liaisons possibles [9]).

Pour assurer la sécurité de la connexion, les protocoles offrent une couche de chiffrement pour les contenus multimédia et les paquets brutes.

Enfin, les agents peuvent s'échanger de la donnée, du contenu multimédia, presque sans latence, grâce aux protocoles et .

**WebRTC** est une technologie complexe qui requiert une certaine expertise quant à la connaissance des protocoles, leur utilisation et la mise en œuvre d'applications en temps réel. Elle sert de base aujourd'hui, la plupart des applications de communication en temps réel.

## 1.3 Software as a Service

Parmi les modèles de distribution de logiciels, le SaaS représente une méthode ou le concepteur ou l'entité tenant l'application, l'héberge en ligne et la rend accessible à ses utilisateurs. En terme de commercialisation, il est possible d'offrir un accès à la plateforme moyennant un abonnement ou l'achat d'une version privée pour les besoins des corporations.

## 1.4 Présentation de solutions existantes

Plusieurs solutions s'inscrivent déjà dans le cadre du déroulement de cours en ligne en temps réel. Nous avons choisi quelques unes à passer en revue.

Il est important de préciser que les insuffisances relevées par rapport à ces outils ne sont aucunement d'ordre technique. Nous nous intéressons plutôt aux aspects logistique et financier. En effet, un des objectifs visés est de minimiser l'investissement requis pour la mise en place d'une solution de classe en ligne, tout en éliminant les barrières possibles.

### 1.4.1 Google Classroom

Google Classroom est un outil de la suite Google pour l'éducation. A défaut de disposer d'un module de visioconférence, il s'intègre parfaitement avec Google Meet, à cette fin. L'application offre une version gratuite et dispose d'une interface accessible. Toutefois, pour les réunions en ligne, le nombre maximum de connexions possibles se limite à 500 participants. Pour les entités universitaires dont l'effectif est considérable par classe, ceci pourrait présenter un désavantage. Grâce à la version payante néanmoins, on peut mettre en place un live stream, pour permettre d'accéder au contenu de la réunion sans toutefois pouvoir interagir avec les participants. Mais le modèle de souscription, basé sur le nombre d'utilisateurs risque d'entraîner des frais assez élevés.

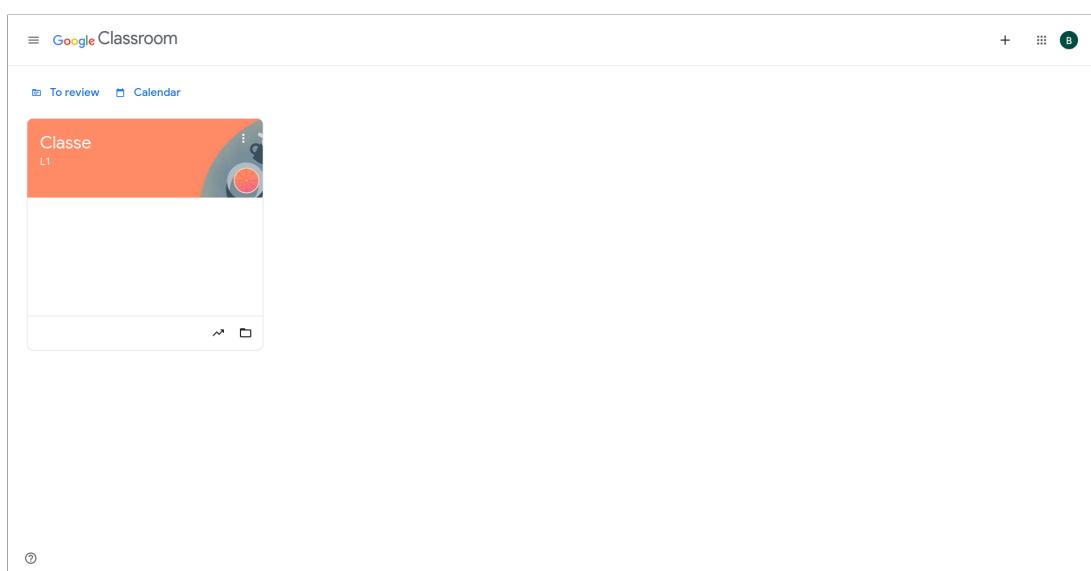


FIGURE 1.2 : Page d'accueil de Google Classroom

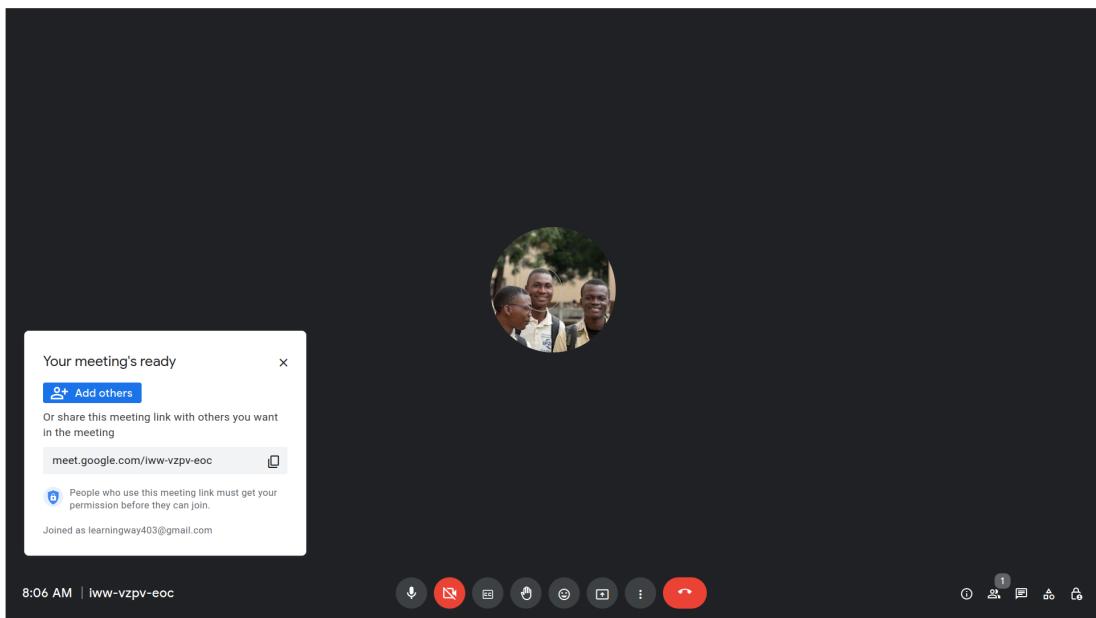


FIGURE 1.3 : Utilisation de Google Meet

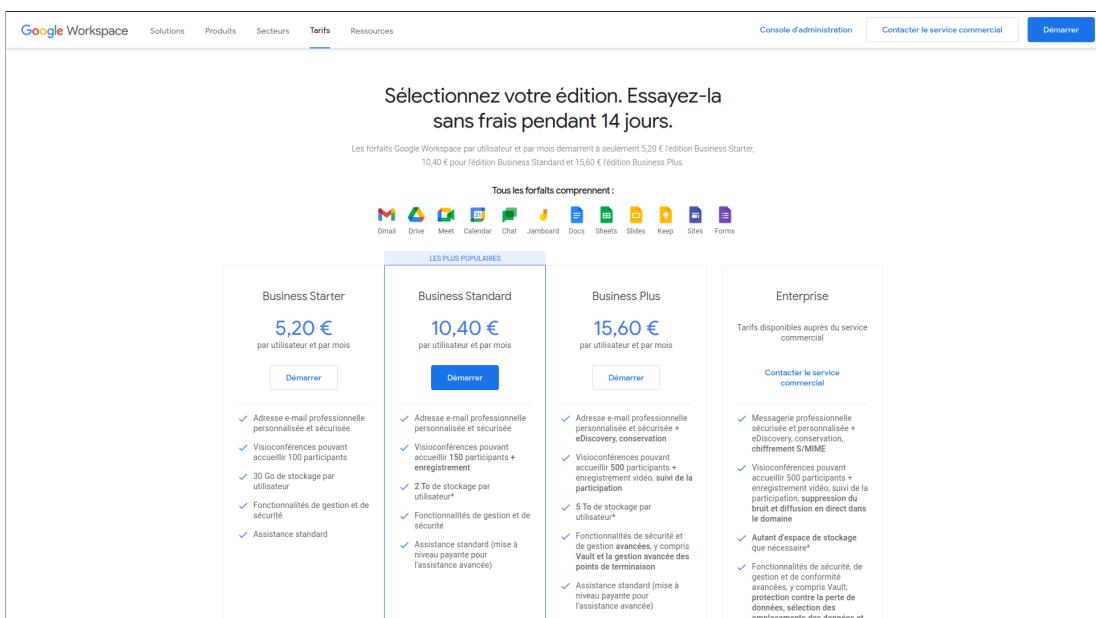


FIGURE 1.4 : Offres de souscription à la suite Google pour éducation

## 1.4.2 Zoom

Zoom est un outil de communication très performant, qui a la capacité de supporter un grand nombre d'utilisateurs. Il dispose de fonctionnalités très utiles pour le déroulement de cours en ligne comme le partage d'écran ou le tableau virtuel. Accéder à ces fonctionnalités dans le cadre d'une utilisation à grande échelle requiert une souscription et les offres de Zoom ne sont pas des plus simples. En effet, Zoom dispose d'un panel large de services associés (figure 1.6) et donc, sans orientation, il est possible de choisir une solution inadéquate en rapport avec le besoin, sans compter la perte financière.



FIGURE 1.5 : Interface de l'application Zoom

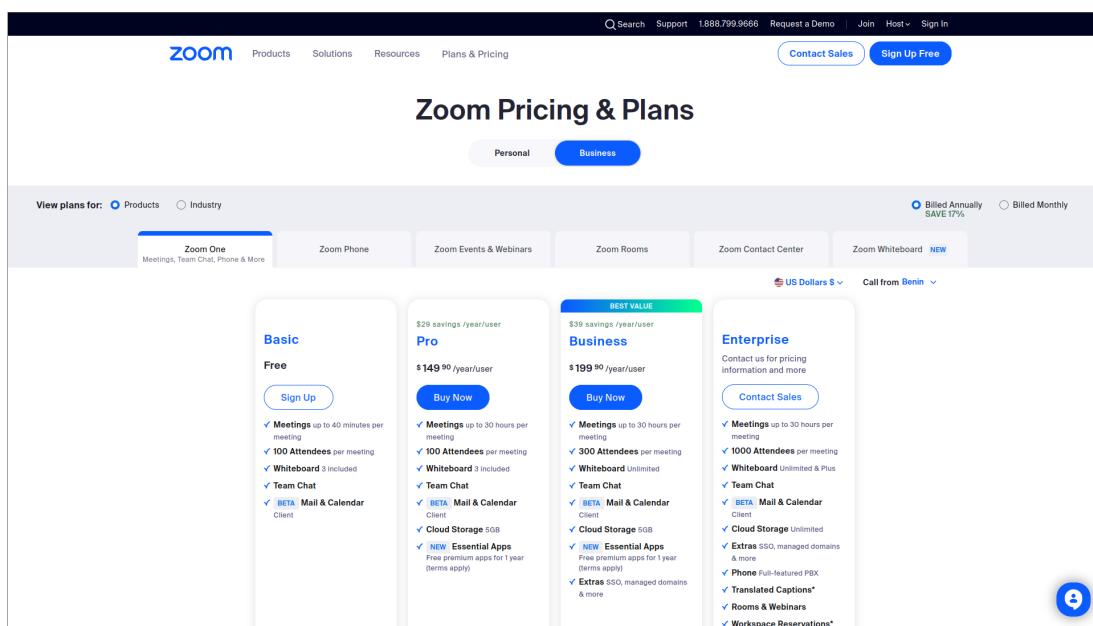


FIGURE 1.6 : Interface de l'application Zoom

### 1.4.3 Moodle

Moodle est un LMS Open Source populaire très connu et utilisé dans les entités de l'enseignement supérieur. Il peut être hébergé ou utilisé en ligne. Il offre un large panel de fonctionnalités et permet l'intégration de divers modules dont des modules de visio-conférence. BigBlueButton (figure 1.8) est une solution Open source employée à cet effet. La mise en place requiert toutefois, une certaine expertise et du matériel spécifique, ce qui en limiterait la portabilité.

