

# Interface Homme Machine sous Qt

---

## *Compte Bancaire*

### *~ TD6 - Tutoriel - IHM-Banque Qt ~*

Version 1.0 – janvier 2019

## Objectif

- Codage C++
  - IHM Qt
  - QWidget
  - QPushButton
  - QLineEdit
  - QListWidget
  - QListWidgetItem
  - QMessageBox
  - QString
  - QDateTime

## Conditions de réalisation

Ressources utilisées :

### **Matériel**

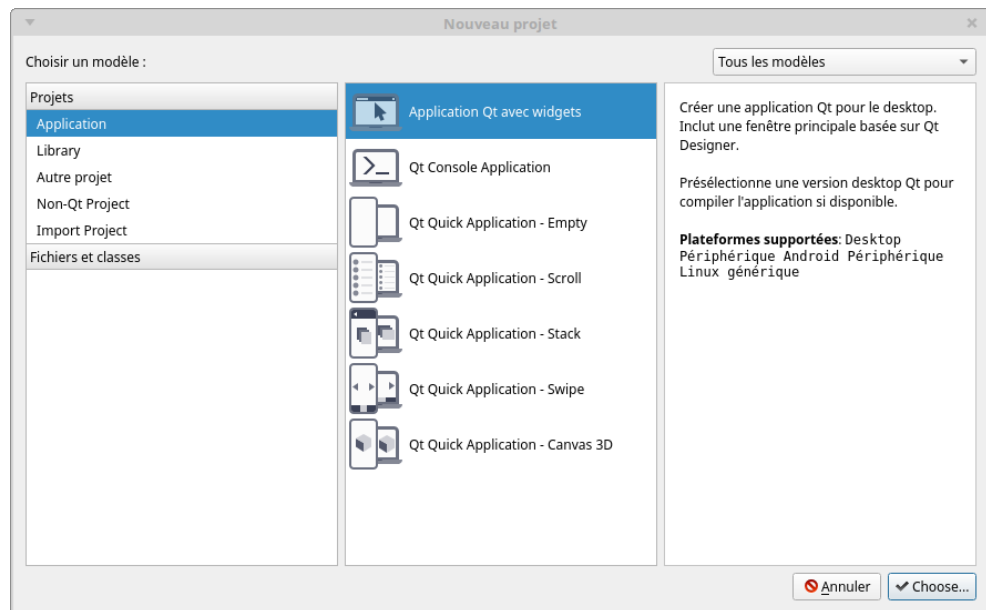
Un PC sous Linux

### **Logiciel**

QT creator

## 1. CRÉATION DU PROJET

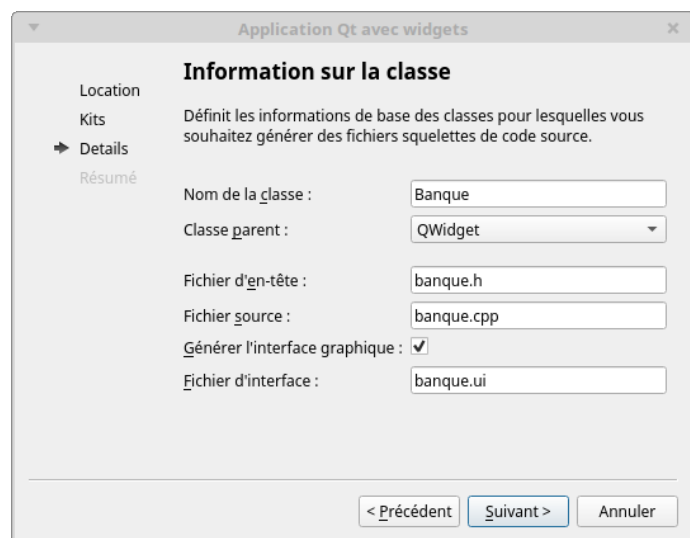
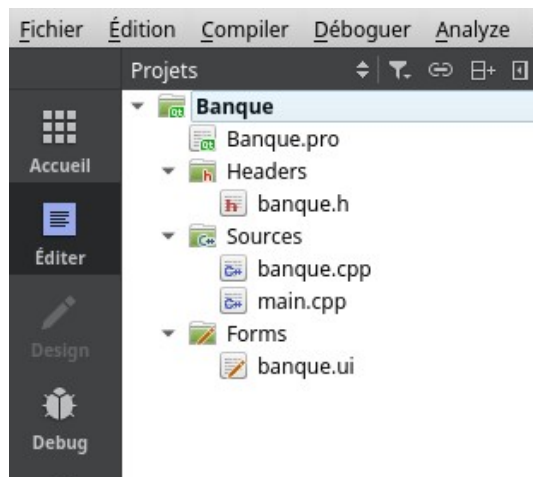
Après avoir lancé **Qt creator**, choisir dans le menu **fichier** l'option **Nouveau fichier ou projet**. Choisir ensuite une **Application Qt avec widgets** comme le montre la figure.



