QT Toolkit
Qt este un framework pentru crerea de aplicatii cross-platform (acelasi cod pentru diferite sisteme de operare, dispozitive) in C++.
Folosind Qt putem crea interfete grafice utilizator. Codul odata scris poate fi rulat pe mai multe platforme fara a necesita modificari in codul sursa.

Qt suporta multiple platforme de 32/64-bit
- Windows
- Linux
- Mac OS
- Mobile / Embedded

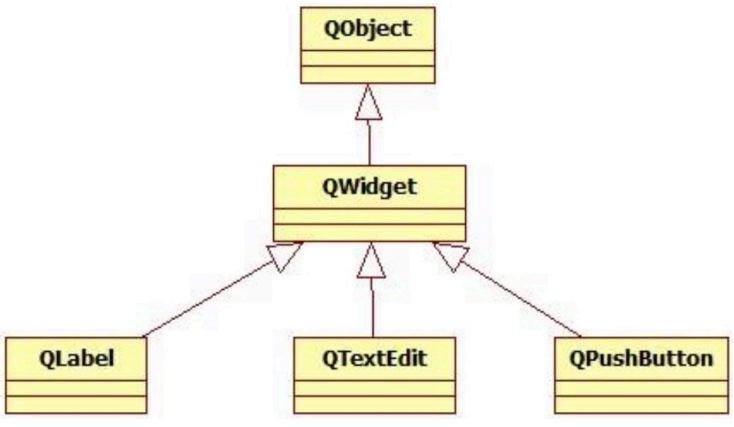
Este o librarie din C++ dar exista posibilitatea de a fi folosit si din alte limbaje. Exemple de aplicatii create folosind Qt : Google Earth, Adobe Photoshop.

QT - Module si utilitare
- **Qt Library** - biblioteca de clase C++, ofera clasele necesare pentru a crea aplicatii.
- **Qt Creator** - mediu de dezvoltare integrat (IDE) pentru a crea aplicatii folosind Qt.
- **Qt Designer** - instrument de creare de interfete grafice utilizator folosind componente Qt.
- **Qt Assistant** - aplicatie ce contine documentatie pentru Qt si faciliteaza accesul la documentatiile diferitelor parti din Qt.
- **Qt Linguist** - suport pentru aplicatii care functioneaza in diferite limbi.

QApplication
Clasa QApplication gestioneaza fluxul de evenimente si setarile generale pentru aplicatiile Qt cu interfata grafica GUI.
QApplication preia evenimentele de la sistemul de operare si distribuie catre Qt, toate evenimentele ce provin de la sistemul de ferestre.
Pentru orice aplicatie Qt cu GUI exista un obiect QApplication, acesta avand responsabilitatile :
- Initializeaza aplicatia conform setarilor sistem.
- Gestioneaza fluxul de evenimente - generate de sistemul de ferestre si distribuite catre componentele grafice Qt (windgets).
- Are referinte catre ferestrele aplicatiei.
- Defineste look and feel.

app.exec(); - porneste procesarea de evenimente din QApplication (event loop). In timp ce ruleaza aplicatia evenimente sunt generate si trimise catre componentele grafice.

Componente grafice QT (widgets)
Sunt elementele de baza folosite pentru a construi interfete grafice utilizator : butoane, etichete, casute de text, etc.
Orice componenta grafica Qt (widget) poate fi adaugata pe o fereastra sau deschisa independent intr-o fereastra separata.



Daca componenta nu este adaugata intr-o componenta parinte avem de fapt o fereastra separata. Ferestrele separa vizual aplicatiile intre ele si sunt decorate cu diferite elemente (bara de titlu, instrumente de pozitionare, redimensionare)

Widget : Eticheta, buton, casuta de text, lista

QLabel

- QLabel este folosit pentru a prezenta un text sau o imagine. Nu ofera interactiune cu utilizatorul.
- QLabel este folosit de obicei ca si o eticheta pentru o componenta interactiva. QLabel ofera mecanism de mnemonic, o scurtatura prin care se seteaza focusul pe componenta atasata.

```
QLabel *label = new QLabel("hello world");
label->show();

QLineEdit txt(parent);
QLabel lblName("&Name:", parent);
lblName.setBuddy(&txt);
```

QPushButton

QPushButton widget - buton

- Se apasa butonul (click) pentru a efectua o operatie.
- Butonul are un text si optionar o iconita. Poate fi specificat si o tasta rapida (shortcut) folosind caracterul & in text.

```
QPushButton btn("&TestBTN");
btn.show();
```

QLineEdit

- Casuta de text (o singura linie)
- Permite utilizatorului sa introduca informatii. Ofera functii de editare (undo, redo, cut, paste, drag and drop)

```
QLineEdit txtName;
txtName.show();
```

