Lo que sigue

1. Search bar.
2. Pantalla de login.
3. Obtener token.
4. Guardar datos en persistencia.
5. Consumir servicios de forma segura.
6. Crear una Master Detail.
7. Base de datos SQLite.
8. Registrar nuevos usuarios por la App.

# 

# Contenido

[**Contenido**](#_ixz623u13nrk) **2**

[**Nuevos literales**](#_ixh11be0veql) **3**

[Inglés](#_51hrc3uy7tqb) 3

[Español](#_xeqruinyi9ze) 4

[Italiano](#_yz207obmvx4b) 5

[Portugués](#_gb3wlb8ymgxc) 6

[Languages](#_11bmuref7q9i) 7

[**Pantalla login**](#_x4zwirtoq6yx) **10**

[**Obtener un token**](#_7mqsq0olas3x) **13**

[**Master Detail**](#_wct6spzbng1y) **15**

[XAML](#_flq1etelkk34) 15

[C#](#_amv0vggja5ez) 15

[MenuPage](#_muxmeuquuynt) 16

[MenuItem ViewModel](#_on0eogd8bifs) 17

[Load menu in MainViewModel](#_z8iu4n7ghxli) 17

[Crear propiedad:](#_sv57hw50zn9k) 17

[Crear método:](#_d2epucxp49j4) 17

[**Consumir servicio de forma segura**](#_oowyst5zffei) **18**

[**Nuevos Literales**](#_h1l85yg1t7dj) **18**

[En Inglés](#_cphz1nbrzzie) 18

[En Español](#_g2xvpqf67jiw) 19

[En Italiano](#_jmiob1wfeesm) 21

[En Portugués](#_bjz996wvlms7) 22

[En Languages](#_sz4rgg20xqnn) 23

[**SQLite**](#_jaizduaqtikd) **25**

[**Registrar Nuevos Usuarios**](#_g9fyxiy4wpit) **29**

# 

# Nuevos literales

## Inglés

<data name="Search" xml:space="preserve">

<value>Search...</value>

</data>

<data name="Login" xml:space="preserve">

<value>Login</value>

</data>

<data name="EMail" xml:space="preserve">

<value>EMail</value>

</data>

<data name="EmailPlaceHolder" xml:space="preserve">

<value>Enter your email...</value>

</data>

<data name="Password" xml:space="preserve">

<value>Password</value>

</data>

<data name="PasswordPlaceHolder" xml:space="preserve">

<value>Enter your password...</value>

</data>

<data name="Rememberme" xml:space="preserve">

<value>Remember me</value>

</data>

<data name="Forgot" xml:space="preserve">

<value>Do you forgot your password?</value>

</data>

<data name="Register" xml:space="preserve">

<value>New user</value>

</data>

<data name="EmailValidation" xml:space="preserve">

<value>You must enter your email.</value>

</data>

<data name="PasswordValidation" xml:space="preserve">

<value>You must enter your password.</value>

</data>

<data name="SomethingWrong" xml:space="preserve">

<value>The email or password are not valid.</value>

</data>

<data name="Menu" xml:space="preserve">

<value>Menu</value>

</data>

<data name="About" xml:space="preserve">

<value>About</value>

</data>

<data name="Setup" xml:space="preserve">

<value>Setup</value>

</data>

<data name="Exit" xml:space="preserve">

<value>Close session</value>

</data>

## Español

<data name="Search" xml:space="preserve">

<value>Buscar…</value>

</data>

<data name="Login" xml:space="preserve">

<value>Inicio de sesión</value>

</data>

<data name="EMail" xml:space="preserve">

<value>Correo electrónico</value>

</data>

<data name="EmailPlaceHolder" xml:space="preserve">

<value>Ingresa tu correo electrónico...</value>

</data>

<data name="Password" xml:space="preserve">

<value>Contraseña</value>

</data>

<data name="PasswordPlaceHolder" xml:space="preserve">

<value>Ingresa tu contraseña...</value>

</data>

<data name="Forgot" xml:space="preserve">

<value>¿Has olvidado la contraseña?</value>

</data>

<data name="Rememberme" xml:space="preserve">

<value>Recordar cuenta</value>

</data>

<data name="Register" xml:space="preserve">

<value>Nuevo usuario</value>

</data>

<data name="EmailValidation" xml:space="preserve">

<value>Debes introducir tu dirección de correo electrónico</value>

</data>

<data name="PasswordValidation" xml:space="preserve">

<value>Debes ingresar tu contraseña</value>

</data>

<data name="SomethingWrong" xml:space="preserve">

<value>El correo electrónico o la contraseña no son válidas.</value>

</data>

<data name="Menu" xml:space="preserve">

<value>Menú</value>

</data>

<data name="About" xml:space="preserve">

<value>Acerca de</value>

</data>

<data name="Setup" xml:space="preserve">

<value>Configuración</value>

</data>

<data name="Exit" xml:space="preserve">

<value>Cerrar sesión</value>

</data>

## Italiano

<data name="Search" xml:space="preserve">

<value>Cerca…</value>

</data>

<data name="Login" xml:space="preserve">

<value>Accesso</value>

</data>

<data name="EMail" xml:space="preserve">

<value>Posta elettronica</value>

</data>

<data name="EmailPlaceHolder" xml:space="preserve">

<value>Inserisci la tua email</value>

</data>

<data name="Password" xml:space="preserve">

<value>Password:</value>

</data>

<data name="PasswordPlaceHolder" xml:space="preserve">

<value>Immettere la password</value>

</data>

<data name="Forgot" xml:space="preserve">

<value>Hai dimenticato la password?</value>

</data>

<data name="Rememberme" xml:space="preserve">

<value>Memorizza account</value>

</data>

<data name="Register" xml:space="preserve">

<value>Nuovo utente</value>

</data>

<data name="EmailValidation" xml:space="preserve">

<value>Devi inserire la tua email</value>

</data>

<data name="PasswordValidation" xml:space="preserve">

<value>Devi inserire la tua password</value>

</data>

<data name="SomethingWrong" xml:space="preserve">

<value>L'e-mail o la password non sono validi.</value>

</data>

<data name="Menu" xml:space="preserve">

<value>Menu</value>

</data>

<data name="About" xml:space="preserve">

<value>Informazioni</value>

</data>

<data name="Setup" xml:space="preserve">

<value>Impostazione</value>

