Tomar Fotos y Acceder a la Galería

1. Cambia el source a la imagen y lo bindas como un **ImageSource**.

<Image

Source="{Binding ImageSource}">

</Image>

private ImageSource imageSource;

public ImageSource ImageSource

{

get { return this.imageSource; }

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

}

Inicializarla en el constructor:

this.ImageSource = "noproduct";

1. Agregar el plug in: **Xam.plugin.media** a todos los proyectos del front.
2. Agregar el **MainActivity**:

public override void OnRequestPermissionsResult(

int requestCode,

string[] permissions,

[GeneratedEnum] Permission[] grantResults)

{

PermissionsImplementation.Current.OnRequestPermissionsResult(

requestCode,

permissions,

grantResults);

}

Y adicione la línea amarilla en este método:

protected override void OnCreate(Bundle savedInstanceState)

{

TabLayoutResource = Resource.Layout.Tabbar;

ToolbarResource = Resource.Layout.Toolbar;

base.OnCreate(savedInstanceState);

CrossCurrentActivity.Current.Init(this, savedInstanceState);

global::Xamarin.Forms.Forms.Init(this, savedInstanceState);

LoadApplication(new App());

}

1. En el **AndroidManifest** agregar las líneas amarillas:

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

<manifest xmlns:android="http://schemas.android.com/apk/res/android" android:versionCode="1" android:versionName="1.0" package="com.companyname.Sales">

<uses-sdk android:minSdkVersion="21" android:targetSdkVersion="27" />

<uses-permission android:name="android.permission.INTERNET" />

<uses-permission android:name="android.permission.ACCESS\_WIFI\_STATE" />

<uses-permission android:name="android.permission.ACCESS\_NETWORK\_STATE" />

<uses-permission android:name="android.permission.CAMERA" />

<uses-permission android:name="android.permission.WRITE\_EXTERNAL\_STORAGE" />

<uses-permission android:name="android.permission.READ\_EXTERNAL\_STORAGE" />

<application android:label="Sales.Android">

<provider android:name="android.support.v4.content.FileProvider"

android:authorities="${applicationId}.fileprovider"

android:exported="false"

android:grantUriPermissions="true">

<meta-data android:name="android.support.FILE\_PROVIDER\_PATHS"

android:resource="@xml/file\_paths"></meta-data>

</provider>

</application>

</manifest>

1. En la carpeta **Resources** en el proyecto de **Android**, se debe crear una carpeta llama **xml**, dentro de esta carpeta se debe agregar un nuevo archivo de tipo xml llamado: **file\_paths.xml** y este debe contener:

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

<paths xmlns:android="http://schemas.android.com/apk/res/android">

<external-files-path name="my\_images" path="Pictures" />

<external-files-path name="my\_movies" path="Movies" />

</paths>

1. En iOS agregar esto al **info.plist**:

<key>NSCameraUsageDescription</key>

<string>This app needs access to the camera to take photos.</string>

<key>NSPhotoLibraryUsageDescription</key>

<string>This app needs access to photos.</string>

<key>NSMicrophoneUsageDescription</key>

<string>This app needs access to microphone.</string>

<key>NSPhotoLibraryAddUsageDescription</key>

<string>This app needs access to the photo gallery.</string>

1. Crear un atributo llamado file de tipo **MediaFile** en la **AddProductViewModel**:

private MediaFile file;

1. Crear los siguientes recursos de traducciones:

Inglés:

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

<value>Where do you take the picture?</value>

</data>

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

<value>From gallery</value>

</data>

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

<value>Take a new picture</value>

</data>

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

<value>Cancel</value>

</data>

Español:

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

<value>¿De dónde vas a tomar la foto?</value>

</data>

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

<value>De la galería</value>

</data>

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

<value>Tomar una nueva imagen</value>

</data>

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

<value>Cancelar</value>

</data>

Italiano:

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

<value>Dove prendi immagine ?</value>

</data>

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

<value>Dalla galleria!</value>

</data>

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

<value>Scatta una nuova foto</value>

</data>

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

<value>Annulla</value>

</data>

Portugués:

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

<value>Onde você tira a foto ?!</value>

</data>

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

<value>Da Galeria</value>

</data>

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

<value>Fotografar novamente</value>

</data>

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

<value>Cancelar</value>

</data>

1. Y agregue los literales la clase Languages:

public static string ImageSource

{

get { return Resource.ImageSource; }

}

public static string FromGallery

{

get { return Resource.FromGallery; }

}

public static string NewPicture

{

get { return Resource.NewPicture; }

}

public static string Cancel

{

get { return Resource.Cancel; }

}

1. Agregar el Tap Recognizer a la imagen y generar el comando y método correspondiente, con el siguiente código:

<Image

Source="{Binding ImageSource}">

<Image.GestureRecognizers>

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

</Image.GestureRecognizers>

</Image>

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 = file.GetStream();

return stream;

});

}

}

1. Pruebe que puede tomar fotos y seleccionar de la galería. Taran!!!!!
2. Ahora vamos a mandar la foto para el servicio. Primero creamos la clase **FilesHelper**:

using System.IO;

public class FilesHelper

{

public static byte[] ReadFully(Stream input)

{

using (MemoryStream ms = new MemoryStream())

{

input.CopyTo(ms);

return ms.ToArray();

}

}

}

1. Agregamos esta propiedad al modelo **Products**:

[NotMapped]

public byte[] ImageArray { get; set; }

1. Adicionamos estas líneas antes de enviar el método **POST** en la **AddProductViewModel**:

byte[] imageArray = null;

if (this.file != null)

{

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

}

var product = new Product

{

Description = this.Description,

Price = price,

Remarks = this.Remarks,

ImageArray = imageArray,

};

1. Ahora vamos al proyecto **API** y creamos el folder **Products** dentro de **Content**. Y adicione al menos un archivo.
2. Creamos el folder **Helpers** y dentro de este la clase **FilesHelper** con el siguiente código:

using System.IO;

using System.Web;

public class FilesHelper

{

public static bool UploadPhoto(MemoryStream stream, string folder, string name)

{

try

{

stream.Position = 0;

var path = Path.Combine(HttpContext.Current.Server.MapPath(folder), name);

File.WriteAllBytes(path, stream.ToArray());

}

catch

{

return false;

}

return true;

}

}

1. Modifique el método POST por lo siguiente:

[ResponseType(typeof(Product))]

public async Task<IHttpActionResult> PostProduct(Product product)

{

product.IsAvailable = true;

product.PublishOn = DateTime.Now.ToUniversalTime();

if (!ModelState.IsValid)

{

return BadRequest(ModelState);

}

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

{

var stream = new MemoryStream(product.ImageArray);

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

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

var folder = "~/Content/Products";

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

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

if (response)

{

product.ImagePath = fullPath;

}

}

this.db.Products.Add(product);

await this.db.SaveChangesAsync();

return CreatedAtRoute("DefaultApi", new { id = product.ProductId }, product);

}

1. Cambiamos la ruta donde va a buscar las imágenes en el modelo **Products**:

return $"[https://salesapiservices.azurewebsites.net/{this.ImagePath.Substring(1)](https://salesapiservices.azurewebsites.net/%7Bthis.ImagePath.Substring(1))}";

1. Publicamos los campos y probamos el envío de imágenes desde la App.