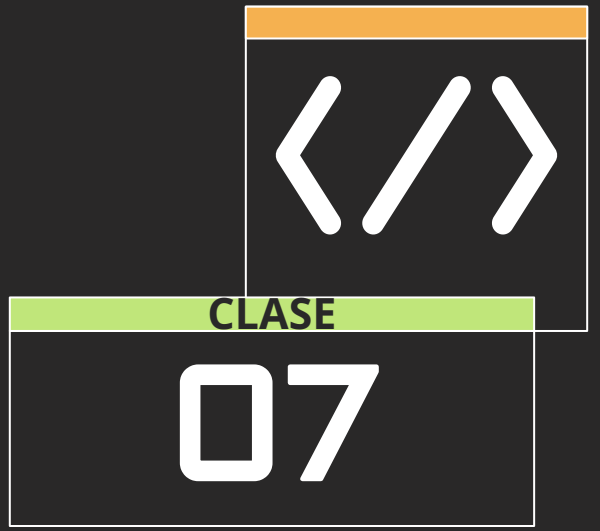
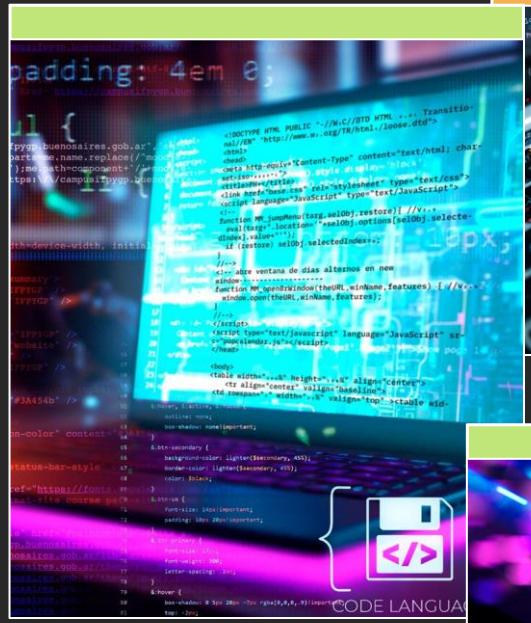
Programacion II [PRACTICA]



1 0 0 1 0 1 0 1 0 1 0 1 0 1 0 0

1 0 1 0 0 1 0 1 0 1 0 0 1 0 1 0

INTERFAZ GRAFICA DE USUARIOS (G.U.I)



1 0 1 0 1 0 1 1 1 0 0 0 1 1 1 0

1 1 0 0 1 0 1 0 1 0 1 0 1 1 1 0

Signals & Slot

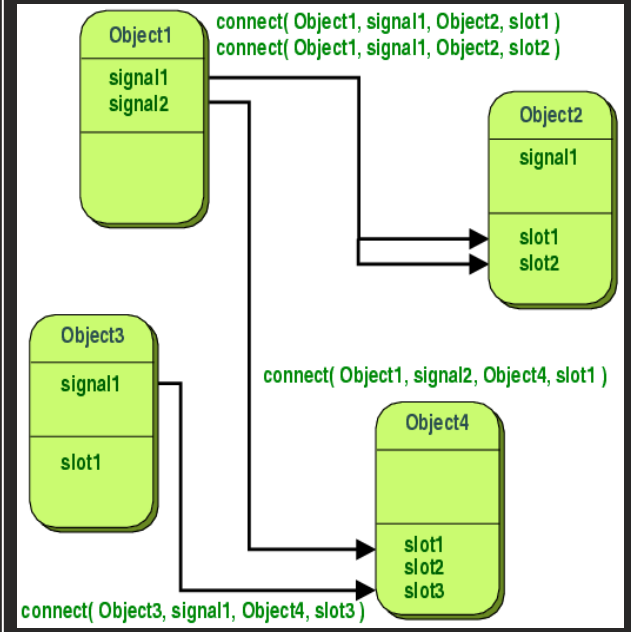
Qt, crea una forma dinámica de comunicar eventos con los cambios de estado que estos provocan y las reacciones de los mismos.

Todo objeto derivado de QObject puede poseer dos tipos de funciones propias especiales:

- **signals.** funciones que permiten emitir una señal cuando hay algún cambio de estado en el objeto al que pertenecen.

- **slots.** funciones que son el final de la conexión, y que ejecutan una serie de acciones una vez reciben el mensaje de la señal.

Signals & Slot



Una señal puede conectarse a más de un slot, para llevar a cabo diferentes actividades. También varios señales diferentes pueden conectarse con un mismo slot, y tendríamos una actividad que puede ser desatada de varias formas diferentes.

