République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Université Abderrahmane Mira Bejaia
Département d'informatique

Mémoire de fin de cycle

En vue de l'obtention du diplôme Master professionnel en Informatique

Spécialité : Génie logiciel

Thème :

Développement d'une plateforme multicritères de diffusion d'alertes multimédia

Réalisé par :

- Mr. ATILOUS Missipssa
- Mr. AMEUR Ali

Jury :

Président : Mr. SAADI Mustapha

Examinateur : Mr. ATMANI Mouloud

Encadrant : Mr. ZERARGA Loutfi

Promotion 2022/2023

Remerciements

Au terme de ce travail nous tenons tout d'abord à remercier :

Notre Dieu, le tout puissant de nous avoir donné le courage, la force et la volonté d'achever ce modeste travail.

Nous Adressons nos vifs remerciements :

A nos parents et à nos frères pour leurs soutient et énormes sacrifices tout au long de notre formation. Nous leur sommes infiniment reconnaissants.

A notre encadreur Mr ZERARGA Loutfi, enseignant à la Faculté de Sciences Exactes de Bejaia, pour son encadrement, ses conseils et pour ses directives précieuses.

A toute personne qui a contribué de près ou de loin à la réalisation de ce travail, nous les remercions vivement.

Dédicaces

ATILOUS Missipssa : Je dédie ce modeste travail à mon chère grand père Larbi paix à son âme. A mes parents et mes frères pour leurs soutient et encouragement. A ma petite sœur en lui souhaitant un grand succès dans sa vie. À tous mes amis sans exception.

AMEUR ALI : Je dédie ce travail à mes chers parents pour tous leurs sacrifices et leur amour durant mes études. À mes frères et sœurs pour leurs soutient moral. A tous mes amis qui ont contribué à la réalisation de ce travail.

Table des matières

Liste des figures

Liste des tableaux

Introduction générale

Les avancées technologiques ont été largement propulsées par Internet. L'émergence d'Internet a donné naissance à une multitude de changements dans notre vie. Les sites web et les applications en ligne sont des résultats concrets de cette révolution. Ces derniers ont ouvert de nouvelles portes en permettant aux gens d'accéder à une abondance de services avec facilité. Des tâches autrefois complexes sont devenues simples grâce à ces applications web. De la gestion de nos finances à la communication avec nos proches, en passant par la planification de nos emplois du temps, les applications web ont étendu nos capacités et ont transformé notre manière de vivre au quotidien.

De nos jours, ces applications web permettent également de sauver des vies, non seulement grâces aux application médicales mais aussi grâce aux applications de diffusion d'alertes. La prévention des catastrophes naturelles, des accidents et de la criminalité est un moyen efficace pour l'État et pour les citoyens de maitriser ce genre d'évènements et de rester en sécurité. En restant à un niveau superficiel, nous pouvons aussi alerter la population des évènements mineurs tel que le bulletin de météo, de l'état des plages et de l'état des routes.

Une alerte est un message critique et qui doit être lu rapidement, il existe plusieurs plateformes de diffusion d'alertes qui connaissent un franc succès, cependant il n'y en a aucune en Algérie, c'est dans ce cadre que s'inscrit notre projet de fin de cycle qui consiste à créer une application de diffusion d'alertes exclusif en Algérie. Nous allons utiliser un middleware orientée message pour relier entre des entités qui publient ces alertes et celles qui les reçoivent. Les destinataires des alertes ne reçoivent que des messages pertinents et ciblés en s'abonnant a des chaines qui publient des alertes sur des sujets bien précis.

Ce mémoire est organisé en quatre (4) chapitres :

Le *chapitre 1* intitulé 'Généralités' définit les plateformes de diffusion multicritère et donne certains exemples existants. Ce chapitre présente les Middlewares orienté message.

Le *chapitre 2* intitulé 'Spécification des besoins' est voué à l'identification des acteurs et les besoins fonctionnels/non fonctionnels ainsi qu'aux diagrammes de cas d'utilisation.

Le *chapitre 3* intitulé ‘Conception’ est dédié à la présentation des diagrammes de séquence, du diagramme de domaine et au schéma de base de données.

Dans le *chapitre 4* intitulé ‘Réalisation’ nous allons présenter les différents outils et technologies utilisé lors du processus de réalisation de notre projet.

Le mémoire s’achève par une conclusion générale et des perspectives.

PREMIER CHAPITRE

Chapitre 1 : Généralités

1.1- Introduction

Dans ce chapitre, nous allons présenter des généralités et des notions de bases qui tournent autour de notre sujet, nous définirons d'abord ce qu'est une plateforme multicritère de diffusion et présenterons quelques exemples. Puis nous nous intéresserons aux middlewares orienté message et plus particulièrement à RabbitMQ.

1.2- Plateforme multicritère de diffusion

Une plateforme est une solution logicielle destiné à répondre aux requêtes des utilisateurs s'y connectant et de stocker l'ensemble des donnés nécessaire à son bon fonctionnement. Le terme multicritère fait référence à une méthode qui évalue ou analyse un élément en prenant en compte plusieurs paramètres afin d'offrir une appréciation globale et pertinente.

1.3- Quelques plateformes de diffusion d'alertes qui existent

Voici quelques exemples des plateformes de diffusion d'alertes les plus populaires.

1.3.1- Everbridge

Everbridge a été fondé en 2002, à la suite des attentats du 11 septembre 2001, en se donnant la mission d'assurer la sécurité des personnes dans les situations de crise. Everbridge est une plateforme bien connue dans le domaine de la gestion des urgences et des alertes. Elle fournit des solutions de communication d'urgence et de gestion de crise pour les entreprises, les gouvernements et les organisations dans plus de 200 pays. La plateforme d'Everbridge permet d'envoyer des alertes par SMS, par e-mail, par appel vocal, par notifications mobiles et par d'autres canaux pour informer rapidement les personnes concernées en cas d'urgence ou d'incident grave. Les services spécifiques et les fonctionnalités disponibles peuvent varier en fonction de l'abonnement et des besoins de l'utilisateur [1].

Figure 1. Gestion de crises d'Everbridge.

1.3.2- Child Focus

Fondée en 1998, Child Focus est une organisation belge à but non lucratif qui se consacre à la protection des enfants et à la lutte contre la disparition et l'exploitation des enfants. La fondation collabore avec les autorités est coordonne le programme "Alertes Enlèvements" qui est utilisé pour diffuser des avis d'enlèvement d'enfants. Les avis d'enlèvement sont diffusés via différents canaux, y compris les médias, les réseaux sociaux, les sites web partenaires, et les téléphones mobiles grâce à un système de diffusion d'alertes SMS [2].

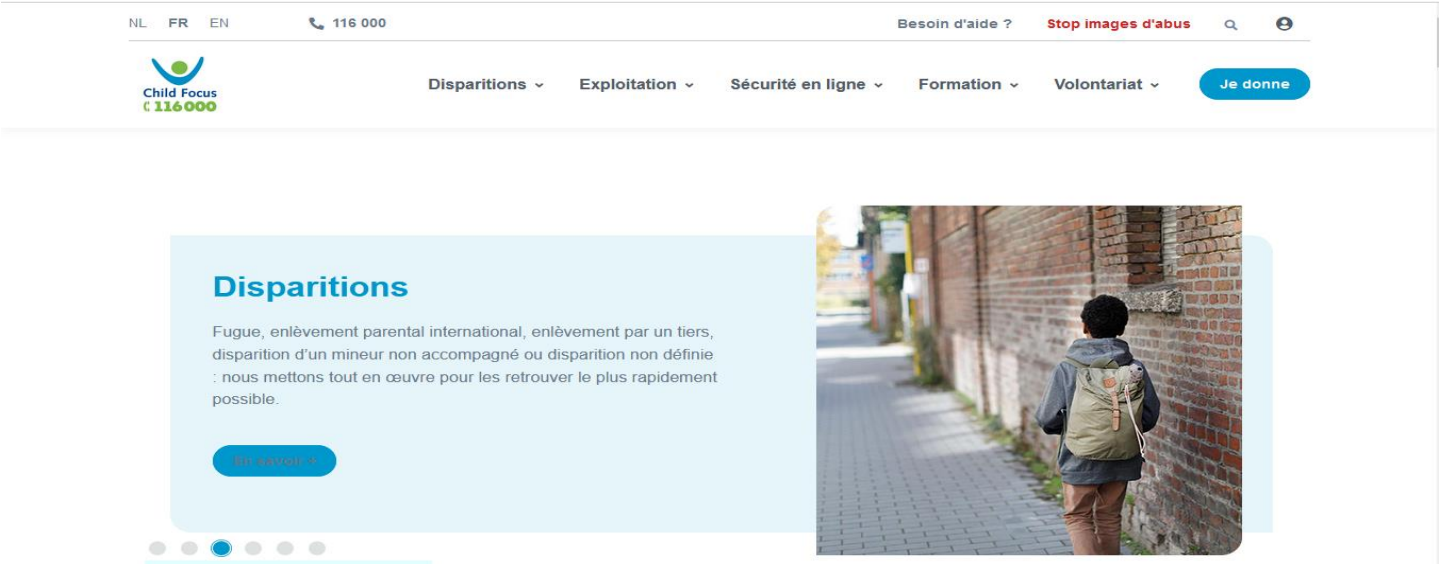

Figure 2. Le programme alerte enlèvement de Child focus.

1.3.3- Nixle

Fondée en 2007 aux états unis, Nixle est destinée aux Résidents, aux entreprises locales et aux agences gouvernementales. Nixle permet une communication bidirectionnelle en temps réel par SMS, par e-mail, par messages vocaux, par les réseaux sociaux et par l'application mobile Nixle.

Le système de notification Nixle est utilisé par plus de 8000 agences, services d'incendie et de police, écoles, hôpitaux aux USA. Les organisations utilisent Nixle pour des situations critiques telles que les événements météorologiques violents, les évacuations, les risques pour la sécurité, les menaces pour la sécurité, les problèmes d'installations, les notifications des employés et les perturbations informatiques/télécom [3].

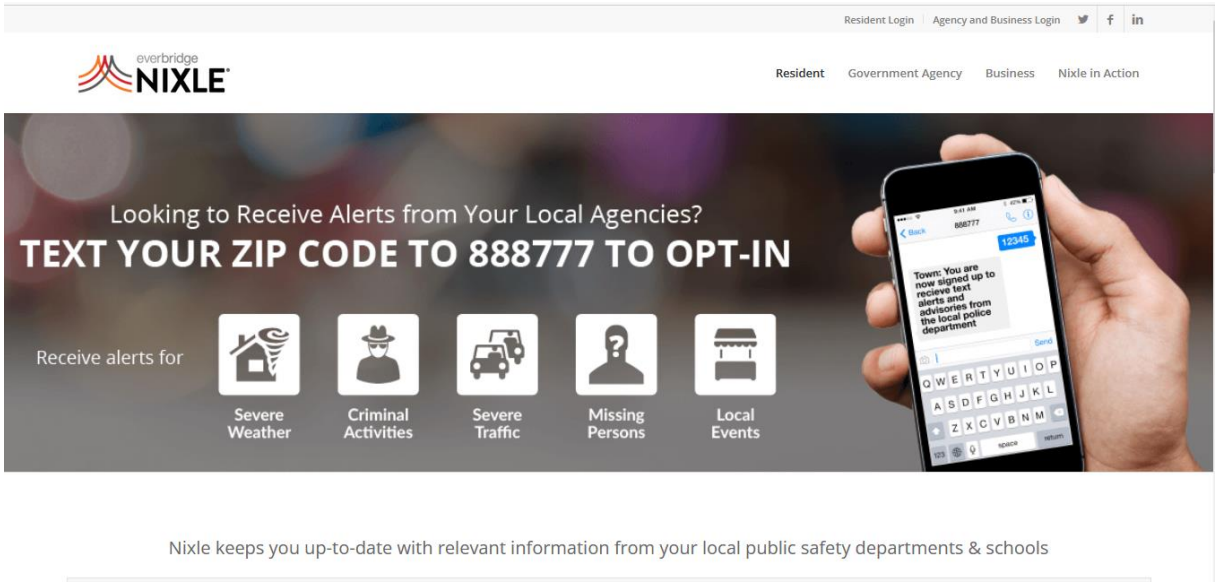

Figure 3. Page d'accueil de Nixle.

1.4- Middleware orienté message

Le middleware orienté message (MOM) est un logiciel qui facilite l'envoi et la réception de messages entre les différents systèmes d'information. Il est utilisé dans les environnements client/serveur où un courtier de messages spécialisé achemine les messages provenant de plusieurs clients vers les applications serveur correspondantes et en transmettant également les réponses des applications serveur aux clients appropriés. Cette approche crée une couche de communication qui permet aux développeurs de se concentrer sur les fonctionnalités des applications plutôt que sur la complexité des systèmes d'exploitation et des protocoles réseau.

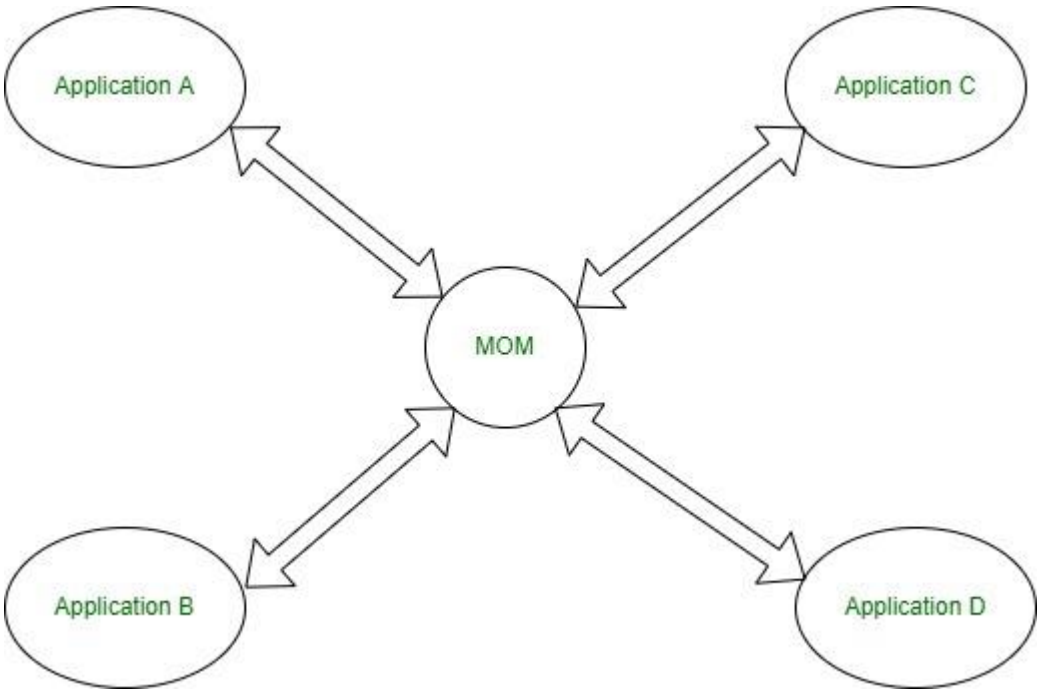

Figure 4. Le MOM interconnectant plusieurs applications.

Le MOM est particulièrement utile lorsque des changements importants se produisent dans une organisation, car plusieurs nouveaux systèmes et produits y sont introduit. Il serait coûteux en temps et en argent de reconfigurer ou de reconstruire l'infrastructure informatique pour s'assurer que tous les systèmes, nouveaux et anciens, fonctionnent ensemble. Un MOM peut combler cet écart et faciliter la communication entre différents systèmes et logiciels sans reconstruire l'infrastructure informatique d'une organisation. Les situations où le MOM se révèle utile sont lors du lancement de nouveaux services, l'adoption de nouveaux services, ainsi que les situations d'acquisition ou de fusion d'entreprises.

Le MOM aide à résoudre les problèmes de performances et d'autres problèmes potentiellement complexes liées à l'interopérabilité et à l'évolutivité, qui peuvent survenir dans un réseau de systèmes en constante évolution. Cette technologie est devenue d'une importance capitale dans les applications IoT[1] (Internet of Things) où les courtiers de messages centralisés facilitent la communication d'appareil à appareil.

[1] IoT est un réseau d'objets physiques connectés à Internet, permettant la collecte et l'échange de données

Le middleware est donc un moyen d'interconnecter asynchroniquement deux ou plusieurs applications et de simplifier la complexité des échanges entre eux.

1.4.1- Avantages des MOM

De ce que nous avons vu précédemment, les MOM offrent plusieurs avantages dans l'intégration des systèmes et de la communication entre eux. Leur capacité à résoudre des problématiques telles que la sécurité, la fiabilité et l'évolutivité ont fait de ces MOM des outils précieux pour les organisations modernes. Parmi les avantages reconnus des MOM figurent les suivant

- **Communication Asynchrone :** les applications peuvent envoyer des messages et continuer à fonctionner sans attendre de réponse immédiate, ce qui améliore l'efficacité et la flexibilité du système.
- **Fiabilité de Communication :** Les messages peuvent être stockés dans des files d'attente et livrés de manière fiable aux destinataires même en cas de panne temporaire ou de congestion du réseau.
- **Sécurité :** Protège la confidentialité et l'intégrité des messages échangés. Ils permettent également l'authentification, le chiffrement des données et la gestion des autorisations, ce qui assure une communication sécurisée entre les différents composants du système.
- **Évolutivité :** Les MOM offrent une évolutivité élevée pour répondre aux changements des besoins des organisations. Ils permettent d'ajouter de nouveaux composants ou de mettre à l'échelle les capacités de traitement sans nécessiter de modifications majeures dans l'architecture globale du système. Cela offre une plus grande souplesse pour s'adapter à la croissance des volumes de données et aux exigences de performance.
- **Flexibilité :** Les MOM offrent une flexibilité élevée. Ils permettent d'ajouter, de supprimer ou de modifier les composants du système de manière indépendante, sans perturber l'ensemble de l'architecture distribuée. Cela permet aux entreprises d'adapter rapidement leur infrastructure aux changements et aux évolutions technologiques

1.4.2- Désavantages des MOM

Malgré les avantages des MOM comme RabbitMQ, IBM MQ et Appache kafka (nous les verrons dans la section qui suit), ils présentent néanmoins quelques désavantages :

- Requière au préalable l'installation, la configuration et la personnalisation du logiciel.

- Complexe à utiliser et nécessite des services de la part de consultants expérimentés pour l'intégration du MOM.
- Latence élevée due au temps nécessaire pour transmettre, mettre en file d'attente et livrer le message, par conséquent ne convient pas aux applications critiques.
- Surcharge du réseau en particulier lorsque de nombreux messages sont échangées.

