INSA Rouen Normandie

Département ASI

4ème année

EC Informatique Répartie

Messagerie instantanée 1

Rapport de projet

Auteurs
Gautier Darchen
Alexandre Huat
Marie-Andrée Jolibois
Romain Judic
Alexandre Le Lain

17 mai 2017

Table des matières

Chapitre 1

Introduction

L'application à développer est une plateforme graphique répartie de messagerie instantanée.

1.1 Fonctions principales

Les fonctions principales de cette application peuvent être scindées en trois catégories :
— l'échange de messages ;
— le filtrage des messages en fonction de leur contenu ;
— l'utilisation d'un avatar.

Concernant la première fonctionnalité, les interlocuteurs pourront s'envoyer des messages écrits de façon instantanée. Ils auront en plus la possibilité de communiquer avec d'autres formats via un système de visio- ou audio-conférence intégré à l'application.

Le filtrage des messages prend en charge le contrôle parental et la modération. Les utilisateurs auront la possibilité de choisir d'activer ou non ce système de filtrage. De plus, ce dernier pourra être personnalisé.

Concernant la dernière fonctionnalité, un système d'avatar pourra être utilisé par les utilisateurs pour les conversations audio et écrites.

1.2 Utilisateurs

Il existe plusieurs types d'utilisateurs.

Utilisateur *lambda* : il peut créer (ouvrir) une conversation, participer à une conversation qu'il sélectionne. Il peut paramétrer ses filtres. À la connexion, il choisit aussi un login et un avatar.

Modérateur : il peut émettre des messages de modération et supprimer des messages d'utilisateurs.

Plusieurs utilisateurs peuvent discuter en même temps sur une même conversation.

Les seuls prérequis à l'utilisation sont savoir exécuter des lignes de commandes pour installer et exécuter une application.

1.3 Contraintes

1.3.1 Contraintes matérielles

L'utilisateur doit disposer d'une machine reliée à internet.

Pour profiter du service visio, l'utilisateur doit posséder un matériel de capture vidéo (webcam) et d'entrées/sorties audio.

Pour utiliser le service audio, l'utilisateur doit posséder une entrée et une sortie audio.

La majorité des calculs est réalisée côté serveur, ce qui n'implique pas un besoin de ressources conséquentes côté client.

1.3.2 Contraintes logicielles

Le client doit disposer du Java Runtime Environnement 8 et de la bibliothèque JavaFX.

Installer JavaFX en ligne de commande sous Ubuntu 16.04+ :

```
sudo apt-get update
sudo apt-get install openjfx
```

Chapitre 2

Spécifications

2.1 Spécifications fonctionnelles

Les comptes utilisateurs ne sont pas persistants, ils n'existent que de l'ouveture à la fermeture d'un service client. L'utilisateur peut paramétrer son compte (pseudo et avatar) avant de participer à une conversation.

Les utilisateurs modérateurs peuvent supprimer des messages.

Tout utilisateur peut ouvrir ou rejoindre une conversation. Les conversations existent indépendamment des utilisateurs, elles sont ouvertes aussi longtemps que le serveur les hébergeant est actif.

L'IA est un helpbot présent sur chaque conversation. L'utilisateur peut l'invoquer en entrant des mots clés commençant pas un backslash.

bonjour salue l'utilisateur et déroule un menu d'aide.

profil indique à l'utilisateur comment paramétrer son profil.

filtre indique à l'utilisateur comment paramétrer les filtres.

chat1 indique à l'utilisateur comment ouvrir une conversation.

chat2 indique à l'utilisateur comment rejoindre une conversation.

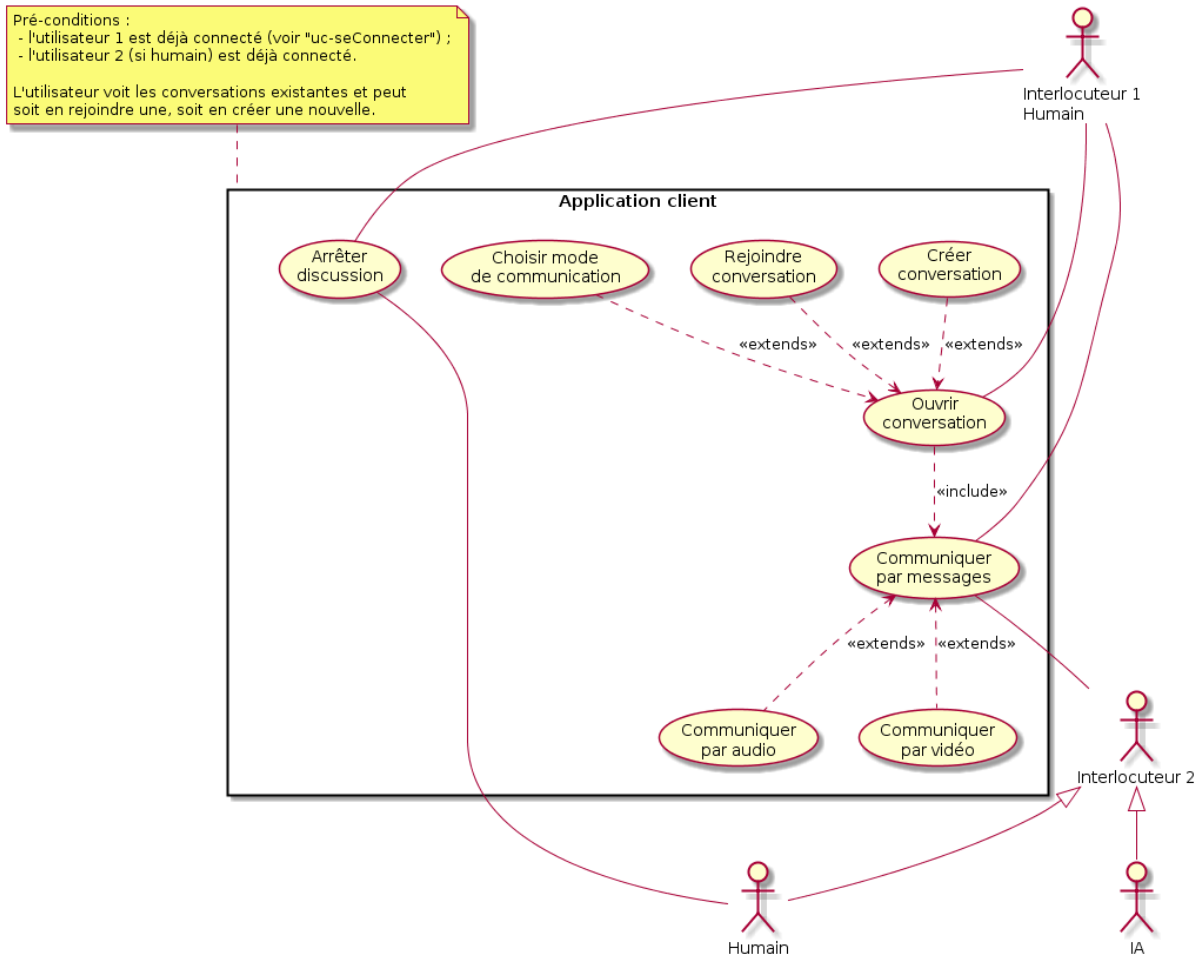
video indique à l'utilisateur comment démarrer une conversation vidéo.

audio indique à l'utilisateur comment démarrer une conversation audio.

Cas d'utilisation

Cette section présente les cas d'utilisation suivants :
— discuter ;
— paramétrer le filtrage ;
— filtrer les messages ;
— se connecter.

FIGURE 2.1 – Cas d'utilisation **discuter**



Le cas d'utilisation (figure 2.1) représente les interactions entre les différents interlocuteurs et les différents moyens mis à leur disposition pour communiquer.

