

# Document Spécification et Conception

PROGRAMMATION RESEAU II

Mini Projet 1 : Application Chat

Développement d'une application réseau utilisant l'architecture des sockets avec les technologies Java.



Filière : GI-2

Année Universitaire :  
2022/2023



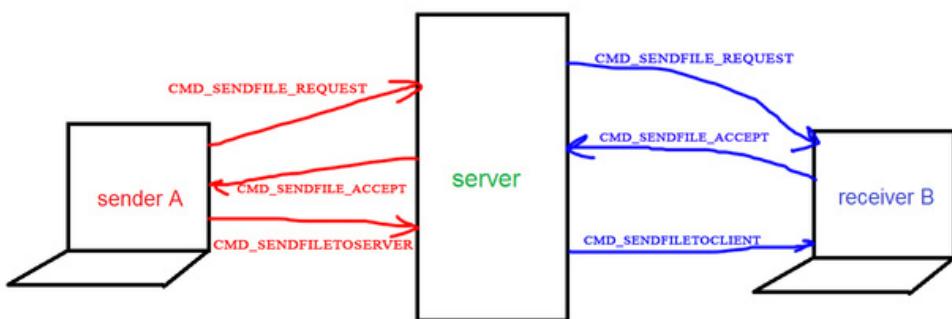
# Introduction

Dans le cadre de module programmation réseau nous sommes amenés à réaliser un mini projet qui consiste à développer une application desktop de chat multi-client, cette dernière doit permettre à un client de communiquer ou plutôt d'échanger des messages de types audio, image et fichier avec un autre client en se basant sur une interface graphique conçue à l'aide du java swing. Pour assurer la communication entre les deux clients nous allons utiliser les sockets, toujours sous le langage java.

Le présent de ce document décrit la réalisation de l'application desktop de chat client/serveur en utilisant les sockets. Ce dernier est le point de communication par lequel un processus peut émettre ou recevoir des données.

L'objectif est la communication entre plusieurs personnes connectées, soit de choisir de parler avec une personne précise ou de parler avec plusieurs en même temps,

L'environnement client/serveur désigne un modèle de communication à travers un réseau entre plusieurs programmes: l'un qualifié de client, envoie des requêtes, l'autre qualifié de serveur attendent les requêtes des clients et y répondent.



# Introduction

**Le protocole IP** est un protocole de niveau réseau qui permet d'échanger des paquets d'octets appelés datagrammes. Ce protocole ne garantit pas l'arrivée à bon port des messages. Cette fonctionnalité peut être implémentée par la couche supérieure, comme par exemple avec TCP. Un datagramme IP est l'unité de transfert à ce niveau. Cette série d'octets contient les informations du message, un en-tête (adresses source et de destination, ...) mais aussi des informations de contrôle. Ces informations permettent aux routeurs de faire transiter les paquets sur l'internet.

La couche de transport est implémentée dans les protocoles UDP ou TCP. Elle permet la communication entre des applications sur des machines distantes. La notion de service permet à une même machine d'assurer plusieurs communications simultanément.

Dans notre projet on a utiliser le protocole **TCP** qui permet une connexion de type point à point entre deux applications. C'est un protocole fiable qui garantit la réception dans l'ordre d'envoi des données. En contre-partie, ce protocole offre de moins bonnes performances mais c'est le prix à payer pour la fiabilité. TCP utilise la notion de port pour permettre à plusieurs applications d'exploiter ce même protocole.

Le système des sockets est le moyen de communication interprocessus développé pour l'Unix Berkeley (BSD). Il est actuellement implémenté sur tous les systèmes d'exploitation utilisant TCP/IP. **Une socket** est le point de communication par lequel un thread peut émettre ou recevoir des informations et ainsi elle permet la communication entre deux applications à travers le réseau.

La communication se fait sur un port particulier de la machine. Le port est une entité logique qui permet d'associer un service particulier à une connexion. Un port est identifié par un entier de 1 à 65535. Par convention les 1024 premiers sont réservés pour des services standard (80 : HTTP, 21 : FTP, 25: SMTP, ...)

Java prend en charge deux protocoles : TCP et UDP.

Les classes et interfaces utiles au développement réseau sont regroupées dans le package **java.net**.

# Description de l'application

L'application de chat que nous allons développer est une application de communication entre les utilisateurs inscrits dans le service à travers leurs emails. L'architecture de l'application est basée sur un modèle Client-Serveur utilisant l'architecture de socket IP TCP.

Les clients peuvent se connecter au service via leurs email et ont accès aux fonctionnalités suivantes :

1. Authentification : le client doit se connecter en utilisant ses informations d'identification, y compris son email et son mot de passe. Si l'authentification échoue, le client est averti par une notification, de plus il a le droit de s'enregistrer à travers un formulaire à remplir.
2. Contacts : une fois connecté, le client peut voir l'état de ses contacts, c'est-à-dire s'ils sont en ligne ou hors ligne.
3. Chat : le client peut communiquer avec ses contacts en envoyant et en recevant des messages texte. Les messages sont affichés dans une zone de chat dédiée, et les notifications sont envoyées lorsque de nouveaux messages sont reçus.
4. Importation : le client peut importer des fichiers envoyés par ses contacts, y compris des fichiers texte, des images et d'autres types de fichiers.
5. Déconnexion : le client peut se déconnecter de l'application à tout moment.
6. Gestion de compte : le client peut gérer son compte et ses contacts en ajoutant ou en supprimant des contacts, en modifiant son profil, en changeant son mot de passe, etc.

# Description de l'application

En termes de mise en œuvre, l'application peut être divisée en deux parties principales : le serveur et le client.

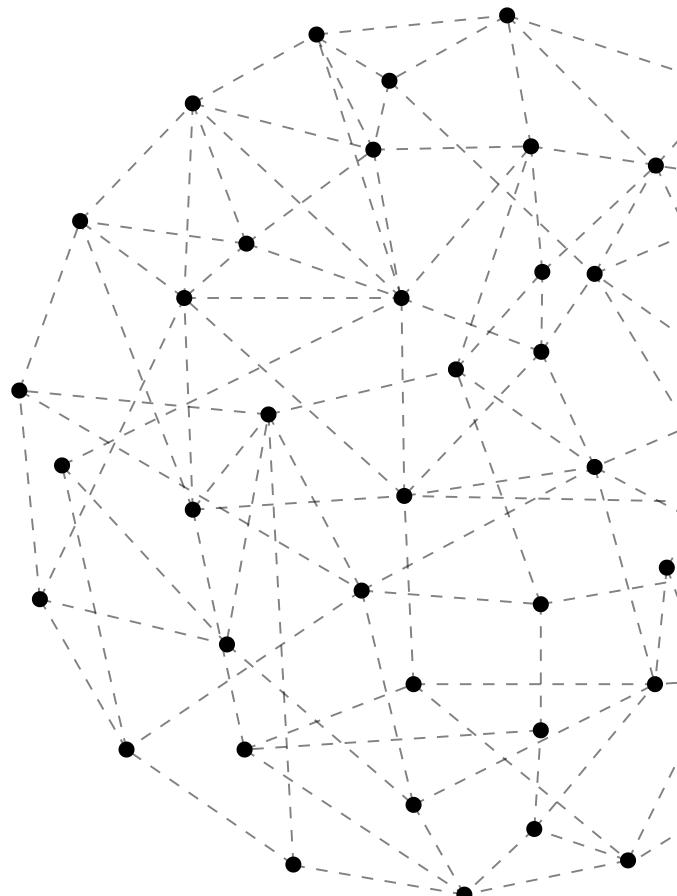
Le serveur est responsable de la gestion de la connexion avec les clients et de la transmission des messages entre eux. Le serveur utilise une structure de données pour stocker les informations sur les clients, y compris leurs emails, leurs noms d'utilisateur et leurs mots de passe. Lorsqu'un client se connecte, le serveur vérifie les informations d'identification du client et les stocke dans la structure de données. Le serveur envoie également des notifications aux autres clients lorsque des nouveaux clients se connectent ou se déconnectent.

Le client est responsable de l'affichage des informations utilisateur et de la gestion de l'interface utilisateur. Lorsqu'un client se connecte, il envoie ses informations d'identification au serveur pour vérification. Une fois authentifié, le client peut voir l'état de ses contacts et communiquer avec eux en envoyant des messages texte ou en important des fichiers. Le client peut également gérer son compte et ses contacts ajouter, supprimer ou modifier selon le choix.

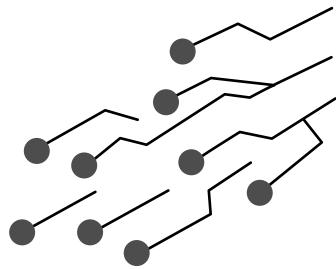
En conclusion, une application de chat en utilisant Java Swing et les sockets est un moyen efficace pour communiquer avec les utilisateurs en temps réel. En utilisant une architecture Client-Serveur basée sur les sockets IP TCP, l'application peut être étendue pour prendre en charge des fonctionnalités supplémentaires telles que la vidéo, l'audio et le partage d'écran.

# But de l'application

L'objectif initial de ce projet est de se familiariser avec les concepts de programmation réseau en utilisant l'interface des sockets, et de mettre en pratique les notions avancées de programmation orientée objet en créant une application de chat client/serveur. Le but final est de permettre aux utilisateurs de choisir un client et de communiquer avec plusieurs contacts simultanément ou d'envoyer des messages privés à un client en particulier, visibles uniquement pour lui. Pour y parvenir, il est nécessaire de créer un serveur de messagerie et un client qui répond aux protocoles du serveur. Le serveur doit être capable de gérer la connexion de plusieurs clients et de permettre des discussions de groupe entre eux.