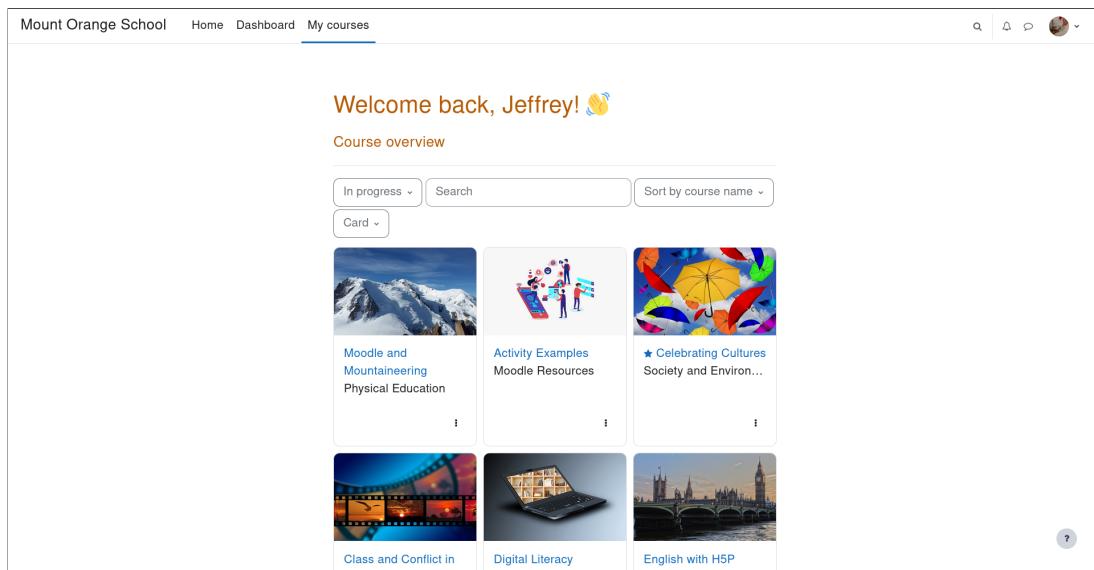


FIGURE 1.7 : Démonstration des capacités de Moodle

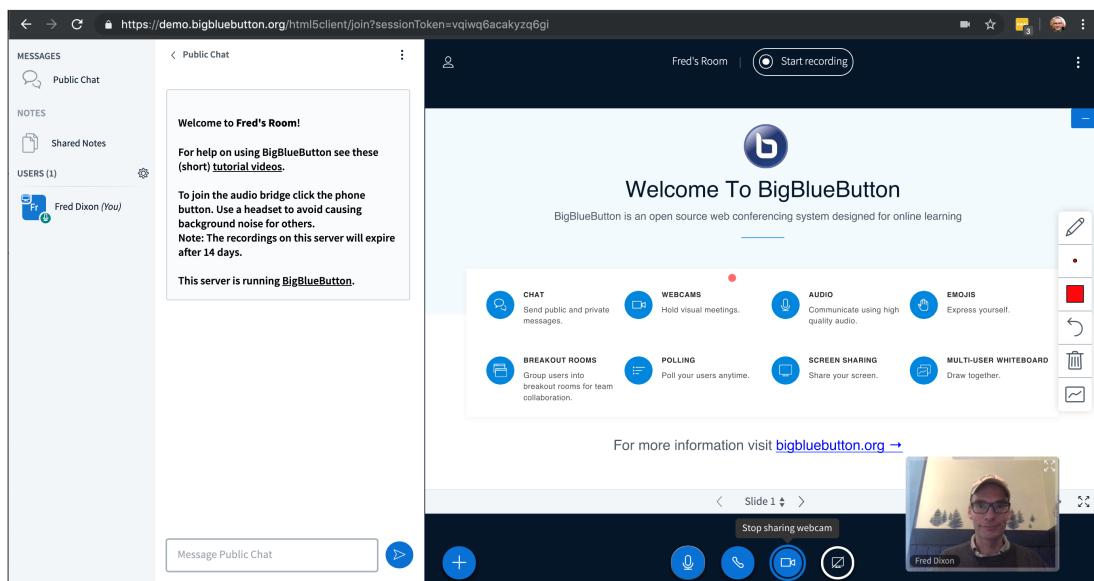


FIGURE 1.8 : Démonstration de BigBlueButton

## Conclusion

Ce chapitre a permis de faire une revue de l'existant et jette les bases des suivants en exposant les concepts clés qui seront développés. Les solutions suscitées conviendraient pour un usage modéré. Elles peuvent s'avérer coûteuses, pour peu qu'elles répondent au besoin. La solution que nous proposons vise à doter les organismes de l'enseignement supérieur, d'un moyen simple mais efficace de tenir les cours en ligne, offrant des outils d'assistance, tout en minimisant les coûts, que cela pourrait engendrer. Pour sa mise en place, il est indispensable d'effectuer une analyse préliminaire dans le but d'identifier les différentes composantes d'un tel système et de faire des choix de conception adaptés. C'est ce à quoi s'attellera le chapitre suivant.

# Matériels et méthodes

## Introduction

Ce chapitre est dédié à la mise en lumière des pratiques d'architecture logicielle employées lors de la conception de notre prototype. Nous y présenterons également les choix techniques effectués.

### 2.1 Méthodes de conception

Dans le souci de décrire de façon fiable, les fonctionnalités du système, nous faisons usage du langage visuel [Unified Modeling Language \(UML\)](#). Il s'agit d'une méthode de visualisation d'architecture logicielle permettant de modéliser l'architecture logicielle d'un système.

Standardisé par [OMG](#), la version actuelle de [UML](#), la 2.5 [2], propose 14 types de diagrammes. N'étant pas une méthode, la norme laisse l'utilisation des diagrammes à l'appréciation des utilisateurs. Dans le cadre de notre prototype, nous avons retenu uniquement les diagrammes de cas d'utilisation, de séquence et de classe, car ils expriment bien la structure de notre application.

#### 2.1.1 Diagramme de cas d'utilisation

Les diagrammes de cas d'utilisation illustrent le comportement fonctionnel du système. Les cas d'utilisation sont utiles pour décrire les interactions possibles entre le/les acteurs acteur(s) et le système.

Les acteurs intervenant dans notre système sont :

- L'étudiant : il dispose d'un accès en lecture aux informations du système ;
- L'enseignant : il dispose d'un accès total en lecture et partiel en écriture sur certaines informations ;
- L'administrateur : il dispose de privilèges élevés pour modifier les informations de la plate-forme et l'administrer.

La figure 2.1 en fait l'illustration.

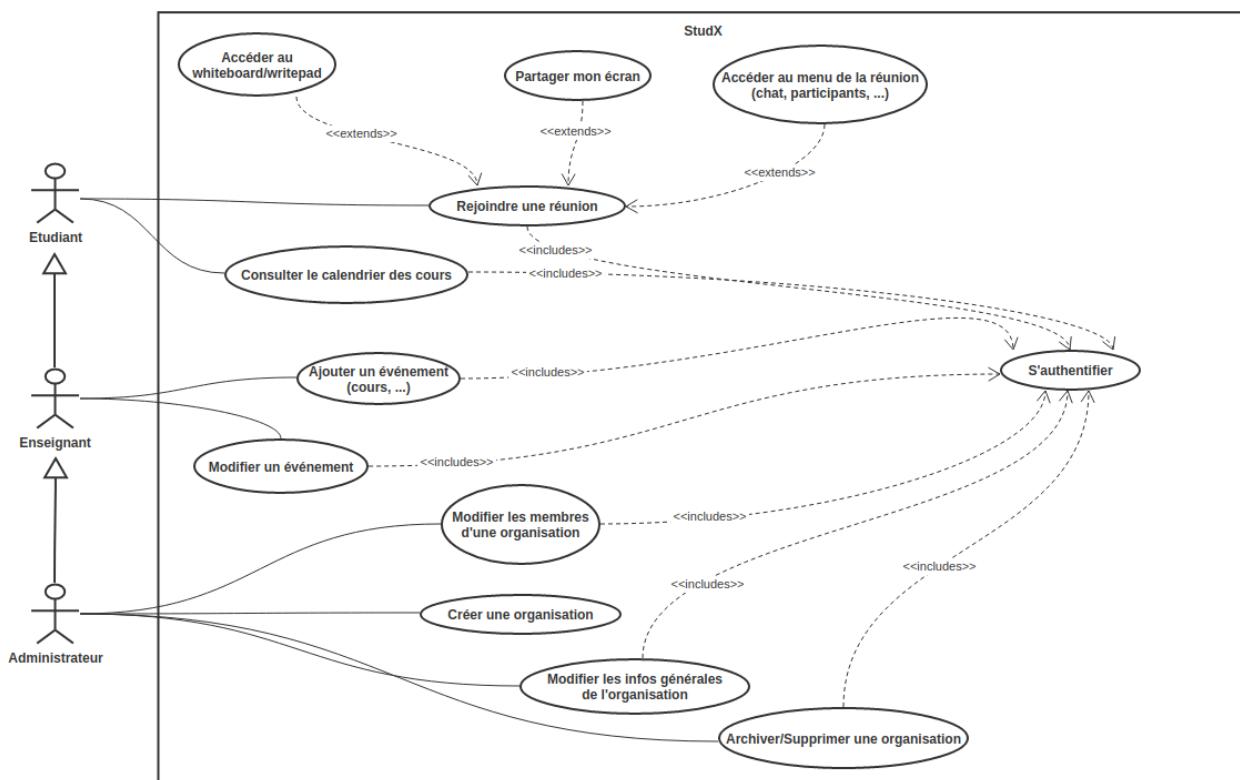


FIGURE 2.1 : Diagramme de cas d'utilisation du prototype StudX

Outre le diagramme, il s'avère parfois nécessaire de fournir, en plus, des descriptions textuelles des cas d'utilisation, dans le but d'apporter plus d'éclaircissements. Ci-dessous, sont présentées les descriptions textuelles des cas “Ajouter un événement” et “Rejoindre une réunion”.

### 2.1.1.1 Description textuelle du cas d'utilisation “Ajouter un événement”

**Titre :** Ajouter un événement

**Objectif :** Planifier les cours en ligne en ajoutant des événements au calendrier

**Acteurs :** Enseignant ou Administrateur

**Pré-conditions :**

- l'utilisateur est authentifié en tant qu'enseignant ou administrateur.

**Séquence nominale :**

1. L'utilisateur accède au calendrier ;
2. Le système renvoie les événements actuellement programmés ;
3. L'utilisateur remplit et soumet un formulaire de création ;
4. Le système enregistre l'événement et les détails associés ;
5. Le système notifie les participants concernés.

**Post-conditions :**

- L'utilisateur accède à l'événement dans son calendrier.

### 2.1.1.2 Description textuelle du cas d'utilisation “Rejoindre une réunion”

**Titre :** Rejoindre une réunion

**Objectif :** Tenir une session de classe en ligne

**Acteurs :** Étudiant ou Enseignant ou Administrateur

**Pré-conditions :**

- l'utilisateur est authentifié.

**Séquence nominale :**

1. l'utilisateur consulte le calendrier;
2. le système affiche les divers événements programmés;
3. l'utilisateur accède aux détails d'un événement;
4. l'utilisateur clique sur le lien pour rejoindre la réunion;
5. le système connecte l'utilisateur aux participants présents.

**Post-conditions :**

- L'utilisateur est en mesure d'interagir, de communiquer avec les participants.

### 2.1.2 Diagramme de séquence

Le diagramme de séquence décrit les interactions, dans l'espace temps, entre objets dans le cadre des scénarios évoqués au niveau des cas d'utilisations. Les figures 2.2 et 2.3 illustrent les diagrammes de séquences pour les deux cas d'utilisation suscités.

