
Plateforme de Chat en Temps Réel avec Appels Vidéo et Vocaux

Réalisé par :

- AIT OUAKOUR Ayoub
 - ABOURAHIM Wissal
 - DOUMAS YOUSRA
 - QASID KARIM
-

Professeur :

- LEMGHARI Nidal
-

SOMMAIRE

Introduction generale

1. Présentation du projet

1.1 Introduction.

1.2 Présentation générale du projet

1.2.1 Problématique

1.2.2 Etude de l'existant..

1.3. Objectifs du projet.

1.4. Solution proposée.

2. Spécification des besoins.

2.1 Introduction

2.2 Besoins fonctionnels

2.3 Besoins non-fonctionnels.

3. Architecture et conception

3.1 Introduction

3.4 Conception détaillée

4. Réalisation

4.1 Introduction

4.2 Environnement logiciel

4.3 Environnement technique

4.4 Travail réalisé

5. Conclusion



Introduction générale

- Dans un monde de plus en plus interconnecté, la communication instantanée est devenue une nécessité fondamentale, tant dans la sphère personnelle que professionnelle. Les plateformes de messagerie et d'appels en temps réel jouent un rôle central dans le quotidien des utilisateurs, facilitant les échanges à distance et supprimant les barrières géographiques. Face à cette évolution, les technologies de communication ont connu un essor considérable, intégrant des fonctionnalités toujours plus avancées telles que le chat en temps réel, les appels vocaux et les visioconférences.
- C'est dans ce contexte que s'inscrit notre projet de Plateforme de Chat en Temps Réel avec Appels Vidéo et Vocaux. L'objectif principal est de concevoir et développer une application web moderne et interactive qui permet aux utilisateurs de communiquer instantanément par messages textuels, tout en ayant la possibilité de passer des appels vocaux et vidéos, individuellement ou en groupe. Ce projet vise à offrir une expérience utilisateur fluide, sécurisée et accessible, en s'appuyant sur des technologies performantes et des protocoles de communication adaptés.
- Ce rapport présente les différentes étapes de réalisation de la plateforme, en partant de l'étude des besoins jusqu'au développement et aux tests de l'application. Il met également en lumière les choix technologiques, les défis rencontrés et les solutions apportées tout au long du processus.



Chapitre 1

Présentation du projet

1.1 Introduction

- Dans le cadre de notre formation à l'ENSA Khouribga, nous avons conçu et développé une plateforme de chat en temps réel offrant des fonctionnalités avancées, telles que la messagerie instantanée et les appels vocaux et vidéo. Ce projet vise à fournir une solution de communication moderne, intuitive et sécurisée.
- La modélisation constitue une étape clé dans le cycle de vie du projet, permettant de structurer les besoins, d'analyser les interactions entre les différents acteurs et de concevoir une architecture logicielle robuste. À travers l'utilisation de diagrammes UML et une conception approfondie de la base de données, nous définissons les principales entités du système, leurs relations et les flux d'échange d'informations.
- Cette démarche garantit une meilleure organisation du code, une évolutivité facilitée et une cohérence fonctionnelle dans le développement de l'application
- Ce rapport présente les différentes étapes de la modélisation et de la conception du projet.



1.2 Présentation générale du projet

1.2.1 Problématique

- À l'ENSA de Khouribga, la collaboration à distance entre étudiants est devenue essentielle dans les projets, travaux pratiques et recherches. Toutefois, les outils existants (Zoom, Teams, WhatsApp) présentent des limites : surcharge fonctionnelle, forte consommation de ressources, dépendance à des services tiers, et confidentialité parfois insuffisante.
- Dès lors, une problématique se pose :

Comment concevoir une plateforme web de communication en temps réel, légère, sécurisée, et adaptée aux besoins académiques, intégrant messagerie instantanée, appels vocaux et vidéo ?

Ce projet implique plusieurs défis :

- Choix de technologies open-source (WebRTC, WebSockets)
- Gestion de connexions simultanées
- Qualité de communication en faible bande passante
- Sécurité des échanges
- Interface simple et ergonomique pour un usage académique

1.2.2 Etude de l'existant

Dans le cadre du développement d'une plateforme de communication en temps réel adaptée à un usage académique, il est important d'analyser les outils actuellement disponibles. Plusieurs applications sont aujourd'hui largement utilisées dans le monde de l'éducation et de la collaboration, chacune avec ses forces et ses limites.

- **Discord** est une plateforme polyvalente offrant chats textuels, vocaux et vidéo. Cependant, son interface peut paraître complexe et certaines fonctions trop distrayantes pour un usage académique classique.
- **Telegram** met l'accent sur la rapidité, la sécurité et la création de groupes avec des fonctionnalités comme les canaux, bots, et appels vocaux et vidéo de groupe. Néanmoins, il ne propose pas encore de visioconférence complète adaptée aux réunions interactives.
- **WhatsApp**, très utilisé pour les messages et appels vidéo, présente des limites pour les groupes académiques, notamment l'absence de support performant pour les appels vidéo de groupe nombreux et de fonctionnalités professionnelles avancées.

Malgré la popularité de ces plateformes, elles présentent plusieurs limitations dans le cadre académique :

- **Complexité** : Les fonctionnalités avancées peuvent rendre l'utilisation de certaines solutions trop complexe pour des utilisateurs cherchant une plateforme simple.
- **Coût et dépendance** : La plupart des solutions exigent un abonnement ou une inscription à un service externe, ce qui n'est pas toujours pratique dans un environnement académique.
- **Confidentialité** : La gestion des données personnelles sur certaines plateformes externes pose des questions de sécurité, surtout dans un cadre universitaire.

1.3 Objectifs du projet.

l'objectif principal de ce projet est de concevoir et développer une plateforme de communication en temps réel intégrant des fonctionnalités avancées telles que la messagerie instantanée, les appels vocaux et vidéo. Cette solution vise à offrir une expérience utilisateur fluide, intuitive et sécurisée tout en garantissant la rapidité et la fiabilité des échanges.

Pour atteindre cet objectif global, nous avons défini les sous-objectifs suivants :

- Développer une messagerie instantanée permettant aux utilisateurs d'échanger des messages textuels en temps réel avec un historique des conversations.
- Intégrer des appels vocaux et vidéo via WebRTC, permettant aux utilisateurs de communiquer de manière plus immersive et interactive.
- Assurer une expérience utilisateur fluide avec une interface ergonomique et réactive, conçue avec React.js.
- Garantir la sécurité et la confidentialité des échanges grâce à un système d'authentification sécurisé basé sur JWT (JSON Web Tokens) et au chiffrement des données.
- Mettre en place un système de notifications en temps réel, informant les utilisateurs des nouveaux messages, des appels entrants et des invitations à des conversations.
- Optimiser la gestion des contacts, permettant aux utilisateurs d'ajouter, supprimer et organiser leurs relations tout en gérant leur statut (en ligne, hors ligne, occupé).



1.4 Solution proposée

Pour répondre efficacement aux objectifs définis, nous proposons une solution complète reposant sur une architecture moderne et modulaire. La plateforme sera développée en utilisant React.js pour garantir une interface utilisateur ergonomique, fluide et responsive. La messagerie instantanée sera implémentée à l'aide de Socket.io, assurant des échanges textuels en temps réel avec conservation de l'historique des conversations. Pour les appels vocaux et vidéo, nous intégrerons la technologie WebRTC, reconnue pour sa capacité à établir des communications peer-to-peer sécurisées et performantes. La sécurité des données sera assurée par une authentification via JWT (JSON Web Tokens). Enfin, la gestion des contacts sera optimisée avec des fonctionnalités permettant d'ajouter, supprimer pour favoriser une communication efficace et personnalisée.



Chapitre 2

Spécification des besoins

2.1 Introduction

Après avoir étudié les solutions existantes et identifié leurs limites, il est nécessaire de définir précisément les besoins auxquels notre plateforme doit répondre. Cette étape est essentielle pour cadrer le développement de l'application en fonction des attentes des utilisateurs finaux, à savoir les étudiants et les enseignants.

La spécification des besoins permet de traduire les exigences fonctionnelles et techniques du projet en objectifs concrets, en décrivant ce que le système doit faire (besoins fonctionnels), les contraintes à respecter (besoins non fonctionnels), ainsi que l'ergonomie attendue.

Cette phase constitue donc une base fondamentale pour la conception, le développement et la validation de la plateforme, en assurant que celle-ci répondra efficacement aux problématiques de communication relevées dans le contexte éducatif.



2.2 Besoins fonctionnels

Les besoins fonctionnels décrivent les fonctionnalités principales que la plateforme de chat en temps réel doit offrir aux utilisateurs.

Authentification et gestion des utilisateurs


- Permettre aux utilisateurs de créer un compte avec une adresse e-mail et un mot de passe.
- Authentifier les utilisateurs via un système sécurisé (JWT).
- Gérer les profils utilisateurs (modification des informations personnelles, ajout d'une photo de profil).
- Gérer les statuts des utilisateurs (en ligne, hors ligne, occupé).

Messagerie instantanée

- Permettre aux utilisateurs d'envoyer et recevoir des messages en temps réel.
- Enregistrer l'historique des conversations et permettre l'accès aux anciens messages.
- Prendre en charge les conversations de groupe et permettre l'ajout ou la suppression de participants.
- Supporter l'envoi de fichiers et d'images dans les messages.

Appels vocaux et vidéo

- Offrir la possibilité de passer des appels vocaux et vidéo en temps réel avec un ou plusieurs utilisateurs.
- Permettre de désactiver le micro ou la caméra pendant l'appel.



Gestion des contacts et des conversations

- Ajouter, supprimer et bloquer des contacts.
- Consulter la liste des contacts avec leurs statuts en temps réel.
- Recevoir des notifications en cas de nouveaux messages ou d'appels entrants.
- Supprimer ou archiver des conversations.

Sécurité et confidentialité

- Chiffrer les messages échangés pour assurer leur confidentialité.
- Permettre aux utilisateurs de configurer leurs paramètres de confidentialité (ex. : qui peut les contacter, voir leur statut).
- Mettre en place un système de signalement et de blocage des utilisateurs indésirables.



2.3 Besoins non fonctionnels

Les besoins non fonctionnels définissent les critères de performance, de sécurité et d'ergonomie que la plateforme doit respecter.

Performance et scalabilité

- Assurer une latence minimale pour les échanges en temps réel.
- Supporter un grand nombre d'utilisateurs simultanés sans perte de performance.
- Optimiser l'architecture pour assurer une montée en charge progressive.

Sécurité

- Utiliser JWT pour l'authentification et chiffrer les mots de passe en base de données (bcrypt).
- Protéger les données échangées à l'aide du chiffrement de bout en bout.

Expérience utilisateur (UX/UI)

- Fournir une interface intuitive et ergonomique pour une prise en main facile.
- Garantir la compatibilité avec les différents appareils (desktop, mobile, tablette).
- Assurer une navigation fluide et réactive avec React.js.



Chapitre 3

Architecture et conception

3.1 Introduction

Dans le présent chapitre, nous allons entamer une partie importante du développement de l'application qui constitue un pont entre la spécification et la réalisation.

Nous allons présenter dans un premier temps la conception détaillée comprenant les vues statiques

3.2 Conception detaille

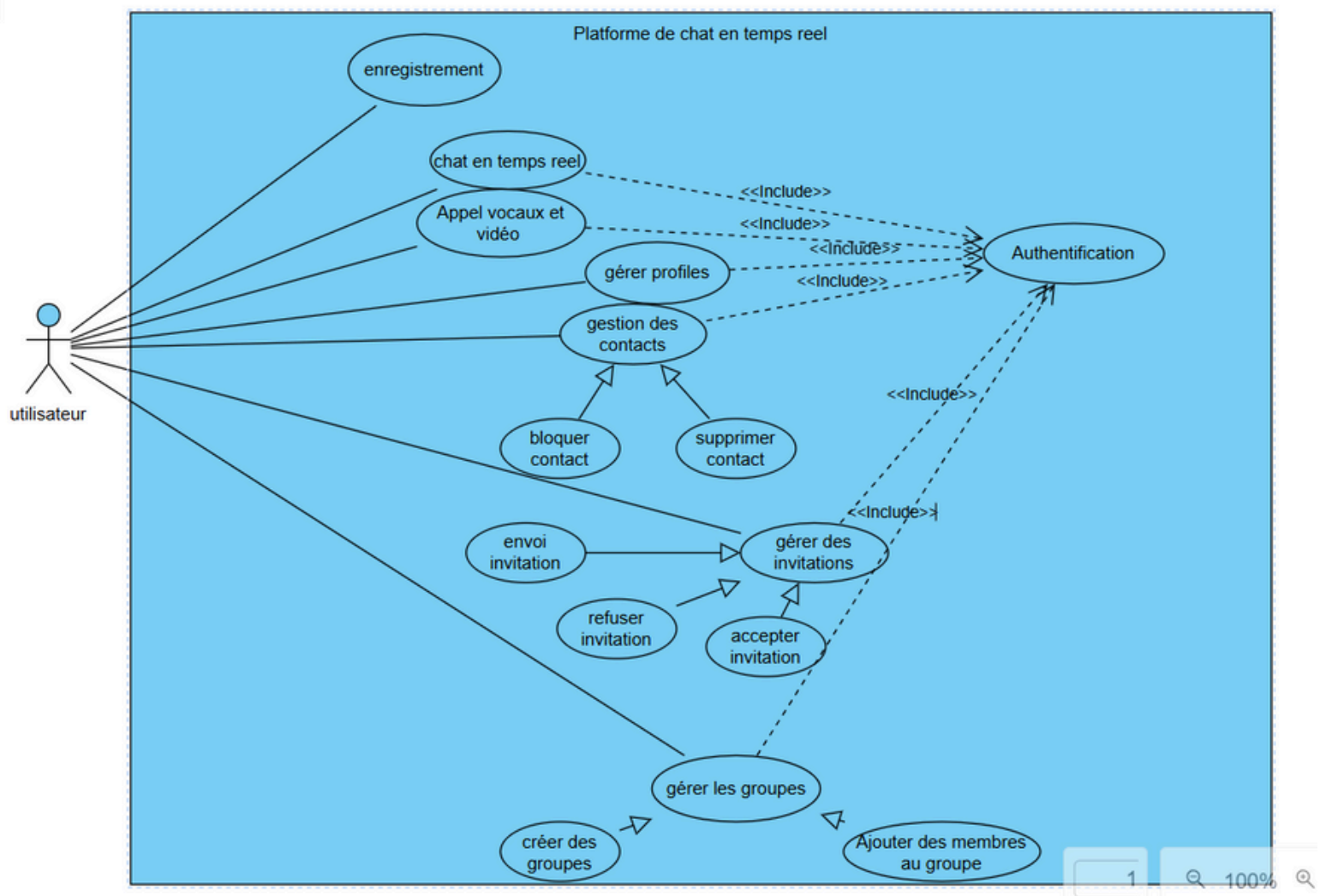
Diagramme de cas d'utilisation :

L'image ci-dessous représente le diagramme de cas d'utilisation de la plateforme de chat en temps réel. Ce diagramme illustre les principales interactions entre l'utilisateur et les différentes fonctionnalités du système.