# Operating Environment



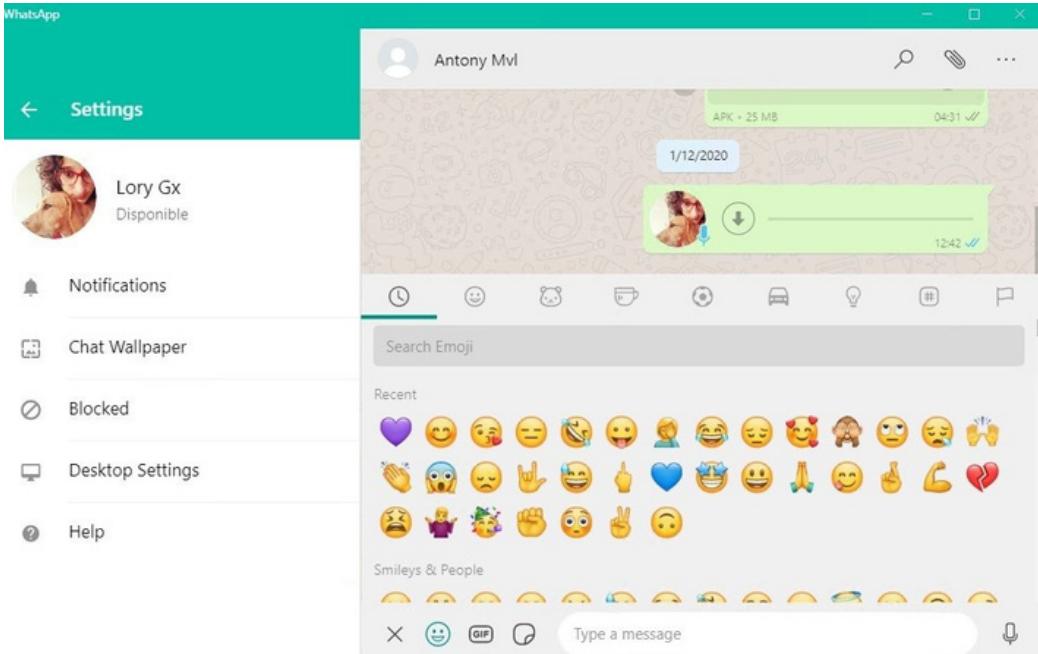
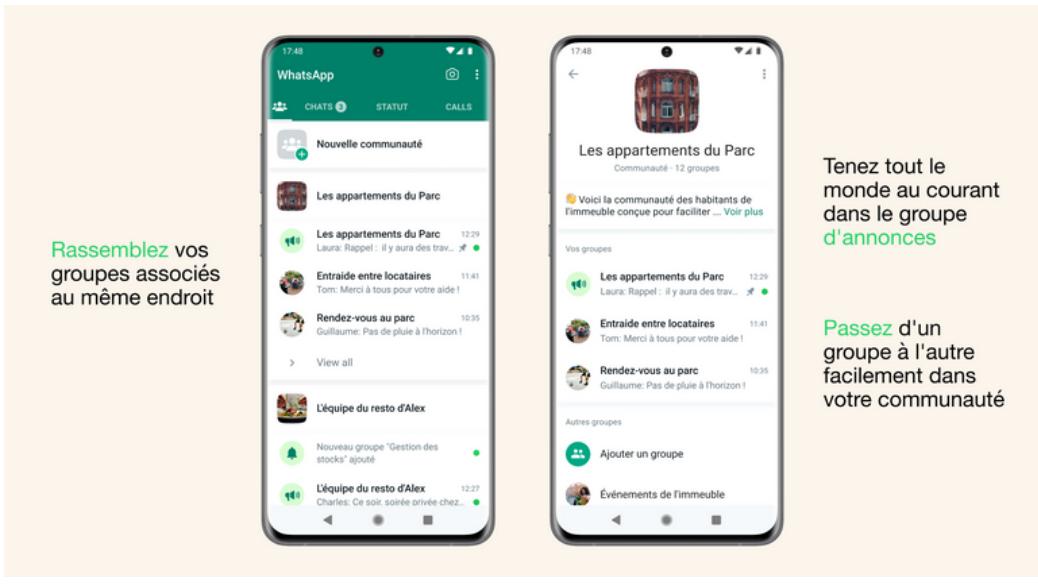
Java est une plateforme multi-plateforme, ce qui signifie qu'elle est compatible avec différents systèmes d'exploitation tels que Windows, Linux et macOS. Pour exécuter l'application de chat utilisant Java Swing et les sockets, il est nécessaire d'avoir sur le système d'exploitation :

- une version de Java compatible
- un environnement de développement Java (IDE) tel que Eclipse, NetBeans ou IntelliJ IDEA
- les bibliothèques Java Swing pour concevoir l'interface utilisateur
- la prise en charge des sockets par le système d'exploitation
- un serveur pour héberger l'application de chat
- un client pour accéder à l'application.



# Etude de l'existant

**WhatsApp** est une application mobile populaire pour fournir un service de messagerie instantanée sur les smartphones. Il utilise les services Internet pour communiquer différents types de messages texte et multimédia entre utilisateurs ou groupes. Ses utilisateurs dans le monde ont franchi le cap du milliard en février 2016. L'effet de WhatsApp sur nos vies, notre culture et notre société ne cesse d'augmenter.



# Spécification des besoins



L'application envisagée doit satisfaire les besoins fonctionnels qui seront exécutés par le système et les besoins non fonctionnels qui perfectionnent la qualité logicielle du système.

Ci-dessous on va citer l'ensemble de ces besoins.

# Exigences fonctionnelles

Les exigences fonctionnelles ou les besoins métiers représentent une déclaration des fonctions prévues d'un système et de ses composants. Elles doivent être clairement décrites afin que ce système soit opérationnel. Notre application doit couvrir principalement les besoins fonctionnels suivants :

Pour les clients :

- Inscription des nouveaux utilisateurs.
- Authentification des utilisateurs inscrit.
- Envoyer et recevoir des messages textes, images, audios, vidéos et fichiers.
- Inviter des nouveaux utilisateurs.
- Consultation et modification du profil.
- Consultation de la liste des contacts.
- Consultation du statut des contacts. (en ligne / hors ligne)
- Gestion des contacts : ajout, modification, suppression.
- Déconnexion (quitter l'application).

Pour le serveur :

- Gestion de toutes les interactions entre les utilisateurs (envoi de messages, des fichiers, d'images).
- Gestion des requêtes de connexion et d'inscription (les données doivent être vérifiée à base de celles contenues dans la base de données).

# Exigences non fonctionnelles

## 1. Utilisabilité

Il s'agit de l'expérience utilisateur et de sa qualité. La navigation dans l'application doit être facile et il faudrait accéder à une certaine fonction rapidement . En plus, l'interface doit être intuitive .Tout cela pour assurer que le système soit facile à utiliser et à comprendre.

## 2. Sécurité

Cette fonction garantit que le système est protégé contre les intrusions et que les données des utilisateurs sont protégées contre les manipulations non autorisées. A vrai dire, l'application doit être sécurisée, notamment en termes d'authentification des utilisateurs, de cryptage des communications et de prévention des attaques de type "man in the middle".

## 3. Évolutivité

L'application doit être facilement adaptable pour répondre aux besoins futurs des utilisateurs .En effet, l'application doit être capable de prendre en charge plusieurs utilisateurs simultanément.

## 4. Performance

La performance est la façon dont le système répond aux entrées de l'utilisateur. Cet attribut inclut la latence le temps d'attente moyen pour une réponse. D'ailleurs, le système doit être interactif et les mises à jour doivent être rapides. Ainsi, dans chaque action-réponse du système, il n'y a pas de retards immédiats. Il convient de dire que la connexion aux comptes d'utilisateurs ne devrait pas prendre plus de 2 secondes.

## 5. Fiabilité

L'application doit être fiable et disponible en tout temps. Des mesures de tolérance aux pannes doivent être mises en place pour minimiser les interruptions de service. Et lorsque l'utilisateur effectue une action, il doit en être accusé réception avec confirmation.

## 6. Extensibilité

L'application doit être facilement extensible, de manière à pouvoir ajouter des fonctionnalités supplémentaires si besoin.

## 7. Réactivité

L'application doit être sensible à l'entrée de l'utilisateur ou à une interruption externe qui est de la plus haute priorité et revenir au même état.

## 8. Convivialité

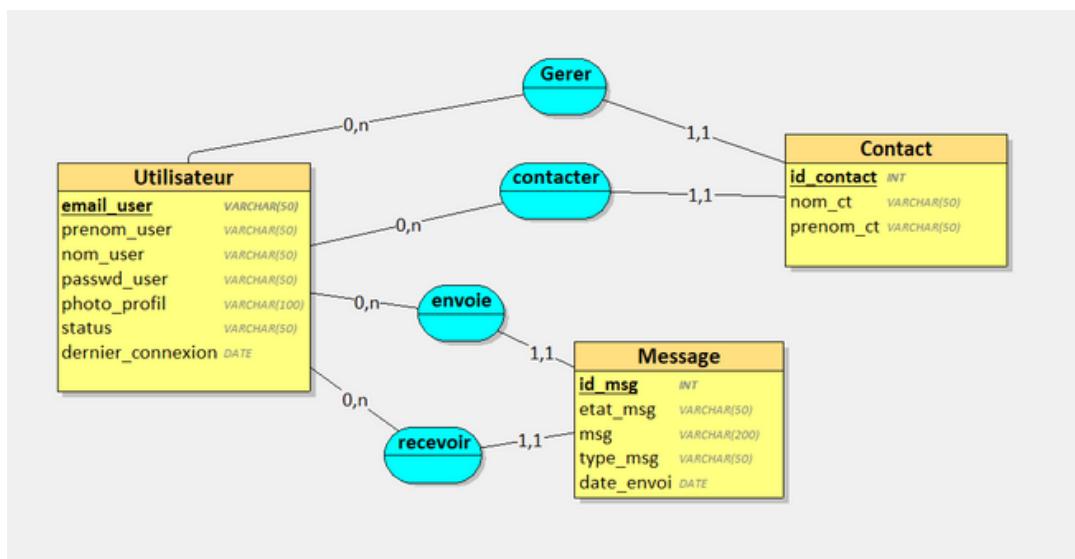
L'utilisateur doit être capable de comprendre facilement le flux de l'application, c'est-à-dire que les utilisateurs doivent pouvoir utiliser l'application sans aucune directive ni aide de manuels.



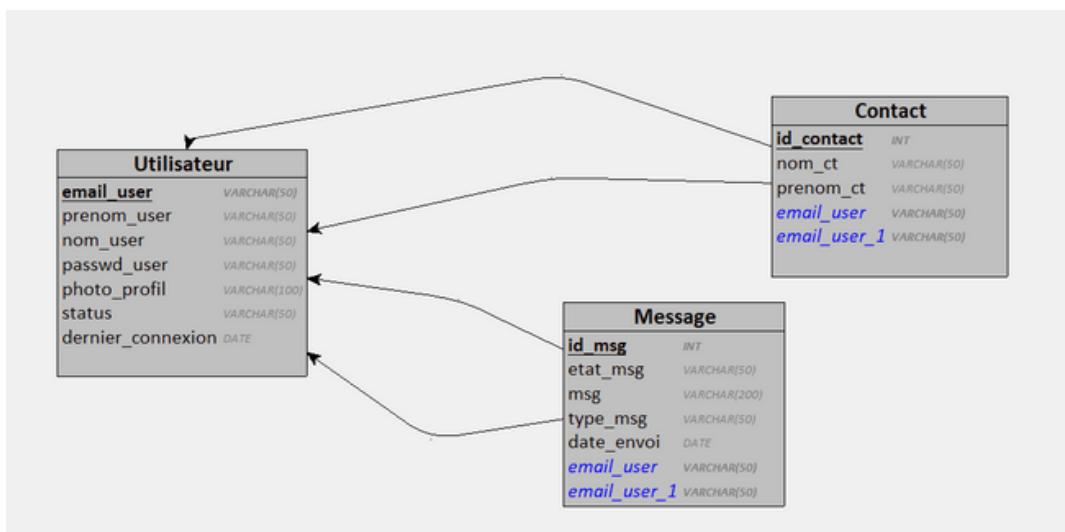
# II. Conception

Comme tout projet informatique, la phase de conception d'un système d'information nécessite des méthodes permettant de mettre en place un modèle sur lequel on va s'appuyer. Il existe plusieurs méthodes d'analyse, la méthode la plus utilisée en France étant la méthode MERISE. **MERISE** est une méthode de conception, de développement et de réalisation de projets informatiques. Le but de cette méthode est d'arriver à concevoir un système d'information. La méthode MERISE est basée sur la séparation des données et des traitements à effectuer en plusieurs modèles conceptuels et physiques.

