

# UNIDADE 1

INTERFACES DE USUARIO  
QT E QT DESIGNER

DESENVOLVIMENTO DE INTERFACES  
CS DESENVOLVIMENTO DE APLICACIÓNS MULTIPLATAFORMA

Autor: Manuel Pacior Pérez



## Índice

<b>1</b>	<b>INTERFACES DE USUARIO.....</b>	<b>5</b>
1.1	INTRODUCCIÓN.....	5
1.2	COMPOÑENTES.....	5
1.3	BIBLIOTECAS DE COMPOÑENTES.....	6
1.3.1	JAVA.....	7
1.3.1.1	JFC (JAVA Foundation Classes).....	7
1.3.1.2	AWT.....	7
1.3.1.3	Swing.....	7
1.3.1.4	SWT.....	7
1.3.2	Bibliotecas MSDN de Microsoft (C#, ASP, ...).....	8
1.3.3	Bibliotecas basadas en XML.....	8
<b>2</b>	<b>QT FRAMEWORK.....</b>	<b>9</b>
2.1	INTRODUCCIÓN.....	9
2.2	INSTALACIÓN.....	10
2.2.1	Debian/Ubuntu.....	10
2.2.2	Windows e MAC OS X.....	10
<b>3</b>	<b>QT DESIGNER.....</b>	<b>10</b>
3.1	INTRODUCCIÓN.....	10
3.2	QWIDGET: CLASE BASE.....	10
3.3	FIESTRAS.....	11
3.3.1	QDialog.....	11
3.3.2	QMainWindow.....	11
3.4	CONTEDORES.....	12
3.4.1	QGroupBox.....	12
3.4.2	QScrollArea.....	12
3.4.3	QToolBox.....	12



3.4.4	QTabWidget.....	13
3.4.5	QTabBar.....	14
3.4.6	QStackedWidget.....	14
3.4.7	QFrame.....	14
3.4.8	QButtonGroup.....	14
3.5	LAYOUTS (MARCOS DE DESEÑO).....	15
3.5.1	QLayout.....	15
3.5.2	QBoxLayout: QHBoxLayout e QVBoxLayout.....	15
3.5.2.1	QHBoxLayout.....	16
3.5.2.2	QVBoxLayout.....	16
3.5.3	QFormLayout.....	16
3.5.4	QGridLayout.....	17
3.5.5	QGraphicsAnchorLayout e QGraphicsAnchor.....	17
3.5.6	QStackedLayout.....	17
3.6	COMPONENTES DE ENTRADA.....	18
3.6.1	QComboBox.....	18
3.6.2	QFontComboBox.....	18
3.6.3	QLineEdit.....	18
3.6.4	QTextEdit.....	18
3.6.5	QPlainTextEdit.....	19
3.6.6	QSpinBox.....	19
3.6.7	QDoubleSpinBox.....	19
3.6.8	QDateTimeEdit.....	20
3.6.9	QTimeEdit.....	20
3.6.10	QDateEdit.....	20
3.6.11	QScrollBar.....	21
3.6.12	QSlider.....	21
3.6.13	QKeySequenceEdit.....	22
3.7	COMPONENTES DE SAÍDA.....	22
3.7.1	QLabel.....	22
3.7.2	QTextBrowser.....	22
3.7.3	QCalendarWidget.....	23
3.7.4	QLCDNumber.....	23



3.7.5 QProgressBar.....	23
3.8 BOTÓNS.....	24
3.8.1 QPushButton.....	24
3.8.2 QToolButton.....	24
3.8.3 QRadioButton.....	24
3.8.4 QCheckBox.....	25
3.8.5 QDialogButtonBox.....	25



# 1 INTERFACES DE USUARIO

## 1.1 INTRODUCCIÓN

A **interface de usuario** é o elemento dunha aplicación informática que lle permite ao usuario comunicarse con ela.

O desenvolvemento de aplicacións hoxe en día non se pode entender sen dedicar unha porcentaxe de tempo bastante significativa a planificar, analizar, deseñar, implementar e probar as súas interfaces de usuario xa que son os medios fundamentais polos que o usuario pode comunicarse coa aplicación e realizar as operacións para as que foi deseñado. Existen diferentes tipos de interfaces de usuario:

- **Textuais:** A comunicación co usuario final faise mediante a inserción de ordes escritas a través dun **intérprete (shell) de comandos** (ex. a consola de Linux e Windows).
- **Gráficas:** A **interface** consiste nun conxunto de elementos visuais como **iconas** ou **menús** coa que interactúa, normalmente, mediante a través dun **dispositivo apuntador** (o rato por exemplo). Son os máis comúns e están orgullosos de popularizar o mundo da informática para usuarios novatos.
- **Táctiles:** A para comunicación ocorre por medio dun dispositivo de contacto, xeralmente unha pantalla que pode reaccionar aos elementos apuntadores, punteiro táctil ou dedos. Adoitan empregarse en dispositivos móbiles, terminais de punto de venda e para deseño de gráficos por ordenador..

Ao longo desta unidade centrarémonos en crear interfaces gráficas. Veremos que unha interface gráfica está composta por un conxunto de fiestras, chamadas **formularios**, e que dentro delas podemos colocar diferentes elementos visuais, que se denominan **controis** ou **compoñentes** cos que damos ordes ou recibimos información ao interactuar.