FIGURE 2.3 – Cas d'utilisation **paramétrer le filtrage**

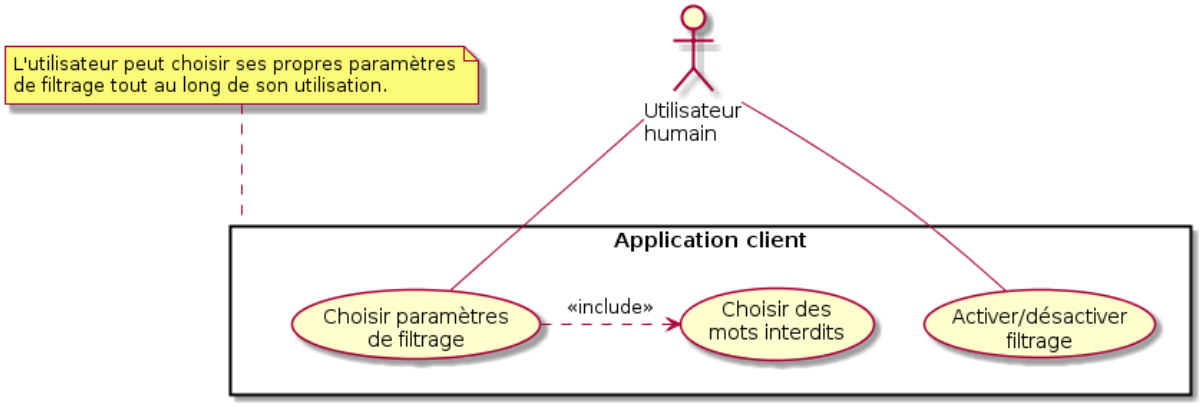


FIGURE 2.2 – Cas d'utilisation **filtrer les messages**

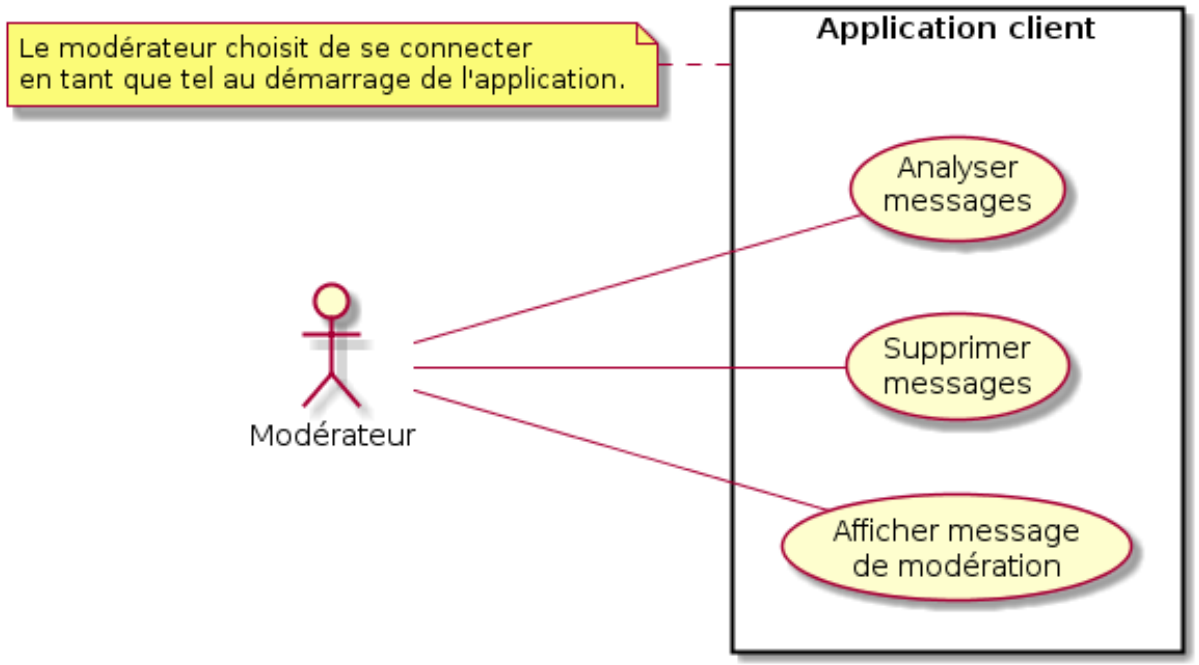
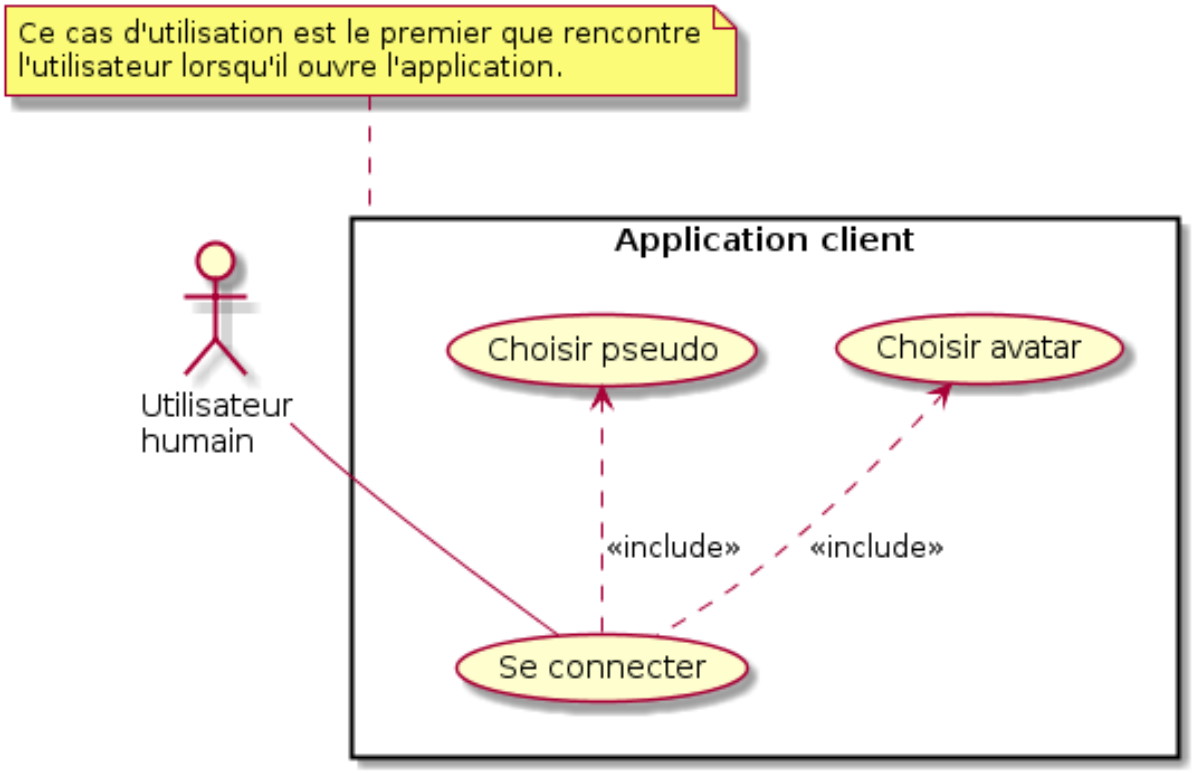


FIGURE 2.4 – Cas d'utilisation **se connecter**



2.2 Spécifications d'interfaces

Maquettes

Cette section présente les maquettes de l'application.