</data>

<data name="Exit" xml:space="preserve">

<value>Chiudi sessione</value>

</data>

## Portugués

<data name="Search" xml:space="preserve">

<value>Pesquisar...</value>

</data>

<data name="Login" xml:space="preserve">

<value>Logon:</value>

</data>

<data name="EMail" xml:space="preserve">

<value>Email</value>

</data>

<data name="EmailPlaceHolder" xml:space="preserve">

<value>Insira o seu e-mail...</value>

</data>

<data name="Password" xml:space="preserve">

<value>Senha</value>

</data>

<data name="PasswordPlaceHolder" xml:space="preserve">

<value>Digite sua senha</value>

</data>

<data name="Forgot" xml:space="preserve">

<value>Esqueceu-se da palavra-passe?</value>

</data>

<data name="Rememberme" xml:space="preserve">

<value>Lembrar-me</value>

</data>

<data name="Register" xml:space="preserve">

<value>Novo utilizador</value>

</data>

<data name="EmailValidation" xml:space="preserve">

<value>Informe seu email</value>

</data>

<data name="PasswordValidation" xml:space="preserve">

<value>Precisa de inserir a sua palavra-passe.</value>

</data>

<data name="SomethingWrong" xml:space="preserve">

<value>O e-mail ou senha não são válidos.</value>

</data>

<data name="Menu" xml:space="preserve">

<value>Menu</value>

</data>

<data name="About" xml:space="preserve">

<value>Sobre</value>

</data>

<data name="Setup" xml:space="preserve">

<value>Configuração</value>

</data>

<data name="Exit" xml:space="preserve">

<value>Fechar sessão</value>

</data>

## Languages

public static string Search

{

get { return Resource.Search; }

}

public static string Login

{

get { return Resource.Login; }

}

public static string EMail

{

get { return Resource.EMail; }

}

public static string EmailPlaceHolder

{

get { return Resource.EmailPlaceHolder; }

}

public static string Password

{

get { return Resource.Password; }

}

public static string PasswordPlaceHolder

{

get { return Resource.PasswordPlaceHolder; }

}

public static string Rememberme

{

get { return Resource.Rememberme; }

}

public static string Forgot

{

get { return Resource.Forgot; }

}

public static string Register

{

get { return Resource.Register; }

}

public static string EmailValidation

{

get { return Resource.EmailValidation; }

}

public static string PasswordValidation

{

get { return Resource.PasswordValidation; }

}

public static string SomethingWrong

{

get { return Resource.SomethingWrong; }

}

public static string Menu

{

get { return Resource.Menu; }

}

public static string Setup

{

get { return Resource.Setup; }

}

public static string About

{

get { return Resource.About; }

}

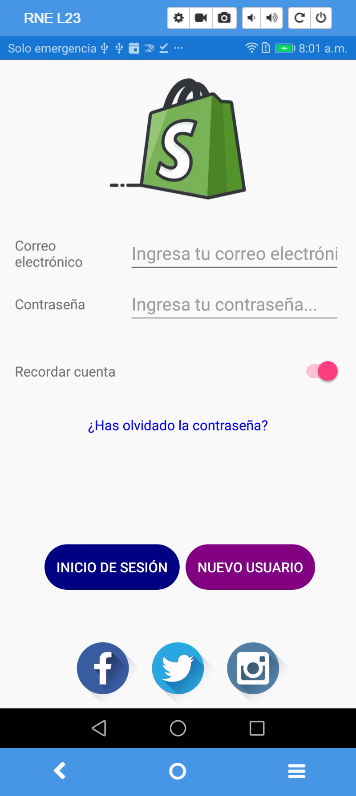
public static string Exit

{

get { return Resource.Exit; }

}

# Pantalla login



La cual tiene este código XAML:

<?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:i18n="clr-namespace:Sales.Helpers"

x:Class="Sales.Views.LoginPage"

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

Title="{i18n:Translate Login}">

<ContentPage.Content>

<ScrollView

BindingContext="{Binding Login}">

<StackLayout

Padding="5">

<Image

HeightRequest="150"

Source="sales">

</Image>

<Grid

Margin="10,0"

VerticalOptions="CenterAndExpand">

<Grid.ColumnDefinitions>

<ColumnDefinition Width="\*"/>

<ColumnDefinition Width="2\*"/>

</Grid.ColumnDefinitions>

<Label

Grid.Row="0"

Grid.Column="0"

Text="{i18n:Translate EMail}"

VerticalOptions="Center">

</Label>

<Entry

Grid.Row="0"

Grid.Column="1"

Keyboard="Email"

Text="{Binding Email}"

Placeholder="{i18n:Translate EmailPlaceHolder}">

</Entry>

<Label

Grid.Row="1"

Grid.Column="0"

Text="{i18n:Translate Password}"

VerticalOptions="Center">

</Label>

<Entry

Grid.Row="1"

Grid.Column="1"

IsPassword="true"

Text="{Binding Password}"

Placeholder="{i18n:Translate PasswordPlaceHolder}">

</Entry>

</Grid>

<StackLayout

Margin="10,0"

Orientation="Horizontal"

VerticalOptions="CenterAndExpand">

<Label

HorizontalOptions="StartAndExpand"

Text="{i18n:Translate Rememberme}"

VerticalOptions="Center">

</Label>

<Switch

IsToggled="{Binding IsRemembered}"

HorizontalOptions="End">

</Switch>

</StackLayout>

<Label

HorizontalOptions="Center"

Text="{i18n:Translate Forgot}"

TextColor="Blue"

VerticalOptions="CenterAndExpand">

<Label.GestureRecognizers>

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

</Label.GestureRecognizers>

</Label>

<ActivityIndicator

IsRunning="{Binding IsRunning}"

VerticalOptions="CenterAndExpand">

</ActivityIndicator>

<StackLayout

Margin="40,0"

Orientation="Horizontal"

VerticalOptions="CenterAndExpand">

<Button

BackgroundColor="Navy"

BorderRadius="23"

Command="{Binding LoginCommand}"

HeightRequest="46"

HorizontalOptions="FillAndExpand"

IsEnabled="{Binding IsEnabled}"

Text="{i18n:Translate Login}"

TextColor="White">

</Button>

<Button

BackgroundColor="Purple"

BorderRadius="23"

Command="{Binding RegisterCommand}"

HeightRequest="46"

HorizontalOptions="FillAndExpand"

IsEnabled="{Binding IsEnabled}"

Text="{i18n:Translate Register}"

TextColor="White">

</Button>

</StackLayout>

<StackLayout

HorizontalOptions="Center"

Orientation="Horizontal"

VerticalOptions="EndAndExpand">

<Image

Aspect="AspectFit"

HeightRequest="70"

Source="facebook\_logo"

WidthRequest="70">

<Image.GestureRecognizers>

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

</Image.GestureRecognizers>

</Image>

<Image

Aspect="AspectFit"

HeightRequest="70"

Source="twitter\_logo"

