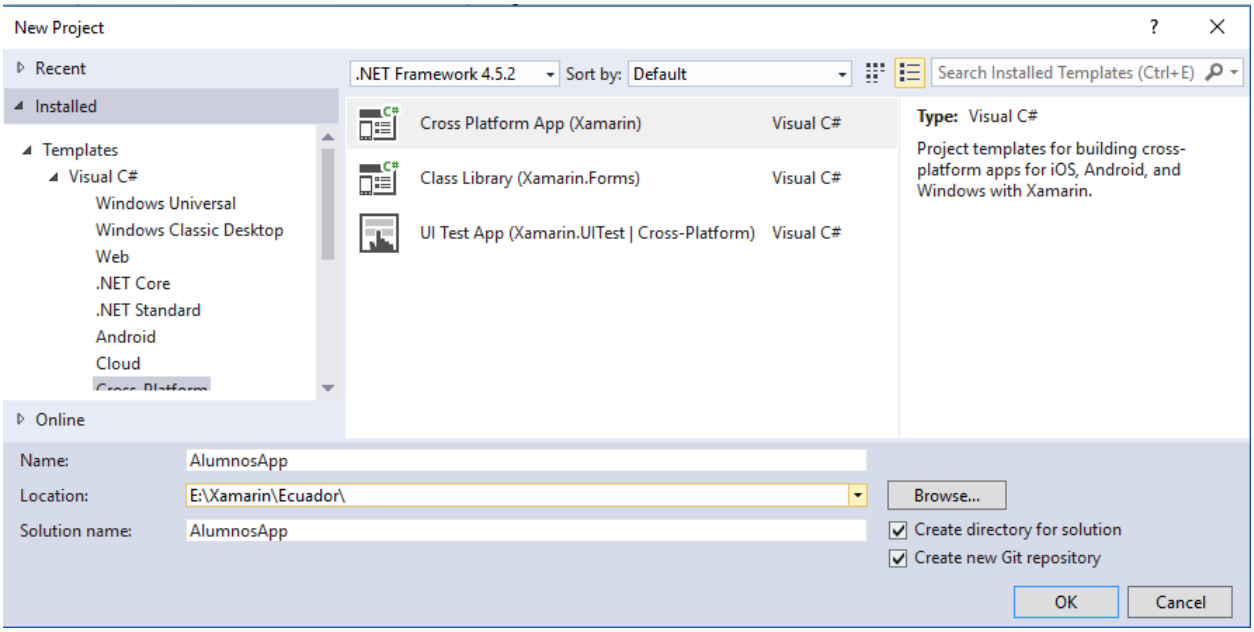
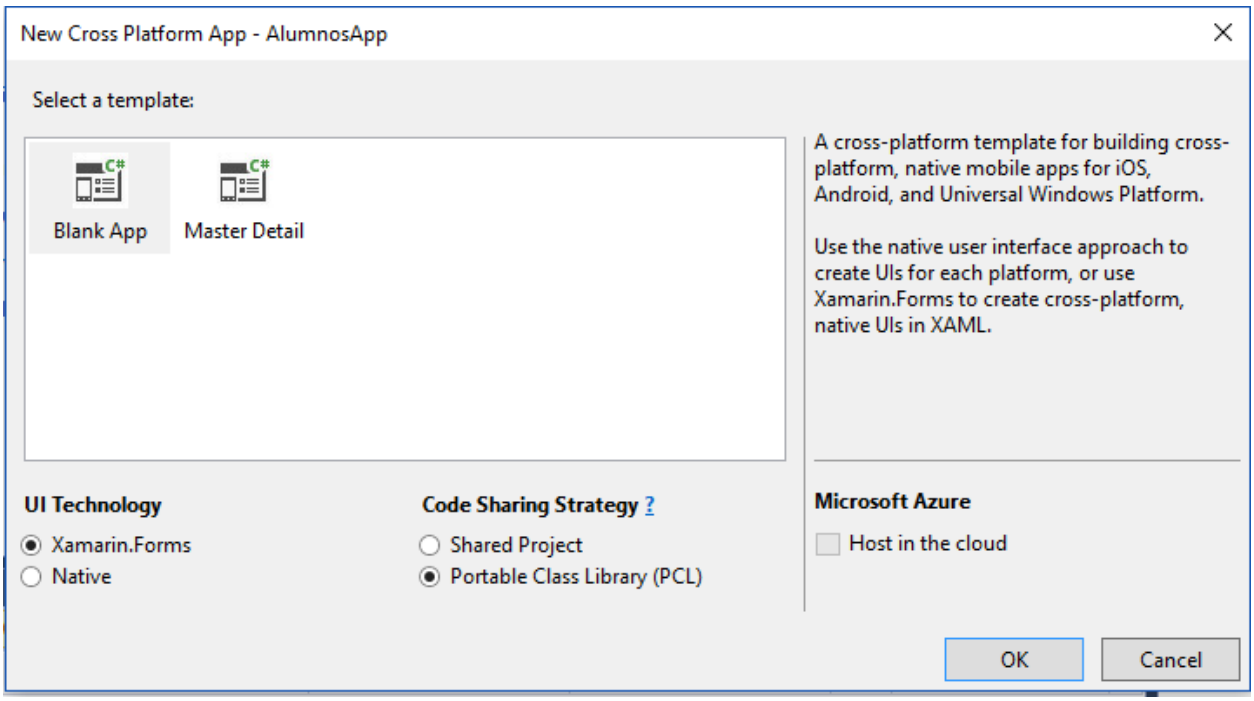
Parte 5: Creación de la app móvil AlumnosApp – Autor: Luis Beltrán

