Application de communication client-serveur avec une interface graphique JavaFX

Introduction

L'application "Application de communication client-serveur avec une interface graphique JavaFX" est un projet développé en Java qui permet la communication entre clients et serveur via une interface utilisateur conviviale. L'objectif principal de cette application est de fournir un moyen simple et efficace de communication en temps réel entre plusieurs utilisateurs connectés à un serveur central.

L'application se compose de deux parties principales : le serveur et le client. Le serveur est responsable de la gestion des connexions des clients, tandis que les clients peuvent se connecter au serveur pour échanger des messages entre eux.

L'interface graphique utilisateur de l'application est développée à l'aide de JavaFX, ce qui offre une expérience utilisateur intuitive et attrayante. Les clients peuvent saisir leurs messages dans une zone de texte dédiée et les envoyer en appuyant sur un bouton "Envoyer". Les messages envoyés sont ensuite affichés dans une liste visible par tous les utilisateurs connectés.

Le serveur, quant à lui, est en charge d'accepter les connexions entrantes des clients, d'écouter les messages envoyés par les clients et de les redistribuer aux destinataires appropriés. Chaque client est identifié par un numéro de client unique, ce qui permet au serveur de gérer efficacement les communications entre les différents utilisateurs.

En exécutant le serveur de l'application, vous pourrez observer la console affichant les informations relatives aux clients connectés, tels que leur numéro de client et leur adresse IP. Ensuite, en exécutant l'application cliente, vous pourrez vous connecter au serveur, envoyer des messages et voir les messages envoyés par d'autres utilisateurs.

Ce projet démontre les concepts clés de la communication client-serveur, la gestion des connexions réseau, la manipulation des flux d'entrée/sortie et l'utilisation de l'interface graphique JavaFX. Il offre une base solide pour développer des applications de communication en temps réel plus complexes et extensibles.

Dans la suite de ce rapport, nous examinerons en détail l'architecture du projet, les fonctionnalités implémentées, ainsi que les principales étapes de son exécution.

Les étapes pour réaliser le projet

Conception de l'interface utilisateur :

Définition des fenêtres, des éléments graphiques et de leur disposition à l'aide de JavaFX.

Création des fichiers FXML pour décrire la structure de l'interface.

Implémentation du contrôleur de la scène MM1 (SceneMM1Controller) :

Gestion de l'authentification de l'utilisateur en vérifiant les informations fournies.

Chargement de la deuxième scène en cas de succès.

Implémentation du contrôleur de la scène MM2 (SceneMM2Controller) :

Gestion des interactions de l'utilisateur avec l'interface de chat.

Connexion au serveur en spécifiant l'hôte et le port.

Envoi et réception de messages avec les autres utilisateurs.

Implémentation du serveur de chat (ServerChat) :

Écoute des connexions entrantes des clients à l'aide d'un socket.

Gestion des communications entre les clients connectés.

Utilisation de threads pour permettre une communication simultanée avec plusieurs clients.

Implémentation de la classe de communication (Communication) :

Gestion de la communication avec un client spécifique.

Réception, analyse et redistribution des messages aux destinataires appropriés.

Utilisation de flux d'entrée/sortie pour l'échange de données avec le client.

Intégration de l'interface utilisateur avec le serveur :

Transmission des messages reçus du serveur à l'interface utilisateur pour affichage.

Transmission des messages envoyés depuis l'interface utilisateur au serveur pour distribution.

Tests et débogage :

Réalisation de tests approfondis pour s'assurer du bon fonctionnement de l'application.

Détection et correction des éventuels problèmes et bogues rencontrés.

Compilation et exécution :

Compilation de l'ensemble du projet pour obtenir les fichiers exécutables.

Exécution du serveur et des clients pour tester la communication en temps réel.

Documentation et rapport :

Rédaction d'une documentation détaillée sur l'architecture du projet, les fonctionnalités implémentées et les étapes d'exécution.

Préparation d'un rapport expliquant le projet, ses objectifs, son fonctionnement et les résultats obtenus.