L'utilisateur peut effectuer plusieurs actions essentielles :

- Enregistrement et authentification : Permet à l'utilisateur de créer un compte et de se connecter à la plateforme.
- Chat en temps réel : Fonctionnalité de messagerie instantanée entre les utilisateurs.
- Appels vocaux et vidéo : Possibilité d'effectuer des appels en audio ou en vidéo.
- Gestion de profil : Modification des informations personnelles et des préférences de l'utilisateur.
- Gestion des contacts : Inclut l'ajout, la suppression et le blocage de contacts.

- Gestion des invitations : Permet l'envoi, l'acceptation et le refus d'invitations.
- Gestion des groupes : Comprend la création de groupes et l'ajout de membres à un groupe.
- Ce diagramme met en évidence la manière dont les différents cas d'utilisation sont interconnectés, offrant ainsi une vue globale des interactions possibles au sein de la plateforme.



• Description textuelle :

Partie 1 : IDENTIFICATION

Titre: Gestion des invitations

Résumé: Ce cas d'utilisation permet à l'utilisateur de gérer les invitations reçues et envoyées (acceptation, refus, consultation et annulation).

Acteurs Principal : Utilisateur

Secondaire : Système de notification

Dates: 29/03/2025

Responsable :

Version :

Partie 2: Description des scénarios

Pré-conditions :

- * L'utilisateur est authentifié.
- * Le système est opérationnel.
- * Des invitations ont été envoyées ou reçues.
- * Scénarios :
- * Scénario nominal (Acceptation d'une invitation) :
 1. L'utilisateur consulte la liste des invitations reçues.
 2. Il sélectionne une invitation et clique sur "Accepter".
 3. Le système valide l'acceptation et ajoute le contact à la liste.
 4. Une notification est envoyée à l'expéditeur.

Scénarios alternatifs :

Refus d'une invitation :

1. L'utilisateur consulte les invitations.
2. Il clique sur "Refuser".
3. Le système supprime l'invitation et notifie l'expéditeur.

Annulation d'une invitation envoyée :

1. L'utilisateur consulte ses invitations envoyées.
2. Il sélectionne une invitation et clique sur "Annuler".
3. Le système supprime l'invitation



Scénarios d'exception :

Erreur réseau : Le système affiche un message d'erreur et propose de réessayer.

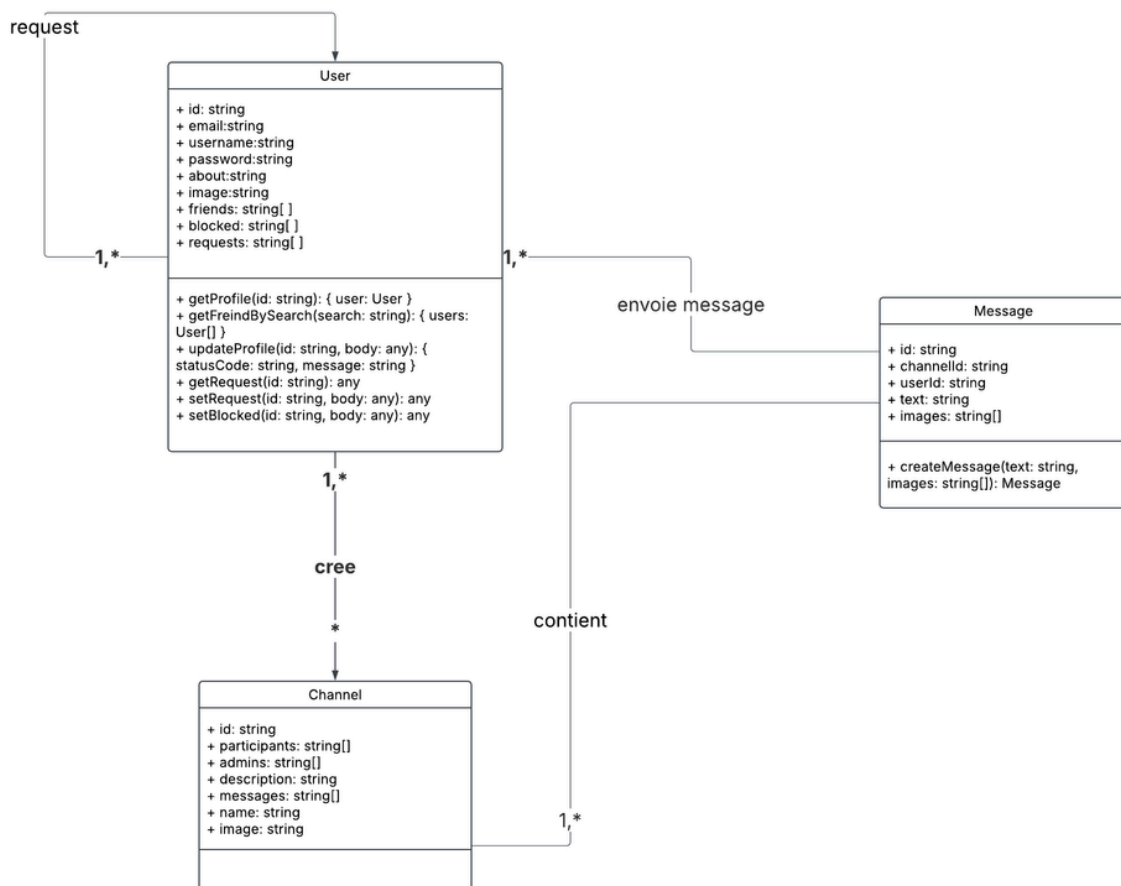
Invitation expirée : Le système informe l'utilisateur que l'invitation n'est plus valide

Post-conditions :

- * La liste des contacts est mise à jour (si acceptation).**
- * Les invitations refusées/annulées sont supprimées.**
- * Les notifications sont envoyées aux parties concernées.**

Diagramme de classe:

- Le diagramme de classes est un outil essentiel en modélisation orientée objet, permettant de représenter la structure statique d'un système. Il illustre les classes, leurs attributs, leurs méthodes et les relations entre elles. Ce diagramme est crucial pour comprendre la conception d'un système, en offrant une vue claire des entités et de leurs interactions. Dans ce chapitre, nous explorerons les concepts clés tels que les classes, les attributs, les méthodes et les relations, et leur rôle dans la modélisation de systèmes complexes.



- classe Channel**

Attributs :

- id: string : Identifiant unique du canal.
- participants: string[] : Liste des identifiants des utilisateurs participants au canal.
- admins: string[] : Liste des identifiants des administrateurs du canal.
- description: string : Description du canal.
- messages: string[] : Liste des identifiants des messages envoyés dans le canal.
- name: string : Nom du canal.
- image: string : URL ou chemin de l'image associée au canal.

• classe User

- Attributs :
 - id: string : Identifiant unique de l'utilisateur.
 - email: string : Adresse e-mail de l'utilisateur.
 - username: string : Nom d'utilisateur.
 - password: string : Mot de passe de l'utilisateur.
 - about: string : Description ou biographie de l'utilisateur.
 - image: string : URL ou chemin de l'image de profil de l'utilisateur.
 - friends: string[] : Liste des identifiants des amis de l'utilisateur.
 - blocked: string[] : Liste des identifiants des utilisateurs bloqués.
 - requests: string[] : Liste des identifiants des demandes d'amis en attente.
- Méthodes :
 - getProfile(id: string): \varnothing user: User \varnothing : Récupère le profil d'un utilisateur par son identifiant.
 - getFriendBySearch(search: string): \varnothing users: User[] \varnothing : Recherche des amis par un critère donné.
 - updateProfile(id: string, body: any): \varnothing statusCode: string, message: string \varnothing : Met à jour le profil de l'utilisateur.
 - getRequest(id: string): any : Récupère une demande d'ami.
 - setRequest(id: string, body: any): any : Ajoute ou modifie une demande d'ami.
 - setBlocked(id: string, body: any): any : Bloque un utilisateur.

• classe Message

- Attributs :
 - id: string : Identifiant unique du message.
 - channelId: string : Identifiant du canal auquel appartient le message.
 - userId: string : Identifiant de l'utilisateur qui a envoyé le message.
 - text: string : Contenu textuel du message.
 - images: string[] : Liste des URL ou chemins des images envoyées.
- Méthodes :
 - createMessage(text: string, images: string[]): Message : Crée un nouveau message

Diagramme d'Objet:

- Ce diagramme d'objet représente une instance d'un système de messagerie instantanée. Il montre deux utilisateurs (user1 et user2), un canal de discussion (channel1), et un message (message1). Les utilisateurs participent au canal, où user1 envoie un message. Le canal contient ce message, illustrant l'interaction entre les objets dans un chat en groupe.

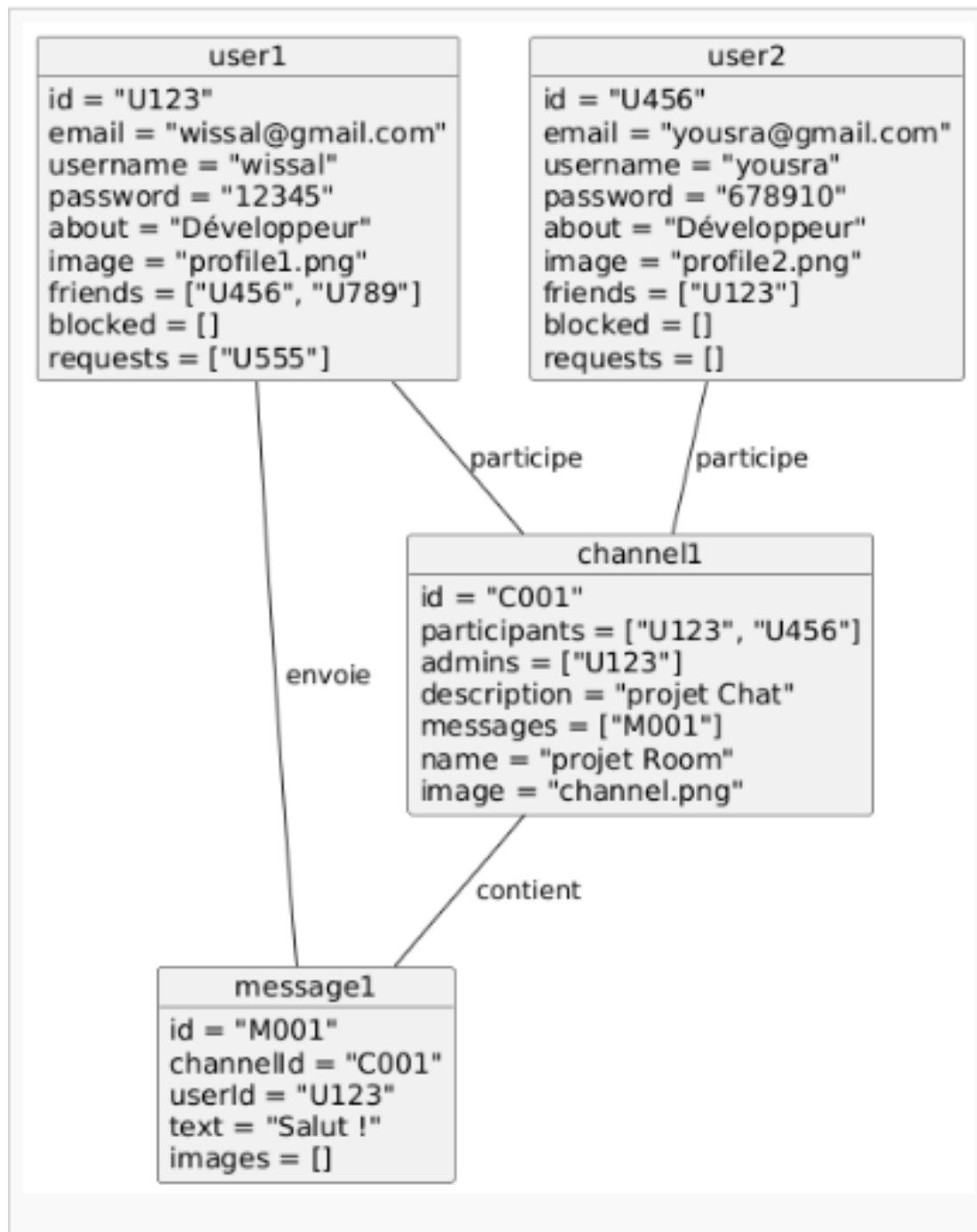


Diagramme de sequence:

1. Diagramme de Séquence : Inscription d'un Utilisateur

Ce diagramme illustre le processus d'inscription d'un nouvel utilisateur sur la plateforme. Il détaille les interactions entre l'utilisateur, l'interface utilisateur (UI), le serveur et la base de données afin de garantir la validité des informations fournies avant la création du compte.

Étapes du processus :

1. Saisie des informations d'inscription
 - L'utilisateur remplit un formulaire avec son email, son nom d'utilisateur (username) et son mot de passe via l'interface UI.
2. Envoi des données au serveur
 - L'interface utilisateur transmet les informations saisies au serveur pour validation.
3. Vérification des données
 - Le serveur effectue plusieurs contrôles pour assurer l'intégrité des données :
 - Vérification du format de l'email.
 - Vérification de la longueur minimale du nom d'utilisateur (≥ 3 caractères).
 - Vérification de l'unicité de l'email en interrogeant la base de données.
4. Gestion des résultats de la vérification
 - Si l'email est invalide, le username est trop court ou l'email est déjà utilisé, une erreur est envoyée à l'interface UI et affichée à l'utilisateur.
 - Si les données sont valides, le processus continue.
5. Création du compte
 - Le serveur crée le compte de l'utilisateur dans la base de données.
 - La base de données confirme l'enregistrement.
6. Retour d'information à l'utilisateur
 - Le serveur envoie une confirmation de succès à l'interface UI.
 - L'interface affiche le message "Compte créé avec succès" à l'utilisateur.

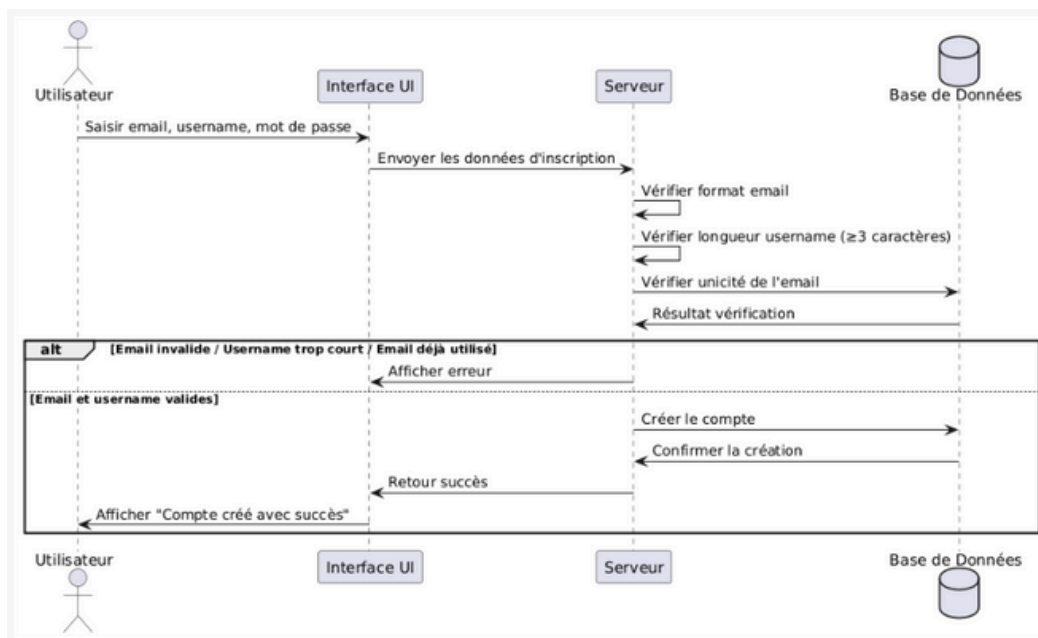


Diagramme de sequence:

2. Diagramme de Séquence : Ajouter un Contact

Ce diagramme illustre le processus par lequel un utilisateur ajoute un contact à sa liste de contacts sur la plateforme de chat en temps réel.

