1.- Crear las carpetas Models, MainViewModel y Views

1.1.- Crear una ViewModel principal, MainViewModel, desde donde arrancará el proyecto

1.2.- Crear la BaseViewModel para implementar el INotifyPropertyChanged:

public abstract class ViewModelBase: INotifyPropertyChanged

{

#region INotifyPropertyChanged implementation

public event PropertyChangedEventHandler PropertyChanged;

#endregion

protected void Notify(string propertyName)

{

if (this.PropertyChanged != null)

this.PropertyChanged(this, new PropertyChangedEventArgs(propertyName));

}

}

2.- Crear la infraestructura con el InstanceLocator

3.- Crear la clase para el servicio de mensajería. Para ello, se crea una carpeta Services en la raíz del proyecto, y dentro de la carpeta se crea la clase MessageService.cs la cual va a contener el siguiente código:

public class MessageService

{

public async Task SendMessage(string title, string message)

{

await MyStock.App.Current.MainPage.DisplayAlert(title, message, "Accept");

}

}

4.- Crear las View

4.1.- Se crea la primera View, LoginView, esta será la pantalla de inicio de la app. Esto hace que, en el constructor de la MainViewModel se cree una instancia de la LoginView, ya que siendo la MainViewModel, la principal, esta tiene que instanciar la vista de arranque del proyecto.

4.2.- Dentro del LoginView.xaml se debe agregar el bindingcontext principal hacia la MainViewModel, y luego, dentro del cuerpo del stacklayout se incorpora el bindingcontext hacia la LoginViewModel.

5.- Indicar en el code behind de App.axml el MainPage apuntando a una NavigationPage que arranca por LoginView

6.- Desarrollar la estructura de la LoginView.xam

Register

Login with Facebook

Login

Forgot your password?

Remember me

Enter your email….

Password

Enter your password….

Email

PRODUCTS

Login

El código para hacer la anterior interfaz en el LoginView.xaml es:

<?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="MyStock.Views.LoginView"

BindingContext="{Binding MvM, Source={StaticResource Locator}}"

Title="Login">

<ContentPage.Content>

<StackLayout

BindingContext="{Binding Login}"

Padding="8"

Spacing="16">

<Label Text="Products"

FontSize="Large"

FontAttributes="Bold"

HorizontalOptions="Center" />

<Grid

Margin="50,10">

<Grid.ColumnDefinitions>

<ColumnDefinition Width="\*"/>

<ColumnDefinition Width="2\*"/>

</Grid.ColumnDefinitions>

<Label Text="Email"

Grid.Row="0"

Grid.Column="0"/>

<Entry Text="{Binding Email, Mode=TwoWay}"

Placeholder="Enter your email..."

Grid.Row="0"

Grid.Column="1"

Keyboard="Email"/>

<Label Text="Password"

Grid.Row="1"

Grid.Column="0"/>

<Entry Text="{Binding Password, Mode=TwoWay}"

Placeholder="Enter your password..."

Grid.Row="1"

Grid.Column="1"

IsPassword="True"/>

</Grid>

<StackLayout Orientation="Horizontal">

<Label Text="Remember me"

VerticalOptions="Center"/>

<Switch

IsToggled="{Binding IsToggled, Mode=TwoWay}"/>

</StackLayout>

<Label Text="Forgot your password?"

TextColor="Blue"

HorizontalOptions="Center">

<Label.GestureRecognizers>

<TapGestureRecognizer Command="{Binding RecoverPasswordCommand}"/>

</Label.GestureRecognizers>

</Label>

<ActivityIndicator

IsRunningLabel="{Binding IsRunning, Mode=TwoWay}"/>

<StackLayout Margin="50,0">

<Button Text="Login"

BackgroundColor="Navy"

Style="{StaticResource GeneralButton}"

IsEnabled="{Binding IsEnabled, Mode=TwoWay}"

Command="{Binding LoginCommand}"/>

<Button Text="Register"

BackgroundColor="Blue"

Style="{StaticResource GeneralButton}"

IsEnabled="{Binding IsEnabled, Mode=TwoWay}"

Command="{Binding LoginWithFacebookCommand}"/>

<Button Text="Register"

BackgroundColor="Blue"

Style="{StaticResource GeneralButton}"

IsEnabled="{Binding IsEnabled, Mode=TwoWay}"

Command="{Binding RegisterNewUserCommand}"/>

</StackLayout>

</StackLayout>

</ContentPage.Content>

</ContentPage>

7.- Instalar a través del Manager NuGet Package el MvvmLightLibs, la cual va a permitir asociar un binding sobre un command.