WidthRequest="70">

<Image.GestureRecognizers>

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

</Image.GestureRecognizers>

</Image>

<Image

Aspect="AspectFit"

HeightRequest="70"

Source="instagram\_logo"

WidthRequest="70">

<Image.GestureRecognizers>

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

</Image.GestureRecognizers>

</Image>

</StackLayout>

</StackLayout>

</ScrollView>

</ContentPage.Content>

</ContentPage>

# Obtener un token

public async Task<TokenResponse> GetToken(

string urlBase,

string username,

string password)

{

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;

}

}

namespace Sales.Common.Models

{

using System;

using Newtonsoft.Json;

public class TokenResponse

{

[JsonProperty(PropertyName = "access\_token")]

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; }

[JsonProperty(PropertyName = "error\_description")]

public string ErrorDescription { get; set; }

}

}

# Master Detail

## XAML

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

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

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

xmlns:pages="clr-namespace:Sales.Views"

x:Class="Sales.Views.MasterPage">

<MasterDetailPage.Master>

<pages:MenuPage/>

</MasterDetailPage.Master>

<MasterDetailPage.Detail>

<NavigationPage x:Name="Navigator">

<x:Arguments>

<pages:ProductsPage/>

</x:Arguments>

</NavigationPage>

</MasterDetailPage.Detail>

</MasterDetailPage>

## C#

namespace Sales.Views

{

using Xamarin.Forms;

using Xamarin.Forms.Xaml;

[XamlCompilation(XamlCompilationOptions.Compile)]

public partial class MasterPage : MasterDetailPage

{

public MasterPage ()

{

InitializeComponent ();

}

protected override void OnAppearing()

{

base.OnAppearing();

App.Navigator = Navigator;

}

}

}

## MenuPage

<?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:i18n="clr-namespace:Sales.Helpers"

x:Class="Sales.Views.MenuPage"

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

Title="{i18n:Translate Menu}"

BackgroundColor="Gray">

<ContentPage.Content>

<StackLayout

Padding="5">

<Image

Source="sales">

</Image>

<ListView

ItemsSource="{Binding Menu}"

HasUnevenRows="True">

<ListView.ItemTemplate>

<DataTemplate>

<ViewCell>

<Grid>

<Grid.GestureRecognizers>

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

</Grid.GestureRecognizers>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="Auto"/>

<ColumnDefinition Width="\*"/>

</Grid.ColumnDefinitions>

<Image

Grid.Column="0"

Source="{Binding Icon}"

WidthRequest="50"

HeightRequest="50">

</Image>

<Label

Grid.Column="1"

VerticalOptions="Center"

TextColor="White"

Text="{Binding Title}">

</Label>

</Grid>

</ViewCell>

</DataTemplate>

</ListView.ItemTemplate>

</ListView>

</StackLayout>

</ContentPage.Content>

</ContentPage>

## MenuItemViewModel

public string Icon { get; set; }

public string Title { get; set; }

public string PageName { get; set; }

## Load menu in MainViewModel

### Crear propiedad:

public ObservableCollection<MenuItemViewModel> Menu { get; set; }

### Crear método:

private void LoadMenu()

{

this.Menu = new ObservableCollection<MenuItemViewModel>();

this.Menu.Add(new MenuItemViewModel

{

Icon = "ic\_info",

PageName = "AboutPage",

Title = Languages.About,

});

this.Menu.Add(new MenuItemViewModel

{

Icon = "ic\_phonelink\_setup",

PageName = "SetupPage",

Title = Languages.Setup,

});

this.Menu.Add(new MenuItemViewModel

{

Icon = "ic\_exit\_to\_app",

PageName = "LoginPage",

Title = Languages.Exit,

});

}

# Consumir servicio de forma segura

client.DefaultRequestHeaders.Authorization = new AuthenticationHeaderValue(tokenType, accessToken);

# Nuevos Literales

## En Inglés

<data name="NoProductsMessage" xml:space="preserve">

<value>No products information available.</value>

</data>

<data name="FirstName" xml:space="preserve">

<value>First name</value>

</data>

<data name="LastName" xml:space="preserve">

<value>Last name</value>

</data>

<data name="Phone" xml:space="preserve">

<value>Phone</value>

</data>

<data name="Address" xml:space="preserve">

<value>Address</value>

</data>

<data name="FirstNamePlaceholder" xml:space="preserve">

<value>Enter your first name...</value>

</data>

<data name="LastNamePlaceholder" xml:space="preserve">

<value>Enter your last name...</value>

</data>

<data name="PhonePlaceHolder" xml:space="preserve">

<value>Enter your phone number...</value>

</data>

<data name="AddressPlaceHolder" xml:space="preserve">

<value>Enter your address...</value>

</data>

<data name="PasswordConfirm" xml:space="preserve">

<value>Confirm</value>

</data>

<data name="PasswordConfirmPlaceHolder" xml:space="preserve">

<value>Enter your password confirm...</value>

</data>

<data name="FirstNameError" xml:space="preserve">

<value>You must enter a fist name.</value>

</data>

<data name="LastNameError" xml:space="preserve">

<value>You must enter a last name.</value>

</data>

<data name="EMailError" xml:space="preserve">

<value>You must enter a valid email.</value>

</data>

<data name="PhoneError" xml:space="preserve">

<value>You must enter a phone number.</value>

</data>

<data name="PasswordError" xml:space="preserve">

<value>You must enter a password. At least 6 characters.</value>

</data>

<data name="PasswordConfirmError" xml:space="preserve">

<value>You must enter a password confirm.</value>

</data>

<data name="PasswordsNoMatch" xml:space="preserve">

<value>The password and confirm must be equals.</value>

</data>

<data name="RegisterConfirmation" xml:space="preserve">

<value>The user was created successfully. Now you can enter with the email and the assigned password.</value>