### PARA SABER MÁIS

Na seguinte ligazón atoparás unha web sobre elementos que poden aparecer en interfaces gráficas de usuario e formas de deseñalos: [Interfaces gráficas de usuario](#)

## 1.2 COMPOÑENTES

Unha interface gráfica compórtase como un todo para proporcionar un servizo ao usuario, permitíndolle realizar solicitudes e mostrando o resultado das accións realizadas pola aplicación. Non obstante, está composto por unha serie de elementos gráficos



atómicos que teñen as súas propias características e funcións e que se combinan para formar a interface. Estes elementos chámanse compoñentes ou controis.

Algúns dos compoñentes máis típicos son:

- **Etiquetas:** permiten colocar un texto na interface. Non son interactivos e pódense usar para escribir texto en varias liñas.
- **Campos de texto:** caixas dunha soa liña nas que podemos escribir algúns datos.
- **Áreas de texto:** caixas de varias liñas nas que podemos escribir parágrafos.
- **Botóns:** áreas rectangulares que se poden premer para realizar unha acción.
- **Botóns de opción:** botóns de opción que se agrupan para facer unha selección dun único elemento entre eles. Úsanse para solicitar datos nun conxunto. O botón marcado está representado por un círculo.
- **Caixas de verificación:** botóns con forma de rectángulo. Úsanse para marcar unha opción. Cando se marca, aparece cunha marca dentro. Pódense seleccionar varios nun conxunto.
- **Imaxes:** úsanse para engadir información gráfica á interface.
- **Contrasinal:** é un cadro de texto no que aparecen ocultos os caracteres. Úsase para escribir contrasinais que outros usuarios non deben ver.
- **Listas:** conxunto de datos presentados nunha táboa entre os que é posible escoller un ou máis.
- **Listas despregables:** combinación de caixa de texto e lista, permite escribir unha información ou seleccionala da lista que aparece oculta e que se pode amosar.

### 1.3 BIBLIOTECAS DE COMPOÑENTES

Os compoñentes que poden formar parte dunha interface gráfica normalmente agrúpanse en bibliotecas<sup>1</sup> coa posibilidade de que o usuario poida xerar os seus propios compoñentes e engadilos ou crear as súas propias bibliotecas. Están compostos por un conxunto de clases que se poden incluír en proxectos de software para crear interfaces gráficas. O uso dalgunhas bibliotecas ou outras depende de varios factores, un dos máis importantes, por suposto, é a linguaxe de programación ou o ambiente de



desenvolvemento que se vai empregar. A continuación enumeramos algunhas delas.

### **1.3.1 JAVA**

#### **1.3.1.1 JFC (JAVA Foundation Classes)**

As JAVA Foundation Classes (JFC) son un marco gráfico para construír interfaces gráficas de usuario portátiles baseadas en Java.

JFC está composto por Abstract Window Toolkit (AWT), Swing, SWT e Java 2D. Xuntos, proporcionan unha interface de usuario consistente para programas Java multiplataforma (Windows, Linux e MacOS X).

#### **1.3.1.2 AWT**

Primeira biblioteca Java para a creación de interfaces gráficas. É común a todas as plataformas, pero cada unha ten os seus propios compoñentes, escritos en código nativo para eles. Atópase en desuso, malia que serviu de base para outros proxectos posteriores.

#### **1.3.1.3 Swing**

Trátase dun proxecto que xurdiu despois de AWT, os seus compoñentes son totalmente multiplataforma porque non teñen código nativo.

Moitos compoñentes Swing derivan de AWT, só hai que engadir un "J" ao comezo do nome de AWT para ter o nome Swing, por exemplo, o elemento AWT "Button" ten a súa correspondencia Swing en "JButton" aínda que se engadiron moitos compoñentes novos. Comeza a estar en desuso.

#### **1.3.1.4 SWT**

[SWT \(Standard Widget Toolkit\)](#) é un conxunto de compoñentes para construír interfaces gráficas Java (Widgets) desenvolvido polo proxecto Eclipse.

Recupera a idea orixinal da biblioteca AWT de usar compoñentes nativos, adoptando así un estilo máis consistente en todas as plataformas, pero evitando as súas limitacións.

A biblioteca Swing, por outra banda, está codificada enteiramente en Java e é frecuentemente acusada de non proporcionar unha experiencia idéntica á dunha aplicación nativa.

Non obstante, o prezo a pagar con SWT por esta mellora, é a dependencia (visual e non como interface de programación) da aplicación resultante do sistema operativo no que se executa.



A interface do banco de traballo eclipse tamén depende dunha capa media de interface gráfica de usuario (GUI) chamada JFace que simplifica a creación de aplicacións baseadas en SWT.

### **1.3.2 Bibliotecas MSDN de Microsoft (C#, ASP, ...)**

NET Framework refírese tanto ao compoñente integral que permite a compilación e execución de aplicacións e webs, como á biblioteca de compoñentes que permite a súa creación. Para o desenvolvemento de interfaces gráficas a biblioteca inclúe ADO.NET, ASP.NET, Windows Forms e o WPF (Windows Presentation Foundation<sup>1</sup>).

#### **PARA SABER MÁIS**

Para saber un pouco máis sobre as bibliotecas de .NET Framework, en xeral e para o desenvolvemento de interfaces gráficas, pode acceder á páxina web que MSDN ten sobre o tema na seguinte ligazón.

[MSDN. Biblioteca de clases de .NET Framework](#)

### **1.3.3 Bibliotecas baseadas en XML**

Tamén hai bibliotecas implementadas en linguaxes intermedias baseadas en tecnoloxías XML (que serán tratadas neste módulo).

