



Ministère de l'Enseignement
Supérieur et de la Recherche
Scientifique

Année Universitaire
2023/2024



Ecole Internationale Supérieure
Privée Polytechnique de Sousse

RAPPORT DE PROJET DE FIN D'ÉTUDES

Pour l'obtention du Diplôme National d'ingénieur en
Informatique

Spécialité: Génie logiciel

Intitulé

Conception et réalisation d'une Application Web
"Midad" avec Auto-completion des Messages

Lieu du stage

Smooth Algo

Réalisé par

Mariem Hajji

Encadré par

Mr. Adel Dahmane
Mr. Abdelmajid Dkhaila

Remerciements

Je voudrais exprimer ma gratitude à tous ceux qui ont contribué à l'élaboration de ce travail.

À tous les agents administratifs de l'EPI DIGITAL,

Pour votre sérieux, votre sens du devoir et vos compétences qui nous ont énormément marqués, et pour nous avoir fourni les outils nécessaires à la réussite durant ces trois années de formation.

À Monsieur Adel Dahmane, mon encadrant,

Pour votre aide et vos judicieux conseils qui ont guidé ma réflexion. Je vous remercie également pour votre disponibilité tout au long de la réalisation de ce travail.

À Monsieur AbdelMajid Dkhila, mon tuteur de stage,

Pour le suivi que vous avez apporté à mon stage, vos conseils et vos explications. Veuillez trouver ici le témoignage de ma profonde gratitude et de mes sincères remerciements.

Que tous ceux qui m'ont soutenu de près ou de loin trouvent ici l'expression de ma profonde gratitude.

Dédicaces

C'est avec un grand plaisir que je dédie ce mémoire à ceux que j'aime le plus au monde.

À mes très chers parents Lotfi et Chadlia ;

En signe de reconnaissance pour tous les efforts et les sacrifices fournis jour et nuit pour mon éducation et mon bien-être. Aucune dédicace ne saurait exprimer l'amour, l'estime et le respect que j'ai toujours eu pour vous. Recevez à travers ce travail, toute ma gratitude et mes profonds sentiments. Que Dieu vous donne longue vie et bonne santé.

À mes frères Ahmed et Houssein ;

Pour votre soutien et vos encouragements tout au long de mes années d'étude.

À tous mes amis et particulièrement à Mariem ;

Pour votre soutien moral, votre gentillesse sans égale, vos conseils et vos encouragements. Je vous dédie ce travail avec tous mes vœux de bonheur, de santé et de réussite.

TABLE DES MATIÈRES

INTRODUCTION GÉNÉRALE.....	1
CHAPITRE 1 : PRESENTATION GENERALE	3
Introduction	3
1.1. Cadre du travail.....	3
1.2. Présentation de la société d'accueil.....	3
1.3. Concepts de base.....	4
1.3.1 Problématique.....	4
1.3.2 Objectifs du projet	5
1.3.3 Processus de développement	6
1.4. Etude théorique	7
1.4.1 Prospection de solutions existantes	7
A. Solution 1 : Telegram.org.....	7
B. Solution 2 : whatsapp.com	8
C. Solution 3 : Messenger.com	9
1.4.2 Critiques et recommandations	10
Conclusion.....	12
CHAPITRE 2 : CAPTURE ET SPECIFICATION DES BESOINS	13
Introduction	13
1.1. Vision du projet	13
1.1.1 Exigences fonctionnelles	14
A. Module de gestion de la messagerie	14
B. Module de gestion des amis	16
C. Module de gestion des appels.....	16
D. Module de gestion des groupes	17
1.1.2 Exigences non fonctionnelles	18
1.2. Spécification des exigences.....	18
1.2.1 Identification des acteurs	18
1.2.2 Identification des cas d'utilisation.....	19
1.2.3 Classement des cas d'utilisation	21
1.2.4 Planification du projet en itérations.....	23
1.2.5 Diagrammes des cas d'utilisation	24
A. Diagramme de cas d'utilisation globale	25
B. Diagramme de cas d'utilisation du module messagerie	26
1.2.6 Description textuelle des cas d'utilisation majeurs	26
A. S'inscrire.....	26
B. Envoyer un message	27
C. Consulter Conversation	27

D. Gérer un groupe.....	28
E. S'authentifier	29
1.2.7 Diagrammes de séquence système	29
A. Diagramme de séquence système « S'inscrire ».....	30
B. Diagramme de séquence système « S'authentifier »	31
C. Diagramme de séquence système « Consulter un conversation »	31
D. Diagramme de séquence système « Envoyer un message »	32
E. Diagramme de séquence système « Gérer un groupe »	33
Conclusion	34
CHAPITRE 3 : ANALYSE ET CONCEPTION OBJET	35
Introduction	35
1.1. Diagrammes de séquence conception.....	35
1.1.1 Diagramme de séquence conception « S'inscrire ».....	35
1.1.2 Diagramme de séquence conception « Envoyer un message »	36
1.1.3 Diagramme de séquence conception « Consulter conversation ».....	36
1.1.4 Diagramme de séquence conception « Gérer un groupe »	37
1.1.5 Diagramme de séquence conception « S'authentifier »	37
1.2. Modélisation diagramme de classe.....	38
1.3. Modélisation non relationnelle	40
Conclusion	41
CHAPITRE 4 : PRESENTATION TECHNIQUE.....	42
Introduction	42
1.1. Choix technologiques.....	42
1.1.1 Architecture générale.....	42
A. Serveur Frontend	43
B. Serveur Backend.....	46
1.1.2 Chemin développement	47
A. Architecture MVC.....	47
B. Architecture React.....	48
1.1.3 Environnement technique	50
A. Environnement technologique	50
B. Environnement logiciel	51
1.2. Processus DevOps.....	52
1.2.1 Contrôle du code source avec Gitlab.....	52
1.2.2 Tests Unitaires	53
1.2.3 Approche CI/CD.....	54
1.2.4 Cloud & Conteneurisation	56
1.3. Intelligence artificielle	58
1.3.1 Algorithme de Prédiction des Réponses	58

A.	Analyse des Données Pertinentes	59
B.	Extraction et transformation des données.....	59
C.	Elaboration du modèle d'IA	60
D.	Évaluation et Interprétation des Résultats	61
1.3.2	Algorithme auto-complétion	61
1.4.	Autres fonctionnalités intégrées	63
1.4.1	Système de notifications.....	63
1.4.2	Système d'appel audio/vidéo.....	64
1.4.3	Système d'invitation d'amis	64
1.5.	Jeu d'essais	65
1.5.1	Présentation des interfaces de l'application.....	65
Conclusion	70
CONCLUSION GÉNÉRALE & PERSPECTIVES	71
RÉFÉRENCES CONSULTÉES	72

LISTE DES FIGURES

Figure 1 : Logo de la société [1].....	3
Figure 2 : Coordonnées de l'entreprise.....	4
Figure 3 : Schéma complet de processus de développement adopté [2]	7
Figure 4 : Page d'accueil du site Telegram.org [3]	8
Figure 5 : Page d'accueil du site Whatsapp.com [4].....	9
Figure 6 : Page d'accueil du site Messenger.com [5].....	9
Figure 7 : L'expression préliminaire des besoins [6].....	13
Figure 8 : Diagramme de cas d'utilisation globale	25
Figure 9 : Diagramme de cas d'utilisation du module messagerie.....	26
Figure 10 : Diagramme de séquence de système « S'inscrire »	30
Figure 11 : Diagramme de séquence système « S'authentifier »	31
Figure 12 : Diagramme de séquence système « Consulter conversation »	32
Figure 13 : Diagramme de séquence système « Envoyer un message »	33
Figure 14 : Diagramme de séquence de système « Gérer un groupe »	34
Figure 15 : Diagramme de séquence conception du scénario nominal « S'inscrire »	35
Figure 16 : Diagramme de séquence de la conception du scénario nominal « Envoyer un message»	36
Figure 17 : Diagramme de séquence de la conception du scénario nominal « Consulter conversation »	36
Figure 18 : Diagramme de séquence de la conception du scénario nominal « Ajouter un membre»	37
Figure 19 : Diagramme de séquence de la conception du scénario nominal « S'authentifier».....	38
Figure 20 : Diagramme de classes.....	39
Figure 21 : Modélisation non relationnelle.....	41
Figure 22 : Le serveur frontend [7].....	43
Figure 23 : Le serveur backend.....	46
Figure 24 : Illustration du l'architecture MVC	48
Figure 25 : Midad dans Gitlab[8].....	52
Figure 26 : Exemple de test unitaire 3A	54
Figure 27 : Représentation de l'approche CI/ CD avec Jenkins[9].....	55
Figure 28 : Exemple de Pipeline [10]	56
Figure 29 : Configuration du fichier Dockerfile	57
Figure 30 : Capture d'analyse des données pertinentes	59
Figure 31 : Capture d'extraction et transformation des données	60
Figure 32 : Capture d'élaboration du modèle.....	60
Figure 33 : Capture d'évaluation et interprétation des résultats.....	61
Figure 34 : Capture d'implémentation de la Fonction d'Auto-complétion.....	62
Figure 35 : Gestion des Requêtes d'Auto-complétion	63
Figure 36 : Interface de connexion.....	65
Figure 37 : Interface de l'inscription	66
Figure 38 : Interface de la messagerie	66
Figure 39 : Interface de profil	67
Figure 40 : Interface de groupe	68
Figure 41 : Interface de la zone des messages	69
Figure 42 : Interface de l'appel audio/video	69
Figure 43 : Interface d'acceptation/refuse.....	70

LISTE DES TABLEAUX

Tableau 1 : Tableau comparatif des solutions existantes [1]	10
Tableau 2 : Identification des cas d'utilisation [2].....	19
Tableau 3 : Classement des cas d'utilisation [3]	21
Tableau 4 : Planification des itérations grâce aux cas d'utilisation [4]	23
Tableau 5 : Description textuelle de cas d'utilisation « S'inscrire ».....	26
Tableau 6 : Description textuelle de cas d'utilisation « Envoyer un message »	27
Tableau 7 : Description textuelle de cas d'utilisation « Consulter conversation »	27
Tableau 8 : Description textuelle de cas d'utilisation « Gérer un groupe »	28
Tableau 9 : Description textuelle de cas d'utilisation « S'authentifier »	29
Tableau 10 : Description de diagramme de classes	39
Tableau 11 : Description des attributs du diagramme de classes	40
Tableau 12 : Un aperçu de l'architecture frontend [5]	44
Tableau 13 : Un aperçu de l'architecture backend[6].....	46
Tableau 14 : Présentation de l'environnement technologique.....	50
Tableau 15 : Présentation de l'environnement logiciel	51

INTRODUCTION GÉNÉRALE

La montée en puissance des technologies de communication a profondément modifié la façon dont les individus communiquent. Les outils de communication indispensables sont les applications de messagerie instantanée, qui permettent de se connecter rapidement et efficacement. Dans cette situation, notre initiative vise à créer une application de chat novatrice qui intègre deux fonctionnalités essentielles : l'auto-complétions et la prédiction des messages. L'objectif de ces caractéristiques est d'optimiser l'expérience utilisateur en prévenant et en simplifiant la rédaction des messages.

Notre projet utilise des concepts de l'intelligence artificielle, du traitement automatique du langage naturel (NLP) et de l'apprentissage automatique comme base théorique. Les progrès en matière d'interfaces utilisateur intuitives et adaptatives nous permettent également de créer une expérience de messagerie fluide et performante.

Il est parfois difficile de rédiger des messages dans les applications de chat, surtout sur les appareils mobiles où la saisie est restreinte. En outre, les utilisateurs ont souvent du mal à exprimer leurs idées de manière précise. L'objectif de notre projet est de relever ces défis en créant une application de chat qui peut anticiper les besoins des utilisateurs et leur offrir une assistance contextuelle tout au long de leur appel.

Afin d'atteindre nos objectifs, nous utilisons une méthodologie de travail qui combine des méthodes de traitement automatique du langage naturel, d'apprentissage automatique et de conception d'interfaces utilisateur. Dans un premier temps, nous allons recueillir et analyser un vaste ensemble de données de chat afin d'entraîner nos modèles de prédiction et d'auto-complétions. Par la suite, ces modèles seront développés et perfectionnés en utilisant des algorithmes d'apprentissage automatique sophistiqués. En même temps, nous élaborerons une interface conviviale qui intègre de manière transparente ces fonctionnalités, en mettant l'accent sur l'accessibilité et la facilité d'utilisation.

L'objectif principal de ce mémoire est de concevoir et évaluer une application de chat novatrice qui simplifie grandement l'expérience utilisateur grâce à l'auto-complétions et à la prédiction des connexions. C'est dans cette situation que notre projet de fin d'études a été réalisé au sein de l'entreprise Smooth Algo. Nous visons à souligner l'efficacité de ces caractéristiques en ce qui concerne la rapidité de saisie, la réduction des erreurs et la facilité de prise de contact. En outre, nous cherchons à étudier les effets de cette technologie sur les interactions sociales et la manière dont les utilisateurs interagissent avec les applications de messagerie.

Pour atteindre cet objectif, notre rapport est scindé en quatre chapitres :

- Le premier chapitre, intitulé « **Présentation générale** » présente l'organisme d'accueil, le contexte général, la problématique et les objectifs du projet. La méthodologie de travail ainsi que l'étude de l'existant et des solutions existantes.
- Le deuxième chapitre, intitulé « **Capture et spécification des besoins** » sera consacré sur la vision du projet et la spécification des besoins avec une explication approfondie des fonctionnalités, ainsi qu'à la planification des sprints.
- Le troisième chapitre, intitulé « **Analyse et conception objet** » est dédié à la réalisation des cas d'utilisations majeurs, la conception et la modélisation relationnelle.
- Le quatrième chapitre, intitulé « **Présentation technique** », il contient les choix technologiques, l'architecture logicielle et du projet ainsi que l'intelligence artificielle et la réalisation du projet.

Finalement, nous clôturons par une conclusion générale qui présente une récapitulation du travail réalisé avec quelques perspectives.

CHAPITRE 1 : Présentation générale

Introduction

Ce chapitre est une première étape qui vise à présenter l'étude préliminaire avant de mettre en œuvre notre solution. Il vise à examiner la problématique, à définir les objectifs de notre travail, à décrire le cadre méthodologique, et à expliquer l'approche systématique proposée.

Finalement, nous effectuerons une analyse approfondie des solutions disponibles dans le domaine.

1.1. Cadre du travail

Le projet est réalisé dans le cadre de la préparation de stage de fin d'études pour l'obtention du diplôme d'ingénierie en génie logiciel à l'École Pluridisciplinaire Internationale (EPI-Digital).

Notre projet a été réalisé au sein de la société Smooth Algo pendant une période de cinq mois dans le but d'atteindre un objectif final qui consiste à concevoir et à développer une application de chat avec auto-complétions et prédiction des messages. Nous avons cherché à créer une application robuste et conviviale qui a permis de simplifier efficacement le processus de rédaction et d'envoi de messages instantanés. En intégrant des fonctionnalités d'auto-complétions et de prédiction, nous avons visé à améliorer l'expérience utilisateur en proposant des suggestions pertinentes et en anticipant les besoins de l'utilisateur lors de la rédaction de messages.

1.2. Présentation de la société d'accueil



Figure 1 : Logo de la société [1]

Smooth Algo est une société de services en ingénierie informatique située à Sahloul Sousse spécialisée dans le conseil et le développement de systèmes d'information, fondée en 2017, présentée dans la figure 1.

Malgré sa taille modeste, Smooth Algo est une organisation dynamique avec une équipe dédiée de 5 personnes professionnelles. Chaque année, l'entreprise s'engage activement dans des programmes de mentorat, guidant les stagiaires

À travers leurs projets de fin d'études et leurs stages d'été. Cet investissement dans les talents le développement profite non seulement à l'entreprise, mais contribue également à la croissance et au succès de l'avenir diplômés en informatique.

Les coordonnées de l'entreprise sont détaillées comme le montre la figure 2 ci-dessous :




	Sahloul Sousse, Tunisie
	+216 54 991 891
	contact@smoothalgo.com

Figure 2 : Coordonnées de l'entreprise

1.3. Concepts de base

1.3.1 Problématique

La nécessité d'outils de communication performants dans le contexte numérique actuel est mise en évidence par des chiffres étonnants qui mettent en évidence la dépendance grandissante à la collaboration à distance et aux interactions virtuelles. D'après une récente étude de Business Research Insights, l'état actuel du monde est mondial.

En 2022, le marché des applications de messagerie a atteint une valeur de 24 210 millions USD. De plus, l'estimation du marché est d'atteindre 58 843 millions USD d'ici 2032, avec un taux de croissance annuel composé remarquable (TCAC) de 9,3 % pendant la période prévisible.

Bien que des solutions de communication existantes soient disponibles, il reste une lacune sur le marché des solutions de communication. Une application web intégrée qui intègre de manière fluide la messagerie instantanée et l'audio/vidéo tout en accordant une grande importance à la sécurité et à l'expérience utilisateur. Souvent, les solutions disponibles manquent de certaines fonctionnalités ou font des compromis sur les mesures de sécurité, ce qui laisse les utilisateurs avec des expériences et des préoccupations fragmentées quant à la confidentialité des données.

Dans ce contexte, Le projet « Midad.tn » vise à développer une application de chat avec des fonctionnalités avancées telles que l'auto-complétion et la prédiction des réponses pour améliorer l'expérience de communication des utilisateurs. Cependant, plusieurs problématiques se posent dans la réalisation de ce projet :

- Quels sont les moyens les plus efficaces pour promouvoir l'application et attirer de nouveaux utilisateurs, en particulier dans un marché concurrentiel où de nombreuses applications de chat existent déjà ?
- Quels mécanismes de sécurité avancés doivent être mis en place pour assurer la confidentialité des conversations et des données personnelles des utilisateurs ?
- Comment garantir une adoption massive de l'application par les utilisateurs potentiels et les encourager à utiliser régulièrement comme leur principal moyen de communication en ligne ?

1.3.2 Objectifs du projet

Midad sera développée principalement pour répondre aux besoins de communication sécurisée des professionnels et ciblera les utilisateurs. Cette solution de messagerie vise à offrir une plateforme 100% tunisienne qui facilite la communication entre les utilisateurs. Voici les objectifs spécifiques de cette application :

- **Auto-complétions lors de la rédaction des messages** : intégrer cette fonctionnalité pour aider les utilisateurs à rédiger des messages plus rapidement et plus facilement en suggérant des mots ou des phrases basées sur ce qu'ils ont tapé précédemment.
- **Prédiction des réponses** : Utiliser des algorithmes d'intelligence artificielle pour prédire les réponses possibles aux messages, offrant ainsi aux utilisateurs des suggestions de réponses rapides.
- **Gestion des contacts et des conversations** : Permettre aux utilisateurs de créer et de gérer des conversations individuelles ou de groupe.
- **Fonctionnalités de chat en temps réel** : Permettre aux utilisateurs d'envoyer et de recevoir des messages en temps réel.
- **Statut en ligne** : Permet à chaque utilisateur de voir ses amis déjà connectés.
- **Historique des messages** : Permet aux utilisateurs de rechercher et de consulter les anciens messages entre eux.
- **Notification des messages** : Permet aux utilisateurs de recevoir des alertes lorsqu'ils reçoivent des messages de leurs amis.

- **Réactions aux messages** : Permet aux utilisateurs de réagir en utilisant des réponses textuelles.
- **Appel vocale/Appel vidéo** : La possibilité de passer des appels vocales/vidéos individuels ou en groupe, permettant aux utilisateurs de communiquer visuellement.
- **Partage les fichiers** : On peut partager des fichiers tels que des images entre les utilisateurs au sein des conversations.
- **Authentification par e-mail et vérification par code** : il s'agit d'un système d'authentification sécurisé basé sur l'e-mail, permettant aux utilisateurs de créer des comptes et de se connecter à l'application.

1.3.3 Processus de développement

Le processus de développement logiciel est un terme général décrivant le processus global de développement d'un produit logiciel. Étant donné qu'il est important pour notre application d'être au premier abord, il est donc nécessaire pour nous de suivre un processus étape par étape pour la créer. Pour cela, nous adopterons le processus unifié (UP) qui est une méthode itérative et incrémentale de développement logiciel. Voici une description sommaire de ses principales caractéristiques :

- **Itératif et incrémental** : Le développement se fait par cycles courts, chacun produisant une version fonctionnelle du logiciel.
- **Piloté par les cas d'utilisation et fondé sur l'utilisation d'un sous-ensemble nécessaire et suffisant du langage UML** : Les fonctionnalités sont définies à partir des besoins des utilisateurs.
- **Centré sur l'architecture** : Une importance particulière est accordée à la conception de l'architecture logicielle.
- **Quatre phases principales** :
 - Inception : Définition du projet et de sa portée
 - Élaboration : Analyse détaillée et conception de l'architecture
 - Construction : Implémentation du système
 - Transition : Déploiement du système

Tout au long d'un projet informatique, une des questions fondamentales auxquelles nous avons essayé de répondre s'est avérée très simple : comment passer des besoins des utilisateurs au code de l'application ? Autrement dit : « J'ai une bonne idée de ce que mon application doit faire, des fonctionnalités attendues par les futurs utilisateurs. Comment obtenir le plus efficacement possible un code informatique opérationnel, complet, testé, et qui réponde parfaitement au besoin ? ».

La figure 3 montre clairement les différents diagrammes UML à modéliser dans notre projet en phases d'analyse et de conception afin de passer des besoins au code. [Modéliser une application Web].

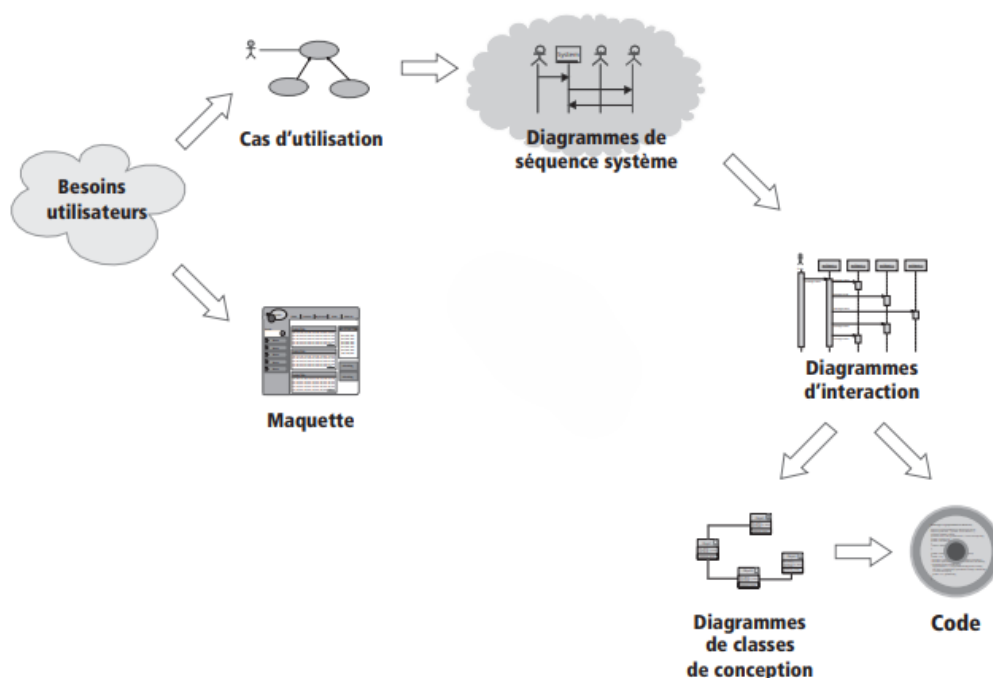


Figure 3 : Schéma complet de processus de développement adopté [2]

1.4. Etude théorique

Dans cette section, notre objectif est de mener une analyse comparative des différentes plateformes disponibles sur le marché. Par la suite, nous réaliserons une analyse critique des solutions déjà disponibles et des suggestions.

