

MPAEE: Expression Blend



**Escuela Politécnica Superior
Universidad de Alicante**

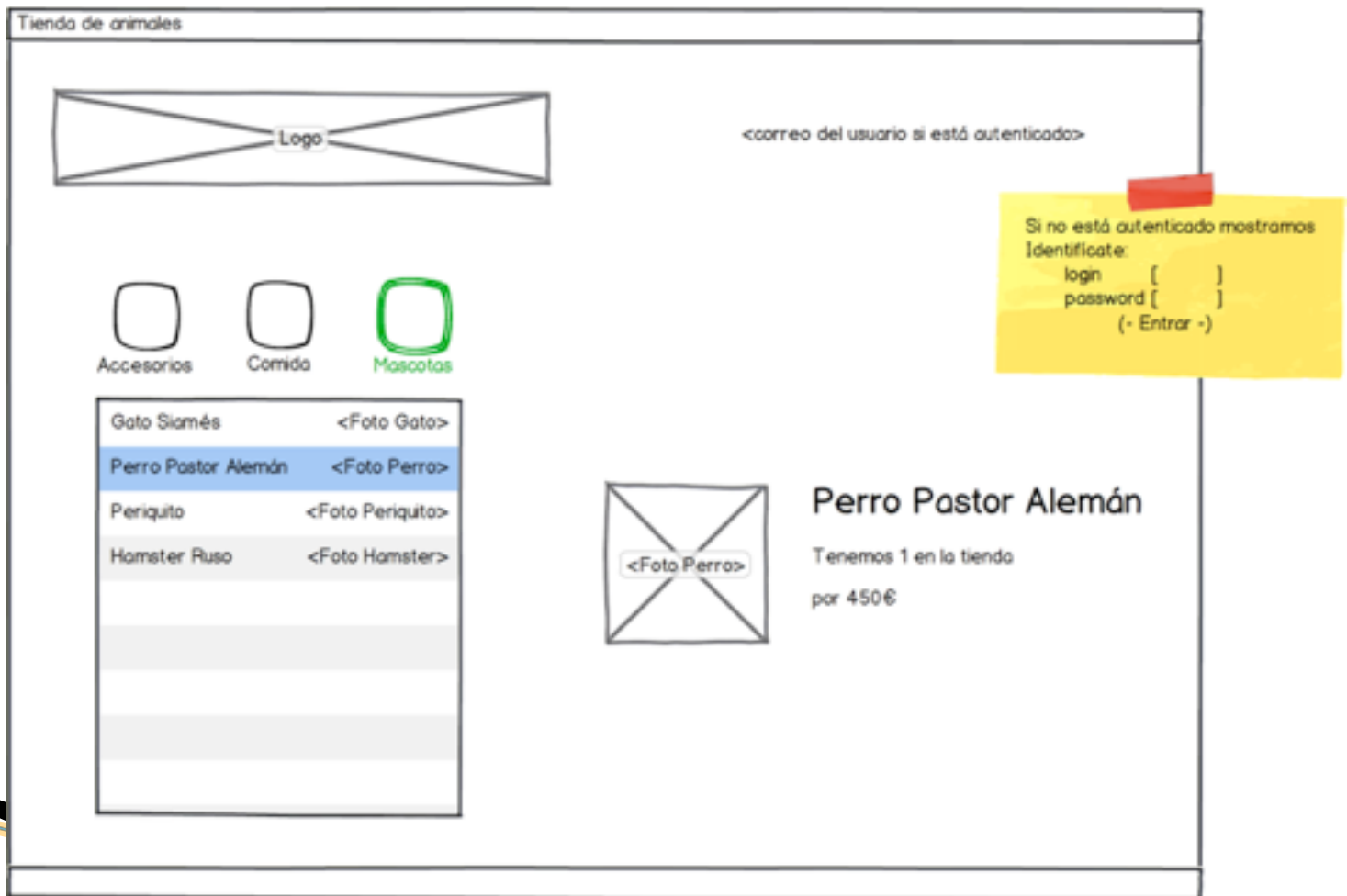
Especificación

► Casos de uso

- consultar el catálogo de productos de la tienda de mascotas

Usuario	Sistema
(si el usuario está autenticado en el sistema va a querer saberlo)	
Elige el tipo de producto o categoría (mascotas, comida, accesorios) Es improbable que se añadan más categorías	Muestra todos productos que pertenecen a la categoría
Elige uno de los productos	Muestra los detalles del producto

Mockup



Guía de estilo: logotipo

► Logotipo



(www.verdezoo.eu)

- usos válidos



(otros ...)

- usos no válidos



(otros ...)

Guía de estilo: tipografías

Arial

ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz
1234567890

Arial Bold

**ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz
1234567890**

DIN-MEDIUM

ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz
1234567890

* Esta tipografía siempre deberá utilizarse como imagen, al no estar disponible (como norma) en los sistemas operativos.





DIN Schrift
1451 Mittelschrift

ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz
1234567890

* Esta tipografía siempre deberá utilizarse como imagen, al no estar disponible (como norma) en los sistemas operativos.

Guía de estilo: colores

Textos

	Titulares color: #366595
	Entradilla color: #333333
	Antetítulo color: #666666
	Texto color: #333333







Cajas, bordes

	Fondos color: #FDFDFD
	Borde color: #CECECE
	Fondo texto rollover color: #666666
	Fondo cajas Modulos color: #f4f4f4

Fondos con degradado

	Fondos textos: color 1: #FDFDFD color 2: #F4F4F4
	botones: color 1: #7EB3D5 color 2: #3075A3
	botones: color 1: #E07F1F color 2: #EDA444
	botones: color 1: #B8B7B7 color 2: #808080
	botones: color 1: #38A8D4 color 2: #49D4EE
	botones: color 1: #808080 color 2: #B8B7B7

Botones

	Complementario 1 color: #E07F1F
	Complementario 2 color: #EDA444
	BOT 1 color: #3075A3
	BOT 2 color: #7EB3D5
	BOT 1 color: #B8B7B7
	BOT 2 color: #808080

Ejemplos de uso

					
normal	rollover	normal	rollover	normal	rollover
					
		normal	rollover	normal	rollover



* Este tipo de botones siempre como imagen, ya que la tipografía utilizada (DIN regular) no es de sistema.

Guía de estilo: iconos

Ejemplos

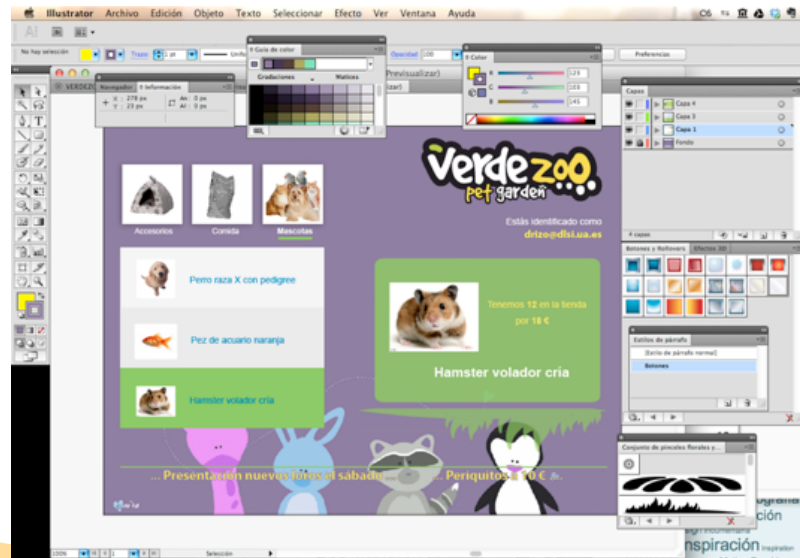


Guía de estilo

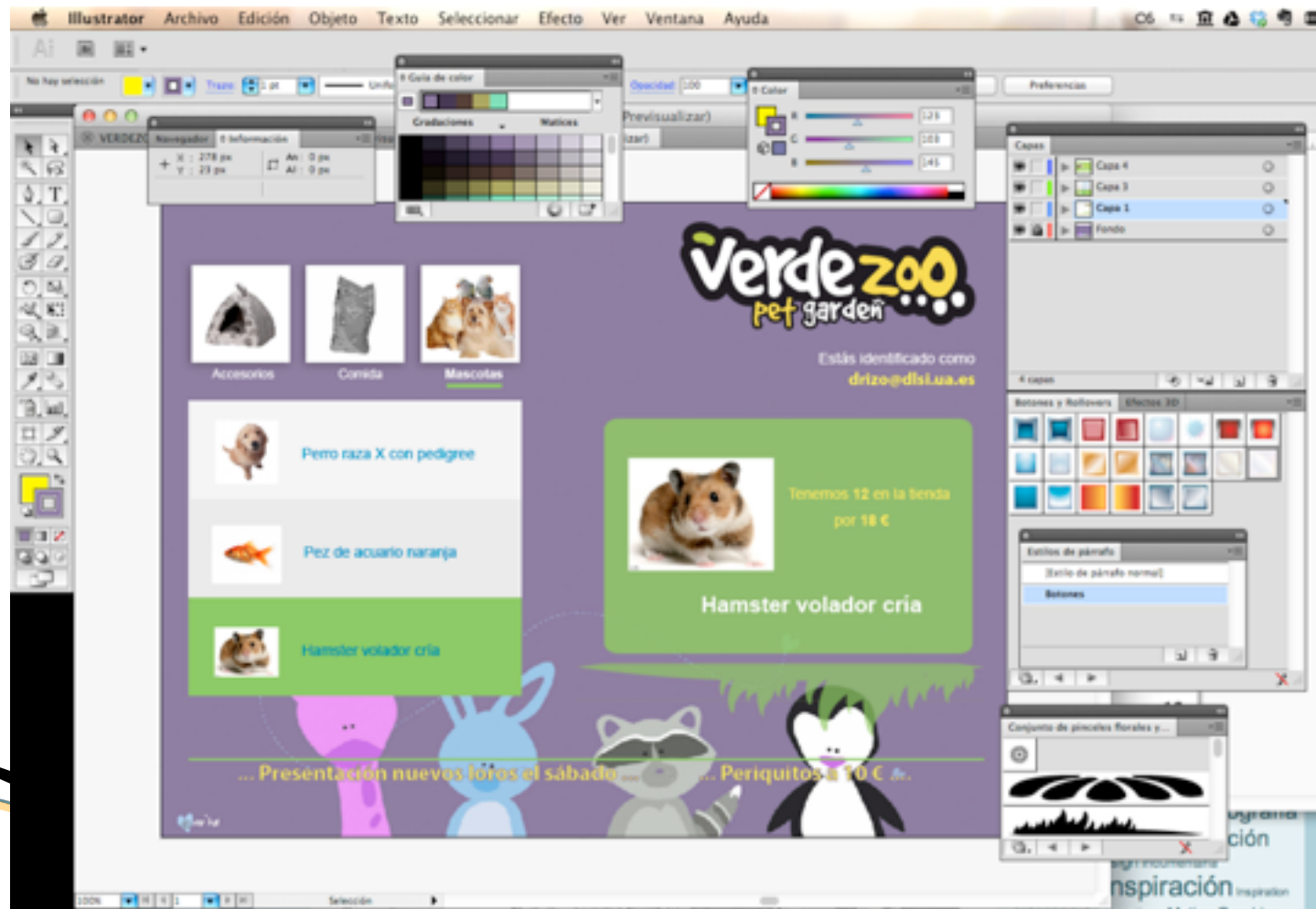
- ▶ Otras consideraciones a tener en cuenta
 - botones
 - orden (p.ej Cancelar izquierda, Aceptar derecha)
 - texto: Aceptar | Cancelar ó Sí | No
 - tipos de control preferido
 - p.ej. por encima de 4 opciones usamos listBox, por bajo radioButton
 - fondos de pantalla
 - imágenes
 - animaciones
 - sonidos
 - etc...

Diseño alta fidelidad

- ▶ (no tenemos en cuenta la guía de estilo anterior por ser un ejemplo diferente al petStore)
- ▶ Realizamos una propuesta en una aplicación de diseño vectorial



Diseño alta fidelidad

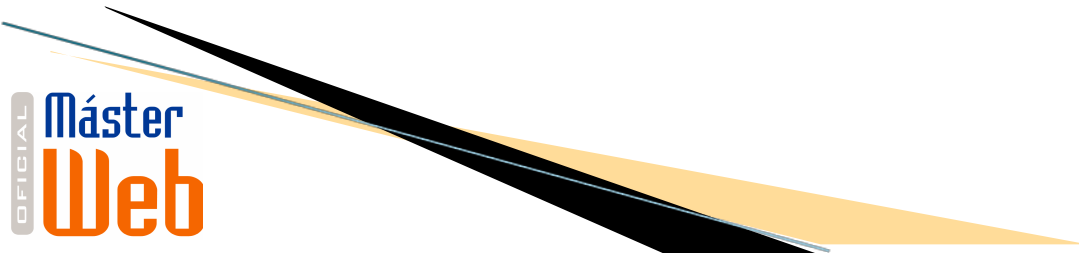


¿Una herramienta vectorial más?

