school everis

FORMAMOS PROFESIONALES

Módulo 7

Código: 488

Desarrollo de Interfaces

Técnico Superior en Desarrollo de Aplicaciones Multiplataforma

school everis

FORMAMOS PROFESIONALES

UNIDAD 2
GENERACIÓN DE INTERFACES A PARTIR DE DOCUMENTOS XML

Contenidos teóricos

Índice

1. Objetivos generales de la unidad

Objetivos curriculares

1. Reconocer las ventajas de generar interfaces gráficas de usuario a partir de su descripción en XML.
2. Generar la descripción de la interfaz en XML usando un editor gráfico.
3. Analizar y modificar el documento XML generado.
4. Asignar acciones a eventos.

2. Competencias y contribución en la unidad

Desarrollar interfaces gráficas en XML, conocer distintas herramientas y aplicaciones libres y propietarias para la creación y validación de interfaces multiplataforma.

3. Contenidos

school everis FORMAMOS PROFESIONALES

CONTENIDOS CONCEPTUALES	CONTENIDOS PROCEDIMENTALES

• Diseño de interfaces estáticas o interfaces dinámicas.
• XML bien formados.
• Lenguajes de descripción de interfaces basados en XML.
• XUL, UIML, SVG, MXML.
• Herramientas libres y propietarias para la creación de interfaces de usuario multiplataforma.
• Análisis y edición de documentos XML.
 	
.
	• Diseño de una interfaz:
	  • GLADE.
	  • BLEND.

4. Evaluación

school everis FORMAMOS PROFESIONALES

Lenguajes de descripción de interfaces basadas en XML: ámbito de aplicación.

Elementos, etiquetas, atributos y valores.

Herramientas libres e propietarias para la creación de interfaces de usuario multiplataforma.

Controles: propiedades.

4. Evaluación

school everis FORMAMOS PROFESIONALES

Eventos: controladores.

Editores de documentos XML.

Generación de código para diferentes plataformas.

2.1. DISEÑO DE INTERFACES ESTÁTICAS O INTERFACES DINÁMICAS

Interfaz de Usuario UI (User Interface), la interfaz con la que las personas interaccionan con los dispositivos:

- Interfaz Estática: No cambia en tiempo de ejecución. Por ejemplo la interfaz de aplicaciones de escritorio.
- Interfaz Dinámica: Cambia en tiempo de ejecución. Por ejemplo aplicaciones de correo que pueden actualizarse en cualquier momento.

Factores para conseguir una UI dinámica:

- Diseñar la aplicación con los componentes, los controles y los patrones de navegación más recientes.
- Diseño de la UI y flujo de navegación, para aprovechar los distintos tamaños de las pantallas de los dispositivos, conviene usar diferentes diseños para diferentes tamaños de pantalla.
- Resolución de la pantalla y densidad. Además del diseño de la UI, si la aplicación tiene gráficos habrá que ajustar la resolución dependiendo del dispositivo.

2.2 XML

XML son las siglas de eXtensible Markup Language (lenguaje de marcas extensible), desarrollado por el World Wide Web Consortium (W3C) es un lenguaje de marcas utilizado para almacenar datos en forma legible y viene del SGML (Standard Generalized Markup Language).
XML da soporte a bases de datos, a diferencia de otros lenguajes, siendo útil cuando múltiples aplicaciones se comunican entre sí o integrar información.
Es extensible después de ser lanzado es posible extender XML con nuevas etiquetas de manera muy sencilla. Un componente importante aunque no necesario es el analizador que nos permite evitar bugs y acelera el desarrollo de las aplicaciones.
Una etiqueta consiste en una marca hecha en el documento, que señala una porción de éste como un elemento. Un pedazo de información con un sentido claro y definido. Las etiquetas tienen la forma <nombre>, donde nombre es el nombre del elemento que se está señalando.
A continuación se muestra un ejemplo para entender la estructura de un documento XML:

2.2.2 ESTRUCTURA DE UN DOCUMENTO XML

Ejemplo de estructura de un XML:

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE Edit_Mensaje SYSTEM "Edit_Correo.dtd">
 <Edit_Mensaje>
  <Mensaje>
   <Remitente>
     <Nombre>Nombre del remitente</Nombre>
     <Mail> Correo del remitente </Mail>
   </Remitente>
   <Destinatario>
     <Nombre>Nombre del destinatario</Nombre>
     <Mail>Correo del destinatario</Mail>
   </Destinatario>
   <Texto>
    <Asunto>
     Este es mi documento con una estructura muy sencilla
     no contiene atributos ni entidades...
   </Asunto>
    <Parrafo>
     Este es mi documento con una estructura muy sencilla
     no contiene atributos ni entidades...
    </Parrafo>
   </Texto>
  </Mensaje>
 </Edit_Mensaje>
```

Ejemplo del código del DTD de "Edit_correo.dtd":

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!-- Este es el DTD de Edit_Mensaje -->
<!ELEMENT Mensaje (Remitente, Destinatario, Texto)*>
<!ELEMENT Remitente (Nombre, Mail)>
<!ELEMENT Nombre (#PCDATA)>
<!ELEMENT Mail (#PCDATA)>
<!ELEMENT Destinatario (Nombre, Mail)>
<!ELEMENT Nombre (#PCDATA)>
<!ELEMENT Mail (#PCDATA)>
<!ELEMENT Texto (Asunto, Parrafo)>
<!ELEMENT Asunto (#PCDATA)>
<!ELEMENT Parrafo (#PCDATA)>
```

2.2.3 DOCUMENTOS XML BIEN FORMADOS Y CONTROL DE ERRORES

Denominaremos los documentos como "**bien formados**" si cumplen una serie de normas básicas:

- Que incluyan una declaración XML como etiqueta obligatoria:
  - Información de la versión(obligatoria), 1.0 es la versión más utilizada.
  - La codificación de caracteres (opcional), normalmente UTF-8 o UTF-16.
  - Declaración independiente (opcional), documento independiente (standalone="yes") o si depende de declaraciones de marcas externas (standalone="no").
- Que solo exista un elemento raíz por documento.
- Que todas las entidades utilizadas se declaren en la DTD interna.
- Que todos los elementos, atributos y entidades estén escritos de forma correcta.

Si además de estar bien formados cumplen las especificaciones de la DTD, del Schema o del elemento que los valide se consideraran **Documentos Válidos**.

Tendríamos una serie de ventajas en el uso de Schemas frente a los DTD:

- Son extensibles.
- Permiten especificar los tipos de datos
- Usan sintaxis de XML, y los DTD no.

school everis FORMAMOS PROFESIONALES

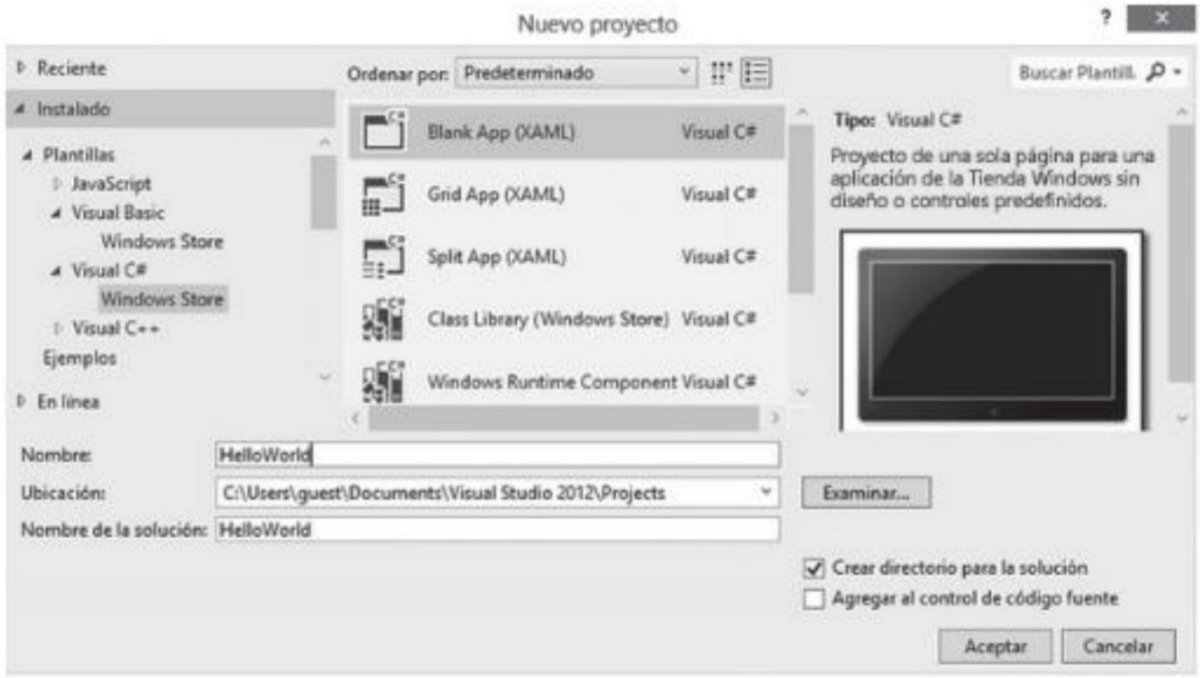
2.3 LENGUAJES DE DESCRIPCIÓN DE INTERFACES BASADOS EN XML. ÁMBITO DE APLICACIÓN. ELEMENTOS, ETIQUETAS, ATRIBUTOS Y VALORES

XAML (acrónimo pronunciado xammel, del inglés, extensible Application Markup Language, Lenguaje Extensible de Formato para Aplicaciones) Es un lenguaje declarativo desarrollado por Microsoft para su plataforma .NET basado en XML, optimizado para describir interfaces de usuario desde el punto de vista gráfico.

Ejemplo de aplicación "Hello world"

Creamos un nuevo proyecto en Visual Studio:

- Inicia Visual Studio
- Picha Archivo->Nuevo proyecto
- En el panel izquierdo,
  expande Instalado->Plantillas
  expande Visual Basic y elige
  el tipo plantilla Tienda Windows



school everis FORMAMOS PROFESIONALES

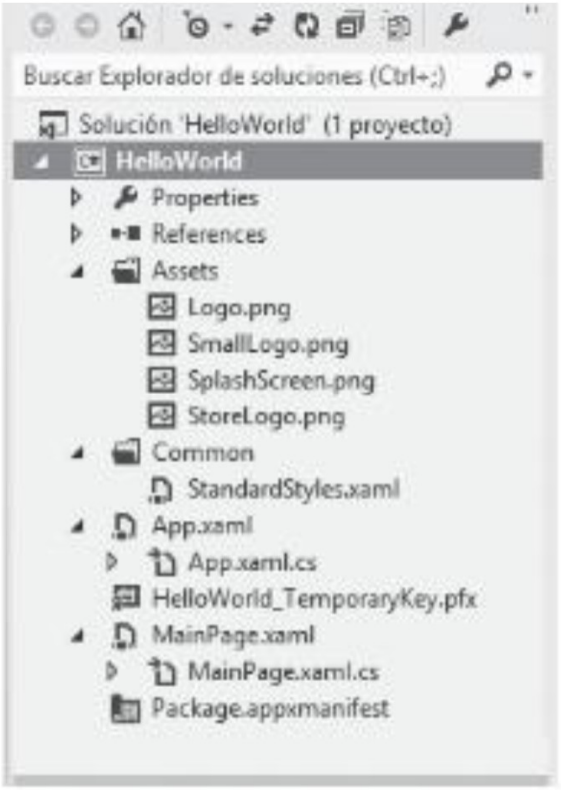
2.3 LENGUAJES DE DESCRIPCIÓN DE INTERFACES BASADOS EN XML. ÁMBITO DE APLICACIÓN. ELEMENTOS, ETIQUETAS, ATRIBUTOS Y VALORES

- Selecciona en el panel central la plantilla *Aplicación vacía, en el cuadro de texto escribe "HelloWorld"*
- Para crear el proyecto click en aceptar.

