Hack@Home

Pon a prueba tus conocimientos

Versión: 1.0.0
Junio de 2017

Miguel Muñoz Serafín
@msmdotnet

Introducción

El objetivo de esta actividad es la de poner a prueba los conocimientos que has adquirido en el ***Xamarin Diplomado 3.0***.

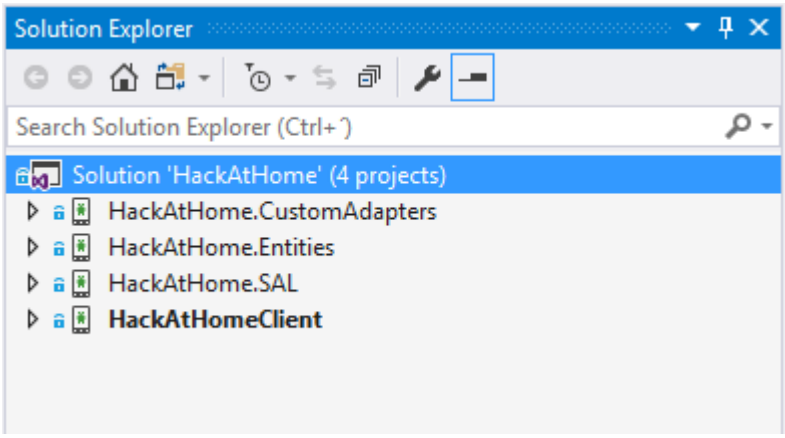
La actividad consiste en desarrollar una aplicación *Xamarin Android* que muestre la lista de actividades (laboratorios) que has desarrollado en este Diplomado. La aplicación que desarrollarás consumirá servicios Web API REST hospedados por TI Capacitación y servicios Azure Mobile hospedados por Microsoft.

Especificaciones.

1. La aplicación debe estar dividida en las siguientes capas:

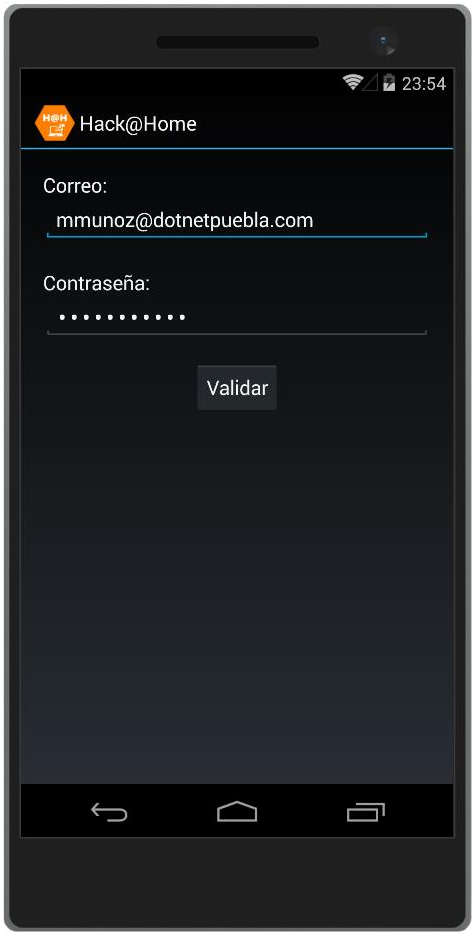
    - **Capa de Acceso a Servicio**. Esta capa deberá tener todo el código necesario para consumir las Web APIs y el servicio Azure Mobile de Microsoft. Deberá ser implementada en un proyecto ***Class Library (Android)***.
    - **Capa de Entidades**. Esta capa tendrá el código necesario para definir las clases utilizadas para intercambiar información con los servicios. Deberá ser implementada en un proyecto ***Class Library (Android)***.
    - **Capa de componentes.** En esta capa desarrollarás el código para implementar una clase **Adapter** que utilizarás como fuente de datos de un Widget *ListView* que mostrará la lista de actividades Deberá ser implementada en un proyecto ***Class Library (Android)***.
    - **Capa de Aplicación**. Esta capa tendrá el código de la aplicación Android. Deberá ser implementada en un proyecto ***Blank App (Android)***.

