# **Introduction**

## **Contexte**

Les temps changent; l’enseignement change; les universités changent. 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 de l’emploi 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 sections 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 mémoire**

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 ou l’enseignant et l'étudiant sont séparés dans le temps et par l’espace [1].

**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 Londre 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 [2]. En 2020, par exemple, sous l’impulsion de la pandémie alors en cours, plusieurs universités dont celle d’Abomey-Calavi [url uac], effectuent la transition partielle ou totale vers les classes virtuelles [3].

On distingue deux environnements d’apprentissage. D’une part, les environnements asynchrones [4] 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[], professeur de marketing, propose une approche simplifiée basée sur les facteurs comme le temps, la distance et le nombre d’apprenants. Le tableau ci-dessous, en fait un récapitulatif.

|  |  |  |  |
| --- | --- | --- | --- |
| Type | Description | Nombre d’apprenants | Type d’environnement |
| MOOCs | Massive Online Open Courses | illimité (en théorie) | asynchrone |
| SMOCs |  | illimité (en théorie) | synchrone |
| SPOCs |  | limité | asynchrone |
| SSOCs |  | limité | synchrone |

Fig. Tableau récapitulatif des types de cours en ligne

A ce stade, il est important de préciser que notre projet s'intéresse aux environnement d’apprentissage synchrones, avec le support d’un grand nombre d’apprenants.

**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 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 [ref webrtcforthecurious]. Pour établir une connexion, il faut quatre étapes à savoir la signalisation, la connexion proprement dite, la sécurisation puis la communication.

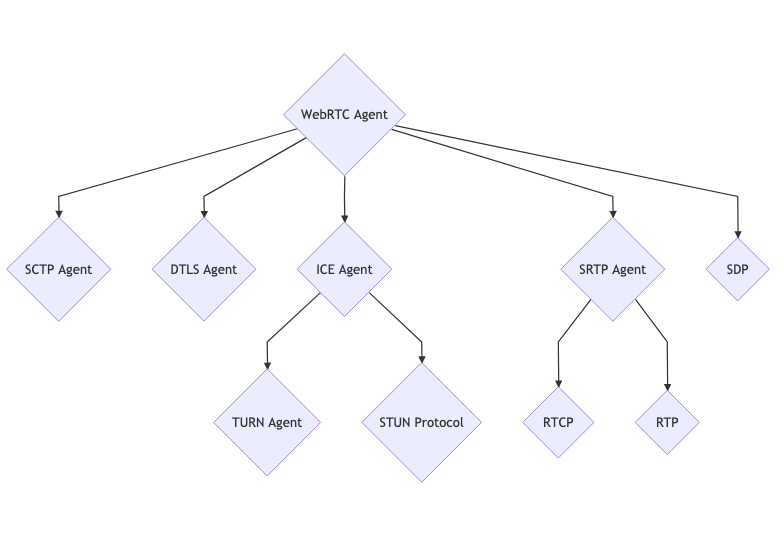


Fig. Protocoles employés par WebRTC

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 SDP est utilisé et permet la transmission d’informations capitales comme:

* l’IP et le port de chaque agent WebRTC (plusieurs variantes, en réalité)
* les codecs multimédia supporters
* 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’IP et du port), le protocole ICE permet de choisir le meilleur candidat, en faisant recours au serveur STUN et parfois, au serveur TURN. Le serveur TURN 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 (NAT et les types de liaisons possibles) [webrtc for the curious types of link].

Pour assurer la sécurité de la connexion, les protocoles DTLS et SRTP 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 au protocoles RTP et SCTP.

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.

**1.3. 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 a ces outils ne sont aucunement d’ordre technique. Nous nous interessons plutot 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 eliminant les barrieres possibles.

**1.3.1. Google Classroom**

Google Classroom est un outil de la suite Google pour education. A defaut de disposer d’un module de visio-conference, il s’integre parfaitement avec Google Meet, a cette fin. L’application offre un version gratuite et dispose d’une interface accessible. Toutefois, poour les reunions en ligne, le nombre maximum de connexions possibles se limite a 500 participants. Pour les entites universitaires dont l’effectif est considerable par classe, ceci pourrait presenter un desavantage. Grace a la version payante neanmoins, on peut mettre en place un *live stream*, pour permettre d’acceder au contenu de la reunion sans toutefois pouvoir interagir avec les participants. Le modele de souscription aussi, base sur le nombre d’utilisateurs risque d’entrainer des frais assez eleves.

