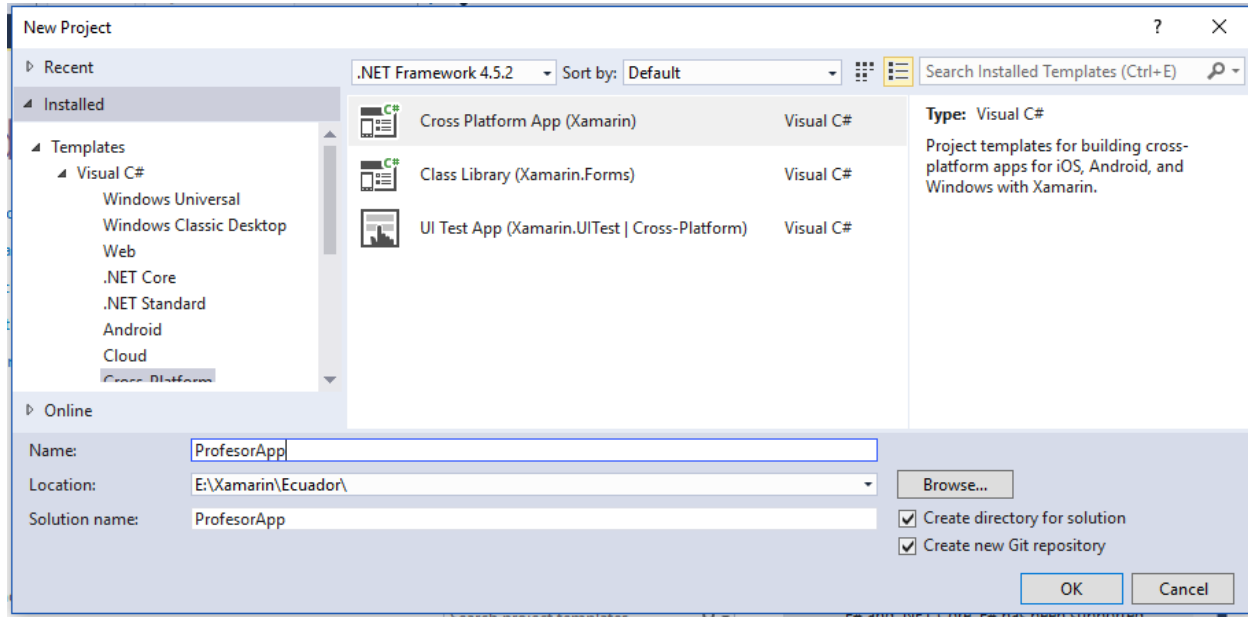
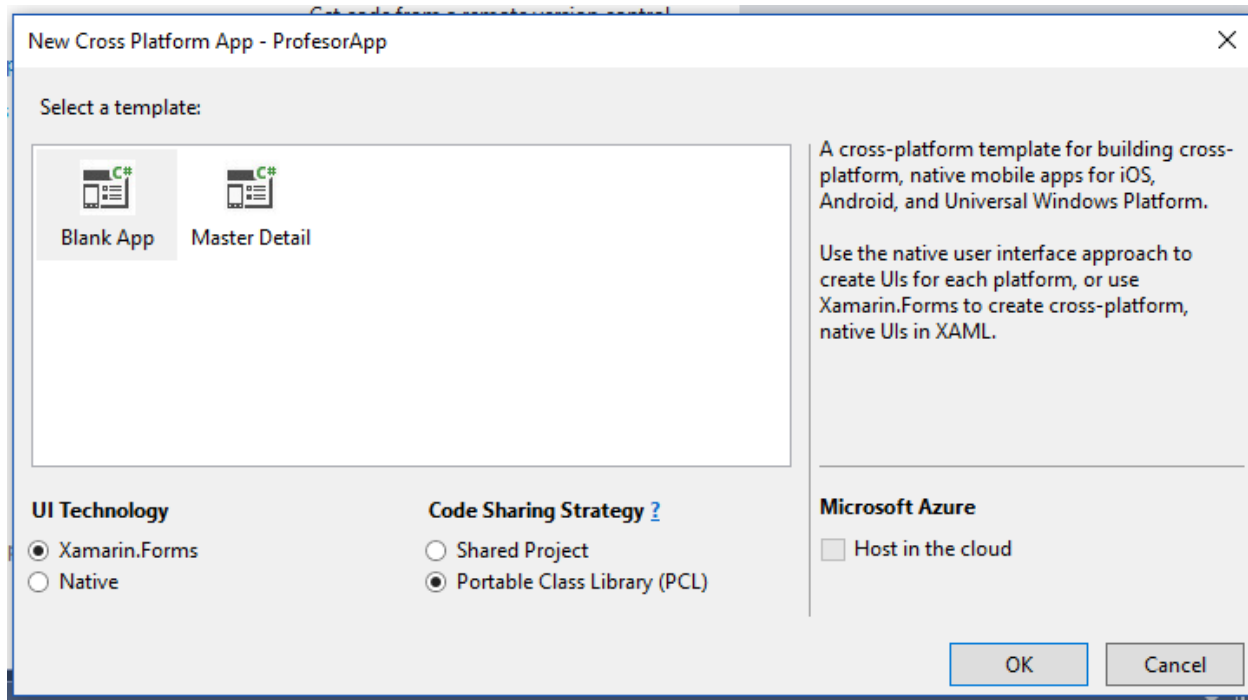


Parte 4: Creación de la app móvil ProfesorApp – Autor: Luis Beltrán

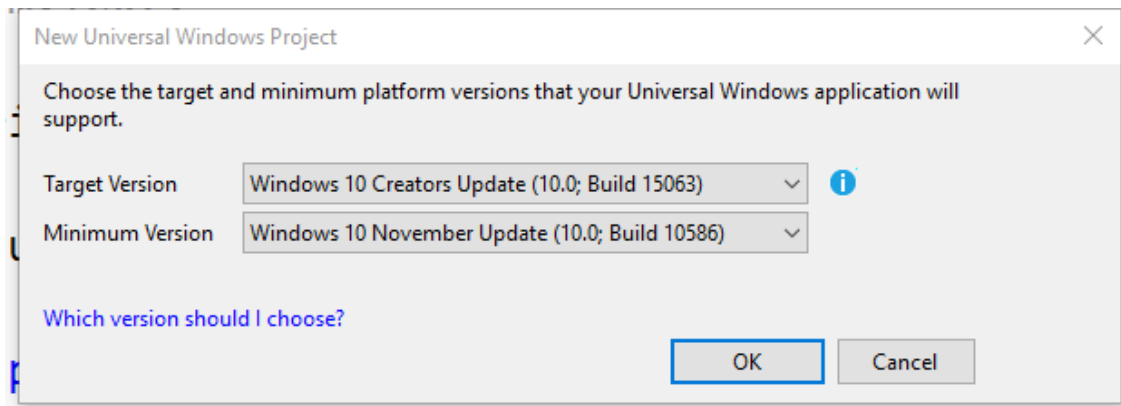
Paso 1. Crea un nuevo proyecto de la categoría **Cross-Platform** selecciona **Aplicación multiplataforma (Xamarin.Forms o nativa)** y coloca el nombre de proyecto **ProfesorApp**. Además, la ruta del proyecto debe ser una ubicación corta para evitar problemas de ruta larga.



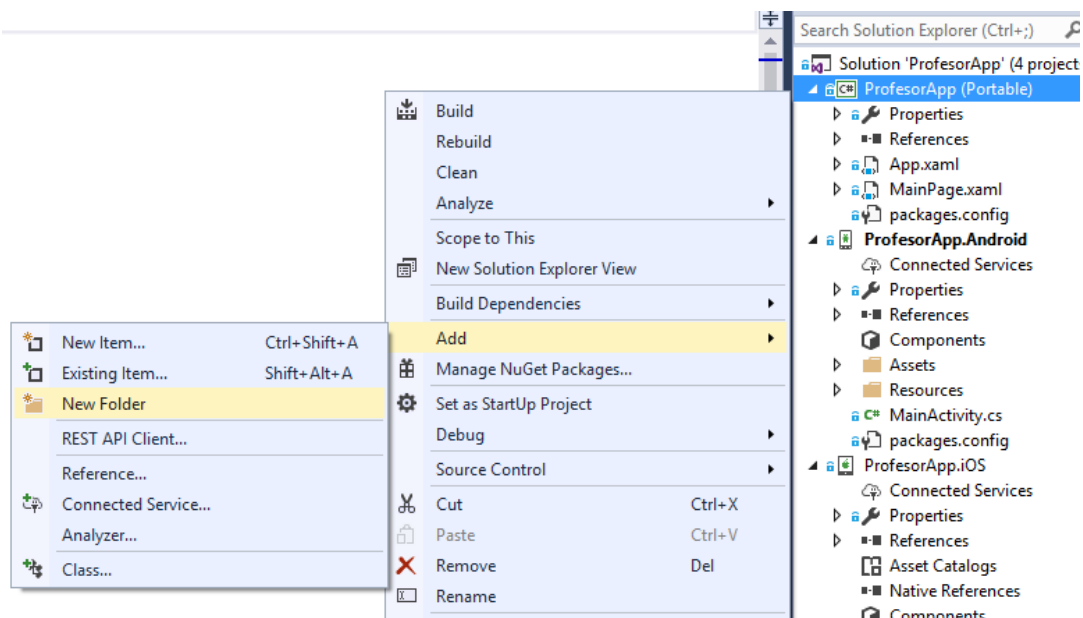
Paso 2. Selecciona la plantilla **Aplicación en blanco**, la tecnología de IU **Xamarin.Forms** y la estrategia de uso compartido de código **Biblioteca de clases portátil (PCL)**. Da clic en **OK**.