- ▶ Herramienta de diseño vectorial
 - como Inkscape, Illustrator, ...
 - mucho más limitada
- ▶ Pero
 - el diseñador trabaja sobre en el mismo entorno para el que vamos a programar
 - los problemas de compatibilidad (desde AI, PSD) se resuelven en Expression Design por el diseñador
 - exportamos a XAML

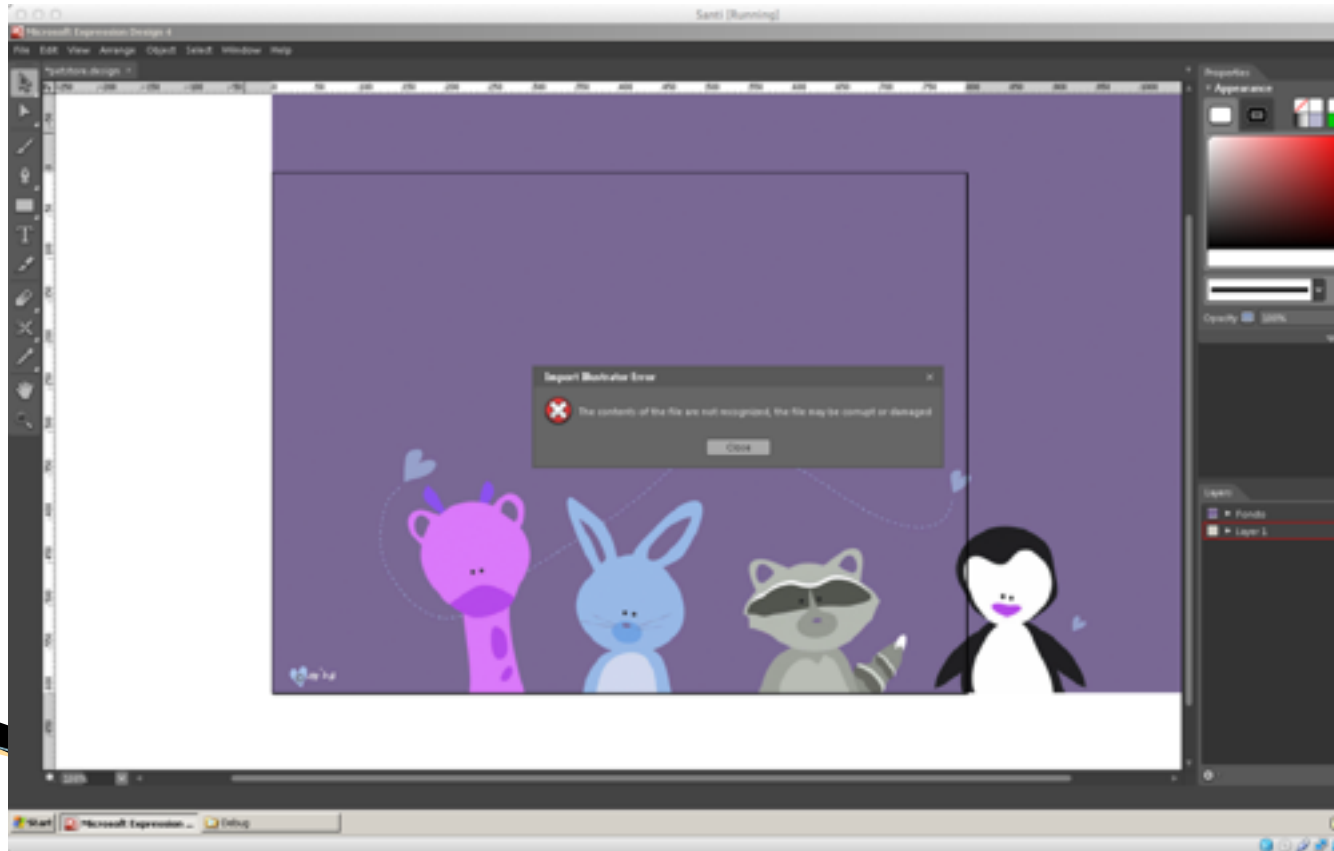
Expression Design

- ▶ Creamos un nuevo proyecto
 - resolución web y OsX: 72ppp
 - resolución Windows: 96ppp
- ▶ Hacemos la pantalla inicial del PetStore a partir del fichero vectorial hecho en Illustrator



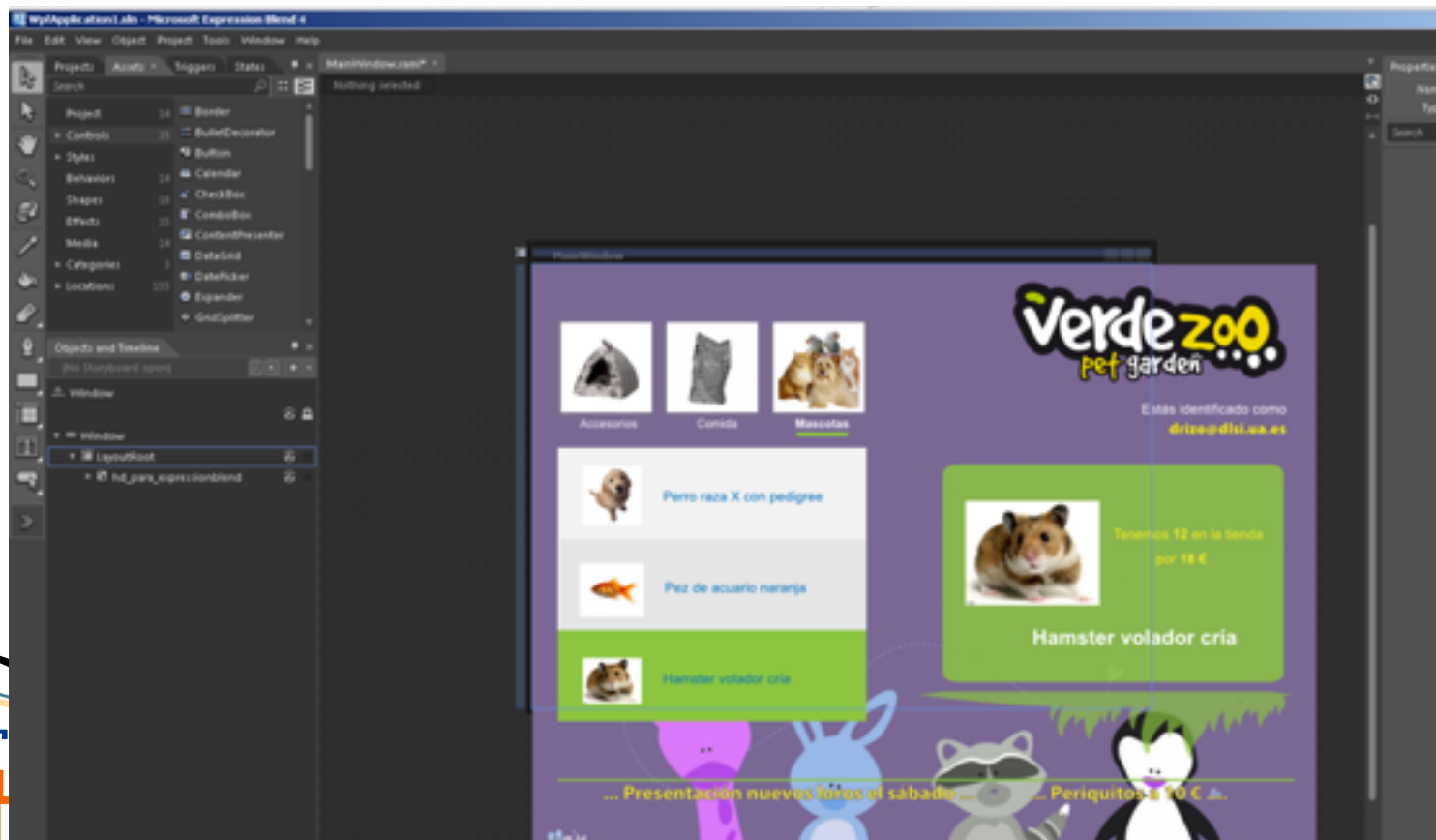
Expression Design

➤ Posibles errores de importación



Expression Blend

- Sin embargo, sin error en Expression Blend, descartamos E.Design



Expression Blend

- ▶ Vamos a integrar la solución actual con el nuevo diseño



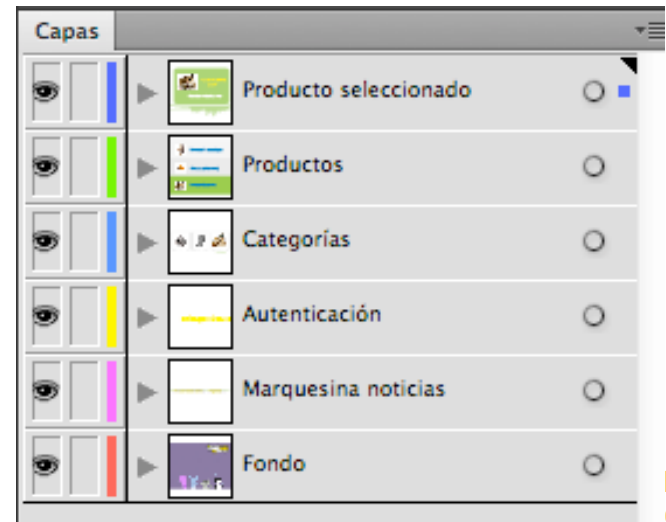
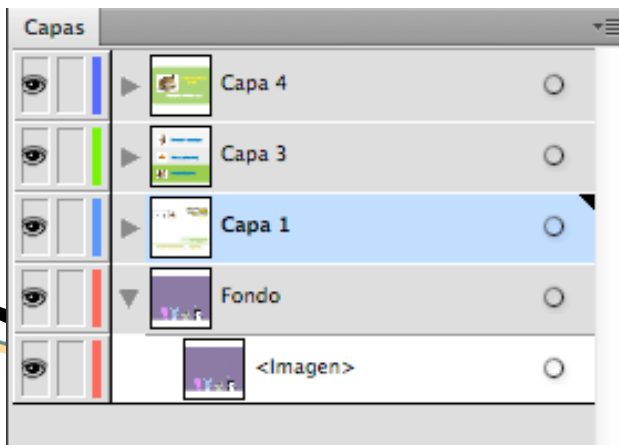
Expression Blend

- ▶ Vamos a integrar la solución actual con el nuevo diseño



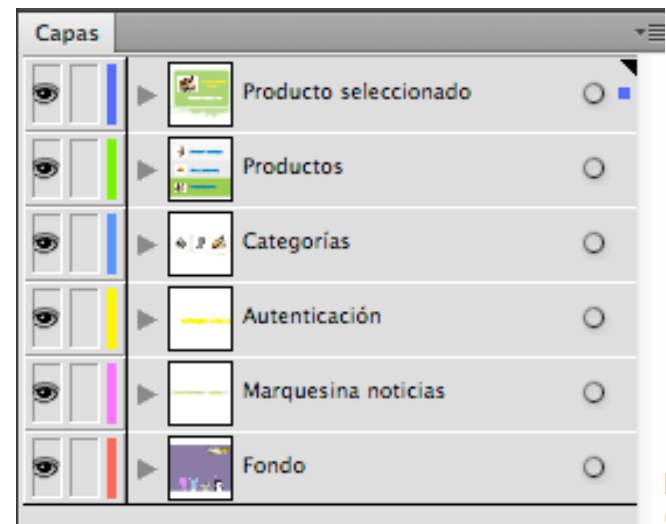
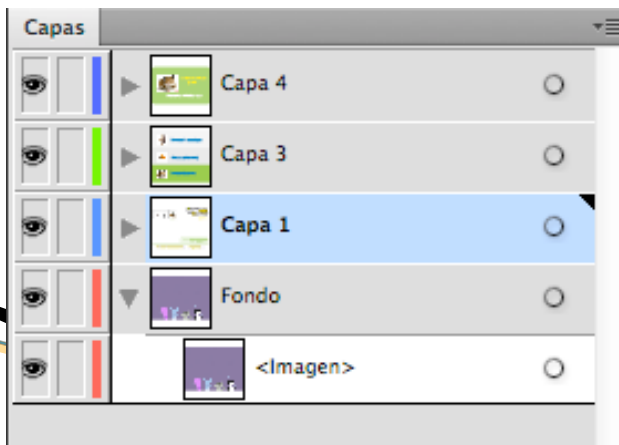
Importación .AI

- “ Abrimos la solución .sln desde Expression Blend
- “ Importamos el .AI en el MainWindow.xaml
 - es importante que las capas en el AI estén organizadas según la maquetación que queramos realizar