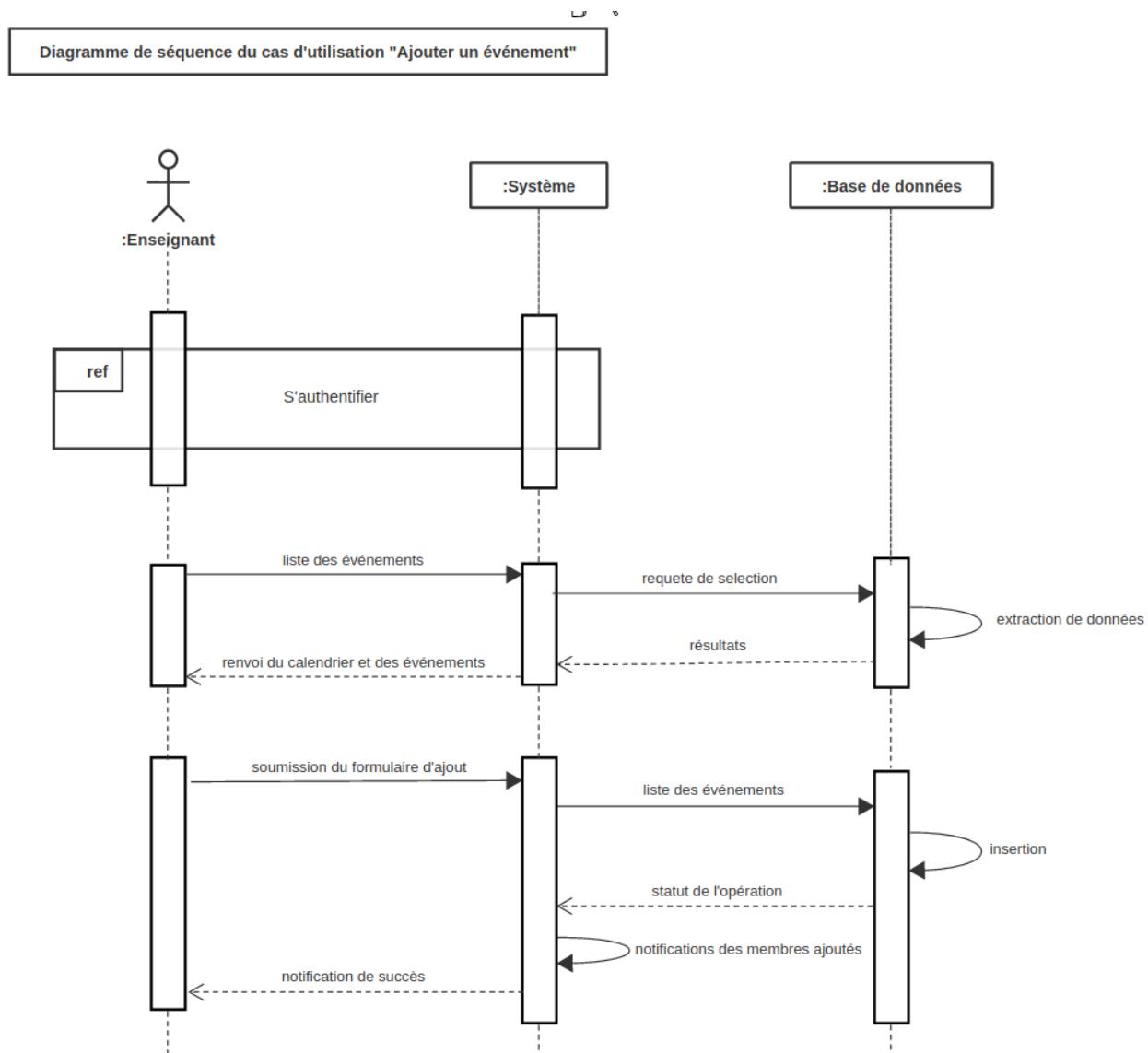


FIGURE 2.2 : Diagramme de séquence pour le cas d'utilisation “Ajouter un événement”

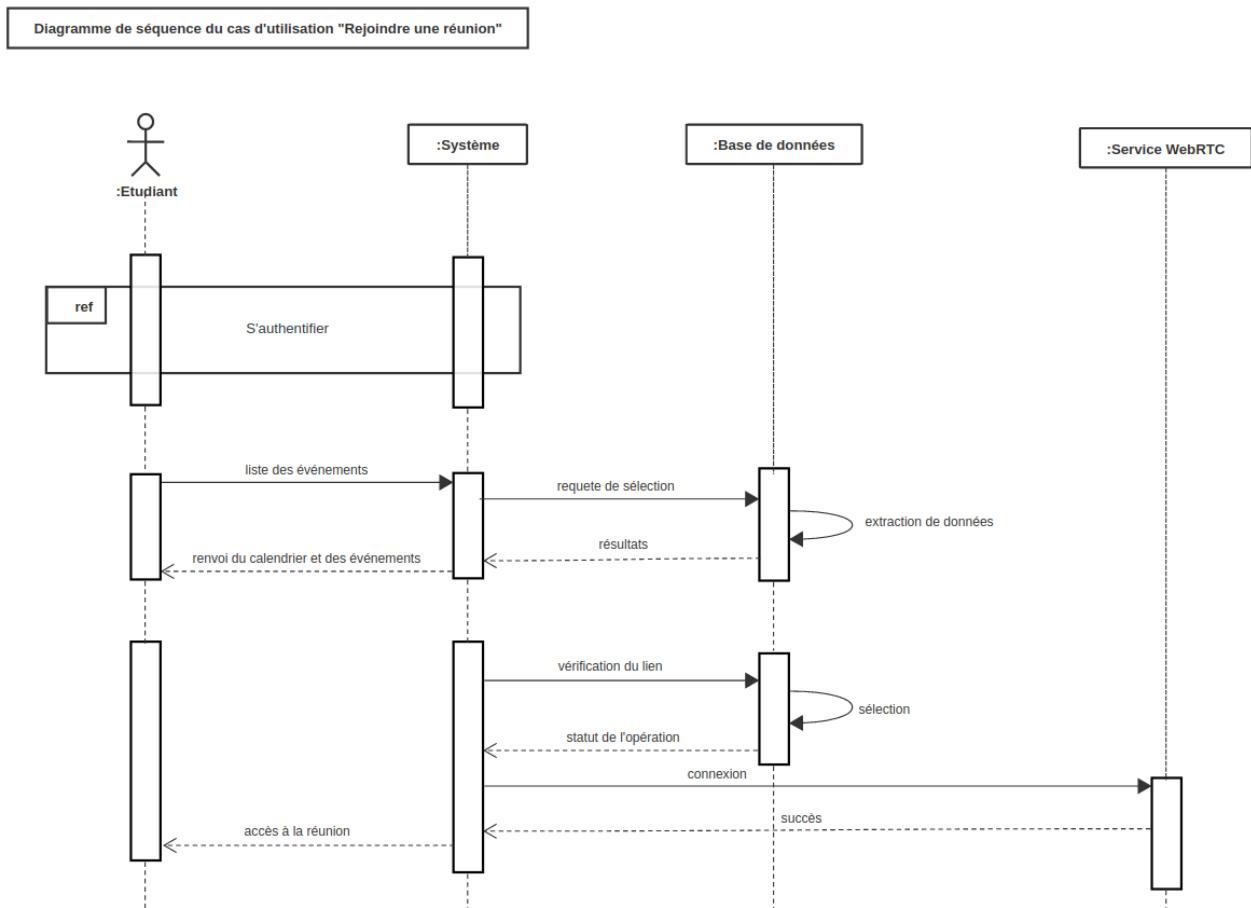


FIGURE 2.3 : Diagramme de séquence pour le cas d'utilisation “Rejoindre une réunion”

### 2.1.3 Diagramme de classe

Le diagramme de classe illustre les classes et les interfaces du système ainsi que les relations qui les lient. Le diagramme à la figure 2.4 en dessous décrit les diverses entités de notre prototype.

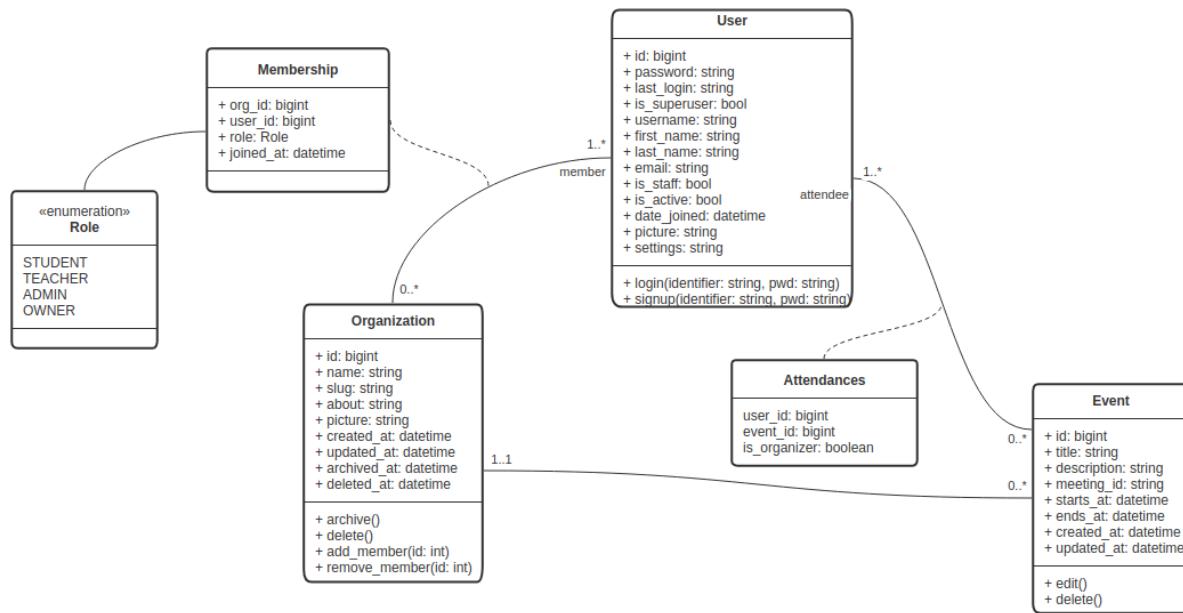


FIGURE 2.4 : Diagramme de classe

### 2.1.4 Architecture du système

Pour assurer la scalabilité des systèmes, il est important de bien en concevoir l'architecture. Dans ce but, nous avons adopté une approche découpée, isolant les composantes du système. Il s'agit de microservices. Toutefois, il est important de noter qu'à l'échelle d'un prototype, l'architecture proposée reste très simplifiée et ne prend pas en compte des facteurs comme la résilience [7]. La figure de l'encart 2.5 présente les diverses composantes de notre architecture.

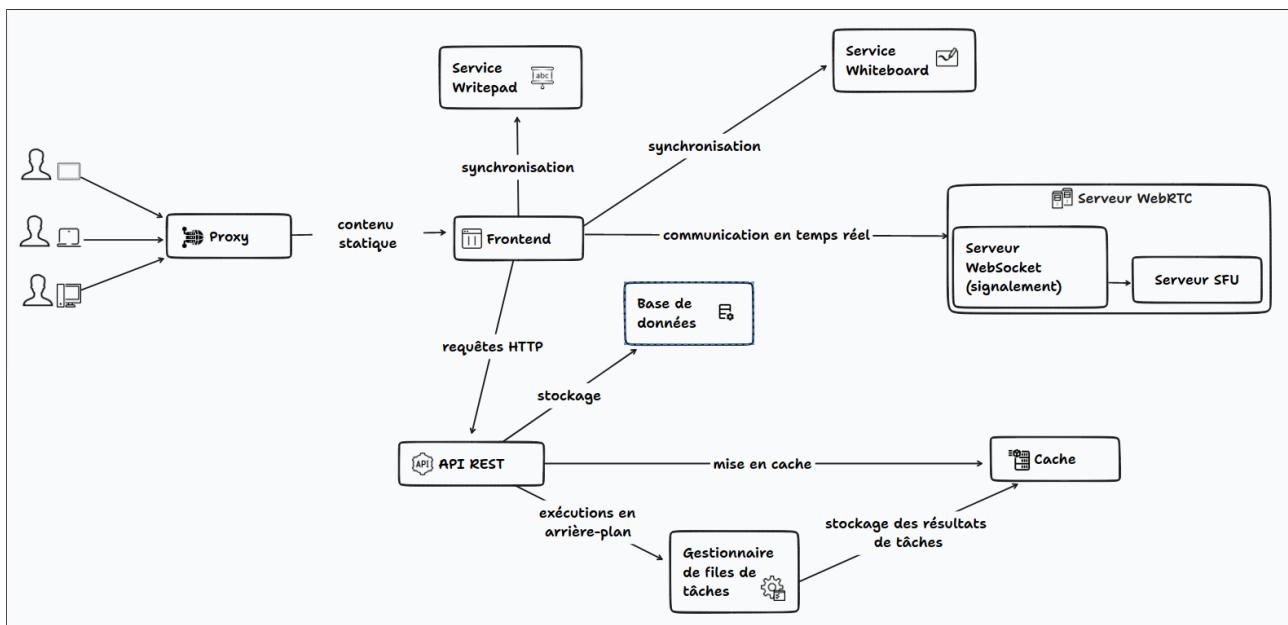


FIGURE 2.5 : Architecture du système du prototype StudX

On peut notamment remarquer que toutes les interactions entre le système et les utilisateurs passent toutes par un proxy. Ceci s'explique principalement par la volonté d'éviter les problèmes

de CORS, qui occurrent dès lors que les services ne sont pas tous sur un même domaine.

Outre ces détails, il faut préciser également que le prototype implémente l'architecture de multi-entité [14], en regroupant les utilisateurs et les données qui leur sont communes en entités que nous qualifions d'organisation. Le but est de pouvoir servir plusieurs regroupements sans pour autant avoir à répliquer le matériel.

## 2.2 Matériels

### 2.2.1 Choix techniques

Faisant référence à l'architecture suscitée, voyons à présent les technologies employées dans la mise en place de la solution.

#### 2.2.1.1 Proxy

Un serveur proxy sert de relais entre différentes parties, notamment entre le client et le serveur dans notre contexte. Il s'agit dans ce cas, d'un reverse proxy, car le relai va du client vers le serveur.

**Caddy** est un serveur Web moderne qui offre un large panel de fonctionnalités. Il offre un large éventail de fonctionnalités que l'on peut mettre en place via un fichier de configuration spéciale nommé Caddyfile. La figure 2.6 en présente un exemple, extrait du code de notre prototype.

