REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITE IBN KHALDOUN - TIARET

MEMOIRE

Présenté à :

FACULTÉ DES MATHEMATIQUES ET D'INFORMATIQUE
DÉPARTEMENT D'INFORMATIQUE

Pour l'obtention du diplôme de :

MASTER

Spécialité : [Génie Logiciel]

Par :
BENYAGOUB Oussama

BENMOUHOUB Belkacem Mohamed Salah

Sur le thème

Conception et réalisation d'une application chat multi-clients avec WebSocket.

Soutenu publiquement le 12 / 06 / 2025 à Tiaret devant le jury composé de :

Mr MOKHTARI Ahmed	MAA	Université de Tiaret	Président
Mr DAHMANI Youcef	Professeur	Université de Tiaret	Encadrant
Mme BENATHMANE Lalia	MAA	Université de Tiaret	Examinatrice

2024-2025

Remerciements

Nos remerciements vont particulièrement à notre encadrant qui nous a suggéré le sujet de recherche, pour son aide, et ses conseils, Le professeur ***DAHMANI Youcef***

Nos remerciements s'adressent également à l'ensemble des enseignants du département d'Informatique pour la qualité de l'enseignement dispensé au cours de notre cursus universitaire

Nous tenons de également de remercier les chers enseignants qui ont accepté d'examiner notre travail et de faire partie du jury pour le temps qu'ils nous ont accordé ainsi que pour leurs remarques constructives.

Dédicaces

Je dédie ce modeste travail,

*À mes **chers parents** pour tout leur amour, soutien, et leurs prières tout au long de mes études.*

*À **ma sœur et à mon frère**, pour leur motivation, leur confiance et leur soutien.*

*Aux **membres de ma famille** qui ont constamment été présents pour moi, m'apportant un soutien moral, spirituel et émotionnel inestimable*

*À la mémoire de **mes chers grands-parents** que dieu leur accorde sa miséricorde, à jamais dans mon cœur.*

*À **mes amis** , pour les moments partagés , leur aide et leur soutien.*

*Et particulièrement à **mon binôme et ami Oussama** avec qui j'ai eu le plaisir de partager tout mon cursus universitaire*

BENMOUHOUB Belkacem

Dédicaces

Avec merveilleux sentiments je dédie ce travail :
À mes très ***chers parents*** *pour tout leur amour, soutien, et leurs prières tout au long de mes études.*

À ceux qui comptent le plus dans ma vie, ***mes frères et mes sœurs***

À ***mes chers amis et camarades****, pour leur présence et les bons souvenirs partagés tout au long de ce parcours.*

À ***mon binôme et cher ami Belkacem*** *avec qui j'ai partagé des moments merveilleux au long de mon parcours universitaire*

BENYAGOUB Oussama

Sommaire

Résumé :

Objectif :

Face à l'absence de développeurs Algériens spécialisés dans le développement de réseaux sociaux et d'applications de communication, nous avons décider de créer et développer une application de chat en temps réel destinée aux utilisateurs.

Cette application a pour objectif principale protéger les utilisateurs Algériens des impacts négatifs des autres plateformes, tout en offrant un espace sécurisé qui garantir la liberté d'expression et facilite les échanges d'une manière respectueuse et responsable.

Méthodologies :

L'objectif de ce projet est de concevoir et de développer une application de chat en temps réel en s'appuyant sur les technologies modernes, notamment **WebSocket** et **Nodejs**. Cette application sera multiplateforme, accessible via un navigateur web, et permettra aux utilisateurs de communiquer instantanément dans des salles de discussion dédiées, que ce soit depuis un PC ou un appareil mobile.

Résultats clés :

En résumé, ce projet vise à répondre à un besoin essentiel dans le domaine de la communication en ligne, tout en promouvant le savoir-faire technologies algérien.

Grâce a l'utilisation de technologies modernes telles que **Websocket** et **Nodejs**, l'application offrira une solution efficace, sécurisée et accessible via un navigateur web, répondant ainsi aux attentes des utilisateurs sur des plateformes différentes.

Ce projet représente une opportunité unique pour mettre en pratique les compétences acquises tout au long du cycle de master en génie logiciel, tout en contribuant à la création d'un environnement numérique respectueux et adapté aux besoins des utilisateurs Algériens.

Mots clés :

Chat, WebSocket, Application web, Communication en temps réel, Multi-clients, Messagerie instantanée, Node.js, React,

Sommaire

Liste des figures

Liste des figures

Liste des tableaux

Introduction générale

Introduction générale :

Dans un contexte où les relations social en Algérie sont en pleine expansion et où la communication en temps réel devient essentielle. Cependant, le manque d'une application de messagerie locale digne de confiance suscite des inquiétudes concernant la confidentialité et l'accessibilité. Cela nous a poussés à concevoir un réseau social local pour le public Algérien

Ce projet a pour objectif de développer une plateforme de communication local qui permet l'envoi et la réception de données en temps réel d'une manière sécurisée et adaptée aux besoins des utilisateurs , il vise à garantir la facilité d'utilisation en mettant en œuvre une interface claire et fluide , la confidentialité des données en renforçant la confiance des utilisateurs en garantissant la protection de leurs échanges et l'échange rapide des messages entre les utilisateurs. . Parmi les fonctionnalités que vont être incluses dans l'application seront l'inscription l'authentification, la gestion des profils, l'échange des messages entre amis et en groupe.

Techniquement , l'application va être accessible via un navigateur web et probablement hébergée dans un serveur local. Enfin , quelques fonctionnalités ne seront pas présentes dans cette version initiale comme les appels vocaux et vidéos , partage des fichiers volumineux (seuls les images et document légers seront pris en charge). Développement mobile (l'application ne sera pas disponible sur les stores mobiles pour cette version).

Le **premier chapitre** constitue un **état de l'art** , dans lequel nous représentons les solutions existantes et les technologies utilisées , ainsi qu'un aperçu sur le protocole WebSocket.

Le deuxième chapitre est dédié à l'**analyse fonctionnelle et à la conception du projet ainsi que l'architecture choisie**, en identifiant les besoins des utilisateurs et en représentant les différentes interactions à l'aide de diagrammes UML.

Le troisième chapitre décrit la **réalisation de l'application**, en détaillant la création de la base de données, l'interface utilisateur, le serveur WebSocket, et l'implémentation des

principales fonctionnalités telles que l'authentification, la gestion des profils et la messagerie.

Ce mémoire vise ainsi à présenter l'ensemble du processus de conception, de modélisation et de développement de l'application.

Chapitre 01 : Etat de l’Art

1. Introduction :

L'objectif de cette étude est d'analyser les solutions existantes dans le domaine des applications de chat en temps réel ainsi que leurs forces, leurs faiblesse et les améliorations possibles pour créer une solution innovante adaptée aux besoins et aux aspirations des utilisateurs.

2. Étude des solutions existantes :

2.1. Application populaires :

Le tableau ci-dessus présente une comparaison entre trois applications de messagerie largement utilisées : WhatsApp, Telegram et Viber. Chaque application est analysée selon ses fonctionnalités principales, ses points forts et ses limitations.

Application	**Fonctionnalités**	**Points forts**	**Points faibles**
WhatsApp	Messages, Appels (audio, Video). Chiffremment de bout en bout Création de communautés (groupes).	communication mondiale gratuite. (Somanathan, 2024) Accès multi-plateformes (Somanathan, 2024)	Les fonctionnalités de sécurité insuffisantes. (Somanathan, 2024) Interdiction arbitraire de comptes. (Somanathan, 2024)
Telegram	Messages, Appels, canaux et bots.	Tous vos conversations sont cryptées. (downloadsource) Groupes jusqu'a 200 000 membres. (downloadsource)	Consommation d'internet élevée. (Capterra) Restrictions fréquentes. (Capterra)
Viber	Messages, Appels (audio, video) et stickers.	Créer vos propres stickers. Partagé des stickers et des GIF. démmarer des discussion (Simon, 2023)	Moins populaire que ces concurrents. (Simon, 2023) Interface utilisateur encombrée. (Simon, 2023)

***Tableau 1:** Applications populaires de chat*

2.2. Technologies utilisées :

Le développement d'une application de messagerie repose sur 2 parties le **frontend** et le **backend**, le frontend est ce que l'utilisateur voit sur l'interface, c'est-à-dire l'interface utilisateur qu'il utilise pour envoyer et recevoir les différents événements. Concernant le backend, c'est la partie invisible de l'application qui gère la logique du système qui comprend le stockage des données des utilisateurs, l'envoi et la réception de messages et la notification en temps réel, ces deux parties se connectent entre elles pour assurer un bon fonctionnement de l'application. Nous pouvons voir ci-dessous les technologies utilisées dans les applications comme **WhatsApp**, **Telegram** et **Viber**.

- **WhatsApp :**

Une application mobile multiplateforme qui fournit un système de messagerie instantanée chiffrée de bout en bout, elle offre la possibilité d'envoyer des messages ainsi que de passer des appels vocaux et vidéo.

a. Front-end :

✓ **Android** : Java.

✓ **iOS** : Swift.

✓ **SiteWeb** : Html, Css, Js.

✓ **PC-Desktop app** : C, C#, Java.

b. Back-end :

✓ **Earlang** : Pour son back-end, whatsapp déploie une multitude de technologies, dont le pricipale langage de programmation est Earlang, un langage de programmation fonctionnel. Earlang l'un des éléments qui permettent à whatsApp de fonctionner à si grande échelle.

✓ **Os des serveurs FreeBSD :** Le système d'exploitation sur lequel fonctionnent les serveurs whatsapp est FreeBSD, qui est un système d'exploitation gratuit et open source basé sur UNIX.

✓ **VM BEAM :** utilisée par whatsapp, est une machine virtuelle qui compile et éxécute le code source Earlang. BEAM est spécialement conçue pour les applications hautement concurrentes. Elle est hautement évolutive et résistante aux pannes pouvant être causées par un trafic important et les mise a jour système.

✓ **Base de donnes Mnesia :** est un système de gestion de base de données distribué (SGBD) écrit en Earlang. Il permet de stocker des données pertinentes et des messages temporaires.

✓ **YAWS :** est également un serveur web basé sur Earlang qui prend en charge le contenu dynamique. Whatsapp l'utilisée pour stocker les données multimédias. (; opengenus)

- ## Telegram :

Telegram est une application de messagerie mobile et de bureau basée sur le cloud qui met l'accent sur la sécurité et la rapidité.

a. Front-end :

✓ **Android** : Java et Kotlin

✓ **iOS** : Swift et Objective-C.

✓ **Web** : VueJS, React.

✓ **Desktop** : les applications desktop sont construites à l'aide d'une combinaison de technologies, notamment Electron.

b. Back-end :

✓ **Langage principal :** c/c++ , ces langages sont connus pour leurs performances et sont parfaitement adaptés à la création d'applications à haut début.

✓ **Protocole de communication MTProto (Telegram protocole) :** personnalisé de telegram sécurise la communication de bout en bout, **Websocket** prend en charge

les connexions persistantes et bidirectionnelles pour les interactions pour en temps réel.

✓ **Base de donnes relationnelle PostgreSQL :** Stocké des données structurées telles que des profiles utilisateurs et des métadonnées.

✓ **Base de donnes NoSQL Cassandra ou MongoDB :** gère les données non structurées, y compris les grandes ensembles de données tels que les journaux de discussion et les historiques de messages. (Komilov, 2024)

- **Viber :**

Viber est une application multiplateforme de messagerie instantanée pour smartphone, tablette et ordinateur qui permet d'échanger par Internet des messages texte et des fichiers, mais également des appels audio et vidéo.

a. Front-End :

✓ **Android** : kotlin.

✓ **iOS** : Swift et Objective-C.

✓ **Web** : JavaScript, html, css. (quora)

b. Back-End :

✓ **Langage principal :** le cœur de viber est développé en C++, ce qui offre des performances élevées et une gestion de la mémoire des bas niveau, essentielle pour la communication en temps réel.

✓ **Base de données :** Viber utilise MongoDB comme base de donnes principal qui fournit des emplacements pour stocker les données et sécuriser le chat au sein de l'application et améliorer la plage de performances.

✓ **Protocole de communication SIP :** Viber utilise session initiation protocol comme technologie principal pour passer des appels téléphoniques à n'importe qui sur internet via viber.

✓ **Nginx :** En tant qu'équilibreur de charge et serveur web, viber est implémenté avec serveur HTTP hautes performances pour configurer les performances et stabilité du serveur. (johnvincentt27, 2019)

3. Limites des solutions actuelles :

Malgré la popularité et la diffusion de ces applications, elles peuvent présenter certains inconvénients et limites, notamment dans le contexte Algérien :

- Contrôle des plateformes, restrictions sur certain sujets sensibles et Manque de liberté d'expression , des plateformes comme **Facebook** et **Instagram** sont souvent critiquées pour la modération du contenu et suppression arbitraire des contenus , ce qui limite la liberté d'expression .ces applications sont développées majoritairement par des pays occidentales ce qui leur permet d'imposer leurs propres règles sur le contenu autorisé, par exemple plusieurs utilisateurs se sont plaints de la suppression et du blocage de contenus liés à la Palestine. C'est pourquoi nous avons besoin des solutions alternatives qui garantissent une véritable liberté d'expression.