    La siguiente imagen muestra la estructura de proyectos sugerida para la solución.



2. La aplicación deberá ser desarrollada para soportar los lenguajes Español, Inglés y Portugués.

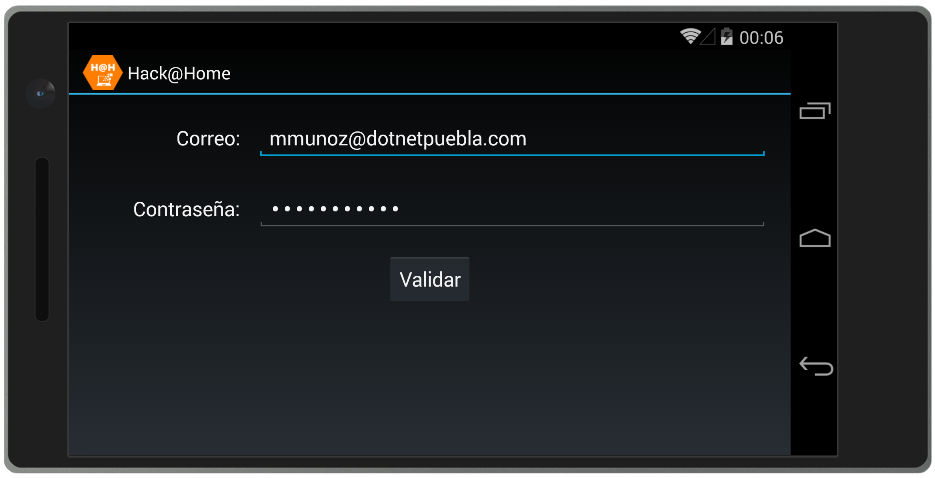
3. Al abrir la aplicación en modo vertical se deberá mostrar una pantalla solicitando las credenciales del usuario.



Los puntos que se evalúan en este paso son:

- Debe mostrar un icono personalizado.
- Debe mostrar el título ***Hack@Home*** a la derecha del icono.
- Debe mostrar las etiquetas ***Correo*** y ***Contraseña*** arriba del cuadro de texto correspondiente.
- Debe mostrar un botón centrado de forma horizontal sobre la pantalla.
- Los textos "**Correo**", "**Contraseña**" y "**Validar**" deben mostrarse en el lenguaje actual del dispositivo (Español, Inglés o Portugués).

4. Al abrir la aplicación en modo horizontal se deberá mostrar una pantalla solicitando las credenciales del usuario.



Los puntos que se evalúan en este paso son:

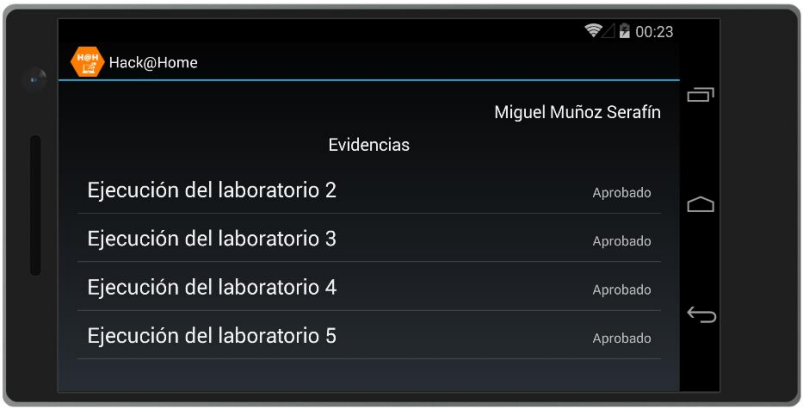
- Debe mostrar un icono personalizado.
- Debe mostrar el título ***Hack@Home*** a la derecha del icono.
- Debe mostrar las etiquetas Correo y Contraseña a la izquierda del cuadro de texto correspondiente.
- El contenido de las etiquetas Correo y Contraseña debe estar alineado a la derecha.
- Debe mostrar un botón centrado de forma horizontal sobre la pantalla.
- Los textos "**Correo**", "**Contraseña**" y "**Validar**" deben mostrarse en el lenguaje actual del dispositivo (Español, Inglés o Portugués).

