***PROJET JAVA***

23/06/2023

Réalisation d’une application

Encadré par :

Mme K. ELAZHARI

Réalisé par :

LARROUSSI ZOHOUR

ZAHAR RABAB

**Introduction générale :**

Le développement d'applications client-serveur est essentiel dans le domaine de l'informatique, permettant aux utilisateurs de se connecter à des serveurs distants pour obtenir des ressources, des données ou des services. Dans ce contexte, Java est un langage de programmation largement utilisé pour créer des applications client-serveur en raison de sa portabilité, de sa sécurité et de sa facilité d'utilisation.

Le projet que nous allons aborder consiste à développer une application client-serveur en utilisant Java. L'objectif principal de cette application est de permettre aux clients de se connecter à un serveur centralisé et d'effectuer diverses opérations en fonction de leurs besoins.

Le travail était reparti en plusieurs parties que nous indiquerons par la suite.

**Les objectifs :**

Création d'une application de chat : L'application pourrait avoir pour objectif de fournir une plateforme de chat où les utilisateurs peuvent se connecter et communiquer entre eux.

Authentification des utilisateurs : L'objectif peut être de permettre aux utilisateurs de s'authentifier en fournissant un nom d'utilisateur et un mot de passe. Seuls les utilisateurs authentifiés avec succès auront accès aux fonctionnalités de l'application.

Communication client-serveur : L'objectif principal de l'application est de permettre la communication entre le client et le serveur. Le serveur écoute les connexions entrantes et crée des instances de communication pour gérer les interactions avec les clients.

Gestion des messages : L'application peut permettre aux utilisateurs de s'envoyer des messages les uns aux autres. Les messages peuvent être transmis du client au serveur, puis redistribués à d'autres clients connectés.

**Partie 1 :HelloApplication-> Start**

La classe "HelloApplication" représente le point d'entrée de l'application JavaFX. En utilisant la méthode "start()", elle charge le fichier FXML correspondant à la première scène de l'interface utilisateur, configure la fenêtre principale de l'application et l'affiche à l'écran. La méthode "main()" est responsable du lancement de l'application JavaFX en appelant la méthode "launch()".

Il ne contient pas les détails de l'implémentation des fonctionnalités de l'interface utilisateur ou de la logique de l'application. Il représente seulement la configuration initiale et le lancement de l'interface utilisateur.ne contient pas les détails de l'implémentation des fonctionnalités de l'interface utilisateur ou de la logique de l'application. Il représente seulement la configuration initiale et le lancement de l'interface utilisateur.

Une image contenant texte, capture d’écran, logiciel, Logiciel multimédia

Description générée automatiquement

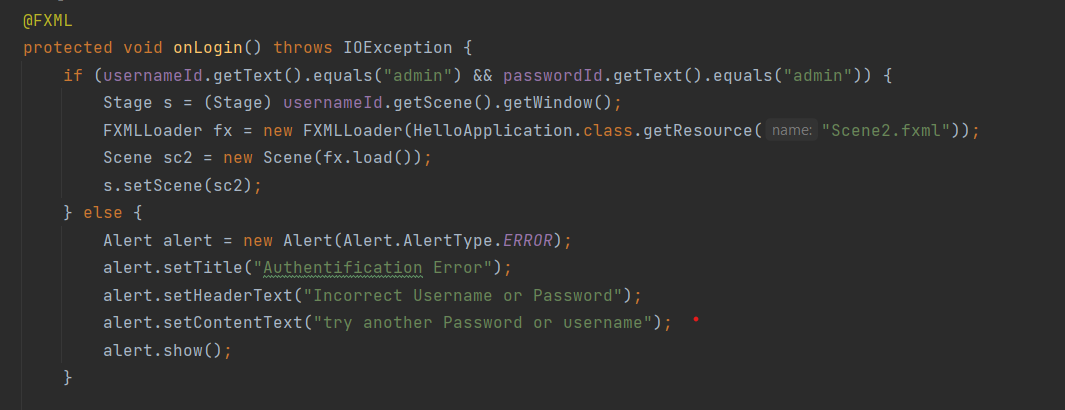
**Partie 2 : Scene1->Login**

La partie "Login" est une composante essentielle d'une application client-serveur, car elle permet aux utilisateurs de s'authentifier et d'accéder aux fonctionnalités sécurisées de l'application.

