UNIVERSIDAD TECNOLÓGICA DE IZÚCAR DE MATAMOROS

MVVM Pattern

Model View View Model

Programación de Aplicaciones

A. Felipe Lima

2016

Explicación teórica

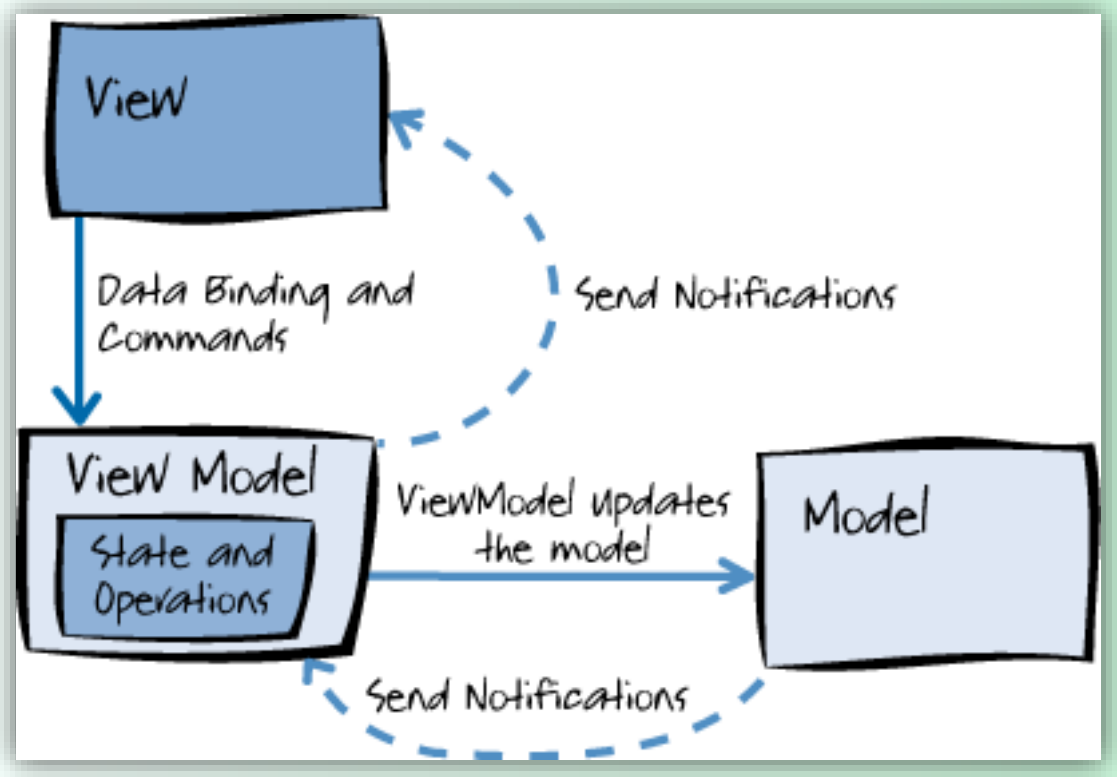
Refiérase al articulo The MVVM Pattern

en https://msdn.microsoft.com/en-us/library/hh848246.aspx

- A continuación extractos del contenido…

The MVVM pattern lends itself naturally to XAML application platforms such as Silverlight. This is because the MVVM pattern leverages some of the specific capabilities of Silverlight, such as data binding, commands, and behaviors. MVVM is similar to many other patterns that separate the responsibility for the appearance and layout of the UI from the responsibility for the presentation logic; for example, if you're familiar with the Model-View-Controller (MVC) pattern, you'll find that MVVM has many similar concepts. (Microsoft. 2012)

The MVVM Pattern



- Intercambio de componentes
- Implementación por separado
- Componentes con trabajo independiente

Model

The model in MVVM is an implementation of the application's domain model that includes a data model along with business and validation logic. Examples of model objects include repositories, business objects, data transfer objects (DTOs), Plain Old CLR Objects (POCOs), and generated entity and proxy objects. (Microsoft. 2012)

View

The view is responsible for defining the structure, layout, and appearance of what the user sees on the screen. Ideally, the view is defined purely with **XAML**, with a limited code-behind that does not contain business logic. (Microsoft. 2012)

View

A view can have its own view model, or it can inherit its parent's view model. **A view gets data from its view model through bindings**, or invoking methods on the view model. At run time, the view changes when UI controls respond to view model properties **raising change notification events**. (Microsft. 2012)