Normalmente teñen mecanismos para desenvolver as interfaces e traducilas a diferentes linguaxes de programación, para posteriormente integrarse na aplicación final. Neste eido destacan as bibliotecas QT que pertencen actualmente á compañía Nokia.





## 2 QT FRAMEWORK

### 2.1 INTRODUCCIÓN

Qt é un framework (marco de traballo) orientado a obxectos multiplataforma moi popular para desenvolver programas que empregan interface gráfica de usuario.

Trátase dunha solución de código aberto que se distribúe baixo os termos da Licenza GNU LGPL e outras comerciais, destas ocúpase Digia. Polo tanto no proxecto participan tanto a comunidade, como desenvolvedores de empresas tecnolóxicas.

Qt usa a linguaxe de programación C++ de forma nativa, máis unha *das súas mellores vantaxes é a súa versatilidade, ao poder usalo en múltiples linguaxes de programación a través de bindings (enlaces)*. Entre as linguaxes que soportan Qt están: Java, Rust, Go, Ring, C# ou *Python*.

Funciona en todas as principais plataformas e ten un amplo soporte. Tamén se usa en sistemas informáticos integrados para automóbiles, navegación aérea e electrodomésticos (lavadoras, frigoríficos, ...).

Ten unha API (Application Programming Interface ou Interface de Programación de Aplicacións) moi completa. Unha das súas características máis destacables é a posibilidade de *deseñar GUI (interfaces gráficas de usuario) mediante o uso de XML (usan a extensión .ui)*.

Os ficheiros UI deseñados co Qt permítenos xerar as interfaces gráficas de dous xeitos:

- *Importando directamente o XML desde algunha das linguaxes* con integración con Qt escollida. Isto fai que a nosa interface se xere en tempo de execución, a través do XML que contén o ficheiro UI. Trátase dun mecanismo máis áxil e inmediato, pola contra, faise máis complexo modificar e/ou adaptar o comportamento da interface.
- *A través de compoñentes que definen a interface de usuario na linguaxe integrada con Qt escollida*. A través da integración, por exemplo, PyQt en Python, tomando como referencia o XML contido nos ficheiros UI, xeramos o ficheiros Python (.py) correspondentes.

Malia que non é motivo de estudo nesta unidade, a linguaxe Python será usada ao longo deste curso nas vindeiras unidades, como ferramenta transversal.



## 2.2 INSTALACIÓN

### 2.2.1 Debian/Ubuntu

Para instalar o sistema en [Debian ou Ubuntu](#), podemos facelo a través do xestor de paquetes escribindo isto a seguinte instrución a través da liña de comandos:

```
$ sudo apt-get install qt5-default qttools5-dev-tools
```

### 2.2.2 Windows e MAC OS X

Tamén se pode instalar a aplicación en calquera destes dous sistemas. No caso de usar Windows ou MacOS X, deberá descargar o instalador na [web do proxecto](#), e executalo para comezar o proceso de instalación.

## 3 QT DESIGNER

### 3.1 INTRODUCCIÓN

Qt Designer é a ferramenta Qt para deseñar e construír interfaces gráficas de usuario (GUI) con Qt Widgets. Permite crear e personalizar as nosas fiestras e caixas de diálogo, de xeito que ves o que estás construíndo en tempo real, e realízanse as modificacións de xeito visual e sinxelo.

Os compoñentes deseñador con Qt Designer intégranse perfectamente co código programado, utilizando mecanismos como sinais e slots,

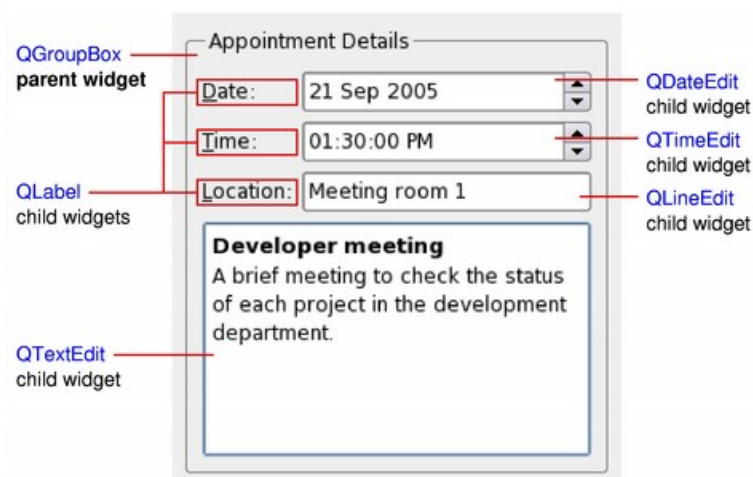
Para poder asignar, de xeito doado, comportamentos ao elementos gráficos, todas as propiedades establecidas en Qt Designer, pódense cambiar dinamicamente desde o código (nós faremolo con Python).

#### PARA SABER MÁIS

Trátase dun proxecto cunha gran comunidade e unha documentación oficial moi completa, desde a que se pode acceder a través da seguinte [ligazón](#).

### 3.2 QWIDGET: CLASE BASE

Qt manexa un conxunto bastante completo de elementos gráficos sendo *QWidget* a clase base para todos os compoñentes que se poden amosar. Calquera clase baseada en [QWidget](#), e que non ten outra pai, pode amosarse como unha xanela.

