ACTIVIDAD #3

Uso de Triggers.

Ronald Ernesto Tejada Ríos

Tabla de contenido

[Creamos un projecto 2](#_Toc19881093)

[Preparando MODELOS DE Styles en XAML 2](#_Toc19881094)

[Creamos las carpetas a utilizar 3](#_Toc19881095)

[Procedemos a instalar un nudget en todos los proyectos 3](#_Toc19881096)

[Agregamos el modelo “user” 4](#_Toc19881097)

[Hacemos el diseño de la MainPage 4](#_Toc19881098)

[Programamos el CodeBehind de la vista MainPage 6](#_Toc19881099)

[Creamos un ContentPage en Views llamado “MyStands” 9](#_Toc19881100)

[Ahora diseñamos el XAML de MyStands 9](#_Toc19881101)

[Ahora ejecutamos, ya tenemos la aplicación 10](#_Toc19881102)

# Creamos un projecto

* Lo llamaremos: TriggersStanding

# Preparando MODELOS DE Styles en XAML

* **En app.cs**

Modificamos de la siguiente manera:

public App()

{

InitializeComponent();

MainPage = new NavigationPage(new MainPage());

}

A modo que sea Navigation page para poder mostrar la barra de navegación

* **En app.XAML**

Agregamos lo siguiente:

<Application.Resources>

<ResourceDictionary>

<Color x:Key="appCryan">#0BD7E6</Color>

<Color x:Key="appDarkBlue">#1F4980</Color>

<Color x:Key="appDarkTone">#033C40</Color>

</ResourceDictionary>

<Style x:Key="pageFrame" TargetType="Frame">

<Setter Property="BackgroundColor" Value="LightGray"/>

<Setter Property="BorderColor" Value="{StaticResource appDarkBlue}" />

<Setter Property="CornerRadius" Value="25"/>

<Setter Property="HorizontalOptions" Value="Fill" />

<Setter Property="VerticalOptions" Value="Fill" />

<Setter Property="Margin" Value="12" />

</Style>

<Style x:Key="labelHeader" TargetType="Label">

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

<Setter Property="FontFamily" Value="IBMPlexMono-Bold"/>

<Setter Property="TextColor" Value="{StaticResource appDarkBlue}"/>

<Setter Property="Margin" Value="5,5"/>

</Style>

<Style x:Key="EntryStyle" TargetType="Entry">

<Setter Property="BackgroundColor" Value="LightGray"/>

<Setter Property="FontAttributes" Value="Bold"/>

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

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

<Setter Property="FontFamily" Value="IBMPlexMono-SemiBoldItalic"/>

</Style>

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

</Style>

</Application.Resources>

* **En el proyecto de ANDROID/Resources/Drawable**

Agregamos dos imágenes:

datelove.png



StandingLayer.png



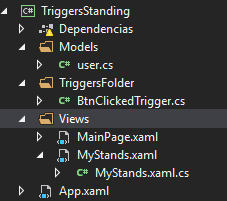
* **En el proyecto de ANDROID/Assets**

Agregamos las tipografías que se encuentran en el proyecto

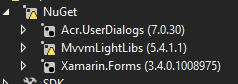
\* IBMPlexMono-SemiBoldItalic

\* IBMPlexMono-Bold

## Creamos las carpetas a utilizar y de paso los archivos



## Procedemos a instalar un nudget en todos los proyectos



NO OLVIDES INICIALIZAR EL NUDGET DE USERDIALOGS

En MainActivity del Proyecto de Android:

Acr.UserDialogs.UserDialogs.Init(this);

Agrega esa línea justo antes de: LoadApplication(new App());

* MVVMLightLibs nos va a facilitar el uso de Commands y ICommans
* Acr.UserDialogs para mostrar mensajes más vistosos de alertas