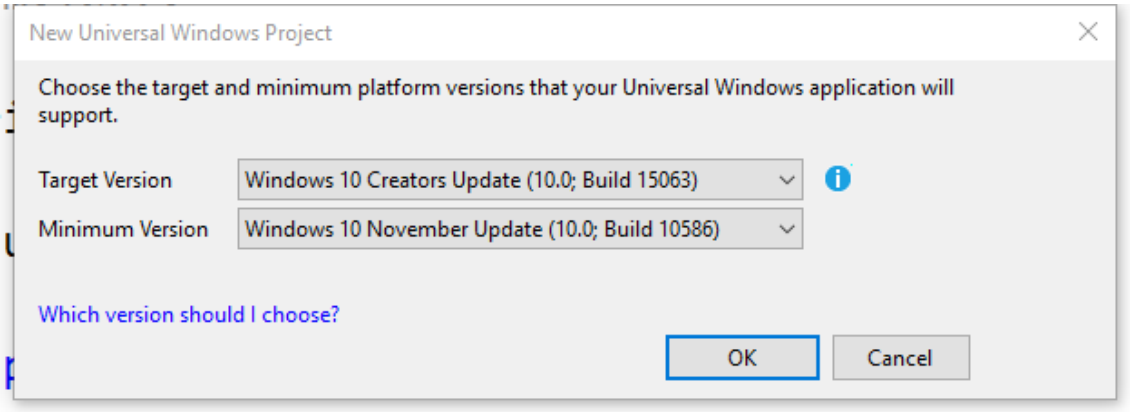
Paso 1. Crea un nuevo proyecto de la categoría **Cross-Platform** selecciona **Aplicación multiplataforma (Xamarin.Forms o nativa)** y coloca el nombre de proyecto **AlumnosApp**. Además, la ruta del proyecto debe ser una ubicación corta para evitar problemas de ruta larga.



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**.

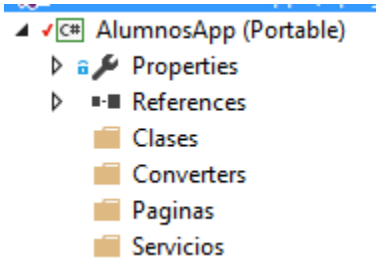


Paso 3. Si tienes instalado el SDK de Windows 10, aparecerá la ventana de selección del Target y Minimum Version. Selecciónalas a conveniencia, según la versión que tengas instalada.



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

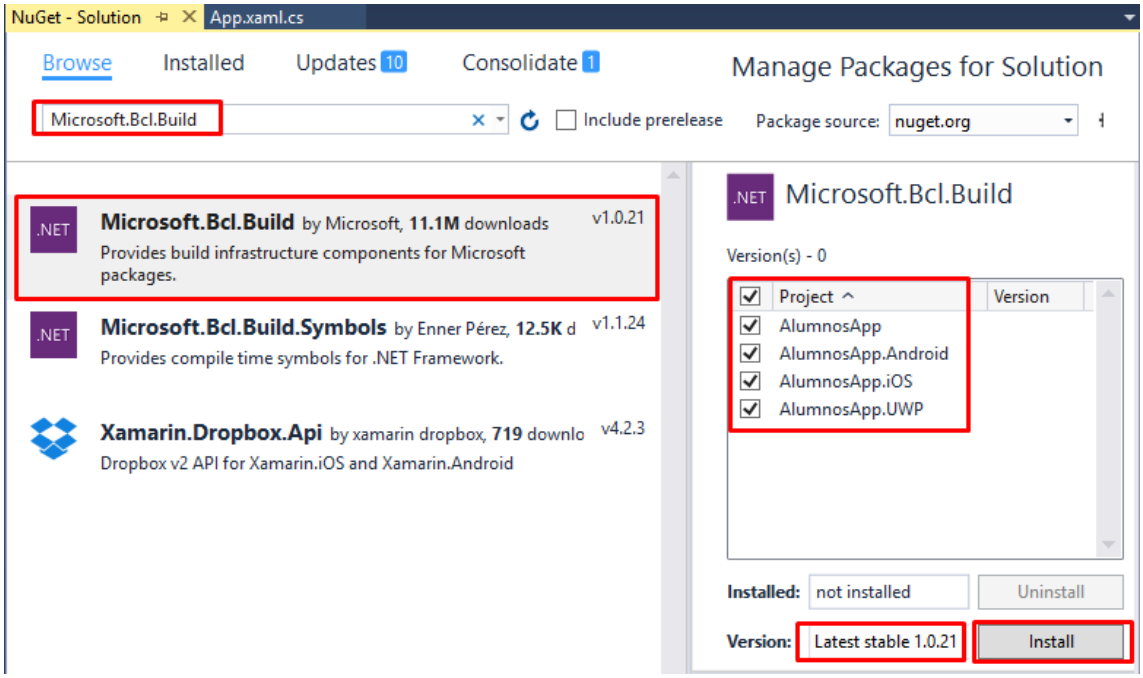
Paso 5. Agrega las carpetas **Clases, Converters, Paginas y Servicios** al proyecto