- Problèmes de confidentialité des données, Plusieurs plateformes telles que **Whatsapp**, **Messenger** et **Instagram** collectent des données personnelles (numéros de téléphones, contacts, messages, localisation, etc.) pour améliorer leurs services. Aussi certains entreprises comme **Meta** (Whatsapp, Messenger, Instagram), partage ces données avec des tiers, en outre certains gouvernements exigent ces plateformes de fournir l'accès aux données personnelles des citoyens, cela menace et réduit la liberté d'expression.

4. Protocole de Communication WebSocket :

4.1. Définition :

Le protocole webSocket fournit un moyen d'échanger des données entre le navigateur et le serveur via une connexion persistante.

WebSocket permet d'ouvrir une session de communication interactive bidirectionnelle entre le navigateur de l'utilisateur et le serveur. Grâce à webSocket vous pouvez envoyer des messages à un serveur et recevoir des réponses sans a avoir à interroger le serveur source spécifiée. (Mozila)

4.2. Avantages :

l'utilisation de websocket apporte de nombreux avantages, on peut citer :

- Réduire la latence, vue que websocket maintient une connexion persistante entre le client et le serveur, cela éliminé la nécessité de rétablir la connexion à chaque interaction contrairement au HTTP. (AppMaster, 2023)

- Le protocole websocket est pris en charge et fonctionne sur la plupart des navigateurs web modernes ce qui facilite son intégration. (AppMaster, 2023)

- Considéré comme une solution idéal pour les applications qui nécessitent des mise à jour instantanées (chat, jeux en lignes etc.)

4.3. Points faibles de WebSocket :

Malgré les nombreux avantages des WebSockets , a des points faibles aussi , on peut citer :

- contrairement à la plupart des navigateurs modernes, certains anciens navigateurs ne prennent pas en charge les WebSockets, ce qui peut affecter le fonctionnement de votre application et sa portée.

- en cas de perte de connexion, aucun mécanisme d'équilibrage de charge ou de reconnexion n'est inclus.

- en raison de la connexion persistante, il est nécessaire de prendre des mesures de sécurité, il faut implémenter des chiffrements tel que SSL/TLS pour sécuriser la connexion WebSocket (wss://) , cela peut protéger votre app contre les vulnérabilités potentielles.

- Au contraire de HTTP qui est stateless ça veut dire il ne conserve pas l'état du client entre deux requêtes, WebSocket est stateful ce qui signifie qu'il conserve l'état du client entre deux requêtes, ce qui entraîne une utilisation accrue de la mémoire et des problèmes potentiels d'évolutivité et de consommation des ressources .

(team, 2023)

4.4. Utilisation :

Le protocole WebSocket est utilisé principalement dans les domaines qui nécessitent des mises à jour en temps réel , comme :

- Chat en direct, en raison de sa capacité à fournir une communication bidirectionnelle à faible latence et en duplex intégral entre le client et le serveur, le technologie websocket est fréquemment utilisé pour offrir des expériences de chat en direct.

- Collaboration en temps réel, travailler avec d'autres personnes dans un environnement en ligne. Figma par exemple utilisé comme un outil qui permet de collaborer avec d'autres personnes et de voir leurs modifications en temps réel.

- Suivi de localisation en temps réel, websocket est utilisé pour assurer une mise à jour instantanée des positions GPS des appareils.

- Plusieurs applications utilisent websocket pour gérer la communication et l'échange de données et temps réel entre le client et le serveur telles que :

- **Discord,** les websockets permettent à discord d'implémenter des fonctionnalités telles que la messagerie de chat en temps réel, la présence (statut en ligne/hors ligne), les mises à jour d'activité et les notifications.

- **WhatsApp web** utilise websocket pour la messagerie de chat en temps réel et pour fournir d'autres fonctionnalités, telles que des mises à jour en temps réel pour la livraison des messages et les accusées de lecture, ainsi que des indicateurs qui s'affichent lorsque quelqu'un tape un message.

- **Youtube** la fonctionnalité de streaming utilisé les websocket pour permettre le chat en temps réel entre les spectateurs et les streamers. De même, le système de notifications de youtube utilise les websockets pour envoyer des notifications en temps réel aux utilisateurs.

- **Uber** utilise websockets pour envoyer les mises à jours de locations en temps réel aux clients. (Diaconu, 2023)

4.5. Fonctionnement :

- Premièrement le client établie un **Handshake** avec le serveur via le protocole HTTP
- Si le serveur accepté la connexion, il répond avec le code de statut 101 qui indique que le serveur à validé la connexion
- Apres la réponse du serveur, le client et le serveur peuvent échanger des messages de différent formats (html, json, texte etc.). via une connexion persistante.

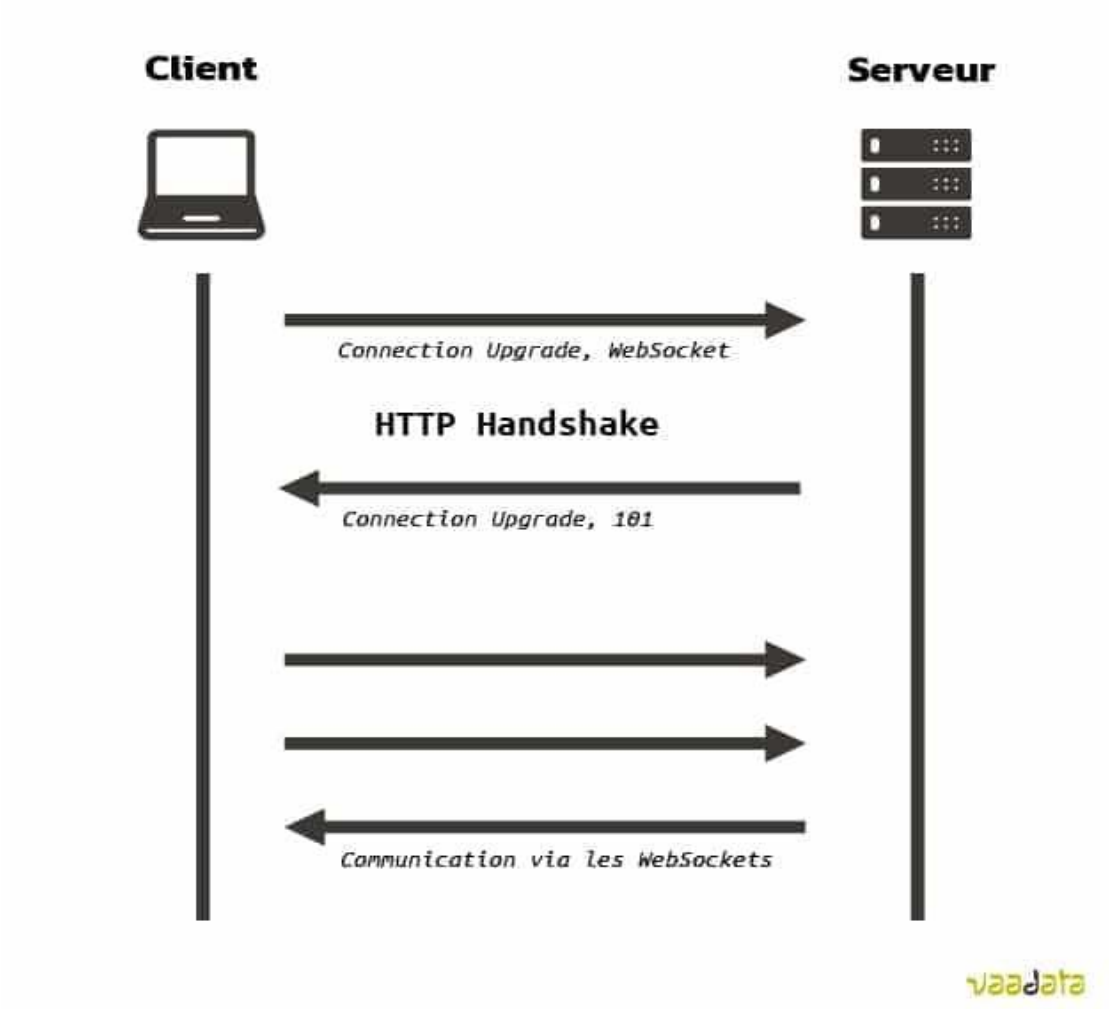

Figure 1: *Protocole de communication WebSocket* **(*vaadata, 2025*)**

5. Conclusion :

ce chapitre nous a permis de mieux analyser et comprendre les technologies utilisées dans les applications de messagerie modernes telles que Whatsapp, Telegram et Viber ainsi que leurs fonctionnalités et leurs limites, de plus, l'étude du protocole Websocket nous a aidé à comprendre son utilisation et les domaines dans lesquels nous l'utilisons ainsi que ses avantages et ses inconvénients ; cette étude représente donc une technique solide qui nous guide pour mettre en œuvre et concevoir une application de messagerie locale.

Chapitre 02 : Conception du projet

1. Introduction :

après avoir analysé les solutions existantes dans le chapitre précédent, ce chapitre présente la conception du projet, qui définit les besoins et les fonctionnalités que l'application couvrira. dans ce chapitre, nous discuterons de l'architecture du projet le Business Model Canvas ainsi que des diagrammes UML.

L'objectif est de définir une base solide avant de commencer la phase d'implémentation.

2. Expression des besoins :

2.1. Définition :

L'expression des besoins consiste à recueillir et à documenter d'une manière détaillée les attentes, les exigences et les fonctionnalités souhaitées par les parties prenantes et les utilisateurs d'un projet.

Les parties prenantes sont l'ensemble des individus concernés par le projet , dans le cadre du projet de l'application de messagerie , les parties prenantes sont :

- ✓ **Les utilisateurs finaux** : les utilisateurs qui vont utiliser l'application pour communiquer entre eux .
- ✓ **L'équipe de développement** : l'équipe qui est responsable de développer et implémenter l'application.

Les besoins fonctionnels expriment comment est le système d'un point de vue utilisateur.

Un besoin technique exprime comment est le système d'un point de vue interne (technique, technologies).

2.2. Besoins fonctionnels :

a. Utilisateur :

- ✓ Créer un compte.

- ✓ Authentifier
- ✓ Envoyer des messages.
- ✓ Envoyer des invitations.
- ✓ Créer des groupes.
- ✓ Consulter les conversations.
- ✓ Bloquer d'autres utilisateurs.
- ✓ Modifier son compte.

b. Système :

- ✓ Authentification des utilisateurs
- ✓ Envoi et réception des messages en temps réel.
- ✓ Envoyer des notifications.
- ✓ Afficher les listes des amis.
- ✓ Afficher les utilisateurs connectés.

c. Administrateur :

- ✓ Gérer les utilisateurs et les messages.

2.3. Diagramme de cas d'utilisation :

Un diagramme de cas d'utilisation représente un diagramme UML (Unified Modeling Language) qui illustre les interactions entre les acteurs externes (comme les utilisateurs et systèmes) et les cas d'utilisation spécifiques d'un système. Il offre une perspective utilisateur sur les principales caractéristiques du système et les interactions entre ces divers éléments. (Booch, 2005)

