ACTIVIDAD #4

Uso de Behaviors.

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)

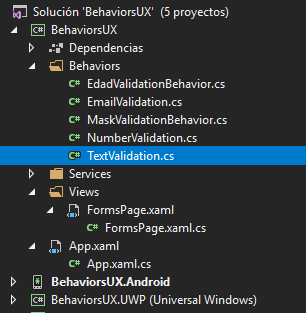
[Ahora diseñamos el XAML de MyStands 9](#_Toc19881101)

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

# Creamos un projecto

* Lo llamaremos: BehaviorsUX

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



Todas las clases son publicas

# Agregamos en Behaviors Folder la clase “EdadValidationBehavior”

using System;

using Xamarin.Forms;

public class EdadValidationBehavior : Behavior<Entry>

{

protected override void OnAttachedTo(Entry bindable)

{

bindable.TextChanged += EdadValidationMethod;

base.OnAttachedTo(bindable);

}

private void EdadValidationMethod(object sender, TextChangedEventArgs e)

{

try

{

int edad = Int32.Parse(e.NewTextValue);

bool isValid = (edad >=13 && edad <=110);

//Casting del objeto sender para poder modificarlo directamente a él!!

((Entry)sender).TextColor = isValid ? Color.Green : Color.Red;

((Entry)sender).BackgroundColor = isValid ? Color.LightGreen : Color.LightCoral;

}

catch (Exception)

{

((Entry)sender).TextColor = Color.Red;

((Entry)sender).BackgroundColor = Color.LightCoral;

}

}

protected override void OnDetachingFrom(Entry bindable)

{

base.OnDetachingFrom(bindable);

}

# Agregamos en Behaviors Folder la clase “EmailValidation”

using System.Text.RegularExpressions;

using Xamarin.Forms;

public class EmailValidation : Behavior<Entry>

{

protected override void OnAttachedTo(Entry bindable)

{

base.OnAttachedTo(bindable);

bindable.TextChanged += ValidandoEmail;

}

protected override void OnDetachingFrom(Entry bindable)

{

base.OnDetachingFrom(bindable);

bindable.TextChanged -= ValidandoEmail;

}

bool IsValidEmail(string email)

{

try

{

var mail = new System.Net.Mail.MailAddress(email);

return true;

}

catch

{

return false;

}

}

private void ValidandoEmail(object sender, TextChangedEventArgs e)

{

Regex reg = new Regex(@"^([a-zA-Z0-9\_\-\.]+)@([a-zA-Z0-9\_\-\.]+)\.(com|org|sv|edu|es)(\.(com|org|sv|edu|es))?$");

var correo = e.NewTextValue;

bool IsValid = reg.IsMatch(correo);

((Entry)sender).BackgroundColor = IsValid ? Color.LightGreen : Color.LightCoral;

}

}

# Agregamos en Behaviors Folder la clase “MaskValidationBehavior”

using Xamarin.Forms;

public class MaskValidationBehavior : Behavior<Entry>

{

private string \_mask = null;

public string Mask

{

get => \_mask;

set

{

\_mask = value;

}

}

protected override void OnAttachedTo(Entry entry)

{

entry.TextChanged += OnNumChanged;

base.OnAttachedTo(entry);

}

private void OnNumChanged(object sender, TextChangedEventArgs args)

{

if (!string.IsNullOrWhiteSpace(Mask))

{

var entry = sender as Entry; //Asi me aseguro de seguir usando el mismo objeto!

var text = entry.Text;

//Agregando el tamaño maximo del texto en caso de que tenga otro distinto al patrom

if (entry.MaxLength != \_mask.Length)

entry.MaxLength = \_mask.Length;

//Evaluando para que solo tome cuando escribe y no cuando borra

if ((args.OldTextValue == null) || (args.OldTextValue.Length <= args.NewTextValue.Length))

//Evaluando posiciones de la mascara

if (Mask[(text.Length - 1)] != 'X')

{

text = text.Insert((text.Length - 1), Mask[(text.Length - 1)].ToString());

}

entry.Text = text;

entry.BackgroundColor = (text.Length == Mask.Length) ? Color.LightGreen : Color.LightCoral;

}

}

protected override void OnDetachingFrom(Entry entry)

{

entry.TextChanged -= OnNumChanged;

base.OnDetachingFrom(entry);

}

}

# Agregamos en Behaviors Folder la clase “NumberValidation”

using System.Text.RegularExpressions;

using Xamarin.Forms;

class NumberValidation : Behavior<Entry>

{

protected override void OnAttachedTo(Entry bindable)

{

bindable.TextChanged += NumberValidationMethod;

}

private void NumberValidationMethod(object sender, TextChangedEventArgs e)

{

Regex reg = new Regex("^[0-9]{8}$");

bool isValid = reg.IsMatch(e.NewTextValue);

//Casting del objeto sender para poder modificarlo directamente a él!!