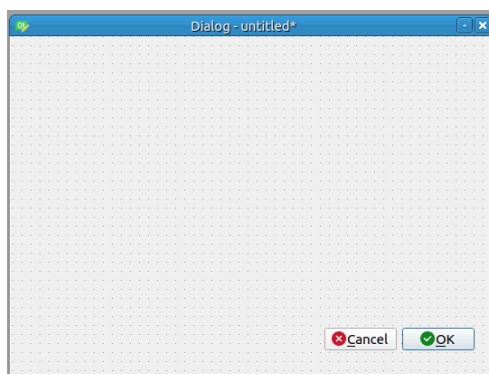


Imaxe dun [QWidget](#)

### 3.3 FIESTRAS

#### 3.3.1 QDialog

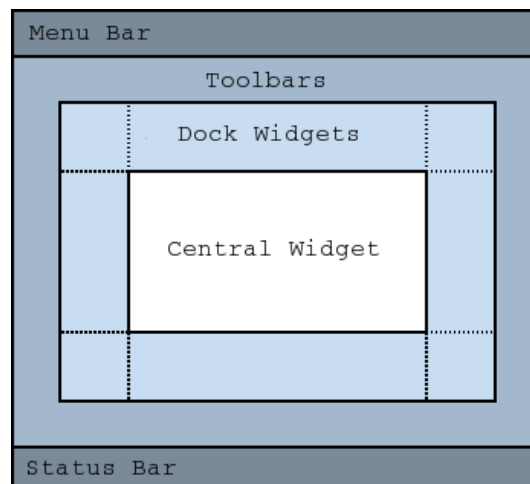
A clase [QDialog](#) está baseada en [QWidget](#), tal como indicamos antes, máis está deseñada para amosarse como unha fiestra pre-configurada cunha serie de cadros de diálogo que permiten realizar operacións comúns (aceptar, rexeitar, etc.).



Imaxe dun [QDialog](#)

#### 3.3.2 QMainWindow

A clase [QMainWindow](#) que define este compoñente está baseada en [QWidget](#) e ofrece as características comúns para crear unha xanela principal, con sitios predefinidos para unha barra de menú, unha barra de estado, unha barra de ferramentas e outros Widgets. Ao contrario que [QDialog](#), non ten accións con botóns predefinidas. A seguinte imaxe resume como se estrutura o Layout deste compoñente.

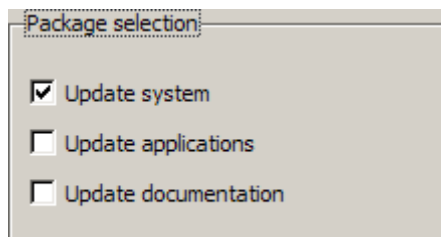


Imaxe. Estrutura dunha [QMainWindow](#)

## 3.4 CONTEDORES

### 3.4.1 QGroupBox

[QGroupBox](#) representa un compoñente cun cadro que ten un título e que pode conter unha colección de elementos.



Imaxe dun [QgroupBox](#)

### 3.4.2 QScrollArea

O [QScrollArea](#) proporciona unha área de desprazamento para amosar o contido dun compoñente fillo dentro dun marco. Se o compoñente fillo supera o tamaño do marco, a vista pode fornecer barras de desprazamento para que se poida ver toda a área do Widget.

### 3.4.3 QToolBox

Unha [QToolBox](#) é un compoñente que amosa unha columna de lapelas unha sobre a outra. A lapela que se atopa seleccionada amosa o seu contido debaixo da pestana actual. Cada pestana ten unha posición de índice dentro da columna de separadores e contén un compoñente de tipo QWidget.



Imaxe dun [QToolBox](#)

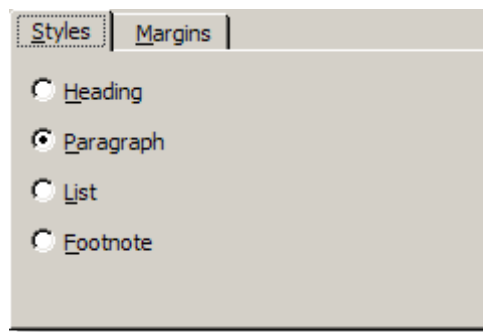
### 3.4.4 QTabWidget

O [QTabWidget](#) ofrece unha barra de pestanas (ver [QTabBar](#)) e unha "área de páxina" que se usa para amosar as páxinas relacionadas con cada separador. Cada pestana está asociada a un compoñente diferente chamado páxina.

Por defecto, a barra de pestanas amósase sobre a área da páxina, máis hai diferentes configuracións dispoñibles (ver `TabPosition`). Só se amosa a páxina actual na área da páxina; todas as outras páxinas permanecerán ocultas. O usuario pode amosar unha páxina diferente facendo clic na súa pestana ou premendo o seu atallo de letra se o ten definido (`Alt+letra`) .

O xeito normal de usar [QTabWidget](#) é facer o seguinte:

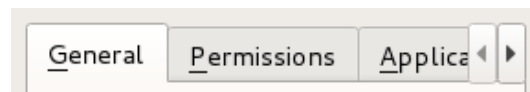
1. Creamos un [QTabWidget](#).
2. Definimos agora un [QWidget](#) para cada unha das páxinas do noso diálogo de separadores, sen especificar o compoñentes pais para eles
3. Insira os compoñentes fillo en cada páxina.
4. Chama a `addTab ()` ou `insertTab ()` para colocar os compoñentes de páxina no [QTabWidget](#), dándolle a cada pestana unha etiqueta adecuada cun atallo de teclado opcional (se queremos).