Étapes du processus :

- Saisie du contact : L'utilisateur entre le nom du contact qu'il souhaite ajouter via l'interface utilisateur (UI).
- Vérification de l'existence du contact :
- L'interface transmet la demande au serveur.
- Le serveur interroge la base de données pour vérifier si l'utilisateur recherché existe.

Gestion des résultats :

- Si l'utilisateur n'existe pas, le serveur envoie un message à l'interface pour informer l'utilisateur que le contact est introuvable.
- Si l'utilisateur existe, le serveur enregistre le contact dans la liste de l'utilisateur dans la base de données.
- Confirmation de l'ajout : Une fois l'ajout effectué, une confirmation est envoyée à l'interface et affichée à l'utilisateur.

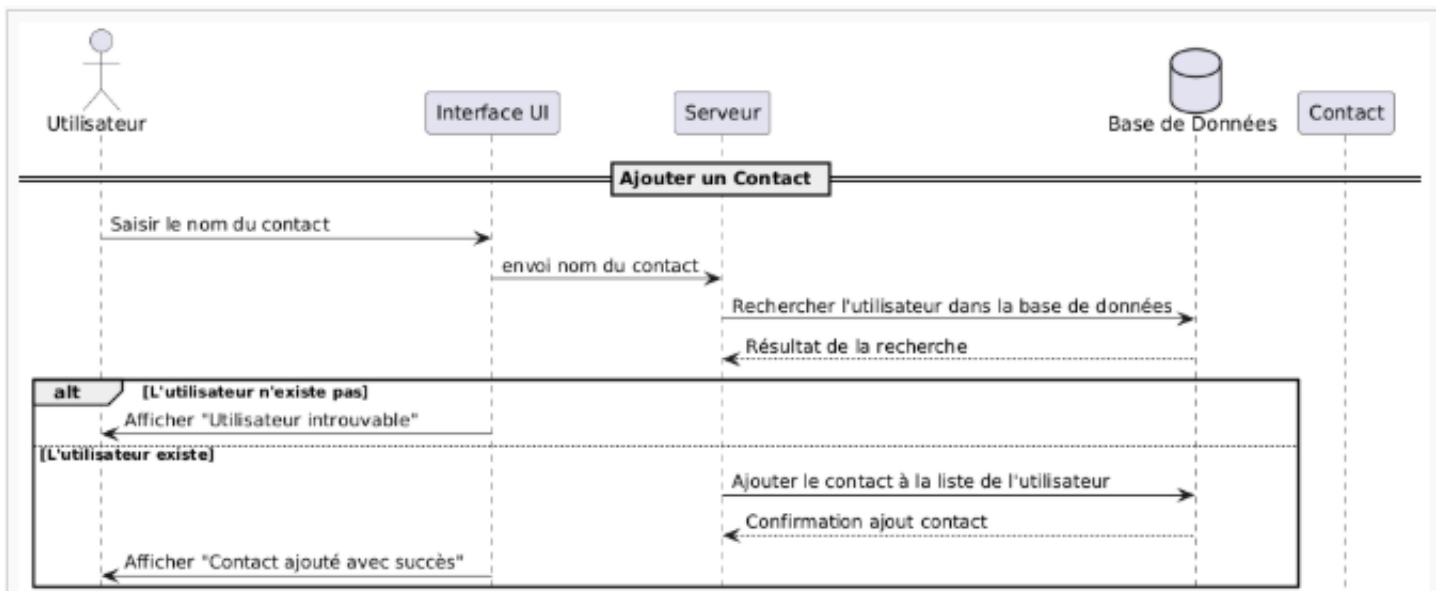


Diagramme de sequence:

3. Diagramme de Séquence : Envoyer un Message Instantané

Ce diagramme représente l'envoi d'un message instantané d'un utilisateur à un contact sélectionné.

Étapes du processus :

1. Saisie et sélection du destinataire : L'utilisateur rédige un message et choisit un contact via l'interface.
2. Transmission du message :
 - L'interface envoie le message au serveur.
 - Le serveur stocke le message dans la base de données.
 - Une confirmation d'enregistrement est renvoyée au serveur.
3. Notification du destinataire (en parallèle) :
 - Une notification en temps réel est envoyée au destinataire pour l'informer de la réception du message.
4. Affichage de confirmation : Une fois le message envoyé avec succès, une notification est affichée à l'expéditeur.

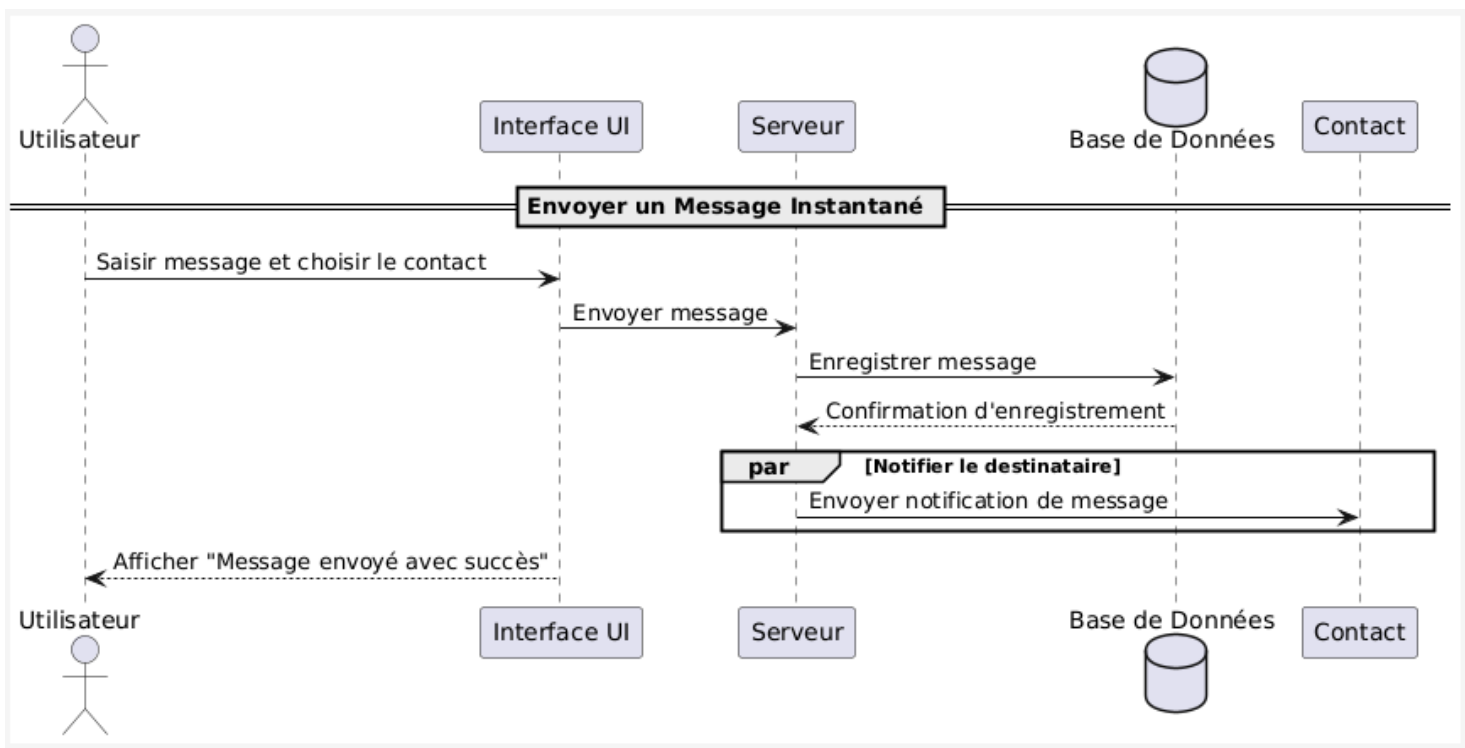


Diagramme d'Activite:

- Le diagramme d'activité décrit un processus de création de compte et d'interaction sociale dans une application. Il commence par l'inscription d'un utilisateur, suivi de la saisie des informations et d'une vérification de l'existence de l'utilisateur. Si l'utilisateur existe déjà, un message d'erreur est affiché, sinon, il est redirigé vers la page de connexion.
- Ensuite, l'utilisateur saisit ses informations d'identification. Si elles sont invalides, un message d'erreur est affiché. Une fois l'authentification réussie, l'utilisateur peut rechercher des amis par leur nom et leur envoyer une invitation. Si l'invitation est refusée, le processus se termine. Si elle est acceptée, l'utilisateur peut commencer une conversation, envoyer des messages et passer un appel.

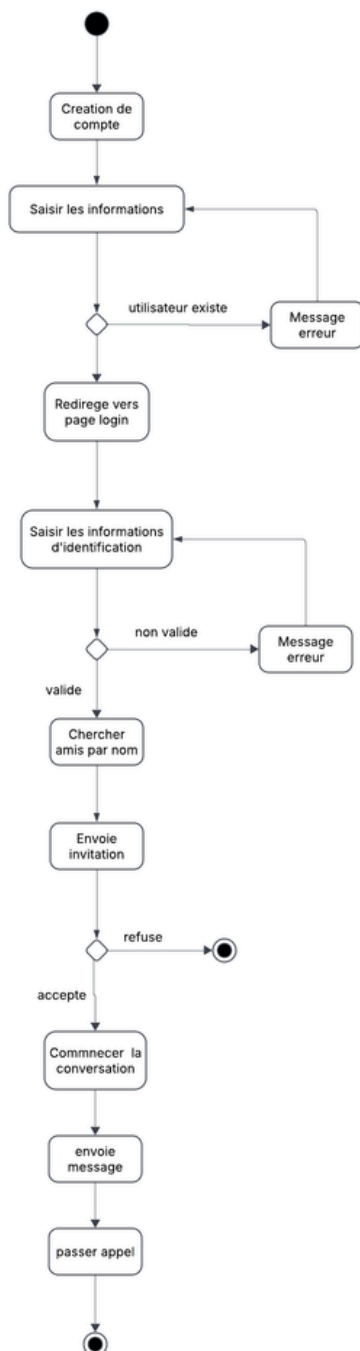


Diagramme de package:

Diagramme de package décrit les relations entre différents modules ou bien systèmes et les sous-systèmes d'un programme.

Comme conçu dans le diagramme au-dessous, présente le système de l'application. Ce forme des packages avec des flèches qui représentent les relations entre les packages.

Par exemple:

- Le package **"User"** représente le système *utilisateurs* qui est accessible à les deux packages **"user details"** et **"friends list"** qui sont les sous-systèmes du système *utilisateur*.
- Le package **"Channel"** est le système parent des sous-systèmes **"Admin"** et **"Participant"** qui sont dépendants du package **"User"** et importent le package **"Message"**, etc.

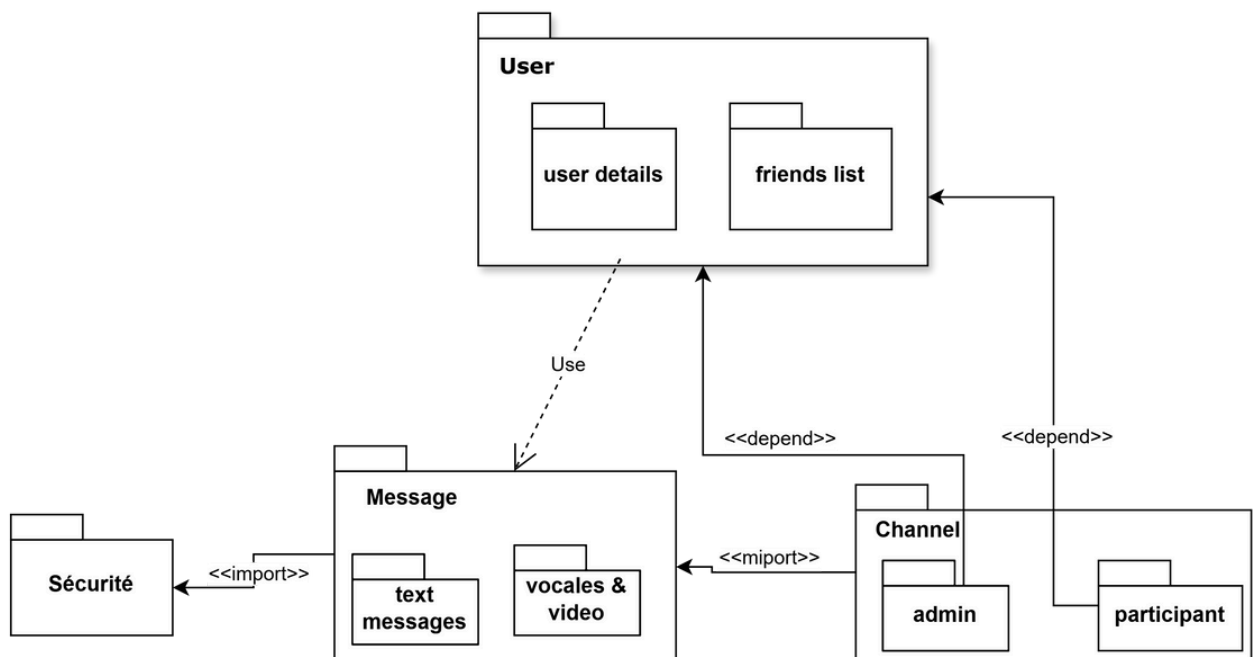
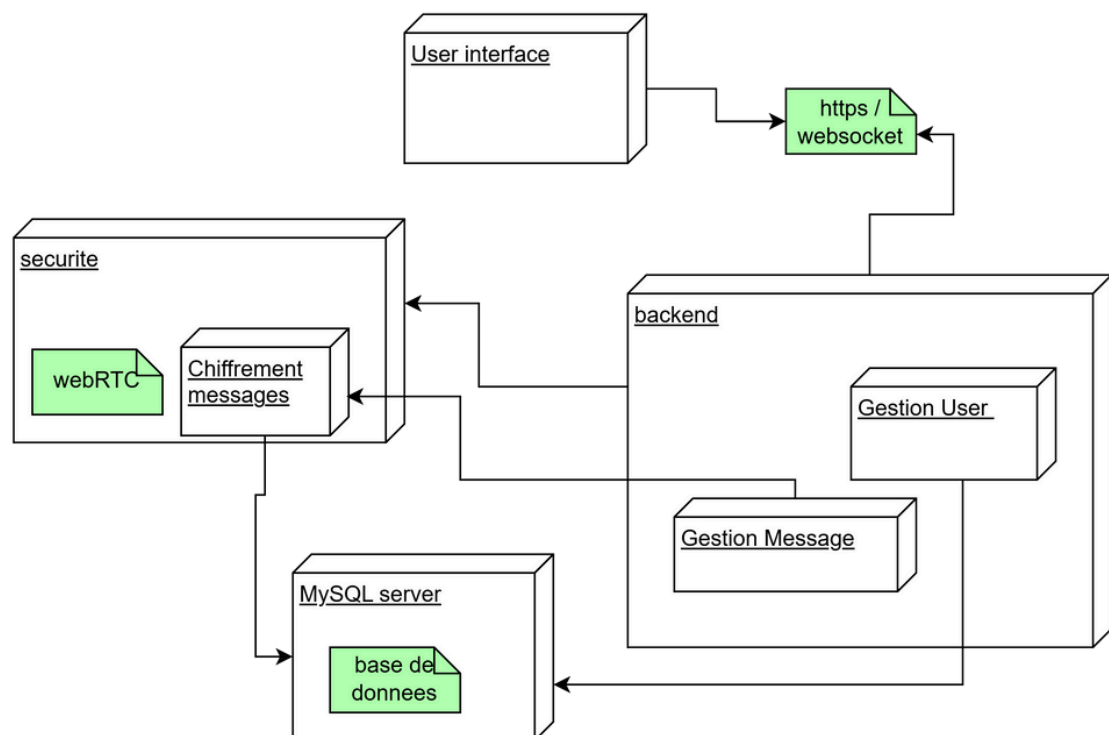


Diagramme de déploiement :

Le diagramme de déploiement illustre les interactions entre les composants matériels et logiciels répartis sur serveurs et clients. Il détaille le processus matériel et logiciel de l'application de messagerie. L'interface client permet à l'utilisateur d'envoyer des messages, de lancer des appels vocaux et vidéo, et se connecte au serveur backend via HTTPS et WebSocket pour la messagerie en temps réel. Le serveur backend gère les informations des utilisateurs et les demandes d'amis, stockées dans une base de données MySQL Server après vérification par une couche de sécurité qui sécurise les messages et les appels via WebRTC.