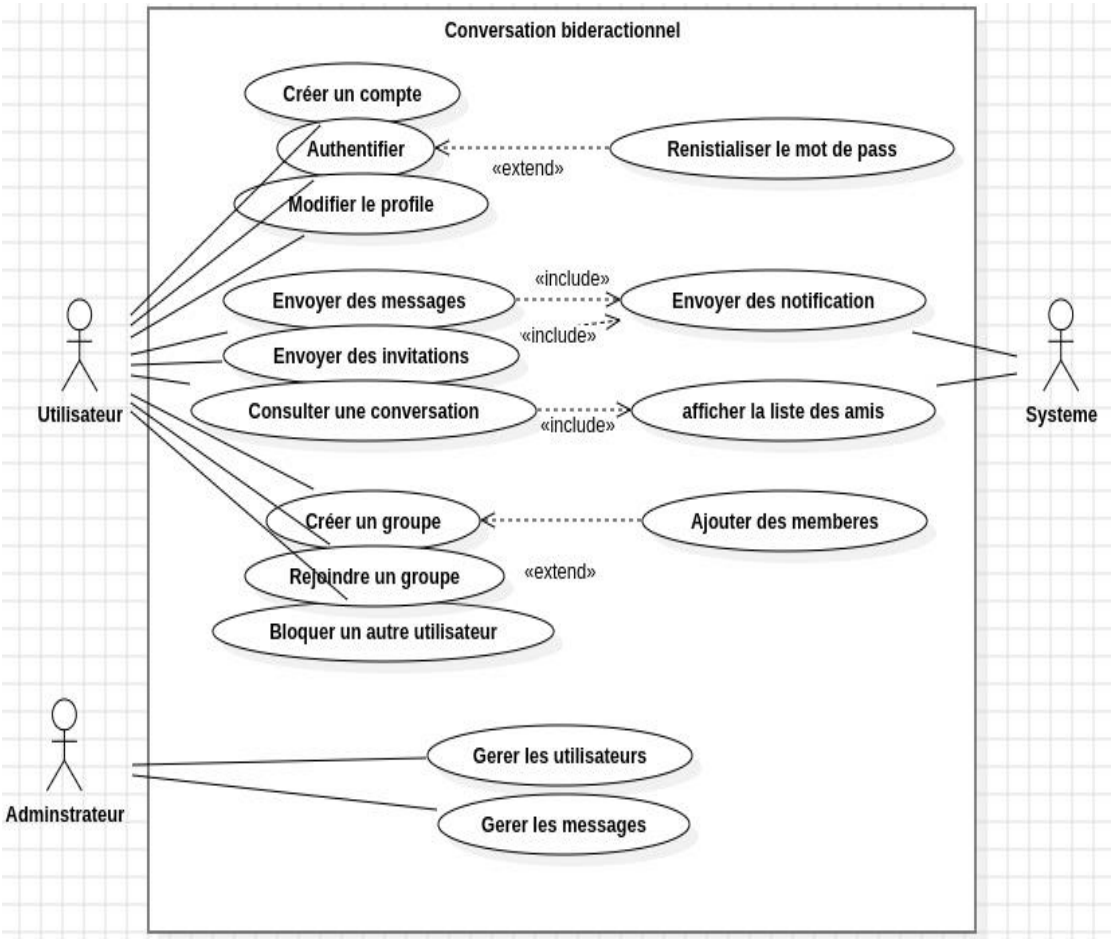

Figure 2: *Diagramme de cas d'utilisation d'une application client serveur de chat*

3. Analyse des exigences :

3.1. Technique d'analyse grammaticale :

Cette technique permet d'analyser les informations collectées dans l'étape précédente (cas d'utilisation, spécification textuelle…)

L'analyse grammaticale consiste à analyser le texte pour trouver les classes, les attributs et les opérations.

a. Les noms :

- ✓ Utilisateur
- ✓ Système
- ✓ Administrateur
- ✓ Compte
- ✓ Message
- ✓ Invitation
- ✓ Conversation
- ✓ Groupe
- ✓ Privé
- ✓ Notification
- ✓ Amis

b. Les verbes :

- ✓ Créer
- ✓ Authentifié
- ✓ Modifier
- ✓ Envoyer
- ✓ Consulter
- ✓ Rejoindre
- ✓ Bloquer
- ✓ Gérer
- ✓ Réinitialiser
- ✓ Afficher
- ✓ Ajouter

3.2. Technique des stéréotypes :

Cette technique est complémentaire avec la technique d'analyse grammaticale et elle se concentre sur trois stéréotypes :

- **Boundary :** représente les limites du système , ces classes communiquent avec les acteurs.

- **Entity :** représente les concepts manipulés par le système.
- **Control :** est responsable sur la coordination des actions qui permettent d'accomplir un cas d'utilisation.

Acteurs	**Entités**	**Contrôleurs**	**Boundary**
Utilisateur	Compte Compte Message Invitation Groupe	Créer Authentifier Envoyer Envoyer Créer	InterfaceInscreption InterfaceAuthentification Conversation BarreDeRecherche InterfaceCréation
Système	Notification Amis	Envoyer Afficher	Liste Liste
Administrateur	Utilisateur Message	Gérer Gérer	InterfaceDeGestion InterfaceDeGestion

***Table 2**: Différents stéréotypes de l'application*

3.3. Diagramme de classe :

- **Définition :**

Le diagramme de classe est un modèle qui représente un ensemble d'objets qui partagent les mêmes attributs, méthodes et relations. Il est l'un des plus utiles en génie logiciel, car il décrit clairement la structure du système. (Larman, 2004)

- **Eléments principaux :**

✓ **Classe :** représentée par des rectangles contenant le nom, les attributs et les méthodes.

✓ **Attributs :** définissent les propriétés des objets de la classe.

✓ **Méthodes :** Définissent les comportements des objets.

✓ **Relations :** incluent l'héritage, l'association, l'agrégation et la composition.

C'est un diagramme fondamentale en **Modélisation Orientée Objet,** Notamment en **conception logicielle** et en développement **UML**.

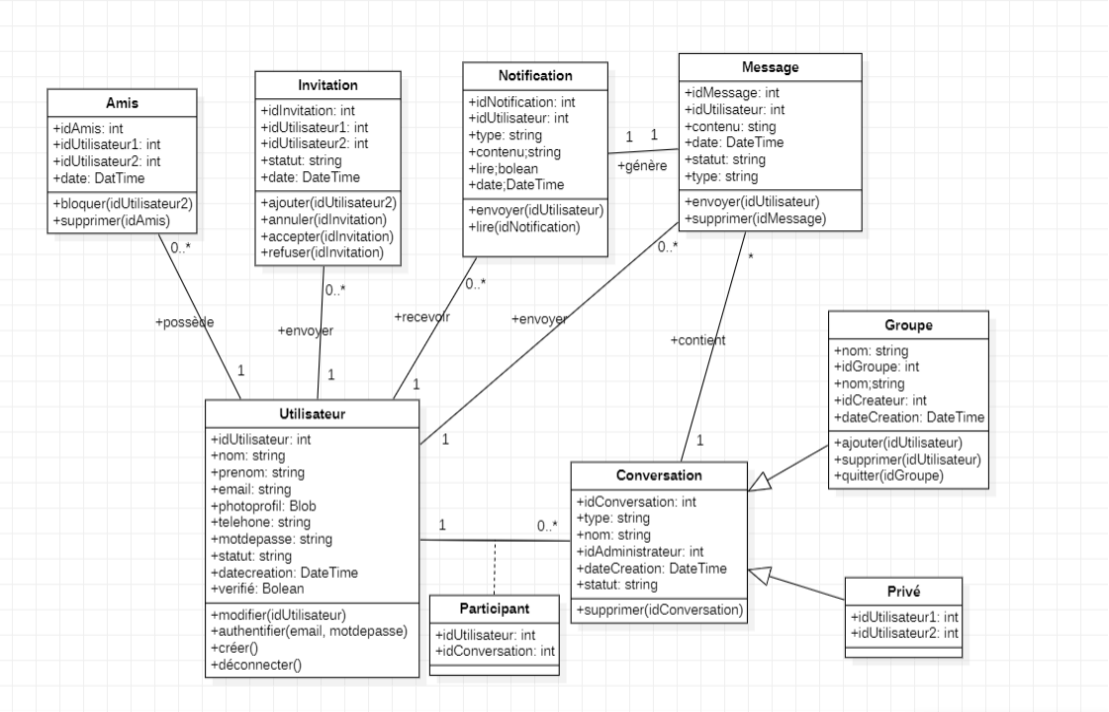

Figure 3: *Diagramme de classe d'une application de chat*

4. Comportement et interactions :

Les interactions expriment comment les instances de ces classes interagissent pour réalise une fonction du système.

- ✓ Trouver quelles classes interagissent pour un cas d'utilisation donné.
- ✓ Trouver les messages envoyés entre les classes pour réaliser un certain comportement.
- ✓ Ne pas créer d'interaction les plus importants et les plus complexes.

4.1. Diagramme de séquence :

Ce diagramme décrit une action ordonnée dans le temps et est composé de trois concepts principaux :

- ✓ Les lignes de vie.
- ✓ Les messages.

✓ Les fragments.

Dans notre système nous allons traiter trois cas d'utilisation importants et qui sont :

✓ Création de compte et authentifier.

✓ Envoi des invitations.

✓ Envoi des messages et réception des réponses.

- **Création d'un compte et authentification :**

L'utilisateur saisit les données à partir d'une interface web laquelle vérifie leurs format et valide leur conformité selon des critères spécifiques. une fois validées, ces données sont transférées vers le serveur, si le serveur accepte la requête, les données sont stockées dans la base données à l'aide d'une méthode d'insertion. Cela permet à l'utilisateur d' authentifier, accéder au système et de modifier ultérieurement ces données personnelles.

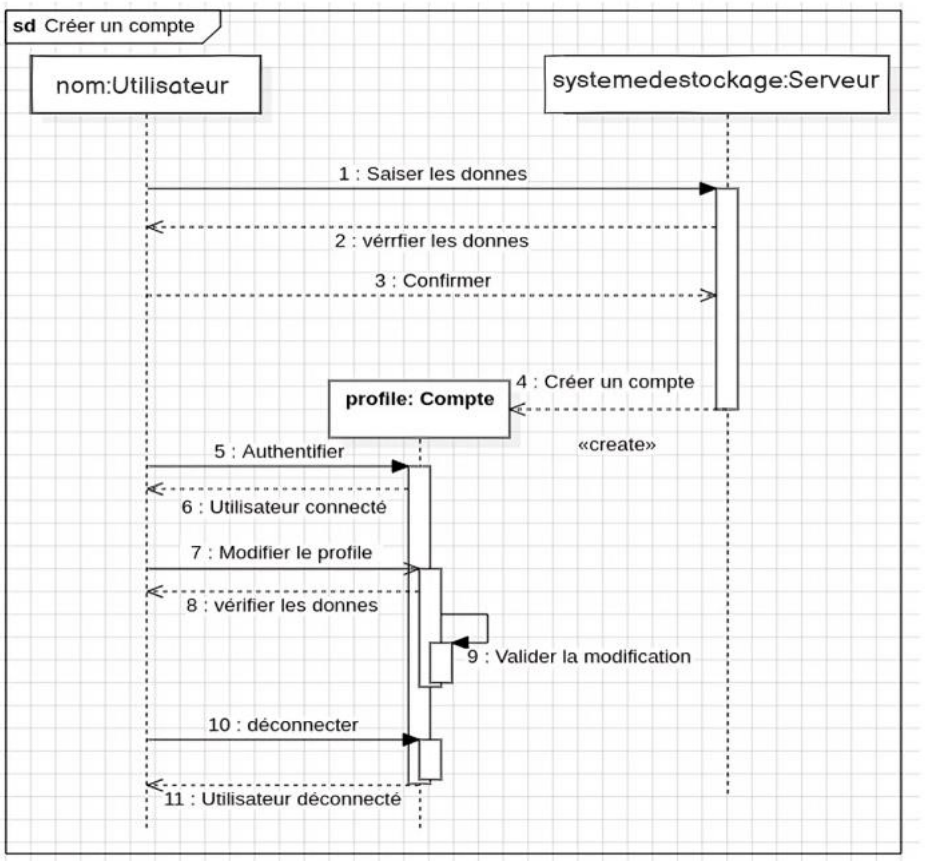

Figure 4: *Diagramme de séquence de la création d'un compte et authentification*

- **L'envoi des invitations :**

L'utilisateur recherche un autre utilisateur par un numéro de téléphone via une barre de recherche intégrée à l'interface web ce qui permet de déclencher une requête SQL de sélection permettant de récupérer le profil correspondant. Une fois le profile affiché un bouton de « **Envoyer une invitation** » est proposé .si l'utilisateur accepte l'invitation une relation d'amitié est créée entre les deux utilisateurs.

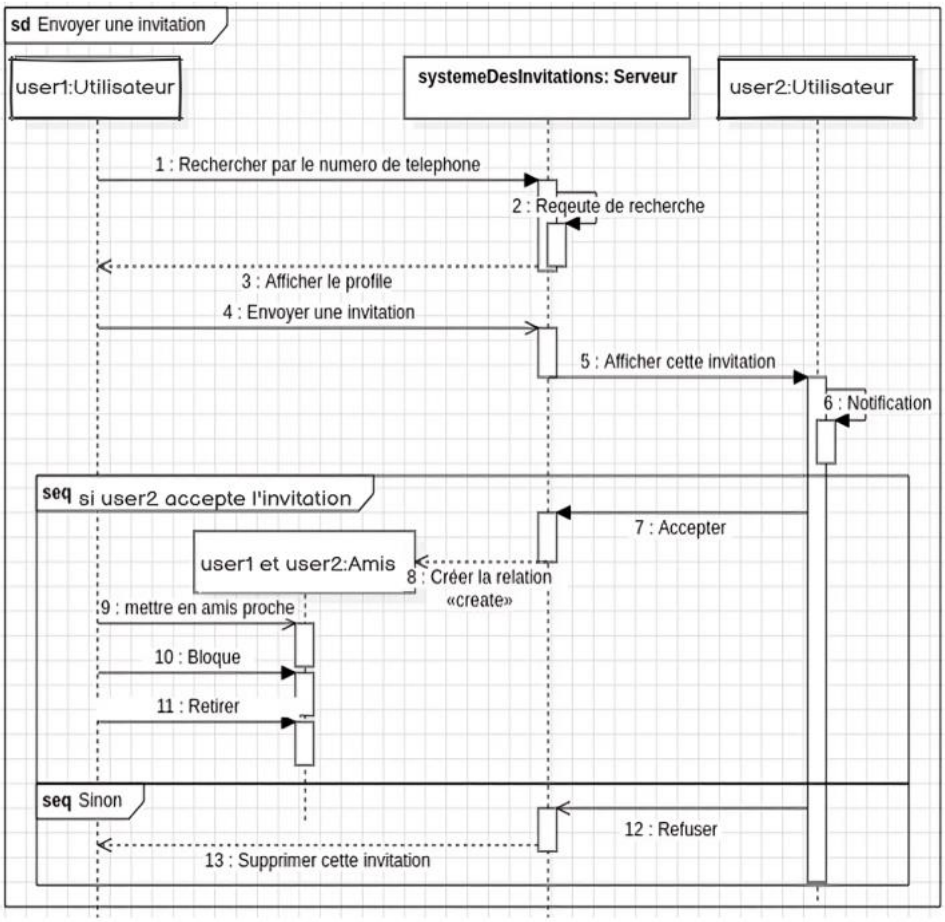

Figure 5: *Diagramme de séquence de l'envoi d'une invitation*

- **L'envoi des messages :**

Une fois un utilisateur est connecté, il peut envoyer un message à un autre utilisateur déjà présent dans sa liste des amis ou envoyer des messages dans des groupes où il est rejoint.