Il est important de noter que ces inconvénients ne sont pas nécessairement présents dans tous les cas d'utilisation des middlewares orientés messages. Ils dépendent du contexte spécifique, de la mise en œuvre et des exigences du système dans lequel ils sont utilisés.

1.4.3- Exemples de quelque MOM

1.4.3.1- RabbitMQ [4]

RabbitMQ est un logiciel open source et gratuit développé par Pivotal Software Inc. C'est un courtier (Broker[2]) de messages se basant sur le standard AMQP (Advanced message queuing protocol), nous détaillerons ce protocole d'avantage dans ce qui suit.

RabbitMQ est l'un des courtiers de messages open source les plus populaires, il est léger et facile à déployer dans les serveurs interne ou dans le cloud.

Figure 5. Logo de RabbitMQ

1.4.3.2- Apache Kafka [5]

Kafka est un projet initialement développé par LinkedIn, utilisé par Airbnb et d'autres grandes entreprises, il est actuellement géré par la fondation apache. Il s'agit d'une plateforme de streaming distribuée, conçue pour la gestion et le traitement de flux de données en temps réel à grande échelle. Apache Kafka est un projet open source et gratuit, distribué sous la licence Apache 2.0.

[2] Entité logicielle qui facilite l'échange de messages entre des applications.

Figure 6. Logo de Apache Kafka.

1.4.3.3- IBM MQ [6]

Anciennement connu sous le nom de WebSphere MQ, IBM MQ est un logiciel payant et disponible depuis plus de 25ans il est développé et géré par IBM[3]. Ce logiciel offre une messagerie fiable et sécurisée pour les environnements distribués.

Figure 7. Logo de IMB MQ.

1.4.4- Tableau comparatif

Nous allons présenter les principales différences entre les MoM cité dans un tableau comparatif

MOM	Apache kafka	RABBITMQ	IBM MQ
Avantages	-Open source -Faible latence -Haut débit	-Grande vitesse -Volume élevé -Documentation détaillé	-Intégration Simplifié -Fiabilité élevé -Hautement évolutif
Désavantages	-Réduit les performances -Se comporte maladroit	-Difficile à personnaliser -Besoins documentaires	-Mauvais support client -Configuration complexe
Open source	Oui	Oui	Non

Tableau 1. Tableau comparatif entre quelques MOM

[3] International Business Machines Corporation

1.5- Présentation de RabbitMQ

Nous optons pour l'utilisation de RabbitMQ pour plusieurs raisons. Il offre des moyens divers qui facilitent l'envoi et la réception des messages tel que les filles d'attentes, les échanges fanout, les échanges topic, etc. Nous verrons ces éléments en détail dans ce qui suit. RabbitMQ dispose également d'autres fonctionnalités avancées comme la gestion des priorités des messages et de leur durabilité. Ce dernier possède une documentation détaillée et une vaste communauté qui pourra répondre aux problèmes qui pourraient nous arriver lors de la phase de réalisation. Enfin, étant gratuit et en plus de tous ces avantages. RabbitMQ reste le choix idéal pour nous.

1.5.1- Les composants principaux de RabbitMQ

RabbitMQ possède plusieurs composants essentiels qui travaillent ensemble pour assurer la communication entre les applications.

1.5.1.1- Producer

Les producteurs sont les applications ou les services qui envoient des messages à RabbitMQ. Ils utilisent un client RabbitMQ pour publier des messages sur des files d'attente ou des échanges.

1.5.1.2- Queues

Ou bien files d'attentes sont l'endroit où sont stocker les messages envoyés par les producteurs en attente d'être consommé ou expiré.

1.5.1.3- Exchanges (échanges) :

Composant qui reçoit les messages des Producteurs et les acheminent aux queues. Ils ont plusieurs types différents : Direct, Fanout, topic, Headers.

L'exchange Direct route les messages vers des files d'attente en utilisant une clé de routage. Cette clé est un simple string qui indique le type de routage.

L'exchange Fanout diffuse un message à toutes les files d'attente liées à cet échange, sans tenir compte des clés de routage ou des critères de correspondance, permettant une diffusion à grande échelle.

L'exchange topic permet de router les messages en fonction de mots-clés ou de motifs spécifiques présents dans les clés de routage des messages, permettant une correspondance flexible et basée sur des critères de mot-clé. Le fonctionnement de cet exchange est similaire à celui de l'exchange direct

L'exchange Headers utilise des arguments avec des en-têtes et des valeurs optionnelles pour acheminer les messages. Les échanges basés sur les en-têtes sont identiques aux échanges de type "topic", à la différence qu'au lieu d'utiliser des clés de routage, les messages sont routés en fonction des valeurs des en-têtes.

1.5.1.4- Consumer :

Les Consommateurs sont les clients qui s'abonnent aux files d'attentes et récupèrent les messages pour les consommer et traiter.

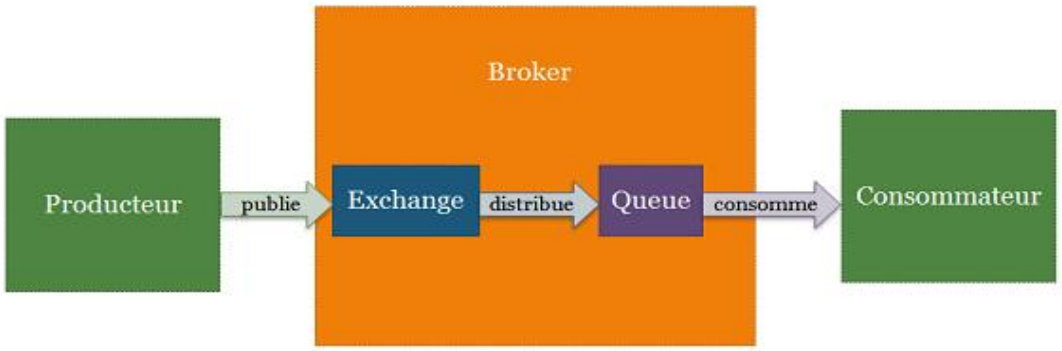

Figure 8. Fonctionnement de RabbitMQ

1.5.1.5- Bindings (Liaisons)

Les liaisons relient les échanges aux files d'attentes en définissant les règles de routages basées sur le type de message, les clés de routage (Rooting keys[4]) et les files d'attentes de destination. En d'autres termes, le binding est un lien qui dit que tel échange est connecté à tel fil d'attente.

1.5.1.6- VHost

Le rôle du VHost est de créer différentes instances pour contrôler l'accès aux ressources de RabbitMQ telles que les filles d'attentes et les échanges en plusieurs domaines (isolation et séparation)

1.5.1.7- Permissions

RabbitMQ gère les autorisations pour contrôler l'accès des utilisateurs aux différentes ressources (queues, échanges, vhost…) et détermine les opérations qu'ils peuvent effectuer (Administration, écriture, lecture).

[4] Permet au système de messagerie RabbitMQ de déterminer qui recevra une copie du message.

1.5.2- Les protocoles de communication

RabbitMQ Utilise principalement le protocole AMQP pour sa communication, mais prends également en charge d'autre protocoles comme HTTP/REST, MQTT et STOMP.

1.5.2.1- AMQP (Advanced Message Queuing Protocol)

S'agissant d'un protocole d'échange de message on retrouve naturellement la notion de file ou Queue ! Il s'agit d'une liste FIFO (First In First Out) dans laquelle sont ajoutés les messages qui seront ensuite redistribués aux différents consommateurs connectés à cette file.

Une différence fondamentale d'AMQP avec les autres protocoles est qu'un message n'est jamais envoyé directement à une file ! Les producteurs publient (publish) les messages dans des zones d'échanges ou Exchange. L'Exchange joue le rôle d'un aiguilleur, il est possible d'aiguiller le même message sur plusieurs Queue. Les consommateurs s'inscrivent (subscribe) aux files pour recevoir les messages.[7]

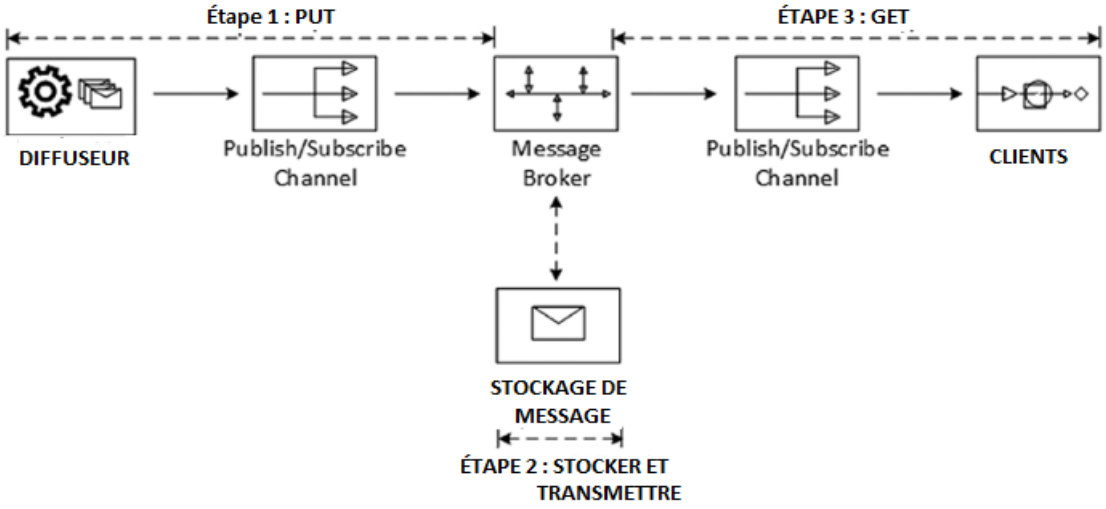

Figure 9. Le protocole AMQP

1.5.2.2- STOMP (Simple/Streaming Text Oriented Messaging Protocol)

STOMP est un protocole de messagerie simple qui permet aux clients de se connecter à des courtiers de messagerie, y compris RabbitMQ. Il fournit des commandes pour envoyer et recevoir des messages, ainsi que pour gérer les files d'attente et les échanges.

1.5.2.3- HTTP/REST

RabbitMQ propose une API HTTP/RESTful qui permet aux clients d'interagir avec le courtier en utilisant des requêtes HTTP. Cette interface facilite l'intégration avec des langages de programmation et des Framework qui prennent en charge les appels HTTP.

1.5.2.4- MQTT (Message Queuing Telemetry Transport)

MQTT est un protocole de messagerie basé sur la publication/souscription qui utilise le protocole TCP/IP. Il permet la communication entre deux appareils distants via des messages, avec une faible consommation de bande passante, une charge de données réduite et une faible consommation d'énergie. RabbitMQ offre une extension pour prendre en charge MQTT, son intégration lui permet d'interagir avec d'autres systèmes et applications qui utilisent MQTT.

1.6- Problématique

Chaque pays dans le monde dispose de moyens pour communiquer rapidement avec les citoyens pour les alerter des cas d'urgences et des cas particuliers qui pourraient atteindre à leur sécurité. L'un de ses moyens est les plateformes de diffusion d'alertes.

Notre projet de fin de cycle consiste à créer et développer une plateforme de diffusion d'alertes et qui permet aux utilisateurs de rechercher, évaluer et s'abonner aux sujets qui les intéressent en prenant en compte différents formats : les alertes par SMS ou par Email.

1.7- Conclusion :

Le middleware orienté message est donc un moyen d'interconnecter de manière asynchrone deux ou plusieurs applications. Dans notre plateforme, RabbitMQ jouera le rôle de passerelle entre le diffuseur et le client. Dans le prochain chapitre, nous modéliserons de manière simple les besoins de notre future plateforme avec des diagrammes et tableaux pour avoir une meilleure vue du projet.

DEUXIEME CHAPITRE

Chapitre 2 : Spécification des besoins

2.1- Introduction

Ce chapitre est dédié à l'expression et à la spécification des besoins de notre plateforme. Nous allons procéder à une analyse des besoins des utilisateurs. Nous commencerons par identifier les acteurs et les présenter dans le diagramme de contexte. Nous identifierons par la suite les besoins fonctionnels et les besoins non fonctionnels. Nous terminerons par la présentation du diagramme de cas d'utilisation globale, des diagrammes de cas d'utilisation détaillés et de leurs descriptions textuelles.

2.2- Expression des besoins

Nous avons opté pour la méthode UP (Unified Process) pour la gestion de notre projet, et avons choisi le langage UML (Unified Modeling Language) pour modéliser les différents diagrammes de notre analyse.

2.2.1- Identification des acteurs

Un acteur est une entité qui interagit directement avec le système pour réaliser une tâche. Ainsi durant notre analyse nous avons identifiés les différents acteurs qui interagissent avec notre système.

Acteur	**Rôle**
Administrateur	C'est l'administrateur principal qui s'occupe de la gestion de la plateforme et de son bon fonctionnement.
Diffuseur	Un utilisateur inscrit à la plateforme en tant que diffuseur possède une ou plusieurs chaînes dans lesquelles il diffuse du contenu. Ce contenu est acheminé vers les clients abonnés à ses chaînes spécifiques.
Client	C'est un utilisateur inscrit à la plateforme en tant que client, il possède un compte et peut accéder à certaines tâches qu'un simple visiteur ne peut pas faire tel que recevoir les alertes, voir le catalogue, l'abonnement et l'évaluation des chaines.

Visiteur	C'est un internaute qui consulte la page d'accueil et qui peut s'inscrire.

Tableau 1. Identification des acteurs du système

2.2.2- Diagramme de contexte statique

Un diagramme de contexte statique illustre les interactions entre un système et ses acteurs externes, sans se préoccuper des détails internes du système.

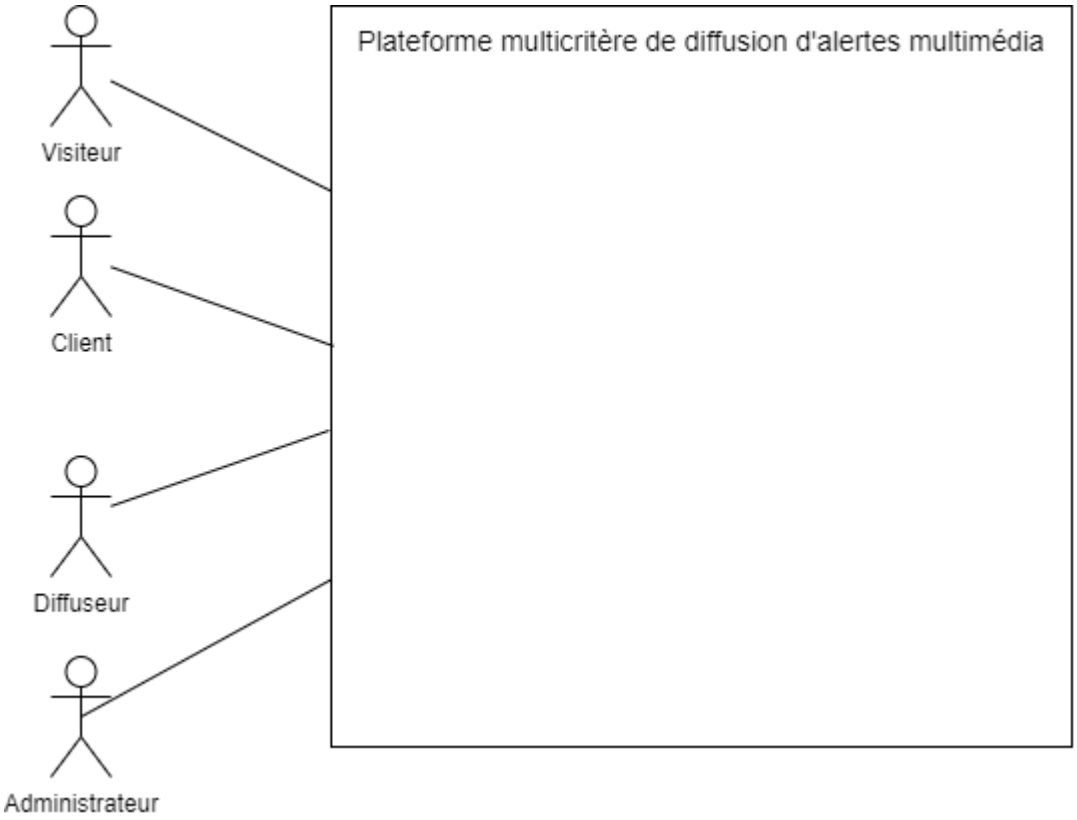

Figure 1. Diagramme de contexte statique.

2.2.3- Identification des besoins

L'identification des besoins fonctionnels et non fonctionnels permet une définition claire des objectifs en termes de fonctionnalités et de caractéristiques que le système doit atteindre. Ces besoins orientent ensuite les étapes suivantes du processus de développement.

2.2.3.1- Identification des besoins fonctionnels

Les besoins fonctionnels expriment les fonctionnalités concrètes de la plateforme, le tableau 3 ci-dessous représente les besoins fonctionnels de notre plateforme.

Tableau 2. Besoins fonctionnels de la plateforme.

Besoins	Description
Espace d'accueil	- Présentation de la plateforme et de ses avantages. - Affiche le catalogue des chaines aux utilisateurs.
Espace de connexion	- Permet aux utilisateurs et l'administrateur de se connecter ou de créer un compte.
Espace administrateur	- Permet à l'administrateur de gérer les réclamations, les signalements et les comptes. - Gérer les comptes. - Gérer les chaines.
Espace Diffuseur	- Gérer son profil - Permet au diffuseur de créer ou de supprimer une chaine. - Envoyer du contenu à ses abonnés via les chaines. - Soumettre une réclamation a l'administrateur. - Voir la liste des abonnés de ses chaines.
Espace Client	- Gérer son profil - Consulter le catalogue des chaines - Gérer ses abonnements - Evaluer et signaler une chaine - Soumettre une réclamation à l'administrateur

2.2.3.2- Identification des besoins non fonctionnels

Les besoins non fonctionnels sont les contraintes que le système doit respecter pour satisfaire le futur utilisateur, ce sont des indicateurs de qualité. Le tableau ci-dessous représente les besoins fonctionnels de notre plateforme.

Besoin	Description
Sobriété	- Simplicité du Design. - Plateforme peu chargé.
Rapidité	- Temps de chargement rapide. - Chargement des images au fur et à mesure. - Changement des valeurs immédiat après modification.
Lisibilité	- Clarté et aération du texte. - Structuration et organisation des éléments.
Utilisabilité	- Facilité et liberté de navigation.
Sécurité	- Garantir la sécurité d'accès aux comptes et aux différents espaces de la plateforme.