</data>

## En Español

<data name="NoProductsMessage" xml:space="preserve">

<value>No hay información de productos disponible.</value>

</data>

<data name="FirstName" xml:space="preserve">

<value>Nombre</value>

</data>

<data name="LastName" xml:space="preserve">

<value>Apellidos</value>

</data>

<data name="Phone" xml:space="preserve">

<value>Teléfono</value>

</data>

<data name="Address" xml:space="preserve">

<value>Dirección</value>

</data>

<data name="FirstNamePlaceholder" xml:space="preserve">

<value>Escribe tu nombre...</value>

</data>

<data name="LastNamePlaceholder" xml:space="preserve">

<value>Escribe tu apellido...</value>

</data>

<data name="PhonePlaceHolder" xml:space="preserve">

<value>Escribir el número de teléfono...</value>

</data>

<data name="AddressPlaceHolder" xml:space="preserve">

<value>Introduzca su dirección</value>

</data>

<data name="PasswordConfirm" xml:space="preserve">

<value>Confirmación</value>

</data>

<data name="PasswordConfirmPlaceHolder" xml:space="preserve">

<value>Introduzca su confirmación de contraseña...</value>

</data>

<data name="FirstNameError" xml:space="preserve">

<value>Debe introducir un nombre.</value>

</data>

<data name="LastNameError" xml:space="preserve">

<value>Debe escribir apellidos</value>

</data>

<data name="EMailError" xml:space="preserve">

<value>Hay que introducir una dirección de correo válida.</value>

</data>

<data name="PhoneError" xml:space="preserve">

<value>Debe introducir un número de teléfono.</value>

</data>

<data name="PasswordConfirmError" xml:space="preserve">

<value>Debes ingresar una contraseña.</value>

</data>

<data name="PasswordsNoMatch" xml:space="preserve">

<value>La contraseña y la confirmación deben ser iguales.</value>

</data>

<data name="PasswordError" xml:space="preserve">

<value>Debe introducir una contraseña. Por lo menos 6 caracteres.</value>

</data>

<data name="RegisterConfirmation" xml:space="preserve">

<value>El usuario se ha creado correctamente. Ahora puede ingresar con el correo electrónico y la contraseña asignada.</value>

</data>

## En Italiano

<data name="NoProductsMessage" xml:space="preserve">

<value>Nessuna informazione sui prodotti disponibile.</value>

</data>

<data name="FirstName" xml:space="preserve">

<value>Nome</value>

</data>

<data name="LastName" xml:space="preserve">

<value>Cognome</value>

</data>

<data name="Phone" xml:space="preserve">

<value>Telefono</value>

</data>

<data name="Address" xml:space="preserve">

<value>Indirizzo</value>

</data>

<data name="FirstNamePlaceholder" xml:space="preserve">

<value>Immetti il tuo nome...</value>

</data>

<data name="LastNamePlaceholder" xml:space="preserve">

<value>Immetti il tuo cognome....</value>

</data>

<data name="PhonePlaceHolder" xml:space="preserve">

<value>Immetti il numero di telefono...</value>

</data>

<data name="AddressPlaceHolder" xml:space="preserve">

<value>Inserisci il tuo indirizzo...</value>

</data>

<data name="PasswordConfirm" xml:space="preserve">

<value>Conferma</value>

</data>

<data name="PasswordConfirmPlaceHolder" xml:space="preserve">

<value>Inserisci la password</value>

</data>

<data name="FirstNameError" xml:space="preserve">

<value>È necessario immettere un nome di pugno.</value>

</data>

<data name="LastNameError" xml:space="preserve">

<value>Immettere un cognome</value>

</data>

<data name="EMailError" xml:space="preserve">

<value>Si deve inserire un indirizzo email valido.</value>

</data>

<data name="PhoneError" xml:space="preserve">

<value>È necessario immettere un numero di telefono.</value>

</data>

<data name="PasswordConfirmError" xml:space="preserve">

<value>Devi inserire una password per confermare.</value>

</data>

<data name="PasswordsNoMatch" xml:space="preserve">

<value>La password e conferma devono essere uguali.</value>

</data>

<data name="PasswordError" xml:space="preserve">

<value>È necessario immettere una password. Almeno 6 caratteri.</value>

</data>

<data name="RegisterConfirmation" xml:space="preserve">

<value>L'utente è stato creato correttamente. Ora è possibile immettere con l'email e la password assegnata.</value>

</data>

## En Portugués

<data name="NoProductsMessage" xml:space="preserve">

<value>Nenhuma informação de produtos disponível.</value>

</data>

<data name="FirstName" xml:space="preserve">

<value>Nome</value>

</data>

<data name="LastName" xml:space="preserve">

<value>Sobrenome</value>

</data>

<data name="Phone" xml:space="preserve">

<value>Telefone</value>

</data>

<data name="Address" xml:space="preserve">

<value>Endereço</value>

</data>

<data name="FirstNamePlaceholder" xml:space="preserve">

<value>Insira seu nome...</value>

</data>

<data name="LastNamePlaceholder" xml:space="preserve">

<value>Insira seu sobrenome...</value>

</data>

<data name="PhonePlaceHolder" xml:space="preserve">

<value>Insira o seu número de telefone...</value>

</data>

<data name="AddressPlaceHolder" xml:space="preserve">

<value>Insira o seu endereço...</value>

</data>

<data name="PasswordConfirm" xml:space="preserve">

<value>Confirmar</value>

</data>

<data name="PasswordConfirmPlaceHolder" xml:space="preserve">

<value>Digite sua senha para confirmar...</value>

</data>

<data name="FirstNameError" xml:space="preserve">

<value>Você deve inserir um nome.</value>

</data>

<data name="LastNameError" xml:space="preserve">

<value>Digite um sobrenome.</value>

</data>

<data name="EMailError" xml:space="preserve">

<value>Você deve inserir um endereço de email válido.</value>

</data>

<data name="PhoneError" xml:space="preserve">

<value>Você deve digitar um número de telefone.</value>

</data>

<data name="PasswordConfirmError" xml:space="preserve">

<value>Tem de introduzir uma palavra-passe</value>

</data>

<data name="PasswordsNoMatch" xml:space="preserve">

<value>A senha e confirme a deve ser igual a.</value>

</data>

<data name="PasswordError" xml:space="preserve">

<value>Informe uma nova senha com no mínimo 6 caracteres.</value>

</data>

<data name="RegisterConfirmation" xml:space="preserve">

<value>O usuário foi criado com êxito. Agora você pode entrar com o e-mail e a senha atribuída.</value>

</data>

## En Languages

public static string NoProductsMessage

{

get { return Resource.NoProductsMessage; }

}

public static string FirstName

{

get { return Resource.FirstName; }

}

public static string FirstNamePlaceholder

{

get { return Resource.FirstNamePlaceholder; }

}

public static string LastName

{

get { return Resource.LastName; }

}

public static string LastNamePlaceholder

{

get { return Resource.LastNamePlaceholder; }

}

public static string Phone

{

get { return Resource.Phone; }

}

public static string PhonePlaceHolder

{

get { return Resource.PhonePlaceHolder; }

}

public static string PasswordConfirm

{

get { return Resource.PasswordConfirm; }

}

public static string PasswordConfirmPlaceHolder

{

get { return Resource.PasswordConfirmPlaceHolder; }

}

public static string Address

{

get { return Resource.Address; }

}

public static string AddressPlaceHolder

{

get { return Resource.AddressPlaceHolder; }

}

public static string FirstNameError

{

get { return Resource.FirstNameError; }

}

public static string LastNameError

{

get { return Resource.LastNameError; }

}

public static string EMailError

{

get { return Resource.EMailError; }

}

public static string PhoneError

{

get { return Resource.PhoneError; }

}

public static string PasswordError

{

get { return Resource.PasswordError; }

}

public static string PasswordConfirmError

{

get { return Resource.PasswordConfirmError; }

}

public static string PasswordsNoMatch

{

get { return Resource.PasswordsNoMatch; }

}

public static string RegisterConfirmation

{

get { return Resource.RegisterConfirmation; }

}

# SQLite

1. Agrega el Nuget: **sql-net-pcl** a todos los proyectos del front.
2. Crear la interfaz **IPathService**:

public interface IPathService

{

string GetDatabasePath();