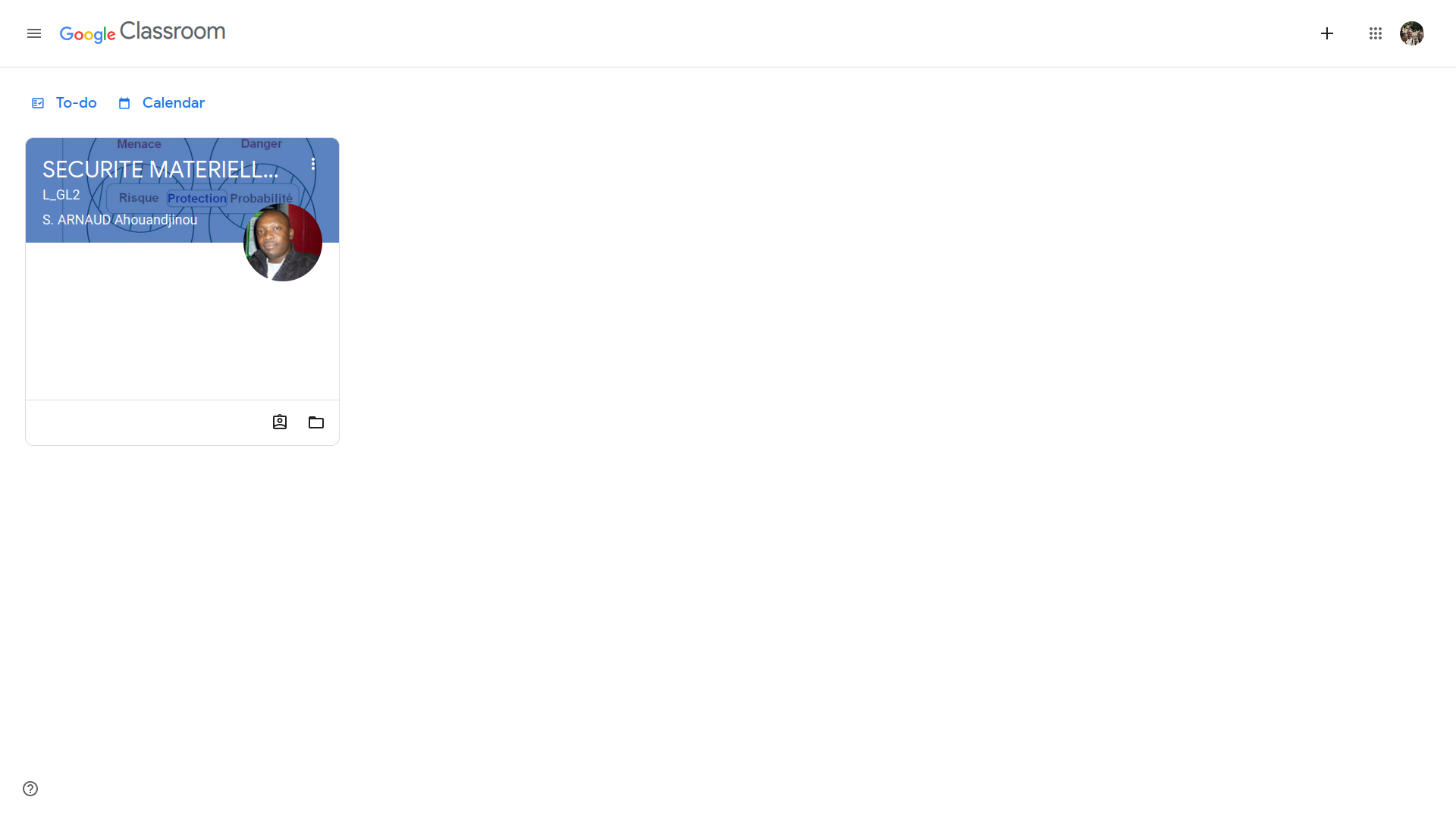


Fig. Interface d’acceuil de Google Classrooms

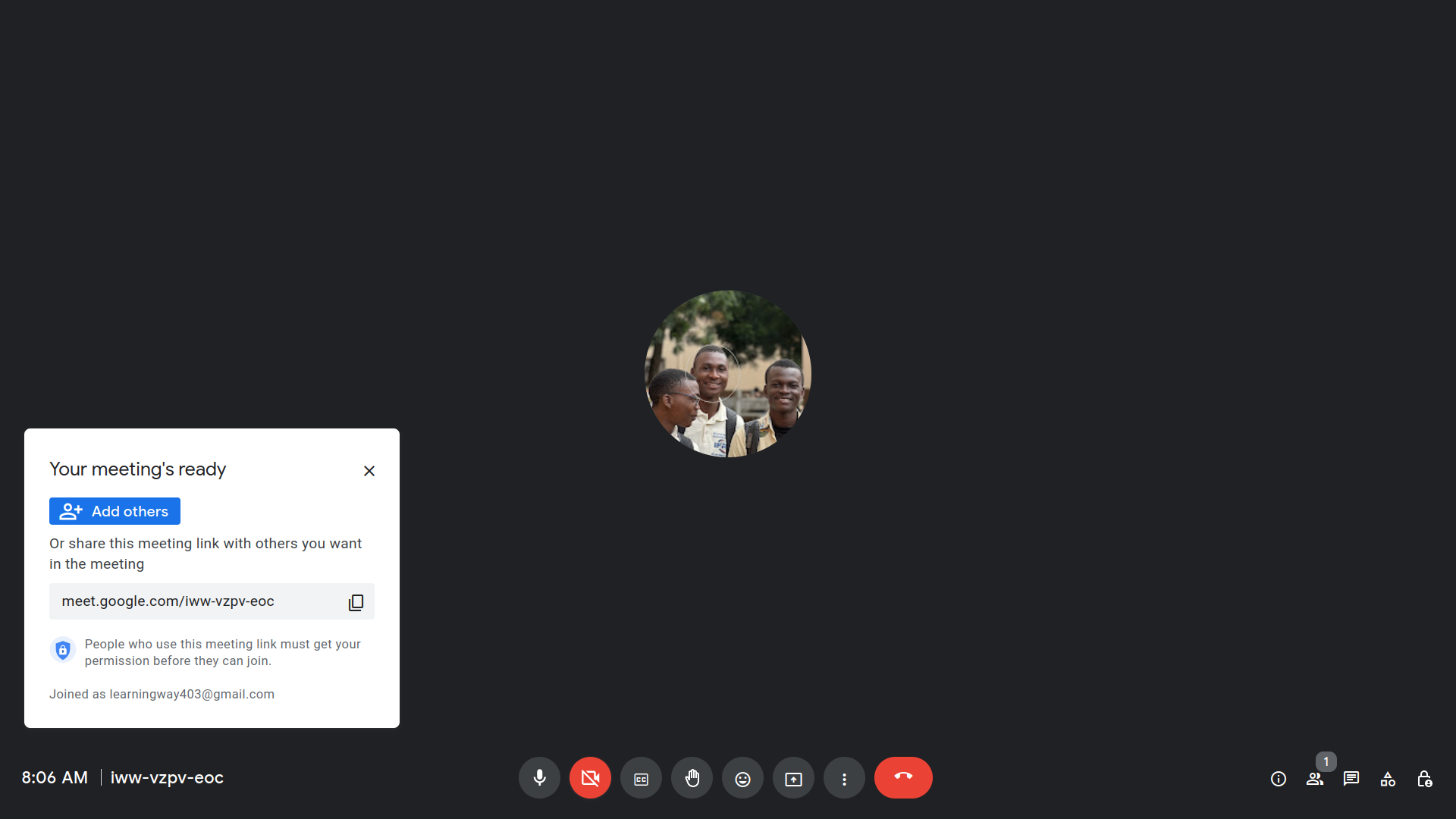


Fig. Interface de Google Meet

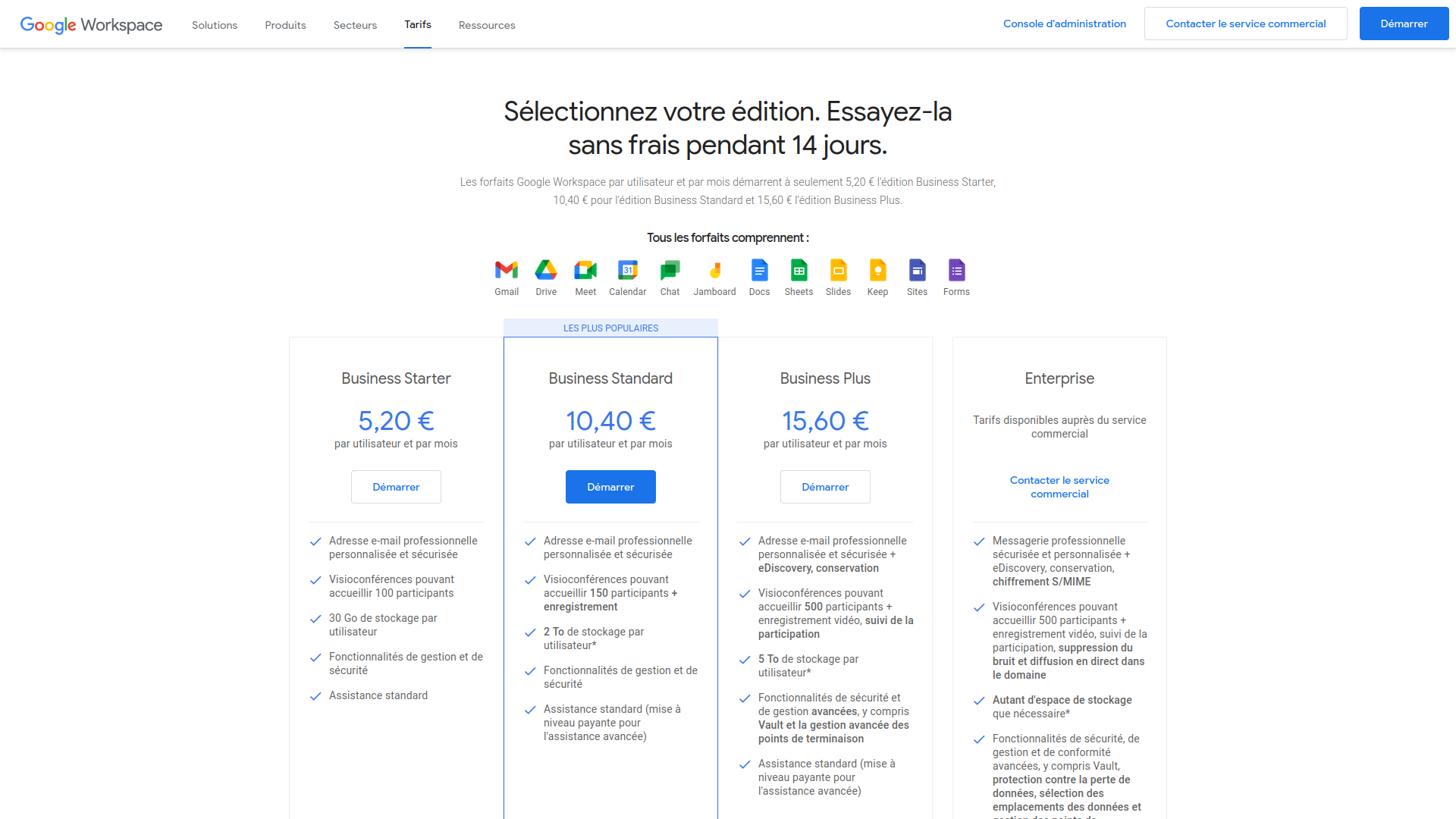


Fig. Page de souscription de la suite Google

**1.3.2. Zoom**

Zoom est un outil de communication tres performant, qui a la capacite de supporter un grand nombre d’utilisateurs. Il dispose de fonctionnalites tres utiles pour le deroulement de cours en ligne comme le partage d’ecran ou le tableau virtuel. Acceder a ces fonctionnalites dans le cadre d’une utilisation a grande echelle requiert une souscription et les offres de Zoom no sont pas des plus simples. En efffet, Zoom dispose d’un panel large de services associés et donc, sans orientation, il est probable de choisir une solution inadequate en rapport avec le besoin.



