



Rapport de projet tutoré  
3ème année d'ingénierie à l'école High Tech

**SecuChat : une plateforme de  
discussion sécurisée**

**Présenté par :** - Alae-eddine SAHBOU  
- Bouchra Griou

**Encadré par :** - Pr Nora El Amrani

**Année universitaire : 2022-2023**

# Résumé

La communication en ligne est un moyen de connecter les personnes à travers le monde, mais la sécurité et la confidentialité des données sont également des préoccupations importantes pour les utilisateurs. L'application de chat sécurisé "SecuChat" offre une expérience de chat en ligne sécurisée aux utilisateurs grâce à son chiffrement de bout en bout. Cette étude analyse les technologies de chiffrement et de sécurité utilisées par l'application SecuChat et les compare à d'autres applications de messagerie instantanée sécurisées telles que Telegram et Signal.

En plus de l'accent mis sur la sécurité et la confidentialité des données, l'application SecuChat offre une gamme de fonctionnalités pour améliorer l'expérience utilisateur. Les utilisateurs peuvent se connecter, s'inscrire et mettre à jour leur profil, ajouter des contacts favoris pour un accès rapide, et envoyer des messages texte en utilisant la messagerie instantanée de l'application.

L'application permet également la création de salles de discussion pour se connecter avec des groupes spécifiques. Bien que les fonctionnalités de suppression de message et de groupe n'existent pas actuellement dans l'application, elles pourraient être ajoutées à l'avenir pour améliorer encore l'expérience utilisateur.

L'application SecuChat offre une expérience de chat en ligne sécurisée et conviviale qui répond aux besoins des utilisateurs soucieux de leur sécurité et de leur vie privée. La base de données utilisée est SQLITE.

**Mots clés :** communication en ligne, sécurité, confidentialité, chat en ligne, chiffrement de bout en bout, technologie AES, comparaison d'applications de messagerie, développement logiciel, méthodologie de conception, gestion de projet, Django.

# Abstract

Online communication is a way to connect people across the world, but data security and privacy are also important concerns for users. The secure chat application "SecuChat" offers a secure online chat experience to users through end-to-end encryption. This study analyzes the encryption and security technologies used by the SecuChat application and compares them to other secure instant messaging applications such as Telegram and Signal.

In addition to the emphasis on data security and privacy, the SecuChat application offers a range of features to enhance the user experience. Users can log in, register, and update their profiles, add favorite contacts for quick access, and send text messages using the application's instant messaging feature.

The application also allows for the creation of chat rooms to connect with specific groups. While message deletion and group features do not currently exist in the application, they could be added in the future to further improve the user experience.

The SecuChat application provides a secure and user-friendly online chat experience that meets the needs of security and privacy-conscious users. The database used is SQLITE.

**Keywords:** online communication, security, privacy, online chat, end-to-end encryption, AES technology, comparison of messaging applications, software development, design methodology, project management, Django.

# Remerciements

Nous tenons à exprimer notre profonde gratitude à notre directrice de mémoire, Pr Nora El Amrani, pour son soutien, ses conseils et son expertise tout au long de ce travail. Nous avons grandement bénéficié de ses commentaires constructifs et de ses suggestions judicieuses qui ont grandement amélioré la qualité de ce rapport.

Nous tenons également à remercier chaleureusement toutes les personnes qui ont contribué de près ou de loin à la réalisation de ce travail, en particulier les enseignants et le personnel de l'école high-tech qui ont été d'une grande aide tout au long de notre formation.

Enfin, nous souhaitons exprimer notre gratitude à nos familles et amis pour leur soutien inconditionnel et leur encouragement tout au long de notre parcours académique. Leurs encouragements ont été une source de motivation et de détermination pour nous permettre d'atteindre nos objectifs et réaliser ce travail avec succès.

# Liste des abréviations

<b>AES:</b>	Advanced Encryption Standard
<b>API:</b>	Application Programming Interface
<b>DBMS:</b>	Database Management System
<b>GUI:</b>	Graphical User Interface
<b>HTTPS:</b>	Hypertext Transfer Protocol Secure
<b>JSON:</b>	JavaScript Object Notation
<b>MVC:</b>	Model-View-Controller
<b>P2P:</b>	Peer-to-Peer
<b>REST:</b>	Representational State Transfer
<b>SQL:</b>	Structured Query Language
<b>SSL:</b>	Secure Sockets Layer
<b>UML:</b>	Unified Modeling Language
<b>UI:</b>	User Interface

# Table des illustrations

<b>TABEAU 1: ÉTUDE COMPARATIVE DES DIFFÉRENTS PROCESSUS LES PLUS UTILISÉS DANS LE CADRE DE DÉVELOPPEMENT DE PROJETS INFORMATIQUES .....</b>	<b>25</b>
<b>FIGURE 1: CYCLE V .....</b>	<b>26</b>
<b>FIGURE 2: METHODE SCRUM .....</b>	<b>28</b>
<b>FIGURE 3: METHODOLOGIE DE CONCEPTION .....</b>	<b>31</b>
<b>FIGURE 4 : DIAGRAMME DE CAS D'UTILISATION .....</b>	<b>38</b>
<b>FIGURE 5 : DIAGRAMME DE CLASSE .....</b>	<b>39</b>
<b>FIGURE 6 : DIAGRAMME D'ACTIVITÉ .....</b>	<b>40</b>
<b>FIGURE 7 : DIAGRAMME DE SÉQUENCE .....</b>	<b>41</b>

# Sommaire

<b>RESUME .....</b>	<b>2</b>
<b>ABSTRACT .....</b>	<b>3</b>
<b>REMERCIEMENTS .....</b>	<b>4</b>
<b>LISTE DES ABREVIATIONS .....</b>	<b>5</b>
<b>TABLE DES ILLUSTRATIONS .....</b>	<b>5</b>
<b>SOMMAIRE .....</b>	<b>6</b>
<b>INTRODUCTION GENERALE .....</b>	<b>8</b>
<b>ÉTUDE BIBLIOGRAPHIQUE .....</b>	<b>10</b>
I.    APPLICATION CHAT .....	11
1. <i>Exemple de l'application de chat</i> .....	11
2. <i>Comparaison entre les applications de chat</i> .....	11
II.   LA SECURITE DE BOUT EN BOUT .....	12
1. <i>Le chiffrement de bout en bout</i> .....	12
2. <i>La vérification des clés de chiffrement</i> .....	13
3. <i>La protection des métadonnées</i> .....	13
III.  CONTRIBUTION .....	14
IV.  LES TECHNOLOGIES D'INFORMATION UTILISEES .....	15
1. <i>Framework web : Django</i> .....	15
2. <i>Cryptographie : AES</i> .....	18
3. <i>API RESTful : Django REST Framework</i> .....	19
4. <i>Base de données : SQLite</i> .....	19
5. <i>WebSocket : Daphne</i> .....	20
<b>ETUDE FONCTIONNELLE ET PLANIFICATION DU PROJET .....</b>	<b>21</b>
I.    PRESENTATION DU PROJET .....	22
II.   ANALYSE DU PROJET .....	22
III.  OBJECTIF DE L'ÉTUDE .....	22
IV.  CAHIER DES CHARGES .....	23
<i>Spécification des besoins fonctionnels :</i> .....	23
<i>Spécification des besoins techniques :</i> .....	23
<i>Etape de planification :</i> .....	23
V.    GESTION DU PROJET INFORMATIQUE .....	23
<i>Cycle de vie d'un logiciel :</i> .....	24
<i>Méthodes de gestion d'un projet informatique</i> .....	24

<i>Méthodologie de développement</i> .....	27
<i>La méthode SCRUM</i> : .....	27
VI. METHODOLOGIE AGILE ET SCRUM DANS LE PROJET.....	28
VII. METHODOLOGIE DE CONCEPTION.....	29
1. <i>Etude comparative entre MERISE et UML</i> .....	29
2. <i>La démarche adoptée</i> .....	30
<b>ANALYSE DE BESOIN, CONCEPTION ET METHODOLOGIE DE REALISATION</b> .....	<b>32</b>
I. ANALYSE DES BESOINS .....	33
II. ETUDE CONCEPTUELLE .....	37
<i>Diagramme de cas d'utilisation</i> .....	37
<i>Diagramme de classe</i> .....	39
<i>Diagramme d'activité</i> .....	40
<i>Diagramme de séquence</i> .....	41
<b>LE TRAVAIL TECHNIQUE</b> .....	<b>42</b>
I. INSTALLATION D'ENVIRONNEMENT DU TRAVAIL .....	43
II. CODE SQL DE LA BASE DE DONNEES.....	45
III. APPLICATION WEB (QUELQUES CAPTURES D'ECRANS) .....	50
<b>CONCLUSION</b> .....	<b>53</b>
<b>CONCLUSION AND FUTURE IMPROVEMENTS</b> .....	<b>54</b>

# Introduction générale

La communication en ligne est devenue un moyen incontournable pour les personnes cherchant à rester connectées. Toutefois, avec la popularité croissante des applications de messagerie instantanée, il y a également eu une préoccupation croissante quant à la sécurité et la confidentialité des données échangées. De nombreux utilisateurs de ces applications sont préoccupés par la protection de leurs informations personnelles, ainsi que la confidentialité de leurs conversations.

Dans ce rapport, nous allons examiner l'application de chat sécurisé "SecuChat", qui est une plateforme de messagerie instantanée conçue pour offrir une expérience de discussion en ligne sécurisée aux utilisateurs. Nous allons aborder plusieurs points clés pour fournir une analyse complète de l'application SecuChat, de sa technologie de chiffrement et de sa position sur le marché des applications de messagerie instantanée sécurisées.

Tout d'abord, nous présenterons l'application de chat SecuChat et expliquerons son fonctionnement. Nous étudierons également la technologie de chiffrement de bout en bout utilisée par cette application, qui garantit que seuls les utilisateurs qui participent à une conversation peuvent accéder aux messages échangés. Nous explorerons également les technologies de chiffrement couramment utilisées dans les applications de messagerie instantanée sécurisées, telles que AES et RSA.

Ensuite, nous comparerons SecuChat avec d'autres applications de messagerie instantanée sécurisées populaires telles que WhatsApp, Telegram et Signal. Nous évaluerons les avantages et les inconvénients de chaque application en matière de sécurité et de confidentialité.

Le rapport se compose de plusieurs chapitres qui approfondissent notre compréhension des différents concepts liés à la sécurité et la confidentialité dans les applications de messagerie instantanée sécurisées. Nous aborderons également l'étude fonctionnelle et la planification du projet, ainsi que l'analyse de besoin, conception et méthodologie de réalisation. Nous décrirons en détail les différentes étapes de développement de SecuChat, son installation et son fonctionnement, ainsi que les tests de sécurité effectués pour garantir la fiabilité de l'application.

En conclusion, ce rapport vise à fournir une analyse complète de l'application SecuChat, de sa technologie de chiffrement et de sa position sur le marché des



applications de messagerie instantanée sécurisées. Nous aborderons chaque aspect de l'application de manière détaillée, en présentant des exemples concrets et en fournissant des comparaisons avec d'autres applications de messagerie instantanée sécurisées populaires.

.

# **Étude Bibliographique**

## **I. Application chat**

### **1. Exemple de l'application de chat**

Il existe plusieurs applications de chat sécurisées sur le marché, dont Signal, Telegram et WhatsApp. Signal est une application de messagerie instantanée open-source qui utilise le protocole de chiffrement de bout en bout pour garantir la sécurité et la confidentialité des communications de ses utilisateurs. La plateforme propose également la possibilité de créer des groupes privés, des appels vocaux et vidéo sécurisés et la suppression automatique des messages. Telegram est une application de messagerie instantanée cloud-based qui permet aux utilisateurs de communiquer via des conversations individuelles et des groupes. La plateforme utilise également une technologie de chiffrement de bout en bout pour protéger les communications des utilisateurs. Telegram propose également des fonctionnalités telles que la possibilité de créer des groupes avec jusqu'à 200 000 membres, des appels vocaux et vidéo sécurisés, des messages programmables, des bots et bien plus encore. Enfin, WhatsApp est une application de messagerie instantanée populaire qui utilise également le chiffrement de bout en bout pour protéger les communications de ses utilisateurs. La plateforme propose également des fonctionnalités telles que les appels vocaux et vidéo, la création de groupes, les messages vocaux et la possibilité d'envoyer des fichiers. En fin de compte, le choix entre Signal, Telegram et WhatsApp dépendra des besoins et des préférences individuels de l'utilisateur..

### **2. Comparaison entre les applications de chat**

Lorsqu'il s'agit de choisir entre différentes applications de chat sécurisées, il est important de prendre en compte plusieurs facteurs tels que la sécurité, la facilité d'utilisation, les fonctionnalités disponibles et la popularité de la plateforme. En termes de sécurité, Signal est considérée comme la plateforme la plus sûre grâce à son protocole de chiffrement de bout en bout. Telegram, quant à elle, utilise également le chiffrement de bout en bout, mais il n'est pas activé par défaut pour toutes les conversations.

En termes de fonctionnalités, Telegram offre plus de possibilités que Signal, telles que la création de groupes plus importants et des fonctionnalités de bot plus avancées. Cependant, Signal se concentre davantage sur la simplicité d'utilisation et propose une expérience utilisateur plus épurée.

En termes de popularité, Telegram est plus populaire que Signal car elle est disponible dans plus de pays et propose des fonctionnalités plus avancées pour les utilisateurs. Cependant, Signal gagne en popularité grâce à sa réputation de sécurité et de confidentialité des données des utilisateurs.

Il convient également de mentionner l'application WhatsApp, qui est l'une des applications de chat les plus populaires au monde. Bien que WhatsApp utilise également le chiffrement de bout en bout, l'application a été critiquée pour ses politiques de confidentialité controversées. En janvier 2021, WhatsApp a mis à jour ses conditions d'utilisation pour permettre le partage de certaines données des utilisateurs avec Facebook, sa société mère, ce qui a suscité des inquiétudes quant à la confidentialité des données des utilisateurs.

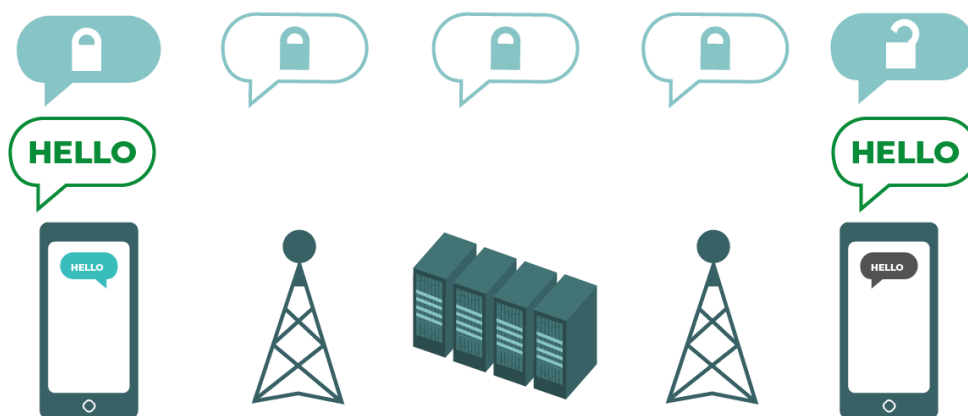
En fin de compte, le choix entre Signal, Telegram et WhatsApp dépendra des besoins et des préférences individuels de l'utilisateur, ainsi que de sa tolérance aux compromis en matière de sécurité et de confidentialité des données.