Ce message est accepté par le serveur et renvoyé à l'utilisateur cible, qui le reçoit et peut réagir ou répondre par un autre message.

Une notification de type « **Nouveau message** » est activée dès que le message est accessible.

L'utilisateur qui a envoyé ce message peut également le retirer, ce qui entraine sa Suppression par le serveur.

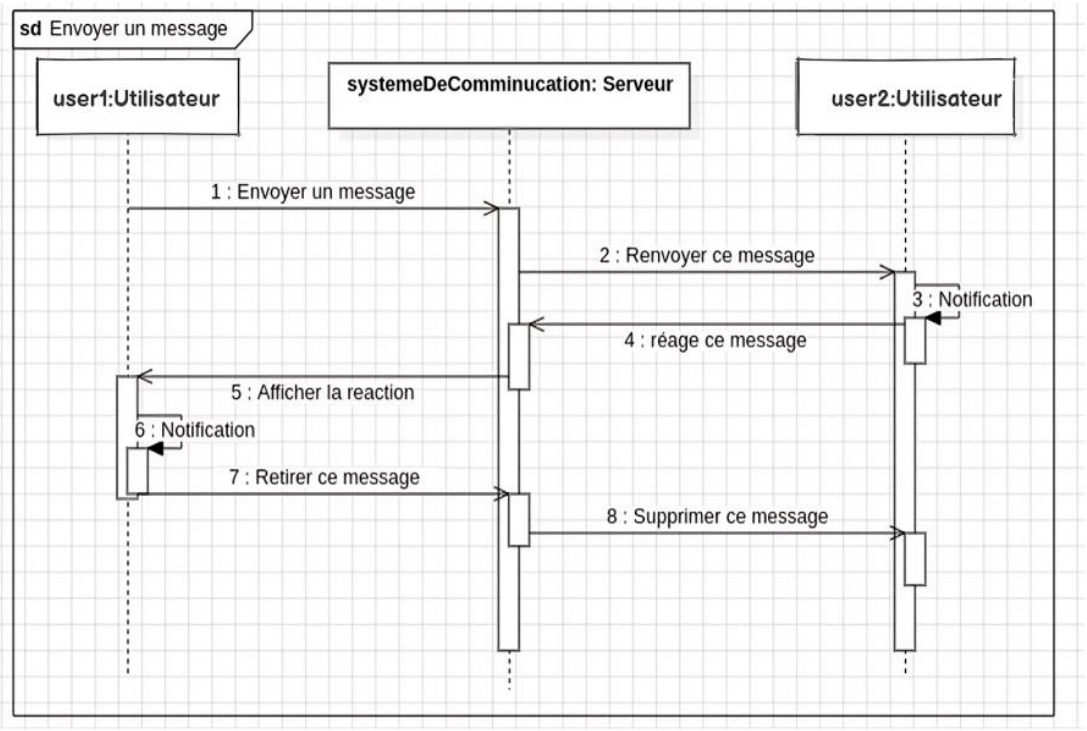

Figure 6: *Diagramme de séquence pour l'envoi un message*

5. Conception architecturale :

5.1. Architecturale logicielle :

L'architecture d'un système informatique est la structure du système qui comprend les éléments logiciels et leurs relations.

N'importe quel système complexe est composé de sous-systèmes qui interagissent entre eux.

La conception est le processus qui consiste à identifier ces sous-systèmes ainsi que les relations qu'ils entretiennent entre eux.

5.2. Représentation des architectures :

- **Composants et connecteurs (c&c) :**

Un composant est un module logiciel ou un entrepôt de données (Client, Serveur, base de donnes…) identifié par un nom qui indique son rôle dans le système.

Un connecteur représente une interaction entre ces composants. (Szyperski, 2002)

Cette représentation est un graphe contenant des composants et des connecteurs.

- ✓ Les composants dans notre système sont :
- ✓ Client web
- ✓ Serveur WebSocket
- ✓ Navigateur
- ✓ Interface Utilisateur (identification/authentification, Principal, Profile, Paramètres).

5.3. Diagramme de composants :

Représente la vue logique d'une architecture et permettent de modéliser les composants et leurs interactions. Il aide à comprendre comment les différentes parties d'un système interagissent. (Larman, 2004)

- Un composant définit un sous système de n'importe quelle taille en terme d'interface fournies et interface requises.
- Une interface fournies définit les fonctions qu'un composant **pourrait faire**.
- Une interface requise définit qu'un composant **attend** de son environnement.
- Si une même interface est requise pour un composant et fournie par l'autre est à dire un **assemblage** entre deux composants.

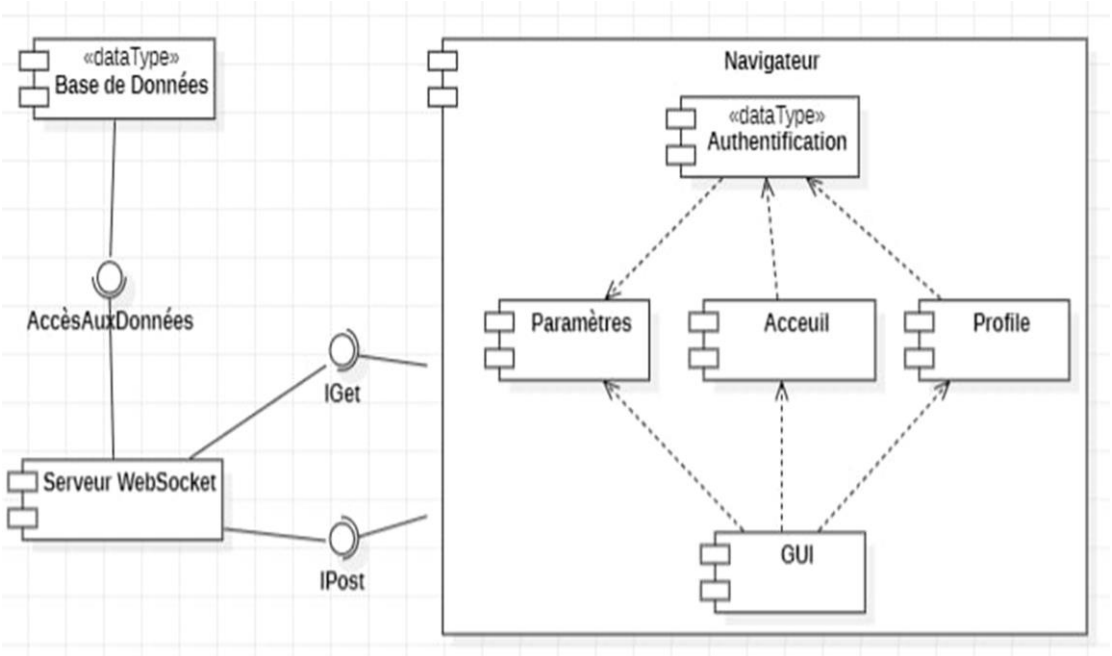

Figure 7: *Diagramme de composants pourl'application de chat*

5.4. Style architectural :

Un style architectural est un modèle définit comment sera le système. Tous les systèmes ont des points communs, ces systèmes auront des architectures qui se ressemblent. Le regroupement de ces architectures est appelé **style architectural.**

Avec un style architectural on peut connaître les composants, les connecteurs et les contraintes définissant l'architecture d'un système. (Bass, 2012)

Nous avons choisir l'architectures **client-serveur** pour concevoir l'application de chat.

- **Architecture client-serveur :**

Est composé de deux composants principaux et dans des machines séparées :

- ✓ **Client :** envoie des requêtes au serveur.
- ✓ **Serveur :** réagit aux requêtes en renvoyant des réponses

L'interface utilisateur se trouve au niveau client.

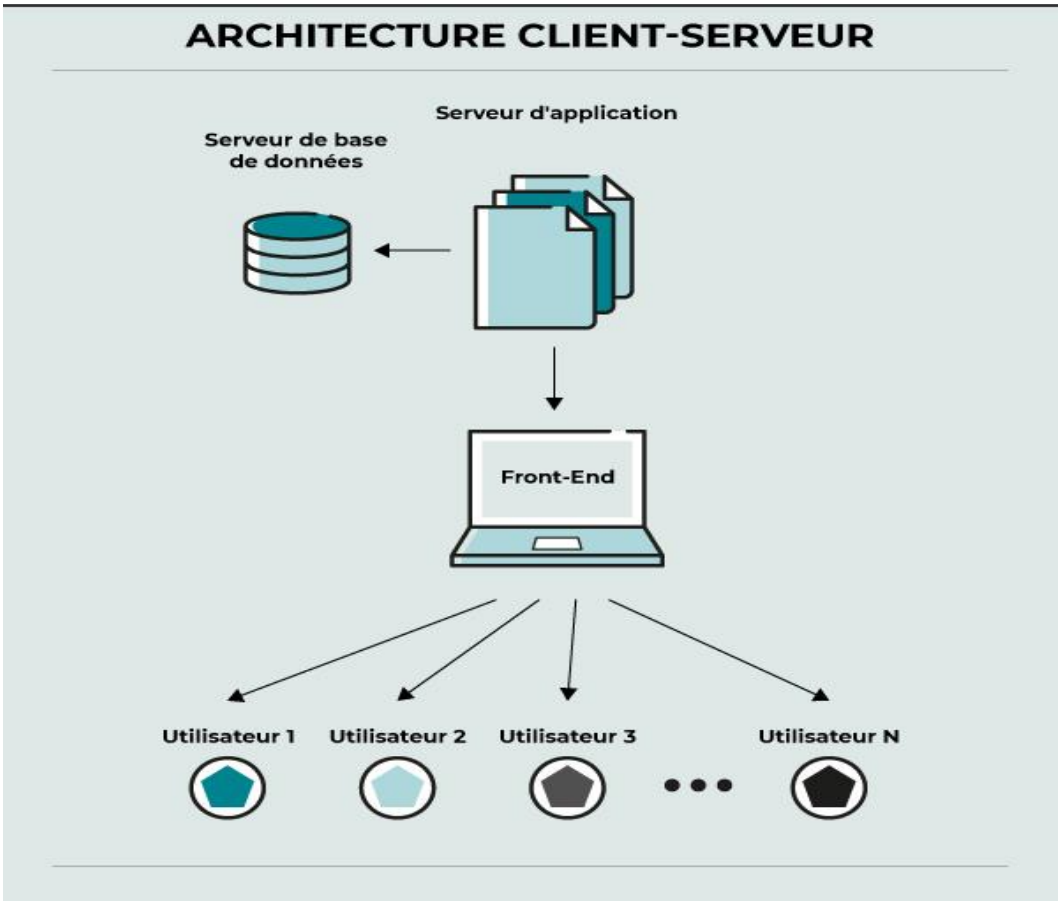

Figure8:*Fonctionnement de l'architecture client serveur* **(*openclassrooms*)**

6. Conclusion :

Au final, ce chapitre a clairement défini les interactions entre les différents acteurs du système, l'architecture ainsi que les fonctionnalités du système qui répondent correctement aux besoins des utilisateurs. Cela garantira une bonne mise en œuvre et assurera le succès et une meilleure qualité du projet.

Chapitre 03 : Implémentation

1. Introduction :

ce chapitre représente les différentes étapes utilisées pour implémenter et développer notre application de messagerie en temps réel suite à l'identification des besoins de l'utilisateur et de l'architecture du système. L'objectif attendu de ce chapitre est de mettre en œuvre le projet en respectant les fonctionnalités identifiées précédemment. L'implémentation comprend principalement, la création de la base de données, la mise en place du backend qui comprend un serveur dédié à la communication avec la base de données, l'élaboration du serveur WebSocket qui garantit la communication en temps réel, également l'implémentation du frontend qui signifie le développement de l'interface utilisateur qui communique avec le backend pour assurer le bon fonctionnement de l'application.