View Model

The view model acts as an intermediary between the view and the model, and is responsible for handling the view logic. Typically, the view model interacts with the model by invoking methods in the model classes. The view model then provides data from the model in a form that the view can easily use. (Microsoft. 2012)

Connecting View Models to Views

MVVM leverages the data-binding capabilities to manage the link between the view and view model, along with behaviors and event triggers. These capabilities limit the need to place business logic in the view's code-behind. (Microsoft. 2012)

Connecting View Models to Views

There are many approaches to connecting a view model to a view, including direct relations and container-based approaches. However, all share the same aim, which is for the view to have a view model assigned to its DataContext property. (Microsoft. 2012)

UNIVERSIDAD TECNOLÓGICA DE IZÚCAR DE MATAMOROS

MVVM

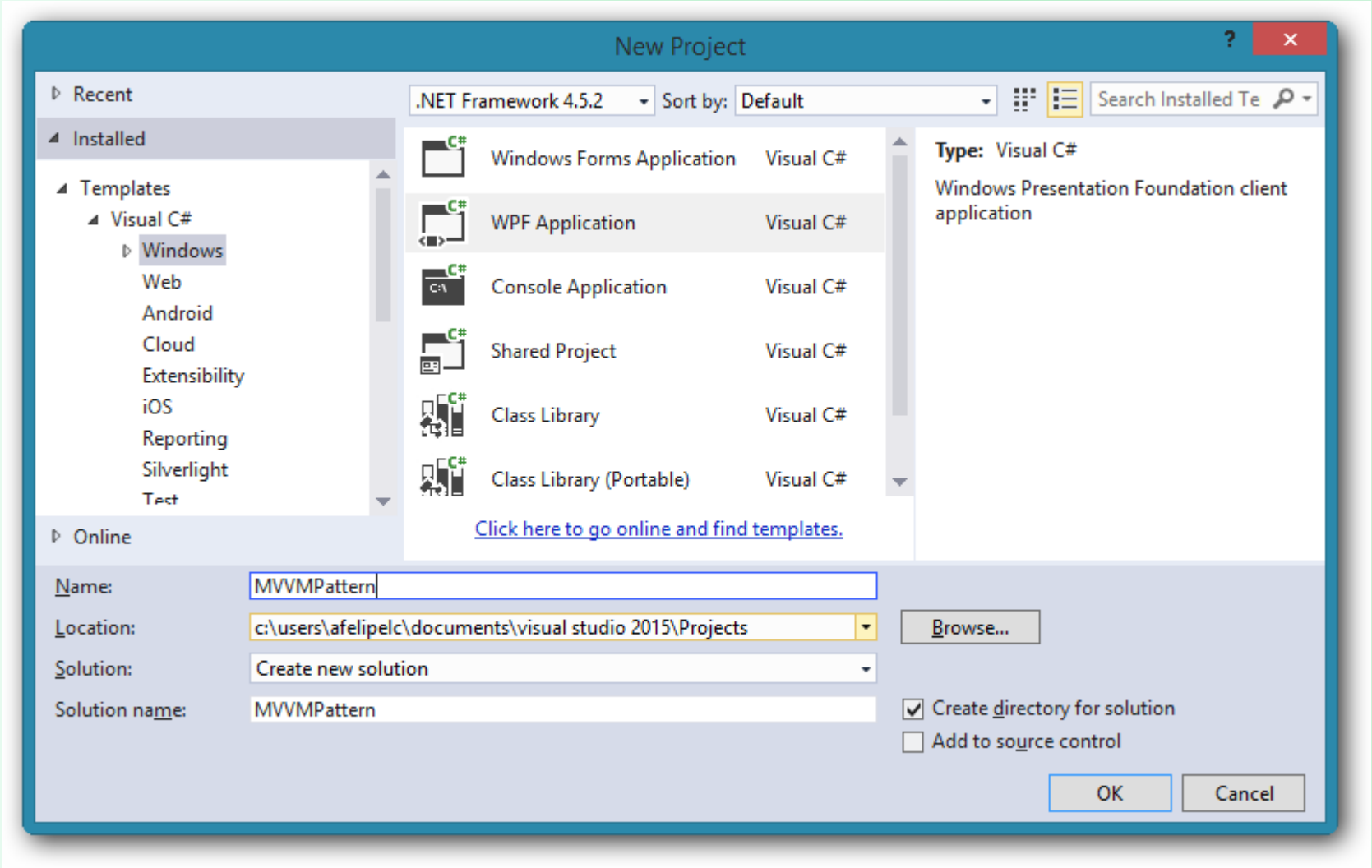
Ejemplo

2016

Palabras clave

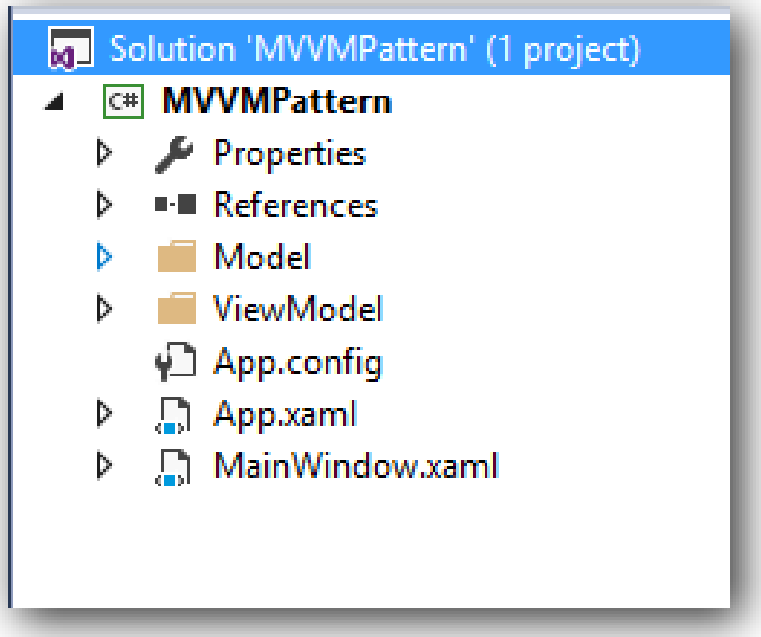
- WPF
- Clases
- Herencia
- Interfaces
- Model
- View Model
- INotifyPropertyChanged
- RelayCommand
- DataContext
- Xaml

Crear proyecto WPF



2016

Estructura del proyecto



- Separar los espacios de nombre por medio de carpetas
- Ó
- Crear proyectos de bibliotecas de clase por separado.

Model

Clase Persona

```
namespace MVVMPattern.Model
{
    public class Persona
    {
        public string Nombre { get; set; }
        public string Apellidos { get; set; }
        public int Edad { get; set; }