5. Al hacer clic en el botón Validar:
    - Deberá invocarse al servicio de autenticación de TI Capacitación.
    - Deberá invocarse al servicio de validación de Microsoft.
    - Cuando la autenticación sea exitosa, deberá mostrarse la pantalla de evidencias.
6. La pantalla de evidencias en modo vertical deberá mostrar la lista de actividades del diplomado Xamarin 3.0.

Los puntos que se evalúan en este paso son:

- Debe mostrar un icono personalizado.
- Debe mostrar el título ***Hack@Home*** a la derecha del icono.
- Debe mostrar el nombre del participante en la parte superior de la pantalla y alineado a la derecha.
- Debe mostrar el título **Evidencias** en el lenguaje correspondiente.
- Debe mostrar la lista de evidencias con el título de la evidencia en un tamaño de letra mayor al Estatus de la evidencia.
- El estatus de la evidencia deberá mostrarse debajo del título de la evidencia.

7. La pantalla de evidencias en modo horizontal deberá mostrar la lista de actividades del diplomado Xamarin 3.0.

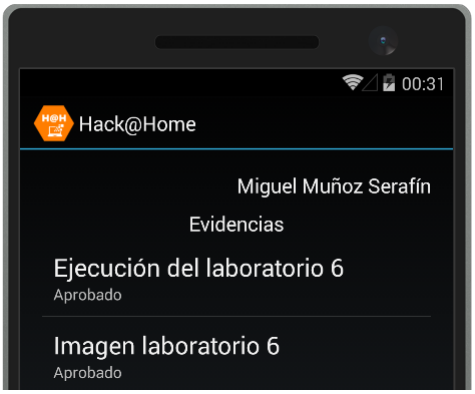


Los puntos que se evalúan en este paso son:

- Debe mostrar un icono personalizado.
- Debe mostrar el título ***Hack@Home*** a la derecha del icono.
- Debe mostrar el nombre del participante en la parte superior de la pantalla y alineado a la derecha.
- Debe mostrar el título **Evidencias** en el lenguaje correspondiente.
- Debe mostrar la lista de evidencias con el título de la evidencia en un tamaño de letra mayor al Estatus de la evidencia.
- El estatus de la evidencia deberá mostrarse en el extremo derecho de la pantalla y en el mismo renglón que el título de la evidencia.

8. La aplicación debe mantener el estado para invocar al servicio Web API únicamente al mostrar la pantalla y no al girar el dispositivo. Los puntos que se evalúan en este paso son:

- El usuario podrá desplazarse por la lista de evidencias. Por ejemplo, ubicando el laboratorio 6.



- El usuario podrá girar a modo horizontal el dispositivo y seguirá viendo la misma información que se veía en modo vertical.

- El usuario podrá presionar el botón home del dispositivo.
- Al regresar a la aplicación, será mostrada la misma información.

9. Al hacer clic en una actividad se mostrará la pantalla con el detalle de la actividad.

Los puntos que se evalúan en este paso son:

- Debe mostrar un icono personalizado.
- Debe mostrar el título ***Hack@Home*** a la derecha del icono.
- Debe mostrar el nombre del participante en la parte superior de la pantalla y alineado a la derecha.
- Debe mostrar el título de la evidencia centrado horizontalmente.
- Debe mostrar el estatus de la evidencia centrado horizontalmente.
- Debe mostrar el texto ***Descripción*** en el lenguaje correspondiente.
- Debe mostrar la descripción de la evidencia.
- Debe mostrar la imagen de la evidencia.

Condiciones.

1. Únicamente se podrán utilizar los Widgets utilizados en los laboratorios y los que se indican en la sección de **TIPs** de este mismo documento.
2. Únicamente se podrá utilizar funcionalidad Android que haya sido vista en los laboratorios y los que se indican en la sección de **TIPs** de este mismo documento.

