Lycée TOUCHARD WASHINGTON

Section des Techniciens Supérieurs
Cybersécurité Informatique Electronique
Option Informatique et Réseaux

TD3 – Serveur Banque

Socket sous Qt - Côté serveur & Base de données

- *Date : décembre 2023*
- *Version : 2.1*
- *Référence : TD3 – Banque Serveur.odt*

---

1. Objectif

- *Utilisation des sockets avec la bibliothèque **Qt***
- *Comprendre le fonctionnement des sockets en mode événementiel.*
- *Codage de l'information*
- *Communication réseau*

2. Conditions de réalisation

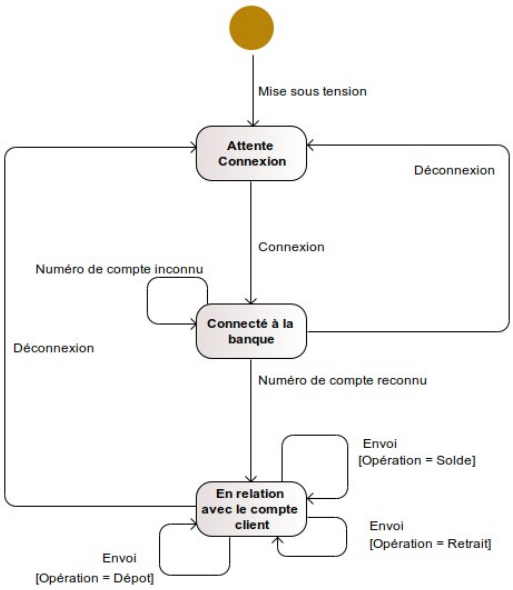
- *Ce fichier contient des liens hypertextes.*
- *Ressources utilisées :*
  - *Un PC sous Linux*
  - *Un client est disponible*
  - *Qt-creator*

3. Ressources

*Les classes sockets dans la technologie QT, consulter le site https://doc.qt.io/qt-6/qtnetwork-programming.html et plus particulièrement la classe, **QTcpServer** https://doc.qt.io/qt-6/qtcpserver.html.*

4. Rappel du besoin

Les distributeurs automatiques de billets (DAB) permettent de se connecter à une banque. Ils offrent la possibilité d'interroger un compte pour en connaître le solde, de déposer de l'argent et d'en retirer. Les fonctionnalités attendues côté client sont décrites par le diagramme d'état du système suivant :



Le diagramme à états ci-contre indique que c'est seulement lorsque le client est connecté à sa banque et que son numéro de compte est reconnu que les opérations vers la banque peuvent être réalisées. La déconnexion reste possible à chaque état.

4.1. Rappel du protocole d'échange entre les clients et le serveur

La trame en provenance du client pour sélectionner un numéro de compte :

Taille des données	**Commande**	**Valeur**
De type **quint16**	'N' - Un caractère sous la forme d'un **QChar**	Un entier de type **int**

La trame en provenance du client pour effectuer un retrait :

Taille des données	**Commande**	**Valeur du retrait**
De type **quint16**	Pour le retrait : 'R' - Un caractère sous la forme d'un **QChar**	Un réel de type **float**

La trame en provenance du client pour effectuer un dépôt :

Taille des données	**Commande**	**Valeur du dépôt**
De type **quint16**	Pour le dépôt : 'D' - Un caractère sous la forme d'un **QChar**	Un réel de type **float**

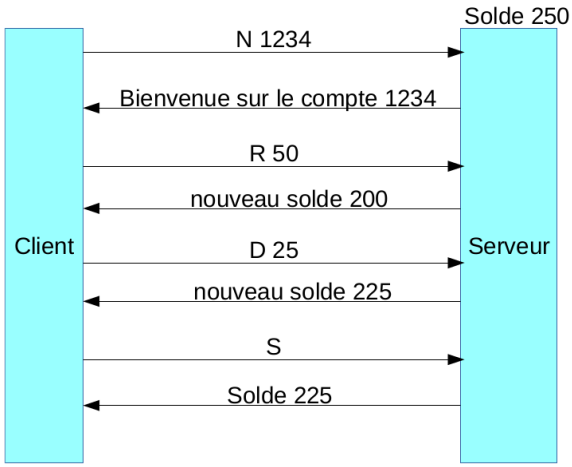
La trame en provenance du client pour obtenir le solde du compte :

