# **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.

# 

# 

1. **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 | Accès ouvert à un grand public sans restriction spatiale ou temporelle | illimité (en théorie) | asynchrone |
| SMOCs | Ouvert à un large nombre de participants et implique leur presence simultanee | illimité (en théorie) | synchrone |
| SPOCs | accès restreint qui requiert une forme de souscription au préalable | limité | asynchrone |
| SSOCs | Accès réservé à un petit nombre et impliquant la présence simultanées de tous les participants | limité | synchrone |

Fig. Tableau récapitulatif des types de cours en ligne

Notre étude s'intéresse notamment aux environnements d’apprentissage synchrones, avec le support d’un grand nombre d’apprenants, ce qui correspond bien aux 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 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.

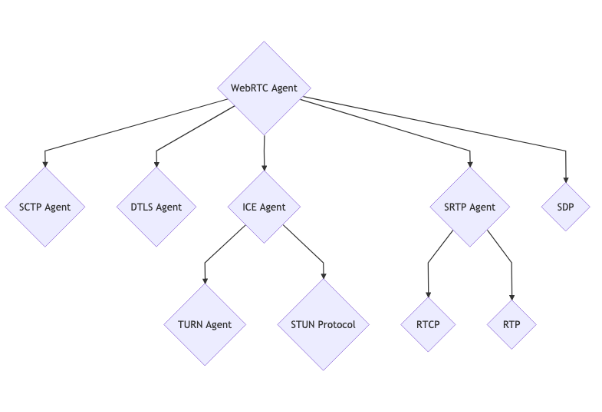


Fig. Protocoles employés par WebRTC

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 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.

**1.3.1. 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, a 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. Le modèle de souscription aussi, basé sur le nombre d’utilisateurs risque d'entraîner des frais assez élevés.