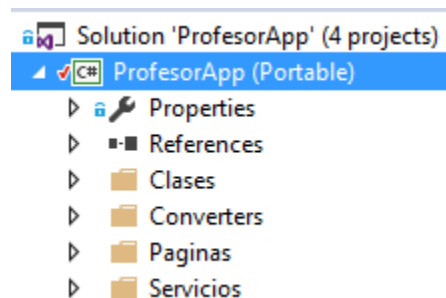
Paso 3. Si tienes instalado el SDK de Windows 10, aparecerá la ventana de selección del Target y Minimum Version. Selecciónalas a conveniencia, según la versión que tengas instalada.



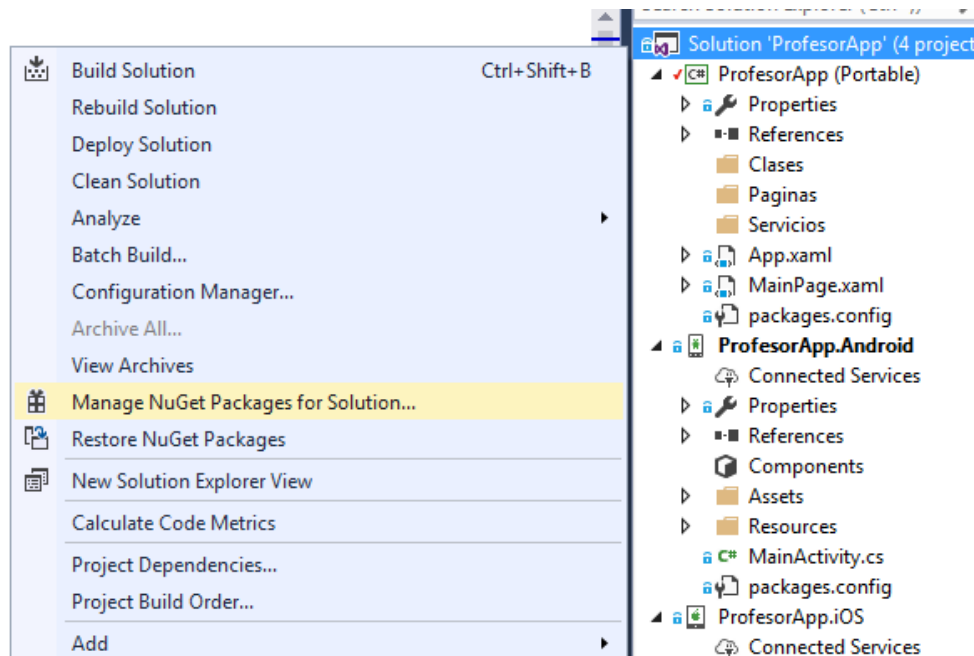
Paso 4. Da clic en **Agregar → Nueva carpeta** en el menú contextual del proyecto PCL.



Paso 5. Agrega las carpetas **Clases**, **Converters**, **Paginas** y **Servicios** al proyecto

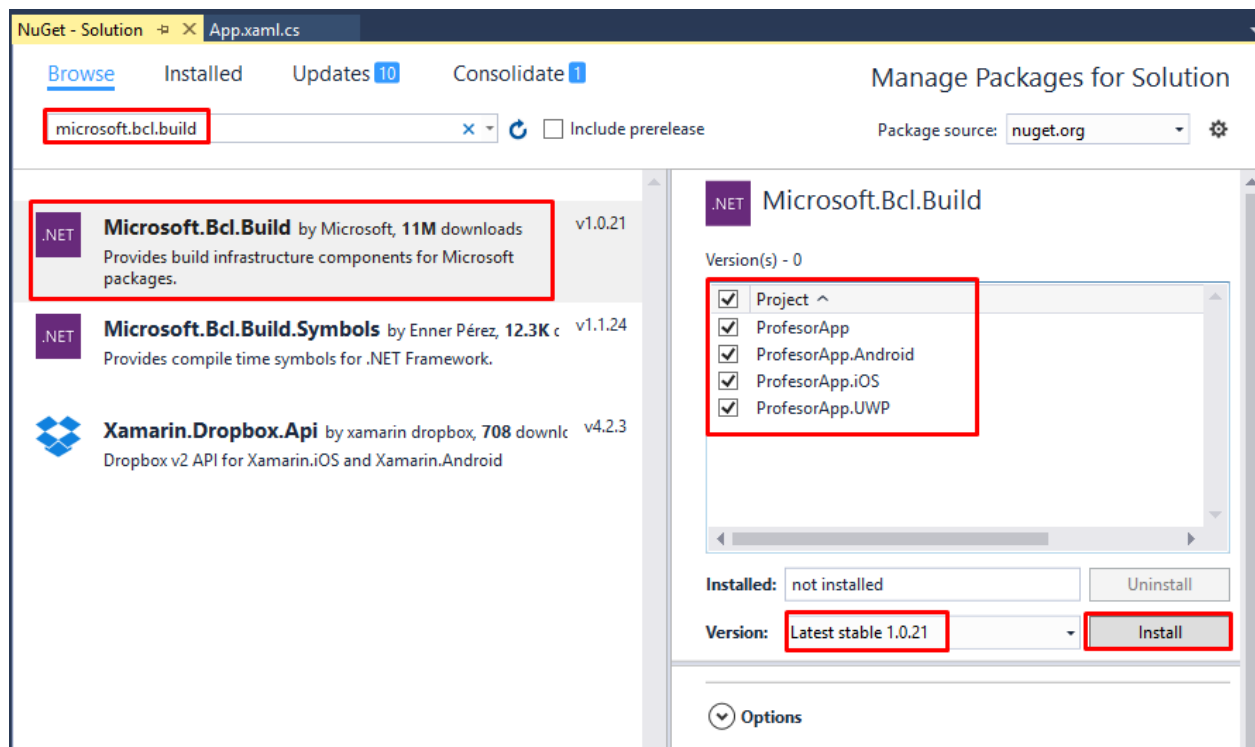


Paso 6. Da clic en **Administrar paquetes Nuget para la solución** en el menú contextual de la solución.

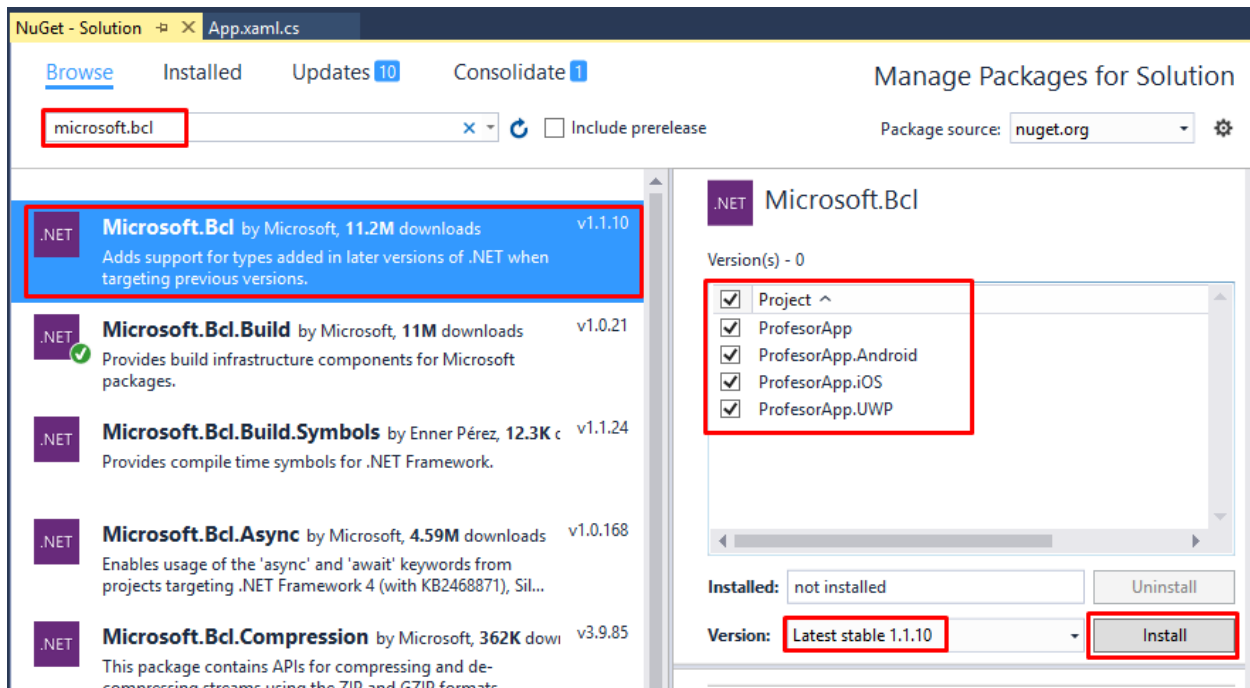


Paso 7. Agrega los siguientes paquetes. Revisa la versión en cada caso:

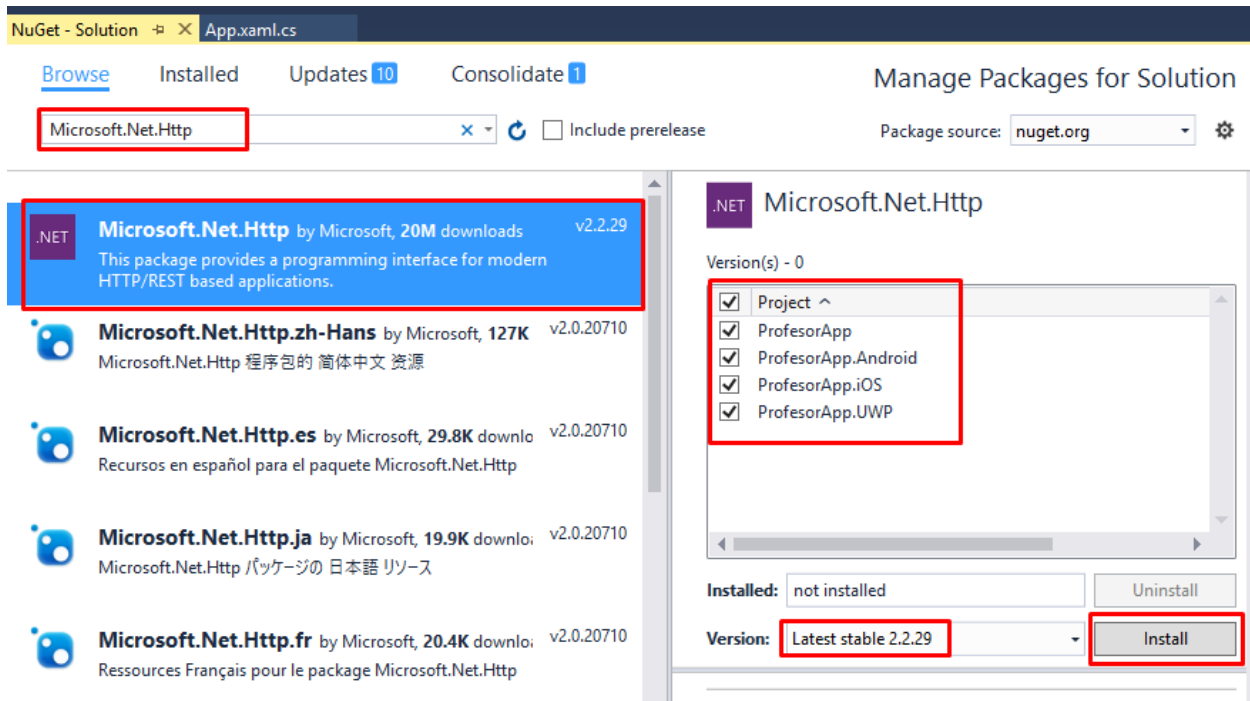
a) **Microsoft.Bcl.Build** (requisito de Microsoft.Net.Http)



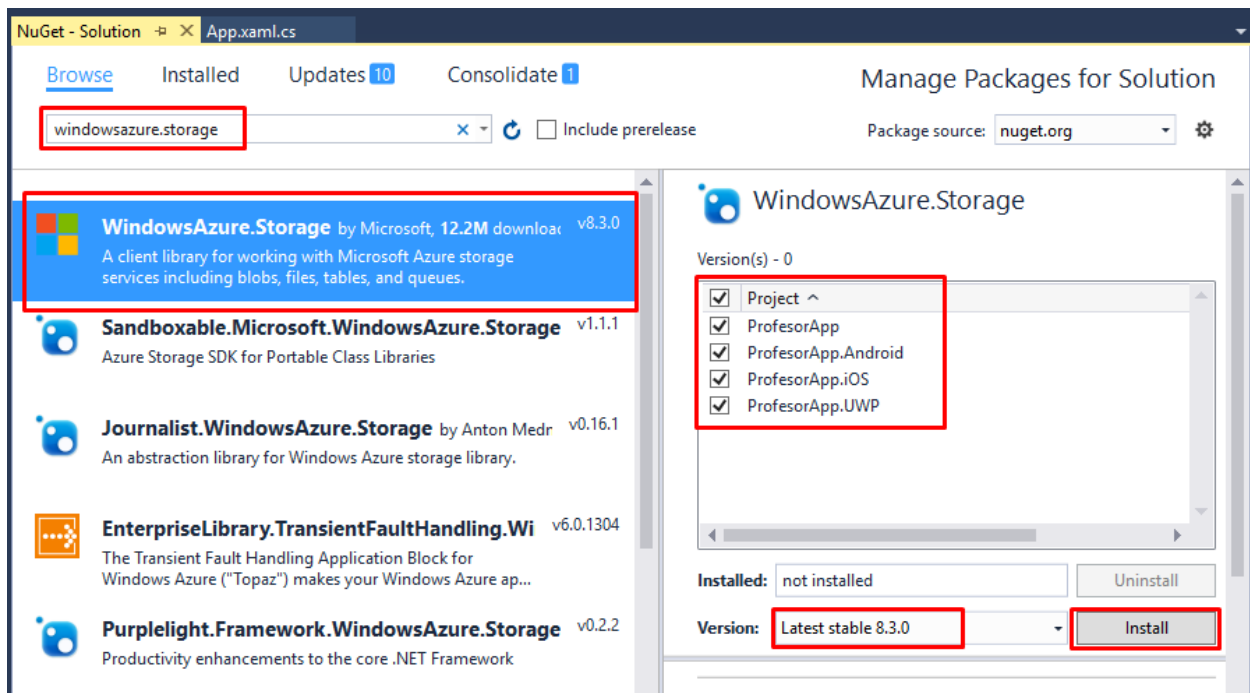
b) **Microsoft.Bcl** (requisito de Microsoft.Net.Http)



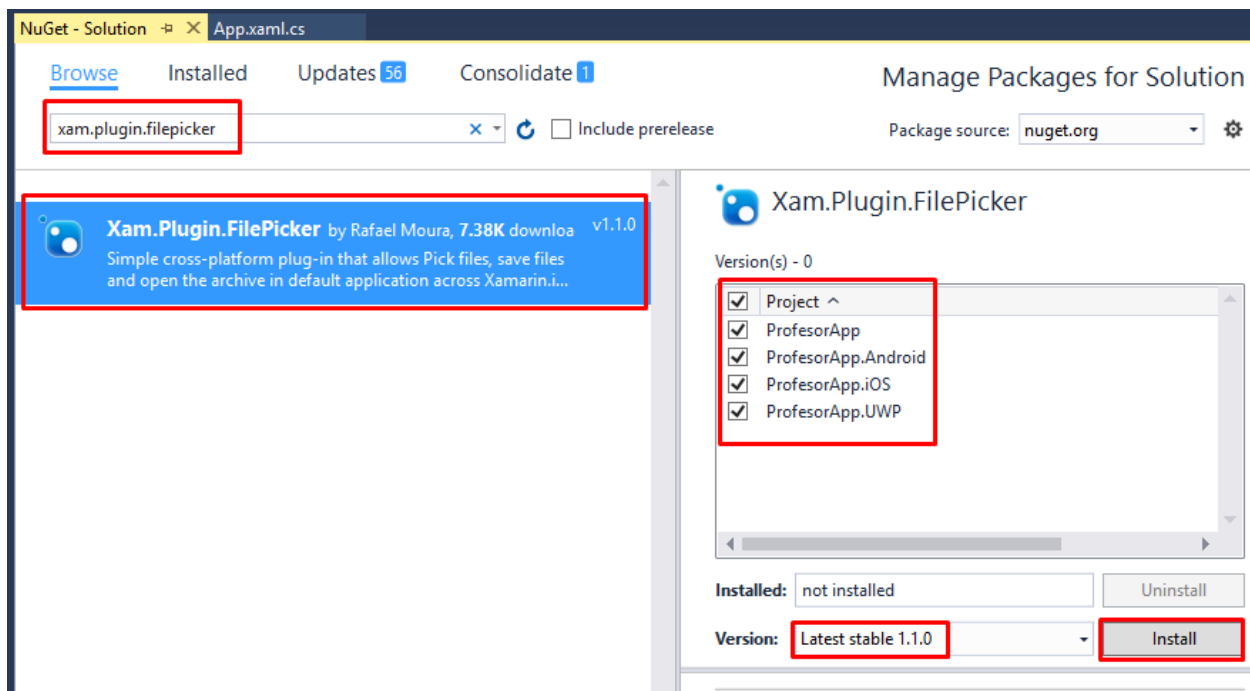
c) **Microsoft.Net.Http** (requisito de para las peticiones al servicio web)



d) **WindowsAzure.Storage** (para la conexión con el Azure Storage)



e) **Xam.Plugin.FilePicker** (para que el usuario pueda seleccionar un archivo de su dispositivo)



f) **Newtonsoft.Json** (actualizar el paquete de UWP)

The screenshot shows the NuGet Package Manager interface in Visual Studio. The left pane lists installed packages, with **Newtonsoft.Json** highlighted by a red box. The right pane shows the details for **Newtonsoft.Json**, including a table of installed versions across projects. In this table, **ProfesorApp.UWP** is selected with a red box. Below the table, the **Version** dropdown is set to **9.0.1** (highlighted with a red box), and the **Install** button is also highlighted with a red box.

Newtonsoft.Json by James Newton-King, **76.7M** downloads v10.0.3
Json.NET is a popular high-performance JSON framework for .NET

NUnit by Charlie Poole, **11.9M** downloads v3.7.1
NUnit is a unit-testing framework for all .NET languages with a strong TDD focus.