Ce projet se nomme **Banque**, il est placé sur le disque de travail dans votre dossier **Projet\_Qt**.

Dans un premier temps, réaliser la classe **Banque** dont le parent sera de type **QWidget**. (relation d'héritage)

L'EDI fabrique les fichiers suivants :



Le fichier **Banque.pro** est fabriqué automatiquement pour l'instant, il n'est pas à modifier.

<b>Banque.pro</b>	Contient la structure du projet
<b>banque.h</b>	Déclaration de la classe Banque
<b>banque.ui</b>	Contient l'interface utilisateur
<b>banque.cpp</b>	Code de la classe Banque
<b>main.cpp</b>	Programme principal

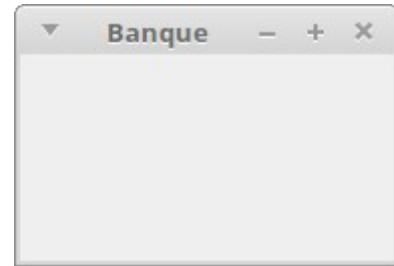
Après exécution, le résultat obtenu est la boîte de dialogue si contre :



*L'icône compile et lance le programme.*



*L'icône effectue uniquement la compilation .*



```
main.cpp
#include "banque.h"
#include <QApplication>

int main(int argc, char *argv[])
{
    QApplication a(argc, argv);
    Banque w;
    w.show();

    return a.exec();
}
```

Vous n'avez pas à intervenir dans le fichier **main.cpp**. Son rôle est de déclarer une instance de la classe Banque, d'afficher l'interface utilisateur associée puis d'exécuter l'application.

```
banque.h
#ifndef BANQUE_H
#define BANQUE_H

#include <QWidget>

namespace Ui {
class Banque;
}

class Banque : public QWidget
{
    Q_OBJECT

public:
    explicit Banque(QWidget *parent = 0);
    ~Banque();

private:
    Ui::Banque *ui;
};

#endif // BANQUE_H
```

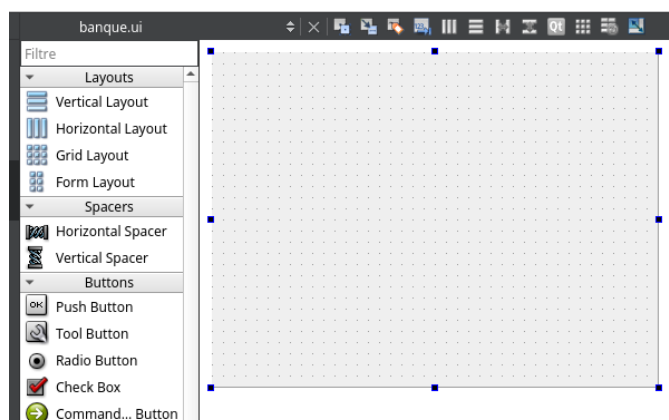
Le pointeur **ui** représente l'accès aux éléments de l'interface homme-machine.

```
banque.cpp
#include "banque.h"
#include "ui_banque.h"

Banque::Banque(QWidget *parent) :
    QWidget(parent),
    ui(new Ui::Banque)
{
    ui->setupUi(this);
}

Banque::~Banque()
{
    delete ui;
}
```

