Rapport de projet

WISHCORD

BERTHIER Thomas

CARUELLE Jérémy

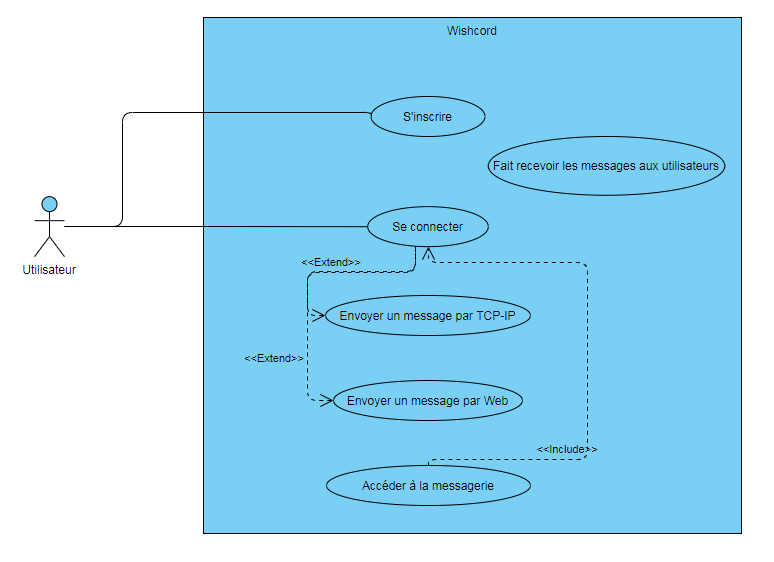
BOUCHER Louis

Sommaire :

* Analyse
* Serveur
* Client
* Client Web

Analyse

Use Case



-L’utilisateur peut s’inscrire.

-L’utilisateur peut se connecter.

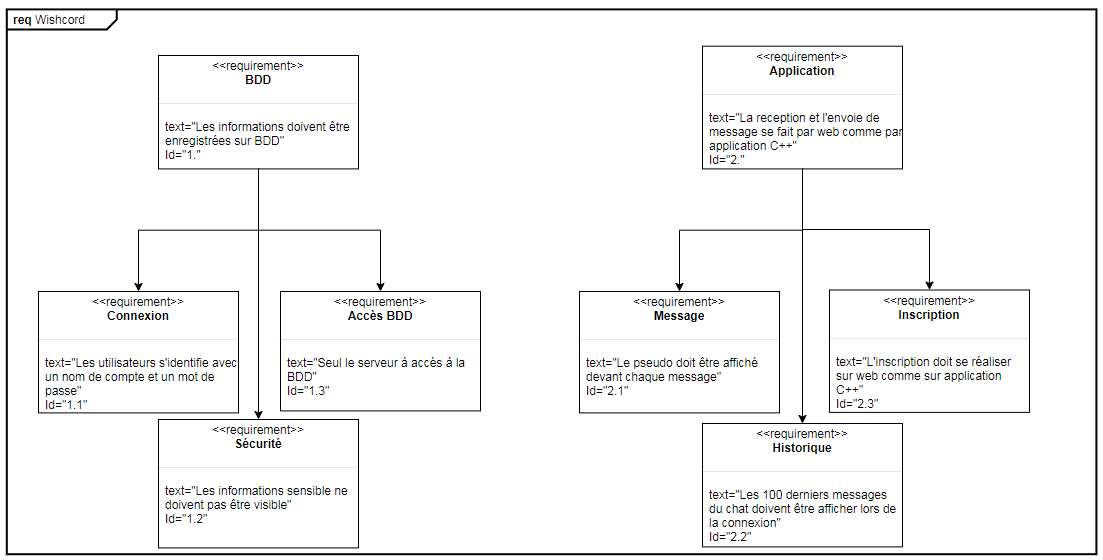
-L’utilisateur accède automatiquement à la messagerie.

-L’utilisateur peut envoyer un message par TCP-IP.