EntityFramework by Microsoft, **32.9M** downloads v6.1.3
Entity Framework is Microsoft's recommended data access technology for new applications.

HtmlAgilityPack by ZZZ Projects, Simon Mourrier, Jeff Klawiter, S v1.5.1
This is an agile HTML parser that builds a read/write DOM and supports plain XPATH or XSLT (you actually don't HAVE to understand...)

jQuery by jQuery Foundation, Inc., **38.1M** downloads v3.1.1
jQuery is a new kind of JavaScript Library.
jQuery is a fast and concise JavaScript Library that simplifies HT...

bootstrap by Twitter, Inc., **11.8M** downloads v3.3.7
Bootstrap framework in CSS. Includes fonts and JavaScript

Newtonsoft.Json

Version(s) - 1

| Project ^ | Version |
|---|---------|
| <input type="checkbox"/> ProfesorApp | 9.0.1 |
| <input type="checkbox"/> ProfesorApp.Android | 9.0.1 |
| <input type="checkbox"/> ProfesorApp.iOS | 9.0.1 |
| <input checked="" type="checkbox"/> ProfesorApp.UWP | 9.0.1 |

Installed: 9.0.1 Uninstall

Version: 9.0.1 Install

Options

Description

Paso 8. En la carpeta **Clases**, agrega las siguientes clases:

- a) **Alumno:** Esta clase recibe los datos de la tabla Alumno creada en el servicio web.

```
namespace ProfesorApp.Clases
{
    public class Alumno
    {
        public int Id { get; set; }
        public string Nombre { get; set; }
        public string FotoURL { get; set; }
        public string Usuario { get; set; }
        public string Password { get; set; }
        public string FotoURLSAS { get; set; }
    }
}
```

- b) **Tarea:** Esta clase recibe los datos de la tabla **Tarea** creada en el servicio web.

```
using System;

namespace ProfesorApp.Clases
{
    public class Tarea
    {
        public int Id { get; set; }
        public string Titulo { get; set; }
        public string ArchivoURL { get; set; }
        public DateTime FechaPublicacion { get; set; } = DateTime.Now;
        public DateTime FechaLimite { get; set; } = DateTime.Now;
    }
}
```

- c) **TareaAlumno:** Esta clase recibe los datos de la tabla **TareaAlumno**

```
using System;

namespace ProfesorApp.Clases
{
    public class TareaAlumno
    {
        public int IdTarea { get; set; }
        public int IdAlumno { get; set; }

        public string Mensaje { get; set; }
        public string ArchivoURL { get; set; }
        public DateTime Fecha { get; set; }
        public int Calificacion { get; set; }
        public bool Evaluado { get; set; }

        public Tarea Tarea { get; set; }
        public Alumno Alumno { get; set; }
    }
}
```

Paso 9. En la carpeta **Servicios**, agrega las siguientes clases:

- a) **ServicioStorage**: Esta clase permite descargar y subir un archivo al blob storage de Azure definido en la parte anterior. Requiere conocer la **URL del storage**, así como la **cadena SAS**.

```
using Microsoft.WindowsAzure.Storage.Blob;
using System.Threading.Tasks;
using System.IO;
using System;

namespace ProfesorApp.Servicios
{
    public class ServicioStorage
    {
        const string StorageURL = "este valor lo debes establecer";
        const string ContainerAlumno = "alumnos";
        const string ContainerTarea = "tareas-asignadas";
        const string ContainerTareaAlumno = "tareas-alumnos";
        const string SASQueryString = "este valor lo debes establecer";

        public async Task<string> UploadTarea(int id, Stream stream)
        {
            string blobSAS = $"{StorageURL}/{ContainerTarea}/{id}.pdf{SASQueryString}";
            return await UploadBlob(blobSAS, stream);
        }

        public async Task<string> UploadAlumno(int id, Stream stream)
        {
            string blobSAS = $"{StorageURL}/{ContainerAlumno}/{id}.jpg{SASQueryString}";
            return await UploadBlob(blobSAS, stream);
        }

        public async Task<Stream> DownloadAlumno(int id)
        {
            string blobSAS = $"{StorageURL}/{ContainerAlumno}/{id}.jpg{SASQueryString}";
            return await DownloadBlob(blobSAS);
        }

        public string GetFullDownloadTareaURL(int id)
        {
            return $"{StorageURL}/{ContainerTarea}/{id}.pdf{SASQueryString}";
        }

        public string GetFullDownloadAlumnoURL(int id)
        {
            return $"{StorageURL}/{ContainerAlumno}/{id}.pdf{SASQueryString}";
        }

        public string GetFullDownloadTareaAlumnoURL(int idTarea, int idAlumno)
        {
            return
                $"{StorageURL}/{ContainerTareaAlumno}/{idTarea}_{idAlumno}.pdf{SASQueryString}";
        }
    }
}
```



```

public async Task<Stream> DownloadTarea(int id)
{
    string blobSAS = $"{StorageURL}/{ContainerTarea}/{id}.pdf{SASQueryString}";
    return await DownloadBlob(blobSAS);
}

public async Task<Stream> DownloadTareaAlumnos(int idTarea, int idAlumno)
{
    string blobSAS =
    $"{StorageURL}/{ContainerTareaAlumno}/{idTarea}_{idAlumno}.pdf{SASQueryString}";
    return await DownloadBlob(blobSAS);
}

private async Task<Stream> DownloadBlob(string blobSAS)
{
    try
    {
        CloudBlockBlob blob = new CloudBlockBlob(new Uri(blobSAS));
        MemoryStream stream = new MemoryStream();
        await blob.DownloadToStreamAsync(stream);
        return stream;
    }
    catch (Exception exc)
    {
        string msgError = exc.Message;
        return null;
    }
}

private async Task<string> UploadBlob(string blobSAS, Stream stream)
{
    string url = "";

    try
    {
        CloudBlockBlob blob = new CloudBlockBlob(new Uri(blobSAS));

        using (stream)
        {
            await blob.UploadFromStreamAsync(stream);
            url = blob.StorageUri.PrimaryUri.AbsoluteUri;
        }
    }
    catch (Exception exc)
    {
        string msgError = exc.Message;
    }

    return url;
}
}
}

```

- b) **ServicioWebApi:** Esta clase define los métodos de acceso al servicio web publicado en la parte 2. Incluye métodos para obtener, agregar, modificar y eliminar datos de las diferentes tablas.

```
using System;
using System.Collections.Generic;
using System.Text;
using System.Threading.Tasks;
using Newtonsoft.Json;
using System.Net.Http;
using ProfesorApp.Clases;
using System.Net.Http.Headers;
using System.Net;

namespace ProfesorApp.Servicios
{
    public static class ServicioWebApi
    {
        const string WebApiURL = "https://alumnosweb-luisb.azurewebsites.net";

        private static HttpClient Cliente = new HttpClient();

        public static async Task<List<Alumno>> GetAlumnos()
        {
            List<Alumno> datos = null;
            Cliente.DefaultRequestHeaders.Accept.Clear();
            Cliente.DefaultRequestHeaders.Accept.Add(new
MediaTyewithQualityHeaderValue("application/json"));

            var url = $"{WebApiURL}/api/Alumnos/";
            var respuesta = await Cliente.GetAsync(url);

            if (respuesta.StatusCode == HttpStatusCode.OK)
            {
                var json = await respuesta.Content.ReadAsStringAsync();
                datos = JsonConvert.DeserializeObject<List<Alumno>>(json);

                var servicioStorage = new ServicioStorage();

                foreach (var alumno in datos)
                {
                    alumno.FotoURLSAS =
servicioStorage.GetFullDownloadAlumnoURL(alumno.Id);
                }

                return datos;
            }

            public static async Task<Alumno> GetAlumno(int id)
            {
                Alumno dato = null;
                Cliente.DefaultRequestHeaders.Accept.Clear();
                Cliente.DefaultRequestHeaders.Accept.Add(new
MediaTyewithQualityHeaderValue("application/json"));

                var url = $"{WebApiURL}/api/Alumnos/{id}";
```

```

        var respuesta = await Cliente.GetAsync(url);

        if (respuesta.StatusCode == HttpStatusCode.OK)
        {
            var json = await respuesta.Content.ReadAsStringAsync();
            dato = JsonConvert.DeserializeObject<Alumno>(json);
        }

        return dato;
    }

    public static async Task<Alumno> AddAlumno(Alumno info)
    {
        Alumno dato = null;
        Cliente.BaseAddress = new Uri(WebApiURL);
        Cliente.DefaultRequestHeaders.Accept.Clear();
        Cliente.DefaultRequestHeaders.Accept.Add(new
MediaTyewithQualityHeaderValue("application/json"));

        var url = $"/api/Alumnos/";
        var jsonContent = JsonConvert.SerializeObject(info);
        var respuesta = await Cliente.PostAsync(url, new
StringContent(jsonContent.ToString(), Encoding.UTF8, "application/json"));

        //if (respuesta.StatusCode == HttpStatusCode.Created)
        {
            var json = await respuesta.Content.ReadAsStringAsync();
            dato = JsonConvert.DeserializeObject<Alumno>(json);
        }

        return dato;
    }

    public static async Task<Alumno> UpdateAlumno(Alumno info)
    {
        Alumno dato = null;
        Cliente.BaseAddress = new Uri(WebApiURL);
        Cliente.DefaultRequestHeaders.Accept.Clear();
        Cliente.DefaultRequestHeaders.Accept.Add(new
MediaTyewithQualityHeaderValue("application/json"));

        var url = $"/api/Alumnos/{info.Id}";
        var jsonContent = JsonConvert.SerializeObject(info);
        var respuesta = await Cliente.PutAsync(url, new
StringContent(jsonContent.ToString(), Encoding.UTF8, "application/json"));

        //if (respuesta.StatusCode == HttpStatusCode.Created)
        {
            var json = await respuesta.Content.ReadAsStringAsync();
            dato = JsonConvert.DeserializeObject<Alumno>(json);
        }

        return dato;
    }

    public static async Task<bool> DeleteAlumno(int id)
    {
        Cliente.BaseAddress = new Uri(WebApiURL);