    }
}
```

UNIVERSIDAD TECNOLÓGICA DE IZÚCAR DE MATAMOROS

View Model

2016

Commands o RelayCommand

Los comandos son empleados para separar controlar la ejecución de eventos sobre los objetos, por ejemplo:

- Al hacer click sobre un botón
- Al presionar una tecla sobre el control
- Entre otros eventos

Son una implementación de la interface **ICommand**

Ver: https://msdn.microsoft.com/en-us/magazine/dn237302.aspx

class RelayCommand

```
namespace MVVMPattern.ViewModel
{
    /// <summary>
    /// Clase que controla la ejecución de eventos en el
ViewModel
    /// </summary>
    public class RelayCommand : ICommand
    {
        public RelayCommand(Action<object> execute)
            : this(execute, null)
        {
        }
        public RelayCommand(Action<object> execute,
Predicate<object> canExecute)
        {
            if (execute == null)
                throw new ArgumentNullException("execute");
            _execute = execute;
            _canExecute = canExecute;
        }
        public bool CanExecute(object parameter)
        {
            return  canExecute == null ? true :
_canExecute(parameter);
        }
```

```
public event EventHandler CanExecuteChanged
        {
            add {
CommandManager.RequerySuggested += value; }
            remove {
CommandManager.RequerySuggested -= value; }
        }
        public void Execute(object parameter)
        {
            _execute(parameter);
        }
        private readonly Action<object>
_execute;
        private readonly Predicate<object>
_canExecute;
    }
}
```