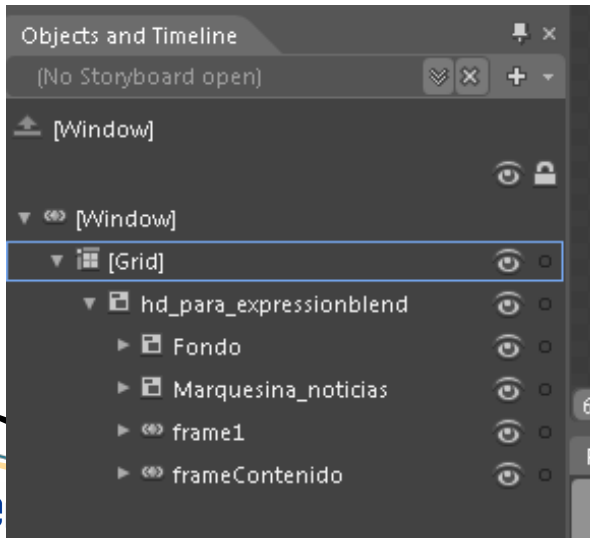
Importación .AI

- “ Abrimos la solución .sln desde Expression Blend
- “ Importamos el .AI en el MainWindow.xaml
 - es importante que las capas en el AI estén organizadas según la maquetación que queremos realizar



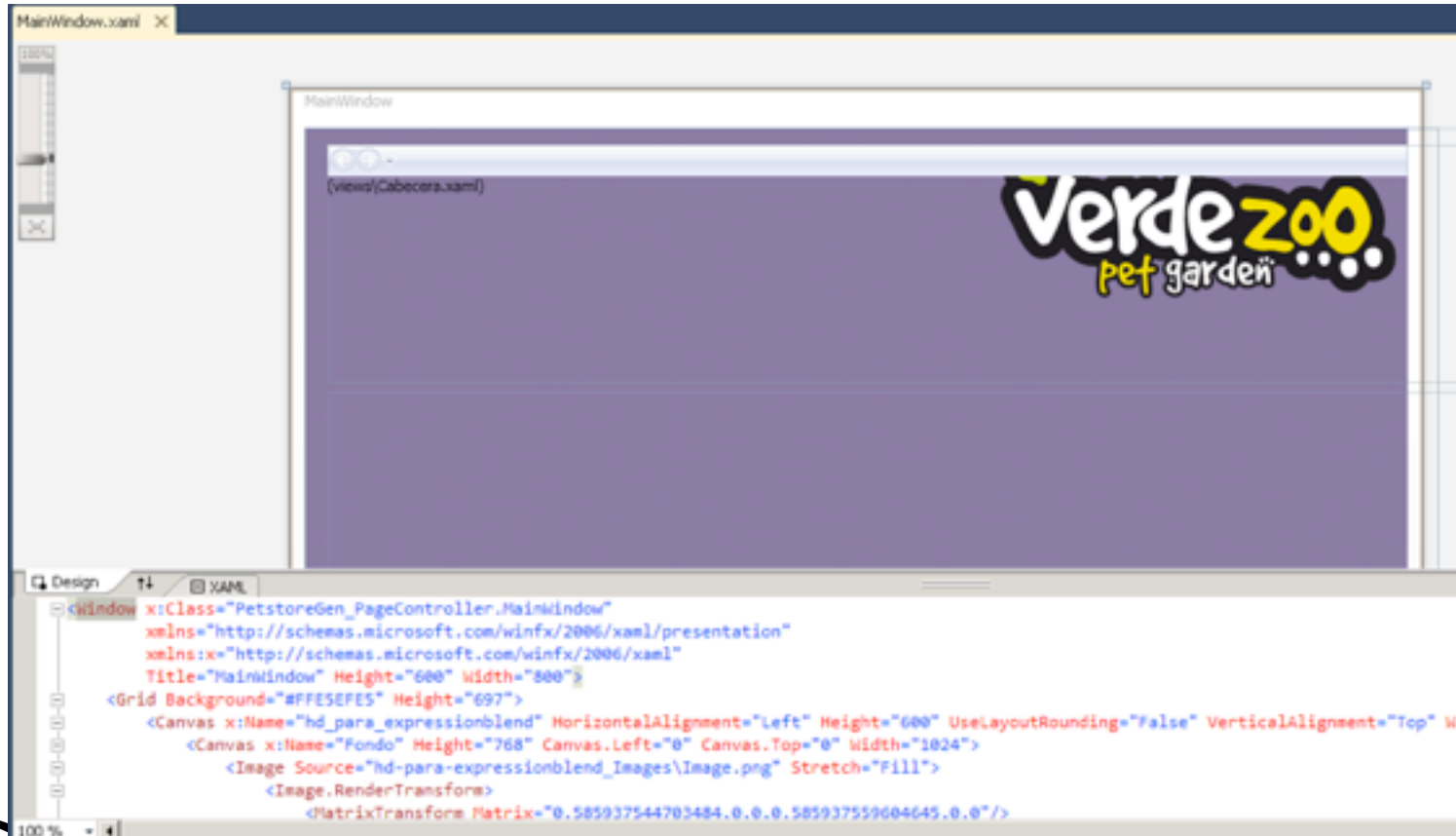
Importación .AI fondo

- “ Reorganizamos las capas quitando las que no necesitamos en la pantalla principal
- “ Redimensionamos



Importación .AI fondo

“ Podemos comprobar cómo se actualiza en VisualStudio



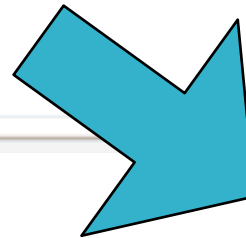
Importación .AI fondo

“ ... y cómo va quedando si lo ejecutamos



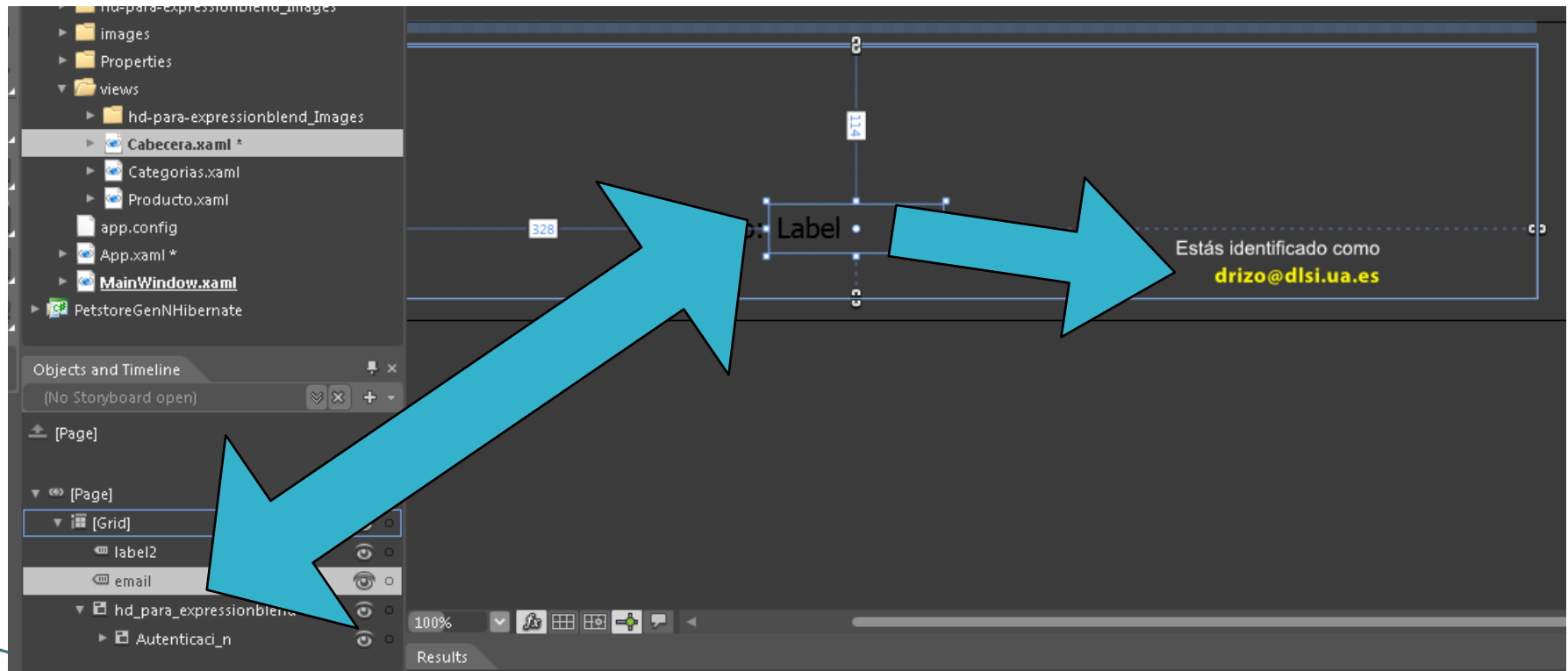
Cabecera

- “ Actualizamos la cabecera, para ello importamos de nuevo el AI usando la vista Cabecera.xaml y cambiamos tamaños
 - podríamos haber aprovechado las capas que hemos quitado antes en la ventana principal



Cabecera

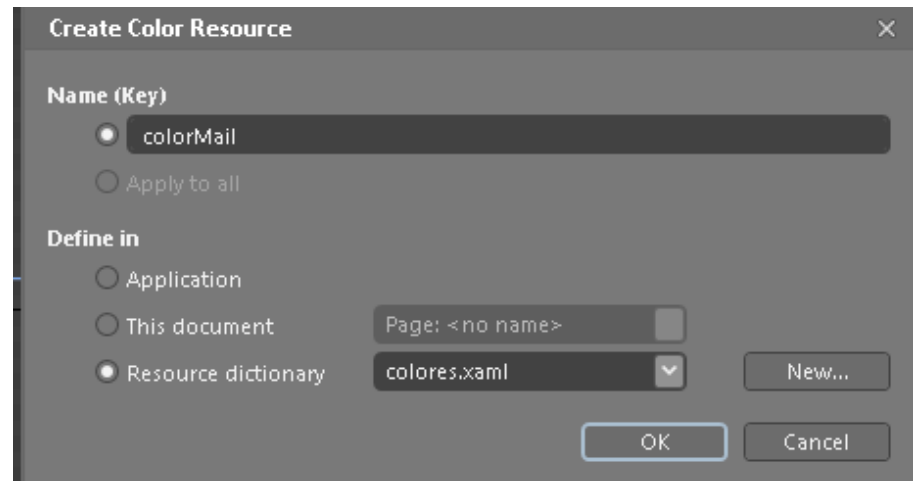
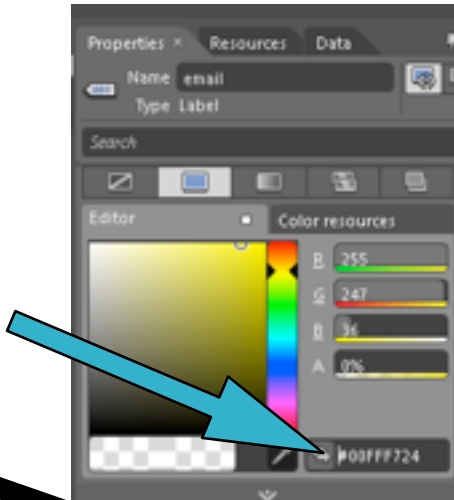
“ Actualizamos la cabecera, para ello importamos de nuevo el AI usando la vista Cabecera.xaml



Cabecera

- “ Los textos se importan como trazados
- “ Creamos nuevo Text para el e-mail manteniendo el antiguo nombre de control (Label no permite color frente)
- “ Creamos un “Resource Dictionary” para crear ahí los estilos (color, fuente)

*convert color
to resource*



Cabecera

“ Corregimos los errores de compilación por cambio de Label a Text

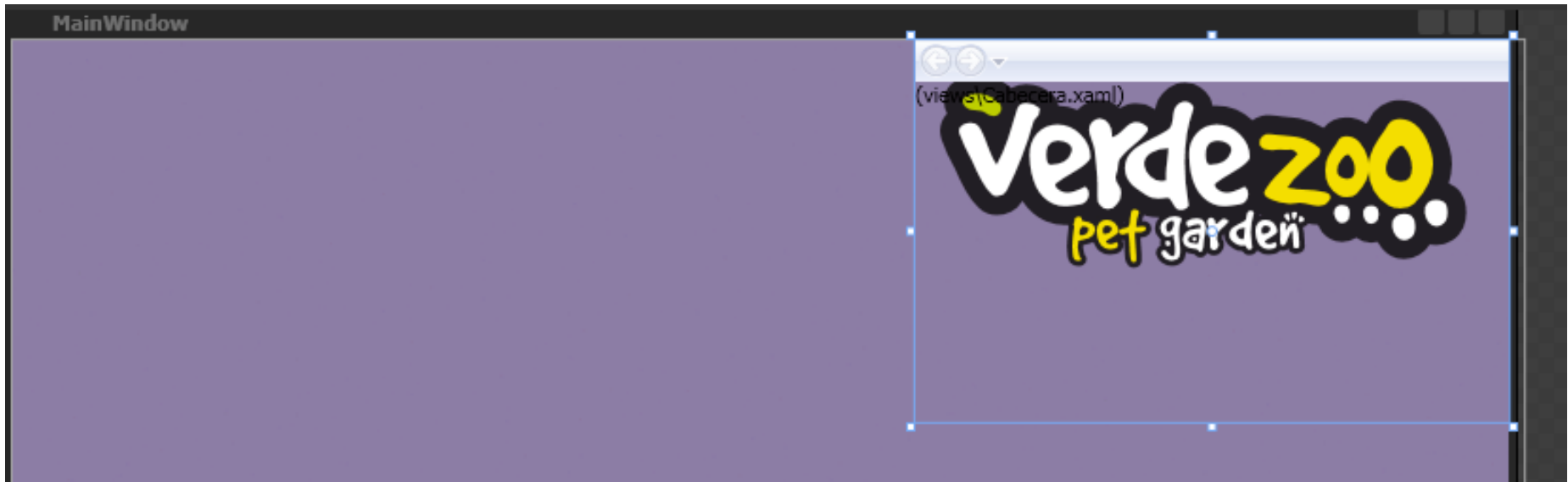
```
namespace PetstoreGen_MVP.views
{
    /// <summary>
    /// Interaction logic for Page1.xaml
    /// </summary>
    public partial class PageController : Page, IVistaMaster
    {
        PresenterMaster presenter = null;
        public PageController()
        {
            presenter = new PresenterMaster(this);
            InitializeComponent();
            presenter.IniciarVista();
        }

        public string Email {
            set{
                email.Content = value;
            }
        }
    }
}
```

email.Text = value;