# Agregamos el modelo “user”

En la carpeta de Models

using System.Collections.Generic;

public class user

{

public string Name { get; set; }

public string Username { get; set; }

public string Pass { get; set; }

public string Img { get; set; }

public bool Active { get; set; }

public List<string> Lcitas { get; set; }

public user()

{

Lcitas = new List<string>();

}

}

# Agregamos en TriggersFolder la clase “BtnClickedTrigger”

using Xamarin.Forms;

public class BtnClickedTrigger : TriggerAction<Button>

{

protected override void Invoke(Button sender)

{

sender.BackgroundColor = Color.Blue;

sender.TextColor = Color.White;

}

}

# Hacemos el diseño de la MainPage

* Cuidado con el XMLNS Si el proyecto se llama distinto

<?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:d="http://xamarin.com/schemas/2014/forms/design"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

mc:Ignorable="d"

x:Class="TheStanding19.Views.MainPage"

NavigationPage.HasNavigationBar="True"

Icon="datelove"

Title="Dating App"

xmlns:TriggersClass="clr-namespace: TriggersStanding.TriggersFolders"

>

<ContentPage.Resources>

<ResourceDictionary>

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

<Setter Property="BackgroundColor" Value="LightGray"/>

<Setter Property="BorderColor" Value="Blue" />

<Setter Property="BorderRadius" Value="20"/>

<Setter Property="BorderWidth" Value="2" />

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

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

<Setter Property="FontAttributes" Value="Bold"/>

<Setter Property="FontFamily" Value="IBMPlexMono-SemiBoldItalic"/>

<!--TRIGGERS FOR STYLE!! Para el evento Clicked!-->

<Style.Triggers>

<EventTrigger Event="Clicked" >

<TriggersClass:BtnClickedTrigger/>

</EventTrigger>

</Style.Triggers>

</Style>

</ResourceDictionary>

</ContentPage.Resources>

<ContentPage.ToolbarItems>

<ToolbarItem Text="My Standings" Command="{Binding MyStanding}"/>

</ContentPage.ToolbarItems>

<ContentPage.Content>

<ScrollView IsClippedToBounds="True">

<StackLayout BackgroundColor="LightGray">

<StackLayout HorizontalOptions="Fill">

<Image

Source="StandinLayer" HorizontalOptions="Fill"

VerticalOptions="Start"

/>

</StackLayout>

<StackLayout>

<Label

Style="{StaticResource labelHeader}"

Text="DATE NAME:"

/>

<Editor

Placeholder="Describe what-s the date about"

HeightRequest="150"

HorizontalOptions="FillAndExpand"

Margin="5,5"

x:Name="txtEditor"

/>

<StackLayout Orientation="Horizontal" HeightRequest="80" HorizontalOptions="Fill">

<Label Margin="6,4,3,4"

Text="Fecha:"

Style="{StaticResource labelHeader}"

HorizontalOptions="Start"

VerticalOptions="CenterAndExpand"

/>

<DatePicker

FontFamily="IBMPlexMono-SemiBoldItalic"

FontSize="Medium"

TextColor="Black"

HorizontalOptions="CenterAndExpand"

VerticalOptions="CenterAndExpand"

x:Name="DateFecha"

/>

</StackLayout>

<Label Text="How do you want me to be?"

FontFamily="IBMPlexMono-SemiBoldItalic"

FontSize="Medium"

TextColor="Blue"

HorizontalOptions="CenterAndExpand"

/>

<Grid Margin="5,10" HorizontalOptions="CenterAndExpand">

<Grid.ColumnDefinitions>

<ColumnDefinition Width="\*"/>

<ColumnDefinition Width="\*"/>

</Grid.ColumnDefinitions>

<!--PAR 1-->

<Button Text="FUNNY"

Style="{StaticResource btnStyle}"

x:Name="btn1"

Grid.Column="0"

Grid.Row="0"