## **II. La sécurité de bout en bout**

La sécurité de bout en bout est l'un des éléments clés des applications de messagerie instantanée. Elle permet de protéger les échanges entre les utilisateurs en garantissant que seuls les destinataires peuvent accéder aux messages envoyés. Dans cette partie du rapport, nous allons explorer en détail les différentes techniques utilisées pour assurer la sécurité de bout en bout des applications de messagerie.

### **1. Le chiffrement de bout en bout**

Le chiffrement de bout en bout est la technique la plus courante utilisée pour garantir la sécurité des échanges dans les applications de messagerie instantanée. Il permet de chiffrer les messages dès qu'ils sont envoyés depuis l'appareil de l'utilisateur et de ne les déchiffrer qu'une fois qu'ils sont arrivés chez le destinataire. Le chiffrement de bout en bout garantit que même les serveurs qui hébergent les messages ne peuvent pas les lire, car ils sont chiffrés avant d'être envoyés.



Les algorithmes de chiffrement utilisés pour le chiffrement de bout en bout peuvent varier selon les applications. Les plus couramment utilisés sont AES et RSA, qui sont des algorithmes de chiffrement symétrique et asymétrique respectivement. AES est plus rapide et plus efficace pour le chiffrement des messages, tandis que RSA est plus sécurisé car il utilise des clés publiques et privées.

## 2. La vérification des clés de chiffrement

La vérification des clés de chiffrement est une autre technique utilisée pour garantir la sécurité de bout en bout des échanges dans les applications de messagerie instantanée. Elle permet aux utilisateurs de s'assurer que la clé de chiffrement utilisée pour chiffrer les messages appartient bien au destinataire. Pour cela, les applications de messagerie instantanée utilisent des **codes QR** (Les codes QR sont des codes-barres en deux dimensions lus rapidement par les smartphones ou les tablettes. Ils contiennent des informations telles que des URL, des numéros de téléphone, des adresses e-mail, des cartes de visite, etc. Ils sont largement utilisés dans les publicités, les magazines, les affiches et les emballages de produits pour un accès facile à des informations en ligne.) ou des empreintes de clé pour vérifier l'identité des utilisateurs.

## 3. La protection des métadonnées

Les métadonnées, également appelées données d'en-tête, sont des informations qui décrivent les caractéristiques d'autres données. Dans le contexte des applications de messagerie instantanée, les métadonnées peuvent inclure des informations telles que l'heure et le lieu d'envoi d'un message, ainsi que les identités des utilisateurs impliqués dans la conversation. La protection des métadonnées est tout aussi importante que la protection du contenu des messages, car les métadonnées peuvent fournir des

informations sensibles qui peuvent être utilisées pour suivre les activités des utilisateurs. Les applications de messagerie instantanée utilisent différentes techniques pour protéger les métadonnées, telles que la diffusion de messages via plusieurs serveurs pour cacher leur provenance. Cela garantit que les métadonnées sont également protégées et que les utilisateurs peuvent échanger des messages en toute sécurité et confidentialité.

En plus de protéger le contenu des messages, il est également important de protéger les métadonnées associées aux messages. Les métadonnées peuvent fournir des informations sensibles telles que l'heure et le lieu d'envoi des messages, ainsi que les identités des utilisateurs. Les applications de messagerie instantanée utilisent différentes techniques pour protéger les métadonnées, comme la diffusion de messages via plusieurs serveurs pour cacher leur provenance.

En résumé, la sécurité de bout en bout est un élément essentiel des applications de messagerie instantanée. Elle permet de protéger les échanges entre les utilisateurs en garantissant que seuls les destinataires peuvent accéder aux messages. Les techniques utilisées pour assurer la sécurité de bout en bout incluent le chiffrement de bout en bout, la vérification des clés de chiffrement et la protection des métadonnées. Ces techniques garantissent une sécurité et une confidentialité maximales pour les utilisateurs des applications de messagerie instantanée.

### **III. CONTRIBUTION**

Dans le cadre de ce mini-projet, nous avons développé l'application de chat SecuChat, avec un accent particulier sur la sécurité de bout en bout. Notre application permet aux utilisateurs de communiquer de manière confidentielle et sécurisée grâce à l'utilisation de la cryptographie. Nous avons implémenté un algorithme de chiffrement AES pour garantir la confidentialité des messages échangés entre les utilisateurs. Nous avons également ajouté une fonctionnalité de génération de clés publiques et privées pour chaque utilisateur, ce qui garantit que seuls les utilisateurs autorisés peuvent accéder aux conversations.

Notre application SecuChat permet également aux utilisateurs de créer des groupes de discussion et d'ajouter des membres en toute sécurité. Nous avons mis en place un système de gestion de groupes qui assure que les utilisateurs non autorisés ne peuvent pas rejoindre les conversations. Les messages échangés dans les groupes sont également chiffrés pour assurer la confidentialité des discussions.

En plus de cela, notre application offre des fonctionnalités de base telles que l'envoi de messages texte. Nous avons conçu une interface utilisateur conviviale et facile à utiliser pour permettre une utilisation intuitive de l'application.

Notre contribution nous a permis de mieux comprendre les enjeux de la sécurité de l'information dans les applications de communication en ligne. Nous avons acquis des connaissances pratiques sur la mise en place de la sécurité de bout en bout dans les applications de chat, ainsi que sur l'importance de la cryptographie pour garantir la confidentialité des données personnelles. Enfin, cette étude nous a également sensibilisés à l'importance de la recherche continue dans le domaine de la sécurité de l'information pour assurer la protection des utilisateurs en ligne.

## **IV. Les technologies d'information utilisées**

Le choix des technologies pour le développement de SecuChat doit être minutieusement étudié pour garantir un résultat optimal et répondre aux besoins de l'application. Nous avons considéré plusieurs critères pour prendre la décision finale.

### **1. Framework web : Django**

Django est un framework de développement web open source écrit en Python. Il est conçu pour simplifier le processus de création de sites web complexes en fournissant une structure de base pour la gestion des tâches courantes de développement web telles que la gestion de base de données, le traitement des formulaires et la gestion de l'authentification utilisateur. Django suit le modèle MVC (Modèle-Vue-Contrôleur) pour séparer la logique de présentation de la logique de traitement. Il est utilisé pour développer des sites web de grande envergure tels que des réseaux sociaux, des portails de commerce électronique et des applications d'entreprise. Django est très apprécié pour sa facilité d'utilisation, sa documentation complète et sa communauté de développeurs active et soutenue.

Le choix de Django comme framework web pour SecuChat a été fait pour plusieurs raisons. Tout d'abord, Django est un framework web populaire qui est connu pour sa sécurité, sa robustesse et sa scalabilité. De plus, il offre une grande flexibilité pour la conception de la base de données et permet une intégration facile avec des outils tiers. Enfin, Django est soutenu par une communauté active et dispose d'une documentation abondante.



## Étude comparative avec d'autres frameworks web

Le choix d'un framework web est crucial pour tout projet de développement web. Django n'est pas le seul framework web disponible, il existe de nombreux autres frameworks tels que Ruby on Rails, Flask, Express.js, Laravel, et bien d'autres. Cette section explore les différences entre Django et certains de ces autres frameworks populaires.

**Ruby on Rails :** Ruby on Rails est un framework web open-source écrit en Ruby. Il est souvent comparé à Django, car ils partagent des similitudes en termes de philosophie de conception et de fonctionnalités. Les deux frameworks sont orientés vers la convention plutôt que la configuration, ce qui signifie qu'ils fournissent des structures prédéfinies pour la création d'applications web. Cependant, Ruby on Rails est souvent considéré comme plus facile à apprendre pour les débutants, car il offre une syntaxe plus concise que Django.

**Flask :** Flask est un framework web minimaliste et flexible pour Python. Il est souvent considéré comme plus facile à apprendre que Django en raison de sa simplicité. Flask ne fournit que les fonctionnalités de base pour la création d'applications web, ce qui permet aux développeurs de personnaliser leur stack de développement. Cependant, Django offre plus de fonctionnalités intégrées que Flask, ce qui peut accélérer le processus de développement.

**Express.js :** Express.js est un framework web pour Node.js, écrit en JavaScript. Express.js est souvent considéré comme plus rapide que Django car il utilise le moteur



de traitement de JavaScript V8 de Google. Express.js est également plus adapté pour la création d'applications web en temps réel telles que les applications de chat, car il offre des fonctionnalités pour la gestion des connexions websocket.

**Laravel** : Laravel est un framework web open-source écrit en PHP. Il est souvent considéré comme plus facile à apprendre que Django, car il utilise une syntaxe plus intuitive. Laravel offre également des fonctionnalités intégrées pour la gestion de tâches de fond, ce qui le rend plus adapté pour la création d'applications de traitement de données en arrière-plan.

En ce qui concerne les exemples d'applications de chat développées avec Django, il existe plusieurs projets open-source disponibles sur GitHub. En voici quelques-uns :

**Channels** - <https://github.com/django/channels> : Channels est une extension de Django qui permet de créer des applications Web temps réel, telles que des chats. Il est basé sur WebSockets et offre une API simple pour la création de canaux de communication bidirectionnels.

**Django Chat** - <https://github.com/madewithtea/django-chat> : Django Chat est un exemple d'application de chat en temps réel développé avec Django et Channels. Il dispose de fonctionnalités telles que l'envoi de messages texte, l'envoi d'images et la possibilité de créer des salles de discussion.

**Django Q&A** - <https://github.com/micropyramid/Django-QA> : Bien que ce ne soit pas un projet de chat en tant que tel, Django Q&A est un exemple d'application de questions-réponses développée avec Django. Il peut facilement être étendu pour inclure des fonctionnalités de chat en temps réel.

**Django Real-Time Chat** - <https://github.com/mrussell247/django-realtime-chat> : C'est un autre exemple d'application de chat en temps réel construit avec Django et Channels. Il offre des fonctionnalités telles que l'envoi de messages en direct, la création de salles de discussion et la gestion des utilisateurs.

Ces exemples montrent comment il est possible de créer des applications de chat en temps réel avec Django. Ils offrent également une base solide pour commencer à construire l'application SecuChat.

**En conclusion**, le choix d'un framework web dépend des besoins spécifiques de chaque projet. Django est souvent choisi pour sa robustesse et sa sécurité, mais il peut être plus difficile à apprendre pour les débutants. Les autres frameworks tels que Flask et Express.js sont plus adaptés pour les projets plus petits et les applications en temps réel. Laravel est souvent choisi pour la création d'applications de traitement de données en arrière-plan. Finalement, le choix d'un framework web doit être basé sur une étude approfondie des besoins du projet.

## **2. Cryptographie : AES**

AES a été choisi comme algorithme de chiffrement pour SecuChat car il est l'un des algorithmes les plus sûrs disponibles. Il est également rapide et facile à mettre en œuvre. En utilisant AES, nous pouvons garantir que les messages envoyés via SecuChat sont cryptés de manière sécurisée et ne peuvent pas être interceptés par des tiers malveillants.



La technologie de chiffrement AES (Advanced Encryptions Standard) est l'un des principaux moyens de sécuriser les communications en ligne. AES est un algorithme de chiffrement de blocs symétriques, ce qui signifie qu'il utilise une seule clé pour le chiffrement et le déchiffrement des données. Cette clé est connue uniquement des deux parties de la communication et est utilisée pour garantir la confidentialité et l'intégrité des données échangées.

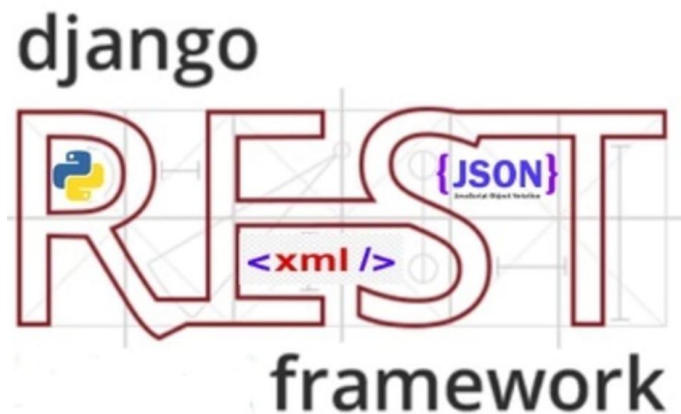
AES est largement utilisé dans les applications de messagerie instantanée sécurisées, y compris SecuChat, car il offre une sécurité élevée et une efficacité de traitement rapide. Les algorithmes AES utilisent des blocs de données de 128 bits pour le chiffrement, ce qui les rend extrêmement difficiles à décrypter pour les personnes non autorisées. De plus, les clés AES peuvent être générées avec une longueur de clé allant jusqu'à 256 bits, ce qui garantit une sécurité encore plus élevée.

La sécurité offerte par AES est également renforcée par le fait qu'il est régulièrement évalué et testé par des experts en sécurité de l'industrie. Le NIST (National Institute of Standards and Technology) des États-Unis est responsable de la normalisation et de la certification de l'algorithme AES. De plus, de nombreux gouvernements et organisations de sécurité utilisent AES pour protéger leurs communications.

En fin de compte, AES est un élément essentiel de la sécurité en ligne et un pilier de la protection des communications privées. En utilisant des technologies telles que AES, SecuChat peut garantir à ses utilisateurs une sécurité et une confidentialité maximales lors de la transmission de données sensibles.

### **3. API RESTful : Django REST Framework**

Django REST Framework a été sélectionné comme l'outil pour créer l'API RESTful de SecuChat. Cela est dû à sa simplicité d'utilisation, sa documentation exhaustive et sa popularité. En utilisant Django REST Framework, nous pouvons facilement créer une API qui est facilement accessible pour les clients.



### **4. Base de données : SQLite**

SQLite a été choisi comme le système de gestion de base de données pour SecuChat car il est facile à utiliser, rapide et léger. Il ne nécessite pas de serveur de base de données distinct, ce qui signifie qu'il est facile à configurer et à déployer. En outre, SQLite est parfait pour les applications légères et peut gérer de grandes quantités de données sans sacrifier la performance.



## 5. WebSocket : Daphne

Daphne a été choisi comme serveur WebSocket pour SecuChat car il est rapide, léger et facile à utiliser. Il a été créé pour être utilisé avec Django, ce qui le rend parfait pour notre application. En utilisant Daphne, nous pouvons garantir que les communications en temps réel entre les clients sont rapides et fiables.



**En conclusion,** ces critères ont été soigneusement examinés et choisis pour garantir que SecuChat est une application sécurisée, rapide, fiable et facile à utiliser. Django a été choisi pour son cadre de développement efficace et sa robustesse en matière de sécurité, tandis que AES a été sélectionné pour sa sécurité élevée et sa rapidité de traitement. Django REST Framework a été utilisé pour faciliter le développement de l'API, SQLite pour la gestion efficace des données, et Daphne pour son support WebSocket. En utilisant ces technologies, nous sommes convaincus que SecuChat répondra aux besoins de nos utilisateurs de manière satisfaisante en leur offrant une expérience sécurisée et conviviale.