((Entry)sender).TextColor = isValid ? Color.Green : Color.Red;

((Entry)sender).BackgroundColor = isValid ? Color.LightGreen : Color.LightCoral;

}

protected override void OnDetachingFrom(Entry bindable)

{

bindable.TextChanged -= NumberValidationMethod;

}

}

# Agregamos en Behaviors Folder la clase “TextValidation”

using System.Text.RegularExpressions;

using Xamarin.Forms;

public class TextValidation : Behavior<Entry>

{

private void TextValidationMethod(object sender, TextChangedEventArgs e)

{

Regex reg = new Regex("^[A-Za-z áéíóúÁÉÍÓÚ]{5,25}$");

bool isValid = reg.IsMatch(e.NewTextValue);

//Casting del objeto sender para poder modificarlo directamente a él!!

((Entry)sender).TextColor = isValid ? Color.DarkGreen : Color.Red;

((Entry)sender).BackgroundColor = isValid ? Color.LightGreen : Color.LightCoral;

}

protected override void OnAttachedTo(Entry bindable)

{

bindable.TextChanged += TextValidationMethod;

base.OnAttachedTo(bindable);

}

protected override void OnDetachingFrom(Entry bindable)

{

base.OnDetachingFrom(bindable);

bindable.TextChanged -= TextValidationMethod;

}

}

# Hacemos el diseño de la vista FormsPage

* 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"

x:Class="BehaviorsUX.Views.FormsPage"

xmlns:be="clr-namespace:BehaviorsUX.Behaviors"

>

<ContentPage.Content>

<StackLayout Margin="10,10,10,0">

<Label Text="Formulario del participante:"

Margin="0,5,0,0"

HorizontalOptions="CenterAndExpand"

/>

<Label Text="Nombres"/>

<Entry Placeholder="Ingrese su nombre">

<Entry.Behaviors>

<be:TextValidation/>

</Entry.Behaviors>

</Entry>

<Label Text="Apellidos"/>

<Entry Placeholder="Ingrese su apellidos">

<Entry.Behaviors>

<be:TextValidation/>

</Entry.Behaviors>

</Entry>

<Label Text="Edad"/>

<Entry Placeholder="Ingrese su edad"

Keyboard="Numeric"

MaxLength="3">

<Entry.Behaviors>

<be:EdadValidationBehavior/>

</Entry.Behaviors>

</Entry>

<Label Text="Telefono"/>

<Entry Placeholder="Ingrese su telefono"

Keyboard="Numeric"

MaxLength="8">

<Entry.Behaviors>

<be:NumberValidation/>

</Entry.Behaviors>

</Entry>

<Label Text="Email"/>

<Entry Placeholder="Ingrese su email"

MaxLength="50">

<Entry.Behaviors>

<be:EmailValidation/>

</Entry.Behaviors>

</Entry>

<Label Text="DUI"/>

<Entry Placeholder="Ingrese su DUI"

Keyboard="Numeric"

>

<Entry.Behaviors>

<be:MaskValidationBehavior Mask="XXXXXXXX-X"/>

</Entry.Behaviors>

</Entry>

<Label Text="NIT"/>

<Entry Placeholder="Ingrese su NIT"

Keyboard="Numeric"

>

<Entry.Behaviors>

<be:MaskValidationBehavior Mask="XXXX-XXXXXX-XXX-X"/>

</Entry.Behaviors>

</Entry>

<!--<Button Text="Confirmar datos"

Clicked="Button\_Clicked">

</Button>-->

</StackLayout>

</ContentPage.Content>

## </ContentPage>

## Nos aseguramos que App empiece por FormsPage

public App()

{

InitializeComponent();

MainPage = new FormsPage();

}

# Ejecutamos finalmente y todo debe de funcionar correctamente