Fig. Interface de Zoom

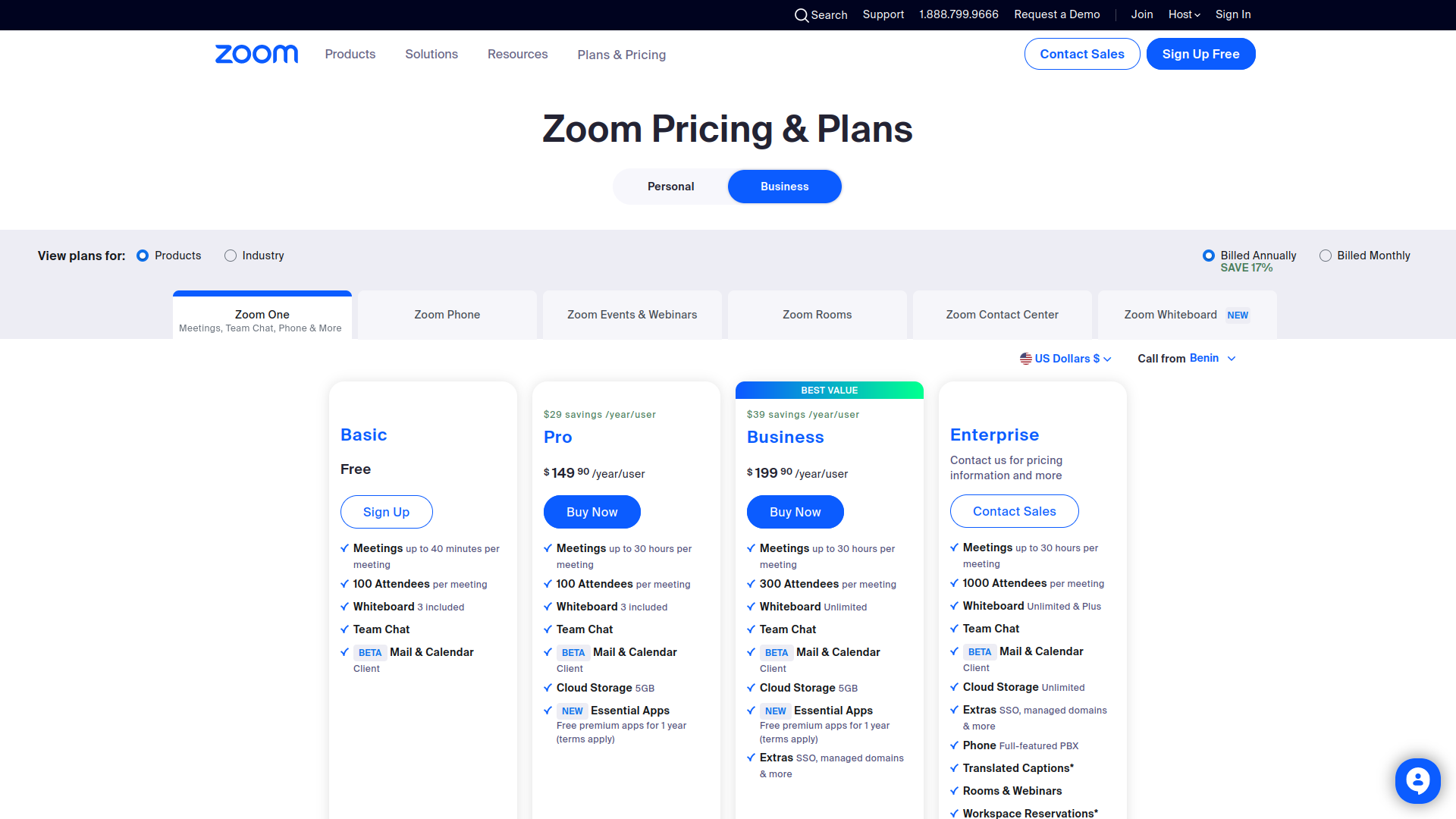


Fig. Page des souscriptions de Zoom

**1.3.3. Moodle**

Moodle est un LMS Open Source populaire tres connu et utilisé dans les entités de l’enseignement supérieur. Il peut etre heberge ou utiliser en ligne. Il offre un large panel de fonctionnalités et permet l’integration de divers modules dont des modules de visio-conférence. BigBlueButton est une solution Open source employé a cet effet. La mise en place, requiert toutefois, une certaine expertise et du materiel spécifique, ce qui en limiterait la portabilité.