}

1. Implementarla en Android y iOS:

[assembly: Xamarin.Forms.Dependency(typeof(Sales.Droid.Implementations.PathService))]

namespace Sales.Droid.Implementations

{

using Interfaces;

using System;

using System.IO;

public class PathService : IPathService

{

public string GetDatabasePath()

{

string path = Environment.GetFolderPath(Environment.SpecialFolder.Personal);

return Path.Combine(path, "Sales.db3");

}

}

}

[assembly: Xamarin.Forms.Dependency(typeof(Sales.iOS.Implementations.PathService))]

namespace Sales.iOS.Implementations

{

using Interfaces;

using System;

using System.IO;

public class PathService : IPathService

{

public string GetDatabasePath()

{

string docFolder = Environment.GetFolderPath(Environment.SpecialFolder.Personal);

string libFolder = Path.Combine(docFolder, "..", "Library", "Databases");

if (!Directory.Exists(libFolder))

{

Directory.CreateDirectory(libFolder);

}

return Path.Combine(libFolder, "Sales.db3");

}

}

}

1. Ahora creamos el **DataService** para nuestras operaciones con datos:

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using Common.Models;

using Interfaces;

using SQLite;

using Xamarin.Forms;

public class DataService

{

private SQLiteAsyncConnection connection;

public DataService()

{

this.OpenOrCreateDB();

}

private async Task OpenOrCreateDB()

{

var databasePath = DependencyService.Get<IPathService>().GetDatabasePath();

this.connection = new SQLiteAsyncConnection(databasePath);

await connection.CreateTableAsync<Product>().ConfigureAwait(false);

}

public async Task Insert<T>(T model)

{

await this.connection.InsertAsync(model);

}

public async Task Insert<T>(List<T> models)

{

await this.connection.InsertAllAsync(models);

}

public async Task Update<T>(T model)

{

await this.connection.UpdateAsync(model);

}

public async Task Update<T>(List<T> models)

{

await this.connection.UpdateAllAsync(models);

}

public async Task Delete<T>(T model)

{

await this.connection.DeleteAsync(model);

}

public async Task<List<Product>> GetAllProducts()

{

var query = await this.connection.QueryAsync<Product>("select \* from [Product]");

var array = query.ToArray();

var list = array.Select(p => new Product

{

Description = p.Description,

ImagePath = p.ImagePath,

IsAvailable = p.IsAvailable,

Price = p.Price,

ProductId = p.ProductId,

PublishOn = p.PublishOn,

Remarks = p.Remarks,

}).ToList();

return list;

}

public async Task DeleteAllProducts()

{

var query = await this.connection.QueryAsync<Product>("delete from [Product]");

}

}

1. Para mayor información ver: <https://www.codeproject.com/Articles/1097179/SQLite-with-Xamarin-Forms-Step-by-Step-guide>

# 

# Registrar Nuevos Usuarios

1. Crear la pantalla de registro:



Con su respectivo XAML:

<?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:i18n="clr-namespace:Sales.Helpers"

x:Class="Sales.Views.RegisterPage"

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

Title="{i18n:Translate Register}">

<ContentPage.Content>

<ScrollView

BindingContext="{Binding Register}">

<StackLayout

Padding="5">

<Image

Source="{Binding ImageSource}">

<Image.GestureRecognizers>

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

</Image.GestureRecognizers>

</Image>

<Label

FontSize="Small"

HorizontalOptions="Center"

Text="{i18n:Translate ChangeImage}">

</Label>

<Grid>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="Auto"/>

<ColumnDefinition Width="\*"/>

</Grid.ColumnDefinitions>

<Label

Grid.Column="0"

Grid.Row="0"

Text="{i18n:Translate FirstName}"

VerticalOptions="Center">

</Label>

<Entry

Grid.Column="1"

Grid.Row="0"

Text="{Binding FirstName}"

Placeholder="{i18n:Translate FirstNamePlaceholder}">

</Entry>

<Label

Grid.Column="0"

Grid.Row="1"

Text="{i18n:Translate LastName}"

VerticalOptions="Center">

</Label>

<Entry

Grid.Column="1"

Grid.Row="1"

Text="{Binding LastName}"

Placeholder="{i18n:Translate LastNamePlaceholder}">

</Entry>

<Label

Grid.Column="0"

Grid.Row="2"

Text="{i18n:Translate EMail}"

VerticalOptions="Center">

</Label>

<Entry

Grid.Column="1"

Grid.Row="2"

Keyboard="Email"

Text="{Binding EMail}"

Placeholder="{i18n:Translate EmailPlaceHolder}">

</Entry>

<Label

Grid.Column="0"

Grid.Row="3"

Text="{i18n:Translate Phone}"

VerticalOptions="Center">

</Label>

<Entry

Grid.Column="1"

Grid.Row="3"

Keyboard="Telephone"

Text="{Binding Phone}"

Placeholder="{i18n:Translate PhonePlaceHolder}">

</Entry>

<Label

Grid.Column="0"

Grid.Row="4"

Text="{i18n:Translate Address}"

VerticalOptions="Center">

</Label>

<Entry

Grid.Column="1"

Grid.Row="4"

Text="{Binding Address}"

Placeholder="{i18n:Translate AddressPlaceHolder}">

</Entry>

<Label

Grid.Column="0"

Grid.Row="5"

Text="{i18n:Translate Password}"

VerticalOptions="Center">

</Label>

<Entry

Grid.Column="1"

Grid.Row="5"

IsPassword="True"

Text="{Binding Password}"

Placeholder="{i18n:Translate PasswordPlaceHolder}">

</Entry>

<Label

Grid.Column="0"

Grid.Row="6"

Text="{i18n:Translate PasswordConfirm}"

VerticalOptions="Center">

</Label>

<Entry

Grid.Column="1"

Grid.Row="6"

IsPassword="True"

Text="{Binding PasswordConfirm}"

Placeholder="{i18n:Translate PasswordConfirmPlaceHolder}">

</Entry>

</Grid>

<ActivityIndicator

IsRunning="{Binding IsRunning}"