# **Etude fonctionnelle et planification du projet**

## I. Présentation du projet

Avant d'entamer toute analyse, il est crucial de poser des questions pertinentes afin de bien comprendre le problème et d'identifier les causes profondes. La méthode QQOCP de questionnement, grâce à sa simplicité et sa logique, nous a permis de collecter les données nécessaires pour dresser un état des lieux du projet. Ainsi, le projet SecuChat consiste à concevoir, réaliser et développer une application web de chat sécurisé accessible à toutes les personnes souhaitant échanger des messages en toute sécurité. Cette application sera développée au sein du service informatique et sera accessible via une plateforme web. La fonctionnalité de l'application sera développée en suivant une méthodologie rigoureuse et en respectant les bonnes pratiques de développement logiciel. Enfin, le projet SecuChat vise à faciliter les échanges sécurisés entre les utilisateurs en centralisant les données et en assurant la confidentialité des échanges.

## II. Analyse du projet

L'analyse du projet "SecuChat" a permis de définir les fonctionnalités de l'application et les exigences du système. Les principales fonctionnalités de l'application incluent :

- **Authentification** : les utilisateurs doivent s'inscrire et se connecter pour accéder aux fonctionnalités de l'application.
- **Gestion des profils d'utilisateurs** : les utilisateurs peuvent modifier leurs noms et ajouter une image de profil.
- **Envoi de messages** : les utilisateurs peuvent envoyer des messages dans les salles de chat.
- **Gestion des salles de chat** : les utilisateurs peuvent modifier les noms des salles et des participants, ainsi que supprimer des salles de chat.
- **Chat de groupe** : les utilisateurs peuvent créer des salles de chat et y ajouter des participants.

## III. Objectif de l'étude

L'objectif de l'étude est de concevoir et de développer une application Web de messagerie sécurisée, qui permet aux utilisateurs de communiquer en temps réel de manière sûre et efficace. L'application doit être conviviale, facile à utiliser et offrir des fonctionnalités avancées pour répondre aux besoins des utilisateurs. La conception de l'application doit être axée sur la sécurité et la confidentialité des données de l'utilisateur, en utilisant des protocoles de sécurité robustes tels que HTTPS pour protéger les communications entre les utilisateurs.

## **IV. Cahier des charges**

Le projet "SecuChat" a pour objectif de fournir une plateforme de chat sécurisée pour les utilisateurs souhaitant communiquer de manière privée et confidentielle. Dans cette section, nous allons établir les spécifications des besoins fonctionnels et techniques pour la réalisation du projet.

### **Spécification des besoins fonctionnels :**

1. Authentification sécurisée pour l'inscription et la connexion des utilisateurs.
2. Système de chiffrement de bout en bout pour garantir la sécurité des communications des utilisateurs.
3. Messagerie instantanée basée sur les salons de discussion.
4. Possibilité de créer, supprimer, rejoindre et quitter des salons de discussion.
5. Possibilité de changer le nom des salons de discussion.
6. Affichage de la liste des utilisateurs présents dans un salon de discussion.
7. Possibilité de personnaliser le profil de l'utilisateur avec des images et de changer le nom d'utilisateur.
8. Possibilité de créer des groupes privés.
9. Possibilité de supprimer les messages envoyés.

### **Spécification des besoins techniques :**

1. Utilisation de technologies de chiffrement telles que AES ou RSA pour protéger les données des utilisateurs.
2. Utilisation de Django pour le développement du backend.
3. Utilisation de JavaScript et des WebSockets pour le développement du frontend.
4. Utilisation de SQLite pour la base de données.
5. Utilisation de Git pour la gestion de version.
6. Utilisation de Visual Studio Code / PyCharm / Atom comme environnement de développement.

### **Etape de planification :**

1. Analyse des besoins et des spécifications.
2. Conception de l'architecture du système.
3. Développement du backend avec Django.
4. Développement du frontend avec JavaScript et des WebSockets.
5. Intégration de la base de données SQLite.
6. Tests de validation pour chaque fonctionnalité développée.
7. Déploiement du système.
8. Maintenance et support continu.

## **V. Gestion du projet informatique**

La gestion de projet (ou conduite de projet) est une démarche visant à organiser de bout en bout le bon déroulement d'un projet.

## Cycle de vie d'un logiciel :

Le cycle de vie d'une application web désigne toutes les étapes du développement d'une application web, de sa conception à sa disparition. L'objectif d'un tel découpage est de permettre de définir des jalons intermédiaires permettant la validation du développement une application web, c'est-à-dire la conformité de l'application web avec les besoins exprimés, et la vérification du processus de développement, c'est-à-dire l'adéquation des méthodes mises en œuvre.

Le cycle de vie du logiciel comprend généralement au minimum les étapes suivantes :

- **Définition des objectifs** : Cette étape consiste à définir la finalité du projet et son Inscription dans une stratégie globale.
- **Analyse des besoins et faisabilité** : C'est-à-dire l'expression, le recueil et la formalisation des besoins du demandeur (le client) et de l'ensemble des contraintes, puis l'estimation de la faisabilité de ces besoins.
- **Spécifications ou conception générale** : Il s'agit de l'élaboration des spécifications de l'architecture générale du logiciel.
- **Conception détaillée** : Cette étape consiste à définir précisément chaque sous-ensemble du logiciel.
- **Codage (Implémentation ou programmation)** : C'est la traduction dans un langage de programmation des fonctionnalités définies lors de phases de conception.
- **Tests unitaires** : Ils permettent de vérifier individuellement que chaque sous-ensemble du logiciel est implémenté conformément aux spécifications.
- **Intégration** : L'objectif est de s'assurer de l'interfaçage des différents éléments (Modules) du logiciel. Elle fait l'objet de tests d'intégration consignés dans un document.
- **Qualification (ou recette)** : C'est-à-dire la vérification de la conformité du logiciel aux spécifications initiales.
- **Documentation** : Elle vise à produire les informations nécessaires pour l'utilisation du logiciel et pour des développements ultérieurs.
- **Mise en production** : C'est le déploiement sur site du logiciel.
- **Maintenance** : Elle comprend toutes les actions correctives (maintenance corrective) et évolutives (Maintenance évolutive) sur le logiciel.

## Méthodes de gestion d'un projet informatique

Tout projet informatique requiert un processus de développement bien défini pour assurer son succès. En fait, le choix d'un processus de développement pour le projet était crucial dans le sens où il affectera aussi bien la planification que les spécifications fonctionnelles et techniques.



*Tableau 1: étude comparative des différents processus les plus utilisés dans le cadre de développement de projets informatiques*

	Description	Avantages	Inconvenient
<b>Modèle en Cascade</b>	Les étapes de développement sont réalisées de façon séquentielle.	<ul style="list-style-type: none"> <li>- Facile à utiliser et à comprendre.</li> <li>- Structure simple pour une équipe inexpérimentée</li> <li>- Fonctionne bien quand la qualité est beaucoup plus importante que les couts et le temps</li> </ul>	<ul style="list-style-type: none"> <li>- Sensibilité aux Nouveaux besoins : refaire tout le procédé</li> <li>-Une phase ne peut démarrer que si l'étape précédente est finie</li> <li>-Le produit n'est visible qu'à la fin.</li> <li>-Les risques se décalent vers la fin.</li> <li>-Très faible implication du client.</li> </ul>
<b>Modèle Incrémental</b>	<ul style="list-style-type: none"> <li>- Il trie les spécifications par priorités.</li> <li>- Chaque incrément est une construction partielle du Logiciel.</li> <li>- Chaque incrément implémente un ou plusieurs spécifications jusqu'à ce que la totalité du produit soit finie.</li> </ul>	<ul style="list-style-type: none"> <li>- Développement de fonctionnalités à risque en premier.</li> <li>- Chaque incrément donne un produit fonctionnel.</li> <li>- Utiliser l'approche « Diviser pour régner ».</li> <li>-Le client entre en relation avec le produit très tôt.</li> </ul>	<ul style="list-style-type: none"> <li>- Exige une bonne planification et une bonne conception</li> <li>- Exige une vision sur le produit fini pour pouvoir bien le diviser en incréments.</li> <li>-Le cout total du système peut être cher.</li> </ul>
<b>Cycle en V</b>	<ul style="list-style-type: none"> <li>- Il est caractérisé par le parallélisme.</li> <li>- Les étapes de développement et horizontalement à la vérification.</li> <li>- Il permet, en cas d'anomalie de limiter un retour aux étapes précédentes.</li> </ul>	<ul style="list-style-type: none"> <li>- Met l'accent sur les tests et la validation et donc accroît la qualité.</li> <li>- Chaque livrable doit être testable.</li> <li>- Facile à utiliser et planifier.</li> <li>- Attitude proactive qui signifie que le travail effectué lors des phases de conception permet de limiter les risques et dérivés pendant les phases de tests. Il s'agit de mettre en face de chaque phase de spécification un moyen de vérification.</li> </ul>	<ul style="list-style-type: none"> <li>- Documentation importante.</li> <li>- Ne contient pas d'activité d'analyse de risque.</li> </ul>

Au terme de cette étude comparative, il s'avère que le processus de développement le plus adéquat afin de mener dans les meilleures conditions notre projet, est le cycle en V.

Cette méthodologie est dérivée du cycle de vie de produit en cascade développé pour l'industrie lourde par l'ingénieur Winston W. Royce en 1970. Mise au point à

l'origine pour le développement de systèmes satellites intégrant des technologies multiples en 1980, ce cycle a été ensuite adapté pour l'informatique et référencé en 1990 par les ingénieurs Kevin John Forsberg et Harold Mooz. Composée de 9 étapes, elle est découpée en 3 parties :

- Sur la pointe du V, la réalisation
- Sur la partie droite du V, la phase de tests

Elle met en avant l'anticipation et la préparation des étapes montantes grâce aux phases en vis-à-vis de la partie descendante. Ceci permet de limiter les retours aux étapes précédentes.

Cette méthodologie met en avant un recueil important des besoins et spécifications en amont afin de planifier l'ensemble du projet précisément. Dans ce contexte, il s'agit d'établir à l'avance tous les éléments nécessaires à la réalisation et la phase de tests.

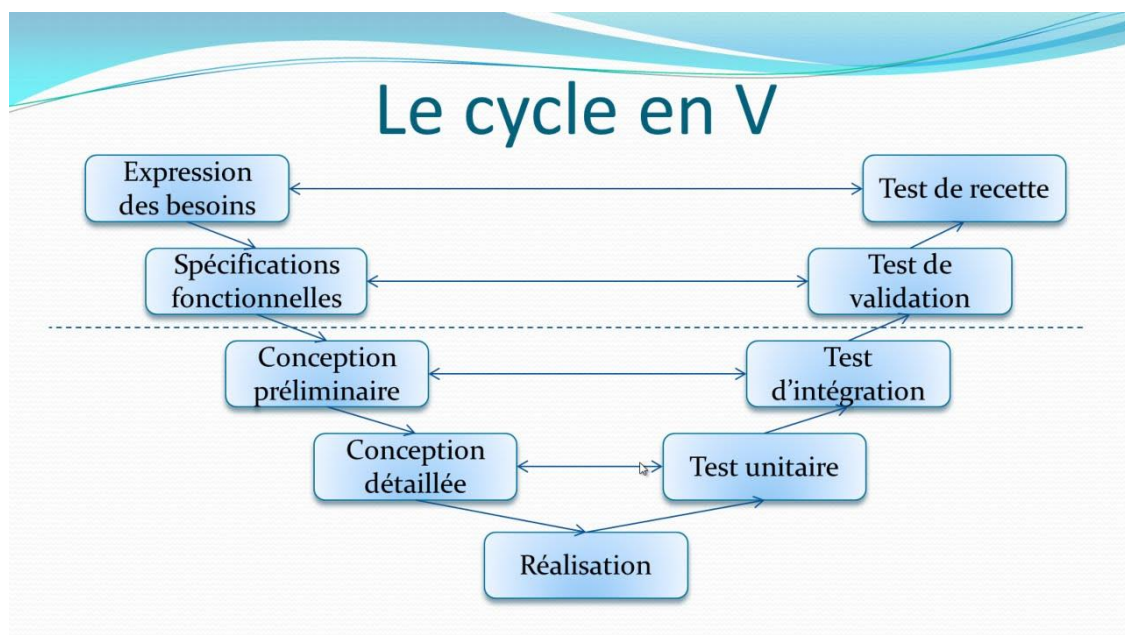


Figure 1: Cycle V

Le processus cycle en V s'appuie sur UML tout au long du cycle de développement, car les différents diagrammes de ce dernier permettent par leur facilité et clarté, de bien modéliser le système à chaque étape.

Pour la réalisation de notre système, nous nous focalisons sur les quatre diagrammes suivants :

- Diagramme de Cas d'utilisation

- Diagramme de classe
- Diagramme d'activité

## **Méthodologie de développement**

Pour la partie codage et réalisation, le cycle en V n'impose aucune règle d'organisation de développement contrairement aux phases en amont. Ainsi, elle laisse la liberté à l'équipe du projet de s'organiser elle-même. Dans notre cas, nous avons adopté l'esprit organisationnel de Scrum. Ce dernier est une méthodologie de développement de logiciels qui s'intéresse plutôt à la gestion du projet qu'aux aspects techniques.

Les méthodes agiles utilisent un principe de développement itératif qui consiste à découper le projet en plusieurs étapes qu'on appelle « itérations ».

Ces itérations sont en fait des mini-projets définis avec le client en détaillant les différentes fonctionnalités qui seront développées en fonction de leur priorité. Le chef de projet établit alors une macro-planning correspondant aux tâches nécessaires pour le développement de ces fonctionnalités.

Le but est d'assumer le fait que l'on ne peut pas tout connaître et anticiper quel que soit notre expérience. On découpe alors le projet en itérations plutôt que de tout prévoir et planifier en sachant que des imprévus arriveront en cours de route.

Voici les avantages du développement itératif :

- Meilleure qualité de la communication : L'utilisateur a la possibilité de clarifier ses exigences au fur et à mesure.
- Meilleure visibilité : Le client a eu meilleure visibilité sur l'avancement des travaux.
- Meilleur contrôle de la qualité : les tests sont effectués en continu.
- Meilleure détection des risques : Les risques sont détectés plus tôt.
- Motivation et confiance de l'équipe : satisfaction d'atteindre un objectif fixé.
- Contrôle des coûts : le projet peut être arrêté s'il n'y a plus de budget.

## **La méthode SCRUM :**

- C'est un cadre de travail permettant de répondre aux problèmes complexes et changeants.

Elle favorise le développement productif et créatif de produits de grande valeur.

- C'est un canevas permettant l'amélioration des pratiques de gestion et de développement.

- SCRUM a été présentée en 1990 par Jeff McKenna et Ken Schwaber, et est appliquée depuis. De nombreux contributeurs ont œuvré pour son amélioration.

