Lycée TOUCHARD WASHINGTON

Section des Techniciens Supérieurs
Systèmes Numériques option
Informatique et Réseaux

TD2 – Accès à une base de données sous Qt

La Banque – Socket et Bdd

- *Date : novembre 2022*
- *Version : 2*
- *Référence : TD2 – Socket Serveur et Acces BDD sous Qt (Banque).odt*

---

1. Objectif

- *Accès aux bases de données avec la bibliothèque **Qt***
- *SQL.*
- *Codage de l'information*

2. Conditions de réalisation

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

3. Ressources

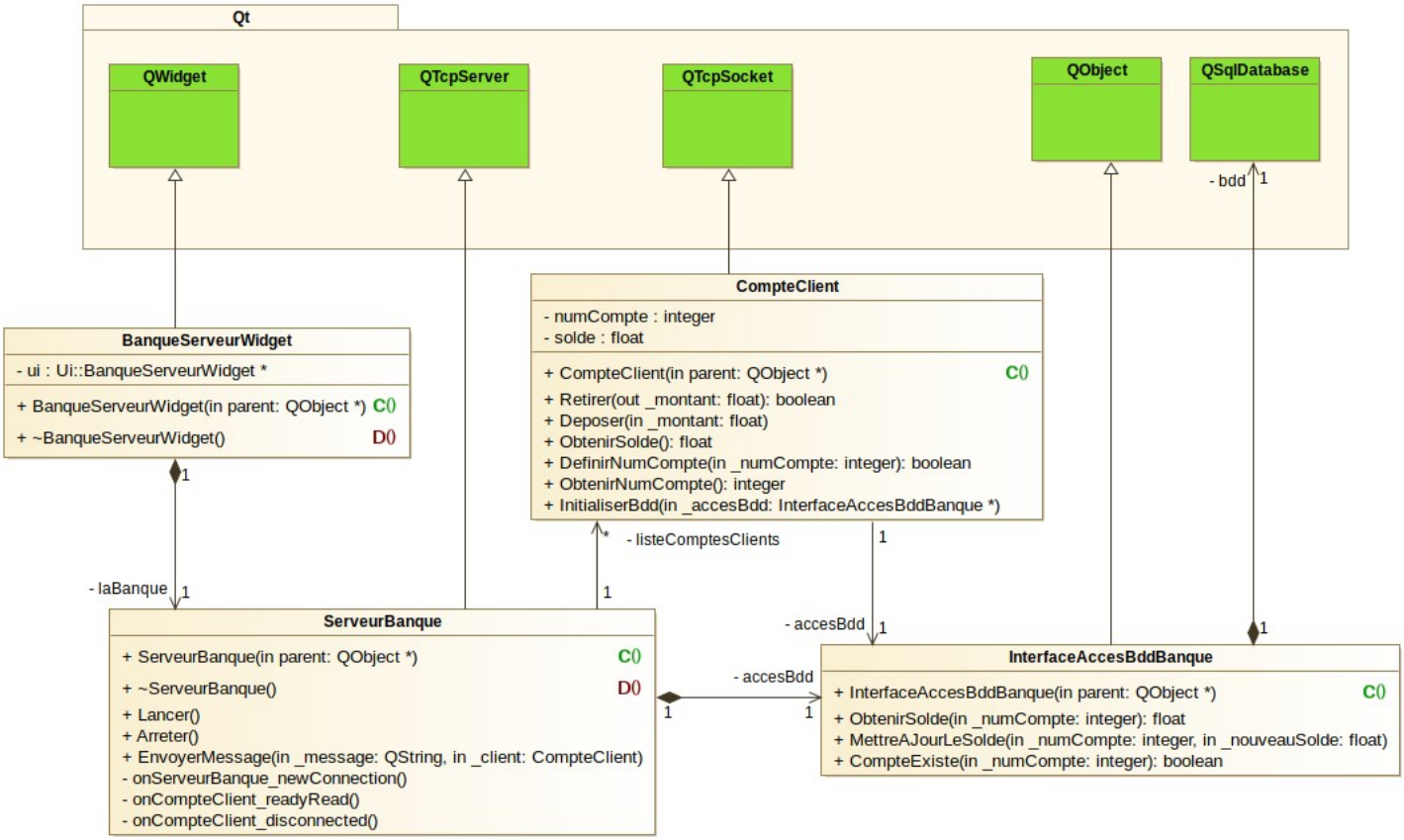
Les classes QSqlDatabase et QSqlQuery dans la technologie QT, consulter les sites : https://doc.qt.io/qt-6/qtsql-index.html https://doc.qt.io/qt-6/sql-programming.html et plus particulièrement

https://doc.qt.io/qt-6/qsqldatabase.html et https://doc.qt.io/qt-6/qsqlquery.html

4. Création du Projet

Dupliquez le projet **ServeurBanque** et renommez le **ServeurBanqueBdd**. Renommez également le fichier .pro en lui donnant le nom de votre projet.

Le diagramme de classes doit subir quelques modifications, il est donné à la figure suivante :



Une nouvelle classe ***InterfaceAccesBddBanque*** apparaît, elle est chargée de toutes les transactions avec la base de données.

Dans la classe ***compteClient*** la méthode ***DefinirNumCompte*** retourne un booléen à vrai si le compte existe dans la base de données, à faux si le numéro de compte n'existe pas. Dans ce dernier cas, il est nécessaire d'utiliser un autre numéro compte, ou, en créer un nouveau avec l'application du **TD1 – Accès à une base de données**.