	File: <b>Caddyfile</b>
1	192.168.122.1:8081 {
2	reverse_proxy http://localhost:8080
3	}
4	
5	192.168.122.1:9091 {
6	reverse_proxy http://localhost:9090
7	}
8	
9	192.168.122.1:9000 {
10	encode gzip
11	
12	handle /ws/* {
13	reverse_proxy http://localhost:5000
14	}
15	
16	handle /api* {
17	reverse_proxy http://localhost:6000
18	}
19	
20	handle_path /media* {
21	root * ./api/media
22	file_server
23	}
24	

FIGURE 2.6 : Example de configuration du proxy **Caddy**

On peut remarquer entre autres, l'utilisation de variables d'environnement qui permettent de rendre la configuration encore plus dynamique. Tous ces atouts en font un bon choix pour notre prototype.

### 2.2.1.2 API REST

Une [API REST](#) définit interface de programmation respectant les contraintes du style d'architecture [Representational State Transfer](#).

Elle doit disposer des caractéristiques suivantes :

- Une architecture client-serveur constituée de clients, de serveurs et de ressources, avec des requêtes gérées via [HTTP](#) ;
- Des communications client-serveur sans état, c'est-à-dire que les informations du client ne sont jamais stockées entre les requêtes, qui doivent être traitées séparément, de manière totalement indépendante ;
- La possibilité de mettre en cache des données afin de rationaliser les interactions client-serveur ;
- Une interface uniforme entre les composants qui permet un transfert standardisé des informations ;
- Un système à couches, invisible pour le client, qui permet de hiérarchiser les différents types de serveurs (pour la sécurité, l'équilibrage de charge, etc.) impliqués dans la récupération des informations demandées ;
- Du code à la demande (facultatif), c'est-à-dire la possibilité d'envoyer du code exécutable depuis le serveur vers le client (lorsqu'il le demande) afin d'étendre les fonctionnalités d'un client.[\[8\]](#)

Pour ce faire, notre choix s'est porté sur **Django**, un framework du langage **Python** offrant une facilité de conception grâce aux nombreuses fonctionnalités déjà incluses par défaut. Avec l'emploi de modules comme **Django Rest Framework**, il est possible de concevoir une [API](#) totalement conforme aux recommandations de la spécification [REST](#).



FIGURE 2.7 : Logo du framework **Django**

### 2.2.1.3 Frontend

La conception de l'interface utilisateur a nécessité l'usage des langages [HTML](#), [CSS](#) et [TypeScript](#).

[HTML](#) est un langage de balisage standardisé, qui permet la conception de documents Web. Assisté du langage de style [CSS](#), il est possible de concevoir une mise en page attrayante favorisant l'expérience utilisateur.

[TypeScript](#) est un langage conçu au-dessus du langage [JavaScript](#). Il vise notamment à améliorer ce dernier en fournissant un système de typage fort. Cela permet entre autres de réduire le nombre de bugs qui finissent en production et d'améliorer l'expérience du développeur.

Afin de faciliter l'intégration de tous les outils suscités, nous avons fait recours au framework [Vue.js](#). [Vue](#) est un framework moderne de conception d'application Web qui se concentre sur le rendu déclaratif et composition de composants. Des solutions complémentaires maintenues officiellement, permettent la gestion du routage, de l'état et bien d'autres fonctionnalités comme les [PWAs](#).

Au vu des avantages qu'il présente, il correspond parfaitement aux besoins de notre plateforme en termes d'interfaces.



FIGURE 2.8 : Logo du framework **Vue.js**

#### 2.2.1.4 Base de données

Les systèmes de gestion de bases de données sont des éléments clés dans la conception d'applications dynamiques. Elles permettent le stockage, le filtrage, la mise à jour et la suppression des données du système. On distingue généralement deux grandes familles de base de données : les bases relationnelles et les bases non relationnelles. La première préconise l'utilisation d'un schéma fixe représentant la structure de la donnée alors que la seconde permet une flexibilité du schéma et autorise l'insertion de colonnes quelconques.

Notre choix s'est porté vers PostgreSQL, un système de gestion de base de données relationnel. C'est d'ailleurs, le seul système Open Source, fournissant des fonctionnalités dignes de concurrencer les maisons d'édition comme Oracle.



FIGURE 2.9 : Logo de **PostgreSQL**

#### 2.2.1.5 Gestionnaire de files de tâches

Effectuer des tâches qui demandent des ressources intensives lors de requêtes [HTTP](#), risque d'en dégrader les performances. Pour éviter cela, nous avons recours à **Celery**, un gestionnaire de files de tâches moderne qui offre une intégration quasi-parfaite avec le framework **Django**.

**Celery** supporte un large panel de protocoles de communication pour la planification des tâches et la récupération des résultats. Pour simplifier l'architecture, nous nous sommes servi du cache comme relai afin de déclencher des tâches.

FIGURE 2.10 : Logo de **Celery**

### 2.2.1.6 Cache

En architecture logicielle, le cache est une composante essentielle. Il permet d'améliorer les performances du système en gardant une copie (en mémoire, à court terme) des données auxquelles les utilisateurs ont précédemment accédé, sans qu'il n'y ait besoin de reprendre le même processus de traitement de la requête. Ceci réduit le temps de réponse mais aussi réduit la pression sur toute l'infrastructure. La base de données est moins sollicitée par exemple. Nous avons opté pour Redis, une solution Open Source qui utilise la mémoire vive de la machine pour permettre un accès en lecture et en écriture très très rapide.

FIGURE 2.11 : Logo de **Redis**

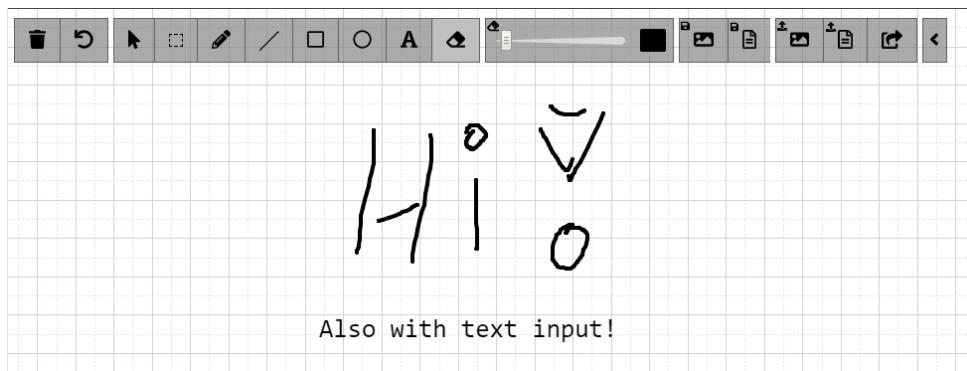
### 2.2.1.7 Service Writepad

Dans le cadre de la conception d'un service synchronisé d'écriture, nous avons employé l'éditeur populaire **TipTap**. Il s'agit d'un éditeur Open Source offrant un large panel de fonctionnalités dont la collaboration entre utilisateurs. Grâce à une documentation extensive, il est d'autant plus facile de l'intégrer à une application.

FIGURE 2.12 : Logo de **TipTap** et version actuelle

### 2.2.1.8 Service Whiteboard

Le service Writepad est le clone d'un projet Open source sous licence MIT, dénommé **whiteboard**. Il est consultable à l'adresse URL suivante : <https://github.com/cracker0dks/whiteboard>. Les modifications effectuées touchent principalement l'interface utilisateur mais aussi font omission de fonctionnalités qui ne nous intéressent pas à l'état actuel du prototype. La figure de l'encart 2.13 en présente l'interface par défaut.

FIGURE 2.13 : Interface par défaut de **whiteboard**

### 2.2.1.9 Serveur WebRTC

**Rust** est un langage compilé qui se veut performant, sûr et productif[6]. Le langage peut notamment donner des garanties d'absence d'erreur de segmentation ou de situation de concurrence dès l'étape de compilation. De plus, ceci se fait sans ramasse-miettes. Ses performances sont comparables à celles du C ou du C++ pour ce qui concerne la vitesse d'exécution. Tout cela en fait un choix excellent pour le développement d'applications de réseau. Parmi les solutions Open Source pour la mise en place d'un serveur [WebRTC](#), nous avons opté pour **Mediasoup**. C'est un serveur de relai qui offre une bibliothèque **Rust** pour l'intégration dans diverses catégories d'applications.

**Actix** est un framework reposant sur le modèle d'acteurs [13] et écrit en **Rust**. Il offre un framework web (**actix-web**) qui permet d'implémenter des services web reposant sur le modèle d'acteurs. C'est un framework toutes batteries incluses qui supporte nativement bien de fonctionnalités comme le support des Websockets, lu protocole [TLS](#) ou encore de la version 2 du protocole [HTTP](#) (HTTP/2). Il offre d'excellentes performances, raison pour laquelle nous l'avons choisi, pour la gestion du signalement lors des connexions [WebRTC](#).

Ci-dessous est un extrait du code source de notre prototype **StudX**, qui présente la syntaxe du langage **Rust**, et montre l'intégration des divers outils suscités :

```
mod webrtc;
mod websocket;

use actix::Actor;
use actix_web::{middleware::Logger, web, App, HttpServer};
use dotenv::dotenv;
use mediasoup::prelude::*;
use std::{env, net};
use tracing::{info, Level};

#[actix_web::main]
async fn main() -> std::io::Result<()> {
    dotenv().ok();
    tracing_subscriber::fmt()
        .compact()
        .with_max_level(Level::INFO)
        .init();
}
```

```

let address = env::var("ADDRESS")
    .expect("ADDRESS env var is not set")
    .parse::<net::SocketAddr>()
    .expect("ADDRESS is invalid");

// Application State
let redis_url = env::var("REDIS_URL")
    .expect("REDIS_URL env var is not set");
let redis = redis::Client::open(redis_url)
    .expect("Unable to connect to Redis");

let server = websocket::Server::new(redis.clone()).start();
let worker_manager = WorkerManager::new();
let registry = webrtc::registry::Registry::default();
let transport_ips = {
    let ip = env::var("MEDIASOUP_IP")
        .expect("MEDIASOUP_IP is not set")
        .parse::<net::IpAddr>()
        .expect("MEDIASOUP_IP is invalid");
    let announced_ip = env::var("MEDIASOUP_ANNOUNCED_IP").ok().map(|ip| {
        ip.parse::<net::IpAddr>()
            .expect("MEDIASOUP_ANNOUNCED_IP is invalid")
    });
    (ip, announced_ip)
};

info!("Server is starting on address: {}", address);

HttpServer::new(move || {
    App::new()
        .app_data(web::Data::new(server.clone()))
        .app_data(web::Data::new(worker_manager.clone()))
        .app_data(web::Data::new(registry.clone()))
        .app_data(web::Data::new(redis.clone()))
        .app_data(web::Data::new(transport_ips.clone()))
        .wrap(Logger::default())
        .service(web::scope("/ws").service(websocket::room))
})
.bind(address)?
.run()
.await
}

```

## 2.2.2 Outils de développement

Le tableau 2.1 présente une liste non exhaustive des outils de développement employés pour la réalisation du prototype.

Outils de développement		
Matériel		
Nom	Description	
Laptop Acer ES1-521	Employé pour les tests de la solution	
Laptop Lenovo Thinkbook	Employé pour le développement et les tests	
Smartphone TECNO Spark 8C	Smartphone pour les besoins de tests d'accessibilité	
Routeur ZTE MF927U	Mise en réseau des appareils	
Systèmes d'exploitation		
Nom	Version	
Manjaro Linux	22.0.4	
Fedora	36	
Windows	10	
Logiciels		
Nom	Description	Version
EDI Jetbrains	Suite de développement logiciel	Versions Pro de Pycharm, CLion et WebStorm
neovim	Editeur de texte modal	0.8
Git, GitHub	Versionnement de code source	git : 2.39.2
Docker	Outil de conteneurisation d'applications	23.0.1

TABLE 2.1 : Liste non exhaustive des outils employés dans la réalisation de **StudX**

## Conclusion

Ce chapitre a permis de passer en revue les choix de conception ainsi que les choix techniques effectués pour la mise en œuvre de notre prototype d'application. Ceci pose des fondations robustes à

l'implémentation de ladite solution. En suivant les décisions techniques prises, nous avons construit notre prototype. Le but du chapitre suivant sera de le présenter puis de faire un bilan d'évaluation.

# Résultats et Discussion

## Introduction

Ce chapitre s'attelle à la présentation du prototype de StudX, l'application de communication en temps réel que nous proposons. Nous en présenterons les diverses fonctionnalités accompagnées de capture d'écran. Puis, au travers d'une discussion, nous en présenterons les limites, les contraintes et les possibilités d'expansion.

### 3.1 Résultats

#### 3.1.1 Authentification

L'accès à l'application est subordonné à l'authentification de l'utilisateur. La figure 3.1 en présente l'interface. Elle offre la possibilité de se connecter ou de s'inscrire.