MVC

Dans ce projet, nous avons utilisé une approche de conception basée sur le modèle MVC (Modèle-Vue-Contrôleur) pour organiser la structure de l'application. Le modèle MVC est un patron de conception qui permet de séparer la logique métier de la présentation des données et de la gestion des interactions.

Modèle (Model) :

Le modèle représente les données et la logique métier de l'application. Dans notre projet, le modèle est principalement représenté par les classes ServerChat et Communication. La classe ServerChat gère les connexions des clients et la communication entre eux. La classe Communication gère la communication avec un client spécifique.

Vue (View) :

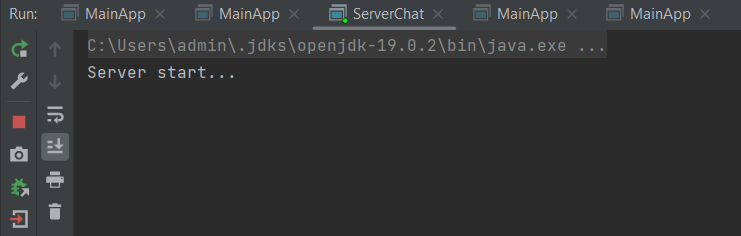
La vue est responsable de l'affichage des données à l'utilisateur et de l'interaction avec lui. Dans notre projet, les vues sont définies à l'aide de fichiers FXML et implémentées dans les classes SceneMM1Controller et SceneMM2Controller. Ces classes sont chargées de gérer l'interface utilisateur et de mettre à jour les éléments graphiques en fonction des actions de l'utilisateur.

Contrôleur (Controller) :

Le contrôleur fait le lien entre le modèle et la vue. Il reçoit les actions de l'utilisateur depuis la vue, effectue les opérations appropriées sur le modèle et met à jour la vue en conséquence. Dans notre projet, les contrôleurs sont représentés par les classes SceneMM1Controller et SceneMM2Controller. Ils sont responsables de la gestion des événements utilisateur, de l'interaction avec les éléments de l'interface utilisateur et de la coordination des actions avec le modèle.

MVC

1-ServerChat



La classe « ServerChat » représente l'application serveur qui écoute les connexions client et facilite la communication.

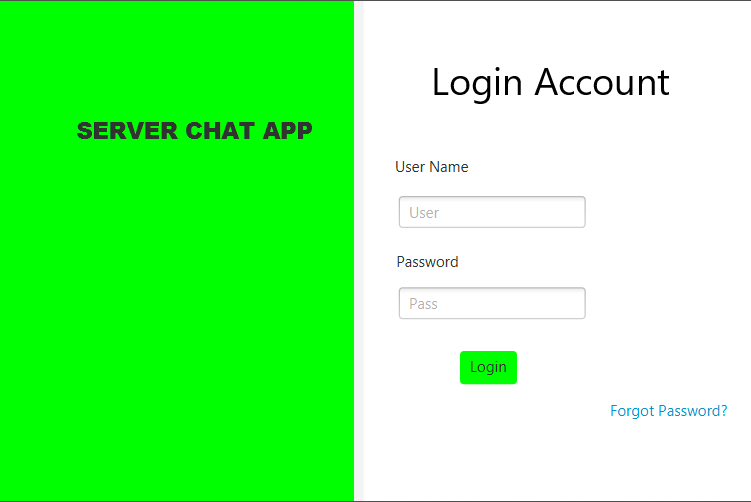
Lorsque vous exécutez la classe « ServerChat », elle démarre un socket serveur sur le port 1234 et attend les connexions client.

Le serveur affichera le message "Démarrage du serveur..." pour indiquer qu'il a démarré avec succès.

2-Authentification

La classe MainApp représente le point d'entrée de l'application JavaFX qui lance l'interface utilisateur.