8.- Desarrollar el código de la LoginViewModel para procesar las acciones que se den desde la LoginView y verificar la existencia del usuario desde una Web API. Se debe agregar el paquete Microsoft.Net.Http V.2.2.29 (incluyendo BCL y Build) y Newtonsoft Json V10. Adicionalmente vamos a requerir el plugin Xam.Plugin.Connectivity V 3.1.1 para verificar la conectividad a Internet. Este último plugin debe agregarse también dentro del proyecto .Android. El código quedaría así:

using MyStock.Services;

using System;

using System.Collections.Generic;

using System.Text;

using System.Windows.Input;

namespace MyStock.ViewModels

{

public class LoginViewModel : ViewModelBase

{

string email;

public string Email

{

get

{

return email;

}

set

{

email = value;

this.Notify("Email");

}

}

string password;

public string Password

{

get

{

return password;

}

set

{

password = value;

this.Notify("Password");

}

}

bool istoggled;

public bool IsToggled

{

get

{

return istoggled;

}

set

{

istoggled = value;

this.Notify("IsToggled");

}

}

bool isrunning;

public bool IsRunning

{

get

{

return isrunning;

}

set

{

isrunning = value;

this.Notify("IsRunning");

}

}

bool isenabled;

public bool IsEnabled

{

get

{

return isenabled;

}

set

{

isenabled = value;

this.Notify("IsEnabled");

}

}

MessageService messageService;

ApiService apiService; \*\*\*

public LoginViewModel()

{

messageService = new MessageService();

IsEnabled = true;

IsToggled = true;

}

public ICommand LoginCommand

{

get;

set;

}

public ICommand LoginWithFacebookCommand

{

get;

set;

}

public ICommand RegisterNewUserCommand

{

get;

set;

}

async void Login()

{

if (string.IsNullOrEmpty(Email))

{

await messageService.SendMessage("Error", "You must enter an email");

return;

}

if (string.IsNullOrEmpty(Password))

{

await messageService.SendMessage("Error", "You must enter an password");

return;

}

IsRunning = true;

IsEnabled = false;

\*\*\* var connection = await apiService.CheckConnection();

if (connection.IsSuccess)

{

//\*\*\*nuevas líneas de código de acuerdo con el pto 10

}

}

}

}

9.- Crear el servicio API. Para ello creamos una clase ApiService dentro de la carpeta Services. Luego creamos una clase Response dentro de la carpeta Models la cual va a contener el siguiente código:

public class Response

{

public bool IsSuccess { get; set; } //Propiedad para verificar si hay conexión

public string Message { get; set; } //Enviar mensaje dependiendo de si conecto o no

public Object Result { get; set; } //Obtener la respuesta http

}

9.1. Asimismo, se va a crear dentro de la carpeta Models una clase llamada TokenResponse con la finalidad de contener los parámetros del token que se va a generar para validar el acceso al usuario. El código de esta clase queda así:

public class TokenResponse

{

[JsonProperty(PropertyName ="access\_token")] //El uso de esta propiedad de Json permite tomar el nombre de la propiedad tal como es generada y mapearla con un nombre mas adecuado que se le de a dicha propiedad dentro de la clase.

public string AccessToken { get; set; }

[JsonProperty(PropertyName ="token\_type")]

public string TokenType { get; set; }

[JsonProperty(PropertyName = "expires\_in")]

public int ExpiresIn { get; set; }

[JsonProperty(PropertyName = "userName")]

public string UserName { get; set; }

[JsonProperty(PropertyName = ".issued")]

public DateTime Issued { get; set; }

[JsonProperty(PropertyName = ".expires")]

public DateTime Expires { get; set; }

}

9.2 Se desarrollar el código de la clase ApiService quedando como sigue:

public class ApiService

{

public async Task<Response> CheckConnection()

{

if (!CrossConnectivity.Current.IsConnected)

{

return new Response

{

IsSuccess = false,

Message = "Please check your Internet settings.",

};

}

var isReachable = await CrossConnectivity.Current.IsRemoteReachable("google.com");

if (!isReachable)

{

return new Response

{

IsSuccess = false,

Message = "Please check your Internet connection.",

};

}

return new Response

{

IsSuccess = true,

Message = "OK",

};

}

public async Task<TokenResponse> GetToken(string urlBase, string username, string password) //Este es el método que obtiene el token

{

try

{

var client = new HttpClient();

client.BaseAddress = new Uri(urlBase);

var response = await client.PostAsync("Token", new StringContent(string.Format(

"grant\_type=password&username={0}&password={1}",username, password),Encoding.UTF8,

"application/x-www-form-urlencoded"));

var resultJSON = await response.Content.ReadAsStringAsync();

var result = JsonConvert.DeserializeObject<TokenResponse>(resultJSON);

return result;

}

catch

{

return null;

}

}

}

Ver la clase ServiceApi.cs para copiar todo el código

10. Nos regresamos a la LoginViewModel, para instanciar al ApiService creado anteriormente\*\*\*