The screenshot shows the login interface for StudX. At the top, it says "Log into your account." Below that, a sub-instruction reads "Track your class and your time easily." The form itself has two input fields: "ID \*" containing "13368720" and "Password \*" containing "(\*3jkjdN@dj)". A large purple "Sign In" button is positioned below these fields. A horizontal line with the word "or" in the center separates the sign-in area from a "Sign Up" button, which is enclosed in a rounded rectangle.

FIGURE 3.1 : Page d'authentification de **StudX**

### 3.1.2 Calendrier

Après authentification, l'utilisateur accède au calendrier des divers événements planifiés. Il lui est possible de réduire ou d'étendre la vue au jour actuel, aux semaines ou encore aux mois. S'il s'agit d'un administrateur ou d'un enseignant, il peut en ajouter de nouveaux. La figure 3.2 présente le calendrier, qui présente tous les programmes du mois courant.

FIGURE 3.2 : Calendrier des planifications

S'il dispose des permissions nécessaires, l'utilisateur peut ajouter un événement au calendrier en suivant le formulaire que montre la figure 3.3.

FIGURE 3.3 : Calendrier des planifications

Il est possible d'associer à l'événement un lien d'accès à la session de conférence en ligne. Pour y accéder par la suite, les utilisateurs peuvent consulter les détails dudit événement (figure 3.4).

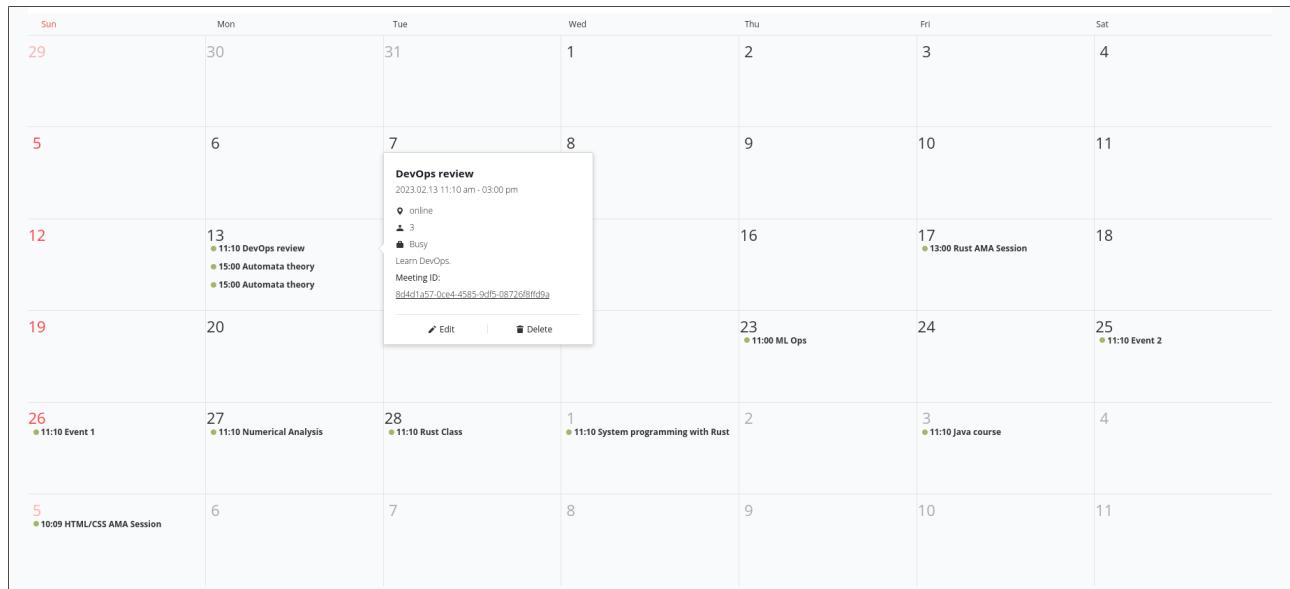
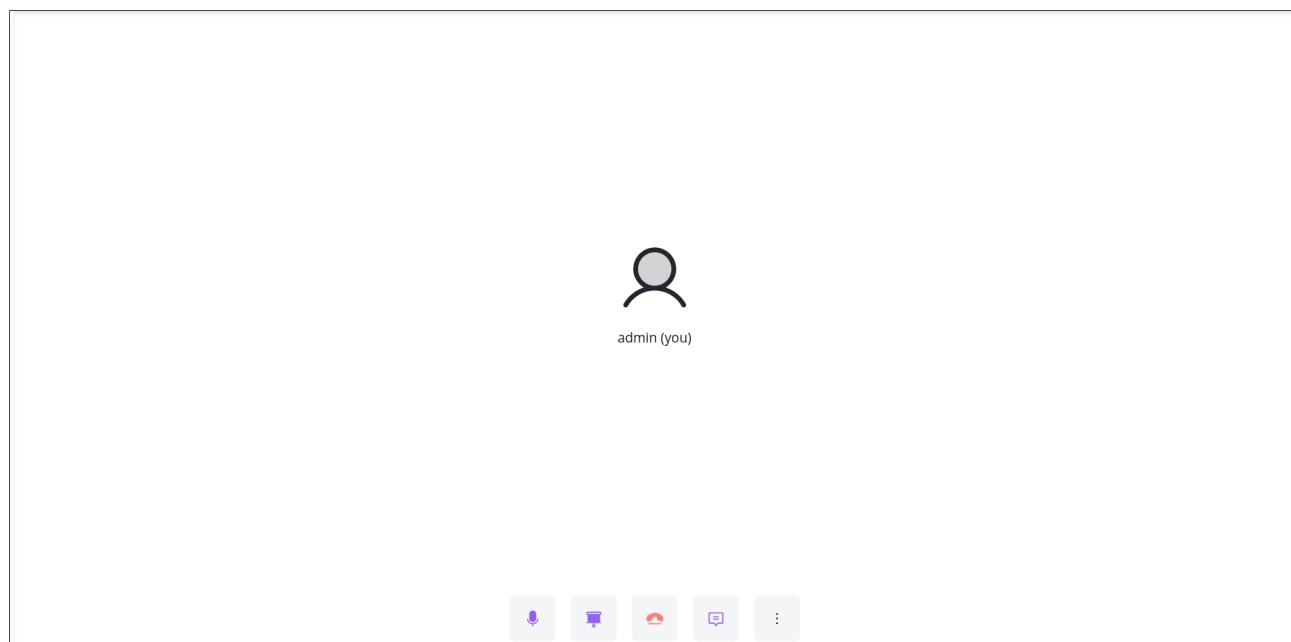


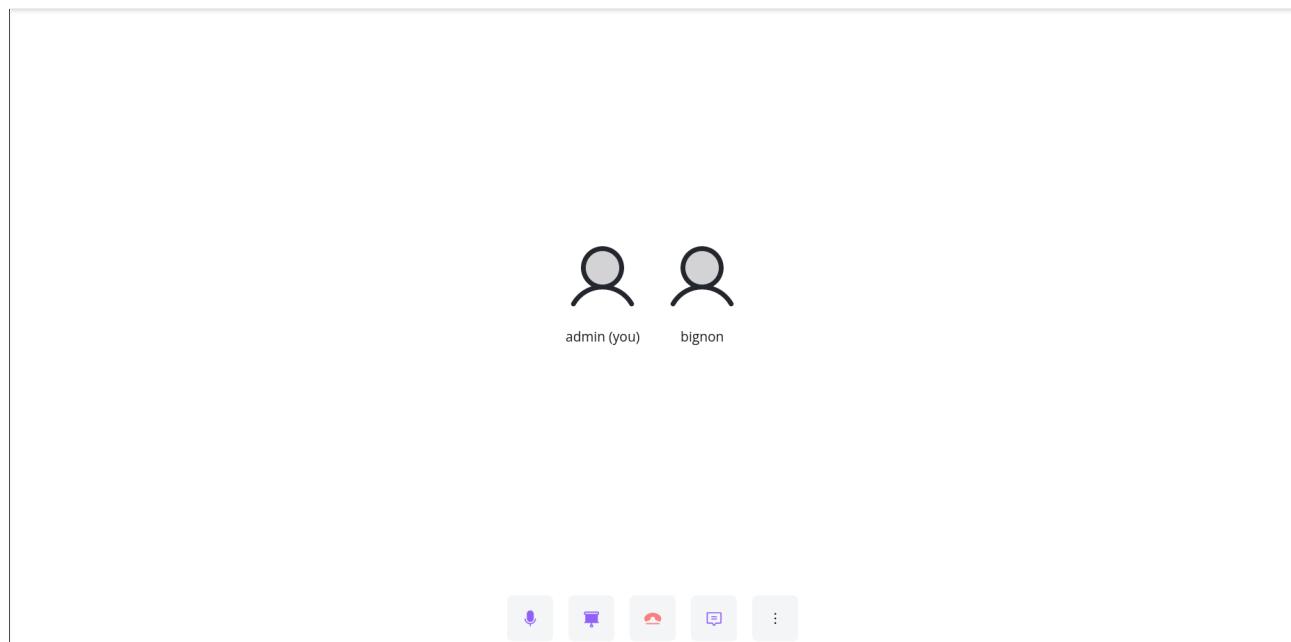
FIGURE 3.4 : Détails d'un événement

### 3.1.3 Sessions en ligne

Les événements incluant un lien donnent accès à une session en ligne que peuvent rejoindre tous les participants disposant du lien. La figure 3.5 présente à quoi ressemble l'interface par défaut, avec la grille des participants et les options de contrôle. Outre la voix, les participants ont la possibilité d'interagir entre eux via des messages écrits (figure 3.6).



(a) Un participant



(b) Deux participants

FIGURE 3.5 : Grille des participants et boutons de contrôle.