2. Conception d'une base de données :

L'application de conversation bidirectionnel contient beaucoup d'informations pour l'échange des données entre les utilisateurs et nécessite une base de données pour stocker les données de ce système.

2.1. Structuration des informations nécessaires :

Dans Le chapitre précédant de conception de projet nous définissons les données nécessaires du système de messagerie instantanée.

Ce système repose essentiellement sur la gestion des utilisateurs et des relations entre eux, á travers des fonctionnalités telles que :

- ✓ Les invitations d'amis.
- ✓ Les relations d'amitié.
- ✓ L'échange de messages.
- ✓ La gestion des conversations (privé ou groupe).
- ✓ Les notifications

Premièrement, nous avons choisi le système de gestion de base de données **Mysql** pour stocker et gérer toutes les informations.

Ensuite, les informations ont été organisées ont été organisées de manière structurée pour garantir une gestion cohérente et efficace des données au sein du système:

- ✓ Les données sont divisées en plusieurs tables.
- ✓ Les informations au sein de chaque table sont organisées en colonnes, représentent les différents attributs de cette entité.
- ✓ Nous avons également spécifier les clés primaires pour chaque table afin de garantir l'unicité des enregistrements et définir des clés étrangères pour assurer l'intégrité des relations entre les différentes tables.

- **Tables principales crées :**

- ✓ **Utilisateur :** contient les informations des utilisateurs du systéme.
- ✓ **Invitation :** gère les demandes d'ajout entre utilisateurs, avec un statut indiquant l'état de l'invitation.
- ✓ **Message :** stocke les messages échangés entre utilisateurs.
- ✓ **Conversation :** représente les conversations (privé ou de groupe).
- ✓ **Conversation utilisateur :** gère l'appartenance des utilisateurs aux conversations.
- ✓ **Notification :** informe les utilisateurs des nouveaux messages ou des invitations reçues.

2.2. Schéma relationnel de données :

Ce schéma représente la base de données de l'application DZChat , il contient plusieurs tables :

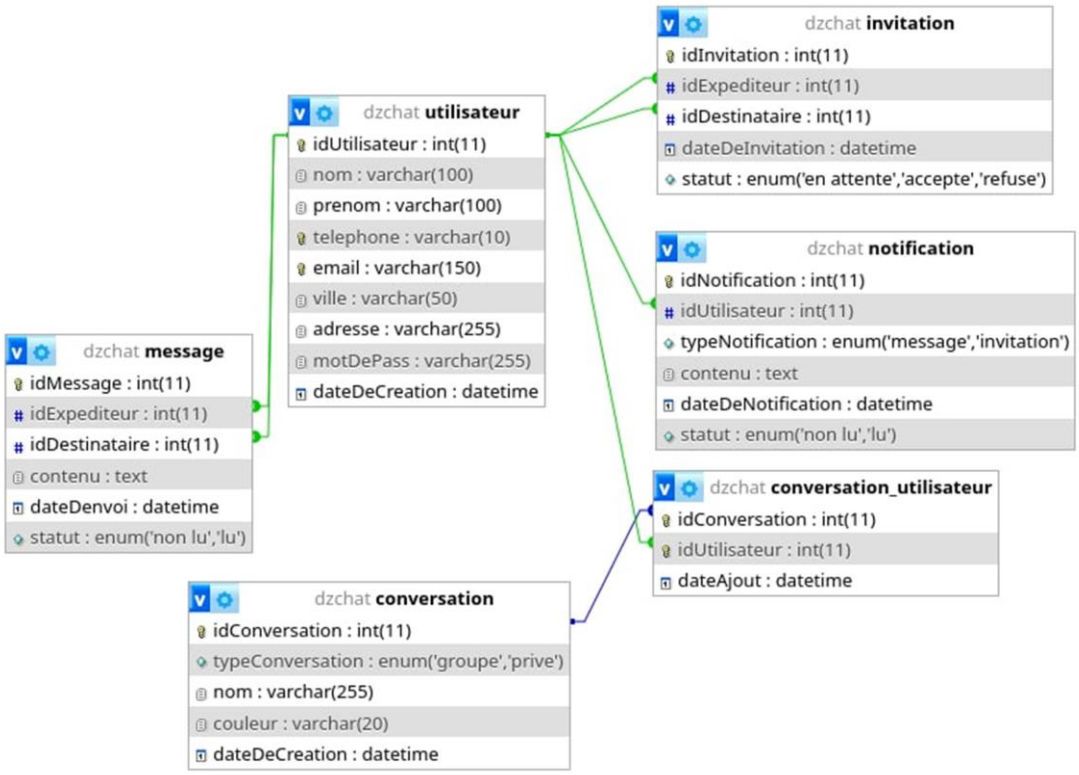

Figure 9: *Schéma conceptuel de la base de données*

3. Conception de l'expérience utilisateur et de l'interface utilisateur :

3.1. Définition :

En design numérique, l'interface utilisateur (UI) désigne l'interactivité, l'apparence et la convivialité d'un écran de produit ou d'une page web, tandis que l'expérience utilisateur (UX) couvre l'expérience globale de l'utilisateur avec le produit ou le site web.

Selon le principe de Pareto, 80 % des utilisateurs de l'application n'utilisent que 20 % de ses fonctionnalités (Mazumdar, 2024) , donc il faut identifier la tâche la plus importante de l'application et la rendez très simple. Dans notre application la tâche la plus importante est : L'envoi des messages.

3.2. Processus de conception d'interface utilisateur :

le processus de conception de l'interface utilisateur consiste à imaginer, structurer et représenter visuellement l'interaction entre l'utilisateur et l'application, et cela comprend des étapes telles que l'esquisse du wireframing et les storyboards.

- **Sketching :**

Le sketching signifie simplement un brouillon visuel des bases d'une interface utilisateur réalisé généralement à la main, avant de passer au wireframe (justinmind, 2024), au prototypage et au codage. C'est-à-dire dessiner des lignes simples et des cases avec quelques mots avant de passer aux étapes plus détaillés

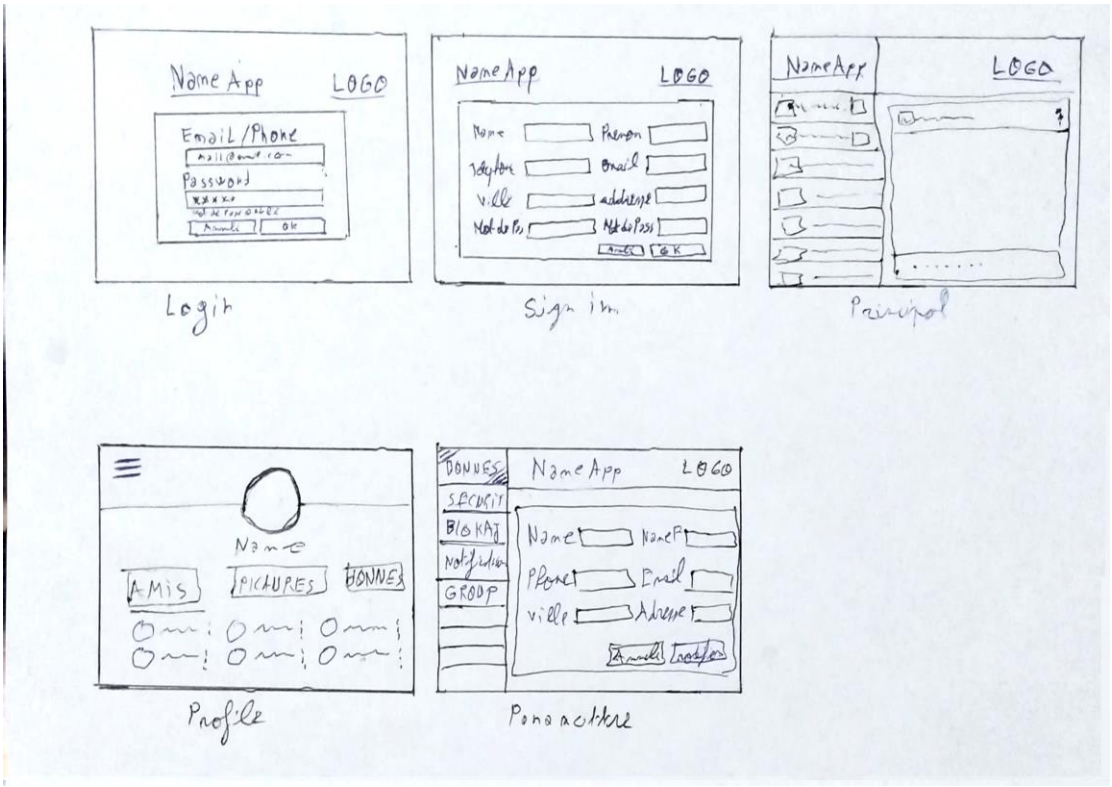

Figure 10:*l'étape Sketching*

- **Wireframing :**

Un wireframe est un guide visuel de base utilisé pour suggérer la structure et la disposition d'une interface numérique (jsutinmind, 2024). Les wireframes sont comme un plan architectural, ils permettent de se concentrer et planifier l'interface sans couleurs, polices de caractères, etc.

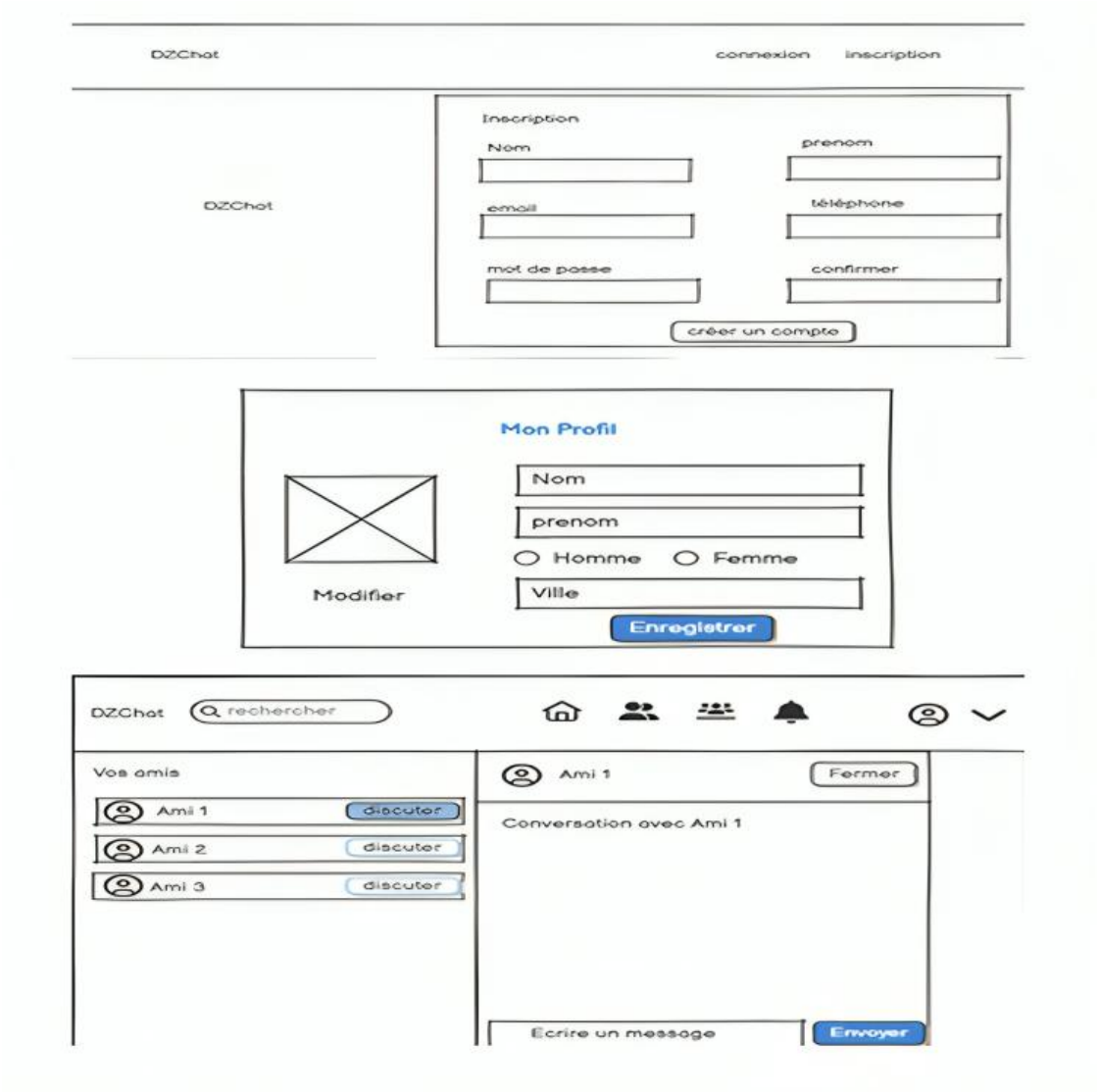

Figure 11:*l'étape de wireframing*

- **Storyboards (flux d'interface utilisateur) :**

Un flux d'interface utilisateur est un ensemble d'écrans ou des pages web qui définissent une tâche logique que l'utilisateur peut effectuer dans cette plateforme de chat.