1.4.1 Prospection de solutions existantes

Au moment de l'étude d'un nouveau projet, il est essentiel d'examiner les applications existantes qui offrent des services similaires à ceux que nous prévoyons de produire. Grâce à cette étape, il est possible d'identifier les principaux points forts et les points faibles des solutions existantes afin de pouvoir offrir des fonctionnalités perfectionnées aux clients. Dans cette partie, nous exposons une étude de quelques applications déjà en place afin de mettre en évidence les principales fonctionnalités à améliorer dans notre application.

A. Solution 1 : Telegram.org

Telegram Messenger est une application et un service de messagerie instantanée multiplateforme en freemium hébergé dans un cloud. Telegram est basée à Dubaï. L'application cliente est gratuite, libre et open source, sous licence GPLv3, disponible sur smartphone ainsi que sur ordinateur et en tant qu'application web. Comme l'illustre la figure 4 ci-dessous, Telegram.org offre à ces utilisateurs la possibilité de communiquer entre eux.



Figure 4 : Page d'accueil du site Telegram.org [3]

B. Solution 2 : whatsapp.com

WhatsApp est une application mobile multiplateforme qui fournit un système de messagerie instantanée chiffrée de bout en bout aussi bien via les réseaux de téléphonie mobiles que par Internet. L'application s'est trouvée à plusieurs reprises au centre de vives critiques portant sur sa sécurité informatique, notamment sur la confidentialité des données personnelles qui y sont échangées. WhatsApp a également été le support de plusieurs campagnes de diffusion de fausses informations. Comme l'illustre la figure 5 ci-dessous la page d'accueil de WhatsApp.

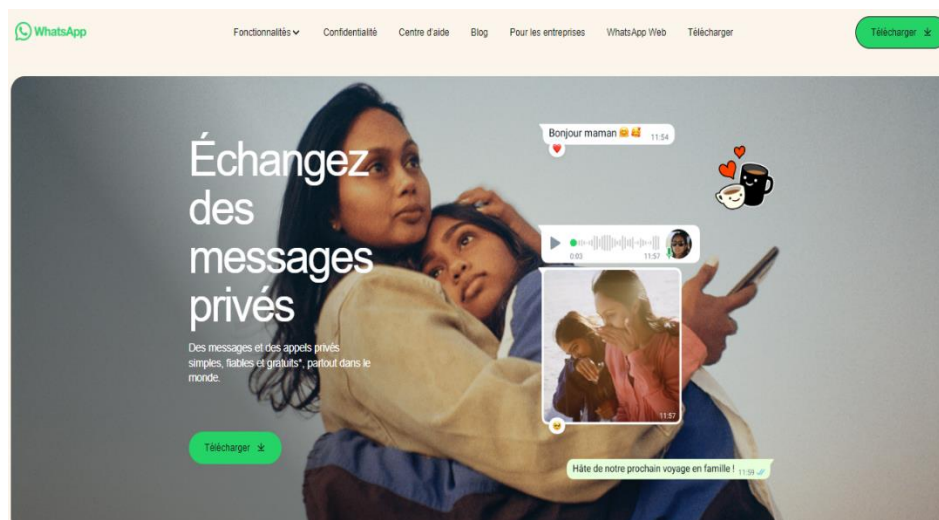


Figure 5 : Page d'accueil du site Whatsapp.com [4]

C. Solution 3 : Messenger.com

Facebook Messenger est un système de messagerie instantanée créé par la société Facebook, et incorporé aux réseaux sociaux Facebook et Instagram. Une application simple qui vous permet d'écrire des messages, de passer des appels vidéo et de rester proche des personnes qui comptent pour vous. Comme l'illustre la figure 6 ci-dessous la page d'accueil de Messenger.

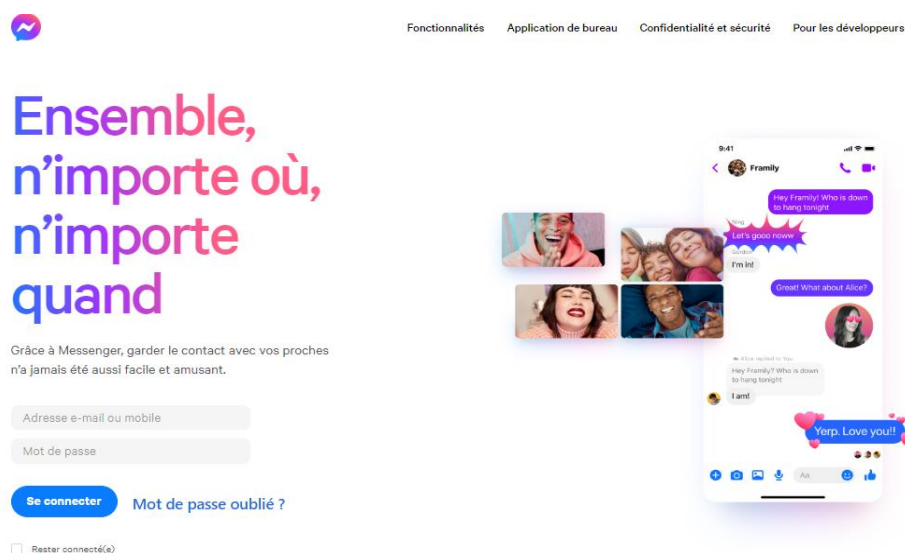


Figure 6 : Page d'accueil du site Messenger.com [5]

1.4.2 Critiques et recommandations

Dans le tableau 1 suivant, nous exposerons une analyse comparative des diverses solutions disponibles sur le marché. Les critères fonctionnels et non fonctionnels, ainsi que les évaluations des utilisateurs.

Tableau 1 : Tableau comparatif des solutions existantes [1]

CRITERES	TELEGRAM	WHATSAPP	MESSANGER
Messagerie instantanée	OUI	OUI	OUI
Gestion des groupes	OUI	OUI	OUI
Partage de fichiers	OUI	OUI	OUI
Gestion des appels audio et vidéo	OUI	OUI	OUI
Authentification par courriel	NON	NON	NON
CRUD message	OUI	OUI	OUI
Système de notification	OUI	OUI	OUI
Statut en ligne	OUI	OUI	OUI
Historique des messages	OUI	NON	OUI
Choisir sticker/emoji	OUI	OUI	OUI
Invitation par email	NON	NON	NON
Contacts en ligne	NON	NON	OUI
Rechercher des amis	OUI	OUI	OUI
Mode sombre	OUI	OUI	OUI
Gérer la conversation	OUI	OUI	OUI
Voir le profil	OUI	OUI	OUI
Modifier la photo de profil	OUI	OUI	OUI
La prédiction des réponses	OUI	OUI	OUI
Auto-complétions lors de la rédaction des messages	NON	NON	OUI
Synchronisation multi-appareils	OUI	OUI	OUI

CRITERES	TELEGRAM	WHATSAPP	MESSENGER
Messages multimédias	NON	OUI	OUI
Voir la liste des groupes	OUI	OUI	OUI
Ajouter des amis	OUI	OUI	OUI
Performance	OUI	OUI	OUI
Fiabilité	OUI	OUI	OUI
Cryptage de bout en bout	NON	OUI	OUI

En réponse aux faiblesses repérées dans les outils de communication actuels, le projet suggéré a pour objectif de concevoir une application web complète pour la messagerie instantanée et les appels audio/vidéo, connue sous le nom de Midad.

- Midad proposera une variété de fonctionnalités solides pour satisfaire les différents besoins des utilisateurs tout en accordant une grande importance à la sécurité, à la convivialité et aux performances.
- Midad aura une interface utilisateur facile à utiliser, ce qui facilitera la messagerie texte transparente et les échanges multimédias entre les utilisateurs. L'application ajoutera des fonctionnalités avancées comme le partage d'écran, l'intégration prédictive des réponses et auto complète lors la rédaction des messages, permettant ainsi à la fois les appels audio/vidéo entre individus et groupes. Grâce au chiffrement des communications en temps réel, les données des utilisateurs seront sécurisées et confidentielles, en plus de pratiques de gestion des données sécurisées.
- Midad intégrera des fonctionnalités avancées comme la traduction linguistique et le stockage des utilisateurs et des conversations dans une base de données MongoDB. Le plan de développement comprend diverses étapes, telles que la création de l'interface utilisateur, la mise en place de la fonctionnalité de messagerie, l'intégration d'appels audio et vidéo via WebRTC, la mise en œuvre du cryptage des communications et l'ajout de fonctionnalités avancées.
- La réalisation de tests complets, incluant les tests unitaires et d'intégration, les tests de sécurité et les tests de performance, assurera la fiabilité et la stabilité de l'application.

Pour résumer, Midad est une solution innovante et sécurisée qui s'adapte parfaitement aux contraintes des outils de communication existants, tout en adoptant de bonnes pratiques de programmation et en incorporant des technologies de pointe.

Conclusion

Dans ce chapitre, nous avons établi notre projet en mentionnant quelques exemples d'applications actuellement utilisées, dans le but de concevoir une application performante qui répond aux attentes des utilisateurs et la solution proposée. Le prochain chapitre traitera plus en détail de la capture et de la spécification des fonctionnalités.

CHAPITRE 2 : CAPTURE ET SPECIFICATION DES BESOINS

Introduction

L'analyse et la spécification des besoins nécessitent de détecter les diverses fonctionnalités que le système devrait offrir. Cette activité fait partie du processus global, explicatif dans le chapitre 1 (fig3). La première étape consiste à formuler les besoins de manière préliminaire, puis à les expliciter à travers des exemples d'utilisation et une maquette d'interface homme-machine, comme illustré dans la figure 7. L'objectif de ce chapitre est de faire la liste des exigences fonctionnelles et non fonctionnelles, ainsi que de définir les acteurs et les situations d'utilisation. Tous les cas d'utilisation seront expliqués à l'aide d'une fiche standard comprenant sa description textuelle, et seront également illustrés par des diagrammes de séquence système pour une analyse accrue.

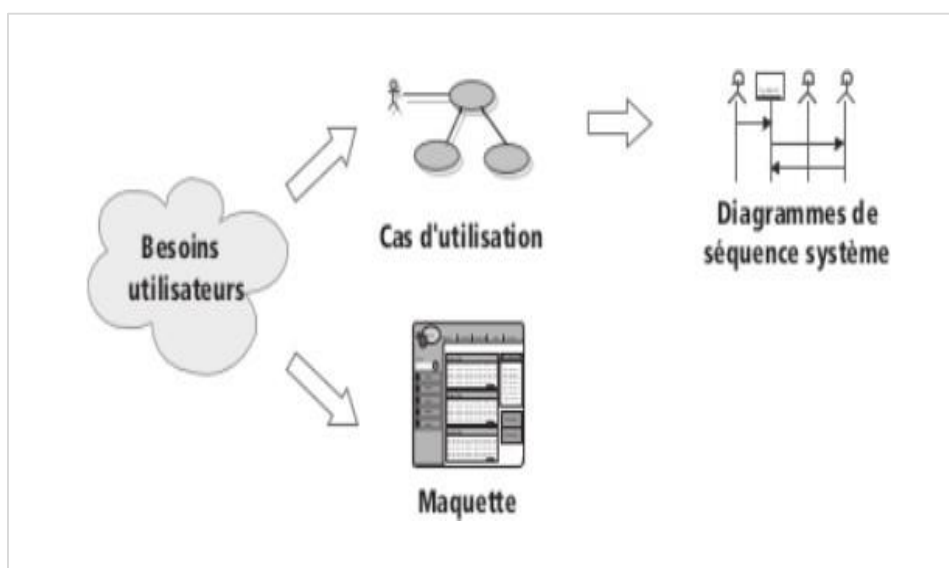


Figure 7 : L'expression préliminaire des besoins [6]

1.1. Vision du projet

L'objectif du projet MIDAD est de découvrir de nouvelles solutions aux défis rencontrés dans la communication en ligne. En offrant une plateforme facile à utiliser et accessible, MIDAD a pour objectif d'améliorer l'interaction entre les utilisateurs et de faciliter une communication efficace et fluide. En proposant des caractéristiques comme la personnalisation des profils, les échanges de messages instantanés et le partage sécurisé de fichiers.

1.1.1 Exigences fonctionnelles

Nous devons rassembler toutes les fonctionnalités essentielles dans notre application pour répondre de manière satisfaisante aux besoins des utilisateurs. Dans cette section, nous allons présenter ces fonctionnalités en les regroupant sous 5 modules distincts, à savoir par :

- Module de gestion de la messagerie
- Module de gestion des amis
- Module de gestion des appels
- Module de gestion des groupes

A. Module de gestion de la messagerie

La version 1 de la plateforme Midad se concentre sur l'amélioration de la communication avec les utilisateurs et de l'interaction sociale. Dans le paysage numérique de la communication, la capacité de se connecter de manière transparente avec des amis et des connaissances est au cœur de chaque plateforme de messagerie.

La version 1 s'engage à renforcer ces liens en introduisant des fonctionnalités visant à améliorer l'expérience de messagerie. Au fur et à mesure que les utilisateurs naviguent dans le domaine dynamique de la communication instantanée, l'accent est mis sur la promotion d'interactions significatives et la garantie d'un échange fluide de pensées et d'idées. Au fur et à mesure que les utilisateurs naviguent dans le domaine dynamique de la communication instantanée, l'accent est mis sur la promotion d'interactions significatives et la garantie d'un échange fluide de pensées et d'idées.

– Envoyer des messages et les recevoir en temps réel :

Dans le domaine dynamique de la messagerie en temps réel, le processus d'envoi et de réception de messages est l'élément vital de l'interaction numérique. Dans cette section, nous nous lançons dans une exploration complète de la dynamique complexe sous-jacente à cette fonctionnalité fondamentale de notre plateforme de messagerie.

– Recherche dans l'historique des messages :

Dans ce module, nous approfondissons la fonctionnalité de recherche de l'historique des messages dans des conversations spécifiques.

En permettant aux utilisateurs de rechercher dans l'historique des messages dans des conversations spécifiques, l'application leur permet de récupérer rapidement les conversations passées, facilitant ainsi la récupération et la référence transparentes des informations.

- **Modifier d'un message en temps réel :**

Dans cette section, nous nous penchons sur la fonctionnalité de mise à jour des messages en temps réel dans l'application Web Midad. Les mises à jour des messages en temps réel représentent un aspect crucial des plateformes de communication modernes, permettant aux utilisateurs d'éditer et de modifier instantanément leurs messages envoyés, même après leur envoi.

- **Supprimer d'un message en temps réel :**

Dans cette section, nous avons vu de la fonctionnalité de supprimer des messages en temps réel dans l'application web Midad.

- **Transférer d'un message :**

Dans cette section, nous explorons la fonctionnalité de transfert de messages en temps réel dans l'application Web Midad. Le transfert de messages est une fonctionnalité importante qui permet aux utilisateurs de partager des informations, des conversations ou des médias importants avec d'autres personnes de manière transparente. Qu'il s'agisse d'un seul message ou de plusieurs messages, les utilisateurs ont la possibilité de sélectionner le contenu souhaité et de le transmettre instantanément à un ou plusieurs destinataires en temps réel.

- **Affichage la liste des utilisateurs connectés :**

Le concept de connectivité transcende la simple interaction, incarnant une synergie dynamique entre les utilisateurs au sein d'une plateforme. Dans ces cas on a parlé sur une fonctionnalité essentielle : montrer les utilisateurs connectés via l'intégration de sockets.

Cette fonctionnalité innovante offre une visibilité en temps réel sur la présence de l'utilisateur, favorisant un sentiment de communauté et facilitant l'engagement instantané. En tirant parti de la technologie des sockets, la plateforme met à jour de manière transparente les statuts des utilisateurs, fournissant des informations précieuses sur la participation active et favorisant les interactions collaboratives.

B. Module de gestion des amis

Nous nous concentrons sur l'amélioration de l'expérience sociale au sein de la plateforme Midad grâce à l'introduction de fonctionnalités de gestion des amis. Ces nouvelles fonctionnalités simplifient le processus de connexion avec les autres, renforçant ainsi le sentiment de communauté et favorisant un plus grand engagement des utilisateurs.

– Ajouter des amis :

La fonction d'ajout d'amis permet aux utilisateurs de se connecter sans effort avec des connaissances, des collègues et des proches, renforçant ainsi le sentiment de communauté au sein de Midad. Notre objectif est de simplifier le processus d'ajout d'amis, de faciliter les connexions précieuses et de créer une expérience utilisateur plus dynamique.

– Envoyer une invitation par e-mail à un utilisateur non-abonné :

L'envoi d'invitations par e-mail à des non-abonnés est une stratégie courante pour attirer de nouveaux utilisateurs vers une plateforme. Dans le contexte de Midad, cela implique l'utilisation d'e-mails ciblés pour inviter des personnes qui ne sont pas encore inscrites sur la plateforme.

– Gérer les contacts et rechercher un ami :

La gestion des contacts et la recherche d'amis sur Midad jouent un rôle crucial dans l'amélioration de l'expérience de l'utilisateur en facilitant les échanges sociaux et professionnels.

– Gérer profil :

La gestion de profil sur Midad permet aux utilisateurs de personnaliser leur identité en ligne en gérant leur nom d'utilisateur, leur adresse e-mail et leur image de profil. Les utilisateurs peuvent choisir et modifier un nom unique pour être identifiés sur la plateforme, et changer la photo de profil.

C. Module de gestion des appels

Nous nous concentrons sur la révolution des capacités de communication de la plateforme Midad grâce à des fonctionnalités avancées de gestion des appels. Ce sprint vise à introduire et à affiner les fonctionnalités qui permettront aux utilisateurs de lancer et d'accepter des appels vidéo de manière transparente, améliorant ainsi leur expérience d'interaction en temps réel. De plus, nous mettrons en œuvre des fonctionnalités permettant d'afficher et de masquer la vidéo, ainsi que de couper et de réactiver le son, offrant aux utilisateurs un contrôle complet sur leurs paramètres de communication.

- **Lancer d'appels audio/vidéo :**

L'initiation d'appels audio/vidéo est une fonctionnalité essentielle de la plateforme Midad, améliorant la communication en temps réel entre les utilisateurs. Cette fonctionnalité permet aux utilisateurs de lancer des appels vidéo, de vérifier la disponibilité du destinataire et de gérer efficacement les réponses, garantissant ainsi une expérience d'appel vidéo transparente.

- **Accepter et refuser les appels audio/vidéo :**

L'acceptation et le refus audio/vidéo pendant les appels offrent aux utilisateurs la possibilité de contrôler leur présence visuelle. Cette fonctionnalité permet aux utilisateurs de d'accepter ou refuser selon le cas de l'utilisateur.

D. Module de gestion des groupes

Le module de gestion des groupes sur Midad a été développé dans le but de simplifier la création et la supervision des communautés sur la plateforme. Les utilisateurs ont la possibilité de créer des groupes de manière simple en définissant des éléments tels que le titre, les membres et la photo du groupe. Après leur création, ces groupes peuvent accueillir des personnes partageant des intérêts similaires, ce qui favorise les échanges et les collaborations.

- **Créer un groupe**

Les utilisateurs ont la possibilité de créer un groupe sur Midad afin de créer un espace dédié aux échanges en indiquant des détails tels que le titre, les membres et la photo du groupe.

- **Listes des groupes**

Les utilisateurs peuvent consulter la liste des groupes sur Midad afin de trouver et de s'intégrer à des communautés en fonction de leurs centres d'intérêt. Cette fonctionnalité facilite l'exploration des différents groupes disponibles.

- **Modifier un groupe**

Les utilisateurs peuvent modifier les informations du groupe sur Midad et enregistrer ces informations comme titre du groupe et liste des membres.

1.1.2 Exigences non fonctionnelles

Les exigences non fonctionnelles définissent toutes les conditions nécessaires pour garantir le bon fonctionnement du système et améliorer la qualité des services de l'application pour l'utilisateur. Notre système doit satisfaire aux exigences suivantes :

- **Ergonomie** : La plateforme doit présenter une interface claire, avec des formulaires assistés et des contrôles simples. Nous devons intégrer les éléments des interfaces d'une façon réutilisable et homogène, suivant les normes de l'expérience utilisateur optimale.
- **Scalabilité** : Le mode d'exécution ne doit pas dépendre d'un environnement spécifique, ou de capacité spécifique. Nous devons élaborer un plan de déploiement flexible qui permet de contrôler l'échelle à laquelle le projet est exécuté.
- **Sécurité** : La plateforme assure la protection des données contre les accès non autorisés, les pertes et les attaques. Pour ce faire, des mesures sensibles des utilisateurs sont mises en place. Une méthode de vérification par email et code sera intégrée. Lors de la connexion, un code sera envoyé à l'email de l'utilisateur. Celui-ci devra entrer ce code pour confirmer son identité et accéder à l'application, renforçant ainsi la sécurité des comptes utilisateurs. De plus, pour garantir une sécurité maximale des communications, un cryptage bout à bout (end-to-end encryption) sera implémenté, basé sur le chiffrement RSA assurant que seuls l'expéditeur et le destinataire peuvent lire les messages échangés.
- **Disponibilité** : La plateforme est conçue pour être mise en fonction depuis plusieurs sites pour garantir une disponibilité maximale en cas de mal fonctionnement de l'environnement hébergeant. Ce point doit être pris en compte dans le plan de déploiement. Par conséquent, nous devons implémenter la solution de « load balancing » pour distribuer le trafic entre plusieurs serveurs c'est-à-dire en cas de défaillance d'un serveur, le trafic est automatiquement redirigé vers les serveurs opérationnels.
- **Maintenabilité** : le code doit être bien documenté et facile à comprendre pour être efficace.

1.2. Spécification des exigences

1.2.1 Identification des acteurs

Dans notre projet nous avons identifié un ensemble de tâches qui sont associées à deux acteurs principaux et trois acteurs secondaires.

- **Abonné** : C'est un acteur principal qui est inscrit dans l'application de chat et à accès à plusieurs fonctionnalités comme l'envoi des messages, partage des fichiers et les appels instantanés.
- **Visiteur** : C'est un acteur principal qui peut faire l'inscription sur la plateforme Midad.
- **Système OTP** : C'est un acteur secondaire basé sur email, qui envoie des codes de vérification pour l'authentification.
- **Système de notification** : C'est un acteur secondaire qui envoie des notifications lors de la réception de messages, sur tous les systèmes d'exploitation.
- **WebRTC** : C'est un acteur secondaire utilisé pour les appels vidéo et audio en temps réel.

1.2.2 Identification des cas d'utilisation

Dans le suivant **tableau 2** nous allons identifier les cas d'utilisation du notre système.