## MCD



## MLD



# III. Modélisation

## DIAGRAMME DE CAS D'UTILISATION:

Les diagrammes de cas d'utilisation sont des diagrammes UML utilisés pour une représentation du comportement fonctionnel d'un système logiciel. Ils sont utiles pour des présentations auprès de la direction ou des acteurs d'un projet, mais pour le développement, les cas d'utilisation sont plus appropriés. En effet, un cas d'utilisation (use cases) représente une unité discrète d'interaction entre un utilisateur (humain ou machine) et un système.

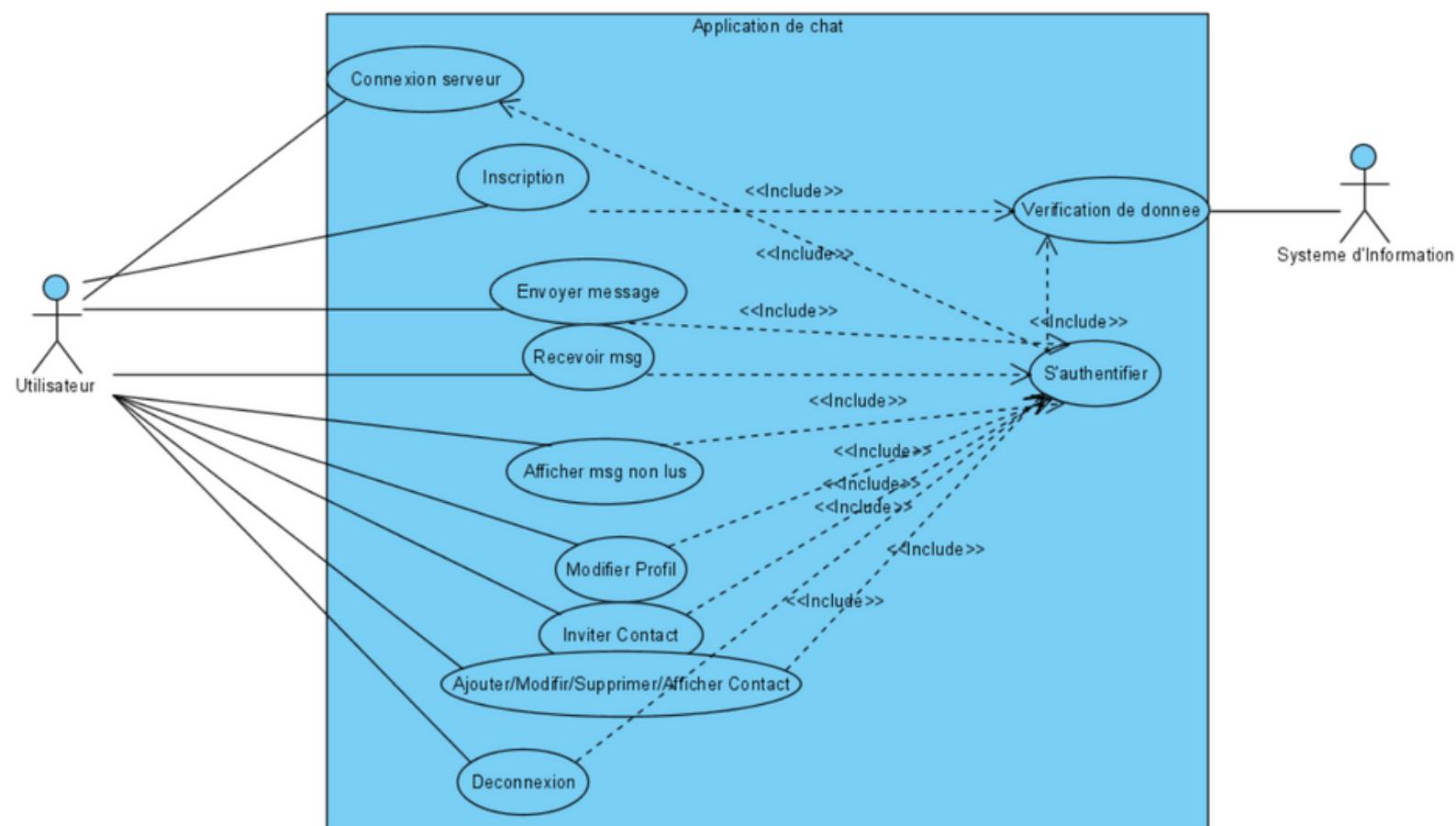


FIGURE 15 : USE CASE DIAGRAM

# III. Modélisation

## DIAGRAMME DE SÉQUENCE:

Le diagramme de séquence fait partie des diagrammes comportementaux (dynamique) et plus précisément des diagrammes d'interactions. Il permet de représenter des échanges entre les différents objets et acteurs du système en fonction du temps.