Tableau 3. Besoins non fonctionnels de la plateforme

2.3- Spécification des besoins

La spécification des besoins doit décrire sans ambigüité notre plateforme. Pour cela nous utiliserons un ensemble de documents (fiches descriptives) et modèles (diagrammes de cas d'utilisation) pour spécifier les différents besoins de notre système.

2.3.1- Diagramme de cas d'utilisation globale

Le diagramme de cas d'utilisation globale va donner une vue générale de tous les acteurs et leur cas d'utilisations.

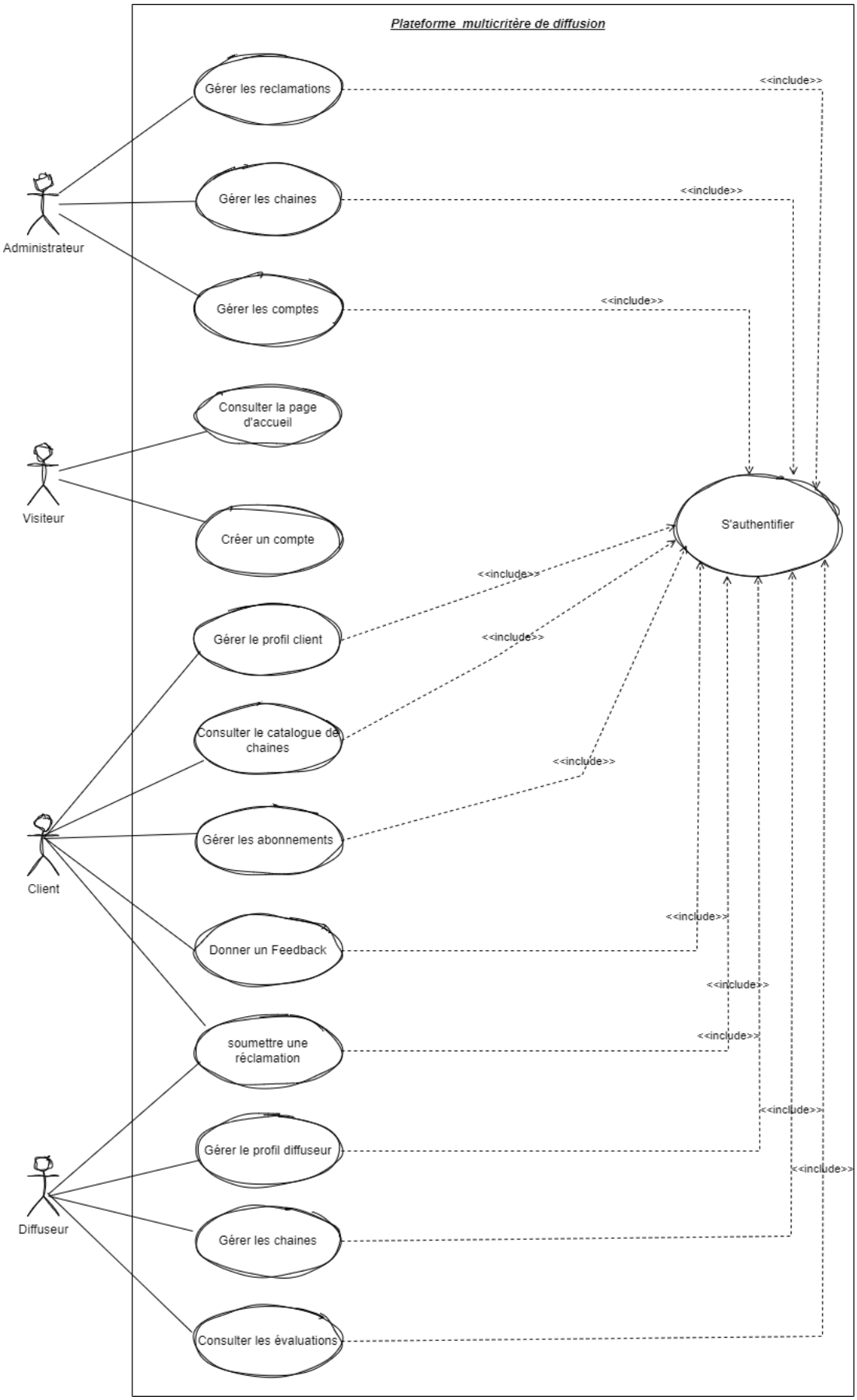

Figure 2. Diagramme de cas d'utilisation globale.

2.3.2- Diagramme de cas d'utilisation détaillé du « Administrateur »

Le diagramme suivant expose le rôle d'un administrateur, il gère les réclamations, gère les signalements de chaines et gère les comptes clients et diffuseur.

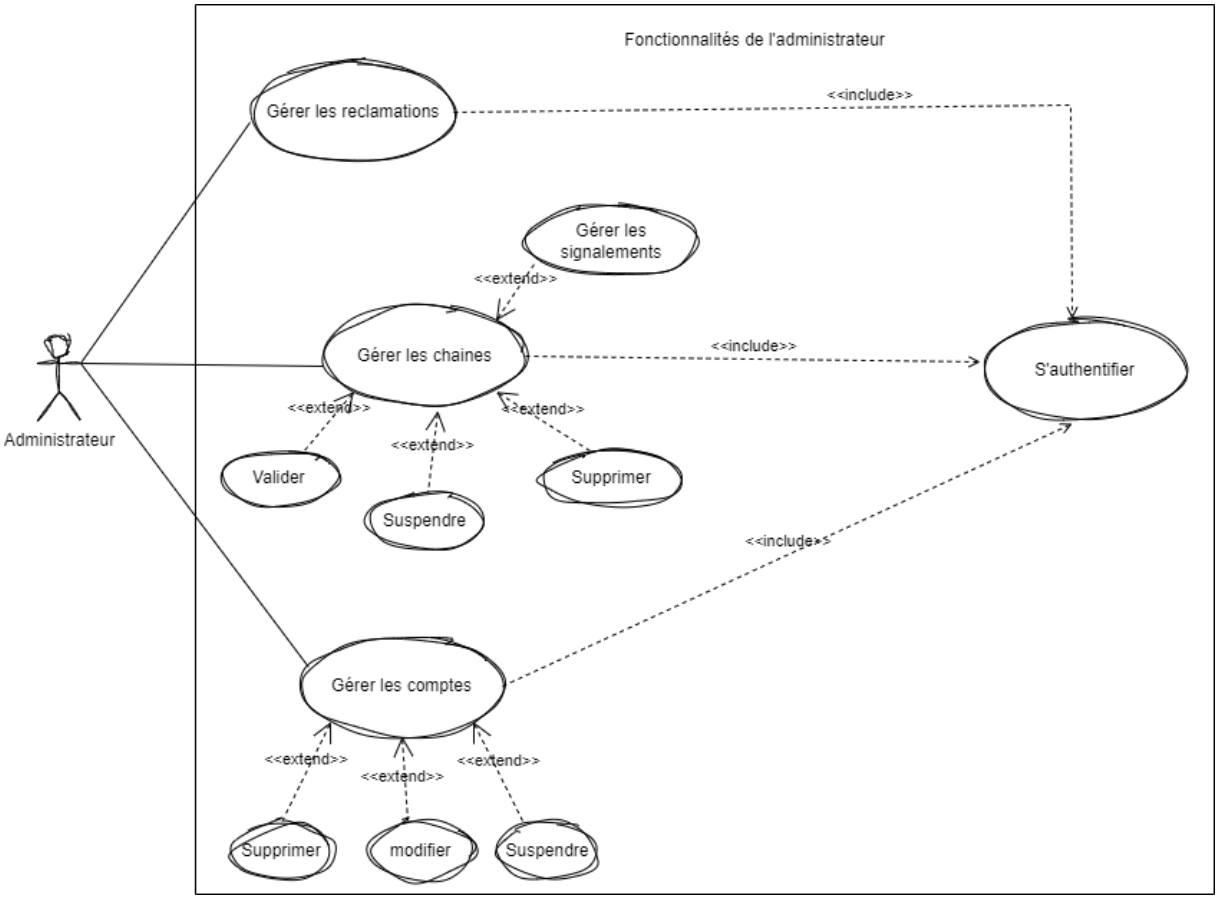

Figure 3. Diagramme de cas d'utilisation détaillé de l'administrateur

2.3.2.1- Description textuelle du cas « s'authentifier »

Sommaire d'identification	
Titre :	Authentification.
Résumer :	L'administrateur introduit son login et son mot de passe.
Acteur :	Administrateur.
Description des enchaînements	
Pré conditions	Poste conditions
L'administrateur doit avoir un compte.	Accès à la page d'accueil de l'administrateur.
Scenario nominal	

1- Demande d'accès au système. 2- Le système lui répond par l'affichage de la boite d'authentification. 3- L'administrateur saisie son login et son mot de passe. 4- Le système vérifie les champs. 5- Le système vérifie l'existence de l'administrateur. • Si l'administrateur existe dans la base de données alors il va réussir à s'authentifier, sinon le système renvoi un message d'erreur (l'administrateur n'existe pas). • Si l'authentification est réussie, il aura accès à la page d'accueil, sinon le système réinitialise la page d'authentification.
Enchainement alternatif
E1. Champs obligatoires vides. ➢ Le système affiche un message d'erreur (champ vide). ➢ Le scenario reprend de l'étape 3. E2. Login ou mot de passe non valide. ➢ Le système affiche un message d'erreur (login ou mot de passe non valide). ➢ Le scenario reprend de l'étape 3.

Tableau 4. Description textuelle du cas d'utilisation « s'authentifier »

2.3.2.2- Description textuelle du cas d'utilisation « Valider une chaine »

Sommaire d'identification	
Titre :	Valider une chaine.
Résumer :	L'administrateur choisit de valider ou pas une chaine crée par un diffuseur.
Acteur :	Administrateur.
Description des enchaînements	
Pré conditions	Poste conditions
S'authentifier en tant qu'administrateur.	➢ Si l'administrateur valide une chaine. Chaine ajoutée.
Scenario nominal	

1- L'administrateur s'authentifie. 2- L'administrateur accède à l'interface des chaines en attente de validation. 3- L'administrateur regarde la description de la chaine. 4- L'administrateur valide la chaine. 5- Le système ajoute la chaine au catalogue.
Enchainement alternatif
E1. L'administrateur refuse de valider la chaine. ➢ Le système affiche une notification au diffuseur (Chaine refusée)

Figure 4. Description textuelle du cas d'utilisation « valider »

2.3.2.3- Description textuelle du cas d'utilisation « Suspendre une chaine »

Sommaire d'identification	
Titre :	Suspendre une chaine.
Résumer :	L'administrateur suspend temporairement une chaine.
Acteur :	Administrateur.
Description des enchaînements	
Pré conditions	Poste conditions
S'authentifier en tant qu'administrateur.	Chaine suspendue.
Scenario nominal	
1- L'administrateur s'authentifie. 2- L'administrateur accède à l'interface de gestion des chaines. 3- L'administrateur sélectionne la chaine à suspendre. 4- L'administrateur sélectionne la durée et le motif de la suspension. 5- Le système ajoute la chaine à la liste des chaines suspendues. 6- Le système empêche le Diffuseur de modifier sa chaine ou de diffusé quoi que ce soit temporairement.	
Enchainement alternatif	
E1. Durée de suspension non précisée. ➢ Le système affiche un message d'erreur (Champ obligatoire)	

Figure 5. Description textuelle du cas d'utilisation « Suspendre »

2.3.3- Diagramme de cas d'utilisation détaillé du « Visiteur »

Le diagramme de cas d'utilisation suivant expose qu'un Visiteur ne peut que consulter la page d'accueil ou créer un compte en tant que Client ou Diffuseur.

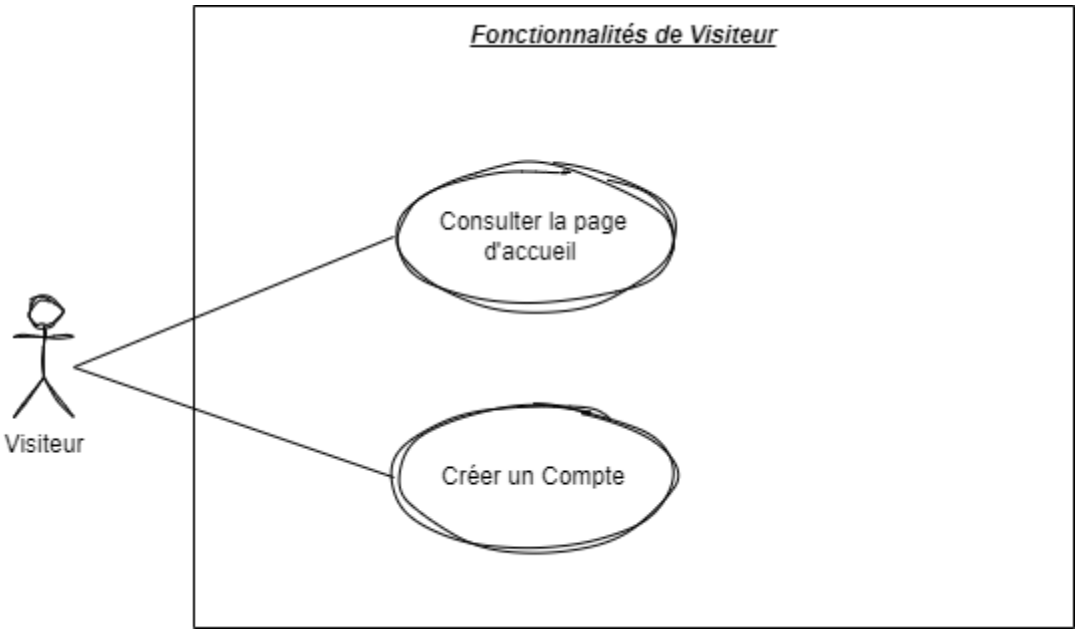

Figure 6. Diagramme de cas d'utilisation détaillé du visiteur

2.3.3.1- Description textuelle du cas d'utilisation « Créer un compte »

Sommaire d'identification	
Titre :	Créer un compte.
Résumé :	Avoir un compte et de profiter des services de la plateforme.
Acteur :	Visiteur.
Description des enchaînements	
Pré conditions	Poste conditions
Avoir accès à la plateforme.	Avoir un compte. Accès au reste des interfaces. Accès au catalogue et pouvoir s'abonner en tant que client. Créer des chaines après validation de l'administrateur et envoyer des alertes en tant que diffuseur.
Scenario nominal	

1- Consulter la page d'inscription. 2- Le visiteur remplit le formulaire. 3- Le système valide les informations. 4- Le système vérifie que l'email n'est pas déjà utilisé. 5- Le système enregistre les informations dans la BD. 6- Le système transfert le visiteur vers la page d'authentification.
Enchainement alternatif
E1. Champs obligatoires vides. Le système affiche un message d'erreur (champ vide). Le scénario reprend à l'étape 2 E2. E-mail déjà utilisé Le système affiche un message d'erreur (E-mail déjà utilisé) Le scénario reprend à l'étape 2.

Tableau 5. Description de diagramme de cas d'utilisation «Créer un compte » pour le visiteur

2.3.4- Diagramme de cas d'utilisation détaillé « Client »

Le diagramme de cas d'utilisation expose qu'un client après l'authentification peut gérer son profil, consulter le catalogue des chaines, gérer ses abonnements, donner un feedback et soumettre une réclamation à un Administrateur.

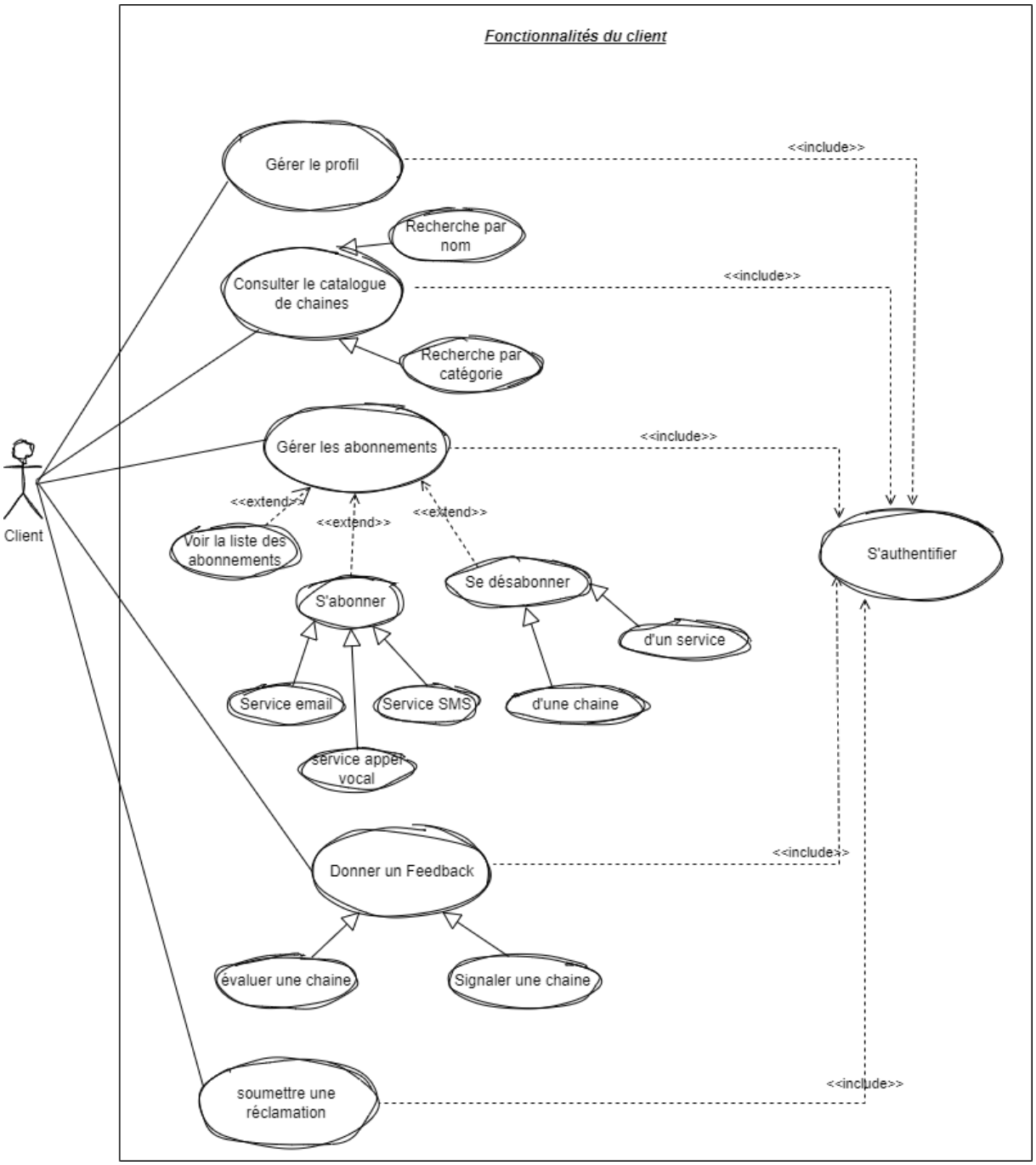

Figure 7. Diagramme de cas d'utilisation du client

2.3.4.1- Description textuelle du cas d'utilisation « Voir la liste des abonnements »

Sommaire d'identification	
Titre :	Voir la liste des abonnements.
Résumé :	Le client visualise une liste de toutes les chaines dont il est abonné.
Acteur :	Client.