Taille des données	**Commande**	**Valeur**
De type **quint16**	Pour le solde : 'S' - Un caractère sous la forme d'un **QChar**	Sans objet

Dans tous les cas de figure, le serveur répond un message sous la forme

Taille des données	**Message**
De type **quint16**	Une chaîne de caractère sous la forme d'une **QString**

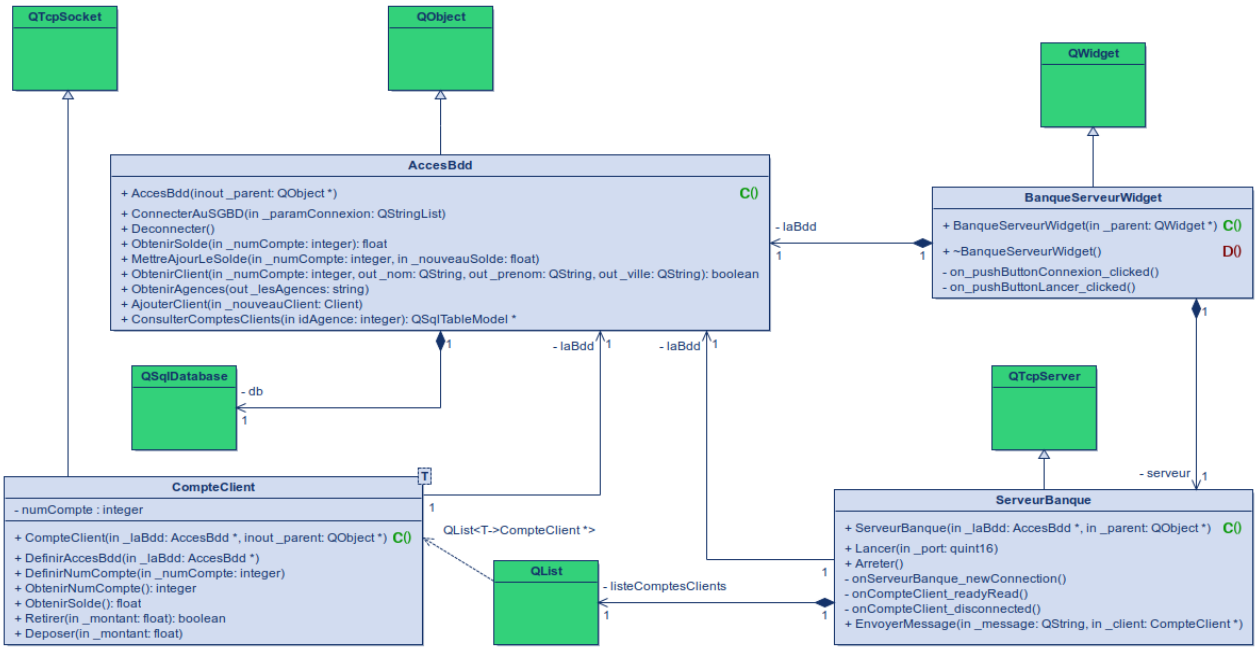
4.2. Exemple de communication client-serveur :



