Document de réponse

Elsass_Chat

Tâches effectuées

1. Création d'un serveur de communication asynchrone
2. Création d'une base de données permettant d'enregistrer les données utilisateurs (des comptes administrateurs et un compte utilisateur sont déjà créé)
3. Authentification des clients auprès du serveur
4. Possibilité d'utiliser un compte existant ou de créer un nouveau compte
5. Création de 5 salons (Général, Blabla, Comptabilité, Informatique, Marketing)
6. Limitation de la diffusion des messages aux utilisateurs du même salon
7. Possibilité de changer de salon (requête au serveur pour 'Blabla', requête aux administrateurs salon pour 'Compta', 'Info' et 'Market')
8. Reconnexion automatique au dernier salon ou était l'utilisateur
9. Enregistrement des informations clients dans le base de données (login, mot de passe, pseudo, salon actuel, salons autorisées, niveau d'autorité, requêtes pour accéder à un salon)
10. Enregistrement des conversations dans la base de données (compte émetteur, salon)
11. Possibilité de bannir temporairement ou de manière permanente les clients, soit en utilisant le pseudo soit en utilisant une adresse IP
12. Division des utilisateurs en trois niveaux d'autorité : utilisateur simple, administrateur d'un salon, administrateur du serveur
13. Seul l'administrateur du salon peut utiliser la commande /kill lorsqu'il est connecté en local sur le serveur
14. Seuls les administrateurs peuvent utiliser les commandes /kick, /unkick, /ban, et /unban et voir les requêtes d'accès (/requests) ainsi que les accepter (/accept_request) ou les refuser (/deny_request)
15. La documentation DocString
16. Le guide d'installation
17. Le guide d'utilisation
18. Le document de réponse

Tâches non effectuées

1. L'interface graphique côté client
2. Possibilité d'établir une communication privée entre deux utilisateurs
3. Affichage du statut de connexion

L’interface graphique était planifiée au départ mais n’a finalement pas pu être complétée dans les délais du contrat. En raison de divers problèmes rencontrés lors de sa programmation, principalement sur le fonctionnement de Qt6 dans un modèle de serveur asynchrone, l’interface graphique client que j’ai pour le moment réalisé n’est pas dans un état fonctionnel.

Pour cette raison, l’application client fonctionne actuellement via une interface de commande.

Limites de ce type d’outil à long terme

A l’origine, il m’a été demandé de développer un serveur de communication asynchrone personnalisé afin que l’entreprise puisse éviter d’utiliser des services similaires plus populaires comme Discord ou Webex, principalement pour des raisons de sécurité des données.

En effet, il n’est pas un secret que ces services, étant majoritairement gratuits, utilisent et revendent les données de leurs utilisateurs à d’autres entreprises. De ce fait, si il y a besoin de discuter de données potentiellement sensibles entre collègues, je pense que créer un outil interne à l’entreprise est un choix judicieux.

Cependant, j’aimerais attirer l’attention du contractant sur deux points qui pourraient potentiellement poser problème sur le long terme.

Le premier est la sécurité des transmissions. Certes, la plupart des entreprises comme Discord revendent les données utilisateurs, ce que nous voudrions éviter. Cependant, il faut également considérer la vulnérabilité du service aux attaques externes, type DDoS ou hacking.

De ce point de vue, une entreprise spécialisée dans le domaine est capable de proposer un produit beaucoup mieux équiper à faire face à ce genre d’attaques, surtout si l’on considère que ces problèmes de sécurité vont être une problématique à long terme.

Le second point est la maintenance du produit sur le long terme. Comme mentionné précédemment, les problématiques d’attaques externes liées à un service de communication vont demander une maintenance et une amélioration constante tant que le service est utilisé.

De plus, si nous considérons que le produit va être déployé à relativement grande échelle ou bien qu’il va continuer de s’améliorer, notamment avec de nouvelles fonctionnalités, il sera nécessaire de dédier des individus à sa maintenance. Cela est dotant plus impératif si il devient le mode principal de communication au sein de l’organisation.

En raison de cela, l’efficacité d’un service de communication asynchrone personnalisé peut être limité par rapport aux alternatives plus populaires. Cependant, cela ne veut pas dire qu’il n’est pas possible de remédier aux deux problèmes cités précédemment.

Si le contractant souhaite tout de même un service interne à l’entreprise pour éviter la revente de ses données mais qu’il n’est pas possible de dédier suffisamment de ressource pour le sécuriser contre des attaques externes, je recommande de limiter l’échelle d’utilisation du service.

En effet, ce service à travers plusieurs réseaux, c’est-à-dire des LANs différents, ouvriraient la possibilité à des attaquants externes d’intercepter les communications, de les modifier ou encore d’essayer de s’introduire dans le réseau de l’entreprise.

Or, limiter l’utilisation du service à un même site, c’est-à-dire que le serveur et les clients sont sur le même LAN, aiderait à sécuriser les transmissions sans y dédier davantage de ressources. Dans ce cas, la sécurité du service dépendrait de la sécurité du réseau, assurée par l’opérateur. Cela permettrait donc d’éviter la revente potentielle des données transmises tout en limitant la vulnérabilité aux attaques externes.

Pensées sur la SAE32

Globalement, je pense que la SAE32 a été un projet très intéressant à réaliser.

Cela nous a permit de développer un outil utilisable dans une situation concrète et pour des raisons qui sont d’actualité (sécurité des données). C’était un projet d’autant plus motivant à faire correctement que nous pourrions être amené à créer des outils similaires une fois embauchés par une entreprise.

Si j’aurais un point à modifier, ce serait concernant la partie interface graphique. Ayant moi-même eu beaucoup de difficultés à ce niveau, j’aurais aimé que nous passions plus de temps à apprendre à faire fonctionner Qt6 avec les threads durant les séances de R309.