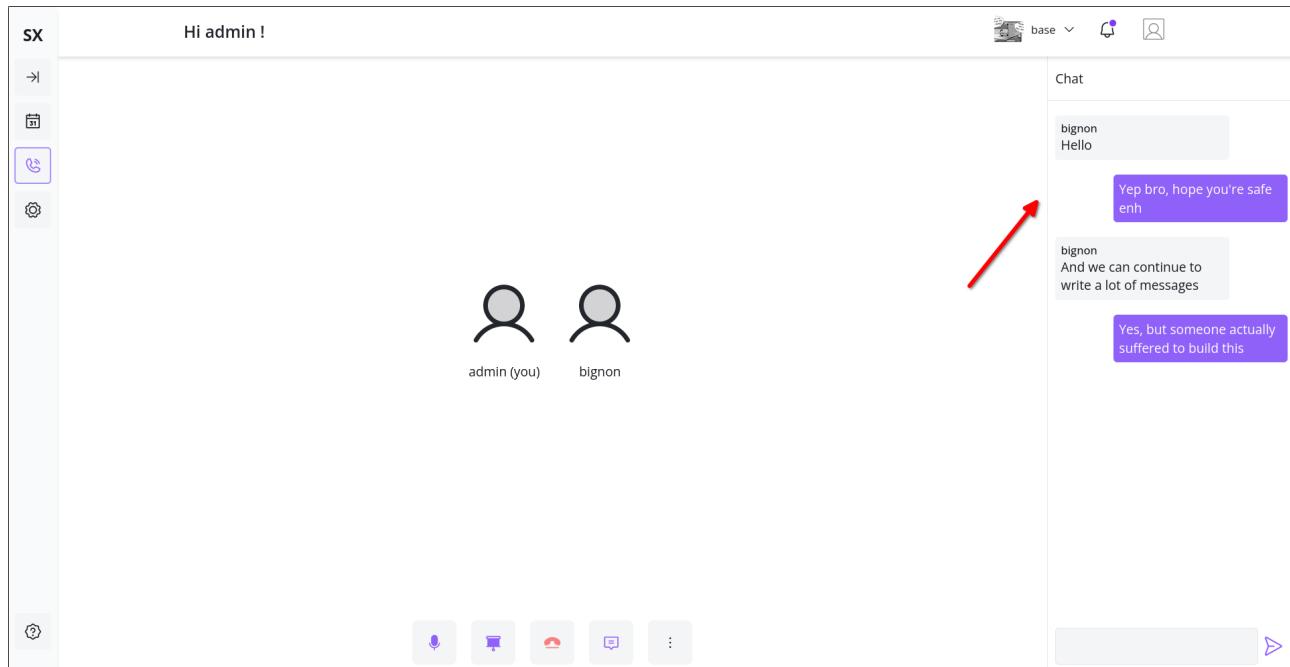
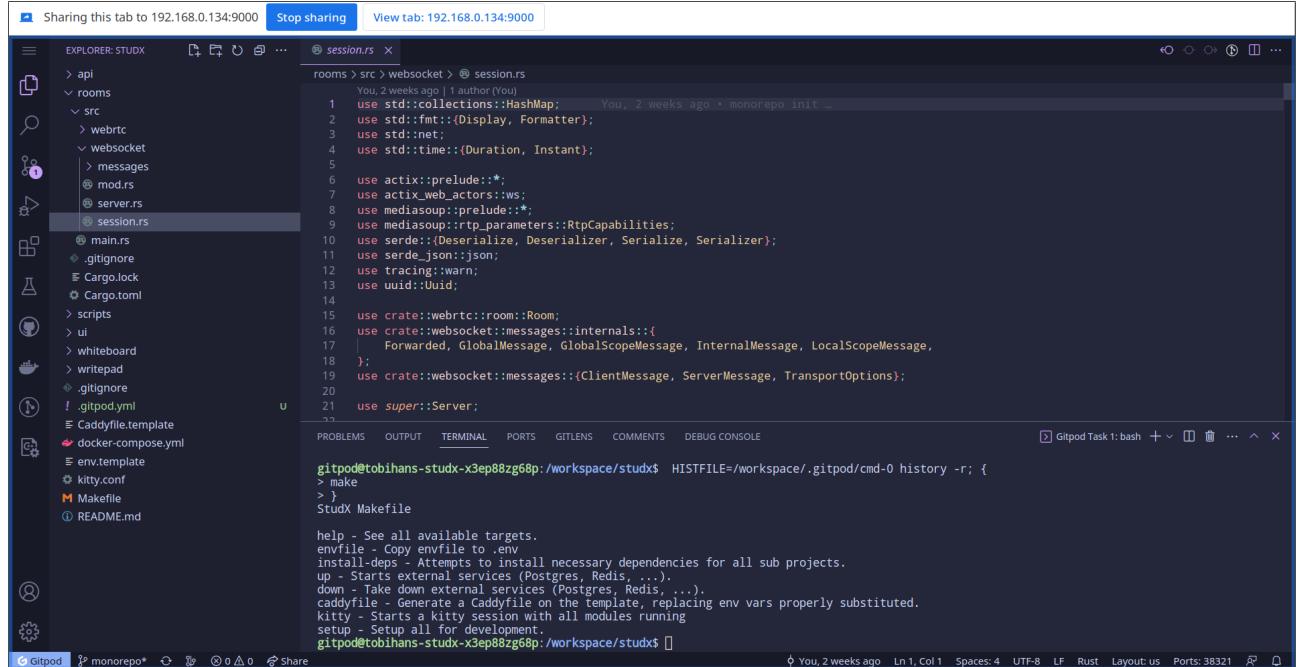
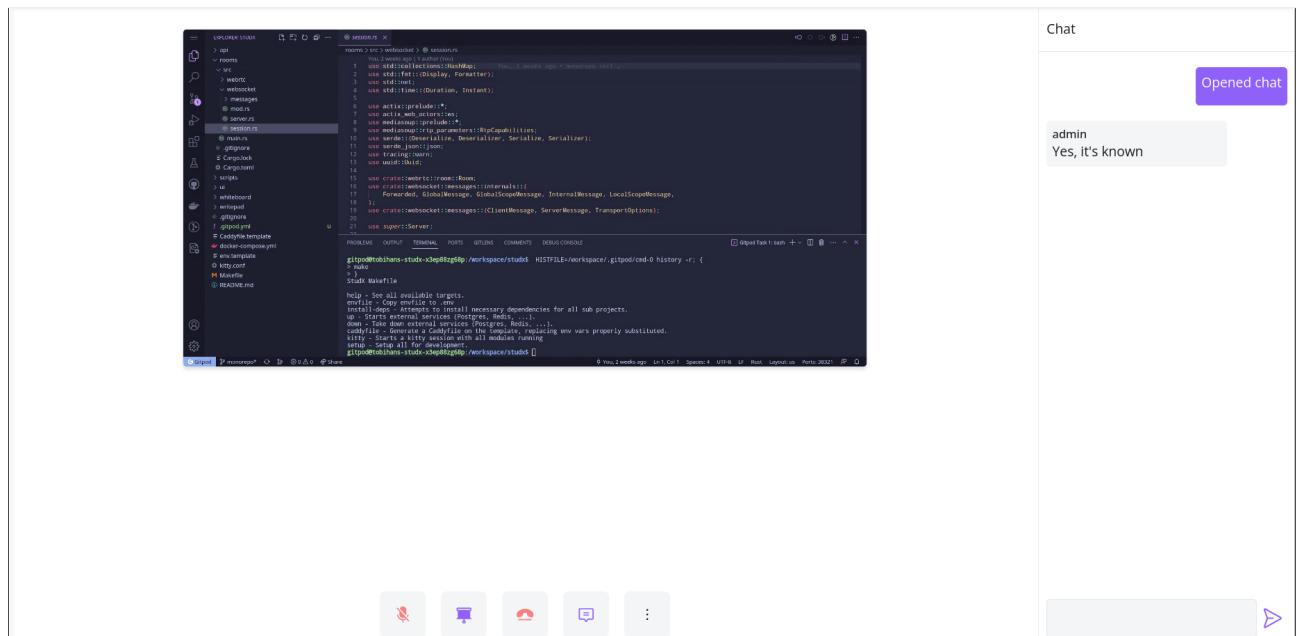


FIGURE 3.6 : Messagerie instantanée intégrée à **StudX**

Plusieurs autres fonctionnalités sont exploitables. L'une d'elles est le partage d'écran. Pour illustrer, nous nous sommes servis de deux appareils avec l'un faisant le partage, comme le montre la figure 3.7.



(a) Participant faisant un partage d'écran.



(b) Visualisation du partage d'écran

FIGURE 3.7 : Partage d'écran.

Les participants disposent également d'un whiteboard, c'est-à- dire un tableau virtuel, pour effectuer des illustrations. Le contenu est synchronisé entre tous les participants. La figure 3.8 fait une démonstration de ladite fonction.

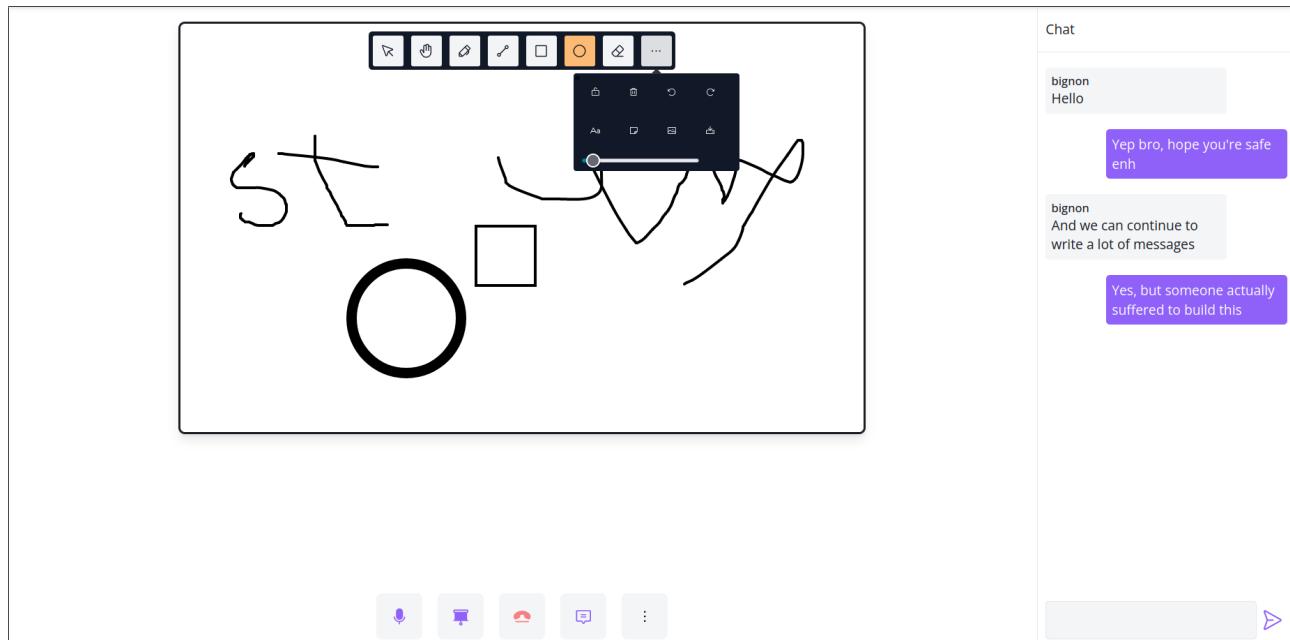


FIGURE 3.8 : Tableau virtuel

On peut également percevoir sur l'image, les modifications apportées au projet Open Source qui a servi de base au développement de cette fonctionnalité. L'application dispose également d'un dispositif de notes intégré, que nous qualifions de **Writepad**. La figure 3.9 en fait la présentation.

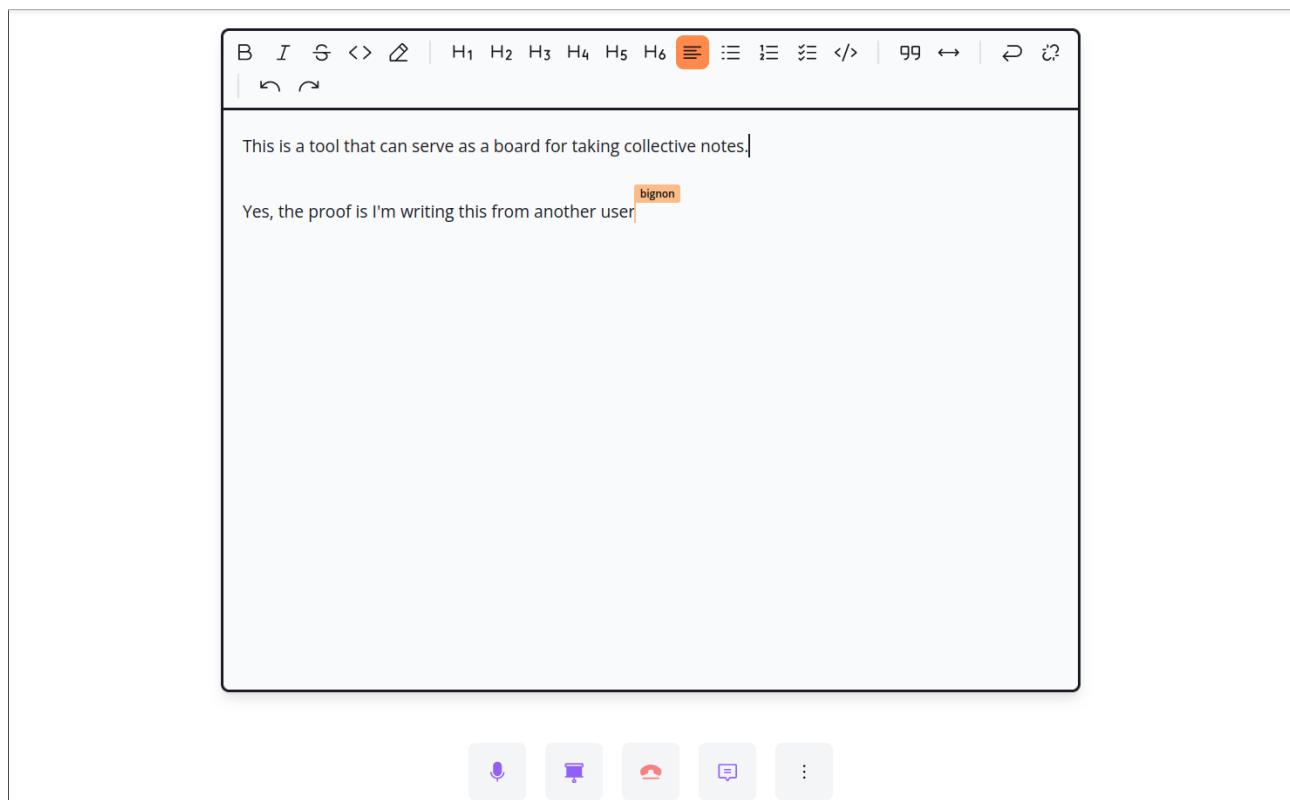


FIGURE 3.9 : Outil de note synchronisé

Les fonctions suscitées rendent inaccessible la grille des participants. Mais il est toujours possible de pourvoir y accéder dans la même section que la messagerie, comme le montre la figure 3.10.