Chapitre 4

Réalisation

4.1 Introduction

Après avoir finalisé l'étape de conception, ce chapitre est consacré à la phase de réalisation de l'application. Nous y décrirons l'environnement de développement mis en place ainsi que les différentes étapes techniques suivies pour aboutir à une première version fonctionnelle de la plateforme.

4.2 Environnement logiciel

Pour mener à bien le développement de l'application, nous avons mis en place un environnement technique complet, regroupant les outils, langages et technologies nécessaires. Cet environnement est structuré comme suit :



VS CODE

- VisualStudio Code est un éditeur de code extensible développé par Microsoft pour Windows, Linux et macOS. Les fonctionnalités incluent la prise en charge du débogage, la mise en évidence de la syntaxe, la complétion intelligente du code, les snippets, la refactorisation du code et Git intégré



Postman

- Un outil de collaboration et de test pour les développeurs, permettant de créer, tester et documenter des API RESTful, mais aussi de simuler et tester des connexions WebSocket pour vérifier la communication en temps réel entre le client et le serveur.

4.3 Environnement technique



MySQL

- Un système de gestion de bases de données relationnelles (RDBMS) open-source, populaire pour son efficacité, sa fiabilité et sa facilité d'utilisation, souvent utilisé pour les applications web.



WebSocket

- WebSocket est un protocole de communication réseau qui permet l'établissement d'une connexion bidirectionnelle persistante entre le client et le serveur. Contrairement au protocole HTTP traditionnel, qui est basé sur le modèle requête-réponse, WebSocket permet au serveur d'envoyer des données au client à tout moment, sans qu'une requête préalable soit nécessaire.



WebRTC

- Technologie open-source permettant des communications audio, vidéo et de partage de données directement entre navigateurs, sans passer par un serveur intermédiaire pour le transport des médias.



NestJs

- Framework backend basé sur Node.js et écrit en TypeScript. Il est conçu pour construire des applications côté serveur robustes, modulaires et bien structurées. Grâce à son architecture inspirée d'Angular, NestJS facilite l'organisation du code et l'intégration de fonctionnalités avancées comme les WebSockets, les API REST ou encore les bases de données.



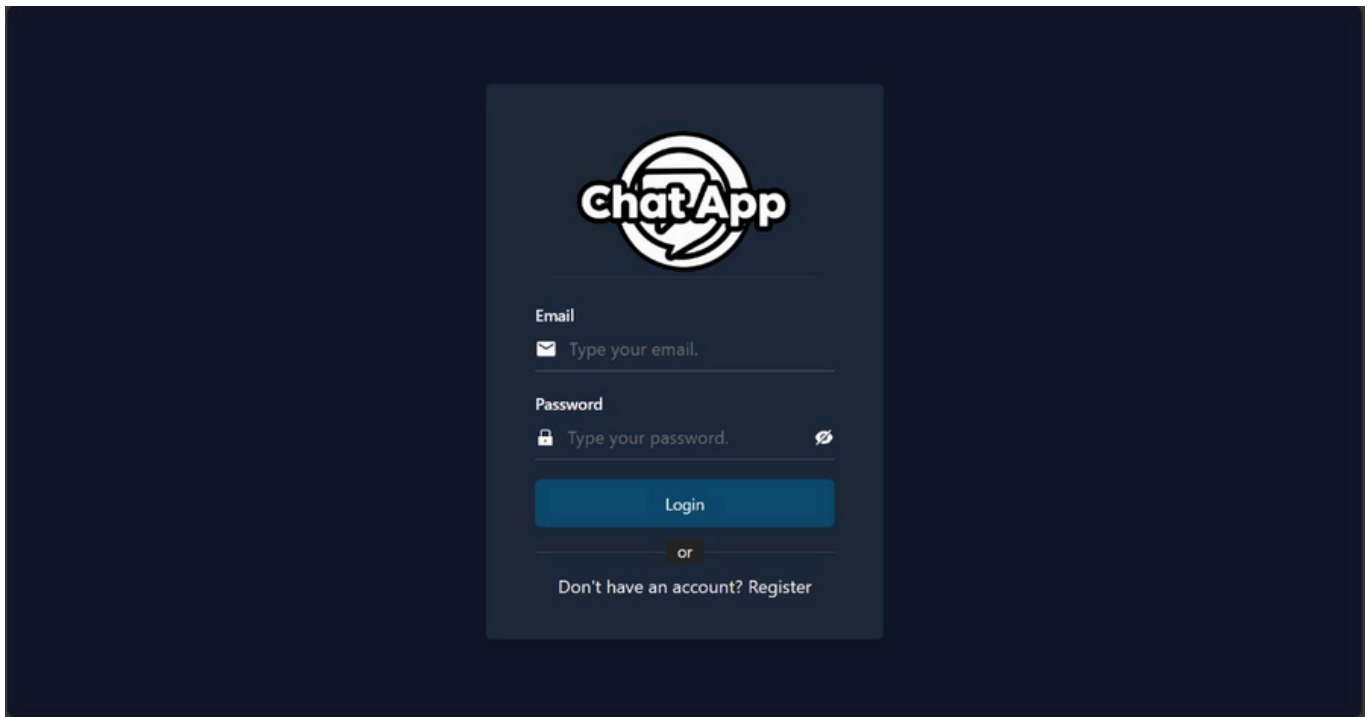
ReactJs

- Bibliothèque JavaScript développée par Meta (anciennement Facebook) qui permet de créer des interfaces utilisateur interactives et dynamiques. Elle est basée sur un concept de composants réutilisables, ce qui facilite le développement et la maintenance des applications web modernes côté client.

4.4 Travail réalisé

Cette partie est consacrée à présenter quelques interfaces de l'application réalisée.

1. Page de connexion (Login Page) :

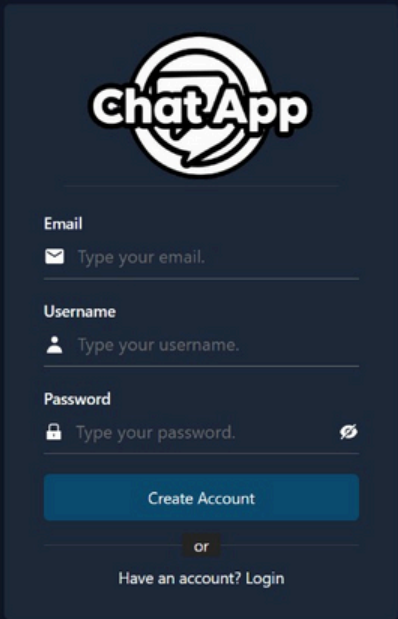


Cette interface permet à un utilisateur déjà inscrit d'accéder à la plateforme en saisissant ses informations d'identification. Elle est composée de :

- **Un champ Email** : pour entrer l'adresse e-mail de l'utilisateur.
- **Un champ Mot de passe** : pour saisir le mot de passe associé au compte.
- **Un bouton "Login"** : pour valider les informations et accéder à l'application.
- **Un lien vers la page d'inscription** : pour permettre à un nouvel utilisateur de créer un compte.

L'interface est sobre, moderne et facile à utiliser, avec des icônes intuitives pour les champs et une bonne lisibilité, ce qui améliore l'expérience utilisateur.

2. Page d'inscription (Registration Page) :

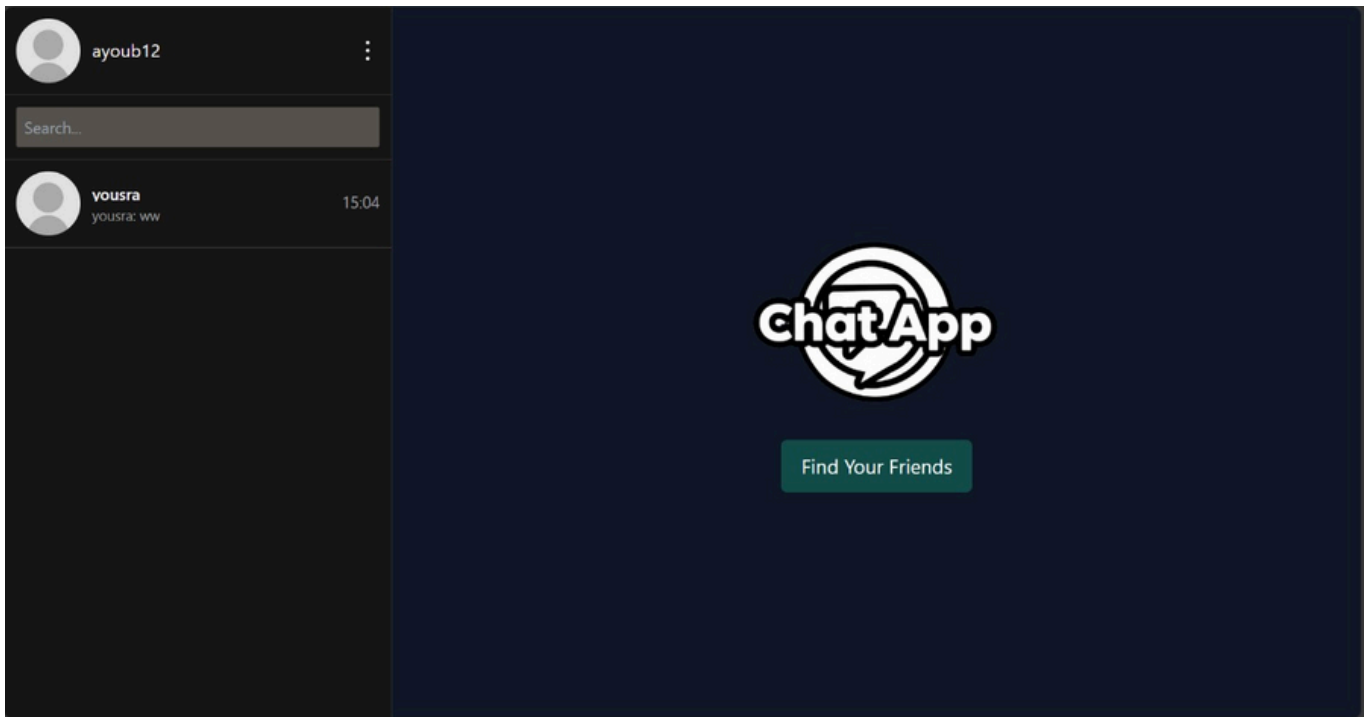
The image shows a registration form for 'Chat App' on a dark blue background. The form is centered and contains the following elements: a logo at the top with the text 'Chat App' inside a speech bubble; three input fields labeled 'Email', 'Username', and 'Password' with placeholder text 'Type your email.', 'Type your username.', and 'Type your password.' respectively; a blue 'Create Account' button; a small 'or' separator; and a link 'Have an account? Login'.

Cette interface permet aux nouveaux utilisateurs de créer un compte sur la plateforme. Elle comprend :

- **Un champ Email** : pour renseigner l'adresse e-mail de l'utilisateur.
- **Un champ Nom d'utilisateur (Username)** : pour choisir un pseudonyme visible dans les conversations.
- **Un champ Mot de passe** : pour définir un mot de passe sécurisé.
- **Un bouton "Create Account"** : pour valider la création du compte.
- **Un lien vers la page de connexion** : "Have an account? Login" pour revenir à l'écran de connexion si l'utilisateur possède déjà un compte.

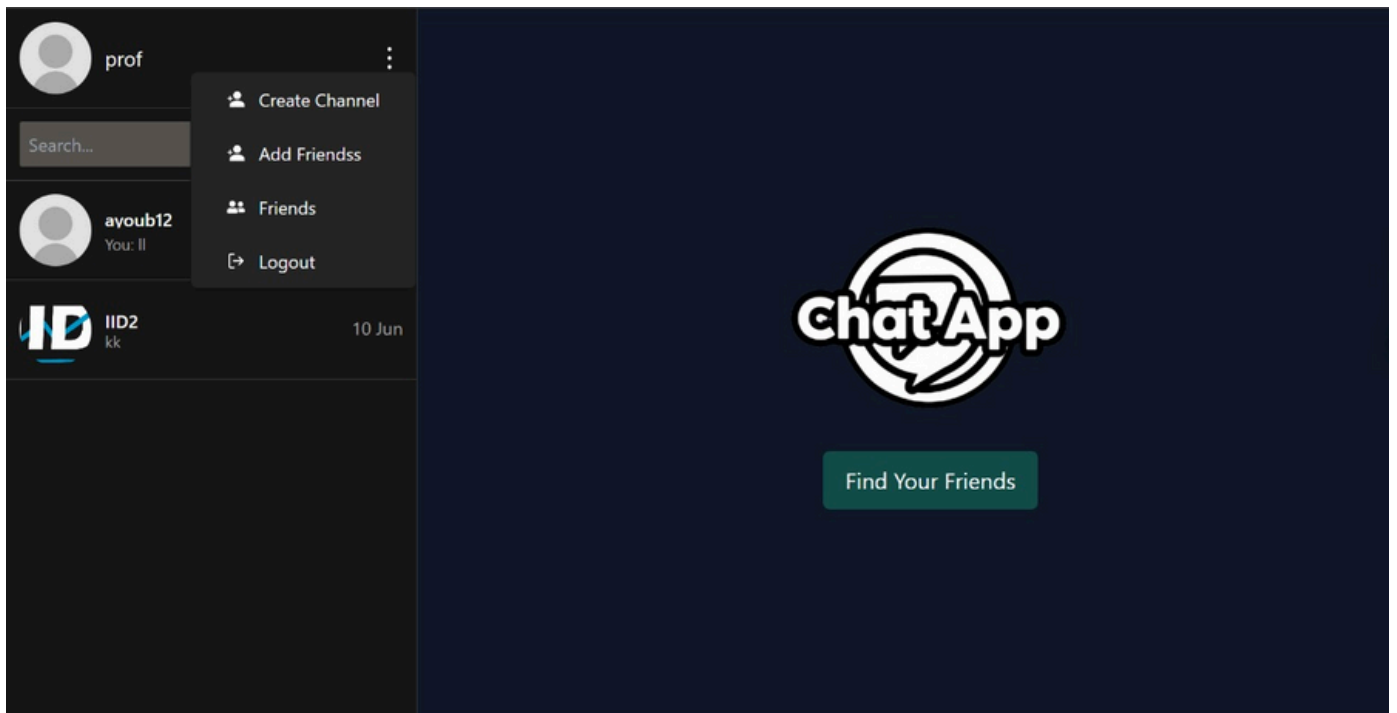
L'interface conserve le même style visuel que la page de connexion, assurant une cohérence graphique et une navigation fluide entre les deux écrans.