Description des enchaînements	
Pré conditions	Poste conditions
Avoir un compte client.	Accès à la liste des abonnements du client.
Scenario nominal	
1- Le client demande à afficher la liste de ses abonnements. 2- Le système récupère les abonnements du client sous forme de tableau. 3- Le client visualise ses abonnements.	
Enchainement Alternatif	
E1. Le client n'est abonné à aucune chaine ➢ Le système ne retourne aucun résultat	

Tableau 6. Description textuelle du cas d'utilisation « Voir la liste des abonnements »

2.3.4.2- Description textuelle du cas d'utilisation « Evaluer une chaine »

Sommaire d'identification	
Titre :	Evaluer une chaine.
Résumé :	Le client note et commente une chaine.
Acteur :	Client.
Description des enchaînements	
Pré conditions	Poste conditions
Avoir un compte client.	Mise à jour de la moyenne d'évaluation de la chaine.
Scenario nominal	
1- Le client s'authentifie. 2- Consulter le profil de la chaine. 3- Cliquer sur le bouton « Evaluer » 4- Donner une note qui varie entre 0 à 5. 5- Rédiger un commentaire. 6- Le système enregistre la saisie. 7- Le système met à jour les évaluations de la chaine.	
Enchainement alternatif	

E1. Client non abonné à la chaine ➢ Le système affiche un message d'erreur (vous devez être abonné à la chaine pour l'évaluer). Le scenario reprend à l'étape 2 E2. Champs évaluation vide ➢ Le système affiche un message d'erreur (vous ne pouvez pas rédiger un commentaire sans donner une note). Le scenario reprend à l'étape 4

Tableau 7. Description textuelle du cas d'utilisation « Evaluer une chaine »

2.3.4.3- Description textuelle du cas d'utilisation « se désabonner d'une chaine »

Sommaire d'identification	
Titre :	Se désabonner d'une chaine.
Résumer :	Le client voit la liste de ses abonnements et se désabonne des chaines de son choix.
Acteur :	Client.
Description des enchaînements	
Pré conditions	Poste conditions
Le client doit avoir un compte. Être abonné à une chaine.	Le système met à jour ses abonnements
Scenario nominal	
1- Le client s'authentifie 2- Le client est sur la liste de ses abonnements. 3- Le client se désabonne d'une chaine. 4- Le système affiche un message de confirmation. 5- Le client confirme pour se désabonner de la chaine. 6- Le système met à jour sa liste d'abonnements. 7- Le système affiche la nouvelle liste.	

Tableau 8. Description textuelle du cas d'utilisation « Se désabonner d'une chaine »

2.3.5- Diagramme de cas d'utilisation détaillé du « Diffuseur »

Le diagramme expose qu'un diffuseur après authentification peut gérer son profil, gérer ses chaines, consulter les évaluations et soumettre une réclamation à un administrateur.

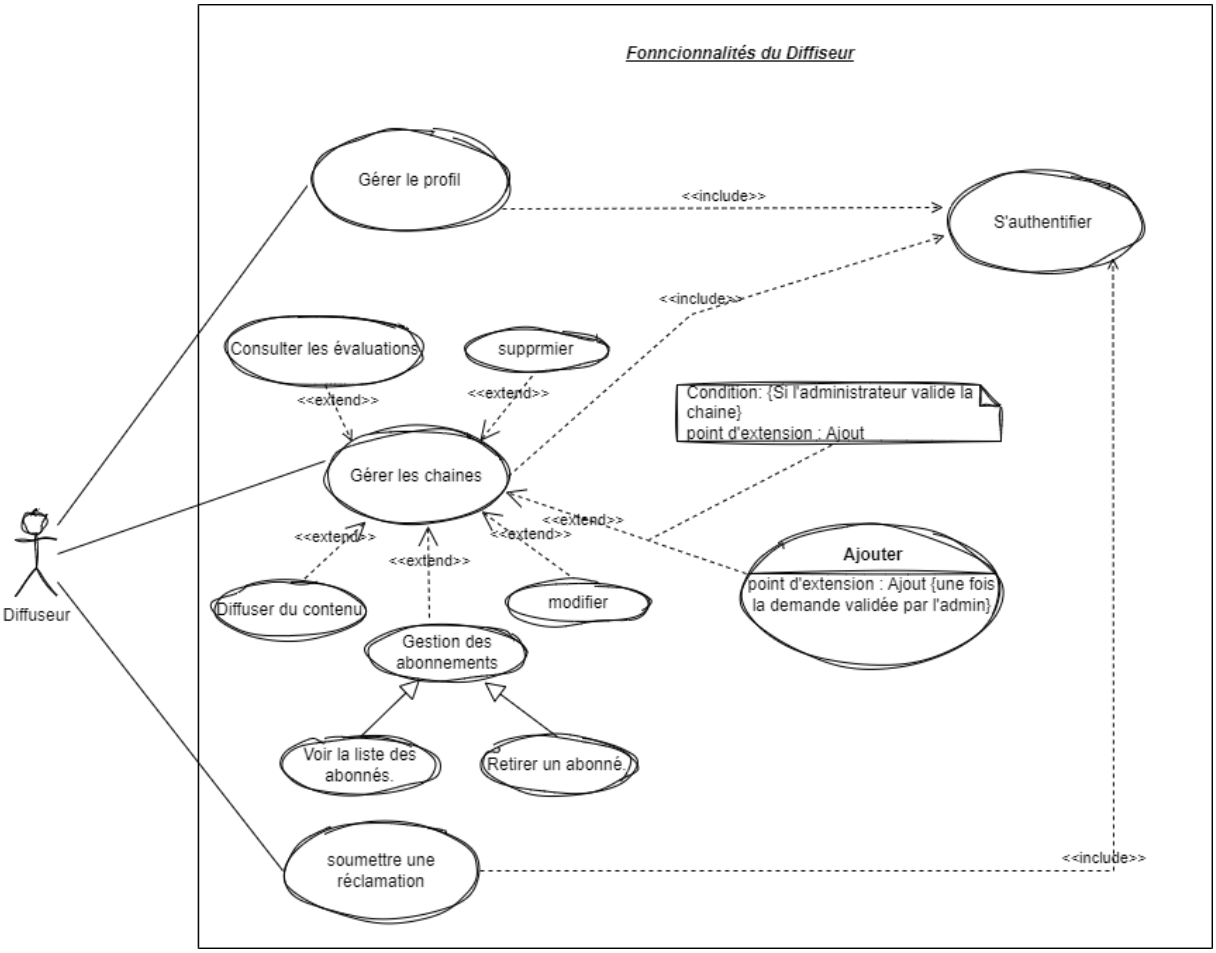

Figure 8. Diagramme de cas d'utilisation détaillé du Diffuseur

2.3.5.1- Description textuelle du cas « Ajouter une chaine »

Sommaire d'identification	
Titre :	Ajouter une chaine.
Résumer :	Le diffuseur créer une nouvelle chaine de diffusion d'alertes multimédia.
Acteur :	Diffuseur.
Description des enchaînements	
Pré conditions	Poste conditions
Avoir un compte diffuseur.	L'administrateur valide la chaine. Chaine ajoutée.

Scenario nominal
1- Le diffuseur s'authentifie. 2- Le diffuseur demande à créer une chaine. 3- Le diffuseur remplit le formulaire. 4- Le système récupère les informations. 5- Le système envoi une notification à l'administrateur. 6- L'administrateur valide la chaine. 7- Le système ajoute la chaine au catalogue. 8- Le diffuseur pourra désormais gérer sa chaine.
Enchainement alternatif
E1. L'administrateur refuse de valider la chaine. ➢ Le système affiche une notification (Chaine refusée) E2. Champs manquant lors du remplissage du formulaire ➢ Le système affiche un message d'erreur (champs vide)

Tableau 9. Description textuelle du cas d'utilisation « ajouter une chaine »

2.3.5.2- Description textuelle du cas d'utilisation « Diffuser du contenu »

Sommaire d'identification	
Titre :	Diffuser du contenu.
Résumer :	Envoyer du contenu (texte ou multimédia) aux abonnés de la chaine.
Acteur :	Diffuseur.
Description des enchaînements	
Pré conditions	Poste conditions
Avoir un compte diffuseur.	Contenu envoyé.
Scenario nominal	

1- Le diffuseur s'authentifie. 2- Le diffuseur sélectionne l'une des chaines qu'il gère. 3- Le diffuseur clique sur le bouton envoyer du contenu. 4- Le diffuseur sélectionne le contenu qu'il veut envoyer à partir de son ordinateur ou téléphone. 5- Le système récupère le contenu. 6- Le système envoie le contenu a RabbitMQ. 7- Le système affiche un message « contenu diffusé ».
Enchainement alternatif
E1. Type de fichier incompatible. ➢ Le système affiche un message d'erreur (type de fichier incompatible) Le scénario reprend à l'étape 4. E2. Fichier trop volumineux ➢ Le système affiche un message d'erreur (fichier trop volumineux) Le scénario reprend à l'étape 4. E3. Contenu vide ➢ Le système affiche une indication (vous ne pouvez pas envoyer du contenu vide) Le scénario reprend à l'étape 4.

Tableau 10. Description textuelle du cas d'utilisation « ajouter une chaine »

2.3.5.3- Description textuelle du cas d'utilisation « Voir la liste des abonnés »

Sommaire d'identification	
Titre :	Voir la liste des abonnés.
Résumer :	Afficher la liste des clients abonnés à une chaine spécifique.
Acteur :	Diffuseur.
Description des enchaînements	
Pré conditions	Poste conditions
Avoir un compte diffuseur.	Liste des clients abonnés à la chaine sélectionnée.
Scenario nominal	

1- Le diffuseur s’authentifie. 2- Le diffuseur sélectionne l’une des chaines qu’il gère. 3- Le diffuseur clique sur le bouton afficher les abonnés 4- Le système affiche la liste des abonnés.

Tableau 11. Description textuelle du cas d'utilisation « afficher la liste des abonnés »

2.4- Conclusion

L’expression et la spécification des besoins représentent la première phase du cycle de développement d’un logiciel. Elle sert à identifier les acteurs actifs du système et associer à chacun un ensemble d'actions avec lesquelles ils interviennent dans l’objectif de donner un résultat optimal et satisfaisant. Dans la phase d’expression, nous avons exprimé les besoins de notre plateforme en identifiant les acteurs et les fonctionnalités du système. Dans la phase de la spécification, nous avons spécifié plus clairement la relation entre les acteurs et leurs fonctionnalités, cette description est représentée à l'aide du diagramme de cas d’utilisation globale et des diagrammes détaillés. Dans le chapitre suivant, nous présenterons la phase de conception de notre plateforme.

TROISIEMME CHAPITRE

Chapitre 3 : Conception

3.1- Introduction

La conception représente une phase primordiale pour produire une application de haute qualité. Dans ce chapitre nous allons présenter quelques diagrammes de séquence essentiels de notre plateforme puis présenterons le modèle de domaine. Nous allons détailler ce dernier en modélisant le diagramme de classe et finirons par la présentation du schéma de base de données

3.2- Diagramme de séquence

Le diagramme de séquence représente l'aspect chronologique de l'échange des messages entre les utilisateurs et les entités qui composent le système.

Nous allons présenter les différents diagrammes de séquences regroupés par notre système :

- Diagramme de séquence du cas d'utilisation ‘’S'authentifier’’.
- Diagramme de séquence du cas d'utilisation ‘’Valider une chaine”
- Diagramme de séquence du cas d'utilisation ‘’ Voir la liste des abonnés ”.
- Diagramme de séquence du cas d'utilisation ‘’ Créer un compte ”.
- Diagramme de séquence du cas d'utilisation ‘’ Diffuser du contenu ”.

3.2.1- Diagramme de séquence du cas d'utilisation s'authentifier

L'authentification est la procédure qui consiste, à vérifier l'identité d'un utilisateur afin d'autoriser l'accès de ce dernier aux ressources demandées. Lors de cette action, il existe deux alternatives, si l'information est correcte, le système affiche l'espace demandé, sinon il affiche un message d'erreur.

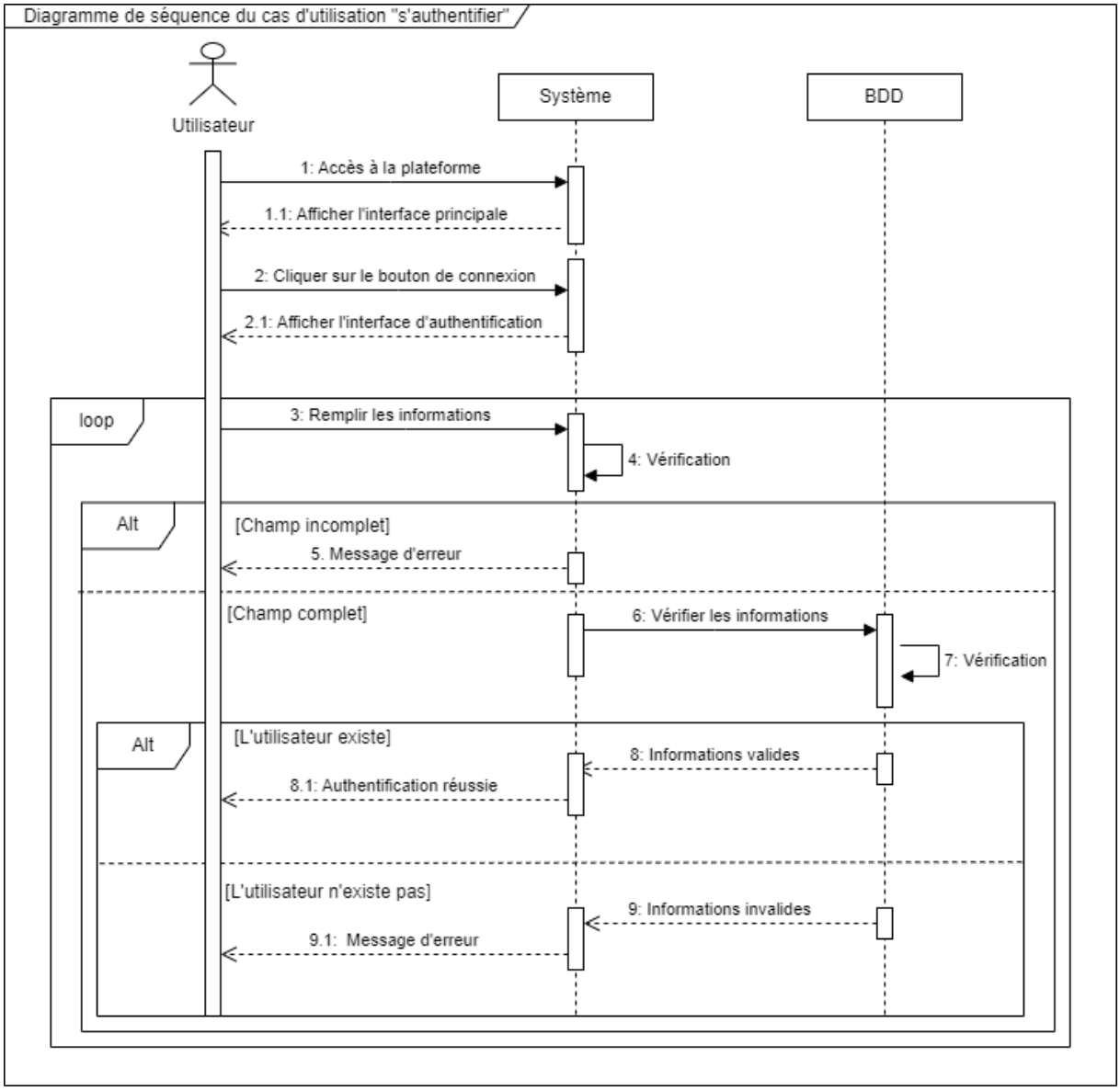

Figure 1. Diagramme de séquence du cas S'authentifier

3.2.2- Diagramme de séquence du cas d'utilisation « Valider une chaine »

Quand un diffuseur décide de créer une chaine, il devra attendre que l'administrateur la valide pour qu'il puisse l'utiliser et y diffuser du contenu. Dans le cas où la chaine n'est pas validée cette dernière se voit supprimée.