Cabecera

- “ Cambiamos la dimensión del contenedor para que se adecúe al nuevo diseño



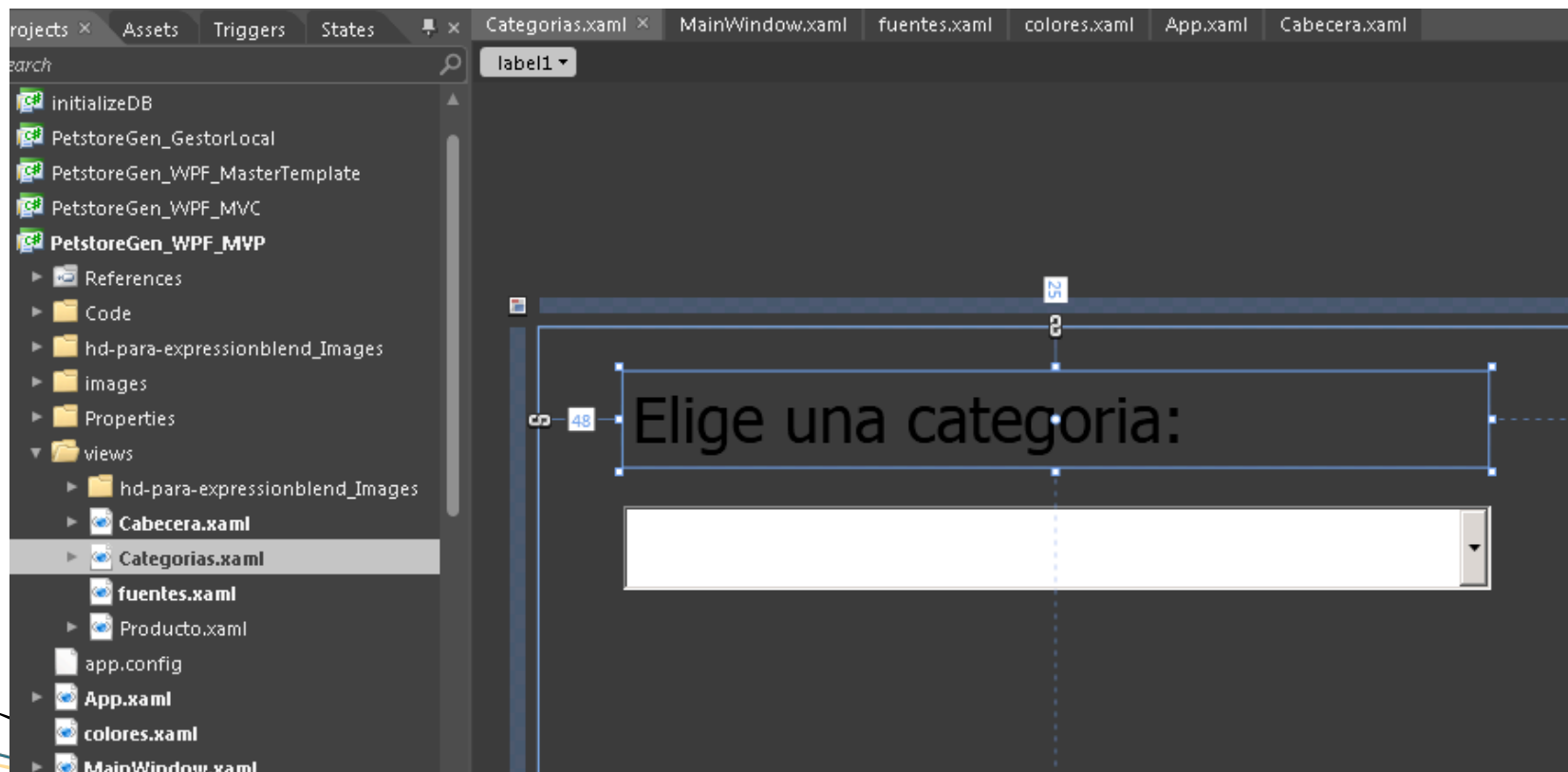
Cabecera

“ Estado actual de ejecución



Categorías

“ Cambiamos la vista de categorías



Categorías

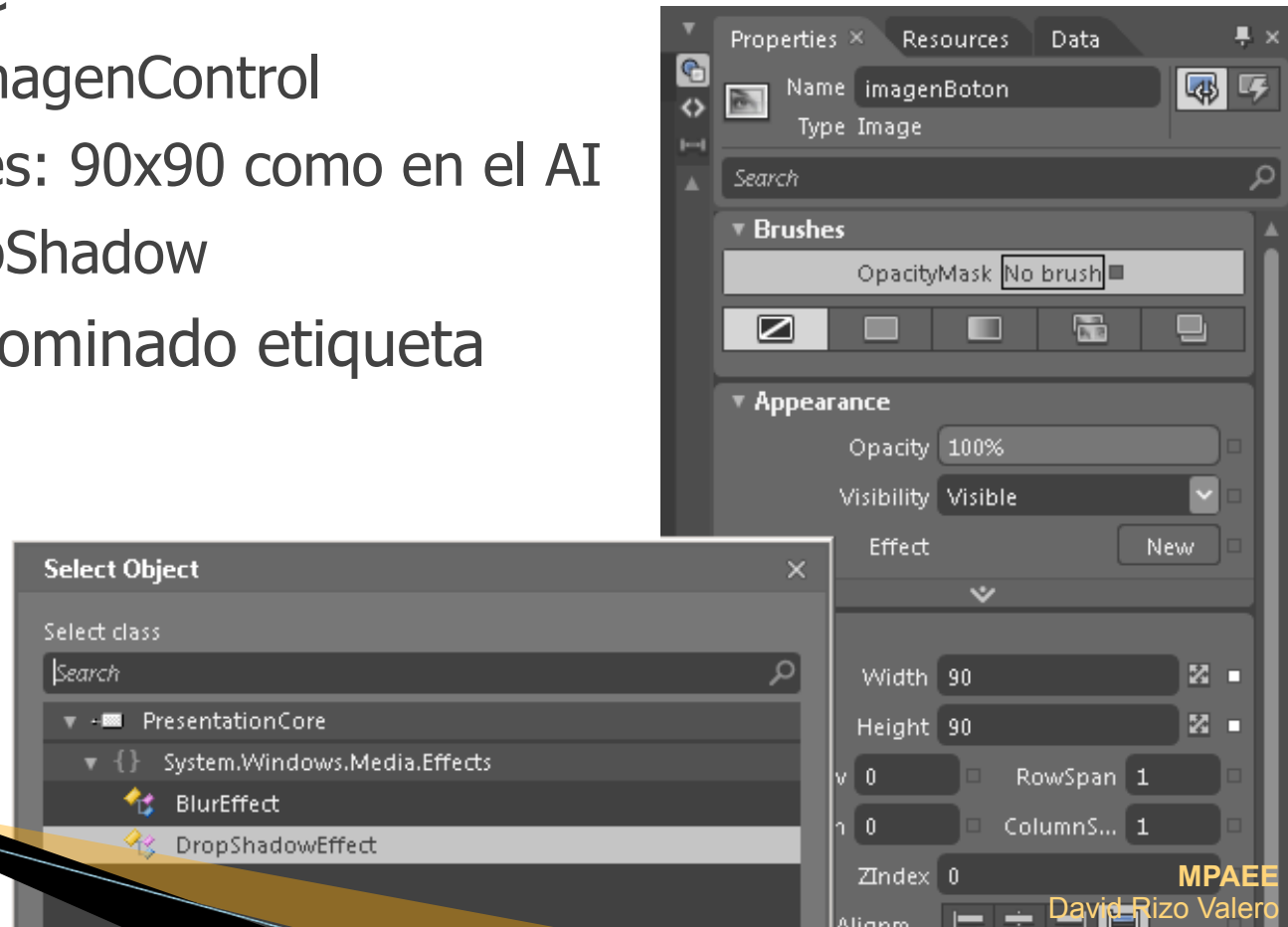
- “ Vamos a usar 3 botones, en la especificación nos dicen que las categorías no van a ser más
 - en BBDD tenemos una tabla, nos prepara para ampliarlas
 - no es necesario que la vista sea un reflejo de la estructura interna
- “ Crearemos un nuevo control personalizado
 - bien porque no existe un control como el que queremos
 - o simplemente para mantener la consistencia



Categorías: nuevo control

Dibujamos una imagen con texto y línea en un StackPanel

- control Image
 - ✎ nombre: imagenControl
 - ✎ dimensiones: 90x90 como en el AI
 - ✎ efecto dropShadow
- TextBox denominado etiqueta
- línea verde

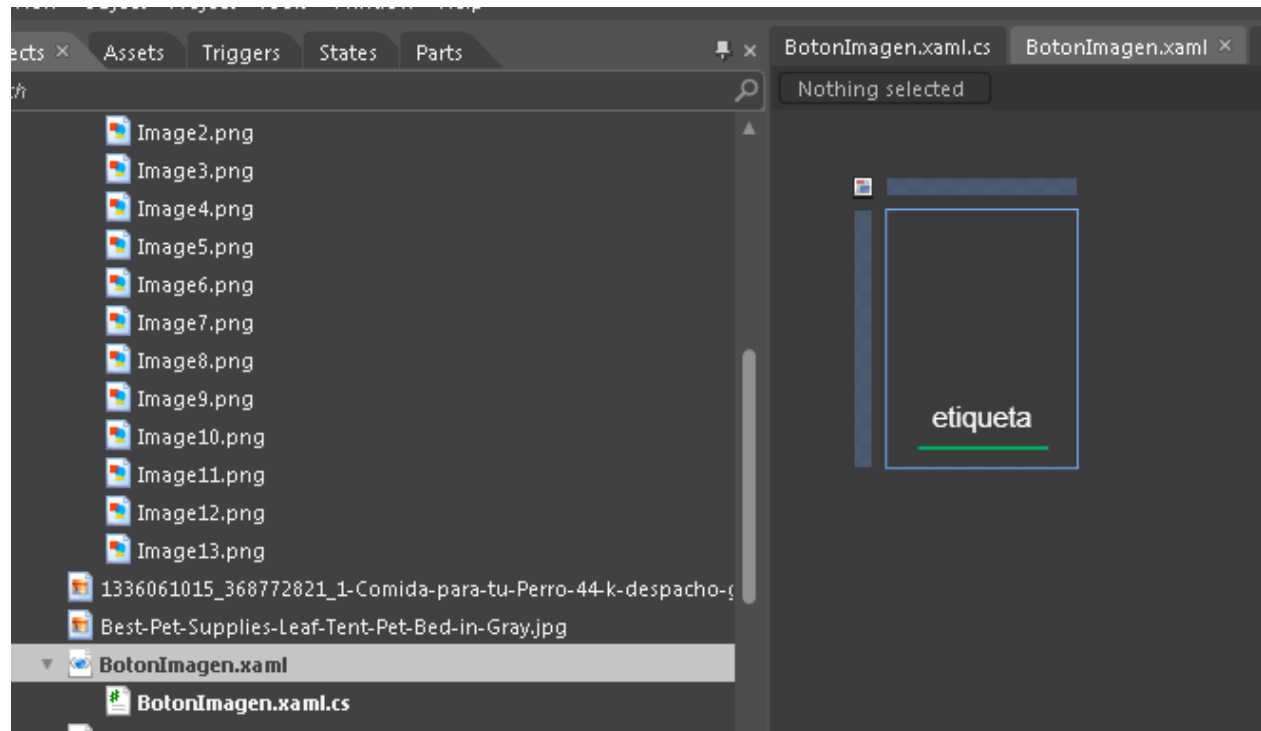


User control vs. custom control

- ▶ User control vs. custom control
 - user control: pequeña ventana, composición de otros controles
 - custom control: especialización de un control existente
- ▶ Los DataTemplates no permiten manipular de forma fácil la VisualStateManager, el UserControl (que dentro podría tener varios DataTemplates) sí