VerticalOptions="CenterAndExpand">

</ActivityIndicator>

<Button

BackgroundColor="Navy"

BorderRadius="23"

Command="{Binding SaveCommand}"

HeightRequest="46"

IsEnabled="{Binding IsEnabled}"

Text="{i18n:Translate Save}"

TextColor="White">

</Button>

</StackLayout>

</ScrollView>

</ContentPage.Content>

</ContentPage>

1. Adicionamos la imagen de “no user”.
2. Creamos el nuevo controlador en el diccionario de recursos:

<x:String x:Key="UrlUsersController">/Users</x:String>

1. Creamos el modelo **UserRequest**:

public class UserRequest

{

public string FirstName { get; set; }

public string LastName { get; set; }

public string EMail { get; set; }

public string Phone { get; set; }

public string Address { get; set; }

public string Password { get; set; }

public string ImagePath { get; set; }

public byte[] ImageArray { get; set; }

}

1. Creamos el helper **RegexHelper**:

using System;

using System.Net.Mail;

public static class RegexHelper

{

public static bool IsValidEmailAddress(string emailaddress)

{

try

{

var email = new MailAddress(emailaddress);

return true;

}

catch (FormatException)

{

return false;

}

}

}

1. Creamos la ViewModel correspondiente:

namespace Sales.ViewModels

{

using System.Windows.Input;

using GalaSoft.MvvmLight.Command;

using Helpers;

using Plugin.Media;

using Plugin.Media.Abstractions;

using Sales.Common.Models;

using Services;

using Xamarin.Forms;

public class RegisterViewModel : BaseViewModel

{

#region Attributes

private MediaFile file;

private ImageSource imageSource;

private ApiService apiService;

private bool isRunning;

private bool isEnabled;

#endregion

#region Properties

public string FirstName { get; set; }

public string LastName { get; set; }

public string EMail { get; set; }

public string Phone { get; set; }

public string Address { get; set; }

public string Password { get; set; }

public string PasswordConfirm { get; set; }

public bool IsRunning

{

get { return this.isRunning; }

set { this.SetValue(ref this.isRunning, value); }

}

public bool IsEnabled

{

get { return this.isEnabled; }

set { this.SetValue(ref this.isEnabled, value); }

}

public ImageSource ImageSource

{

get { return this.imageSource; }

set { this.SetValue(ref this.imageSource, value); }

}

#endregion

#region Constructors

public RegisterViewModel()

{

this.apiService = new ApiService();

this.IsEnabled = true;

this.ImageSource = "nouser";

}

#endregion

#region Commands

public ICommand ChangeImageCommand

{

get

{

return new RelayCommand(ChangeImage);

}

}

private async void ChangeImage()

{

await CrossMedia.Current.Initialize();

var source = await Application.Current.MainPage.DisplayActionSheet(

Languages.ImageSource,

Languages.Cancel,

null,

Languages.FromGallery,

Languages.NewPicture);

if (source == Languages.Cancel)

{

this.file = null;

return;

}

if (source == Languages.NewPicture)

{

this.file = await CrossMedia.Current.TakePhotoAsync(

new StoreCameraMediaOptions

{

Directory = "Sample",

Name = "test.jpg",

PhotoSize = PhotoSize.Small,

}

);

}

else

{

this.file = await CrossMedia.Current.PickPhotoAsync();

}

if (this.file != null)

{

this.ImageSource = ImageSource.FromStream(() =>

{

var stream = this.file.GetStream();

return stream;

});

}

}

public ICommand SaveCommand

{

get

{

return new RelayCommand(Save);

}

}

private async void Save()

{

if (string.IsNullOrEmpty(this.FirstName))

{

await Application.Current.MainPage.DisplayAlert(

Languages.Error,

Languages.FirstNameError,

Languages.Accept);

return;

}

if (string.IsNullOrEmpty(this.LastName))

{

await Application.Current.MainPage.DisplayAlert(

Languages.Error,

Languages.LastNameError,

Languages.Accept);

return;

}

if (string.IsNullOrEmpty(this.EMail))

{

await Application.Current.MainPage.DisplayAlert(

Languages.Error,

Languages.EMailError,

Languages.Accept);

return;

}

if (!RegexHelper.IsValidEmailAddress(this.EMail))

{

await Application.Current.MainPage.DisplayAlert(

Languages.Error,

Languages.EMailError,

Languages.Accept);

return;

}

if (string.IsNullOrEmpty(this.Phone))

{

await Application.Current.MainPage.DisplayAlert(

Languages.Error,

Languages.PhoneError,

Languages.Accept);

return;

}

if (string.IsNullOrEmpty(this.Password))

{

await Application.Current.MainPage.DisplayAlert(

Languages.Error,

Languages.PasswordError,

Languages.Accept);

return;

}

if (this.Password.Length < 6)

{

await Application.Current.MainPage.DisplayAlert(

Languages.Error,

Languages.PasswordError,

Languages.Accept);

return;

}

if (string.IsNullOrEmpty(this.PasswordConfirm))

{

await Application.Current.MainPage.DisplayAlert(

Languages.Error,

Languages.PasswordConfirmError,

Languages.Accept);

return;

}

if (!this.Password.Equals(this.PasswordConfirm))

{

await Application.Current.MainPage.DisplayAlert(

Languages.Error,

Languages.PasswordsNoMatch,

Languages.Accept);

return;

}

this.IsRunning = true;

this.IsEnabled = false;

var connection = await this.apiService.CheckConnection();

if (!connection.IsSuccess)

{

this.IsRunning = false;

this.IsEnabled = true;

await Application.Current.MainPage.DisplayAlert(

Languages.Error,

connection.Message,

Languages.Accept);

return;

}

byte[] imageArray = null;

if (this.file != null)

{

imageArray = FilesHelper.ReadFully(this.file.GetStream());

}

var userRequest = new UserRequest

{

Address = this.Address,

EMail = this.EMail,

FirstName = this.FirstName,

ImageArray = imageArray,

LastName = this.LastName,

Password = this.Password,

};

var url = Application.Current.Resources["UrlAPI"].ToString();

var prefix = Application.Current.Resources["UrlPrefix"].ToString();

var controller = Application.Current.Resources["UrlUsersController"].ToString();

var response = await this.apiService.Post(url, prefix, controller, userRequest);

if (!response.IsSuccess)

{

this.IsRunning = false;

this.IsEnabled = true;

await Application.Current.MainPage.DisplayAlert(

Languages.Error,

response.Message,

Languages.Accept);

return;