2016

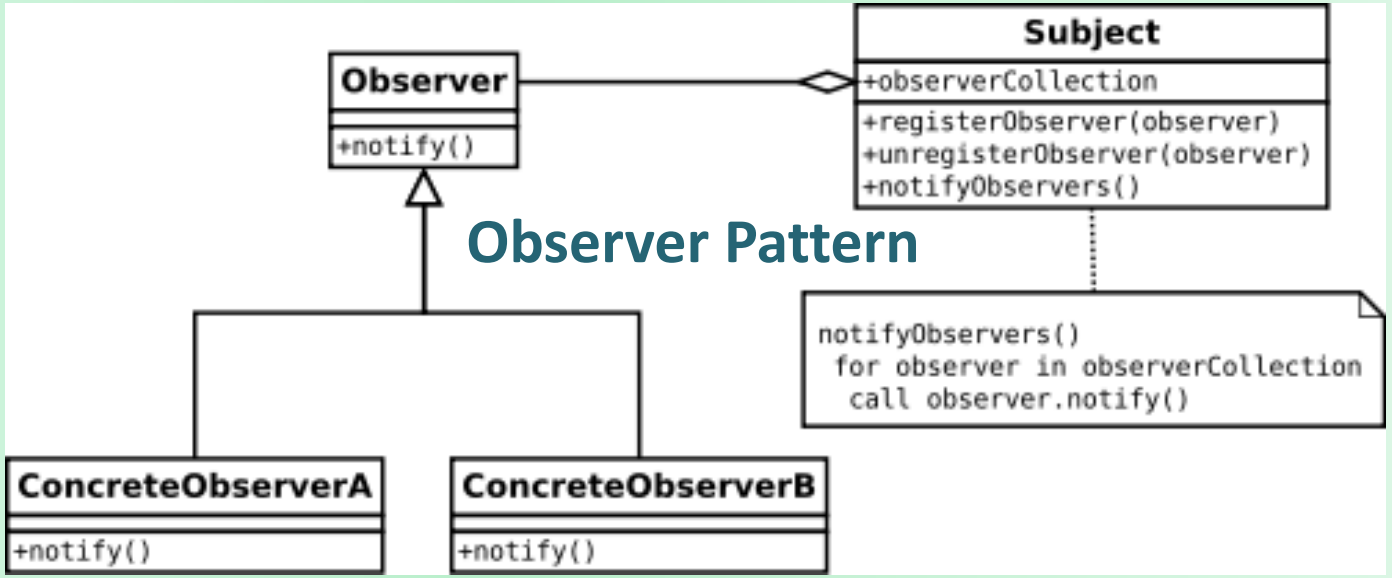
NotifyPropertyChanged en clases modelo

El view model es el responsable de administrar los cambios que ocurren tanto en la vista, como en el modelo o propiedades del view model.

Para poder reaccionar ante tales cambios, es necesario realizar un implementación de **INotifyPropertyChanged**

ViewModelBase

- Será la base de todo view model en la aplicación, será una implementación de **INotifyPropertyChanged**



ViewModelBase

```
namespace MVVMPattern.ViewModel
{
    /// <summary>
    /// Clase que observa los cambios ocurridos en las propiedades del ViewModel
    /// </summary>
    public class ViewModelBase : INotifyPropertyChanged
    {
        public event PropertyChangedEventHandler PropertyChanged;
        protected void OnPropertyChanged(string propertyName)
        {
            if (PropertyChanged != null)
            {
                PropertyChanged(this, new PropertyChangedEventArgs(propertyName));
            }
        }
    }
}
```

ViewModel clases

Los View Model, heredan la clase **ViewModelBase**, por lo tanto, pueden usar el método :

OnPropertyChanged(“PropiedadViewModel”);

OnPropertyChanged()

Al ocurrir un cambio en la propiedad indicada como parámetro de OnPropertyChanged(“propiedad”), la vista será actualizada automáticamente.

PersonasViewModel

```
namespace MVVMPattern.ViewModel
{
    public class PersonasViewModel : ViewModelBase
    {
        ObservableCollection<Persona> personas;
        Persona personaActual;
        RelayCommand nuevoCommand, guardarCommand;

        public PersonasViewModel()
        {
            this.Personas = new ObservableCollection<Persona>();
            this.nuevoCommand = new RelayCommand(p => this.Nuevo(), p => true);
            this.guardarCommand = new RelayCommand(p => this.Guardar(), p =>
this.sePuedeGuardar());
            Nuevo();
        }
```

PersonasViewModel

```
#region Propiedades
        public ObservableCollection<Persona> Personas
        {
            get
            {
                return personas;
            }
            set
            {
                personas = value;
                OnPropertyChanged("Personas");
            }
        }
```

