**RAPPORT DU PROJET JAVA : APPLICATION CLIENT / SERVEUR**

# ***Réalisé par :*** **Salma Nacimi et Oussama Zouitni**

# Introduction

* INTERET DU PROJET
* OBJECTIF DU PROJET

# Étapes du projet

* EXPLICATION APPLICATION

# Modèle mvc

* MODEL
* VIEW
* CONTROLLER

# Fonctionnalités du projet

* COMMUNICATION ENTRE MODEL, VIEW ET CONTROLLER

# Conclusion

# Introduction :

INTERET DU PROJET :

Les projets Java d'application serveur-client sont utiles dans de nombreux domaines comme : la communication bidirectionnelle, la gestion centralisée des données etc. C’est un domaine passionnant ou vous créez des applications interactives, performantes et fiables pour répondre aux besoins des utilisateurs.

Notre projet vise à développer une application de communication client-serveur utilisant Java FX, un langage de programmation orienté objet offrant des fonctionnalités graphiques avancées.

L'application permettra aux utilisateurs de se connecter à un serveur de communication et d'échanger des messages individuels ou des messages diffusés à tous les clients connectés.

OBJECTIF DU PROJET :

Le projet vise à développer une application de communication client-serveur utilisant Java FX, permettant aux utilisateurs de se connecter, d'échanger des messages individuels et de diffuser des messages à tous les clients connectés.

L'objectif est de créer une plateforme conviviale et sécurisée qui facilite la communication et la collaboration entre les utilisateurs, en offrant une interface intuitive et en mettant en place des mécanismes de sécurité pour assurer la confidentialité des échanges.

Ces objectifs peuvent être affinés et ajustés au fur et à mesure que le projet progresse et que les besoins évoluent.

# Étapes du projet :

EXPLICATION APPLICATION :

L'application de communication client-serveur est un système informatique permettant à des utilisateurs de communiquer entre eux via un réseau. Elle est composée de deux parties distinctes : le client et le serveur.

L'application permet aux utilisateurs de communiquer de différentes manières, telles que l'envoi de messages individuels, la diffusion de messages à tous les clients connectés, la création de groupes de discussion, le partage de fichiers, etc. Elle offre également des fonctionnalités telles que la recherche d'utilisateurs, la gestion des contacts, l'historique des messages, etc.

Les étapes qu’on a suivi pour réaliser ce projet sont premièrement l’analyse des besoins et les exigences du projet. On identifie les fonctionnalités principales, les contraintes et les objectifs spécifiques de l'application serveur-client. Cela aidera à définir la portée du projet.

Après, le développement du serveur , cela peut inclure la mise en place d'un environnement de développement, le choix du Framework ou des bibliothèques appropriées, la création des fonctionnalités du serveur, la mise en place de la gestion des requêtes et des réponses, ainsi que la gestion des données .

Ensuite, le développement du client, une fois le serveur en place, . Concevez et créez l'interface utilisateur du client, en s’assurant qu'elle répond aux besoins des utilisateurs. L’implémentation des fonctionnalités du client, telles que la communication avec le serveur, la manipulation des données, l'affichage des résultats, etc.

Finalement, le déploiement et maintenance, une fois que l'application est prête, place au déploiement sur les serveurs appropriés en la rendant accessible aux utilisateurs.

# Modèle mvc :

Le modèle MVC offre une structure claire, une séparation des préoccupations, une réutilisabilité du code, une testabilité améliorée, une flexibilité et une évolutivité accrues, ainsi qu'une gestion simplifiée de l'interface utilisateur. Ces avantages font du modèle MVC une approche populaire et efficace pour le développement d'applications, en particulier dans les projets d'application serveur-client.