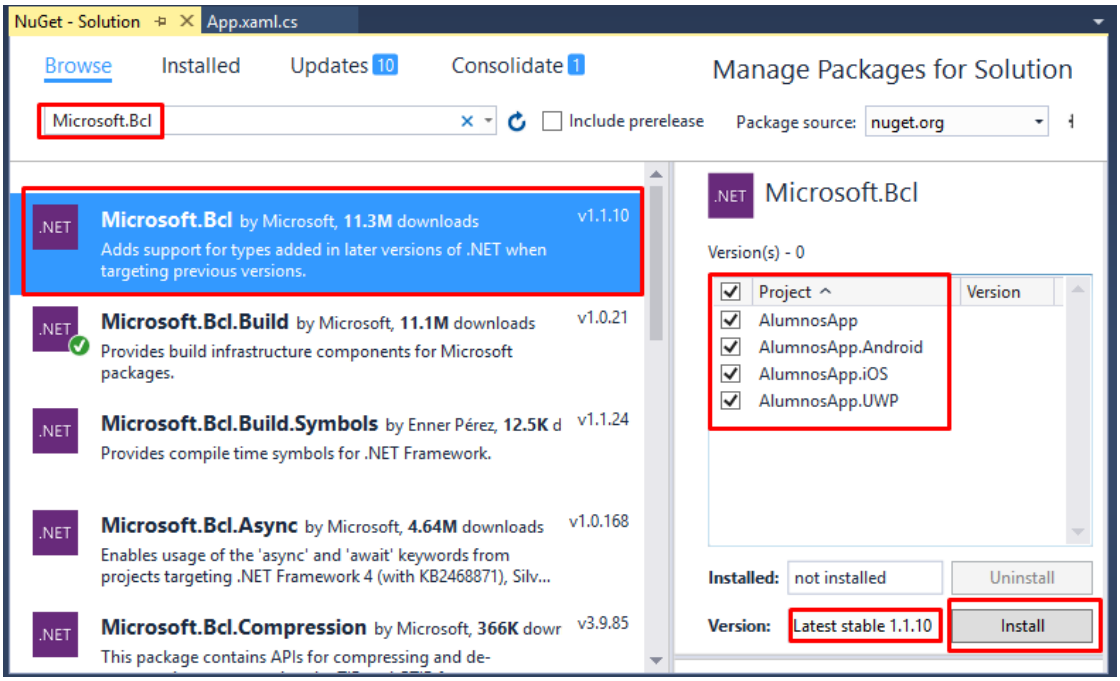
Paso 6. Da clic en **Administrar paquetes Nuget para la solución** en el menú contextual de la solución.

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

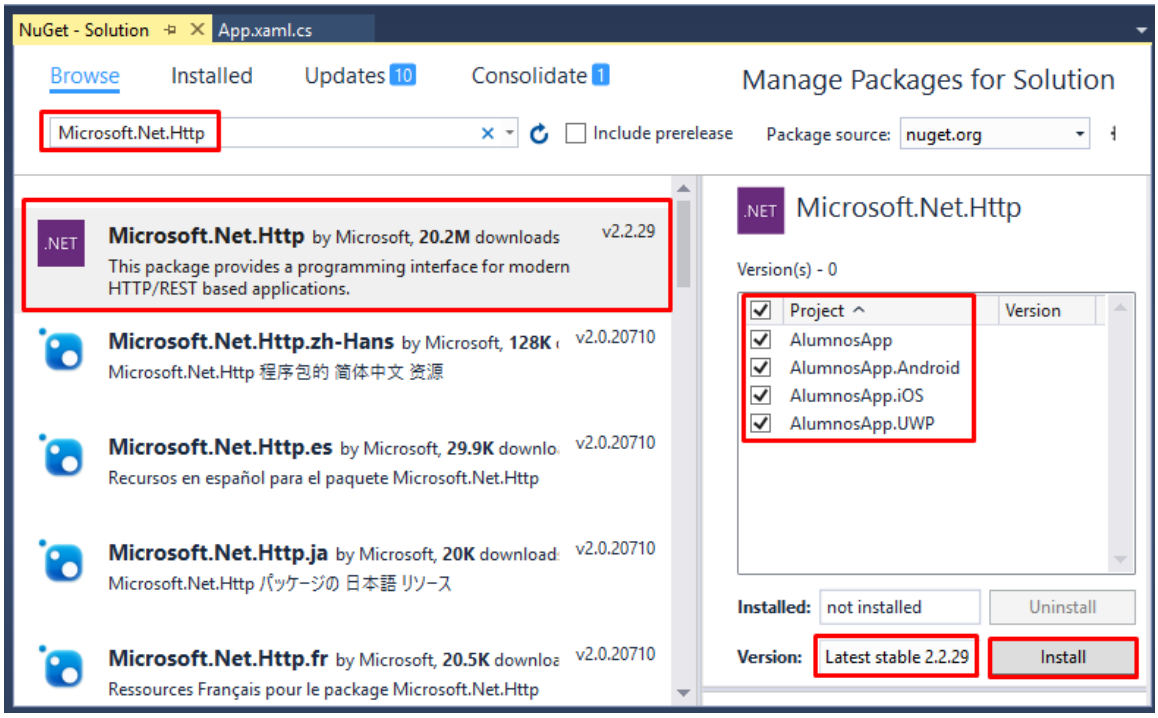
a) **Microsoft.Bcl.Build** (requisito de Microsoft.Net.Http)