Imaxe dun [QTabWidget](#)

### 3.4.5 QTabBar

O compoñente [QTabBar](#) serve para debuxar lapelas usando unha das formas predefinidas, e permite emitir un sinal cando se selecciona unha delas.



Imaxe dun [QTabBar](#)

### 3.4.6 QStackedWidget

O [QStackedWidget](#) pódese usar para crear unha interface de usuario similar á proporcionada por [QTabWidget](#), que representa un compoñente visual construído encima da clase [QStackedLayout](#).

Do mesmo xeito que [QStackedLayout](#), [QStackedWidget](#) pode inserir varios compoñentes por páxina.

### 3.4.7 QFrame

A clase [QFrame](#), é a base de todos aqueles compoñentes que poden ter un marco (frame). Por exemplo:

- O compoñente [QMenu](#) úsase para "elevar" o menú sobre a pantalla.
- [QProgressBar](#) ten un aspecto "afundido".
- [QLabel](#) ten un aspecto plano.
- Pode ser usado directamente para crear marcos de posición sinxelos sen contido

### 3.4.8 QButtonGroup

O compoñente [QButtonGroup](#) fornece un *contedor abstracto* no que se poden colocar elementos de tipo botón (Button). Non proporciona unha representación visual deste



contedor, para iso consulte [QGroupBox](#), senón que serve para xestionar os estados de cada un dos botóns do grupo.

### **3.5 LAYOUTS (MARCOS DE DESEÑO).**

O sistema de [Layouts de Qt](#) ofrece un xeito sinxelo e potente de organizar automaticamente os Widgets fillos dentro dun Widget (pai) para garantir que fagan un uso axeitado do espazo dispoñible.

Qt inclúe un conxunto de clases de xestión de deseño (Layouts) que se usan para describir como se distribúen os compoñentes na interface de usuario dunha aplicación. Estes deseños posicinan e redimensionan automaticamente os Widgets cando cambia a cantidade de espazo dispoñible para eles, garantindo que estean ordenados de xeito consistente e que a interface de usuario no seu conxunto siga sendo utilizable.

Todas as subclases de QWidget poden usar Layouts ou esquemas para definir o deseño dos elementos que contén. Cando se define un Layout nun Widget, este faise cargo das seguintes tarefas:

- Posicionamento dos compoñentes que contén (fillos).
- Tamaños predeterminados para as fiestras.
- Tamaños mínimos para as fiestras.
- Xestión do redimensionamento.
- Actualizacións automáticas cando o contido cambia:
  - Tamaño da letra, texto ou outro contido dos compoñentes contidos.
  - Ocultar ou amosar os compoñentes que contén.
  - Eliminar os compoñentes contidos.

A continuación enumeramos os principais Layouts que fornece o Framework Qt.

#### **3.5.1 QLayout**

Trátase dunha clase base abstracta que herdan os compoñentes de tipo Layout: [QBoxLayout](#), [QGridLayout](#), [QFormLayout](#) e [QStackedLayout](#).

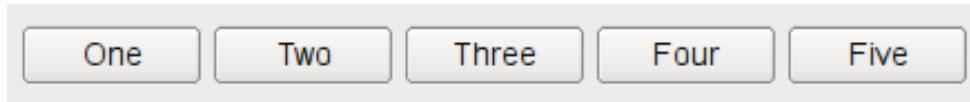
#### **3.5.2 QBoxLayout: QHBoxLayout e QVBoxLayout**

O [QBoxLayout](#) toma o espazo que obtén desde o deseño do seu compoñente pai, divídeo en celas horizontais ou verticais, segundo o configuremos. Os compoñentes fillos distribúense continua e proporcionalmente, enchendo a cela contedor na que é asignado.

A orientación pode ser vertical ("mesma columna") ou horizontalmente ("mesma fila").

### 3.5.2.1 QHBoxLayout

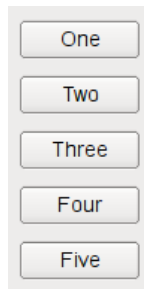
O compoñente [QHBoxLayout](#) especializa ao [QBoxLayout](#) de xeito que aliña os compoñentes que contén.



Imaxe dun [QHBoxLayout](#)

### 3.5.2.2 QVBoxLayout

O compoñente [QVBoxLayout](#) especializa ao [QBoxLayout](#) de xeito que se aliña o compoñentes que contén verticalmente.



Imaxe dun [QVBoxLayout](#)

### 3.5.3 QFormLayout

O [QFormLayout](#) xestiona formularios de compoñentes de entrada (Inputs) e as súas etiquetas (Labels) asociadas. Trátase dunha clase de deseño que amosa aos seus fillos nun formulario con dúas columnas:

- A columna esquerda: contén as etiquetas
- A columna dereita: contén os compoñentes de entrada (editores de liñas, caixas de rotación, etc.).