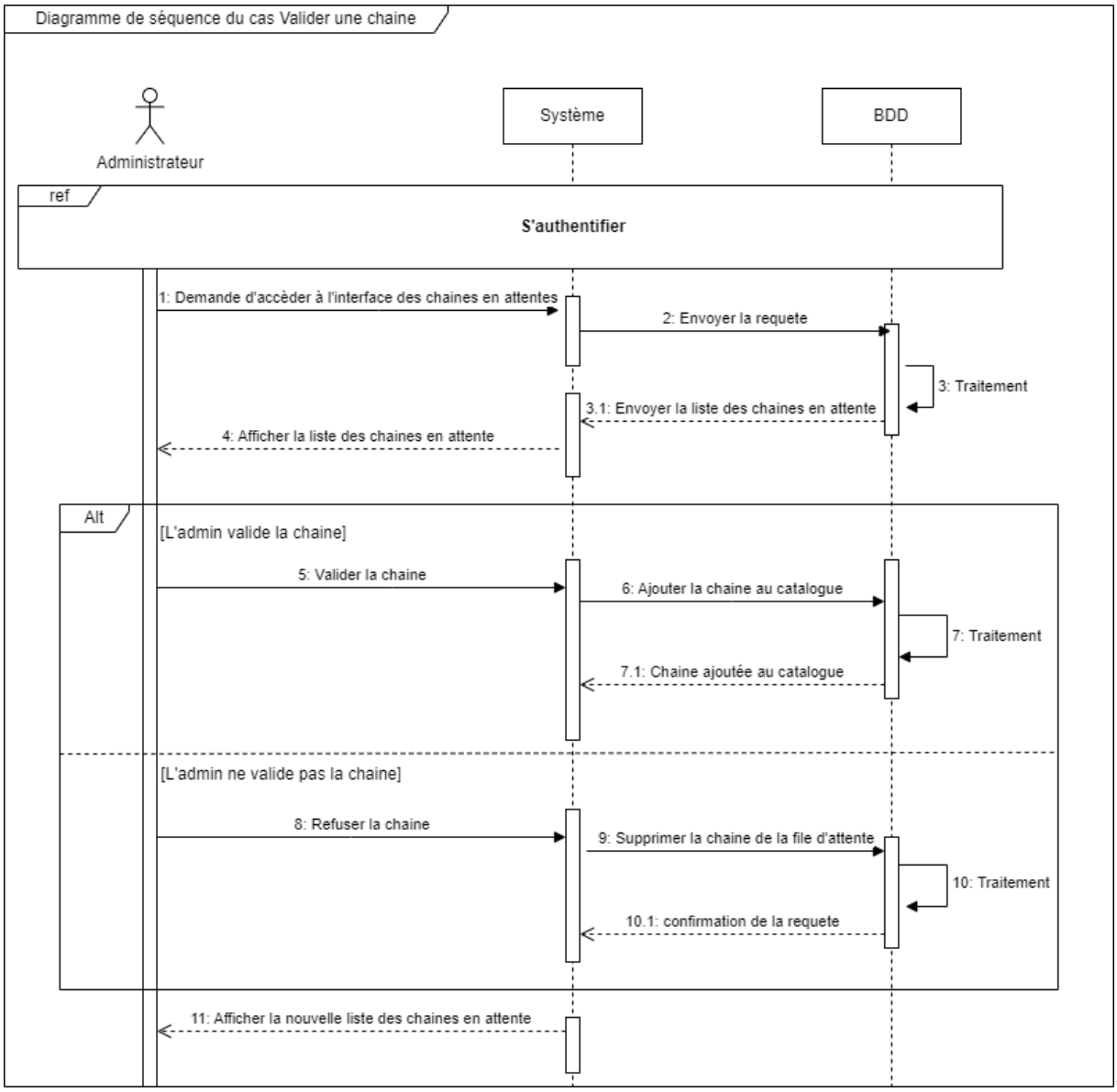

Figure 2. Diagramme de séquence du cas Valider une chaine

3.2.3- Diagramme de séquence du cas d'utilisation « créer un compte »

Quand un visiteur se rends à la plateforme il a la possibilité de devenir un utilisateur en créant un compte, soit en tant que diffuseur ou en tant que client. Le système enregistre alors le nouvel utilisateur à la base de données ou renvoi un message d'erreur dans le cas d'échec d'inscription.

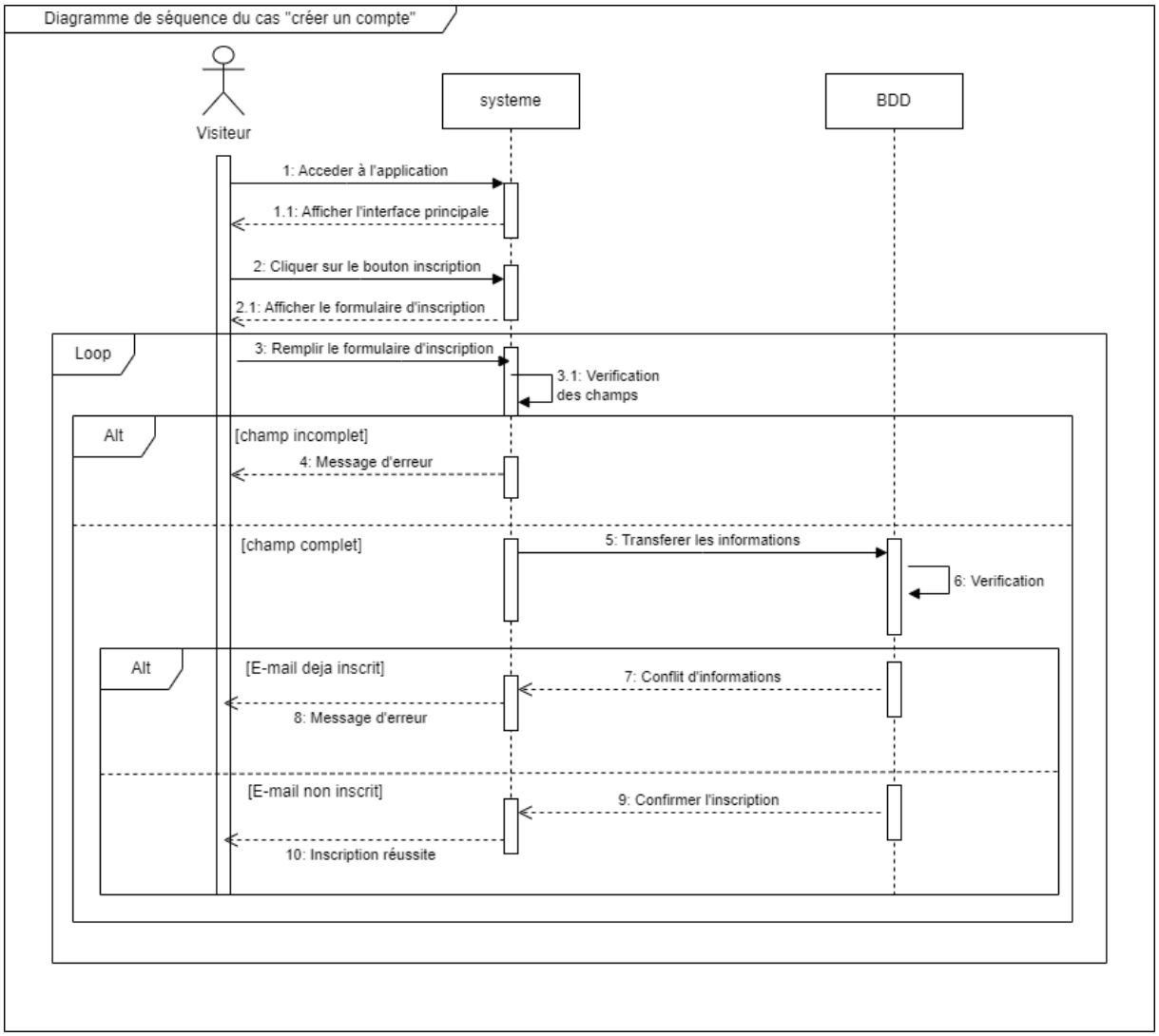

Figure 3. Diagramme de séquence du cas créer un compte

3.2.4- Diagramme de séquence du cas d'utilisation «voir la liste des abonnements »

Un client peut voir la liste des chaines dont il est abonné, le système récupère cette liste de la base de données et le lui l'affiche.

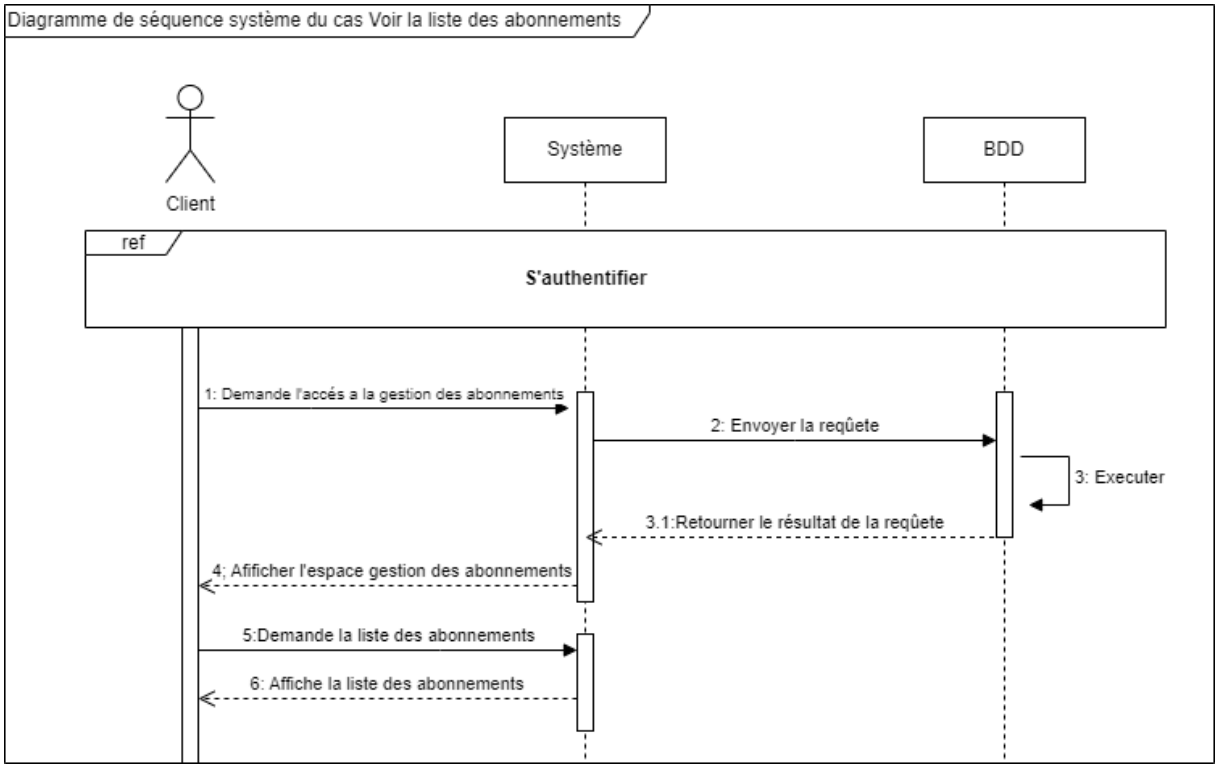

Figure 4.Diagramme de séquence du cas voir la liste des abonnements

3.2.5- Diagramme de séquence du cas « diffuser du contenu »

Le Diffuseur après authentification demande l'accès a son espace de gestion de chaines, le système récupère les chaines gérées par ce diffuseur et le lui les affiche. Le diffuseur sélectionne alors l'une de ses chaines, remplis les champs et clique sur diffuser le contenu, le système récupère alors le message et l'envoie directement a RabbitMQ. Si le message est stocké avec succès RabbitMQ renvoi un accusé de réception et le système informe que son contenu a bien était diffusé. Dans le cas contraire RabbitMQ renvoi une erreur et le système informe le diffuseur que son message n'a pas été diffusé

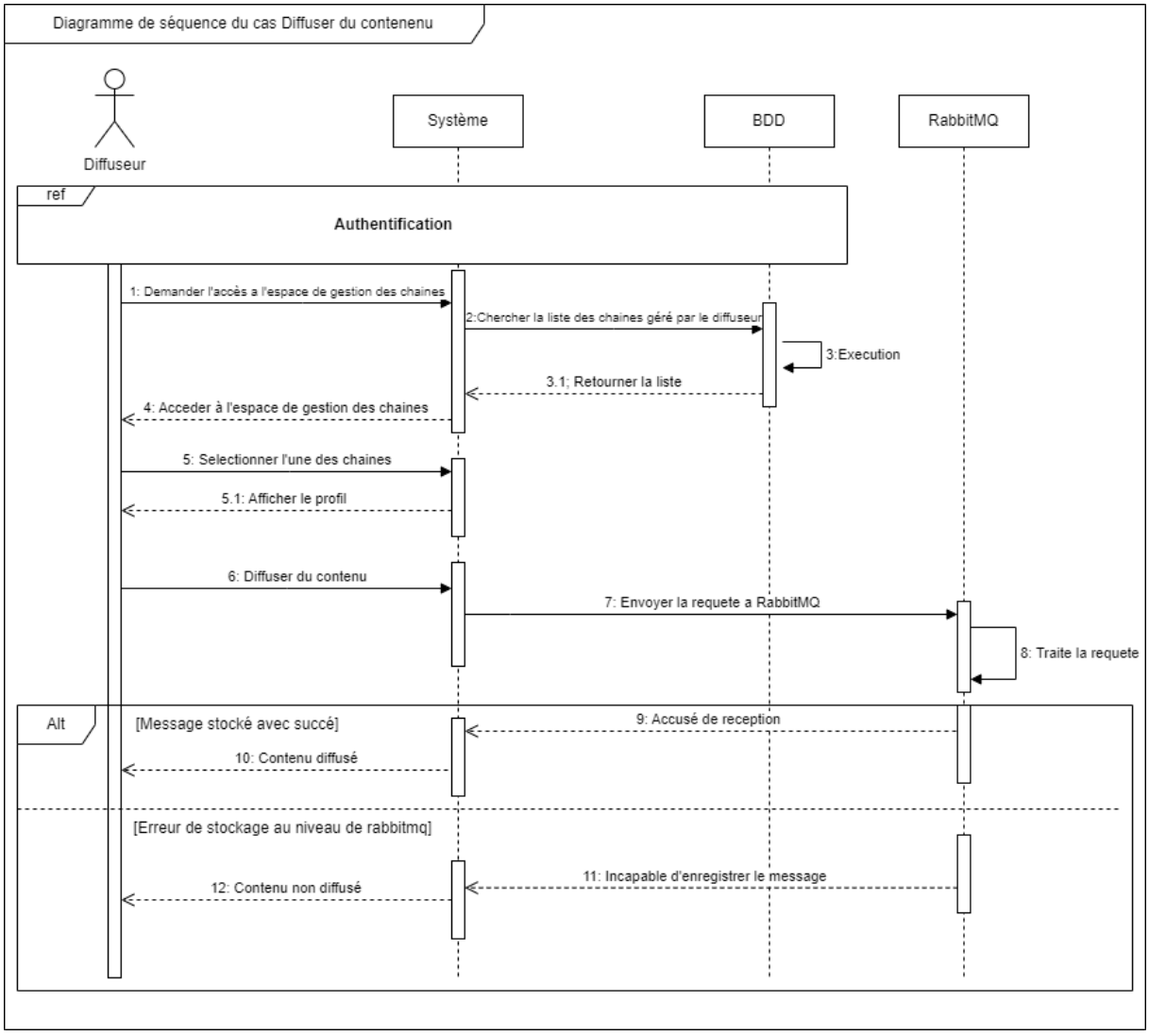

Figure 5. Diagramme de séquence du cas diffuser du contenu

3.3- Modèle de domaine

Le modèle de domaine est une représentation conceptuelle des éléments clés de la plateforme et la relation entre eux.

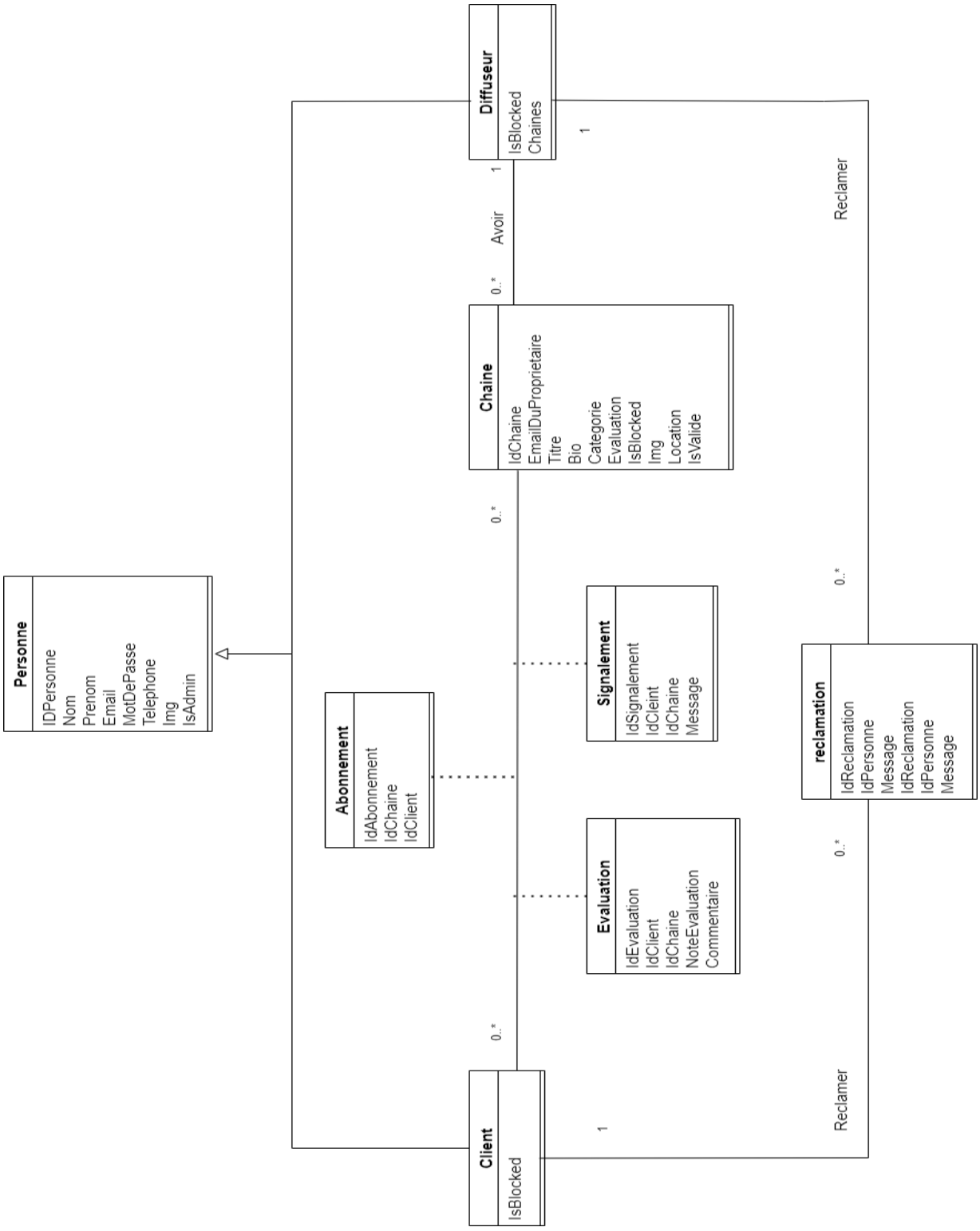

Figure 6. Modèle du domaine.

3.4- Diagramme de classe

Le diagramme de classe décrit clairement la structure de la plateforme en modélisant ses classes, ses attributs, ses opérations et les relations entre ses objets [11]. Il constitue l'un des pivots essentiels de la modélisation orientée objet et il est le seul diagramme considéré comme obligatoire lors de l'étape de conception.