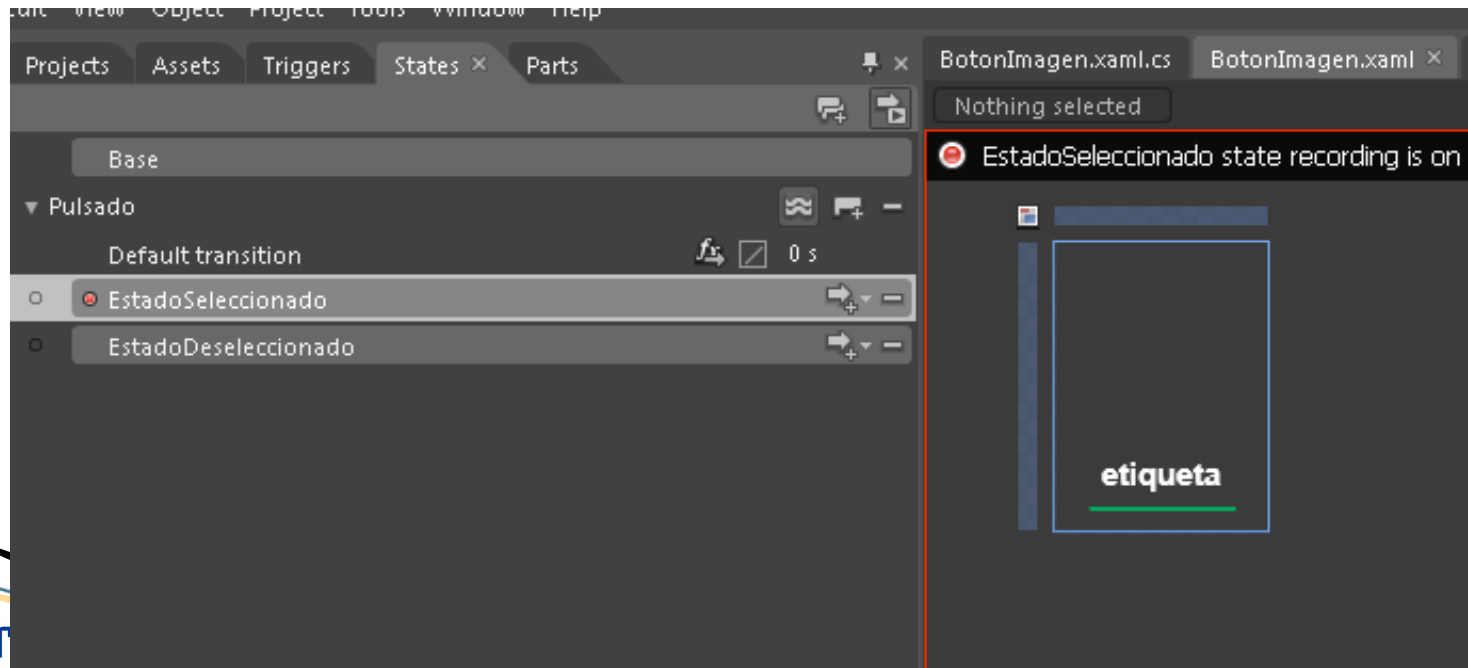
Categorías: nuevo control

- “ Botón derecho sobre StackPanel y pulsamos “Make into user control”
- “ Nos crea un
 - .xaml
 - .CS



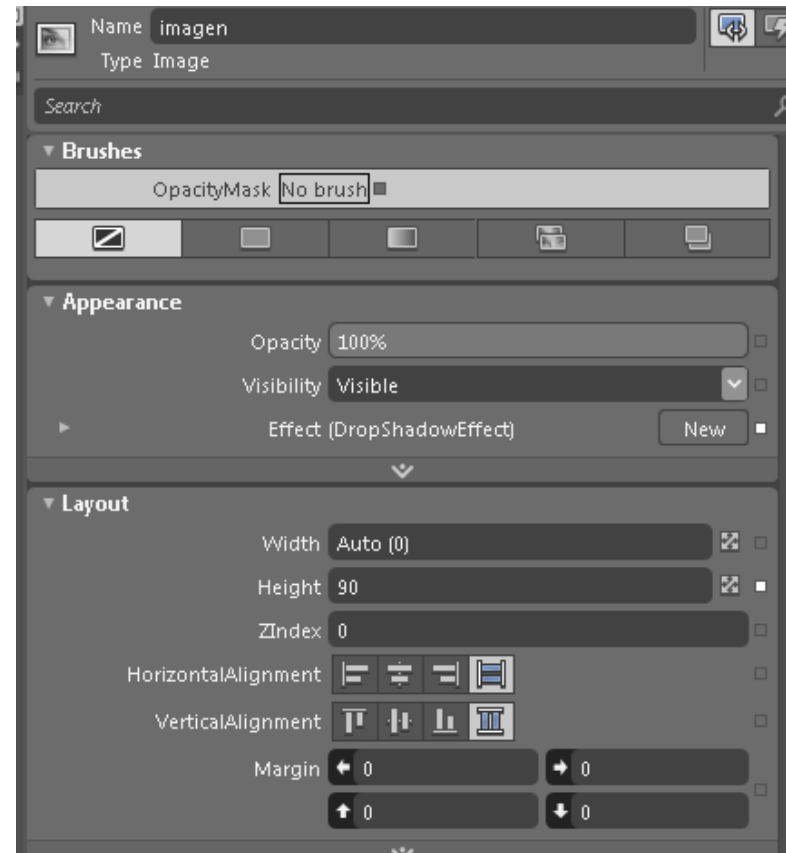
Categorías: nuevo control

- “ Cambiamos la apariencia según el estado
 - esto se hace de manera visual (comienza a grabar cuando pulsamos en un estado)



Categorías: nuevo control

- “ En EstadoSeleccionado ponemos la línea verde, el texto en negrita
- “ En EstadoDeseleccionado ocultamos la línea verde, el texto en normal y añadimos un efecto de sombra paralela a la imagen



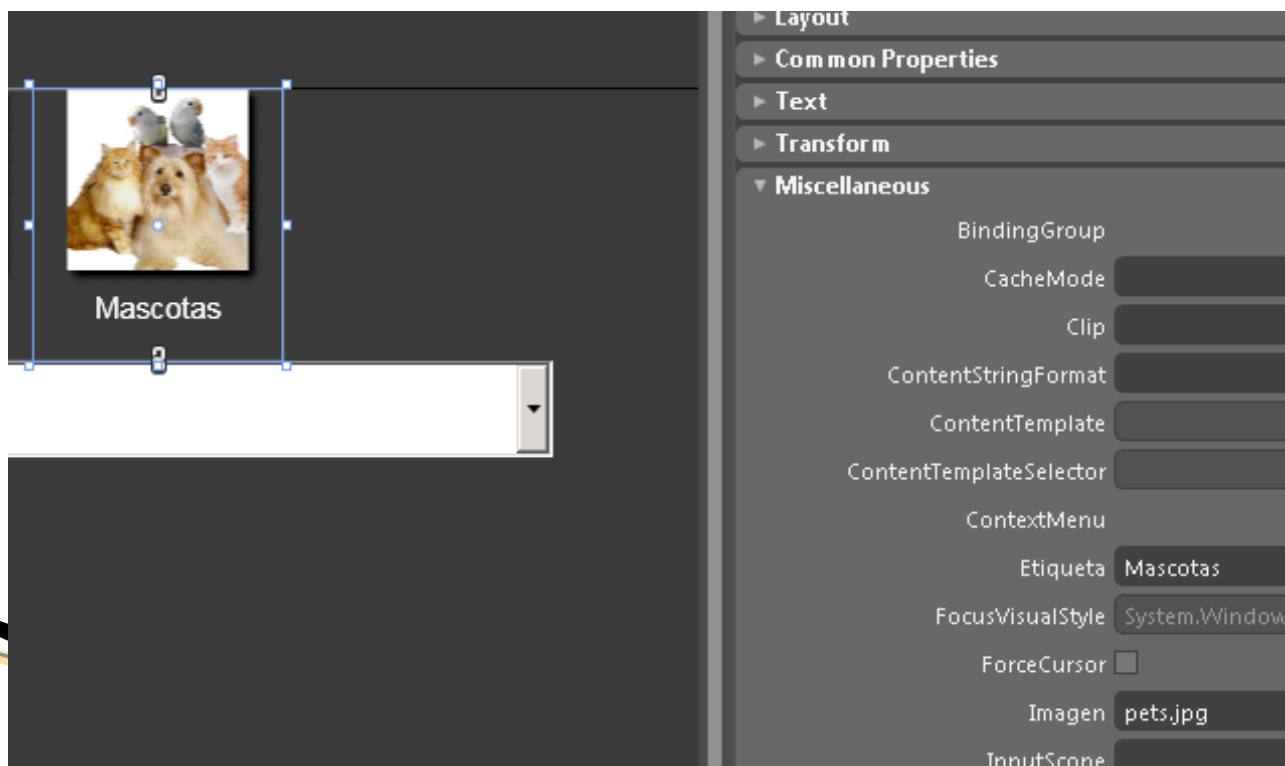
Categorías: nuevo control

- “ Añadimos las propiedades visibles al control usando propiedades
- sólo reencaminan los valores hacia los controles internos

```
public partial class BotonImagen : UserControl {  
    public BotonImagen() {  
        this.InitializeComponent();  
    }  
  
    public string Etiqueta  
    {  
        get { return etiqueta.Text; }  
        set { etiqueta.Text = value; }  
    }  
  
    public ImageSource Imagen  
    {  
        get { return imagen.Source; }  
        set { imagen.Source = value; }  
    }  
}
```

Categorías: nuevo control

“ En categorias.xaml podemos añadir 3 controles nuevos y cambiarles las propiedades de imagen y etiqueta



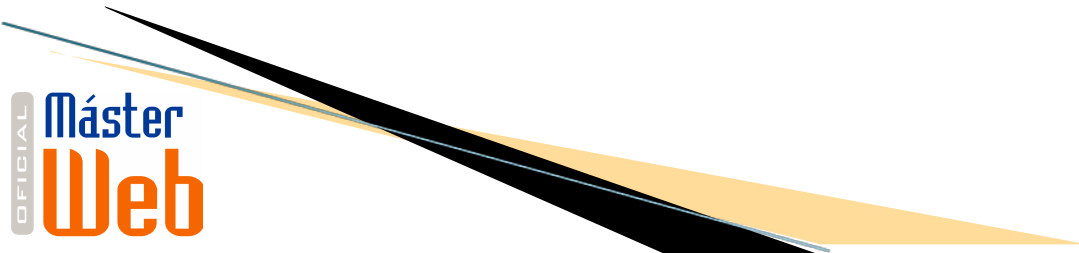
Categorías: nuevo control

“ Añadimos la propiedad Seleccionado que cambia la apariencia (cambiando el estado) e inicializamos

```
BotonImagen.xaml.cs x BotonImagen.xaml Categorías.xaml App.xaml.cs App.xaml MainWindow.xaml
20 public partial class BotonImagen : UserControl
21 {
22     Boolean seleccionado;
23
24     public BotonImagen()
25     {
26         this.InitializeComponent();
27         Seleccionado = false;
28     }
29
30     public Boolean Seleccionado
31     {
32         get { return seleccionado; }
33         set {
34             seleccionado = value;
35             if (seleccionado) {
36                 VisualStateManager.GoToState(this, "EstadoSeleccionado", true);
37             } else {
38                 VisualStateManager.GoToState(this, "EstadoDeseleccionado", true);
39             }
40         }
41     }
42 }
```