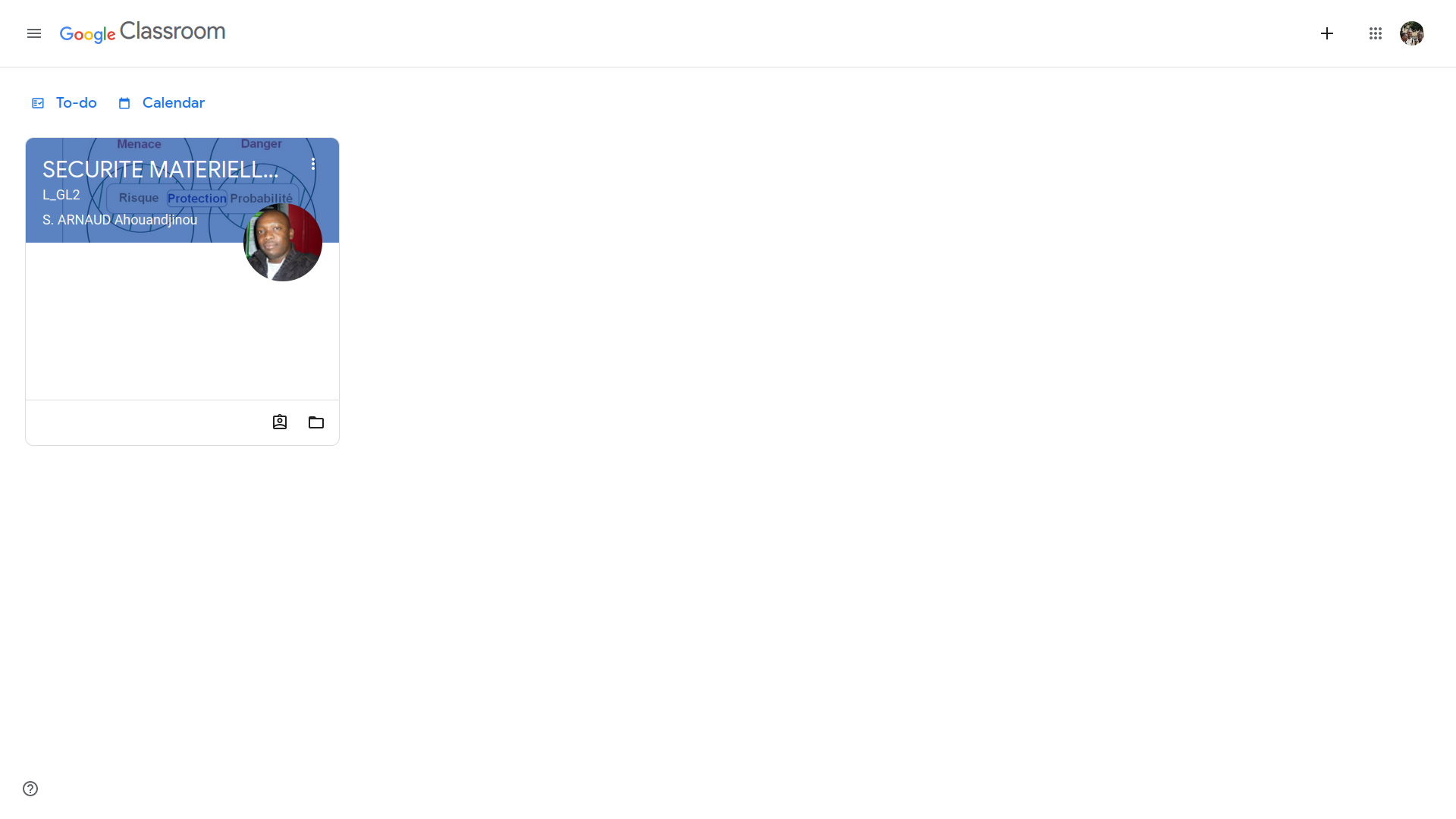


Fig. Interface d'accueil de Google Classrooms

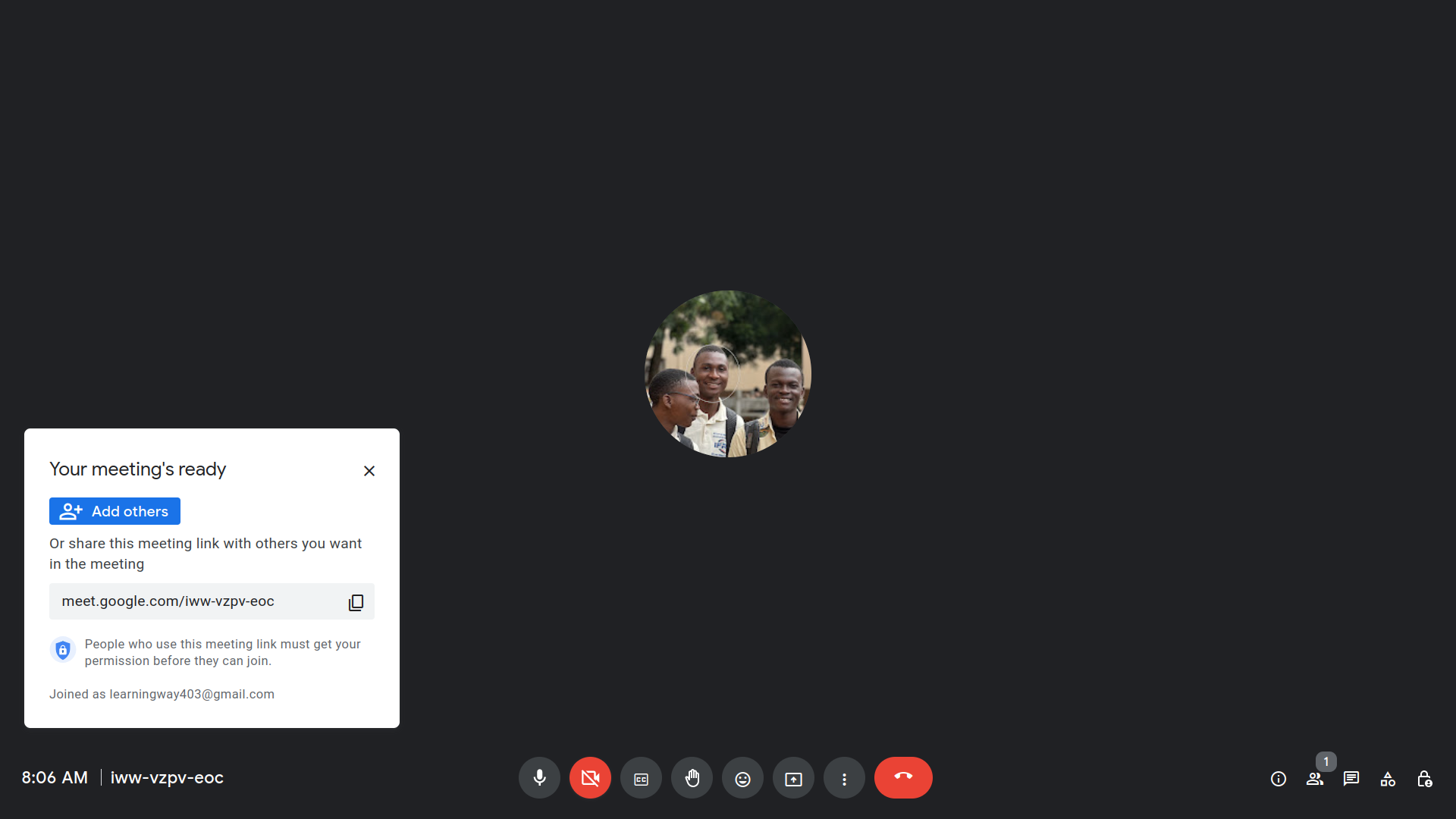


Fig. Interface de Google Meet

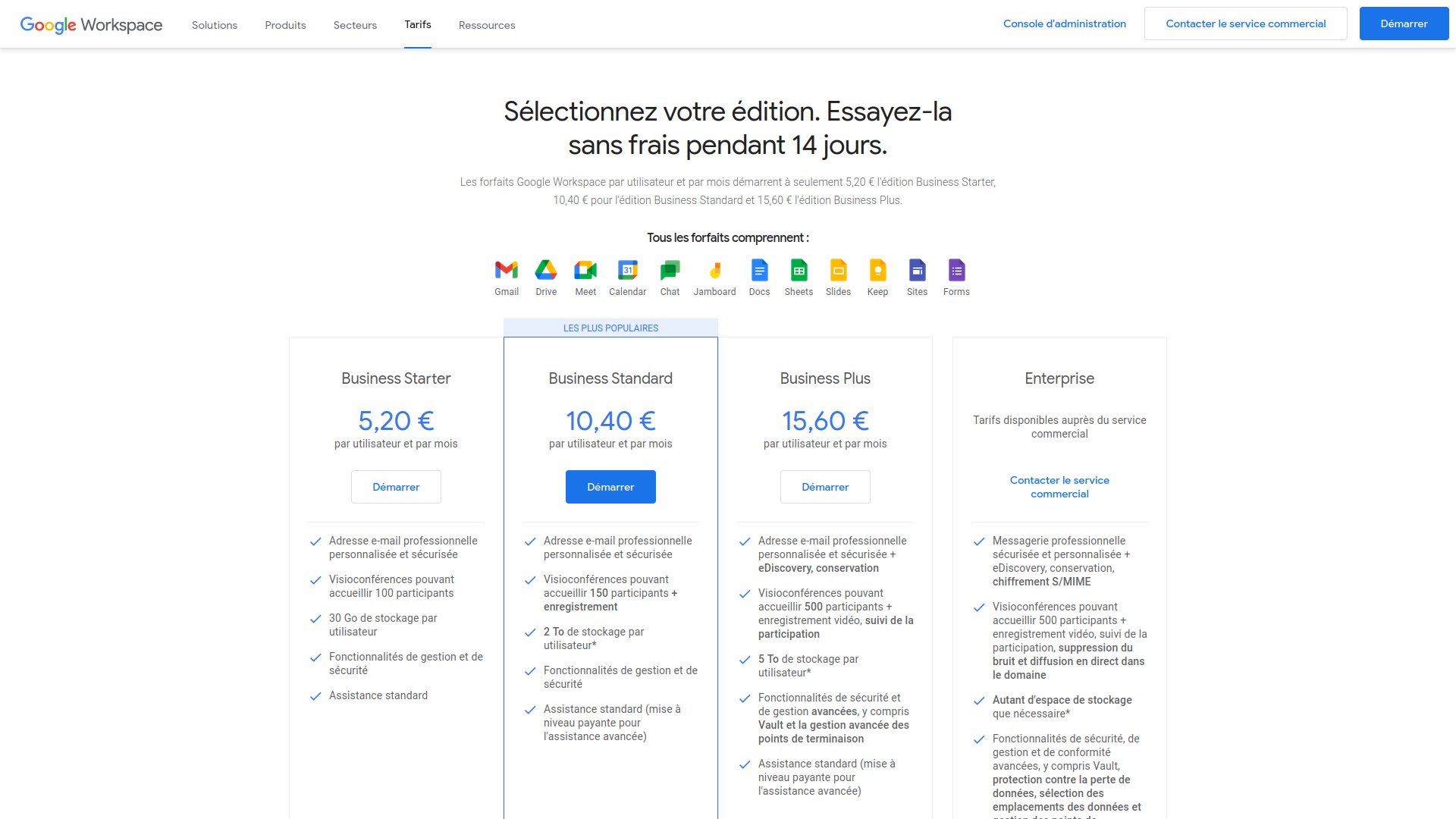


Fig. Page de souscription de la suite Google

**1.3.2. Zoom**

Zoom est un outil de communication 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 et donc, sans orientation, il est possible de choisir une solution inadéquate en rapport avec le besoin.