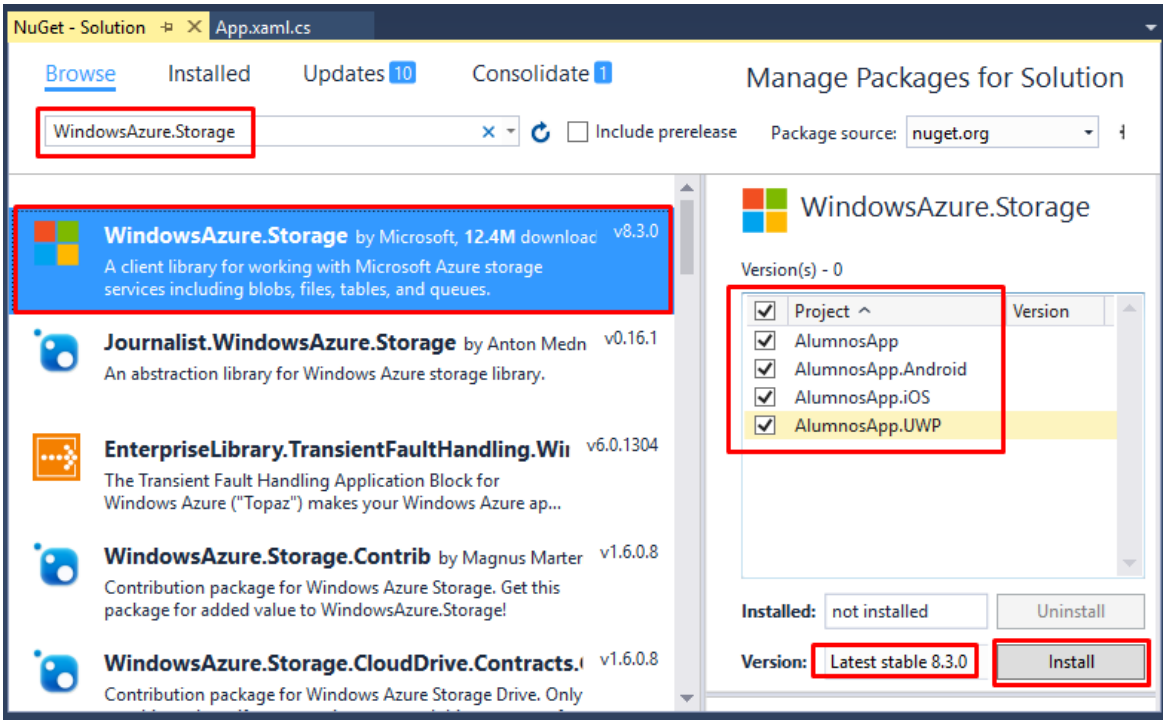
b) **Microsoft.Bcl** (requisito de Microsoft.Net.Http)



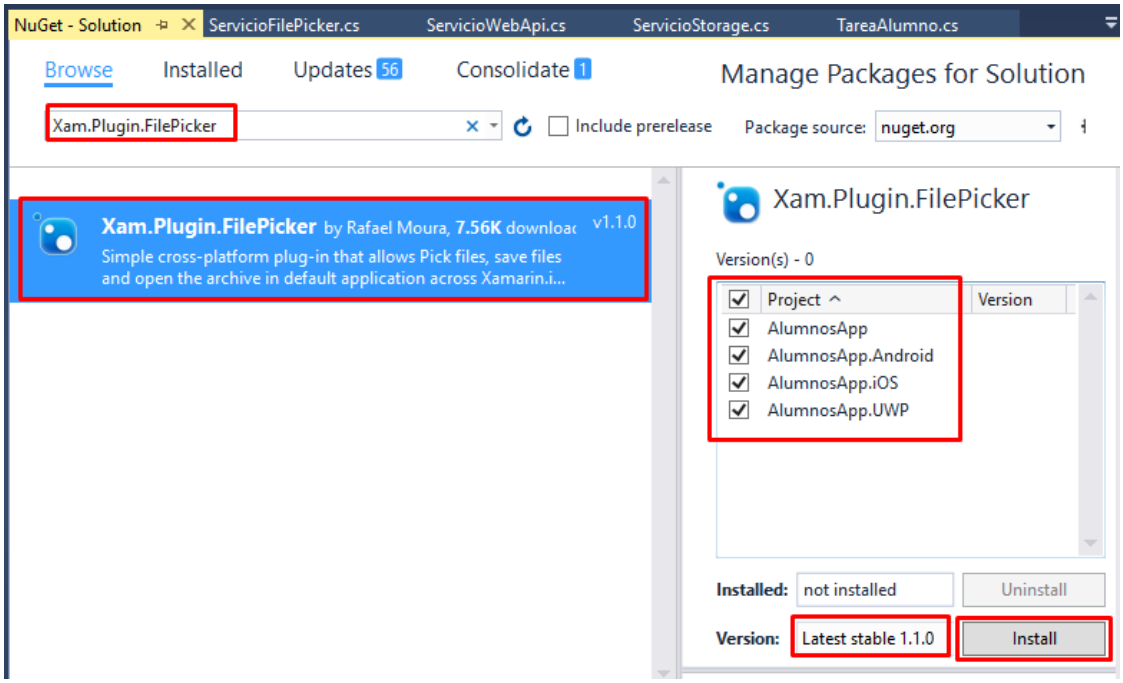
c) **Microsoft.Net.Http** (requisito de para las peticiones al servicio web)