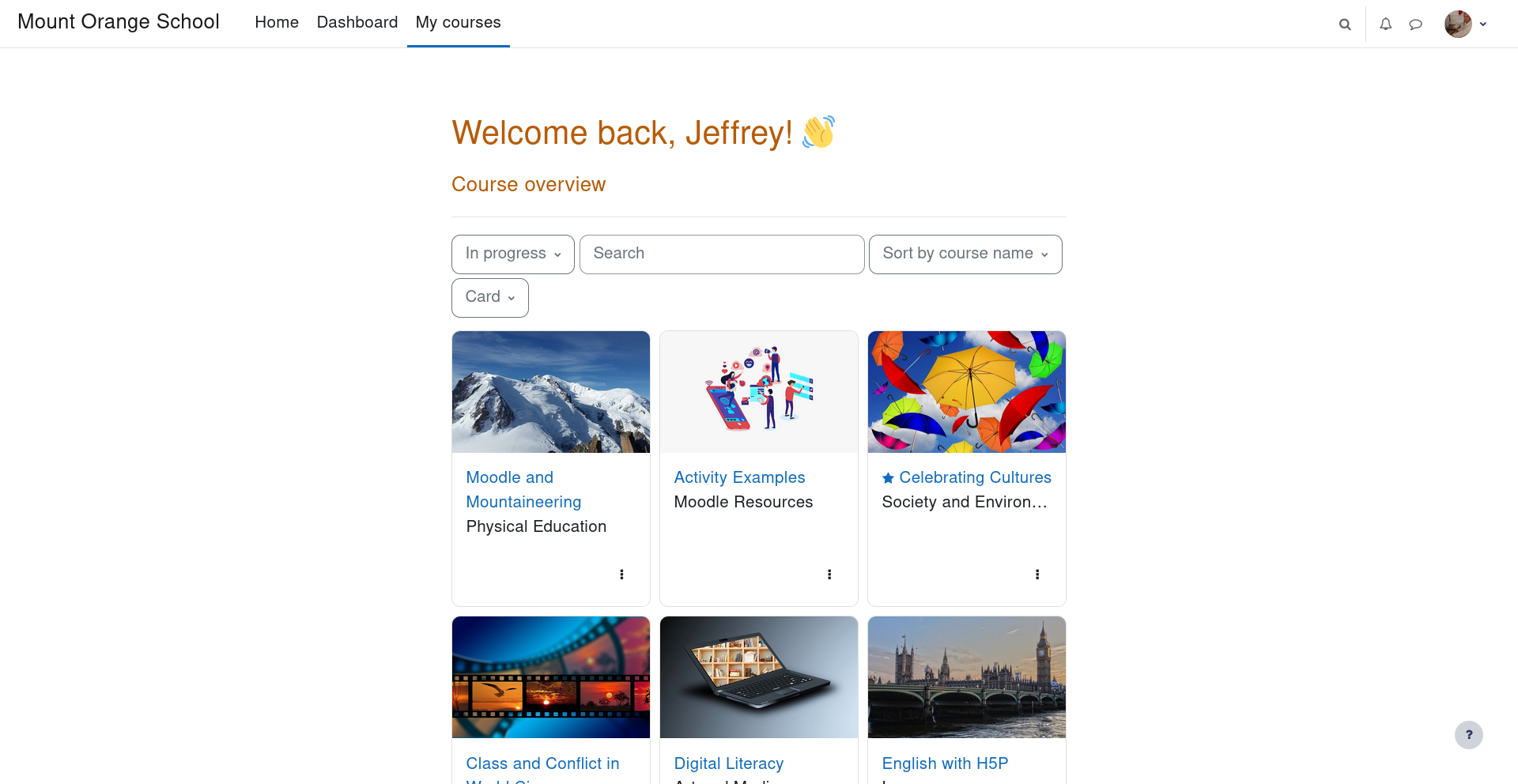


FIg. Site demo de Moodle

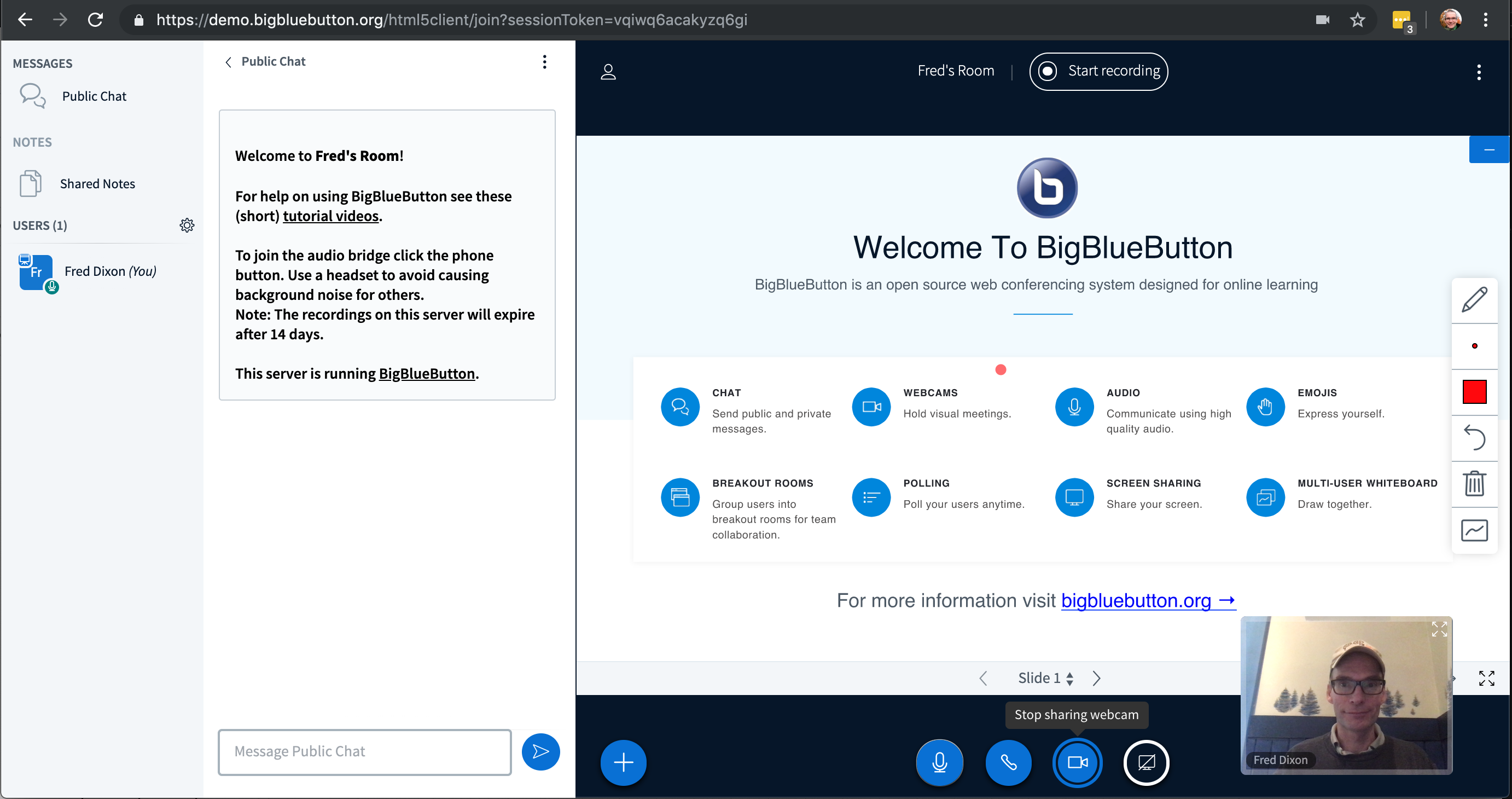


Fig. Demonstration de BigBluebutton

**Conclusion**

# Ce chapitre a permis de faire une revue de l’existant et jette les bases des suivants en exposant les concepts cles qui seront developpes. Les solutions suscitees conviendraient pour un usage modere. Elles peuvent s’averer couteuses, pour peu qu’elles repondent au besoin. La solution que nous proposons vise a doter les organismes de l’enseignement superieur, d’un moyen simple mais efficace de tenir les cours en ligne, offrant des outils d’assistance, tout en minimisant les couts, que cela pourrait engendrer.

# **2. Materiels et methodes**

**Introduction**

Ce chapitre est dedie a la mise en lumiere des pratiques d’architecture logicielle employees lors de la conception de notre prototype. Nous y presenterons egalement les choix techniques effectues.

**2.1. Methodes de conception**

Dans le souci de decrire de facon fiable, les fonctionnalites du system, nous faisons usage du langage visuel UML. Il s’agit d’une methode de visualisation d’architecture logicielle permettant de modeliser l’architecture logicielle d’un systeme.

Standardise par l’OMG, la version actuelle de l’UML, la 2.5, propose 14 types de diagrammes. N’etant pas une methode, la norme laisse l’utilisation des diagrammes a l’appreciation des utilisateurs.

Dans le cadre de notre prototype, nous avonc retenu uniquement les diagrammes de cas d’utilisation, de sequence 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 systeme. Les cas d’utilisation sont utiles pour decrire les interactions possibles entre le/les acteurs acteur(s) et le systeme.

Les acteurs intervenant dans notre systeme sont:

* L’etudiant: il dispose d’un acces en lecture aux informations du systeme
* L’enseignant: il dispose d’un acces total en lecture et partiel en ecriture sir certaines informations
* L’administrateur: il dispose de privileges eleves pour modifier les informations de la plateforme et l’administrer.

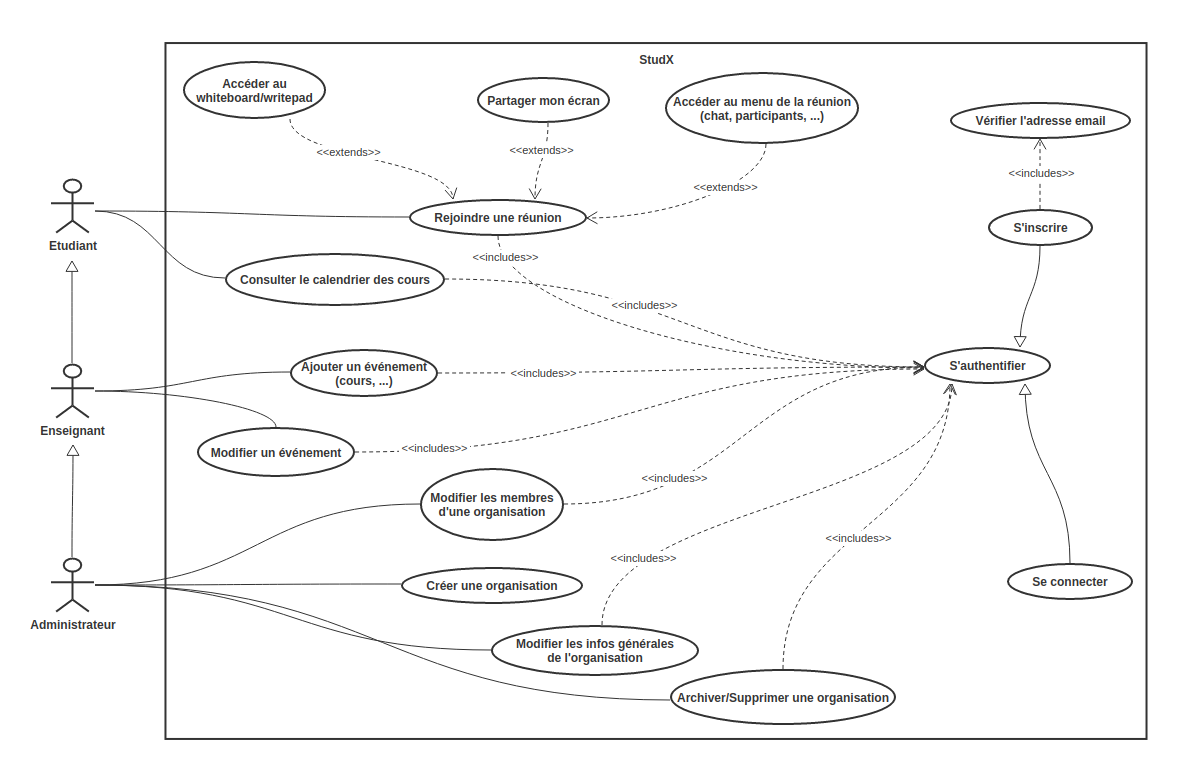


Fig. Diagramme de cas d’utilisation

Outre le diagramme, il s’avere parfois necessaire de fournir, en plus, des descriptions textuelles des cas d’utilisation, dans le but d’apporter plus d’eclaircissements. Ci-dessous, sont presentees les descriptions textuelles des cas “S’authentifier”, “Ajouter un evenement” et “Rejoindre une reunion”.