PersonasViewModel

```
public Persona PersonaActual
        {
            get
            {
                return personaActual;
            }
            set
            {
                personaActual = value;

OnPropertyChanged("PersonaActual");
            }
        }
```

```
 public RelayCommand
NuevoCommand
        {
            get
            {
                return
nuevoCommand;
            }
        }
```

PersonasViewModel

```
public RelayCommand GuardarCommand
        {
            get
            {
                return guardarCommand;
            }
        }
        #endregion
```

PersonasViewModel

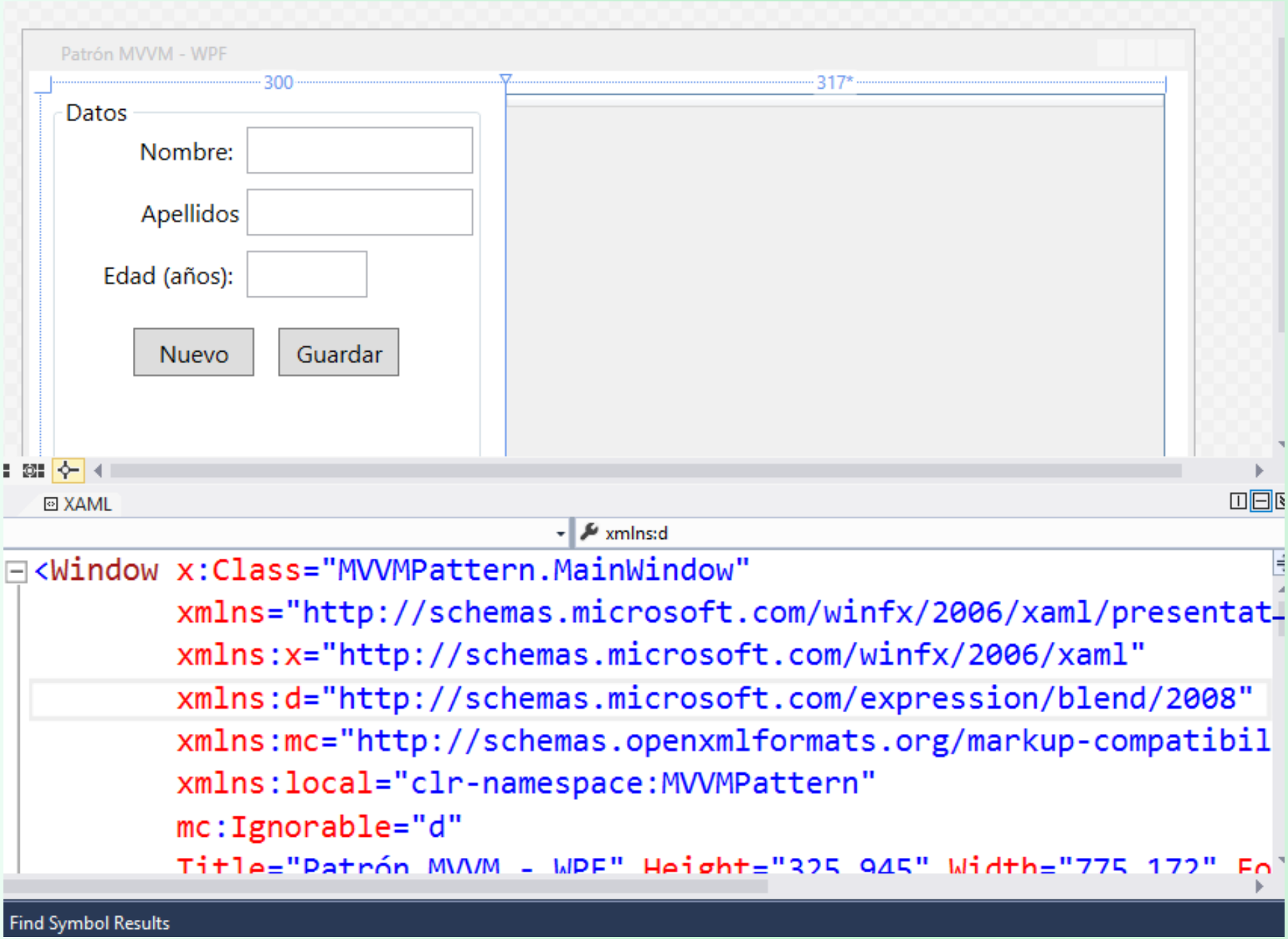
```
#region Métodos
        private void Guardar()
        {
            if (!personas.Contains(personaActual))
                Personas.Add(personaActual);
        }

        private bool sePuedeGuardar()
        {
            return personaActual != null &&
                !string.IsNullOrEmpty(personaActual.Nombre) &&
                !string.IsNullOrEmpty(personaActual.Apellidos);
        }
```