}

this.IsRunning = false;

this.IsEnabled = true;

await Application.Current.MainPage.DisplayAlert(

Languages.Confirm,

Languages.RegisterConfirmation,

Languages.Accept);

await Application.Current.MainPage.Navigation.PopAsync();

}

#endregion

}

}

1. En el API creamos los siguientes valores en parámetros:

<appSettings>

<add key="AdminUser" value="jzuluaga55@hotmail.com" />

<add key="AdminPassWord" value="\*\*\*\*" />

<add key="SMTPName" value="smtp.live.com" />

<add key="SMTPPort" value="25" />

</appSettings>

1. Creamos helper **MailHelper** en el **API**:

namespace Sales.API.Helpers

{

using System.Collections.Generic;

using System.Net;

using System.Net.Mail;

using System.Threading.Tasks;

using System.Web.Configuration;

public class MailHelper

{

public static async Task SendMail(string to, string subject, string body)

{

var message = new MailMessage();

message.To.Add(new MailAddress(to));

message.From = new MailAddress(WebConfigurationManager.AppSettings["AdminUser"]);

message.Subject = subject;

message.Body = body;

message.IsBodyHtml = true;

using (var smtp = new SmtpClient())

{

var credential = new NetworkCredential

{

UserName = WebConfigurationManager.AppSettings["AdminUser"],

Password = WebConfigurationManager.AppSettings["AdminPassWord"]

};

smtp.Credentials = credential;

smtp.Host = WebConfigurationManager.AppSettings["SMTPName"];

smtp.Port = int.Parse(WebConfigurationManager.AppSettings["SMTPPort"]);

smtp.EnableSsl = true;

await smtp.SendMailAsync(message);

}

}

public static async Task SendMail(List<string> mails, string subject, string body)

{

var message = new MailMessage();

foreach (var to in mails)

{

message.To.Add(new MailAddress(to));

}

message.From = new MailAddress(WebConfigurationManager.AppSettings["AdminUser"]);

message.Subject = subject;

message.Body = body;

message.IsBodyHtml = true;

using (var smtp = new SmtpClient())

{

var credential = new NetworkCredential

{

UserName = WebConfigurationManager.AppSettings["AdminUser"],

Password = WebConfigurationManager.AppSettings["AdminPassWord"]

};

smtp.Credentials = credential;

smtp.Host = WebConfigurationManager.AppSettings["SMTPName"];

smtp.Port = int.Parse(WebConfigurationManager.AppSettings["SMTPPort"]);

smtp.EnableSsl = true;

await smtp.SendMailAsync(message);

}

}

}

}

1. Creamos helper **UsersHelper** en el **API**:

namespace Sales.API.Helpers

{

using System;

using System.Security.Claims;

using System.Threading.Tasks;

using System.Web.Configuration;

using Common.Models;

using Domain.Models;

using Microsoft.AspNet.Identity;

using Microsoft.AspNet.Identity.EntityFramework;

using Models;

public class UsersHelper : IDisposable

{

private static ApplicationDbContext userContext = new ApplicationDbContext();

private static DataContext db = new DataContext();

public static bool DeleteUser(string userName, string roleName)

{

var userManager = new UserManager<ApplicationUser>(new UserStore<ApplicationUser>(userContext));

var userASP = userManager.FindByEmail(userName);

if (userASP == null)

{

return false;

}

var response = userManager.RemoveFromRole(userASP.Id, roleName);

return response.Succeeded;

}

public static ApplicationUser GetUserASP(string email)

{

var userManager = new UserManager<ApplicationUser>(new UserStore<ApplicationUser>(userContext));

var userASP = userManager.FindByEmail(email);

return userASP;

}

public static Response CreateUserASP(UserRequest userRequest)

{

try

{

var userManager = new UserManager<ApplicationUser>(new UserStore<ApplicationUser>(userContext));

var oldUserASP = userManager.FindByEmail(userRequest.EMail);

if (oldUserASP != null)

{

return new Response

{

IsSuccess = false,

Message = "001. User already exists.",

};

}

var userASP = new ApplicationUser

{

Email = userRequest.EMail,

UserName = userRequest.EMail,

PhoneNumber = userRequest.Phone,

};

var result = userManager.Create(userASP, userRequest.Password);

if (result.Succeeded)

{

var newUserASP = userManager.FindByEmail(userRequest.EMail);

userManager.AddClaim(newUserASP.Id, new Claim(ClaimTypes.GivenName, userRequest.FirstName));

userManager.AddClaim(newUserASP.Id, new Claim(ClaimTypes.Name, userRequest.LastName));

if (!string.IsNullOrEmpty(userRequest.Address))

{

userManager.AddClaim(newUserASP.Id, new Claim(ClaimTypes.StreetAddress, userRequest.Address));

}

if (!string.IsNullOrEmpty(userRequest.ImagePath))

{

userManager.AddClaim(newUserASP.Id, new Claim(ClaimTypes.Uri, userRequest.ImagePath));

}

return new Response

{

IsSuccess = true,

};

}

var errors = string.Empty;

foreach (var error in result.Errors)

{

errors += $"{error}, ";

}

return new Response

{

IsSuccess = false,

Message = errors,

};

}

catch (Exception ex)

{

return new Response

{

IsSuccess = false,

Message = ex.Message,

};

}

}

public static bool UpdateUserName(string currentUserName, string newUserName)

{

var userManager = new UserManager<ApplicationUser>(new UserStore<ApplicationUser>(userContext));

var userASP = userManager.FindByEmail(currentUserName);

if (userASP == null)

{

return false;

}

userASP.UserName = newUserName;

userASP.Email = newUserName;

var response = userManager.Update(userASP);

return response.Succeeded;

}

public static void CheckRole(string roleName)

{

var roleManager = new RoleManager<IdentityRole>(new RoleStore<IdentityRole>(userContext));

// Check to see if Role Exists, if not create it

if (!roleManager.RoleExists(roleName))

{

roleManager.Create(new IdentityRole(roleName));

}

}

public static void CheckSuperUser()

{

var userManager = new UserManager<ApplicationUser>(new UserStore<ApplicationUser>(userContext));

var email = WebConfigurationManager.AppSettings["AdminUser"];

var password = WebConfigurationManager.AppSettings["AdminPassWord"];

var userASP = userManager.FindByName(email);

if (userASP == null)

{

CreateUserASP(email, "Admin", password);

return;

}

}

public static void CreateUserASP(string email, string roleName)

{

var userManager = new UserManager<ApplicationUser>(new UserStore<ApplicationUser>(userContext));

var userASP = userManager.FindByEmail(email);

if (userASP == null)

{

userASP = new ApplicationUser

{

Email = email,

UserName = email,

};

userManager.Create(userASP, email);

}

userManager.AddToRole(userASP.Id, roleName);