5. Réalisation d'un serveur multi-clients

5.1. Création du projet

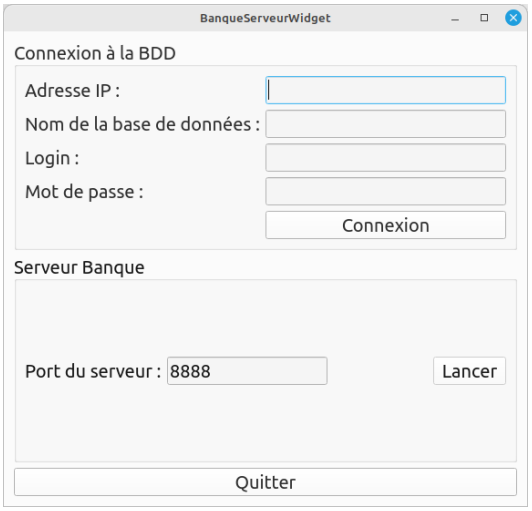
1. Créez le projet **ServeurBanque** de type Application Qt utilisant les Widgets. La classe principale se nomme ***BanqueServeurWidget***. Ce projet utilise une communication réseau entre un serveur et des clients et accède à un système de gestion de bases de données SQL, modifier le fichier **.pro** de votre projet en conséquence.



Attention : Identifiez bien les différentes relations entre chaque classe. Elles sont différentes des Travaux dirigés précédents.

Les relations de composition seront implémentées dynamiquement dans les constructeurs respectifs, et les ressources correspondantes seront libérées dans les destructeurs. Les associations seront également représentées par des pointeurs, initialisés via des paramètres passés au constructeur de la classe.

L'interface utilisateur est illustrée dans la figure ci-contre :



Une fois connecté à la base de données, le bouton **Connexion** devient **Déconnexion**.

La zone intitulée **Serveur Banque** dans l'interface n'est accessible que si l'utilisateur est connecté à la base de données.

Lorsque le serveur est lancé, il est impossible de se déconnecter de la base de données. Le serveur doit d'abord être arrêté.

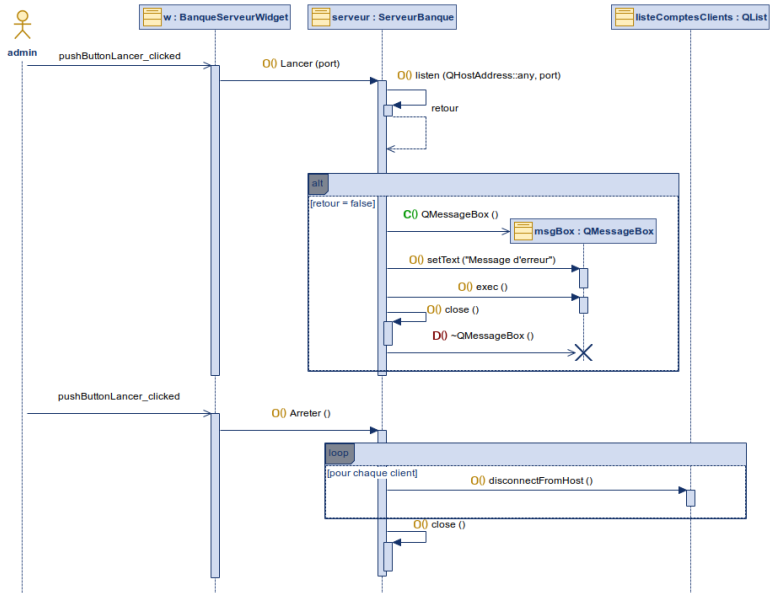
5.2. Réalisation de la classe AccesBdd

Vous pouvez reprendre la classe développée lors du « TD1 – Agence Bancaire », vous ajoutez les nouvelles méthodes.

Codez ces nouvelles méthodes.