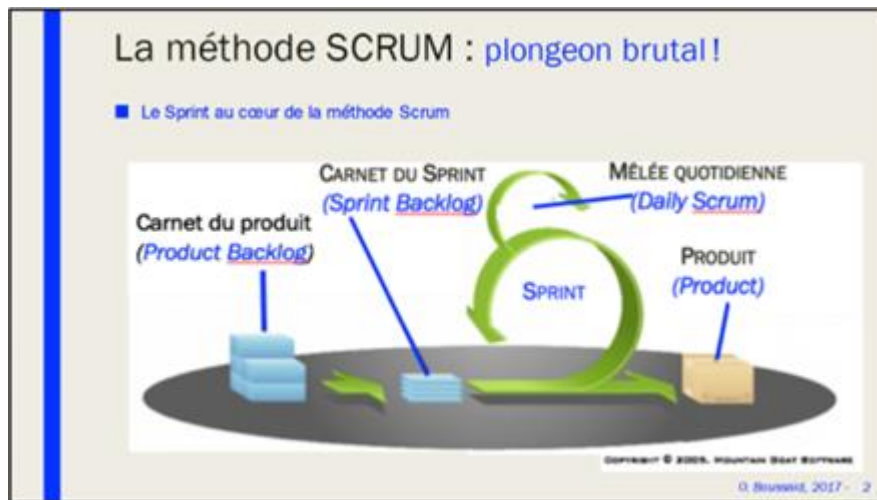


Figure 2: Méthode Scrum

## VI. Méthodologie Agile et Scrum dans le projet

Pour gérer efficacement le projet SecuChat, notre équipe de deux étudiants a adopté une approche Agile de développement logiciel en se basant sur les principes de la méthodologie Scrum. Nous avons cherché à appliquer les bonnes pratiques de gestion de projet afin de livrer un produit de qualité dans les délais impartis.

Nous avons divisé le projet en sprints d'une durée d'un mois, chaque sprint correspondant à une itération du cycle de développement. Au début de chaque sprint, nous avons tenu une réunion de planification pour discuter des fonctionnalités à implémenter et des tâches à accomplir pour atteindre les objectifs du sprint. Nous avons également évalué les risques potentiels et les obstacles éventuels qui pourraient entraver notre progression.

Au cours de chaque sprint, nous avons maintenu un suivi quotidien de l'avancement du projet, en nous réunissant régulièrement pour discuter de notre progression et pour nous entraider en cas de blocages. Nous avons appliqué les principes de collaboration et de communication de Scrum malgré notre petite équipe.

À la fin de chaque sprint, nous avons tenu une réunion de revue de sprint pour présenter les fonctionnalités développées et recueillir des commentaires pour améliorer le projet. Cela nous a permis de nous assurer que nous étions sur la bonne voie et de corriger les erreurs éventuelles.

Enfin, nous avons organisé des réunions de rétrospective de sprint pour évaluer notre travail et identifier les points à améliorer pour le sprint suivant. Nous avons utilisé ces

réunions pour discuter des difficultés rencontrées et pour proposer des idées pour améliorer notre processus de développement.

L'utilisation de la méthodologie Scrum nous a permis de livrer un produit de qualité malgré notre petite équipe. Nous avons réussi à respecter les délais impartis tout en restant alignés sur les objectifs de notre projet. La communication et la collaboration étroite entre les membres de l'équipe ont été des facteurs clés de notre réussite.

## **VII. Méthodologie de conception**

### **1. Etude comparative entre MERISE et UML**

**MERISE** (Méthode d'Etude et de Réalisation Informatique pour les Systèmes d'Entreprise) est une méthode d'analyse et de réalisation des systèmes d'information qui est élaborée en plusieurs étapes : schéma directeur, étude préalable, étude détaillée et la réalisation. Alors qu'UML (Unified Modeling Language), est un langage de modélisation des systèmes standard, qui utilise des diagrammes pour représenter chaque aspect d'un système i.e. : statique, dynamique, ...en s'appuyant sur la notion d'orienté objet qui est un véritable atout pour ce langage.

#### **➤ Merise ou UML ?**

Les "méthodologues" disent qu'une méthode, pour être opérationnelle, doit avoir 3 composantes :

- Une démarche (les étapes, phases et tâches de mise en œuvre),
- Des formalismes (les modélisations et les techniques de transformation),
- Une organisation et des moyens de mise en œuvre.

#### **Merise :**

- S'est attachée, en son temps, à proposer un ensemble "cohérent" sur ces trois composantes. Certaines ont vieilli et ont dû être réactualisées (la démarche), d'autres "tiennent encore la route" (les modélisations).
- Se positionne comme une méthode de conception de SI organisationnel, plus tournée vers la compréhension et la formalisation des besoins du métier que vers la réalisation de logiciel. En sens, Merise se réclame plus de l'ingénierie du SI métier que du génie logiciel.
- Jamais Merise ne s'est voulu une méthode de développement de logiciel ni de programmation.

**Merise** est encore tout à fait valable pour :

- La modélisation des données en vue de la construction d'une base de données relationnelle
- La modélisation des processus métiers d'un SI automatisé en partie par du logiciel.
- La formalisation des besoins utilisateur dans le cadre de cahier des charges utilisateur, en vue de la conception d'un logiciel adapté.

**UML :**

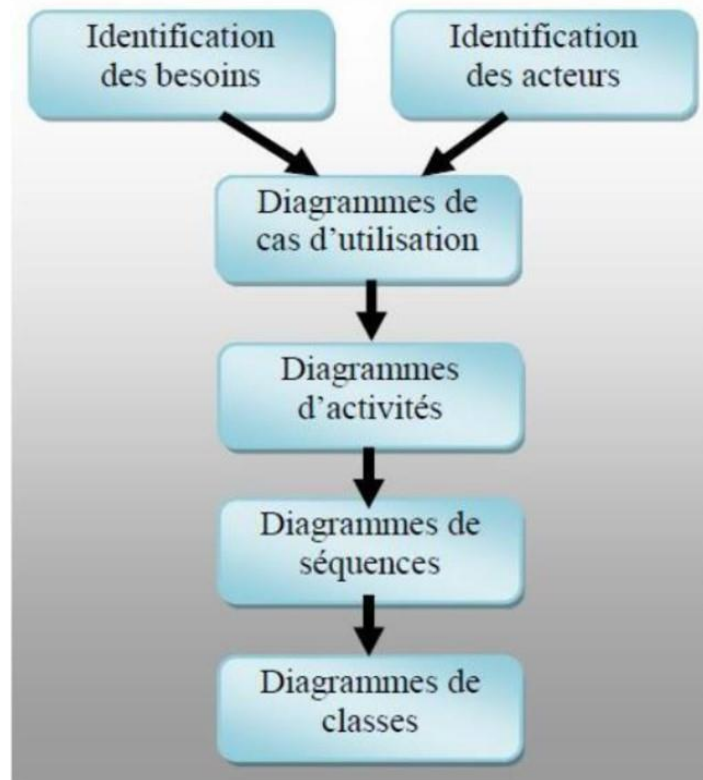
- Se positionne exclusivement comme un ensemble de formalismes. Il faut y associer une démarche et une organisation pour constituer une méthode.
- Par son origine (la programmation objet) s'affirme comme un ensemble de formalismes pour la conception de logiciel à base de langage objet.

**UML** est idéal pour :

- Concevoir et déployer une architecture logicielle développée dans un langage objet (Java, C++...). Certes UML, dans sa volonté "unificatrice" a proposé des formalismes,
- Pour modéliser les données (le modèle de classes réduit sans méthodes et stéréotypé en entités), mais avec des lacunes que ne présentait pas l'entité relation de Merise.

## **2. La démarche adoptée**

Après cette étude comparative, il est certes que nous adoptons UML comme langage de modélisation. Ainsi, la méthodologie de conception adoptée se base sur le choix de diagrammes UML adéquats. Nous avons utilisé quatre diagrammes : diagramme de cas d'utilisation, diagramme d'activités, diagramme de séquence et diagramme de classes. Le schéma suivant représente notre méthodologie de conception.



*Figure 3: Méthodologie de conception*

### ***Conclusion :***

Afin d'identifier les éléments principaux ainsi que les fonctionnalités principales de notre application web, nous avons commencé en premier lieu à faire une description du système et finalement on a présenté et justifié le choix du processus de développement choisie afin de mener notre projet. Dans les chapitres suivants, nous allons entamer le cœur de notre projet qui contient les étapes de conception d'une application web pour gérer les bons de sortie et entrée du matériel informatique.

# **Analyse de besoin, conception et méthodologie de réalisation**



## **I. Analyse des besoins**

Avant de présenter les figures des tâches assignées à chaque type d'utilisateur, nous allons brièvement introduire les quatre types d'utilisateurs que nous avons identifiés dans notre application SecuChat.

Le premier type d'utilisateur est le Super Admin, qui a un accès complet à toutes les fonctionnalités de l'application, y compris la gestion des salons de discussion, des messages et des utilisateurs. En plus de cela, il peut également supprimer des messages d'un utilisateur et voir l'historique des messages d'un utilisateur.

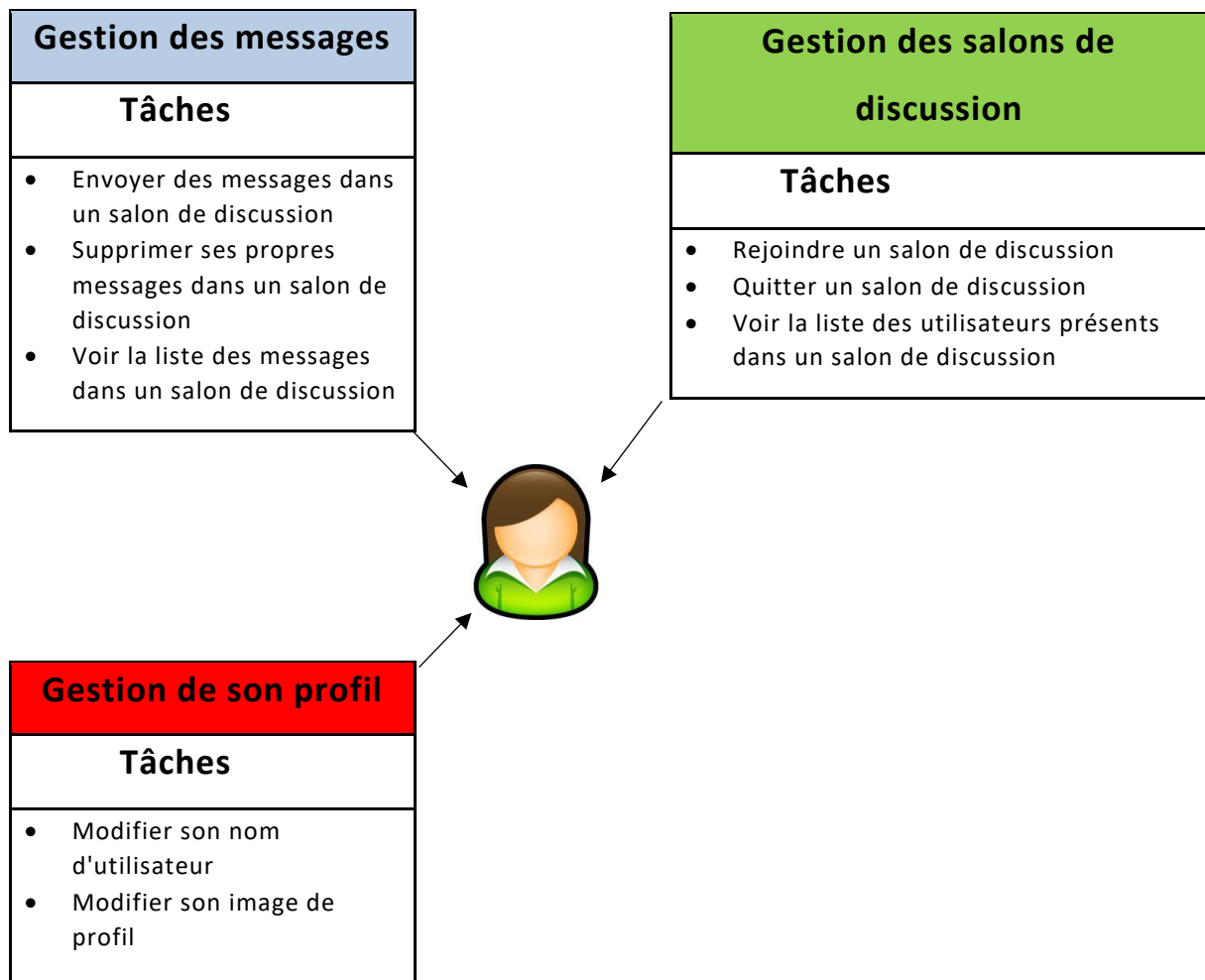
Le deuxième type d'utilisateur est l'Admin, qui peut créer et gérer des salons de discussion, ainsi que supprimer des salons et des messages. Il peut également voir la liste des salons de discussion et des messages dans ces salons.

Le troisième type d'utilisateur est le Modérateur, qui peut seulement supprimer des messages d'un utilisateur dans un salon de discussion et voir la liste des messages dans ce salon.

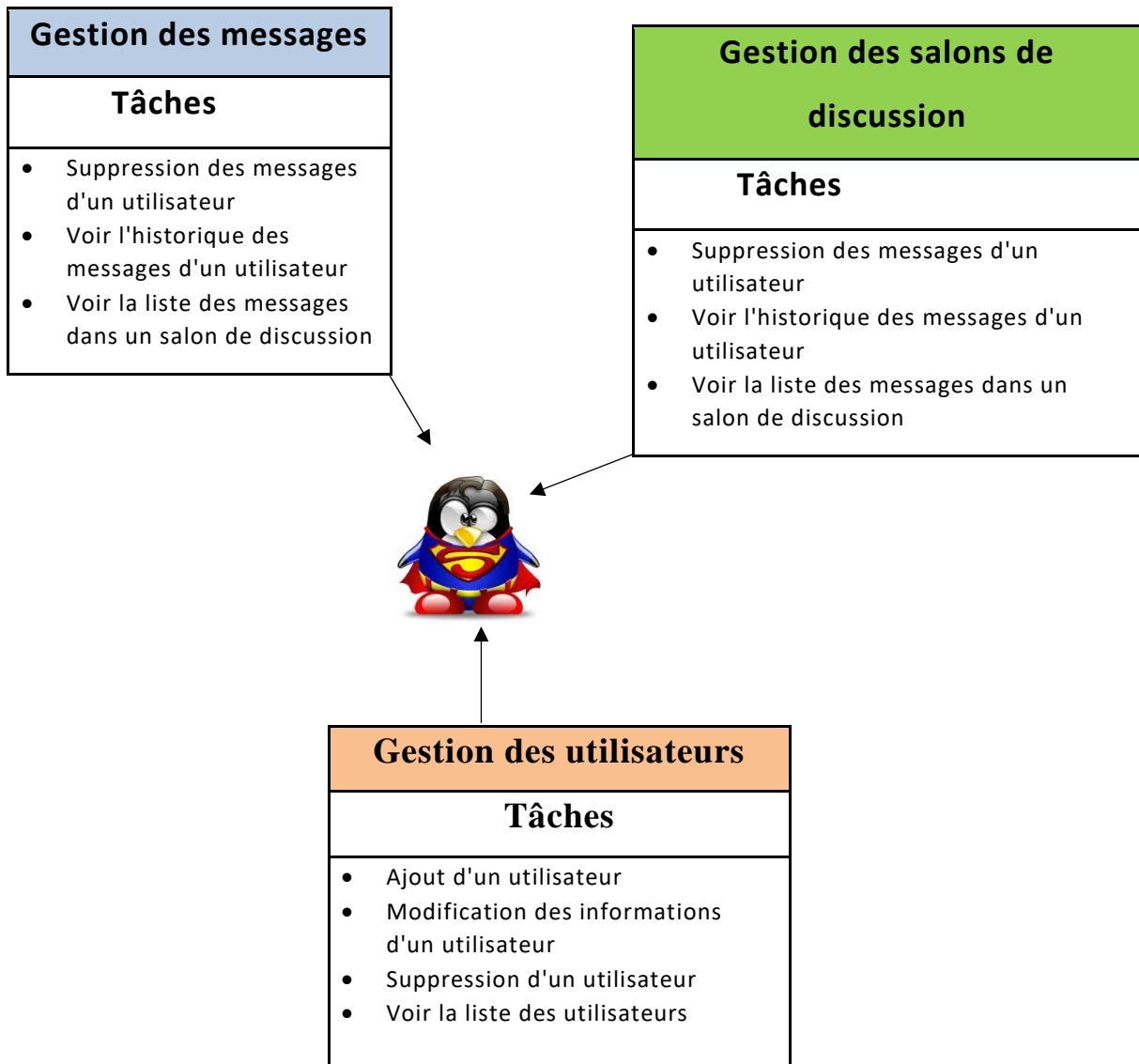
Enfin, le quatrième type d'utilisateur est l'Utilisateur, qui peut rejoindre et quitter des salons de discussion, envoyer des messages et voir la liste des utilisateurs présents dans un salon. L'utilisateur peut également modifier son nom d'utilisateur et son image de profil.

