

INSTITUT NATIONAL DES SCIENCES APPLIQUÉES DE ROUEN

Département ASI

Architecture des Systèmes d'Information

EC INFORMATIQUE RÉPARTIE

Rapport de Projet

Projet

Messagerie instantanée

Auteurs

Gautier DARCHEN
Alexandre HUAT
Marie-Andrée JOLIBOIS
Romain JUDIC
Alexandre LE LAIN

Version

v0.00

15 mai 2017

Table des matières

1	Introduction	2
1.1	Fonctions principales	2
1.2	Utilisateurs	2
1.3	Contraintes	2
1.3.1	Contraintes matérielles	2
1.3.2	Contraintes logicielles	3
2	Spécifications	4
2.1	Spécifications fonctionnelles	4
2.2	Spécifications d'interfaces	7
2.3	Spécifications opérationnelles	9
3	Conception préliminaire	10
3.1	Modèle du domaine	10
3.2	Diagrammes de séquence système	11
3.2.1	Modification du modèle du domaine	13
3.3	Diagrammes d'activités	14
3.4	Diagrammes d'interaction	17
3.5	Diagramme de communication client serveur	20
3.6	Découpage de l'application en <i>packages</i>	20
4	Conception détaillée	21
5	Implémentations et tests	23
5.1	Choix techniques	23
5.1.1	Langage de programmation	23
5.1.2	Bibliothèques	23
5.1.3	Technologie répartie	23
5.1.4	Guide d'utilisation	23
5.2	Tests de validation	24
5.2.1	Tests des spécifications fonctionnelles	24
5.2.2	Tests des spécifications d'interface	24
5.2.3	Tests des spécifications opérationnelles	24
6	Conclusion	25
6.1	Développement	25
6.2	Perspectives	25

Chapitre 1

Introduction

L'application à développer est une plateforme de messagerie instantanée entre deux interlocuteurs. Elle permettra à ces interlocuteurs de communiquer tout en étant connectés sur des machines distantes.

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.

Chapitre 2

Spécifications

2.1 Spécifications fonctionnelles

Les comptes utilisateurs ne sont pas persistants, ils n'existent que de l'ouverture 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 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 un salon de chat.

`\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

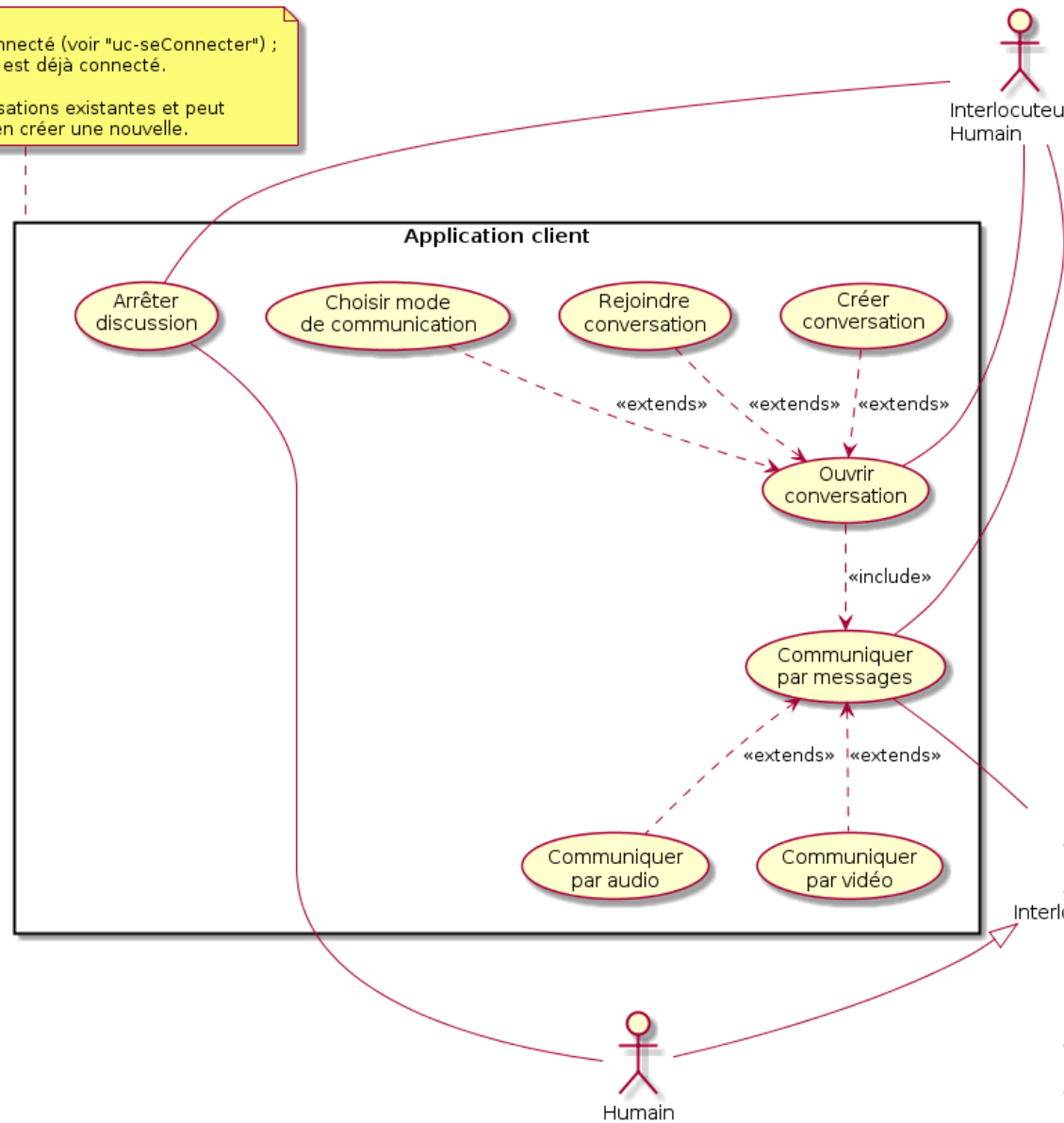
Cette section présente les cas d'utilisation suivants :

- discuter ;
- paramétrer le filtrage ;
- filtrer les messages ;
- se connecter.