FIGURE 2.5 – Exemple de conversation texte

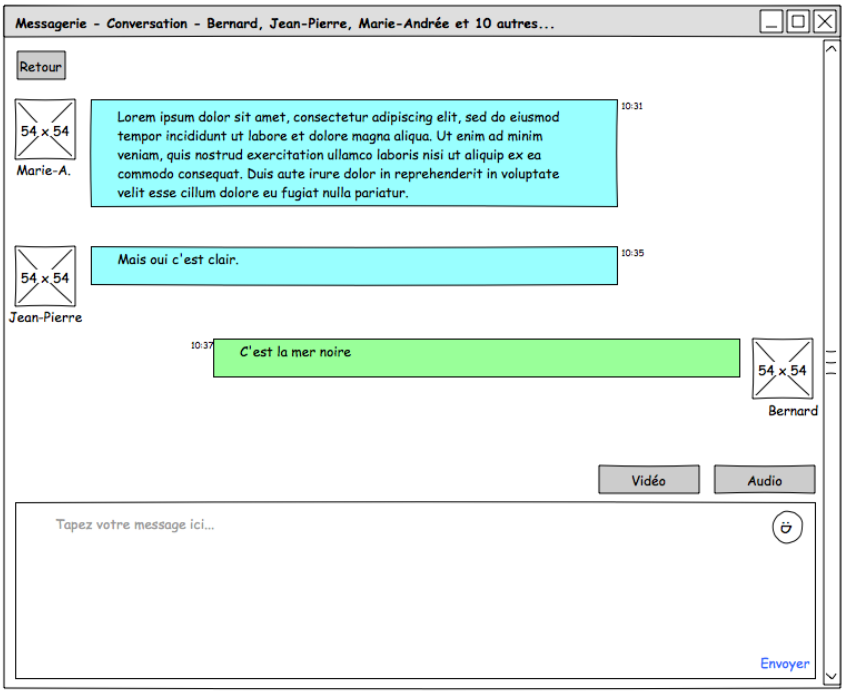


FIGURE 2.6 – Conversation vidéo

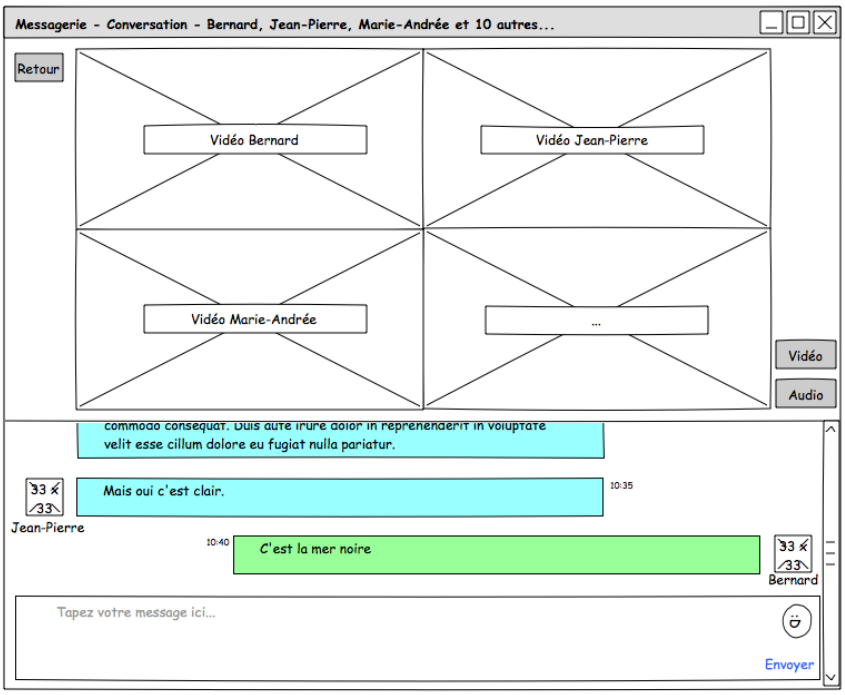


FIGURE 2.7 – Liste des conversations

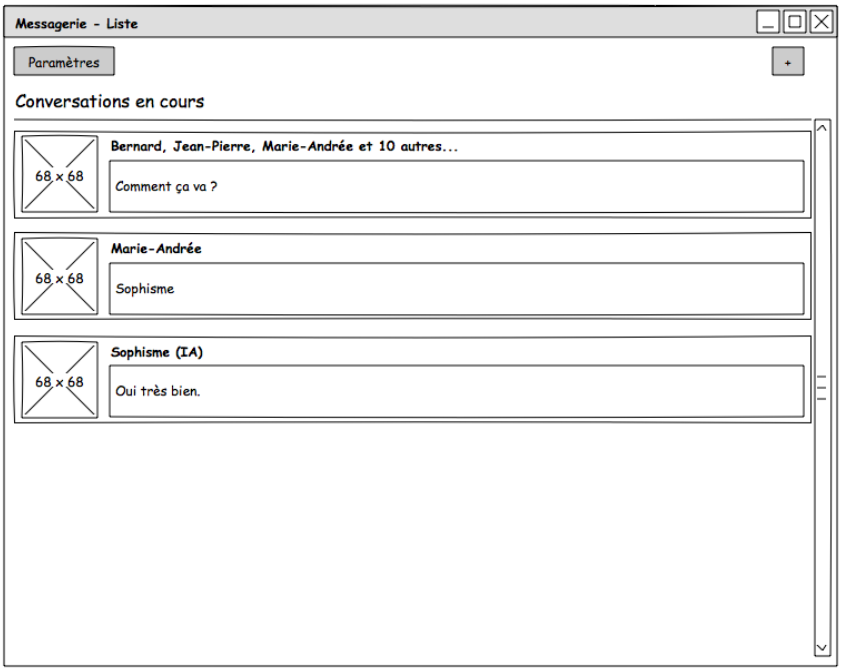
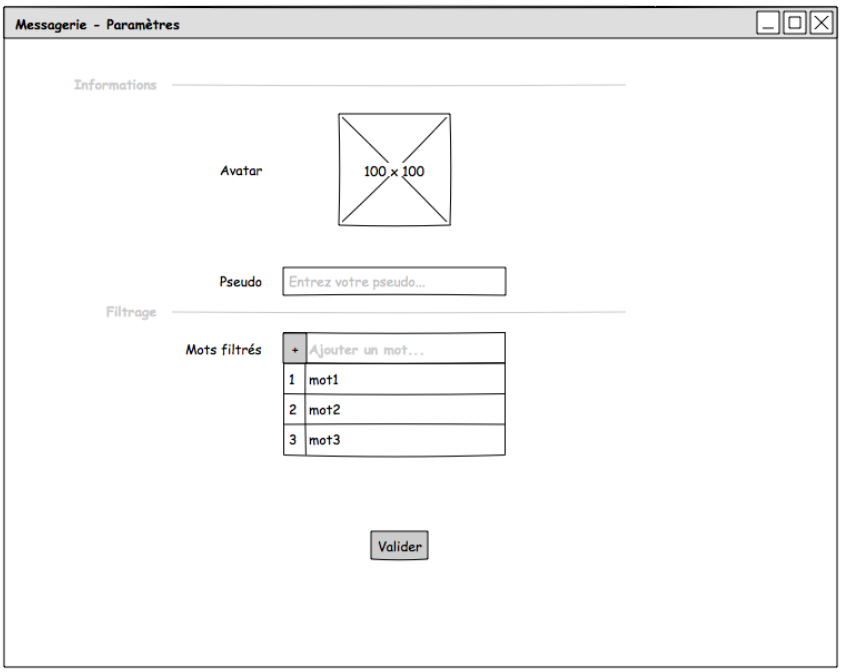


FIGURE 2.8 – Interface des paramètres



2.3 Spécifications opérationnelles

Le système de messagerie répond instantanément.

Les messages sont sauvegardés tant qu'il subsiste un utilisateur connecté à la conversation.

Tant que le serveur est actif, le service client est disponible.

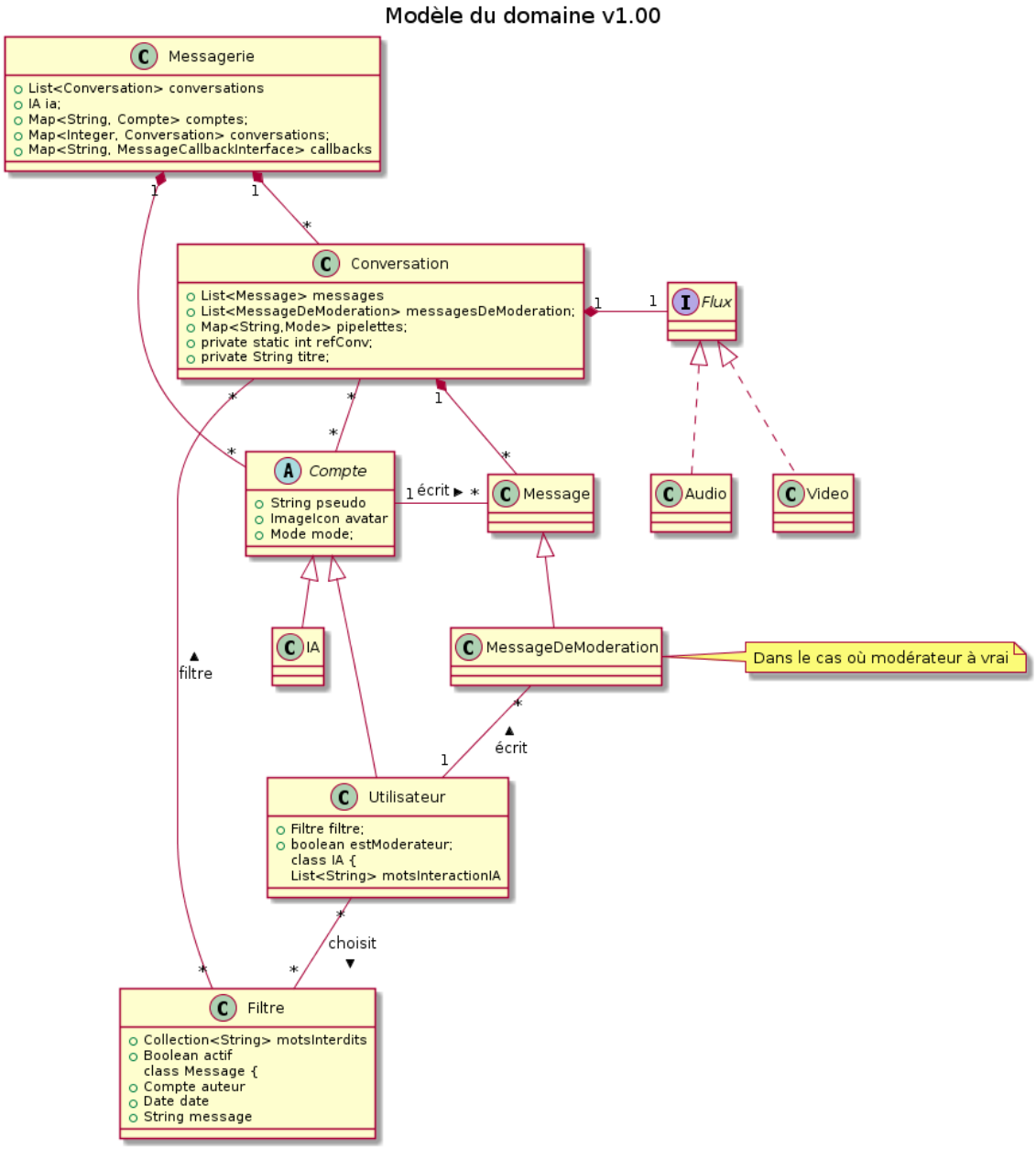
Lors de l'utilisation de l'application, la référence du compte est utilisée pour savoir qui participe à la conversation. Cette référence du compte transite donc entre l'application client et l'application serveur. Aucune mesure de sécurité n'est prévue pour éviter une transmission visible de cette référence.