Ces différents types d'utilisateurs ont des niveaux d'accès différents et des tâches qui leur sont assignées en fonction de leur rôle dans l'application SecuChat. Les figures ci-dessous présente les tâches associées à chaque type d'utilisateur.

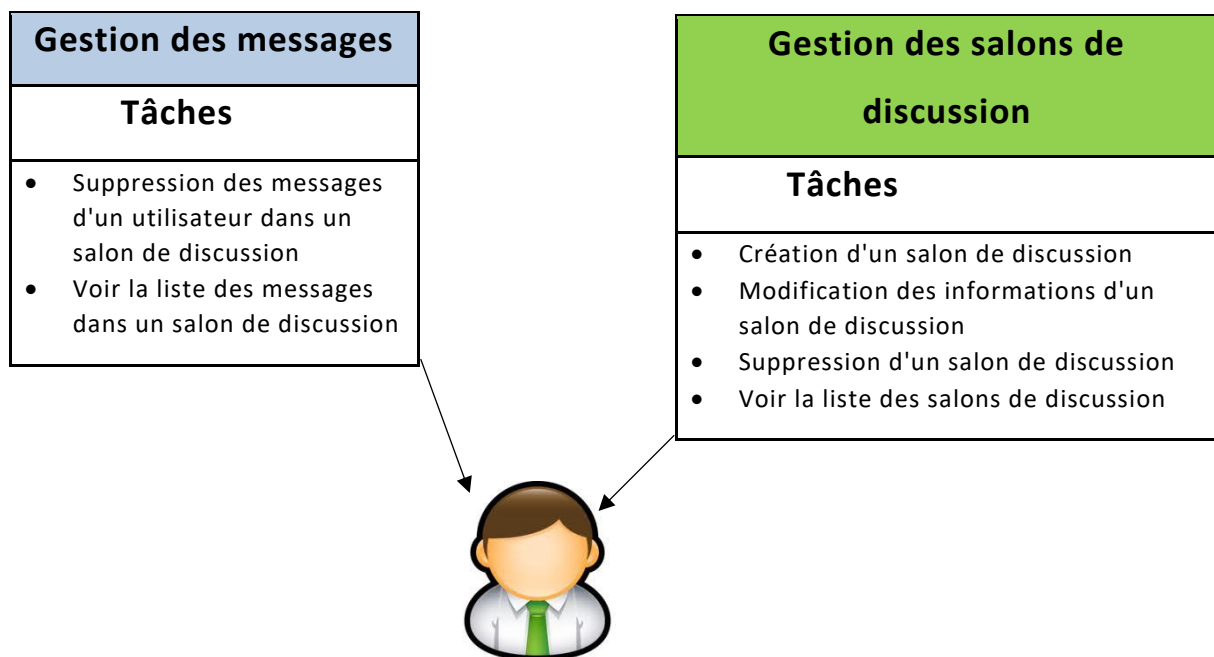
- Utilisateur



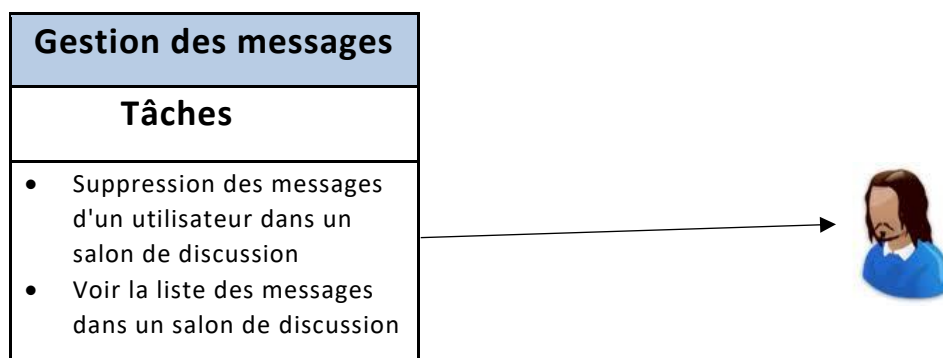
- Super Admin



- **Admin**



- **Modérateur**



- Relation entre les quatre parties

Les quatre types d'utilisateurs dans notre application SecuChat ont des rôles différents mais complémentaires pour assurer le bon fonctionnement de l'application. Le Super Admin a le contrôle total de l'application et peut effectuer toutes les tâches de gestion des salons de discussion, des messages et des utilisateurs. L'Admin est responsable de la création et de la gestion des salons de discussion, tandis que le Modérateur est chargé de la modération des messages. L'Utilisateur final est celui qui utilise l'application pour rejoindre et quitter des salons de discussion, envoyer des messages et gérer son propre profil. Ces quatre types d'utilisateurs sont en relation les uns avec les autres, chaque type étant dépendant de l'autre pour assurer la bonne marche de l'application.

## **II. Etude conceptuelle**

### **Diagramme de cas d'utilisation**

Les diagrammes de cas d'utilisation sont des diagrammes UML utilisés pour donner une vision globale du comportement fonctionnel d'un système logiciel. Ils sont utiles pour des présentations auprès de la direction ou des acteurs d'un projet, mais pour le développement, les cas d'utilisation sont plus appropriés.

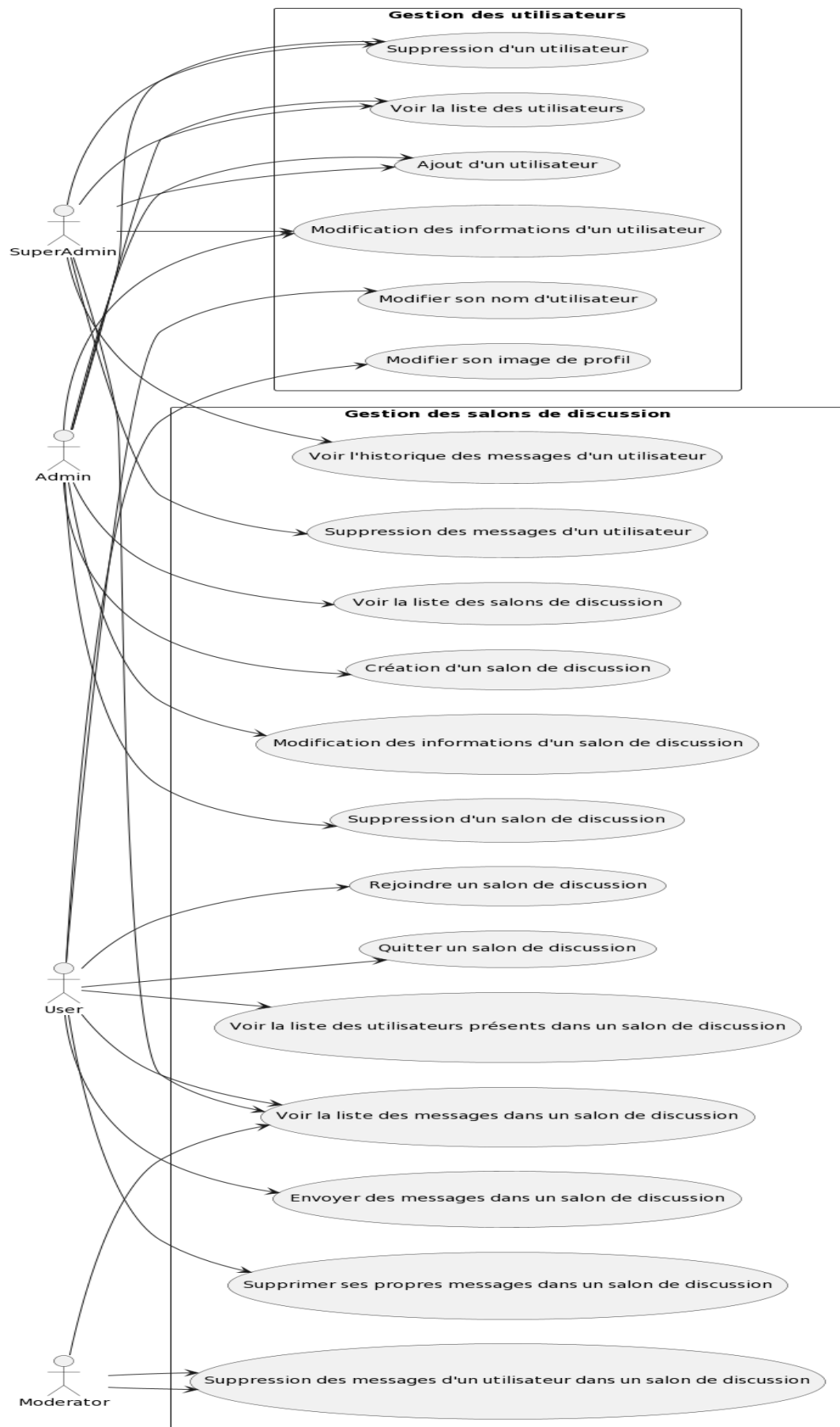


Figure 4 : Diagramme de cas d'utilisation

## Diagramme de classe

Un diagramme de classes dans le langage de modélisation est un diagramme de structure statique qui décrit la structure d'un système classes, leurs attributs, les opérations (ou) les méthodes et les entre les classes.

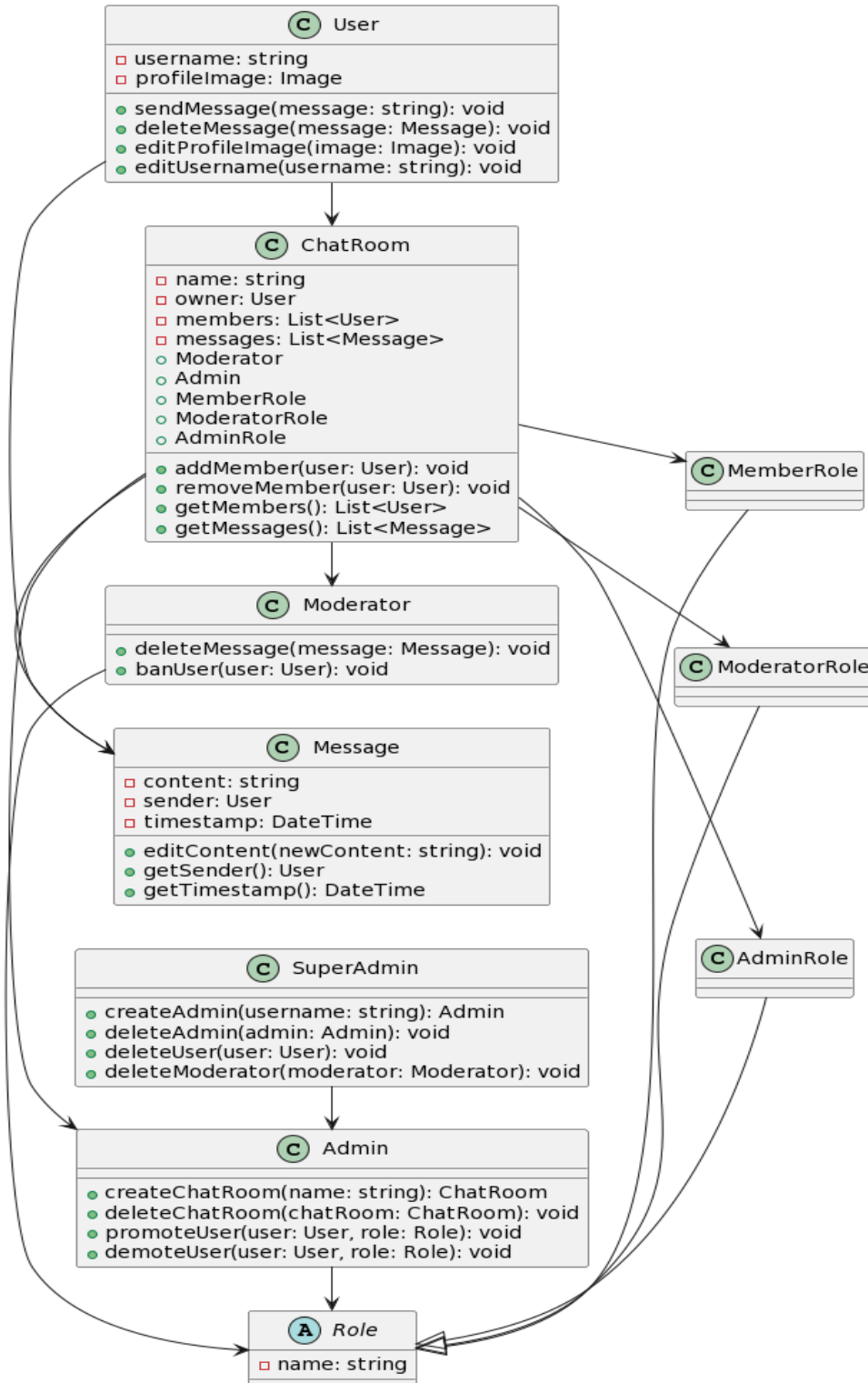
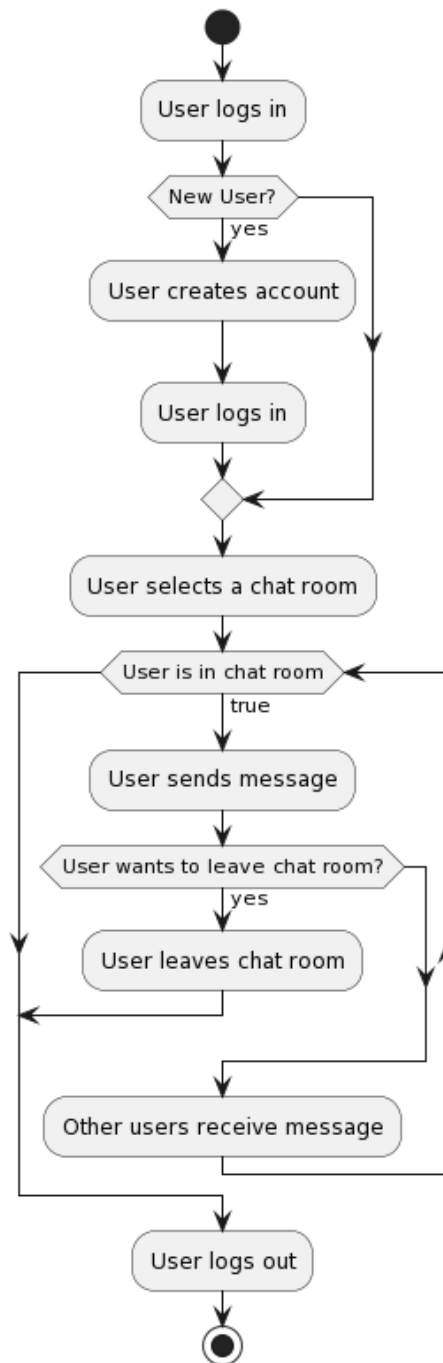