Entrega de la aplicación.

1. Los participantes deberán realizar un video mostrando cada funcionalidad que será evaluada.
2. El video podrá ser subido en Youtube, Facebook o Vimeo.
3. Los participantes deberán proporcionar el enlace de su video en el sitio de evidencias: https://ticapacitacion.com/evidencias/xamarin30
4. Los participantes deberán proporcionar el enlace del código fuente de la aplicación en el sitio de evidencias. Por ejemplo, pueden subir el código fuente en un archivo ZIP a OneDrive.

TIPs.

1. El siguiente código te permite consumir el servicio Web API de autenticación.

```
/// <summary>
/// Realiza la autenticación al servicio Web API.
/// </summary>
/// <param name="studentEmail">Correo del usuario</param>
/// <param name="studentPassword">Password del usuario</param>
/// <returns>Objeto ResultInfo con los datos del usuario y un token de
autenticación.</returns>
public async Task<ResultInfo> AutenticateAsync(
    string studentEmail, string studentPassword)
{
    ResultInfo Result = null;

    // Dirección base de la Web API
    string WebAPIBaseAddress = "https://ticapacitacion.com/hackathome/";
    // ID del diplomado.
    string EventID = "xamarin30";

    string RequestUri = "api/evidence/Authenticate";

    // El servicio requiere un objeto UserInfo con los datos del usuario y evento.
    UserInfo User = new UserInfo
    {
        Email = studentEmail,
        Password = studentPassword,
        EventID = EventID
    };
    // Utilizamos el objeto System.Net.Http.HttpClient para consumir el servicio REST
    // Debe instalarse el paquete NuGet System.Net.Http
```

```
    using (var Client = new HttpClient())
    {
        // Establecemos la dirección base del servicio REST
        Client.BaseAddress = new Uri(WebAPIBaseAddress);

        // Limpiamos encabezados de la petición.
        Client.DefaultRequestHeaders.Accept.Clear();

        // Indicamos al servicio que envie los datos en formato JSON.
        Client.DefaultRequestHeaders.Accept.Add(
            new MediaTypeWithQualityHeaderValue("application/json"));
        try
        {
            // Serializamos a formato JSON el objeto a enviar.
            // Debe instalarse el paquete NuGet Newtonsoft.Json.
            var JSONUserInfo = JsonConvert.SerializeObject(User);

            // Hacemos una petición POST al servicio enviando el objeto JSON
            HttpResponseMessage Response =
                    await Client.PostAsync(RequestUri,
                    new StringContent(JSONUserInfo.ToString(),
                                      Encoding.UTF8, "application/json"));

            // Leemos el resultado devuelto.
            var ResultWebAPI = await Response.Content.ReadAsStringAsync();

            // Deserializamos el resultado JSON obtenido
            Result = JsonConvert.DeserializeObject<ResultInfo>(ResultWebAPI);
        }
        catch (System.Exception )
        {
            // Aquí podemos poner el código para manejo de excepciones.
        }
    }
    return Result;
}
```

Las siguientes son las sentencias ***using*** del código anterior.

```
using Newtonsoft.Json;
using System;
using System.Net.Http;
using System.Net.Http.Headers;
using System.Text;
using System.Threading.Tasks;
```

La siguiente es la estructura de la clase ***ResultInfo***.

```
    public class ResultInfo
    {
        public Status Status { get; set; }
        // El Token expira después de 10 minutos del último acceso al servicio REST
        public string Token { get; set; }

        public string FullName{  get; set; }
    }
```

La siguiente es la estructura de la enumeración ***Status***.

```
public enum Status
{
    Error = 0,
    Success = 1,
    InvalidUserOrNotInEvent = 2,
    OutOfDate = 3,
    AllSuccess = 999
}
```

La siguiente es la estructura de la clase *UserInfo*.

```
public class UserInfo
{
    public string Email { get; set; }
    public string Password { get; set; }
    public string EventID { get; set; }
}
```

