ACTIVIDAD #1

Diseñar interfaces graficas con alto nivel de usabilidad.

Ronald Ernesto Tejada Ríos

Tabla de contenido

[Creamos un projecto 2](#_Toc19870952)

[Creamos 3 carpetas para aplicar el MVVM design Pattern: 2](#_Toc19870953)

[Instalamos MVVMLightLibs 2](#_Toc19870954)

[Hacemos el archivo Operator: 2](#_Toc19870955)

[Creamos el BaseViewModel 2](#_Toc19870956)

[Ahora si vamos a trabajar en la vista de la calculadora: 3](#_Toc19870957)

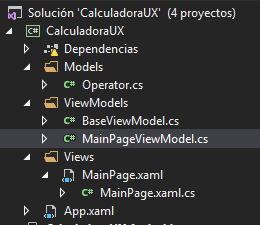
[Ahora solo resta hacer el ViewModel para la calculadora 6](#_Toc19870958)

[Ahora ejecutamos, ya tenemos la calculadora lista 11](#_Toc19870959)

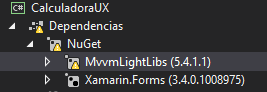
# Creamos un projecto

Lo llamaremos: CalculadoraUX

# Creamos 3 carpetas para aplicar el MVVM design Pattern:

 Debe lucir de esta manera.

## Instalamos MVVMLightLibs



## Hacemos el archivo Operator:

namespace CalculadoraUX.Models

{

public enum Operator

{

Addition,

Subtraction,

Division,

Multiplication

}

}

## Creamos el BaseViewModel

namespace CalculadoraUX.ViewModels

{

using System.ComponentModel;

public class BaseViewModel : INotifyPropertyChanged

{

public event PropertyChangedEventHandler PropertyChanged;

protected virtual void OnPropertyChanged(string propertyName = "")

{

if (PropertyChanged != null)

{

PropertyChanged(this, new PropertyChangedEventArgs(propertyName));

}

}

}

}

# Ahora si vamos a trabajar en la vista de la calculadora:

Lo importante acá es que hemos aplicado Styles para ahorrar el diseño de todos los botones, hay 3 categorías, los botones normales, los naranjas para las operaciones y los oscuros para borrar.

<?xml version="1.0" encoding="utf-8" ?>

<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"

xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"

xmlns:local="clr-namespace:CalculadoraUX"

x:Class="CalculadoraUX.MainPage"

Title="Calculator"

BackgroundColor="#404040"

xmlns:viewModel="clr-namespace:CalculadoraUX.ViewModels"

>

<ContentPage.BindingContext>

<viewModel:MainPageViewModel/>

</ContentPage.BindingContext>

<ContentPage.ToolbarItems>

<ToolbarItem Text="Ronald Ris" />

</ContentPage.ToolbarItems>

<ContentPage.Resources>

<ResourceDictionary>

<Style x:Key="plainButton" TargetType="Button">

<Setter Property="BackgroundColor" Value="#eee"/>

<Setter Property="TextColor" Value="Black" />

<Setter Property="BorderRadius" Value="0"/>

<Setter Property="FontSize" Value="40" />

</Style>

<Style x:Key="darkerButton" TargetType="Button">

<Setter Property="BackgroundColor" Value="#ddd"/>

<Setter Property="TextColor" Value="Black" />

<Setter Property="BorderRadius" Value="0"/>

<Setter Property="FontSize" Value="40" />

</Style>

<Style x:Key="orangeButton" TargetType="Button">

<Setter Property="BackgroundColor" Value="#E8AD00"/>

<Setter Property="TextColor" Value="White" />

<Setter Property="BorderRadius" Value="0"/>

<Setter Property="Command" Value="{Binding OperatorCommand}"/>

<Setter Property="FontSize" Value="40" />

</Style>

</ResourceDictionary>

</ContentPage.Resources>

<Grid x:Name="controlGrid" RowSpacing="1" ColumnSpacing="1"

VerticalOptions="Fill">

<Grid.RowDefinitions>

<RowDefinition Height="100" />

<RowDefinition Height="100" />

<RowDefinition Height="\*" />

<RowDefinition Height="\*" />

<RowDefinition Height="\*" />

<RowDefinition Height="\*" />

<RowDefinition Height="\*" />

</Grid.RowDefinitions>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="\*" />

<ColumnDefinition Width="\*" />

<ColumnDefinition Width="\*" />

<ColumnDefinition Width="\*" />

</Grid.ColumnDefinitions>

<Label Text="{Binding DisplayValueResult,StringFormat='Resultado: {0}'}" Grid.Row="0"

HorizontalTextAlignment="End" VerticalTextAlignment="End" TextColor="White"

FontSize="60" Grid.ColumnSpan="4" />

<Label Text="{Binding DisplayValue}" Grid.Row="1"

HorizontalTextAlignment="End" VerticalTextAlignment="End" TextColor="White"