- ENVOYER MESSAGE

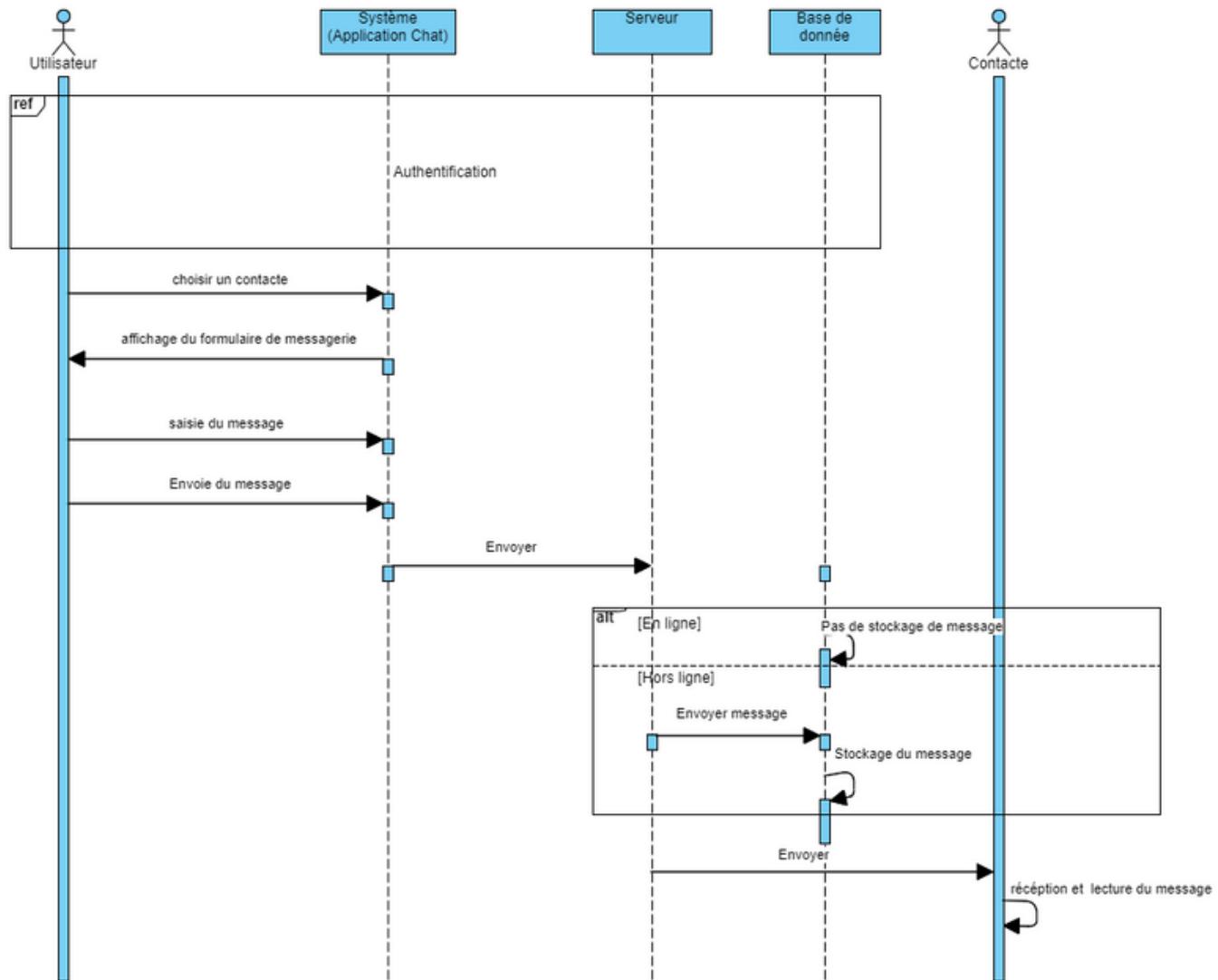


FIGURE 3 : SD ENVOYER MESSAGE

# Modélisation

## DIAGRAMME DE SÉQUENCE:

- AFFICHER MESSAGES PRIVÉS NON LUS

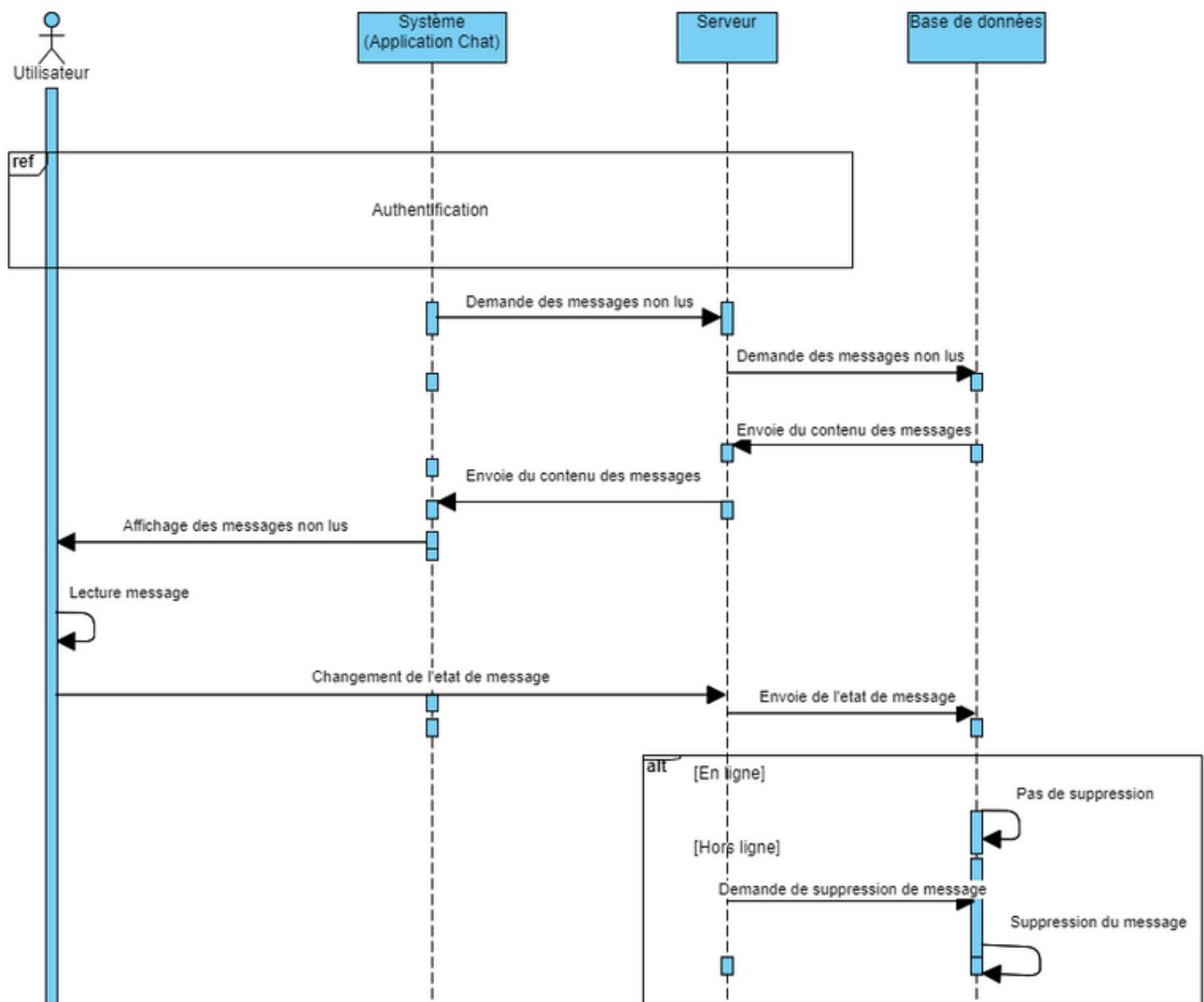


FIGURE 4 : SD AFFICHER MESSAGES NON LUS

# Modélisation

## DIAGRAMME DE SÉQUENCE:

- LOGIN

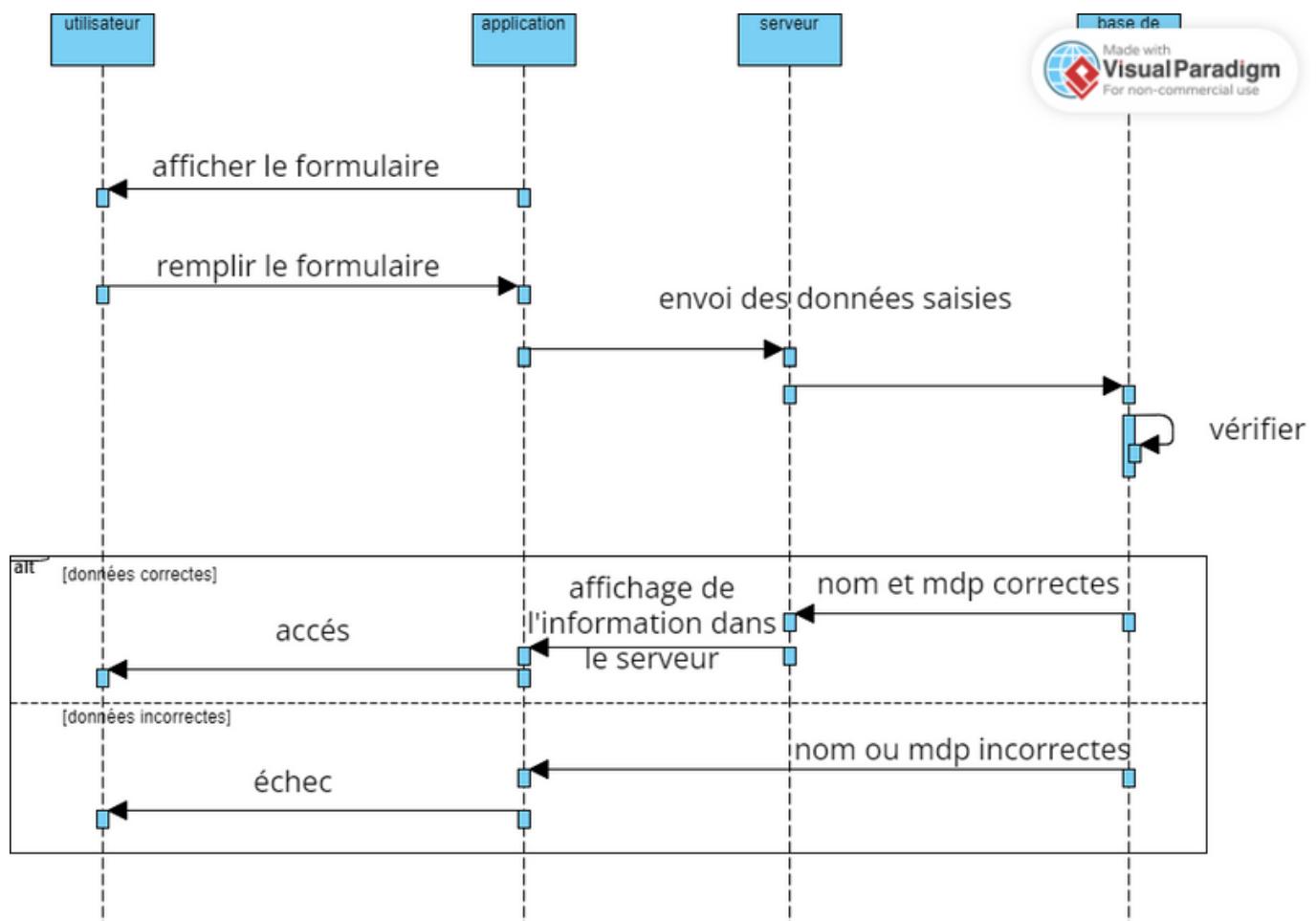


FIGURE 5 : SD S'AUTHENTIFIER

# Modélisation

## DIAGRAMME DE SÉQUENCE:

- INSCRIPTION

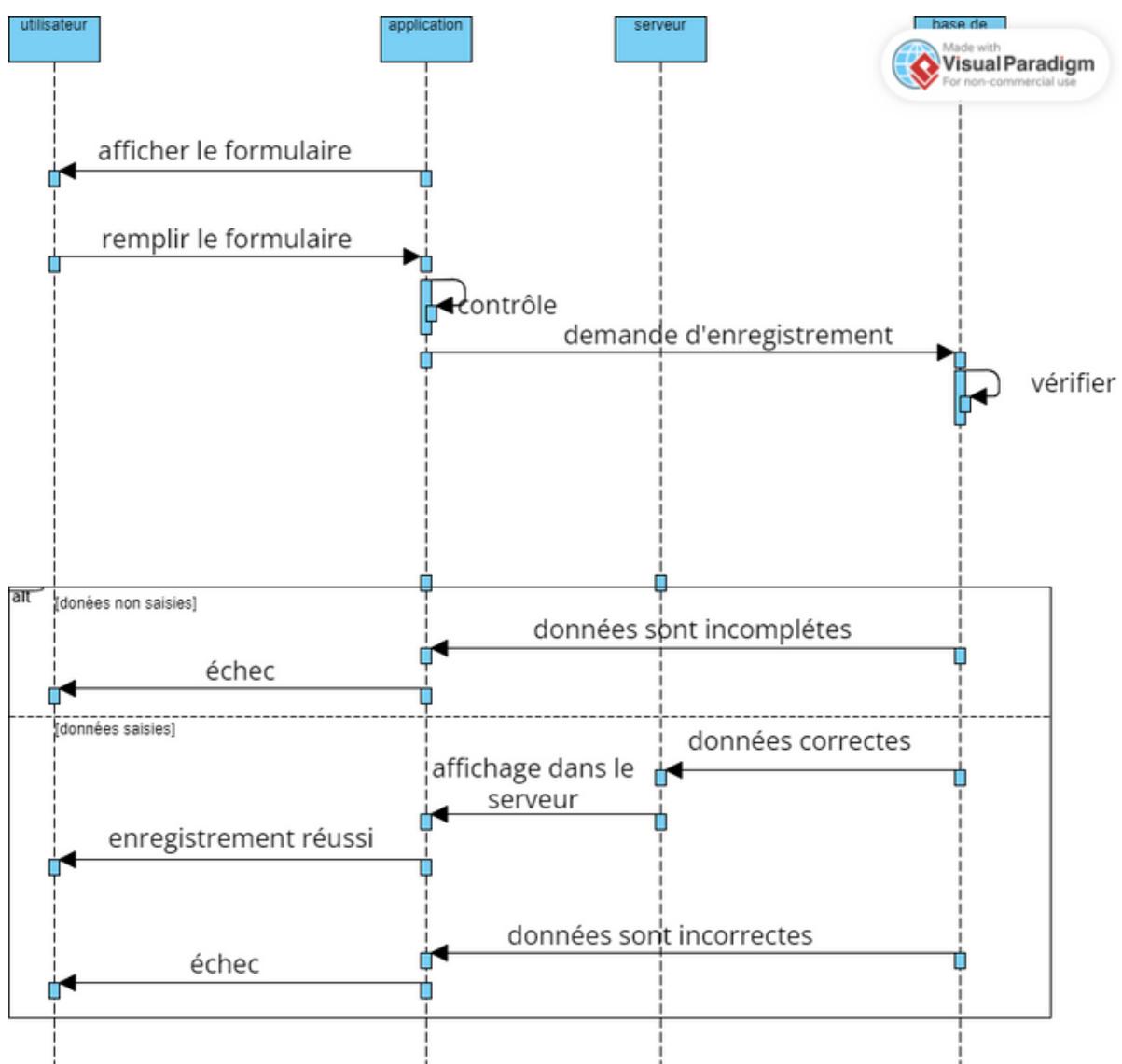


FIGURE 6 : SD S'ENREGISTRER

# Modélisation

## DIAGRAMME DE SÉQUENCE:

- DECONNEXION

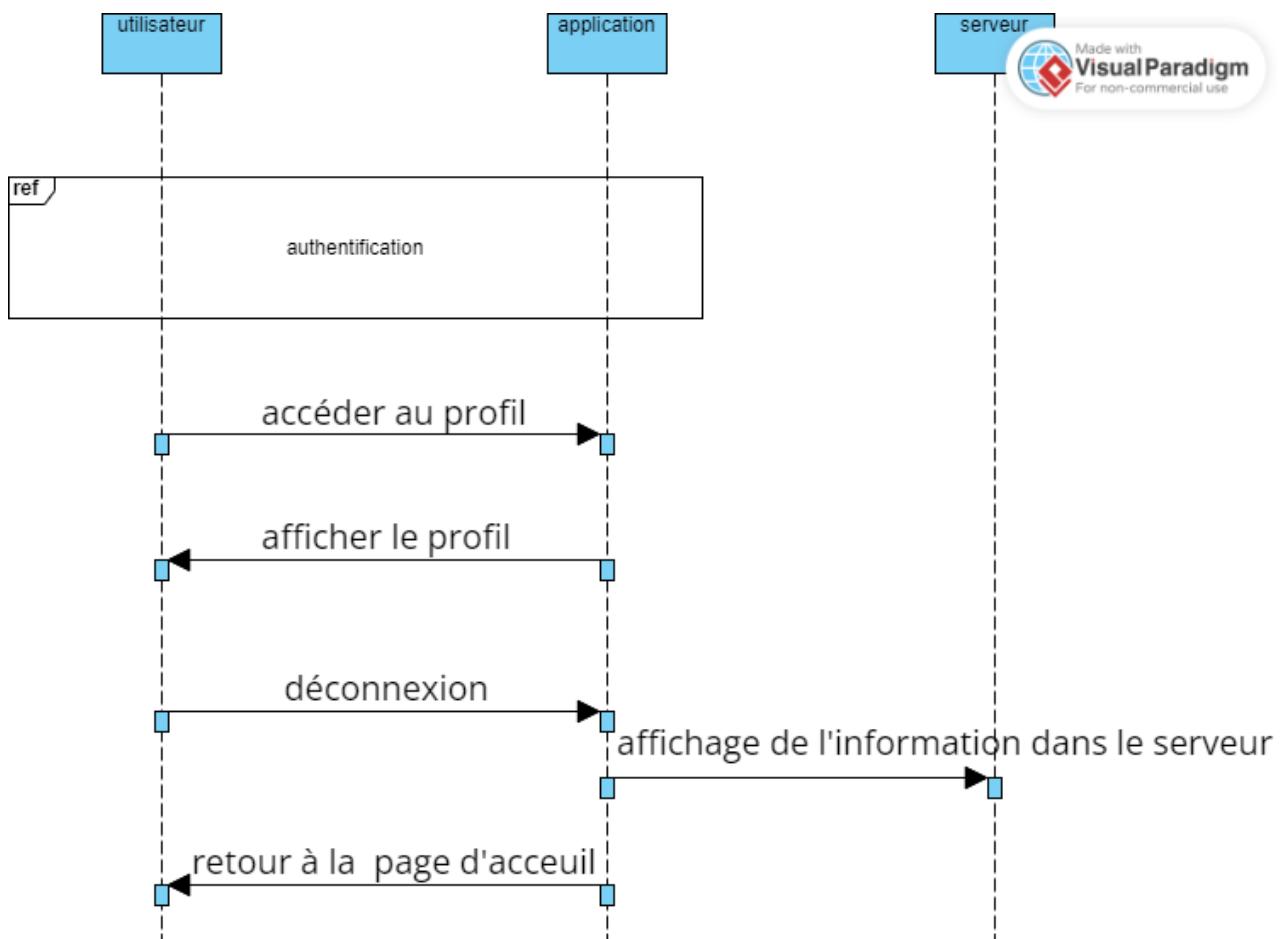


FIGURE 7 : SD SE DECONNECTER

# Modélisation

## DIAGRAMME DE SÉQUENCE:

- AJOUTER CONTACT

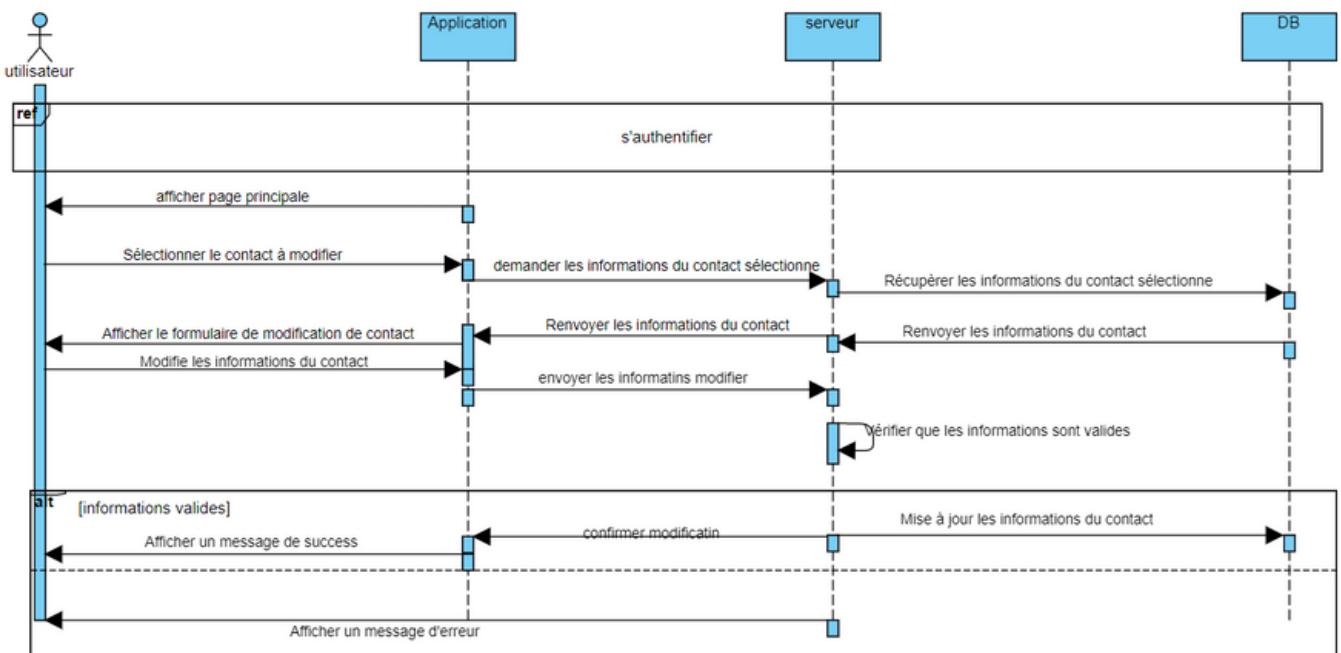


FIGURE 8 : SD AJOUTER CONTACT