Tableau 2 : Identification des cas d'utilisation [2]

#	Cas d'utilisation	Description	Acteur
Module de gestion de la messagerie			
1	Envoyer des messages et les recevoir	Permet à l'abonné d'envoyer et de recevoir des messages texte.	Abonné
2	Chercher conversation	Permet à l'abonné de rechercher des messages dans l'historique de la conversation.	Abonné
3	Modifier un message	Permet à l'abonné de modifier un message déjà envoyé.	Abonné
4	Supprimer un message	Permet à l'abonné de supprimer un message envoyé.	Abonné
5	Consulter conversation groupe	Permet à l'abonné de consulter les conversations de groupe.	Abonné
6	Consulter conversation ami	Permet à l'abonné de consulter les conversations avec des amis.	Abonné
7	Transférer un message	Permet à l'abonné de transférer un message reçu à un autre contact.	Abonné
8	Voir les contacts en ligne	Permet à l'abonné de voir quels contacts sont actuellement en ligne.	Abonné
9	Voir les statuts en ligne	Permet à l'abonné de voir les statuts en ligne de ses contacts lors de message.	Abonné
10	Ajouter un emoji	Permet à l'abonné d'ajouter et envoyer des emojis .	Abonné

#	Cas d'utilisation	Description	Acteur
11	Joindre fichiers	Permet à l'abonné de joindre des fichiers avec ses contacts.	Abonné
12	Envoyer des emojis	Permet à l'abonné d'envoyer des emojis dans la conversation.	Abonné
13	Consulter les notifications	Permet à l'abonné de consulter des notifications pour les nouveaux messages reçus. En fait appel à l'acteur secondaire de notification.	Abonné
14	Visualiser fichiers partagés	Permet à l'abonné de visualiser les fichiers partagés.	Abonné
15	Sélectionner un message	Permet à l'abonné de sélectionner un message.	Abonné
16	Afficher la liste des utilisateurs hors amis	Permet à l'abonné d'afficher la liste des abonnés qui ne sont pas dans sa liste d'amis.	Abonné
17	Gérer la conversation	Permet à l'abonné de gérer les paramètres de la conversation.	Abonné
Module de gestion des amis			
18	Ajouter des amis	Permet à l'abonné d'ajouter de nouveaux amis.	Abonné
19	Inviter des utilisateurs par midad	Permet à l'abonné d'inviter des abonnés via la plateforme Midad.	Abonné
20	Inviter des utilisateurs par email	Permet à l'abonné d'inviter des utilisateurs via email.	Abonné
21	Bloquer un ami	Permet à l'abonné de bloquer un ami.	Abonné
22	Gérer les contacts	Permet à l'abonné de gérer ses contacts.	Abonné
23	Rechercher des amis	Permet à l'abonné de rechercher des amis.	Abonné
24	Visualiser le profil	Permet à l'abonné de visualiser le profil.	Abonné
25	Modifier la photo de profil	Permet à l'abonné de modifier la photo de profil.	Abonné
26	Consulter les listes des amis.	Permet à l'abonné de consulter les listes d'amis.	Abonné
27	S'authentifier	Permet à l'abonné de s'authentifier en fait appel à l'acteur secondaire d'email.	Abonné
28	S'inscrire	Permet à un visiteur de s'inscrire sur la plateforme.	Visiteur

#	Cas d'utilisation	Description	Acteur
29	Valider email par code	Permet à l'abonné de valider son email via un code de vérification.	Abonné
Module de gestion des appels			
30	Lancer des appels audio/vidéo	Permet à l'abonné de lancer des appels audio et vidéo en utilisant le WebRTC.	Abonné
31	Participer à un appel	Permet à l'abonné de participer à un appel en cours.	Abonné
32	Refuser un appel	Permet à l'abonné de refuser un appel entrant.	Abonné
33	Accepter un appel	Permet à l'abonné d'accepter un appel entrant en utilisant le WebRTC.	Abonné
Module de gestion des groupes			
34	Créer un groupe	Permet à l'abonné de créer un nouveau groupe.	Abonné
35	Modifier un groupe	Permet à l'abonné de modifier les informations d'un groupe existant.	Abonné
36	Supprimer un membre	Permet à l'abonné de supprimer un membre existant.	Abonné

1.2.3 Classement des cas d'utilisation

Après avoir effectué l'identification des cas d'utilisation, nous pouvons maintenant les classer en tenant compte les facteurs suivants :

- La priorité fonctionnelle
- Le risque technique

Le **tableau 3** suivant illustre cette démarche sur notre étude de cas :

Tableau 3 : Classement des cas d'utilisation [3]

#	Cas d'utilisation	Priorité	Risque
1	Envoyer des messages et les recevoir	Haute	Haut
2	Supprimer un message	Haute	Moyen
3	Consulter conversation groupe	Haute	Moyen
4	Consulter conversation ami	Haute	Moyen

#	Cas d'utilisation	Priorité	Risque
5	Consulter les notifications	Haute	Moyen
6	Ajouter des amis	Haute	Moyen
7	S'authentifier	Haute	Moyen
8	S'inscrire	Haute	Moyen
9	Valider email par code	Haute	Moyen
10	Lancer des appels audio/vidéo	Haute	Moyen
11	Participer à un appel	Haute	Moyen
12	Refuser un appel	Haute	Moyen
13	Accepter un appel	Haute	Haut
14	Créer un groupe	Haute	Moyen
15	Chercher conversation	Moyen	Moyen
16	Modifier un message	Moyen	Moyen
17	Transférer un message	Moyen	Moyen
18	Voir les contacts en ligne	Moyen	Moyen
19	Voir les statuts en ligne	Moyen	Moyen
20	Visualiser fichiers partagés	Moyen	Moyen
21	Joindre fichiers	Basse	Basse
22	Afficher la liste des utilisateurs hors amis	Moyen	Basse
23	Gérer la conversation	Moyen	Moyen
24	Inviter des utilisateurs par midad	Moyen	Moyen
25	Inviter des utilisateurs par email	Moyen	Moyen
26	Bloquer un ami	Basse	Moyen
27	Rechercher des amis	Moyen	Moyen
28	Visualiser le profil	Moyen	Moyen
29	Gérer les contacts	Moyen	Moyen
30	Consulter les listes des amis	Moyen	Moyen
31	Modifier un groupe	Moyen	Moyen
32	Supprimer un membre	Moyen	Basse
33	Modifier la photo de profil	Moyen	Moyen
34	Sélectionner un message	Moyen	Moyen
35	Envoyer des emojis	Basse	Basse
36	Ajouter un emoji	Basse	Basse

1.2.4 Planification du projet en itérations

À partir du classement précédent, le chef de projet a proposé au comité de pilotage le découpage en itérations dans le tableau 4 suivant :

Tableau 4 : Planification des itérations grâce aux cas d'utilisation [4]

#	Cas d'utilisation	Priorité	Risque	Itération
1	Envoyer des messages et les recevoir	Haute	Haut	1
2	Supprimer un message	Haute	Moyen	1
3	Consulter conversation groupe	Haute	Moyen	1
4	Consulter conversation ami	Haute	Moyen	1
5	Consulter les notifications	Haute	Moyen	1
6	Ajouter des amis	Haute	Moyen	1
7	S'authentifier	Haute	Moyen	1
8	S'inscrire	Haute	Moyen	1
9	Valider email par code	Haute	Moyen	1
10	Lancer des appels audio/vidéo	Haute	Moyen	1
11	Participer à un appel	Haute	Moyen	1
12	Refuser un appel	Haute	Moyen	1
13	Accepter un appel	Haute	Haut	1
14	Créer un groupe	Haute	Moyen	1
15	Chercher conversation	Moyen	Moyen	2
16	Modifier un message	Moyen	Moyen	2
17	Transférer un message	Moyen	Moyen	2
18	Voir les contacts en ligne	Moyen	Moyen	2
19	Voir les statuts en ligne	Moyen	Moyen	2
20	Visualiser fichiers partagés	Moyen	Moyen	2
21	Joindre fichiers	Basse	Basse	2
22	Afficher la liste des utilisateurs hors amis	Moyen	Basse	2
23	Gérer la conversation	Moyen	Moyen	2
24	Inviter des utilisateurs par midad	Moyen	Moyen	2
25	Inviter des utilisateurs par email	Moyen	Moyen	2
26	Bloquer un ami	Basse	Moyen	2
27	Rechercher des amis	Moyen	Moyen	2
28	Visualiser le profil	Moyen	Moyen	2
29	Gérer les contacts	Moyen	Moyen	2

#	Cas d'utilisation	Priorité	Risque	Itération
30	Consulter les listes des amis	Moyen	Moyen	2
31	Modifier un groupe	Moyen	Moyen	2
32	Supprimer un membre	Moyen	Basse	2
33	Modifier la photo de profil	Moyen	Moyen	3
34	Sélectionner un message	Moyen	Moyen	3
35	Envoyer des emojis	Basse	Basse	3
36	Ajouter un emoji	Basse	Basse	3

1.2.5 Diagrammes des cas d'utilisation

En continuant notre méthode structurée, semblable à celle que nous avons adoptée lors de la définition de nos besoins fonctionnels. Pour représenter le diagramme de cas d'utilisation, nous employons la structure modulaire afin de faciliter la visualisation des interactions et des scénarios propres à chaque fonctionnalité.

De plus, la visualisation simplifiée des cas d'utilisation de notre système met l'accent sur la façon dont les utilisateurs interagissent avec la demande. Ces diagrammes présentent divers cas d'utilisation offrant un aperçu concis de toutes les fonctionnalités prises en charge. Dans notre contexte, où un seul utilisateur est impliqué, la mise en évidence de la simplicité et la clarté des interactions avec les utilisateurs deviennent particulièrement pertinentes, soulignant l'efficacité et la conception centrée sur l'utilisateur de l'application.

A. Diagramme de cas d'utilisation globale

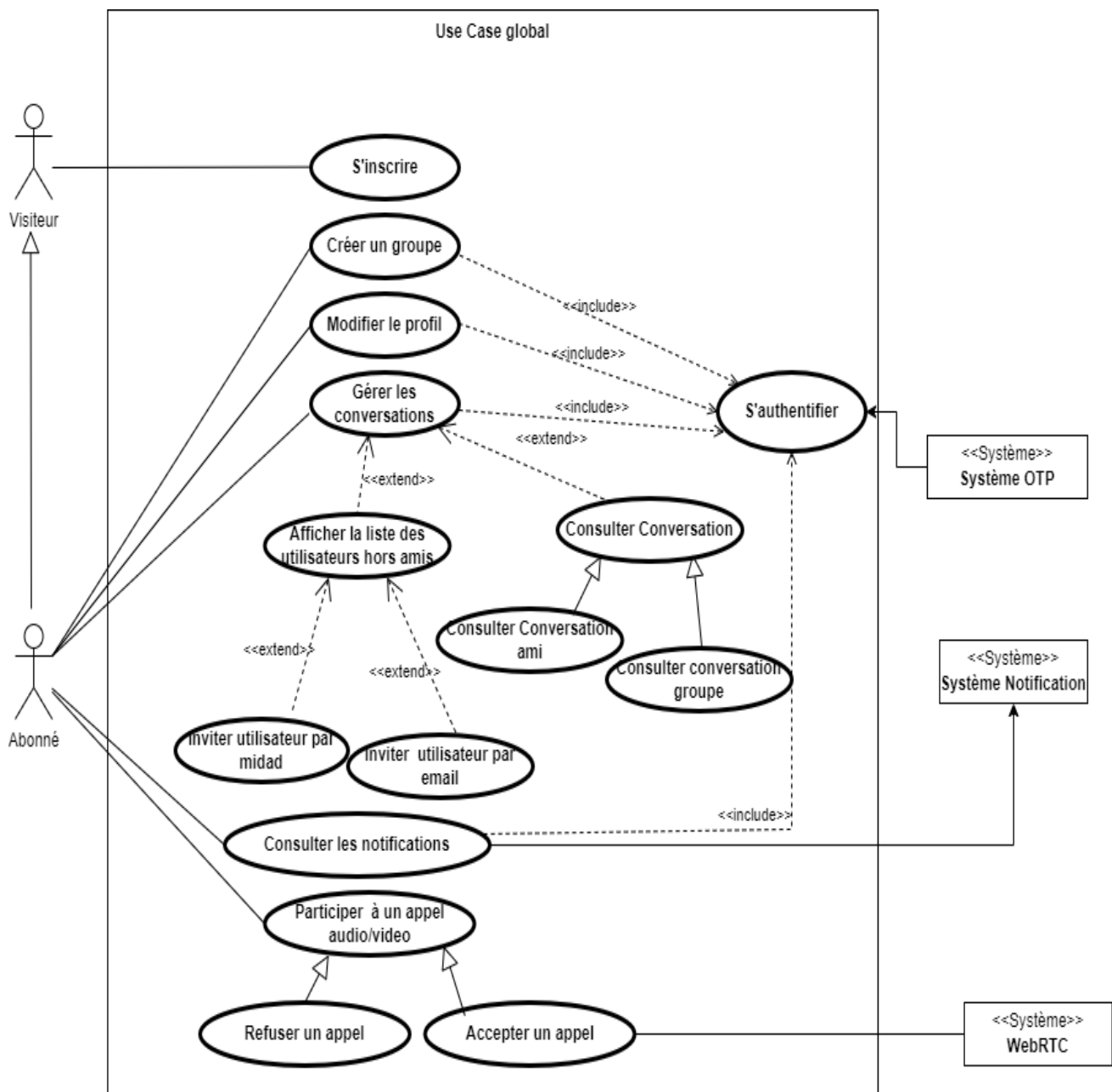


Figure 8 : Diagramme de cas d'utilisation globale

B. Diagramme de cas d'utilisation du module messagerie

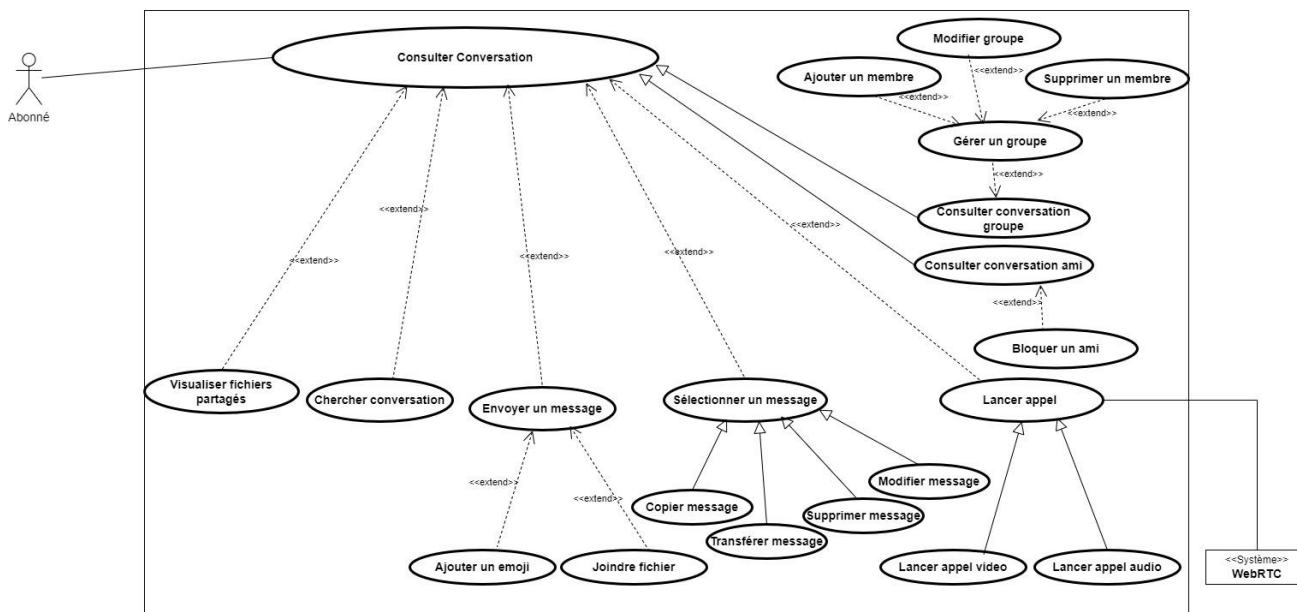


Figure 9 : Diagramme de cas d'utilisation du module messagerie

1.2.6 Description textuelle des cas d'utilisation majeurs

A. S'inscrire

Le tableau 5 ci-dessous représente la description textuelle du cas d'utilisation « S'inscrire », détaillant les étapes et interactions nécessaires pour permettre à un utilisateur de créer un nouveau compte sur la plateforme Midad.

Tableau 5 : Description textuelle de cas d'utilisation « S'inscrire »

Objectif	Permettre à un visiteur de créer un nouveau compte sur la plateforme Midad.
Acteur	Visiteur
Précondition	Le visiteur doit accéder à la page d'inscription.
Postcondition	Le visiteur a un compte actif sur la plateforme et peut se connecter avec ses identifiants.
Scénario nominal	<ol style="list-style-type: none"> 1. Le visiteur accéder à l'interface d'inscription. 2. Il saisit son nom complet, adresse e- mail et photo de profil. 3. Le visiteur clique sur le bouton « S'inscrire ». 4. Le système vérifie la validité des informations. 5. Le compte est activé 6. Le visiteur se connecte avec succès.
Scénario alternatif	<ol style="list-style-type: none"> 2a. Si email invalide ou image n'existe pas, le système envoie un message d'erreur et les informations invalides. 4b. L'adresse e-mail est déjà utilisée : le système informe l'utilisateur et demande une adresse différente.

B. Envoyer un message

Le tableau 6 ci-dessous représente la description textuelle du cas d'utilisation « Envoyer un message », détaillant les étapes et interactions nécessaires pour permettre à l'abonné d'envoyer des messages sur la plateforme Midad.

Tableau 6 : Description textuelle de cas d'utilisation « Envoyer un message »

Objectif	Permettre à un abonné d'envoyer des messages sur la plateforme Midad.
Acteur	Abonné
Précondition	L'abonné doit être authentifié et avoir accès à la plateforme Midad.
Postcondition	Le message est envoyé avec succès et visible par l'autre abonné
Scénario nominale	<ol style="list-style-type: none">1. L'abonné accède à l'interface de conversation.2. L'abonné saisit le message3. L'abonné clique sur le bouton « envoyer »4. Le système envoie le message.5. Le message est affiché dans la conversation.
Scénario alternatif	<ol style="list-style-type: none">1a. L'utilisateur tente d'envoyer un message sans être connecté.1b. Le système demande à l'utilisateur de se connecter.2a. L'utilisateur essaie d'envoyer un message sans connexion internet.2b. Le système affiche un message d'erreur et demande de réessayer plus tard.

C. Consulter Conversation

Le tableau 7 ci-dessous représente la description textuelle du cas d'utilisation « Consulter Conversation ».

Tableau 7 : Description textuelle de cas d'utilisation « Consulter conversation »

Objectif	Permettre à l'abonné de consulter la conversation sur la plateforme Midad.
Acteur	Abonné
Précondition	L'abonné doit être authentifié et avoir accès à la plateforme Midad.
Postcondition	La conversation est affichée et consultée par l'abonné.
Scénario nominal	<ol style="list-style-type: none">1. L'utilisateur accède à la plateforme Midad.2. L'utilisateur sélectionne une conversation.3. Le système charge et affiche la conversation consultée.

Scénario alternatif	1a. L'abonné tente de consulter une conversation sans être connecté. 1b. Le système demande à l'abonné de se connecter. 2a. L'utilisateur essaie de consulter une conversation qui n'existe pas. 2b. Le système affiche un message d'erreur.
----------------------------	---

D. Gérer un groupe

Le tableau 8 ci-dessous représente la description textuelle du cas d'utilisation « Gérer un groupe ».

Tableau 8 : Description textuelle de cas d'utilisation « Gérer un groupe »

Objectif	Permettre à un utilisateur de modifier, supprimer, et gérer les membres d'un groupe sur Midad.
Acteur	Abonné
Précondition	L'abonné doit être authentifié et avoir accès à la plateforme Midad.
Postcondition	Le groupe est modifié, supprimé ou ajouter les membres sont gérés avec succès.
Scénario nominal	1. L'abonné accède à l'application Midad. 2. Le système vérifie la connexion d'abonné. 3. L'abonné choisit de modifier, ou supprimer un membre. 4. L'abonné saisit ou modifie les informations. 5. Le système enregistre les modifications et affiche une confirmation.
Scénario alternatif	3a. L'utilisateur essaie de créer un groupe sans fournir les informations nécessaires. 3b. Le système affiche un message d'erreur et demande de compléter les informations manquantes. 4a. L'utilisateur essaie de modifier un groupe qui n'existe pas. 4b. Le système affiche un message d'erreur.

E. S'authentifier

Le tableau 9 ci-dessous représente la description textuelle du cas d'utilisation « S'authentifier ».

Tableau 9 : Description textuelle de cas d'utilisation « S'authentifier »

Objectif	Permettre à un utilisateur de se connecter à la plateforme Midad.
Acteur	Visiteur ou abonné
Précondition	L'abonné doit avoir un compte valide et accès à l'interface de connexion.
Postcondition	L'abonné est authentifié et a accès à la plateforme Midad.
Scénario nominal	<ol style="list-style-type: none">1. L'abonné accède à l'interface de connexion.2. L'abonné saisit son email.3. L'abonné clique sur « Se connecter ».4. Le système OTP envoie un code de vérification par email.5. L'abonné saisit le code de vérification reçu par email.6. Le système vérifie le code de vérification.7. Si le code est correct, l'utilisateur est redirigé vers la page d'accueil.8. L'abonné connecté avec succès.
Scénario alternatif	<ol style="list-style-type: none">3a. L'abonné saisit un code de vérification invalide.3b. Le système affiche un message d'erreur indiquant que le code est incorrect.

1.2.7 Diagrammes de séquence système

Le cas d'utilisation décrit les interactions des acteurs avec le site web que nous souhaitons définir et créer. Au cours de ces échanges, les acteurs génèrent des messages qui impactent le système informatique et suscitent habituellement une réponse de celui-ci. Nous allons séparer ces messages et les illustrer visuellement à l'aide de diagrammes de séquence UML. Dans cette partie, nous exposons les diagrammes de séquence système des principaux cas d'utilisation, accompagnés d'une description textuelle approfondie.

A. Diagramme de séquence système « S'inscrire »

Ce diagramme de séquence de la figure 10 illustre le processus d'inscription d'un utilisateur au système Midad. Le flux commence lorsque l'utilisateur accède à l'interface d'inscription. Le système affiche alors l'interface appropriée. Une boucle est mise en place pour gérer les éventuelles erreurs lors de la saisie des informations. L'utilisateur entre les données requises (email, nom complet et image de profil) et les valide. Le système vérifie ensuite ces informations. Si l'email est invalide ou si la photo de profil est manquante, le système envoie un message d'erreur à l'utilisateur, et la boucle recommence pour permettre une nouvelle saisie.