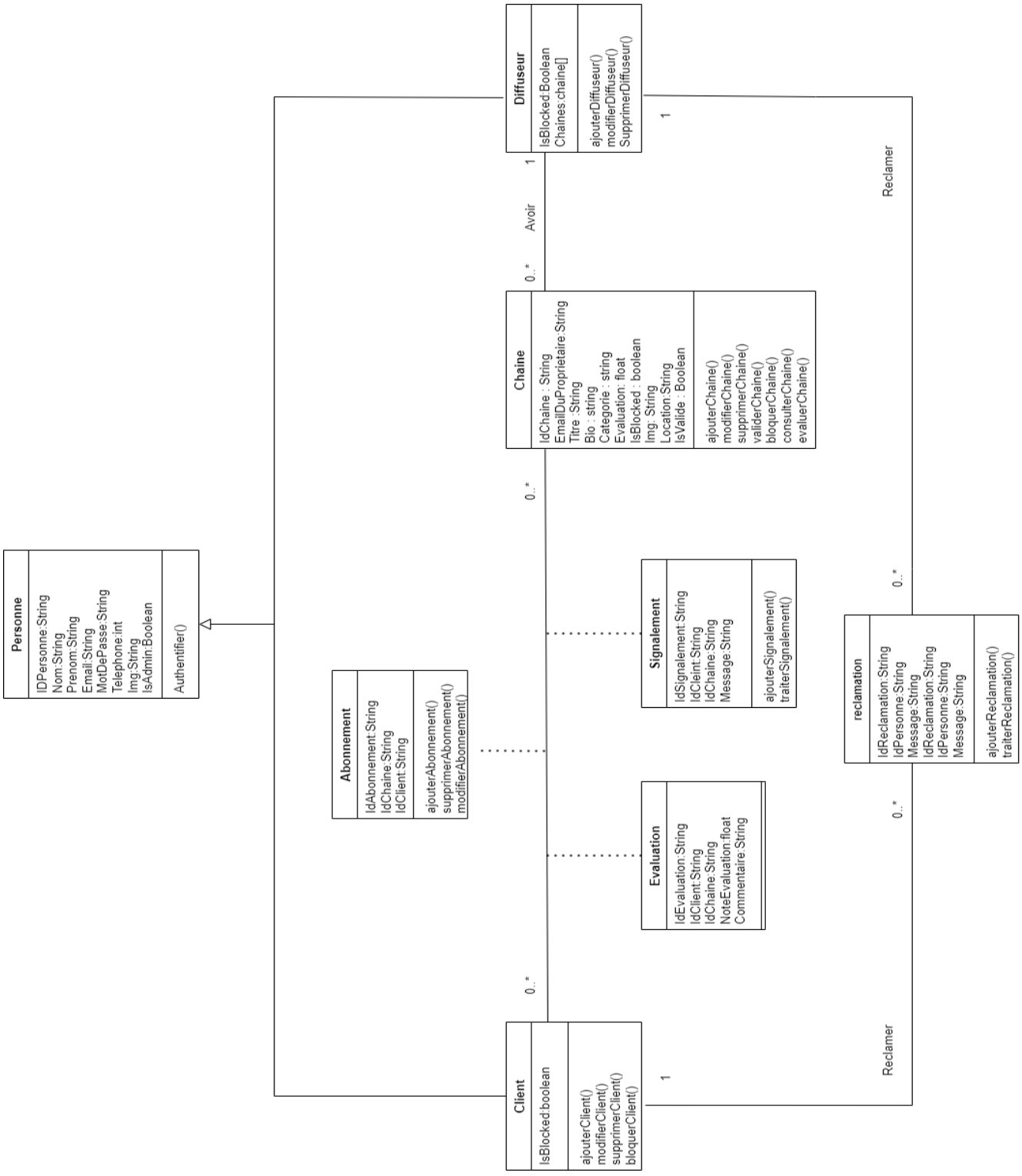

Figure 7. Diagramme de classe.

3.5- Conclusion

Dans ce chapitre, nous somme passer de simples modèles statiques qui décrivent les grandes lignes de notre plateforme à une modélisation dynamique et détaillé qui englobe la quasi-totalité de notre plateforme. Dans le prochain chapitre nous mettrons en pratique ce que nous avons vu précédemment.

QUATRIEME CHAPITRE

Chapitre 4 : Réalisation

4.1- Introduction

Apres avoir mis l'accent sur la conception de la base de données et des différentes fonctionnalités dans les chapitres précédents, on va dans ce chapitre décrire le processus de réalisation de notre plateforme et ce, en présentant l'architecture MVC, les différentes technologies, Framework et outils que nous avons utilisés et un guide d'installation de RabbitMQ. Nous présenterons également quelques interfaces de notre application.

4.2- Single page application (SPA)

4.2.1- Définition d'une SPA

Une Single Page Application (SPA) est un type d'application web qui fonctionne entièrement au sein d'une seule page web, sans nécessiter de rechargement complet de la page lors des interactions de l'utilisateur. Contrairement aux sites web traditionnels, où chaque action entraîne le chargement d'une nouvelle page, une SPA charge initialement toutes les ressources nécessaires et utilise JavaScript pour dynamiquement mettre à jour le contenu de la page en fonction des interactions de l'utilisateur.

4.2.2- Fonctionnement d'une SPA

- Tout d'abord, le client se connecte au serveur et obtient le contenu de la page, qui comprend principalement le code HTML, le CSS et un ensemble de fichiers JavaScript regroupés. Ces fichiers JavaScript regroupés contiennent tous les scripts nécessaires pour exécuter la logique de l'application.
- Lorsqu'un utilisateur effectue une action, cela déclenche l'exécution des scripts JavaScript correspondants, qui envoient des requêtes au serveur. Les données sont généralement retournées au format JSON, et la mise à jour des données ne nécessite pas de rechargement complet de la page web.
- Dans une application multi-tiers (MPA) impliquant des serveurs back-end, le serveur web peut se contenter de télécharger le paquet initial de HTML/CSS/JavaScript. Tous les appels ultérieurs se font directement entre le navigateur et les services back-end.[8]

4.2.3- Avantages d'une SPA

- Expérience utilisateur fluide : Les SPAs offrent une expérience utilisateur fluide et réactive, car elles chargent le contenu initial une seule fois et mettent à jour dynamiquement les sections de la page en utilisant JavaScript. Cela permet aux utilisateurs de naviguer rapidement entre les différentes parties de l'application sans avoir à subir les temps de chargement associés aux rechargements de page complets.
- Réduction des charges serveur : Les SPAs minimisent les échanges de données avec le serveur, car seules les mises à jour nécessaires sont demandées via des appels Ajax. Cela permet de réduire la charge sur le serveur et d'améliorer les performances globales de l'application, en particulier lorsqu'il y a un grand nombre d'utilisateurs ou une utilisation intensive de l'application.
- Réutilisabilité : Les SPAs utilisent des frameworks JavaScript comme React pour créer des composants modulaires réutilisables, simplifiant ainsi le développement et la maintenance des applications web.

4.3- MERN stack

La MERN stack est une combinaison de technologies employées pour créer des applications web dynamiques et modernes. L'acronyme MERN désigne MongoDB, Express.js, React et Node.js, qui sont les éléments centraux de cette pile technologique.

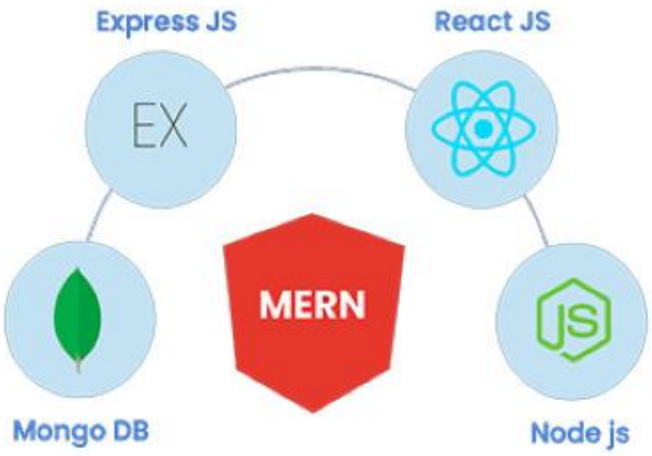

Figure 1. MERN Stack

4.4- Le modèle MVC

MVC a été introduit par Trygve Reenskaug en 1979 comme une solution aux problèmes des utilisateurs qui contrôlent de grandes quantités de données. MVC vise à séparer la logique d'affaire de la couche présentation. Il définit trois couches principales : Le modèle, la vue et le contrôleur [9]. Le modèle représente les données et la logique métier de l'application et est indépendants de la vue et des contrôleurs. La vue est la représentation visuelle des données, elle récupère les données du modèle et les présente de manière claire et compréhensible. Le contrôleur est un coordinateur entre la vue et le modèle. Il reçoit les interactions des utilisateurs depuis la vue, met à jour le modèle en fonction des entrées de l'utilisateur qui envoie une notification à la vue pour qu'elle se mette à jour.

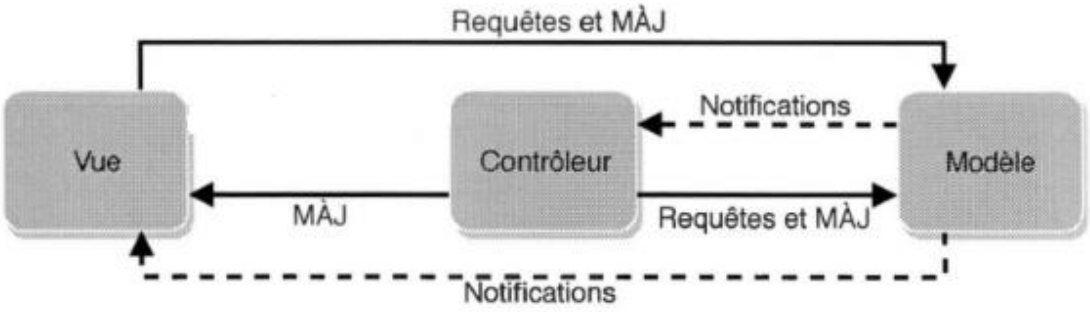

Figure 2. L'architecture MVC

4.5- Les modèles

4.5.1- Mongo DB

MongoDB est une base de données NoSQL populaire et open-source qui stocke les données sous forme de documents JSON[5]. Elle peut gérer de grandes quantités d'informations et s'adapter à différentes applications. MongoDB est largement utilisée dans les sites web, les applications mobiles et d'autres systèmes pour stocker et organiser les données de manière efficace. La documentation officielle se trouve sur le lien : https://docs.mongodb.com/

Figure 3. LOGO MongoDB

[5] JavaScript Object Notation est un format léger d'échange de données, facile à lire et à écrire pour les développeurs, et facile à analyser et à générer pour les machines.

4.5.2- Mongo atlas cloud

MongoDB Atlas Cloud est une plateforme de base de données hébergée dans le cloud qui simplifie la gestion et le déploiement de bases de données MongoDB. Elle offre une infrastructure sécurisée, évolutive et prête à l'emploi, permettant aux développeurs de se concentrer sur la création d'applications plutôt que sur la gestion des serveurs. Avec MongoDB Atlas Cloud, nous pouvons facilement héberger et gérer nos bases de données MongoDB sans nous soucier des détails techniques. La documentation officielle se trouve sur le lien : https://docs.atlas.mongodb.com/

4.5.3- Mongoose

Mongoose est une bibliothèque JavaScript qui facilite l'interaction avec la base de données MongoDB dans le cadre du développement d'applications Node.js. Elle fournit une couche d'abstraction au-dessus de MongoDB, permettant aux développeurs de définir des schémas pour les documents et de bénéficier de fonctionnalités avancées telles que la validation des données, les requêtes complexes et les opérations de mise à jour. Mongoose simplifie également la gestion des connexions à la base de données, la création de modèles de données et l'exécution de requêtes. Grâce à son approche orientée objet, Mongoose permet de structurer les données de manière cohérente et offre une interface fluide pour interagir avec MongoDB. Avec Mongoose, les développeurs peuvent accélérer le processus de développement d'applications Node.js en simplifiant l'accès et la manipulation des données dans MongoDB. La documentation officielle se trouve sur le lien : https://mongoosejs.com/docs/

Figure 4. LOGO Mongoose

4.6- Les vues

4.6.1- HTML5

HyperText Markup Language version 5 est la dernière itération majeure du langage de balisage utilisé pour créer des pages web. Il introduit de nouvelles fonctionnalités et améliorations significatives par rapport aux versions précédentes, offrant aux développeurs des outils plus puissants pour concevoir des sites web modernes et réactifs.

Figure 5. LOGO HTML5

4.6.2- CSS3

Le CSS (Cascading Style Sheets, ou feuilles de styles en cascade) est utilisé pour personnaliser la présentation des éléments HTML tels que la couleur, la taille, la police de caractères, la position, la largeur, la hauteur, et d'autres aspects liés à la mise en page d'un document HTML.

Figure 6. LOGO CSS3

4.6.3- Bootstrap

Bootstrap est un Framework de développement web populaire et open-source. Il offre une collection d'outils et de composants préconçus, tels que des grilles de mise en page, des formulaires, des boutons, des carrousels, et bien d'autres encore, permettant aux développeurs de créer rapidement des sites web réactifs et esthétiquement attrayants. Grâce à son approche de développement basée sur la grille, Bootstrap facilite la création d'interfaces responsives qui s'adaptent de manière fluide à différents appareils et tailles d'écran. De plus, il offre une vaste documentation, une communauté active et de nombreux thèmes personnalisables, ce qui en fait un choix populaire pour les projets web. La documentation officielle se trouve sur le lien : https://getbootstrap.com/docs/5.3/getting-started/introduction/

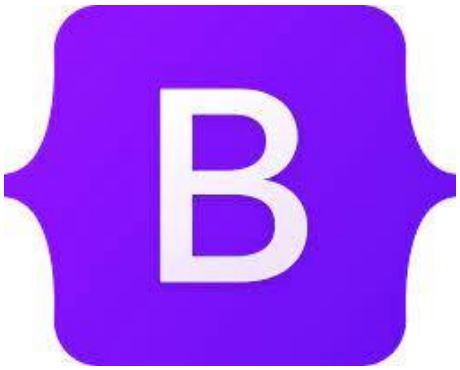

Figure 7. LOGO Bootstrap

4.6.4- JavaScript

JavaScript est un langage de programmation polyvalent, principalement utilisé pour rendre les pages web interactives et dynamiques. Il peut être exécuté côté client, directement dans le navigateur, ce qui lui confère une grande portabilité. JavaScript permet d'ajouter des fonctionnalités telles que des animations et des formulaires interactifs. Il est également utilisé côté serveur avec Node.js pour développer des applications web complètes. JavaScript est un langage orienté objet et basé sur des événements, offrant une syntaxe simple et flexible. La documentation officielle se trouve sur le lien : https://developer.mozilla.org/fr/docs/Web/JavaScript/Reference

Figure 8. LOGO JavaScript

4.6.5- React JS

React JS est une bibliothèque JavaScript open-source largement utilisée pour construire des interfaces utilisateur dynamiques et réactives. Basée sur le concept de composants réutilisables, React simplifie le processus de développement en permettant de diviser l'interface utilisateur en éléments indépendants et modulaires. Grâce à son approche de rendu virtuel efficace, React offre des performances optimisées en mettant à jour uniquement les composants qui nécessitent des changements, ce qui améliore l'expérience utilisateur et l'efficacité du développement. De plus, React adopte un modèle de programmation déclaratif, ce qui permet aux développeurs de décrire facilement l'état souhaité de l'interface utilisateur et React se charge de mettre à jour automatiquement l'interface en fonction des modifications de l'état. Avec sa large communauté, sa richesse en outils et sa compatibilité avec d'autres bibliothèques et Framework, React JS est devenu un choix populaire pour la construction d'applications web modernes et évolutives. La documentation officielle se trouve sur le lien : https://react.dev/

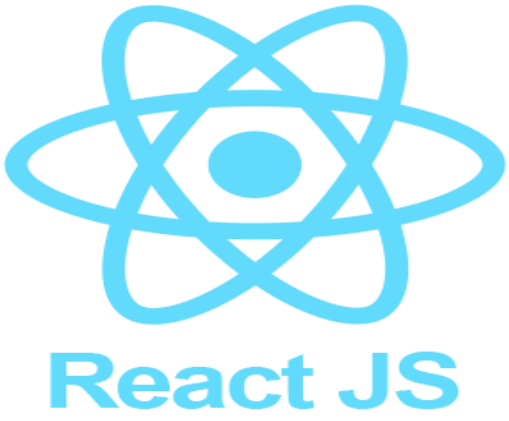

Figure 9. LOGO React JS

4.6.6- JSX

JSX est une extension de syntaxe utilisée avec React JS pour décrire la structure des composants d'interface utilisateur. Il permet d'écrire du code JavaScript et du code HTML de manière combinée, offrant ainsi une syntaxe claire et intuitive pour créer des éléments d'interface. Avec JSX, les développeurs peuvent incorporer du code JavaScript directement dans le balisage HTML, ce qui facilite la manipulation et la gestion dynamique des données. JSX offre également une meilleure lisibilité du code en permettant l'utilisation de balises et d'attributs familiers aux développeurs web. Au moment de la compilation, JSX est transformé en code JavaScript pur, permettant ainsi à React de créer et de mettre à jour efficacement les composants d'interface utilisateur. Grâce à JSX, la création d'interfaces réactives et modulaires avec React JS devient plus simple et plus intuitive. La documentation officielle se trouve sur le lien : https://fr.legacy.reactjs.org/docs/introducing-jsx.html

4.7- Les Contrôleurs

4.7.1- Node JS

Node.js est un environnement de développement utilisant JavaScript pour créer des applications web côté serveur. Il permet aux développeurs de construire des applications rapides et évolutives grâce à son modèle de programmation asynchrone. Node.js est largement utilisé pour les applications en temps réel, les API et les microservices. Il facilite également le partage de code entre le côté client et le côté serveur, offrant ainsi une expérience de développement plus fluide. La documentation officielle se trouve sur le lien : https://nodejs.org/fr

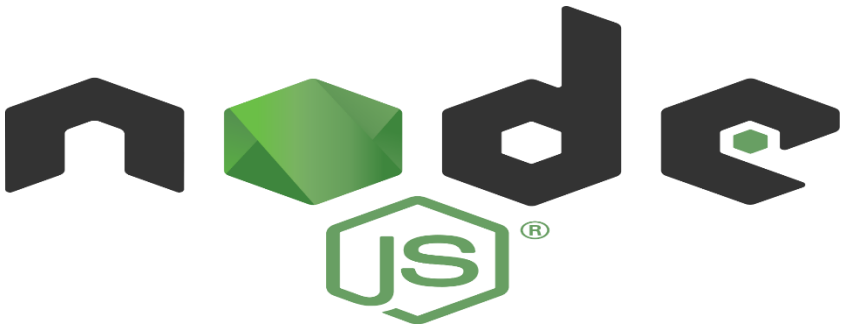

Figure 10. LOGO NodeJS

4.7.2- Express JS

Express.js est un Framework léger et flexible basé sur Node.js utilisé pour construire des applications web. Il simplifie la création de serveurs en offrant des fonctionnalités de base et une gestion facile des routes et des requêtes. Express.js permet aux développeurs de construire rapidement des applications web robustes et réactives en utilisant JavaScript. La documentation officielle se trouve sur le lien : https://expressjs.com/fr/

4.7.3- RESTful API

Une API RESTful (Representational State Transfer) est une architecture logicielle qui permet la communication entre différents systèmes via des protocoles HTTP. Elle repose sur le principe fondamental de l'utilisation des verbes HTTP (GET, POST, PUT, DELETE) pour effectuer des opérations sur des ressources. Une API RESTful est conçue de manière à être légère, scalable et facilement intégrable par d'autres applications. Elle permet aux développeurs d'accéder, de créer, de modifier et de supprimer des données à travers des endpoints bien définis. En utilisant des formats de données courants tels que JSON ou XML, les API RESTful fournissent une interface claire et universelle pour échanger des informations entre les systèmes.