# Modélisation

## DIAGRAMME DE SÉQUENCE:

- MODIFIER CONTACT

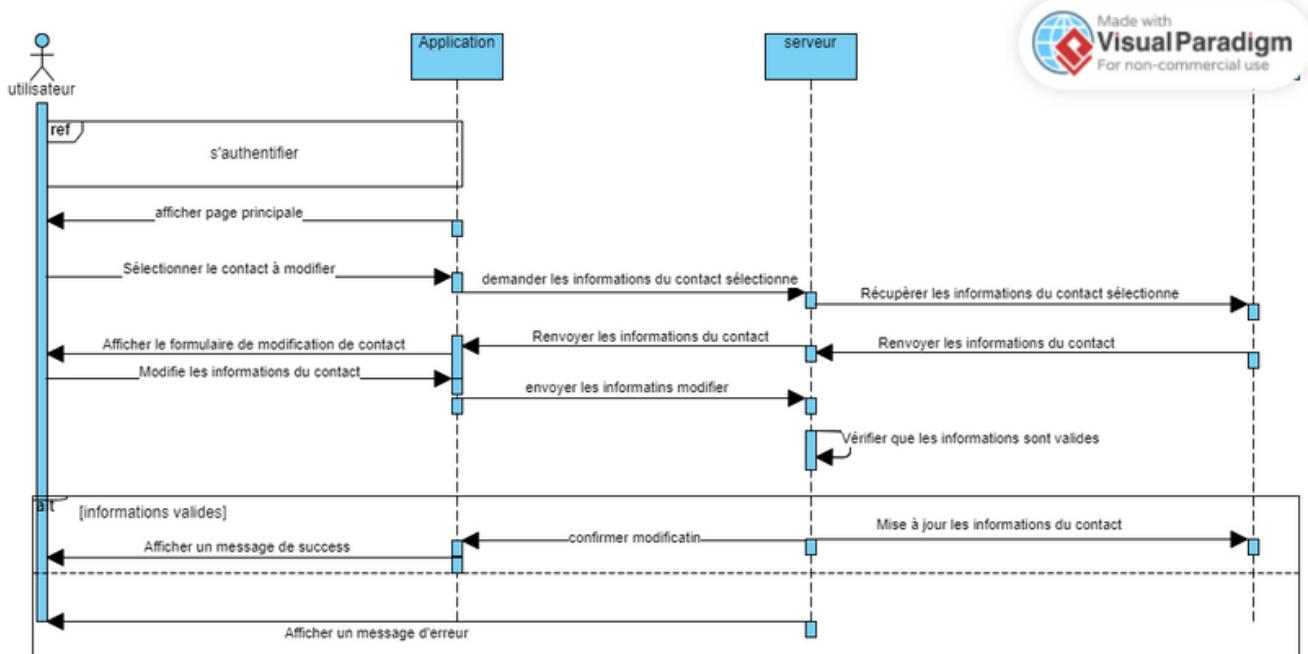


FIGURE 9 : SD MODIFIER CONTACT

# Modélisation

## DIAGRAMME DE SÉQUENCE:

- AFFICHER CONTACTS

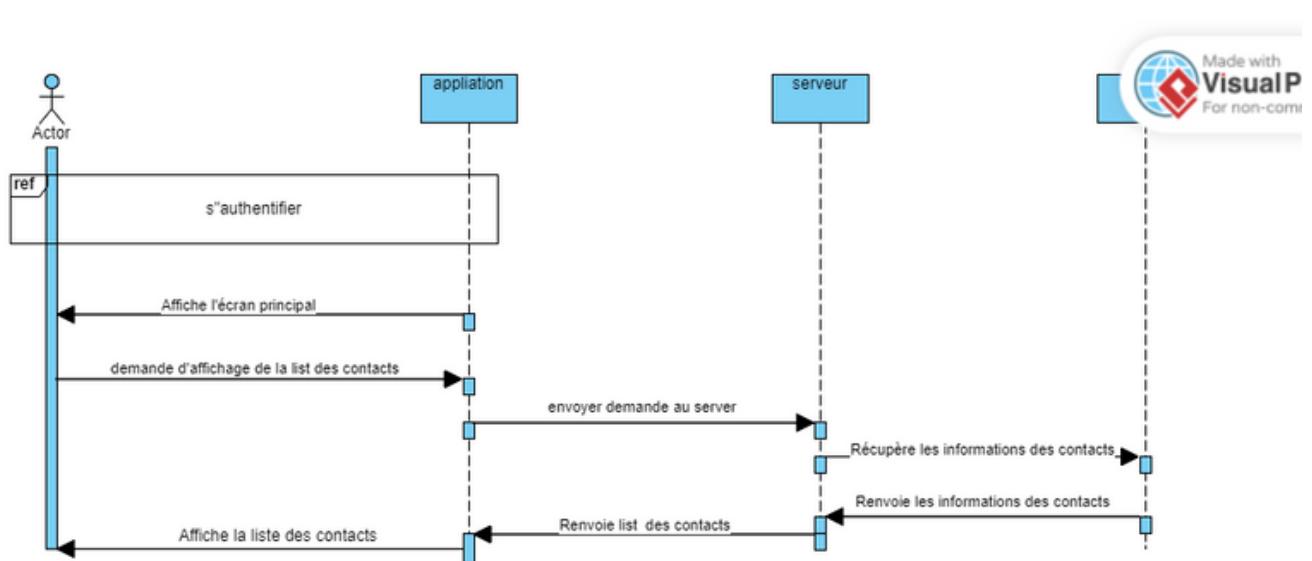


FIGURE 10 : SD AFFICHER CONTACTS

# Modélisation

## DIAGRAMME DE SÉQUENCE:

- SUPPRIMER CONTACT

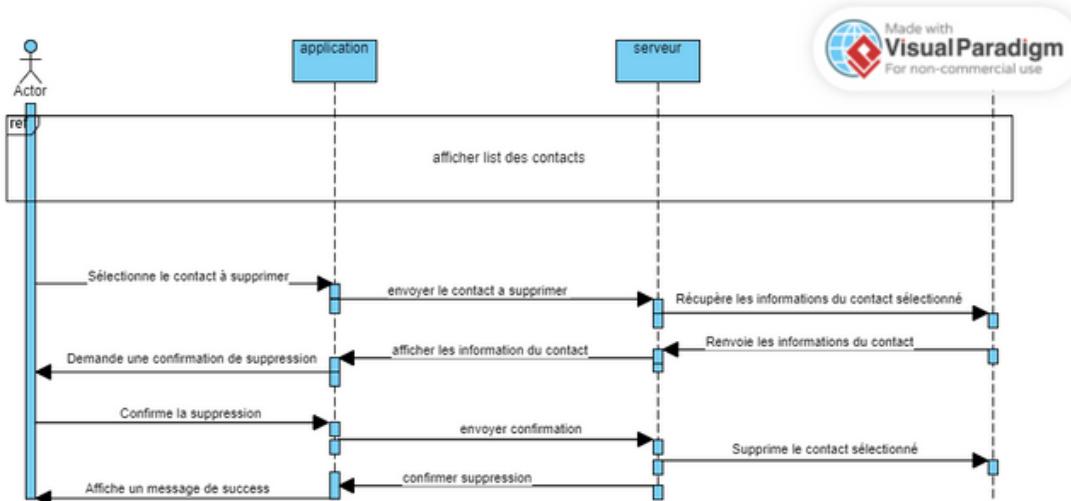


FIGURE 11 : SD SUPPRIMER CONTACT

# Modélisation

## DIAGRAMME DE SÉQUENCE:

- MODIFIER PROFILE

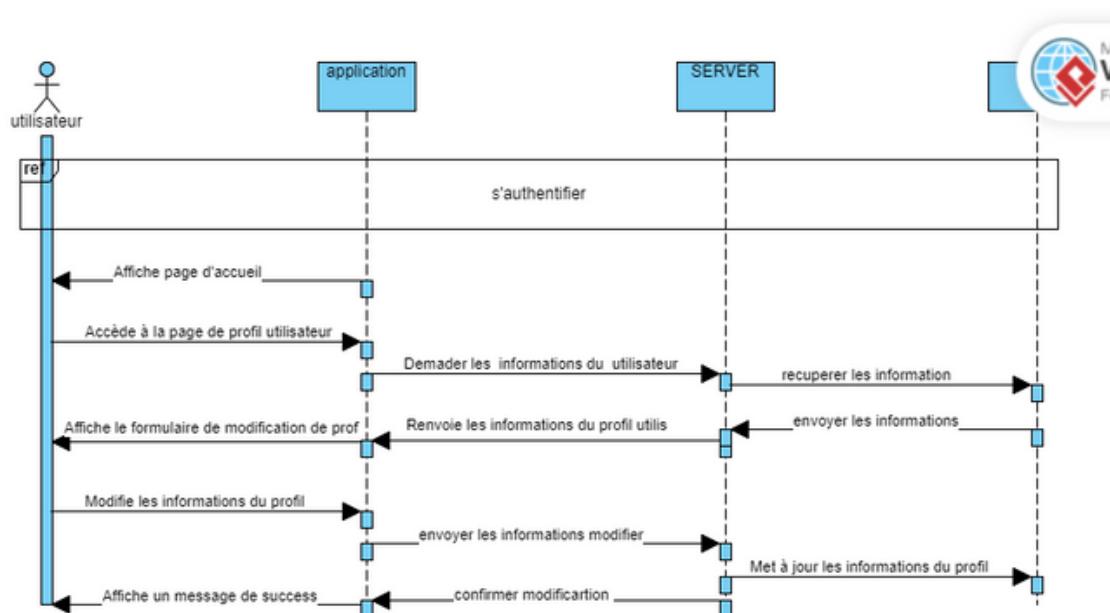


FIGURE 12 : SD MODIFIER PROFILE

# Modélisation

## DIAGRAMME DE SÉQUENCE:

- SUPPRIMER PROFILE

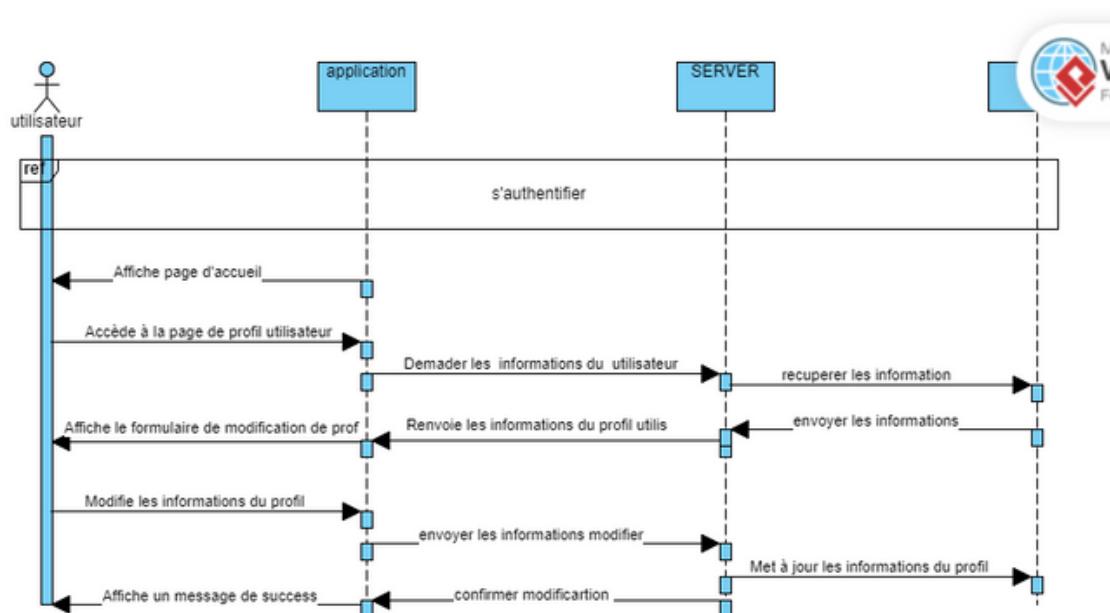


FIGURE 13 : SD MODIFIER PROFILE