Chapitre 3

Conception préliminaire

3.1 Modèle du domaine

FIGURE 3.1 – Modèle du domaine



3.2 Diagrammes de séquence système

FIGURE 3.2 – Diagramme de séquence système de **communiquer**

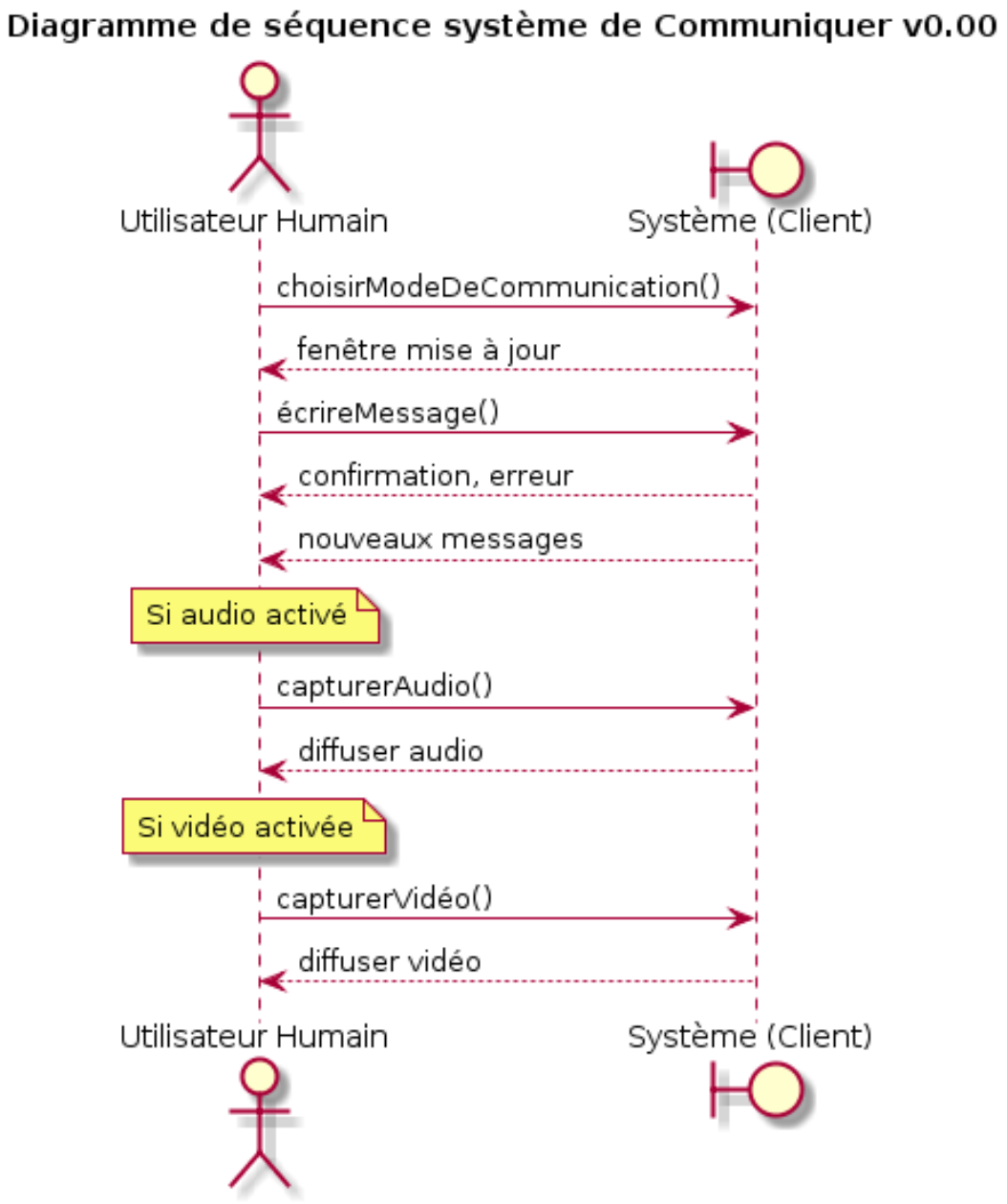


FIGURE 3.3 – Diagramme de séquence système de **se connecter**

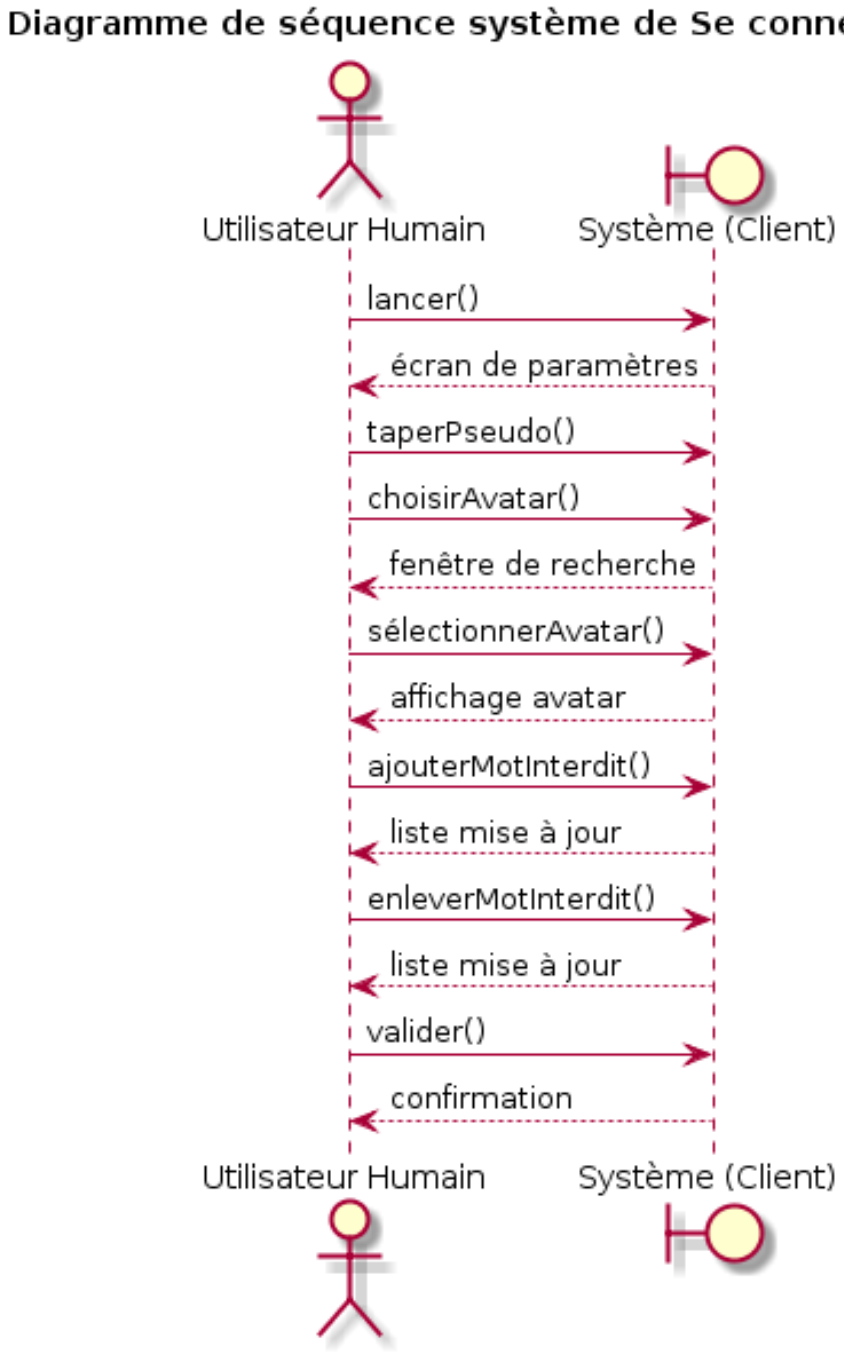
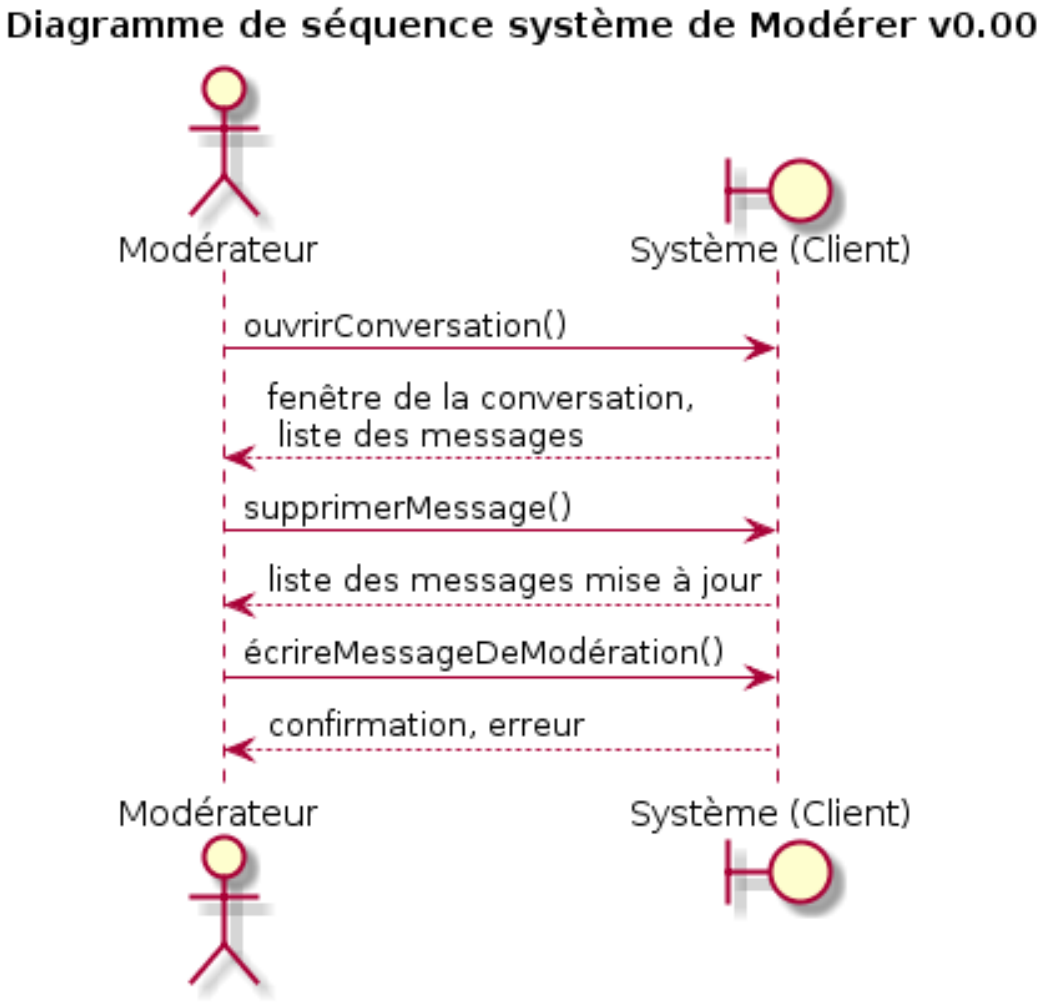


FIGURE 3.4 – Diagramme de séquence système de **modérer**



3.2.1 Modification du modèle du domaine

— Ajout d'attributs dans la classe `Messagerie`.
— L'interface `Moderateur` est remplacé par un attribut booléen de la classe `Utilisateur`.