Fig. Interface de Zoom

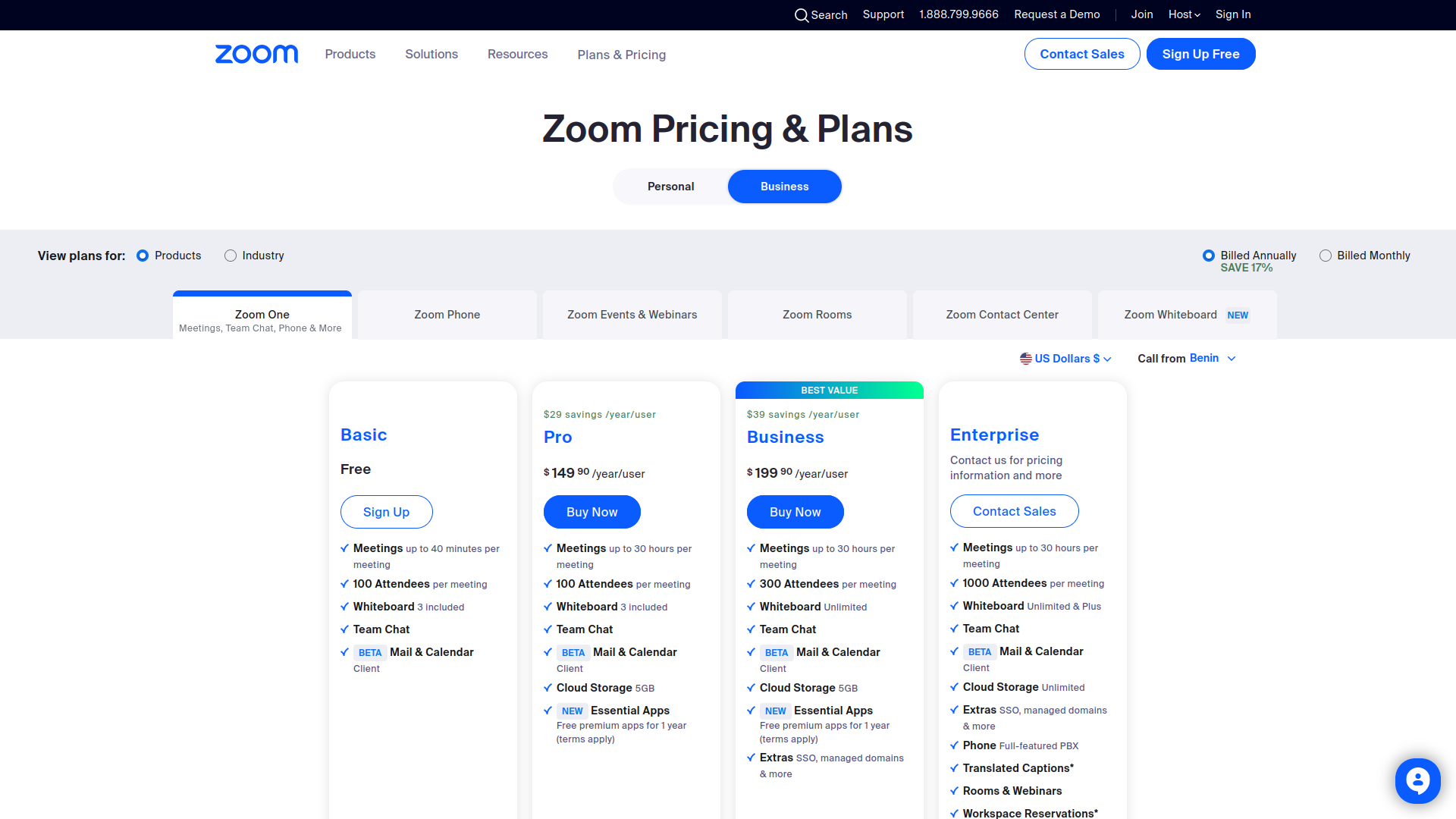


Fig. Page des souscriptions de Zoom

**1.3.3. Moodle**

Moodle est un LMS Open Source populaire 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 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.