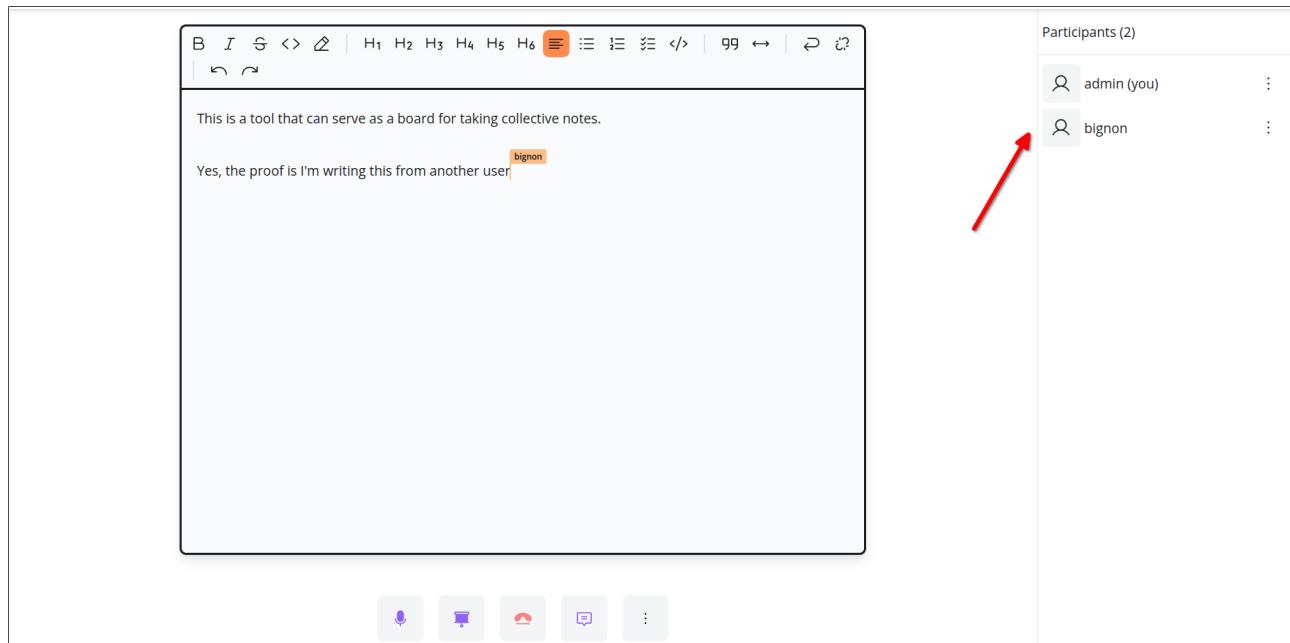
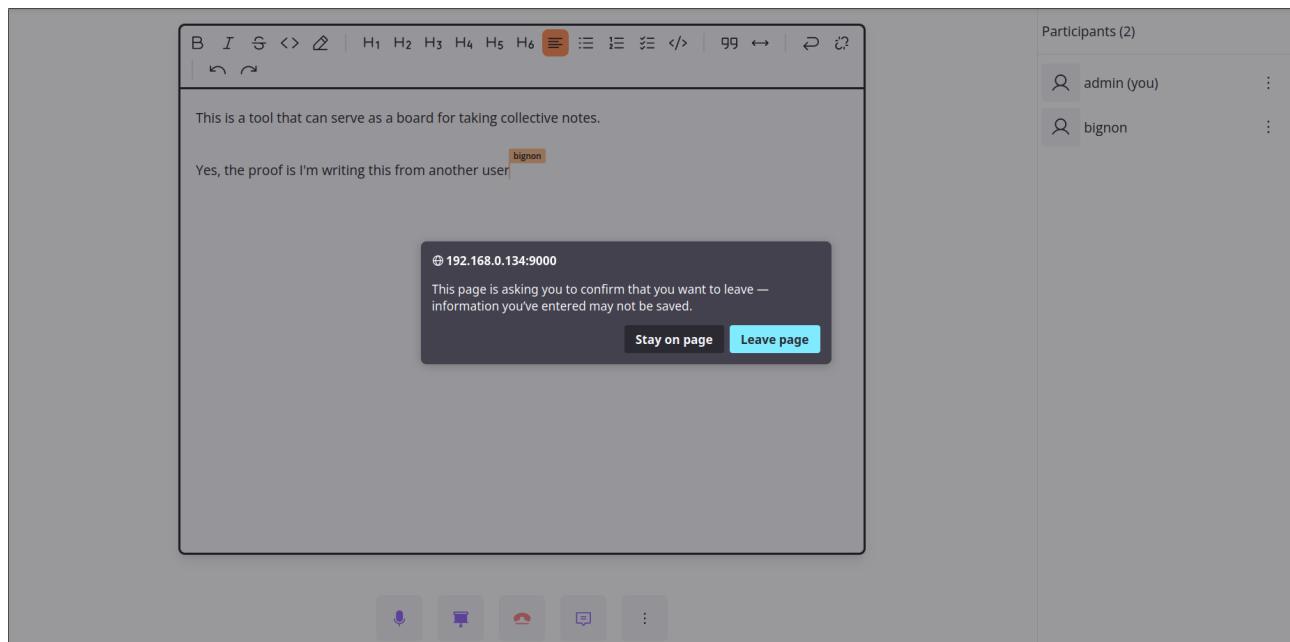
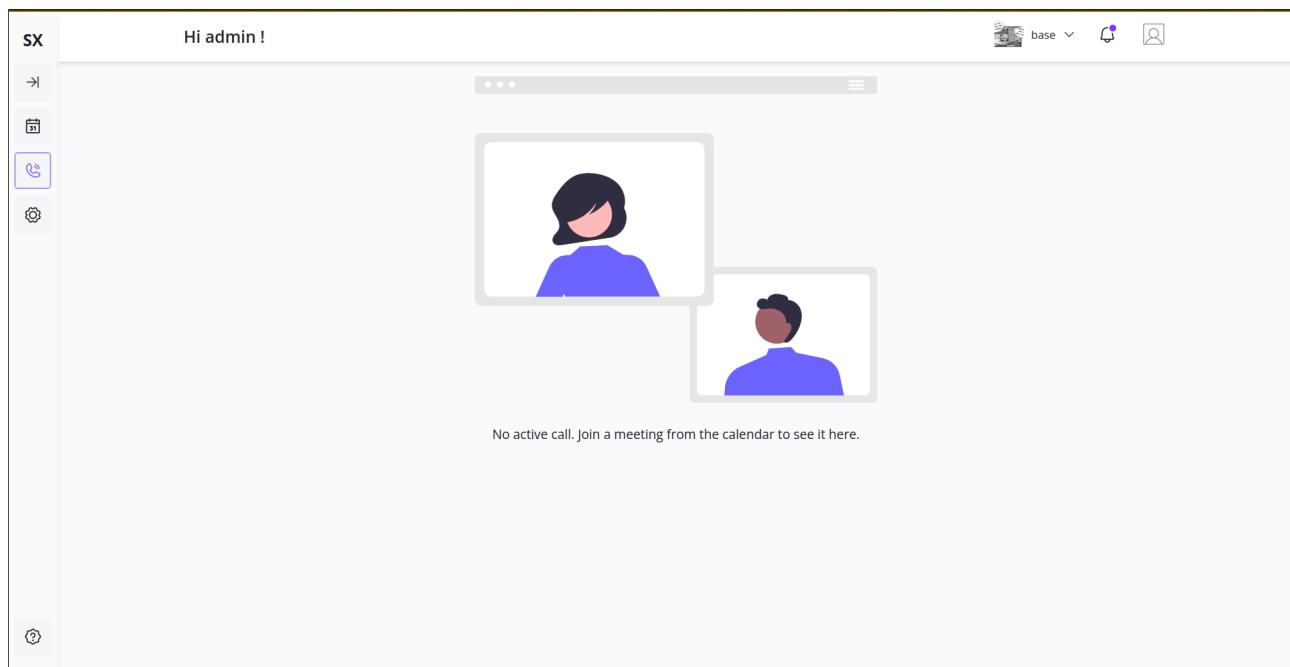


FIGURE 3.10 : Liste des participants

Enfin, chaque utilisateur a la possibilité de quitter la réunion. Si par mégarde, il essaie de recharger par exemple, l'onglet, une confirmation est requise (si le navigateur supporte cette fonctionnalité), comme le montre la figure 3.11.



(a) Confirmation de déconnexion



(b) Page de redirection après déconnexion

FIGURE 3.11 : Processus de déconnexion.

### 3.1.4 Autres fonctionnalités

Dans le but d'améliorer l'expérience utilisateur, nous avons jugé utile d'ajouter quelques fonctionnalités outre celles initialement visées. Parmi elles figurent le mode sombre et la mise en place d'un tutoriel interactif expliquant les diverses composantes de notre application.

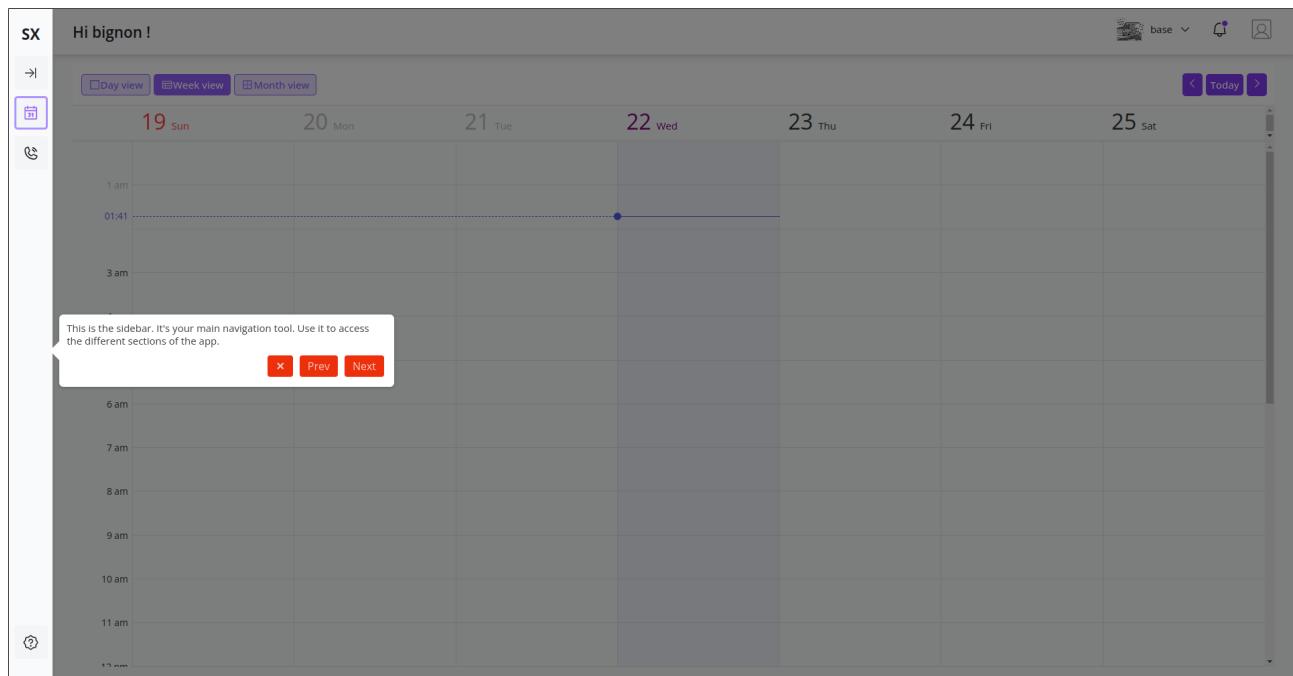


FIGURE 3.12 : Tutoriel interactif d'introduction à StudX

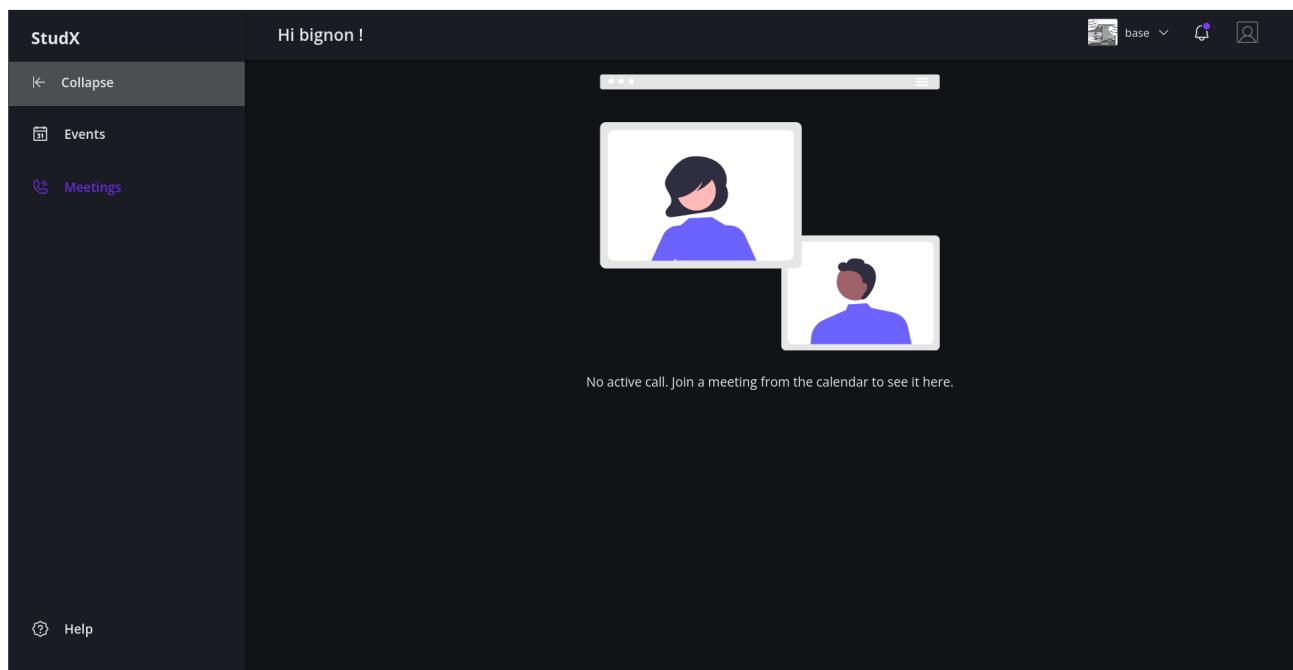


FIGURE 3.13 : Mode sombre

Les administrateurs de la plateforme disposent également d'un accès aux paramètres de l'organisation qu'ils dirigent et peuvent ainsi ajouter ou retirer des membres (figure 3.14).