Aplicación vacía contiene un muchos archivos a pesar de ser Una plantilla mínima:

Archivo de manifiesto(package.appxmanifest)
Logotipos(StoreLog.scale-100.png…)
Código y XAML (app.xaml…)
…

Cualquier proyecto contendrá estos archivos.



school everis FORMAMOS PROFESIONALES

2.3 LENGUAJES DE DESCRIPCIÓN DE INTERFACES BASADOS EN XML. ÁMBITO DE APLICACIÓN. ELEMENTOS, ETIQUETAS, ATRIBUTOS Y VALORES

Sustituiremos la *MainPage* por una página que usa la plantilla *Página básica* (para aprovechar el diseño y las clases auxiliares):

- En el Explorador de soluciones, haz click derecho en *MainPage.xaml* y pulsa Eliminar y Aceptar.
- Selecciona Proyecto -> Agregar nuevo elemento.
- En Visual Basic, elije el tipo de plantilla *Tienda Windows* (en el panel izquierdo).
- En el panel central elije Página básica y escribe "*MainPage.xaml*" como nombre.
- #Importante# No dejar el nombre predeterminado "BasicPage1" porque puede que el proyecto no compile correctamente.
- Clic en Agregar y después clic en Compilar -> Compilar solución.
- #Importante# La nueva página mostrará un error en el diseñador hasta que compile las clases auxiliares de las depende.

2.3 LENGUAJES DE DESCRIPCIÓN DE INTERFACES BASADOS EN XML. ÁMBITO DE APLICACIÓN. ELEMENTOS, ETIQUETAS, ATRIBUTOS Y VALORES

Para ver su apariencia pulsa F5 con lo que compilas, implementas e inicias la aplicación en modo depuración (atl+F4 para cerrar)

Alt+Tab para dejar de depurar.
Depurar -> Detener depuración para cerrar la aplicación.

2.3 LENGUAJES DE DESCRIPCIÓN DE INTERFACES BASADOS EN XML. ÁMBITO DE APLICACIÓN. ELEMENTOS, ETIQUETAS, ATRIBUTOS Y VALORES

Modificar tu página de inicio

Para ver y editar los archivos, haz doble clic en el archivo en el Explorador de soluciones. Puedes expandir un archivo XAML como una carpeta para ver su archivo de código asociado. De manera predeterminada, los archivos XAML se abren en una vista dividida que muestra la superficie de diseño y el editor de XAML.

App.xaml es donde declara recursos que usa en la aplicación.

```
<Application
        x:Class="HelloWorld.App"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:local="using:HelloWorld">
</Application>
```

2.3 LENGUAJES DE DESCRIPCIÓN DE INTERFACES BASADOS EN XML. ÁMBITO DE APLICACIÓN. ELEMENTOS, ETIQUETAS, ATRIBUTOS Y VALORES

App.xaml.vb es el archivo de código subyacente para App.xaml. El código subyacente es el código que se une a la clase parcial de la página XAML. El XAML y el código subyacente forman una clase completa.

App.xaml.vb es el punto de entrada de tu aplicación. Como todas las páginas de código subyacente, contiene un constructor que llama al método InitializeComponent.

El método InitializeComponent lo genera visual Studio, y su principal propósito es inicializar los elementos declarados en el archivo XAML.

App.xaml.vb también contiene métodos para controlar la activación y la suspensión de la aplicación.

school everis FORMAMOS PROFESIONALES

2.3 LENGUAJES DE DESCRIPCIÓN DE INTERFACES BASADOS EN XML. ÁMBITO DE APLICACIÓN. ELEMENTOS, ETIQUETAS, ATRIBUTOS Y VALORES

```
' The Blank Application template is documented at http://go.microsoft.com/fwlink/p/?LinkID=234227
''' <summary>
''' Provides application-specific behavior to supplement the default Application class.
''' </summary>
NotInheritable Class App
Inherits Application
''' <summary>
''' Invoked when the application is launched normally by the end user. Other entry points will be used when the application is launched to open a specific file, to display
''' search results, and so forth.
''' </summary>
''' <param name="e">Details about the launch request and process.</param>
Protected Overrides Sub OnLaunched(e As Windows.ApplicationModel.Activation.LaunchActivatedEventArgs)
#If DEBUG Then
        ' Show graphics profiling information while debugging.
        If System.Diagnostics.Debugger.IsAttached Then
                ' Display the current frame rate counters
                Me.DebugSettings.EnableFrameRateCounter = True
        End If
#End If
Dim rootFrame As Frame = TryCast(Window.Current.Content, Frame)
' Do not repeat app initialization when the Window already has content, just ensure that the window is active
If rootFrame Is Nothing Then
        ' Create a Frame to act as the navigation context and navigate to the first page
        rootFrame = New Frame()
        If e.PreviousExecutionState = ApplicationExecutionState.Terminated Then
                ' TODO: Load state from previously suspended application
        End If
```

2.3 LENGUAJES DE DESCRIPCIÓN DE INTERFACES BASADOS EN XML. ÁMBITO DE APLICACIÓN. ELEMENTOS, ETIQUETAS, ATRIBUTOS Y VALORES

```
            ' Place the frame in the current Window
            Window.Current.Content = rootFrame
End If
If rootFrame.Content Is Nothing Then
            ' When the navigation stack isn't restored navigate to the first page, configuring the new page by passing required information as a navigation parameter
            If Not rootFrame.Navigate(GetType(MainPage), e.Arguments) Then
                        Throw New Exception("Failed to create initial page")
            End If
End If
' Ensure the current window is active
Window.Current.Activate()
End Sub
''' <summary>
''' Invoked when application execution is being suspended. Application state is saved without knowing whether the application will be terminated or resumed with the
''' contents of memory still intact.
''' </summary>
''' <param name="sender">The source of the suspend request.</param>
''' <param name="e">Details about the suspend request.</param>
Private Sub OnSuspending(sender As Object, e As SuspendingEventArgs) Handles
Me.Suspending
Dim deferral As SuspendingDeferral = e.SuspendingOperation.GetDeferral()
' TODO: Save application state and stop any background activity
deferral.Complete()
End Sub
End Class
```

school everis FORMAMOS PROFESIONALES

2.3 LENGUAJES DE DESCRIPCIÓN DE INTERFACES BASADOS EN XML. ÁMBITO DE APLICACIÓN. ELEMENTOS, ETIQUETAS, ATRIBUTOS Y VALORES

MainPage.xaml define la interfaz de usuario de la aplicación. Puede agregar elementos directamente con el marcado XAML o puede usar las herramientas de diseño suministradas por visual Studio. La plantilla Página básica crea una nueva clase llamada MainPage (o cualquier nombre que escriba) que se hereda de Page. También incluye cierto contexto sencillo, como un botón Atrás y un título de página, y proporciona métodos para la navegación y la administración de estado.

```
<Page
        x:Name="pageRoot"
        x:Class="HelloWorld.MainPage"
        DataContext="{Binding DefaultViewModel, RelativeSource={RelativeSource Self}}"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:local="using:HelloWorld"
        xmlns:common="using:HelloWorld.Common"
        xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
        xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
        mc:Ignorable="d">
        <Page.Resources>
                <common:BooleanToVisibilityConverter x:Key="BooleanToVisibilityConverter"/>
                <!-- TODO: Delete this line if the key AppName is declared in App.xaml -->
                <x:String x:Key="AppName">My Application</x:String>
        </Page.Resources>
        <!--
                This grid acts as a root panel for the page that defines two rows:
```

school everis FORMAMOS PROFESIONALES

2.3 LENGUAJES DE DESCRIPCIÓN DE INTERFACES BASADOS EN XML. ÁMBITO DE APLICACIÓN. ELEMENTOS, ETIQUETAS, ATRIBUTOS Y VALORES

```
* Row 0 contains the back button and page title
* Row 1 contains the rest of the page layout
-->
<Grid Background="{StaticResource ApplicationPageBackgroundThemeBrush}">
        <Grid.ChildrenTransitions>
                <TransitionCollection>
                        <EntranceThemeTransition/>
                </TransitionCollection>
        </Grid.ChildrenTransitions>
        <Grid.RowDefinitions>
                <RowDefinition Height="140"/>
                <RowDefinition Height="*"/>
        </Grid.RowDefinitions>
        <!-- Back button and page title -->
        <Grid>
        <Grid.ColumnDefinitions>
                <ColumnDefinition Width="120"/>
                <ColumnDefinition Width="*"/>
        </Grid.ColumnDefinitions>
```

school everis FORMAMOS PROFESIONALES

2.3 LENGUAJES DE DESCRIPCIÓN DE INTERFACES BASADOS EN XML. ÁMBITO DE APLICACIÓN. ELEMENTOS, ETIQUETAS, ATRIBUTOS Y VALORES

```
                <AppBarButton x:Name="backButton" Icon="Back" Height="95" Margin="10,46,10,0" Command="{Binding NavigationHelper.GoBackCommand,
                ElementName=pageRoot}"
                Visibility="{Binding IsEnabled, Converter={StaticResource BooleanToVisibilityConverter}, RelativeSource={RelativeSource Mode=Self}}"
                        AutomationProperties.Name="Back"
                        AutomationProperties.AutomationId="BackButton"
                        AutomationProperties.ItemType="Navigation Button"/>
                <TextBlock x:Name="pageTitle" Text="{StaticResource AppName}" Style="{StaticResource HeaderTextBlockStyle}" Grid.Column="1"
                IsHitTestVisible="false" TextWrapping="NoWrap" VerticalAlignment="Bottom" Margin="0,0,30,40"/>
                </Grid>
        </Grid>
</Page>
```

MainPage.xaml.vb es la página de código subyacente para **MainPage.xaml**. Aquí agrega los controladores de eventos y la lógica de su aplicación. La plantilla Página básica incluye 2 métodos en los que puede guardar y cargar el estado de la página.