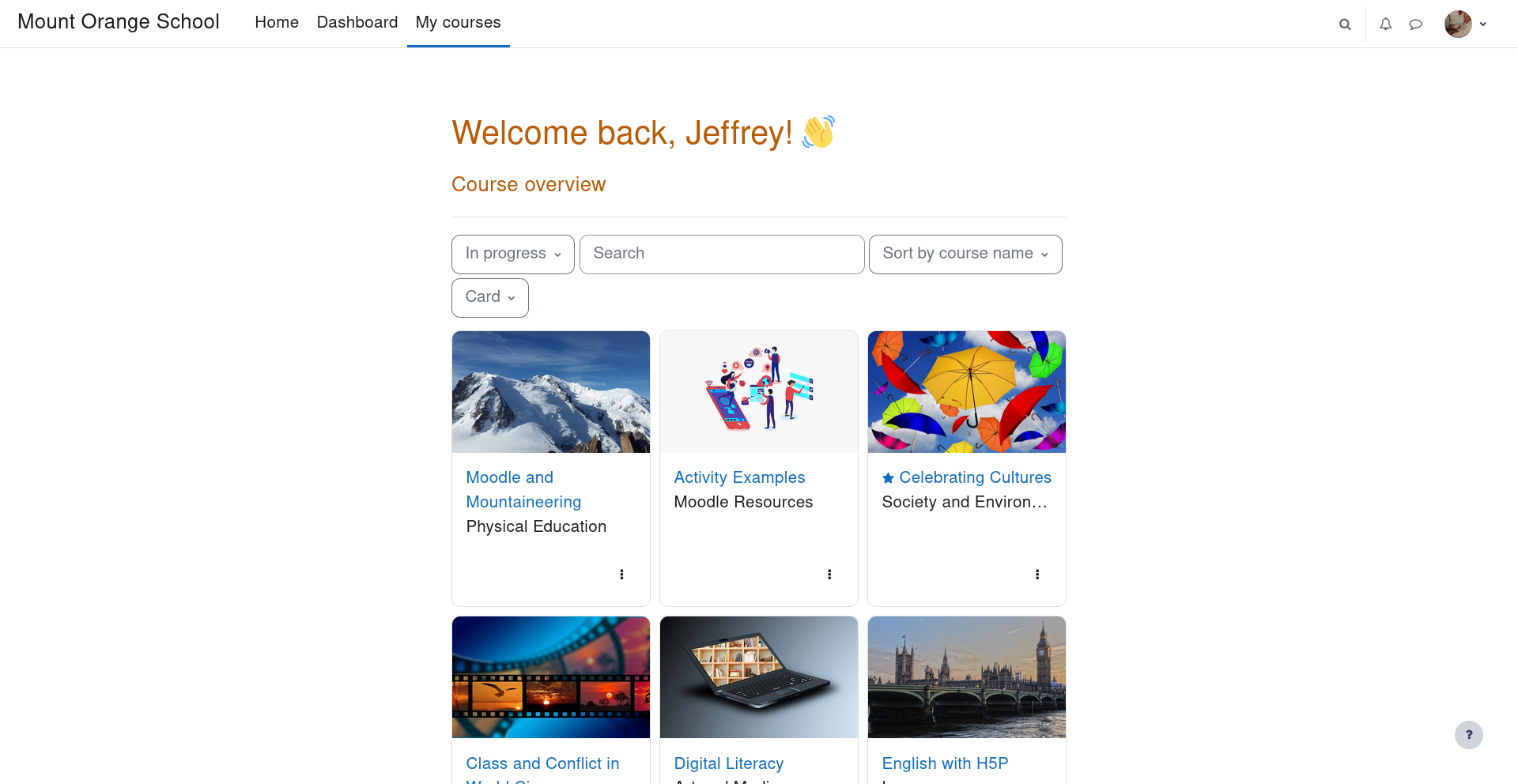


FIG. Site démo de Moodle

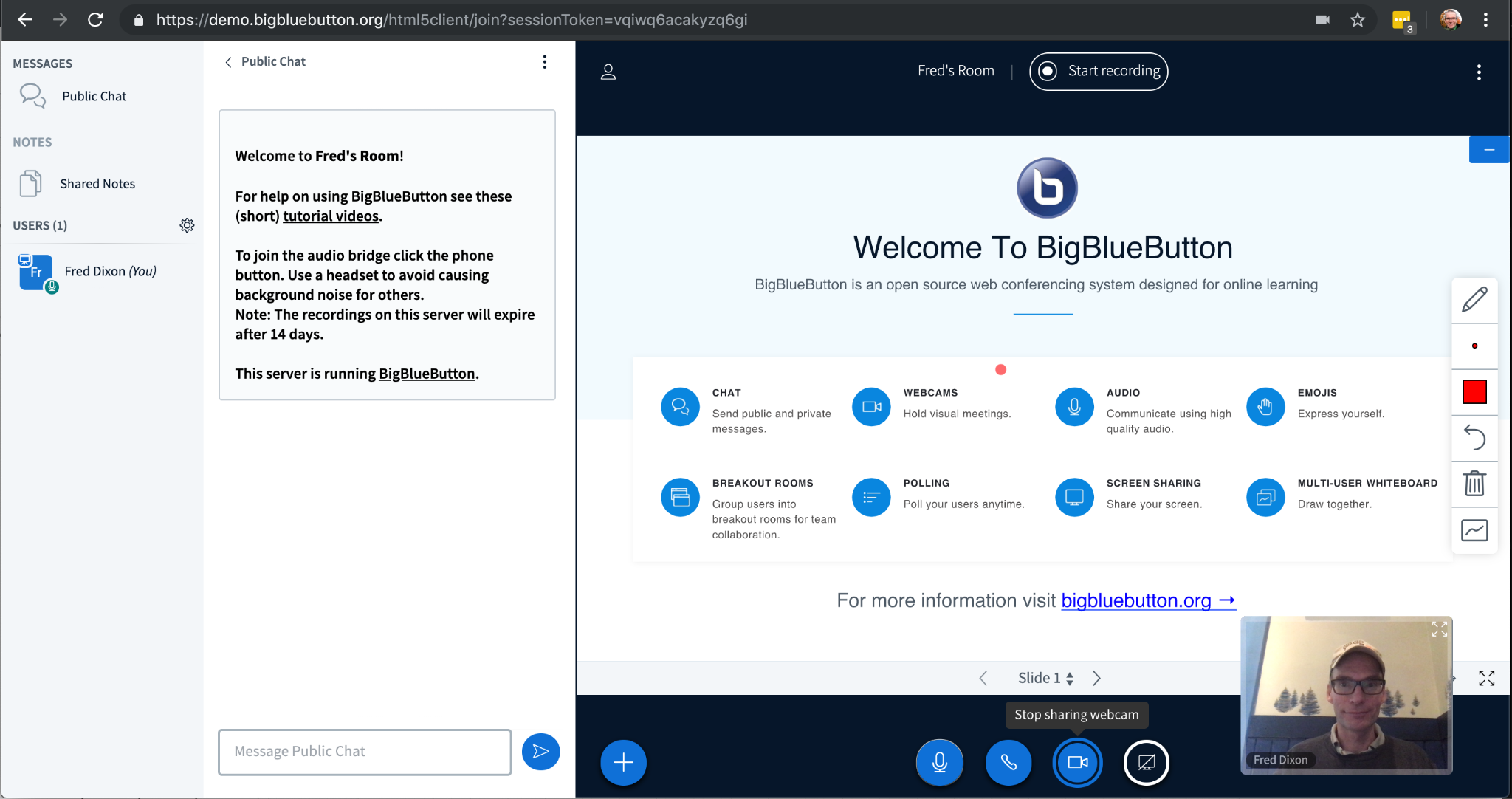


Fig. Démonstration de BigBluebutton

**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.**

# 

# **2. 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 UML. Il s’agit d’une méthode de visualisation d’architecture logicielle permettant de modéliser l’architecture logicielle d’un système.

Standardisé par l’OMG, la version actuelle de l’UML, la 2.5, 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’etudiant: 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 plateforme et l’administrer.