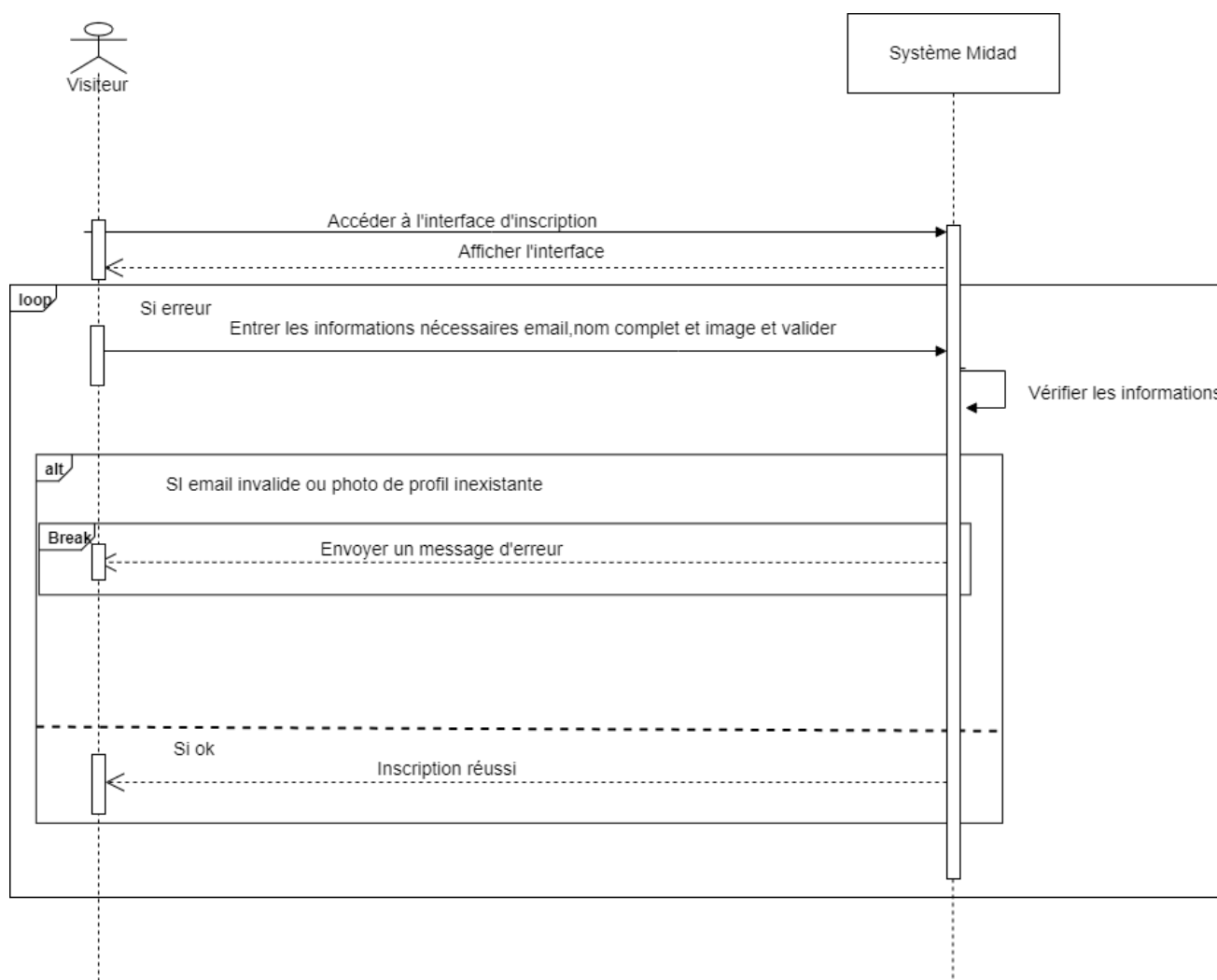


Figure 10 : Diagramme de séquence de système « S'inscrire »

Si toutes les informations sont correctes, le système confirme que l'inscription a réussi. Ce diagramme met en évidence l'interaction entre l'utilisateur et le système, ainsi que les différentes étapes de validation nécessaires pour assurer une inscription réussie et sécurisée.

B. Diagramme de séquence système « S'authentifier »

Ce diagramme de séquence illustre le processus de connexion au système Midad. Il débute lorsque l'utilisateur accède à l'interface de connexion, que le système affiche ensuite. Une boucle est mise en place pour gérer les potentielles erreurs. L'utilisateur saisit son email, et le système envoie un code de vérification par email. Si le code est incorrect, le système envoie un message d'erreur et la boucle recommence, permettant à l'utilisateur de réessayer. Si le code est correct, le processus se termine avec un message confirmant que la connexion est réussie. Ce diagramme met en évidence l'interaction entre l'utilisateur et le système, ainsi que le mécanisme de sécurité utilisant un code de vérification par email pour authentifier l'utilisateur.

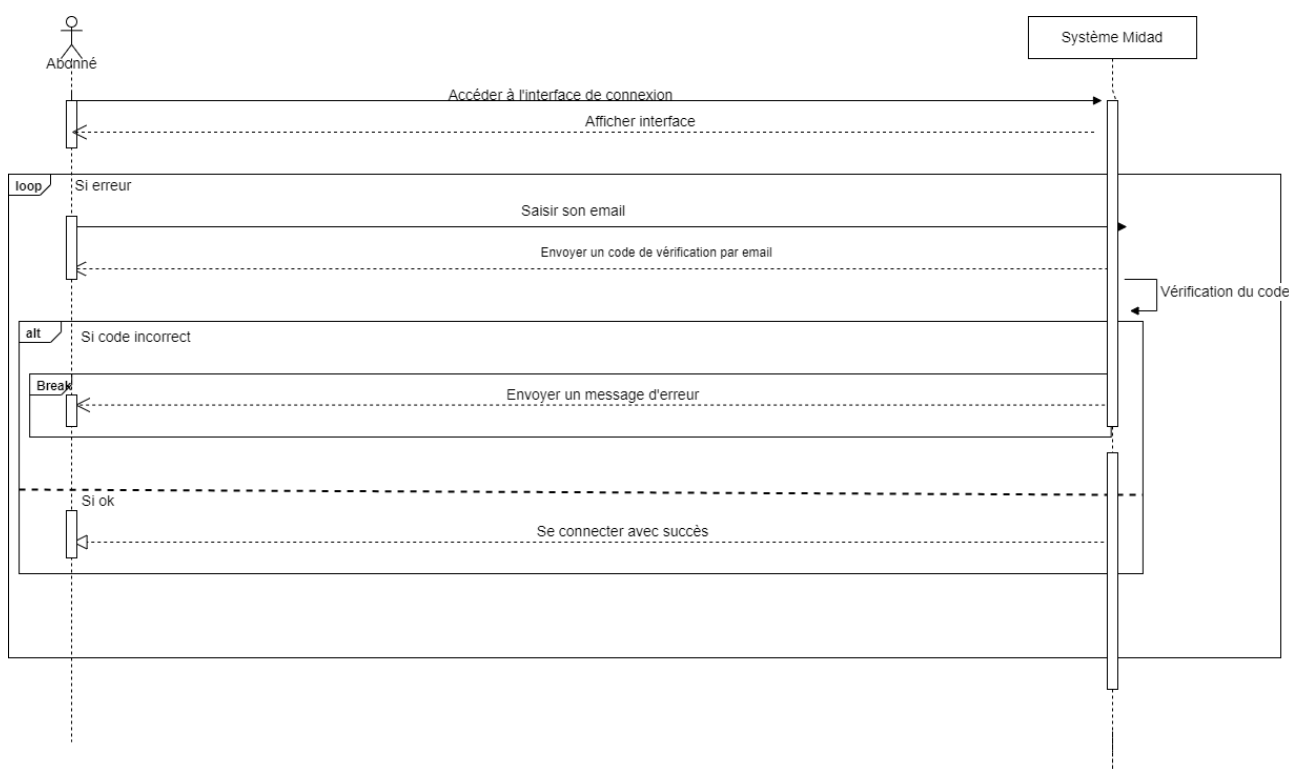


Figure 11 : Diagramme de séquence système « S'authentifier »

C. Diagramme de séquence système « Consulter un conversation »

Ce diagramme de séquence illustre les interactions entre un abonné et le système Midad pour accéder et utiliser les fonctionnalités de messagerie. Le processus commence par une étape obligatoire d'authentification de l'abonné. Ensuite, le diagramme présente deux actions optionnelles : la recherche d'un ami et la visualisation d'une pièce jointe. Dans le cas de la recherche d'un ami, l'abonné peut sélectionner un autre abonné, ce qui déclenche l'affichage de la conversation correspondante par le système. Ces étapes optionnelles sont encadrées par des fragments "opt" (optionnel), indiquant qu'elles

peuvent être exécutées ou non selon le choix de l'utilisateur. Ce diagramme met en évidence la flexibilité du système, permettant à l'abonné de naviguer entre différentes fonctionnalités après s'être authentifié.

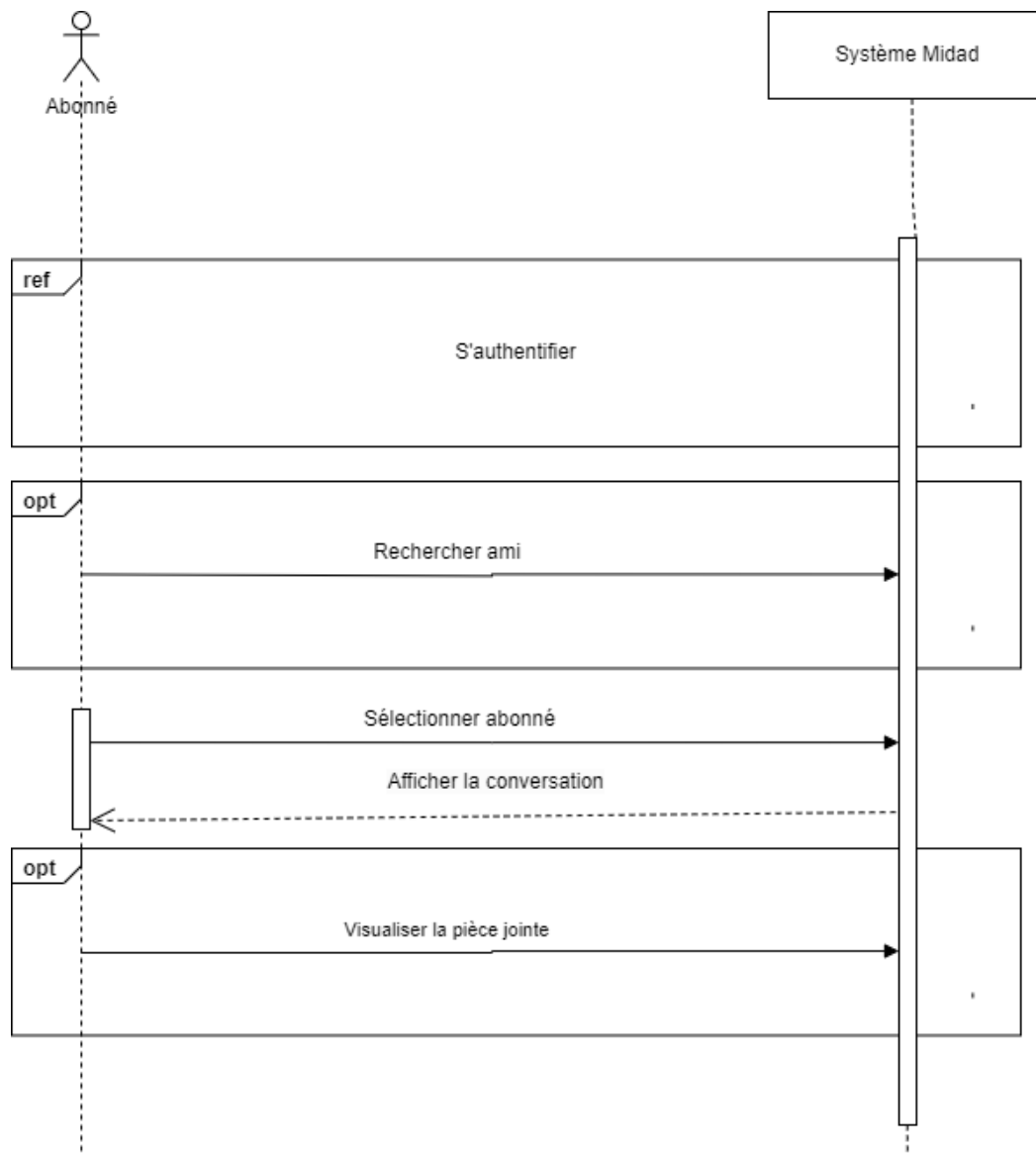


Figure 12 : Diagramme de séquence système « Consulter conversation »

D. Diagramme de séquence système « Envoyer un message »

Ce diagramme de séquence illustre les interactions entre un utilisateur et le système Midad pour la gestion des conversations. Le processus commence par deux étapes obligatoires, indiquées par le fragment "ref" : l'authentification de l'utilisateur et la consultation d'une conversation. Ensuite, l'utilisateur peut rédiger un message, que le système envoie dans la conversation. Le diagramme présente également deux actions optionnelles, encadrées par des fragments "opt" : la possibilité de

joindre des fichiers au message et l'option d'envoyer un emoji. Cette structure met en évidence la flexibilité du système, permettant à l'utilisateur de personnaliser ses messages avec des pièces jointes ou des emojis, tout en maintenant un flux de base simple pour l'envoi de messages textuels.

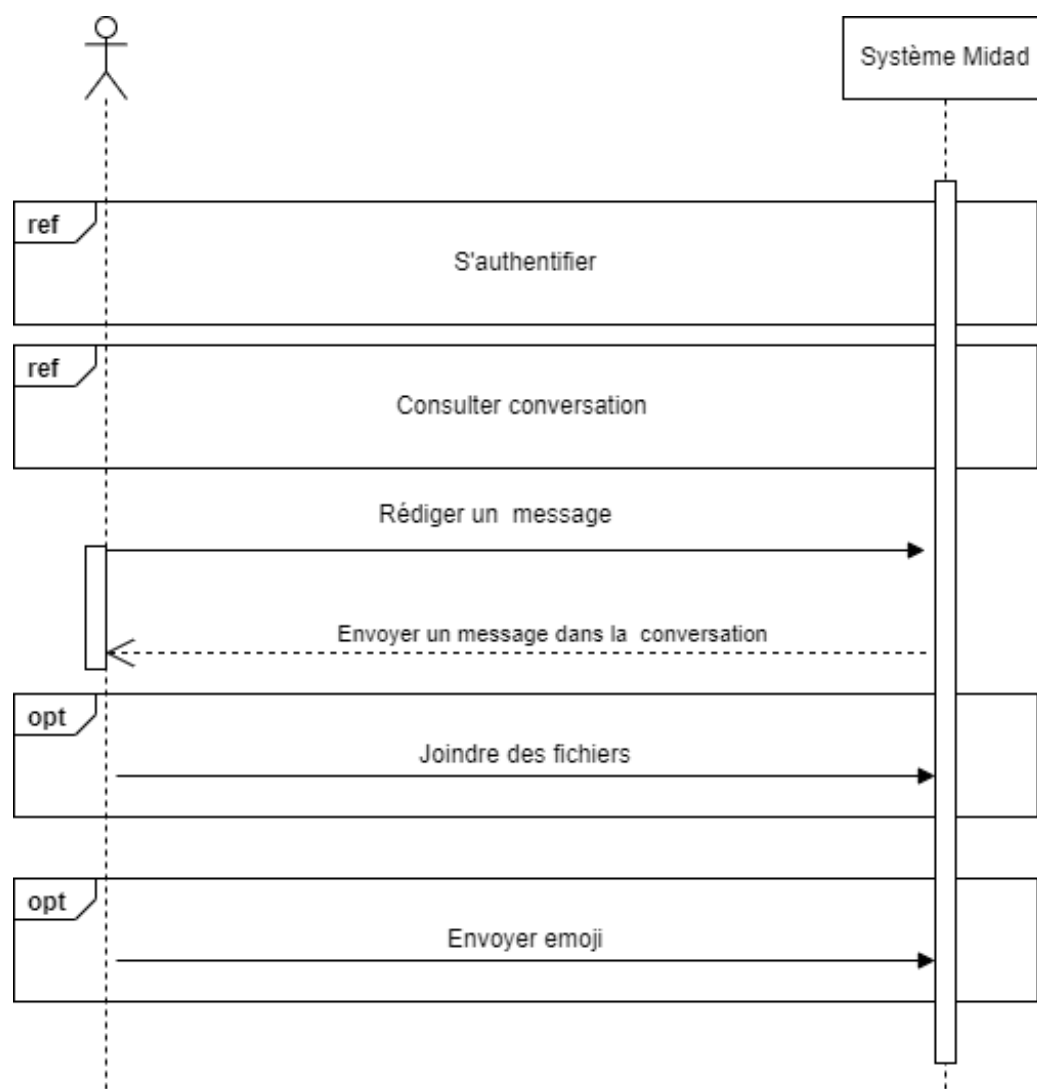


Figure 13 : Diagramme de séquence système « Envoyer un message »

E. Diagramme de séquence système « Gérer un groupe »

Ce diagramme de séquence illustre les interactions entre un abonné et le système Midad pour la gestion des groupes de discussion. Le processus débute par une authentification obligatoire, suivie d'une étape optionnelle de recherche d'ami. L'abonné peut ensuite sélectionner un groupe, ce qui déclenche l'affichage des détails du groupe par le système. Le diagramme présente ensuite une boucle contenant trois actions optionnelles : ajouter un membre, supprimer un membre, et modifier le groupe. Cette structure met en évidence la flexibilité offerte à l'abonné pour gérer les groupes, permettant des modifications répétées selon les besoins. Le diagramme souligne ainsi les fonctionnalités principales

de gestion de groupe dans le système Midad, offrant une vue claire des interactions possibles entre l'utilisateur et le système dans ce contexte.

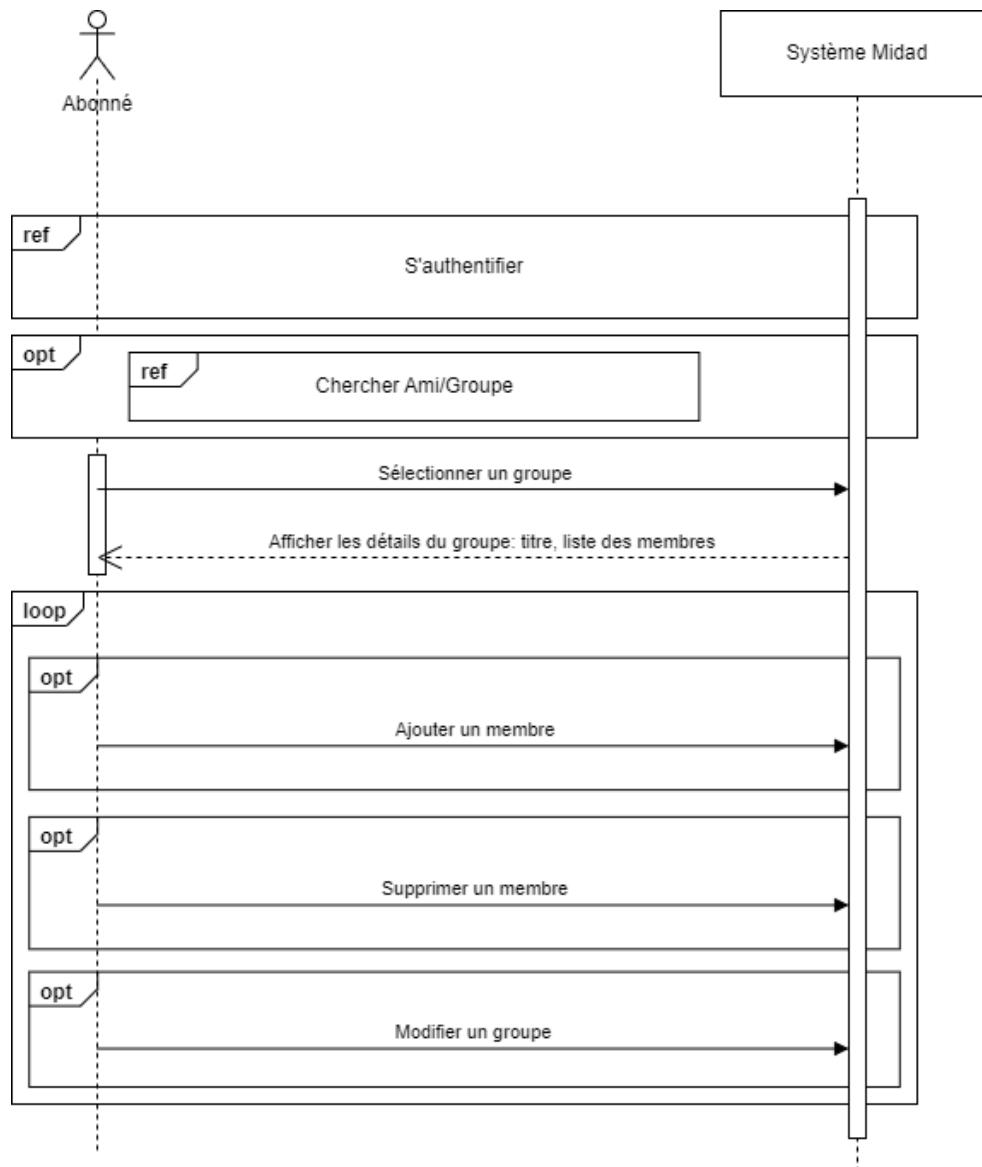


Figure 14 : Diagramme de séquence de système « Gérer un groupe »

Conclusion

Dans ce chapitre, nous avons repéré les différents acteurs qui vont utiliser notre système, expliqué les exigences fonctionnelles et non fonctionnelles qu'il doit satisfaire, et décrit les interactions qui seront établies entre l'acteur et l'application. Au cours du prochain chapitre, nous examinerons la phase de conception de la solution.

CHAPITRE 3 : ANALYSE ET CONCEPTION OBJET

Introduction

Rappelons le positionnement de cette activité de conception par rapport à l'ensemble du processus décrit au **chapitre 1**. Nous avons identifié les cas d'utilisation et leur description détaillée au **chapitre 2**, grâce en particulier aux diagrammes de séquence système. Pour passer maintenant à la conception, nous allons décrire le diagramme de classe pour passer à détailler une exploitation supplémentaire de la maquette, et passer enfin à la description des interactions correspondantes par les diagrammes de séquence (diagramme de séquence détaillé).

1.1. Diagrammes de séquence conception

1.1.1 Diagramme de séquence conception « S'inscrire »

Le diagramme de séquence de la conception « S'inscrire » illustre les interactions entre le visiteur, le système Midas, et les différents composants pour finaliser l'inscription d'un nouvel utilisateur. Les étapes comprennent la soumission des informations d'inscription, la vérification de l'unicité des données, et la création d'un compte dans la base de données, comme illustré dans la figure 15.

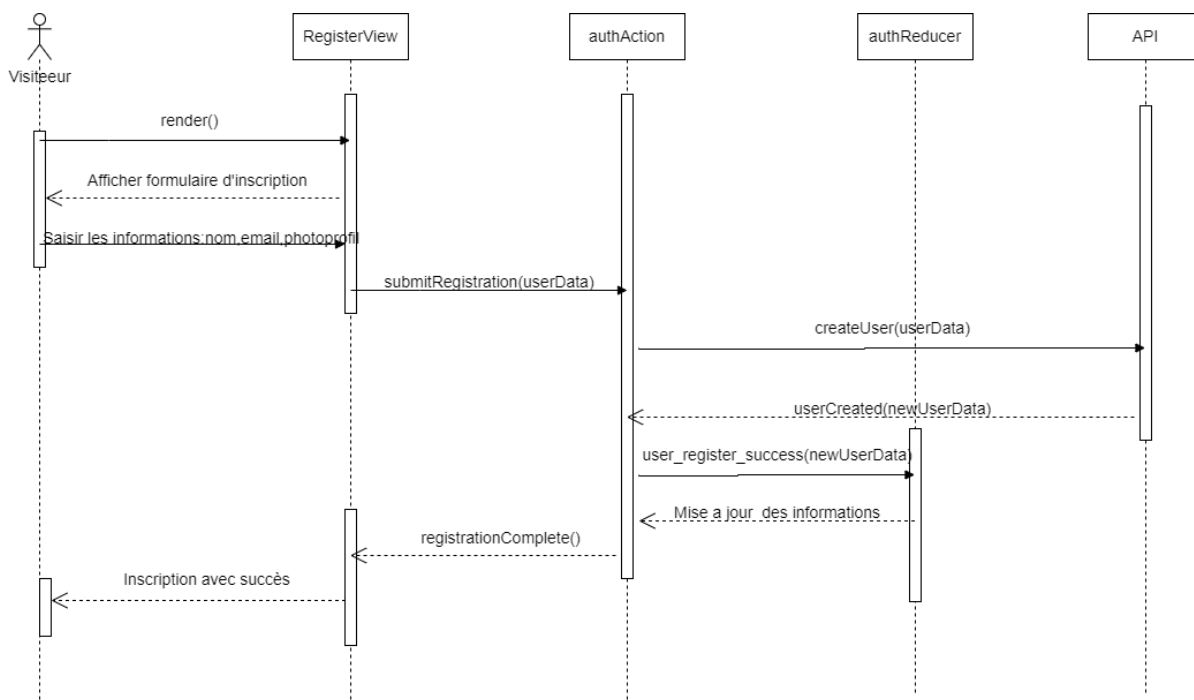


Figure 15 : Diagramme de séquence conception du scénario nominal « S'inscrire »

1.1.2 Diagramme de séquence conception « Envoyer un message »

Le diagramme de séquence de la conception « Envoyer un message », comme illustré dans la figure 16.