2. El siguiente código te permite consumir el servicio REST que te devuelve el listado de evidencias.

```
/// <summary>
/// Obtiene la lista de evidencias.
/// </summary>
/// <param name="token">Token de autenticación del usuario.</param>
/// <returns>Una lista con las evidencias.</returns>
public async Task<List<Evidence>> GetEvidencesAsync(string token)
{
    List<Evidence> Evidences = null;

    // Dirección base de la Web API
    string WebAPIBaseAddress = "https://ticapacitacion.com/hackathome/";

    // URI completo
    string URI = $"{WebAPIBaseAddress}api/evidence/getevidences?token={token}";

    using (var Client = new HttpClient())
    {
        Client.DefaultRequestHeaders.Accept.Clear();
        Client.DefaultRequestHeaders.Accept.Add(
           new MediaTypeWithQualityHeaderValue("application/json"));
        try
        {
            // Realizamos una petición GET
            var Response =
                    await Client.GetAsync(URI);

            if (Response.StatusCode == HttpStatusCode.OK)
            {
               // Si el estatus de la respuesta HTTP fue exitosa, leemos
               // el valor devuelto.

                var ResultWebAPI = await Response.Content.ReadAsStringAsync();
```

```
                Evidences =
                    JsonConvert.DeserializeObject<List<Evidence>>(ResultWebAPI);
            }
        }
        catch (System.Exception )
        {
            // Aquí podemos poner el código para manejo de excepciones.
        }
    }
    return Evidences;
}
```

Las siguientes son las sentencias ***using*** del código anterior.

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

La siguiente es la estructura de la clase ***Evidence***.

```
public class Evidence
{

    // Identificador de la evidencia
    public int EvidenceID { get; set; }

    // Título de la Evidencia
    public string Title { get; set; }

    // Estatus de la Evidencia
    public string Status { get; set; }
}
```

3. El siguiente código te permite consumir el servicio Web API que te devuelve el detalle de una evidencia.

```
/// <summary>
/// Obtiene el detalle de una evidencia.
/// </summary>
/// <param name="token">Token de autenticación del usuario</param>
/// <param name="evidenceID">Identificador de la evidencia.</param>
/// <returns>Información de la evidencia.</returns>
public async Task<EvidenceDetail> GetEvidenceByIDAsync(string token, int evidenceID)
{
    EvidenceDetail Result = null;

    // Dirección base de la Web API
    string WebAPIBaseAddress = "https://ticapacitacion.com/hackathome/";

    // URI de la evidencia.
```

```
    string URI =
$"{WebAPIBaseAddress}api/evidence/getevidencebyid?token={token}&&evidenceid={evidenceI
D}";

    using (var Client = new HttpClient())
    {
        Client.DefaultRequestHeaders.Accept.Clear();
        Client.DefaultRequestHeaders.Accept.Add(
            new MediaTypeWithQualityHeaderValue("application/json"));
        try
        {
            // Realizamos una petición GET
            var Response = await Client.GetAsync(URI);

            var ResultWebAPI = await Response.Content.ReadAsStringAsync();
            if (Response.StatusCode == HttpStatusCode.OK)
            {
                // Si el estatus de la respuesta HTTP fue exitosa, leemos
                // el valor devuelto.
                Result = JsonConvert.DeserializeObject<EvidenceDetail>(ResultWebAPI);
            }
        }
        catch (System.Exception)
        {
            // Aquí podemos poner el código para manejo de excepciones.
        }
    }
    return Result;
}
```

Las siguientes son las sentencias ***using*** del código anterior.

```
using Newtonsoft.Json;
using System.Net;
using System.Net.Http;
using System.Net.Http.Headers;
using System.Threading.Tasks;
```

La siguiente es la estructura de la clase ***EvidenceDetail***.

```
    public class EvidenceDetail
    {
        public string Description { get; set; }

        // URL de la imagen de la evidencia.
        public string Url { get; set; }

    }
```