>

<Button.Triggers>

<DataTrigger TargetType="Button" Binding="{Binding Source={x:Reference btn2},

Path=BackgroundColor}" Value="Blue">

<Setter Property="BackgroundColor" Value="LightGray"/>

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

</DataTrigger>

</Button.Triggers>

</Button>

<Button Text="BORING"

Style="{StaticResource btnStyle}"

x:Name="btn2"

Grid.Column="1"

Grid.Row="0"

>

<Button.Triggers>

<DataTrigger TargetType="Button" Binding="{Binding Source={x:Reference btn1},

Path=BackgroundColor}" Value="Blue">

<Setter Property="BackgroundColor" Value="LightGray"/>

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

</DataTrigger>

</Button.Triggers>

</Button>

<!--PAR 2-->

<Button Text="TALKER"

Style="{StaticResource btnStyle}"

x:Name="btn3"

Grid.Column="0"

Grid.Row="1"

>

<Button.Triggers>

<DataTrigger TargetType="Button" Binding="{Binding Source={x:Reference btn4},

Path=BackgroundColor}" Value="Blue">

<Setter Property="BackgroundColor" Value="LightGray"/>

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

</DataTrigger>

</Button.Triggers>

</Button>

<Button Text="LISTENER"

Style="{StaticResource btnStyle}"

x:Name="btn4"

Grid.Column="1"

Grid.Row="1"

>

<Button.Triggers>

<DataTrigger TargetType="Button" Binding="{Binding Source={x:Reference btn3},

Path=BackgroundColor}" Value="Blue">

<Setter Property="BackgroundColor" Value="LightGray"/>

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

</DataTrigger>

</Button.Triggers>

</Button>

<!--PAR 3-->

<Button Text="CONSERVATIVE"

Style="{StaticResource btnStyle}"

x:Name="btn5"

Grid.Column="0"

Grid.Row="2"

>

<Button.Triggers>

<DataTrigger TargetType="Button" Binding="{Binding Source={x:Reference btn6},

Path=BackgroundColor}" Value="Blue">

<Setter Property="BackgroundColor" Value="LightGray"/>

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

</DataTrigger>

</Button.Triggers>

</Button>

<Button Text="INQUISITIVE"

Style="{StaticResource btnStyle}"

x:Name="btn6"

Grid.Column="1"

Grid.Row="2"

>

<Button.Triggers>

<DataTrigger TargetType="Button" Binding="{Binding Source={x:Reference btn5},

Path=BackgroundColor}" Value="Blue">

<Setter Property="BackgroundColor" Value="LightGray"/>

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

</DataTrigger>

</Button.Triggers>

</Button>

</Grid>

<Button Style="{StaticResource btnStyle}" x:Name="btnEnviar"

Text="Enviar Datos"

Clicked="BtnEnviar\_Clicked"

>

</Button>

</StackLayout>

</StackLayout>

</ScrollView>

</ContentPage.Content>

</ContentPage>

## 

## Programamos el CodeBehind de la vista MainPage

Esto lo vamos a agregar justo dentro de la clase MainPage

public static user ActiveUser;

public ICommand MyStanding { get { return new RelayCommand(MyStandingsPage); } }

public MainPage()

{

InitializeComponent();

ActiveUser = new user { Active = true, Img = "myAvatar.png", Name = "Ronald", Username = "Ris", Pass = "2525" };

BindingContext = this;

RestartColors();

}

private void MyStandingsPage()

{

Navigation.PushAsync(new MyStands(ActiveUser));

}

private void RestartColors()

{

btn1.TextColor = Color.White;

btn3.TextColor = Color.White;

btn5.TextColor = Color.White;

btn1.BackgroundColor = Color.Blue;

btn3.BackgroundColor = Color.Blue;

btn5.BackgroundColor = Color.Blue;

}

private async void BtnEnviar\_Clicked(object sender, EventArgs e)

{

if (txtEditor.Text == "" || txtEditor.Text == null)