5.3. Diagramme de séquence pour le lancement et l'arrêt du serveur

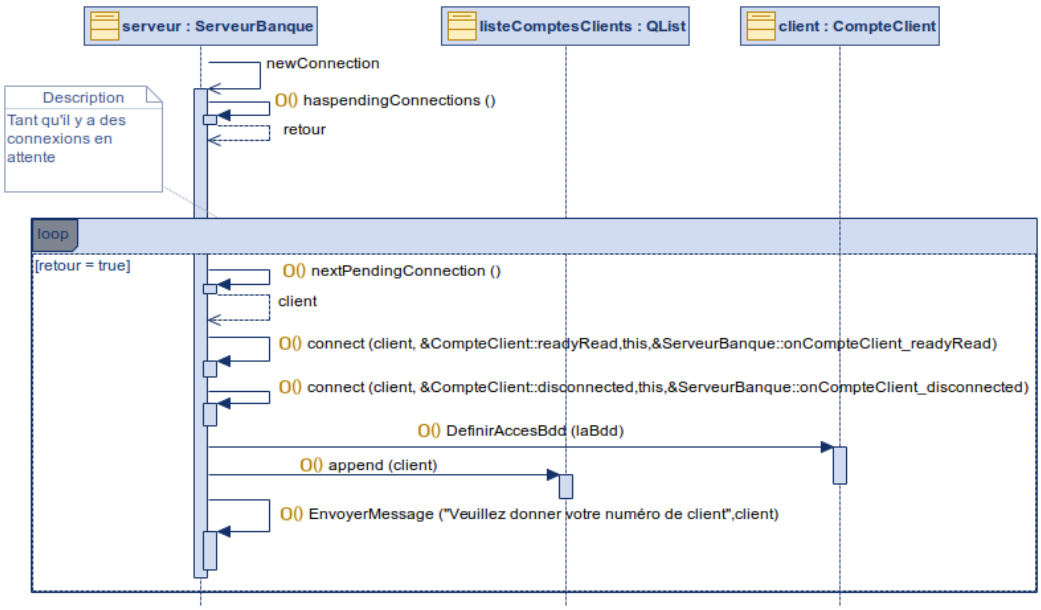
La description du fonctionnement des méthodes ***Lancer()*** et ***Arreter()*** est représentée par le diagramme de séquence suivant :



2. Codez ces deux méthodes ainsi que le destructeur de la classe ***BanqueServeurWidget***.
3. Déclarez la classe ***CompteClient*** en respectant la relation indiquée dans le diagramme de classe et complétez son constructeur. Il reçoit en paramètre un pointeur sur ***QObject*** initialisé par défaut avec la valeur ***nullptr*** et le transmet à sa classe de base, initialise à 0 le numéro de compte et à nullptr le pointeur sur AccesBdd.

4. Codez la relation entre la classe ***ServeurBanque*** et la classe ***CompteClient*** sous la forme d'une **QList,** comme le montre le diagramme de classe.

   Le diagramme de séquence suivant montre l'enchaînement des opérations lorsqu'un client se connecte à la banque.



5. Réalisez le codage du ***slot*** correspondant à ce signal.
6. Codez le ***slot*** en relation avec le signal indiquant qu'un client s'est déconnecté. S'il n'y a pas d'erreur, celui-ci retire le client de la liste, déconnecte les ***signaux*** et les ***slots*** qui lui sont associés et détruit le socket de dialogue vers le client.
7. Lancer votre **serveur banque** et votre **client distributeur automatique de billets** vérifiez la connexion et la déconnexion de votre client.
8. Le codage du dernier slot se fera après le codage de la classe ***CompteClient***.

5.4. Réalisation de la classe CompteClient

10. Complétez le codage du constructeur en initialisant les attributs solde et numéro de compte avec une valeur nulle
11. Les méthodes ne posent aucune difficulté :
    - ***Retirer()*** : si cela est possible, le solde est débité du montant passé en paramètre.
    - ***Deposer***() : crédite le solde si le montant passé en paramètre est supérieur à zéro.
    - ***ObtenirSolde***() : renvoie la valeur du solde
    - ***DefinirNumCompte()*** : Affecte la valeur du paramètre au numéro de compte.
    - ***ObtenirNumCompte()*** : renvoie le numéro du compte

5.5. Finalisation de la classe ServeurBanque

12. Codez le dernier ***slots*** correspondant au signal indiquant que des données sont parvenues au serveur. Vous respecterez le protocole rappelé au début du sujet et vous utiliserez un ***QDataStream*** après avoir vérifié qu'un nombre suffisant de données a bien été reçu.

Extrait de l'utilisation QDataStream

```
if (socketVersClient->bytesAvailable() >= static_cast<qint64>(sizeof(taille)))
    {
        QDataStream in(socketVersClient);
        in >> taille;
        // à compléter

    }
```

SocketVersClient représente un pointeur sur l'objet qui a généré le signal ***readyRead***.