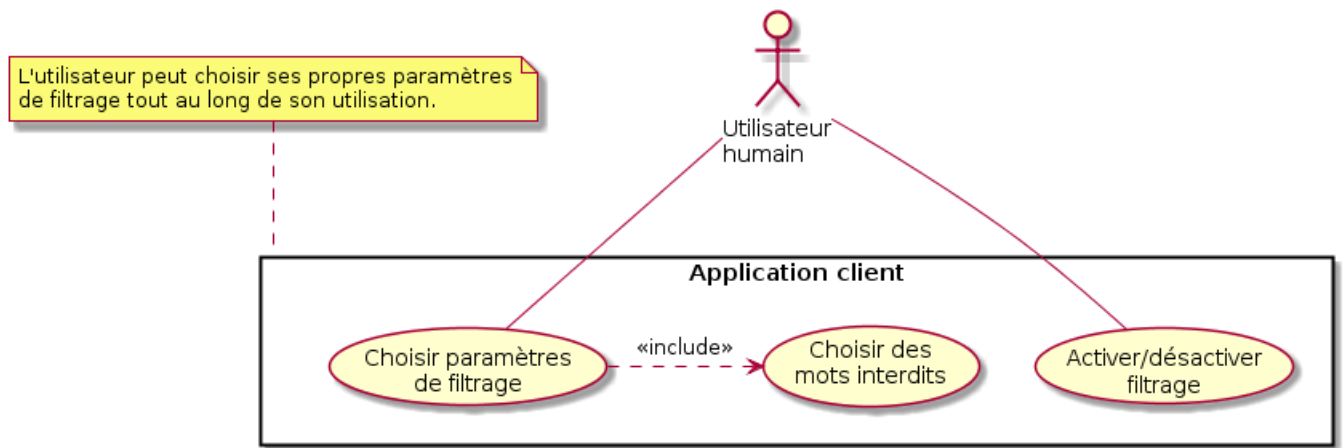
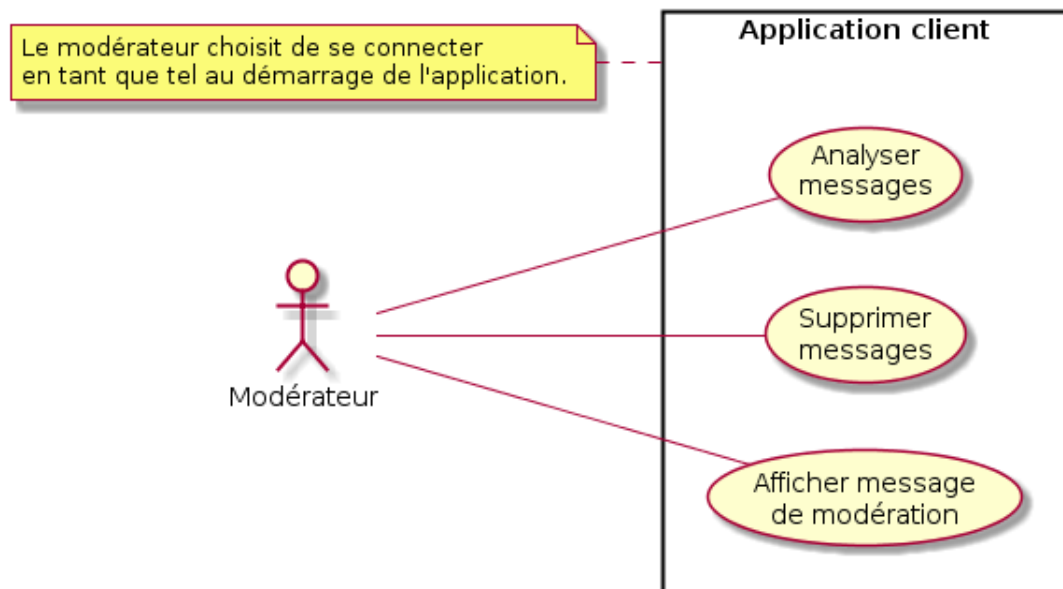
Pré-conditions :

- l'utilisateur 1 est déjà connecté (voir "uc-seConnecter") ;
- l'utilisateur 2 (si humain) est déjà connecté.

L'utilisateur voit les conversations existantes et peut soit en rejoindre une, soit en créer une nouvelle.

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**

Ce cas d'utilisation est le premier que rencontre l'utilisateur lorsqu'il ouvre l'application.

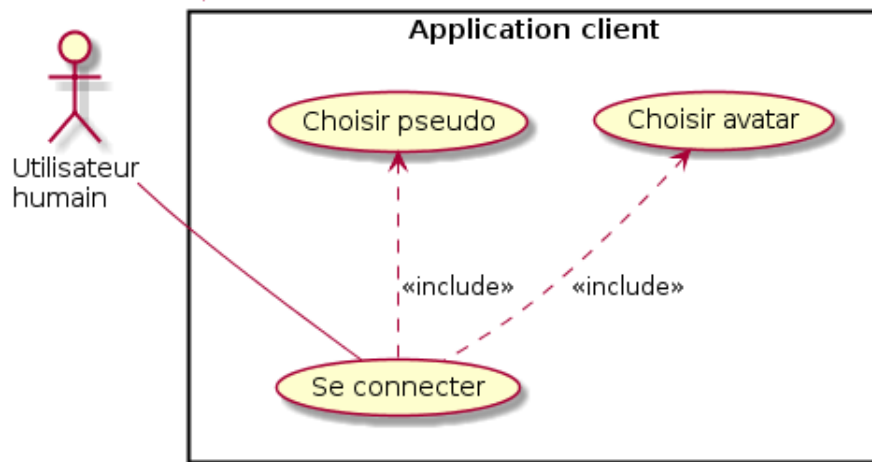


FIGURE 2.4 – Cas d'utilisation **se connecter**

2.2 Spécifications d'interfaces

Maquettes

Cette section présente les maquettes de l'application.