{

UserDialogs.Instance.Toast(new ToastConfig("La descrición no puede quedar vacia").SetPosition(ToastPosition.Top).SetDuration(1500)

.SetBackgroundColor(Color.Red)

.SetMessageTextColor(Color.Black));

}

else

{

string carac = "";

if (btn1.BackgroundColor == Color.Blue)

{

carac += btn1.Text + "\n";

}

if (btn2.BackgroundColor == Color.Blue)

{

carac += btn2.Text + "\n";

}

if (btn3.BackgroundColor == Color.Blue)

{

carac += btn3.Text + "\n";

}

if (btn4.BackgroundColor == Color.Blue)

{

carac += btn4.Text + "\n";

}

if (btn5.BackgroundColor == Color.Blue)

{

carac += btn5.Text + "\n";

}

if (btn6.BackgroundColor == Color.Blue)

{

carac += btn6.Text + "\n";

}

String text = "Contenido: " + txtEditor.Text + "\n" +

"Fecha cita " + DateFecha.Date + "\n" +

"Caracteristicas:\n" + carac;

if (await DisplayAlert("Confirmación", text, "Ok", "Cancel"))

{

UserDialogs.Instance.Toast(new ToastConfig("Procesando cita").SetPosition(ToastPosition.Top).SetDuration(1000)

.SetBackgroundColor(Color.Green)

.SetMessageTextColor(Color.Black));

ActiveUser.Lcitas.Add(text);

txtEditor.Text = "";

RestartColors();

}

else

{

UserDialogs.Instance.Toast(new ToastConfig("Cita cancelada")

.SetPosition(ToastPosition.Top).SetDuration(1000)

.SetBackgroundColor(Color.Red)

.SetMessageTextColor(Color.Black));

}

}

}

Finalmente agregamos los using que hacen falta.

Habrá un error y es porque hay que modificar el Constructor del archivo MyStands

# Views/ “MyStands”

Acá vamos a cambiar el constructor para que reciba un objeto de tipo user

public MyStands(user oUser)

{

InitializeComponent();

BindingContext = oUser;

}

* Y agregamos la referencia para user
* Ahora regresamos al MainPage y agregamos la referencia using para acceder a MyStands

## Ahora diseñamos el XAML de MyStands

<?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:d="http://xamarin.com/schemas/2014/forms/design"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

mc:Ignorable="d"

x:Class="TheStanding19.Views.MyStands"

Title="{Binding Username}">

<ContentPage.Content>

<StackLayout>

<Label Text="Citas agendadas:"

Style="{StaticResource labelHeader}"

/>

<ScrollView>

<StackLayout>

<ListView ItemsSource="{Binding Lcitas}"

RowHeight="200">

<ListView.ItemTemplate>

<DataTemplate>

<ViewCell>

<Frame Style="{Binding pageFrame}">

<Editor Text="{Binding}"

HeightRequest="180"

HorizontalOptions="Center"

VerticalOptions="Center"

TextColor="Black"

FontAttributes="Bold"

