Lycée TOUCHARD WASHINGTON

Section des Techniciens Supérieurs
Systèmes Numériques option
Informatique et Réseaux

TD1 - Socket sous Qt

Audit - Côté client

-
- *Date : novembre 2022*
- *Version : 2*
- *Référence : TD1 – Socket sous Qt (Audit_Client).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 serveur est disponible*
  - *Qt-creator*
  -

3. Ressources

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

4. Le besoin

Les techniciens réseau sont souvent appelés pour une défaillance sur un poste informatique. Avant de se déplacer, il serait intéressant de connaître certaines caractéristiques de la machine. Pour cela, un petit programme «**AuditServeur** » implanté sur chaque machine que le technicien par l'intermédiaire du programme «**AuditClient** » pourra interroger est une aide précieuse. Le travail proposé ici permet de définir et de coder ce client.

4.1. Analyse et conception

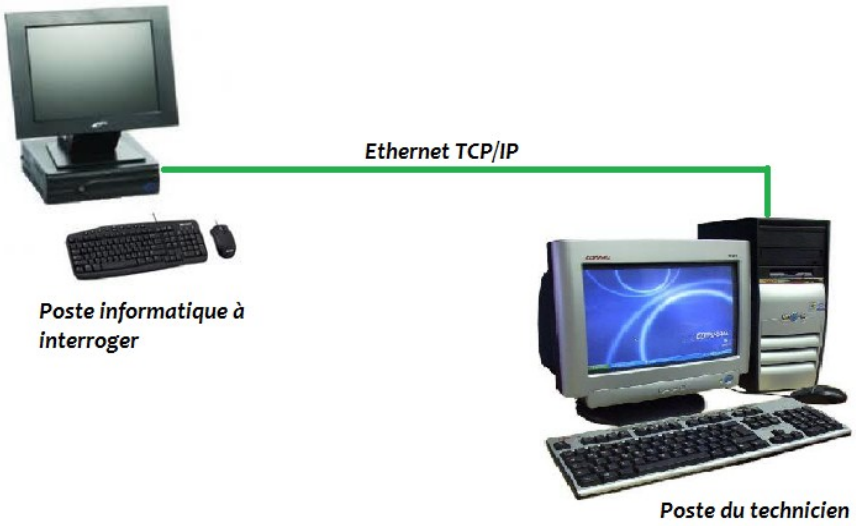
Le serveur est capable de fournir les informations de la machine à l'administrateur client distant.

Selon la demande de celui-ci, le serveur envoie les informations correspondantes :

Demande	*Commande*	*Réponse attendue*
Nom de l'utilisateur	"u"	Le nom de l'utilisateur connecté
Nom de la machine	"c"	Le nom de la machine
Système d'exploitation	"o"	Le type de système d'exploitation
L'architecture du processeur	"a"	Le type de processeur x86 ou amd64 par exemple

D'autres commandes pourront être ajoutées par la suite.