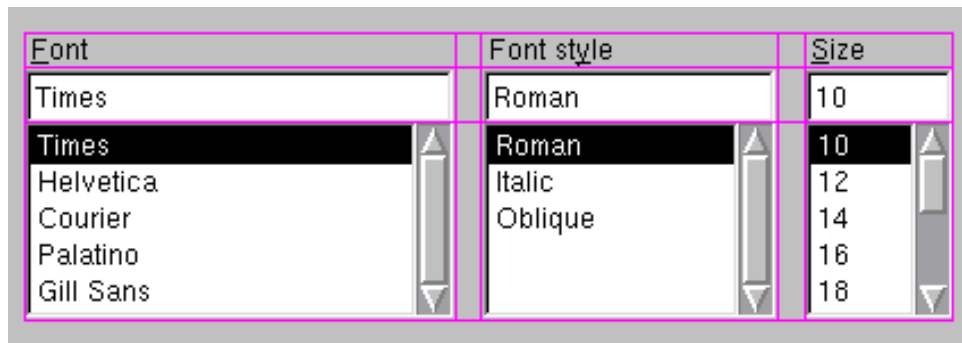


Imaxe dun [QFormLayout](#)



### 3.5.4 QGridLayout

O [QGridLayout](#) colle todo o espazo dispoñible, divídeo en filas e columnas, e pon cada compoñente que xestiona, na cela axeitada.

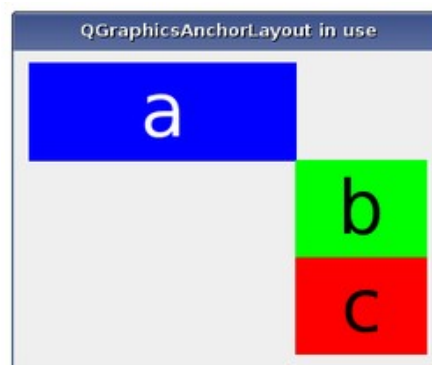


Imaxe dun [QGridLayout](#)

### 3.5.5 QGraphicsAnchorLayout e QGraphicsAnchor

Estes dous compoñentes explícanse xuntos porque están totalmente relacionados:

- O [QGraphicsAnchorLayout](#) permite aos desenvolvedores especificar como se deben colocar os compoñentes en relación a outros.
- O [QGraphicsAnchor](#) representa un enlace entre dous elementos dun [QGraphicsAnchorLayout](#).



Imaxe dun [QgraphicsAnchorLayout](#)

### 3.5.6 QStackedLayout

[QStackedLayout](#) pódese usar para crear unha interface de usuario similar á proporcionada por [QTabWidget](#).

## 3.6 COMPONENTES DE ENTRADA

### 3.6.1 QComboBox

Un [QComboBox](#) proporciona un medio para presentar unha lista de opcións ao usuario de xeito que ocupe a cantidade mínima de espazo na pantalla. Trátase dun compoñente de selección que amosa o elemento actual, e permite ofrecer unha lista de elementos para seleccionar.



Imaxe dun [QComboBox](#)

### 3.6.2 QFontComboBox

Un [QFontComboBox](#) ofrece un Combo preconfigurado coa lista de nomes de familias de tipos de letra ordenados alfabeticamente, como: Arial, Helvetica, Times New Roman, etc. Os nomes da familia amósanse empregando a fonte real cando é posible. Para fontes como Symbol, onde o nome non se pode representar na propia fonte, amósase unha mostra do tipo de letra xunto ao nome da familia.



Imaxe dun [QFontComboBox](#)

### 3.6.3 QLineEdit

Un [QLineEdit](#) define un compoñente que permite ao usuario introducir e editar unha soa liña de texto plano cunha útil colección de funcións de edición (incluíndo desfacer e refacer, cortar e pegar e arrastrar e soltar).

Mudando a propiedade por defecto de "EchoMode" podemos facer que sexa so de escritura e que non se amose, por exemplo, para entradas de contrasinais.



Imaxe dunha [QLineEdit](#)

### 3.6.4 QTextEdit

[QTextEdit](#) é un visor/editor WYSIWYG avanzado que admite o formato de texto enriquecido usando etiquetas de estilo HTML ou como linguaxe de marcado (Markdown). Está optimizado para tratar documentos de gran tamaño e para responder rapidamente á

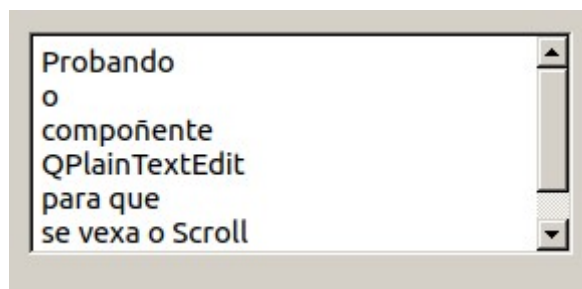
entrada do usuario.



Imaxe dun [QTextEdit](#)

### 3.6.5 QPlainTextEdit

[QPlainTextEdit](#) define un editor avanzado que admite texto plano. Está optimizado para tratar documentos de gran tamaño e para responder rapidamente á entrada do usuario. O contido do texto inicialízase a través da propiedade *plainText*.



Imaxe dun [QPlainTextEdit](#)

### 3.6.6 QSpinBox

[QSpinBox](#) está deseñado para manexar números enteiros e conxuntos discretos de valores (por exemplo, nomes de meses). Permite ao usuario escoller un valor facendo clic nos botóns arriba/abaixo ou premendo arriba/abaixo no teclado para aumentar/diminuír o valor que se amosa actualmente. O usuario tamén pode escribir o valor manualmente.



Imaxe dun [QSpinBox](#)

### 3.6.7 QDoubleSpinBox

[QDoubleSpinBox](#) ten o mesmo comportamento co [QSpinBox](#), máis neste caso permite escoller valores dobres. Para definir en canto aumenta o seguinte valor, debemos usar a

propiedade *singleStep*.