3.3 Diagrammes d'activités

FIGURE 3.5 – Diagramme d'activité de **se connecter**

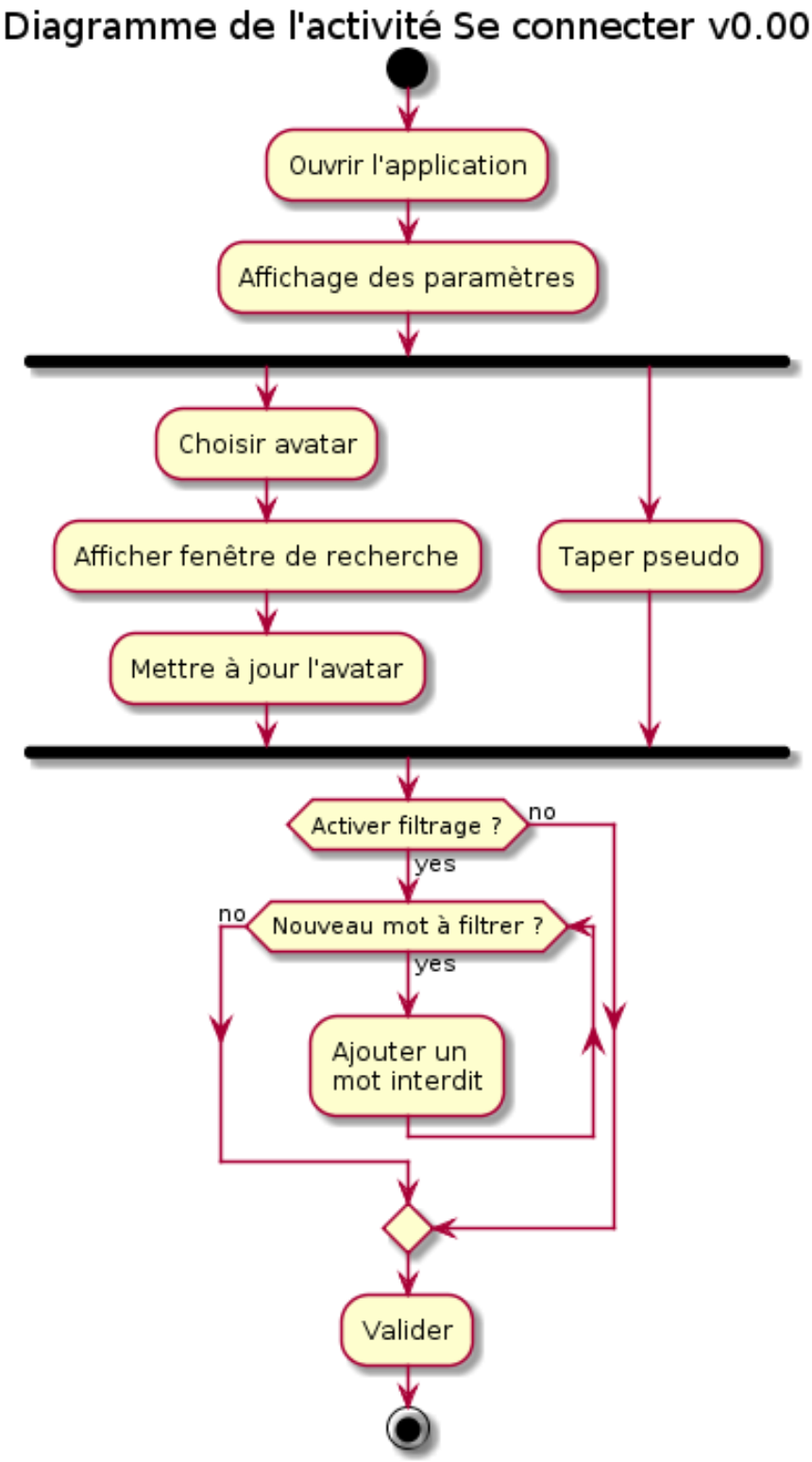


FIGURE 3.6 – Diagramme d'activité de **choisir une conversation**

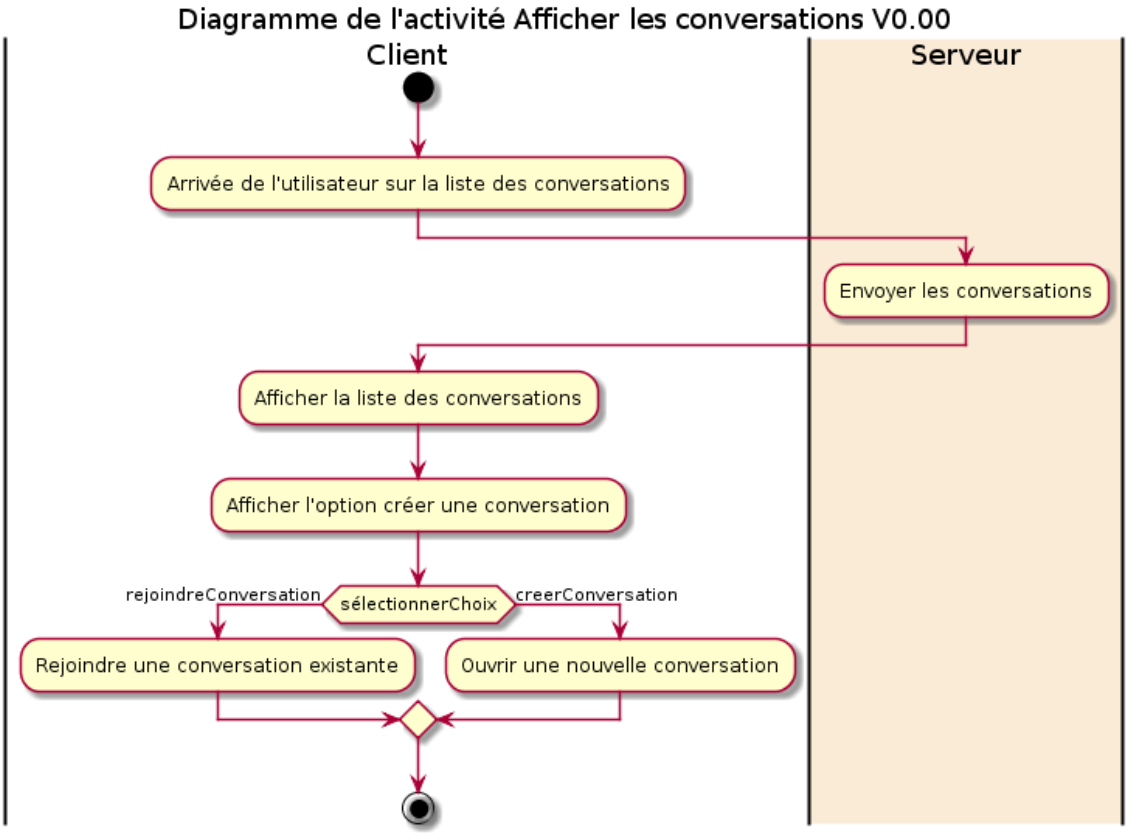


FIGURE 3.7 – Diagramme d'activité d'**ouvrir une conversation**

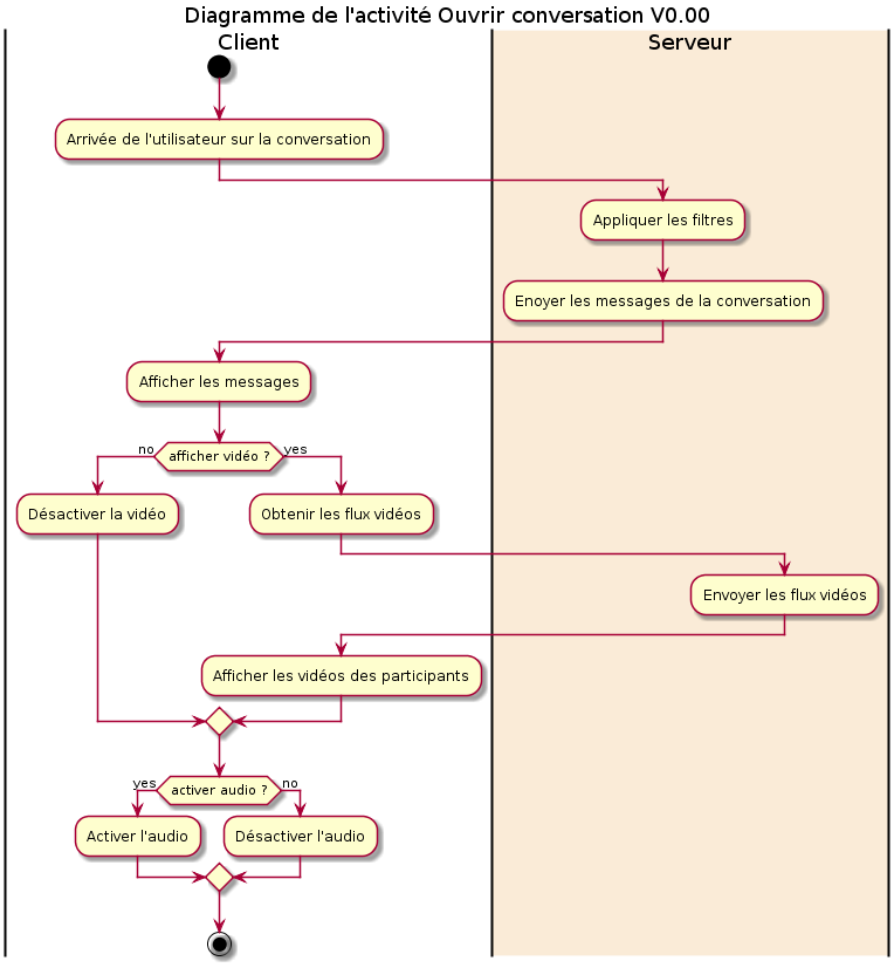
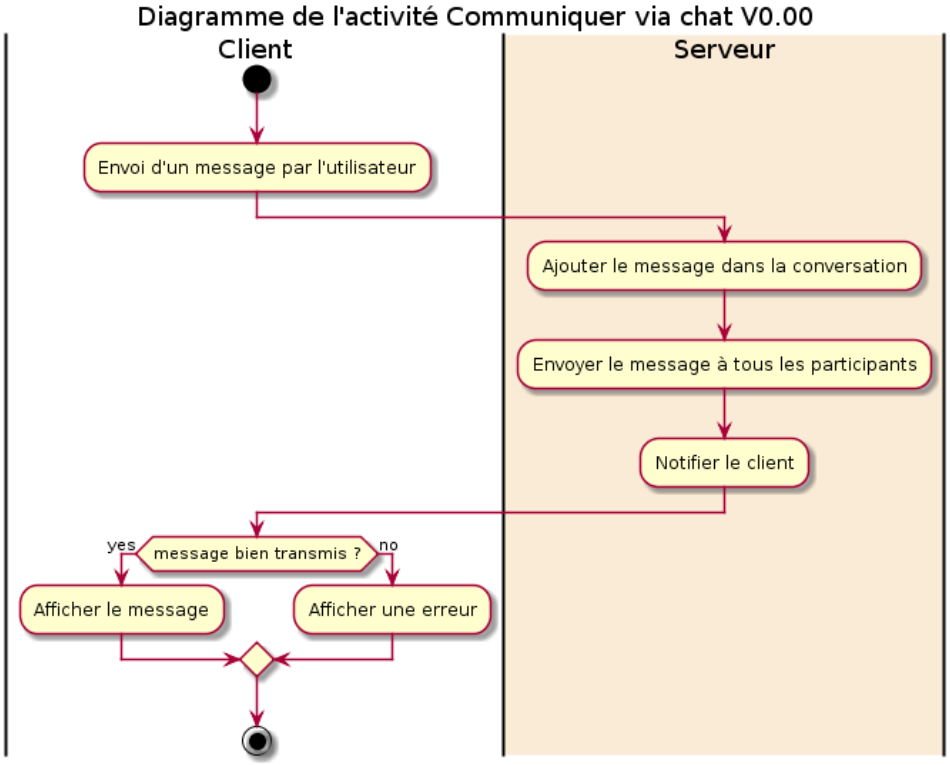