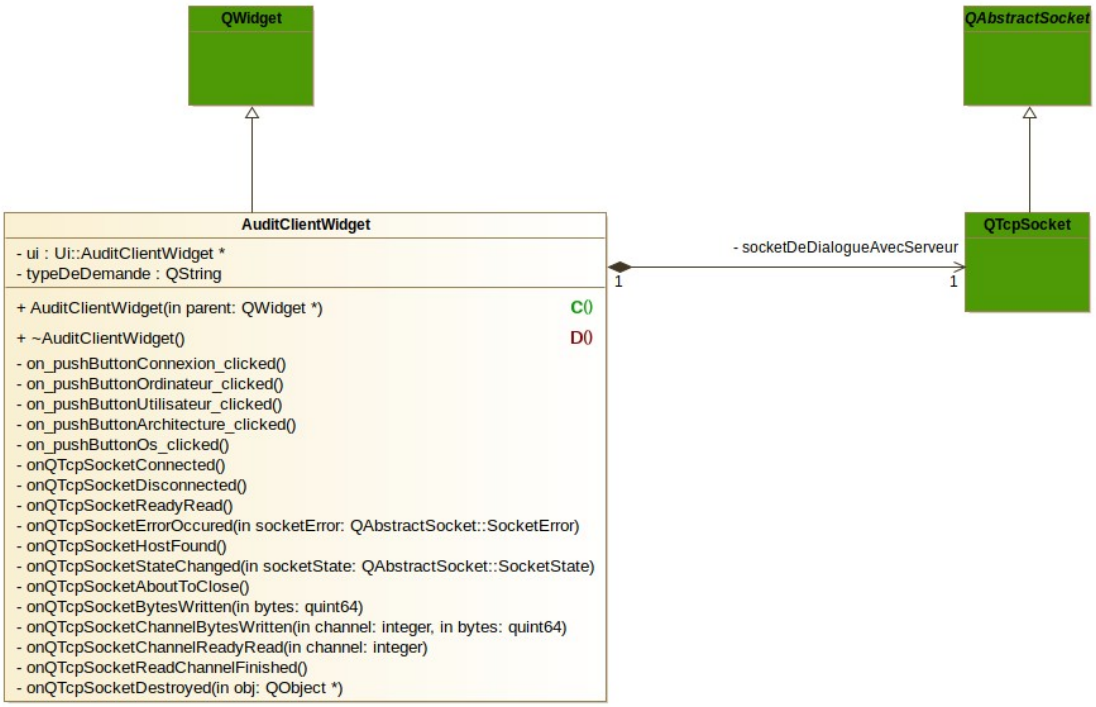
4.2. Mise en situation



5. Réalisation du client

5.1. Création du projet

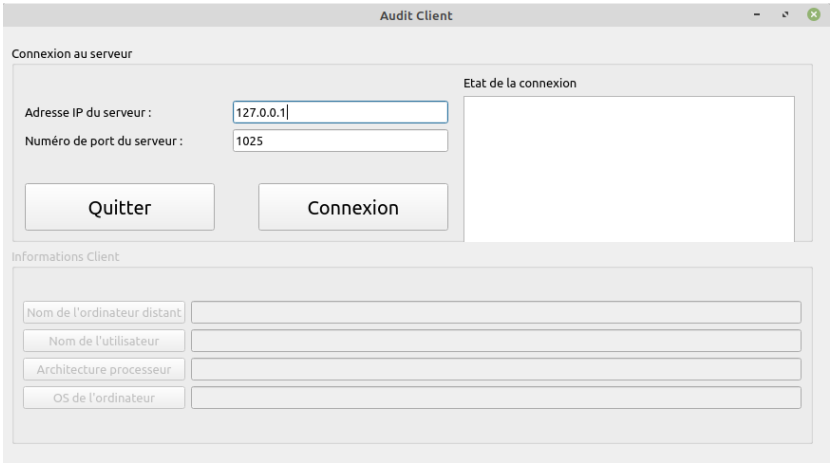
Créez un projet de type Application graphique en C++ sous QT6 avec ***Qt Creator***. La classe principale se nomme ***AuditClientWidget***, elle hérite de ***QWidget*** comme le montre le diagramme de classes ci-dessous.

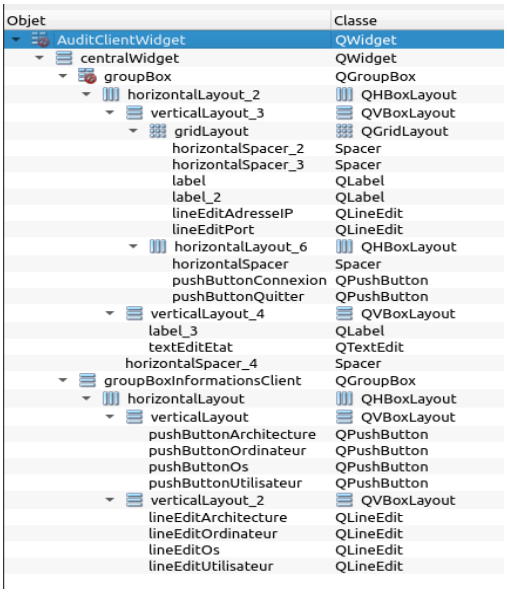


La relation entre la classe **AuditClientWidget** et la classe **QTcpSocket** sera implémentée de manière dynamique. Le moment venu, vous ajouterez, le code nécessaire dans le constructeur et dans le destructeur de la classe.

5.2. Création de l'IHM

L'interface du client aura l'aspect suivant :





L'attribut **enabled** du **groupBoxInformationsClient** est décoché.

1. Respectez les conventions de nommage pour chaque Widget.
2. Associez graphiquement le bouton Quitter au slot ***close()*** de ***Qwidget***.
3. Associez à chaque bouton le slot ***clicked()***
4. Complétez la déclaration de la classe **AuditClientWidget** de manière à correspondre au diagramme de classe.

5.3. Utilisation de la classe QTcpSocket

1. Instanciez la relation ***socketDeDialogueAvecServeur*** comme décrit sous le diagramme de classes.
2. Quel constructeur de ***QTcpSocket*** sera utilisé ?

Après l'instanciation de l'attribut ***socketDeDialogueAvecServeur***, il est nécessaire de lier les signaux de la classe **QTcpSocket** aux slots qui seront implémentés dans la classe **AuditClientWidget.**

3. Pour chacun de ces slots, affichez le message correspondant dans la zone d'édition ***textEditEtat*** contenant l'état de la connexion.

Exemple d'ajout dans la zone d'édition **auditclientwidget.cpp**

```
void AuditClientWidget::onQTcpSocketErrorOccured(QAbstractSocket::SocketError socketError)
{
    Q_UNUSED(socketError); // le paramètre n'est pas utilisé dans cette méthode
    ui->textEditEtat->append(socketDeDialogueAvecServeur->errorString());
}
```

4. Codez le **slot** associé au bouton « Connexion ». Il prend les informations présentes dans les **QLineEdit** correspondant à l'adresse IP du serveur et à son numéro de port pour réaliser la connexion. Il sera également utilisé dans un deuxième temps pour la déconnexion de la socket
5. Codez les **slots** associés aux boutons de la **groupBox** "informations client".

Voici le code pour le système d'exploitation :

Exemple pour le système d'exploitation **auditclientwidget.cpp**

```
void AuditClientWidget::on_pushButtonOs_clicked()
{
    typeDeDemande="o";
    socketDeDialogueAvecServeur->write(typeDeDemande.toLatin1());
}
```

6. Codez le slot de réception des données **onQTcpSocketReadyRead**.

Vous utiliserez la méthode **readAll** pour lire les données en provenance du serveur.

La méthode **data** de la classe **QByteArray** pourrait vous être utile pour mettre les informations dans les **QlineEdit**.

6. Approfondissement

Complétez votre code afin que:

7. Le **groupBox** "informations client" ne soit accessible que si l'on est connecté à un serveur.
8. Le texte du bouton de connexion se change en "Déconnexion" si l'on est connecté à un serveur.
9. Si le texte du bouton de connexion est "Déconnexion" et que l'on clique dessus, on se déconnecte du serveur et le texte du bouton redevient "connexion"
10. Les champs IP et port ne sont accessibles que si l'on n'est pas connecté à un serveur. Les champs IP et port doivent redevenir accessibles lors de la déconnexion du serveur.