**2.1.1.1 Description textuelle du cas d’utilisation “S’authentifier”**

Sommaire d’identification:

Titre: S’authentifier

Objectif: Authentifier un utilisateur afin qu’il accede a la plateforme

Acteurs: Etudiant ou Enseignant ou Administrateur

Scenarii:

Pre-conditions:

- l’utilisateur dispose d’un compte et est admis dans au moins une organisation

Sequence nominale:

1. L’utilisateur remplit le formulaire de connexion

2. Le systeme authentifie l’utilisateur

3. Le systeme redirige l’utilisateur vers la page la page d’acceuil de l’organisation la plus recente.

Post-conditions:

- L’utilisateur dispose d’une session active

Enchainements alternatifs:

A1: Identifiants incorrects

L’enchainement demarre au point 2

2. Le systeme renvoie l’utilisateur vers la page de connexion avec une notification.

Ce dernier effectue a nouveau la saisie. L’enchainement reprend au point 3.

A2: L’utilisateur n’appartient a aucune organisation

L’enchainement prend depart au point 3.

3. Le systeme redirige l’utilisateur vers une page 404.

**2.1.1.2 Description textuelle du cas d’utilisation “Ajouter un evenement”**

Sommaire d’identification:

Titre: Ajouter un evenement

Objectif: Planifier les cours en ligne en ajoutant des evenements au calendrier

Acteurs: Enseignant ou Administrateur

Scenarii:

Pre-conditions:

- l’utilisateur est authentifie en tant qu’enseignant ou administrateur

Sequence nominale:

1. L’utilisateur accede au calendrier

2. Le systeme renvoie les evenements actuellement programmes

3. L’utilisateur remplit et soumet un formulaire de creation

4. Le systeme enregistre l’evenement et les details associés

5. Le system notifie les participants concernés

Post-conditions:

- L’utilisateur accede a l’evenement dans son calendrier

**2.1.1.2 Description textuelle du cas d’utilisation “Rejoindre une reunion”**

Sommaire d’identification:

Titre: Rejoindre une reunion

Objectif: Tenir une session de classe en ligne

Acteurs: Etudiant ou Enseignant ou Administrateur

Scenarii:

Pre-conditions:

- l’utilisateur est authentifie

Sequence nominale:

1. L’utilisateur consulte le calendrier

2. Le systeme affiche les divers evenements programmes

3. L’utilisateur accede aux details d’un evenement

4. L’utilisateur clique sur le lient pour rejoindre la reunion

5. Le system connecte l’utilisateur aux participants presents

Post-conditions:

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

# **2.1.2 Diagramme de sequence**

Le diagramme de sequence decrit les interactions, dans l’espace temps, entre objets dans le cadre des scenarii evoques au niveau des cas d’utilisations. Ci-dessous, sont illustres le diagrammes de sequences pour les trois cas d’utilisation suscites.

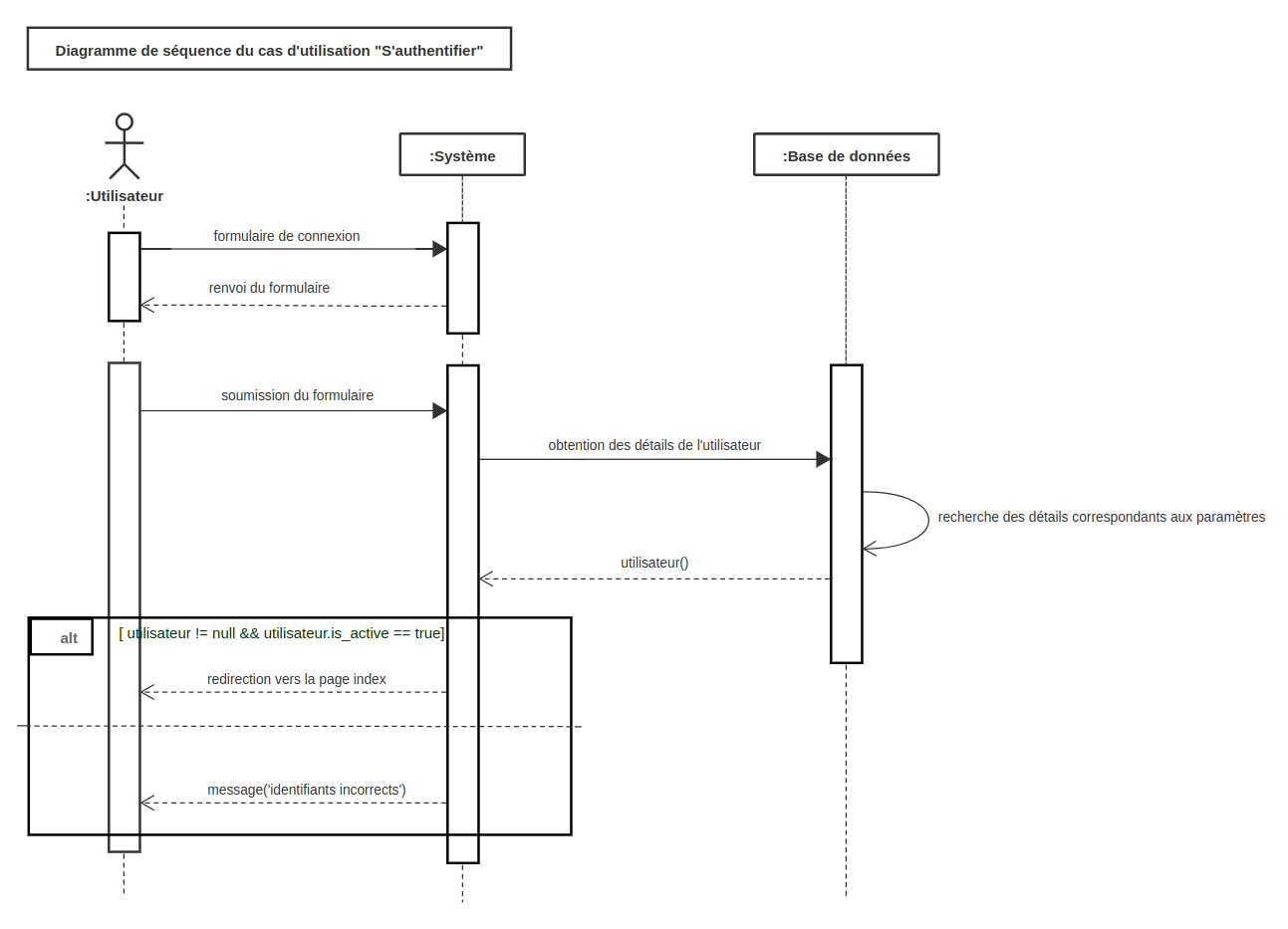


Fig. Diagramme de sequence du cas d’utilisaation “S’authentifier”