1. **Si** l'utilisateur clique sur le bouton **s'inscrire**, **alors** il peut créer un compte.
2. **Si** l'utilisateur saisit correctement ces données d'authentification puis clique sur le bouton **se connecter**, **alors** il sera redirigé vers la page d'accueil.
3. **Si** l'utilisateur clique sur le bouton **ajouter(+)** dans la page d'accueil, **alors** pourra ajouter un autre utilisateur en saisissant son numéro de téléphone.
4. **Si** l'utilisateur clique sur un ami, **alors** la conversation de cet ami est affiché et il peut envoyer et recevoir des messages.
5. **Si** l'utilisateur clique sur la photo de profile dans la page d'accueil, **alors** une liste d'actions s'affiche contenant le profile, les paramètres et la déconnexion.
6. **Si** l'utilisateur clique sur trois points à coté d'un ami dans la liste des amis, **alors** il peut retirer cet ami de la liste, le bloquer ou même supprimer la conversation entre eux.

- **Visual design :**

1. **Mode d'affichage :** light mode.
2. **Palette de couleurs :**

✓ Navy rgb(33, 53, 85).

✓ Blue rgb(62, 88, 121).

✓ Beige rgb(216, 196, 182).

3. **Typographie :** Roboto avec 14px pour les messages et interlignage 1.6.

4. **Icons :** logo de l'application, icone mail et téléphone dans la page de l'authentification, icone plus pour l'ajout des amis, icone pour le bouton de paramètres, des icones pour les amis et les photos.

4. Serveur :

4.1. Introduction :

Le serveur joue un rôle crucial dans le développement , il est chargé de traiter les requêtes envoyés par l'utilisateur et de fournir des réponses. Il est responsable de la logique métier , l'interaction avec les bases de données et d'assurer la sécurité des échanges. Sa performance est très importante pour assurer la continuité des services. Notre serveur est développé en utilisant **Express.js** et s'exécute sur un serveur **Node.js**.

4.2. Connexion de la base de données :

Avant de commencer le développement il faut établir une connexion avec la base de données :

```
const db = mysql.createConnection({
  host: 'localhost',
  user: 'root',
  password: '',
  database: 'dzchat'
});

db.connect((err) => {
  if (err) throw err;
  console.log('✅ Connecté à MySQL');
});
```

***Figure 12**:connexion de la BDD avec Express.js*

4.3. Présentation des APIs utilisées :

Une API, ou interface de programmation d'application, est un ensemble de définitions et de protocoles qui facilite la création et l'intégration des applications

(redhat, 2025). Dans le cadre de notre projet plusieurs APIs sont intégrées pour gérer les différentes fonctionnalités

- **Inscription :**

L'API d'inscription permet la création d'un nouveau compte utilisateur , l'utilisateur envoi son nom, prénom ,email, téléphone, mot de passe via l'API **Signup.**

```
app.post('/api/signup', (req, res) => {
  const { nom, prenom, telephone, email, motDePass } = req.body;

  const checkQuery = 'SELECT * FROM utilisateur WHERE email = ? OR telephone = ?';
  db.query(checkQuery, [email, telephone], (err, results) => {
    if (err) {
      console.error('Erreur lors de la vérification:', err);
      return res.status(500).json({ message: 'Erreur serveur lors de la vérification.' });
    }

    if (results.length > 0) {
      const exists = results[0];
      if (exists.email === email) {
        return res.status(409).json({ message: 'Cet email est déjà utilisé.' });
      }
      if (exists.telephone === telephone) {
        return res.status(409).json({ message: 'Ce numéro de téléphone est déjà utilisé.' });
      }
    }


    bcrypt.hash(motDePass, 10, (err, hashedPassword) => {
      if (err) {
        console.error('Erreur lors du hachage du mot de passe:', err);
        return res.status(500).json({ message: 'Erreur lors du traitement du mot de passe.' });
      }


      const code = crypto.randomInt(100000, 999999).toString();
      const dateDeCreation = new Date();

      const insertQuery = `
    INSERT INTO utilisateur
    ( nom, prenom, genre, telephone, email, motDePass, ville, photoprofil, enligne, dateDeCreation, code, verifie)
    VALUES ( ?, ?, NULL, ?, ?, ?, NULL, NULL, false, ?, ?, false)
`;
```

***Figure 13**: l'API d'inscription utilisée dans DZChat*

A travers l'API Signup un code de vérification est envoyé a l'email de l'utilisateur pour vérifier son compte.

```
    const mailOptions = {
      from: '"DZChat 👋" <mohamedbelkacem2203@gmail.com>',
      to: email,
      subject: 'Votre code de vérification - DZChat',
      html: `
<div style="font-family: Arial, sans-serif; padding: 20px; border: 1px solid #ddd; border-radius: 8px;">
  <h2 style="color: #2c3e50;">Bienvenue sur <span style="color: #007BFF;">DZChat</span> !</h2>
  <p style="font-size: 16px; color: #333;">Voici votre code de vérification :</p>
  <p style="font-size: 32px; font-weight: bold; color: #007BFF; margin: 16px 0;">${code}</p>
  <p style="font-size: 14px; color: #777;">Entrez ce code dans l'application pour confirmer votre adresse e-mail.</p>
  <br>
  <p style="font-size: 13px; color: #aaa;">Merci de nous rejoindre,<br>L'équipe DZChat</p>
</div>
`
    };
```

***Figure 14**:Email de vérification envoyé a l'utilisateur*

- **Authentification :**

Cette API vérifie les identifiants d'un utilisateur pour lui permettre d'accéder a son compte .

Une cookie de session est créer pour maintenir l'utilisateur connecté.

```
app.post('/api/login', (req, res) => {
  const { email, motDePass } = req.body;

  if (!email || !motDePass) {
    return res.status(400).json({ message: 'Email et mot de passe sont requis.' });
  }

  const query = 'SELECT * FROM utilisateur WHERE email = ?';
  db.query(query, [email], (err, results) => {
    if (err) {
      return res.status(500).json({ message: 'Erreur serveur.' });
    }

    if (results.length === 0) {
      return res.status(404).json({ message: 'Utilisateur non trouvé.' });
    }

    const user = results[0];

    bcrypt.compare(motDePass, user.motDePass, (err, match) => {
      if (err) {
        return res.status(500).json({ message: 'Erreur serveur.' });
      }

      if (!match) {
        return res.status(401).json({ message: 'Mot de passe incorrect.' });
      }


  req.session.user = {
  id: user.idUtilisateur,
  nom: user.nom,
  prenom: user.prenom,
  email: user.email,
  verifie: user.verifie,
  photoprofil: `http://localhost:3001${user.photoprofil}`
};

      console.log("🎯 Session de login créée :", req.session.user);

      res.status(200).json({
        message: 'Connexion réussie.',
```

Figure 15: *API authentification*

- **Vérification :**

Cette API permet de vérifier le code envoyé par email a l'utilisateur lors de l'inscription .

```
app.post('/api/verifyCode', (req, res) => {
  const { email, code } = req.body;

  const query = 'SELECT * FROM utilisateur WHERE email = ?';
  db.query(query, [email], (err, results) => {
    if (err) return res.status(500).json({ message: 'Erreur serveur lors de la vérification.' });
    if (results.length === 0) return res.status(404).json({ message: 'Utilisateur introuvable.' });

    const utilisateur = results[0];
    if (utilisateur.code === null) {
      return res.status(400).json({ message: 'Code expiré ou déjà utilisé.' });
    }
    if (utilisateur.code === code) {
      const updateQuery = 'UPDATE utilisateur SET verifie = true, code = NULL WHERE email = ?';
      db.query(updateQuery, [email], (err2) => {
        if (err2) return res.status(500).json({ message: 'Erreur lors de la mise à jour.' });
        if (req.session.user) {
    req.session.user.verifie = true;
  }

        const getUserQuery =
          'SELECT idUtilisateur AS id, nom, prenom, email, ville, genre FROM utilisateur WHERE email = ?';
        db.query(getUserQuery, [email], (err3, rows) => {
          if (err3 || rows.length === 0) {
            return res.status(500).json({ message: 'Erreur lors de la récupération des données utilisateur.' });
          }

          const user = rows[0];
          return res.status(200).json({
            message: '✅ Compte vérifié avec succès.',
            user
          });
        });
      });
    } else {
      return res.status(400).json({ message: '❌ Code de vérification invalide.' });
    }
  });
});
```

Figure 16: *API de vérification du code de confirmation*

- **Configuration du profil :**

A travers cette API l'utilisateur peut configurer son profil en ajoutant son genre, ville et une photo de profil.

```
app.post('/api/updateProfile', upload.single('photo'), (req, res) => {
  const { email, nom, prenom, genre, ville } = req.body;

  if (!email) {
    return res.status(400).json({ message: 'Email manquant.' });
  }

  let photoPath = null;
  if (req.file) {
    photoPath = `/uploads/${req.file.filename}`; // chemin à stocker dans la BDD
  }

  const sql = `
    UPDATE utilisateur
    SET nom = ?, prenom = ?, genre = ?, ville = ?, photoprofil = ?
    WHERE email = ?
  `;

  db.query(sql, [nom, prenom, genre, ville, photoPath, email], (err, result) => {
    if (err) {
      console.error("Erreur lors de la mise à jour du profil :", err);
      return res.status(500).json({ message: 'Erreur serveur.' });
    }

    return res.status(200).json({ message: 'Profil mis à jour avec succès.', photo: photoPath });
  });
});
```

Figure 17:*API de configuration du profil*

- **Recherche d'utilisateur :**

Permet de rechercher un utilisateur via son numéro de téléphone.

```
app.get('/api/recherche-utilisateur', (req, res) => {
  let { telephone } = req.query;
  console.log('Téléphone brut:', telephone);

  if (!telephone) {
    return res.status(400).json({ message: 'Le numéro de téléphone est requis.' });
  }

  telephone = telephone.trim(); // nettoie les espaces

  const query = 'SELECT idUtilisateur, nom, prenom, telephone, photoprofil FROM utilisateur WHERE telephone = ? LIMIT 1';
  db.query(query, [telephone], (err, results) => {
    if (err) {
      console.error('Erreur SQL:', err);
      return res.status(500).json({ message: 'Erreur serveur.' });
    }

    console.log('Résultats:', results);

    if (results.length === 0) {
      return res.status(404).json({ message: 'Utilisateur non trouvé.' });
    }

    res.json({ user: results[0] }); // 👈 dans ton backend Node.js

  });
});
```

Figure 18:*API de recherche d'utilisateur*

- **Envoi d'invitations, acceptation et refus :**

A travers ces trois APIs l'utilisateur peut envoyer une demande d'amitié et contrôler les demandes reçus.

```
app.post('/api/invitation', (req, res) => {
  const { expediteur_id, destinataire_id } = req.body;

  if (!expediteur_id || !destinataire_id) {
    return res.status(400).json({ message: 'Les deux identifiants sont requis.' });
  }


  if (expediteur_id === destinataire_id) {
    return res.status(400).json({ message: 'Vous ne pouvez pas vous inviter vous-même.' });
  }

  const query = 'INSERT INTO invitation (expediteur_id, destinataire_id) VALUES (?, ?)';
  db.query(query, [expediteur_id, destinataire_id], (err, results) => {
    if (err) {

      return res.status(500).json({ message: 'Erreur serveur lors de l\'envoi de l\'invitation.' });
    }

    res.json({ message: 'Invitation envoyée avec succès.' });
  });
});
```

Figure 19:*API d'envoi d'invitation*

- **Récupération des amis :**

Via cette API les amis de l'utilisateur connecté sont récupérés.

```
app.get('/api/mes-amis', (req, res) => {
  if (!req.session.user || !req.session.user.id) {
    return res.status(401).json({ error: 'Utilisateur non connecté' });
  }

  const id = req.session.user.id;
  console.log("ID utilisateur dans session:", id);

  const sql = `
    SELECT u.idUtilisateur, u.nom, u.prenom, u.photoprofil
    FROM amis a
    JOIN utilisateur u
      ON (
        (a.utilisateur1_id = ? AND u.idUtilisateur = a.utilisateur2_id)
        OR
        (a.utilisateur2_id = ? AND u.idUtilisateur = a.utilisateur1_id)
      )
  `;

  db.query(sql, [id, id], (err, results) => {
    if (err) {
      console.error(err);
      return res.status(500).send("Erreur serveur");
    }
    console.log("Résultats amis :", results);
    res.json(results);
  });
});
```

Figure 20: *API de récupération des amis*

- **Création de groupe :**

Cette API permet à l'utilisateur de créer un groupe en spécifiant le nom et en ajoutant les membres.

```
app.post('/creer-groupe', (req, res) => {
  const { nomGroupe, membres, createur } = req.body;

  if (!nomGroupe || !Array.isArray(membres) || membres.length === 0 || !createur) {
    return res.status(400).json({ message: 'Données manquantes ou invalides' });
  }

  const insertGroupeQuery = 'INSERT INTO groupe (nomGroupe, idCreateur) VALUES (?, ?)';
  db.query(insertGroupeQuery, [nomGroupe, createur], (err, result) => {
    if (err) {
      console.error('Erreur insertion groupe :', err);
      return res.status(500).json({ message: 'Erreur lors de la création du groupe' });
    }

    const idGroupe = result.insertId;


    const membresAvecCreateur = [...new Set([...membres, createur])];
    const values = membresAvecCreateur.map(id => [idGroupe, id]);

    const insertMembresQuery = 'INSERT INTO membre_groupe (idGroupe, idUtilisateur) VALUES ?';
    db.query(insertMembresQuery, [values], (err2) => {
      if (err2) {
        console.error('Erreur insertion membres :', err2);
        return res.status(500).json({ message: 'Erreur lors de l'ajout des membres' });
      }

      res.status(201).json({ message: 'Groupe créé avec succès', idGroupe });
    });
  });
});
```

Figure 21:*API de la création d'un groupe*

5. Serveur WebSocket :

5.1. Présentation :

Dans le contexte de l'application DZChat, nous avons mis en place un serveur WebSocket pour faciliter une interaction instantanée entre les utilisateurs. Au lieu de suivre le modèle traditionnel HTTP basé sur des requêtes/réponses, les WebSockets proposent une liaison persistante bidirectionnelle, indispensable pour un échange de messages en temps réel efficace.