Figure 5 : Diagramme de classe

## Diagramme d'activité

Le diagramme d'activité est un diagramme comportemental d'UML, permettant de représenter le déclenchement d'événements en fonction des états du système et de modéliser des comportements parallélisables. Le diagramme d'activité est également utilisé pour décrire un flux de travail

**Chat Application Activity Diagram**



*Figure 6 : Diagramme d'activité*

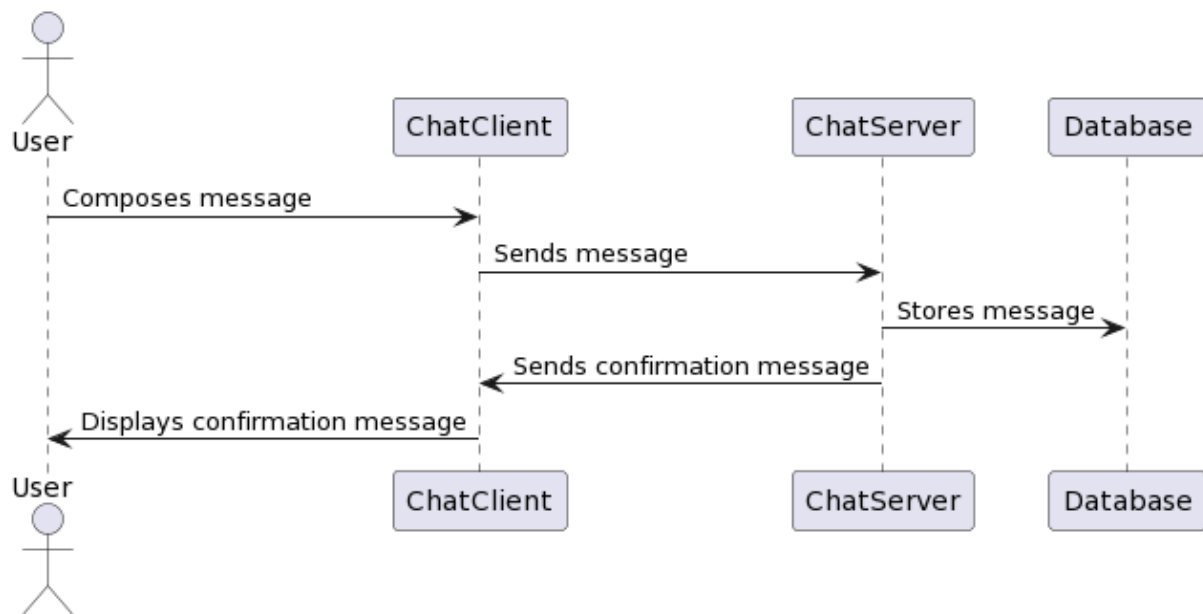


## Diagramme de séquence

Les diagrammes de séquences permettent de représenter graphiquement comment les éléments du système interagissent entre eux et avec les acteurs selon un ordre chronologique dans la formulation de l'envoi des messages.

L'interaction de l'acteur avec le système est représentée par une ligne verticale appelée ligne de vie, les messages s'échangent entre ces lignes de vie.

Ci-dessous les diagrammes de séquence pour les principaux cas d'utilisation soulevés.



*Figure 7 : Diagramme de séquence*

# **Le travail technique**

## I. Installation d'environnement du travail

Pour installer l'environnement de travail pour le développement de SecuChat, il est nécessaire de suivre les étapes suivantes :

**Installer Python :** Django est un framework web qui est basé sur le langage de programmation Python. Il est donc nécessaire d'installer Python pour pouvoir utiliser Django. Il est recommandé d'installer la version Python 3.6 ou supérieure. Pour installer Python, il suffit de télécharger le fichier d'installation à partir du site officiel de Python (<https://www.python.org/downloads/>) et de suivre les instructions d'installation.

**Installer Django :** Une fois que Python est installé, il est temps d'installer Django. Pour cela, vous pouvez utiliser le gestionnaire de packages pip de Python. Ouvrez une invite de commande et tapez la commande suivante :

```
C:\Users\alaes>pip install django
```

**Installer les dépendances :** SecuChat utilise plusieurs bibliothèques tierces pour fonctionner. Pour installer ces bibliothèques, vous pouvez utiliser le fichier requirements.txt fourni avec le code source de SecuChat. Ouvrez une invite de commande et tapez la commande suivante :

```
C:\Users\alaes>pip install -r requirements.txt
```

**Installer SQLite :** SQLite est un système de gestion de base de données relationnelles qui est utilisé par Django pour stocker les données. Il est recommandé d'utiliser SQLite pour le développement de SecuChat car il est facile à installer et à utiliser. Pour installer SQLite, téléchargez le fichier d'installation à partir du site officiel de SQLite (<https://www.sqlite.org/download.html>) et suivez les instructions d'installation.

**Installer Daphne WebSocket :** Daphne est un serveur Web ASGI qui est utilisé pour gérer les connexions WebSocket. Pour installer Daphne, vous pouvez utiliser le

gestionnaire de packages pip de Python. Ouvrez une invite de commande et tapez la commande suivante :

```
C:\Users\alae>pip install daphne_
```

Une fois ces étapes terminées, vous êtes prêt à commencer le développement de SecuChat en utilisant Django et Daphne WebSocket.

### **Structure du projet django**

La structure du projet Django de SecuChat suit la convention de structure de projet standard recommandée par Django. Voici un aperçu de la structure de base du projet :

**manage.py** : fichier exécutable utilisé pour gérer le projet Django.

**requirements.txt** : fichier contenant les dépendances Python requises pour le projet.

**config/** : répertoire contenant les fichiers de configuration du projet.

**static/** : répertoire contenant les fichiers statiques tels que les images, les fichiers CSS et les fichiers JavaScript.

**templates/** : répertoire contenant les fichiers de templates HTML utilisés pour la vue du projet.

**apps/** : répertoire contenant les différentes applications Django créées pour le projet. Chaque application peut avoir son propre modèle de données, ses propres vues et ses propres fichiers statiques.

Dans le répertoire **apps/**, on peut trouver les applications suivantes :

**accounts** : gère l'authentification et l'autorisation des utilisateurs.

**chat** : contient les modèles de données, les vues et les fichiers statiques nécessaires pour implémenter la fonctionnalité de chat en temps réel.

**inventory** : gère la gestion de l'inventaire des matériels.

**orders** : gère la gestion des commandes et des bons de sortie.

Chaque application est organisée selon le modèle MVC (Modèle-Vue-Contrôleur) de Django, avec les modèles de données définis dans les fichiers `models.py`, les vues définies dans les fichiers `views.py` et les fichiers de templates HTML stockés dans le répertoire **templates/**.

**En résumé**, la structure du projet Django de SecuChat suit les meilleures pratiques recommandées par Django et utilise l'architecture MVC pour organiser les différentes applications du projet.

## II. Code SQL de la base de données

La première capture d'écran montre un ensemble de commandes SQL qui permettent de créer trois tables dans une base de données. Pour y parvenir, la commande `PRAGMA foreign keys, off` a été utilisée pour désactiver temporairement les contraintes de clé étrangère, permettant ainsi l'insertion de données dans les tables ultérieurement sans être limité par ces contraintes.

La première table créée est `"auth_group"` avec deux colonnes, `"id"` et `"name"`. La colonne `"id"` est définie comme une clé primaire avec auto-incrémentation, tandis que la colonne `"name"` est définie comme une chaîne de caractères non unique.

La deuxième table créée est `"auth_group_permissions"` avec trois colonnes, `"id"`, `"group_id"` et `"permission_id"`. La colonne `"id"` est définie comme une clé primaire avec auto-incrémentation, tandis que les colonnes `"group_id"` et `"permission_id"` sont définies comme des clés étrangères faisant référence aux colonnes `"id"` des tables `"auth_group"` et `"auth_permission"` respectivement.

La troisième table créée est `"auth_permission"` avec quatre colonnes, `"id"`, `"content_type_id"`, `"codename"` et `"name"`. La colonne `"id"` est définie comme une clé primaire avec auto-incrémentation, tandis que la colonne `"content_type_id"` est définie comme une clé étrangère faisant référence à la colonne `"id"` de la table `"django_content_type"`. Les colonnes `"codename"` et `"name"` contiennent des chaînes de caractères définissant le nom et les permissions de chaque groupe et chaque permission. Ces dernières sont ajoutées à la table `"auth_permission"` à l'aide de commandes `INSERT INTO`. Il est important de noter que la deuxième capture d'écran montre également l'insertion des données dans la table `"auth_permission"`.

Cependant, il est crucial de réactiver les contraintes de clé étrangère une fois la création terminée pour garantir l'intégrité de la base de données.

```

--
-- File generated with SQLiteStudio v3.4.3 on mar. mai 9 22:24:47 2023
--
-- Text encoding used: System
--
PRAGMA foreign_keys = off;
BEGIN TRANSACTION;

-- Table: auth_group
CREATE TABLE IF NOT EXISTS "auth_group" ("id" integer NOT NULL PRIMARY KEY AUTOINCREMENT, "name" varchar(150) NOT NULL UNIQUE);

-- Table: auth_group_permissions
CREATE TABLE IF NOT EXISTS "auth_group_permissions" ("id" integer NOT NULL PRIMARY KEY AUTOINCREMENT, "group_id" integer NOT NULL REFERENCES "auth_group" ("id") DEFERRABLE INITIALLY DEFERRED, "permission_id" integer NOT NULL REFERENCES "auth_permission" ("id") DEFERRABLE INITIALLY DEFERRED);

-- Table: auth_permission
CREATE TABLE IF NOT EXISTS "auth_permission" ("id" integer NOT NULL PRIMARY KEY AUTOINCREMENT, "content_type_id" integer NOT NULL REFERENCES "django_content_type" ("id") DEFERRABLE INITIALLY DEFERRED, "codename" varchar(100) NOT NULL, "name" varchar(255) NOT NULL);
INSERT INTO auth_permission (id, content_type_id, codename, name) VALUES (1, 1, 'add_logentry', 'Can add log entry');
INSERT INTO auth_permission (id, content_type_id, codename, name) VALUES (2, 1, 'change_logentry', 'Can change log entry');
INSERT INTO auth_permission (id, content_type_id, codename, name) VALUES (3, 1, 'delete_logentry', 'Can delete log entry');
INSERT INTO auth_permission (id, content_type_id, codename, name) VALUES (4, 1, 'view_logentry', 'Can view log entry');
INSERT INTO auth_permission (id, content_type_id, codename, name) VALUES (5, 2, 'add_permission', 'Can add permission');
INSERT INTO auth_permission (id, content_type_id, codename, name) VALUES (6, 2, 'change_permission', 'Can change permission');
INSERT INTO auth_permission (id, content_type_id, codename, name) VALUES (7, 2, 'delete_permission', 'Can delete permission');
INSERT INTO auth_permission (id, content_type_id, codename, name) VALUES (8, 2, 'view_permission', 'Can view permission');
INSERT INTO auth_permission (id, content_type_id, codename, name) VALUES (9, 3, 'add_group', 'Can add group');
INSERT INTO auth_permission (id, content_type_id, codename, name) VALUES (10, 3, 'change_group', 'Can change group');
INSERT INTO auth_permission (id, content_type_id, codename, name) VALUES (11, 3, 'delete_group', 'Can delete group');
INSERT INTO auth_permission (id, content_type_id, codename, name) VALUES (12, 3, 'view_group', 'Can view group');

```

```

INSERT INTO auth_permission (id, content_type_id, codename, name) VALUES (13, 4, 'add_user', 'Can add user');
INSERT INTO auth_permission (id, content_type_id, codename, name) VALUES (14, 4, 'change_user', 'Can change user');
INSERT INTO auth_permission (id, content_type_id, codename, name) VALUES (15, 4, 'delete_user', 'Can delete user');
INSERT INTO auth_permission (id, content_type_id, codename, name) VALUES (16, 4, 'view_user', 'Can view user');
INSERT INTO auth_permission (id, content_type_id, codename, name) VALUES (17, 5, 'add_contenttype', 'Can add content type');
INSERT INTO auth_permission (id, content_type_id, codename, name) VALUES (18, 5, 'change_contenttype', 'Can change content type');
INSERT INTO auth_permission (id, content_type_id, codename, name) VALUES (19, 5, 'delete_contenttype', 'Can delete content type');
INSERT INTO auth_permission (id, content_type_id, codename, name) VALUES (20, 5, 'view_contenttype', 'Can view content type');
INSERT INTO auth_permission (id, content_type_id, codename, name) VALUES (21, 6, 'add_session', 'Can add session');
INSERT INTO auth_permission (id, content_type_id, codename, name) VALUES (22, 6, 'change_session', 'Can change session');
INSERT INTO auth_permission (id, content_type_id, codename, name) VALUES (23, 6, 'delete_session', 'Can delete session');
INSERT INTO auth_permission (id, content_type_id, codename, name) VALUES (24, 6, 'view_session', 'Can view session');
INSERT INTO auth_permission (id, content_type_id, codename, name) VALUES (25, 7, 'add_site', 'Can add site');
INSERT INTO auth_permission (id, content_type_id, codename, name) VALUES (26, 7, 'change_site', 'Can change site');
INSERT INTO auth_permission (id, content_type_id, codename, name) VALUES (27, 7, 'delete_site', 'Can delete site');
INSERT INTO auth_permission (id, content_type_id, codename, name) VALUES (28, 7, 'view_site', 'Can view site');
INSERT INTO auth_permission (id, content_type_id, codename, name) VALUES (29, 8, 'add_message', 'Can add message');
INSERT INTO auth_permission (id, content_type_id, codename, name) VALUES (30, 8, 'change_message', 'Can change message');
INSERT INTO auth_permission (id, content_type_id, codename, name) VALUES (31, 8, 'delete_message', 'Can delete message');
INSERT INTO auth_permission (id, content_type_id, codename, name) VALUES (32, 8, 'view_message', 'Can view message');
INSERT INTO auth_permission (id, content_type_id, codename, name) VALUES (33, 9, 'add_userprofile', 'Can add user profile');
INSERT INTO auth_permission (id, content_type_id, codename, name) VALUES (34, 9, 'change_userprofile', 'Can change user profile');
INSERT INTO auth_permission (id, content_type_id, codename, name) VALUES (35, 9, 'delete_userprofile', 'Can delete user profile');
INSERT INTO auth_permission (id, content_type_id, codename, name) VALUES (36, 9, 'view_userprofile', 'Can view user profile');
INSERT INTO auth_permission (id, content_type_id, codename, name) VALUES (37, 10, 'add_chat', 'Can add chat');
INSERT INTO auth_permission (id, content_type_id, codename, name) VALUES (38, 10, 'change_chat', 'Can change chat');
INSERT INTO auth_permission (id, content_type_id, codename, name) VALUES (39, 10, 'delete_chat', 'Can delete chat');
INSERT INTO auth_permission (id, content_type_id, codename, name) VALUES (40, 10, 'view_chat', 'Can view chat');