school everis FORMAMOS PROFESIONALES

2.3 LENGUAJES DE DESCRIPCIÓN DE INTERFACES BASADOS EN XML. ÁMBITO DE APLICACIÓN. ELEMENTOS, ETIQUETAS, ATRIBUTOS Y VALORES

```
' The Basic Page item template is documented at http://go.microsoft.com/fwlink/p/?LinkID=234237
''' <summary>
''' A basic page that provides characteristics common to most applications.
''' </summary>
Public NotInheritable Class MainPage
        Inherits Page
        ''' <summary>
        ''' NavigationHelper is used on each page to aid in navigation and
        ''' process lifetime management
        ''' </summary>
        Public ReadOnly Property NavigationHelper As Common.NavigationHelper
                Get
                        Return Me._navigationHelper
                End Get
        End Property
        Private _navigationHelper As Common.NavigationHelper
        ''' <summary>
        ''' This can be changed to a strongly typed view model.
        ''' </summary>
        Public ReadOnly Property DefaultViewModel As Common.ObservableDictionary
                Get
                        Return Me._defaultViewModel
                End Get
        End Property
```

school everis FORMAMOS PROFESIONALES

2.3 LENGUAJES DE DESCRIPCIÓN DE INTERFACES BASADOS EN XML. ÁMBITO DE APLICACIÓN. ELEMENTOS, ETIQUETAS, ATRIBUTOS Y VALORES

```
Private _navigationHelper As Common.NavigationHelper
''' <summary>
''' This can be changed to a strongly typed view model.
''' </summary>
Public ReadOnly Property DefaultViewModel As Common.ObservableDictionary
        Get
                Return Me._defaultViewModel
        End Get
End Property
Private _defaultViewModel As New Common.ObservableDictionary()
Public Sub New()
        InitializeComponent()
        Me._navigationHelper = New Common.NavigationHelper(Me)
        AddHandler Me._navigationHelper.LoadState, AddressOf NavigationHelper_LoadState
        AddHandler Me._navigationHelper.SaveState, AddressOf NavigationHelper_SaveState
End Sub
''' <summary>
''' Populates the page with content passed during navigation. Any saved state is also
''' provided when recreating a page from a prior session.
''' </summary>
''' <param name="sender">
''' The source of the event; typically <see cref="NavigationHelper"/>
''' </param>
''' <param name="e">Event data that provides both the navigation parameter passed to
''' <see cref="Frame.Navigate"/> when this page was initially requested and
''' a dictionary of state preserved by this page during an earlier
```

2.3 LENGUAJES DE DESCRIPCIÓN DE INTERFACES BASADOS EN XML. ÁMBITO DE APLICACIÓN. ELEMENTOS, ETIQUETAS, ATRIBUTOS Y VALORES

```
        ''' session. The state will be null the first time a page is visited.</param>
        Private Sub NavigationHelper_LoadState(sender As Object, e As Common.LoadStateEventArgs)
        End Sub
        ''' <summary>
        ''' Preserves state associated with this page in case the application is suspended or the
        ''' page is discarded from the navigation cache. Values must conform to the serialization
        ''' requirements of <see cref="Common.SuspensionManager.SessionState"/>.
        ''' </summary>
        ''' <param name="sender">
        ''' The source of the event; typically <see cref="NavigationHelper"/>
        ''' </param>
        ''' <param name="e">Event data that provides a dictionary of state preserved by this page
        ''' during an earlier session. The state will be null the first time a page is visited.</param>
        Private Sub NavigationHelper_SaveState(sender As Object, e As Common.SaveStateEventArgs)
        End Sub
        #Region "NavigationHelper registration"
                ''' The methods provided in this section are simply used to allow NavigationHelper to respond to the page's navigation methods.
                ''' Page specific logic should be placed in event handlers for the <see cref="Common.NavigationHelper.LoadState"/> and <see
                cref="Common.NavigationHelper.SaveState"/>.
                ''' The navigation parameter is available in the LoadState method in addition to page state preserved during an earlier session.
                Protected Overrides Sub OnNavigatedTo(e As NavigationEventArgs)
                        _navigationHelper.OnNavigatedTo(e)
                End Sub
                Protected Overrides Sub OnNavigatedFrom(e As NavigationEventArgs)
                        _navigationHelper.OnNavigatedFrom(e)
                End Sub
        #End Region
End Class
```

2.3 LENGUAJES DE DESCRIPCIÓN DE INTERFACES BASADOS EN XML. ÁMBITO DE APLICACIÓN. ELEMENTOS, ETIQUETAS, ATRIBUTOS Y VALORES

Modificar la página de inicio:

- Haz clic en MainPage.xaml en el explorador de soluciones para abrirlo.
- Haz doble clic en MainPage.xaml en el Explorador de soluciones para abrirlo. Para cambiar el título de la página, selecciona el texto “Mi aplicación” en la parte superior de la página en el diseñador XAML. Asegúrate de que el TextBlock llamado pageTitle aparezca en el panel Propiedades. El panel Propiedades está por debajo del panel explorador de soluciones, de manera predeterminada.
- El panel Propiedades contiene una lista de propiedades y valores para el objeto seleccionado. Después, cada valor de propiedad es un marcador de propiedad, un símbolo de cuadro pequeño en el que puedes hacer clic para abrir un menú de propiedad. El marcador de propiedad Text de color verde indica que se ha establecido en un recurso. En común en el panel Propiedades, haz clic en el marcador de propiedad de la propiedad Text. Se abre el menú Propiedad.
- En el menú Propiedad, selecciona Editar recurso. Se abrirá el cuadro de diálogo editar recurso.
- En el cuadro de diálogo editar recurso, cambia el valor “My Application” a “Hello world”.

2.3 LENGUAJES DE DESCRIPCIÓN DE INTERFACES BASADOS EN XML. ÁMBITO DE APLICACIÓN. ELEMENTOS, ETIQUETAS, ATRIBUTOS Y VALORES

- Haz clic en Aceptar. En lugar de escribir el nombre de la aplicación directamente en el bloque de texto, actualiza un recurso de cadena a la que la propiedad Text del bloque de texto está vinculada. Mediante el uso de un recurso como este, hace que sea más sencillo mantener el texto reutilizable y más fácil de localizar. En MainPage.xaml, el XAML para la definición del recurso Appname se actualiza de esta manera.

```
<x:String x:Key="AppName">Hello, world!</x:String>
```

- En el editor de XAML, agrega los controles para la interfaz de usuario. En la Grid raíz, inmediatamente después de la Grid que contiene el botón Atrás y el título de la página, agrega este XAML. Contiene un stackPanel con un TextBlock que pregunta el nombre del usuario, un elemento TextBox para aceptar el nombre del usuario, un Button y otro elemento TextBlock.

```
<StackPanel Grid.Row="1" Margin="120,30,0,0">
        <TextBlock Text="What's your name?"/>
        <StackPanel Orientation="Horizontal" Margin="0,20,0,20">
                <TextBox x:Name="nameInput" Width="300" HorizontalAlignment="Left"/>
                <Button Content="Say &amp;quot;Hello&amp;quot;"/>
        </StackPanel>
        <TextBlock x:Name="greetingOutput"/>
</StackPanel>
```

2.3 LENGUAJES DE DESCRIPCIÓN DE INTERFACES BASADOS EN XML. ÁMBITO DE APLICACIÓN. ELEMENTOS, ETIQUETAS, ATRIBUTOS Y VALORES

- Presiona F5 para ejecutar la aplicación. Tiene que tener esta apariencia.

Puedes escribir en el elemento TextBox, pero si haces clic en el Button aun no se realiza ninguna acción.

2.3 LENGUAJES DE DESCRIPCIÓN DE INTERFACES BASADOS EN XML. ÁMBITO DE APLICACIÓN. ELEMENTOS, ETIQUETAS, ATRIBUTOS Y VALORES

Crear un controlador de eventos

Los elementos XAML pueden enviar mensajes cuando se producen ciertos eventos. Estos mensajes de eventos te dan la oportunidad de realizar alguna acción como respuesta al evento. El código para responder al evento lo pones en un método de controlador de eventos. Uno de los eventos más comunes en muchas aplicaciones es que un usuario pulse un Button.

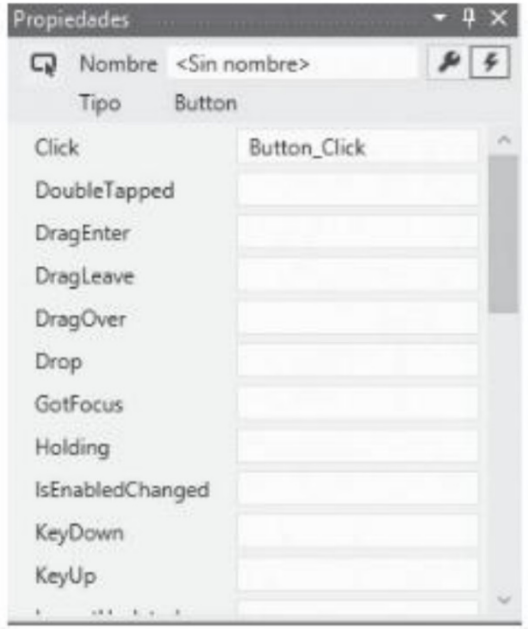
Crearemos un controlador de eventos para el evento Clic del botón. El controlador de eventos obtendrá el nombre del usuario del control TextBox nameInput y lo usará para enviar un saludo al TextBlock greetingOutput.

Tu aplicación también debe poder controlar la entrada desde un ratón o un lápiz. Afortunadamente, los eventos como Click y DoubleTapped no dependen del dispositivo. Si estás familiarizado con la programación de Microsoft .NET, es posible que hayas visto eventos separados para entradas táctiles, de ratón o de lápiz, como MouseMove, TouchMove y stylusMove. En las aplicaciones de la Tienda windows, estos eventos separados se sustituyen por un único evento PointerMoved que funciona igual de bien para entradas táctiles, de ratón o de lápiz.

2.3 LENGUAJES DE DESCRIPCIÓN DE INTERFACES BASADOS EN XML. ÁMBITO DE APLICACIÓN. ELEMENTOS, ETIQUETAS, ATRIBUTOS Y VALORES

Agregar un controlador de eventos

En la vista de diseño o XAML, seleccione el Button "Say Hello" que agregó a MainPage.xaml.
En la Ventana de propiedades, haga clic en el botón de eventos (Botón de eventos).
Busque el evento Clic en la parte superior de la lista de eventos. En el cuadro de texto del evento, escribe el nombre de la función que controla el evento Clic.
Para este ejemplo, escriba "Button_Clic".



Presiona Entrar. El método de controlador de eventos se crea y se abre en un editor de código, de forma que puedes agregar código que se ejecutará cuando se produzca el evento.

2.3.2. XUL

XUL (acrónimo de xML-based user-interface Language, lenguaje basado en XML para la interfaz de usuario) Es un lenguaje basado en XML utilizado para describir y crear interfaces de usuario, que ha sido diseñado para brindar la portabilidad de las mismas, por lo que permite desarrollar aplicaciones multi-plataforma sofisticadas o complejas sin necesidad de herramientas especiales. Inicialmente XUL fue creado para desarrollar los productos de Mozilla (navegador y cliente de e-mail, entre otros) de una forma más rápida y fácil. Al ser un lenguaje basado en XML, contiene todas las características disponibles para XML y sus mismas ventajas.