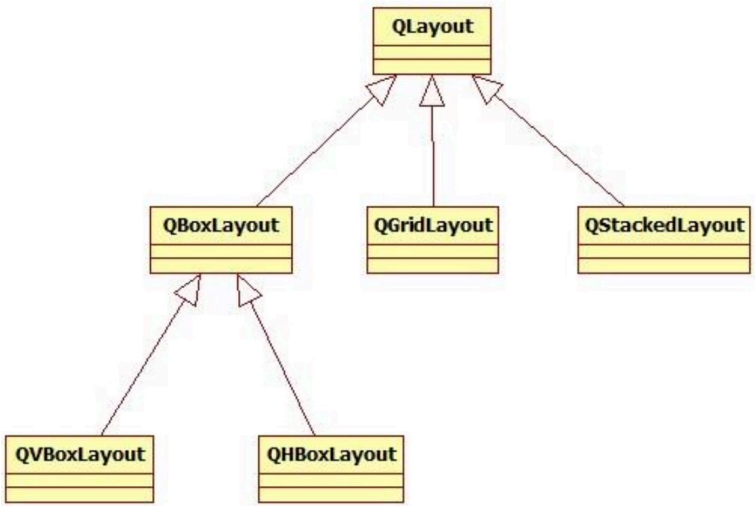
QListWidget

Prezinta o lista de elemente.

```
QListWidget *list = new QListWidget;
new QListWidgetItem("Item 1", list);
new QListWidgetItem("Item 2", list);
QListWidgetItem *item3 = new QListWidgetItem("Item 3");
list->insertItem(0, item3);
list->show();
```

Layout management

- Orice QWidget are o componenta parinte.
- Sistemul Qt layout ofera o metoda de a aranja automat componentele pe interfata.
- Qt include un set de clase pentru layout management, aceste clase ofera diferite strategii de aranjare automata a componentelor.
- Componentele sunt pozitionate / redimensionate automat conform strategiei implementate de layout manager si luand in considerare spatiul disponibil pe ecran. Folosind diferite layouturi putem crea interfete grafice utilizator care acomodeaza diferite dimensiuni ale ferestrei.



Layout management

```
QWidget *wnd = new QWidget;
QHBoxLayout *hLay = new QHBoxLayout();
QPushButton *btn1 = new QPushButton("Bt &1");
QPushButton *btn2 = new QPushButton("Bt &2");
QPushButton *btn3 = new QPushButton("Bt &3");
hLay->addWidget(btn1);
hLay->addWidget(btn2);
hLay->addWidget(btn3);
wnd->setLayout(hLay);
wnd->show();
```

```
QWidget *wnd2 = new QWidget;
QVBoxLayout *vLay = new QVBoxLayout();
QPushButton *btnn1=new QPushButton("B&1");
QPushButton *btnn2= new QPushButton("B&2");
QPushButton *btnn3= new QPushButton("B&3");
vLay->addWidget(btnn1);
vLay->addWidget(btnn2);
vLay->addWidget(btnn3);
wnd2->setLayout(vLay);
wnd2->show();
```

GUI putem compune multiple componente care folosesc diferite strategii de aranjare pentru a crea interfata utilizator dorita.

Layout management

addStretch() se foloseste pentru a consuma spatiu. Practic se adauga un spatiu care redimensioneaza in functie de strategia de aranjare.

```
QHBoxLayout* btnsL = new QHBoxLayout;
btns->setLayout(btnsL);
QPushButton* store = new QPushButton("&Store");
btnsL->addWidget(store);
btnsL->addStretch();
QPushButton* close = new QPushButton("&Close");
btnsL->addWidget(close);
```

Layout management

Cum construim interfetele grafice :

- Cream componentele necesare.
- Setam proprietatile componentelor daca este necesar.
- Adaugam componenta la un layout (layout manager se ocupa cu dimensiunea si pozitia componentelor).
- Conectam componentele intre ele folosind mecanismul de signal si slot.

Avantaje :

- Ofera un comportament consistent indiferent de dimensiunea ecranului/ferestrei, se ocupa de rearanjarea componentelor in caz de redimensionare a componentei.
- Seteaza valori implicite pentru componentele adaugate.
- Se adapteaza in functie de fonturi si alte setari sistem legate de interfetele utilizator.
- Daca adaugam/stergem componente restul componentelor sunt rearanjate automat.

Pozitionare cu coordonate absolute

```
/**
 * Create GUI using absolute positioning
 */
void createAbsolute() {
	QWidget* main = new QWidget();
	QLabel* lbl = new QLabel("Name:", main);
	lbl->setGeometry(10, 10, 40, 20);
	QLineEdit* txt = new QLineEdit(main);
	txt->setGeometry(60, 10, 100, 20);
	main->show();
	main->setWindowTitle("Absolute");
}
/**
 * Create the same GUI using form layout
 */
void createWithLayout() {
	QWidget* main = new QWidget();
	QFormLayout *fL = new QFormLayout(main);
	QLabel* lbl = new QLabel("Name:", main);
	QLineEdit* txt = new QLineEdit(main);
	fL->addRow(lbl, txt);
	main->show();
	main->setWindowTitle("Layout");
	//fix the height to the "ideal" height
	main->setFixedHeight(main->sizeHint().height());
}
```

Dezavantaje :

- Utilizatorul nu poate redimensiona fereastra (la redimensionare componentele raman pe loc si fereastra nu arata bine).
- Nu ia in considerare fontul, dimensiunea textului.
- Pozitiile si dimensiunile trebuie calculate manual (usor de gresit, greu de intretinut)