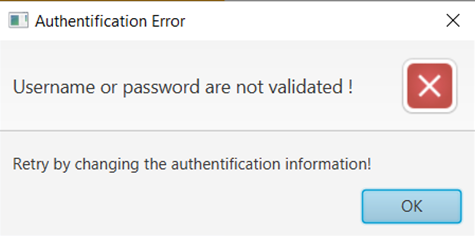
Lorsque vous exécutez la classe MainApp, elle lance l'application JavaFX et affiche l'interface utilisateur définie dans le fichier SceneMM1.fxml.



Dans l'interface utilisateur, entrez le nom d'utilisateur « admin » et le mot de passe « admin » (en supposant qu'il s'agit des bons identifiants de connexion).

Cliquez sur le bouton "Login".

La méthode onLoginMM de la classe SceneMM1Controller est déclenchée.



Si le nom d'utilisateur et le mot de passe saisis correspondent aux valeurs attendues ("admin" et "admin" dans ce cas), la connexion est réussie.

Si non il doit afficher un message « Username or password are not validate ».

Une fois la connexion réussie, l'interface utilisateur passera à la scène suivante, qui est contrôlée par le fichier SceneMM2.fxml.

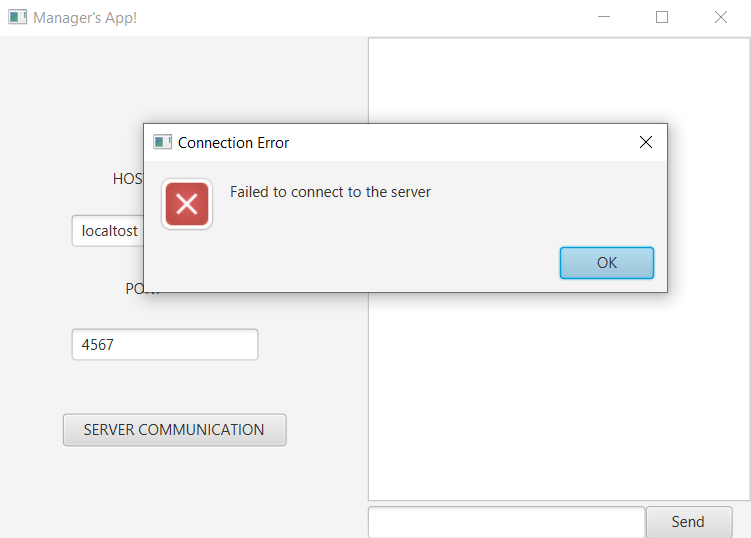
3-Communication

Après une connexion réussie, l'interface utilisateur passera à une nouvelle scène définie par le fichier SceneMM2.fxml.

La nouvelle scène se compose de deux sections : une pour la configuration du serveur et une pour la communication.

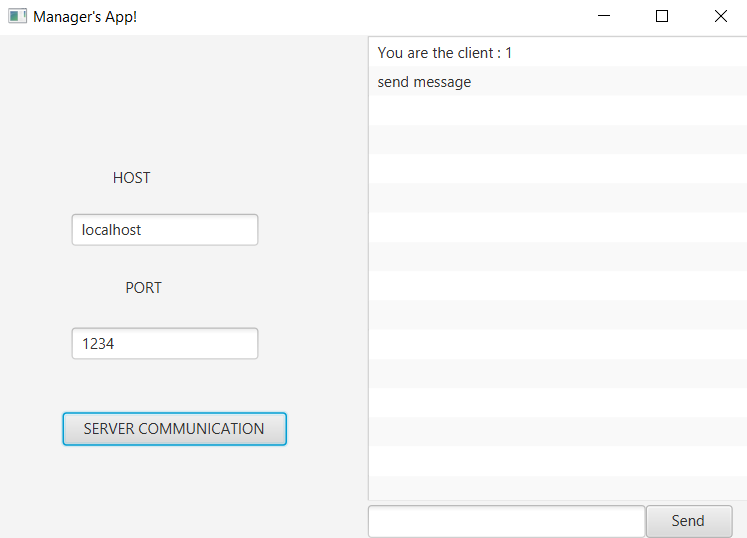
La section de configuration du serveur vous permet d'entrer un hôte et un port auxquels le serveur se connectera.