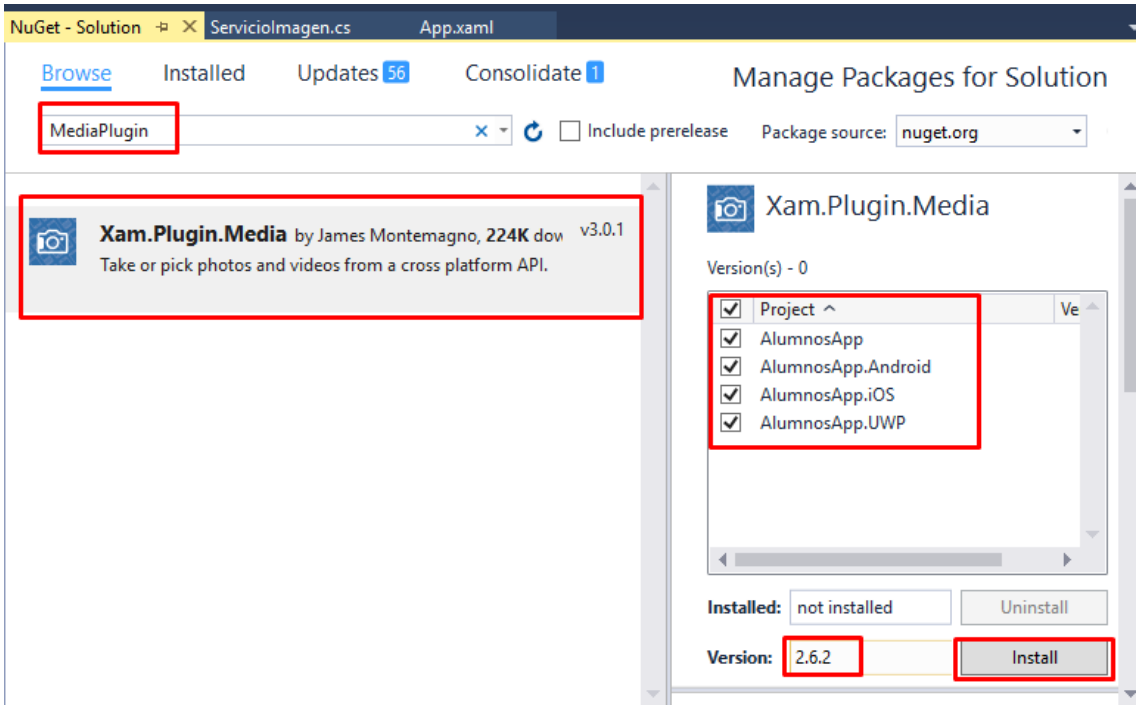
d) **WindowsAzure.Storage** (para la conexión con el Azure Storage)



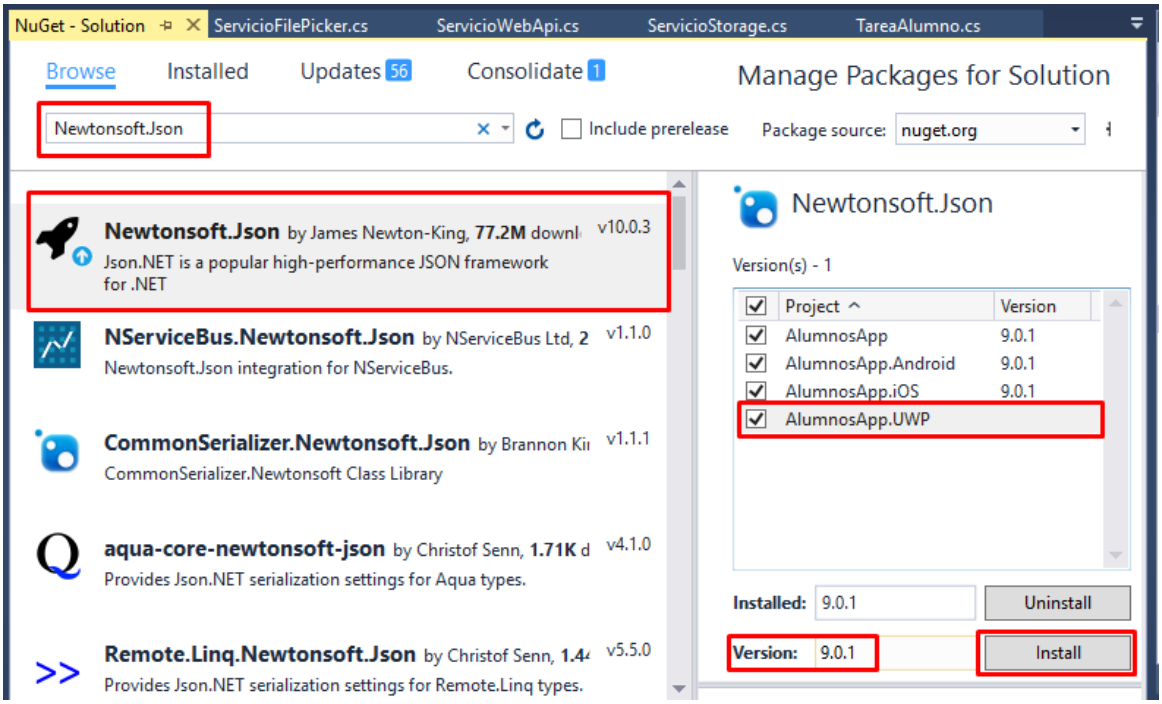
e) **Xam.Plugin.FilePicker** (para que el usuario pueda seleccionar un archivo de su dispositivo)



f) **Xam.Plugin.Media:** Permite al usuario seleccionar una fotografía de la carpeta pública de imágenes del dispositivo o tomar una fotografía en tiempo real.



g) Newtonsoft.Json (actualizar el paquete de UWP)



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 AlumnosApp.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 AlumnosApp.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 AlumnosApp.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 AlumnosApp.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> UploadTareaAlumno(int idTarea, int idAlumno, Stream
stream)
        {
            string blobSAS =
$"{StorageURL}/{ContainerTareaAlumno}/{idTarea}_{idAlumno}.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}.jpg{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 algunas diferentes tablas.