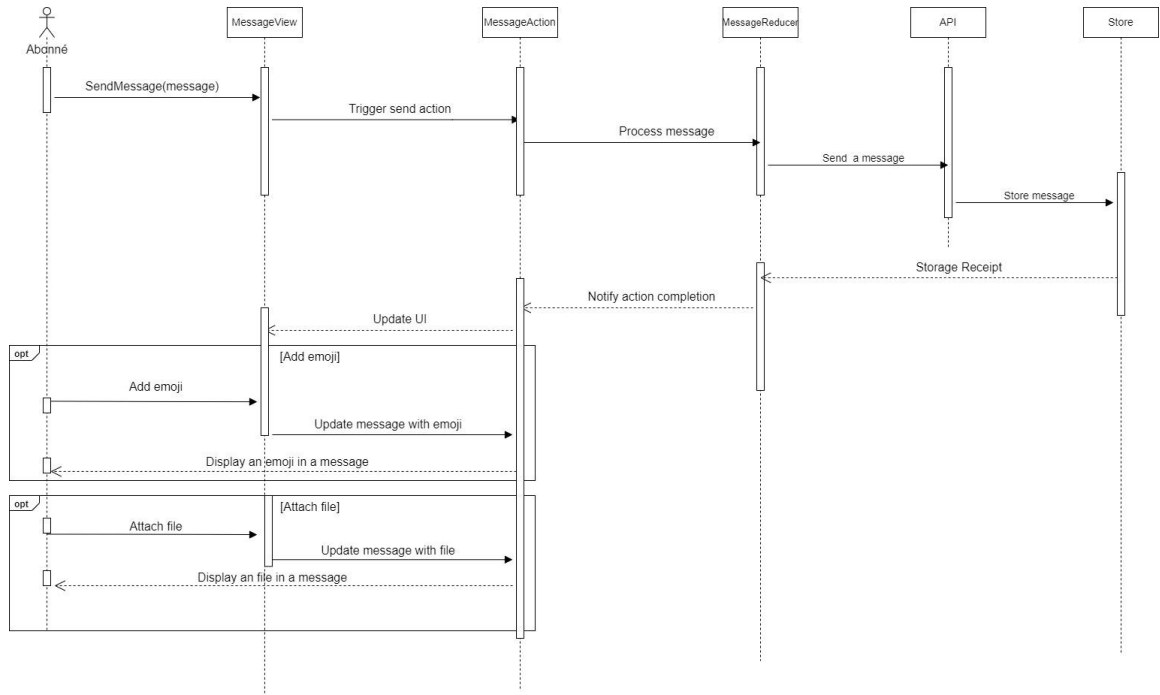


Figure 16 : Diagramme de séquence de la conception du scénario nominal « Envoyer un message »

1.1.3 Diagramme de séquence conception « Consulter conversation »

Le diagramme de séquence de la conception « Consulter conversation », comme illustré dans la figure 17.

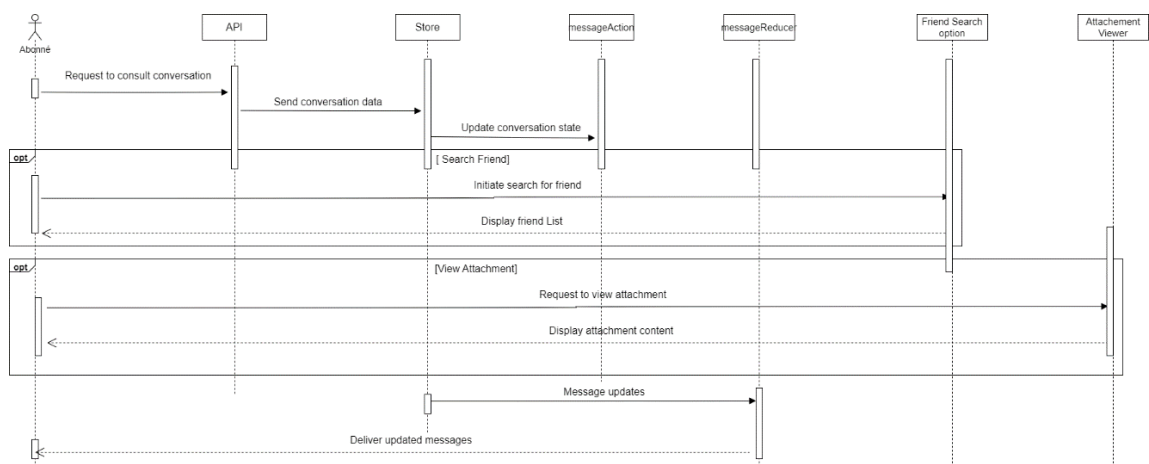


Figure 17 : Diagramme de séquence de la conception du scénario nominal « Consulter conversation »

1.1.4 Diagramme de séquence conception « Gérer un groupe »

Le diagramme de séquence de la conception « Gérer un groupe », comme illustré dans la figure 18, contient trois actions principales : addMember, removeMember et updateGroup. Actuellement, nous nous concentrons spécifiquement sur l'ajout d'un membre.

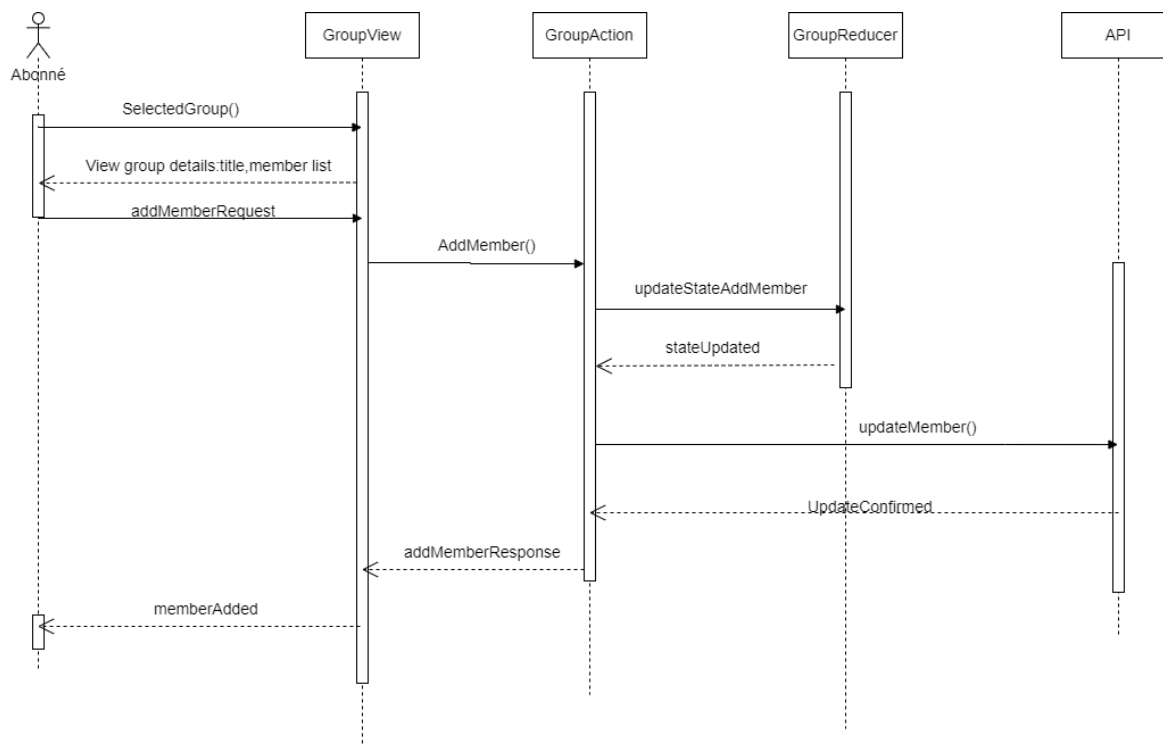


Figure 18 : Diagramme de séquence de la conception du scénario nominal « Ajouter un membre »

1.1.5 Diagramme de séquence conception « S'authentifier »

Ce diagramme de séquence illustre le processus d'authentification d'un abonné dans un système. Il met en évidence les interactions entre l'Abonné, le composant Login, authAction, authReducer et l'API.

Le flux commence lorsque l'Abonné accède à la page de connexion. Le formulaire s'affiche, permettant la saisie de l'email. Une vérification de l'existence de l'utilisateur est effectuée via authAction. Si l'utilisateur existe, un code de vérification est envoyé par email. L'Abonné saisit ensuite ce code, qui est vérifié par l'API via authAction. Une fois le code OTP validé, une demande de connexion est initiée. AuthAction communique avec authReducer pour mettre à jour l'état de connexion (user_login_success). Enfin, l'Abonné est informé du succès de la connexion.

Ce diagramme met en lumière l'utilisation d'une authentification à deux facteurs (email + code OTP) et l'architecture du système, qui sépare les responsabilités entre la présentation (Login), la logique métier (authAction), la gestion de l'état (authReducer) et les opérations côté serveur (API). Cette approche renforce la sécurité et la modularité de l'application, comme illustré dans la figure 19.

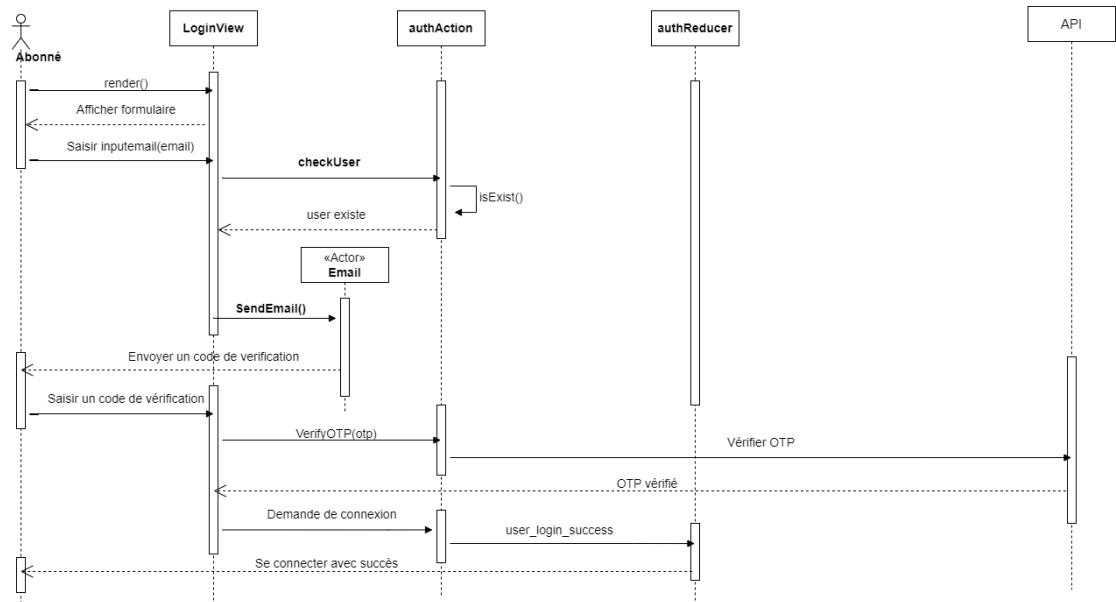


Figure 19 : Diagramme de séquence de la conception du scénario nominal « S'authentifier »

1.2. Modélisation diagramme de classe

Afin de continuer notre approche, nous allons maintenant examiner le diagramme de classes générales de notre application. Grâce à cette étape, nous pourrions obtenir une vision globale de la structure de notre application et mettre en évidence les divers modules qui la composent.

En parallèle, nous poursuivrons la définition des diagrammes de classe d'entités, en donnant une description approfondie des classes et des attributs qui pourraient susciter des confusions. Cette méthode nous permettra d'acquérir une meilleure compréhension de la structure interne de notre application et de clarifier les interactions entre les différentes catégories. Le figure 20 ci-dessous présente le diagramme de classe global

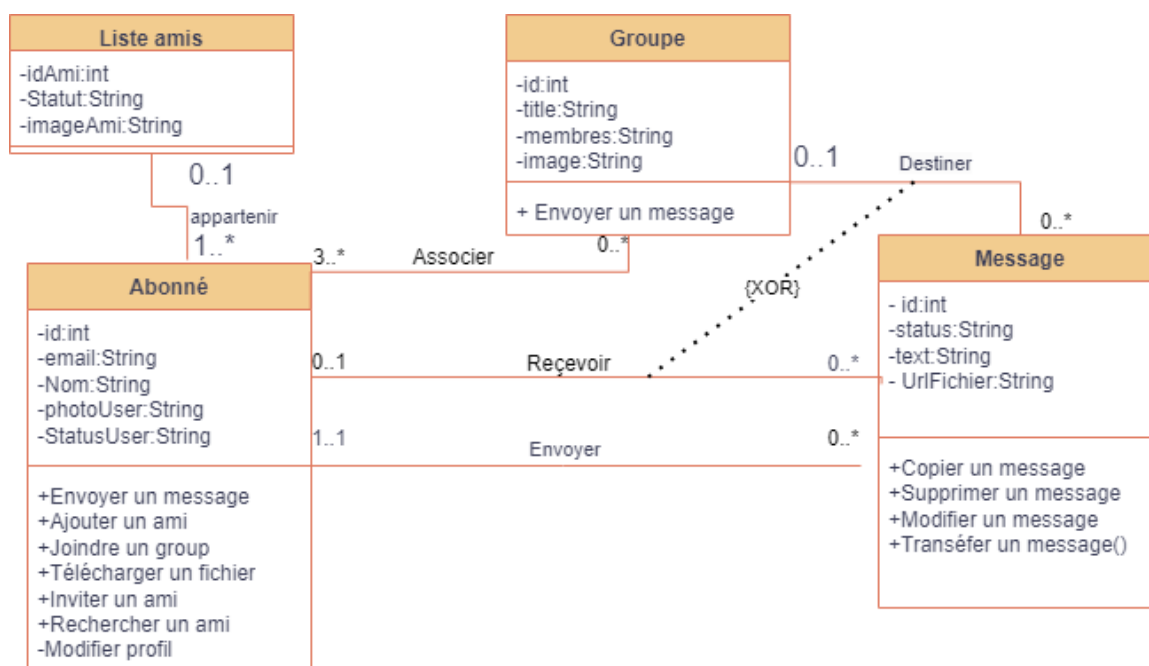


Figure 20 : Diagramme de classes

Le tableau 10 ci-dessous clarifie le diagramme de classe en fournissant des descriptions détaillées des classes.

Tableau 10 : Description de diagramme de classes

Classe	Description classe
Abonné	Représente un utilisateur de l'application.il possède des informations de base comme email, nom complet et la photo de profil .il peut envoyer des messages ajouter des amis, télécharger des fichiers et rechercher des amis.
Liste amis	Représente un ami de l'utilisateur. Stocke des informations sur le statut de l'amitié et l'image de l'ami.
Groupe	Représente un groupe auquel les utilisateurs peuvent appartenir. Contient des informations sur le titre, les membres et l'image du groupe. Permet d'envoyer des messages au groupe.
Message	Représente un message envoyé par un utilisateur. Contient le statut et le texte du message. Permet des opérations comme copier, supprimer, modifier et transférer le message.

Présentons ensuite une description des attributs qui peuvent être non clairs dans le *tableau 11* ci-dessous :

Tableau 11 : Description des attributs du diagramme de classes

Classe	Attribut	Description Attribut
Message	id	Identifiant unique du message.
	UrlFichier	Lien du fichier
	status	Statut du message (envoyé, reçu, lu, etc.).
	Text	Contenu texte du message.
Groupe	id	Identifiant unique du groupe.
	titre	Titre du groupe.
	membres	Liste des membres du groupe.
	image	L'image du groupe.
Liste Amis	idAmi	Identifiant unique de l'ami.
	status	Statut de l'ami (ami, bloqué, etc.).
	imageAmi	L'image de l'ami.
Abonné	id	Identifiant unique de l'utilisateur.
	Email	Adresse email de l'utilisateur.
	Nom complet	Nom complet de l'utilisateur.
	PhotoAbonné	La photo de l'utilisateur.
	StatusAbonné	Statut de l'utilisateur (en ligne, hors ligne, etc.).

1.3. Modélisation non relationnelle

Dans le cadre de ce projet, comme pour toute autre application, la base de données n'est pas conçue à partir de zéro. Elle doit plutôt être générée à partir du diagramme de classe de l'application, en appliquant des règles et des choix pour passer du modèle orienté objet au modèle non relationnel. La figure suivante (figure 21) présente le modèle non relationnel de notre application.



Figure 21 : Modélisation non relationnelle

Conclusion

Dans ce chapitre, nous avons exposé la mise en œuvre de diagramme de classe. Dans le chapitre suivant, qui sera le dernier, nous allons démontrer davantage la partie réalisation de notre application à savoir l'environnement de travail, l'architecture physique et logique, le déploiement ainsi que les jeux d'essai.

CHAPITRE 4 : PRESENTATION TECHNIQUE

Introduction

Dans ce dernier chapitre, nous examinons tout d'abord les choix technologiques qui ont orienté le développement de notre projet. Cela inclut une vue d'ensemble du modèle de développement que nous avons choisi, de l'architecture logicielle développée et de l'environnement technique utilisé.

Par la suite, notre attention se porte sur le processus DevOps, qui constitue le fondement de notre approche de gestion de projet. Nous exposons notre méthode de gestion du code source, en utilisant l'outil Git. Il est également possible de comprendre nos méthodes d'intégration et de déploiement continus en utilisant nos pipelines de CI/CD (Intégration Continue/Déploiement Continu), ainsi que nos pratiques de tests continus, qui incluent les tests unitaires et d'intégration. Nous concluons en présentant le travail réalisé à travers une série d'aperçus d'écran, mettant en évidence les différentes fonctionnalités que nous avons mises en place dans l'application.

1.1. Choix technologiques

Dans cette première partie, nous exposons le schéma de développement choisi pour le projet, ainsi que l'architecture logicielle et l'environnement technique déployés. Nous détaillons les outils et les technologies utilisés pour le développement et la mise en production de l'application, en expliquant leur rôle dans la réalisation de nos objectifs. En outre, nous examinons certains aspects du développement sur lesquels nous avons pris soin de nous focaliser tout au long de la création de notre application.

1.1.1 Architecture générale

Notre système est basé sur une architecture minutieusement choisie : l'architecture à trois niveaux. Cette architecture assure non seulement sa robustesse, mais aussi sa flexibilité et son évolutivité. Elle représente le fondement de notre application, ce qui simplifie l'intégration et la gestion des diverses composantes du système en les divisant en trois niveaux distincts : la présentation, la logique métier et l'accès aux données.

Cette architecture est basée sur deux des modèles de conception les plus couramment utilisés : le Modèle-Vue-Contrôleur (MVC) et le Modèle redux. Ces modèles n'ont pas été choisis par hasard, car ils ont été sélectionnés en raison de leurs multiples bénéfices, notamment en ce qui concerne la séparation des préoccupations et l'amélioration de la testabilité du code. En outre, l'alliance entre le MVC et le redux renforce l'architecture globale de notre système. Notre application bénéficie d'une structure solide grâce à cette association, ce qui assure une expérience utilisateur fluide et une maintenance facile. Dans cette section, nous n'aborderons que les deux grands serveurs frontend et backend.

A. Serveur Frontend

L'architecture interne du serveur frontend est représentée dans la figure 22 ci-dessous.

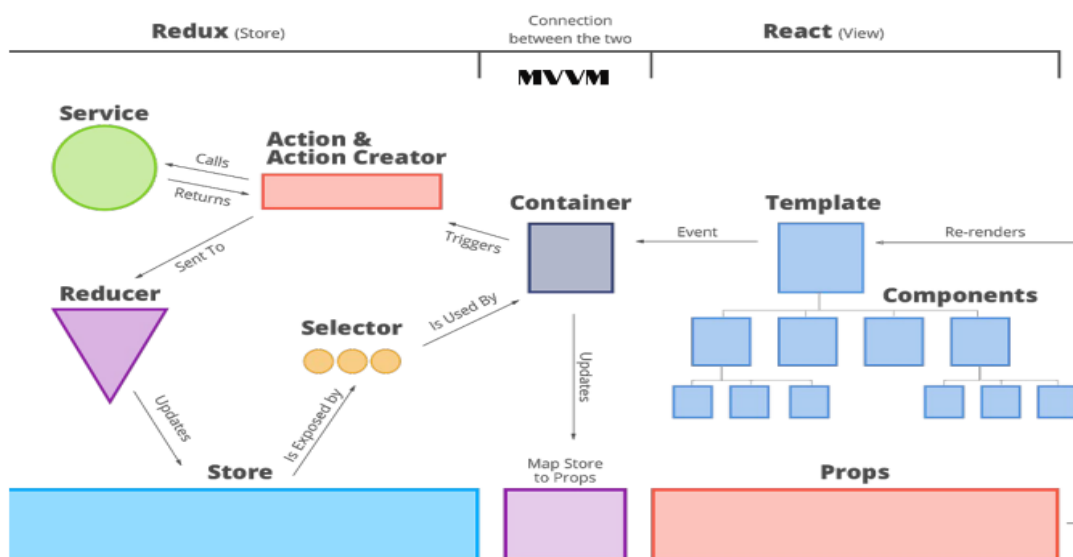


Figure 22 : Le serveur frontend [7]

Maintenant, nous avons expliqué mieux les composants de l'architecture interne dans un tableau 12 qui définit et décrit toutes les interactions entre les éléments mentionnés dans la figure 22.

Tableau 12 : Un aperçu de l'architecture frontend [5]

Composant	Explication	Interaction
Service	La couche de service gère les appels API et renvoie les données au front-end. Il agit comme un intermédiaire entre le frontend et le backend, faisant abstraction des complexités de communication back-end.	Le frontend envoie des appels API au service, qui ensuite renvoie les données requises
Reducers	C'est le cerveau de Redux. C'est là que l'on va mettre la logique de notre application. Un Reducer Redux est une fonction qui reçoit le state et une action en paramètre, et qui retourne un nouveau state. Un premier state et une action sont envoyés au reducer, qui retourne un nouveau state	Lorsque les actions sont expédiées, les réducteurs mettent à jour l'état dans le magasin en conséquence.
Store	Permet de stocker des données de manière centralisée et partagées entre tous les composants de notre application. Auparavant, les dépendances tierces comme Flux, Redux ou Mobx permettaient de gérer les données et coordonner les composants de nos applications complexes.	Les actions envoient des données au store, et le store met à jour son état via des réducteurs. La mise à jour l'État est alors mis à la disposition des composants.
Actions	Ce sont des charges utiles d'informations qui envoyer des données de l'application au magasin(store). Ils sont le principal moyen d'envoyer des données de l'application vers la boutique Redux.	Les actions sont déclenchées par l'utilisateur interactions ou autres événements, envoyer des demandes d'API à Service et expédition réponses au magasin.