Première instruction du constructeur, initialise l'interface homme-machine

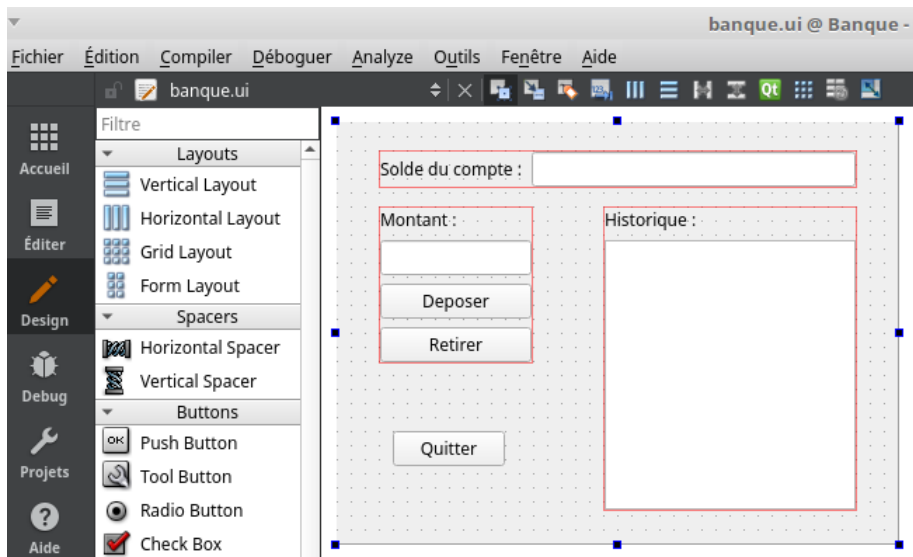


Un double clic sur le fichier **banque.ui** lance la partie design et permet de fabriquer l'interface graphique.

La première partie de l'écran contient les Widgets à déposer sur la grille de la partie centrale. Les propriétés de chaque élément.

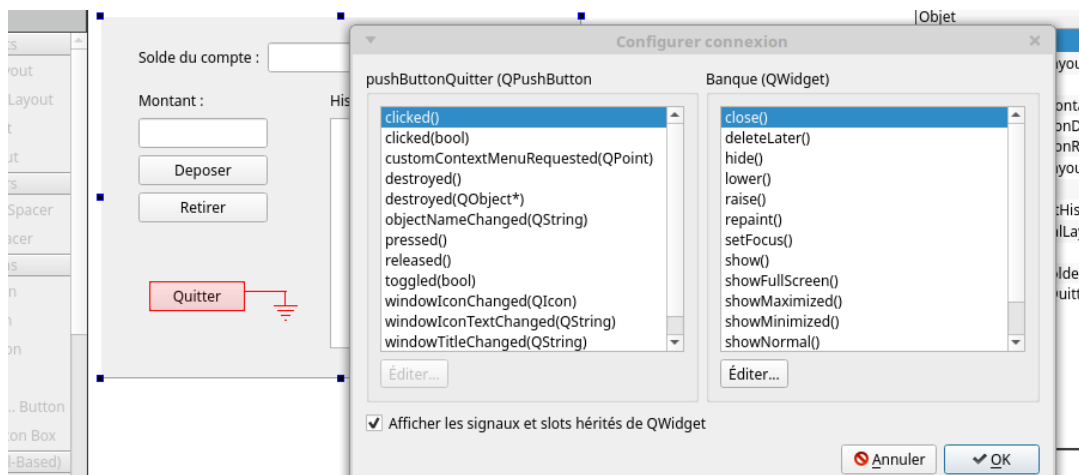
## 2. CRÉATION DE L'IHM

À l'aide du designer, réalisez l'interface homme-machine suivante en respectant les types de widget et en utilisant des noms significatifs comme le montre le tableau des objets.

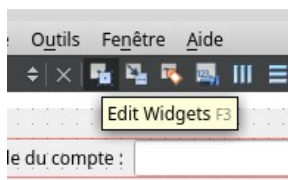


Objet	Classe
Banque	QWidget
verticalLayout	QVBoxLayout
label_3	QLabel
lineEditMontant	QLineEdit
pushButtonDeposer	QPushButton
pushButtonRetirer	QPushButton
verticalLayout_2	QVBoxLayout
label_2	QLabel
listWidgetHistorique	QListWidget
horizontalLayout	QHBoxLayout
label	QLabel
lineEditSolde	QLineEdit
pushButtonQuitter	QPushButton

À l'aide de l'icône « **Edit Signals/slots** » ou la touche F4 connectez le signal **clicked()** du bouton **Quitter** au slot **close()** hérité de **QWidget**.