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

namespace AlumnosApp.Servicios
{
    public static class ServicioWebApi
    {
        const string WebApiURL = "actualizar-valor";

        private static HttpClient Cliente = new HttpClient() { BaseAddress = new
Uri(WebApiURL) };

        public static async Task<Alumno> GetAlumnoByCredentials(string usuario, string
password)
        {
            Alumno dato = null;
            Cliente.DefaultRequestHeaders.Accept.Clear();
            Cliente.DefaultRequestHeaders.Accept.Add(new
MediaTypeWithQualityHeaderValue("application/json"));

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

            if (respuesta.StatusCode == HttpStatusCode.OK)
            {
                var json = await respuesta.Content.ReadAsStringAsync();

                if (!string.IsNullOrEmpty(json))
                {
                    dato = JsonConvert.DeserializeObject<Alumno>(json);
                    dato.FotoURLSAS = new
ServicioStorage().GetFullDownloadAlumnoURL(dato.Id);
                }
            }

            return dato;
        }

        public static async Task<Alumno> GetAlumno(int id)
        {
            Alumno dato = null;
            Cliente.DefaultRequestHeaders.Accept.Clear();
            Cliente.DefaultRequestHeaders.Accept.Add(new
MediaTypeWithQualityHeaderValue("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);
                dato.FotoURLSAS = new
ServicioStorage().GetFullDownloadAlumnoURL(dato.Id);
            }

            return dato;
        }

        public static async Task UpdateAlumno(Alumno info, MemoryStream stream)
        {
            if (stream != null)
            {
                var servicioStorage = new ServicioStorage();
                info.FotoURL = await servicioStorage.UploadAlumno(info.Id, stream);
            }

            Cliente.DefaultRequestHeaders.Accept.Clear();
            Cliente.DefaultRequestHeaders.Accept.Add(new
MediaTypeWithQualityHeaderValue("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"));
        }

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

            //var url =
$"{WebApiURL}/api/Tareas/GetTareasNoRealizadasByAlumno/{idAlumno}";
            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<List<TareaAlumno>> GetTareasRealizadasByAlumno(int
idAlumno, 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 url =
$"{WebApiURL}/api/TareaAlumnos/GetTareasRealizadasByAlumno/{idAlumno}/{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,
MemoryStream stream)
        {
            try
            {

                if (stream != null)
                {
                    var servicioStorage = new ServicioStorage();
                    info.ArchivoURL = await
servicioStorage.UploadTareaAlumno(info.IdTarea, info.IdAlumno, stream);
                }

                info.Alumno = null;
                info.Tarea = null;

                TareaAlumno dato = null;
                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;
            }
            catch(Exception ex)
            {
                return null;
            }
        }

        public static async Task UpdateTareaAlumno(TareaAlumno info, MemoryStream stream)
        {
            if (stream != null)
            {
                var servicioStorage = new ServicioStorage();
                info.ArchivoURL = await servicioStorage.UploadTareaAlumno(info.IdTarea,
info.IdAlumno, stream);
            }

            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"));
        }

        public static async Task<bool> DeleteTareaAlumno(int idTarea, int idAlumno)
        {
            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 AlumnosApp.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; }
        }
    }
}
```

d) **ServicioImagen:** Esta clase utiliza el plugin **MediaPlugin** para permitir al usuario seleccionar una fotografía de la biblioteca o tomarla directamente de la cámara:

```
using Plugin.Media;
using Plugin.Media.Abstractions;
using System;
using System.Threading.Tasks;

namespace AlumnosApp.Servicios
{
    public static class ServicioImagen
    {
        public static async Task<MediaFile> TomarFoto(bool usarCamara)
        {
            await CrossMedia.Current.Initialize();

            if (usarCamara)
                if (!CrossMedia.Current.IsCameraAvailable ||
!CrossMedia.Current.IsTakePhotoSupported)
                    return null;

            var file = usarCamara
                ? await CrossMedia.Current.TakePhotoAsync(StoreCameraMediaOptions {
Directory = "Alumno", Name = $"{DateTime.UtcNow}.jpg" })
                : await CrossMedia.Current.PickPhotoAsync();
            return file;
        }
    }
}
```

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

        </ResourceDictionary>

    </Application.Resources>
</Application>
```

Paso 12. Modifica App.xaml.cs para establecer la página de inicio y un elemento Alumno estático global, accesible desde toda la aplicación:

```
using AlumnosApp.Clases;
using Xamarin.Forms;

namespace AlumnosApp
{
    public partial class App : Application
    {
        public static Alumno Alumno;

        public App()
        {
            InitializeComponent();

            MainPage = new NavigationPage (new AlumnosApp.Paginas.PaginaLogin());
        }
    }
}
```

Paso 13. En la carpeta **Paginas**, agrega las siguientes páginas:

a) **PaginaLogin:** Es una página de inicio para que el alumno pueda ingresar con sus credenciales.

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="AlumnosApp.Paginas.PaginaLogin">
    <StackLayout VerticalOptions="Center">
        <Label Text="Usuario: " Style="{StaticResource LabelTitulo}" Margin="10,0,10,5"/>
        <Entry x:Name="txtUsuario" Style="{StaticResource CajaTexto}"
Margin="10,5,10,10"/>

        <Label Text="Password: " Style="{StaticResource LabelTitulo}"
Margin="10,10,10,5"/>
        <Entry x:Name="txtPassword" IsPassword="True" Style="{StaticResource CajaTexto}"
Margin="10,5,10,10"/>

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

    <ContentPage.ToolbarItems>
        <ToolbarItem x:Name="Login" Text="Iniciar sesión" Order="Primary" Priority="0"
Clicked="Login_Clicked"/>
    </ContentPage.ToolbarItems>
</ContentPage>
```

Código C#

```
using System;
using AlumnosApp.Servicios;
using Xamarin.Forms;
using Xamarin.Forms.Xaml;