# III. Modélisation

## DIAGRAMME DE CLASSE:

Le diagramme de classe constitue un élément très important de la modélisation. Il permet de définir quelles seront les composantes du système final.

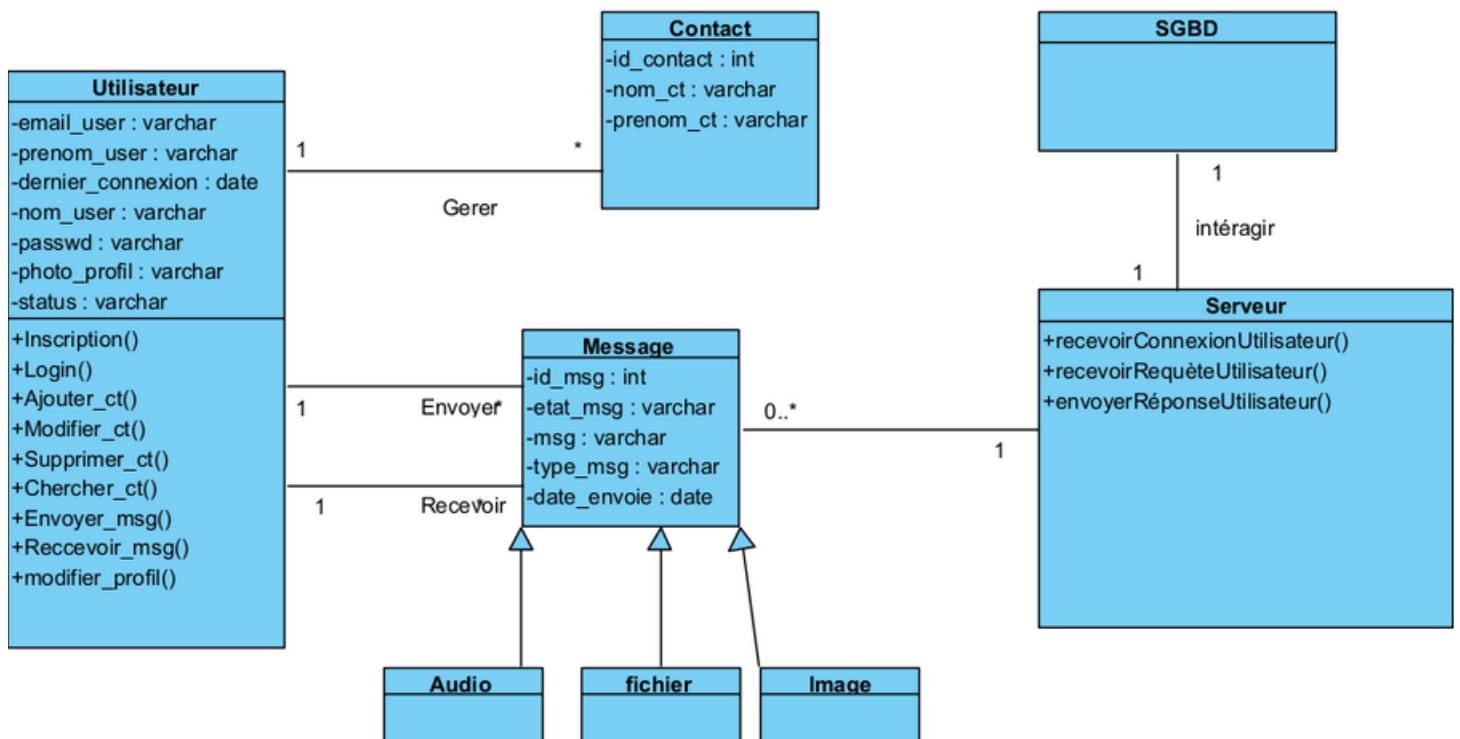


FIGURE 14 : CLASS DIAGRAM

# IV. Réalisation

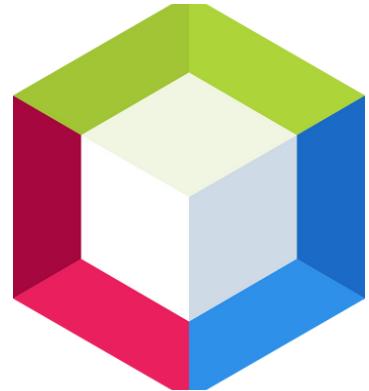
## TECHNOLOGIES UTILISEES :

### IDE : NETBEANS

**NetBeans** est un environnement de développement intégré (EDI), placé en open source par Sun en juin 2000 sous licence CDDL (Common Development and Distribution License) et GPLv2. En plus de Java, NetBeans permet la prise en charge native de divers langages tels le C, le C++, le JavaScript, le XML, le Groovy. Il offre toutes les facilités d'un IDE moderne (éditeur avec coloration syntaxique, projets multi-langage, refactoring, éditeur graphique d'interfaces et de pages Web).