The screenshot shows the StudX application's settings page for an organization named "base". The interface includes a sidebar with icons for navigation and a main area for configuration. In the main area, there is a section for "General" settings, which includes a profile picture of a cartoon bear, a "Change" button, a "Name" field set to "base", a "Description (optional)" field containing "The base organization.", and a "Save" button. Below this is a "Members (3)" section with a "Add a new member" button and a "Username" input field containing "admin".

(a) Paramètres généraux.

The screenshot shows the StudX application's member management page. It displays a list of three members: "admin" (ADMIN, joined 2 months ago), "bignon" (STUDENT, joined 6 days ago), and "hans" (TEACHER, joined 6 days ago). Each member entry includes a profile icon, their name, their role, and a delete button. Below the member list is a "Danger Zone" section with buttons for "Archive organization" and "Delete organization".

(b) Gestion des membres.

FIGURE 3.14 : Paramètres d'une organisation.

## 3.2 Discussion

Le développement de l'application StudX a été un projet ambitieux, nécessitant la mise en place de nombreuses technologies modernes pour répondre aux besoins d'un environnement en ligne en constante évolution. Les fonctionnalités prévues pour l'application ont toutes été implémentées, permettant une tenue efficace de classes virtuelles en temps réel. Le code source de l'application est disponible à l'adresse suivante : <https://github.com/tobihans/studx.git>.

Cependant, la mise en œuvre de la technologie WebRTC a été un défi technique majeur tout au long du processus de développement. Le recours à cette technologie nécessite une compréhension approfondie de son fonctionnement, ainsi que des compétences en programmation avancées pour l'adapter à nos besoins spécifiques. Cela a été rendu encore plus complexe par le fait que WebRTC est une technologie relativement nouvelle, qui ne cesse de se développer.

En outre, l'utilisation de code Open Source dans l'application a été également un facteur clé. Il a fallu comprendre le fonctionnement du code et le modifier pour répondre aux exigences de notre projet. Cela a nécessité un travail minutieux, impliquant une analyse approfondie du code existant, ainsi que des compétences en programmation avancées pour le modifier efficacement.

Malgré ces défis et bien d'autres, notre prototype est en place. Le choix du moteur Mediasoup pour les besoins de relais a été un choix judicieux. Ce moteur supporte un grand nombre de participants et intègre un concept de routeur pouvant supporter jusqu'à 500 personnes. Les routeurs peuvent également être connectés entre eux, offrant une grande évolutivité et une flexibilité pour répondre aux besoins futurs de l'application. Le choix de s'abstenir de diffuser les flux vidéo permet également d'alléger le traffic et va permettre le support d'un grand nombre.

Outre celà, si l'application est déployée dans un réseau local, il est possible d'éviter totalement les couts liées à Internet. Les usagers proches du point de connectivité disposent d'u accès qui ne requiert pas du tout Internet. Plus la couverture du réseau est grande, plus facile il est de donner accès aux utilisateurs sans que ces derniers ne dépensent dans la consommation de forfaits Internet, ce qui engendre ainsi des économies, sur le plan financier.

Le développement de StudX a été une expérience enrichissante. Bien que des défis aient été rencontrés tout au long du processus, l'application fonctionne maintenant et répond aux objectifs initialement fixés.

## Conclusion

Après la conception et le développement de l'application, il est clair que le prototype répond aux besoins identifiés et remplit les objectifs fixés pour ce projet. Les tests effectués ont démontré la fiabilité et la performance de l'application, ainsi que sa capacité à gérer les fonctionnalités attendues. Cependant, cela ne signifie pas qu'elle est parfaite, car de nombreux aspects peuvent être encore améliorés et développés pour répondre à des besoins futurs ou pour satisfaire des demandes plus spécifiques.

# Conclusion Générale

L'évolution des méthodes d'enseignement implique l'utilisation croissante des technologies de l'information et de la communication. Cette tendance est motivée par la nécessité de s'adapter aux nouveaux besoins d'apprentissage et de compenser les insuffisances logistiques et financières auxquelles font face les établissements d'enseignement supérieur. Dans ce contexte, le projet auquel nous avons contribué visait à fournir un cadre moderne de communication en temps réel, permettant d'explorer de nouvelles possibilités en matière de formation virtuelle. Nous sommes convaincus que l'application que nous avons développée ouvre la voie à une large gamme d'opportunités en matière d'exploitation des classes virtuelles, et nous espérons que notre travail sera une contribution utile à l'ensemble de la communauté éducative. Bien qu'ayant atteint ses objectifs initiaux et répondu aux besoins identifiés, il reste encore de nombreuses possibilités d'extension pour en faire un outil encore plus puissant et adapté aux besoins évolutifs de l'enseignement en ligne. Il est donc envisageable d'explorer des pistes de développement pour étendre l'application au-delà de ses fonctions actuelles.

## Perspectives

Le prototype StudX propose une variété de fonctionnalités intéressantes pour la tenue de classes virtuelles. Toutefois, il présente certaines limites, outre les choix délibérés de conception comme l'absence de flux vidéo.

Bien que des règles d'accessibilité basiques aient été prises en compte, elles ne répondent pas entièrement aux besoins des personnes malentendantes, qui ne peuvent pas bénéficier des échanges vocaux entre les participants. Pour remédier à cela, il serait envisageable d'intégrer un modèle de Machine Learning pour la conversion des signaux audio en gestes du langage des signes.

Par ailleurs, dans le cadre d'un prototype, les fonctionnalités sont plutôt restreintes. On pourrait ajouter ou améliorer des fonctions comme la persistance de la messagerie instantanée ou encore la gestion des utilisateurs. L'enregistrement des sessions pour un usage ultérieur, par exemple, est une fonctionnalité dont l'implémentation a démarré (branche **room-recording**) dans le but de permettre une amélioration de l'expérience utilisateur.

# Bibliographie

- [1] Eugène C. EZIN. "Les ressources éducatives libres et les Moocs : Perspectives pour l'Université d'Abomey-Calavi". In : *Research Gate*. Sept. 2015. URL : [https://www.researchgate.net/publication/283225586\\_Les\\_ressources\\_educatives\\_libres\\_et\\_les\\_Moocs\\_Perspectives\\_pour\\_l'Universite\\_d'Abomey-Calavi](https://www.researchgate.net/publication/283225586_Les_ressources_educatives_libres_et_les_Moocs_Perspectives_pour_l'Universite_d'Abomey-Calavi).
- [3] Simon HOLM et Alexander LÖÖF. "The design and architecture of a WebRTC application". Bachelor's degree 180 hp. Malmö University, 2019. URL : <https://www.diva-portal.org/smash/get/diva2:1480111/FULLTEXT01.pdf>.
- [4] Santeri JUSLENIUS. *WebSocket vs WebRTC in the stream overlays of the Streamr Network*. 2021. URL : [https://helda.helsinki.fi/bitstream/handle/10138/332587/Juslenius\\_Santeri\\_gradu\\_2021.pdf](https://helda.helsinki.fi/bitstream/handle/10138/332587/Juslenius_Santeri_gradu_2021.pdf).
- [5] Andreas KAPLAN. *Contemporary Issues in Social Media Marketing*. Routledge, 2017.

# Webographie

- [2] Object Management GROUP. *UML Specification*. URL : <https://www.omg.org/spec/UML/>.
- [6] Rust LANG. *Site officiel du langage Rust*. URL : <https://rust-lang.org>.
- [7] Microsoft LEARN. *Résilience et haute disponibilité dans les microservices*. URL : <https://learn.microsoft.com/fr-fr/dotnet/architecture/microservices/architect-microservice-container-applications/resilient-high-availability-microservices>.
- [8] REDHAT. *Une API REST, qu'est-ce que c'est ?* URL : <https://www.redhat.com/fr/topics/api/what-is-a-rest-api>.
- [9] Document Open SOURCE. *WebRTC for the curious*. URL : <https://webrtcforthe curious.com/docs/03-connecting/>.
- [10] UAC. *Site d'education de l'Université d'Abomey-Calavi*. URL : <https://etudiant.bj>.
- [11] WIKIPÉDIA. *Distance education*. URL : [https://en.wikipedia.org/wiki/Distance\\_education](https://en.wikipedia.org/wiki/Distance_education).
- [12] WIKIPÉDIA. *Enseignement supérieur*. URL : [https://fr.wikipedia.org/wiki/Enseignement\\_sup%C3%A9rieur](https://fr.wikipedia.org/wiki/Enseignement_sup%C3%A9rieur).
- [13] WIKIPÉDIA. *Modèle d'acteur*. URL : [https://fr.wikipedia.org/wiki/Mod%C3%A8le\\_d%27acteur](https://fr.wikipedia.org/wiki/Mod%C3%A8le_d%27acteur).
- [14] WIKIPÉDIA. *Multitenant*. URL : <https://fr.wikipedia.org/wiki/Multitenant>.
- [15] WIKIPÉDIA. *Nuxt.js*. URL : <https://fr.wikipedia.org/wiki/Nuxt.js>.
- [16] WIKIPÉDIA. *TypeScript*. URL : <https://fr.wikipedia.org/wiki/TypeScript>.
- [17] WIKIPEDIA. *Online Learning*. URL : [https://en.wikipedia.org/wiki/Online\\_learning\\_in\\_higher\\_education](https://en.wikipedia.org/wiki/Online_learning_in_higher_education).

# Table des matières

<b>Dédicace</b>	ii
<b>Remerciements</b>	iii
<b>Résumé</b>	iv
<b>Abstract</b>	v
<b>Liste des acronymes</b>	viii
<b>Glossaire</b>	xi
<b>Introduction</b>	2
<b>1 Revue de littérature</b>	4
Introduction .....	4
1.1 Formation à distance .....	4
1.1.1 Origines .....	4
1.1.2 Internet et formations en ligne .....	4
1.2 Communication en temps réel .....	5
1.3 Software as a Service .....	7
1.4 Présentation de solutions existantes .....	7
1.4.1 Google Classroom .....	7
1.4.2 Zoom .....	8
1.4.3 Moodle .....	9
Conclusion .....	10
<b>2 Matériels et méthodes</b>	11
Introduction .....	11
2.1 Méthodes de conception .....	11
2.1.1 Diagramme de cas d'utilisation .....	11
2.1.1.1 Description textuelle du cas d'utilisation "Ajouter un événement" ..	12
2.1.1.2 Description textuelle du cas d'utilisation "Rejoindre une réunion" ..	12
2.1.2 Diagramme de séquence .....	13
2.1.3 Diagramme de classe .....	14
2.1.4 Architecture du système .....	15
2.2 Matériels .....	16
2.2.1 Choix techniques .....	16
2.2.1.1 Proxy .....	16

---

2.2.1.2	API REST . . . . .	17
2.2.1.3	Frontend . . . . .	17
2.2.1.4	Base de données . . . . .	18
2.2.1.5	Gestionnaire de files de tâches . . . . .	18
2.2.1.6	Cache . . . . .	19
2.2.1.7	Service Writepad . . . . .	19
2.2.1.8	Service Whiteboard . . . . .	19
2.2.1.9	Serveur WebRTC . . . . .	20
2.2.2	Outils de développement . . . . .	22
Conclusion	. . . . .	22
<b>3</b>	<b>Résultats et Discussion</b>	<b>24</b>
Introduction	. . . . .	24
3.1	Résultats . . . . .	24
3.1.1	Authentification . . . . .	24
3.1.2	Calendrier . . . . .	25
3.1.3	Sessions en ligne . . . . .	26
3.1.4	Autres fonctionnalités . . . . .	32
3.2	Discussion . . . . .	34
Conclusion	. . . . .	36
Conclusion	. . . . .	36
<b>Bibliographie</b>		<b>37</b>
<b>Webographie</b>		<b>38</b>