FIGURE 3.8 – Diagramme d'activité de **communiquer**



3.4 Diagrammes d'interaction

FIGURE 3.9 – Diagramme de séquence d'analyse de **se connecter**

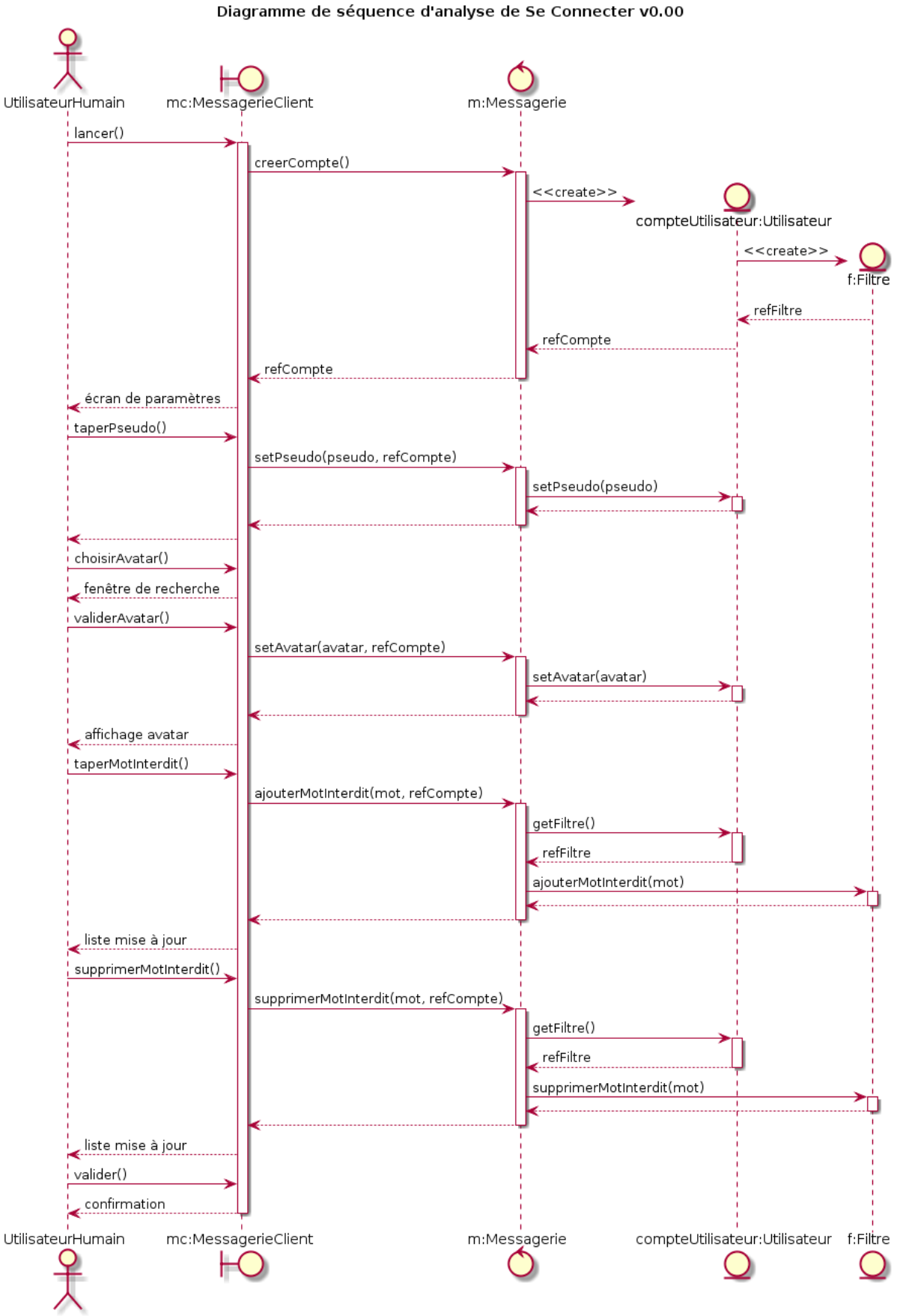


FIGURE 3.10 – Diagramme de séquence d'analyse de **communiquer**

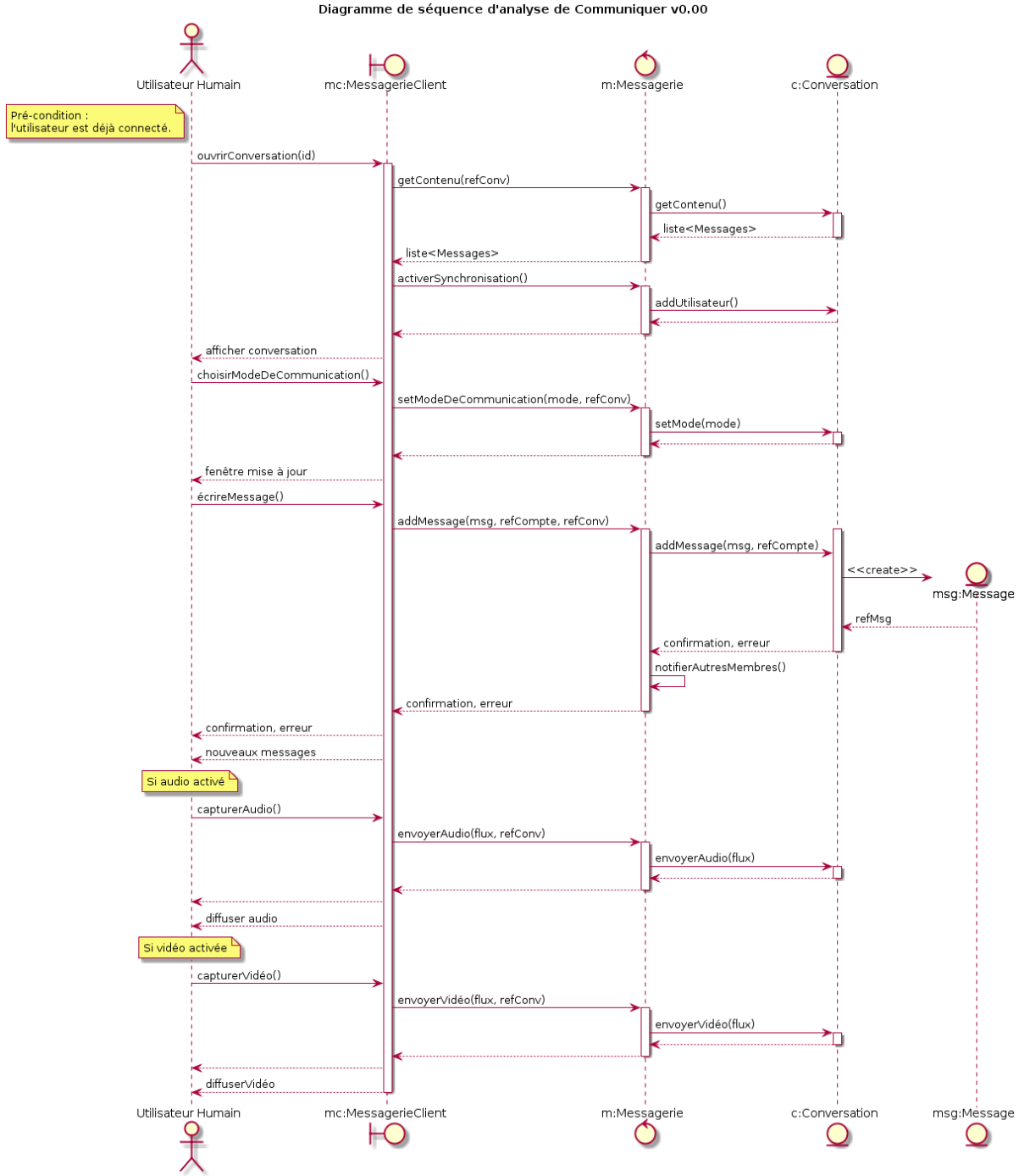
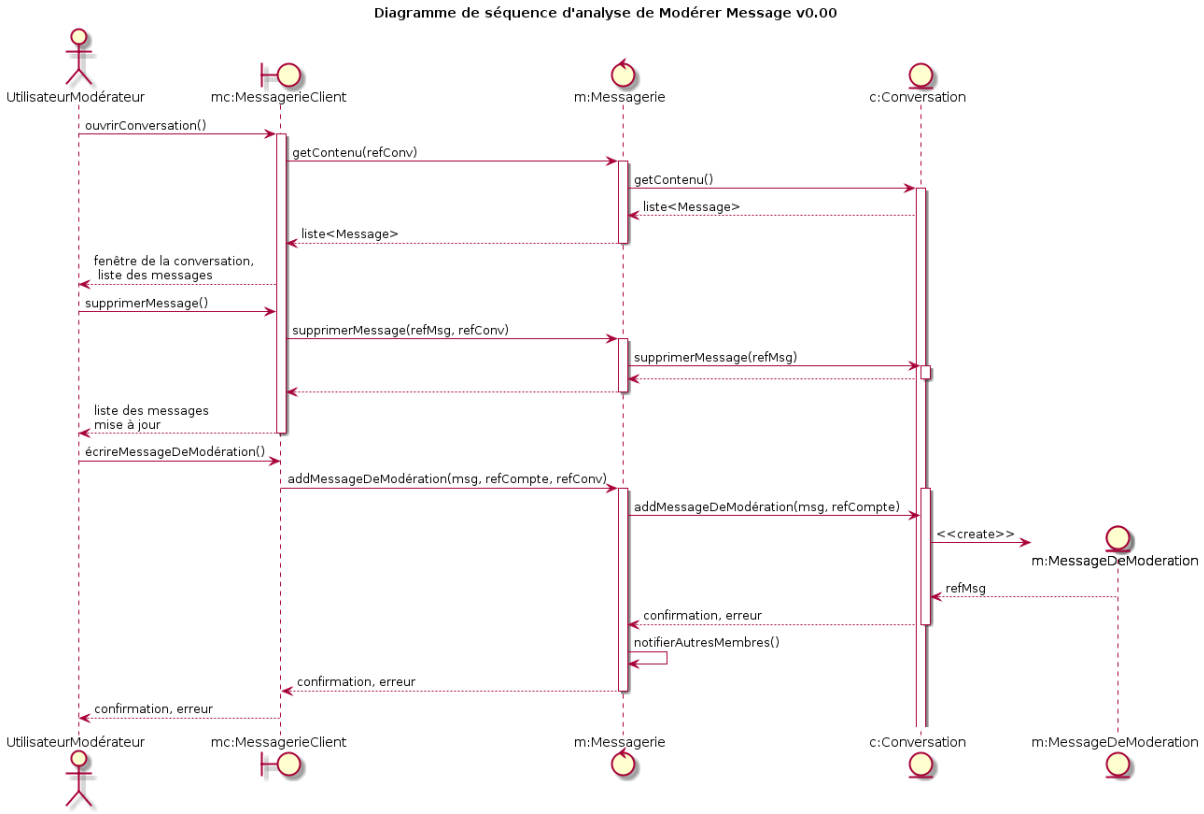
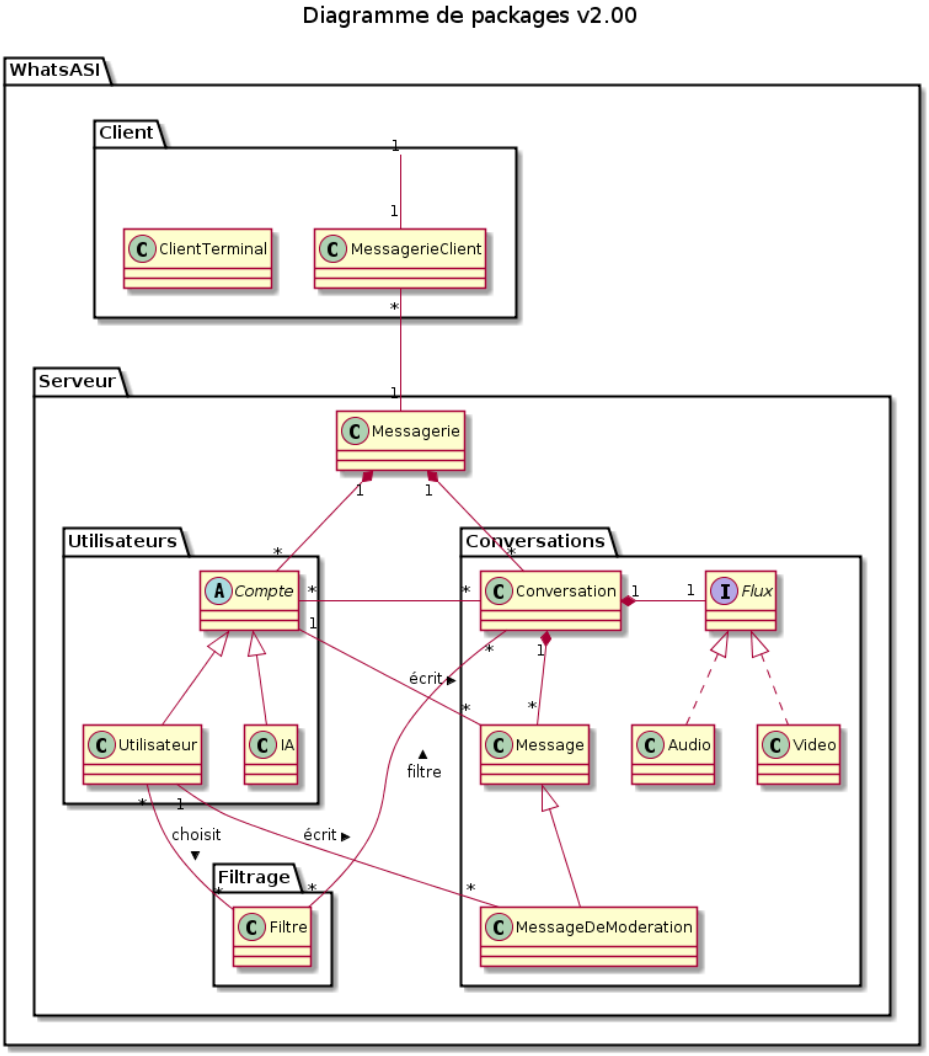