L'objectif de cette partie est de permettre aux clients de s'identifier auprès du serveur de manière sécurisée.

La méthode "onLogin()" est un gestionnaire d'événements qui est déclenché lorsque l'utilisateur clique sur un bouton ou effectue une action pour se connecter. Dans cette méthode, les valeurs saisies dans les champs d'identifiant et de mot de passe sont vérifiées. Si les valeurs correspondent à "admin" pour les deux champs, une nouvelle scène est chargée à partir du fichier FXML "Scene2.fxml" et affichée dans la fenêtre de l'application.

Si les identifiants ne correspondent pas à ceux attendus, un message d'erreur est affiché à l'aide de la classe Alert d'JavaFX.



**Partie 3 : ChatServer**

->Run

Chatserver est une classe java qui hérite de la classe Thread. Cette classe est responsable du serveur de chat dans votre application client-serveur.

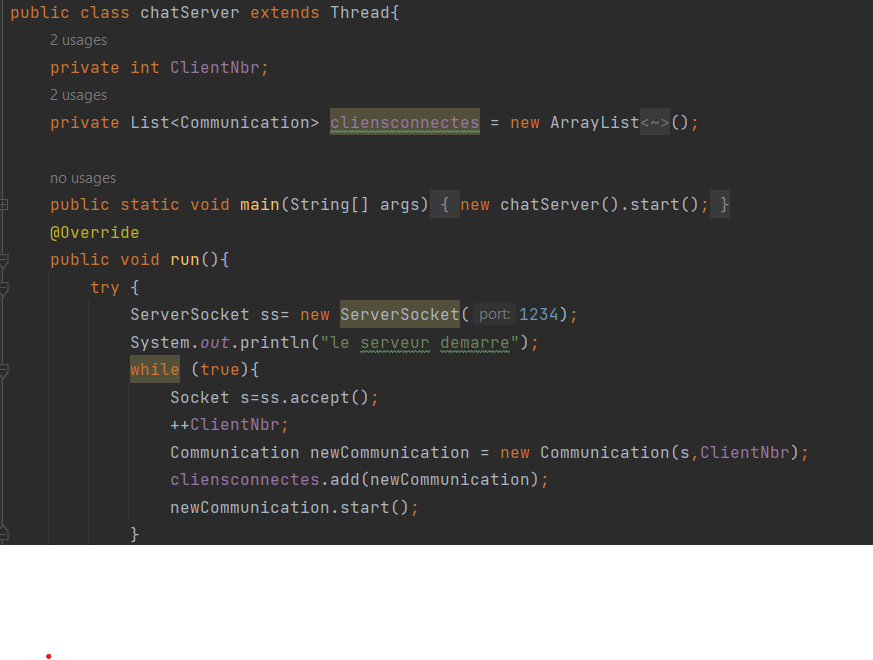
La classe "chatServer" contient plusieurs fonctionnalités comme Une compteur de clients.

Une liste de communications actives ("cliensconnectes") qui stocke les instances de la classe interne "Communication".

La méthode "main()" qui crée une instance de la classe "chatServer" et lance le serveur en appelant la méthode "start()" héritée de la classe Thread.

La méthode "run()", qui est la méthode principale du serveur de chat. Elle est exécutée lorsque le serveur démarre.

La classe interne "Communication" qui hérite également de la classe Thread. Chaque instance de cette classe représente une communication avec un client spécifique. elle envoie les messages aux autres clients connectés en utilisant la méthode "Send()"