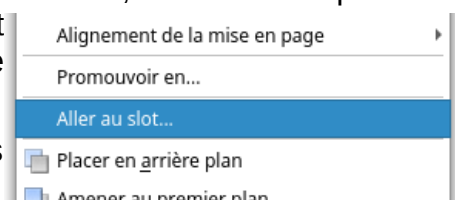


Exécuter le programme, la fenêtre apparaît l'action sur le bouton **Quitter** permet de fermer la boîte de dialogue et terminer le programme.



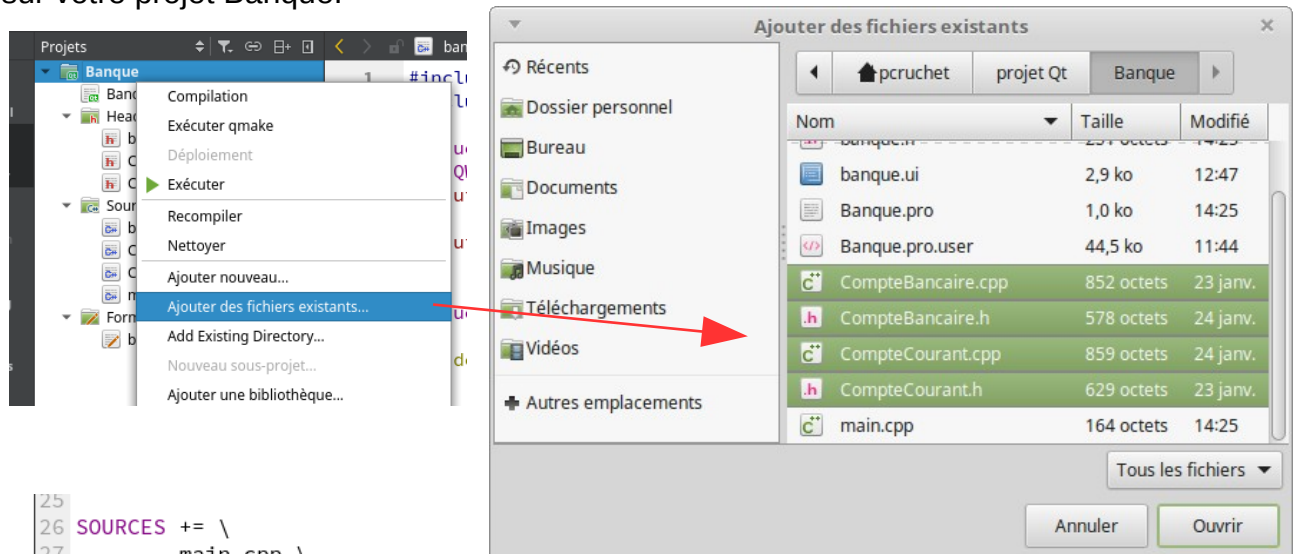
Pour la suite revenir au mode « **Edit Widgets** » ou la touche F3. Un clic droit sur chacun des boutons Déposer et Retirer permet de faire apparaître une boîte menu, sélectionnez pour chacun d'eux, « **Aller au slot...** ». et sélectionnez le signal **clicked()** de la classe **QabstractButton** dont **QPushButton** a hérité.

Deux nouvelles méthodes apparaissent dans les fichiers **banque.h** et **banque.cpp**.



### 3. CODAGE C++

Recopiez dans votre projet les fichiers relatifs aux classes **CompteBancaire** et **CompteCourant**, puis ajoutez les fichiers existants à votre projet avec un clic droit sur votre projet Banque.



```

25 SOURCES += \
26     main.cpp \
27     banque.cpp \
28     CompteBancaire.cpp \
29     CompteCourant.cpp
30
31 HEADERS += \
32     banque.h \
33     CompteBancaire.h \
34     CompteCourant.h
35
36 FORMS += \
37     banque.ui
38
39

```

Le fichier **Banque.pro** s'est enrichi avec les nouveaux noms de fichiers.