3. Page d'accueil (Home Page) :



Cette interface sert de page d'accueil pour "chatApp". Elle comprend :

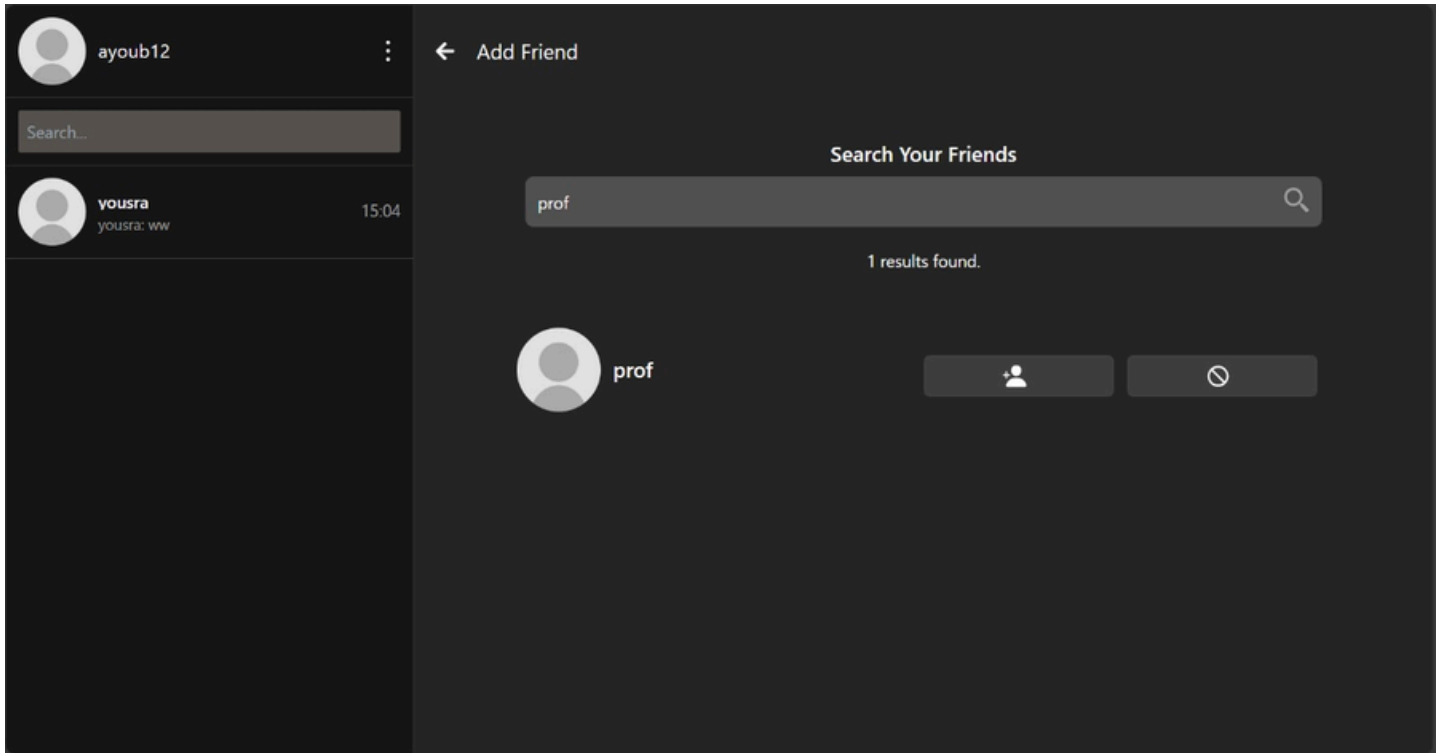
- Une liste de contacts : affichant des avatars et des noms d'utilisateurs .
- Une barre de recherche : pour trouver des utilisateurs ou conversations.
- Un bouton "Find Your Friends" : pour découvrir de nouveaux contacts.
- L'interface utilise un fond sombre, assurant une cohérence visuelle et une navigation intuitive.



Ce menu contextuel s'affiche lorsqu'on clique sur les trois boutons. Il comprend :

- Une option "**Create Channel**" : pour créer un nouveau canal de discussion.
- Une option "**Add Friends**" : pour ajouter de nouveaux contacts.
- Une option "**Friends**" : pour afficher la liste des amis.
- Une option "**Logout**" : pour se déconnecter de la plateforme.

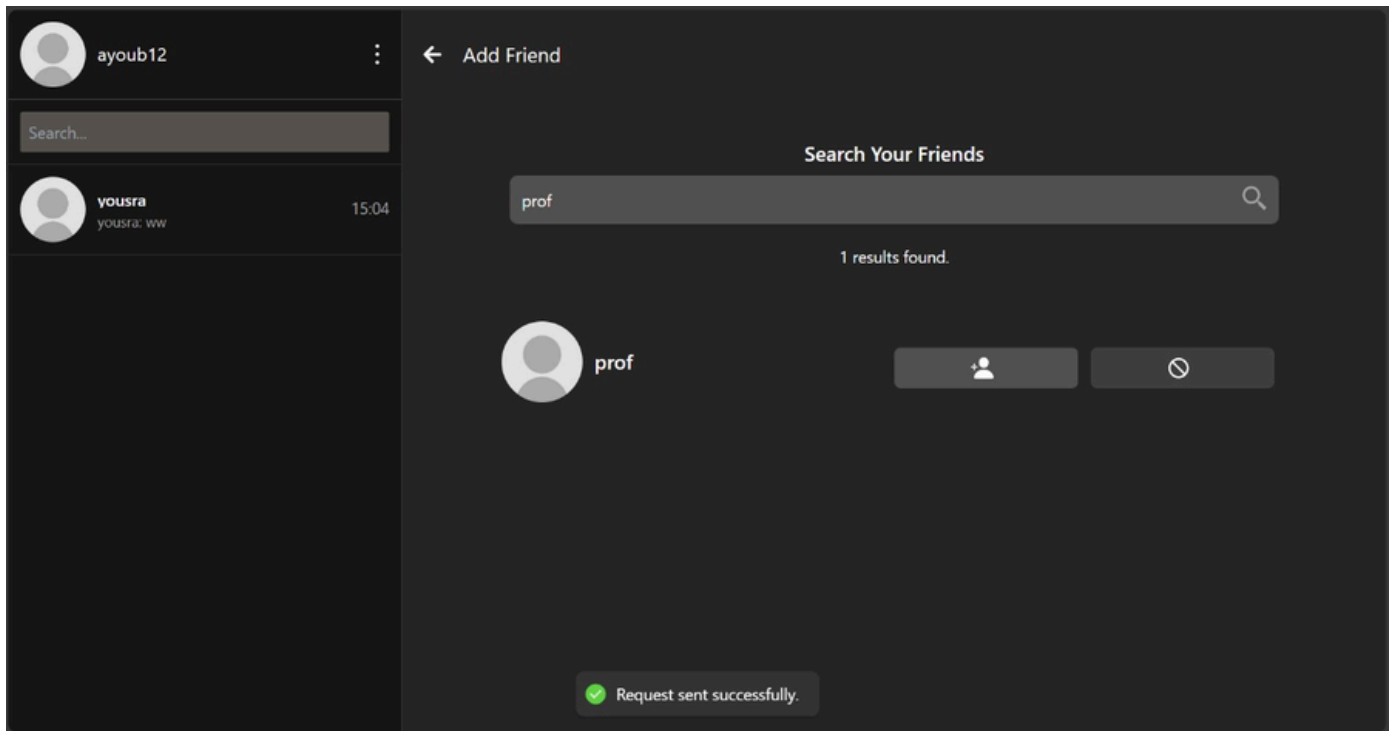
4.Interface d'ajout d'ami – Application de messagerie



L'image montre une interface utilisateur sombre où l'utilisateur connecté (nom d'utilisateur : ayoub12) a la possibilité de rechercher et d'ajouter de nouveaux amis. Dans la section de recherche, l'utilisateur a saisi le mot-clé "prof", ce qui a affiché un résultat correspondant à un utilisateur nommé "prof". Deux options sont disponibles pour cet utilisateur :

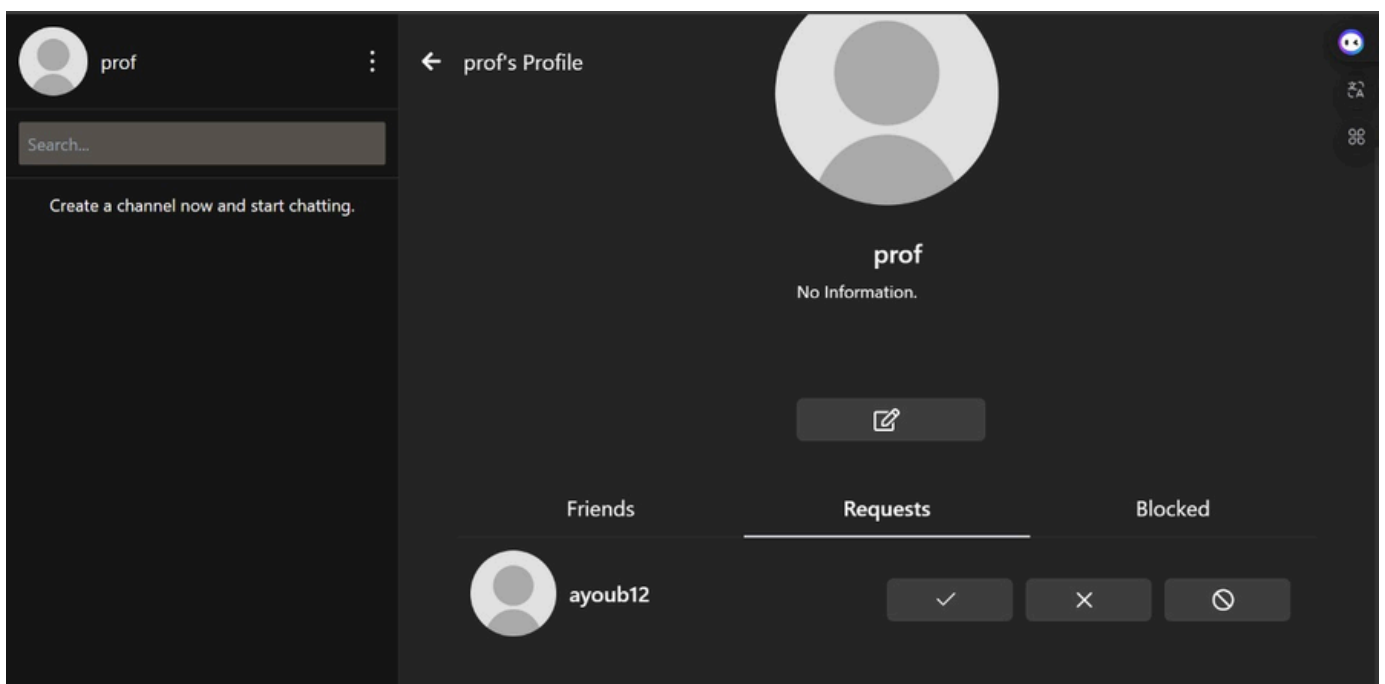
- Un bouton avec une icône d'ajout pour envoyer une demande d'ami.
- Un bouton avec une icône de blocage pour ignorer ou bloquer cet utilisateur.

5. Ajout bien effectué



Une notification "Request sent successfully" confirme l'envoi d'une demande.

6. Interface de gestion de demandes d'amis – Profil utilisateur



Cette interface permet à l'utilisateur de gérer les demandes d'ajout d'amis dans une application de messagerie. Elle offre plusieurs fonctionnalités essentielles :

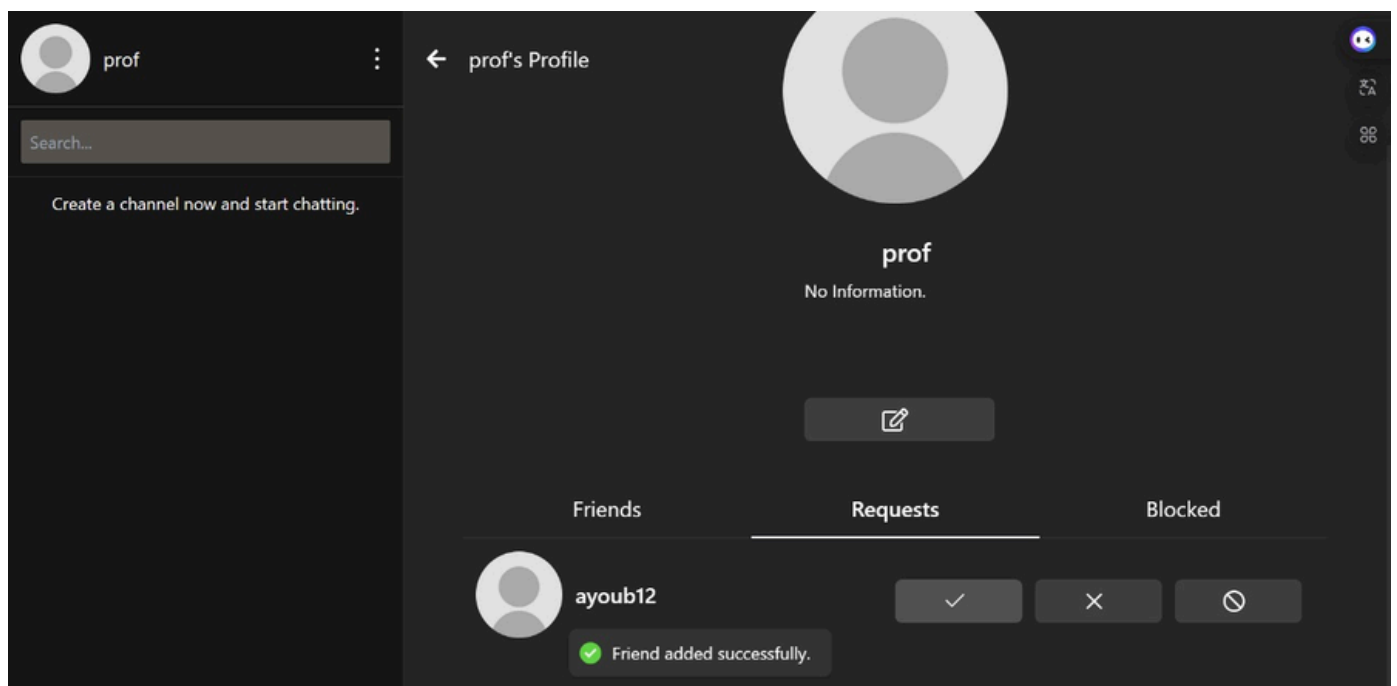
- Afficher les demandes reçues : l'utilisateur peut voir qui lui a envoyé une demande d'ami.
- Accepter une demande : ajoute la personne à la liste des amis.
- Refuser une demande : ignore la demande sans ajouter la personne.
- Bloquer un utilisateur : empêche toute interaction future avec cet utilisateur.