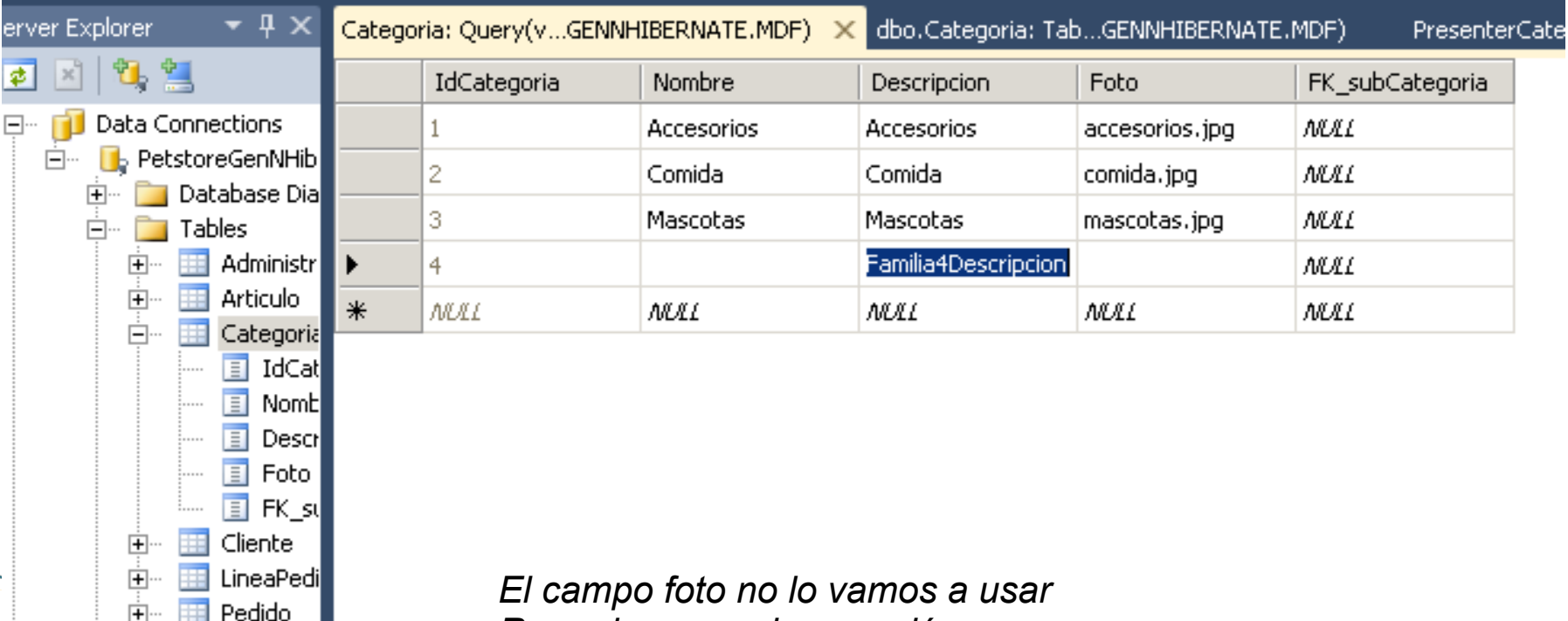
Categorías: nuevo control

“ Añadimos la propiedad ID que cambia tendrá el OID del objeto seleccionado



Datos de prueba

► En Visual Studio



Server Explorer

Data Connections

PetstoreGenNHib

Database Dia

Tables

Administr

Articulo

Categoria

IdCat

Nomb

Descr

Foto

FK_su

Cliente

LineaPedi

Pedido

Categoria: Query(v...GENNHIBERNATE.MDF) X

dbo.Categoria: Tab...GENNHIBERNATE.MDF) PresenterCate

	IdCategoria	Nombre	Descripcion	Foto	FK_subCategoria
	1	Accesorios	Accesorios	accesorios.jpg	NULL
	2	Comida	Comida	comida.jpg	NULL
	3	Mascotas	Mascotas	mascotas.jpg	NULL
►	4		Familia4Descripcion		NULL
*	NULL	NULL	NULL	NULL	NULL

*El campo foto no lo vamos a usar
Recordar cerrar la conexión*

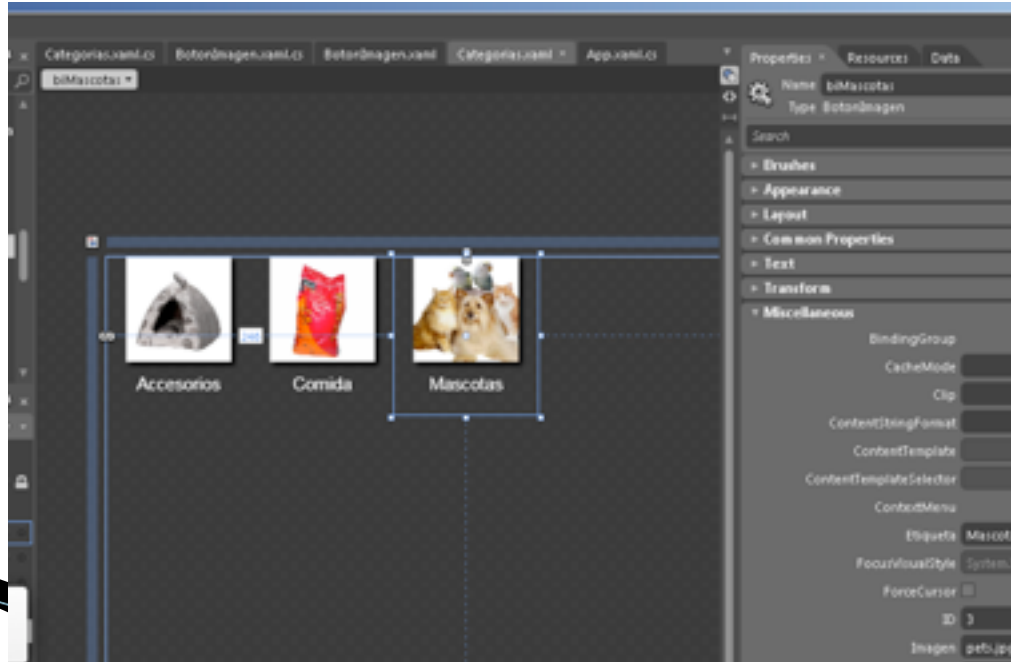
Enlazamos los BotonImagen

- ▶ Añadimos una nueva propiedad ID para que cuando se pulse se sepa qué categoría debemos consultar

```
public int ID
{
    get { return id; }
    set { id = value; }
}
```


Enlazamos los BotonImagen

- ▶ Damos nombre e ID a cada control BotonImagen en categorias.xaml (el ID lo podríamos obtener de BBDD, pero suponemos que es constante)



Enlazamos los BotonImagen

- Quitamos el ComboBox y añadimos un manejador de eventos para cada BotonImagen (ver código comentado)

```
public partial class Categorías : Page, IVistaCategorías
{
    private PresenterCategorías presenter = null;

    public Categorías()
    {
        InitializeComponent();
        //comboBox1.FontSize = 20;
        //comboBox1.DisplayMemberPath = "Nombre";
        //comboBox1.SelectedValue = "Id";
        presenter = new PresenterCategorías(this);
        //presenter.ObtenerProductos += new EventHandler(comboBox1_SelectionChanged);
        presenter.IniciarVista();
    }

    public IList<CategoriaEN> TodasCategorías
    {
        set{
            comboBox1.ItemsSource = value;
        }
    }

    /*private void comboBox1_SelectionChanged(object sender, EventArgs e)
    {
        int categoriaElegida = ((CategoriaEN)comboBox1.SelectedValue).Id;
        this.NavigationService.Navigate(new Productos(categoriaElegida));
    }*/
}
```

En EB

```
namespace PetstoreGen_MVP.code
{
    public class PresenterCategorías
    {
        //public event EventHandler ObtenerProductos;
        private IVistaCategorías vista;
        private Service servicio = null;

        public PresenterCategorías(IVistaCategorías vista)
        {
            this.vista = vista;
            servicio = new Service();
        }

        public void IniciarVista()
        {
            // lo dejamos para que las cargue en memoria
            IList<CategoriaEN> categorias = servicio.Gestor_DameTodasCategorías();
            //vista.TodasCategorías = categorias;

            // Ejemplo de redireccionamiento definido por el propio Presenter.
            //if (categorias.Count == 1) ObtenerProductos(this, new EventArgs());
        }
    }
}
```

En VS

Enlazamos los BotonImagen

- ▶ Añadimos un manejador de evento MouseDown a biAccesorios

```
private void biAccesorios_MouseDown(object sender, System.Windows.Input.MouseButtonEventArgs e)
{
    if (biAccesorios.Seleccionado)
    {
        int categoriaElegida = biAccesorios.ID;
        this.NavigationService.Navigate(new Productos(categoriaElegida));
    }
}
```

Enlazamos los BotonImagen

- ▶ Hacemos que manejador valga para los 3 botones y lo asignamos a los tres botones

```
private void biAccesorios_MouseDown(object sender, System.Windows.Input.MouseButtonEventArgs e)
{
    BotonImagen bi = (BotonImagen)sender;
    if (bi.Seleccionado)
    {
        int categoriaElegida = bi.ID;
        this.NavigationService.Navigate(new Productos(categoriaElegida));
    }
}
```

Estado actual

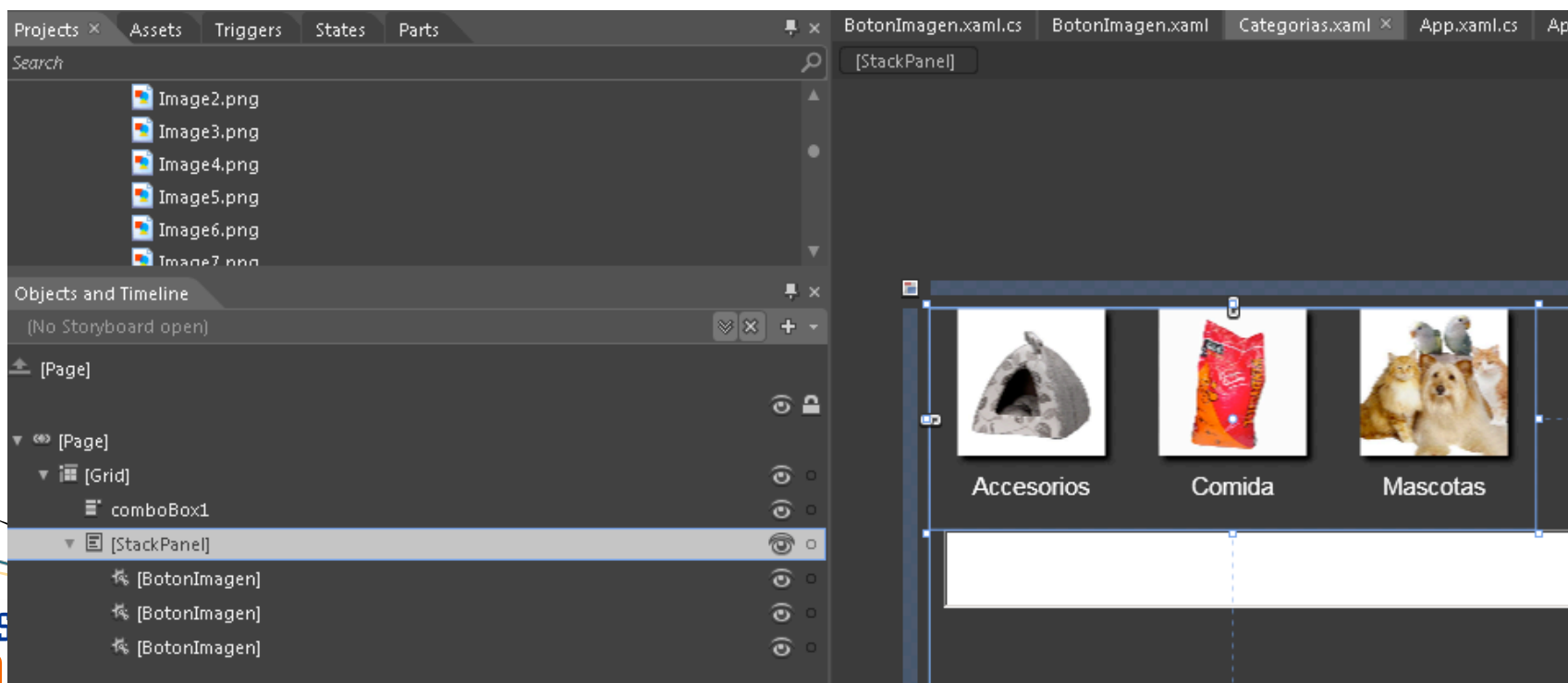


Estado actual



Categorías: nuevo control

- “ Hacemos que cuando se pulse un botón se suelten los demás
- Para ello necesitamos que estén dentro de un mismo panel



Categorías: nuevo control

“ Hacemos que cuando se pulse un botón se suelten los demás

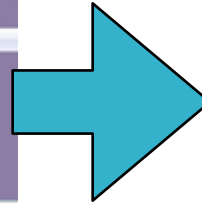
```
private void imagen_MouseDown(object sender, System.Windows.Input.MouseButtonEventArgs e)
{
    Seleccionado = !Seleccionado;

    if (Seleccionado) {
        if (Parent is Panel) {
            Panel p = (Panel) Parent;

            foreach(Control c in p.Children)
                if (c is BotonImagen && !c.Equals(this))
                    ((BotonImagen)c).Seleccionado = false;
        }
    }
}
```