FontSize="60" Grid.ColumnSpan="4" />

<Button Text = "DEL" Grid.Row="2" Grid.Column="0"

Command="{Binding DeleteCommand}"

Style="{StaticResource darkerButton}" />

<Button Text = "CE" Grid.Row="2" Grid.Column="1"

Command="{Binding ClearEntryCommand}"

Style="{StaticResource darkerButton}" />

<Button Text = "C" Grid.Row="2" Grid.Column="2"

Command="{Binding ClearCommand}"

Style="{StaticResource darkerButton}" />

<Button Text = "÷" Grid.Row="2" Grid.Column="3"

CommandParameter="/"

Style="{StaticResource orangeButton}" />

<Button Text = "7"

Grid.Row="3" Grid.Column="0"

Command="{Binding NumeroAddedCommand}" CommandParameter="7"

Style="{StaticResource plainButton}" />

<Button Text = "8"

Grid.Row="3" Grid.Column="1"

Command="{Binding NumeroAddedCommand}" CommandParameter="8"

Style="{StaticResource plainButton}" />

<Button Text = "9"

Grid.Row="3" Grid.Column="2"

Command="{Binding NumeroAddedCommand}" CommandParameter="9"

Style="{StaticResource plainButton}" />

<Button Text = "X"

Grid.Row="3" Grid.Column="3"

CommandParameter="\*"

Style="{StaticResource orangeButton}" />

<Button Text = "4"

Grid.Row="4" Grid.Column="0"

Command="{Binding NumeroAddedCommand}" CommandParameter="4"

Style="{StaticResource plainButton}" />

<Button Text = "5"

Grid.Row="4" Grid.Column="1"

Command="{Binding NumeroAddedCommand}" CommandParameter="5"

Style="{StaticResource plainButton}" />

<Button Text = "6"

Grid.Row="4" Grid.Column="2"

Command="{Binding NumeroAddedCommand}" CommandParameter="6"

Style="{StaticResource plainButton}" />

<Button Text = "-"

Grid.Row="4" Grid.Column="3"

CommandParameter="-"

Style="{StaticResource orangeButton}" />

<Button Text = "1"

Command="{Binding NumeroAddedCommand}" CommandParameter="1"

Grid.Row="5" Grid.Column="0"

Style="{StaticResource plainButton}" />

<Button Text = "2"

Command="{Binding NumeroAddedCommand}" CommandParameter="2"

Grid.Row="5" Grid.Column="1"

Style="{StaticResource plainButton}" />

<Button Text = "3"

Command="{Binding NumeroAddedCommand}" CommandParameter="3"

Grid.Row="5" Grid.Column="2"

Style="{StaticResource plainButton}" />

<Button Text = "+"

Grid.Row="5" Grid.Column="3"

CommandParameter="+"

Style="{StaticResource orangeButton}" />

<Button Text = "0"

Grid.ColumnSpan="2"

Command="{Binding NumeroAddedCommand}" CommandParameter="0"

Grid.Row="6" Grid.Column="0" Style="{StaticResource plainButton}" />

<Button Text = "."

Grid.Row="6" Grid.Column="2"

Command="{Binding NumeroAddedCommand}" CommandParameter="."

Style="{StaticResource plainButton}" />

<Button Text = "="

Grid.Row="6" Grid.Column="3"

Command="{Binding ComputeCommand}"

Style="{StaticResource orangeButton}" />

</Grid>

</ContentPage>

# Ahora solo resta hacer el ViewModel para la calculadora

Lo llamaremos: MainPageViewModel

namespace CalculadoraUX.ViewModels

{

using CalculadoraUX.Models;

using GalaSoft.MvvmLight.Command;

using System;

using System.Windows.Input;

public class MainPageViewModel : BaseViewModel

{

#region Atributos

/// <summary>

///Variables para uso de la clase y por eso son privadas

/// </summary>

private decimal? \_operandOne;

private decimal? \_operandTwo;

private decimal? resultPreviousDATO;

private Operator? \_operation;

private string \_displayValue;

private string \_displayValueResult;

private bool \_hasOperation;

#endregion

#region Propiedades

/// <summary>

/// Propiedades que me permite hacer publicos los cambios en los atributos

/// </summary>

public string DisplayValue

{

get { return \_displayValue; }

set

{

\_displayValue = value;

OnPropertyChanged("DisplayValue");

}

}

public string DisplayValueResult

{

get { return \_displayValueResult; }

set

{

\_displayValueResult = value;

OnPropertyChanged("DisplayValueResult");

}

}

public Operator? Operation

{

get { return \_operation; }

set

{

\_operation = value; OnPropertyChanged("Operation");

}

}

public decimal? OperandOne

{

get { return \_operandOne; }

set

{

\_operandOne = value; OnPropertyChanged("OperandOne");

}

}

public decimal? OperandTwo

{

get { return \_operandTwo; }

set

{

\_operandTwo = value; OnPropertyChanged("OperandTwo");