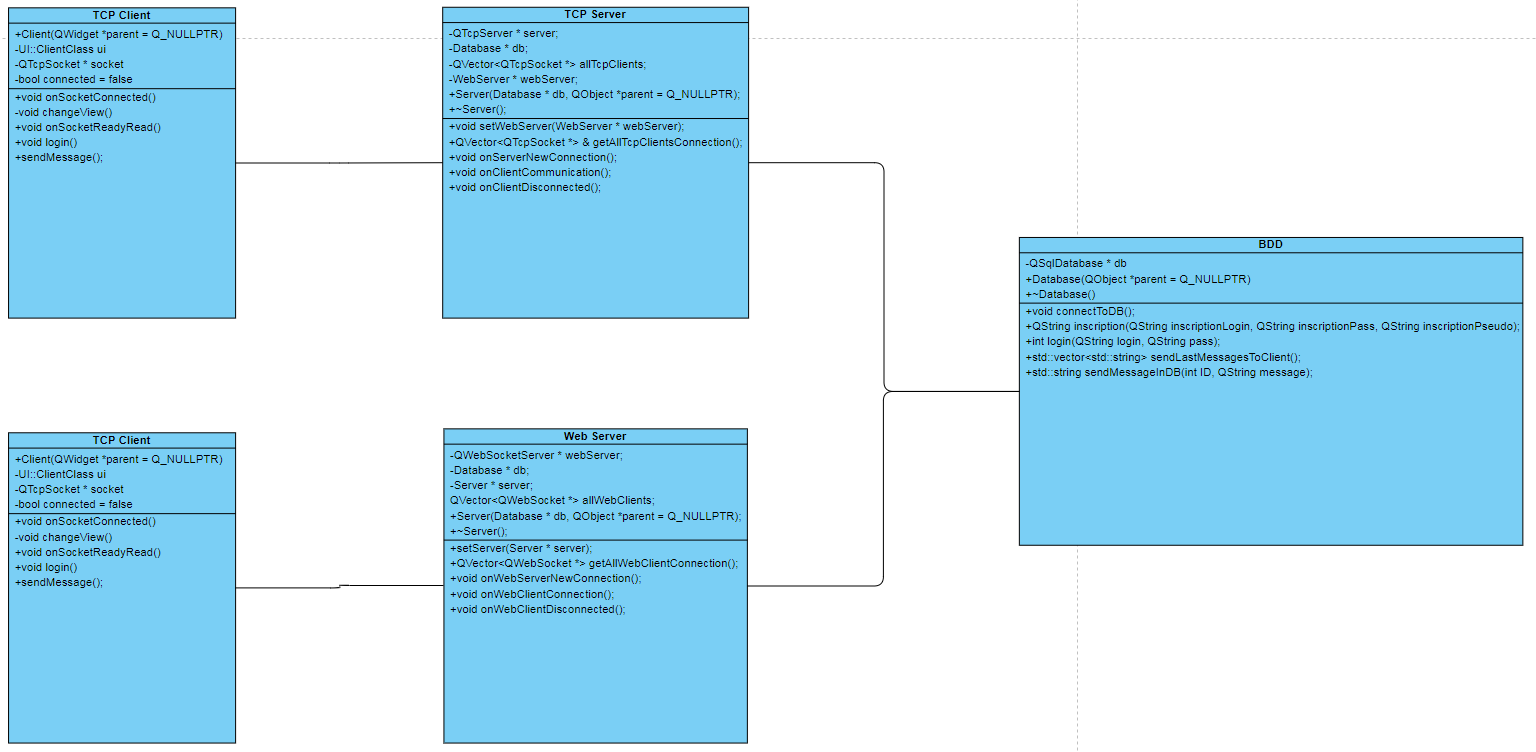
-L’utilisateur peut envoyer un message par Web.

-L’application fait recevoir les messages au utilisateurs.