Composant	Explication	Interaction
Child Components	<p>Les composants enfants sont des sous-composants qui reçoivent des accessoires de leurs composants parents.</p> <p>Ils contribuent à fragmenter l'interface utilisateur en plus petites parties, des pièces réutilisables.</p>	La commande accessoires fournie par les composants enfants permet de rendre des parties spécifiques de l'interface utilisateur et de gérer la logique locale.
Props	<p>Les props sont le mécanisme par lequel les données et les onctions sont transmises du parent à l'enfant composants dans React.</p> <p>Ils permettent la communication entre les composants.</p>	Les composants reçoivent des données et fonctionnent comme des accessoires, qui qu'ils utilisent pour rendre l'interface utilisateur et gérer les interactions avec les utilisateurs.
Selector	Les sélecteurs sont des outils qui collectent des informations précises à partir des données de l'État. Ils offrent la possibilité aux composants d'obtenir les données nécessaires au magasin de manière plus efficace.	Les sélecteurs extraient les informations de l'état de la propriété et les transmettent aux connexions ou directement aux éléments.
State	<p>L'état correspond à l'état actuel de l'appli.</p> <p>Reducers le conserve dans le magasin et le met à jour.</p>	L'accès à l'état est effectué par sélecteurs pour fournir des données à Composants.
Hooks	<p>Ce sont des fonctions qui vous permettent d'utiliser l'état et d'autres fonctionnalités React dans les fonctions Composants. Ils utilisent des sélecteurs pour accéder à</p> <p>Indiquer les données et les actions de répartition</p>	Les crochets sont utilisés par Composants pour récupérer des données depuis le magasin et déclenchez actions.

B. Serveur Backend

Notre structure de backend repose sur la modularité, avec des éléments structurés en fonction de leurs fonctionnalités et de leurs responsabilités. Bien qu'il puisse s'inspirer du motif MVC, il met l'accent sur la souplesse et l'adaptabilité en s'adaptant aux exigences de notre application. Comme le montre la figure ci-dessous 23 :

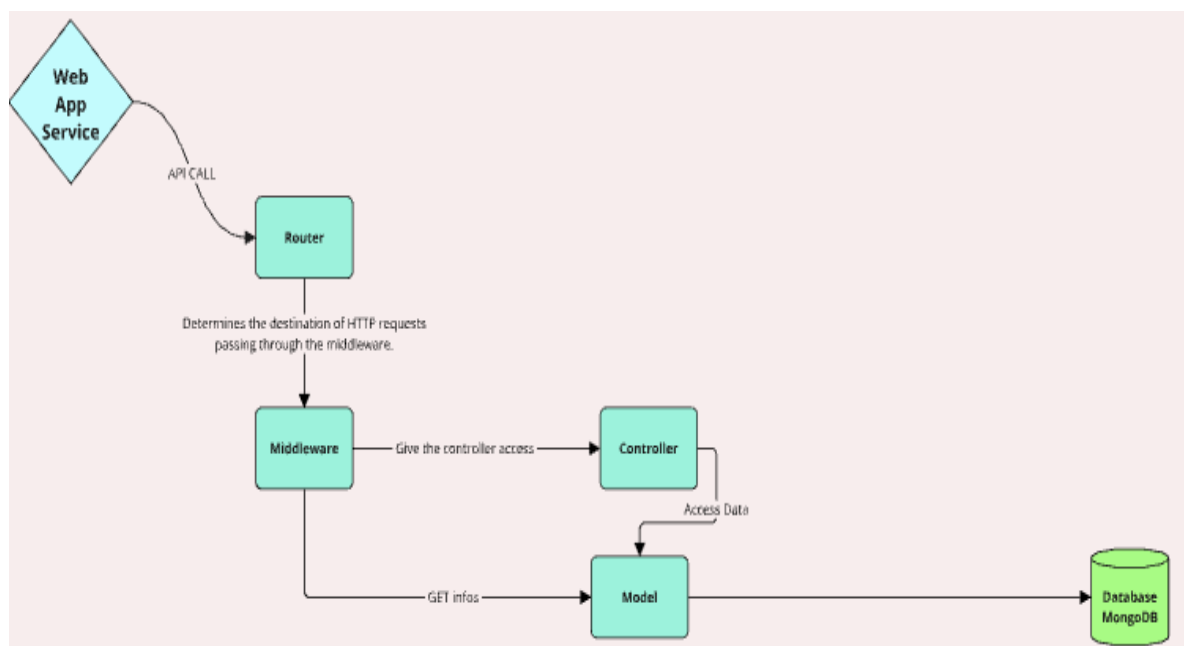


Figure 23 : Le serveur backend

Les éléments de la figure 23 sont définis comme suit :

Tableau 13 : Un aperçu de l'architecture backend[6]

Élément	Définition
Controllors	<p>Les contrôleurs jouent le rôle d'orchestre de notre backend, étant chargés de gérer le flux de contrôle et les données entre les diverses parties du système. Ils analysent les requêtes des utilisateurs, gèrent les interactions entre les modèles, le middleware et les réponses. En réalité,</p> <p>Les contrôleurs jouent un rôle essentiel en liant la logique de notre application, ce qui facilite la communication et le flux de données.</p>
Routes	<p>Les routes déterminent les finalités de notre application et précisent la manière dont les demandes entrantes doivent être traitées. Dans le modèle MVC, ils ont un rôle semblable à celui des contrôleurs en associant les méthodes HTTP et les URI à des actions de contrôleur spécifiques. En structurant les trajets selon les différentes catégories de ressources ou les regroupements logiques,</p> <p>Notre objectif est de garantir la clarté et la durabilité de la conception de notre API.</p>

Élément	Définition
Models	Les modèles regroupent les structures de données ainsi que la logique métier de notre application. Ils symbolisent des éléments tels que les
	utilisateurs, les produits ou les transactions, et déterminent la façon dont les données sont conservées, récupérées et manipulées." Malgré leur focalisation principalement sur la gestion des données, les modèles peuvent également inclure des éléments intégrés. Une logique de validation et des règles métier sont établies en fonction des exigences de notre application.
Middleware	Les fonctionnalités de middleware captent et traitent les demandes entrantes ou les réponses sortantes, en ajoutant une couche supplémentaire de fonctionnalités à notre backend. Le middleware effectue fréquemment les tâches suivantes : authentification, autorisation, journalisation, validation des données et gestion des erreurs. En réduisant la complexité . Nous privilégions la réutilisabilité et la maintenance en middleware, ce qui nous permet de créer aisément des pipelines complexes de traitement des demandes.
Configuration and Environment Variables	Les fichiers de configuration et les variables d'environnement contrôlent les paramètres et les configurations d'une application, tels que les connexions à la base de données, les clés API ou les niveaux de logging. L'externalisation de la configuration de notre codebase permet d'améliorer la souplesse et la sécurité, ce qui nous permet de personnaliser notre application. Comportement sans modification du code source. De plus, des configurations adaptées à chaque environnement garantissent la cohérence à travers différents environnements de mise en œuvre, de l'élaboration à la production.

1.1.2 Chemin développement

A. Architecture MVC

- Le modèle de conception MVC, illustré dans la figure 24, divise une application en trois aspects principaux : modèle, vue et contrôleur.

- **Modèle**

Le modèle signifie les données qui doivent s'afficher dans la vue. Le modèle représente une collection de classes qui décrit la logique métier (modèle métier et modèle de données). Il définit également les règles métier pour les moyens de données comme la manière dont les données peuvent être modifiées et manipulées.

▪ Vue

La vue représente les composants de l'interface utilisateur (XML, HTML, etc.) en utilisant les données reçues du contrôleur. Elle surveille également les changements du modèle via le modèle Observer, pour mettre à jour l'affichage en conséquence.

▪ Contrôleur

Le contrôleur est responsable du traitement des demandes entrantes. Il traite les données de l'utilisateur via le modèle et transmet les résultats à View.

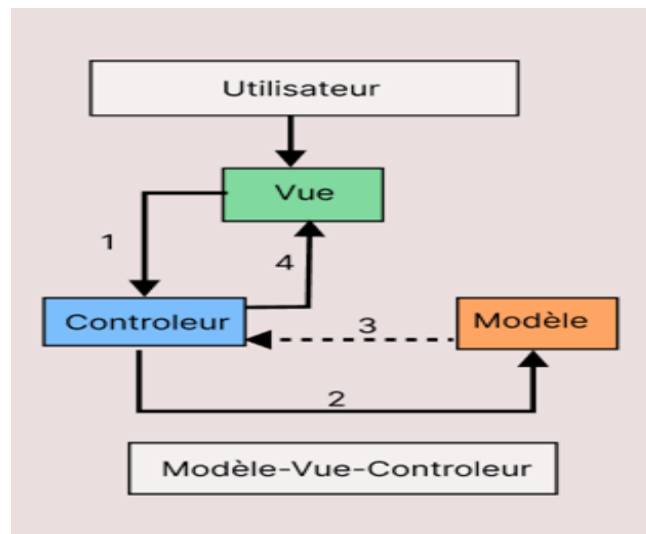


Figure 24 : Illustration de l'architecture MVC

L'architecture MVC est largement utilisée car elle offre de nombreux bénéfices tels que :

- La Possibilité de proposer plusieurs perspectives.
- Support de la technique asynchrone.
- La modification n'affecte pas le modèle entier.

B. Architecture React

L'architecture React Redux est un modèle de conception populaire pour gérer l'état des applications React de manière prévisible et efficace. Voici une explication détaillée de cette architecture :

- **Store** : C'est l'objet central qui détient l'état global de l'application. Il agit comme un conteneur unique pour toutes les données de l'application.
- **Actions** : Ce sont des objets JavaScript simples qui décrivent les changements à apporter à l'état. Elles sont la seule source d'information pour le store.

- **Reducers** : Ce sont des fonctions pures qui spécifient comment l'état de l'application change en réponse aux actions. Elles prennent l'état actuel et une action, et retournent un nouvel état.
- **Dispatch** : C'est la méthode utilisée pour envoyer des actions au store, déclenchant ainsi les changements d'état.
- **Selectors** : Ce sont des fonctions qui extraient des parties spécifiques de l'état du store, permettant aux composants d'accéder uniquement aux données dont ils ont besoin.
- **Middleware** : Il s'agit de fonctions optionnelles qui s'exécutent entre l'envoi d'une action et le moment où elle atteint le reducer. Elles sont utiles pour la gestion des effets secondaires, comme les appels API asynchrones.
- **Provider** : C'est un composant React qui enveloppe l'application et rend le store accessible à tous les composants enfants

Le flux de données dans cette architecture suit un cycle unidirectionnel :

1. L'utilisateur interagit avec l'interface utilisateur.
2. Cette interaction déclenche une action.
3. L'action est envoyée (dispatched) au store.
4. Les reducers traitent l'action et mettent à jour l'état.
5. Le store notifie les composants abonnés du changement d'état.
6. Les composants se re-renderisent avec le nouvel état.

Cette architecture est largement utilisée car elle offre de nombreux bénéfices tels que :

- **Prévisibilité** : Le flux de données unidirectionnel rend le comportement de l'application plus prévisible.
- **Maintenabilité** : La séparation claire des préoccupations facilite la maintenance du code.
- **Débogage** : Les outils de développement Redux permettent de suivre facilement les changements d'état.
- **Testabilité** : Les composants et les reducers étant des fonctions pures, ils sont faciles à tester.

En résumé, l'architecture React Redux fournit une structure robuste pour gérer l'état des applications React, particulièrement bénéfique pour les applications de grande envergure et complexes comme Midad.

1.1.3 Environnement technique

Dans cette partie, nous allons exposer les technologies et les logiciels qui ont été employés pour la réalisation de notre solution.

A. Environnement technologique

Le tableau ci-dessous présente en détail les éléments constitutifs de notre environnement technologique :

Tableau 14 : Présentation de l'environnement technologique

	React JS est une bibliothèque JavaScript pour les interfaces utilisateur. Basée sur des composants réutilisables, elle utilise un DOM virtuel pour des rendus rapides. Son style de programmation est déclaratif et elle est souvent combinée à d'autres bibliothèques pour des applications web complètes.
	Node.js est un environnement d'exécution JavaScript côté serveur. Il utilise un modèle asynchrone et non bloquant, idéal pour les applications web à forte charge. Node.js est un choix populaire pour la création de serveurs web rapides et évolutifs.
	Express.js est un framework web minimaliste et flexible pour Node.js. Il simplifie le développement d'applications web en fournissant des fonctionnalités essentielles telles que le routage, la gestion des requêtes et des réponses, et la création de serveurs....
	MongoDB est une base de données NoSQL orientée documents, développée par MongoDB Inc. Elle est conçue pour stocker de grandes quantités de données non structurées ou semi-structurées, en utilisant un modèle de données basé sur des documents JSON (JavaScript Object Notation).
	Miro est un tableau blanc en ligne pour la collaboration d'équipe qui vous permet de visualiser des concepts, des idées et des solutions avec votre équipe depuis n'importe où.
	Python est un langage de programmation puissant et facile à apprendre. Il dispose de structures de données de haut niveau et permet une approche simple mais efficace de la programmation orientée objet.



Socket.io est un module de Node.js qui permet de créer des Web Sockets, c'est-à-dire des connections bidirectionnelles entre clients et serveur qui permettent une communication en temps réel sur un autre protocole que le protocole http normalement utilisé dans les pages web.



Draw.io est un logiciel de diagramme en ligne gratuit permettant de créer des organigrammes, des diagrammes de processus, des organigrammes des diagrammes UML, ER et de réseau.

B. Environnement logiciel

Le tableau suivant présente tous les logiciels que nous avons utilisés pour réaliser notre système.

Tableau 15 : Présentation de l'environnement logiciel



Visual Studio Code est un éditeur de code source gratuit développé par Microsoft.



Swagger est un langage de description d'interface permettant de décrire des API exprimées à l'aide de JSON. Swagger est utilisé avec toute une série d'outils logiciels open source pour concevoir, créer, documenter et utiliser des services Web.



Trello est un outil de gestion de projet en ligne, lancé en septembre 2011 et inspiré par la méthode [Kanban](#) de [Toyota](#). Il repose sur une organisation des projets en planches listant des cartes, chacune représentant des tâches.



Figma est une plateforme de conception d'interfaces utilisateur (UI) et d'expérience utilisateur (UX) basée sur le cloud, qui facilite la collaboration entre les designers et les développeurs pour créer des prototypes interactifs, des maquettes et des designs d'interface.



Docker est une plateforme permettant de lancer certaines applications dans des conteneurs logiciels lancée en 2013.



GitLab est un logiciel libre de forge basé sur git proposant les fonctionnalités de wiki, un système de suivi des bugs, l'intégration continue et la livraison continue. Développé par GitLab Inc

1.2. Processus DevOps

1.2.1 Contrôle du code source avec Gitlab

Le contrôle du code source, notamment grâce à l'utilisation de Git, est essentiel dans le domaine du développement logiciel moderne. Il simplifie la collaboration, la gestion des versions et le suivi des modifications du code, ce qui améliore l'efficacité et diminue le risque de conflits ou de perte de travail.

GitLab, un outil très prisé pour la gestion de dépôts Git, est une option exceptionnelle pour notre projet, surtout étant donné que nous utilisons déjà Trello pour la gestion de projet. L'illustration 4-3 présente le repo du projet MIDAD sur GitLab.

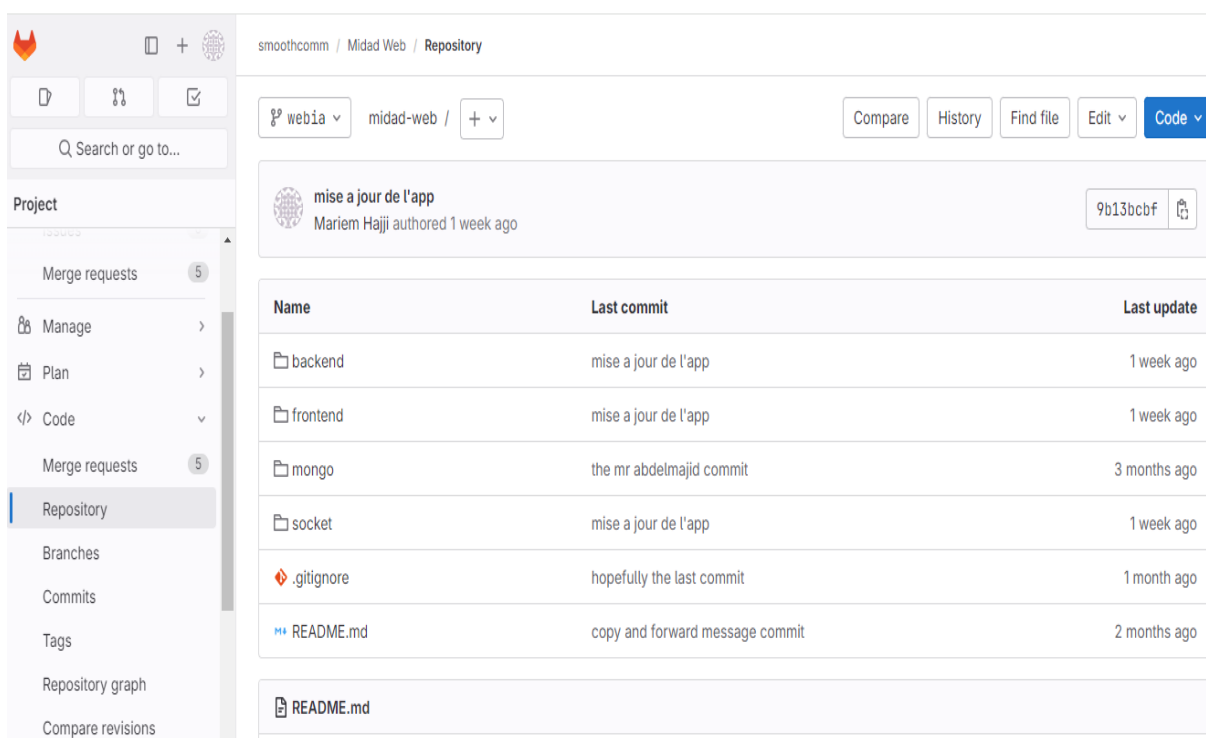


Figure 25 : Midad dans Gitlab[8]

Grâce à l'utilisation de GitLab, on peut assurer non seulement un contrôle de version solide avec Git. Cela nous donne la possibilité de disposer d'un flux de travail cohérent où le suivi des modifications, la création de branches, la révision du code et la gestion des tâches deviennent un processus coordonné.

Le contrôle du code source et les révisions de code sont assurés par GitLab, tandis que Trello assure le suivi des problèmes et la gestion de projet. Quand ils sont utilisés

conjointement, un développeur a la possibilité de connecter directement un commit ou une demande de pull spécifique dans GitLab à une carte Trello, ce qui permet de suivre l'étape de planification jusqu'à la modification finale du code.

1.2.2 Tests Unitaires

"Tests unitaires ne sont pas une perte de temps. Ils font gagner du temps au final." - **Martin Fowler**

Notre philosophie de tests pour notre application React est guidée par cette citation, qui a été réalisée en utilisant la bibliothèque React Testing Library.

Dans cette bibliothèque, nous trouvons des outils qui respectent des principes essentiels afin de rédiger des tests qui reflètent l'utilisation réelle de notre logiciel, que ce soit pour des tests unitaires ou des tests d'intégration. Elle encourage l'interaction avec les nœuds du DOM plutôt qu'avec les instances de composants et ne propose pas l'utilisation directe des instances de composants. Cette méthode offre la possibilité d'évaluer les éléments de l'application de la façon dont l'utilisateur les utiliserait.

Pour les tests unitaires, nous avons employé ces outils afin de vérifier le comportement de chaque composant de manière individuelle. Grâce aux utilitaires offerts par React Testing Library, nous avons pu vérifier que chaque unité de l'application fonctionne conformément à nos attentes. Dans le cadre des tests d'intégration, la bibliothèque de tests React a été employée afin de créer des scénarios d'utilisation réels, en examinant non seulement le fonctionnement des composants individuels, mais également leurs interactions au sein de l'application. Pour structurer nos tests, nous utilisons également la méthode des 3A (Arrange, Action, Assert).

```
import { describe, it, expect } from "vitest";
import GroupList from "../CreateGroup";

describe('CreateGroup Component', () => {
  it('should render a list of groups', () => {
    // Arrange: Préparer les données nécessaires pour le test
    const groups = [
      { id: 1, title: 'Girls' }, { id: 2, title: 'azerty' }
    ];
    // Action: Exécuter l'action
    const groupListComponent = GroupList({ groups });
    // Assert: Vérification du rendu attendu
    expect(groupListComponent).toBeTruthy();
  });
});
```

Figure 26 : Exemple de test unitaire 3A

1.2.3 Approche CI/CD

Dans notre projet, nous avons mis en œuvre une intégration et une livraison continues. Ainsi, à chaque push vers la branche midadweb, un build de code est automatiquement déclenché et le code est testé sur un conteneur d'image spécifique. Le flux de travail Gitflow, que nous utilisons, propose une méthode évolutive, souple et performante pour gérer notre projet. Ce flux de travail est particulièrement bénéfique dans un contexte CI/CD (Intégration Continue/Déploiement Continu), car il assure l'automatisation des processus. Pour cette automatisation, nous avons utilisé Jenkins, un outil puissant pour orchestrer les processus de développement et de déploiement. Comme le montre la figure 27 ci-dessous, cette intégration permet une gestion fluide et continue des modifications apportées au projet.

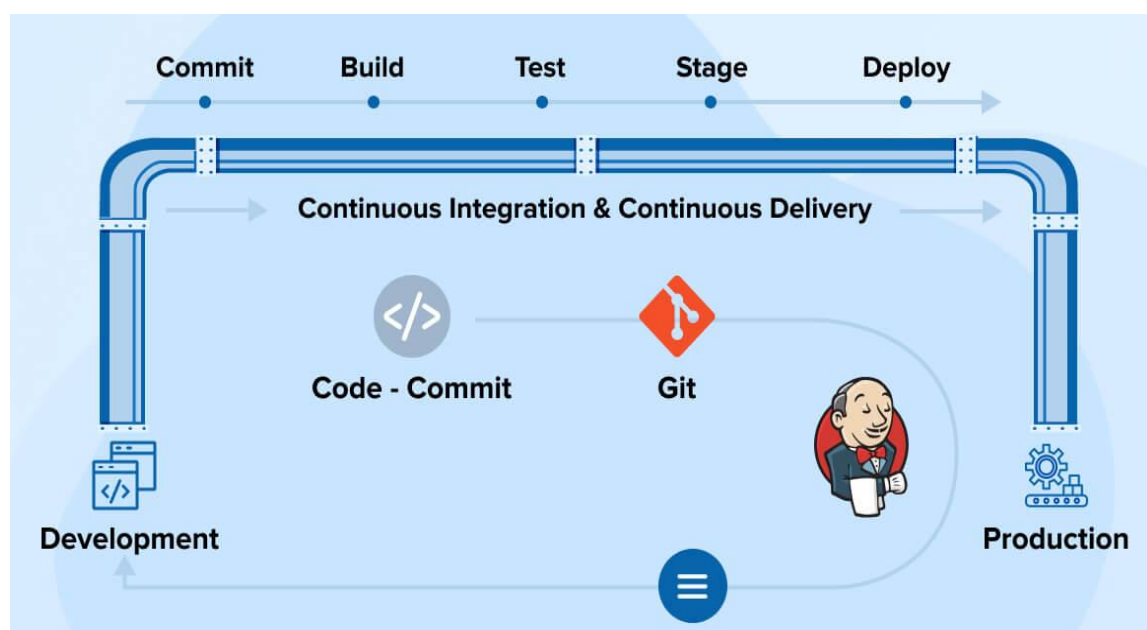


Figure 27 : Représentation de l'approche CI/ CD avec Jenkins[9]

Cette illustration met en évidence le processus d'intégration continue et de livraison continue (CI/CD) dans un contexte de développement logiciel. Elle illustre les différentes étapes du processus, allant du développement à la production.