4. El siguiente código te permite enviar la evidencia de tu actividad al servicio Azure Mobile de Microsoft.

```
public class MicrosoftServiceClient
{
    // Cliente para acceder al servicio Mobile
```

```
    MobileServiceClient Client;
    // Objeto para realizar operaciones con Tablas de Mobile Service
    private IMobileServiceTable<LabItem> LabItemTable;

    /// <summary>
    /// Envía una evidencia.
    /// </summary>
    /// <param name="userEvidence">Objeto con los datos de la evidencia.</param>
    /// <returns></returns>
    public async Task SendEvidence(LabItem userEvidence)
    {
        Client =
            new MobileServiceClient(@"http://xamarin-diplomado.azurewebsites.net/");
        LabItemTable = Client.GetTable<LabItem>();
        await LabItemTable.InsertAsync(userEvidence);
    }
}
```

Las siguientes son las sentencias ***using*** del código anterior.

```
using Microsoft.WindowsAzure.MobileServices;
using System.Threading.Tasks;
```

Esta es la estructura de la clase ***LabItem***.

```
public class LabItem
{
    public string Id { get; set; }
    public string Email { get; set; }
    public string Lab { get; set; }
    public string DeviceId { get; set; }
}
```

Este es un ejemplo del consumo del método ***SendEvidence***.

```
var MicrosoftEvidence  = new LabItem
{
    Email = Email.Text,
    Lab = "Hack@Home",
    DeviceId = Android.Provider.Settings.Secure.GetString(
                    ContentResolver, Android.Provider.Settings.Secure.AndroidId)
};
var MicrosoftClient = new MicrosoftServiceClient();
await MicrosoftClient.SendEvidence(MicrosoftEvidence);
```

5. El siguiente es el código que implementa el objeto *Adapter* necesario para el *ListView* que mostrará las evidencias.

```
using Android.App;
using Android.Views;
using Android.Widget;
using HackAtHome.Entities;
using System.Collections.Generic;

namespace HackAtHome.CustomAdapters
{
    /// <summary>
```

```
    /// EvidenceAdapter es un tipo Adapter que permite administrar los datos de un
ListView.
    /// Los datos que maneja son de tipo Evidence.
    /// </summary>
    public class EvidencesAdapter : BaseAdapter<Evidence>
    {
        List<Evidence> Items; // Datos de cada evidencia de laboratorio.
        Activity Context; // Activity donde se utilizará este Adapter.
        int ItemLayoutTemplate; // Layout a utilizar para mostrar los datos de un
elemento.
        int EvidenceTitleViewID; // ID del TextView donde se mostrará el nombre de la
evidencia.
        int EvidenceStatusViewID; // ID del TextView donde se mostrará el estatus de
la evidencia.

        /// <summary>
        /// Constructor para recibir la información que necesita el Adapter
        /// </summary>
        /// <param name="context">Activity donde se aloja el ListView.</param>
        /// <param name="evidences">La lista de elementos.</param>
        /// <param name="itemLayoutTemplate">ID del Layout para mostrar cada elemento
del ListView.</param>
        /// <param name="evidenceTitleViewID">ID del TextView donde se mostrará el
título de la evidencia.</param>
        /// <param name="evidenceStatusViewID">ID del TextView donde se mostrará el
estatus de la evidencia.</param>
        public EvidencesAdapter(Activity context, List<Evidence> evidences, int
itemLayoutTemplate, int evidenceTitleViewID, int evidenceStatusViewID)
        {
            this.Context = context;
            this.Items = evidences;
            this.ItemLayoutTemplate = itemLayoutTemplate;
            this.EvidenceTitleViewID = evidenceTitleViewID;
            this.EvidenceStatusViewID = evidenceStatusViewID;
        }

        /// <summary>
        ///  Devuelve el elemento de la lista localizado en la posición especificada.
        /// </summary>
        /// <param name="position">Posición del elemento dentro de la lista.</param>
        /// <returns></returns>
        public override Evidence this[int position]
        {
            get
            {
                return Items[position];
            }
        }

        /// <summary>
        /// Devuelve el número de elementos de la lista.
        /// </summary>
        public override int Count
        {
            get
            {
```

```
                return Items.Count;
            }
        }

        /// <summary>
        /// Devuelve el ID del elemento localizado en la posición especificada.
        /// </summary>
        /// <param name="position">Posición del elemento dentro de la lista.</param>
        /// <returns></returns>
        public override long GetItemId(int position)
        {
            return Items[position].EvidenceID;
        }

        //
        /// <summary>
        /// Devuelve el View que muestra los datos de un elemento del conjunto de
datos.
        /// </summary>
        /// <param name="position">Posición del elemento a mostrar.</param>
        /// <param name="convertView">View anterior que puede ser reutilizada.</param>
        /// <param name="parent">View padre al que podria adjuntarse el View
devuelto.</param>
        /// <returns></returns>
        public override View GetView(int position, View convertView, ViewGroup parent)
        {
            // Obtenemos el elemento del cual se requiere la Vista
            var Item = Items[position];
            View ItemView; // Vista que vamos a devolver
            if(convertView == null)
            {
                // No hay vista reutilizable, crear una nueva
                ItemView = Context.LayoutInflater.Inflate(ItemLayoutTemplate, null /*
No hay View padre*/);
            }
            else
            {
                // Reutilizamos un View existente para ahorrar recursos
                ItemView = convertView;
            }

            // Establecemos los datos del elemento dentro del View
            ItemView.FindViewById<TextView>(EvidenceTitleViewID).Text = Item.Title;
            ItemView.FindViewById<TextView>(EvidenceStatusViewID).Text = Item.Status;

            return ItemView;
        }
    }
}
```