Imaxe. dun [QDoubleSpinBox](#)

### 3.6.8 QDateTimeEdit

A clase [QDateTimeEdit](#) ofrece un compoñente para editar datas e horas. Permite ao usuario editar datas usando o teclado ou as teclas coas frechas, para aumentar e diminuír os valores de data e hora.



Imaxe dun [QDateTimeEdit](#)

### 3.6.9 QTimeEdit

A clase [QTimeEdit](#) proporciona un compoñente para editar os tempos, baseado no [QDateTimeEdit](#). Moitas das propiedades e funcións proporcionadas por [QTimeEdit](#) están implementadas en [QDateTimeEdit](#). Estas son as propiedades relevantes desta clase:

- ***time***: garda o tempo que amosa o compoñente.
- ***minimumTime***: define o tempo mínimo que pode configurar o usuario.
- ***maximumTime***: define o tempo máximo que pode configurar o usuario.
- ***displayFormat***: contén unha cadea que se usa para dar formato á hora que amosa o compoñente.



Imaxe dun [QTimeEdit](#)

### 3.6.10 QDateEdit

A clase [QDateEdit](#) proporciona un compoñente para editar os tempos, baseado no [QDateTimeEdit](#). Moitas das propiedades e funcións proporcionadas por [QDateEdit](#) están implementadas en [QDateTimeEdit](#). Estas son as propiedades relevantes desta clase:

- ***date***: garda a data que amosa o compoñente
- ***minimumDate***: define a data mínima que pode establecer o usuario.

- **maximumDate**: define a data máxima (última) que pode configurar o usuario.
- **displayFormat**: contén unha cadea que se usa para dar formato á data que se amosa no compoñente.

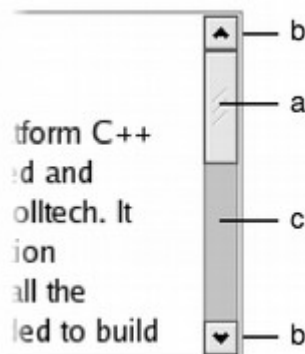


Imaxes dun [QDateEdit](#)

### 3.6.11 QScrollBar

Unha [QScrollBar](#) representa unha barra de desprazamento, trátase dun control que permite ao usuario acceder a partes dun documento que é máis grande que o compoñente usado para amosalo.

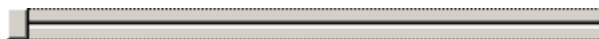
Ofrece unha indicación visual da posición actual do usuario dentro do documento e da cantidade do documento visible. As barras de desprazamento adoitan estar equipadas con outros controis que permiten unha navegación máis precisa. Qt amosa as barras de desprazamento dun xeito adecuado a cada plataforma.



Imaxe dunha [QScrollBar](#)

### 3.6.12 QSlider

O [QSlider](#) é un compoñente clásico para definir un valor delimitado. Permite ao usuario mover un control que se despraza ao longo dunha ranhura horizontal ou vertical, e traduce a posición do controlador nun valor enteiro que se atopa dentro do rango definido.



Imaxe dun [QSlider](#)

### 3.6.13 QKeySequenceEdit

O compoñente [QKeySequenceEdit](#) permítelle ao usuario escoller unha secuencia, que normalmente se usa como atallo. A gravación iníciase cando o compoñente recibe o foco e remata un segundo despois de que o usuario solte a última tecla.



Imaxe dunha [QKeySequenceEdit](#)

## 3.7 COMPOÑENTES DE SAÍDA

### 3.7.1 QLabel

O compoñente [QLabel](#) proporciona unha visualización para texto ou imaxe. Non se ofrece ningunha funcionalidade de interacción do usuario. O aspecto visual da etiqueta pódese configurar de varios xeitos, polo que pode conter calquera dos seguintes tipos de contido:

- **Texto simple**: Pasando un elemento QString a través de setText().
- **Texto enriquecido**: Pasando un elemento QString que conteña texto enriquecido (HTML ou Markdown) a través de setText().
- **Unha imaxe**: Pasando un elemento de tipo QPixmap mediante o método setPixmap().
- **Un vídeo**: Pasando un elemento de tipo QMovie a través de setMovie().
- **Un número**: Pasa un enteiro un un número decimal a través de setNum, que o converte a texto plano.
- **Nada**: O mesmo que un texto sinxelo baleiro. Este é o valor predeterminado que tamén se pode establecer a través do método clear().

**Exemplo de QLabel:**

Imaxe dun [QLabel](#)

### 3.7.2 QTextBrowser

O compoñente [QTextBrowser](#) proporciona un navegador de texto enriquecido con navegación en hipertexto. Esta clase amplía a [QTextEdit](#), en modo de só lectura, e engadindo algunhas funcións de navegación para que os usuarios poidan seguir ligazóns

en documentos de hipertexto.

Se desexa proporcionar aos seus usuarios un editor de texto enriquecido editable, use [QTextEdit](#).

### 3.7.3 QCalendarWidget

A clase [QCalendarWidget](#) ofrece un calendario mensual que permite ao usuario seleccionar unha data. O widget inicialízase co mes e o ano actuais, máis [QCalendarWidget](#) ofrece varios espazos públicos para mudar o ano e o mes que se amosan.