# Présentation du modèle MVC et de son importance dans le projet :

Dans le modèle de notre application de communication client-serveur, plusieurs classes ou structures de données peuvent être utilisées pour représenter les entités et gérer la logique métier. Voici quelques exemples :

* Classe Utilisateur : Cette classe représente un utilisateur et peut contenir des attributs tels que l'identifiant, le nom d'utilisateur, le mot de passe, les informations de profil, etc. Elle peut également inclure des méthodes pour la création, la modification et l'authentification des utilisateurs.
* Classe Message : Cette classe représente un message échangé entre les utilisateurs. Elle peut avoir des attributs tels que l'expéditeur, le destinataire, le contenu, la date et l'heure, etc. La classe peut inclure des méthodes pour l'envoi, la réception et la manipulation des messages.
* Classe Connexion : Cette classe représente la connexion entre le client et le serveur. Elle peut contenir des attributs tels que l'adresse IP, le port, l'état de la connexion, etc. La classe peut avoir des méthodes pour établir, fermer et gérer les connexions.
* Classe Groupe de discussion : Cette classe représente un groupe de discussion dans lequel les utilisateurs peuvent échanger des messages. Elle peut contenir des attributs tels que l'identifiant du groupe, le nom du groupe, la liste des membres, les messages échangés, etc. La classe peut inclure des méthodes pour la création, la gestion des membres et l'envoi de messages dans les groupes de discussion.
* Structure de données File d'attente (Queue) : Une file d'attente peut être utilisée pour gérer les messages en attente d'envoi ou de réception. Elle permet de maintenir l'ordre d'arrivée des messages et de gérer efficacement les opérations d'ajout et de retrait.
* Structure de données Liste (List) : Une liste peut être utilisée pour stocker les utilisateurs, les groupes de discussion, les messages reçus, etc. Elle permet d'organiser et de manipuler les données de manière séquentielle
* MODEL :

Dans le modèle de notre application de communication client-serveur, nous avons plusieurs entités telles qu’Utilisateur, Message, Connexion et Groupe de discussion, qui représentent les différentes composantes de la communication. Le modèle gère la logique métier associée à ces entités, telle que la création et la gestion des utilisateurs, l'envoi et la réception des messages, la gestion des connexions et la création et la gestion des groupes de discussion. La logique métier du modèle garantit l'intégrité des données, valide les entrées utilisateur, coordonne les interactions entre les entités et applique les règles métier spécifiques à l'application. En résumé, le modèle constitue le cœur de l'application, assurant la cohérence et la gestion des données, et fournissant les fonctionnalités essentielles pour la communication entre les utilisateurs

* VIEW :

La présentation des données et l'interface utilisateur dans notre application de communication client-serveur jouent un rôle essentiel pour offrir une expérience utilisateur agréable et intuitive. La présentation des données doit être claire et organisée, facilitant la lecture et la compréhension des messages, des utilisateurs, des groupes de discussion et des connexions. L'interface utilisateur doit être conviviale, offrant une disposition bien structurée des éléments et des fonctionnalités facilement accessibles. Les interactions avec l'interface doivent être fluides, réactives et permettre aux utilisateurs d'effectuer des actions telles que l'envoi et la réception de messages, la gestion des contacts et des groupes, avec facilité. Une attention particulière doit être portée à l'ergonomie, à l'esthétique et à l'expérience globale pour garantir une interaction fluide et satisfaisante avec l'application.

* Controller :

Le contrôleur occupe une position clé dans l'architecture MVC en tant que médiateur entre la vue et le modèle. Son rôle principal consiste à recevoir les actions de l'utilisateur depuis la vue, à mettre à jour le modèle en conséquence et à mettre à jour la vue avec les nouvelles données du modèle.