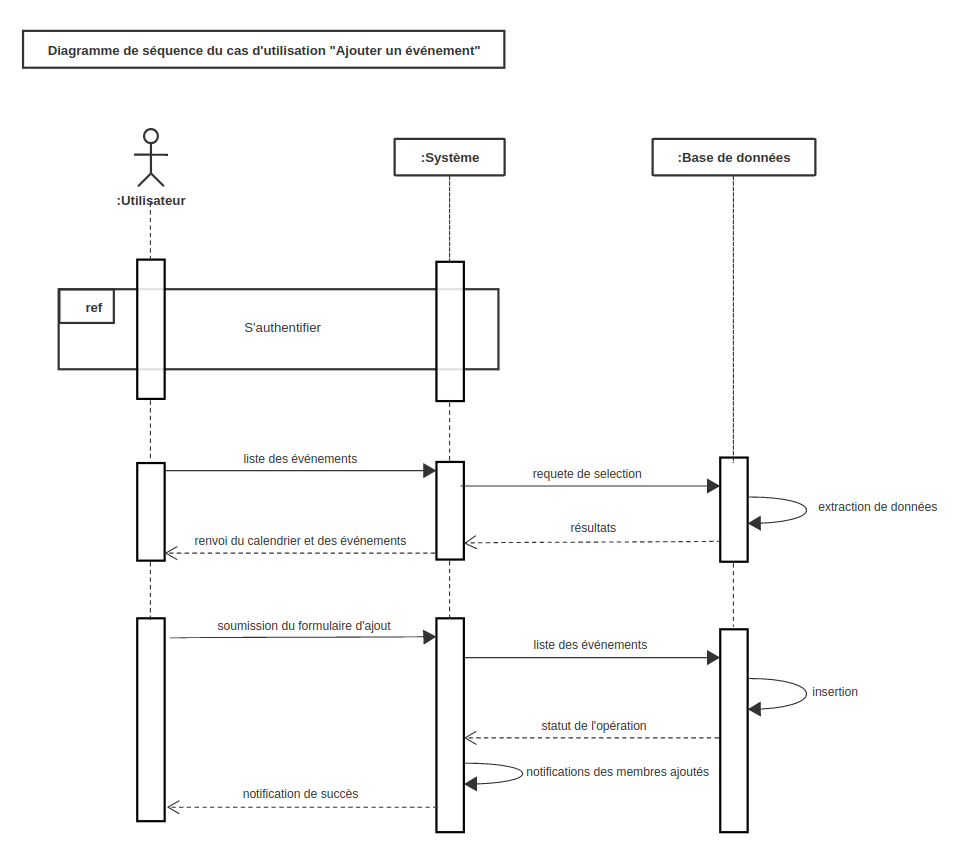


Fig. Diagramme de sequence pour le cas d’utilisation “Ajouter un evenement”

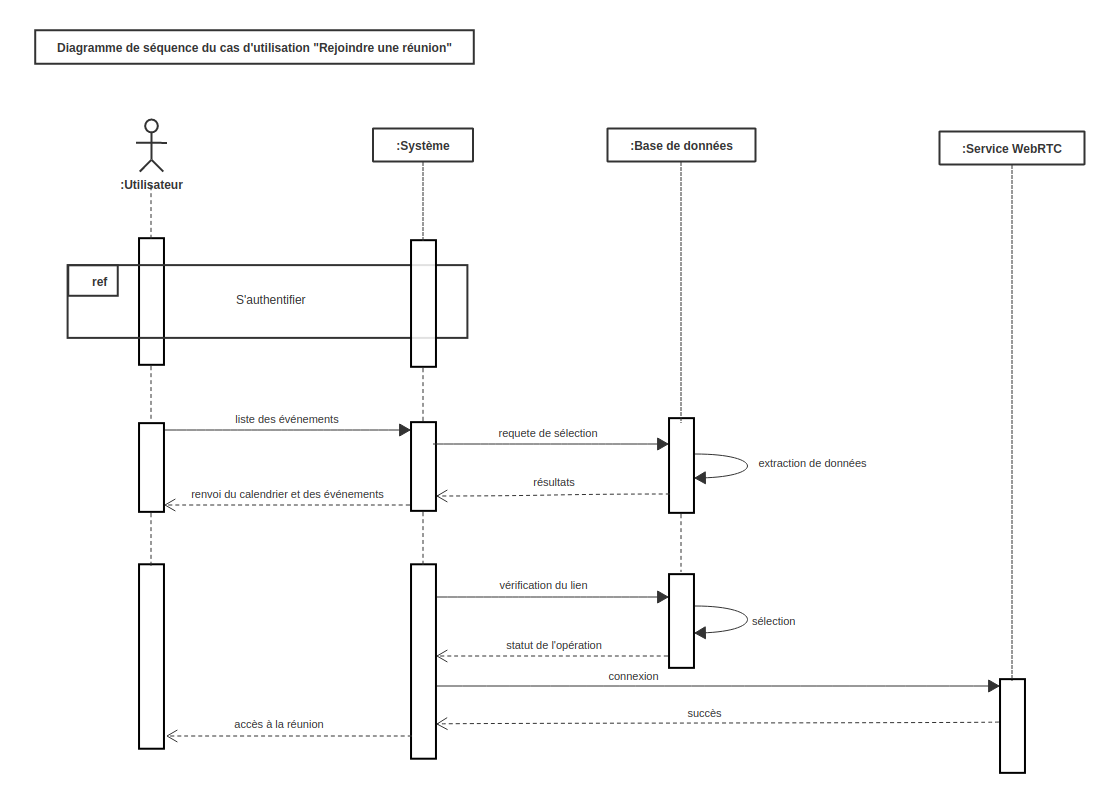


Fig. Diagramme de sequence pour le cas d’utilisation “Ajouter un evenement”

# **2.1.3 Diagramme de classe**

Le diagramme de classe illustre les classes et les interfaces du systeme ainsi que les relations qui les lient. Le diagramme ci-dessous decrit les entites de notre prototype.