1. **Développement** : Les développeurs écrivent du code et le committent dans un système de gestion de versions (Git).
2. **Commit** : Une fois que le code est comité, il est intégré dans le dépôt Git, déclenchant le pipeline CI/CD.
3. **Build (CI)** : Le code est ensuite compilé ou construit. Cette étape assure que le code peut être assemblé sans erreurs.
4. **Test (CI)** : Une série de tests automatisés est exécutée pour vérifier la qualité et la fonctionnalité du code.
5. **Stage(CD)** : Le code validé est déployé dans un environnement de staging, qui est une réplique de l'environnement de production, pour des tests supplémentaires.
6. **Deploy(CD)** : Si tous les tests passent avec succès, le code est déployé en production.

Le pipeline CI/CD est orchestré par Jenkins, un outil d'automatisation populaire représenté dans l'image par le personnage en bas à droite. Jenkins surveille le dépôt Git pour tout nouveau commit et déclenche automatiquement le pipeline de build, de test et de déploiement. Ce processus permet de s'assurer que les modifications de code sont intégrées et

livrées de manière continue, rapide et fiable, minimisant ainsi les risques et les erreurs lors des déploiements en production.

L'image ci-dessous montre un exemple de pipeline CI/CD utilisé pour le projet midad-web. Ce pipeline comporte plusieurs étapes essentielles à l'automatisation du déploiement continu :

- Clone **Repository** : Cloner le dépôt de code source.
- Build **Docker Image** : Construire l'image Docker à partir du code source.
- Deploy **Container** : Déployer le conteneur Docker construit.
- Clean **Up** : Nettoyer les ressources temporaires utilisées pendant le processus.

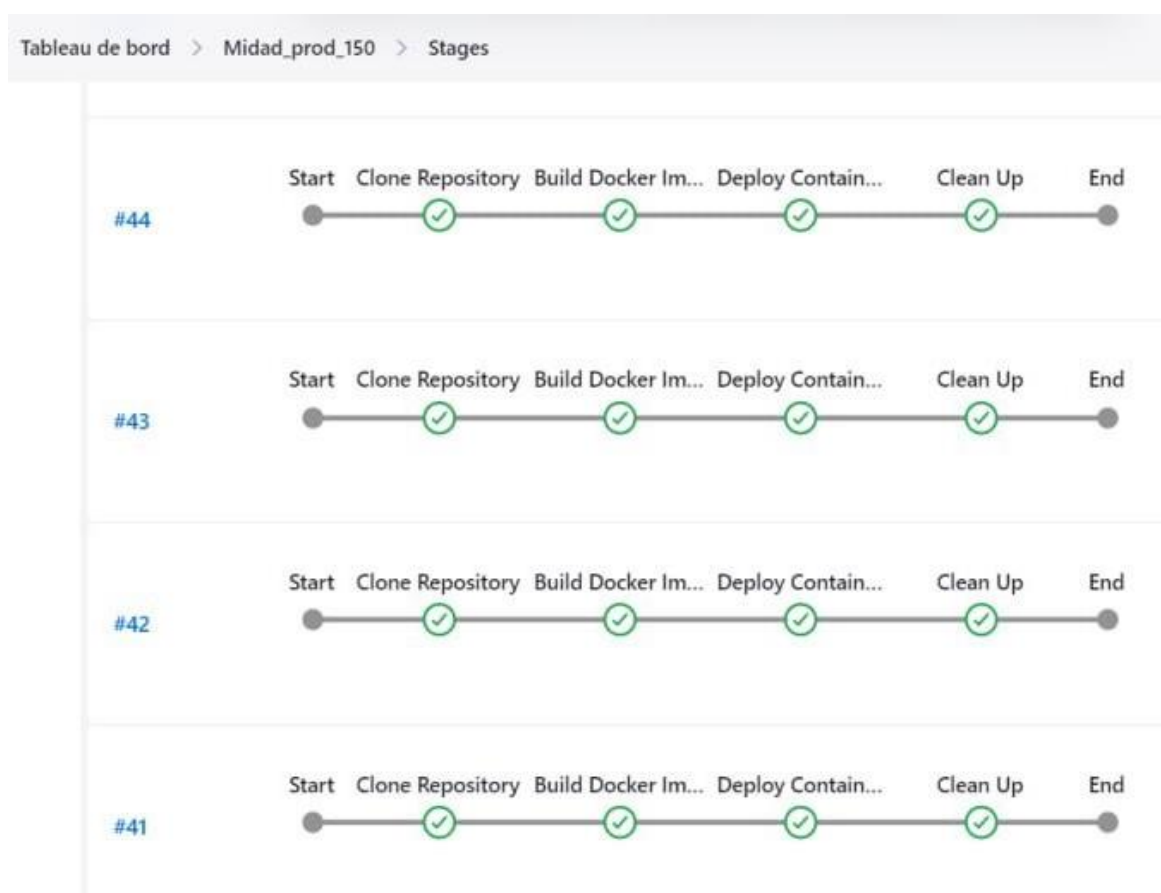


Figure 28 : Exemple de Pipeline [10]

1.2.4 Cloud & Conteneurisation

Nous avons opté pour Docker, une plateforme de conteneurisation puissante qui facilite la création d'environnements isolés et reproductibles, afin de conteneuriser notre application. Grâce à Docker, nous avons réussi à organiser notre application et ses dépendances dans des

conteneurs légers et portables, assurant ainsi une cohérence et une facilité de mise en place dans divers environnements.

Pour notre application, nous avons développé quatre fichiers Docker distincts, comprenant une description détaillée de la configuration et les instructions nécessaires pour générer une image Docker pour chaque partie : le frontend, le backend , le socket et le mongo. Quatre Dockerfiles différents sont utilisés afin de séparer et d'optimiser la création de chaque partie de notre application. Cela offre la possibilité d'une gestion modulaire et d'une configuration précise des environnements de développement, de test et de production pour chaque composant. L'enregistrement des données sera stocké dans le conteneur MongoDB même en l'absence d'une base de données définie. MongoDB créera automatiquement la base de données. On peut observer l'un de ces fichiers Docker dans la figure 29 ci-dessous.

```
1 FROM node:20
2 WORKDIR /usr/src/app
3 VOLUME /usr/src/app/src/images
4 COPY package*.json ./
5 RUN npm install --legacy-peer-deps
6 COPY . .
7 EXPOSE 3000
8 CMD ["npm", "start"]
```

Figure 29 : Configuration du fichier Dockerfile

Pour déployer notre application, nous avons utilisé le cloud privé de Smooth Algo, ce qui constitue une forme d'auto-hébergement. Nous utilisons un serveur Nginx pour assurer le routage avec le domaine midad.tn. Dans une application utilisant Docker et Nginx, Nginx est souvent configuré comme un reverse proxy. Voici comment cela fonctionne en quelques mots :

- a. Nginx reçoit les requêtes des clients.
- b. Nginx les transmet aux conteneurs Docker appropriés qui hébergent les services backend de l'application.
- c. Nginx renvoie les réponses des conteneurs Docker aux clients. Cela permet de gérer les requêtes de manière efficace, sécurisée et évolutive.

Un "cloud privé" de Smooth Algo désigne une infrastructure de cloud computing dédiée exclusivement à l'usage interne de cette société. Cette approche nous permet d'avoir un contrôle total sur notre environnement d'hébergement, incluant la gestion des ressources, la sécurité et les mises à jour.

1.3. Intelligence artificielle

L'intelligence artificielle améliore notre application en facilitant la rédaction de messages et en proposant des réponses prédictives. Grâce à des modèles avancés comme GPT-4, l'auto-complétion suggère des mots ou des phrases au fur et à mesure que l'utilisateur tape, accélérant la rédaction et réduisant les erreurs. De plus, la prédiction des réponses utilise le contexte de la conversation pour offrir des réponses pertinentes rapidement. Ainsi, notre application rend la communication plus fluide, efficace et agréable.

1.3.1 Algorithme de Prédiction des Réponses

L'algorithme de prédiction des réponses dans le traitement du langage naturel (NLP) est un processus complexe qui vise à générer des réponses pertinentes en fonction d'une entrée textuelle donnée. Ce processus repose sur plusieurs étapes clés : l'analyse des données, leur transformation, la construction du modèle d'intelligence artificielle, et enfin, l'évaluation et l'interprétation des résultats.

La technique la plus cruciale et centrale dans cet algorithme est l'Embedding des Mots (Word Embedding). L'Embedding transforme les mots en vecteurs denses et continus dans un espace de dimension fixe. Cela permet au modèle de capturer les similarités sémantiques entre les mots, contrairement aux représentations de mots simples (comme one-hot encoding) qui sont souvent clairsemées et n'ont pas cette propriété. En réduisant la dimensionnalité des entrées textuelles, l'Embedding rend le traitement plus efficace en termes de calcul et de mémoire, tout en préservant l'information sémantique. De plus, les vecteurs d'Embedding apprennent à capturer les relations entre les mots en fonction de leur contexte dans les données d'entraînement, ce qui aide le modèle à générer des prédictions plus précises [inspiré Introduction to Word Embedding by Towards Data Science].

Voici comment chaque composant du code contribue à l'efficacité globale de la solution d'IA, en suivant les étapes clés du processus :

A. Analyse des Données Pertinentes

Le code débute en collectant et en normalisation les données textuelles à l'aide d'une API, en convertissant les textes en caractères minuscules afin d'uniformiser les entrées. Il est essentiel de passer par cette étape afin d'assurer la cohérence des données et de les rendre prêtes à être converties en un format exploitable.

La figure 30 ci-dessous représente l'analyse des données pertinentes à intégrer à la solution d'intelligence artificielle.

```
#Lire les données texte
df = pd.read_json("http://localhost:5000/api/messenger/guessWord/" + friendId)
# Convertir des données textuelles en minuscules
df = df[0].str.lower()
# Décomposer les données textuelles en phrases
sentences = df.values.tolist()
```

Figure 30 : Capture d'analyse des données pertinentes

B. Extraction et transformation des données

Afin de préparer les données à l'apprentissage, le code tokenise les textes en les transformant en séquences numériques, puis applique un padding pour calibrer la longueur des séquences. L'objectif de ce processus est de convertir les données en un format que le modèle peut utiliser de manière efficace pour l'entraînement.

La figure 31 ci-après illustre le processus d'extraction et de transformation des données.

```
# Créer un Tokenizer pour convertir des mots en nombres
tokenizer = Tokenizer(filters='!"#$%&()*+,-./:;<=>?@[\\]^_`{|}~\t\n', lower=True)
tokenizer.fit_on_texts(sentences)
# Nombre de mots (taille du dictionnaire)
vocab_size = len(tokenizer.word_index) + 1
# Convertir des phrases en séquences de nombres
sequences = tokenizer.texts_to_sequences(sentences)
# Créer des séquences d'entrée et de sortie
input_sequences = []
output_sequences = []

for sequence in sequences:
    for i in range(1, len(sequence)):
        input_sequences.append(sequence[:i]) # Le premier mot de la phrase
        output_sequences.append(sequence[i]) # Le ième mot de la phrase

# Remplir les chaînes d'entrée à la même longueur
max_length = max([len(x) for x in input_sequences])
input_sequences = pad_sequences(input_sequences, maxlen=max_length, padding="pre")

# Rendre les séquences de sortie encodées à chaud
output_sequences = tf.keras.utils.to_categorical(output_sequences, num_classes=vocab_size)
```

Figure 31 : Capture d'extraction et transformation des données

C. Elaboration du modèle d'IA

L'architecture séquentielle du modèle est basée sur des couches d'Embedding et LSTM, spécialement conçues pour traiter des séquences textuelles et prendre en compte les dépendances temporelles. Les prédictions des mots suivants sont générées par la couche finale, Dense, qui utilise une activation softmax pour la classification. Grâce à cette conception, le modèle peut acquérir des connaissances sur les relations complexes entre les mots et produire des réponses adéquates.

La figure 32 suivante montre le processus d'élaboration du modèle d'intelligence artificielle appliquée au traitement du langage naturel.

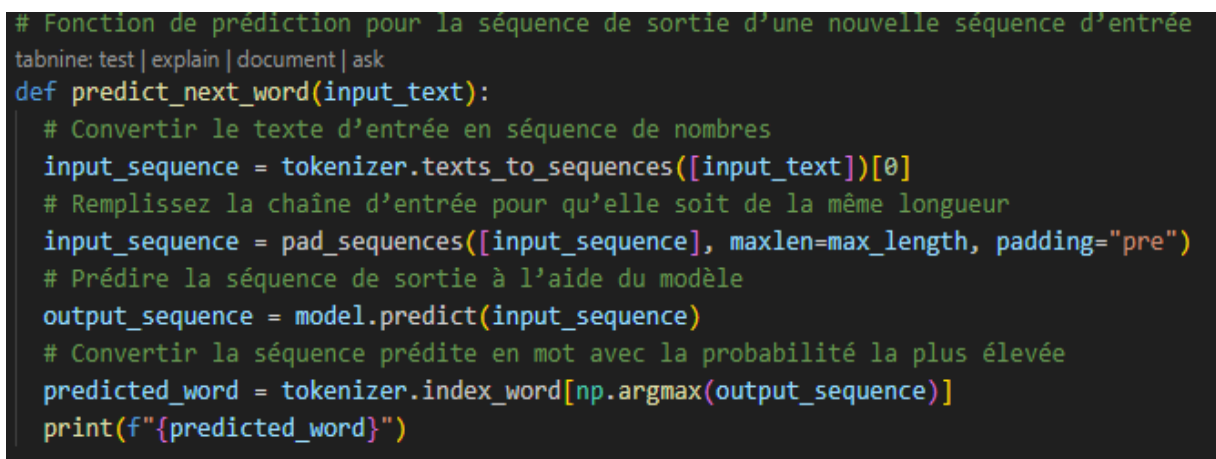
```
# Construire le modèle de Deep Learning
model = Sequential([
    Embedding(vocab_size, 64, input_length=max_length),
    LSTM(64),
    Dense(vocab_size, activation="softmax")
])
# Construire le modèle
model.compile(loss="categorical_crossentropy", optimizer="adam", metrics=["accuracy"])
model.summary()
# Entraîner le modèle
model.fit(input_sequences, output_sequences, epochs=10, batch_size=128, workers=4, use_multiprocessing=True)
```

Figure 32 : Capture d'élaboration du modèle

D. Évaluation et Interprétation des Résultats

Après l'entraînement, le modèle est utilisé pour prédire des mots dans de nouvelles séquences d'entrée. La fonction de prédiction transforme le texte d'entrée en une séquence numérique, effectue une prédiction et retourne le mot avec la probabilité la plus élevée. Cette étape est cruciale pour évaluer la précision du modèle et ajuster les paramètres si nécessaires.

La figure 33 suivante montre le processus d'évaluation et interprétation des résultats.



```
# Fonction de prédiction pour la séquence de sortie d'une nouvelle séquence d'entrée
tabnine: test | explain | document | ask
def predict_next_word(input_text):
    # Convertir le texte d'entrée en séquence de nombres
    input_sequence = tokenizer.texts_to_sequences([input_text])[0]
    # Remplissez la chaîne d'entrée pour qu'elle soit de la même longueur
    input_sequence = pad_sequences([input_sequence], maxlen=max_length, padding="pre")
    # Prédire la séquence de sortie à l'aide du modèle
    output_sequence = model.predict(input_sequence)
    # Convertir la séquence prédite en mot avec la probabilité la plus élevée
    predicted_word = tokenizer.index_word[np.argmax(output_sequence)]
    print(f"{predicted_word}")
```

Figure 33 : Capture d'évaluation et interprétation des résultats.

Les différentes parties du code illustrent la manière dont les données sont analysées, extraites, transformées et exploitées afin de concevoir une solution d'intelligence artificielle. Par la suite, les résultats sont étudiés et analysés afin de prédire le prochain mot dans une séquence de texte donnée. On peut appliquer cette méthode à différentes applications de traitement du langage naturel, comme la correction de texte, la création de texte, et bien d'autres encore.

1.3.2 Algorithme auto-complétion

L'algorithme d'auto-complétion utilise principalement la technique de recherche par préfixe (prefix search) pour générer des suggestions d'auto-complétion en fonction des mots déjà tapés par l'utilisateur. Cette technique, implémentée à l'aide de la fonction reduce, parcourt une liste de mots possibles et filtre ceux qui correspondent au préfixe de recherche, jusqu'à un maximum de 6 suggestions. L'algorithme est simple et efficace pour des ensembles de données de taille modérée, transformant les mots en vecteurs denses et continus pour capturer les similarités sémantiques, et réduisant la dimensionnalité des entrées textuelles pour un traitement plus rapide et précis. [Inspiré Couchbase]

Ces deux figures 34 et 35 de code démontrent l'implémentation d'une fonctionnalité d'auto-complétion ainsi que la gestion de requêtes pour les fichiers publics dans une application Node.js, illustrant ainsi une partie essentielle du système de suggestion de mots. Voici une analyse détaillée de chaque fonction :

```
const getAutoCompleteWords = (objectKeys, search) => {  
  return objectKeys.reduce((acc, currentObj) => {  
    if (acc === undefined || acc.length < 6)  
      if (search === currentObj.substr(0, search.length))  
        return acc.concat(currentObj)  
    return acc  
  }, [])  
}  
  
module.exports = {  
  ...  
  getAutoCompleteWords  
}
```

Figure 34 : Capture d'implémentation de la Fonction d'Auto-complétion

```
const handleAutoComplete = (res, url, data) => {
  data = JSON.parse(data);
  let searchobj = search[0].toUpperCase();
  let results = getWords.getAutoCompleteWords(Object.keys(data[searchobj]), search);
  res.writeHead(200, { 'Content-Type': 'application/json' });
  res.end(JSON.stringify(results));
}

const handlePublic = (response, url) => {
  let extension = url.split(".")[1];
  let filePath = path.join(__dirname, "..", "public", url);
  fs.readFile(filePath, (error, file) => {
    if (error) {
      response.writeHead(500);
      response.end("error500");
    } else {
      response.writeHead(200, { 'Content-Type': extension });
      response.end(file);
    }
  });
}

module.exports = {
  handleAutoComplete,
  handlePublic,
  handleHome
}
```

Figure 35 : Gestion des Requêtes d'Auto-complétion

On conclure, ces deux figures de code montrent comment le système de suggestion de mots est implémenté, avec des fonctions pour générer des suggestions basées sur des préfixes de recherche et pour gérer les requêtes de fichiers publics.

1.4. Autres fonctionnalités intégrées

1.4.1 Système de notifications

Le système de notification dans cette application de messagerie fonctionne principalement à travers l'utilisation de Socket.IO pour les notifications en temps réel, et des notifications du navigateur pour les notifications système.

L'application Midad utilise un système de notification à deux niveaux pour assurer que les utilisateurs restent informés des nouveaux messages, qu'ils soient actifs sur la page ou non.

Premièrement, si l'utilisateur a accordé les permissions nécessaires, des notifications du navigateur sont générées pour chaque nouveau message reçu. Ces notifications apparaissent même si l'utilisateur navigue sur d'autres onglets ou applications, affichant le nom de l'expéditeur et un aperçu du message. Elles permettent à l'utilisateur de revenir rapidement à l'application d'un simple clic. Deuxièmement, à l'intérieur de l'interface de Midad, la bibliothèque react-hot-toast est employée pour afficher des notifications visuelles élégantes et non intrusives. Ces notifications toast apparaissent brièvement à l'écran, généralement dans un coin, pour informer l'utilisateur des nouveaux messages, des statuts de connexion des amis, ou d'autres événements pertinents. Elles offrent une expérience fluide et réactive, complétant ainsi les notifications du navigateur pour une couverture complète des besoins de notification des utilisateurs.

1.4.2 Système d'appel audio/vidéo

Le système d'appel audio/vidéo utilise WebRTC pour établir une communication en temps réel entre deux utilisateurs. Lorsqu'un utilisateur initie un appel, le composant Vocall gère le processus. Il commence par accéder aux périphériques média de l'utilisateur (caméra et microphone) via `navigator.mediaDevices.getUserMedia()`. Une fois les flux média obtenus, il les affiche dans le composant vidéo local. L'appel est ensuite signalé à l'autre utilisateur via une connexion socket. Quand l'appel est accepté, un objet Peer (créé avec la bibliothèque PeerJS) est utilisé pour établir une connexion peer-to-peer. Les flux audio et vidéo sont alors échangés entre les pairs, permettant une communication bidirectionnelle. Le système gère également les changements d'état comme la mise en sourdine, l'arrêt de la vidéo, et la fin de l'appel, en mettant à jour l'interface utilisateur et en informant l'autre participant via des événements socket. Ce processus permet une communication fluide et en temps réel entre les utilisateurs, avec la possibilité de contrôler divers aspects de l'appel.

1.4.3 Système d'invitation d'amis

Le système d'invitation d'amis dans l'application Midad permet aux utilisateurs d'ajouter des amis via deux méthodes : invitation par email et invitation via la plateforme Midad. Lorsqu'un utilisateur souhaite ajouter un ami directement via la plateforme Midad, il peut le faire en utilisant la fonctionnalité d'ajout d'ami. Cette fonctionnalité présente une interface où l'utilisateur peut confirmer son choix d'ajouter un ami. En cliquant sur le bouton "Ajouter", une action est déclenchée : l'action `addFriend` est dispatchée avec les informations de l'utilisateur

actuel et de l'utilisateur à ajouter. Ensuite, un événement add-friend est émis via Socket.io, informant le serveur de l'ajout d'ami. Enfin, l'interface utilisateur est mise à jour pour refléter l'ajout de l'ami, en masquant certaines options et en réinitialisant les champs pertinents. Pour inviter un ami par email, l'utilisateur peut utiliser la fonctionnalité d'envoi d'invitation par email. Cette fonctionnalité permet à l'utilisateur d'entrer l'adresse email de la personne à inviter. L'utilisateur entre l'adresse email et soumet le formulaire. Si l'email est valide, une action `sendInvitation` est dispatchée, envoyant l'invitation à l'adresse email spécifiée. Des alertes sont affichées pour informer l'utilisateur du succès ou de l'échec de l'envoi de l'invitation. Enfin, l'email est réinitialisé et le formulaire d'envoi d'email est masqué. Ces deux méthodes offrent une flexibilité aux utilisateurs de Midad pour agrandir leur réseau d'amis, soit en utilisant des invitations directes via la plateforme, soit en envoyant des invitations par email à des personnes extérieures à Midad.