}

}

#endregion

#region Comandos

/// <summary>

/// Enlazan los comando del XAML

/// </summary>

public ICommand NumeroAddedCommand { get { return new RelayCommand<string>(NumeroAdded); } }

public ICommand OperatorCommand { get { return new RelayCommand<string>(OperatorCommandExecute); } }

public ICommand DeleteCommand { get { return new RelayCommand(DeleteCommandExecute); } }

public ICommand ClearEntryCommand { get { return new RelayCommand(ClearEntryCommandExecute); } }

public ICommand ClearCommand { get { return new RelayCommand(ClearCommandExecute); } }

public ICommand ComputeCommand { get { return new RelayCommand(ComputeCommandExecute); } }

#endregion

#region Constructor ctor

public MainPageViewModel()

{

resultPreviousDATO = null;

}

#endregion

#region Methodos

/// <summary>

/// vcuando presiona un numero o .

/// </summary>

/// <param name="value"></param>

private void NumeroAdded(string value)

{

if (resultPreviousDATO!=null)

{

resultPreviousDATO = null;

if (\_hasOperation)

{

Clear();

\_hasOperation = false;

}

}

decimal result;

string newDisplay = DisplayValue + value;

if (Decimal.TryParse(newDisplay, out result))

{

if (\_operation.HasValue)

{

\_operandTwo = result;

}

else

{

\_operandOne = result;

}

DisplayValue = newDisplay;

}

}

/// <summary>

/// Cuando presiona un operatio \* - + /

/// </summary>

/// <param name="op"></param>

private void OperatorCommandExecute(string op)

{

Operator? operation;

switch (op)

{

case "+":

operation = Operator.Addition;

break;

case "-":

operation = Operator.Subtraction;

break;

case "\*":

operation = Operator.Multiplication;

break;

case "/":

operation = Operator.Division;

break;

default:

throw new ArgumentException("Invalid Operator!");

}

if (resultPreviousDATO != null) //Esto lo agrego por si al terminar una operacion quiere seguir usando el resultado anterior

{

\_operandOne = resultPreviousDATO;

resultPreviousDATO = null;

\_operation = operation;

DisplayValue = string.Empty;

}

else

{

if (\_operandTwo.HasValue)

{

var calculation = Calculate();

if (calculation != 0)

{

\_operandOne = calculation;

\_operation = operation;

DisplayValueResult = \_operandOne.ToString();

DisplayValue = string.Empty;

}

else

{

Clear();

}

}

else if (\_operandOne.HasValue)

{

\_operation = operation;

DisplayValue = string.Empty;

}

}

}

/// <summary>

/// Borra un solo digito

/// </summary>

private void DeleteCommandExecute()

{

if (\_hasOperation)

return;

if (!String.IsNullOrWhiteSpace(DisplayValue))

{

DisplayValue = DisplayValue.Remove(DisplayValue.Length - 1);

}

}

/// <summary>

/// Borra el primer o segundo dato de la operacion

/// </summary>

private void ClearEntryCommandExecute()

{

if (\_operandTwo.HasValue)

{

\_operandTwo = null;

}

else

{

\_operation = null;

\_operandOne = null;

}

DisplayValue = string.Empty;

}

/// <summary>

/// Borra todos los datos

/// </summary>

private void ClearCommandExecute()

{

Clear();

}

private void Clear()

{

\_operandOne = null;

\_operandTwo = null;

\_operation = null;

DisplayValue = string.Empty;

}

/// <summary>

/// Cuand presiona = igual

/// </summary>

private void ComputeCommandExecute()

{

resultPreviousDATO = Calculate();

DisplayValueResult = decimal.Round((Decimal)resultPreviousDATO, 5).ToString();

DisplayValue = string.Empty;

\_operandOne = null;

\_operandTwo = null;

\_hasOperation = true;

}

public decimal Calculate()

{

if (\_operandOne.HasValue

&& \_operation.HasValue

&& \_operandTwo.HasValue)

{

switch (\_operation.Value)

{

case Operator.Addition:

return (\_operandOne.Value + \_operandTwo.Value);

case Operator.Subtraction:

return (\_operandOne.Value - \_operandTwo.Value);

case Operator.Multiplication:

return (\_operandOne.Value \* \_operandTwo.Value);

case Operator.Division:

try

{

return (Decimal)(\_operandOne.Value / \_operandTwo.Value);

}

catch (Exception s) { return 0.0m; }

default:

return 0.0m;

}

}

return 0.0m;

}

#endregion

}

}

# Ahora ejecutamos, ya tenemos la calculadora lista

El resultado debería de ser el siguiente:



**PREGUNTAS GUÍAS:**

1. ¿Qué comprende por usabilidad?

* Facilitarle la vida al usuario creando una app que tenga un proceso de aprendizaje fácil e intuitivo.

1. ¿Cómo se evalua la usabilidad en una aplicación movil?

* Creando contenido muy visual y que sea muy intuitivo para quien sea que use la aplicación.