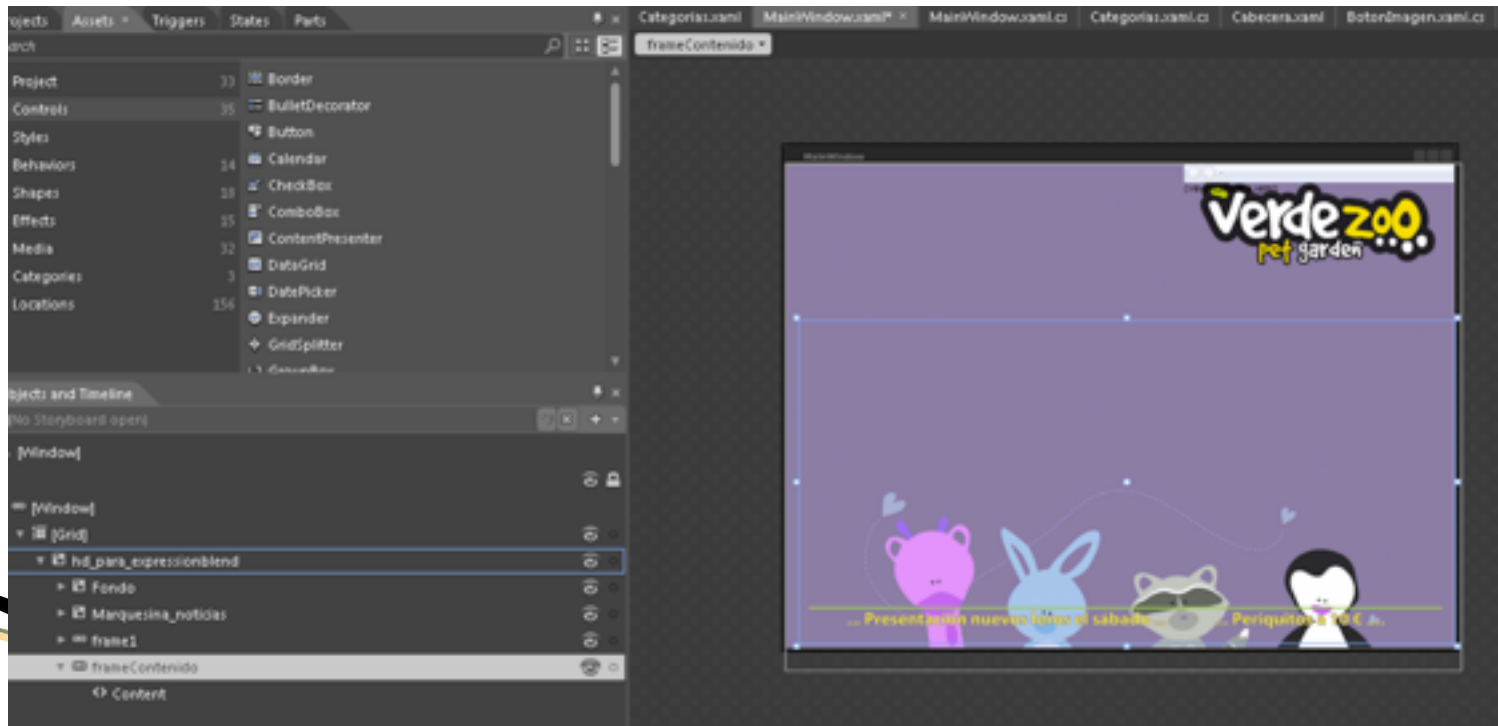

Productos

- ▶ Vamos a cambiar la organización de las vistas de forma que no se reemplacen las categorías



Productos

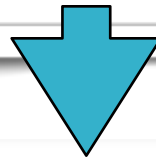
- ▶ Estado actual de frames, frameContenido tiene categorías y productos:



Productos

- Añadimos un frame sólo para las categorías y cambiamos su carga

```
public partial class MainWindow : Window
{
    public MainWindow()
    {
        InitializeComponent();
        frameContenido.NavigationService.Navigate(new Categorias());
    }
}
```

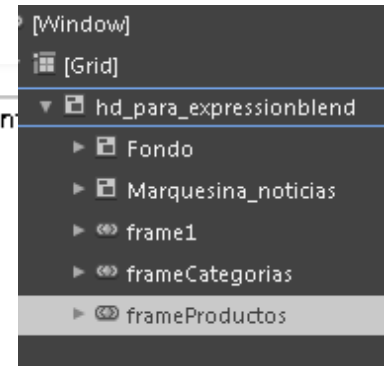


```
public partial class MainWindow : Window
{
    public MainWindow()
    {
        InitializeComponent();
        frameCategorias.NavigationService.Navigate(new Categorias());
    }
}
```

Productos

- ▶ Cuando se pulsa la categoría se deben cambiar los productos en otro frame

```
private void biAccesorios_MouseDown(object sender, System.Windows.Input.MouseButtonEventArgs e)
{
    BotonImagen bi = (BotonImagen)sender;
    if (bi.Seleccionado)
    {
        int categoriaElegida = bi.ID;
        this.NavigationService.Navigate(new Productos(categoriaElegida));
    }
}
```



```
public partial class MainWindow : Window
{
    public MainWindow()
    {
        InitializeComponent();
        frameProductos.NavigationUIVisibility = NavigationUIVisibility.Hidden;
        frameCategorias.NavigationService.Navigate(new Categorias(frameProductos));
    }
}
```

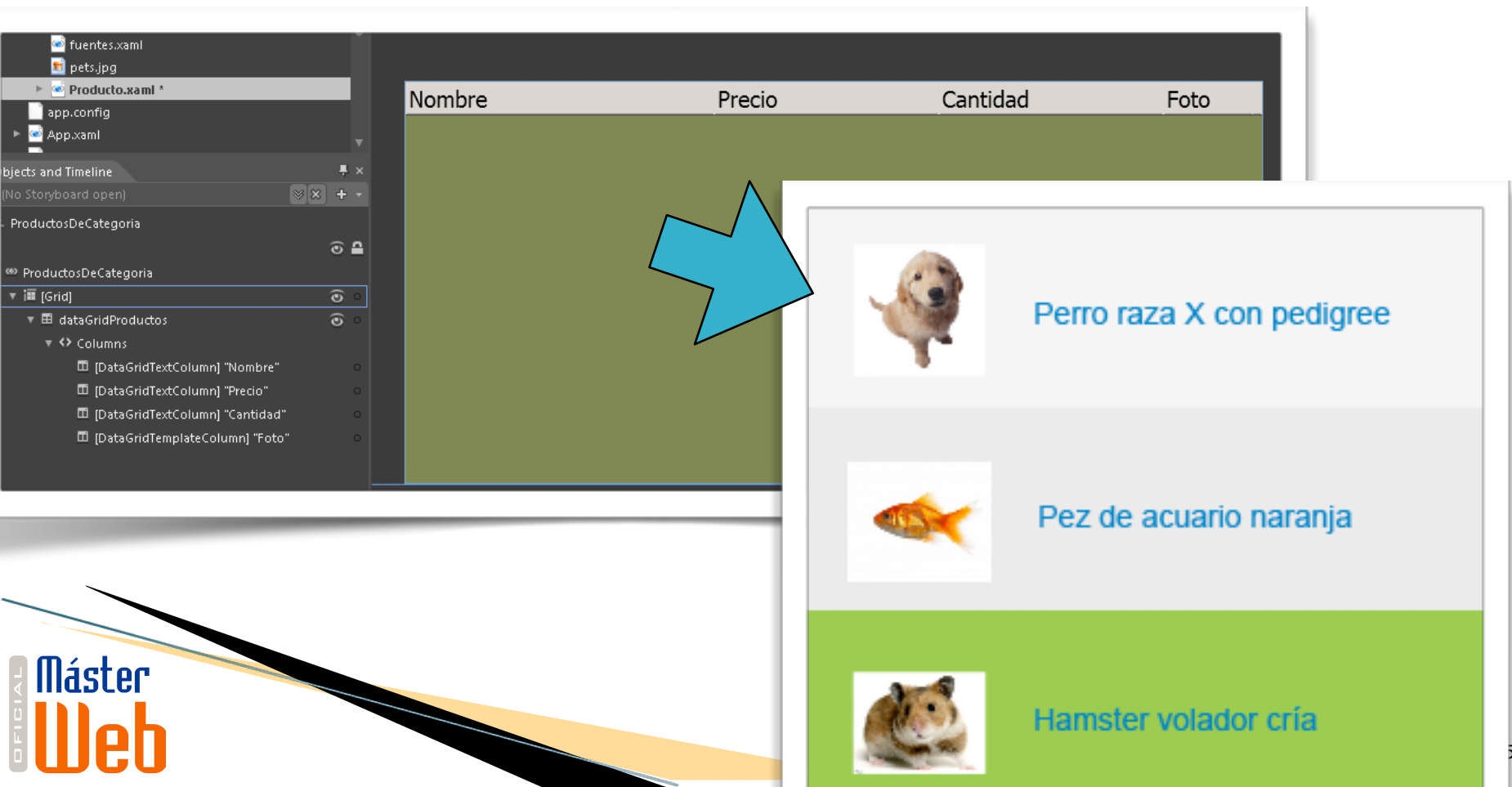
```
private void biAccesorios_MouseDown(object sender, System.Windows.Input.MouseButtonEventArgs e)
{
    BotonImagen bi = (BotonImagen)sender;
    if (bi.Seleccionado)
    {
        int categoriaElegida = bi.ID;
        productosPage.Navigate(new Productos(categoriaElegida));
    }
}
```

Productos

- ▶ Estado actual tras el cambio






Productos: estilo a la tabla/ dataGrid



The image shows a development environment with a file explorer on the left containing files like `fuentes.xml`, `pets.jpg`, `Producto.xml *`, `app.config`, and `App.xaml`. Below the file explorer is the 'Objects and Timeline' panel, which shows a tree structure for `ProductosDeCategoria` containing a `[Grid]` with a `dataGridProductos` control. The `Columns` property of `dataGridProductos` is expanded, showing four columns: `[DataGridTextColumn] "Nombre"`, `[DataGridTextColumn] "Precio"`, `[DataGridTextColumn] "Cantidad"`, and `[DataGridTemplateColumn] "Foto"`. A large blue arrow points from the `dataGridProductos` control to a visual representation of the data grid on the right.

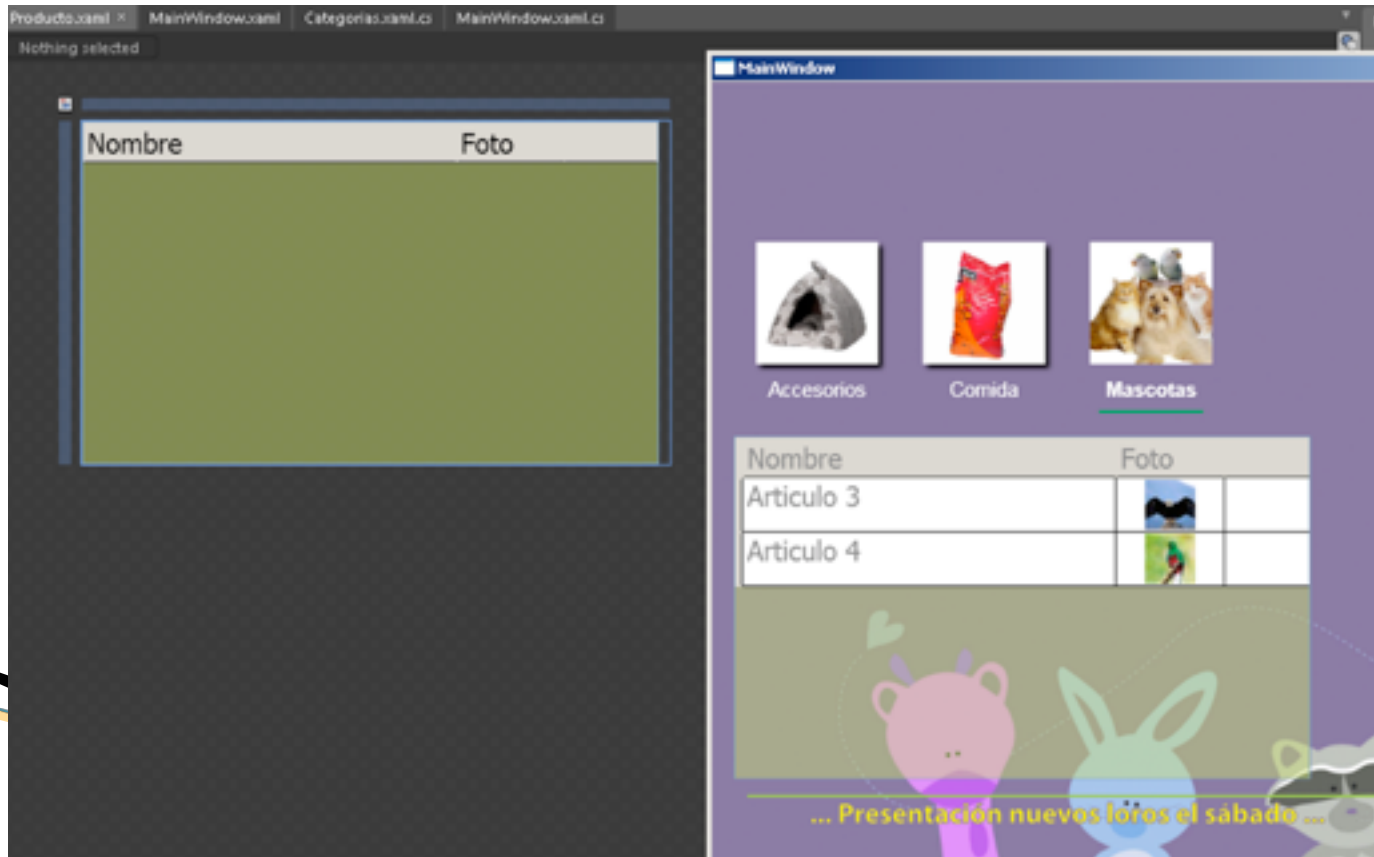
The visual representation of the data grid is a table with four columns: **Nombre**, **Precio**, **Cantidad**, and **Foto**. The table contains three rows of data:

Nombre	Precio	Cantidad	Foto
Perro raza X con pedigree			
Pez de acuario naranja			
Hamster volador cría			

At the bottom left, there is a logo for 'OFICIAL Máster Web'.