	Mon	Tue	Wed	Thu	Fri	Sat	Sun
44	29	30	31	1	2	3	4
45	5	6	7	8	9	10	11
46	12	13	14	15	16	17	18
47	19	20	21	22	23	24	25
48	26	27	28	29	30	1	2
49	3	4	5	6	7	8	9

Imaxe dun [QCalendarWidget](#)

### 3.7.4 QLCDNumber

O compoñente [QLCDNumber](#) amosa un número con díxitos tipo LCD de case case calquera tamaño. Mediante a configuración da propiedade **mode** podemos indicar o tipo de dato: números decimais, hexadecimais, octais ou binarios.



Imaxe dun [QLCDNumber](#)

### 3.7.5 QProgressBar

[QProgressBar](#) proporciona unha barra de progreso horizontal ou vertical que se soe usar para informar de dous aspectos:

1. Informar ao usuario da porcentaxe de progreso dunha determinada operación
2. Informar ao usuario de que a aplicación aínda está en execución.

A barra de progreso utiliza o concepto de pasos. Configúreo especificando os valores de paso mínimos e máximos posibles e amosará a porcentaxe de pasos que se completaron

a medida que se vai modificando o valor de paso actual. A porcentaxe calcúlase dividindo o progreso:  $(\text{value}() - \text{minimum}()) / (\text{maximum}() - \text{minimum}())$ .



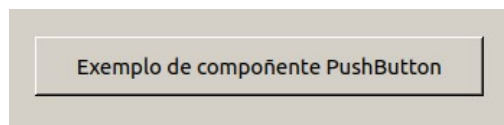
Imaxe dunha [QProgressBar](#)

## 3.8 BOTÓNS

### 3.8.1 QPushButton

O compoñente [QPushButton](#) proporciona un botón de comando, e pode que sexa un dos máis usados en calquera interface gráfica de usuario. Premer nun botón para que a aplicación realice algunha acción ou responda a algunha pregunta. **Os botóns típicos son: Aceptar, Aplicar, Cancelar, Pechar, Si, Non e Axuda.**

Un botón de comando represéntase como un elemento rectangular, e normalmente amosa unha etiqueta de texto coa que describe a acción que realiza. Pódese especificar unha tecla de atallo.



Imaxe dun [QPushButton](#)

### 3.8.2 QToolButton

A clase [QToolButton](#) proporciona un botón de acceso rápido a comandos ou opcións, usados normalmente dentro dunha [QToolBar](#). A diferenza dun botón de comando normal, un botón de ferramenta normalmente non amosa unha etiqueta de texto, senón que amosa unha icona.




Imaxe. dunha [QToolButton](#)

### 3.8.3 QRadioButton

O compoñente [QRadioButton](#) ofrece un botón de opción cunha etiqueta de texto que se pode activar (marcar) ou desactivar (desmarcar). Os botóns de radio normalmente presentan ao usuario unha opción "entre moitos".



Nun grupo de botóns [QRadioButton](#), só se pode marcar unha opción á vez; se o usuario selecciona outro botón, o botón seleccionado previamente está desactivado.



RadioButton de exemplo 1

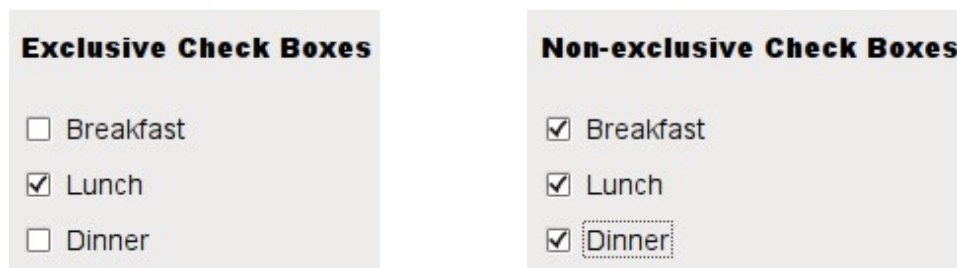
Imaxe dunha [QToolButton](#)

### 3.8.4 QCheckBox

O [QCheckBox](#) é un botón de opción que se pode activar (marcar) ou desactivar (desmarcar). As caixas de verificación úsanse normalmente para representar funcións nunha aplicación que se pode activar ou desactivar sen afectar a outras.

Pódense implementar diferentes tipos de comportamento. Por exemplo, pódese usar un [QButtonGroup](#) para agrupar os botóns de comprobación lóxicamente, permitindo caixas de verificación exclusivas ou múltiples.

A seguinte imaxe ilustra as diferenzas entre as caixas de verificación exclusivas e non exclusivas.



Imaxe de dous [QCheckBox](#) o da esquerda exclusivo e o da dereita múltiple

### 3.8.5 QDialogButtonBox

A clase [QDialogButtonBox](#) é un compoñente que presenta botóns nun deseño axeitado. Os diálogos e caixas de mensaxes normalmente presentan botóns nun deseño que se axusta ás directrices da interface desa plataforma.

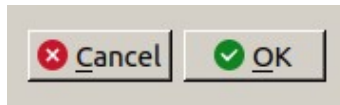
[QDialogButtonBox](#) permite que un desenvolvedor engada botóns e empregue automaticamente o deseño axeitado para o ambiente de escritorio do usuario.

A maioría dos botóns dun diálogo seguen certos roles. Estes papeis inclúen:

- Aceptar ou rexeitar o diálogo.
- Pedir axuda.
- Realización de accións no propio diálogo (como restablecer campos ou aplicar



cambios).



Imaxe dun [QDialogButtonBox](#)