4.8- Outils utilisés et autres

4.8.1- Visual studio code

Visual Studio Code est un éditeur de code source gratuit et open-source, développé par Microsoft. Il offre une expérience de développement moderne et polyvalente, avec une interface utilisateur conviviale et une large gamme de fonctionnalités. Visual Studio Code prend en charge de nombreux langages de programmation et offre des fonctionnalités telles que la coloration syntaxique, l'autocomplétions intelligente, le débogage intégré, la gestion des versions avec Git, les extensions personnalisées et bien plus encore. Il dispose également d'une intégration étroite avec des outils et services couramment utilisés, tels que les outils de développement web, les frameworks populaires et les services cloud. Avec sa communauté active et sa disponibilité sur différentes plateformes, Visual Studio Code est devenu un choix populaire parmi les développeurs pour leur productivité et leur confort lors de la création

d'applications. La documentation officielle se trouve sur le lien :
https://code.visualstudio.com/docs

Figure 11. LOGO VS Code

4.8.2- Postman :

Postman est un outil simple et pratique utilisé par les développeurs pour tester et travailler avec des API. Il permet d'envoyer des requêtes HTTP à des endpoints spécifiques, de visualiser les réponses et de vérifier le bon fonctionnement des API. Avec son interface conviviale, Postman facilite le processus de test et de débogage des API, ce qui en fait un outil incontournable pour les développeurs. La documentation officielle se trouve sur le lien :
https://learning.postman.com/docs/

Figure 12. LOGO Postman

4.8.3- Draw.io

Draw.io est une application de dessin et de création de diagrammes en ligne. C'est un outil polyvalent qui permet de créer facilement des schémas, des organigrammes, des diagrammes de flux, des diagrammes UML et bien d'autres types de diagrammes. Avec son interface intuitive et conviviale, Draw.io offre une large gamme d'éléments graphiques préconçus, de formes et de connexions pour aider les utilisateurs à créer des diagrammes professionnels et visuellement attrayants. Il permet également de collaborer en temps réel avec d'autres utilisateurs, facilitant ainsi le travail d'équipe et le partage d'idées. La documentation officielle se trouve sur le lien : https://www.drawio.com/doc/

Figure 13. LOGO Draw.io

4.8.4- JSON Web token

La signature Web JSON (JWS) représente un contenu sécurisé par des signatures numériques ou des codes d'authentification des messages (MAC) à l'aide de structures de données basées sur JSON[10].

Le token contient les informations essentielles d'un utilisateur lorsqu'il se connecte tel que son ID, son rôle, etc. Le client enregistre le token dans son navigateur et l'envoie avec chaque requête qu'il envoie au système. Le système vérifie la validité du JWT et extrait les informations sur l'utilisateur pour authentifier et autoriser les requêtes. La documentation officielle se trouve sur le lien : https://jwt.io/introduction

4.8.5- Node Mailer :

NodeMailer est une bibliothèque Node.js populaire utilisée pour envoyer des e-mails à partir d'une application Node.js. Elle simplifie le processus d'envoi d'e-mails en fournissant une interface conviviale pour la création et l'envoi d'e-mails.

4.9- Installation, configuration et exploitation de RabbitMQ

Nous allons montrer comment installer et configurer RabbitMQ ainsi que comment il procède pour l'envoi des messages

4.9.1- Installation et configuration [12]

Pour installer RabbitMQ il faut visiter le site officiel accessible via ce lien :
https://www.rabbitmq.com/install-windows.html

Apres installation, Nous activons par la suite le plugin de gestion de RabbitMQ avec la commande : rabbitmq-plugins enable rabbitmq_management

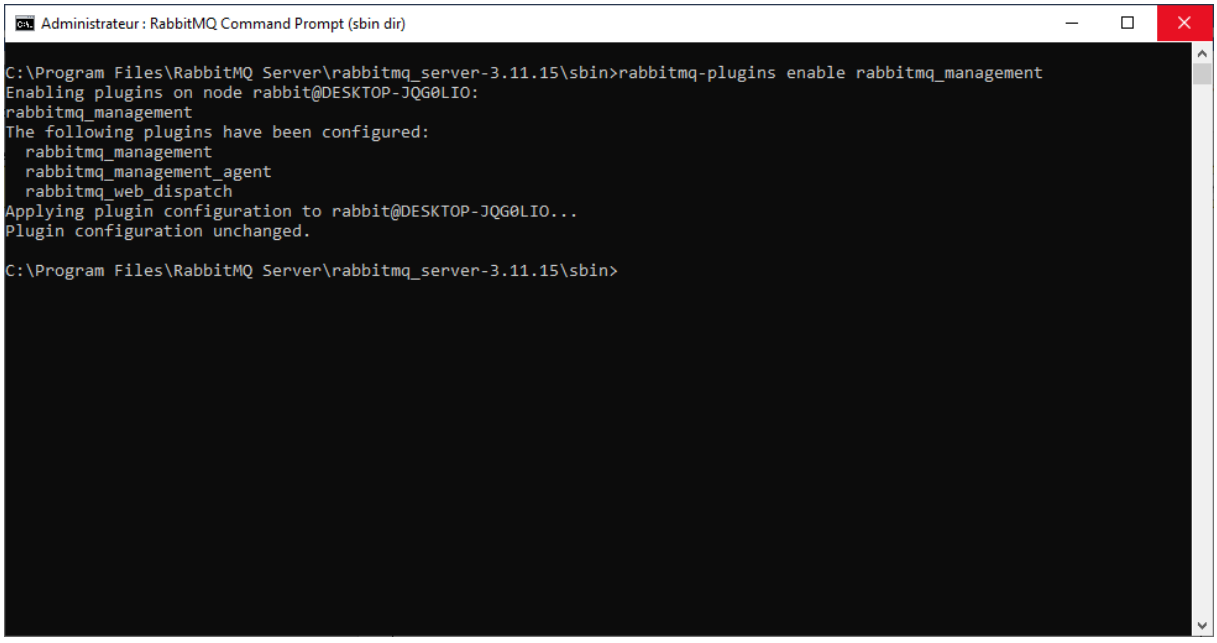

Figure 14. Activation du plugin de gestion de RabbitMQ

Le plugin de gestion RabbitMQ aide les utilisateurs à faire fonctionner RabbitMQ à l'aide d'une interface utilisateur graphique.

L'étape suivante est d'ouvrir les services dans la barre de recherche de l'ordinateur et de redémarrer RabbitMQ

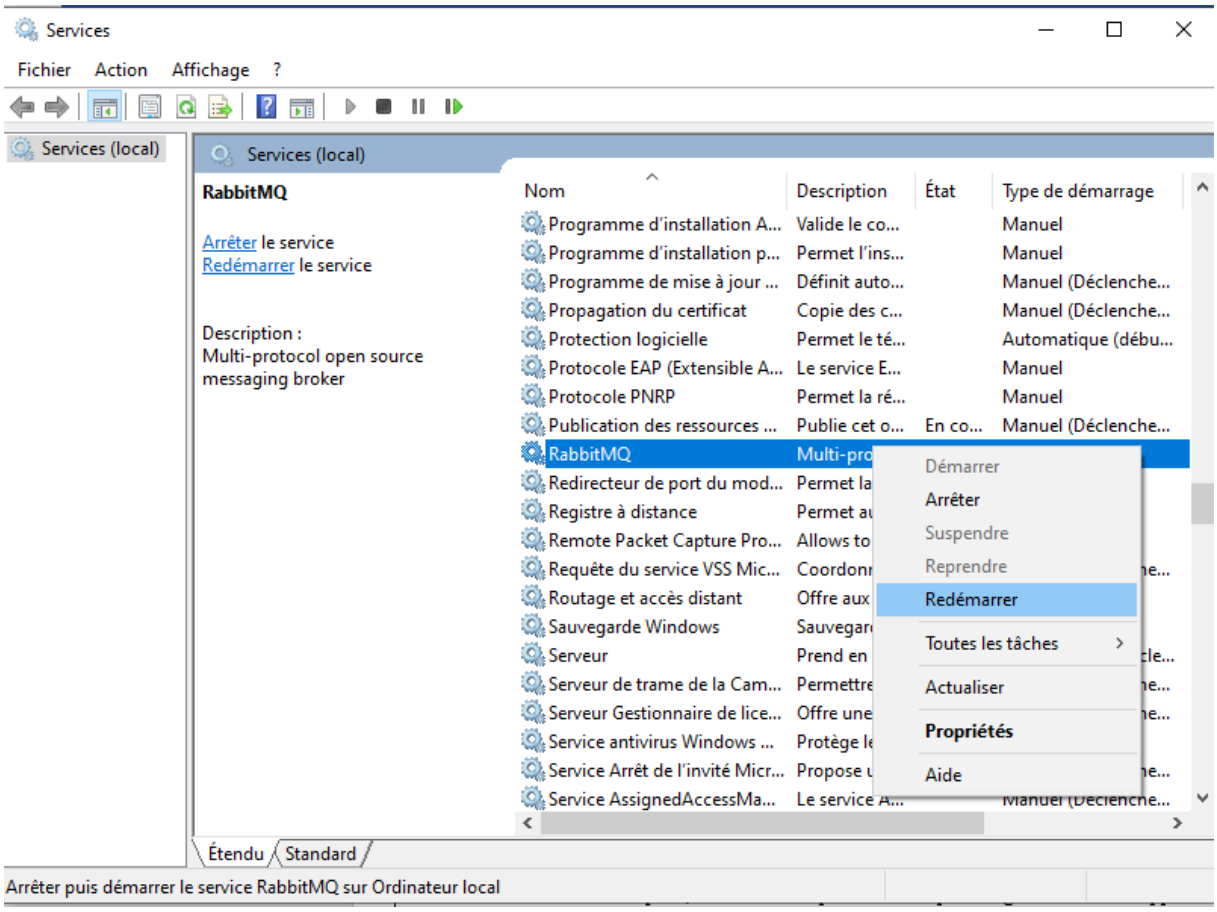

Figure 15. Redémarrage des serveurs de RabbitMQ

Nous ouvrons maintenant notre invite de commande dans notre projet et lancent la commande suivante : npm install amqplib

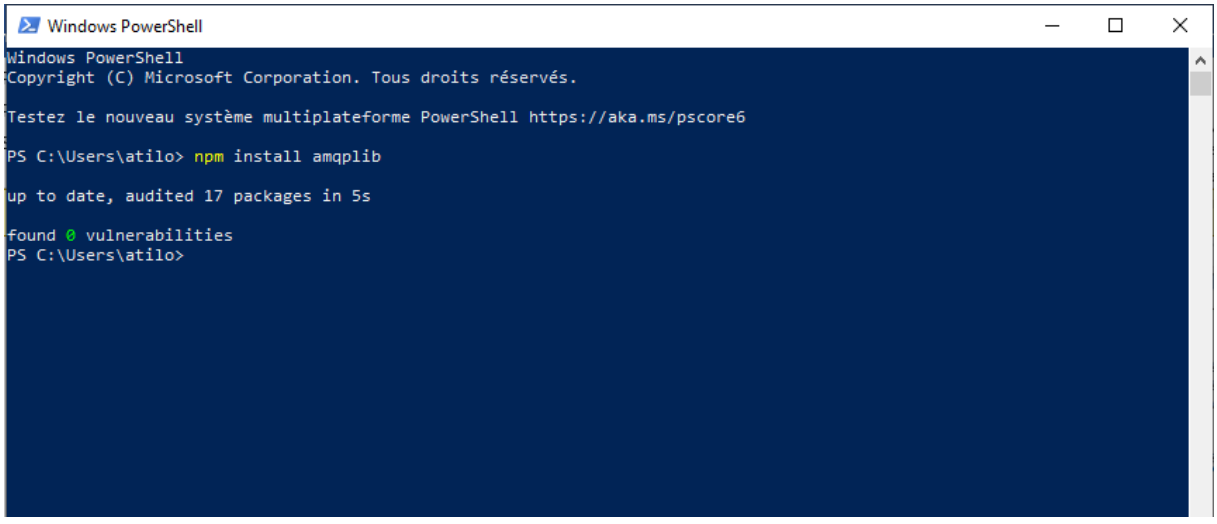

Figure 16. Installation de la bibliothèque AMQPLIB

Amqplib est une bibliotheque de nodeJS qui permet d'utiliser les services qu'offre RabbitMQ

4.9.2- Exemple d'utilisation de RabbitMQ

Nous allons à présent voir comment utiliser amqplib pour qu'un diffuseur publie un message

```
const send = async (req,res) => {
    const channel_id = req.params.id
    const {title,subject,content} = req.body
    if(subject===''){return res.status(400).json('you must add a subject')}
    if(content===''){return res.status(400).json('there is no content!')}
    const message = title.concat('µ*µ',subject,'µ*µ',content)
    const connection = await amqplib.connect('amqp://localhost')//open connection with our rabbitmq server
    const channel = await connection.createChannel()//open channel with the previous connection
    await channel.assertExchange(exchangeName , 'direct' , {durable : true})//check if this echange exists if not create one!
    try{
        //send this message to this exchange to ditrubue it using this binding key
       channel.publish(exchangeName , channel_id , Buffer.from(message),{persistent : true})//channel-id is the binding key

        setTimeout(()=>{
            connection.close();
        },500)
        res.status(200).json('ok')


    }catch(err){
        res.status(400).json({ err: err.message })
    }
    }
```

Figure 17. Code source pour diffuser un message

Comme nous le voyons sur la capture, nous récupérons le binding key (channel-id) et le message aux lignes 2 et 3. Nous ouvrons une connexion avec le serveur rabbitMQ avec la méthode connect de amqplib pour ouvrir une chaine et interagir avec lui. A la ligne 13, nous envoyant le message à l'exchange en utilisant le binding key.

4.10. Présentation des interfaces

4.10.1- Interface page d'accueil Get-Notified

La page d'accueil sert à présenter le site et offre la possibilité de s'inscrire ou bien de se connecter à son compte.

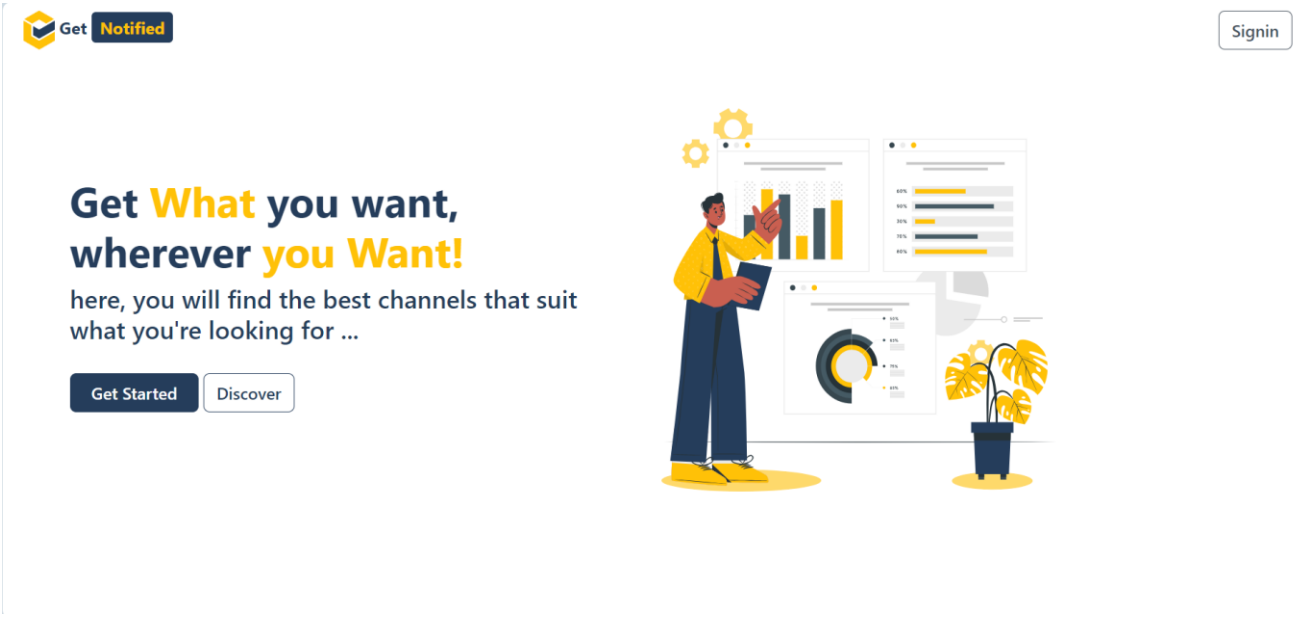

Figure 18. Page d'accueil

4.10.2- Interface d'inscription

Cette interface permet à un diffuseur de créer une chaine et attendre qu'elle soit validée ou non par un admin.

