JMessenger - Rapport de projet Java

BECART Quentin - GOURMELON Gwendal

ISEN AP4 - Mars 2019

1 Introduction

Nous avons au cours de cette période école et dans le cadre du module de programmation Java, été amenés à réaliser un programme de messagerie instantanée.

https://github.com/MyGg29/JIRC

2 Rappel du cahier des charges

- Le programme devra permettre à tous les utilisateurs d'envoyer et de recevoir des messages sur un salon ou à des utilisateurs spéciqués identiés (1 ou plusieurs).
- Le programme devra pouvoir permettre de créer/modier des salons et d'y inviter des utilisateurs
- Chaque utilisateur possèdera ou non les droits de modication d'un salon. Droits qu'il pourra répliquer ou non aux autres utilisateurs du salon.
- Le programme doit disposer d'une interface graphique claire pour la lecture de l'historique des salons, la liste des discussions en cours, les utilisateurs connectés sur chaque salon.
- Le programme devra permettre, pour les administrateurs d'un salon, d'exporter l'historique de discussion d'un salon via un chier XML.
- Chaque utilisateur peut, à tout moment, se désinscrire d'un salon.
- La déconnexion du salon ne doit pas entraîner la perte de l'historique des conversations.

- Le programme devra fournir des statistiques sur l'activité des utilisateurs et des diérents salons.

3 Structure du programme

3.1 MVC

Dans le cadre de ce projet, nous avons utilisé l'architecture logicielle MVC (pour Modèle-Vue-Contrôleur) se présentant de la manière suivante :

- Le Modèle contient l'ensemble des données à afficher
- La Vue constitue l'interface graphique (elle est l'élément d'interaction avec l'utilisateur)
- Le Contrôleur représente le lien entre la vue et le modèle et définit les actions qui seront effectuées lors d'un évènement (interaction de l'utilisateur)

Le modèle et le contrôleur sont tous deux des fichiers .java, la vue est quant à elle un fichier .fxml.

3.2 Packages

Nous avons actuellement 4 packages :

- Client : côté client
- Database : partie base de données
- Models : modèles, autrement dit les classes génériques d'informations utilisées
- Server : côté serveur
- Util : partie utilitaire (fonctions génériques, ...)

3.3 Classes importantes

- client.Main.java : le point d'entrée du Client.
- serveur.Server.java : le point d'entrée du serveur. Créer un Thread pour chaque nouvelle connexion

- client.controllers.MainController.java : le contrôleur de la fenêtre principale de chat
- models.Client.java : La logique client. S'occupe du transfert de données entre le client et le serveur
- models.ClientListener.java : Thread d'écoute du client. Permettant de parler et d'écouter le serveur en même temps.
- serveur.SSocket.java : La logique du serveur.

3.4 Diagramme de classes et diagramme client

Voir pages suivantes. L'image du diagramme représenté est disponible en meilleure résolution sur le repository GitHub du projet.[1]

4 Conclusion

Ce projet nous a permis de mettre en pratique les différents points du langage Java abordés en cours. Nous nous sommes également familiarisés avec le framework FXML et le développement d'interfaces graphiques d'une manière générale. Il aura aussi été interessant de développer deux applications qui communiquent entre elles : le serveur d'un côté et le client de l'autre.

References

[1] Q. BECART and G. GOURMELON. *JMessenger*. Available at `https://github.com/MyGg29/JIRC`.