5.2. Rôle dans l'application :

Le serveur WebSocket fonctionne en tant qu'entité de liaison entre les clients. Quand un utilisateur transmet un message, celui-ci est transféré au serveur par le biais du WebSocket, qui le réoriente sans délai vers le destinataire en ligne. Ceci autorise :

- Une latence diminuée.
- Une actualisation en temps réel des discussions.

5.3. Fonctionnement :

- **Etablissement d'une connexion :**

Au démarrage du serveur , il se met a l'écoute pour les connexions entrantes sur le port 8080, lorsque qu'un client se connecte un identifiant unique est attribué (UUID) pour le suivre pendant toute la durée de session.

Comme vous voyez dans la capture suivante un client ouvre une connexion WebSocket avec l'UUID **1f3f022d-dc7e-4c75-8f16-38c296ab01c8**. Ensuite le client envoi une requête de type **register** contenant son nom et le lien vers s photo de profil.

Figure 22: *enregistrement d'un client via WebSocket*

- **Echange de messages** :

Le serveur WebSocket reçoit en temps réel les évènement transmis par les clients qui sont connectés , dans la figure montrée ci-dessous le client identifié par l'UUID **4f8bfa52-12c8-428b-9191-e6c63d9c19c6** participe dans une salle de discussion identifié par l'id 3 .

Au premier lieu cet client envoi une série d'évènements du type typing indiquant qu'il est en train de rédiger un message , le serveur capte ces évènements et informe les autres utilisateurs connectés que l'utilisateur mohamed est en train d'écrire un message

Finalement , le message l'utilisateur envoi un message réel et ce message est transmis aux utilisateurs connectés dans la même salle de discussion.

```
📦 Données reçues du client 4f8bfa52-12c8-428b-9191-e6c63d9c19c6: { type: 'typing', groupeId: 3, nom: 'mohamed' }
📦 Données reçues du client 4f8bfa52-12c8-428b-9191-e6c63d9c19c6: { type: 'typing', groupeId: 3, nom: 'mohamed' }
📦 Données reçues du client 4f8bfa52-12c8-428b-9191-e6c63d9c19c6: { type: 'typing', groupeId: 3, nom: 'mohamed' }
📦 Données reçues du client 4f8bfa52-12c8-428b-9191-e6c63d9c19c6: { type: 'typing', groupeId: 3, nom: 'mohamed' }
📦 Données reçues du client 4f8bfa52-12c8-428b-9191-e6c63d9c19c6: { type: 'typing', groupeId: 3, nom: 'mohamed' }
📦 Données reçues du client 4f8bfa52-12c8-428b-9191-e6c63d9c19c6: {
  type: 'message',
  text: 'hello',
  groupeId: 3,
  id: '5f165f22-92a3-46af-9730-082571851145',
  sender: 62,
  nom: 'mohamed',
  photoprofil: 'http://localhost:3001/uploads/photo-1747559070016-830029159.jpg'
}
📩 Message reçu de 4f8bfa52-12c8-428b-9191-e6c63d9c19c6 (groupe 3): hello
```

Figure 23 :*réception des évènements et réception d'un message dans un groupe via WebSocket*

- **Fermeture de connexion :**

Quand un client se déconnecte , l'évènement est détecté par le serveur et enregistré dans les logs

Dans l'exemple suivant les clients ont été déconnectés.

```
❌ Client déconnecté : 0665e25d-8ae4-489b-90ba-f7a3c61823e1
❌ Client déconnecté : bd38dc20-1042-4f31-aae3-9fe4f073e670
❌ Client déconnecté : 4f8bfa52-12c8-428b-9191-e6c63d9c19c6
```

Figure 24 :*notification de déconnexion d'un client WebSocket*

6. Client :

6.1. Introduction :

Cette partie décrit le processus utilisateur c'est-à-dire la séquence d'actions qu'utilisateur effectue pour utiliser l'app DZChat depuis l'inscription et jusqu'à l'envoi de messages. L'interface utilisateur est développé en utilisant React une bibliothèque JavaScript moderne qui permet de créer des interfaces dynamiques et interactives.

6.2. Etapes processus utilisateur :

- **Etape 1 : inscription :**

l'utilisateur ouvre l'app DZChat depuis se machine pour la première fois ,il choisit l'option **S'inscrire**.

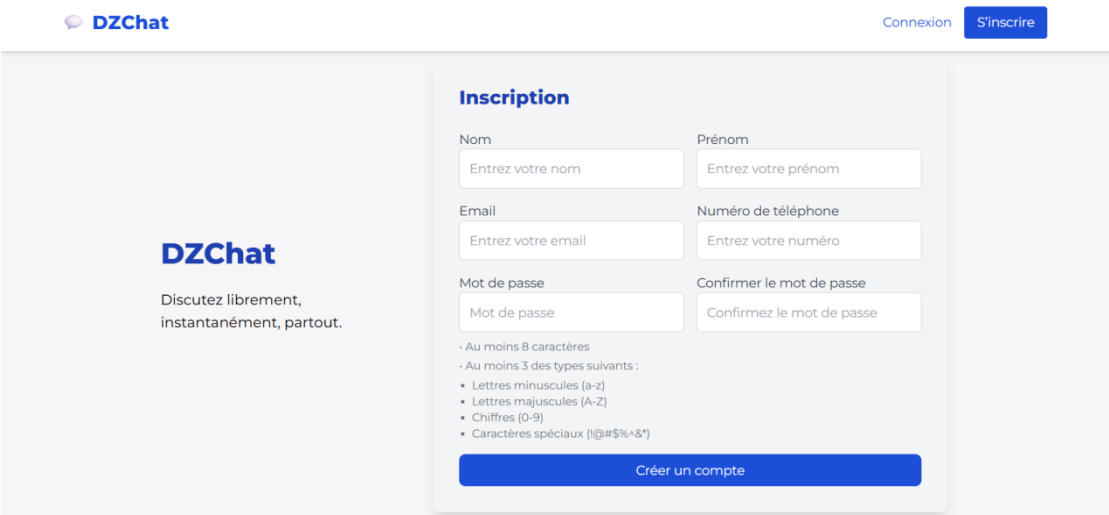

Figure 25: *interface inscription de DZChat*

- **Etape 2 : vérification de l'email :**

Après avoir saisir ses informations correctement dans la page de l'inscription , et notamment l'adresse email, l'application DZchat envoie un code de vérification automatiquement généré a l'utilisateur via **NodeMailer** .

Figure 26: *email de vérification envoyé par DZchat*

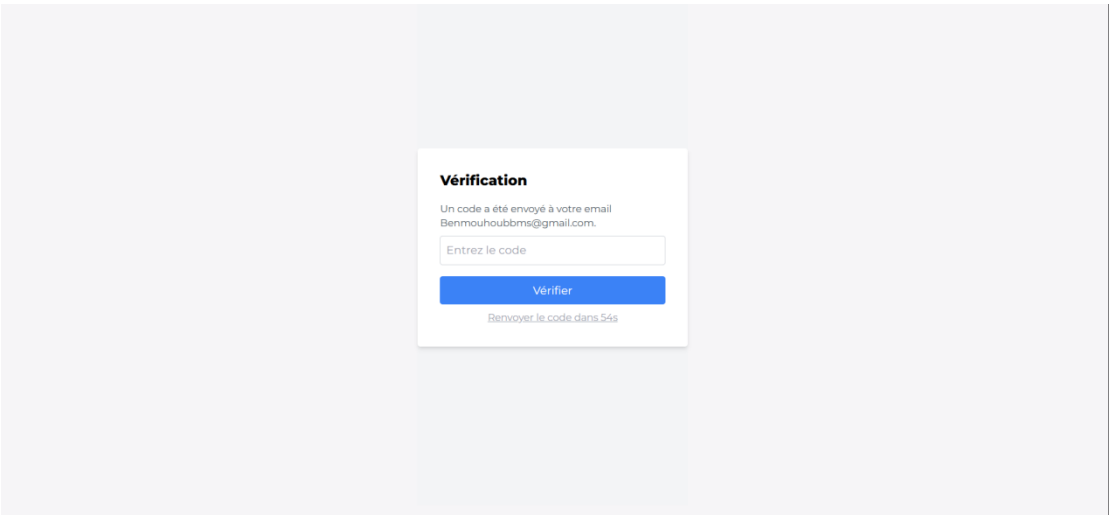

Figure 27: *Page de saisie de code de vérification*

- **Etape 3 : configuration du profil :**

Après la saisie du code correcte et la vérification de l'email , l'utilisateur est dirigé vers la page **Profil** pour configurer son profil personnel **.**

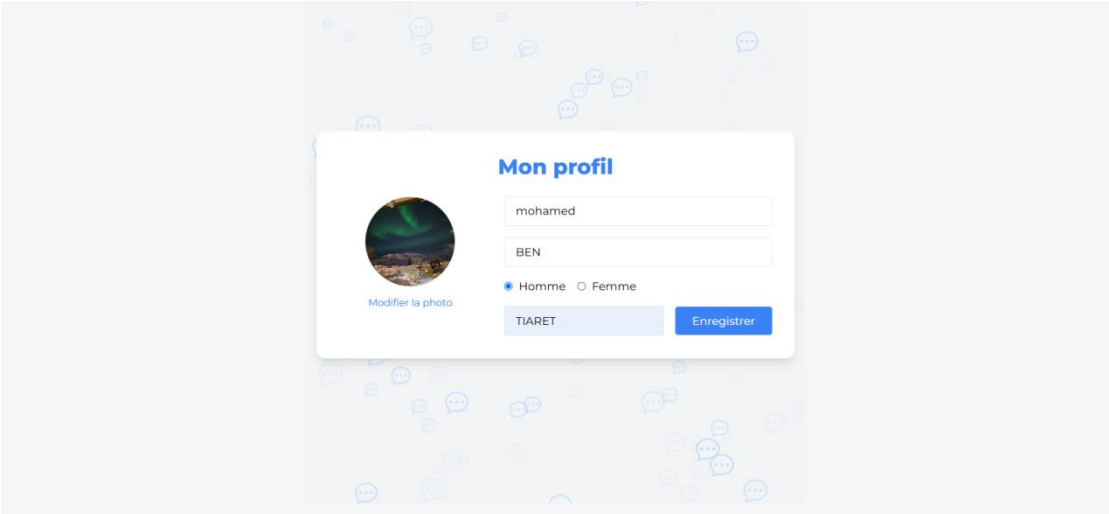

Figure 28:*interface de la configuration du profil*

- **Etape 4 : recherche d'utilisateur et envoi d'invitation :**

Après la configuration du profil l'utilisateur est dirigé vers la page Accueil ou il peut rechercher des utilisateurs via leurs numéro de téléphone.

Figure 29:*recherche d'utilisateur et envoi d'unvitation*

- **Etape 5 : discussion :**

Une fois les invitations sont envoyées et acceptées par d'autres utilisateurs, une relation d'amitié est créée par les deux parties

Les amis de l'utilisateur connecté sont affichées par la suite dans la page d'accueil sous forme de cartes interactives

Quand l'utilisateur séléctionne le bouton discuter , une fenêtre de chat s'ouvre en lui offrant la possibilité de discuter en temps réel avec cet ami via WebSocket.