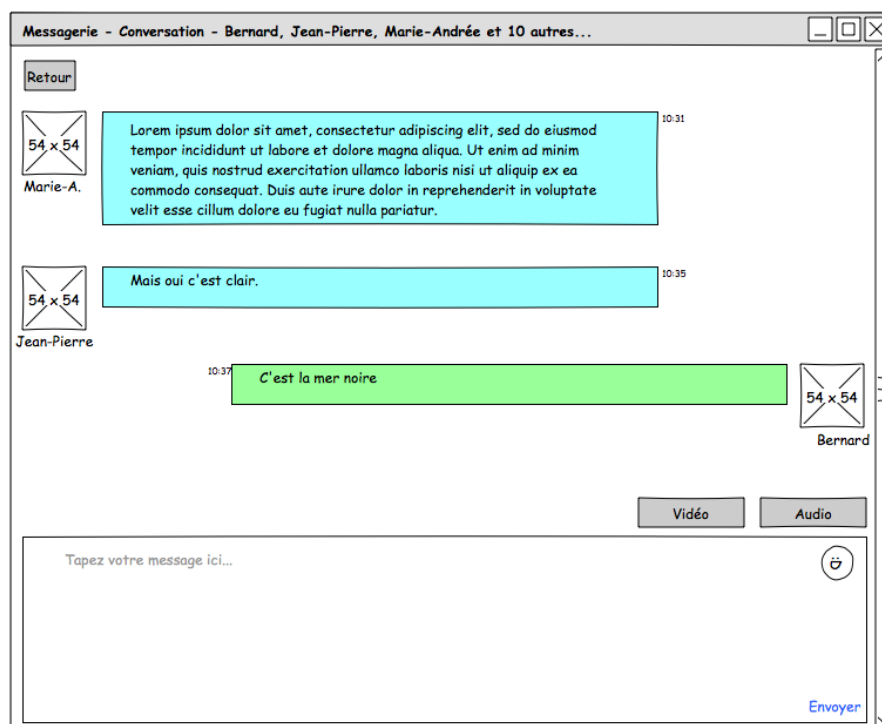


FIGURE 2.5 – Un exemple de conversation texte

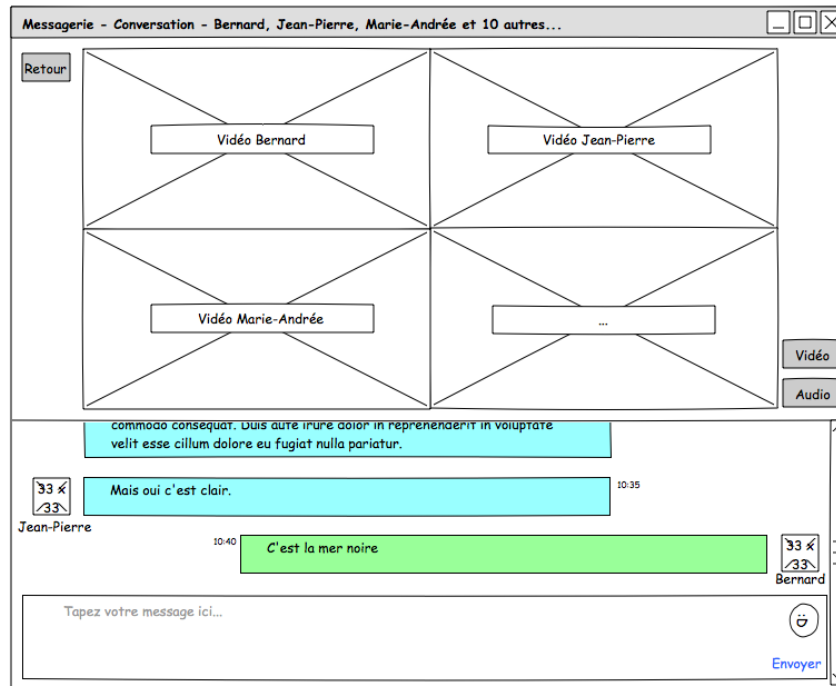


FIGURE 2.6 – Conversation vidéo

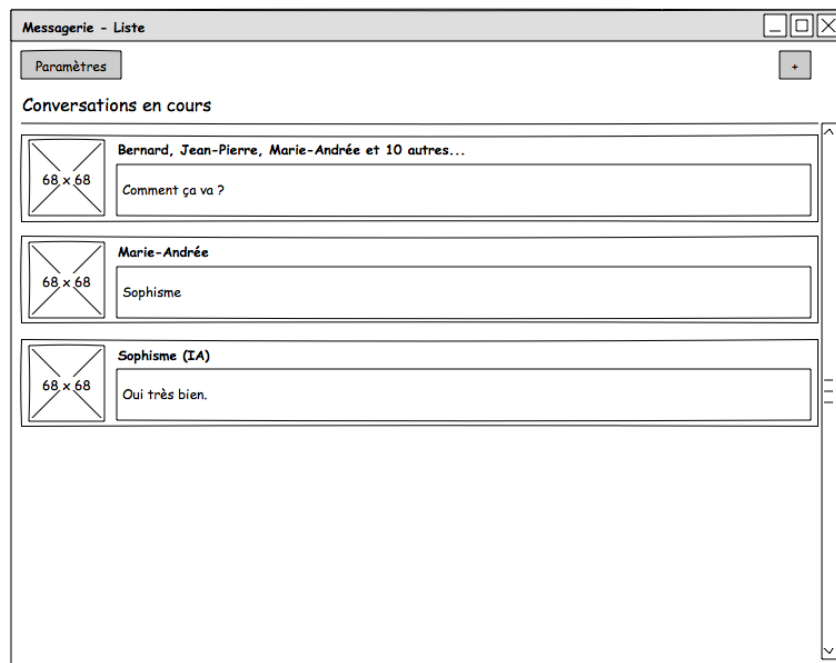
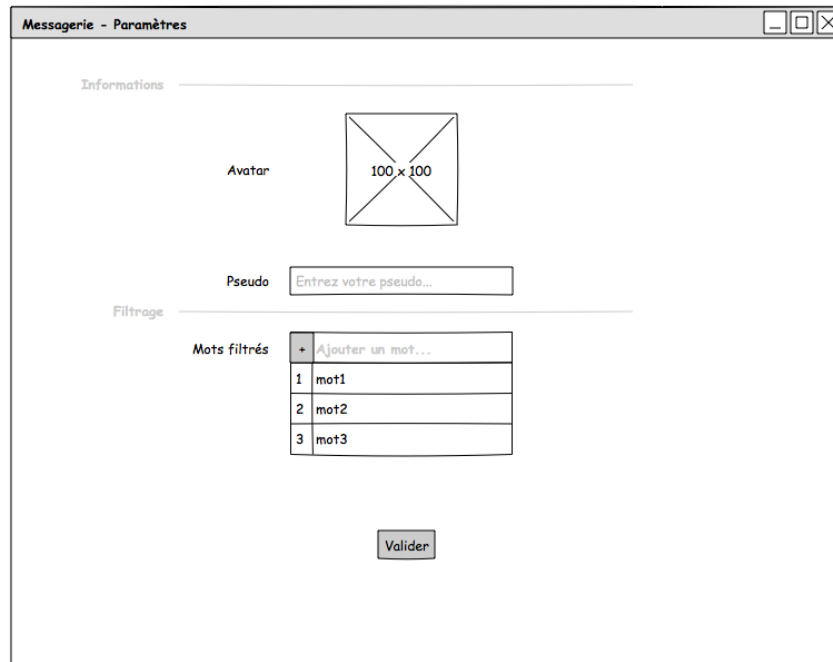
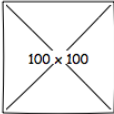


FIGURE 2.7 – Liste des conversations



Messagerie - Paramètres

Informations

Avatar 

Pseudo

Filtrage

Mots filtrés

+	Ajouter un mot...
1	mot1
2	mot2
3	mot3

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.

Le serveur qui héberge l'application est disponible à tout instant t pour permettre aux utilisateurs d'utiliser le service à n'importe quel moment.