6. Para pasar un valor simple (string, int, etc.) a un *Activity* a través de un *Intent* utilizamos:

```
Intent.PutExtra("NombreDelValor", Valor);
```

7. Para obtener un valor cadena desde un ***Intent*** podemos utilizar la siguiente instrucción:

```
Intent.GetStringExtra("NombreDelValor");
```

8. Para obtener un valor ***int*** desde un ***Intent*** podemos utilizar la siguiente instrucción:

```
Intent.GetIntExtra("NombreDelValor",0);
```

9. El siguiente código permite determinar si el dispositivo se encuentra en modo *Landscape*.

```
bool IsLandScape(Activity activity)
{
    var Orientation = activity.WindowManager.DefaultDisplay.Rotation;
    return Orientation == SurfaceOrientation.Rotation90 ||
                          Orientation == SurfaceOrientation.Rotation270;

}
```

10. El componente ***UrlImageViewHelper*** te permite mostrar una imagen desde un URL.
    - Guía para instalar un componente: https://developer.xamarin.com/guides/cross-platform/xamarin-studio/components_walkthrough/

11. El siguiente es un ejemplo de cómo establecer la imagen a mostrar en un Widget *ImageView* utilizando el componente ***UrlImageViewHelper***.

```
Koush.UrlImageViewHelper.SetUrlDrawable(WidgetImageView, ImageUrl);
```

12. Puedes utilizar el widget *WebView* para mostrar código HTML en una aplicación Android. La descripción de una evidencia contiene código HTML.

13. El siguiente código muestra una forma de establecer el contenido HTML de una variable de tipo *string* que podrá ser mostrado por el Widget *WebView*.

```
WidGetWebView.LoadDataWithBaseURL(
                                null, WebViewContent, "text/html", "utf-8", null);
```

    WebViewContent es una variable ***string*** con el código HTML a mostrar.

14. El evento ***ItemClick*** de un ***ListView*** se origina cuando el usuario selecciona un elemento en un ***ListView***.

Si encuentras problemas durante la realización de esta actividad, puedes solicitar apoyo en los grupos de Facebook siguientes:

https://www.facebook.com/groups/iniciandoconxamarin/
https://www.facebook.com/groups/xamarindiplomadoitc/

Check List

La siguiente lista te puede servir de guía al momento de grabar tu video.

1. ☐ ¿Muestra la estructura de la solución dividida en 4 capas?
2. ☐ ¿Muestra la estructura expandida de cada uno de los subdirectorios del directorio ***Resources*** de la aplicación Android?
3. ☐ ¿Muestra la referencia al Assembly ***Microsoft.Azure.Mobile.Client*** en la capa de Servicio?