Veamos como es la forma del método connect:

```
bool QObject::connect ( const QObject *sender, SIGNAL(*signal), const QObject *receiver, SLOT(*slot) )
```

Usamos las macros SIGNAL y SLOT para envolver a las funciones signal y slot con los tipos de sus parámetros.

QMessageBox

Es una ventana modal que presenta al usuario una pregunta, y recibe una respuesta. Las hay de 4 tipos:

- Warning
- Question
- Information
- Critical

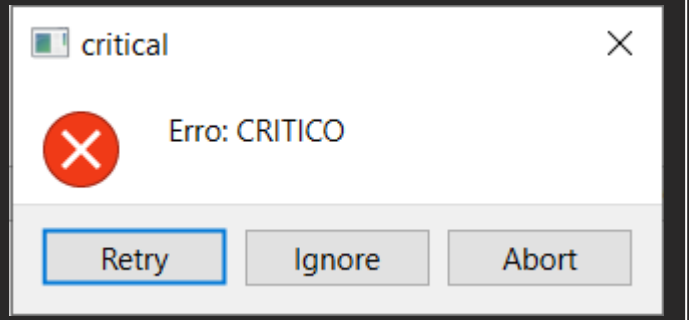
QMessageBox::critical

StandardButton	**critical**(QWidget * *parent*, const QString & *title*, const QString & *text*, StandardButtons *buttons* = Ok, StandardButton *defaultButton* = NoButton)

```
 int devolucion;
 devolucion = QMessageBox::critical (this,"critical", "Erro: CRITICO", QMessageBox::Abort
|QMessageBox::Retry | QMessageBox::Ignore);

    qDebug() <<"reply: "<<devolucion;

// codigo de Abort
    if (devolucion == QMessageBox::Abort) qDebug ("ABORTAR");
// codigo de Retry
else if (devolucion == QMessageBox::Retry) qDebug ("Retry");
 else qDebug ("IGNORAMOS");// codigo de Ignore
```

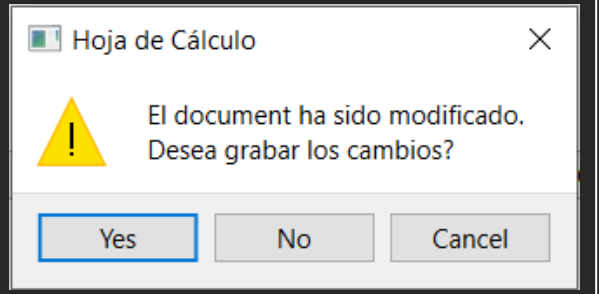


QMessageBox::warning

StandardButton warning(QWidget * parent, const QString & title, const QString & text, StandardButtons buttons = Ok, StandardButton defaultButton = NoButton)

```
QMessageBox::warning(this, tr(“Hoja de Cálculo"), tr(“El document ha sido modificado.\n“ “Desea guardar los cambios?"), QMessageBox::Yes | QMessageBox::Default, QMessageBox::No, QMessageBox::Cancel | MessageBox::Escape);
```

```
Int devolucion = QMessageBox::warning(this, tr("Hoja de Cálculo"), tr("El document ha sido modificado.\ n" "Desea grabar los cambios?"), QMessageBox::Yes | QMessageBox::Default, QMessageBox::No, QMessageBox::Cancel | QMessageBox::Escape);
qDebug() <<"devolucion: "<<devolucion;
if(devolucion==QMessageBox::Yes)qDebug() <<"SI";
if(devolucion==QMessageBox::No)qDebug() <<"No";
if(devolucion==QMessageBox::Cancel)qDebug() <<"Cancel";
if(devolucion==QMessageBox::Escape)qDebug() <<"Escape";
```



QMessageBox::question

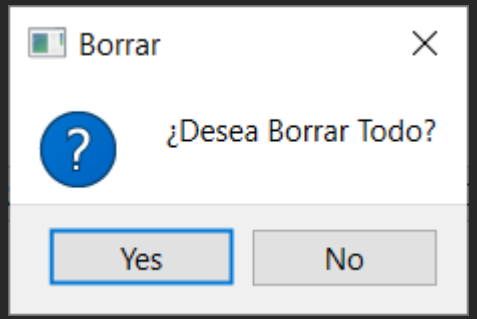
StandardButton	**question**(QWidget * *parent*, const QString & *title*, const QString & *text*, StandardButtons *buttons* = StandardButtons(Yes | No), StandardButton *defaultButton* = NoButton)

```
QMessageBox::question(this, tr("Borrar CD"), tr("Borrado\"%1\" y todas las pistas?").arg(record.value(CdArtistId).toString()), QMessageBox::Yes | QMessageBox::Default, QMessageBox::No, QMessageBox::Cancel | MessageBox::Escape);
```

```
Int devolucion = QMessageBox::question(this, tr("Borrar"), tr("¿Desea Borrar Todo?"), QMessageBox::Yes | QMessageBox::Default, QMessageBox::No | QMessageBox::Escape);

if (devolucion == QMessageBox::No) qDebug() <<"No";

 if (devolucion == QMessageBox::Yes) qDebug() <<"Yes";
```