```

```

        Cliente.DefaultRequestHeaders.Accept.Clear();
        Cliente.DefaultRequestHeaders.Accept.Add(new
MediaTyewithQualityHeaderValue("application/json"));

        var url = $"/api/Alumnos/{id}";
        var respuesta = await Cliente.DeleteAsync(url);
        return respuesta.IsSuccessStatusCode;
    }

    public static async Task<List<Tarea>> GetTareas()
    {
        List<Tarea> datos = null;
        Cliente.DefaultRequestHeaders.Accept.Clear();
        Cliente.DefaultRequestHeaders.Accept.Add(new
MediaTyewithQualityHeaderValue("application/json"));

        var url = $"{WebApiURL}/api/Tareas/";
        var respuesta = await Cliente.GetAsync(url);

        if (respuesta.StatusCode == HttpStatusCode.OK)
        {
            var json = await respuesta.Content.ReadAsStringAsync();
            datos = JsonConvert.DeserializeObject<List<Tarea>>(json);
        }

        return datos;
    }

    public static async Task<Tarea> GetTarea(int id)
    {
        Tarea dato = null;
        Cliente.DefaultRequestHeaders.Accept.Clear();
        Cliente.DefaultRequestHeaders.Accept.Add(new
MediaTyewithQualityHeaderValue("application/json"));

        var url = $"{WebApiURL}/api/Tareas/{id}";
        var respuesta = await Cliente.GetAsync(url);

        if (respuesta.StatusCode == HttpStatusCode.OK)
        {
            var json = await respuesta.Content.ReadAsStringAsync();
            dato = JsonConvert.DeserializeObject<Tarea>(json);
        }

        return dato;
    }

    public static async Task<Tarea> AddTarea(Tarea info)
    {
        Tarea dato = null;
        Cliente.BaseAddress = new Uri(WebApiURL);
        Cliente.DefaultRequestHeaders.Accept.Clear();
        Cliente.DefaultRequestHeaders.Accept.Add(new
MediaTyewithQualityHeaderValue("application/json"));

        var url = $"/api/Tareas/";
        var jsonContent = JsonConvert.SerializeObject(info);

```

```

        var respuesta = await Cliente.PostAsync(url, new
StringContent(jsonContent.ToString(), Encoding.UTF8, "application/json"));

        //if (respuesta.StatusCode == HttpStatusCode.Created)
        {
            var json = await respuesta.Content.ReadAsStringAsync();
            dato = JsonConvert.DeserializeObject<Tarea>(json);
        }

        return dato;
    }

    public static async Task<Tarea> UpdateTarea(Tarea info)
    {
        Tarea dato = null;
        Cliente.BaseAddress = new Uri(WebApiURL);
        Cliente.DefaultRequestHeaders.Accept.Clear();
        Cliente.DefaultRequestHeaders.Accept.Add(new
MediaTypeWithQualityHeaderValue("application/json"));

        var url = $"/api/Tareas/{info.Id}";
        var jsonContent = JsonConvert.SerializeObject(info);
        var respuesta = await Cliente.PutAsync(url, new
StringContent(jsonContent.ToString(), Encoding.UTF8, "application/json"));

        //if (respuesta.StatusCode == HttpStatusCode.Created)
        {
            var json = await respuesta.Content.ReadAsStringAsync();
            dato = JsonConvert.DeserializeObject<Tarea>(json);
        }

        return dato;
    }

    public static async Task<bool> DeleteTarea(int id)
    {
        Cliente.BaseAddress = new Uri(WebApiURL);
        Cliente.DefaultRequestHeaders.Accept.Clear();
        Cliente.DefaultRequestHeaders.Accept.Add(new
MediaTypeWithQualityHeaderValue("application/json"));

        var url = $"/api/Tareas/{id}";
        var respuesta = await Cliente.DeleteAsync(url);
        return respuesta.IsSuccessStatusCode;
    }

    public static async Task<List<TareaAlumno>> GetTareaAlumnosByEval(bool evaluado)
    {
        List<TareaAlumno> datos = null;
        Cliente.DefaultRequestHeaders.Accept.Clear();
        Cliente.DefaultRequestHeaders.Accept.Add(new
MediaTypeWithQualityHeaderValue("application/json"));

        var url = $"{WebApiURL}/api/TareaAlumnos/GetTareaAlumnosByEval/{evaluado}";
        var respuesta = await Cliente.GetAsync(url);

        if (respuesta.StatusCode == HttpStatusCode.OK)
        {

```

```

        var json = await respuesta.Content.ReadAsStringAsync();
        datos = JsonConvert.DeserializeObject<List<TareaAlumno>>(json);
    }

    return datos;
}

public static async Task<TareaAlumno> GetTareaAlumno(int idTarea, int idAlumno)
{
    TareaAlumno dato = null;
    Cliente.DefaultRequestHeaders.Accept.Clear();
    Cliente.DefaultRequestHeaders.Accept.Add(new
MediaTypeWithQualityHeaderValue("application/json"));

    var url = $"{WebApiURL}/api/TareaAlumnos/{idTarea}/{idAlumno}";
    var respuesta = await Cliente.GetAsync(url);

    if (respuesta.StatusCode == HttpStatusCode.OK)
    {
        var json = await respuesta.Content.ReadAsStringAsync();
        dato = JsonConvert.DeserializeObject<TareaAlumno>(json);
    }

    return dato;
}

public static async Task<TareaAlumno> AddTareaAlumno(TareaAlumno info)
{
    TareaAlumno dato = null;
    Cliente.BaseAddress = new Uri(WebApiURL);
    Cliente.DefaultRequestHeaders.Accept.Clear();
    Cliente.DefaultRequestHeaders.Accept.Add(new
MediaTypeWithQualityHeaderValue("application/json"));

    var url = $"/api/TareaAlumnos/";
    var jsonContent = JsonConvert.SerializeObject(info);
    var respuesta = await Cliente.PostAsync(url, new
StringContent(jsonContent.ToString(), Encoding.UTF8, "application/json"));

    //if (respuesta.StatusCode == HttpStatusCode.Created)
    {
        var json = await respuesta.Content.ReadAsStringAsync();
        dato = JsonConvert.DeserializeObject<TareaAlumno>(json);
    }

    return dato;
}

public static async Task<TareaAlumno> UpdateTareaAlumno(TareaAlumno info)
{
    TareaAlumno dato = null;
    Cliente.BaseAddress = new Uri(WebApiURL);
    Cliente.DefaultRequestHeaders.Accept.Clear();
    Cliente.DefaultRequestHeaders.Accept.Add(new
MediaTypeWithQualityHeaderValue("application/json"));

    var url = $"/api/TareaAlumnos/{info.IdTarea}/{info.IdAlumno}";
    var jsonContent = JsonConvert.SerializeObject(info);

```

```

        var respuesta = await Cliente.PutAsync(url, new
StringContent(jsonContent.ToString(), Encoding.UTF8, "application/json"));

        //if (respuesta.StatusCode == HttpStatusCode.Created)
        {
            var json = await respuesta.Content.ReadAsStringAsync();
            dato = JsonConvert.DeserializeObject<TareaAlumno>(json);
        }

        return dato;
    }

    public static async Task<bool> DeleteTareaAlumno(int idTarea, int idAlumno)
    {
        Cliente.BaseAddress = new Uri(WebApiURL);
        Cliente.DefaultRequestHeaders.Accept.Clear();
        Cliente.DefaultRequestHeaders.Accept.Add(new
MediaTypeWithQualityHeaderValue("application/json"));

        var url = $"/api/TareaAlumnos/{idTarea}/{idAlumno}";
        var respuesta = await Cliente.DeleteAsync(url);
        return respuesta.IsSuccessStatusCode;
    }
}
}

```

- c) **ServicioFilePicker:** Esta clase utiliza el plugin de **FilePicker** para permitir al usuario seleccionar un archivo.

```

using Plugin.FilePicker;
using System;
using System.IO;
using System.Threading.Tasks;

namespace ProfesorApp.Servicios
{
    public class ServicioFilePicker
    {
        public async Task<MemoryStream> GetFile()
        {
            try
            {
                var data = await CrossFilePicker.Current.PickFile();
                var bytes = data.DataArray;
                return new MemoryStream(bytes);
            }
            catch (Exception ex)
            {
                return null;
            }
        }
    }
}

```

Paso 10. En la carpeta **Converters**, agrega las siguientes clases, las cuales servirán para mostrar un formato de datos diferente en la interfaz de usuario, dependiendo el tipo de valor suministrado.