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

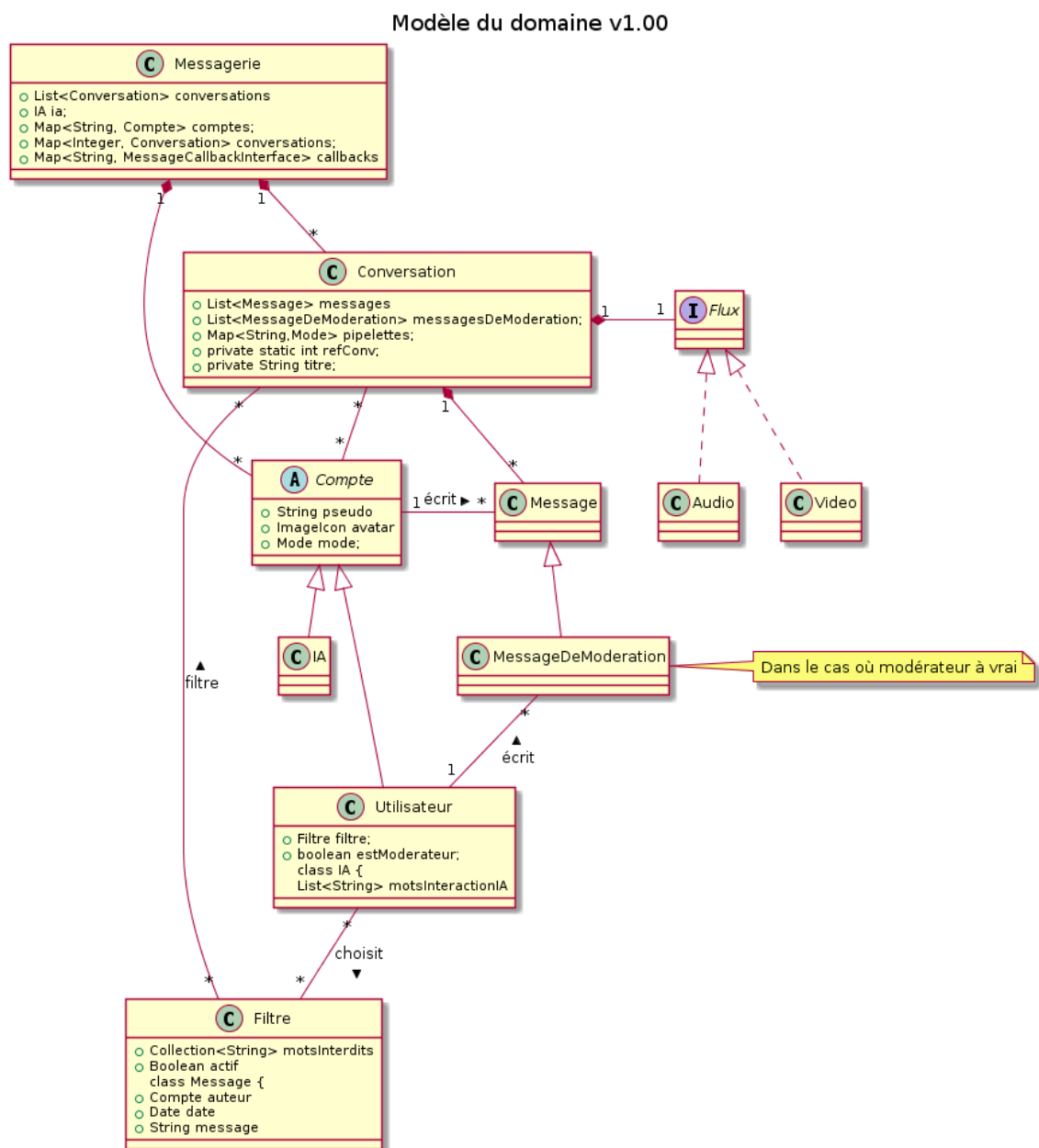


FIGURE 3.1 – Modèle du domaine

3.2 Diagrammes de séquence système

Diagramme de séquence système de Communiquer v0.00



FIGURE 3.2 – Le diagramme de séquence système de « communiquer »

Diagramme de séquence système de Se connecter v0.00

FIGURE 3.3 – Le diagramme de séquence système de « se connecter »

Diagramme de séquence système de Modérer v0.00

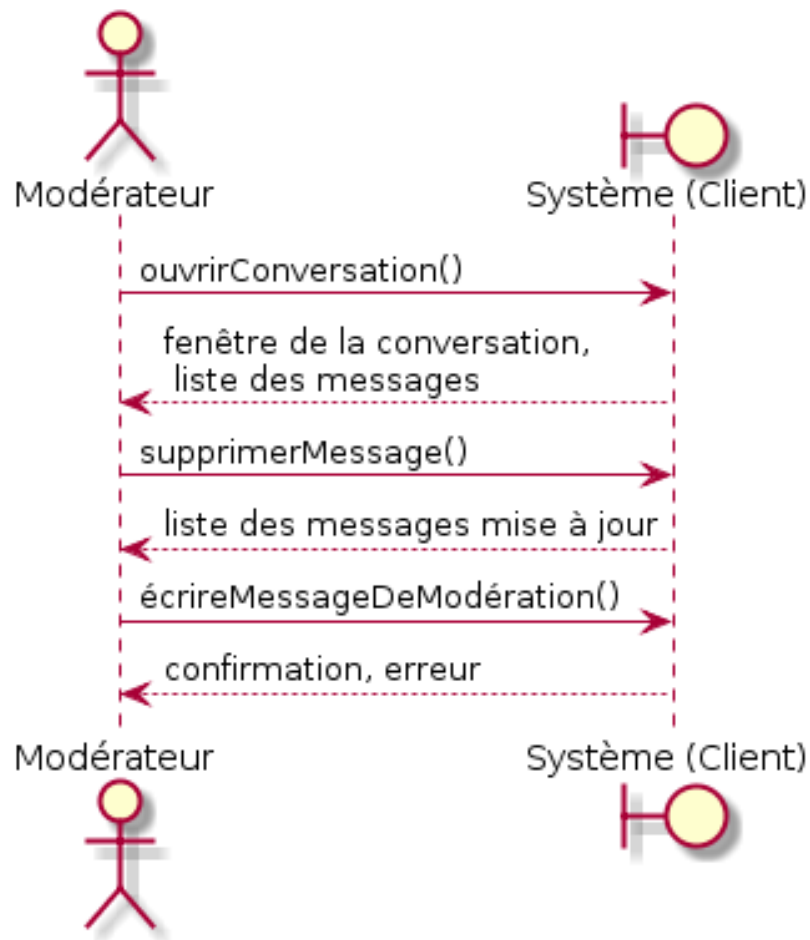


FIGURE 3.4 – Le diagramme de séquence système de « modérer »

3.2.1 Modification du modèle du domaine

La logique utilisée lors de la conception est restée la même. Nous avons cependant effectuée 2 changements principaux. Nous avons inclus beaucoup plus de champs dans la classe Messagerie. En effet nous avons tous centraliser via cette classe pour structurer les encapsulations. Par exemple, cette classe contient l'IA.

De plus, nous avons initialement une classe modérateur que nous avons supprimé et transformé en un booléen dans la classe utilisateur. Choix qui nous paraissait beaucoup plus pertinent.