```

Le troisième screenshot montre un extrait de code SQL pour la création de plusieurs tables dans une base de données. Les tables incluent "auth\_user" pour stocker les informations des utilisateurs, "auth\_user\_groups" pour stocker les groupes auxquels les utilisateurs appartiennent, "auth\_user\_user\_permissions" pour stocker les permissions des utilisateurs, "chat\_chat" pour stocker les informations sur les chats, "chat\_chat\_users" pour associer les utilisateurs aux chats, "chat\_message" pour stocker les messages envoyés dans les chats, "chat\_userprofile" pour stocker les informations sur les profils des utilisateurs, "django\_admin\_log" pour stocker les logs des actions des administrateurs, "django\_content\_type" pour stocker les informations sur les types de contenu, "django\_migrations" pour stocker les informations sur les migrations de base de données, "django\_session" pour stocker les sessions des utilisateurs et "django\_site" pour stocker les informations sur les sites web.

```
-- Table: auth_user
CREATE TABLE IF NOT EXISTS "auth_user" ("id" integer NOT NULL PRIMARY KEY AUTOINCREMENT, "password" varchar(128) NOT NULL, "last_login" datetime NULL, "is_superuser" bool NOT NULL, "username" varchar(150) NOT NULL UNIQUE, "last_name" varchar(150) NOT NULL, "email" varchar(254) NOT NULL, "is_staff" bool NOT NULL, "is_active" bool NOT NULL, "date_joined" datetime NOT NULL, "first_name" varchar(150) NOT NULL);

-- Table: auth_user_groups
CREATE TABLE IF NOT EXISTS "auth_user_groups" ("id" integer NOT NULL PRIMARY KEY AUTOINCREMENT, "user_id" integer NOT NULL REFERENCES "auth_user" ("id") DEFERRABLE INITIALLY DEFERRED, "group_id" integer NOT NULL REFERENCES "auth_group" ("id") DEFERRABLE INITIALLY DEFERRED);

-- Table: auth_user_user_permissions
CREATE TABLE IF NOT EXISTS "auth_user_user_permissions" ("id" integer NOT NULL PRIMARY KEY AUTOINCREMENT, "user_id" integer NOT NULL REFERENCES "auth_user" ("id") DEFERRABLE INITIALLY DEFERRED, "permission_id" integer NOT NULL REFERENCES "auth_permission" ("id") DEFERRABLE INITIALLY DEFERRED);

-- Table: chat_chat
CREATE TABLE IF NOT EXISTS "chat_chat" ("id" integer NOT NULL PRIMARY KEY AUTOINCREMENT, "name" varchar(128) NOT NULL);

-- Table: chat_chat_users
CREATE TABLE IF NOT EXISTS "chat_chat_users" ("id" integer NOT NULL PRIMARY KEY AUTOINCREMENT, "chat_id" bigint NOT NULL REFERENCES "chat_chat" ("id") DEFERRABLE INITIALLY DEFERRED, "user_id" integer NOT NULL REFERENCES "auth_user" ("id") DEFERRABLE INITIALLY DEFERRED);

-- Table: chat_message
CREATE TABLE IF NOT EXISTS "chat_message" ("id" integer NOT NULL PRIMARY KEY AUTOINCREMENT, "content" text NOT NULL, "createTime" datetime NOT NULL, "chat_id" bigint NOT NULL REFERENCES "chat_chat" ("id") DEFERRABLE INITIALLY DEFERRED, "user_id" integer NOT NULL REFERENCES "auth_user" ("id") DEFERRABLE INITIALLY DEFERRED);

-- Table: chat_userprofile
CREATE TABLE IF NOT EXISTS "chat_userprofile" ("id" integer NOT NULL PRIMARY KEY AUTOINCREMENT, "avatar" varchar(100) NULL, "user_id" integer NOT NULL UNIQUE REFERENCES "auth_user" ("id") DEFERRABLE INITIALLY DEFERRED);

-- Table: django_admin_log
CREATE TABLE IF NOT EXISTS "django_admin_log" ("id" integer NOT NULL PRIMARY KEY AUTOINCREMENT, "object_id" text NULL, "object_repr" varchar(200) NOT NULL, "action_flag" smallint unsigned NOT NULL CHECK ("action_flag" >= 0), "change_message" text NOT NULL, "content_type_id" integer NULL REFERENCES "django_content_type" ("id") DEFERRABLE INITIALLY DEFERRED, "user_id" integer NOT NULL REFERENCES "auth_user" ("id") DEFERRABLE INITIALLY DEFERRED, "action_time" datetime NOT NULL);

-- Table: django_content_type
CREATE TABLE IF NOT EXISTS "django_content_type" ("id" integer NOT NULL PRIMARY KEY AUTOINCREMENT, "app_label" varchar(100) NOT NULL, "model" varchar(100) NOT NULL);

-- Table: django_migrations
CREATE TABLE IF NOT EXISTS "django_migrations" ("id" integer NOT NULL PRIMARY KEY AUTOINCREMENT, "app" varchar(255) NOT NULL, "name" varchar(255) NOT NULL, "applied" datetime NOT NULL);

-- Table: django_session
CREATE TABLE IF NOT EXISTS "django_session" ("session_key" varchar(40) NOT NULL PRIMARY KEY, "session_data" text NOT NULL, "expire_date" datetime NOT NULL);

-- Table: django_site
CREATE TABLE IF NOT EXISTS "django_site" ("id" integer NOT NULL PRIMARY KEY AUTOINCREMENT, "name" varchar(50) NOT NULL, "domain" varchar(100) NOT NULL UNIQUE);
```

La quatrième capture d'écran montre une série de requêtes SQL pour créer des index sur des tables de la base de données. Les index sont utilisés pour accélérer les recherches dans une base de données en créant des structures de données qui permettent de localiser rapidement les informations stockées dans une table. Les requêtes commencent par l'instruction "CREATE INDEX IF NOT EXISTS", qui crée l'index uniquement s'il n'existe pas déjà. Les tables incluent "auth\_group\_permissions", "auth\_permission", "auth\_user\_groups", "auth\_user\_user\_permissions", "chat\_chat\_users", "chat\_message", "django\_admin\_log", "django\_content\_type" et "django\_session". Les noms des index sont également spécifiés, tels que "auth\_group\_permissions\_group\_id\_b120cbf9" ou "auth\_permission\_content\_type\_id\_codename\_01ab375a\_uniq". Enfin, la requête se termine par une instruction "COMMIT TRANSACTION", qui confirme les modifications apportées à la base de données et active les contraintes de clé étrangère avec l'instruction "PRAGMA foreign\_keys = on".



```

-- Index: auth_group_permissions_group_id_b120cbf9
CREATE INDEX IF NOT EXISTS "auth_group_permissions_group_id_b120cbf9" ON "auth_group_permissions" ("group_id");

-- Index: auth_group_permissions_group_id_permission_id_0cd325b0_uniq
CREATE UNIQUE INDEX IF NOT EXISTS "auth_group_permissions_group_id_permission_id_0cd325b0_uniq" ON
"auth_group_permissions" ("group_id", "permission_id");

-- Index: auth_group_permissions_permission_id_84c5c92e
CREATE INDEX IF NOT EXISTS "auth_group_permissions_permission_id_84c5c92e" ON "auth_group_permissions"
("permission_id");

-- Index: auth_permission_content_type_id_2f476e4b
CREATE INDEX IF NOT EXISTS "auth_permission_content_type_id_2f476e4b" ON "auth_permission" ("content_type_id");

-- Index: auth_permission_content_type_id_codename_01ab375a_uniq
CREATE UNIQUE INDEX IF NOT EXISTS "auth_permission_content_type_id_codename_01ab375a_uniq" ON "auth_permission"
("content_type_id", "codename");

-- Index: auth_user_groups_group_id_97559544
CREATE INDEX IF NOT EXISTS "auth_user_groups_group_id_97559544" ON "auth_user_groups" ("group_id");

-- Index: auth_user_groups_user_id_6a12ed8b
CREATE INDEX IF NOT EXISTS "auth_user_groups_user_id_6a12ed8b" ON "auth_user_groups" ("user_id");

-- Index: auth_user_groups_user_id_group_id_94350c0c_uniq
CREATE UNIQUE INDEX IF NOT EXISTS "auth_user_groups_user_id_group_id_94350c0c_uniq" ON "auth_user_groups"
("user_id", "group_id");

-- Index: auth_user_user_permissions_permission_id_1fbb5f2c
CREATE INDEX IF NOT EXISTS "auth_user_user_permissions_permission_id_1fbb5f2c" ON "auth_user_user_permissions"
("permission_id");

-- Index: auth_user_user_permissions_user_id_a95ead1b
CREATE INDEX IF NOT EXISTS "auth_user_user_permissions_user_id_a95ead1b" ON "auth_user_user_permissions"
("user_id");

-- Index: auth_user_user_permissions_user_id_permission_id_14a6b632_uniq
CREATE UNIQUE INDEX IF NOT EXISTS "auth_user_user_permissions_user_id_permission_id_14a6b632_uniq" ON
"auth_user_user_permissions" ("user_id", "permission_id");

-- Index: chat_chat_users_chat_id_d52bf378
CREATE INDEX IF NOT EXISTS "chat_chat_users_chat_id_d52bf378" ON "chat_chat_users" ("chat_id");

-- Index: chat_chat_users_chat_id_user_id_dfd3d335_uniq
CREATE UNIQUE INDEX IF NOT EXISTS "chat_chat_users_chat_id_user_id_dfd3d335_uniq" ON "chat_chat_users" ("chat_id",
"user_id");

-- Index: chat_chat_users_user_id_4126ebf1
CREATE INDEX IF NOT EXISTS "chat_chat_users_user_id_4126ebf1" ON "chat_chat_users" ("user_id");

-- Index: chat_message_chat_id_21483fa7
CREATE INDEX IF NOT EXISTS "chat_message_chat_id_21483fa7" ON "chat_message" ("chat_id");

-- Index: chat_message_user_id_a47c01bb
CREATE INDEX IF NOT EXISTS "chat_message_user_id_a47c01bb" ON "chat_message" ("user_id");

-- Index: django_admin_log_content_type_id_c4bce8eb
CREATE INDEX IF NOT EXISTS "django_admin_log_content_type_id_c4bce8eb" ON "django_admin_log" ("content_type_id");

-- Index: django_admin_log_user_id_c564eba6
CREATE INDEX IF NOT EXISTS "django_admin_log_user_id_c564eba6" ON "django_admin_log" ("user_id");

-- Index: django_content_type_app_label_model_76bd3d3b_uniq
CREATE UNIQUE INDEX IF NOT EXISTS "django_content_type_app_label_model_76bd3d3b_uniq" ON "django_content_type"
("app_label", "model");

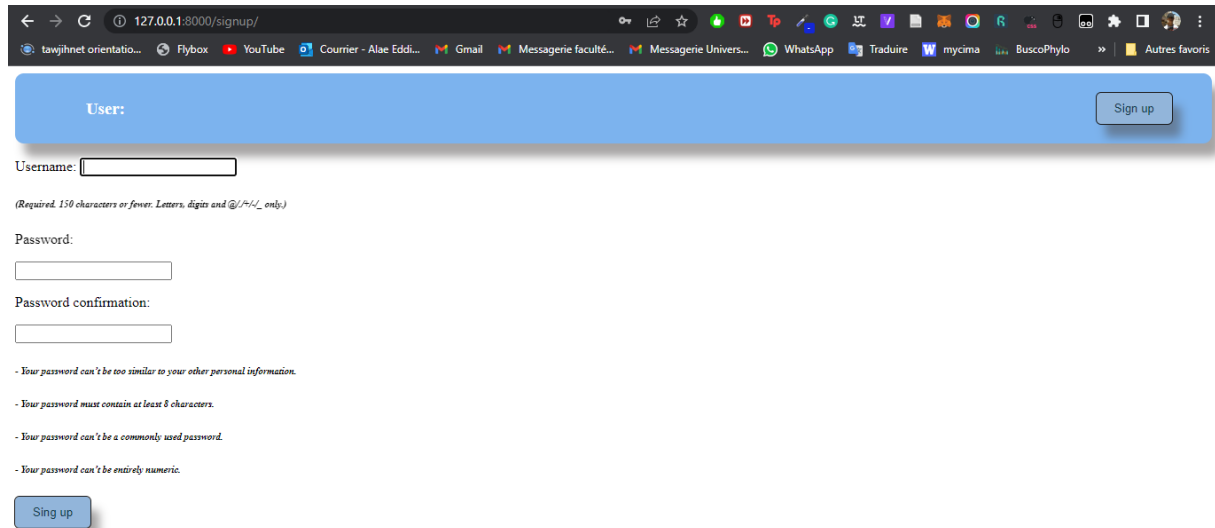
-- Index: django_session_expire_date_a5c62663
CREATE INDEX IF NOT EXISTS "django_session_expire_date_a5c62663" ON "django_session" ("expire_date");

COMMIT TRANSACTION;
PRAGMA foreign_keys = on;

```

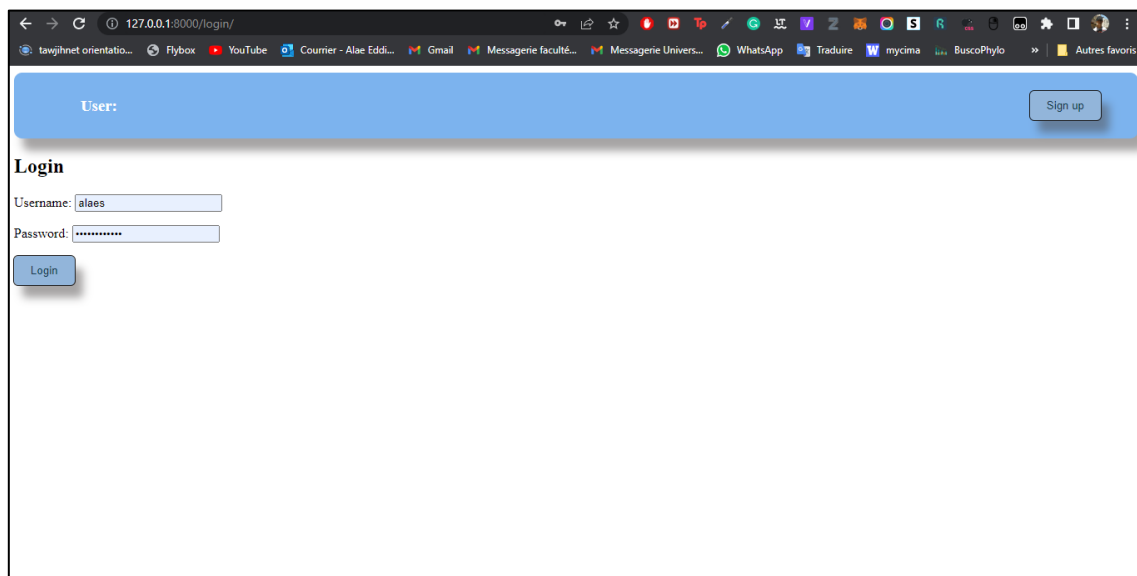
### III. Application Web (quelques captures d'écrans)

La capture d'écran de la page d'inscription comporte trois champs de saisie : un pour le nom d'utilisateur et les deux autres pour le mot de passe et la confirmation du mot de passe. Le mot de passe doit contenir au moins huit caractères, dont une lettre majuscule, un symbole et un chiffre.