- a) **ConvertidorFecha:** Toma un objeto de tipo DateTime y lo convierte a una cadena en formato día/mes/año

```
using System;
using System.Globalization;
using Xamarin.Forms;

namespace ProfesorApp.Converters
{
    public class ConvertidorFecha : IValueConverter
    {
        public object Convert(object value, Type targetType, object parameter,
CultureInfo culture)
        {
            if (value is DateTime)
                return ((DateTime)value).ToString("dd/MM/yyyy");
            return string.Empty;
        }

        public object ConvertBack(object value, Type targetType, object parameter,
CultureInfo culture)
        {
            throw new NotImplementedException();
        }
    }
}
```

- b) **ConvertidorEvaluacionTarea:** Sirve para mostrar al usuario un mensaje si la tarea ha sido evaluada o está pendiente. El valor convertido es un booleano.

```
using System;
using System.Globalization;
using Xamarin.Forms;

namespace ProfesorApp.Converters
{
    public class ConvertidorEvaluacionTarea : IValueConverter
    {
        public object Convert(object value, Type targetType, object parameter,
CultureInfo culture)
        {
            if (value is bool)
                return ((bool)value) ? "Tarea evaluada" : "Pendiente de evaluar";
            return string.Empty;
        }

        public object ConvertBack(object value, Type targetType, object parameter,
CultureInfo culture)
        {
            throw new NotImplementedException();
        }
    }
}
```


- c) **ConvertidorMensajeCorto:** Muestra los primeros 50 caracteres de una cadena en caso de que ésta sea muy larga

```
using System;
using System.Globalization;
using Xamarin.Forms;

namespace ProfesorApp.Converters
{
    public class ConvertidorMensajeCorto : IValueConverter
    {
        public object Convert(object value, Type targetType, object parameter,
CultureInfo culture)
        {
            if (value is string)
            {
                var Mensaje = value.ToString();
                return (Mensaje.Length > 50) ? Mensaje.Substring(0, 50) : Mensaje;
            }

            return string.Empty;
        }

        public object ConvertBack(object value, Type targetType, object parameter,
CultureInfo culture)
        {
            throw new NotImplementedException();
        }
    }
}
```

Paso 11: Modifica **App.xaml**. En este archivo vamos a colocar las referencias a los convertidores para que sean accesibles de manera general en la aplicación. Además, vamos a definir **estilos** en nuestra aplicación, disponibles para los diferentes controles. Observa el código, donde se agrega una referencia al espacio de nombres **Converters** a través del alias **Convertidor**. Tanto los **Converters** como los estilos se definen dentro de un **ResourceDictionary**, incluido dentro de **ApplicationResources**.

```
<?xml version="1.0" encoding="utf-8" ?>
<Application xmlns="http://xamarin.com/schemas/2014/forms"
    xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
    xmlns:Convertidor="clr-namespace:ProfesorApp.Converters"
    x:Class="ProfesorApp.App">
    <Application.Resources>

        <ResourceDictionary>
            <Convertidor:ConvertidorEvaluacionTarea x:Key="ConvertidorEvaluacionTarea"/>
            <Convertidor:ConvertidorFecha x:Key="ConvertidorFecha"/>
            <Convertidor:ConvertidorMensajeCorto x:Key="ConvertidorMensajeCorto"/>

            <Style x:Key="LabelNormal" TargetType="Label">
                <Setter Property="FontSize" Value="20"/>
                <Setter Property="TextColor" Value="Black"/>
                <Setter Property="HorizontalOptions" Value="Start"/>
                <Setter Property="Margin" Value="10"/>
                <Setter Property="HorizontalTextAlignment" Value="Start"/>
            </Style>

            <Style x:Key="LabelTitulo" TargetType="Label" BasedOn="{StaticResource
LabelNormal}">
                <Setter Property="FontAttributes" Value="Bold"/>
            </Style>

            <Style x:Key="LabelDetalle" TargetType="Label">
                <Setter Property="LineBreakMode" Value="WordWrap"/>
                <Setter Property="FontSize" Value="15"/>
                <Setter Property="TextColor" Value="#030303"/>
                <Setter Property="HorizontalOptions" Value="Start"/>
                <Setter Property="Margin" Value="12,5,12,1"/>
                <Setter Property="HorizontalTextAlignment" Value="Start"/>
            </Style>

            <Style x:Key="Indicador" TargetType="ActivityIndicator">
                <Setter Property="Color" Value="Blue"/>
            </Style>

            <Style x:Key="Lista" TargetType="ListView">
                <Setter Property="HasUnevenRows" Value="True"/>
            </Style>

            <Style x:Key="Linea" TargetType="BoxView">
                <Setter Property="HeightRequest" Value="1"/>
                <Setter Property="BackgroundColor" Value="Black"/>
                <Setter Property="HorizontalOptions" Value="FillAndExpand"/>
            </Style>

            <Style x:Key="CajaTexto" TargetType="Entry">
```

```

        <Setter Property="HorizontalOptions" Value="FillAndExpand"/>
        <Setter Property="FontSize" Value="20"/>
        <Setter Property="TextColor" Value="White"/>
        <Setter Property="BackgroundColor" Value="Black"/>
        <Setter Property="FontAttributes" Value="Bold"/>
        <Setter Property="Margin" Value="10,0"/>
        <Setter Property="HorizontalTextAlignment" Value="Start"/>
    </Style>

    <Style x:Key="SliderCalificacion" TargetType="Slider">
        <Setter Property="Minimum" Value="0"/>
        <Setter Property="Maximum" Value="100"/>
        <Setter Property="Margin" Value="10"/>
        <Setter Property="HorizontalOptions" Value="FillAndExpand"/>
    </Style>

    <Style x:Key="FotoAlumno" TargetType="Image">
        <Setter Property="WidthRequest" Value="150"/>
        <Setter Property="HeightRequest" Value="150"/>
        <Setter Property="Aspect" Value="AspectFit"/>
        <Setter Property="HorizontalOptions" Value="Center"/>
    </Style>

    <Style x:Key="SelectorFecha" TargetType="DatePicker">
        <Setter Property="Format" Value="dd/MM/yyyy"/>
        <Setter Property="BackgroundColor" Value="White"/>
        <Setter Property="TextColor" Value="Black"/>
        <Setter Property="Margin" Value="10,0"/>
    </Style>

</ResourceDictionary>

</Application.Resources>
</Application>

```

Paso 12. En la carpeta **Paginas**, agrega las siguientes páginas:

a) **PáginaMenu:** Una página con 3 botones para acceder a las demás páginas.

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"
             x:Class="ProfesorApp.Paginas.PaginaMenu">
    <ContentPage.ToolbarItems>
        <ToolBarItem x:Name="Tareas" Text="Tareas" Order="Primary" Priority="0"
Clicked="Tareas_Clicked"/>
        <ToolBarItem x:Name="Alumnos" Text="Alumnos" Order="Primary" Priority="2"
Clicked="Alumnos_Clicked"/>
        <ToolBarItem x:Name="Respuestas" Text="Respuestas" Order="Primary" Priority="2"
Clicked="Respuestas_Clicked"/>
    </ContentPage.ToolbarItems>
</ContentPage>
```

Código C#:

```
using System;

using Xamarin.Forms;
using Xamarin.Forms.Xaml;

namespace ProfesorApp.Paginas
{
    [XamlCompilation(XamlCompilationOptions.Compile)]
    public partial class PaginaMenu : ContentPage
    {
        public PaginaMenu()
        {
            InitializeComponent();
        }

        private async void Tareas_Clicked(object sender, EventArgs e)
        {
            await Navigation.PushAsync(new PaginalistaTareas());
        }

        private async void Alumnos_Clicked(object sender, EventArgs e)
        {
            await Navigation.PushAsync(new PaginalistaAlumnos());
        }

        private async void Respuestas_Clicked(object sender, EventArgs e)
        {
            await Navigation.PushAsync(new PaginalistaTareasAlumnos());
        }
    }
}
```

- b) **PaginaListaTareas:** Esta página muestra las tareas que han sido publicadas por el profesor en una lista. Permite navegar hacia otra página para agregar, modificar, eliminar o ver el detalle de una tarea específica.

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"
              x:Class="ProfesorApp.Paginas.PaginaListaTareas">
    <ScrollView>
        <StackLayout>
            <ActivityIndicator x:Name="activityIndicator" Style="{StaticResource
Indicador}"/>
            <ListView x:Name="lsvTareas" ItemSelected="lsvTareas_ItemSelected"
Style="{StaticResource Lista}">
                <ListView.ItemTemplate>
                    <DataTemplate>
                        <TextCell Text="{Binding Titulo}" TextColor="Blue" />
                    </DataTemplate>
                </ListView.ItemTemplate>
            </ListView>
        </StackLayout>
    </ScrollView>
    <ContentPage.ToolbarItems>
        <ToolbarItem x:Name="Agregar" Text="Agregar" Order="Primary" Priority="0"
Clicked="Agregar_Clicked"/>
    </ContentPage.ToolbarItems>
</ContentPage>
```

Código C#:

```
using System;
using ProfesorApp.Servicios;
using ProfesorApp.Clases;
using Xamarin.Forms;
using Xamarin.Forms.Xaml;

namespace ProfesorApp.Paginas
{
    [XamlCompilation(XamlCompilationOptions.Compile)]
    public partial class PaginaListaTareas : ContentPage
    {
        public PaginaListaTareas()
        {
            InitializeComponent();
        }

        private void ActualizarActivityIndicator(bool estado)
        {
            activityIndicator.IsRunning = estado;
            activityIndicator.IsEnabled = estado;
            activityIndicator.IsVisible = estado;
        }

        protected async override void OnAppearing()
        {
            base.OnAppearing();

            ActualizarActivityIndicator(true);
            lsvTareas.ItemsSource = await ServicioWebApi.GetTareas();
            ActualizarActivityIndicator(false);
        }

        private async void lsvTareas_ItemSelected(object sender,
SelectedItemChangedEventArgs e)
        {
            try
            {
                Tarea dato = (Tarea)e.SelectedItem;
                await Navigation.PushAsync(new PaginaDetalleTarea(dato));
            }
            catch (Exception ex)
            {
            }
        }

        private async void Agregar_Clicked(object sender, EventArgs e)
        {
            await Navigation.PushAsync(new PaginaDetalleTarea(new Tarea()));
        }
    }
}
```

- c) **PaginaDetalleTarea:** Esta página permite ver el detalle de una tarea seleccionada, así como agregarla, modificarla o eliminarla.