FIGURE 3.11 – Diagramme de séquence d'analyse de **modérer un message**



3.5 Découpage de l'application en *packages*

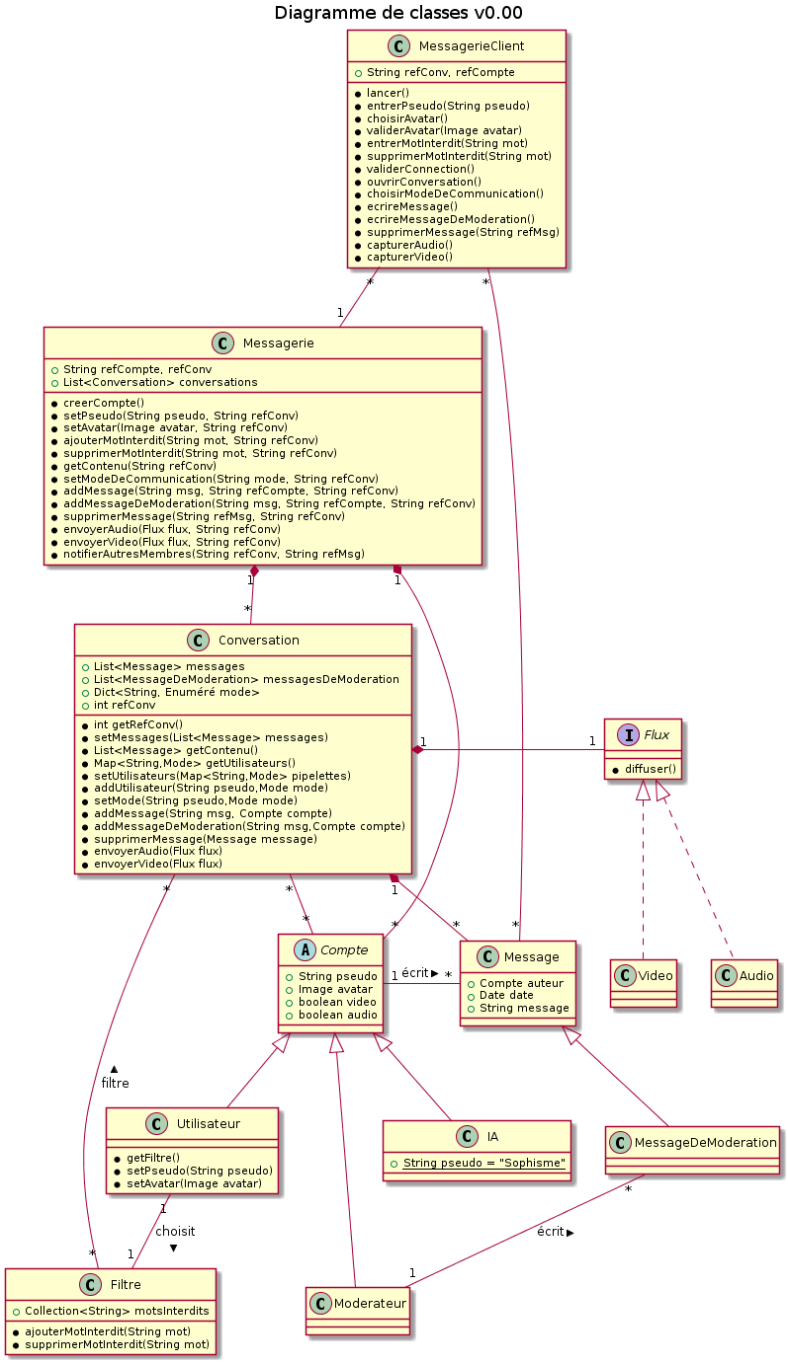
FIGURE 3.12 – Diagramme de packages



Chapitre 4

Conception détaillée

FIGURE 4.1 – Diagramme de classes de conception détaillée



Chapitre 5

Implémentations et tests

Cette partie précise et justifie nos choix techniques et restitue les résultats des tests de validation.