3.3 Diagrammes d'activités

Diagramme de l'activité Se connecter v0.00

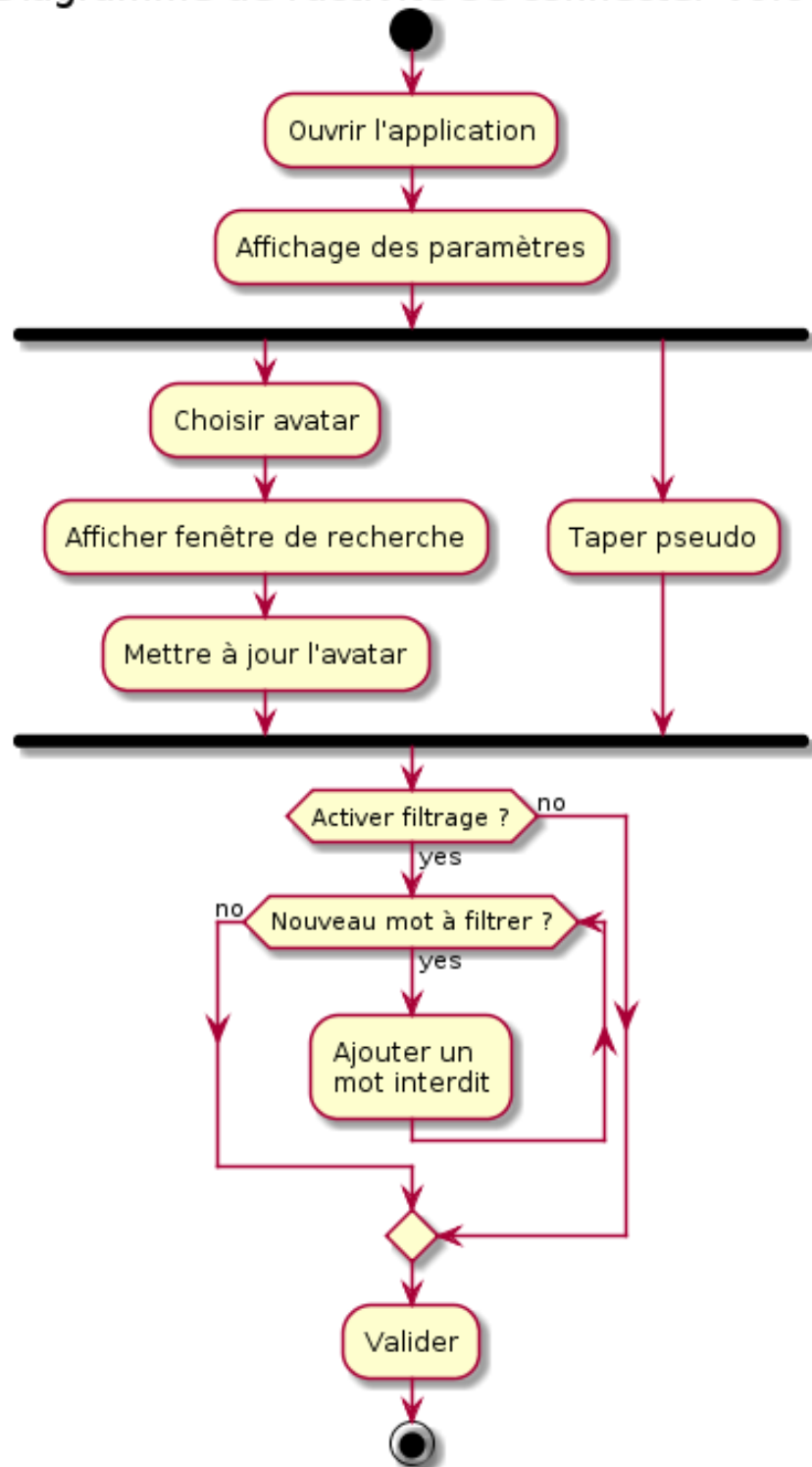


FIGURE 3.5 – Le diagramme de l'activité « se connecter »

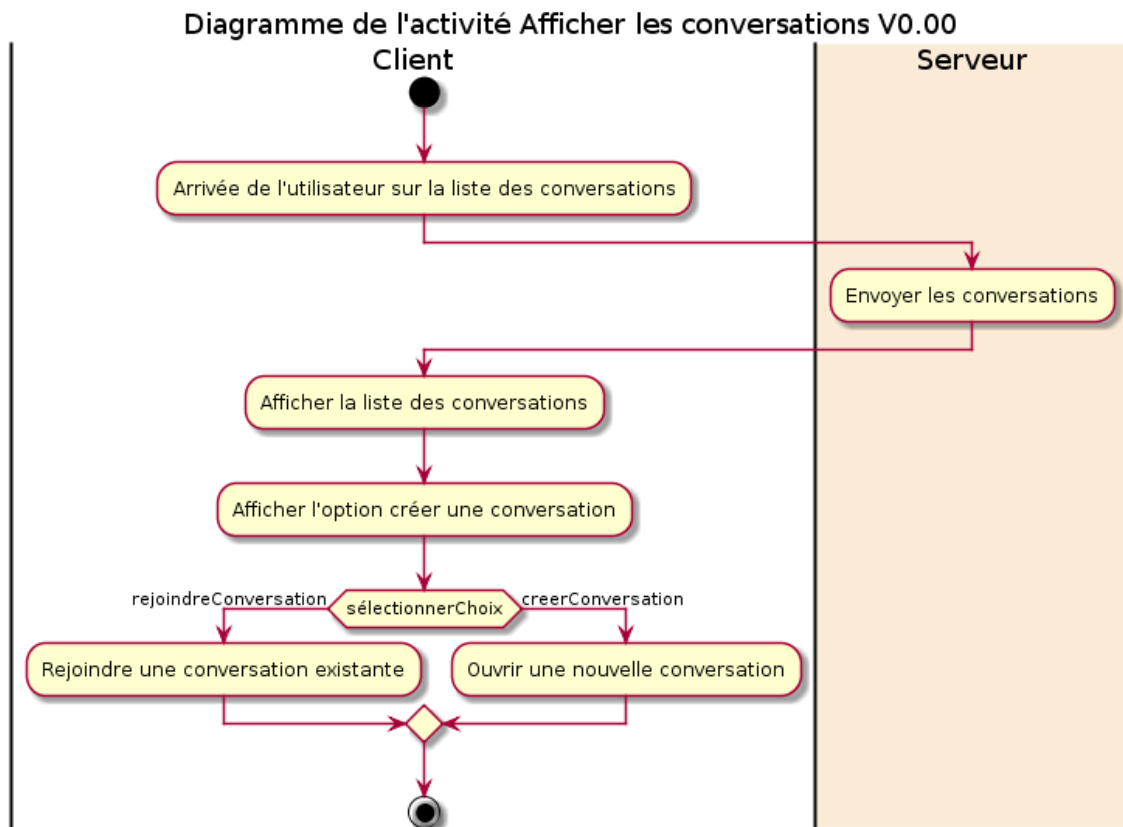


FIGURE 3.6 – Le diagramme de l'activité « choisir une conversation »