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:Convertidor="clr-namespace:ProfesorApp.Converters"
    x:Class="ProfesorApp.Paginas.PaginaDetalleTarea">

    <StackLayout Padding="10" Spacing="10" BackgroundColor="White">
        <Label Text="Titulo:" Style="{StaticResource LabelTitulo}"/>
        <Entry Text="{Binding Titulo}" Style="{StaticResource CajaTexto}"/>

        <Label Text="Fecha límite:" Style="{StaticResource LabelTitulo}"/>
        <DatePicker Date="{Binding FechaLimite}" Style="{StaticResource SelectorFecha}"/>

        <Label Text="Fecha publicación:" Style="{StaticResource LabelTitulo}"/>
        <Label Text="{Binding FechaPublicacion, Converter={StaticResource
ConvertidorFecha}}" Style="{StaticResource LabelNormal}"/>

        <ActivityIndicator x:Name="activityIndicator" Style="{StaticResource
Indicador}"/>
    </StackLayout>

    <ContentPage.ToolbarItems>
        <ToolbarItem x:Name="Archivo" Text="Archivo" Order="Primary" Priority="0"
Clicked="Archivo_Clicked"/>
        <ToolbarItem x:Name="Guardar" Text="Guardar" Order="Primary" Priority="1"
Clicked="Guardar_Clicked"/>
        <ToolbarItem x:Name="Eliminar" Text="Eliminar" Order="Primary" Priority="2"
Clicked="Eliminar_Clicked"/>
        <ToolbarItem x:Name="Ver" Text="Archivo" Order="Primary" Priority="3"
Clicked="Ver_Clicked"/>
    </ContentPage.ToolbarItems>
</ContentPage>
```

Código C#:

```
using System;
using ProfesorApp.Servicios;
using ProfesorApp.Clases;
using Xamarin.Forms;
using Xamarin.Forms.Xaml;
using System.IO;

namespace ProfesorApp.Paginas
{
    [XamlCompilation(XamlCompilationOptions.Compile)]
    public partial class PaginaDetalleTarea : ContentPage
    {
        Tarea dato;
        MemoryStream stream;

        public PaginaDetalleTarea(Tarea dato)
        {
            InitializeComponent();

            ActualizarActivityIndicator(true);

            this.dato = dato;
            this.BindingContext = dato;

            ActualizarActivityIndicator(false);
        }

        private void ActualizarActivityIndicator(bool estado)
        {
            activityIndicator.IsRunning = estado;
            activityIndicator.IsEnabled = estado;
            activityIndicator.IsVisible = estado;
        }

        private async void Archivo_Clicked(object sender, EventArgs e)
        {
            ServicioFilePicker servicioFilePicker = new ServicioFilePicker();
            stream = await servicioFilePicker.GetFile();
        }

        private async void Guardar_Clicked(object sender, EventArgs e)
        {
            if (stream != null || dato.Id > 0)
            {
                ActualizarActivityIndicator(true);

                if (dato.Id == 0)
                    dato = await ServicioWebApi.AddTarea(dato);

                if (stream != null)
                {
                    var servicioStorage = new ServicioStorage();
                    dato.ArchivoURL = await servicioStorage.UploadTarea(dato.Id, stream);
                }

                await ServicioWebApi.UpdateTarea(dato);
            }
        }
    }
}
```



```

        ActualizarActivityIndicator(false);

        await DisplayAlert("Información", "Dato registrado con éxito", "OK");
        await Navigation.PopAsync();
    }
    else
    {
        await DisplayAlert("Información", "Debes agregar un archivo primero",
"OK");
    }
}

private async void Eliminar_Clicked(object sender, EventArgs e)
{
    if (dato.Id > 0)
    {
        if (await DisplayAlert("Eliminar", "¿Deseas eliminar el registro?", "Si",
"No"))
        {
            ActualizarActivityIndicator(true);
            await ServicioWebApi.DeleteTarea(dato.Id);
            ActualizarActivityIndicator(false);

            await DisplayAlert("Información", "Dato eliminado con éxito", "OK");
            await Navigation.PopAsync();
        }
    }
}

private void Ver_Clicked(object sender, EventArgs e)
{
    if (dato.Id > 0)
    {
        var servicioStorage = new ServicioStorage();
        Device.OpenUri(new
Uri(servicioStorage.GetFullDownloadTareaURL(dato.Id)));
        //var stream = await servicioStorage.DownloadTarea(dato.Id);
    }
}
}
}

```

- d) **PaginaListaAlumnos:** Esta página muestra los alumnos registrados por el profesor. También permite navegar a otra página para ver el detalle de un registro específico, así como agregar, modificar o eliminar registros.

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"
              x:Class="ProfesorApp.Paginas.PaginaListaAlumnos">
    <ScrollView>
        <StackLayout>
            <ActivityIndicator x:Name="activityIndicator" Style="{StaticResource
Indicador}"/>
            <ListView x:Name="lsvAlumnos" ItemSelected="lsvAlumnos_ItemSelected"
Style="{StaticResource Lista}">
                <ListView.ItemTemplate>
                    <DataTemplate>
                        <ImageCell ImageSource="{Binding FotoURLSAS}" Text="{Binding
Nombre}" TextColor="Blue"/>
                    </DataTemplate>
                </ListView.ItemTemplate>
            </ListView>
        </StackLayout>
    </ScrollView>
    <ContentPage.ToolbarItems>
        <ToolbarItem x:Name="Agregar" Text="Agregar" Order="Primary" Priority="0"
Clicked="Agregar_Clicked"/>
    </ContentPage.ToolbarItems>
</ContentPage>
```

Código C#:

```
using System;
using ProfesorApp.Servicios;
using ProfesorApp.Clases;
using Xamarin.Forms;
using Xamarin.Forms.Xaml;

namespace ProfesorApp.Paginas
{
    [XamlCompilation(XamlCompilationOptions.Compile)]
    public partial class PaginaListaAlumnos : ContentPage
    {
        public PaginaListaAlumnos()
        {
            InitializeComponent();
        }

        private void ActualizarActivityIndicator(bool estado)
        {
            activityIndicator.IsRunning = estado;
            activityIndicator.IsEnabled = estado;
            activityIndicator.IsVisible = estado;
        }

        protected async override void OnAppearing()
        {
            base.OnAppearing();

            ActualizarActivityIndicator(true);
            lsvAlumnos.ItemsSource = await ServicioWebApi.GetAlumnos();
            ActualizarActivityIndicator(false);
        }

        private async void lsvAlumnos_ItemSelected(object sender,
SelectedItemChangedEventArgs e)
        {
            try
            {
                Alumno dato = (Alumno)e.SelectedItem;
                await Navigation.PushAsync(new PaginaDetalleAlumno(dato));
            }
            catch (Exception ex)
            {
            }
        }

        private async void Agregar_Clicked(object sender, EventArgs e)
        {
            await Navigation.PushAsync(new PaginaDetalleAlumno(new Alumno()));
        }
    }
}
```

- e) **PaginaDetalleAlumno:** Permite agregar, modificar, eliminar o ver el detalle de un alumno específico.

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"
              x:Class="ProfesorApp.Paginas.PaginaDetalleAlumno">
    <StackLayout Padding="10" Spacing="10" BackgroundColor="White">
        <Image Source="{Binding FotoURLSAS}" Style="{StaticResource FotoAlumno}"/>

        <Label Text="Nombre:" Style="{StaticResource LabelTitulo}"/>
        <Entry Text="{Binding Nombre}" Style="{StaticResource CajaTexto}"/>

        <Label Text="Usuario:" Style="{StaticResource LabelTitulo}"/>
        <Entry Text="{Binding Usuario}" Style="{StaticResource CajaTexto}"/>

        <Label Text="Password:" Style="{StaticResource LabelTitulo}"/>
        <Entry Text="{Binding Password}" IsPassword="True" Style="{StaticResource
CajaTexto}"/>

        <ActivityIndicator x:Name="activityIndicator" Style="{StaticResource
Indicador}"/>
    </StackLayout>

    <ContentPage.ToolbarItems>
        <ToolbarItem x:Name="Guardar" Text="Guardar" Order="Primary" Priority="0"
Clicked="Guardar_Clicked"/>
        <ToolbarItem x:Name="Eliminar" Text="Eliminar" Order="Primary" Priority="1"
Clicked="Eliminar_Clicked"/>
    </ContentPage.ToolbarItems>
</ContentPage>
```

Código C#:

```
using System;
using ProfesorApp.Servicios;
using ProfesorApp.Clases;
using Xamarin.Forms;
using Xamarin.Forms.Xaml;

namespace ProfesorApp.Paginas
{
    [XamlCompilation(XamlCompilationOptions.Compile)]
    public partial class PaginaDetalleAlumno : ContentPage
    {
        Alumno dato;

        public PaginaDetalleAlumno(Alumno dato)
        {
            InitializeComponent();

            ActualizarActivityIndicator(true);
        }
    }
}
```

```

        this.dato = dato;
        this.BindingContext = dato;

        ActualizarActivityIndicator(false);
    }

    private void ActualizarActivityIndicator(bool estado)
    {
        activityIndicator.IsRunning = estado;
        activityIndicator.IsEnabled = estado;
        activityIndicator.IsVisible = estado;
    }

    private async void Guardar_Clicked(object sender, EventArgs e)
    {
        ActualizarActivityIndicator(true);

        if (dato.Id == 0)
        {
            dato = await ServicioWebApi.AddAlumno(dato);
        }
        else
        {
            await ServicioWebApi.UpdateAlumno(dato);
        }

        ActualizarActivityIndicator(false);

        await DisplayAlert("Información", "Dato registrado con éxito", "OK");
        await Navigation.PopAsync();
    }

    private async void Eliminar_Clicked(object sender, EventArgs e)
    {
        if (dato.Id > 0)
        {
            if (await DisplayAlert("Eliminar", "¿Deseas eliminar el registro?", "Si",
"No"))
            {
                ActualizarActivityIndicator(true);
                await ServicioWebApi.DeleteAlumno(dato.Id);
                ActualizarActivityIndicator(false);

                await DisplayAlert("Información", "Dato eliminado con éxito", "OK");
                await Navigation.PopAsync();
            }
        }
    }
}

```

- f) **PaginaListaTareasAlumnos:** Muestra la lista de tareas registradas por los alumnos, con la posibilidad de seleccionar una y calificarla posteriormente.

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"
    x:Class="ProfesorApp.Paginas.PaginaListaTareasAlumnos">
    <ScrollView>
        <StackLayout>
            <ActivityIndicator x:Name="activityIndicator" Color="Blue"/>
            <ListView x:Name="lsvTareasAlumnos"
                ItemSelected="lsvTareasAlumnos_ItemSelected" HasUnevenRows="True">
                <ListView.ItemTemplate>
                    <DataTemplate>
                        <ViewCell>
                            <StackLayout BackgroundColor="White" Spacing="5" Padding="5">
                                <Label Text="{Binding MensajeCorto}"
                                    LineBreakMode="WordWrap" FontSize="15" TextColor="#030303" HorizontalOptions="Start"
                                    Margin="12,5,12,1" HorizontalTextAlignment="Start"/>
                                <Label Text="{Binding FechaRespuestaDate}"
                                    LineBreakMode="WordWrap" FontSize="15" TextColor="#030303" HorizontalOptions="Start"
                                    Margin="12,5,12,1" HorizontalTextAlignment="Start"/>
                                <StackLayout Orientation="Horizontal">
                                    <Label Text="{Binding Calificacion}"
                                        LineBreakMode="WordWrap" FontSize="15" TextColor="#030303" HorizontalOptions="Start"
                                        Margin="12,5,12,1" HorizontalTextAlignment="Start"/>
                                    <Label Text="{Binding EvaluadoString}"
                                        LineBreakMode="WordWrap" FontSize="15" TextColor="#030303" HorizontalOptions="Start"
                                        Margin="12,5,12,1" HorizontalTextAlignment="Start"/>
                                </StackLayout>
                            </StackLayout>
                        </ViewCell>
                    </DataTemplate>
                </ListView.ItemTemplate>
            </ListView>
        </StackLayout>
    </ScrollView>
</ContentPage>
```

Código C#:

```
using System;
using ProfesorApp.Servicios;
using ProfesorApp.Clases;
using Xamarin.Forms;
using Xamarin.Forms.Xaml;
using System.Threading.Tasks;

namespace ProfesorApp.Paginas
{
    [XamlCompilation(XamlCompilationOptions.Compile)]
    public partial class PaginaListaTareasAlumnos : ContentPage
    {
        public PaginaListaTareasAlumnos()
        {
            InitializeComponent();

            private void ActualizarActivityIndicator(bool estado)
            {
                activityIndicator.IsRunning = estado;
                activityIndicator.IsEnabled = estado;
                activityIndicator.IsVisible = estado;
            }

            private async Task ObtenerTareasAlumnos()
            {
                ActualizarActivityIndicator(true);
                lsvTareasAlumnos.ItemsSource = await
ServicioWebApi.GetTareaAlumnosByEval(switchTareaEvaluada.IsToggled);
                ActualizarActivityIndicator(false);
            }

            private async void switchTareaEvaluada_Toggled(object sender, ToggledEventArgs e)
            {
                await ObtenerTareasAlumnos();
            }

            protected async override void OnAppearing()
            {
                base.OnAppearing();
                await ObtenerTareasAlumnos();
            }

            private async void lsvTareasAlumnos_ItemSelected(object sender,
SelectedItemChangedEventArgs e)
            {
                try
                {
                    TareaAlumno dato = (TareaAlumno)e.SelectedItem;
                    await Navigation.PushAsync(new PaginaCalificarTareaAlumno(dato));
                }
                catch (Exception ex) { }
            }
        }
    }
}
```

- g) **PaginaCalificarTareaAlumno:** Esta página permite ver el detalle de una tarea subida por el alumno, así como evaluarla asignando una calificación.

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"
    x:Class="ProfesorApp.Paginas.PaginaListaTareasAlumnos">
    <ScrollView>
        <StackLayout>
            <StackLayout Orientation="Horizontal">
                <Label Text="Mostrar tareas evaluadas" Style="{StaticResource
LabelTitulo}" />
                <Switch x:Name="switchTareaEvaluada"
Toggled="switchTareaEvaluada_Toggled" IsToggled="False" />
            </StackLayout>

            <ActivityIndicator x:Name="activityIndicator" Style="{StaticResource
Indicador}" />
            <ListView x:Name="lsvTareasAlumnos"
ItemSelected="lsvTareasAlumnos_ItemSelected" Style="{StaticResource Lista}">
                <ListView.ItemTemplate>
                    <DataTemplate>
                        <ViewCell>
                            <StackLayout BackgroundColor="White" Spacing="5" Padding="5">
                                <Label Text="{Binding Mensaje, Converter={StaticResource
ConvertidorMensajeCorto}}" Style="{StaticResource LabelDetalle}" />
                                <Label Text="{Binding Fecha, Converter={StaticResource
ConvertidorFecha}}" Style="{StaticResource LabelDetalle}" />
                                <StackLayout Orientation="Horizontal">
                                    <Label Text="{Binding Calificacion}"
Style="{StaticResource LabelDetalle}" />
                                    <Label Text="{Binding Evaluado,
Converter={StaticResource ConvertidorEvaluacionTarea}}" Style="{StaticResource
LabelDetalle}" />
                                </StackLayout>
                            </StackLayout>
                        </ViewCell>
                    </DataTemplate>
                </ListView.ItemTemplate>
            </ListView>
        </StackLayout>
    </ScrollView>
</ContentPage>
```


Código C#:

```
using System;
using ProfesorApp.Servicios;
using ProfesorApp.Clases;
using Xamarin.Forms;
using Xamarin.Forms.Xaml;

namespace ProfesorApp.Paginas
{
    [XamlCompilation(XamlCompilationOptions.Compile)]
    public partial class PaginaCalificarTareaAlumno : ContentPage
    {
        TareaAlumno dato;

        public PaginaCalificarTareaAlumno(TareaAlumno dato)
        {
            InitializeComponent();

            this.dato = dato;
        }

        protected async override void OnAppearing()
        {
            base.OnAppearing();

            ActualizarActivityIndicator(true);
            dato = await ServicioWebApi.GetTareaAlumno(dato.IdTarea, dato.IdAlumno);
            this.BindingContext = dato;

            ActualizarActivityIndicator(false);
        }

        private void ActualizarActivityIndicator(bool estado)
        {
            activityIndicator.IsRunning = estado;
            activityIndicator.IsEnabled = estado;
            activityIndicator.IsVisible = estado;
        }

        private async void Calificar_Clicked(object sender, EventArgs e)
        {
            ActualizarActivityIndicator(true);

            dato.Evaluado = true;
            await ServicioWebApi.UpdateTareaAlumno(dato);
            ActualizarActivityIndicator(false);

            await DisplayAlert("Información", "Dato registrado con éxito", "OK");
            await Navigation.PopAsync();
        }

        private void VerTarea_Clicked(object sender, EventArgs e)
        {
            var servicioStorage = new ServicioStorage();
            Device.OpenUri(new
Uri(servicioStorage.GetFullDownloadTareaURL(dato.IdTarea)));
        }
    }
}
```

```

private void VerRespuesta_Clicked(object sender, EventArgs e)
{
    var servicioStorage = new ServicioStorage();
    Device.OpenUri(new
Uri(servicioStorage.GetFullDownloadTareaAlumnoURL(dato.IdTarea, dato.IdAlumno)));
}
}

```

Paso 13. Modifica App.xaml.cs para establecer la página de inicio:

```

public App()
{
    InitializeComponent();

    MainPage = new NavigationPage(new Paginas.PaginaMenu());
}

```

Paso 14. Compila y ejecuta la app