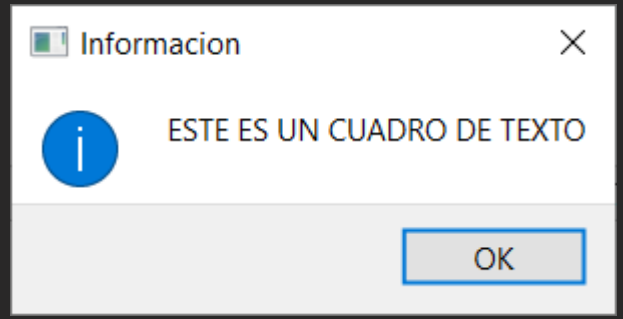


QMessageBox::information

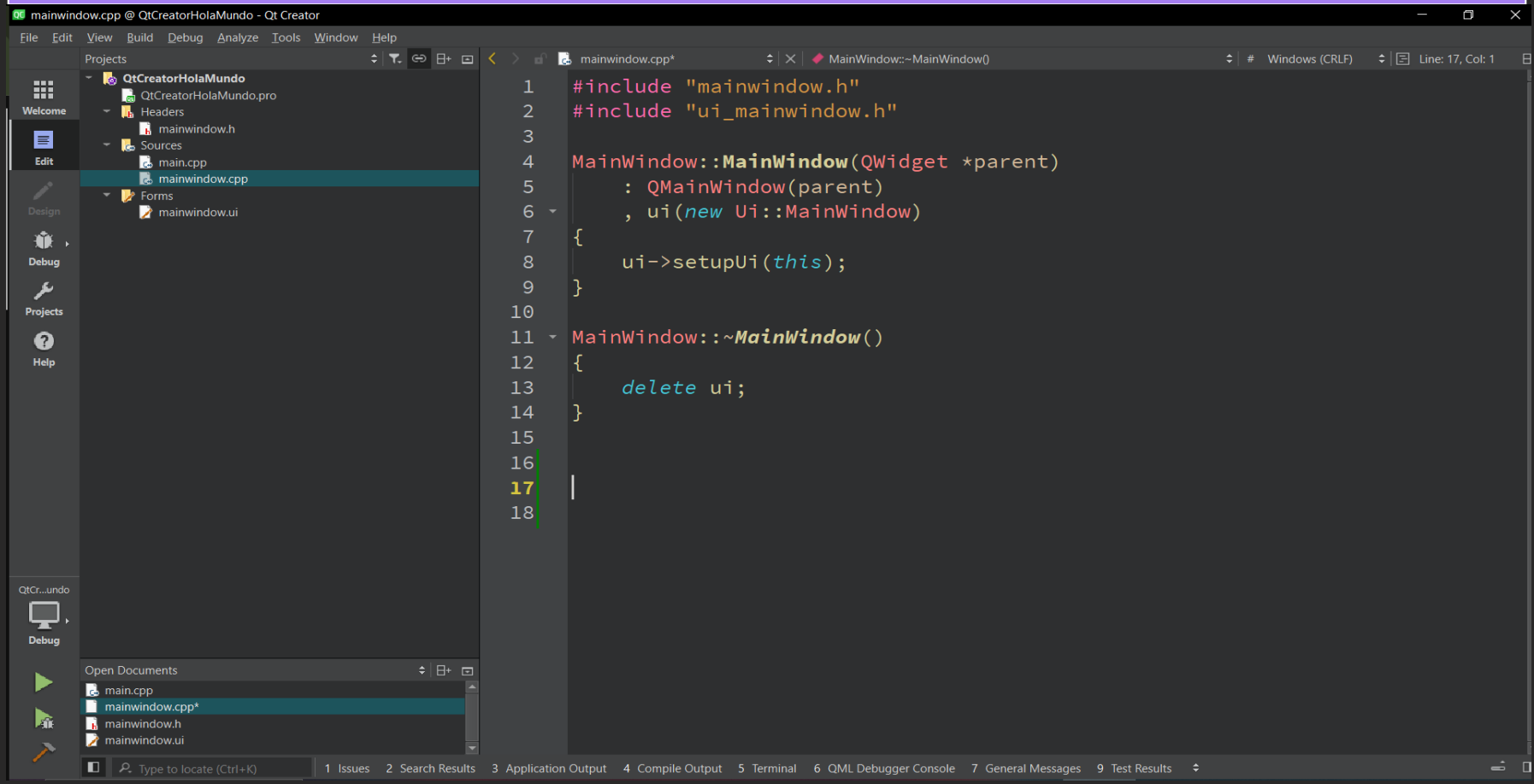
StandardButton

information(QWidget * parent, const QString & title, const QString & text, StandardButtons buttons = Ok, StandardButton defaultButton = NoButton)

```
QMessageBox::information (this, tr("Informacion"),tr("ESTE ES UN CUADRO DE TEXTO"),QMessageBox::Ok,QMessageBox::Ok);
```