L'interface est également divisée en trois onglets :

- Amis : liste des contacts acceptés.
- Demandes : liste des utilisateurs en attente de réponse.
- Bloqués : liste des utilisateurs interdits.

Objectif : Offrir un espace de gestion sécurisé et intuitif pour contrôler les relations sociales dans l'application.

7.Utilité de l'interface – Confirmation d'ajout d'un ami

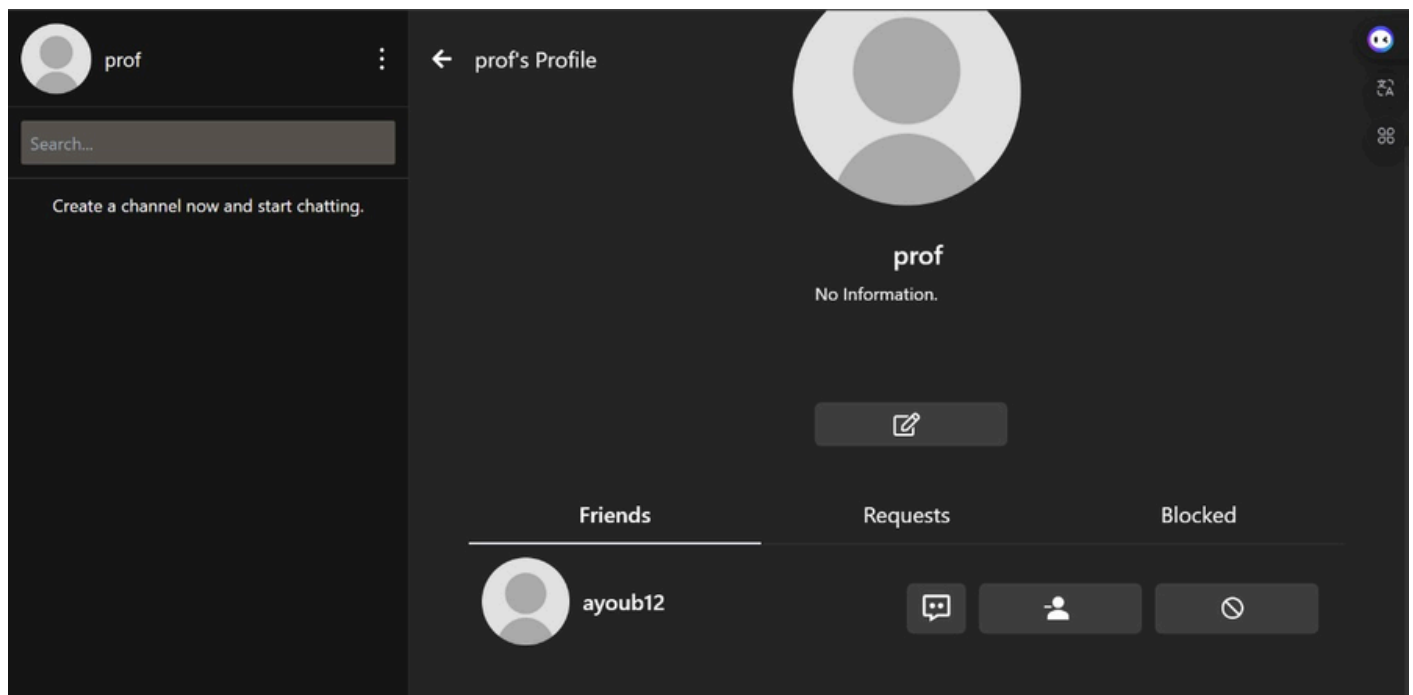


Cette interface confirme que l'utilisateur a accepté une demande d'ami dans l'application. Elle est essentielle pour la gestion des contacts entre utilisateurs.

- Le message "Friend added successfully." indique que l'utilisateur ayoub12 a été ajouté avec succès à la liste des amis.
- Cela permet désormais aux deux utilisateurs de communiquer entre eux via les fonctionnalités de l'application (messagerie, appels, etc.).
- L'écran reste sur l'onglet "Requests", mais met à jour l'état de la demande après validation.
- Les options de refus ou de blocage sont également disponibles avant validation.

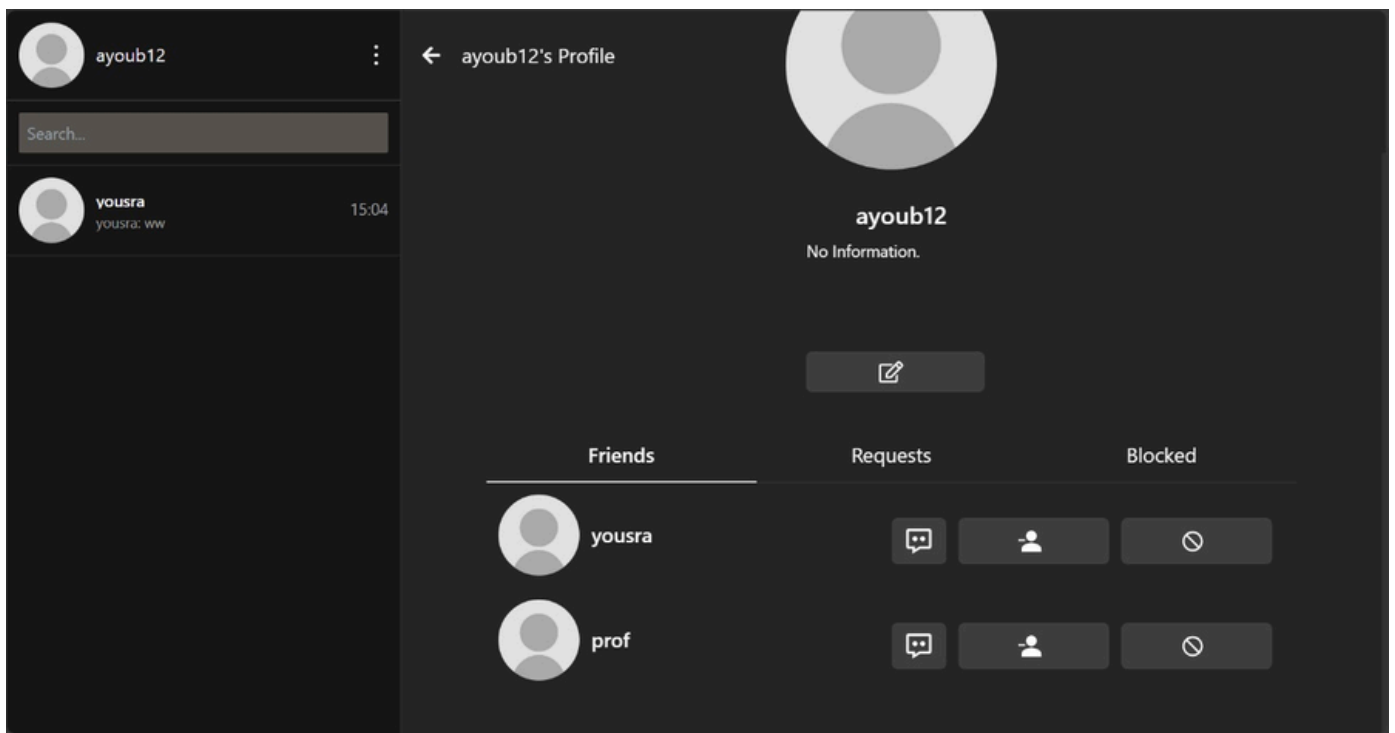
Objectif : offrir un retour clair à l'utilisateur sur l'action effectuée et assurer une gestion fluide des relations entre membres.

8.Profil Utilisateur - Interface de Messagerie



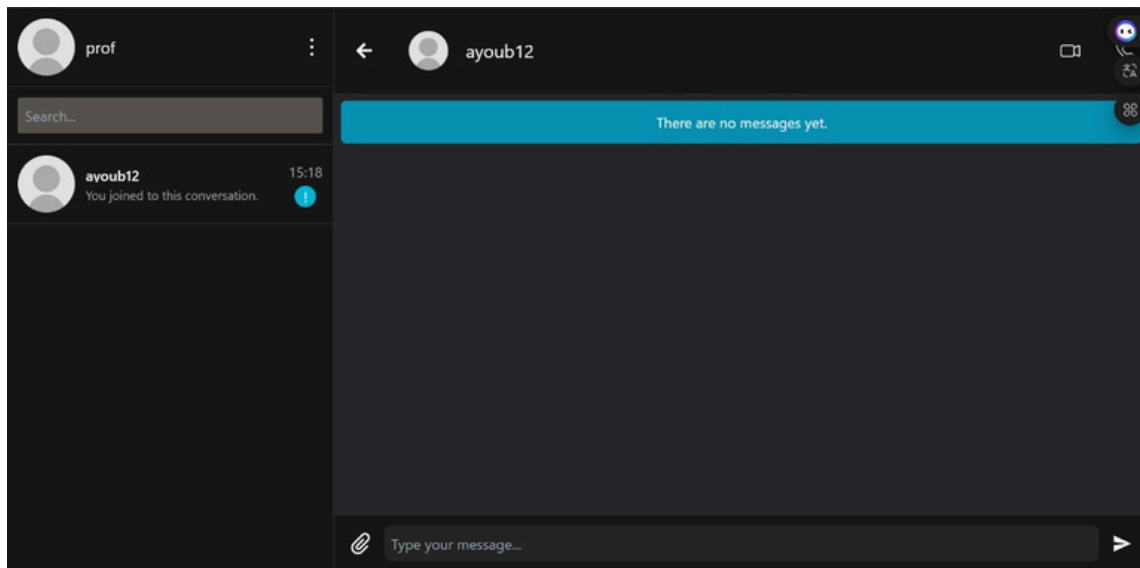
Une fois l'invitation d'amitié acceptée, l'étudiant est automatiquement ajouté à la liste des amis du prof . Cette interface offre une vue claire de l'état de la relation et permet une gestion simple des contacts.

9.Profil Utilisateur - Interface de Messagerie




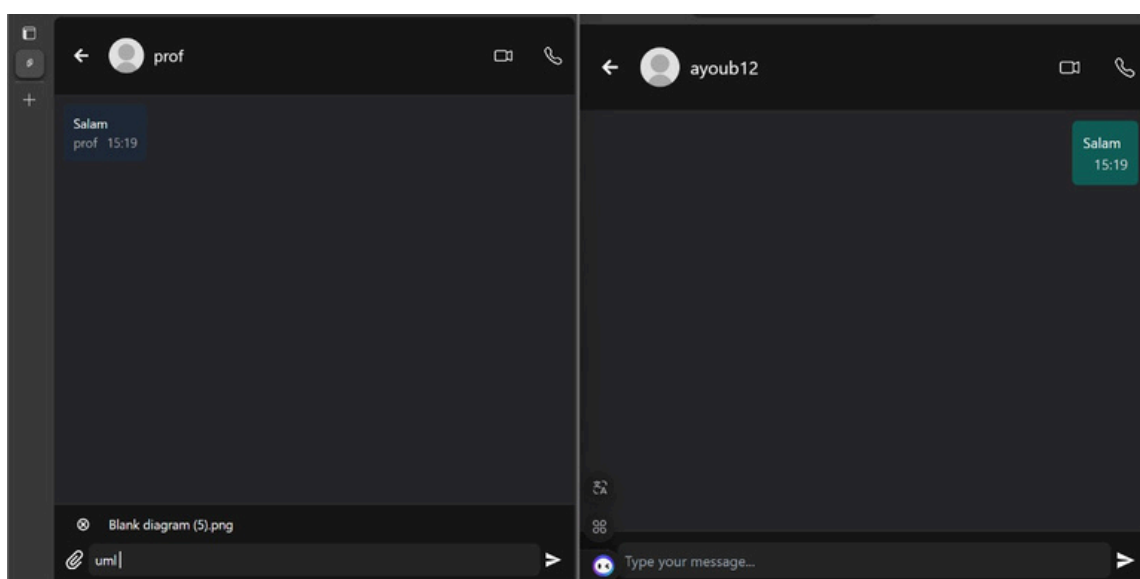
Une fois l'invitation d'amitié acceptée, le professeur est automatiquement ajouté à la liste d'un étudiant . Cette interface offre une vue claire de l'état de la relation et permet une gestion simple des contacts.