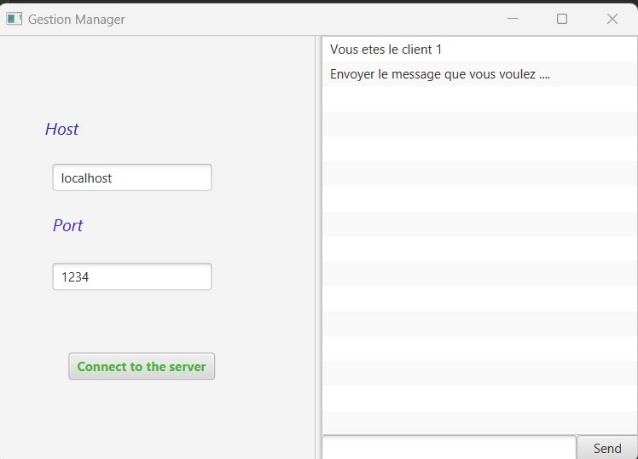
Il traite diverses actions, telles que la connexion et la déconnexion, l'envoi de messages, la gestion des contacts et la configuration de la connexion. Lorsque l'utilisateur se connecte, le contrôleur récupère les informations d'identification et les valide auprès du modèle, mettant à jour l'état de la vue en conséquence. Lorsque l'utilisateur envoie un message, le contrôleur récupère le contenu du message et le destinataire, puis les transmet au modèle pour traitement. Le contrôleur gère également les actions liées aux contacts, telles que l'ajout, la suppression ou la modification des informations de contact, en mettant à jour le modèle et en actualisant la vue. En outre, le contrôleur permet à l'utilisateur de configurer la connexion en saisissant le port et l'adresse du serveur, ainsi que le client de destination. En résumé, le contrôleur est responsable de la coordination et de la gestion des actions de l'utilisateur, assurant ainsi une interaction fluide entre la vue et le modèle de notre application.

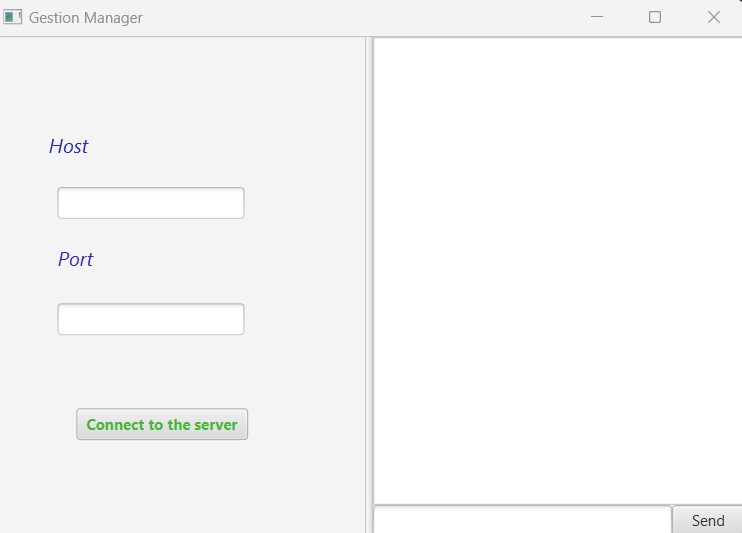
# Fonctionnalités du projet :

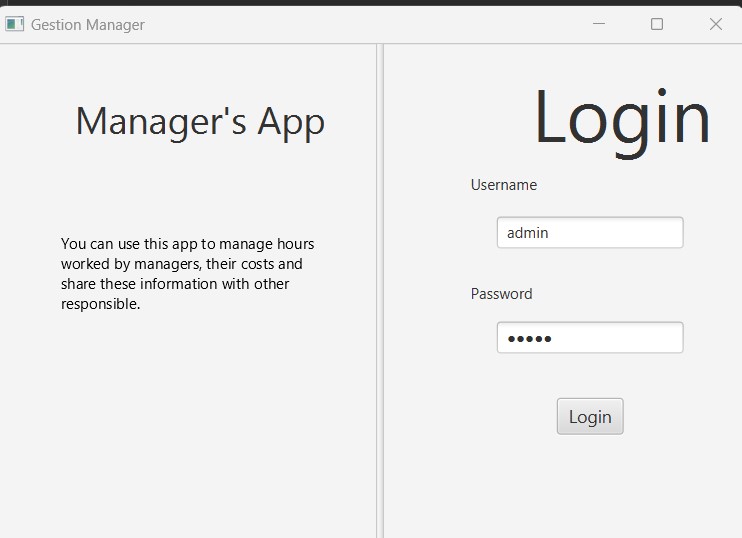
* COMMUNICATION ENTRE MODEL, VIEW ET CONTROLLER :

