 

Mini Projet Systèmes Réparti

2023/2024

Réalisé par Achraf saoud IGL3

1-Gestion d'une liste de tâches

Java RMI :Gestion d'une liste de tâches :

* Ajouter une nouvelle tâche à la liste.
* Supprimer une tâche existante de la liste.
* Récupérer la liste complète des tâches.

La réalisation d'un service de gestion de tâches avec Java RMI requiert plusieurs étapes. Tout d'abord, il est nécessaire de définir une interface distante décrivant les opérations disponibles pour la manipulation des tâches. Ensuite, cette interface est implémentée côté serveur pour fournir les fonctionnalités nécessaires à la gestion des tâches, incluant l'ajout, la suppression et la récupération. Le serveur RMI est ensuite configuré pour héberger cette implémentation, permettant aux clients d'accéder aux fonctionnalités du service via le réseau. En parallèle, le client RMI est développé pour interagir avec le serveur, fournissant une interface utilisateur pour faciliter la manipulation des tâches. Enfin, des tests approfondis sont réalisés pour garantir le bon fonctionnement de l'ensemble du système, assurant ainsi une gestion efficace et fiable des activités pour les utilisateurs.

Les avantages de java rmi:

* le client peut être mis à jour de manière transparente puisqu'un simple redéploiement au niveau du serveur permet de modifier le comportement du client
* l'exécution distante de la méthode est masquée au code client, le code appelant travaillant avec les instances implémentant les interfaces comme si l'exécution était locale.

Les inconvenientes de java rmi:

* Dépendance à Java : RMI est spécifique à Java, ce qui signifie que les clients et les serveurs doivent être écrits en Java.
* Limitations de performance : dans certains cas, RMI peut présenter des limitations de performance par rapport à d'autres technologies plus légères comme les sockets.

2-Service de chat

Sockets :Service de chat :

* Envoi de messages texte à un salon de discussion commun.
* Récupération des messages envoyés par d'autres utilisateurs.

Nous avons mis en place un serveur de chat en Java utilisant des sockets pour la communication. Le serveur gère de manière efficace les connexions entrantes des clients, en créant des threads dédiés pour chaque client connecté. Ces threads assurent la réception des messages des clients, leur diffusion à tous les autres clients et la gestion des déconnexions. Ce modèle permet une communication bidirectionnelle entre le serveur et chaque client, ainsi qu'une communication multilatérale entre tous les clients connectés au salon de discussion.

Les avantages de Sockets:

* Flexibilité : les sockets offrent un contrôle total sur la communication entre les processus, ce qui permet de mettre en œuvre des protocoles personnalisés.
* Les sockets peuvent être utilisés avec n'importe quel langage de programmation qui prend en charge les opérations de réseau.

Les inconvenients de Sockets:

* Complexité : l'utilisation directe des sockets peut être complexe, surtout lorsqu'il s'agit de gérer des tâches comme la sérialisation des données ou la gestion des connexions simultanées.
* Maintenance : la gestion manuelle des connexions et des protocoles peut entraîner une complexité de maintenance accrue.

3-Service de chat

**gRPC :**Service de messagerie :

* Envoi de messages texte à un destinataire spécifié.
* Récupération des messages reçus par un utilisateur donné.

La mise en place d'un service de messagerie avec gRPC en Java nécessite la définition du service et des messages dans un fichier .proto est essentielle pour définir les fonctionnalités du service et les données échangées entre le client et le serveur. Ensuite, la configuration de Maven avec les dépendances appropriées et le plugin protobuf-maven-plugin permet de compiler le fichier .proto et de générer les classes Java correspondantes. Une fois les classes générées, l'implémentation du serveur gRPC consiste à créer une classe Java pour configurer et lancer le serveur, tout en implémentant les méthodes du service pour répondre aux demandes des clients. De même, l'implémentation du client GRPC implique la création d'une classe Java pour configurer le client et utiliser ses méthodes pour interagir avec le service sur le serveur. En suivant ces étapes, il est possible de mettre en place un service de messagerie fonctionnel et performant avec gRPC en Java.

Les avantages de gRPC:

* Performance élevée : gRPC est conçu pour être performant, avec un support pour la sérialisation binaire et des mécanismes de streaming efficaces.
* gRPC prend en charge plusieurs langages de programmation, facilitant l'intégration entre différents systèmes hétérogènes.
* Génération automatique de code : gRPC génère automatiquement le code pour les clients et les serveurs à partir d'un fichier de protocole, ce qui facilite le développement.

Les inconvenients de gRPC:

* Complexité accrue : bien que la génération de code automatique facilite le développement, la configuration et la gestion de gRPC peuvent être complexes, surtout pour les applications plus complexes.