5.1 Choix techniques

5.1.1 Langage de programmation

Nous avons réaliser toute l'application en Java car ce langage est connu de toute l'équipe et permettait une implémentation rapide des technologies d'informatique répartie apprises avant le projet.

5.1.2 Bibliothèques

Le client graphique est une application JavaFX. L'ensemble de l'équipe avait été initiée à Swing en ASI3 mais JavaFX est la bibliothèque graphique officielle de Java depuis la sortie de la Java SE 8 en 2014. Se tenir à jour sur les dernières technologies justifiait ce choix.

5.1.3 Technologie répartie

Nous avons choisi la technologie d'informatique répartie RMI, plutôt que REST par exemple, car nous avons réalisé un logiciel non Web.

5.1.4 Guide d'utilisation

Installation/Exécution

1. Compilez le projet via le fichier `compile.sh`.

```
./compile.sh
```

2. Lancez le RMIRegistry.

```
./launchRmiregistry.sh
```

3. Lancez le serveur (dans un autre terminal).

```
./launchServer.sh
```

4. Lancez le client graphique (dans un autre terminal).

```
./launchClient.sh
```

5. ... ou en ligne de commande.

```
./launchClient.sh terminal
```

Utilisation L'IHM est intuitive.

1. Choisissez un avatar et un pseudo sur le panneau de connexion et cliquez sur Se connecter.
2. Un panneau s'ouvre. Saisissez des mots à filtrer dans les conversations.
3. Le panneau des conversations s'ouvre. Créez une nouvelle conversation en cliquant sur + et cliquez sur un titre de conversation dans la barre latérale. Vous pouvez maintenant envoyer des messages comme dans n'importe d'autre application standards.
4. Les informations détaillées sont fournies par l'IA d'aide, invocable en écrivant `\bonjour` dans n'importe quelle conversation.

Exemples d'utilisation La figure 5.1 illustre la connexion d'un utilisateur modérateur.

La figure 5.2 illustre le paramétrage des filtres de ce même utilisateur.

La figure 5.3 illustre une conversation entre deux utilisateurs dont celui de gauche, A. Pauchet, a filtré le mot « 20 ». On constate sur la fenêtre de droite, du modérateur, que celui-ci peut supprimer un message.

FIGURE 5.1 – Panneau de connexion

FIGURE 5.2 – Panneau de filtrage

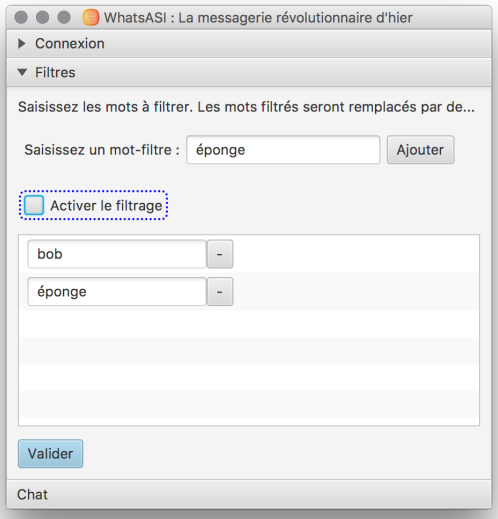
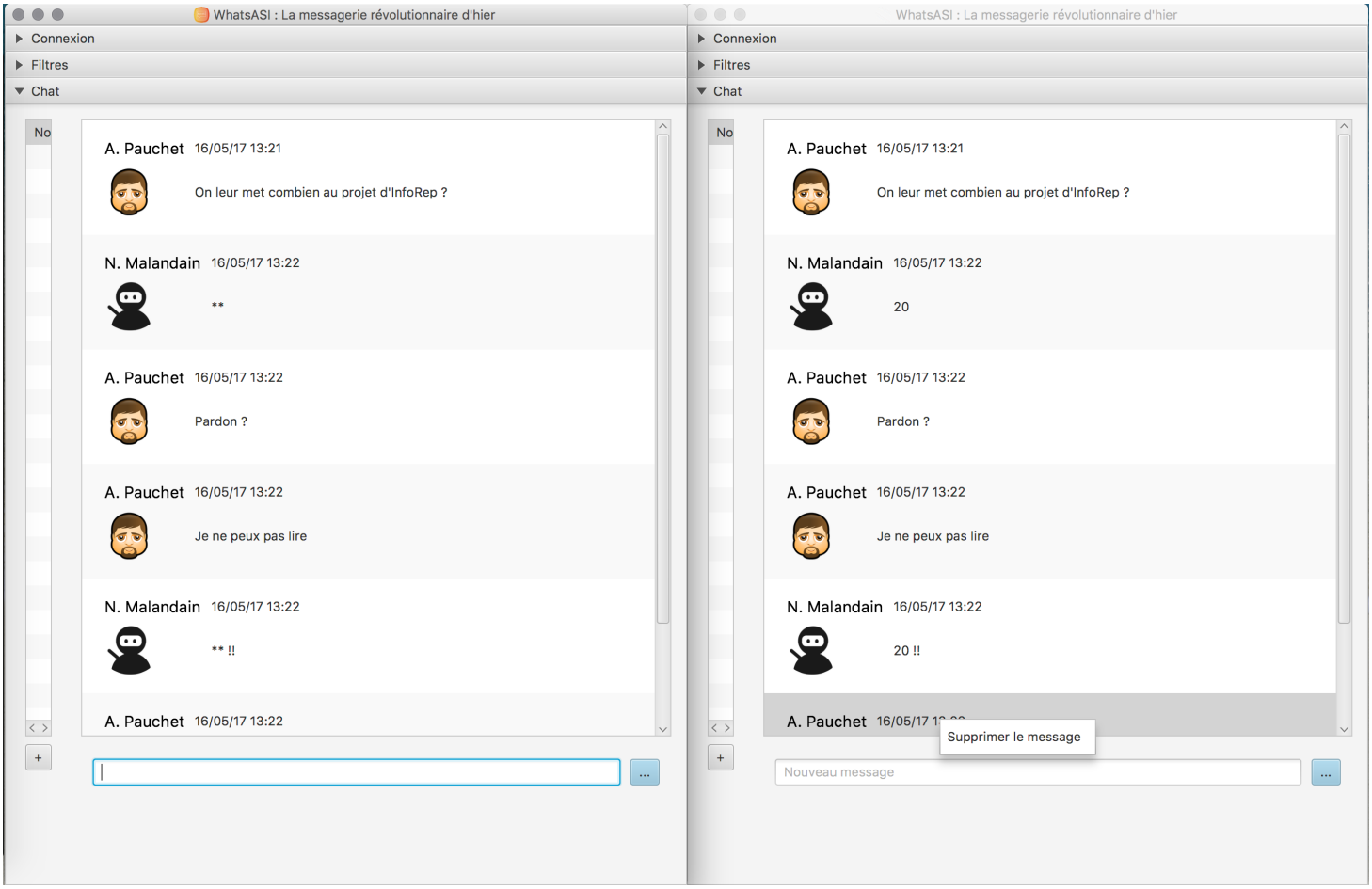


FIGURE 5.3 – Panneau de conversation



5.2 Tests de validation

5.2.1 Tests des spécifications fonctionnelles

1. Les comptes utilisateurs ne persistent pas au-delà de la durée de vie d'un client de messagerie : OK
2. L'utilisateur peut paramétrer son compte, choisir un pseudo et un avatar : OK
3. Tout utilisateur peut rejoindre une conversation : OK
4. Les conversations persistent même quand aucun utilisateur n'y est actuellement connecté : OK
5. L'IA répond à toutes les commandes prévues : OK
6. Les modérateurs peuvent supprimer des messages et la suppression est propagée à la conversation et tous les utilisateurs : OK
7. Les mots à filtrer sélectionnés par un utilisateur sont filtrés dans toutes les conversations pour cet utilisateur seulement : OK

5.2.2 Tests des spécifications d'interface

1. Il existe un panneau de connexion : OK
2. Il existe un panneau de filtrage : OK
3. Il existe un panneau de conversation où chaque message est affiché avec sa date, le pseudo et l'avatar de son expéditeur : OK
4. Une vue étendue s'affiche pour les conversations vidéos : KO (pas implémenté)
5. Les conversations audios sont supportées : KO

5.2.3 Tests des spécifications opérationnelles

1. Le système de messagerie répond instantanément : OK
2. Les messages sont sauvegardés tant qu'il subsiste un utilisateur sur une conversation : OK (et plus longtemps encore)
3. Le service client est toujours accessible tant qu'est activé le serveur : OK
4. Chaque compte possède une référence propre : OK

Chapitre 6

Conclusion

6.1 Développement

6.1.1 Ce que nous avons développé

1. Un serveur de messagerie hébergeant des conversations.
2. Un client graphique qui permet de créer un compte, filtrer des messages et participer à des conversations écrites.
3. Un client terminal.
4. Un helpbot.

6.1.2 Ce que nous avons abandonné

La gestion de flux audio et flux vidéo. Solution : utiliser un plugin permettant de gérer les flux audios et vidéos comme n'importe quel autre flux entre serveur et clients.

6.2 Perspectives

— Gérer les flux audios/vidéos.
— Déployer un client Web.
— Sécuriser le passage des références de compte au serveur avec un chiffrement asymétrique.

6.3 Ce que nous avons appris

— Développer une application RMI.
— Quelques bases de JavaFX, notamment le fait que la mise à jour de ses éléments graphiques est asynchrone et difficelement forçable.
— Nous avons été confronté à un problème de sérialisation avec les images. Seule la classe `IconImage` de Java est sérialisable, mais la taille d'une image ne peut être sauvegardée, cela a posé des problèmes d'affichage que nous avons résolus en choisissant une taille d'image fixe.