10. Page de conversation - Page de messagerie :

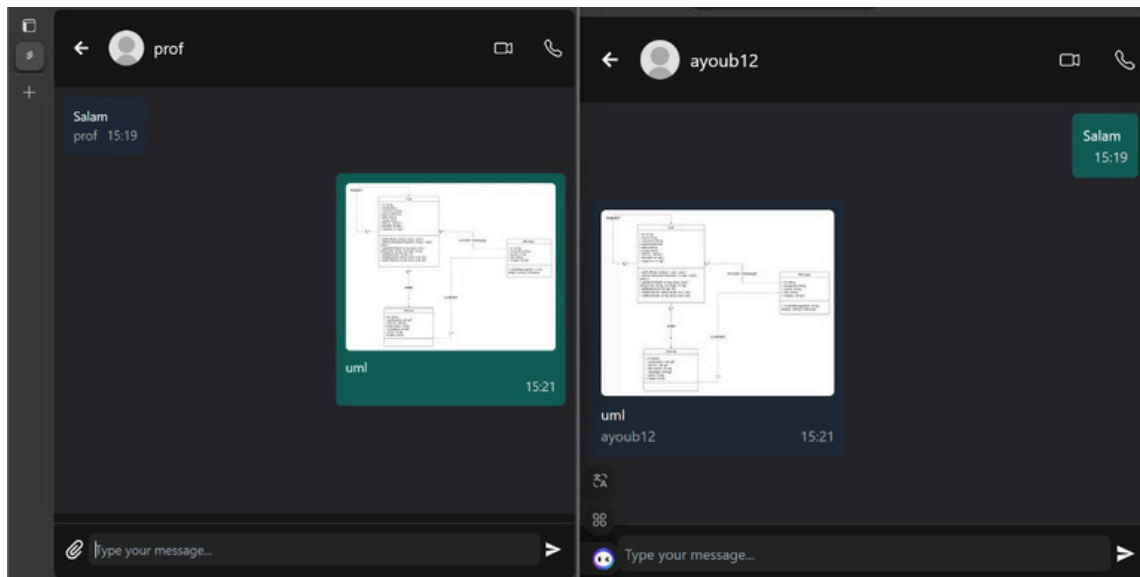


Cette interface permet aux utilisateurs de concevoir les messages entre eux et d'envoyer de nouveaux. Elle comprend :

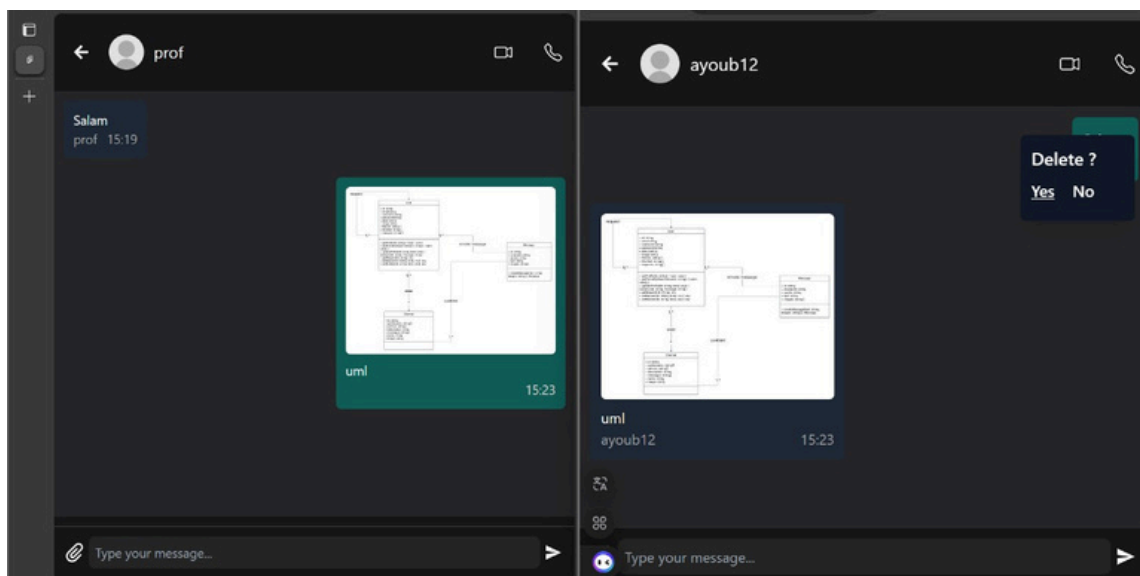
- **Un champ pour écrire le message** : pour saisir le message et l'envoyer en cliquant sur le bouton envoyer
- **Le bouton  attacher document** : pour attacher des documents, fichiers ou bien image aux messages.
- **les boutons appel vocal et vidéo** : pour effectuer des appels vocaux ou vidéo.



Comme le montre la figure, les messages envoyés sont verts et les messages reçus sont bleus.



Ce figure illustre l'envoi des images en utilisant le button attache, et comme les messages, les images envoyés sont verts et les images reçus sont bleus.



Pour supprimer un message, l'application offre aux utilisateurs l'option de le supprimer partout via un clic droit, qui affiche un menu proposant de supprimer ou non.

Après la suppression, le message sera remplacé par " the message has been deleted " sur les pages des deux utilisateurs.

11 Page de création groupe

ayoub12

Search...

prof
You: uml
15:21

yousra
You: cc
15:20

← Create Channel

Name
Channel Name

Description
Channel Description

Friends

Search a friend...

yousra +

prof +

Participants (1)

ayoub12 - ☆

Create Channel

ayoub12

Search...

prof
You: uml
15:21

yousra
You: cc
15:20

← Create Channel

IID

Name
IID2

Description
IID2 ENSAKH

Friends

Search a friend...

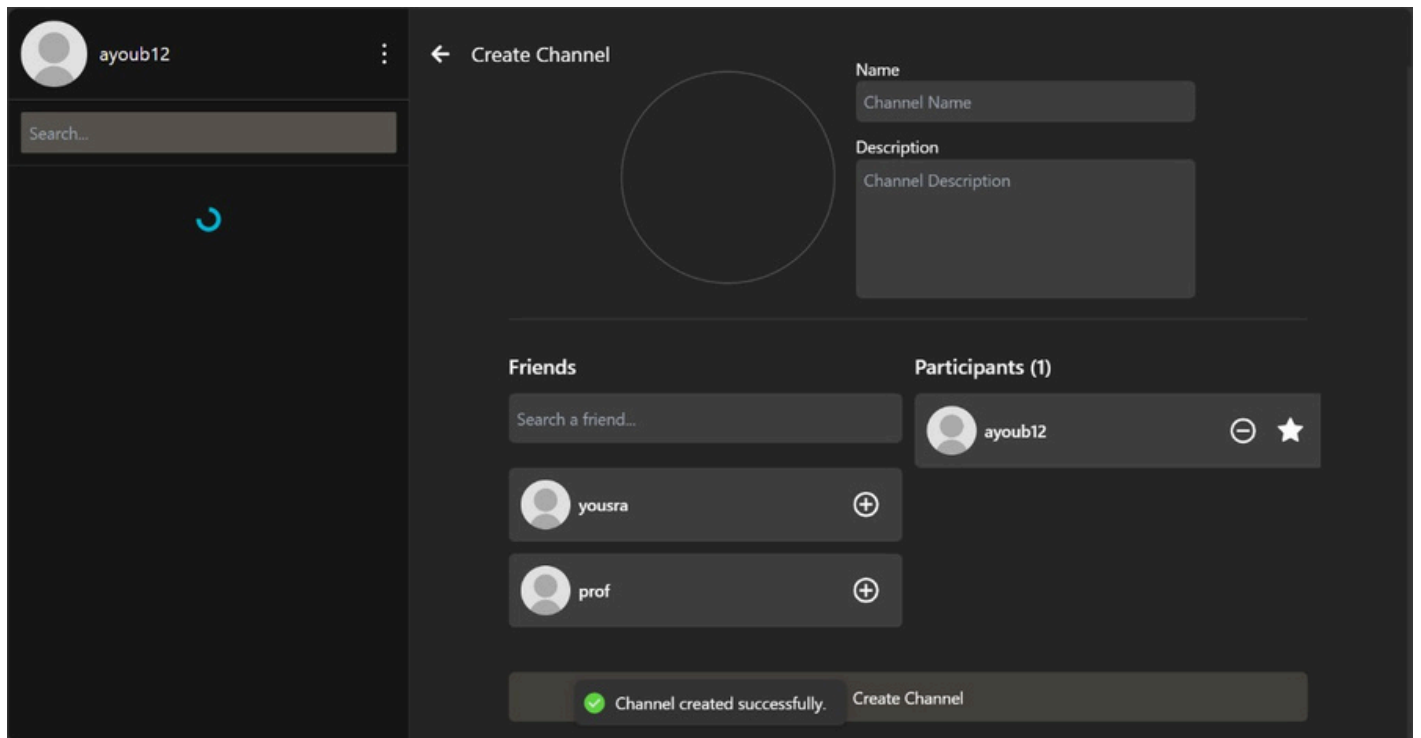
Participants (3)

ayoub12 - ☆

yousra - ☆

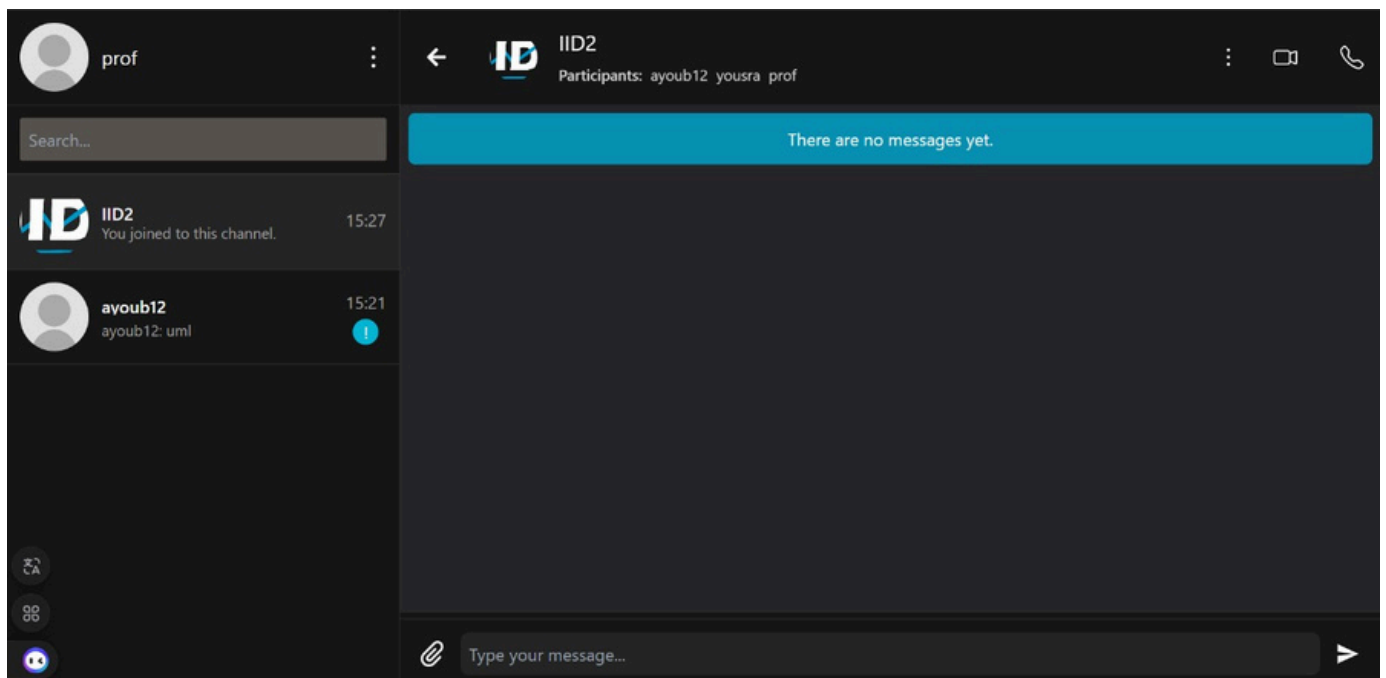
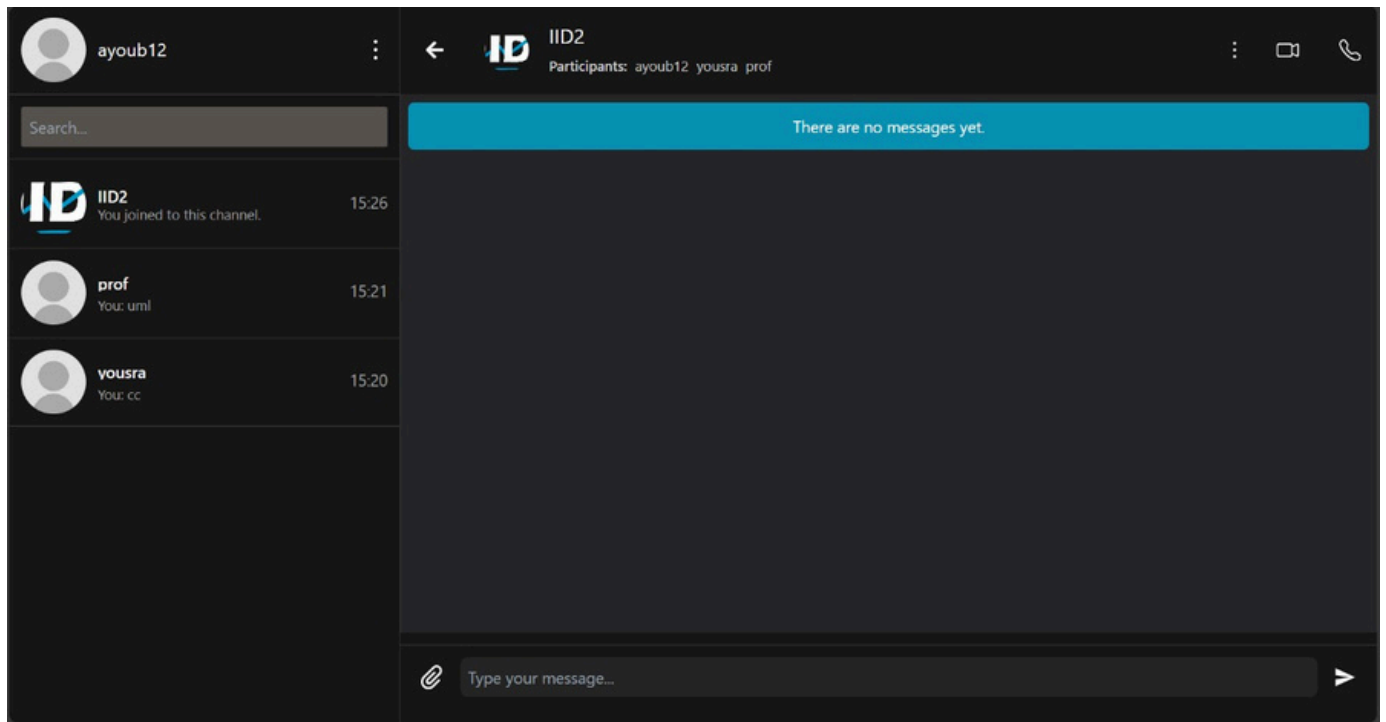
prof - ☆