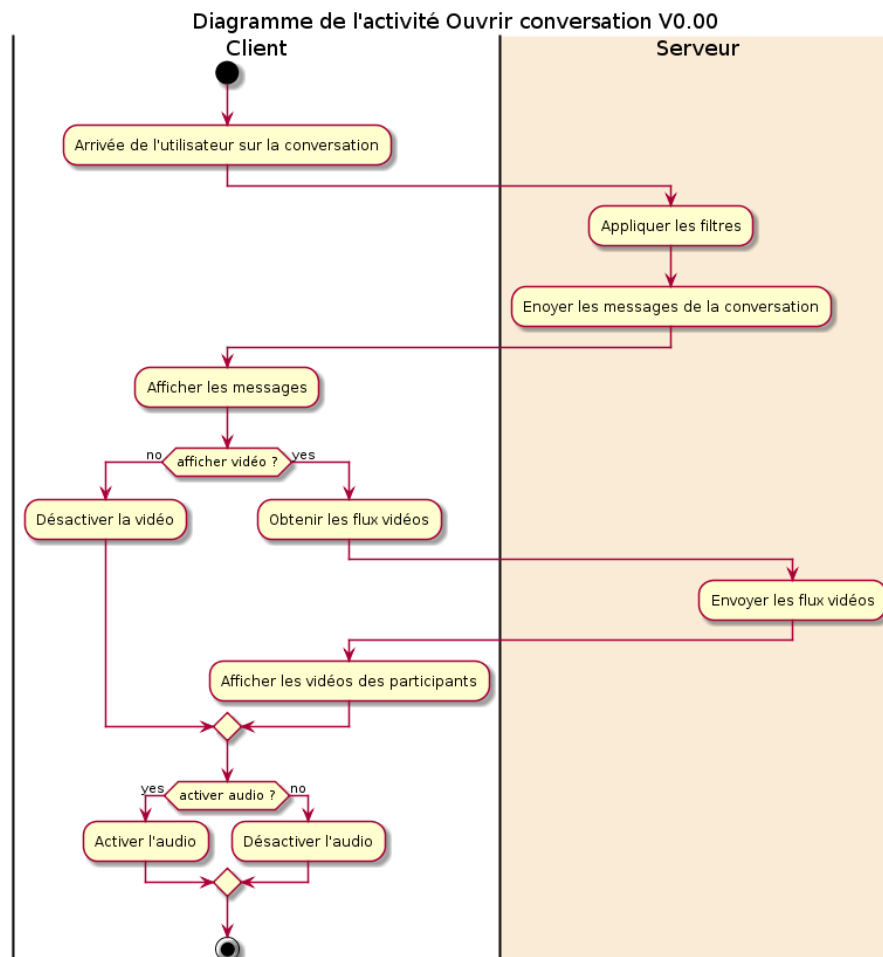


FIGURE 3.7 – Le diagramme de l'activité « ouvrir une conversation »

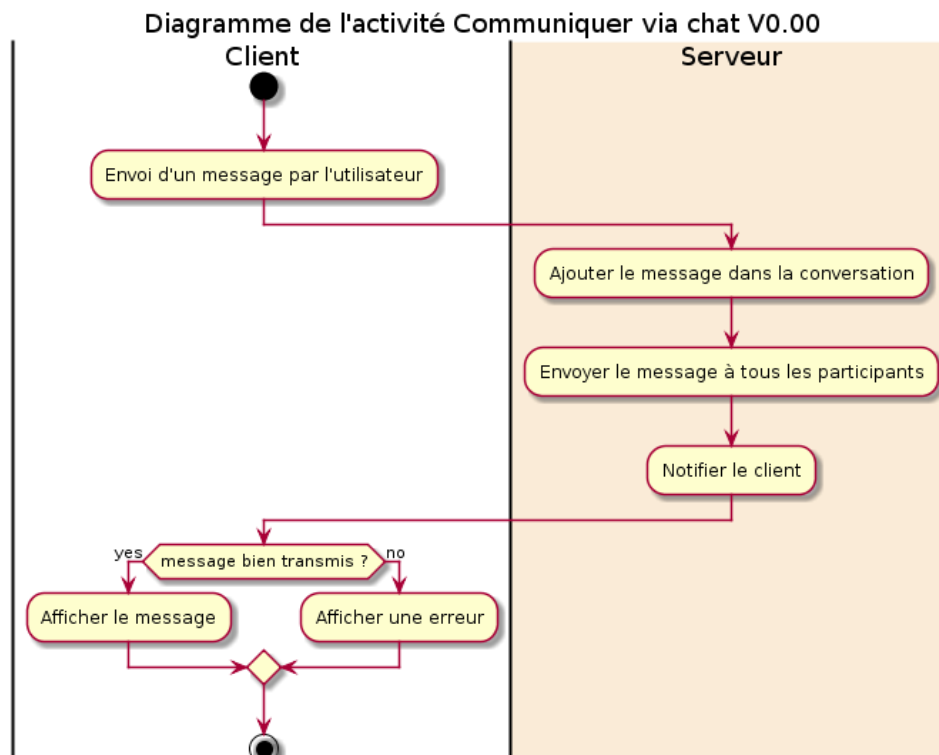


FIGURE 3.8 – Le diagramme de l'activité « communiquer »

3.4 Diagrammes d'interaction

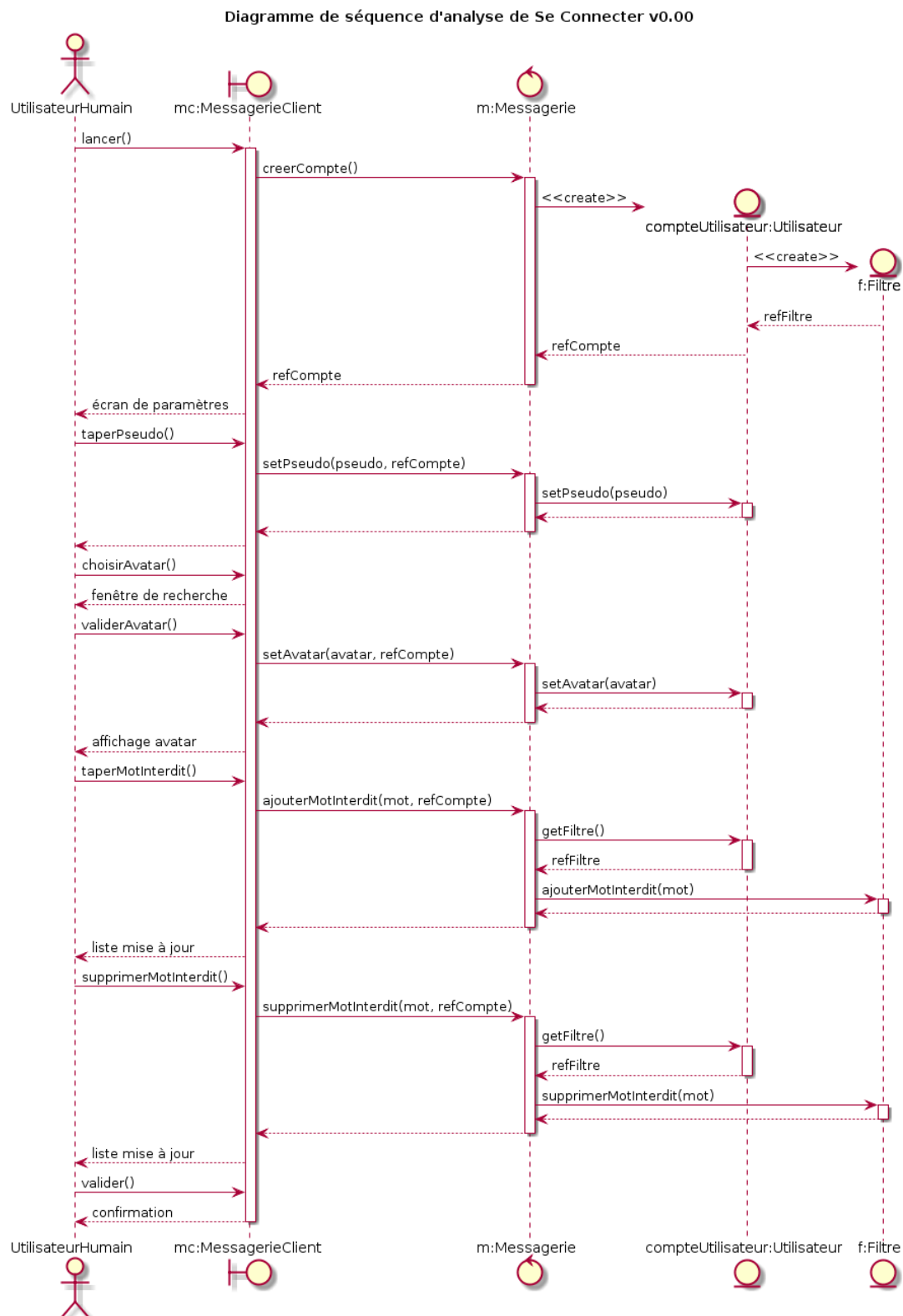


FIGURE 3.9 – Le diagramme de séquence d'analyse « se connecter »