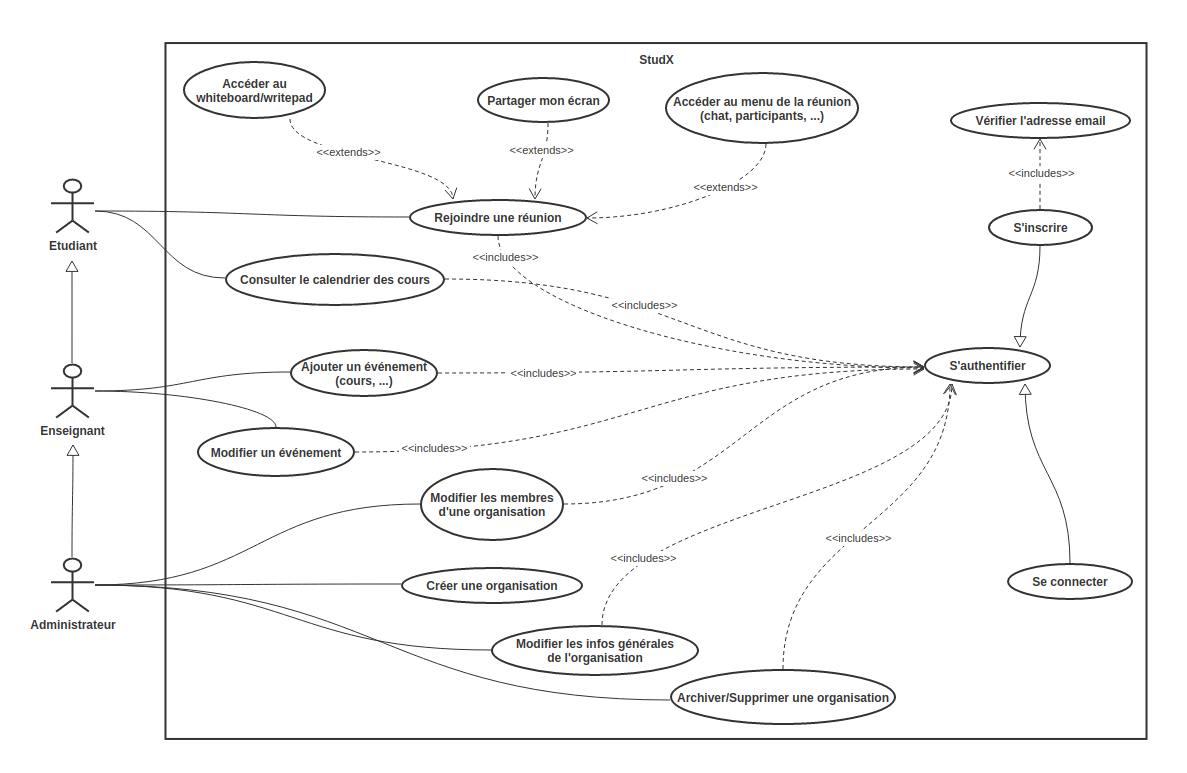


Fig. Diagramme de cas d’utilisation

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 “S’authentifier”, “Ajouter un événement” et “Rejoindre une réunion”.

**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 accède à la plateforme

Acteurs: Étudiant ou Enseignant ou Administrateur

Scenarii:

Pré-conditions:

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

Séquence nominale:

1. L’utilisateur remplit le formulaire de connexion

2. Le système authentifie l’utilisateur

3. Le système redirige l’utilisateur vers la page la page d'accueil de l’organisation la plus récente.

Post-conditions:

- L’utilisateur dispose d’une session active

Enchaînements alternatifs:

A1: Identifiants incorrects

L'enchaînement démarre au point 2

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

Ce dernier effectue à nouveau la saisie. L'enchaînement reprend au point 3.

A2: L’utilisateur n’appartient à aucune organisation

L'enchaînement prend départ au point 3.

3. Le système 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 événement

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

Acteurs: Enseignant ou Administrateur

Scenarii:

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 programmes

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 reunion”**

Sommaire d’identification:

Titre: Rejoindre une réunion

Objectif: Tenir une session de classe en ligne

Acteurs: Étudiant ou Enseignant ou Administrateur

Scenarii:

Pré-conditions:

- l’utilisateur est authentifié

Séquence nominale:

1. L’utilisateur consulte le calendrier

2. Le système affiche les divers événements programmes

3. L’utilisateur accède aux détails d’un événement

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

5. Le système connecté 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 scénarii évoqués au niveau des cas d’utilisations. Ci-dessous, sont illustrés le diagrammes de séquences pour les trois cas d’utilisation suscités.