/>

</Frame>

</ViewCell>

</DataTemplate>

</ListView.ItemTemplate>

</ListView>

</StackLayout>

</ScrollView>

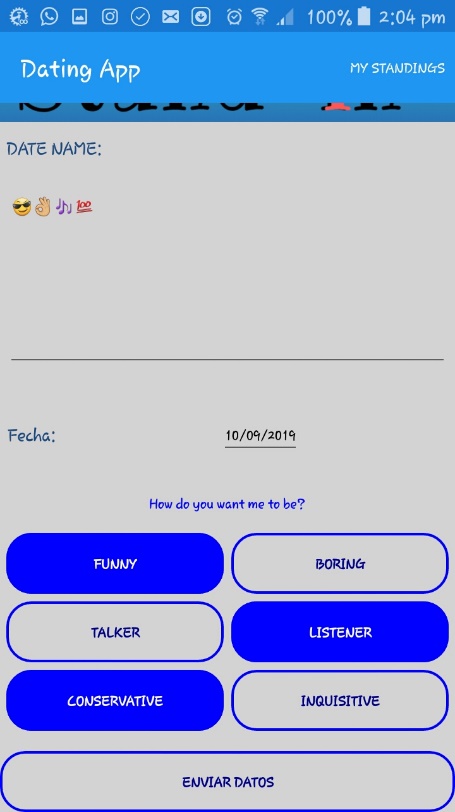
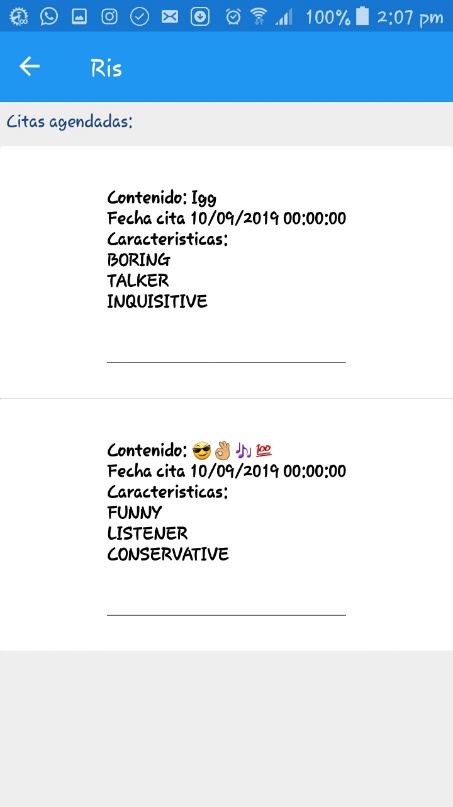
</StackLayout>

</ContentPage.Content>

</ContentPage>

# Ahora ejecutamos, ya tenemos la aplicación

El resultado debería de ser el siguiente:

Igual que la app de la Actividad 2, pero ahora funciona con triggers

**PREGUNTAS GUÍAS:**

1. ¿Cuál es la utilidad de los Triggers en Xamarin Form?

Ayuda a cambiar atributos del UI a través de eventos manejados internamente desde las vistas o bien con clases externas que ejecuten cierto código según el evento que sucedió a X Target

1. ¿Defina la sintaxis de los Triggers en Xamarin?

## Property Triggers

<Entry Placeholder="enter name">

<Entry.Triggers>

<Trigger TargetType="Entry"

Property="IsFocused" Value="True">

<Setter Property="BackgroundColor" Value="Yellow" />

</Trigger>

</Entry.Triggers>

</Entry>

### Applying a Trigger using a Style

<ContentPage.Resources>

<ResourceDictionary>

<Style TargetType="Entry">

<Style.Triggers>

<Trigger TargetType="Entry"

Property="IsFocused" Value="True">

<Setter Property="BackgroundColor" Value="Yellow" />

</Trigger>

</Style.Triggers>

</Style>

</ResourceDictionary>

</ContentPage.Resources>

## Data Triggers

<Entry x:Name="entry"

Text=""

Placeholder="required field" />

<Button x:Name="button" Text="Save"

FontSize="Large"

HorizontalOptions="Center">

<Button.Triggers>

<DataTrigger TargetType="Button"

Binding="{Binding Source={x:Reference entry},

Path=Text.Length}"

Value="0">

<Setter Property="IsEnabled" Value="False" />

</DataTrigger>

</Button.Triggers>

</Button>

## Event Triggers

<EventTrigger Event="Clicked">

<local:NumericValidationTriggerAction />

</EventTrigger>

public class NumericValidationTriggerAction : TriggerAction<Entry>

{

protected override void Invoke (Entry entry)

{

double result;

bool isValid = Double.TryParse (entry.Text, out result);

entry.TextColor = isValid ? Color.Default : Color.Red;

}

}