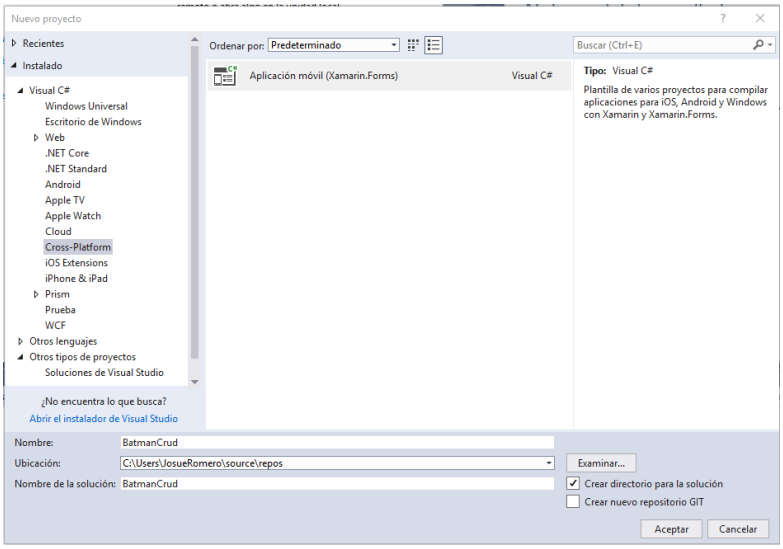
En el aeropuerto militar, se llevaba a cabo una exhibición de aviones de reconocimiento y destrezas tácticas como saltos por paracaídas. Sucede que una nave fue tripulada y manipulada por un sujeto todavía no identificado, el cual hizo colisionar una nave contra un helicóptero, ambos cayendo sobre el graderío de los espectadores.

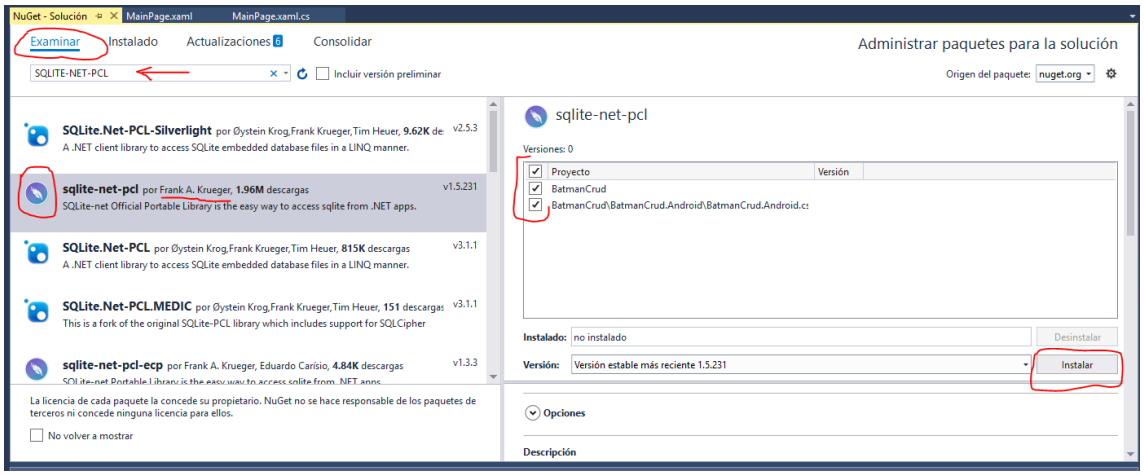
Debido a la cantidad de personas tratando llamar por su teléfono móvil hacia emergencias las antenas de telefonía colapsaron, dejando sin señal a la zona, por lo que se activaron los cuerpos de emergencia destinados a los paracaidistas e incendios, siendo estos insuficientes para el público.

Debido a que no había forma para conectarse con los sistemas centrales, los socorristas al rescatar a una persona la registraban de forma local en su teléfono móvil atreves de una aplicación xamarin.

Se solicitó de forma urgente al departamento de tecnología militar desarrollar una aplicación para tener un control de las personas restadas del incidente y de forma local para cualquier dispositivo android. Para ello debían registrar solamente el nombre de la persona, el número de documento ya sea este dui, nit, carnet de menoridad o pasaporte, e indicar si este es o no menor de edad. Este fue el resultado.

CONSTRUYAMOS EL PROYECTO Y AGREGEMOS EL NUGGET.





Model

Creamos la carpeta Model en el proyecto.

Creamos la siguiente clase llamada **VictimaModel,** para el registro de cada víctima, entendiendo que si es mayor de edad el número de documento pertenece a un DUI y si no, es de un documento de minoridad.

Importante especificar la llave primaria y su propiedad auto-incrementable.

```
using System;
using System.Collections.Generic;
using System.Text;
using SQLite;

namespace BatmanCrud.Model
{
    public class VictimaModel
    {
        [PrimaryKey, AutoIncrement]
        public int Id { get; set; }
        public string Nombre { get; set; }
        public int Documento { get; set; }
        public bool EsMayor { get; set; }
    }
}
```

Siempre en nuestra carpeta Model, vamos a Crear la clase base, que tendrá un código que en lugar de estar repitiendo solo estaremos llamando donde le necesitemos. Será una **ClaseBase.**

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Runtime.CompilerServices;
using System.Text;

namespace BatmanCrud.Model
{
    public class ClaseBase : INotifyPropertyChanged
    {
        public event PropertyChangedEventHandler PropertyChanged;

        protected virtual void OnPropertyChanged([CallerMemberName] string propertyName = null)
        {
            PropertyChanged?.Invoke(this, new PropertyChangedEventArgs(propertyName));
        }
    }
}
```

Services

De alguna forma he tratado de simplificar el uso de interfaces, debido a que es necesario que por tema de plataformas obtener la ruta y también la lista de funcionalidades, aquí, directamente consumimos las funcionalidades en una ruta que solo funciona para Android. Creamos la carpeta Services, y dentro una clase llamada **Repository**.