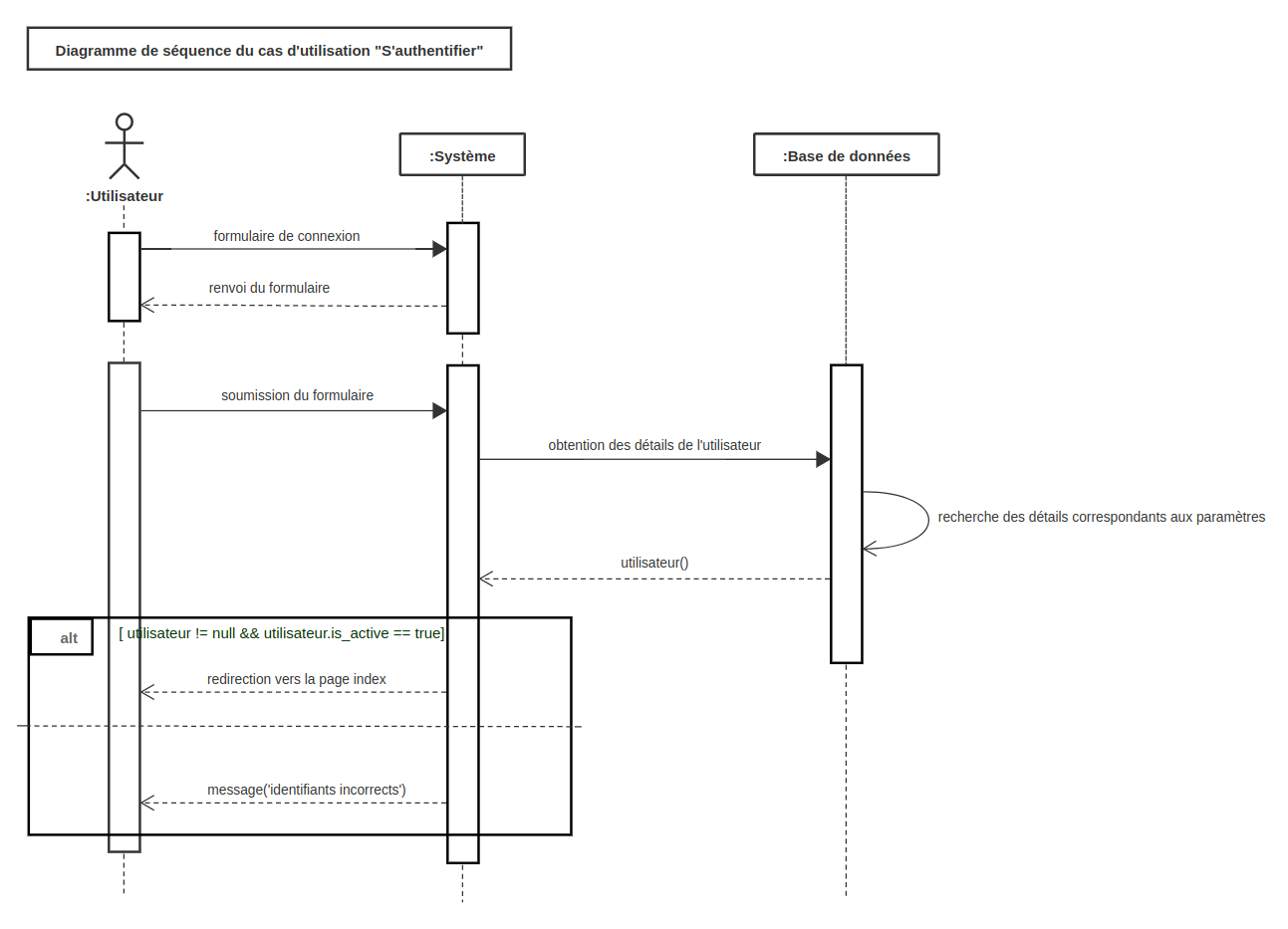


Fig. Diagramme de séquence du cas d'utilisation “S’authentifier”

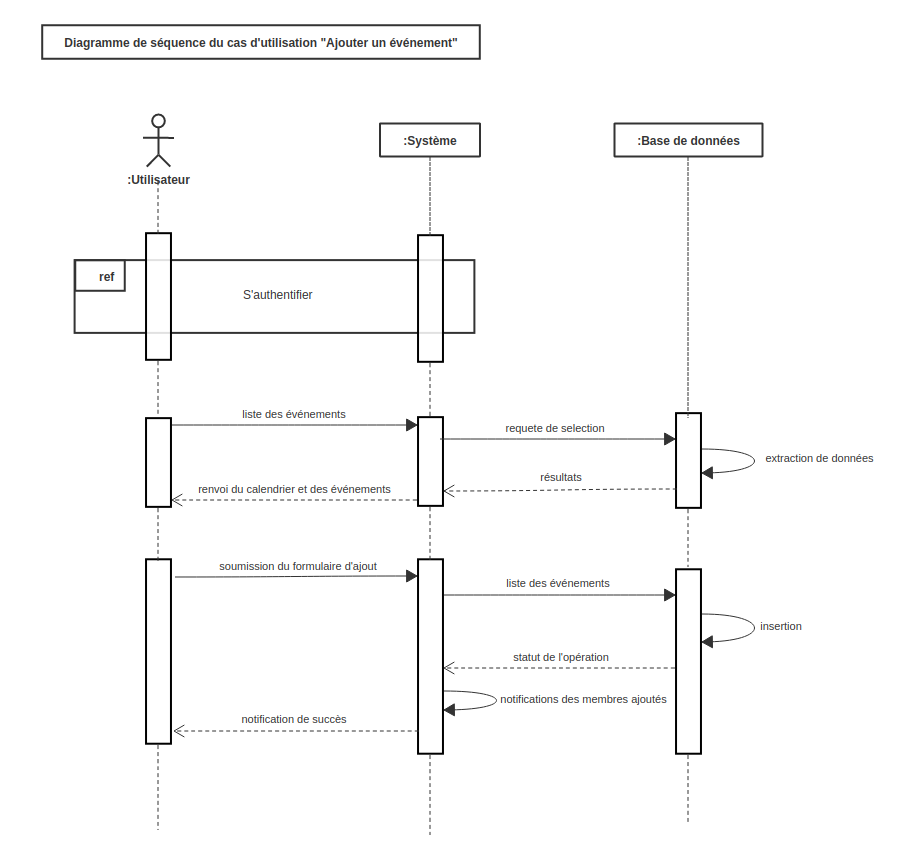


Fig. Diagramme de séquence pour le cas d’utilisation “Ajouter un événement”

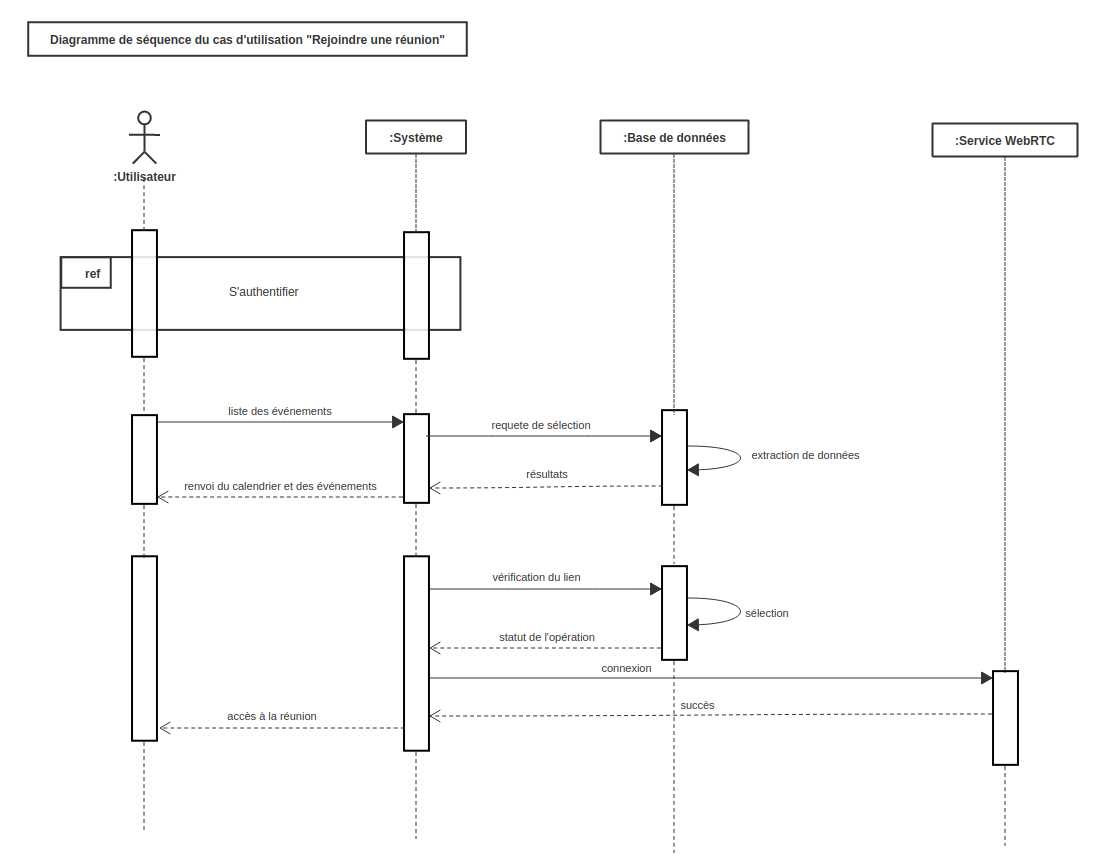


Fig. Diagramme de séquence pour le cas d’utilisation “Ajouter un événement”