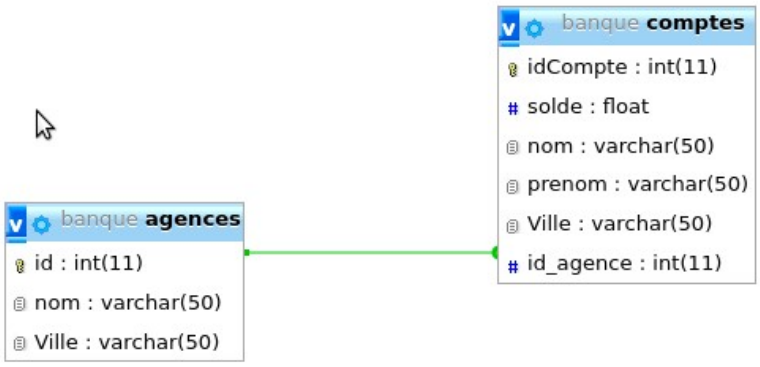
Dans cette même classe, la méthode ***InitialiserBdd*** initialise la relation entre la classe ***CompteClient*** et ***InterfaceAccesBddBanque***, elle est implémentée sous la forme d'un pointeur.

1. Réalisez les modifications demandées dans les différentes classes existantes.
2. Créez le squelette de la classe ***InterfaceAccesBddBanque*** et implémentez la relation entre la classe ***ServeurBanque*** et cette classe de manière dynamique. Le destructeur sera chargé de la libération de la mémoire.
3. Complétez la méthode ***onServeurBanque_newConnection*** de manière à pouvoir initialiser le lien avec la base de données pour chaque nouveau compte client.

5. Accès à la base de données

5.1. Connexion à la base de données

La base de données "**banque**" reste inchangée. Elle contient les deux tables nommées "**comptes**" et "**agences**", pour rappel, voici leurs caractéristiques :



Les informations concernant la connexion à la base de données restent inchangées. Elles sont juste stockées dans le fichier ***ServeurBanqueBdd.ini*** sous la forme :

*ServeurBanqueBdd.ini

```
[BDD]
adresseIp=172.18.58.8
nomDeLaBase=banque
login=snir
motDePasse=snir
```

Le constructeur de la classe ***InterfaceAccesBddBanque*** ouvre l'accès à cette base de données à partir des informations sauvegardées dans le fichier ***ServeurBanqueBdd.ini*** et en cas d'erreur il affiche la nature du problème dans une ***QmessageBox***.

4. Codez le constructeur de la classe ***InterfaceAccesBddBanque***.

5.2. Interaction avec la base de données

Dans toutes les méthodes qui vont suivre, vous aurez besoin d'utiliser les méthodes ***prepare*** , ***bindValue*** et ***exec*** de la classe ***QsqlQuery***.

5. En ligne de commande, réalisez la requête **SQL** permettant de déterminer si un compte existe connaissant son numéro de compte.

Recherche si un compte existe

```
bool InterfaceAccesBddBanque::CompteExiste(const int _numCompte)
{
    QSqlQuery requete;
    bool existe=false;
    // recherche si le compte existe

    if (requete.size()!=0) // si le compte existe on passe existe a vrai
        existe=true;

    return existe;
}
```

6. Complétez le code de la méthode ci-dessus.
7. Codez la méthode ***InterfaceAccesBddBanque::ObtenirSolde***. Elle prend comme la méthode précédente le numéro de compte pour lequel on souhaite obtenir le solde.

Affectation du numéro de compte client si le client est présent dans la Bdd

```
bool CompteClient::DefinirNumCompte(const int _numCompte)
{
    bool retour = true;
    if(accesBdd != nullptr)
    {
        if(!accesBdd->CompteExiste(_numCompte))
            retour = false;
        else
            solde = accesBdd->ObtenirSolde(_numCompte);
        numCompte  = _numCompte;
    }
    return retour;
}
```

8. En utilisant le code ci-dessus pour la méthode ***DefinirNumCompte***, réalisez les modifications nécessaires à la méthode ***onCompteClient_ReadyRead*** pour le cas de l'envoi du numéro de compte 'N'. Ajoutez, si le numéro de compte n'est pas présent dans la base de données, un message annonçant que le compte n'existe pas et referme ensuite la socket de dialogue avec client.
9. En ligne de commande, réalisez la requête SQL permettant de mettre à jour le solde d'un compte connaissant son numéro.
10. Codez la méthode ***InterfaceAccesBddBanque::MettreAJourLeSolde*** à partir du résultat obtenu. Il est toujours nécessaire de vérifier que l'accès à la base de données est bien valide.
11. Complétez le codage des méthodes ***Retirer***, ***Deposer*** et ***ObtenirSolde*** de la classe ***CompteClient*** afin qu'elles fassent appel aux méthodes de la classe ***InterfaceAccesBddBanque***.
12. Ajoutez à la classe ***InterfaceAccesBddBanque*** une méthode ***ObtenirClient*** qui, à partir du numéro de compte passé en paramètre d'entrée, fournit en paramètre de sortie son nom, son prénom et sa ville et en paramètre de retour un booléen vrai si l'opération c'est bien passé, faux sinon.
13. Modifier le code de la méthode ***ServeurBanque::onCompteClient_readyRead*** afin d'obtenir sur l'interface du client pour le compte n° 4 la figure suivante :