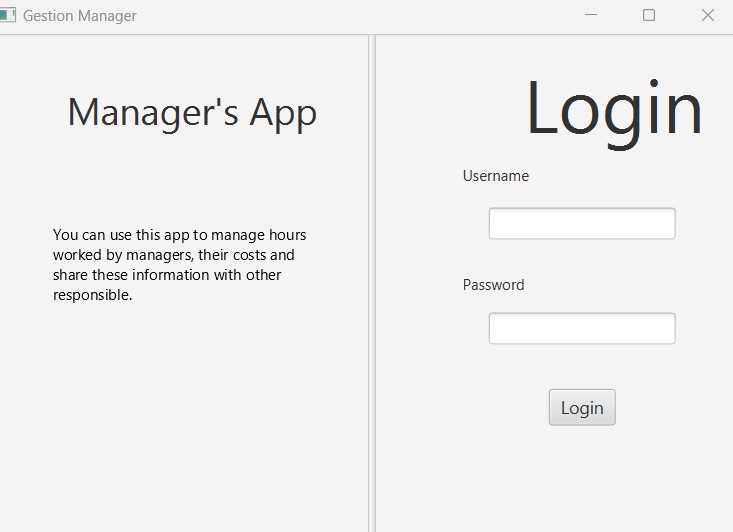
La communication entre le modèle, la vue et le contrôleur dans notre projet d'application client-serveur suit le flux de l'architecture MVC. La vue transmet des commandes et des événements au contrôleur en réponse aux interactions de l'utilisateur, tandis que le contrôleur échange des informations avec le modèle pour mettre à jour les données et obtenir les informations nécessaires à l'affichage. Ensuite, le modèle informe la vue des changements apportés aux données, ce qui permet à la vue de les récupérer et de les afficher à l'utilisateur. Cette communication fluide entre les composants assure la synchronisation des données et la mise à jour de l'affichage, offrant ainsi une expérience utilisateur homogène et réactive.

* L’interaction entre les utilisateurs (VIEW):







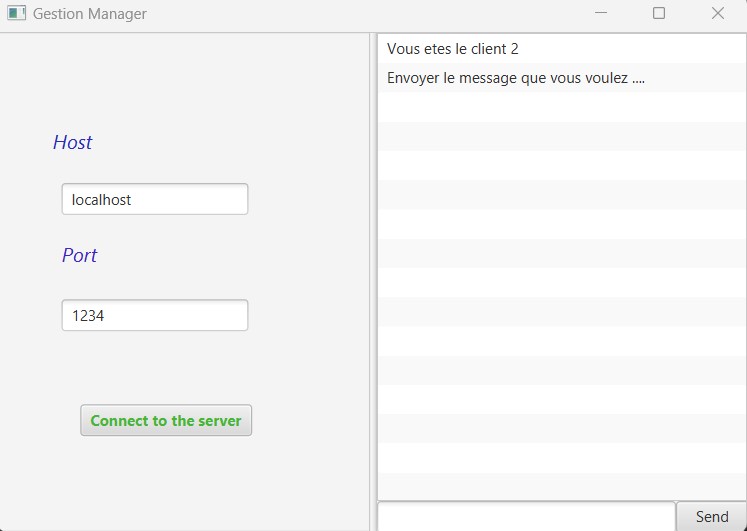


# 

# 

# 

# 



# Conclusion :

En conclusion, le projet d'application client-serveur de communication basé sur l'architecture MVC a abouti au développement d'une application fonctionnelle et conviviale qui facilite les échanges de messages entre les clients.

Dans l'ensemble, le projet a été couronné de succès, offrant une base solide pour de futures évolutions et extensions de l'application de communication client-serveur. Les améliorations envisagées permettraient de renforcer les fonctionnalités de l'application et de répondre encore mieux aux besoins des utilisateurs.