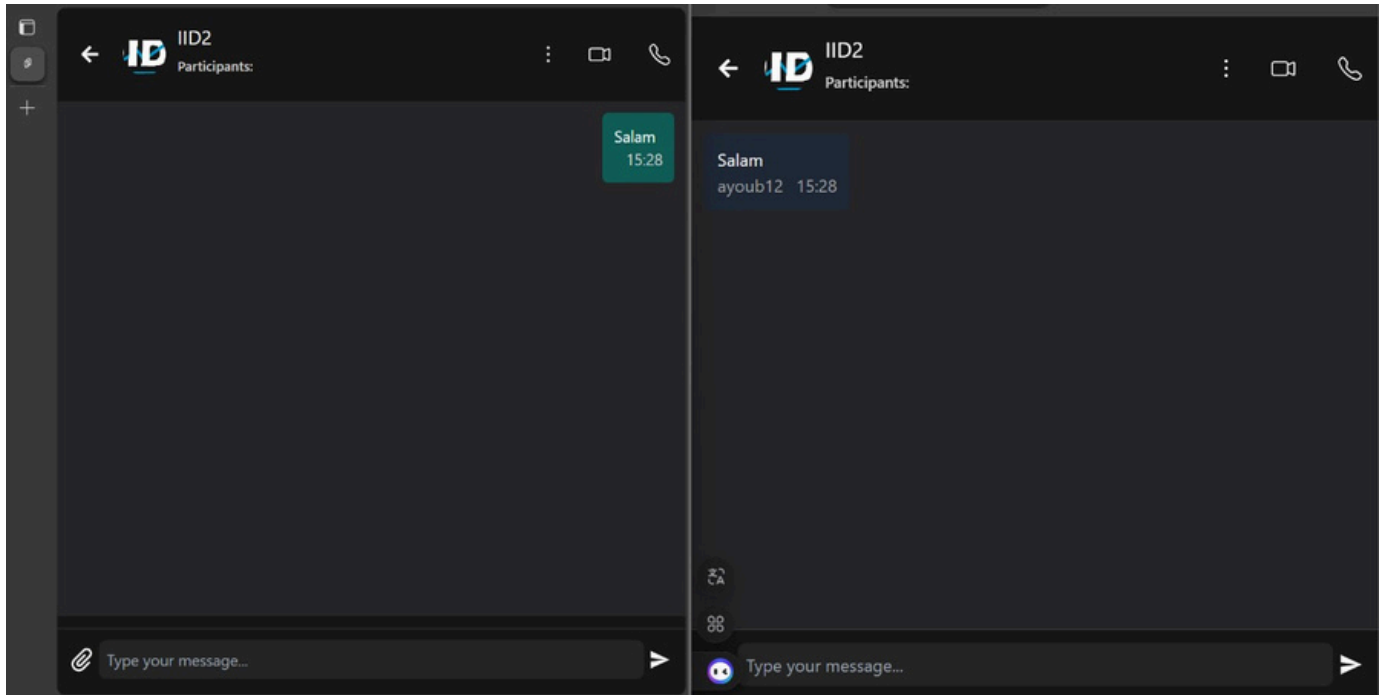
Create Channel



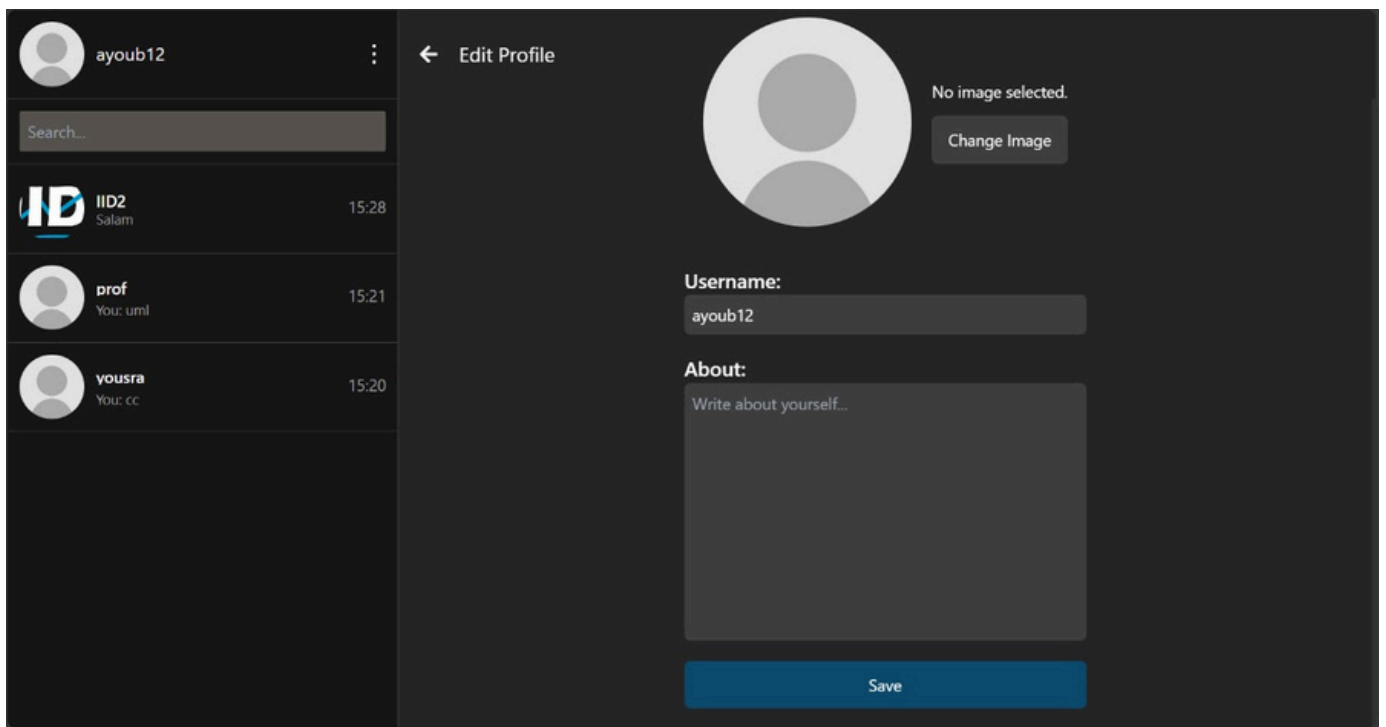
- Les images illustrent le processus de création d'un canal au sein d'une plateforme de communication collaborative. Dans un premier temps, l'interface propose un formulaire de configuration, permettant à l'utilisateur de définir le nom et la description du canal. La zone latérale affiche une liste d'amis disponibles, suggérant la possibilité d'inviter d'autres membres à rejoindre le canal. Une notification s'affiche pour confirmer la création réussie du canal.
- Dans l'étape suivante, le canal apparaît avec un nom et une description personnalisés, et plusieurs participants y sont désormais intégrés. L'interface présente également des fonctionnalités supplémentaires telles que la gestion des membres, la recherche d'amis, et un espace dédié à l'ajout d'un logo. L'ensemble reflète un système structuré, pensé pour faciliter la collaboration, la communication en équipe, et la coordination de projets dans un contexte académique ou professionnel.

12 Page de conversation groupe





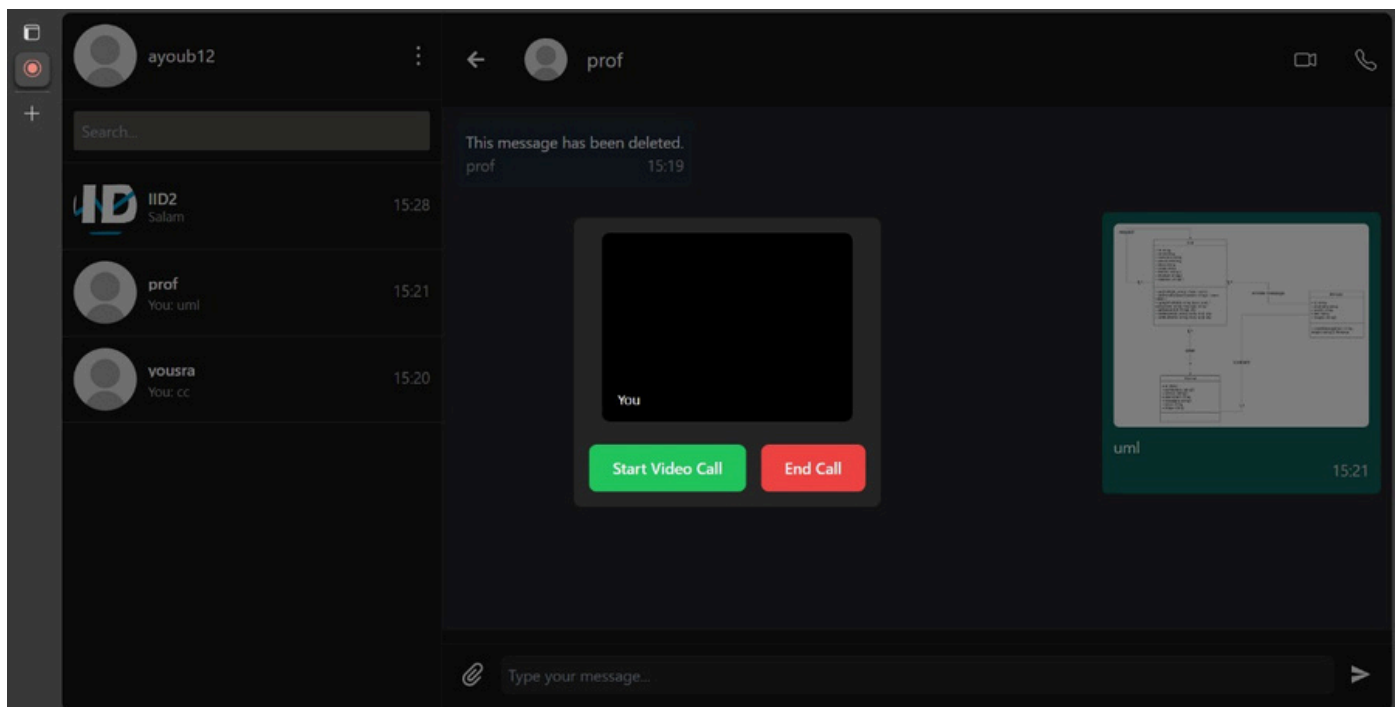
13 Page de modification profile

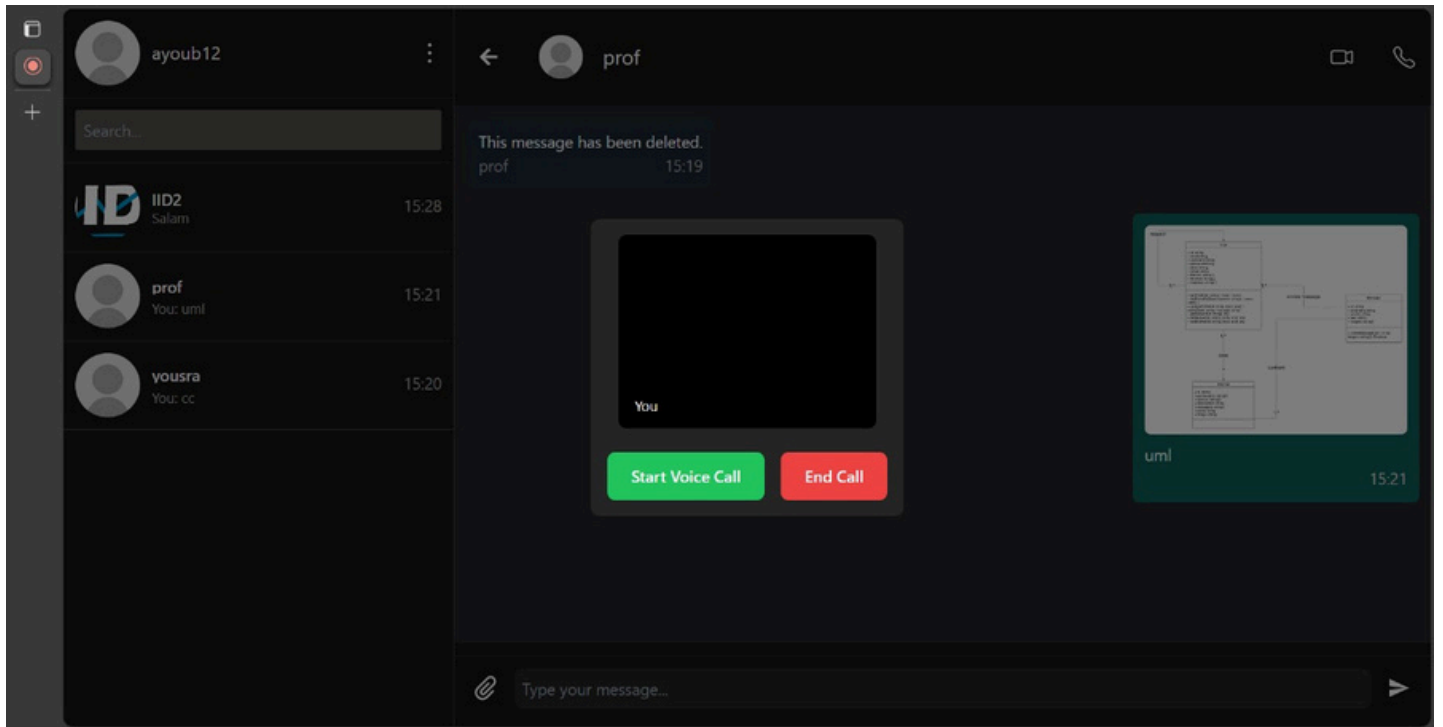


- Comprend un espace pour ajouter ou modifier une photo de profil à l'aide d'un bouton « Change Image ». Juste en dessous, deux champs sont proposés : l'un pour saisir ou modifier le nom d'utilisateur, et un autre, intitulé « About », permettant d'écrire une brève description personnelle. Un bouton « Save » est placé en bas pour enregistrer les changements effectués. L'interface est sobre, intuitive et centrée sur l'essentiel pour une personnalisation rapide du profil.

14 Module d'appels vocaux et vidéo

- L'application propose deux interfaces distinctes pour les appels : l'une dédiée aux appels vidéo, l'autre aux appels vocaux. Dans les deux cas, lorsqu'un appel est initié ou reçu, une fenêtre modale s'affiche à l'écran. Cette modale permet à l'utilisateur de démarrer ou de quitter l'appel en un clic.
- Pour les appels vidéo, l'activation de la caméra est automatique dès que l'appel commence, permettant une communication visuelle en temps réel. À l'inverse, pour les appels vocaux, seule l'activation du microphone est requise, sans affichage vidéo. Cette séparation des interfaces permet une expérience utilisateur fluide et adaptée selon le type d'appel choisi.





Conclusion

Dans ce chapitre, nous avons présenté quelques interfaces et nous avons démontré quelques scénarios de tests que nous avons appliqué pour vérifier que l'application fonctionne correctement.



Difficultés rencontrées dans notre projet

- WebRTC (Web Real-Time Communication) est une technologie essentielle que nous avons utilisée pour implémenter les appels vocaux et vidéo en temps réel directement via le navigateur, sans nécessiter d'extensions ni d'applications tierces. Elle permet d'établir une communication peer-to-peer sécurisée entre utilisateurs, avec une faible latence et une qualité audio/vidéo optimisée. WebRTC prend en charge la capture, l'encodage, la transmission et la lecture des flux multimédias, tout en assurant la traversée des pare-feux et des NAT, ce qui en fait un outil puissant pour notre application.
- Cependant, son intégration a soulevé plusieurs défis techniques. L'un des principaux obstacles a été la gestion de l'accès aux périphériques (caméra et microphone), notamment sur certains navigateurs où les permissions doivent être gérées dynamiquement et avec attention à chaque session. De plus, la mise en place d'un système de notification d'appel entrant, capable d'alerter un utilisateur même lorsqu'il n'est pas dans l'interface de conversation active, a nécessité l'utilisation de modales personnalisées et d'un échange de signaux précis via WebSocket pour assurer la synchronisation côté client et serveur.
- Malgré ces difficultés, l'intégration de WebRTC s'est révélée être un élément clé de notre application, offrant une expérience fluide et interactive à travers deux interfaces distinctes : l'une pour les appels vocaux où seul le microphone est activé, et l'autre pour les appels vidéo, où la caméra est également mise en marche.



Conclusion Générale

Ce projet a abouti à la réalisation d'une application de messagerie en temps réel intégrant des fonctionnalités de chat, d'appels vocaux et vidéo. L'architecture choisie repose sur un front-end en React.js pour une interface fluide et réactive, et un back-end en NestJS qui offre une structure modulaire, maintenable et robuste.

La communication en temps réel a été assurée grâce à WebSocket via socket.io, tandis que les appels audio et vidéo ont été implémentés à l'aide de WebRTC, garantissant une interaction directe entre les utilisateurs. Les tests ont été facilités par Postman, qui a permis de vérifier la fiabilité des échanges en WebSocket.

Par ailleurs, l'utilisation des diagrammes UML a renforcé la compréhension et la cohérence du système, en modélisant clairement les différents flux et composants. Cette base solide permet d'envisager des améliorations futures comme le support des appels groupés ou l'optimisation de la scalabilité.

Enfin, malgré certaines difficultés techniques rencontrées tout au long du développement, ce projet nous a permis de renforcer nos compétences, d'approfondir notre maîtrise des technologies utilisées, et d'acquérir une expérience concrète en conception et mise en œuvre d'une application web complète.