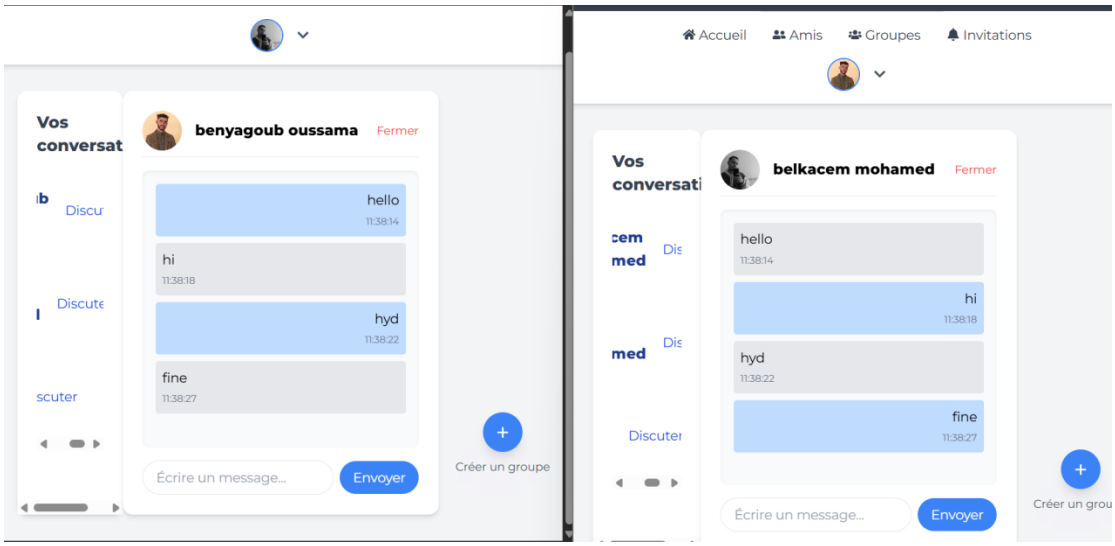

Figure 30:*interface de discussion en temps réel entre deux amis*

- **Etape 6 : création de groupe :**

Le bouton plus affiché dans l'écran d'accueill permet a l'utilisateur connecté de créer une nouvelle salle de discussion multi-clients

Lors du clic une boite de dialogue est affichée pour entre le nom du groupe et pour choisir les participants.

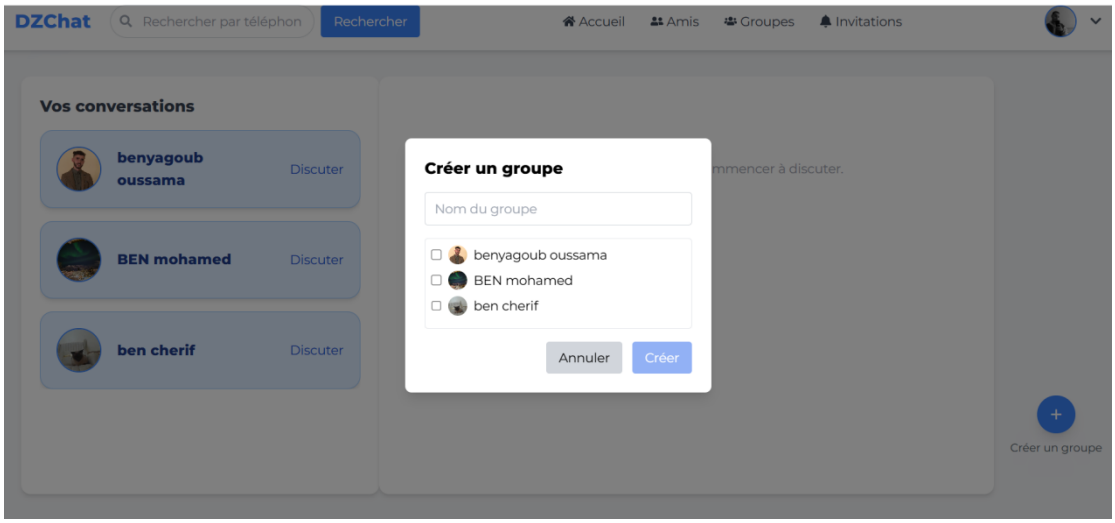

Figure 31:*Interface de création du groupe*

- **Etape 7 : discussion du groupe :**

Les groupes auxquels l'utilisateur connecté appartient sont affichés dans une page dédiée intitulée **mes groupes** . en appuyant sur le bouton **ouvrir** une fenêtre de chat s'affiche lui offrant la possibilité de discuter avec les membres de groupes en temps réel.

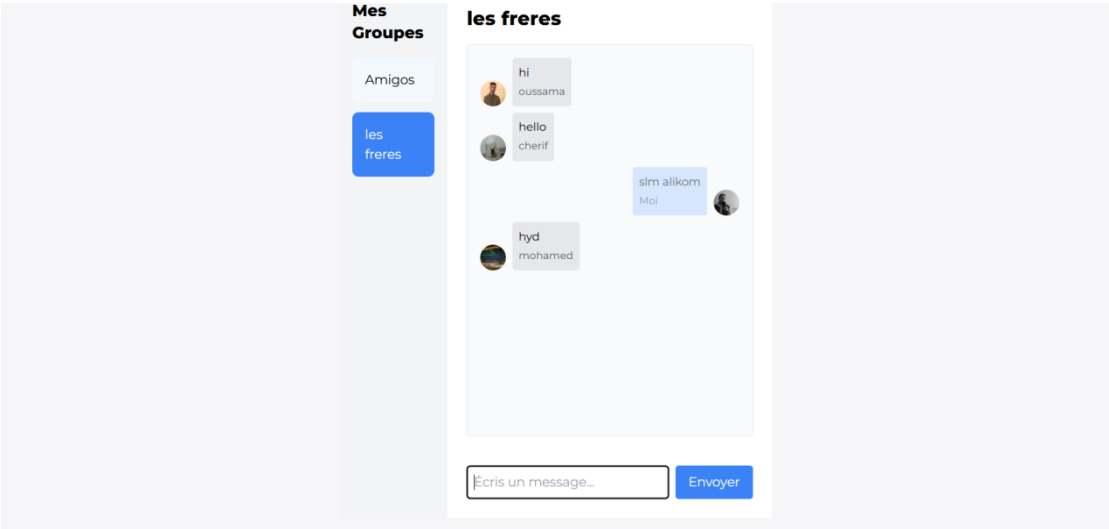

***Figure 32**: Interface de discussion en temps réel dans un groupe*

7. Conclusion :

ce chapitre a présenté les différentes étapes d'implémentation du projet En commençant par le serveur et la configuration de la connexion à la base de données, puis la mise en place du serveur websocket pour garantir la messagerie temps réel, l'intégration des API nécessaires, et enfin le développement de l'interface utilisateur.

Conclusion générale

Conclusion générale :

Ce projet de fin d'étude nous a permis de concevoir une application de messagerie en temps réel qui répond au problème posé au départ qui est l'absence d'un réseau social local , Pour cela, nous avons combiné des technologies modernes telles que **React, Tailwind , Node.Js** et **Express.Js ,** ainsi que **MySQL** que nous avons géré via **phpMyAdmin**.

Cependant , Il est à noter que cette version n'est qu'une version initiale. qui répond au besoin de base qui est la messagerie en temps réel. plusieurs améliorations peuvent être planifiées et appliquées au future telles que l'ajout de nouvelles fonctionnalités et l'amélioration de la sécurité.

Concernant les perspectives, ce projet peut évoluer pour une version plus complète et améliorée qui couvre tous les besoins des utilisateurs comme les publications, les Courtes vidéos. Les appels vocales et vidéo , ou encore le développement d' une version mobile native.

Bibliographie

Bass, L., Clements, P., et Kazman, R. 2012. *Software Architecture in Practice. 3rd ed.* Boston : Addison-Wesley, 2012.
Booch, G., Rumbaugh, J. et Jacobson, I. 2005. *The Unified Modeling Language User Guide.* Boston : Addison-Wesley, 2005.
Larman, C. 2004. *Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development.* Upper Saddle River, NJ : Prentice Hall, 2004.
Szyperski, C., Gruntz, D. et Murer, S. 2002. *Component Software: Beyond Object-Oriented Programming.* Boston : Addison-Wesley, 2002.

Webographie

AppMaster. 2023. Quels sont les avantages de l'utilisation de WebSocket pour la communication IoT ? *AppMaster.* [En ligne] 30 10 2023. [Citation : 5 3 2025.] https://appmaster.io/fr/blog/websocket-pour-la-communication-iot.
Capterra. Avis clients sur Telegram. *Capterra.* [En ligne] [Citation : 24 février 2025.] https://www.capterra.fr/reviews/180347/telegram.
Diaconu, Alex. 2023. Ably. *Ably.* [En ligne] 25 avril 2023. [Citation : 14 mars 2025.] https://ably.com/topic/what-are-websockets-used-for.
—. 2023. What are WebSockets used for? *ably.* [En ligne] 25 4 2023. [Citation : 12 3 2025.] https://ably.com/topic/what-are-websockets-used-for.
downloadsource. Telegram: les avantages et les inconvénients de cette application de messagerie instantanée. *downloadsource.* [En ligne] [Citation : 24 février 2025.] https://www.downloadsource.fr/telegram-et-sa-comparaison-avec-dautres-applications-similaires/n/4593/.
johnvincentt27. 2019. How to build a secure chat app like whatsapp, wechat & viber? *Medium.* [En ligne] 27 3 2019. [Citation : 3 3 2025.] https://medium.com/@johnvincentt27/how-to-build-a-secure-chat-app-like-whatsapp-wechat-viber-1245fa29ab3.
jsutinmind. 2024. Qu’est-ce qu’un wireframe ? *justinmind.* [En ligne] 15 juilliet 2024. https://www.justinmind.com/fr/wireframe.
justinmind. 2024. Guide du débutant pour le sketching de l’UI. *justinmind.* [En ligne] 8 aout 2024. https://www.justinmind.com/ui-design/sketching.
Komilov, Yodgorbek. 2024. System Design of the Telegram Android App: Layers and Technologies. *Medium.* [En ligne] 27 10 2024. [Citation : 1 3 2025.] https://medium.com/@YodgorbekKomilo/system-design-of-the-telegram-android-app-layers-and-technologies-6bc1965f4287.
Mazumdar, Stuti. 2024. The Pareto Principle in UX: Designing for the 80%. *Think Design.* [En ligne] Decembre 2024. https://think.design/blog/the-pareto-principle-in-ux.
Mozila, Developer. The WebSocket API (WebSockets). *developer mozilla.* [En ligne] [Citation : 8 3 2025.] https://developer.mozilla.org/en-US/docs/Web/API/WebSockets_API.
openclassrooms. Apprenez l'architecture client-serveur. *openclassrooms.* [En ligne] https://openclassrooms.com/fr/courses/7210131-definissez-votre-architecture-logicielle-grace-aux-standards-reconnus/7370196-apprenez-larchitecture-client-serveur.
opengenus. System Design of WhatsApp. *opengenus.* [En ligne] [Citation : 28 2 2025.] https://iq.opengenus.org/system-design-of-whatsapp/..
pubnub. 2023. pubnub. *pubnub.* [En ligne] pubnub, 3 septembre 2023. [Citation : 13 mars 2025.] https://www.pubnub.com/guides/websockets/.
quora. quora. *quora.* [En ligne] [Citation : 2 3 2025.] https://www.quora.com/What-frameworks-and-languages-were-used-to-develop-the-following-apps-full-stack-and-why-1-WhatsApp-2-Viber%20.
redhat. 2025. Une API, qu'est-ce que c'est ? *redhat.* [En ligne] 21 février 2025. https://www.redhat.com/fr/topics/api/what-are-application-programming-interfaces.

Simon, Kris Timothy. 2023. Advantages and Disadvantages of Viber. *Proflus.* [En ligne] 2 octobre 2023. [Citation : 24 février 2025.] https://www.profolus.com/topics/advantages-disadvantages-of-viber/.
—. 2023. Advantages and Disadvantages of Viber. *proflus.* [En ligne] 2 10 2023. [Citation : 24 2 2025.] https://www.profolus.com/topics/advantages-disadvantages-of-viber/.
Somanathan, Sudarshan. 2024. clickUp. *clickUp.* [En ligne] 21 décembre 2024. [Citation : 22 février 2025.] https://clickup.com/fr-FR/blog/245386/avantages-et-inconvenients-de-whatsapp.
team, Developer relations. 2023. What are WebSockets ws:// & wss:// connections. *pubnub.* [En ligne] 3 septembre 2023. https://www.pubnub.com/guides/websockets/.
vaadata. 2025. WebSockets : fonctionnement, vulnérabilités et bonnes pratiques sécurité. *vaadata.* [En ligne] 11 3 2025. [Citation : 14 3 2025.] https://www.vaadata.com/blog/fr/websockets-fonctionnement-vulnerabilites-et-bonnes-pratiques-securite/.