# **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 ci-dessous décrit les entités de notre prototype.

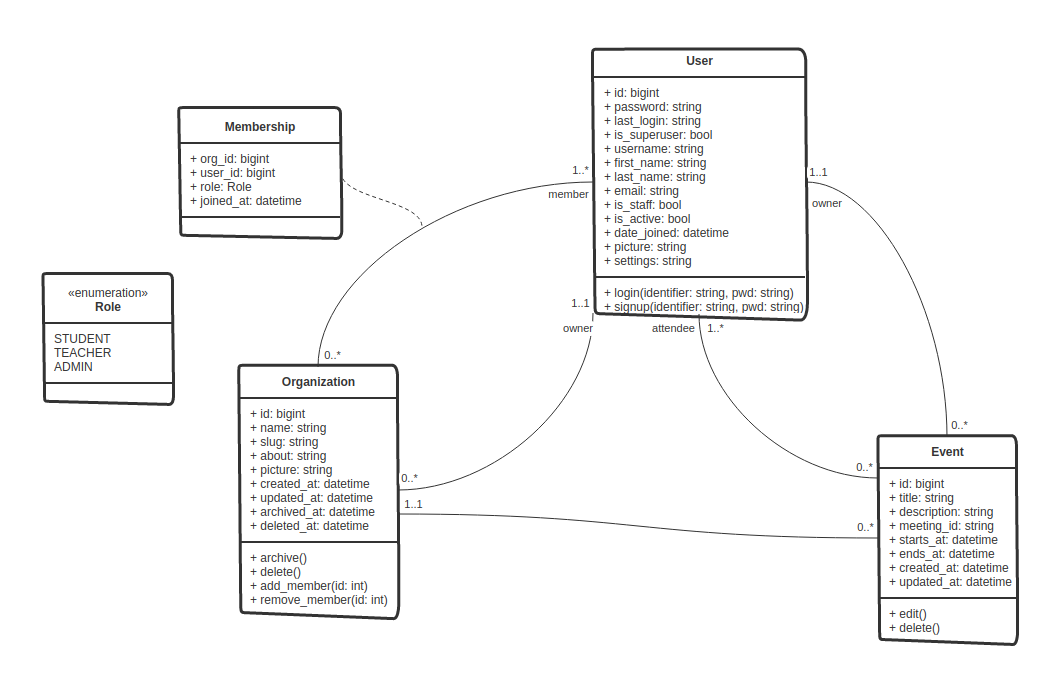
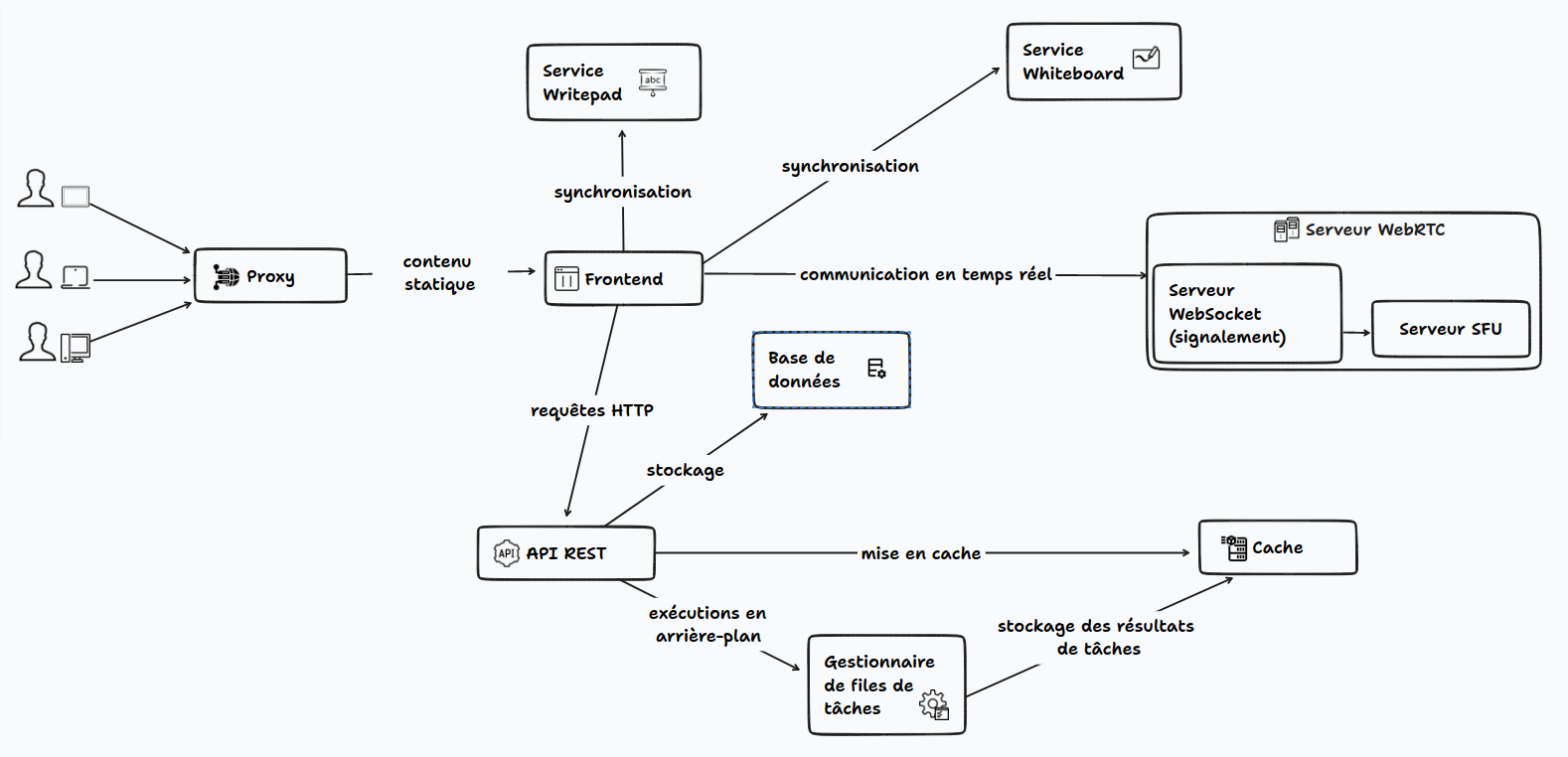


Fig. Diagramme de classe

# **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 découplée, 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 la résilience. Voici ci-dessous, une illustration des diverses composantes de notre architecture.