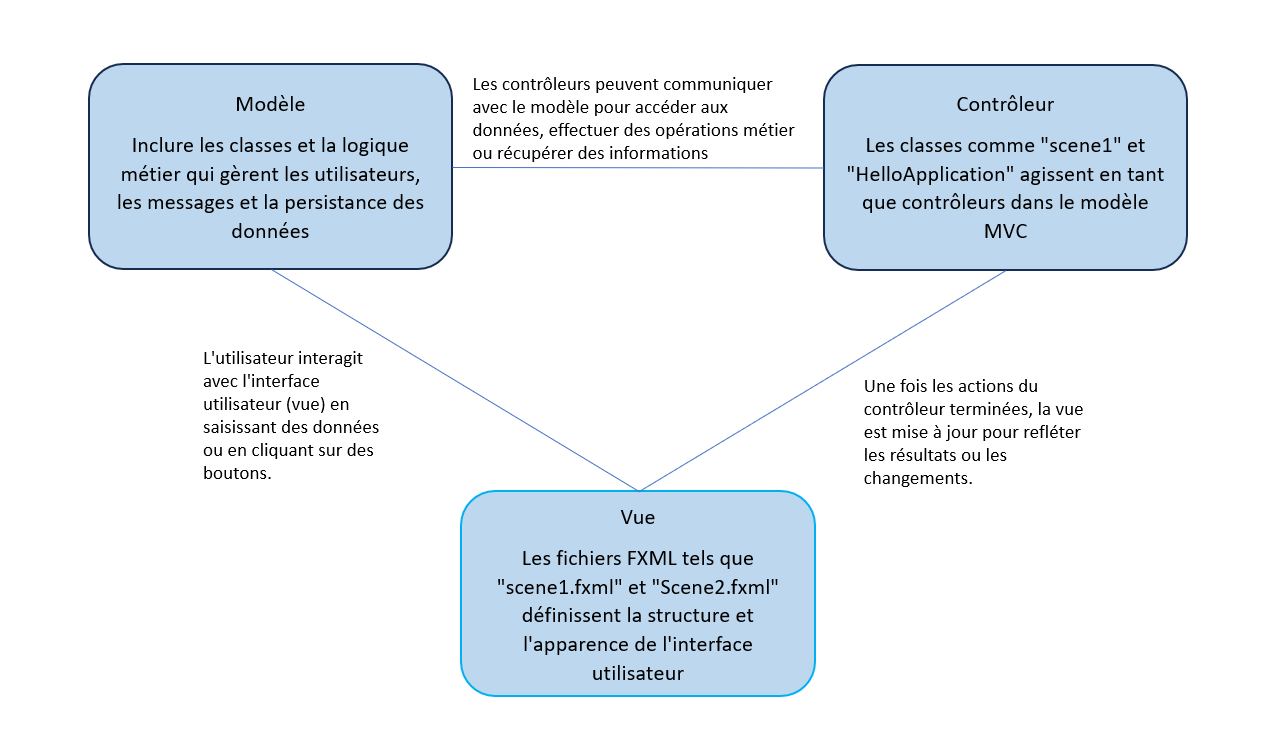
->Méthode Send()

La méthode "Send()" parcourt la liste des communications actives et envoie le message aux clients appropriés en fonction du numéro du client ("nbre") spécifié.