Compilé en Java, NetBeans est disponible sous Windows, Linux, Solaris (sur x86 et SPARC), Mac OS X ou sous une version indépendante des systèmes d'exploitation (requérant une machine virtuelle Java). Un environnement Java Development Kit JDK est requis pour les développements en Java.

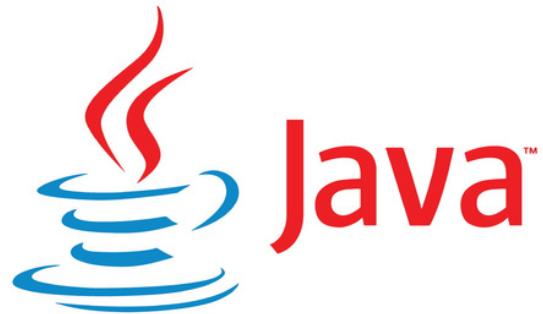
NetBeans constitue par ailleurs une plateforme qui permet le développement d'applications spécifiques (bibliothèque Swing (Java)). L'IDE NetBeans s'appuie sur cette plate forme.



# IV. Réalisation

## TECHNOLOGIES UTILISEES :

Pour la partie réalisation du projet, on a utilisé **Java** qui est un langage de programmation inspiré du langage C++, avec un modèle de programmation orienté objet. Un autre avantage est son caractère universel : le même système peut être utilisé pour une grande variété d'applications.



Depuis son origine, Java fournit plusieurs classes et interfaces destinées à faciliter l'utilisation du réseau par programmation en reposant sur les sockets. Celles-ci peuvent être mises en oeuvre pour réaliser des échanges utilisant le protocole réseau IP avec les protocoles de transport TCP ou UDP. Les échanges se font entre deux parties : un client et un serveur.

Le système des sockets est le moyen de communication interprocessus développé pour l'Unix Berkeley (BSD). Il est actuellement implémenté sur tous les systèmes d'exploitation utilisant TCP/IP.

Java prend en charge deux protocoles : **TCP** et **UDP**.

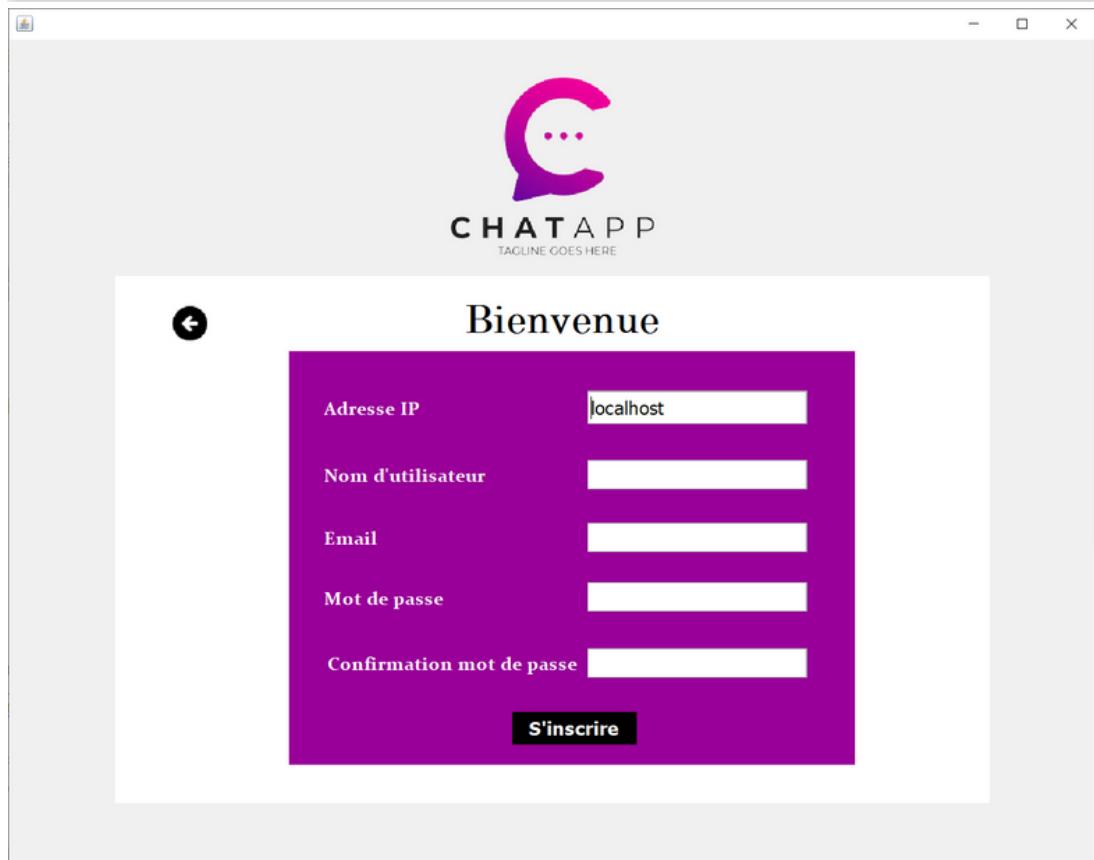
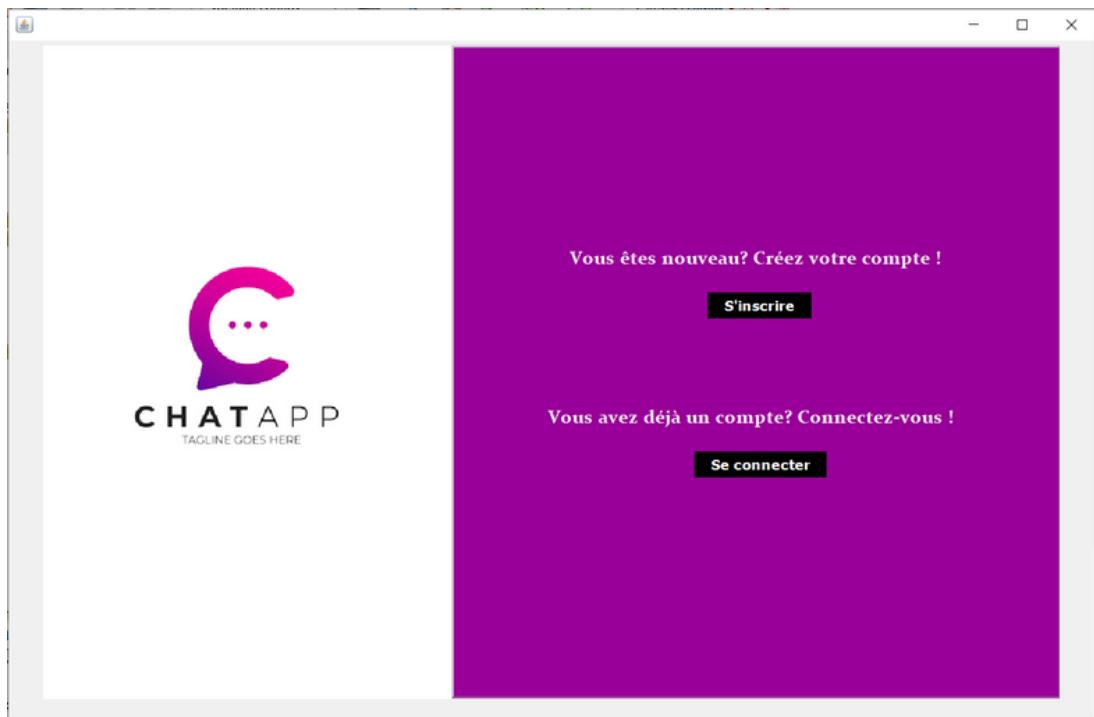
Les classes et interfaces utiles au développement réseau sont regroupées dans le package **java.net**.

**Swing** fait partie de la bibliothèque Java Foundation Classes (JFC). C'est une API dont le but est similaire à celui de l'API AWT mais dont les modes de fonctionnement et d'utilisation sont complètement différents. Swing a été intégré au JDK depuis sa version 1.2. Cette bibliothèque existe séparément pour le JDK 1.1.

Swing propose de nombreux composants dont certains possèdent des fonctions étendues, une utilisation des mécanismes de gestion d'événements performants (ceux introduits par le JDK 1.1) et une apparence modifiable à la volée (une interface graphique qui emploie le style du système d'exploitation Windows ou Motif ou un nouveau style spécifique à Java nommé Metal).