Exigences

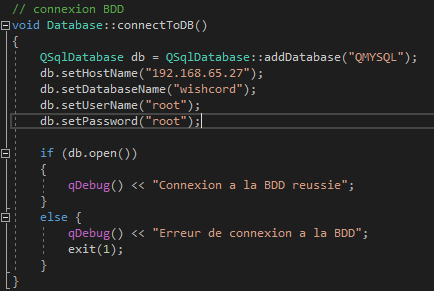


Classes



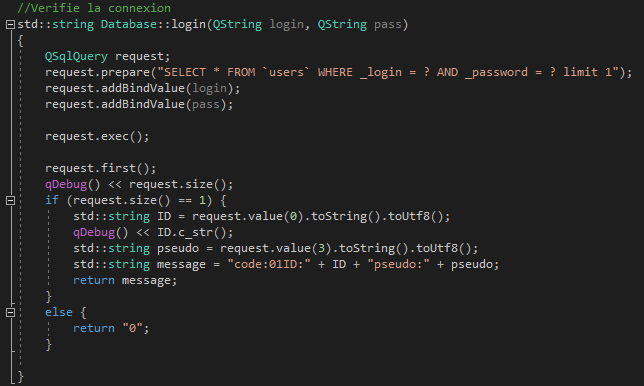
Serveur

Database

void Database::connectToDB() 

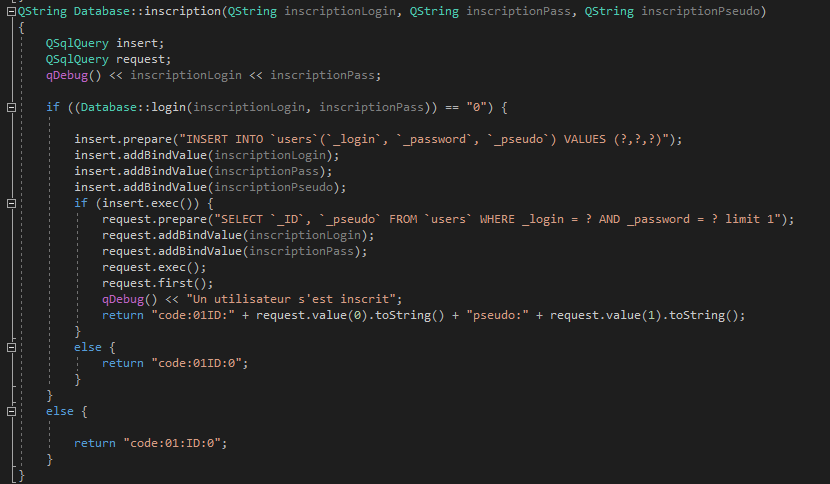
Instancie une nouvelle connexion à la base de donnée puis essaye de se connecter à celle-ci. Si l’opération est échouée, termine le programme.

std::string Database::login(QString login, QString pass)



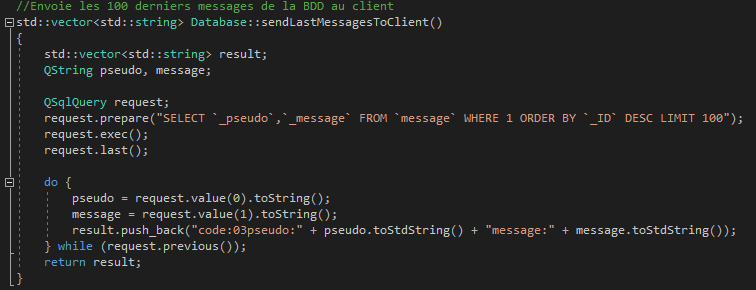
Effectue une requête SQL prenant le login et le mot de passe envoyé par le client pour vérifier si le compte existe. Si un résultat est trouvé, un message formaté contenant des informations nécessaires au client est retourné, sinon, 0 est retourné.