La section de communication comprend une vue de liste pour afficher les messages reçus des clients et un champ de texte avec un bouton "Envoyer" pour envoyer des messages.



Dans la section de configuration du serveur, entrez l'hôte et le port correspondant au serveur auquel vous souhaitez vous connecter.

Cliquez sur le bouton "SERVER COMMUNICATION" pour établir une connexion avec le serveur spécifié.

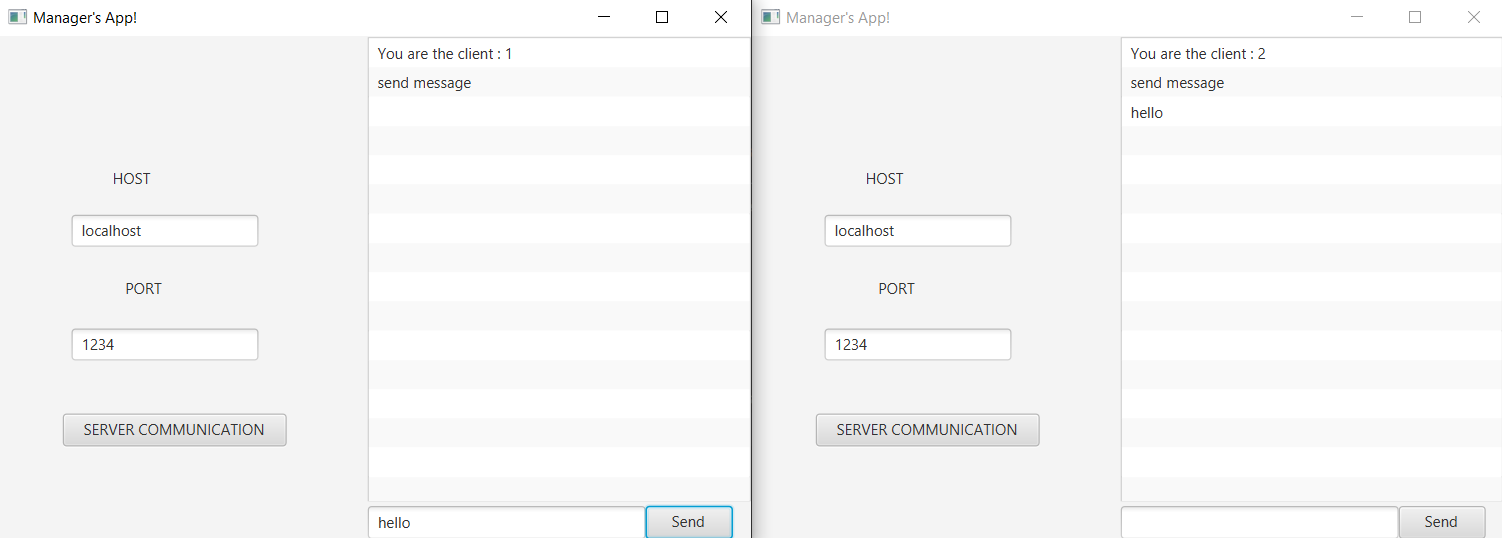


Après avoir établi une connexion avec le serveur, l'application commencera à écouter les messages du serveur.

Tous les messages reçus du serveur seront affichés dans la vue de liste.

Pour envoyer un message au serveur, entrez le message dans le champ de texte en bas et cliquez sur le bouton "Envoyer".

Le message sera envoyé au serveur, et si le serveur répond, la réponse sera affichée dans la vue liste.



Conclusion :

Ce projet de développement d'une application de communication client-serveur avec une interface graphique JavaFX permettra aux utilisateurs de se connecter, d'échanger des messages en temps réel et de communiquer de manière interactive. En suivant les différentes étapes décrites ci-dessus, nous avons pu concevoir, implémenter et exécuter avec succès l'application. Les tests réalisés ont démontré la fiabilité et la fonctionnalité de l'application, ouvrant ainsi des perspectives d'utilisation dans divers contextes de communication en temps réel.