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

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

            ActualizarActivityIndicator(false);
        }

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

        private async void Login_Clicked(object sender, EventArgs e)
        {
            ActualizarActivityIndicator(true);
            App.Alumno = await ServicioWebApi.GetAlumnoByCredentials(txtUsuario.Text,
txtPassword.Text);
            ActualizarActivityIndicator(false);

            if (App.Alumno != null)
                await Navigation.PushAsync(new PaginaMenu());
            else
                await DisplayAlert("Error", "Alumno no encontrado. Revisa usuario y
contraseña", "OK");
        }
    }
}
```

b) PaginaMenu

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="AlumnosApp.Paginas.PaginaMenu">

    <StackLayout>
        <StackLayout Orientation="Horizontal">
            <Label Text="Bienvenido" Style="{StaticResource LabelTitulo}"/>
            <Label Text="{Binding Nombre}" Style="{StaticResource LabelTitulo}"/>
        </StackLayout>
        <Image Source="{Binding FotoURLSAS}" Style="{StaticResource FotoAlumno}"/>
    </StackLayout>

    <ContentPage.ToolbarItems>
        <ToolbarItem x:Name="TareasPendientes" Text="Tareas Pendientes" Order="Primary"
Priority="0" Clicked="TareasPendientes_Clicked"/>
        <ToolbarItem x:Name="MisTareas" Text="Mis Tareas" Order="Primary" Priority="1"
Clicked="MisTareas_Clicked"/>
        <ToolbarItem x:Name="MiPerfil" Text="Mi Perfil" Order="Primary" Priority="2"
Clicked="MiPerfil_Clicked"/>
    </ContentPage.ToolbarItems>
</ContentPage>
```

Código C#

```
using System;
using Xamarin.Forms;
using Xamarin.Forms.Xaml;

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

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

            this.BindingContext = App.Alumno;
        }

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

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

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

c) PaginaPerfil

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="AlumnosApp.Paginas.PaginaPerfil">
    <StackLayout Padding="10" Spacing="10" BackgroundColor="White">
        <Image x:Name="imgFoto" 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="Foto" Text="Foto" Order="Primary" Priority="0"
Clicked="Foto_Clicked"/>
        <ToolbarItem x:Name="Guardar" Text="Guardar" Order="Primary" Priority="1"
Clicked="Guardar_Clicked"/>
    </ContentPage.ToolbarItems>
</ContentPage>
```

Código C#

```
using System;
using AlumnosApp.Servicios;
using Xamarin.Forms;
using Xamarin.Forms.Xaml;
using System.IO;

namespace AlumnosApp.Paginas
{
       [XamlCompilation(XamlCompilationOptions.Compile)]
       public partial class PaginaPerfil : ContentPage
       {
        MemoryStream stream;

        public PaginaPerfil ()
              {
                     InitializeComponent ();
              }
```

```
        protected override void OnAppearing()
        {
            base.OnAppearing();
            this.BindingContext = App.Alumno;
        }

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

        private async void Foto_Clicked(object sender, EventArgs e)
        {
            var action = await DisplayActionSheet("Foto del alumno", "Cancelar", null,
"Usar cámara", "Seleccionar de la biblioteca");

            bool camara = action.Contains("cámara");
            var file = await ServicioImagen.TomarFoto(camara);

            imgFoto.Source = ImageSource.FromStream(() => {
                var stream = file.GetStream();
                this.stream = new MemoryStream();
                stream.CopyTo(this.stream);
                stream.Seek(0, SeekOrigin.Begin);
                file.Dispose();
                return stream;
            });
        }

        private async void Guardar_Clicked(object sender, EventArgs e)
        {
            ActualizarActivityIndicator(true);
            stream?.Seek(0, SeekOrigin.Begin);
            await ServicioWebApi.UpdateAlumno(App.Alumno, stream);
            App.Alumno = await ServicioWebApi.GetAlumno(App.Alumno.Id);
            ActualizarActivityIndicator(false);

            await DisplayAlert("Información", "Perfil actualizado", "OK");
            await Navigation.PopAsync();
        }
    }
}
```

d) PaginaTareasPendientes

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="AlumnosApp.Paginas.PaginaTareasPendientes">
    <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"
                                  Detail="{Binding FechaLimite, Converter={StaticResource
ConvertidorFecha}}"/>
                    </DataTemplate>
                </ListView.ItemTemplate>
            </ListView>
        </StackLayout>
    </ScrollView>
</ContentPage>
```

Código C#

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

namespace AlumnosApp.Paginas
{
	[XamlCompilation(XamlCompilationOptions.Compile)]
	public partial class PaginaTareasPendientes : ContentPage
	{
		public PaginaTareasPendientes ()
		{
			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.GetTareasNoRealizadasByAlumno(App.Alumno.Id);
            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)
            {
            }
        }

    }
}
```

e) PaginaDetalleTarea

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="AlumnosApp.Paginas.PaginaDetalleTarea">

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

        <Label Text="Publicada el día" Style="{StaticResource LabelTitulo}"/>
        <Label Text="{Binding FechaPublicacion, Converter={StaticResource
ConvertidorFecha}}"  Style="{StaticResource LabelNormal}"/>
        <BoxView Style="{StaticResource Linea}"/>

        <Label Text="Fecha máxima de entrega" Style="{StaticResource LabelTitulo}"/>
        <Label Text="{Binding FechaLimite, Converter={StaticResource ConvertidorFecha}}"
Style="{StaticResource LabelNormal}"/>