Dans le fichier Banque.h, ajoutez un attribut **compte** de type **CompteCourant** dans la section privée de la classe Banque.

La touche **F4** fait basculer alternativement du fichier **.h** au fichier **.cpp**.

```

#ifndef BANQUE_H
#define BANQUE_H

#include <QWidget>
#include "CompteCourant.h"

namespace Ui {
class Banque;
}

class Banque : public QWidget
{
    Q_OBJECT

public:
    explicit Banque(QWidget *parent = 0);
    ~Banque();

private slots:
    void on_pushButtonDeposer_clicked();
    void on_pushButtonRetirer_clicked();

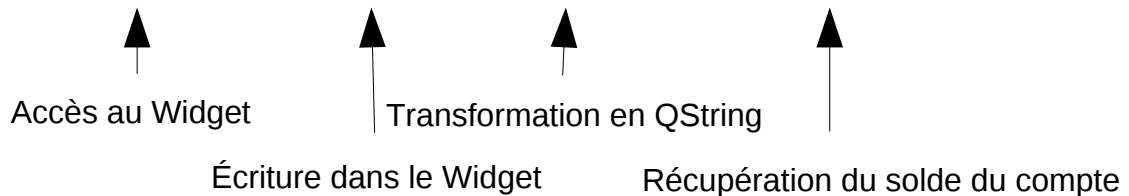
private:
    Ui::Banque *ui;
    CompteCourant compte;
};

#endif // BANQUE_H

```

Dans le constructeur de la classe Banque, ajoutez la ligne suivante pour faire afficher le solde du compte :

```
Banque::Banque(QWidget *parent) :
    QWidget(parent),
    ui(new Ui::Banque)
{
    ui->setupUi(this);
    ui->lineEditSolde->setText(QString::number(compte.ObtenirSolde()));
}
```



Pour déposer une somme sur le compte bancaire, complétez la méthode :

```
void Banque::on_pushButtonDeposer_clicked()
{
    double montant = ui->lineEditMontant->text().toDouble();
    compte.Deposer(montant);
    ui->lineEditSolde->setText(QString::number(compte.ObtenirSolde()));
}
```

La première ligne récupère le texte contenu dans **lineEditMontant** et la convertie en double.

Pour retirer une somme du compte bancaire, l'opération est similaire. Il faut simplement tester que le retrait a bien été effectué. Si ce n'est pas le cas, on affiche une boîte de message d'alerte :

```
void Banque::on_pushButtonRetirer_clicked()
{
    double montant = ui->lineEditMontant->text().toDouble();
    if(!compte.Retirer(montant))
    {
        QMessageBox message(QMessageBox::Warning,
                            "Alerte de la banque",
                            "Votre solde est insuffisant");
        message.exec();
    }
    else
    {
        ui->lineEditSolde->setText(QString::number(compte.ObtenirSolde()));
    }
}
```

icône

Titre

Message

Exécution de la boite de dialogue

Il est nécessaire d'inclure `<QMessageBox>`

Pour réaliser l'historique des opérations, il est nécessaire d'ajouter le code suivant dans la méthode `void Banque::on_pushButtonDeposer_clicked()`

```
QDateTime dateTime = QDateTime::currentDateTime();
QString texte = "+" + QString::number(montant) + " € ";
texte += dateTime.toString("ddd d MMMM yyyy hh:mm");
ui->listWidgetHistorique->addItem(texte);
```

À partir de liens suivants, expliquez le code ci-dessus :

<http://doc.qt.io/qt-5/qdatetime.html#toString>

<http://doc.qt.io/qt-5/qdatetime.html#currentDateTime>

Ajout dans le QListWidget

Pour le retrait, on souhaite afficher de la même manière avec un signe - à la place du signe +. La construction de la chaîne reste identique.

Maintenant, on souhaite afficher en rouge ce retrait dans l'historique, l'insertion dans la liste est modifiée de la manière suivante :

```
QListWidgetItem *item = new QListWidgetItem(texte);
item->setForeground(Qt::red);
ui->listWidgetHistorique->addItem(item);
```

Ajoutez les éléments nécessaires pour modifier le découvert autorisé.