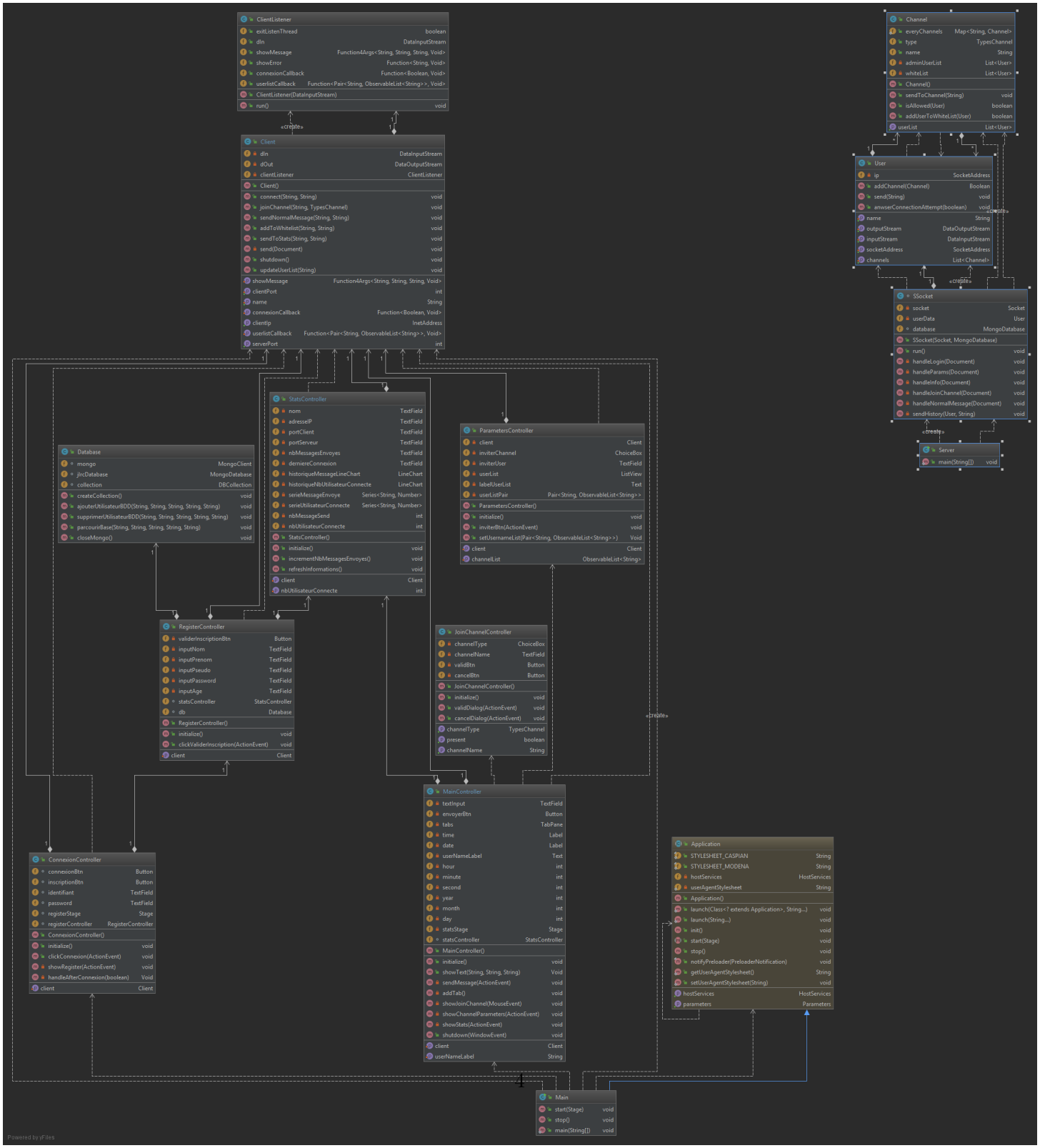

Figure 1: Diagramme de classes

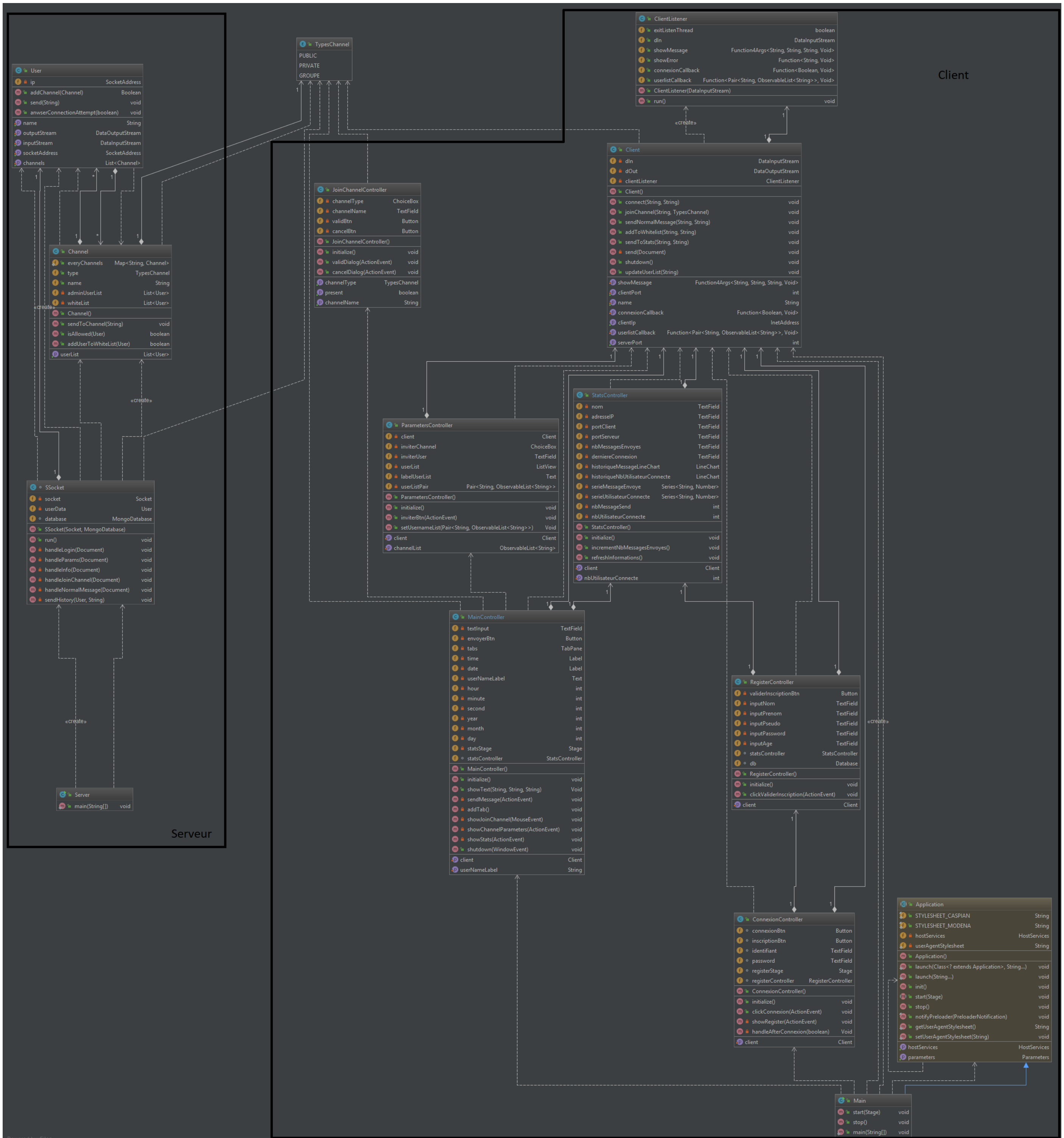

Figure 2: Diagramme Client