A diferencia de HTML, XUL provee un gran conjunto herramientas para crear menús, paneles, barras de herramientas, wizards, entre otras. Gracias a esto, no será necesario utilizar un lenguaje de programación propietario o incluir un gran código JavaScript para manejar el comportamiento de la interfaz de usuario.

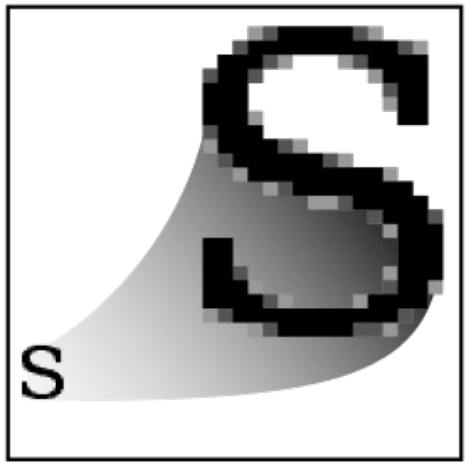
2.3.3. UIML

UIML (acrónimo de user Interface Markup Language, lenguaje de marcas de interfaz de usuario) basado en XML para el diseño de interfaces de usuario en computadores. Permite escribir interfaces de usuario en términos declarativos (por ejemplo como texto) y abstraerlos. La abstracción significa que no debe especificar exactamente como será la interfaz de usuario, aunque si indicará los elementos que serán mostrados y su comportamiento.
Por ejemplo, podrías escribir para escribir un mensaje por pantalla:

```
<part class="DialogMessage" name="HelloWorld"/>
```

2.3.4. SVG

SVG (Scalable Vector Graphics) Gráficos Vectoriales Redimensionables son una especificación para describir gráficos vectoriales bidimensionales, tanto estáticos como animados en formato XML.



Raster
.jpeg .gif .png

Vector
.svg

school everis FORMAMOS PROFESIONALES

2.3.5. MXML

MXML (Macromedia eXtensible Markup Language) . Es un lenguaje que describe interfaces de usuario, crea modelos de datos y tiene acceso a los recursos del servidor, del tipo RIA (Rich Internet Application). MXML tiene una mayor estructura en base a etiquetas, similar a HTML, pero con una sintaxis menos ambigua, proporciona una gran variedad e inclusive permite extender etiquetas y crear sus propios componentes. Una vez compilado genera .swf.

Puesto que los archivos MXML son archivos XML, se puede escribir código MXML en un editor de texto simple, un editor XMLdedicado, o un entorno de desarrollo integrado (IDE) que admita la edición de texto. Flex proporciona un IDEdedicado, llamado Adobe ® Flex ™ Builder ™, que puede utilizar para desarrollar sus aplicaciones.

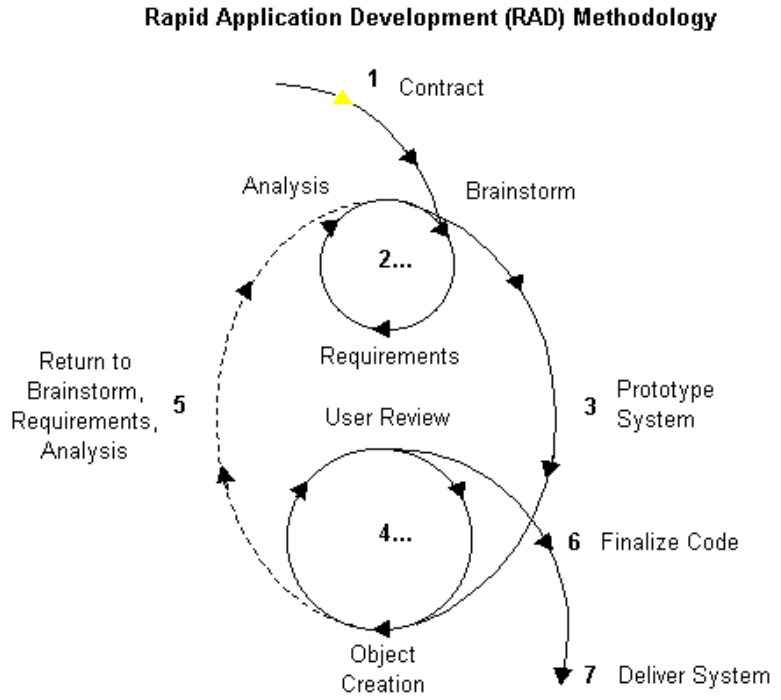
Ejemplo de aplicación "Hola Mundo":

```
<mx:Application
xmlns:mx=http://www.adobe.com/2006/mxml
viewSourceURL="src/HelloWorld/index.html
"horizontalAlign="center" verticalAlign="middle"
width="350" height="150">
        <mx:Panel paddingTop="15" paddingBottom="15" paddingLeft="15"paddingRight="15"title="Test Application">
                <mx:Label text="Hello World!" fontWeight="bold" fontSize="20"/>
        </mx:Panel>
</mx:Application>
```

2.4. HERRAMIENTAS LIBRES Y PROPIETARIAS PARA LA CREACIÓN DE INTERFACES DE USUARIO MULTIPLATAFORMA

Las herramientas de diseño de interfaces graficas de usuario forman parte de las herramientas RAD (rapid Application Development, Desarrollo Rápido de Aplicaciones).
Algunas plataformas serian:

- Glade.
- Blend para Visual Studio.
- Dreamweaver.
- Lazarus.
- Gambas.
- Embarcadero Delphi.
- Visual Foxpro.
- Anjuta.
- Game Maker:Studio.
- GeneXus.
- Clarion.



school everis FORMAMOS PROFESIONALES

2.4.1 GLADE

Glade (o Glade Interface Designer), "Diseñador de interfaces Glade", es una herramienta de desarrollo visual de interfaces gráficas mediante GTK/GNOME. Es independiente del lenguaje de programación y no genera código fuente sino un archivo XML. Se puede utilizar de manera independiente aunque esta integrado en Anjuta. Se encuentra bajo licencia GPL.

Las herramientas de Glade (paleta, editor, etc.) son implementadas como widgets. Esto permite una fácil integración con IDEs como Anjuta o Scaffold, y hace que sea más fácil cambiar la interfaz.

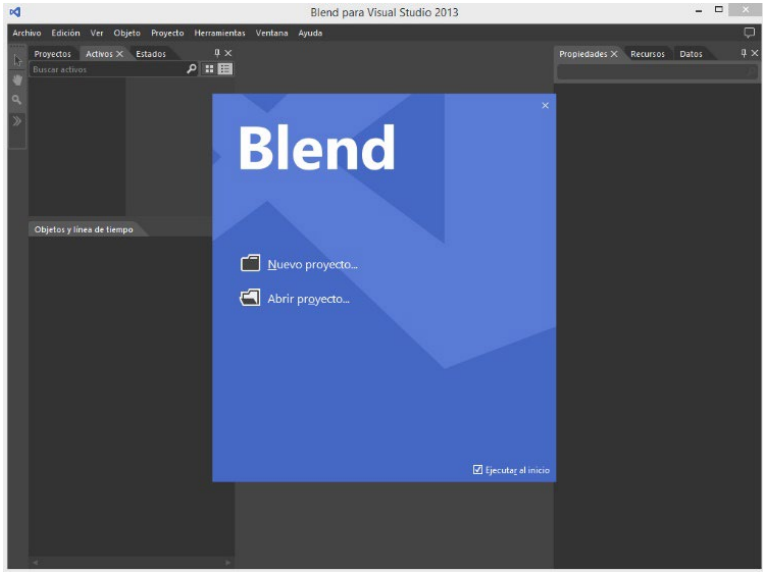
GtkBuilder es un formato XML que Glade usa para almacenar los elementos de las interfaces diseñadas.

Estos archivos pueden emplearse para construirla en tiempo de ejecución mediante el objeto GtkBuilder de GTK+.

school everis FORMAMOS PROFESIONALES

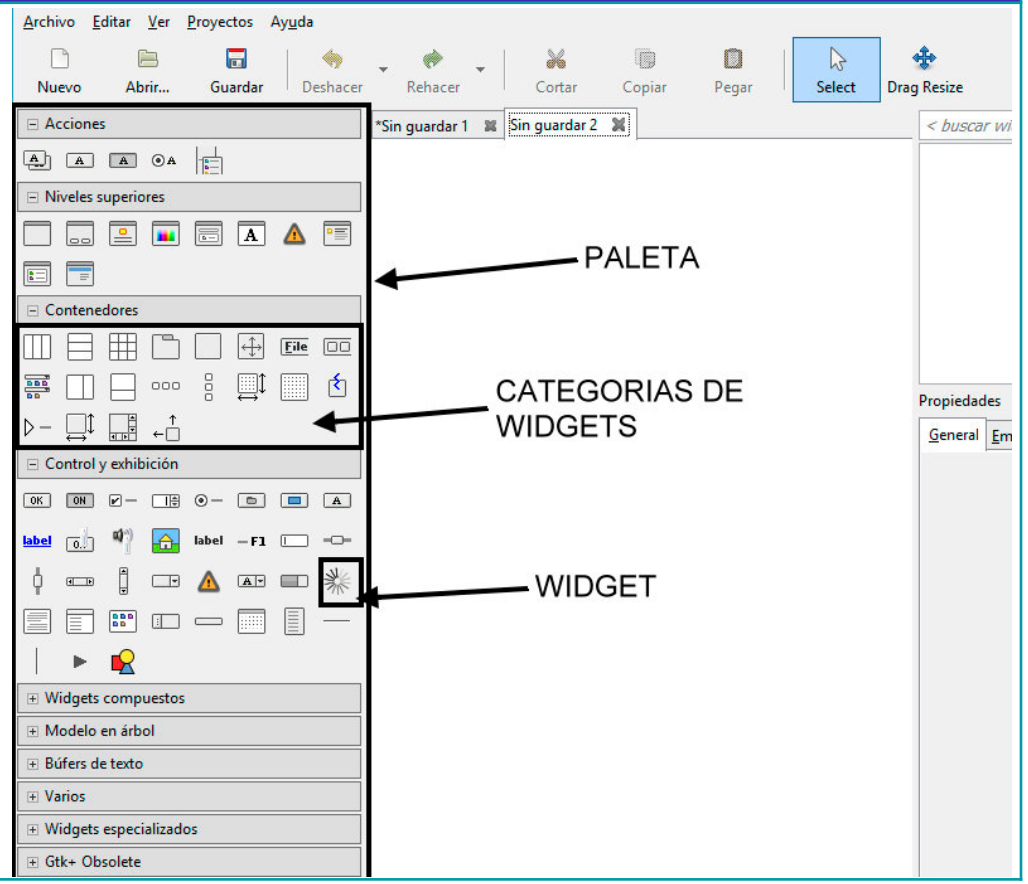
2.4.2 BLEND

Blend para visual Studio, introducido con visual Studio 2012 y mejorado en visual Studio 2013, proporciona capacidades avanzadas centradas en el diseño para crear aplicaciones para la Tienda windows, windows Phone, wPF y Silverlight.
Puede descargar Blend para visual Studio 2013 con visual Studio Express para windows, visual Studio Express para windows Phone y visual Studio Professional 2013 y versiones posteriores.