On peut notamment remarquer que toutes les interactions entre le système et les utilisateurs passent toutes par un proxy. Ceci s’explique par la volonté d'éviter les problèmes de CORS, qui occurrent dès lors que les services ne sont pas tous sur un même domaine.

**2.2. Méthodes de conception**

# **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.

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**. En voici un exemple, extrait du code de notre prototype:

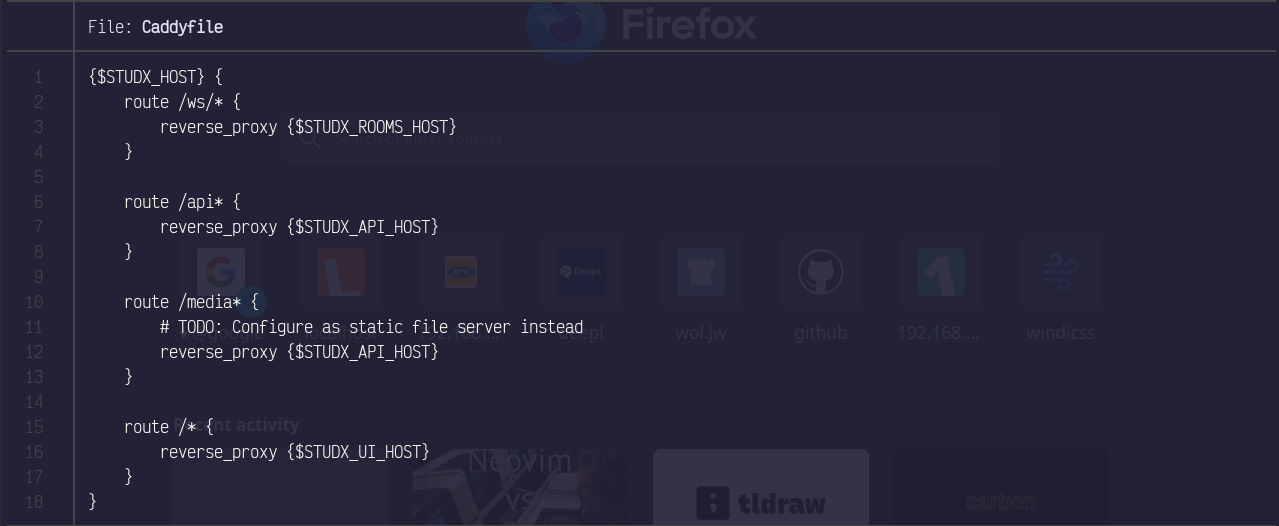


Fig. Caddyfile

On peut remarquer l’utilisation de variables d’environnement qui permettent de rendre la configuration encore plus dynamique. Tous ces atouts en font 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-serveursans etat, 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 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 auto descriptifs 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 facilité de conception grâce aux nombreuses fonctionnalités déjà incluses par défaut. Ajoutez à l’emploi de modules comme Django Rest Framework, il est possible de concevoir une API totalement conforme aux recommandations de la spécification REST.

Fig. Python, Django de gauche vers droite

**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.

**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 le 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 SGBD Open Source , fournissant des fonctionnalités dignes de concurrencer les maisons d'édition comme Oracle.

**2.2.1.5 Gestionnaire de files de tâches**

Tourner des tâches aux 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 protocole 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.

**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.

**2.2.1.7 Service Writepad**

Dans le cadre de la conception d’un service synchronisé d'écriture, nous avons employé l'éditeur populaire Tip Tap. 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.

**2.2.1.8 Service Whiteboard**

Le service Writepad est un fork1(définition) d’un projet Open source sous licence MIT, dénommé **whiteboard**. Il est consultable a 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 présente l’interface par défaut.

Fig. Image page gh et projet demo

**2.2.1.9 Serveur WebRTC**

Rust est un langage compilé qui se veut performant, sûr et productif [url rust-lang.org]. Le langage peut notamment donner des garanties d'absence d'[erreur de segmentation](https://fr.wikipedia.org/wiki/Erreur_de_segmentation) ou de [situation de concurrence](https://fr.wikipedia.org/wiki/Situation_de_concurrence)[14](https://fr.wikipedia.org/wiki/Rust_(langage)#cite_note-14) dès l'étape de compilation. De plus, ceci se fait sans [ramasse-miettes](https://fr.wikipedia.org/wiki/Ramasse-miettes_(informatique))[15](https://fr.wikipedia.org/wiki/Rust_(langage)#cite_note-15). Ses performances sont comparables à celles de C ou C++[16](https://fr.wikipedia.org/wiki/Rust_(langage)#cite_note-16) pour ce qui concerne la vitesse d'exécution [wiki-rust]. 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 [lien mediasoup]. C’est un serveur de relai qui offre une bibliothèque Rust pour l'intégration dans diverses catégories d'applications.

Fig. Exemple de code Rust

Actix est un framework reposant sur le modèle d’acteurs [url-wiki] 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, pour la gestion du signalement lors des connexions WebRTC.

# **2.2.2 Outils de développement**

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

| Outils de développement | | | |
| --- | --- | --- | --- |
| Matériel | | | |
| Nom | | Description | |
| Laptop HP Probook 4540s | | Spécifications:  Système d’exploitation: Manjaro Linux  8GB de RAM et 500GB de ROM, a servi au développement et au test de la solution. | |
| Laptop Acer Es6 Aspire | | Spécifications:  6GB de RAM et 1To de ROM  Système d’exploitation: Fedora  A servi unique pour les besoins de tests en réseau | |
| Laptop Lenovo Thinkbook | | Spécifications:  Système d’exploitation: Manjaro Linux  32Gb de RAM et 1To de ROM  Utilisé pour les besoins de développement et de test | |
| Téléphone Spark 8C | | Smartphone de 4Gb de RAM et de 64Gb de ROM utilise pour les tests et la vérification des critères d'accessibilité | |
| Routeur ZTE MF927U | | Un routeur moderne pour établir une connectivité réseau entre les appareils | |
| Logiciels | | | |
| Nom | Description | | Version |
| EDIs Jetbrains | Suite de developpement logiciels offrant des fonctionnalités très avancées | | Versions professionnelles de Pycharm, CLion et WebStorm |
| neovim | Éditeur modal de texte, qui peut servir d’EDI aussi. Il offre des fonctionnalités d'édition très avancées. | | 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 effectués pour la mise en œuvre de notre prototype d’application. Le chapitre suivant présentera la solution qui en résulte.