The screenshot shows a web browser at the URL 127.0.0.1:8000/signup/. The page has a blue header with the text "User:" and a "Sign up" button. Below the header, there are three input fields: "Username:", "Password:", and "Password confirmation:". The "Password:" field has a small note below it: "(Required: 150 characters or fewer. Letters, digits and @/./#/-/\_ only.)". Below the "Password confirmation:" field, there are four lines of feedback text: "Your password can't be too similar to your other personal information.", "Your password must contain at least 8 characters.", "Your password can't be a commonly used password.", and "Your password can't be entirely numeric.". At the bottom, there is a "Sing up" button.

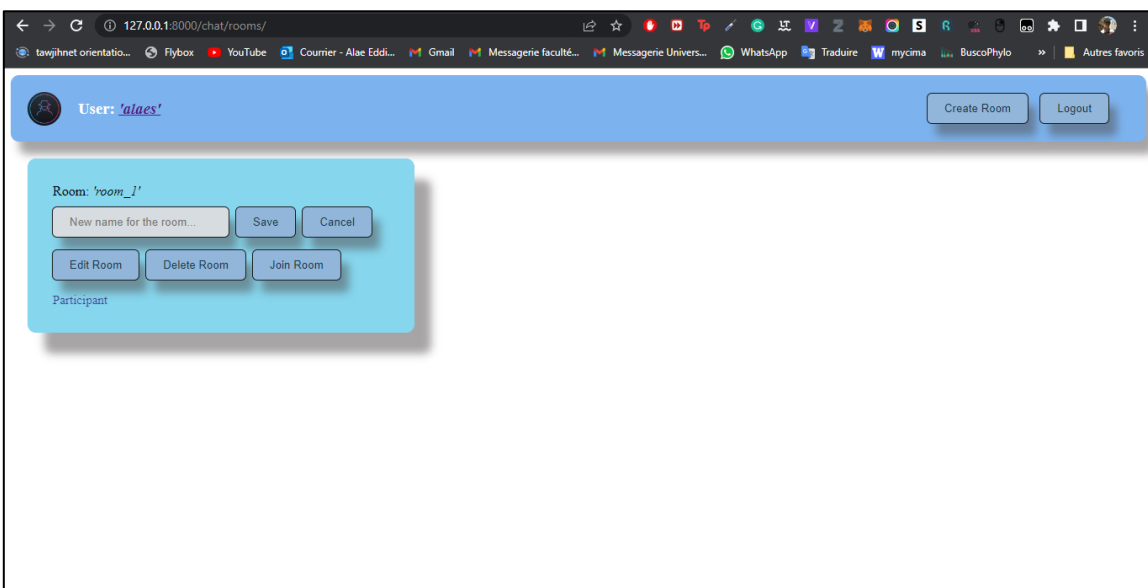
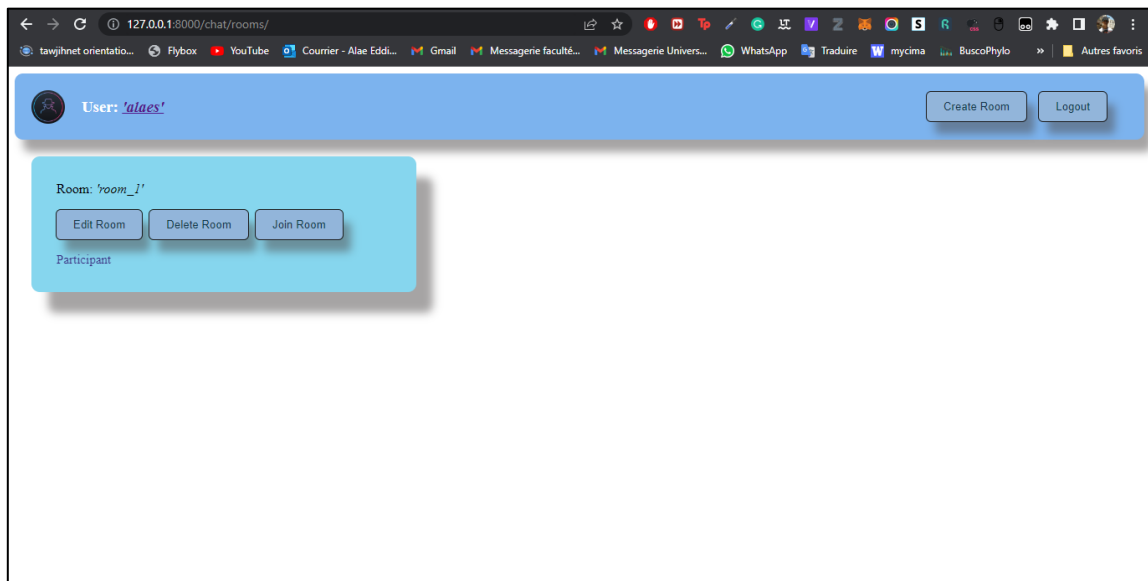
La capture d'écran de la page de connexion présente deux champs de saisie pour le nom d'utilisateur et le mot de passe, ainsi qu'un bouton de connexion et un bouton d'inscription dans le coin de la barre de navigation.



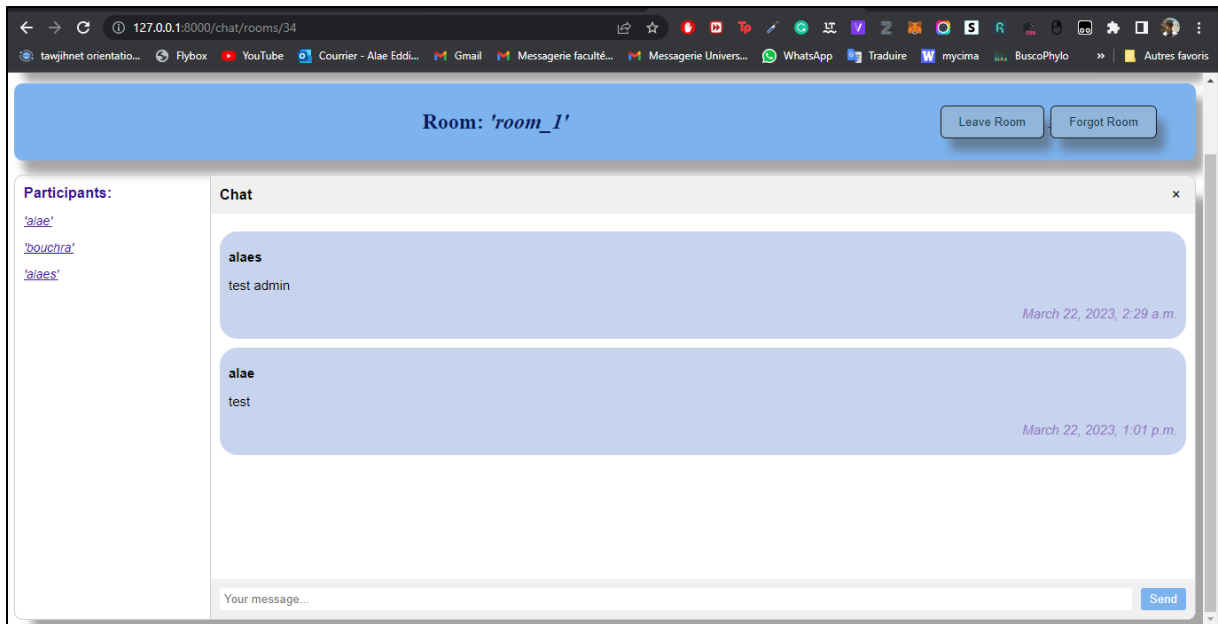
The screenshot shows a web browser at the URL 127.0.0.1:8000/login/. The page has a blue header with the text "User:" and a "Sign up" button. Below the header, there is a "Login" section with two input fields: "Username:" (containing the text "alaes") and "Password:" (containing masked characters). Below the "Password:" field, there is a "Login" button.

Sur la page après la connexion, le nom d'utilisateur de l'utilisateur connecté est affiché dans le coin supérieur gauche de l'écran. Dans le coin supérieur droit de la page, il y a un bouton de déconnexion ainsi qu'un bouton pour créer une nouvelle salle. Dans la

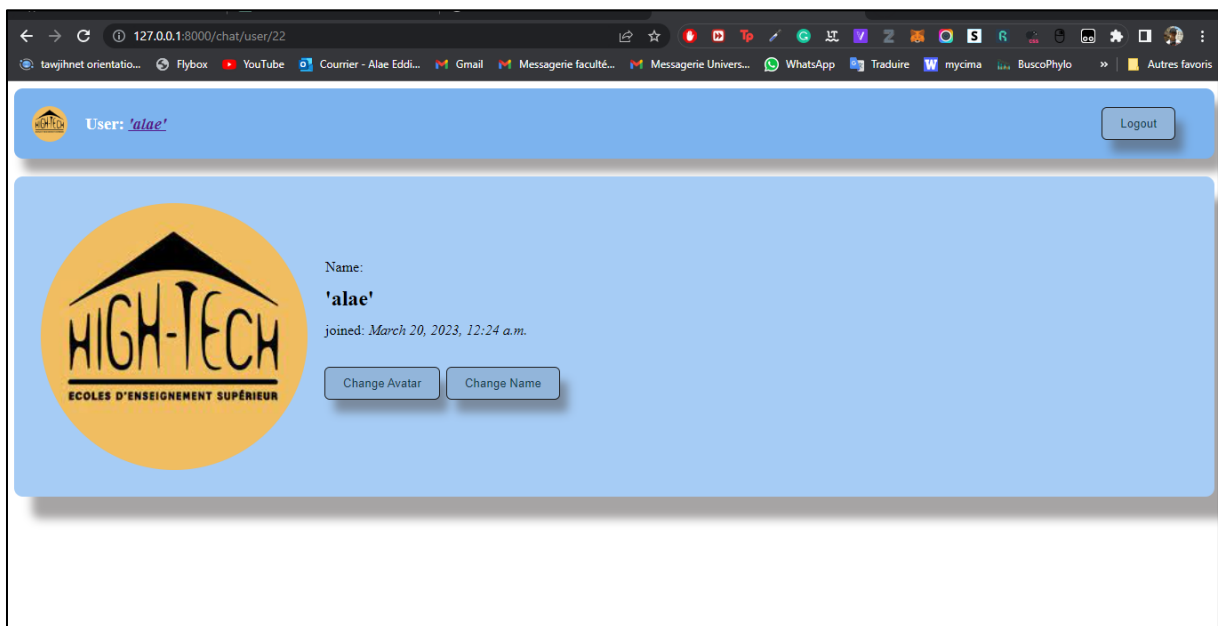
partie centrale de la page, les différentes salles auxquelles l'utilisateur peut participer sont affichées. Si l'utilisateur est propriétaire d'une salle, il aura la possibilité de la modifier ou de la supprimer.



Sur la page de chat, une fois que l'utilisateur a rejoint la salle, la liste des utilisateurs est affichée à gauche, tandis que le chat est affiché à droite. Dans la partie chat, il y a une zone de texte pour entrer et soumettre les messages, tandis que les autres participants peuvent voir les messages en temps réel avec l'heure de leur envoi.



Dans la section "profil", l'utilisateur peut voir son avatar, son nom et la date et l'heure de la création de son profil. Il peut également modifier son nom et son avatar à partir d'un bouton dédié.



## Conclusion

Basé sur l'ensemble de nos discussions, il est clair que la mise en place d'une application de chat sécurisée nécessite une planification minutieuse et l'utilisation d'outils et de technologies appropriés. Nous avons opté pour le framework web Django pour la création de SecuChat en raison de ses nombreuses fonctionnalités avancées, de sa sécurité, de sa robustesse et de sa scalabilité. Nous avons également choisi d'utiliser le chiffrement AES pour garantir la sécurité des données des utilisateurs.

Le choix de SQLite comme système de gestion de base de données a été décidé pour sa légèreté et sa facilité d'utilisation. Pour les connexions en temps réel, nous avons décidé d'utiliser Daphne Websocket, qui permet des connexions bidirectionnelles en temps réel pour les clients Web.

En termes de perspectives pour l'application, il y a plusieurs fonctionnalités que nous pourrions ajouter à l'avenir pour améliorer l'expérience utilisateur. Cela inclut l'amélioration du design et l'ajout d'options pour envoyer des images, des emojis et des stickers. Nous pourrions également ajouter la possibilité de passer des appels vidéo et audio. Une fonctionnalité intéressante pourrait être la possibilité de supprimer les messages après leur envoi.

En fin de compte, nous avons réussi à créer une application de chat sécurisée et efficace qui répond aux exigences élevées en matière de sécurité et de fonctionnalité. Les connaissances et l'expérience acquises dans la création de SecuChat peuvent être utilisées pour d'autres projets qui nécessitent des fonctionnalités similaires et une sécurité élevée.

.

## **Conclusion and Future Improvements**

Based on our discussions, it is clear that developing a secure chat application requires careful planning and the use of appropriate tools and technologies. We chose the Django web framework for creating SecuChat due to its advanced features, security, robustness, and scalability. We also opted to use AES encryption to ensure the security of user data.

The choice of SQLite as the database management system was made due to its lightweight and user-friendliness. For real-time connections, we decided to use Daphne Websocket, which allows for bidirectional real-time connections for web clients.

In terms of future improvements for the application, there are several features that we could add to enhance the user experience. This includes improving the design and adding options for sending images, emojis, and stickers. We could also add the ability to make video and audio calls. An interesting feature could be the ability to delete messages after sending.

Ultimately, we have succeeded in creating a secure and effective chat application that meets high requirements for security and functionality. The knowledge and experience gained in creating SecuChat can be used for other projects that require similar features and high security.