}

public static void CreateUserASP(string email, string roleName, string password)

{

var userManager = new UserManager<ApplicationUser>(new UserStore<ApplicationUser>(userContext));

var userASP = new ApplicationUser

{

Email = email,

UserName = email,

};

var result = userManager.Create(userASP, password);

if (result.Succeeded)

{

userManager.AddToRole(userASP.Id, roleName);

}

}

public static async Task PasswordRecovery(string email)

{

var userManager = new UserManager<ApplicationUser>(new UserStore<ApplicationUser>(userContext));

var userASP = userManager.FindByEmail(email);

if (userASP == null)

{

return;

}

var random = new Random();

var newPassword = string.Format("{0}", random.Next(100000, 999999));

var response = await userManager.AddPasswordAsync(userASP.Id, newPassword);

if (response.Succeeded)

{

var subject = "Sales App - Recuperación de contraseña";

var body = string.Format(@"

<h1>Sales App - Recuperación de contraseña</h1>

<p>Su nueva contraseña es: <strong>{0}</strong></p>

<p>Por favor no olvide cambiarla por una de fácil recordación",

newPassword);

await MailHelper.SendMail(email, subject, body);

}

}

public void Dispose()

{

userContext.Dispose();

db.Dispose();

}

}

}

1. Creamos controlador **UsersController** en el **API**:

namespace Sales.API.Controllers

{

using System;

using System.IO;

using System.Web.Http;

using Common.Models;

using Helpers;

using Newtonsoft.Json.Linq;

[RoutePrefix("api/Users")]

public class UsersController : ApiController

{

public IHttpActionResult PostUser(UserRequest userRequest)

{

if (userRequest.ImageArray != null && userRequest.ImageArray.Length > 0)

{

var stream = new MemoryStream(userRequest.ImageArray);

var guid = Guid.NewGuid().ToString();

var file = $"{guid}.jpg";

var folder = "~/Content/Users";

var fullPath = $"{folder}/{file}";

var response = FilesHelper.UploadPhoto(stream, folder, file);

if (response)

{

userRequest.ImagePath = fullPath;

}

}

var answer = UsersHelper.CreateUserASP(userRequest);

return Ok(answer);

}

[HttpPost]

[Authorize]

[Route("GetUser")]

public IHttpActionResult GetUser(JObject form)

{

try

{

var email = string.Empty;

dynamic jsonObject = form;

try

{

email = jsonObject.Email.Value;

}

catch

{

return BadRequest("Incorrect call.");

}

var user = UsersHelper.GetUserASP(email);

return Ok(user);

}

catch (Exception ex)

{

return BadRequest(ex.Message);

}

}

}

}

1. Publicamos el controlador y probamos todo.
2. Ahora obtengamos el usuario luego de ingresar a la App. Crea estos modelos en **Models**:

public class Claim

{

public int Id { get; set; }

public string UserId { get; set; }

public string ClaimType { get; set; }

public string ClaimValue { get; set; }

}

Y:

using System.Collections.Generic;

public class UserASP

{

public string Email { get; set; }

public bool EmailConfirmed { get; set; }

public string PasswordHash { get; set; }

public string SecurityStamp { get; set; }

public string PhoneNumber { get; set; }

public bool PhoneNumberConfirmed { get; set; }

public bool TwoFactorEnabled { get; set; }

public object LockoutEndDateUtc { get; set; }

public bool LockoutEnabled { get; set; }

public int AccessFailedCount { get; set; }

public List<Claim> Claims { get; set; }

public string Id { get; set; }

public string UserName { get; set; }

}

Y:

public class GetUserRequest

{

public string Email { get; set; }

}

1. Y el método **GetUser** en el API:

public async Task<Response> GetUser(string urlBase, string prefix, string controller, string email, string tokenType, string accessToken)

{

try

{

var getUserRequest = new GetUserRequest

{

Email = email,

};

var request = JsonConvert.SerializeObject(getUserRequest);

var content = new StringContent(request, Encoding.UTF8, "application/json");

var client = new HttpClient();

client.BaseAddress = new Uri(urlBase);

client.DefaultRequestHeaders.Authorization = new AuthenticationHeaderValue(tokenType, accessToken);

var url = $"{prefix}{controller}";

var response = await client.PostAsync(url, content);

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

if (!response.IsSuccessStatusCode)

{

return new Response

{

IsSuccess = false,

Message = answer,

};

}

var user = JsonConvert.DeserializeObject<UserASP>(answer);

return new Response

{

IsSuccess = true,

Result = user,

};

}

catch (Exception ex)

{

return new Response

{

IsSuccess = false,

Message = ex.Message,

};

}

}

1. En el **LoginViewModel**, adicionar las siguientes líneas:

Settings.IsRemembered = this.IsRemembered;

Settings.AccessToken = token.AccessToken;

Settings.TokenType = token.TokenType;

var prefix = Application.Current.Resources["UrlPrefix"].ToString();

var controller = Application.Current.Resources["UrlUsersController"].ToString();

var response = await this.apiService.GetUser(url, prefix, $"{controller}/GetUser", this.Email, token.TokenType, token.AccessToken);

if (response.IsSuccess)

{

var userASP = (UserASP)response.Result;

MainViewModel.GetInstance().UserASP = userASP;

Settings.UserASP = JsonConvert.SerializeObject(userASP);

}

this.IsRunning = false;

this.IsEnabled = true;

this.Email = string.Empty;

1. Lo mismo en la clase **App**:

public App()

{

InitializeComponent();

var mainViewModel = MainViewModel.GetInstance();

if (Settings.IsRemembered)

{

if (!string.IsNullOrEmpty(Settings.UserASP))

{

mainViewModel.UserASP = JsonConvert.DeserializeObject<UserASP>(Settings.UserASP);

}

mainViewModel.Products = new ProductsViewModel();

this.MainPage = new MasterPage();

}

else

{

mainViewModel.Login = new LoginViewModel();

this.MainPage = new NavigationPage(new LoginPage());

}

}

1. Crear la propiedad en la **MainViewModel**:

public string UserFullName

{

get

{

if (this.UserASP != null && this.UserASP.Claims != null && this.UserASP.Claims.Count > 1)

{

return $"{this.UserASP.Claims[0].ClaimValue} {this.UserASP.Claims[1].ClaimValue}";

}

return null;

}

}

1. Adicionar label en el **MenuPage**:

<Image

Source="sales">

</Image>

<Label

FontAttributes="Bold"

FontSize="Large"

HorizontalOptions="Center"

Text="{Binding UserFullName}"

TextColor="White">

</Label>

<ListView

ItemsSource="{Binding Menu}"

HasUnevenRows="True">

<ListView.ItemTemplate>