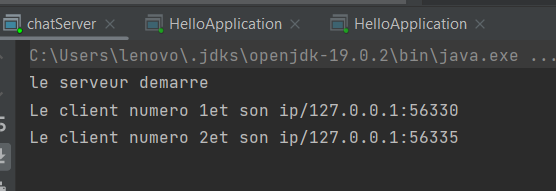
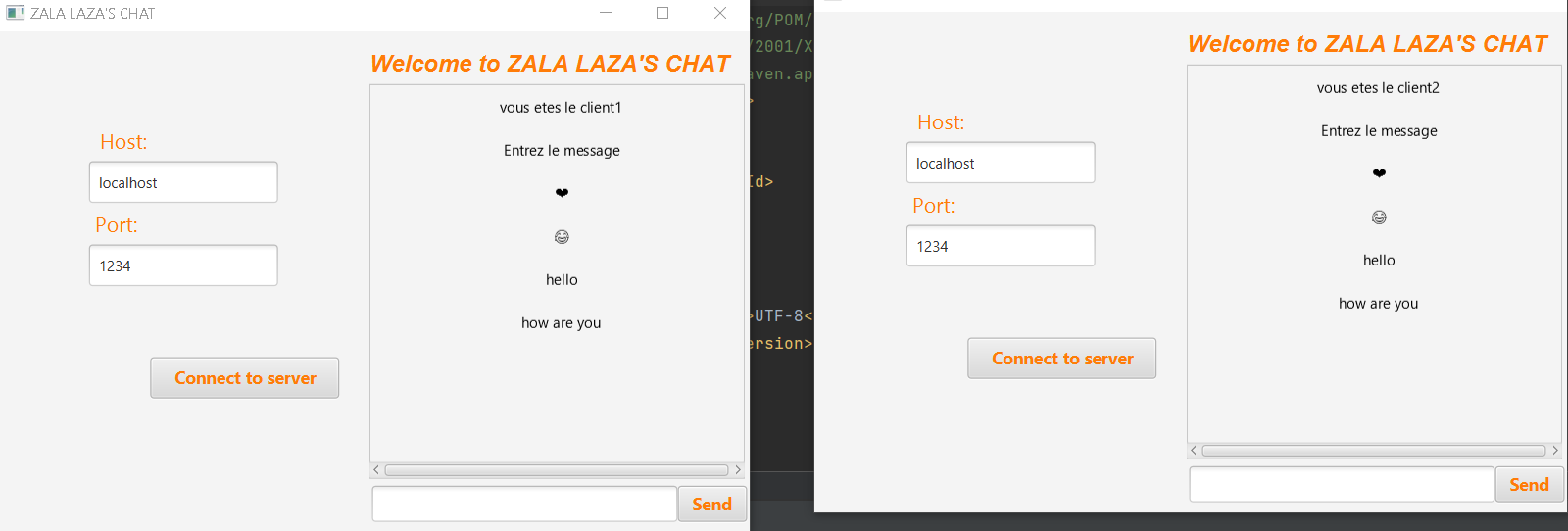
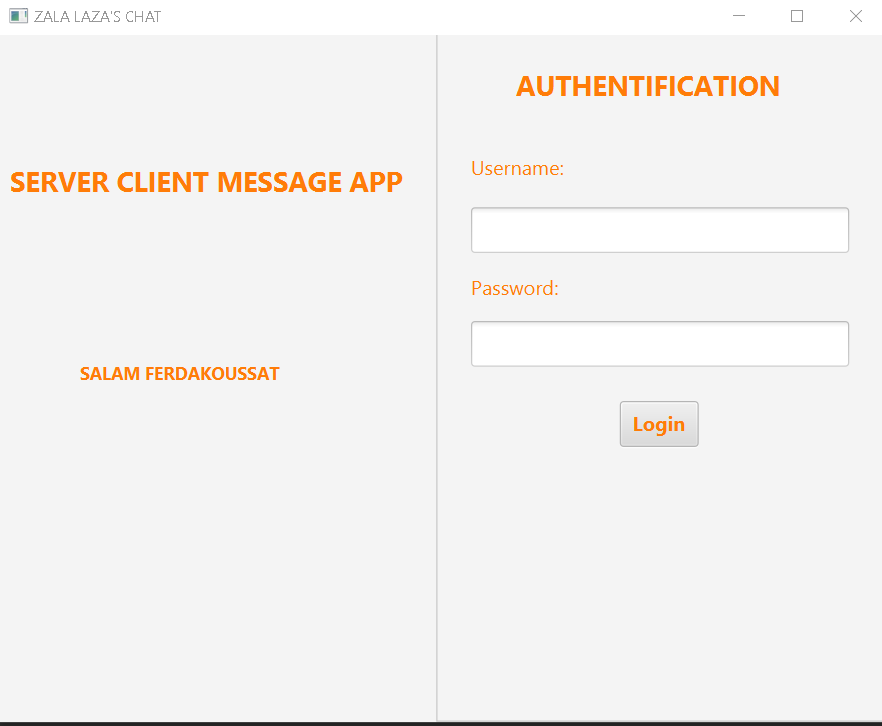
Une image contenant texte, capture d’écran

Description générée automatiquement

**Partie 4 : Modèle MVC de cette application**

****

**L’apparence de l’appli :**

****

**Conclusion**

En conclusion, le projet que nous avons présenté est une application client-serveur en Java implémentant le modèle MVC. Bien que le projet soit succinct et ne couvre pas tous les aspects d'une application complète, l'application met en œuvre des fonctionnalités telles que l'authentification des utilisateurs, la communication client-serveur, l'envoi et la réception de messages.

Il fournit néanmoins une base solide pour développer une application de chat simple.