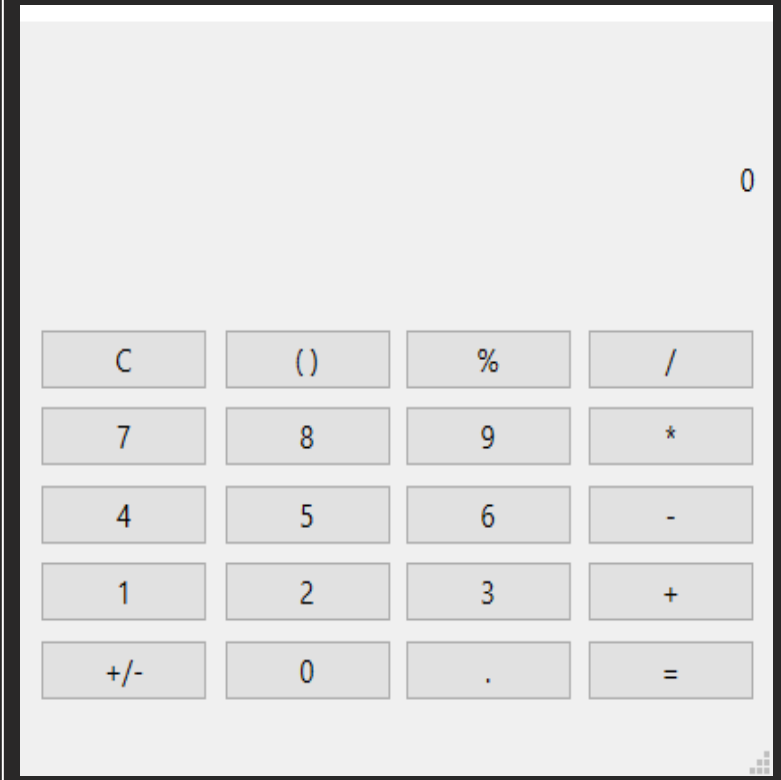


<< A PROGRAMAR >> ●●●


mainwindow.cpp @ QtCreatorHolaMundo - Qt Creator
File Edit View Build Debug Analyze Tools Window Help
Projects
QtCreatorHolaMundo
QtCreatorHolaMundo.pro
Headers
mainwindow.h
Sources
main.cpp
mainwindow.cpp
Forms
mainwindow.ui
Welcome
Edit
Design
Debug
Projects
Help
QtCr...undo
Debug
Open Documents
main.cpp
mainwindow.cpp*
mainwindow.h
mainwindow.ui
mainwindow.cpp*
MainWindow::~MainWindow()
Windows (CRLF)
Line: 17, Col: 1
1 #include "mainwindow.h"
2 #include "ui_mainwindow.h"
3
4 MainWindow::MainWindow(QWidget *parent)
5 : QMainWindow(parent)
6 , ui(new Ui::MainWindow)
7 {
8 ui->setupUi(this);
9 }
10
11 MainWindow::~MainWindow()
12 {
13 delete ui;
14 }
15
16
17
18
Type to locate (Ctrl+K)
1 Issues 2 Search Results 3 Application Output 4 Compile Output 5 Terminal 6 QML Debugger Console 7 General Messages 9 Test Results

Retos de Programacion

01 Calculadora (Front)



Teniendo el Front realizado, Desarrollar todas las operaciones de una calculadora sensilla.

1
0
1
0
0
1
0
0
1
0
1
1
0
1
1
1

02 VARIOS

1. Generar todos los tipo de QMessageBox, y ver las devolucion en in valor int de sus botones.

2. Usando Signal y Slot (Connect) con un boton abrir algunas de las QMessageBox (Critical,Informacion,Adevertencia,Pregunta)

3. Usando Signal & Slot (connect) poner tres botones en el front usando el layout horizontal (uno debajo del otro) y cambiar a 3 colores diferentes el fondo de la ventana Widget

4. Usando Signal & Slot (connect) poner tres botones en el front y un Qlabel, uno de los botones generar un mensaje, otro limpiara el Qlabel y el ultimo mostrara otro mensaje.