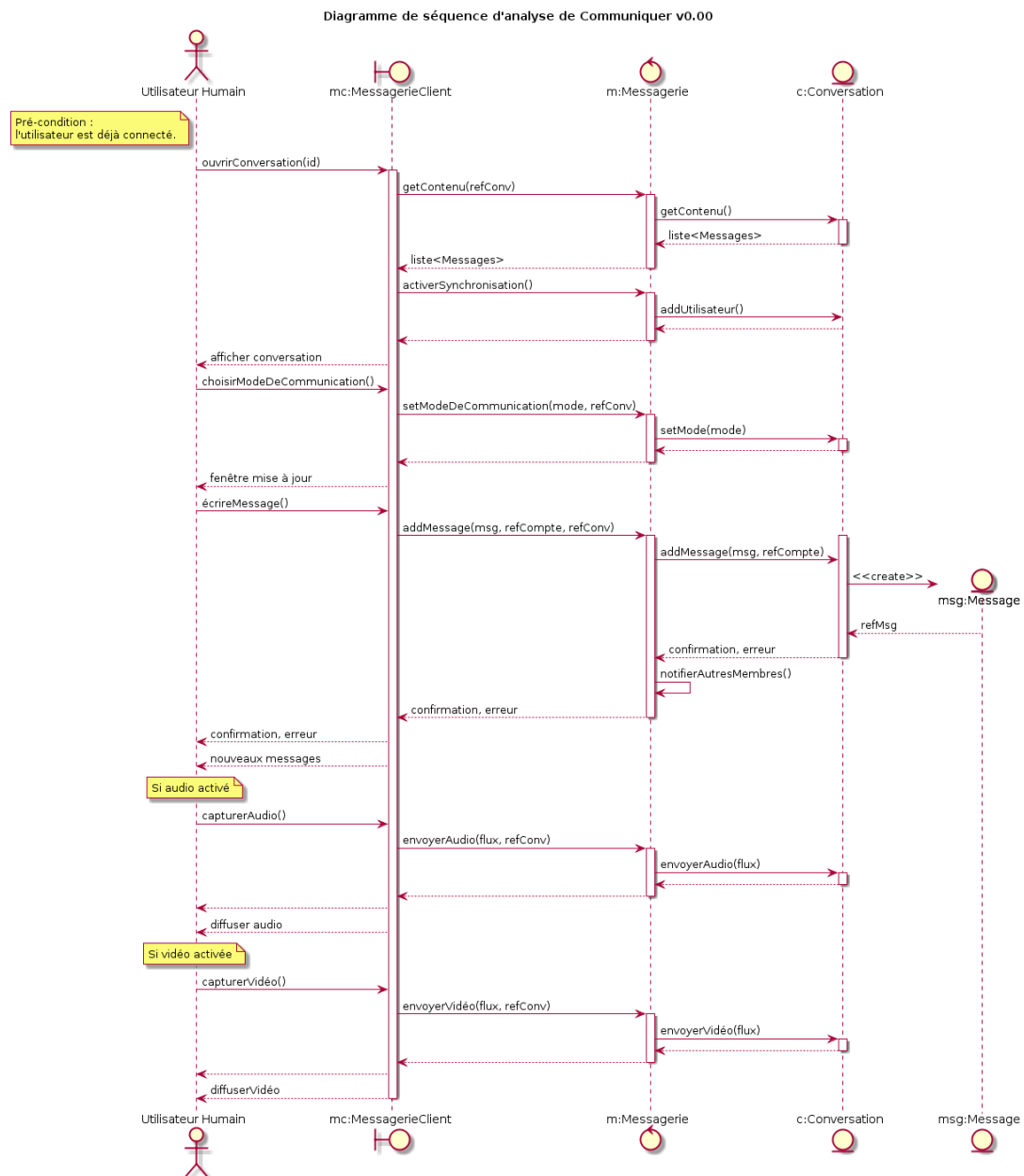


FIGURE 3.10 – Le diagramme de séquence d'analyse « Communiquer »

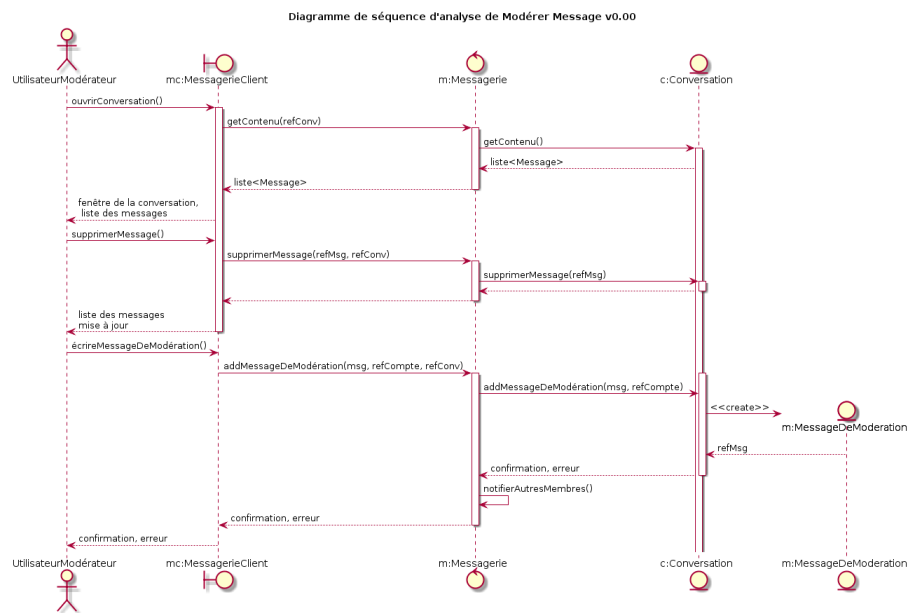


FIGURE 3.11 – Le diagramme de séquence d'analyse « Modérer Message »