PersonasViewModel

```
 private void Nuevo()
            {
                this.PersonaActual = new Persona();
            }
            #endregion
        }
}
```

UNIVERSIDAD TECNOLÓGICA DE IZÚCAR DE MATAMOROS

View


Patrón MVVM - WPF
300
317*
Datos
Nombre:
Apellidos
Edad (años):
Nuevo
Guardar
XAML
xmlns:d
<Window x:Class="MVVMPattern.MainWindow"
xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentat
xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
xmlns:mc="http://schemas.openxmlformats.org/markup-compatibil
xmlns:local="clr-namespace:MVVMPattern"
mc:Ignorable="d"
Find Symbol Results

Escrita en **Xaml**

La vista hace uso del ViewModel para mostrar los datos al usuario, sus componentes están:

- Enlazados a las propiedades
- Enlazados a los command

Tutorial de Xaml: http://www.wpf-tutorial.com/xaml/what-is-xaml/

View

```
<Window x:Class="MVVMPattern.MainWindow"
        …
        Title="Patrón MVVM - WPF" Height="325.945" Width="775.172" FontSize="16">
    <Grid Margin="15">
        <Grid.ColumnDefinitions>
            <ColumnDefinition Width="300"/>
            <ColumnDefinition Width="317*"/>
        </Grid.ColumnDefinitions>

        <GroupBox Header="Datos" Margin="0,0,15,0">
            <StackPanel>
                <StackPanel Orientation="Horizontal" HorizontalAlignment="Right">
                    <Label Content="Nombre: " />
                    <TextBox Width="150" Text="{Binding PersonaActual.Nombre, Mode=TwoWay,
UpdateSourceTrigger=Default}" x:Name="nombreTxt"/>
                </StackPanel>
                <StackPanel Orientation="Horizontal" HorizontalAlignment="Right" Margin="0,10,0,0">
                    <Label Content="Apellidos" />
                    <TextBox Width="150" Text="{Binding PersonaActual.Apellidos, Mode=TwoWay,
UpdateSourceTrigger=Default}" />
                </StackPanel>
```

View

```
<StackPanel Orientation="Horizontal" HorizontalAlignment="Right"
Margin="0,10,0,0">
                    <Label Content="Edad (años): " />
                    <TextBox Width="80" Text="{Binding PersonaActual.Edad,
Mode=TwoWay, UpdateSourceTrigger=Default, StringFormat=\{0:0\}}"
Margin="0,0,70,0"/>
                </StackPanel>
                <StackPanel Orientation="Horizontal" Margin="0,20"
HorizontalAlignment="Center">
                    <Button Height="32" Width="80" Margin="8,0" Content="Nuevo"
Command="{Binding NuevoCommand}" x:Name="nuevoBtn" Click="nuevoBtn_Click"/>
                    <Button Height="32" Width="80" Margin="8,0"
Content="Guardar" Command="{Binding GuardarCommand}"/>
                </StackPanel>
            </StackPanel>
        </GroupBox>
```

View

```
<DataGrid Grid.Column="1" HorizontalAlignment="Stretch"
                    VerticalAlignment="Top" MaxHeight="300"
Height="250"
                    ItemsSource="{Binding Personas}"
                    SelectedItem="{Binding PersonaActual}"
      SelectionMode="Single" CanUserAddRows="False"
IsReadOnly="True" CanUserDeleteRows="False"/>
    </Grid>
</Window>
```

View behind-code

```
public partial class MainWindow : Window
    {
        PersonasViewModel viewContext = new PersonasViewModel();
        public MainWindow()
        {
            this.DataContext = viewContext;
            InitializeComponent();
            this.nombreTxt.Focus();
        }

        private void nuevoBtn_Click(object sender, RoutedEventArgs e)
        {
            this.nombreTxt.Focus();
        }
    }
```