2.5 PALETAS Y VISTAS

Los **widgets** o **controles** que podemos utilizar desde **Glade** se encuentran en la paleta. Esta paleta aparecerá en el lado izquierdo de la aplicación y mostrará los widgets organizados por categorías: Actions, TopLevels, Containers, Color and Display, Composite widgets y Miscellaneus.



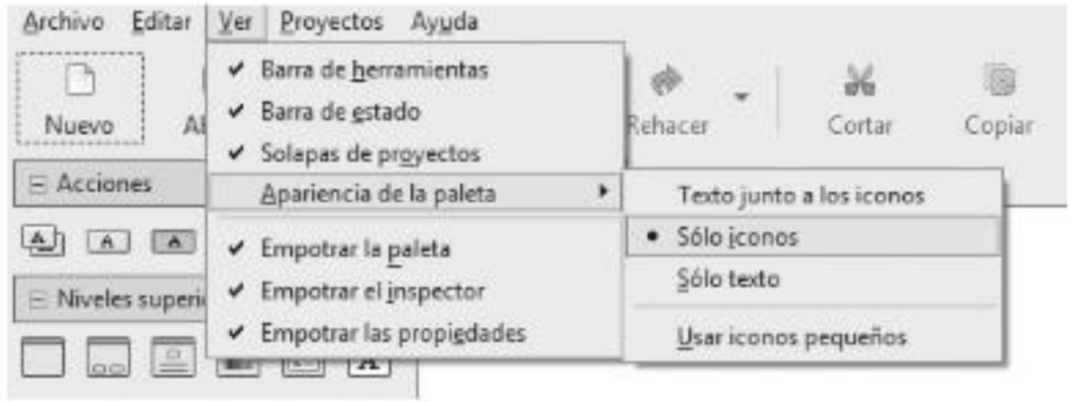
2.5 PALETAS Y VISTAS

Ahora veremos cómo podemos trabajar con los widgets. Para empezar debemos seleccionar los widgets que queramos utilizar en la ventana de Paleta. Esto se puede realizar de tres maneras:

- **Modo de Selección**: que se ejecuta pulsando la flecha de Selector, y se indicará que está activado cuando el puntero del ratón cambie a una flecha, lo que quiere decir que ya podemos seleccionar widgets para nuestro proyecto y editarlos en la ventana de Propiedades. Abriendo el menú contextual del widget (clic derecho) podemos también seleccionar un widget. y podemos desde la misma Paleta añadir varios widgets de un mismo tipo si mantenemos la tecla Control al seleccionarlo. Pulsando otra vez sobre el selector o sobre otro widget en la Paleta volvemos al modo normal.
- **Modo Emplazamiento**: para usarlo debemos seleccionar un widget de la Paleta, y el puntero cambiará para tener forma de cruz. Entonces podemos emplazar el widget seleccionado dentro de los contenedores o widgets de nivel superior. Después el modo vuelve automáticamente a Selección.
- **Modo Emplazamiento de Nivel Superior**: para ello necesitamos seleccionar un widget de nivel superior en la ventana de Paleta y el widget aparecerá directamente en el escritorio, quedando listo para ser editado. Después el modo vuelve automáticamente a Selección.

2.5 PALETAS Y VISTAS

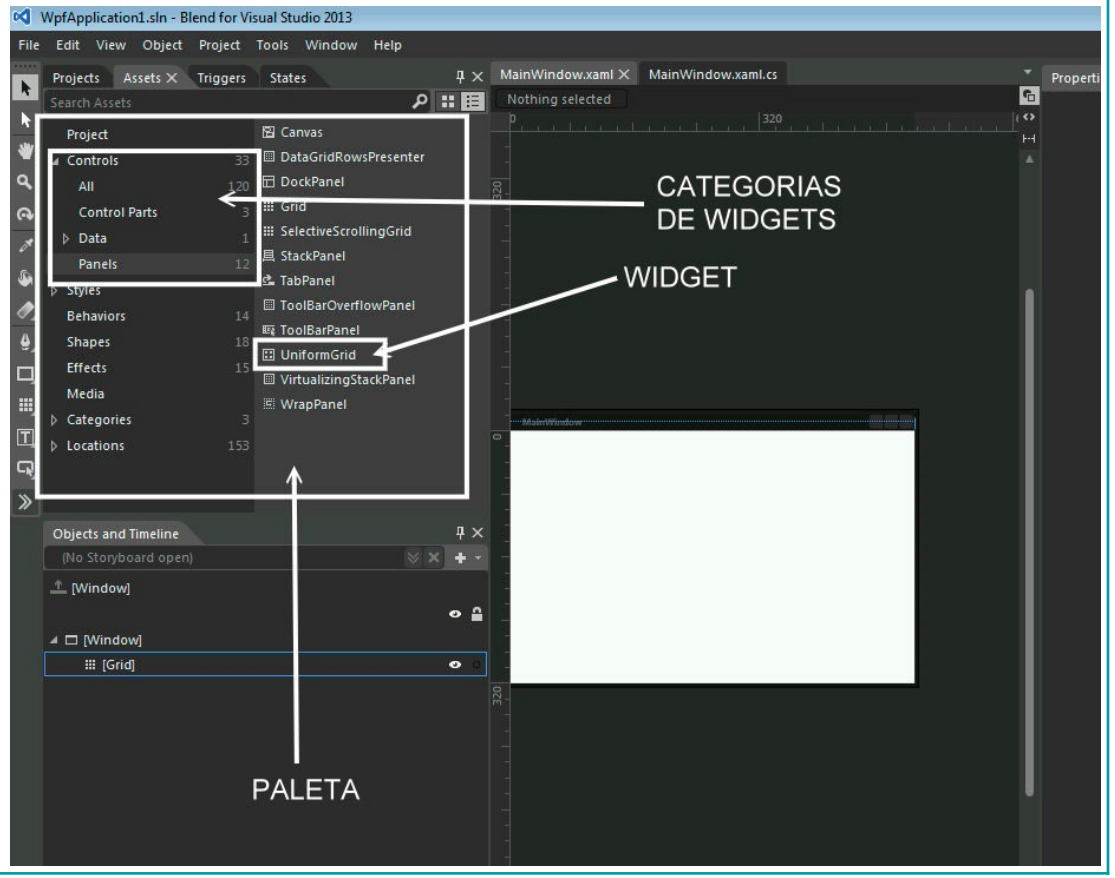
Podemos también cambiar la vista en la que se nos muestran los widgets en la paleta, para ello debemos ir a Ver -> Apariencia de la paleta -> …



2.5 PALETAS Y VISTAS

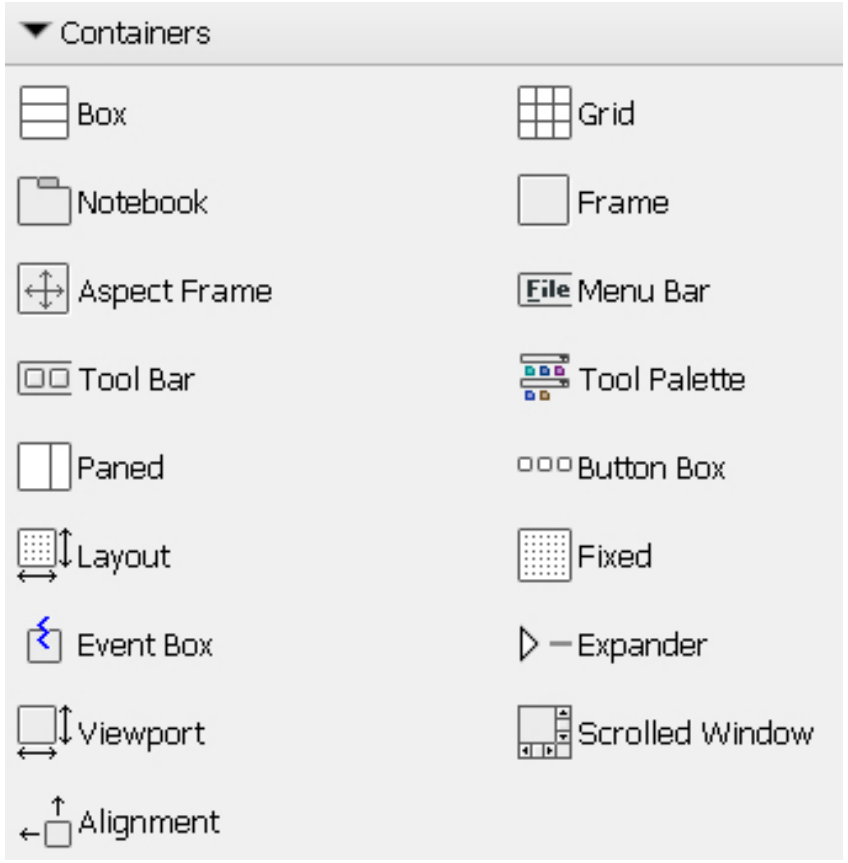
Los controles que podemos utilizar desde Blend se encuentran en la paleta. Esta paleta aparecerá en el lado izquierdo de la aplicación y mostrará los widgets organizados por categorías:

- Project.
- Control.
- Styles.
- Behaviors.
- Shapes.
- Effects.
- Media.
- Categories.
- Locations.