3.5 Diagramme de communication client serveur

3.6 Découpage de l'application en *packages*

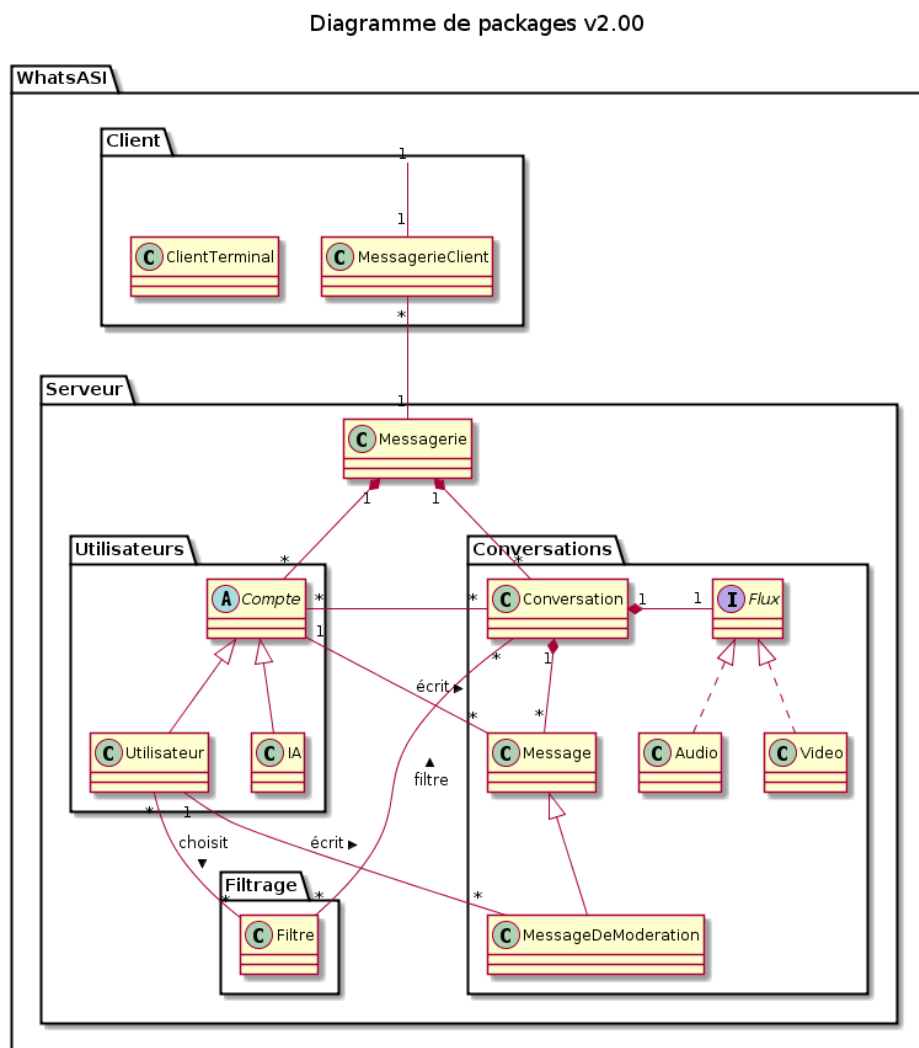


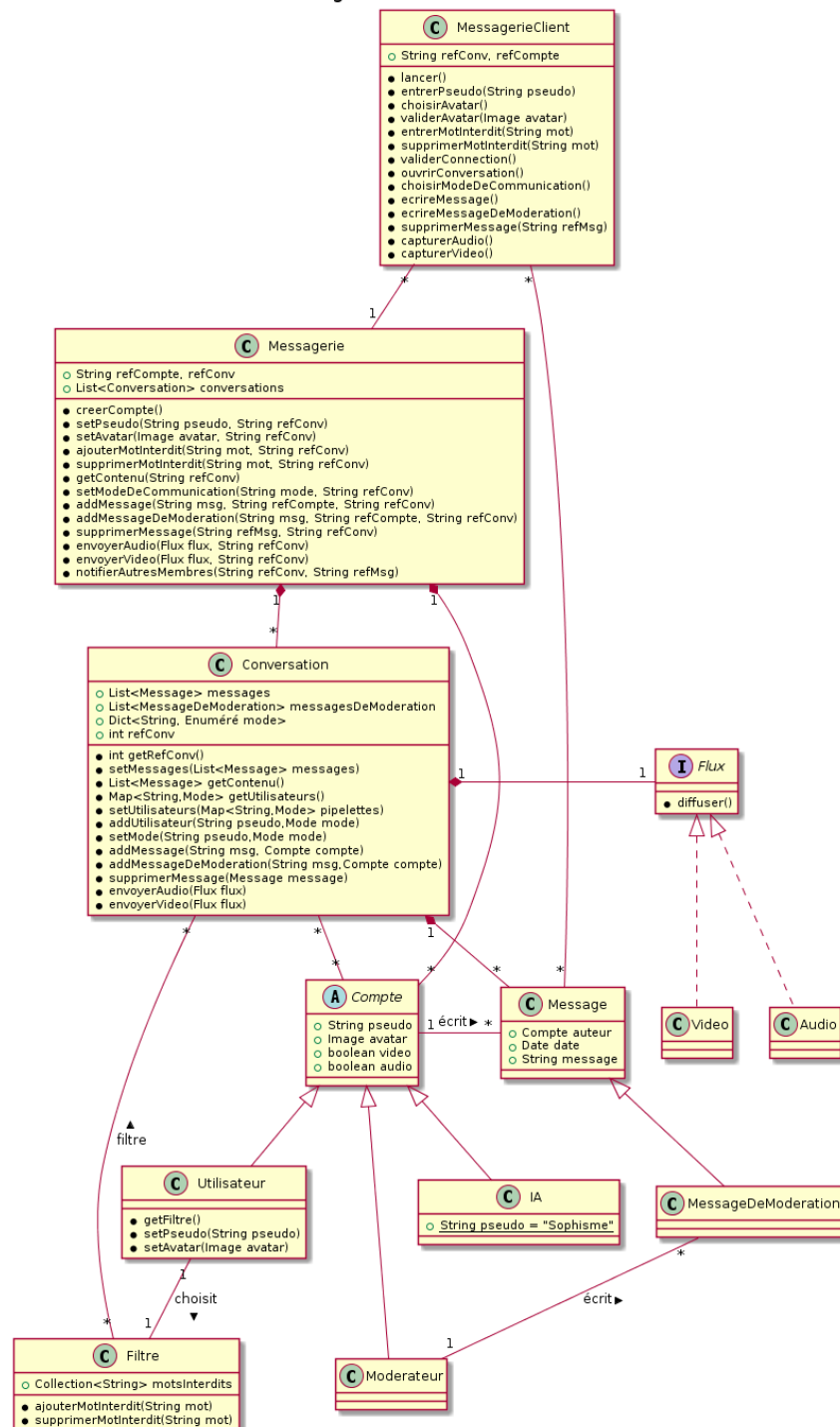
FIGURE 3.12 – Le diagramme de packages

Nous avons choisi de travailler avec des threads et callback. Comme indiqué sur la figure ci dessous. Nous sommes en multiciel : il y a donc un callback par client.

Chapitre 4

Conception détaillée

Diagramme de classes v0.00



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

L'application est intégralement réalisée en Java. En effet, Java 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 justifie donc 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 indépendant du monde du Web. De plus, RMI permet de gérer de multiples clients (multi-threading).

5.1.4 Guide d'utilisation

Installation/Exécution

1. Compilez le projet via le fichier `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.

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 connectée : 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 affichée 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)

5.2.3 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 serveur hébergeant les conversations est disponible en tout temps : 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.