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

    <ContentPage.ToolbarItems>
        <ToolbarItem x:Name="Enviar" Text="Enviar" Order="Primary" Priority="0"
Clicked="Enviar_Clicked"/>
        <ToolbarItem x:Name="Ver" Text="Archivo" Order="Primary" Priority="1"
Clicked="Ver_Clicked"/>
    </ContentPage.ToolbarItems>
</ContentPage>
```

Código C#

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

namespace AlumnosApp.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 void Ver_Clicked(object sender, EventArgs e)
	 {
		if (dato.Id > 0)
		{
			var servicioStorage = new ServicioStorage();
			Device.OpenUri(new
Uri(servicioStorage.GetFullDownloadTareaURL(dato.Id)));
		}
	 }

	 private async void Enviar_Clicked(object sender, EventArgs e)
	 {
		TareaAlumno dato = new TareaAlumno() {
			Alumno = App.Alumno,
			Tarea = this.dato,
			IdAlumno = App.Alumno.Id,
			IdTarea = this.dato.Id,
			ArchivoURL = string.Empty,
			Fecha = DateTime.Now,
			Calificacion = 0, Evaluado = false, Mensaje = string.Empty
		};
		await Navigation.PushAsync(new PaginaTareaAlumno(dato, true));
	 }
    }
}
```

f) PaginaListaTareasAlumno

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="AlumnosApp.Paginas.PaginaListaTareasAlumno">
    <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 AlumnosApp.Servicios;
using AlumnosApp.Clases;
using Xamarin.Forms;
using Xamarin.Forms.Xaml;
using System.Threading.Tasks;

namespace AlumnosApp.Paginas
{
	[XamlCompilation(XamlCompilationOptions.Compile)]
	public partial class PaginaListaTareasAlumno : ContentPage
	{
		public PaginaListaTareasAlumno ()
		{
			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.GetTareasRealizadasByAlumno(App.Alumno.Id, switchTareaEvaluada.IsToggled);
            ActualizarActivityIndicator(false);
        }

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

        protected async override void OnAppearing()
        {
            base.OnAppearing();
            try { await ObtenerTareasAlumnos();}
            catch (Exception ex) { }
        }

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

g) PaginaTareaAlumno

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="AlumnosApp.Paginas.PaginaTareaAlumno">
    <ScrollView>
        <StackLayout Padding="10" Spacing="10" BackgroundColor="White">
            <ActivityIndicator x:Name="activityIndicator" Style="{StaticResource
Indicador}"/>
            <Label Text="{Binding Tarea.Titulo}" Style="{StaticResource LabelTitulo}" />
            <BoxView Style="{StaticResource Linea}"/>

            <StackLayout Orientation="Horizontal">
                <Label Text="Fecha límite:" Style="{StaticResource LabelTitulo}"/>
                <Label Text="{Binding Tarea.FechaLimite, Converter={StaticResource
ConvertidorFecha}}" Style="{StaticResource LabelNormal}"/>
            </StackLayout>
            <BoxView Style="{StaticResource Linea}"/>

            <Label Text="Mensaje del alumno:" Style="{StaticResource LabelTitulo}"/>
            <Entry Text="{Binding Mensaje}" Style="{StaticResource CajaTexto}"/>
            <BoxView Style="{StaticResource Linea}"/>

            <StackLayout Orientation="Horizontal">
                <Label Text="Fecha respuesta:" Style="{StaticResource LabelTitulo}"/>
                <Label Text="{Binding Fecha, Converter={StaticResource
ConvertidorFecha}}"  Style="{StaticResource LabelNormal}"/>
            </StackLayout>
            <BoxView Style="{StaticResource Linea}"/>
        </StackLayout>
    </ScrollView>

    <ContentPage.ToolbarItems>
        <ToolbarItem x:Name="SeleccionarArchivo" Text="Seleccionar" Order="Primary"
Priority="0" Clicked="SeleccionarArchivo_Clicked"/>
        <ToolbarItem x:Name="VerTarea" Text="Tarea" Order="Primary" Priority="1"
Clicked="VerTarea_Clicked"/>
        <ToolbarItem x:Name="VerRespuesta" Text="Respuesta" Order="Primary" Priority="2"
Clicked="VerRespuesta_Clicked"/>
        <ToolbarItem x:Name="EnviarTarea" Text="Enviar" Order="Primary" Priority="3"
Clicked="EnviarTarea_Clicked"/>
    </ContentPage.ToolbarItems>
</ContentPage>
```

Código C#

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

```
namespace AlumnosApp.Paginas
{
        [XamlCompilation(XamlCompilationOptions.Compile)]
        public partial class PaginaTareaAlumno : ContentPage
        {
         TareaAlumno dato;
         MemoryStream stream;
         bool esNuevo;

         public PaginaTareaAlumno(TareaAlumno dato, bool esNuevo)
         {
             InitializeComponent();
             this.dato = dato;
             this.esNuevo = esNuevo;
         }

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

             if (!esNuevo)
             {
                 ActualizarActivityIndicator(true);
                 dato = await ServicioWebApi.GetTareaAlumno(dato.IdTarea, dato.IdAlumno);
                 ActualizarActivityIndicator(false);
             }

             this.BindingContext = dato;
         }

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

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

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

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

         private async void EnviarTarea_Clicked(object sender, EventArgs e)
```

```
        {
            if (stream != null || !esNuevo)
            {
                ActualizarActivityIndicator(true);

                if (esNuevo)
                    await ServicioWebApi.AddTareaAlumno(dato, stream);
                else
                    await ServicioWebApi.UpdateTareaAlumno(dato, stream);
                ActualizarActivityIndicator(false);

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

Paso 14. Modifica **MainActivity** en el proyecto de Android, agregando los permisos requeridos por el **MediaPlugin**. Simplemente agrega este método dentro de la clase principal:

```
        public override void OnRequestPermissionsResult(int requestCode, string[]
permissions, Permission[] grantResults)
        {

Plugin.Permissions.PermissionsImplementation.Current.OnRequestPermissionsResult(requestCo
de, permissions, grantResults);
        }
```

Paso 15. Compila y ejecuta la app