2.6 COMPONENTES CONTENEDORES DE CONTROLES

Una vez seleccionado el widget debemos organizarlo dentro del proyecto, para lo cual es posible utilizar otros widgets contenedores o cajas. Estos widgets están, como todos, situados en la ventana Paleta, y son los siguientes:

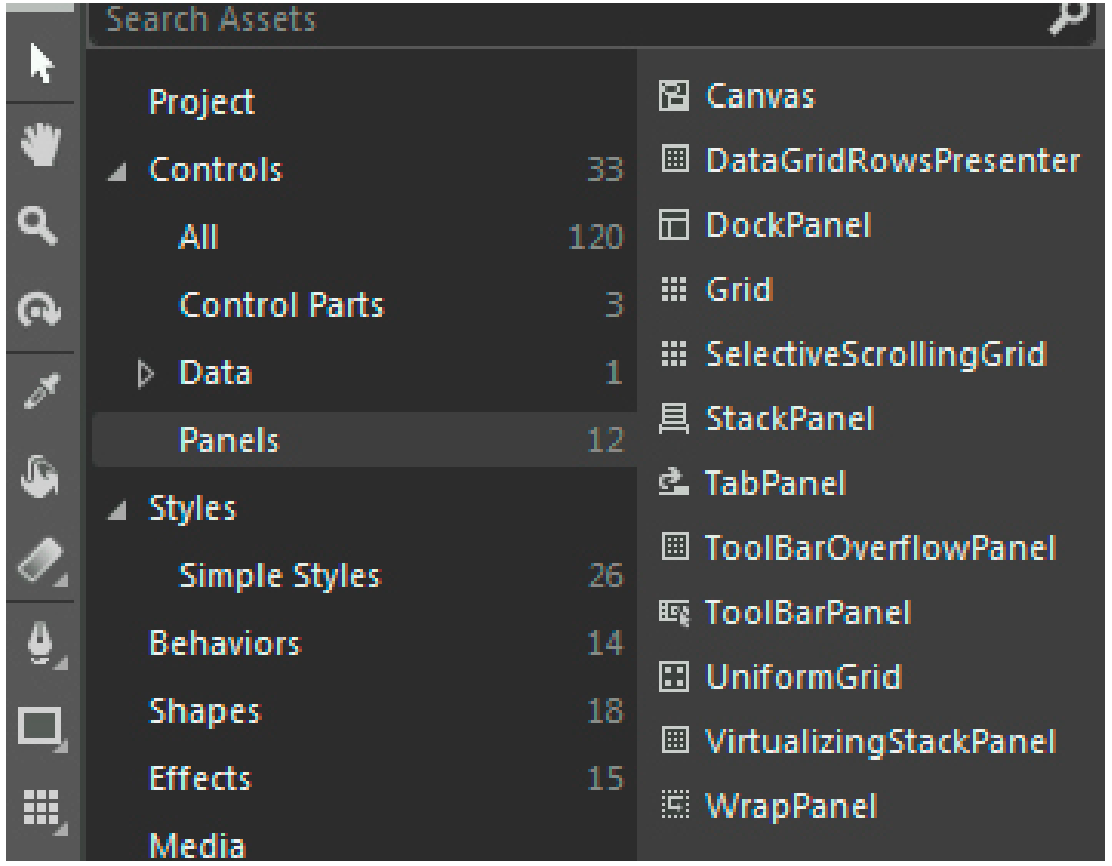


2.6 COMPONENTES CONTENEDORES DE CONTROLES

Estas cajas pueden ser anidadas entre ellas para crear esquemas estructurales más complejos. Una característica de las cajas horizontales y verticales es que Glade te pregunta cuántas filas o columnas quieres crear, aunque más tarde es posible modificar este número, tanto añadiendo como eliminando casillas. Una vez hayamos creado todas las cajas que necesitemos, podemos añadir widgets como etiquetas o botones o incluso algunos más complicados dentro de las mismas. Para facilitar el uso y la vista de estos contenedores, Glade nos ofrece una vista en forma de esquema de los mismos.
Los widgets, una vez en el Portapapeles, pueden ser cortados, copiados, pegados o borrados desde el menú Editar o haciendo en ellos clic derecho seleccionado la opción deseada. Todos estos widgets deben tener un nombre único dentro de cada proyecto de Glade. En caso de que no sea así, las operaciones anteriores nos generará nombres nuevos para los mismos.

2.6 COMPONENTES CONTENEDORES DE CONTROLES

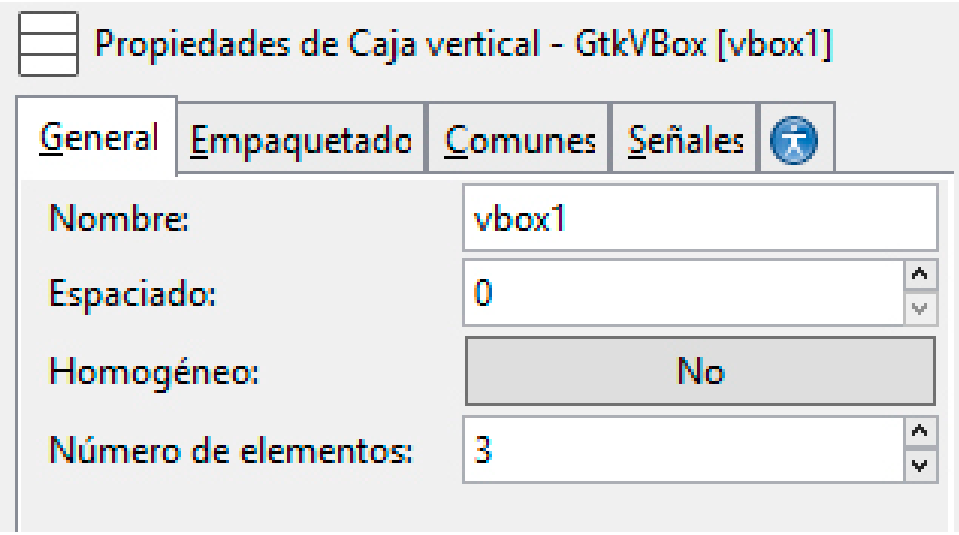
En el caso de **Blend** estos widgets contenedores o cajas estan situados en la ventana de la paleta dentro del menú **Controls -> Data -> Panels**, y son los siguientes:



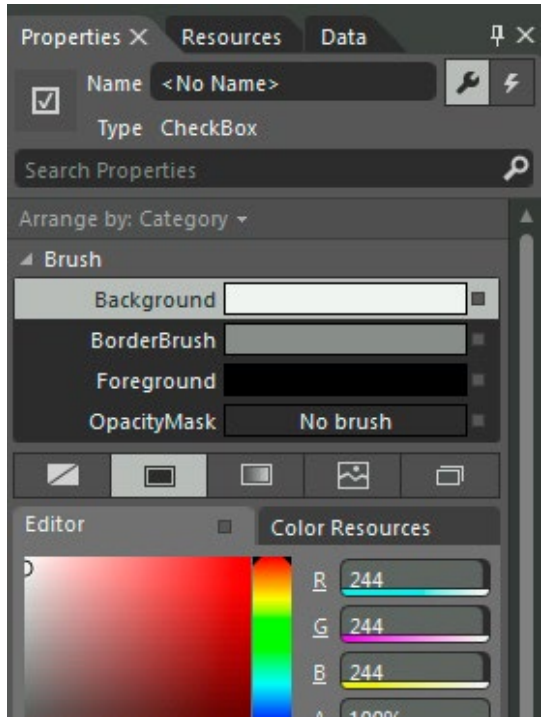
2.7 CONTROLES, PROPIEDADES

Todos los controles (o casi todos) que disponen los entornos de desarrollo de interfaces tienen una serie de propiedades que pueden ser modificadas. Ejemplos de estas propiedades son: el tamaño, la posición, el color, etc.

Glade



Blend

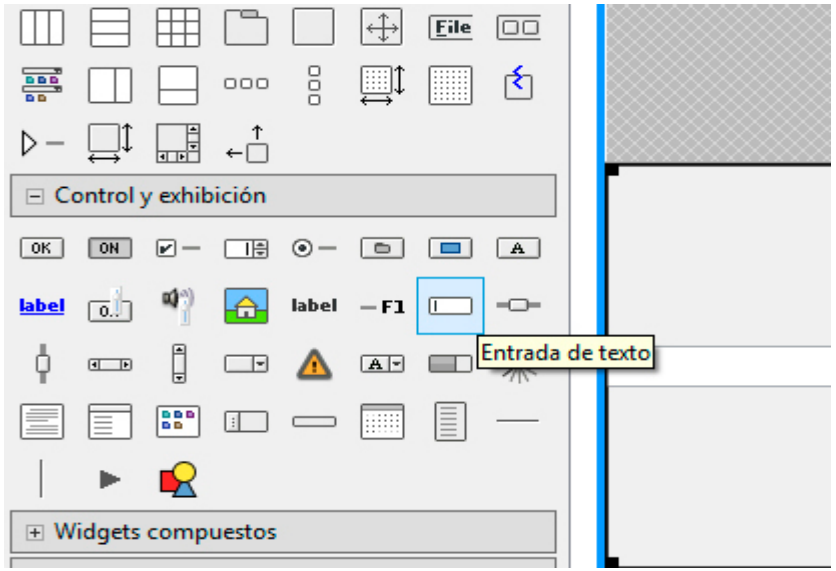


2.7 CONTROLES, PROPIEDADES

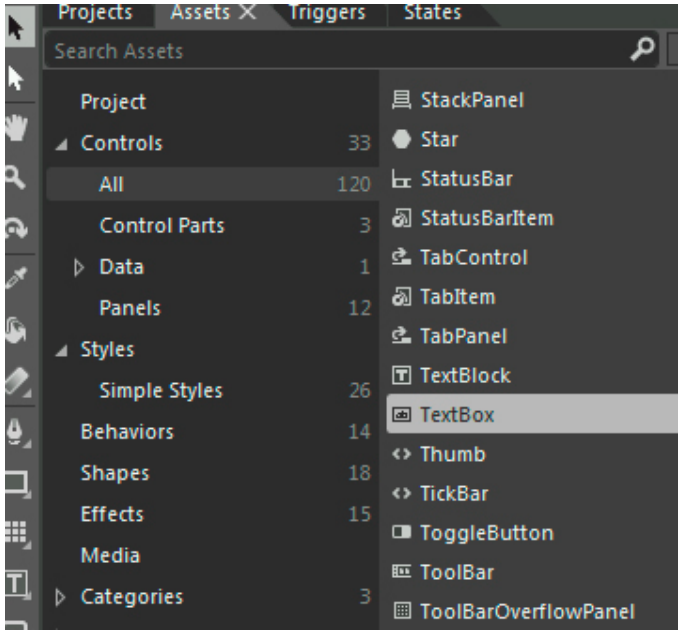
Se presentan ahora los controles más comunes que parecen en los diferentes entornos de desarrollo de interfaces.

- **Entrada de texto** (Text entry o TextBox): podemos introducir datos en nuestras aplicaciones a través de este control, también es posible mostrar información desde nuestra aplicación en estos controles.