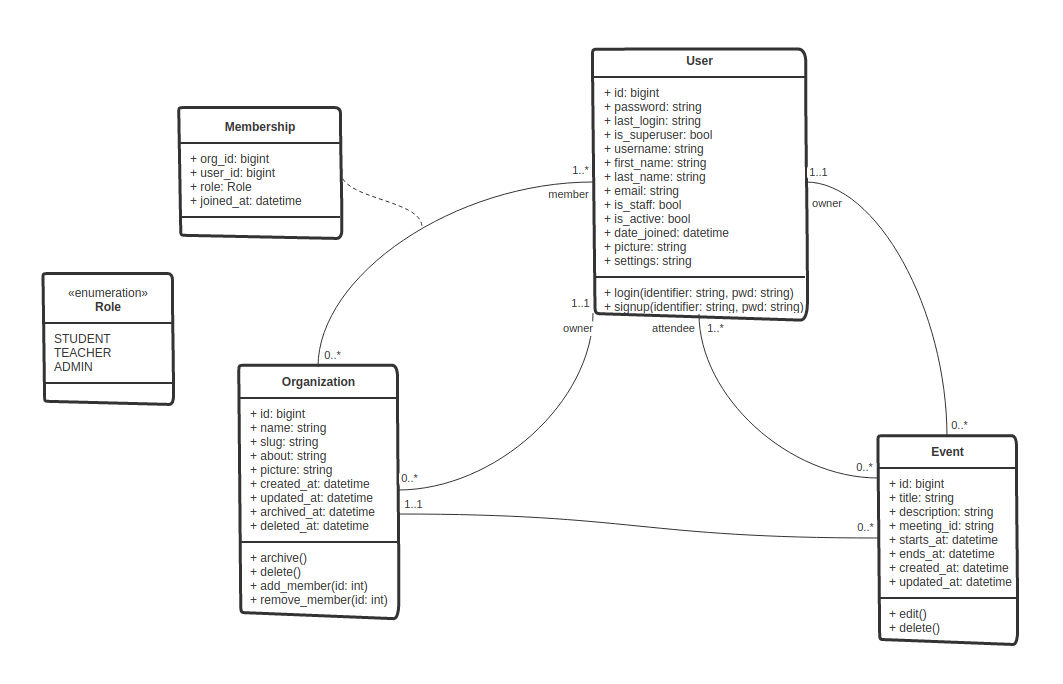
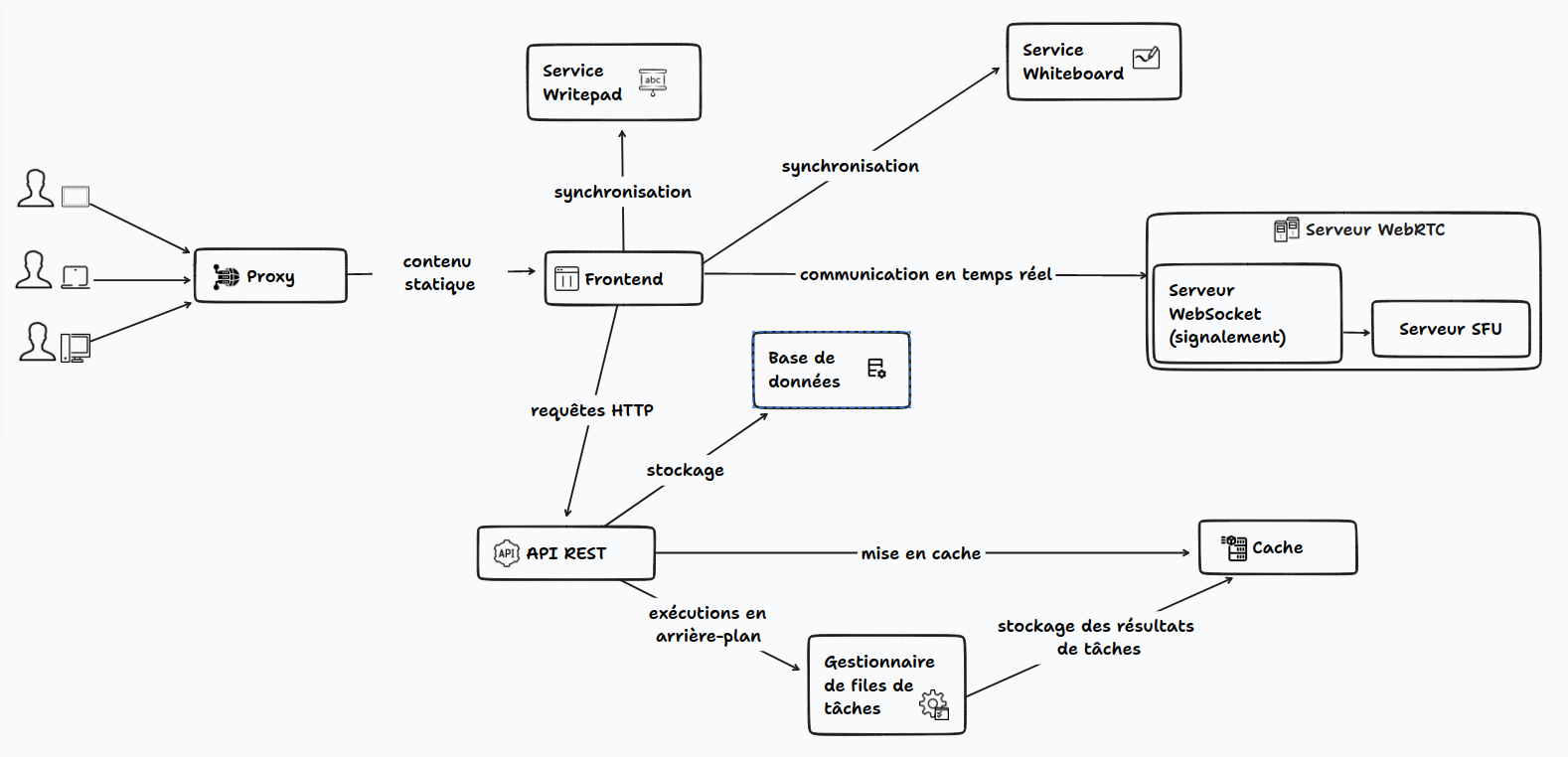


Fig. Diagramme de classe

# **2.1.4 Architecture du sytème**

Pour assurer le scabilite des systemes, il est important de bien en concevoir l’architecture. Dans ce but, nous avons adopte une approche decouplee, isolant les composantes du systeme. Il s’agit de microservices. Toutefois, il est important de noter qu’a l’echelle d’un prototype, l’architecture proposée reste tres simplifiee et ne prend pas en compte la resilience. Voici ci-dessous, une illustration des diverses composantes de notre architecture.



On peut notamment remarquer que toutes les interactions entre le systeme et les utilisateurs passent toutes par un proxy. Ceci s’explique par la volonte d’eviter les problemes de CORS, qui occurent des lors que les services ne sont pas tous un meme domaine.

**2.2. Methodes de conception**

# **2.2.1 Choix techniques**

# Faisant reference a l’architecture suscitee, voyons a present les technologies employees dans la mise en place de la solution.

**2.2.1.1 Proxy**

Un serveur proxy sert de relai entre differentes parties, notamment entre le client et le serveur dans notre contexte. Il s’agit dans ce cas, d’un reverse proxy.

Caddy esst un serveur Web moderne qui offre un large panel de fonctionnalites. Il offre un large eventail de fonctionnalites que l’on peut mettre en place via un fichier de configuration speciale nomme **Caddyfile**. En voici un example, extrait du code de notre prototype:

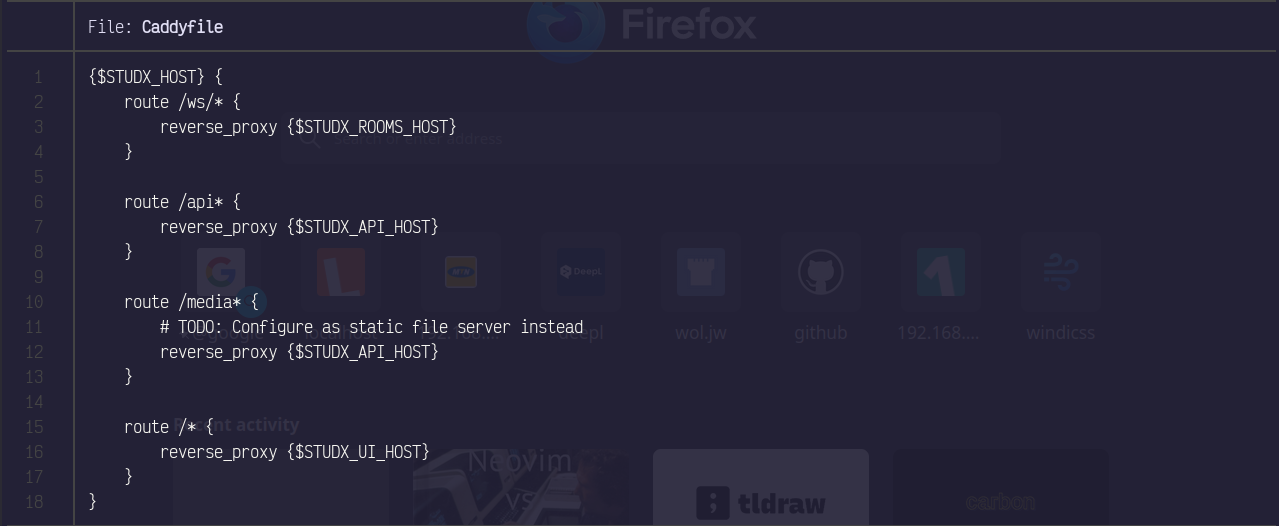


Fig. Caddyfile

On peut remarquer l’utilisation de variables d’environnement qui permettent de rendre la configuration encore plus dynamique. Tous ces atouts en on fait un bon choix pour notre prototype.

**2.2.1.2 API REST**

Une API REST offre interface de programmation respectant les contraintes du style d’architecture REST.

Une API RESTful doit remplir les critères suivants :

* Une architecture client-serveur constituée de clients, de serveurs et de ressources, avec des requêtes gérées via HTTP
* Des communications client-serveur [stateless](https://www.redhat.com/fr/topics/cloud-native-apps/stateful-vs-stateless), c'est-à-dire que les informations du client ne sont jamais stockées entre les requêtes GET, 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 Cela implique que :
* les ressources demandées soient identifiables et séparées des représentations envoyées au client ;
* les ressources puissent être manipulées par le client au moyen de la représentation reçue, qui contient suffisamment d'informations ;
* les messages autodescriptifs renvoyés au client contiennent assez de détails pour décrire la manière dont celui-ci doit traiter les informations ;
* l'API possède un hypertexte/hypermédia, qui permet au client d'utiliser des hyperliens pour connaître toutes les autres actions disponibles après avoir accédé à une ressource.
* 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 [ref url origine redhat]

Pour ce faire, notre choix s’est porté sur Django, un framework du langage Python offrant une facilite de conception grace aux nombreuses fonctionnalites deja incluses par defaut. Ajoute a l’emploi de modules comme DjangoRestFramework, il est possible de concevoir une API totalement conforme aux recommandations de la specification REST.