QString Database::inscription(QString inscriptionLogin, QString inscriptionPass, QString inscriptionPseudo)



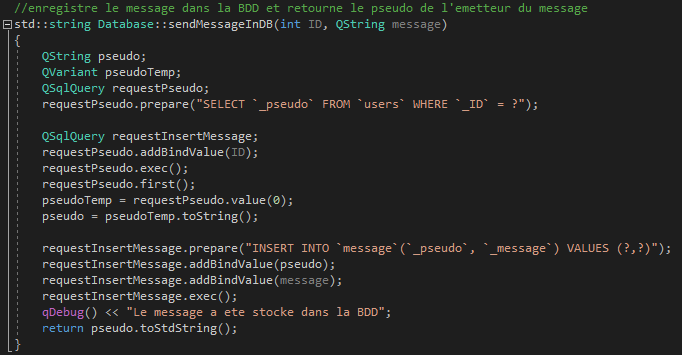
Effectue la méthode login () en prenant en compte le login et mot de passe envoyé par le client, si cette méthode retourne 0 cela signifie qu’aucun compte n’a été créé avec ces identifiants et donc qu’une requête SQL peut être effectuée pour créer ce compte. Si la création est réussie, une autre requête SQL est effectuée pour obtenir l’ID du compte crée dans la BDD puis retourne un message formaté contenant des informations nécessaires au client. Si la méthode login () ou l’insertion SQL n’est pas effectuée, un message formaté différent est retourné.

std::vector<std::string> Database::sendLastMessagesToClient()



Effectue une requête SQL qui récupère les 100 derniers messages. Ces messages sont formatés et enfin stocker dans un tableau dynamique le temps que toutes les tuples n’ont pas été enregistrés dans le tableau dynamique avant de retourner le tableau.

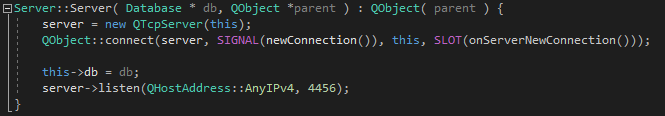
std::string Database::sendMessageInDB(int ID, QString message)



Effectue une requête SQL avec l’ID envoyé par le client qui récupère le pseudo du compte lié à l’ID. Une autre requête est ensuite effectuée pour insérer le message et le pseudo de l’émetteur dans la base de donnée puis retourne le pseudo de l’émetteur.

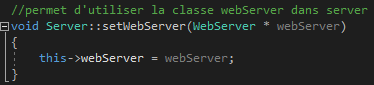
Server

Constructeur



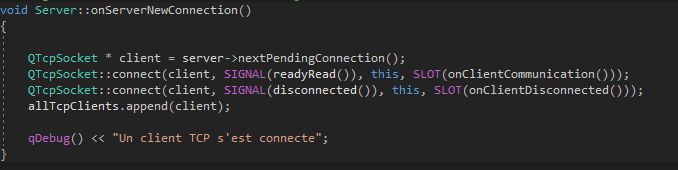
Créer un nouveau serveur QTcp et connecte le slot/la méthode onServerNewConnection () au signal newConnection () pour l’appeler à chaque nouvelle connexion. Notre classe Database est ensuite stocké dans les attributs privés de la class Server. Pour finir, on fait écouter le serveur sur le port 4456.

void Server::setWebServer(WebServer \* webServer)



Méthode permettant d’utiliser la class WebServer en l’ajoutant aux attributs privés.

void Server::onServerNewConnection()



Créer un QTcpSocket égale à la connexion du client au serveur puis connecte le slot/la méthode onClientCommunication () au signal readyRead () pour l’appeler lorsqu’un échange a lieu entre le client et le serveur et connecte également le slot/la méthode onClientDisconnected () au signal disconnected () pour l’appeler quand le client se déconnecte du serveur. Pour finir la connexion du client est ajouté dans un tableau dynamique crée dans les attributs privés qui regroupe l’ensemble des connexions.