Glade



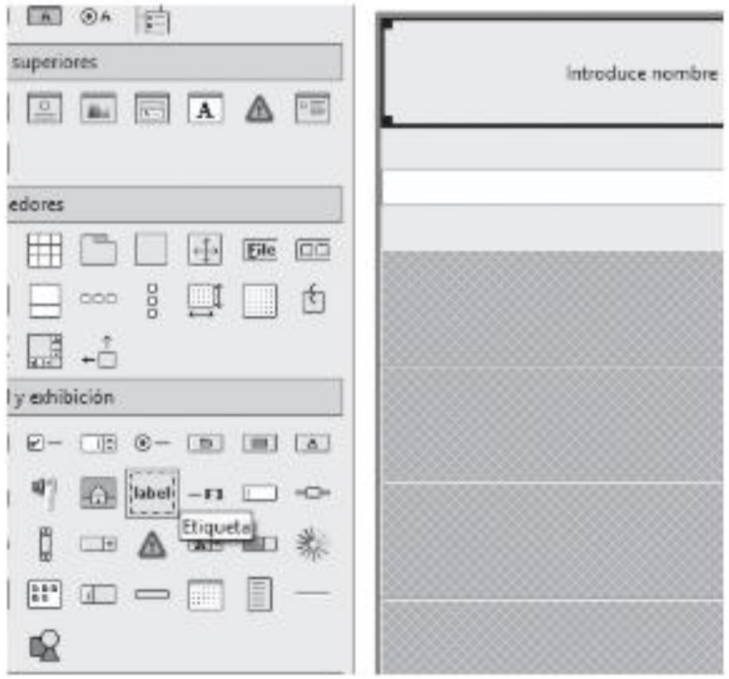
Blend



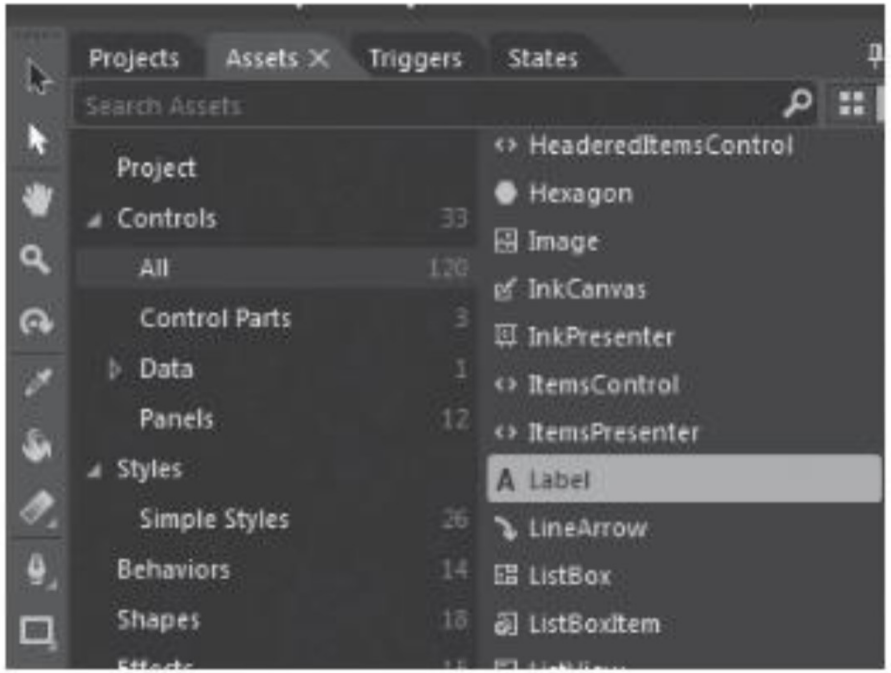
2.7 CONTROLES, PROPIEDADES

- **Etiqueta (Label):** nos permite mostrar por pantalla texto estático, ya que, a diferencia del anterior, no puede modificarse su contenido en la ejecución de la aplicación.

Glade

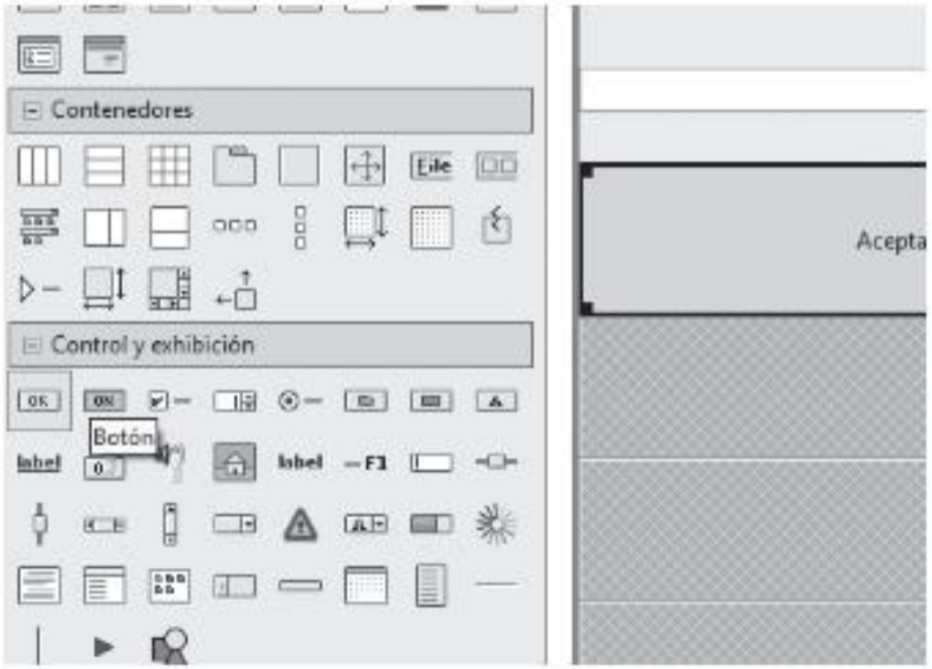


Blend

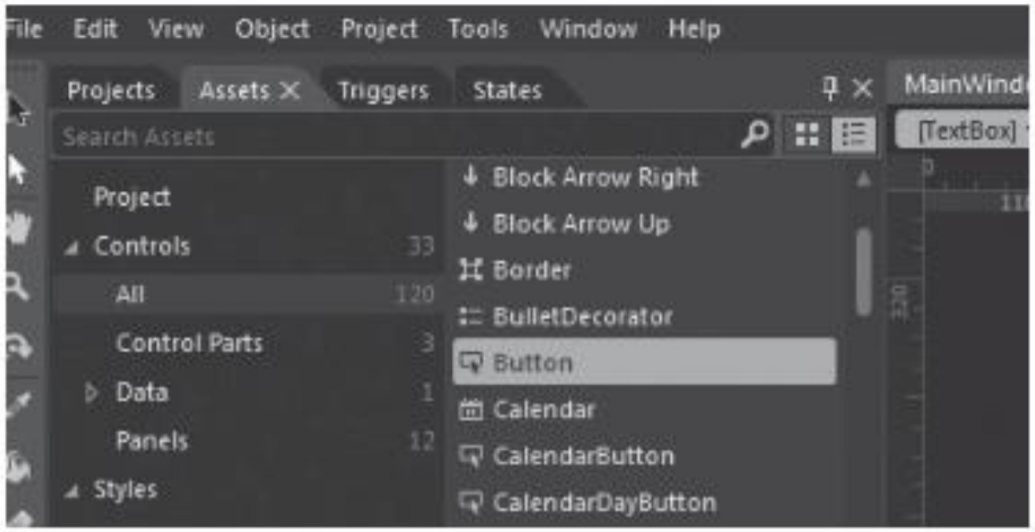


2.7 CONTROLES, PROPIEDADES

- **Botón (Button):** nos permite realizar operaciones concretas al presionar sobre el mismo. Normalmente el texto que aparezca en el botón deberá informar acerca de la función del mismo (Imprimir, Salir, Borrar, etc.).



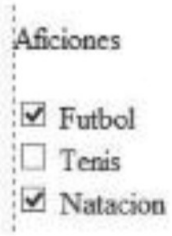
Glade



Blend

2.7 CONTROLES, PROPIEDADES

- Opciones (radio / radioButton) y casillas de verificación (Check / CheckBox): en las primeras (radio) solo se permite seleccionar una opción de las que se presentan. En las segundas (check) se permiten seleccionar ninguna, varias o incluso todas las opciones que se nos presentan.



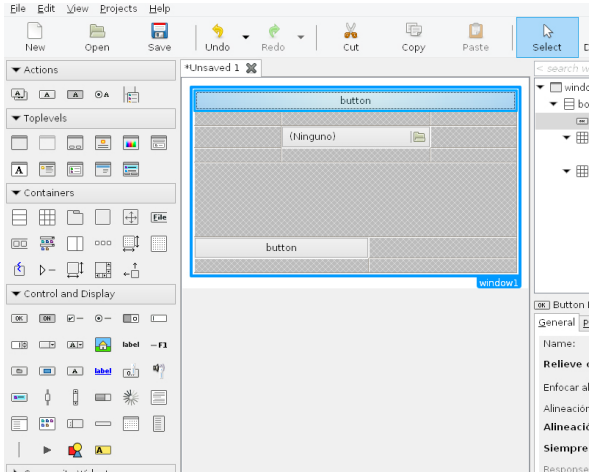
2.7 CONTROLES, PROPIEDADES

- Lista desplegable (Combo / select / ComboBox): se nos despliega una lista para seleccionar el valor deseado.

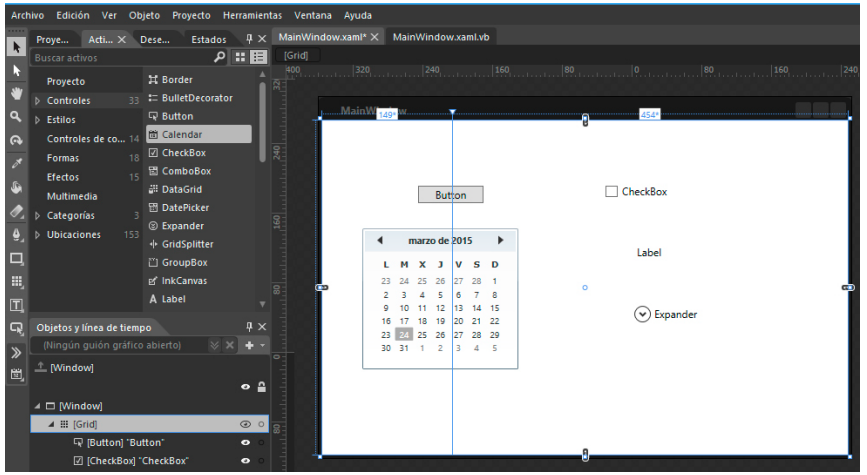
2.8 UBICACIÓN, TAMAÑO Y ALINEAMIENTO DE CONTROLES

Podemos ubicar los controles en cualquier parte de la interfaz que estamos diseñando, al igual podemos variar su tamaño y alineación.
Con Glade una vez hemos definido los contenedores adecuados para nuestra aplicación solamente deberemos arrastrar los controles deseados a los contenedores creados a tal efecto. El tamaño, posición y la alineación vendrán indicados por la distribución de los contenedores seleccionado previamente por nosotros.

Glade



Blend



2.9 EVENTOS, CONTROLADORES. SECUENCIA DE LOS EVENTOS

La programación dirigida por eventos es un paradigma de programación en el que tanto la estructura como la ejecución de los programas van determinados por los sucesos que ocurran en el sistema, definidos por el usuario o que ellos mismos provoquen.

Para entender la programación dirigida por eventos, podemos oponerla a lo que no es: mientras en la programación secuencial (o estructurada) es el programador el que define cuál va a ser el flujo del programa, en la programación dirigida por eventos será el propio usuario (o lo que sea que esté accionando el programa) el que dirija el flujo del programa. Aunque en la programación secuencial puede haber intervención de un agente externo al programa, estas intervenciones ocurrirán cuando el programador lo haya determinado, y no en cualquier momento como puede ser en el caso de la programación dirigida por eventos.