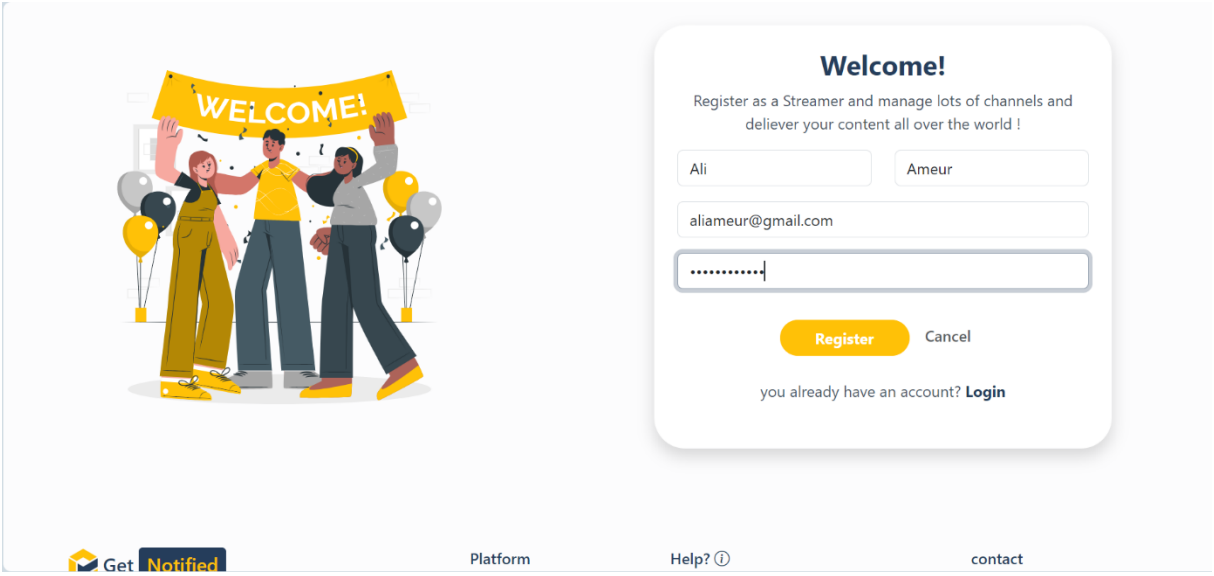

Figure 19. Interface d'inscription

4.10.3- Interface de connexion

L'interface de connexion permet à l'utilisateur de s'authentifier et d'accéder à son compte

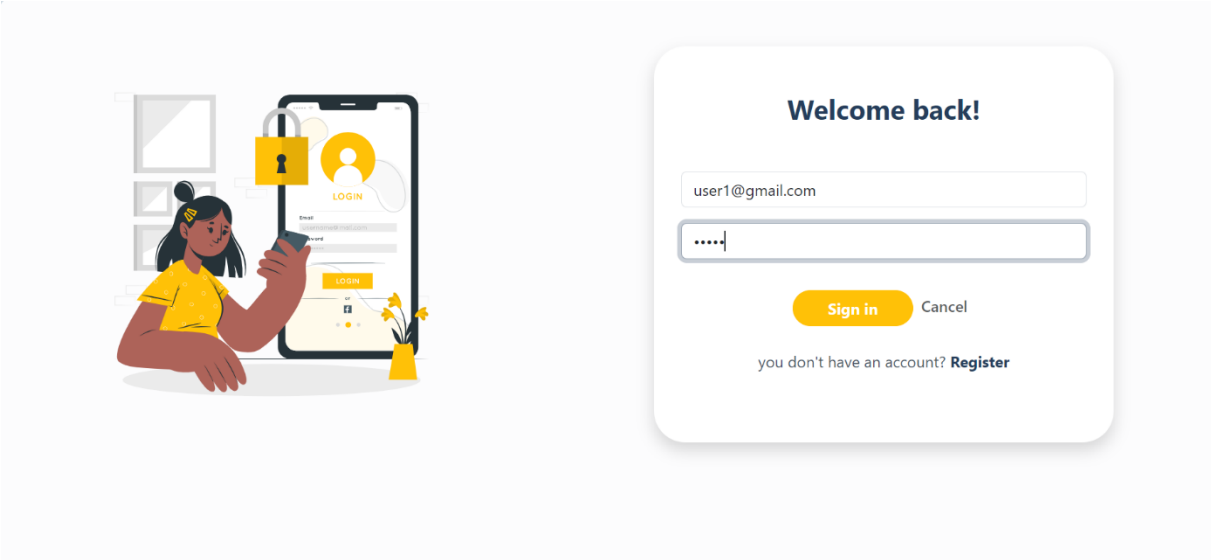

Figure 20. Interface de connexion

4.10.4- Interface de la liste des chaines

L'interface de la liste des chaines expose les chaines existantes sur la plateforme, l'utilisateur peut faire une recherche selon le nom de la chaine les trier selon leurs catégories ou leurs location (wilaya)

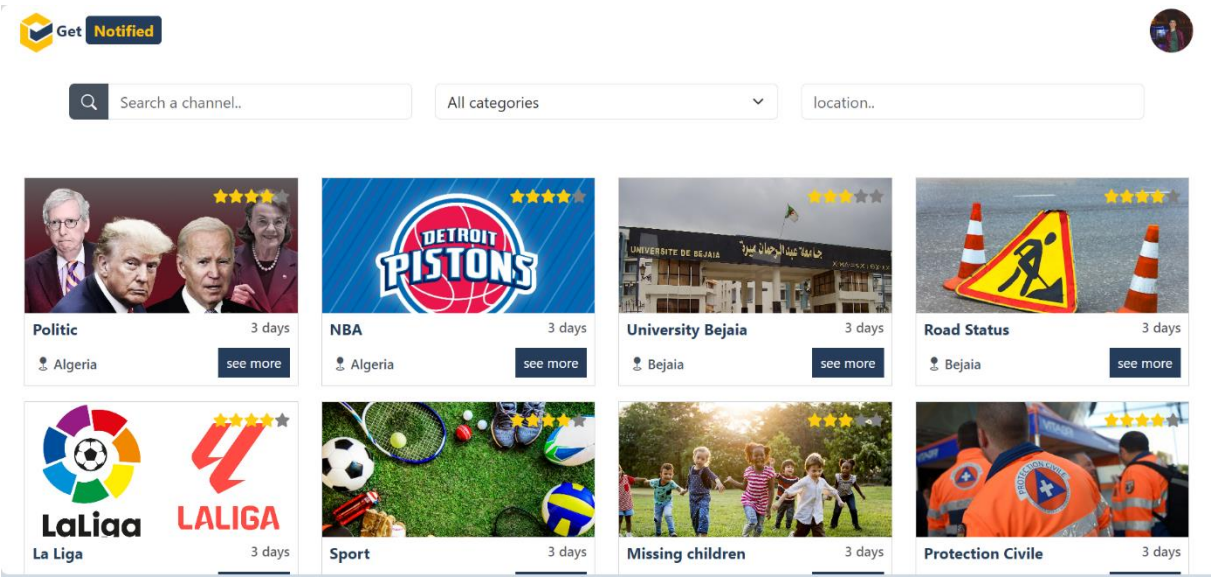

Figure 21. Interface de la liste des chaines

4.10.5- Interface créer une chaine

Cette interface permet à un diffuseur de créer une chaine et attendre qu'elle soit validée ou non par un admin.

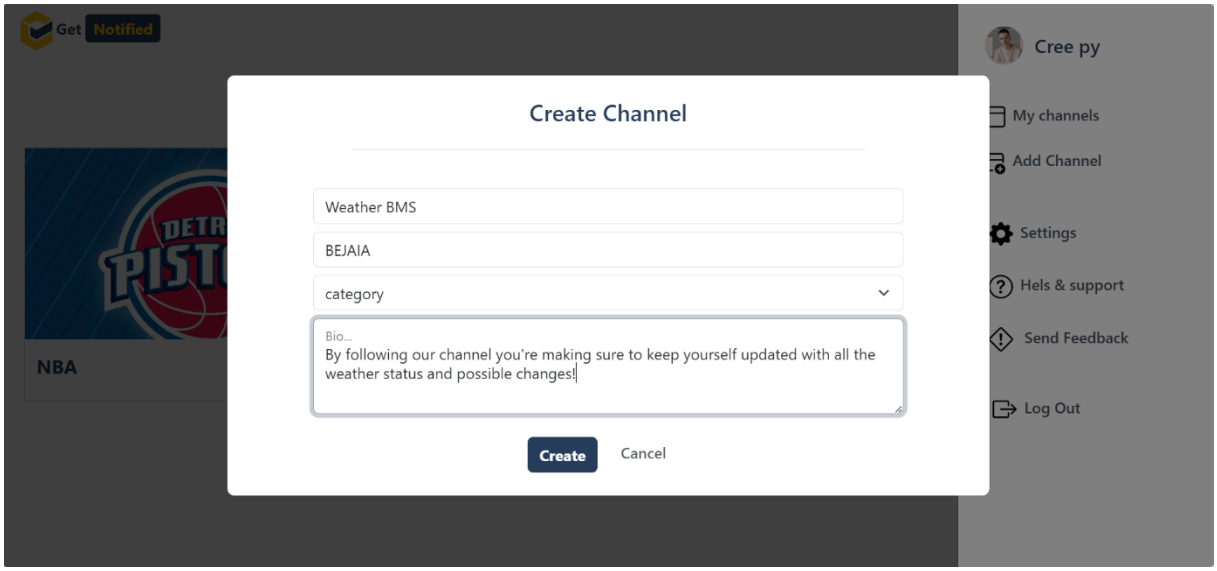

Figure 22. Interface créer une chaine

4.10.6- Interface d'une chaine

Cette interface affiche les informations d'une chaine et permet au client de s'y abonner, se désabonner, l'évaluer ou la signaler.

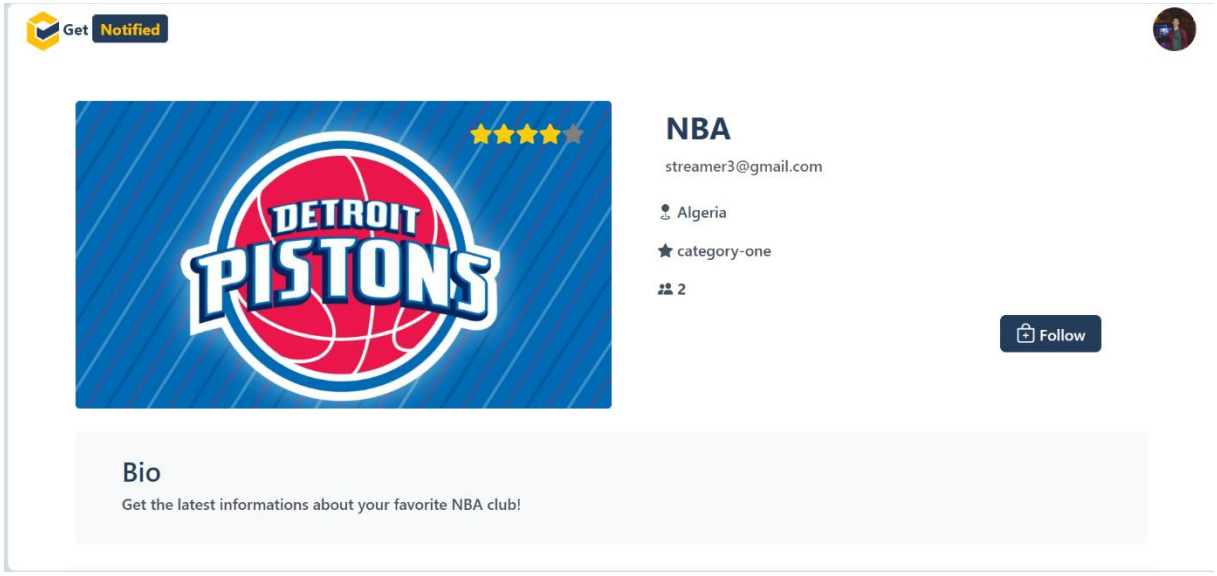

Figure 23. Interface d'une chaine

4.10.7- Interface profile utilisateur

Cette interface permet à l'utilisateur de voir ses informations personnelles et de les modifier et d'également voir la liste de ses abonnements.

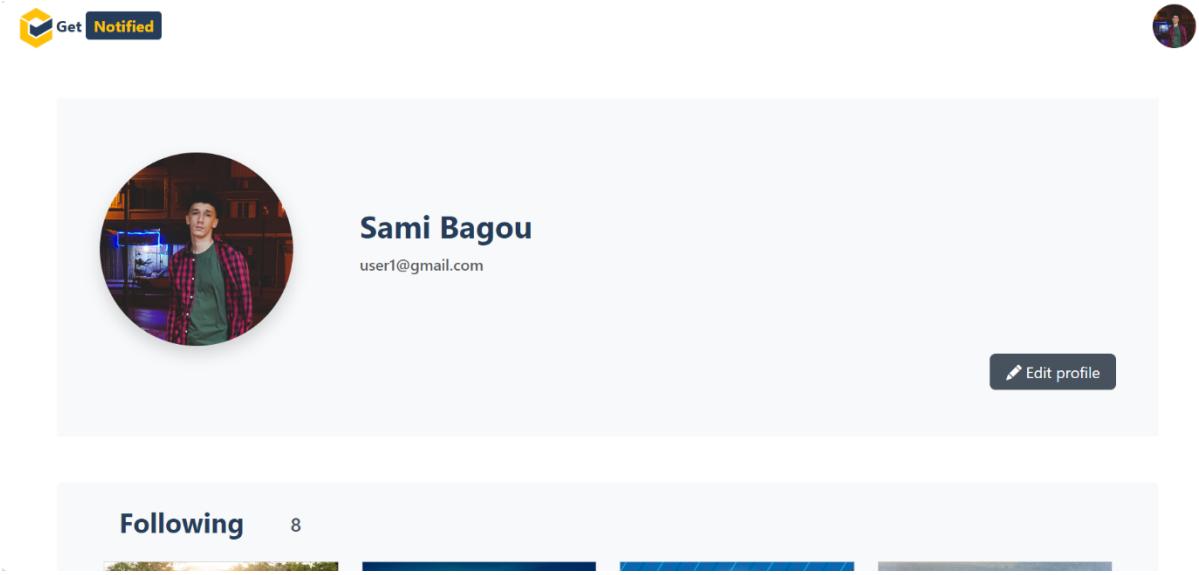

Figure 24. Interface profile utilisateur

4.11- Conclusion

Au cours de ce chapitre, nous sommes passés de simples diagrammes à une application fonctionnelle, passant de la théorie à la pratique. À cet effet, nous avons défini la combinaison des technologies et outils que nous avons utilisés pour atteindre les objectifs décrits dans les chapitres précédents.

Conclusion générale et perspectives

Dans ce mémoire nous avons conçu et réalisé une plateforme multicritère de diffusion d'alertes multimédia. L'objectif de notre projet consiste à contacter des personnes et leurs envoyer des alertes qui concernent des sujets bien précis qui les intéressent. Nous nous sommes munis de plusieurs outils à cette fin.

Dans ce cadre nous avons tout d'abord commencé par examiner les plateformes de diffusion d'alertes existantes et d'analyser les besoins de notre plateforme, nous avons par la suite étudié le fonctionnement des middleware orientés message et plus particulièrement RabbitMQ, l'étape d'installation et de configuration fut rapide et facile. Son intégration dans notre plateforme a posé un défi supplémentaire, car nous avons rapidement rencontré des erreurs. Malgré ces difficultés initiales, nous avons réussi à surmonter la plupart des obstacles. Notre plateforme permet à un diffuseur d'avoir plusieurs chaines et à un client de s'abonner à plusieurs chaines, de se désabonner, de les évaluer et de les signaler à un administrateur en cas de besoin. Chaque client à sa propre file où sont stocker les alertes qui lui sont destinée. Dès qu'il se connecte il les reçoit par E-mail et sa file d'attente sur RabbitMQ se vide. Il nous était difficile d'intégrer l'envoie des alertes par SMS et message vocaux car les API de ces derniers sont payante et nos moyens restreints.

Ce projet nous a permis d'avoir une approche complète du développement logiciel, d'appliquer nos acquis et d'obtenir de nouvelles connaissances en génie logiciel.

D'une manière globale, nous reconnaissant en toute sincérité que notre projet n'a pas atteint la perfection et que nous pouvons encore l'améliorer, nous avons l'ambition d'enrichir la plateforme Get-Notified par :

- L'utilisation des API d'envoi automatique d'SMS et d'enregistrement vocaux
- L'ajout d'avantages de fonctionnalités comme l'envoi de notifications et messages au tableau de bord du client.
- La gestion des degrés des alertes
- Ajouter d'autres langue au site comme le français et l'arabe.

Bibliographie

[1] Everbridge, https://www.everbridge.com/ Consulté le 10 Maï 2023

[2] Childfocus, https://childfocus.be/fr-be/ Consulté le 10 Maï 2023

[3] Nixle https://www.nixle.com/ Consulté le 11 Maï 2023

[4] RabbitMQ https://www.rabbitmq.com/ Consulté le 19 Maï 2023

[5] Apache Kafka https://kafka.apache.org/ Consulté le 23 Maï 2023

[6] IBM MQ https://www.ibm.com/docs/en/ibm-mq/8.0?topic=overview-introduction-mq Consulté le 28 Maï 2023

[7] AMQP part 1. Novembre 2012 https://www.arolla.fr/blog/2012/11/amqp-101-part-1/

[8] Qu'est-ce que c'est qu'une application single page. Septembre 2021 https://mobiskill.fr/blog/conseils-emploi-tech/quest-ce-quune-single-page-application/

[9] Aymen Daoudi, étude des patrons architecturaux de type MVC dans les applications Android. Décembre 2018

[10] Michael Jones, John Bradley, and Nat Sakimura. Json web token (jwt). Technical report, May 2015.

[11] Lucidchart, Qu'est-ce qu'est un diagramme de classe UML, https://www.lucidchart.com/pages/fr/diagramme-de-classes-uml , Consulté le 03 Juin 2023

[12] K. Jackson. OpenStack Cloud Computing Cookbook. September 2012.

Résumé :

Les plateformes de diffusion d'alertes jouent un rôle essentiel dans notre monde en constante évolution, permettant de diffuser rapidement des informations cruciales à un large public. Ce mémoire expose le processus de conception et de développement d'une plateforme multicritère de diffusion d'alertes, intégrant les technologies modernes telles que ReactJS et Node.Js tout en suivant la méthodologie MVC.

L'objectif central de ce projet était de créer une solution efficace pour la diffusion rapide et fiable des alertes dans divers contextes en Algérie. L'intégration de RabbitMQ, un système de messagerie asynchrone, a été essentielle pour optimiser la diffusion des alertes. Cette technologie a permis de garantir la fiabilité de la transmission des alertes.

Les résultats de ce travail contribuent à l'évolution des plateformes de diffusion d'alertes en Algérie offrant une solution efficace. Cette plateforme offre une réponse aux besoins variées des personnes selon leurs besoin et centre d'intérêts.

Abstract :

Alert broadcasting platforms play an essential role in our ever-evolving world, enabling the rapid dissemination of crucial information to a wide audience. This thesis outlines the design and development process of a multi-criteria alert broadcasting platform, incorporating modern technologies such as ReactJS and Node.js while following the MVC methodology.

The central objective of this project was to create an efficient solution for the swift and reliable dissemination of alerts in various contexts in Algeria. The integration of RabbitMQ, an asynchronous messaging system, was vital in optimizing alert distribution. This technology ensured the reliability of alert transmission.

The results of this work contribute to the advancement of alert broadcasting platforms in Algeria, providing an effective solution. This platform caters to the diverse needs of individuals based on their requirements and interests.