Ejercicio 1

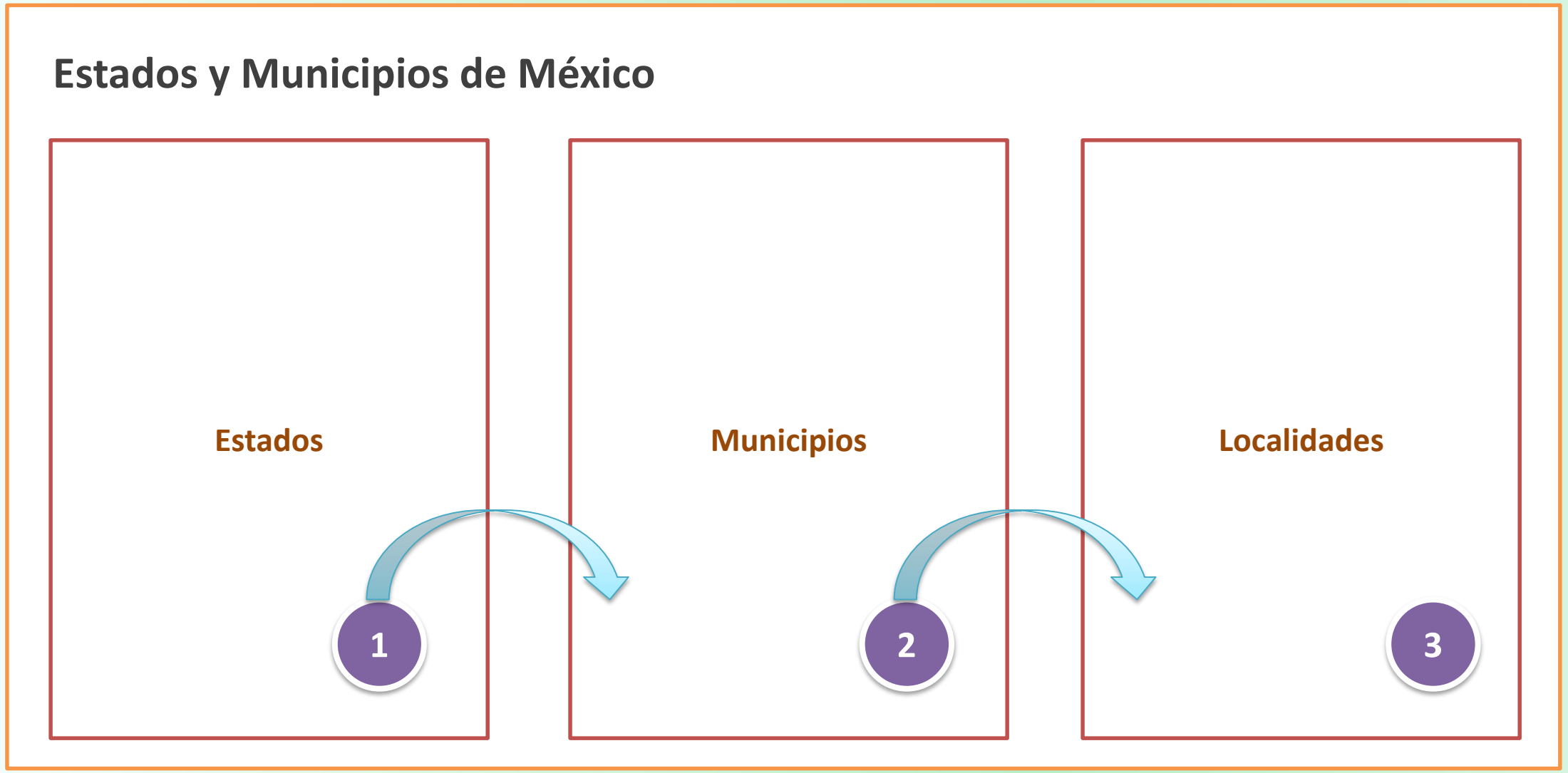
- Basado en el código anterior, genera la aplicación WPF y además:
  - Agrega la propiedad: FechaNacimiento
  - Modifica la propiedad **Edad**, a sólo lectura; hacer que devuelva la edad cumplida en años (considerar meses).
  - Adaptar la vista a los cambios anteriores; la edad no debe ser editable.

Ejercicio 2

A partir de la base de datos de Municipios de México, crea una aplicación MVVM donde:

- Muestre la lista de estados con No. de estado
- Al seleccionar un estado, muestre la lista de municipios con No. de municipio
- Al seleccionar un municipio, muestre la lista de localidades con CP.

Propuesta de diseño



2016