Fig. Python, Django de gauche vers droite

**2.2.1.3 Frontend**

La conception de l’interface utilisateur a necessite 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 u emise en page attrayante favorisant l’experience utilisateur.

TypeScript est un langage concu au dessus du langage JavaScript. Il vise notamment a ameliorer ce dernier en fournissant un systeme de typage fort. Cela permet entre autres de reduire la quantite de bugs qui finissent en production et d’ameliorer l’experience du developpeur.

Afin de faciliter l’integration de tous les outils suscites, nous avons fait recours au framework Vue.js. Vue est un framework moderne de conception d’application Web qui se concentre sur le rendu declaratif et composition de composants. Des solutions complementaires maintenues officiellement, permettent la gestion du routage, de l’etat et bien d’autres fonctionnalites comme les PWAs. Au vu des avantages qu’il presente, il correspond parfaitement aux besoins de notre plateforme en termes d’interfaces.

Fig. Logos de HTML, CSS, TypeScript et Vue, de la gauche vers la droite

**2.2.1.4 Base de données**

Les bases de donnees sont des elements cles dans la conception d’applications dynamiques. Elles permettent le stockage, le filtrage, la mise a jour et la suppression des donnees du systeme. On distingue generalement deux grandes familles de base de donnees: les bases relationnelles et les bases non relationnelles. La premiere preconise l’utilisation d’un schema fixe representant la structure de la donnee alors que le seconde permet une fexibilite du schema et autorise l’insertion de colomnes quelconques.

Notre choix s’est porte vers PostgreSQL, un systeme de gestion de base de donnees relationnel. C’est d’ailleurs, le seul SGDB Open Source , fournissant des fonctionnalites dignes de concurrencer les maisons d’edition comme Oracle.

Fig. PostgreSQL

**2.2.1.5 Gestionnaire de files de tâches**

Tourner des taches aux ressources intesives lors de requetes HTTP risque d’en degrader les performances. Pour eviter cela, nous avons recours a Celery, un gestionnaire de files de taches moderne qui offrent ue integration quasi-parfaire avec le framework Django.

Celery supporte un largde protocole de communication pour la planification des taches et la recuperation des resultats. Pour simplifier l’architecture, nous nous sommes servi du cache comme relai afin de declencher des taches.

**2.2.1.6 Cache**

En architecture logicielle, le cache est une composante essentielle. Il permet d’ameliorer les performances du systeme en gardant une copie (en memoire, a court terme) des donnees auxquelles les utilisateurs ont precedement acceder, sans qu’il n’y ait besoin de reprendre le meme processus de traitement de la requete. Ceci reduit le temps de reponse mais aussi reduit la pressions sur toute l’infrastructure. La base de donnees est moins sollicitee par exemple. Nous avons opte pour Redis, une solution Open Source qui utilise la memoire vive de la machine pour permettre un acces en lecture et en ecriture tres tres rapide.

Fig. Logo Redis

**2.2.1.7 Service Writepad**

Dans le cadre de la conception d’un service synchronise d’ecriture, nous avons employe l’editeur populaire Tiptap.

**2.2.1.8 Service Whiteboard**

Le service Writepad est un fork1(definition) d’un project Open source licensé sous MIT. Les modifications effectuees touchent principalement l’interface utilisateur mais aussi font omission de fonctionnalites qui ne nous interessent pas a l’etat actuel du prototype.

Fig. Image page gh et projet demo

**2.2.1.9 Serveur WebRTC**

Parmi les solutions Open Source pour la mise en place d’un serveur WebRTC, nous avons opte pour Mediasoup.

# **2.2.2 Outils de developpement**

Le tableau ci-dessous presente une liste non exhaustive des outils de developpement employes pour la realisation du prototype.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Outils de developpement | |  |
|  | Materiel | |  |
|  | Nom | Description |  |
| Laptop HP Probook 4540s | | Specifications:  Systeme d’exploitation: Manjaro Linux  8GB de RAM et 500GB de ROM, a servi au developpement et au test de la solution. | |
| Laptop Acer Es6 Aspire | | Specifications:  6GB de RAM et 1To de ROM  Systeme d’exploitation: Fedora  A servi unique pour les besoins de tests en reseau | |
| Laptop Lenovo Thinkbook | | Specifications:  Systeme d’exploitation: Manjaro Linux  32Gb de RAM et 1To de ROM  Utilise pour les besoins de developpement et de test | |
| Telephone Spark 8C | | Smartphone de 4Gb de RAM et de 64Gb de ROM utilise pour les tests et le verification des criteres d’accessibilite | |
| Routeur ZTE MF927U | | Un routeur moderne pour etablir une connectivite reseau entre les appareils | |
| Logiciels | | | |
| Nom | Description | | Version |
| EDIs Jetbrains | Suite de developpement logiciels offrant des fonctionnalites tres avancees | | Versions pro de Pycharm, CLion et WebStorm |
| neovim | Editeur modal de texte, qui peut servir d’EDI aussi. Il offre des fonctionnalites d’edition tres avancees. | | 0.8 |
| Git & GIthub | Outils de versionnement du code source | | GIt: 2.39.2 |
| Docker | OUtil open source de conteneurisations d’applications | | 23.0.1 |

**Conclusion**

Ce chapitre a permis de passer en revue les choix de conception ainsi que les choix techniques effectues pour la mise en oeuvre de notre prototype d’application. Le chapitre suivant, presentera la solution qui en resulte.

# **Matériels**

* **Rust**

Rust est un langage de programmation annoncé par Mozilla en 2010, compilé, multi-paradigme qui se veut fiable, concurrent et pratique >>. Il offre des performances proches de celles du C/C++. Il permet d’éviter les erreurs de segmentation ou de concurrence déjà à l'étape de compilation; tout ceci sans **Garbage Collector**, ce qui lui confère une excellente vitesse d’exécution. Il supporte les styles de programmation fonctionnelle, procédurale, modèle d’acteur et sous certains angles, orienté objet. C’est un langage fiable pour la conception d’applications web, les outils en ligne de commande (CLI/TUI), la programmation système, réseau ou embarquée. C’est la raison pour laquelle nous l’avons choisi pour le développement du backend de notre application web.

* **Actix**

Actix est un framework reposant sur le modèle d’acteurs 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 les *Websockets,* le *TLS* ou encore *HTTP/2.* Il offre d’excellentes performances, raison pour laquelle nous l’avons choisi.

* **Typescript**

Typescript est un langage libre et open-source développé par Microsoft. Il est trans compilé en JavaScript et peut donc être exécuté dans n’importe quel navigateur web ou moteur JavaScript. Il offre un typage statique optionnel des variables et des fonctions; la création et l’import de module. Le but principal de ce langage est de combler les lacunes de JavaScript et de permettre une expérience de développement plus agréable avec une détection des erreurs à l'étape de développement plutôt qu’en production.

* **Nuxt**

Nuxt est un méta framework basé sur le framework frontend Vue.js qui facilite la conception d’applications web universelles (rendues côté serveur SSR ou SPA). Il fournit une abstraction facilitant le développement d’applications Vue.js permettant aux développeurs de se consacrer à la logique métier.

* **WebRTC**

WebRTC (Web Real-Time Communication) est un projet open-source offrant une API de communication audio et vidéo aux navigateurs web et aux applications mobiles. Il offre une connexion P2P éliminant le besoin de plugins ou d’applications spécifiques pour la communication entre utilisateurs. Il est supporté par tous les navigateurs majeurs et ses spécifications sont maintenues par le W3C consortium et l’IETF (Internet Engineering Task Force).

Compilation des ressources pour la webographie et la bibliographie: <https://tobihans.notion.site/Refs-51187c42f1bb4e8b9b0b33cac975794d>