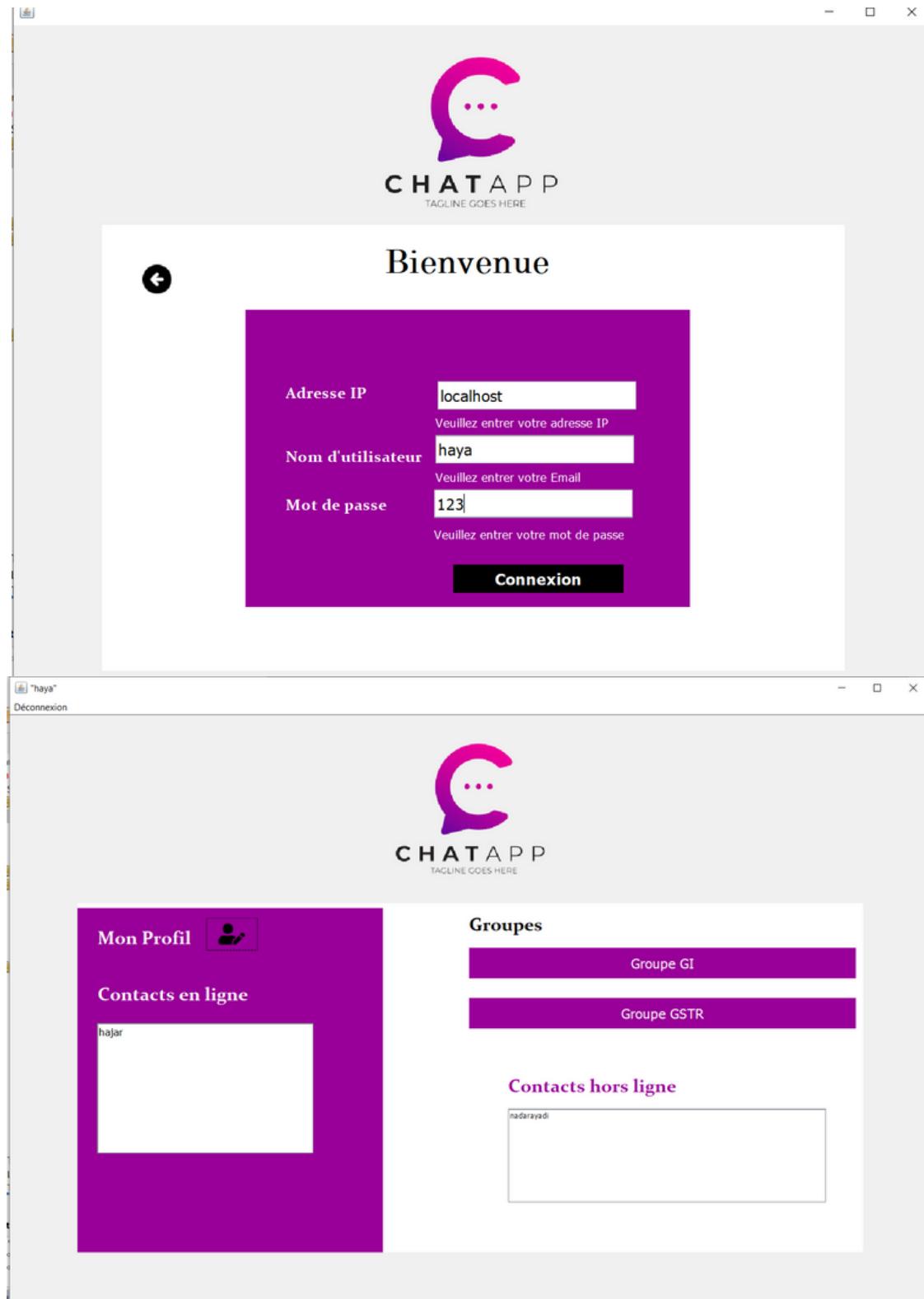
# IV. Réalisation

## CAPTURE DE L'INTERFACE DE L'APPLICATION :



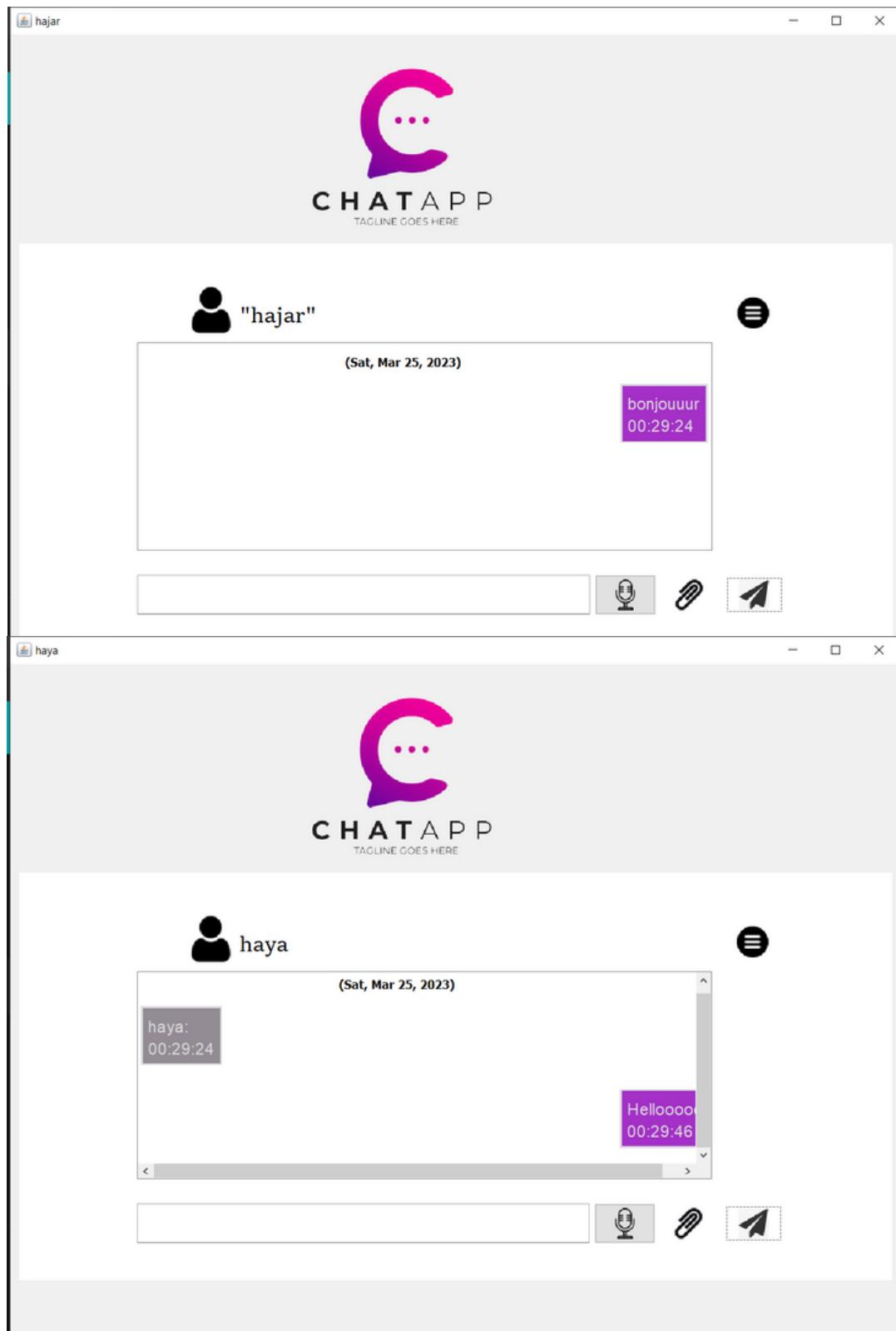
# IV. Réalisation

## CAPTURE DE L'INTERFACE DE L'APPLICATION :



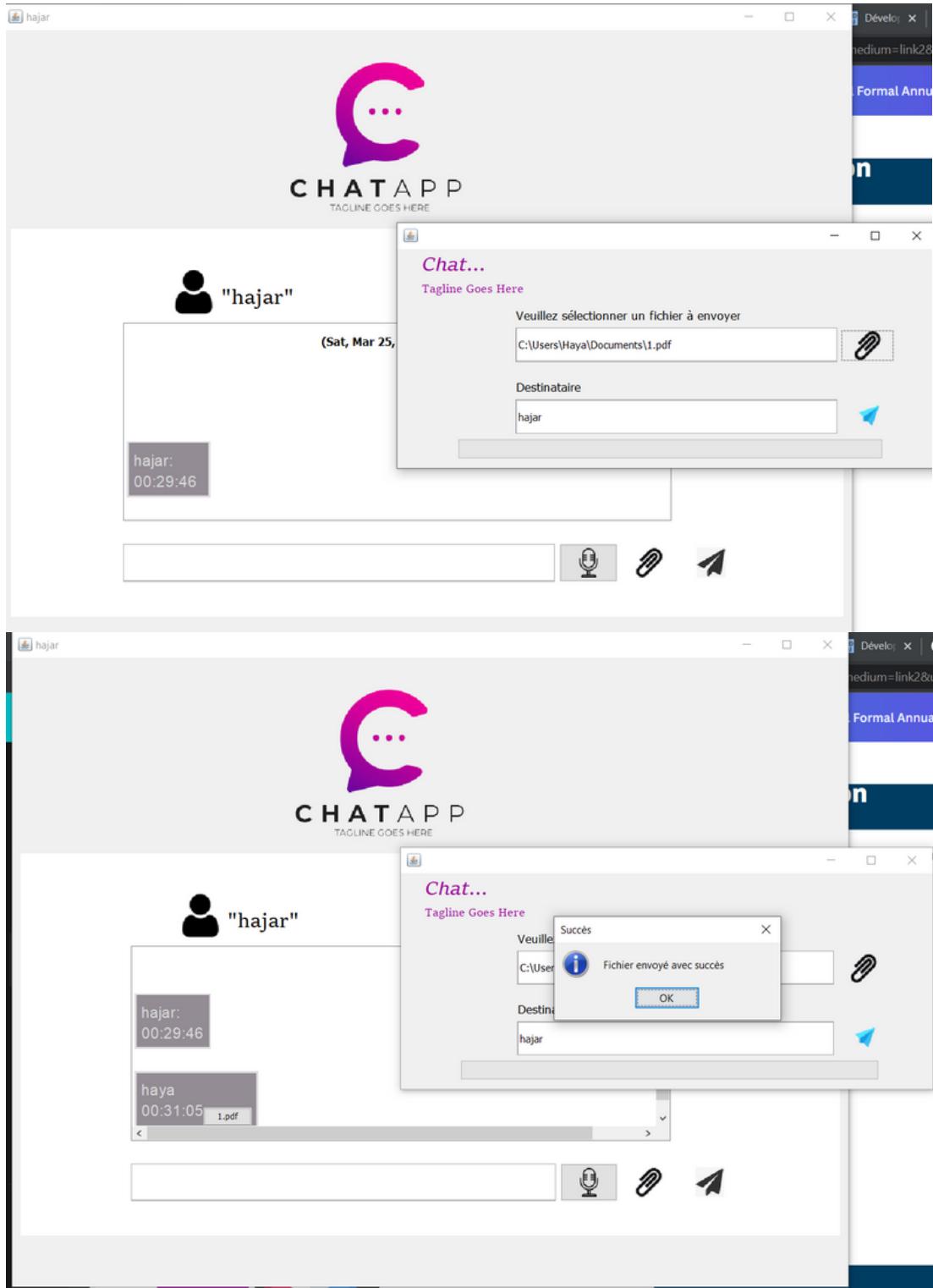
# IV. Réalisation

## CAPTURE DE L'INTERFACE DE L'APPLICATION :



# IV. Réalisation

## CAPTURE DE L'INTERFACE DE L'APPLICATION :



# IV. Réalisation

## CAPTURE DE L'INTERFACE DE L'APPLICATION :



# IV. Réalisation

## CAPTURE DE L'INTERFACE DE L'APPLICATION :