Al abrir la aplicación en modo vertical.

4. ☐ ¿Muestra un icono personalizado?
5. ☐ ¿Muestra el título ***Hack@Home*** a la derecha del icono?
6. ☐ ¿Muestra las etiquetas *Correo* y *Contraseña* arriba del cuadro de texto correspondiente?
7. ☐ ¿Muestra un botón centrado de forma horizontal sobre la pantalla?
8. ☐ ¿Los textos "**Correo**", "**Contraseña**" y "**Validar**" se muestran en el lenguaje Español, Inglés y Portugués al cambiar la configuración del dispositivo?

Al abrir la aplicación en modo horizontal.

9. ☐ ¿Muestra un icono personalizado?
10. ☐ ¿Muestra el título ***Hack@Home*** a la derecha del icono?
11. ☐ ¿Muestra las etiquetas *Correo* y *Contraseña* a la izquierda del cuadro de texto correspondiente?
12. ☐ ¿El contenido de las etiquetas *Correo* y *Contraseña* está alineado a la derecha?
13. ☐ ¿Muestra un botón centrado de forma horizontal sobre la pantalla?
14. ☐ ¿Los textos "**Correo**", "**Contraseña**" y "**Validar**" se muestran en el lenguaje Español, Inglés y Portugués al cambiar la configuración del dispositivo?

La pantalla de evidencias en modo Vertical.

15. ☐ ¿Muestra un icono personalizado?
16. ☐ ¿Muestra el título ***Hack@Home*** a la derecha del icono?
17. ☐ ¿Muestra el nombre del participante en la parte superior de la pantalla y alineado a la derecha?
18. ☐ ¿Muestra el título **Evidencias** en el lenguaje Español, Inglés y Portugués al cambiar la configuración del dispositivo?
19. ☐ ¿Muestra la lista de evidencias con el título de la evidencia en un tamaño de letra mayor al Estatus de la evidencia?
20. ☐ ¿Muestra el estatus de la evidencia debajo del título de la evidencia?

La pantalla de evidencias en modo Horizontal.

21. ☐ ¿Muestra un icono personalizado?
22. ☐ ¿Muestra el título ***Hack@Home*** a la derecha del icono?
23. ☐ ¿Muestra el nombre del participante en la parte superior de la pantalla y alineado a la derecha?

24. ☐ ¿Muestra el título **Evidencias** en el lenguaje Español, Inglés y Portugués al cambiar la configuración del dispositivo?
25. ☐ ¿Muestra la lista de evidencias con el título de la evidencia en un tamaño de letra mayor al Estatus de la evidencia?
26. ☐ ¿Muestra el estatus de la evidencia en el extremo derecho de la pantalla y en el mismo renglón que el título de la evidencia?

La aplicación debe mantener el estado.

27. ☐ ¿Muestra el desplazamiento por la lista de evidencias ubicando alguna evidencia de la lista?
28. ☐ ¿Al girar el dispositivo se mantiene la posición ubicada en la lista de evidencias?
26. ☐ ¿Presiona el botón *Home* del dispositivo?
30. ☐ Al regresar a la aplicación después de presionar el botón *Home* del dispositivo ¿Se mantiene la posición ubicada en la lista de evidencias?

Al hacer clic en una actividad se mostrará la pantalla con el detalle de la actividad.

31. ☐ ¿Muestra un icono personalizado?
32. ☐ ¿Muestra el título ***Hack@Home*** a la derecha del icono?
33. ☐ ¿Muestra el nombre del participante en la parte superior de la pantalla y alineado a la derecha?
34. ☐ ¿Muestra el título de la evidencia centrado horizontalmente?
35. ☐ ¿Muestra el estatus de la evidencia centrado horizontalmente?
36. ☐ ¿Muestra el texto ***Descripción*** en el lenguaje Español, Inglés y Portugués al cambiar la configuración del dispositivo?
37. ☐ ¿Muestra la descripción de la evidencia?
38. ☐ ¿Muestra la imagen de la evidencia?