El creador de un programa dirigido por eventos debe definir los eventos que manejarán su programa y las acciones que se realizarán al producirse cada uno de ellos, lo que se conoce como el administrador de evento. Los eventos soportados estarán determinados por el lenguaje de programación utilizado, por el sistema operativo e incluso por eventos creados por el mismo programador.

2.9 EVENTOS, CONTROLADORES. SECUENCIA DE LOS EVENTOS

En la programación dirigida por eventos, al comenzar la ejecución del programa se llevarán a cabo las inicializaciones y demás código inicial y a continuación el programa quedará bloqueado hasta que se produzca algún evento. Cuando alguno de los eventos esperados por el programa tenga lugar, el programa pasará a ejecutar el código del correspondiente administrador de evento. Por ejemplo, si el evento consiste en que el usuario ha hecho clic en el botón de play de un reproductor de películas, se ejecutará el código del administrador de evento, que será el que haga que la película se muestre por pantalla.

Un ejemplo claro lo tenemos en los sistemas de programación Léxico y visual Basic, en los que a cada elemento del programa (objetos, controles, etcétera) se le asignan una serie de eventos que generará dicho elemento, como la pulsación de un botón del ratón sobre él o el redibujado del control.

La programación dirigida por eventos es la base de lo que llamamos interfaz de usuario, aunque puede emplearse también para desarrollar interfaces entre componentes de Software o módulos del núcleo.

Con la aparición y popularización de los PC, el software empezó a ser demandado para usos alejados de los clásicos académicos y empresariales para los cuales era necesitado hasta entonces, y quedó patente que el paradigma clásico de programación no podía responder a las nuevas necesidades de interacción con el usuario que surgieron a raíz de este hecho.

2.10 ANÁLISIS y EDICIÓN DEL DOCUMENTO XML

La **validación XML** es la comprobación de que un documento en lenguaje XML está bien formado y se ajusta a una estructura definida. Un documento bien formado sigue las reglas básicas de XML establecidas para el diseño de documentos. Un documento válido además respeta las normas dictadas por su DTD (definición de tipo de documento) o esquema utilizado.

La validación se encarga de verificar:

- La **corrección de los datos**: aunque validar contra un esquema no garantiza al 100% que los datos son correctos, nos permite detectar formatos nulos o valores fuera de rango y por tanto incorrectos.
- La **integridad de los datos**: al validar, se comprueba que toda la información obligatoria está presente en el documento.
- El **entendimiento compartido de los datos**: a través de la validación se comprueba que el emisor y receptor perciban el documento de la misma manera, que lo interpreten igual.

La creación manual de documentos XML e incluso su manipulación pueden introducir todo tipo de errores, tipográficos, sintácticos y de contenido. Existen editores de XML que facilitan la tarea de crear documentos válidos y bien formados, ya que pueden advertir de los errores básicos cometidos e incluso escribir automáticamente la sintaxis más sencilla necesaria.

2.10 ANÁLISIS y EDICIÓN DEL DOCUMENTO XML

Cuando necesitamos obtener un documento válido, el editor XML ha de ser capaz de:

- Leer la DTD del documento y presentarle una lista desplegable con los elementos disponibles enumerados en la DTD, evitando así la inclusión de algún elemento no definido en el esquema.
- Advertir el olvido de una etiqueta obligatoria e incluso no permitir este tipo de descuidos o errores, no dando por finalizado el documento si existen errores de este tipo.

A falta de un editor de XML (o como alternativa), pueden crearse documentos XML con editores de texto convencionales. Las labores de validación de instancias de documentos XML y de DTD y Esquemas XML, las de generación automática de DTD y esquemas XML a partir de un documento XML dado, o las de transformación de una DTD en un esquema XML, pueden entonces encomendarse a herramientas de validación y conversión que tenemos disponibles como servicio web.

2.10 ANÁLISIS y EDICIÓN DEL DOCUMENTO XML

Así, por ejemplo:

- validación XML:
  http://www.w3schools.com/xml/xml_validator.asp (w3 Schools).
  http://www.stg.brown.edu/service/xmlvalid (Brown U.; Richard Goerwitz).
  http://www.xmlvalidation.com/ (EPSOforum).
  http://www.validome.org/xml/ (validome.org).
- validación DTD / Schema:
  http://www.w3.org/2001/03/webdata/xsv (w3 Consortium).
  http://www.validome.org/grammar/ (validome.org).
  http://schneegans.de/sv/ (Christoph Schneegans).
- Herramientas de conversión:
  http://www.hitsw.com/xml_utilites/

2.10.1 EDITORES DE XML

Un editor de XML es un editor de lenguaje de marcado con funcionalidades añadidas para facilitar la edición de XML. Esto podemos hacerlo con un editor de texto plano, con todo el código visible, pero los editores de XML han añadido facilidades como finalización de etiquetas, menús y botones para las tareas que son comunes en la edición de XML, en base a datos suministrados con la definición de tipo de documento (Document Type Definition o DTD) o el árbol XML.
También tenemos editores gráficos XML que ocultan el código en el fondo y presentan el contenido al usuario en un formato más fácil de usar, aproximándola a la versión renderizada o a la edición de formularios. Esto es útil para situaciones en las que las personas que no dominan el código XML necesitan introducir la información de los documentos basados en XML, tales como hojas de tiempo y los informes de gastos. E incluso si el usuario está familiarizado con XML, el uso de estos editores, que nos ayudan con los detalles de la sintaxis, es más rápido y más conveniente.

school everis FORMAMOS PROFESIONALES

2.10.1 EDITORES DE XML

- Gratuitas:
  XMLPad 3 de wMHelp (http://www.wmhelp.com/) (windows).
  XML Copy Editor, comunitario, con licencia GNU (http://xml-copy-editor.sourceforge.net/) (windows,Linux).
  XMLQuire, de Qutoric (http://qutoric.com/xmlquire/) (windows).
  XMLSpear, de Donkey Development (http://www.donkeydevelopment.com) (windows, Linux, Mac; programado en Java).
  XPontus XML Editor, comunitario, con licencia GNU (http://xpontus.sourceforge.net) (windows, Linux, Mac).
- versiones gratuitas de software comercial:
  EditiX XML Editor, de JAPISoft (http://www.editix.com) (comercial, con evaluación de 30 días; versión gratuita Free XML Editor, con funcionalidad reducida, en http://www.freexmleditorsite.com) (windows, Linux, Mac).

2.10.1 EDITORES DE XML

- Comerciales, con período de evaluación:
  Altova XMLSpy, de Altova (http://www.altova.com, http://www.xmlspy.com) (evaluación: 30 días) (windows).
  Liquid XML Studio, de Liquid Technologies (http://www.liquid-technologies.com/xml-studio.aspx) (evaluación: 30 días) (windows).
  <oXygen/> XML Editor, de SyncRO Soft (http://www.oxygenxml.com) (evaluación: 30 días) (windows, Mac, Linux, Solaris).
  Serna Enterprise XML editor, de InfoTrust (http://www.serna-xmleditor.com) (sin período de evaluación) (windows, Linux, Mac).
  Stylus Studio, de Progress Software (http://www.stylusstudio.com) (evaluación: 15 días) (windows)
  XMetaL, de JustSystems (http://xmetal.com) (evaluación de 30 días) (windows).
  XMLmind XML Editor, de Pixware (http://www.xmlmind.com/xmleditor) (evaluación: 30 días) (windows, Linux, Mac).
  XMLwriter, de wattle Software (http://xmlwriter.net) (evaluación: 30 días) (windows).

2.11 GENERACIÓN DE CÓDIGO PARA DIFERENTES PLATAFORMAS

Los documentos XML se procesan a través de analizadores, aplicaciones que leen el documento, lo interpretan y generan una salida basada en sus contenidos y en la marca utilizada para su descripción. El resultado se muestra en un dispositivo de visualización, como una ventana de navegación o una impresora. Los procesadores hacen posible la presentación y distribución de documentos XML.
Una aplicación que consume información XML debe leer un fichero de texto codificado según dicho estándar, cargar la información en memoria y, desde allí, procesar esos datos para obtener unos resultados (que posiblemente también almacenará de forma persistente en otro fichero XML).
El proceso anterior se enmarca dentro de lo que se conoce en informática como "parsing" o análisis léxico-sintáctico, y los programas que lo llevan a cabo se denominan "**parsers**" o **analizadores** (léxico-sintácticos). Más específicamente podemos decir que el "parsing XML" es el proceso mediante el cual se lee y se analiza un documento XML para comprobar que está bien formado para, posteriormente, pasar el contenido de ese documento a una aplicación cliente que necesite consumir dicha información.

2.11 GENERACIÓN DE CÓDIGO PARA DIFERENTES PLATAFORMAS

Hay dos tipos de analizadores para documentos XML: analizadores dirigidos por la estructura (parsers DOM) y analizadores orientados a eventos (parsers SAX).

- DOM: XML es un metalenguaje de estructura jerárquica, las marcas dentro de un documento XML tiene una relación padre-hijo estricta. Esta estructura lleva a que la representación natural para documentos de este tipo sea la denominada estructura de "árbol". DOM ofrece, a través de sus interfaces, una visión abstracta de esa estructura de árbol. Cada uno de los elementos constructivos del documento XML viene representado por objetos de tipo "nodo" basado en los principios del diseño orientado a objetos. Estos nodos serán: elementos, atributos, comentarios, instrucciones de procesamiento, etc.
- SAX: El API SAX no especifica cómo ha de ser la implementación del parser, sino como ha de comportarse, es decir, especifica interfaces de programación y no clases. Por lo tanto, SAX no es ningún programa concreto, sino una especificación abstracta que envuelve (mediante la técnica de "layering" habitual en POO) a implementaciones de parsers XML distintos.

En su núcleo SAX está compuesto por dos interfaces: el XMLReader que representa al parser; y el ContentHandler que recibe datos del parser. Estos dos interfaces son suficientes para realizar el 90% de la funcionalidad que podemos necesitar de SAX.

school everis FORMAMOS PROFESIONALES

2.11 GENERACIÓN DE CÓDIGO PARA DIFERENTES PLATAFORMAS

XML no ha nacido solo para su aplicación en Internet, sino que se propone como un estándar para el intercambio de información estructurada entre diferentes plataformas. Cuando nos referimos a plataformas incluimos Linux, Android, iOS, Microsoft en sus desarrollos visual Studio o XPS. También se puede usar en bases de datos, editores de texto, hojas de cálculo, dispositivos móviles, portátiles y casi cualquier cosa imaginable.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android=http://schemas.android.com/apk/res/android android:id="@+id/principal" android:layout_width="fill_parent"
android:layout_height="fill_parent" android:orientation="vertical">
        <TextView>
                android:id="@+id/etiqueta1"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="Esto es una etiqueta de texto">
        </TextView>
        <Button>
                android:id="@+id/boton1"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="Esto es un botón">
        </Button>
</LinearLayout>
```

XML es la base para la creación de interfaces gráficas de muchos generadores de código o IDEs como visual Studio, Android Studio, NetBeans y Eclipse.