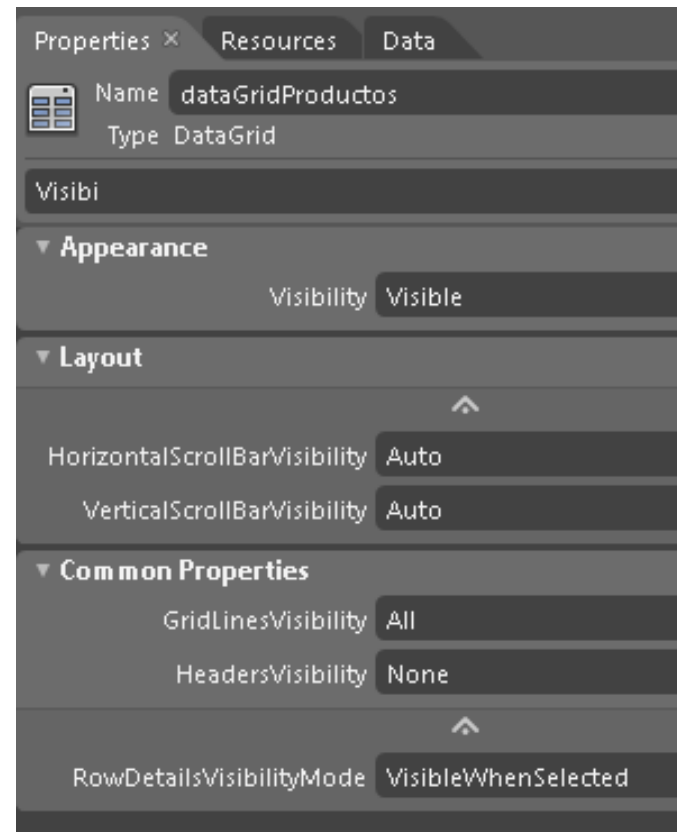
Productos: estilo a la tabla/ dataGrid

- ▶ Estado sólo cambiando tamaños



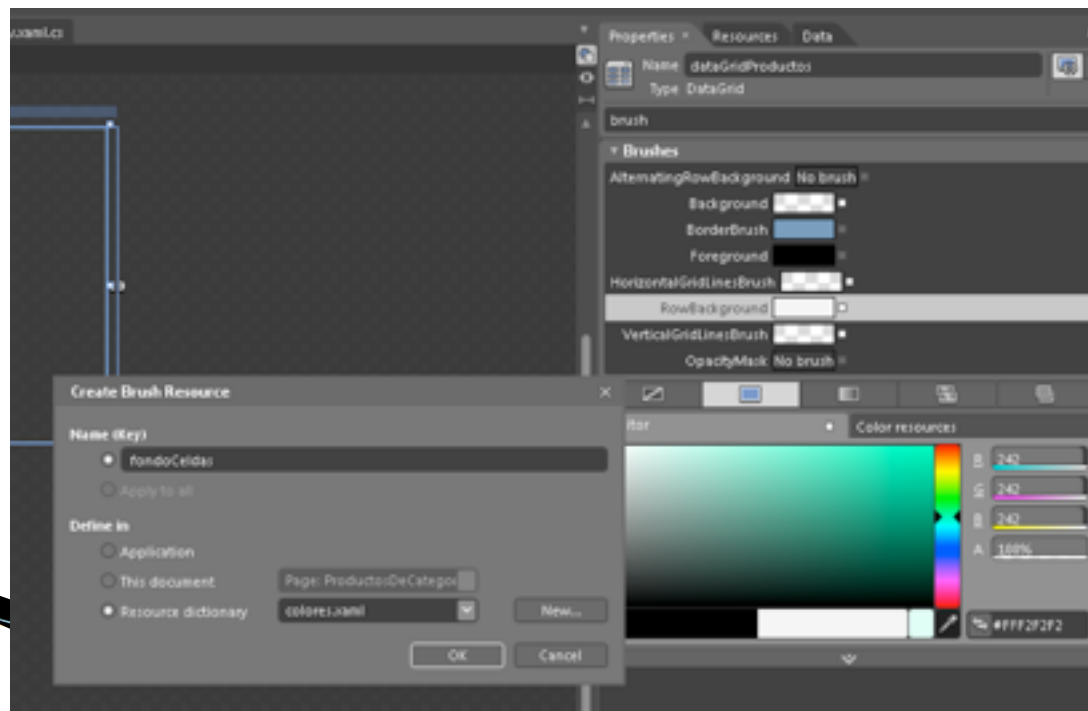
Productos: estilo a la tabla/ dataGrid

- ▶ Quitamos cabecera
- ▶ Cambiamos anchuras y orden columnas
- ▶ Quitamos scroll horizontal
- ▶ Color transparente a líneas (ó GridLinesVisibility)
- ▶ Colores (ver siguiente diapositiva)



Productos: estilo a la tabla/ dataGrid

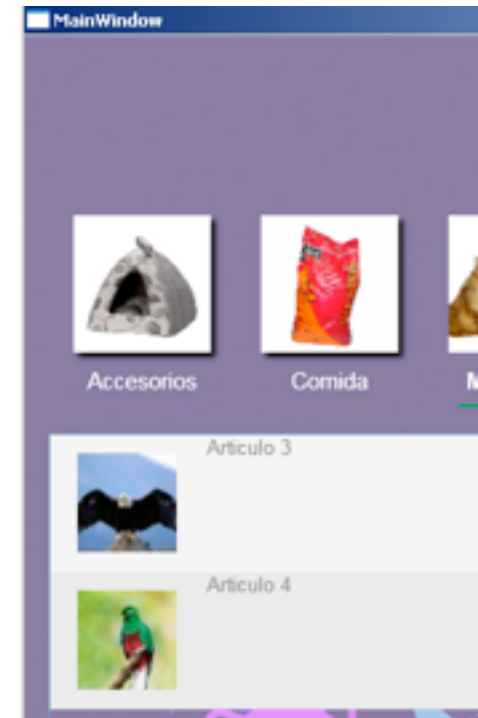
- Cambiamos los colores, pero los llevamos al recurso (convert to new resource)



Productos: estilo a la tabla/ dataGrid

- ▶ Para el tamaño de las imágenes usamos directamente el intellisense del xaml y añadimos Height a Image

```
<Grid>
    <DataGrid AutoGenerateColumns="False" IsReadOnly="True" Height="253"
        <DataGrid.Columns>
            <DataGridTemplateColumn Header="Foto" Width="100" IsReadOnly
                <DataGridTemplateColumn.CellTemplate>
                    <DataTemplate>
                        <Image Source="{Binding Foto}" Height="65" />
                    </DataTemplate>
                </DataGridTemplateColumn.CellTemplate>
            </DataGridTemplateColumn>
            <DataGridTextColumn Header="Nombre" Width="325" Binding="{Bi
        </DataGrid.Columns>
    </DataGrid>
```



Productos: estilo a la tabla/ dataGrid

- Para centrar el texto de las celdas

```
<Grid>
  <DataGrid AutoGenerateColumns="False" IsReadOnly="True" Height="253" Horizontal
    <DataGrid.Columns>
      <DataGridTemplateColumn Header="Foto" Width="100" IsReadOnly="True" Ca
        <DataGridTemplateColumn.CellTemplate>
          <DataTemplate>
            <Image Source="{Binding Foto}" Height="65" />
          </DataTemplate>
        </DataGridTemplateColumn.CellTemplate>
      </DataGridTemplateColumn>
      <DataGridTextColumn Header="Nombre" Width="325" Binding="{Binding Nomb
        <DataGridTextColumn.ElementStyle>
          <Style TargetType="TextBlock">
            <Setter Property="VerticalAlignment" Value="Center" />
          </Style>
        </DataGridTextColumn.ElementStyle>
      </DataGridTextColumn>
    </DataGrid.Columns>
  </DataGrid>
</Grid>
```

Productos: estilo a la tabla/ dataGrid

► Estado final



Ejercicio

- ▶ Añadir la visualización del producto seleccionado conforme se muestra en el diseño



WPF Routed Commands

Cliente

```
public interface ICommandSource
{
    ICommand Command { get; }
    object CommandParameter { get; }
    IInputElement CommandTarget { get; }
}
```

Es notificado de cambios en canExecute
CommandTarget: enlace con el control (foco actual si es null)

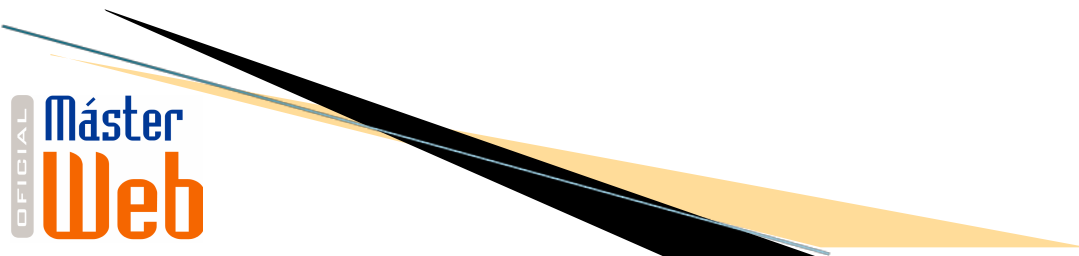
Comando

```
public interface ICommand
{
    event EventHandler CanExecuteChanged;
    bool CanExecute(object parameter);
    void Execute(object parameter);
}
```

ButtonBase, MenuItem, Hyperlink e InputBinding implementan ICommandSource

WPF Routed Commands

- ▶ Creamos un comando. En lugar de realizar la interfaz ICommand usamos RoutedUICommand
- ▶



```

namespace PetstoreGen_PageController
{
    public static class MisComandos
    {
        private static RoutedUICommand comandoEjemplo =
            new RoutedUICommand("ComandoEjemplo", "ComandoEjemplo", typeof(MisComandos));

        public static RoutedCommand ComandoEjemplo
        {
            get { return comandoEjemplo; }
        }

        static MisComandos()
        {
            CommandBinding comandoEjemploBinding = new CommandBinding(comandoEjemplo);
            CommandManager.RegisterClassCommandBinding(typeof(MisComandos), comandoEjemploBinding);
        }
    }

    /// <summary>
    /// Interaction logic for EjemploCommands.xaml
    /// </summary>
    public partial class EjemploCommands : Window

```



```

        CommandManager.RegisterClassCommandBinding(typeof(MisComandos), comandoEjemploBinding)
    }
}

/// <summary>
/// Interaction logic for EjemploCommands.xaml
/// </summary>
public partial class EjemploCommands : Window
{
    public EjemploCommands()
    {
        this.InitializeComponent();

        // Insert code required on object creation below this point.
    }

    private void ComandoEjemploPuedoEjecutar(object sender, CanExecuteRoutedEventArgs e)
    {
        e.CanExecute = true;
        e.Handled = true;
    }

    private void ComandoEjemploEjecutado(object sender, ExecutedRoutedEventArgs e)
    {
        MessageBox.Show("Comando de ejemplo ejecutado");
        e.Handled = true;
    }
}
}

```

```
ploCommands.xaml.cs  EjemploCommands.xaml  Window1.xaml  BotonImagen.xaml.cs  Cabecera.xaml
1  <Window
2      xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
3      xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
4      x:Class="PetstoreGen_PageController.EjemploCommands"
5      xmlns:comandos="clr-namespace:PetstoreGen_PageController"
6
7      x:Name="Window"
8      Title="EjemploCommands"
9      Width="640" Height="480">
10
11      <Window.CommandBindings>
12          <CommandBinding Command="{x:Static comandos:MisComandos.ComandoEjemplo}"
13              CanExecute="ComandoEjemploPuedoEjecutar"
14              Executed="ComandoEjemploEjecutado" />
15      </Window.CommandBindings>
16
17      <Grid x:Name="LayoutRoot">
18          <Button x:Name="botonEjemploHola" Content="Lanza comando ejemplo con parámetro &#xd;&#x"
19              Command="{x:Static comandos:MisComandos.ComandoEjemplo}" />
20          <Button x:Name="botonEjemploAdios" Content="Lanza comando ejemplo con parámetro &#xa;&q"
21      </Grid>
22  </Window>
```