1.5. Jeu d'essais

1.5.1 Présentation des interfaces de l'application

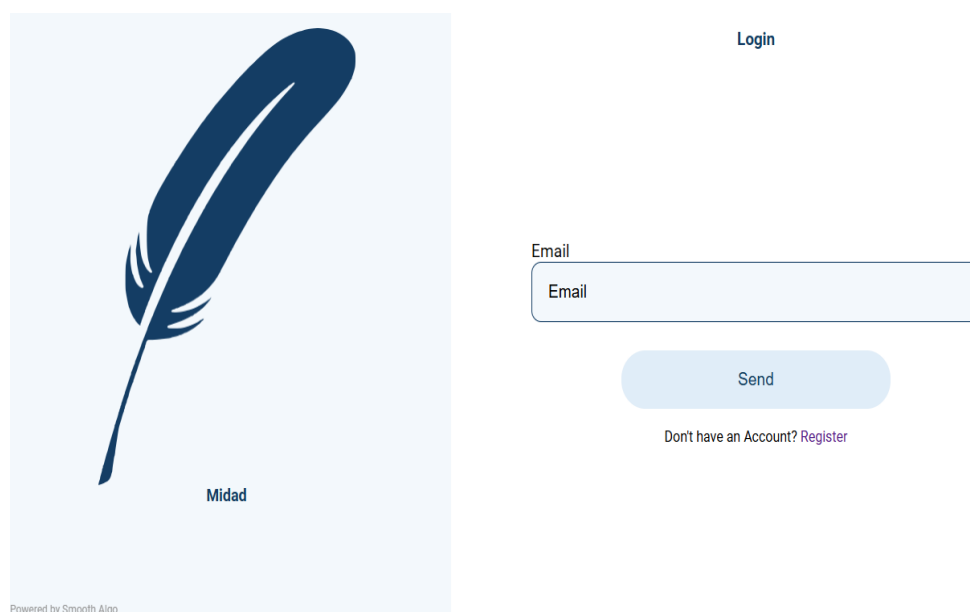
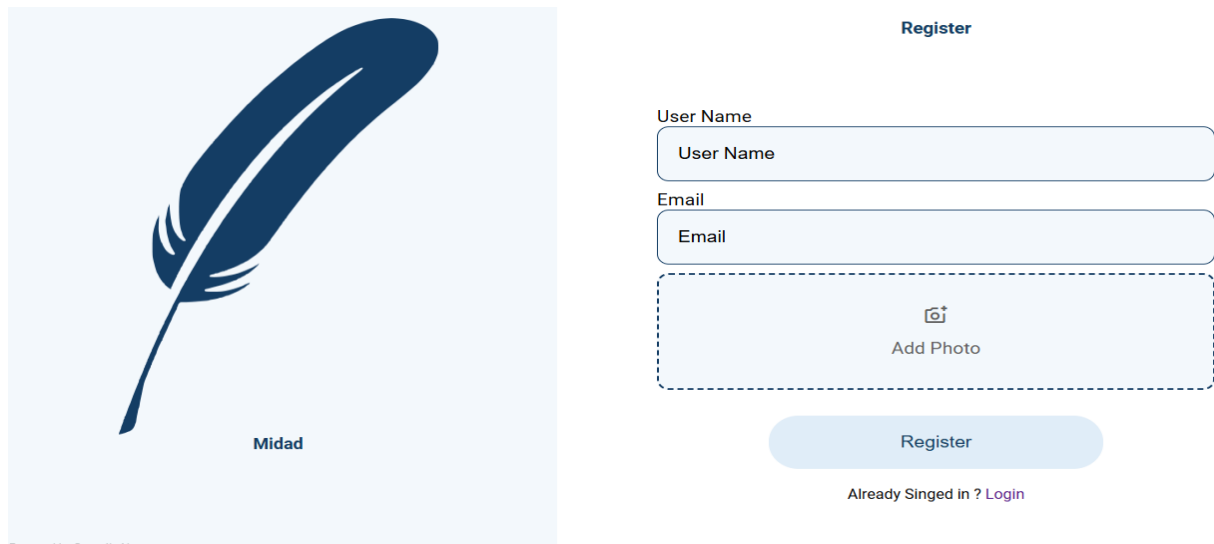


Figure 36 : Interface de connexion

Le screen de connexion contient le champ email avec une fonction sur le champ email qui corrige l'adresse mail contenant des caractères interdits.



The registration interface consists of two main parts. On the left is a profile picture placeholder featuring a dark blue quill icon and the name 'Midad' below it. On the right is the registration form, which includes a title 'Register' at the top. Below the title are three input fields: 'User Name', 'Email', and a dashed box for 'Add Photo' with a camera icon. A 'Register' button is positioned below the photo upload area. At the bottom, there is a link that says 'Already Singed in ? Login'.

Figure 37 : Interface de l'inscription

Le formulaire d'inscription comporte les champs nécessaires pour la création d'un compte avec une validation front sur tous les champs avant de passer la requête au contrôleur.

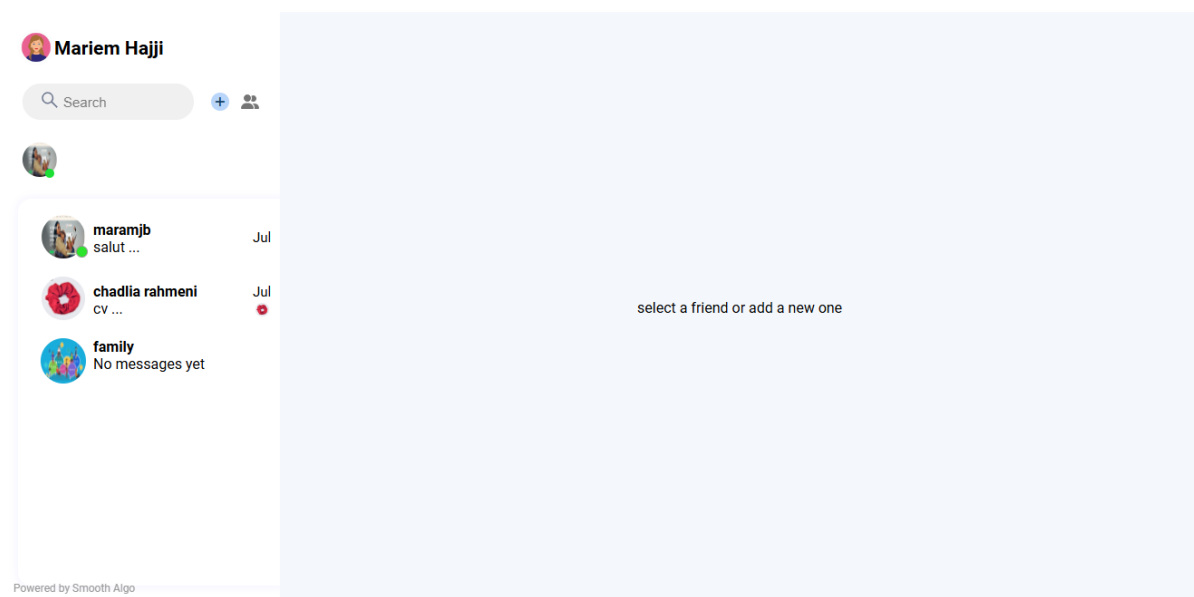
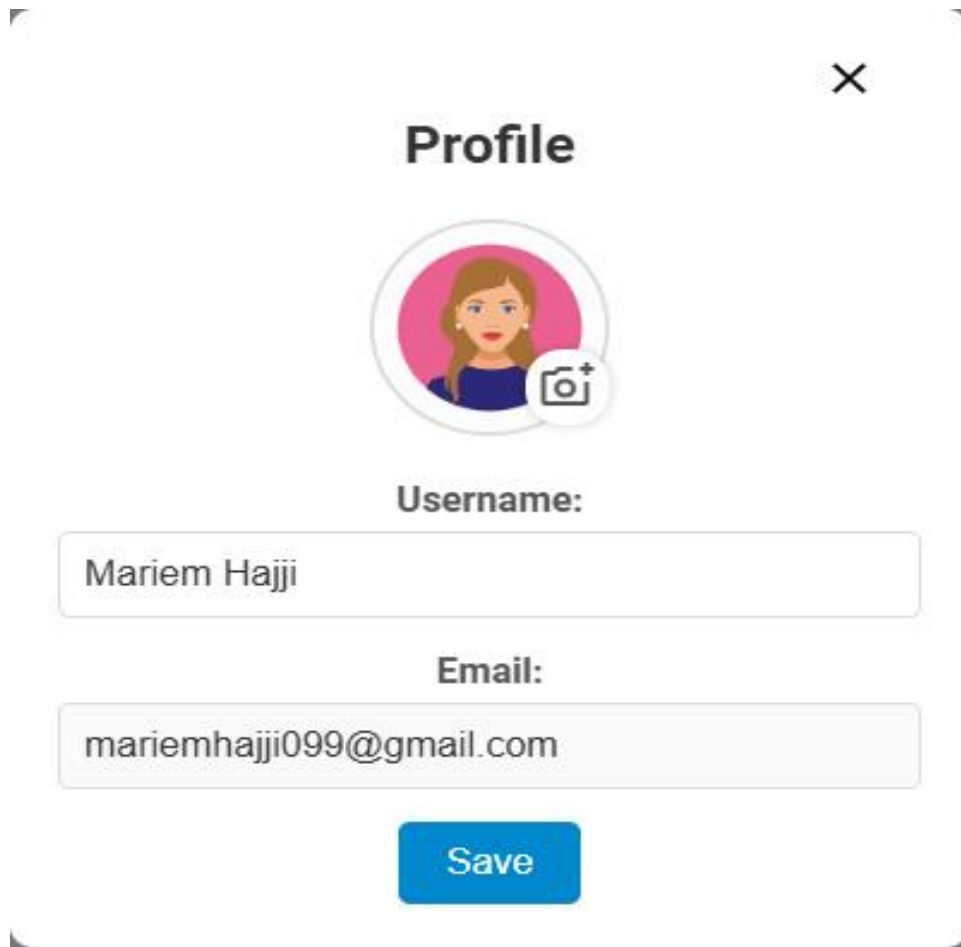


Figure 38 : Interface de la messagerie

L'interface de la messagerie permet une gestion intuitive des conversations, groupes et amis, avec une fenêtre de chat centralisée pour l'échange de messages et fichiers. Une barre de recherche facilite la navigation, et les paramètres de messagerie.




The image shows a mobile application profile screen. At the top right is a close button (X). The title 'Profile' is centered. Below it is a circular profile picture of a woman with a camera icon overlay. Underneath the picture is the label 'Username:' followed by a text input field containing 'Mariem Hajji'. Below that is the label 'Email:' followed by a text input field containing 'mariemhajji099@gmail.com'. At the bottom center is a blue 'Save' button.

Figure 39 : Interface de profil

Une interface de profil doit inclure une image de profil, un nom complet et une adresse email. L'image de profil et le nom complet peuvent être modifiés, tandis que l'adresse email reste inchangée pour garantir la sécurité et la fiabilité des données.

Create New Group



The illustration shows a group of seven diverse people interacting with large, colorful speech bubbles. The people are depicted in various poses: some are sitting, some are standing, and some are using laptops or mobile devices. The speech bubbles are in shades of green, yellow, blue, and purple, and some contain ellipses to represent text. The background is a solid blue color.

Title

Members

CANCEL CREATE

Figure 40 : Interface de groupe

Un groupe en ligne se compose d'un titre et de membres. Le titre reflète l'objectif du groupe, tandis que les membres sont ajoutés lors de la création du groupe. Chaque membre contribue aux échanges, créant un environnement collaboratif.

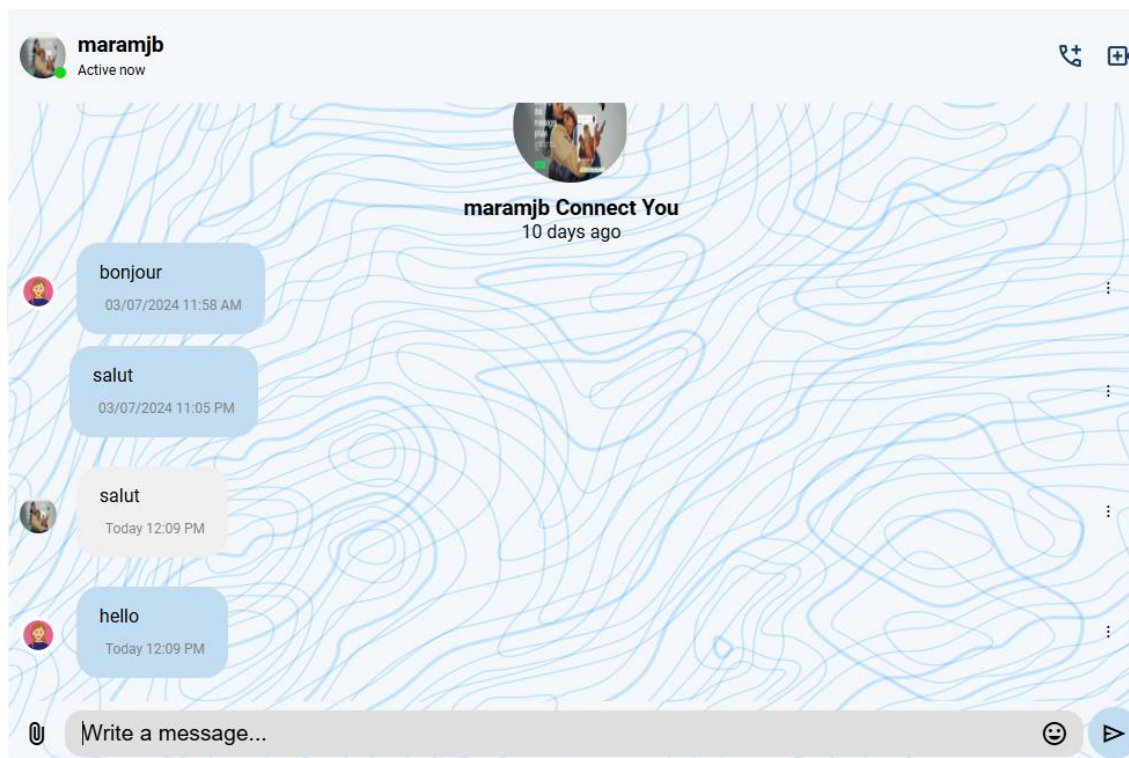


Figure 41 : Interface de la zone des messages

La figure 42 présente une interface de messagerie instantanée. Elle montre une conversation simple et sporadique.

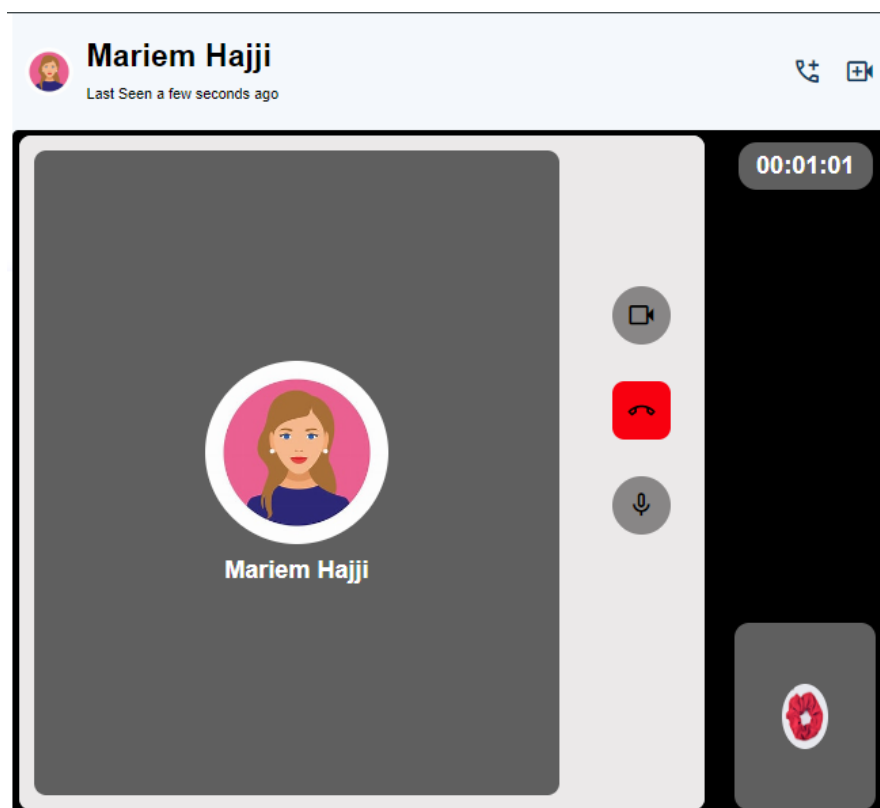


Figure 42 : Interface de l'appel audio/video

La figure précédente montre l'interface d'un appel vidéo en cours avec un utilisateur. L'écran principal est gris et affiche un avatar circulaire. À droite, on peut voir les contrôles de l'appel, notamment les boutons pour la caméra, pour raccrocher, et pour le microphone. Un chronomètre en haut à droite indique que l'appel dure depuis 1 minute et 1 seconde. L'interface suggère qu'il s'agit d'une application de communication vidéo moderne.

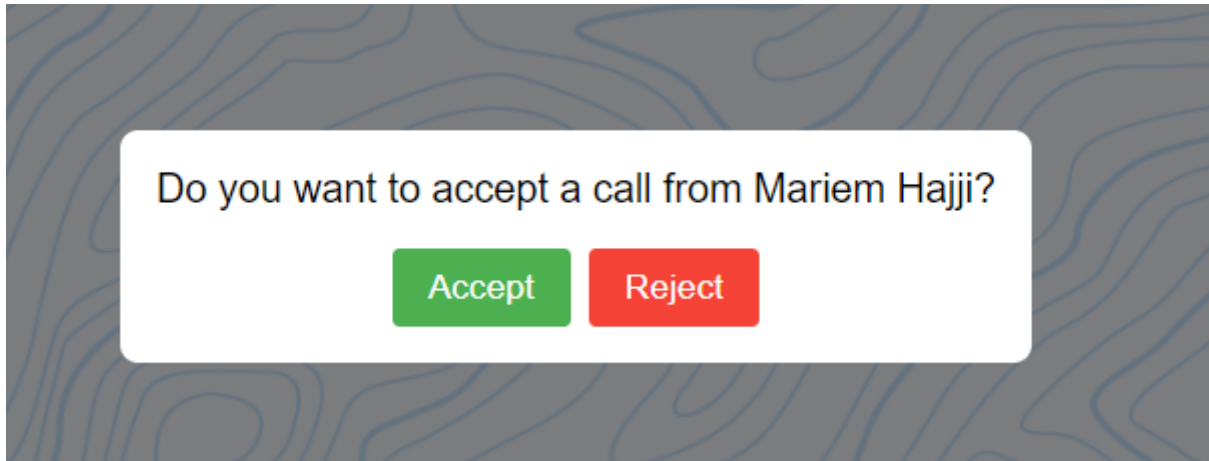


Figure 43 : Interface d'acceptation/refuse

La figure 43 ci-dessus montre une interface d'appel entrant sur une application de midad. Un message demande à l'utilisateur s'il souhaite accepter un appel de Mariem Hajji. Deux options sont présentées sous forme de boutons : "Accept » pour répondre à l'appel, et "Reject» pour le refuser.

Conclusion

Ce dernier chapitre représente le dernier volet de ce rapport. Il a pour objectif de présenter l'environnement logiciel, les architectures physiques et logiques de notre application et sa dernière partie est dédiée à la navigation sur les différentes fonctionnalités quelle propose et elle est représentée sous forme de captures d'écrans.

CONCLUSION GÉNÉRALE & PERSPECTIVES

En réponse à la demande croissante d'outils de communication fluides, l'équipe du projet Midad a travaillé d'arrache-pied pour concevoir une application web complète de messagerie instantanée et d'appels audio/vidéo. En adoptant des technologies de pointe et une méthode de développement itérative, Midad vise à établir de nouvelles normes en matière de technologie de communication.

En s'appuyant sur la pile MERN et en intégrant le chiffrement en temps réel, MIDAD assure à la fois la sécurité des données et une expérience utilisateur conviviale. L'adoption de la méthodologie Scrum pour la gestion du projet favorise l'agilité et l'amélioration continue, permettant ainsi une adaptation rapide aux évolutions des exigences et des demandes du marché.

Au fur et à mesure que nous progressons dans les versions et les sprints prévus, Midad reste déterminé à livrer des fonctionnalités innovantes et intuitives. En comblant les lacunes des solutions existantes et en se concentrant sur les principes de conception centrés sur l'utilisateur, Midad s'efforce de devenir le choix privilégié pour les particuliers et les organisations qui s'appuient sur le télétravail, la collaboration et les interactions virtuelles. Dans un futur proche, Midad se rapprochera de sa vision de devenir un leader, permettant aux utilisateurs de se connecter, de collaborer et de s'épanouir dans le paysage numérique actuel.

Nous avons également envisagé d'ajouter plusieurs fonctionnalités avancées afin d'améliorer encore davantage l'expérience Midad. Les utilisateurs pourront utiliser les fonctionnalités de gestion de groupe afin de créer, gérer et collaborer de manière efficace en groupe, répondant ainsi à leurs besoins personnels et professionnels. En outre, l'emploi de la réponse automatique Les interactions avec les utilisateurs seront améliorées grâce à l'IA qui fournira des informations intelligentes et contextuelles, ce qui améliorera l'efficacité de la communication et la satisfaction des utilisateurs.

Les perspectives de l'application Midad incluent l'intégration de bots intelligents et la traduction en temps réel.

RÉFÉRENCES CONSULTÉES

- [1] SmoothAlgo, logo, Consulté en avril 2024, <https://smoothalgo.com/>.
- [2] Modéliser une application web, livre, Consulté en avril 2024, [Modeliser_une_application_web.pdf](#)
- [3] Telegram, page d'accueil, Consulté en avril 2024, <https://web.telegram.org/>.
- [4] Whatsapp, page d'accueil, Consulté en mai 2024, <https://www.whatsapp.com/>.
- [5] Messenger, page d'accueil, Consulté en mai 2024, <https://www.messenger.com/>.
- [6] Modéliser une application web, livre, Consulté en mai 2024, [Modeliser_une_application_web.pdf](#).
- [7] Xourse, image, Consulté en juin 2024, <https://medium.com/@Xourse/reactjs-app-architecture-7a33d7ae9834>.
- [8] Midad Gitlab, capture, Consulté en juin 2024, <https://gitlab.com/smoothcomm/midad-web>.
- [9] Approche CD/CI, image, Consulté en juillet 2024, <https://www.tatvasoft.com/blog/ci-cd-jenkins/>, Approche CD/CI avec Jenkins.
- [10] Pipeline de déploiement continu, capture, Consulté en juillet 2024, 'Tableau de bord Midad' .

Résumé

Ce projet, réalisé dans le cadre du projet de fin d'études à l'École EPI, vise à développer une application web innovante nommée "Midad". Conçu au sein de l'entreprise SMOOTHALGO, ce projet fait partie d'une formation en ingénierie informatique, avec une spécialisation en génie logiciel. L'application "Midad" se distingue par ses fonctionnalités avancées de gestion de communication, incluant l'initiation et l'acceptation d'appels vidéo, la gestion des amis, des groupes de discussion, et l'échange sécurisé de messages grâce à un cryptage de bout en bout. L'objectif principal de "Midad" est d'améliorer la satisfaction des utilisateurs en leur offrant une expérience intuitive, sécurisée et fluide, que ce soit pour une conversation entre abonnés ou au sein de groupes de discussion.

Mots clés : MongoDB, React, Node.js, Express, Socket.IO.

Abstract

This project, carried out as part of the final year project at École EPI, focuses on developing an innovative web application named "Midad." Created within the company SMOOTHALGO, this project is a component of a training program in computer engineering, specializing in software engineering. The "Midad" application stands out for its advanced communication management features, including initiating and accepting video calls, managing friends and discussion groups, and securely exchanging messages through end-to-end encryption. The primary goal of "Midad" is to enhance user satisfaction by offering an intuitive, secure, and seamless experience, whether for conversations between subscribers or within discussion groups.

Keywords: MongoDB, React, Node.js, Express, Socket.IO.