```
using System;
using System.Collections.Generic;
using System.IO;
using System.Text;
using System.Threading.Tasks;
using BatmanCrud.Model;
using SQLite;

namespace BatmanCrud.Services
{
    public class Repository
    {
        SQLiteAsyncConnection BaseDeDatos;

        public Repository()
        {
            var UbicacionBD = Path.Combine(Environment.GetFolderPath(Environment.SpecialFolder.LocalApplicationData), "NombreDB.db3");
            BaseDeDatos = new SQLiteAsyncConnection(UbicacionBD);
            BaseDeDatos.CreateTableAsync<VictimaModel>().Wait();
        }

        public Task<int> CrearVictimaAsync(VictimaModel NvaVictima)
        {
            return BaseDeDatos.InsertAsync(NvaVictima);
        }

        public Task<int> ActulizarVictimaAsync(VictimaModel VictimaModificada)
        {
            return BaseDeDatos.UpdateAsync(VictimaModificada);
        }

        public Task<int> EliminarVictimaAsync(VictimaModel VictimaEliminada)
        {
            return BaseDeDatos.DeleteAsync(VictimaEliminada);
        }

        public  Task<List<VictimaModel>> ObtenerVictimasAsync()
        {
            return  BaseDeDatos.Table<VictimaModel>().ToListAsync();
        }
    }
}
```

ViewModel

Crear carpeta ViewMode. Al inicio de la aplicación queremos mostrar las victimas ya registradas. Además de identificar de entre las victimas cual ha sido seleccionada. También Crear una nueva o de haber seleccionado uno, modificar su información o eliminarle.

InicioVictimaViewModel.

Nuestra clase hereda de nuestra ClaseBase, para no volver a copiar la implementación del INotifyPropertyChanged.

Creamos una propiedad de tipo ObservableCollection que es nuestra lista de victimas almacenadas.

Una propiedad para identificar la victima selecionada.

Y los comandos para crear uno nuevo o ver su detalle. Al crear uno

```
using BatmanCrud.Model;
using BatmanCrud.Services;
using BatmanCrud.View;
using System.Collections.ObjectModel;
using Xamarin.Forms;

namespace BatmanCrud.ViewModel
{
   public class InicioVictimaViewModel:ClaseBase
    {
        private ObservableCollection<VictimaModel> _Victimas;

        public ObservableCollection<VictimaModel> Victimas
        {
            get { return _Victimas; }
            set { _Victimas = value; OnPropertyChanged(); }
        }

        public Command GuardarCommand { get; set; }
        public Command DetalleCommand { get; set; }
        public VictimaModel VictimaSeleccionada { get; set; }

        public InicioVictimaViewModel()
        {
            ObtenerVictimas();
            GuardarCommand = new Command(Guardar);
            DetalleCommand = new Command(Detalle);
        }

        private async void Detalle()
        {
            var Pagina = new NuevaVictimaPage();
            Pagina.BindingContext = new NuevaVictimaViewModel(VictimaSeleccionada);
            await App.Current.MainPage.Navigation.PushAsync(Pagina);
        }
        private async void Guardar()
        {
            var Pagina = new NuevaVictimaPage();
            await App.Current.MainPage.Navigation.PushAsync(Pagina);
        }

        async void ObtenerVictimas()
        {
            var Repo = new Repository();
            var Resultado = await Repo.ObtenerVictimasAsync();
            Victimas = new ObservableCollection<VictimaModel>(Resultado);
        }
    }
}
```

nuevo **No enviamos ningún VictimaModel** como parámetro, y llegamos a digitar los datos de la nueva victima.

Si le damos ver detalle enviamos como parámetro **la victima seleccionada.**

View: Creamos la carpeta view y creamos una pagina de contenido con el siguiente diseño.

InicioVictimaPage

```
<?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:vm="clr-namespace:BatmanCrud.ViewModel"
             x:Class="BatmanCrud.InicioVictimaPage">

    <ContentPage.BindingContext>
        <vm:InicioVictimaViewModel></vm:InicioVictimaViewModel>
    </ContentPage.BindingContext>

    <ContentPage.ToolbarItems>
        <ToolbarItem Text="+ Agregar" Command="{Binding GuardarCommand}">
        </ToolbarItem>
    </ContentPage.ToolbarItems>

    <ContentPage.Content>

        <ListView ItemsSource="{Binding Victimas}" SelectedItem="{Binding VictimaSeleccionada}" HasUnevenRows="True">
            <ListView.Header>
                <Button Text="Ver Detalle Victima" Command="{Binding DetalleCommand}"></Button>
            </ListView.Header>

            <ListView.ItemTemplate>
                <DataTemplate>
                    <TextCell Text="{Binding Nombre}"   Detail="{Binding Documento}"></TextCell>
                </DataTemplate>
            </ListView.ItemTemplate>
        </ListView>

    </ContentPage.Content>
</ContentPage>
```

ViewModel

NuevaVictimaViewModel

1. Creamos las propiedades. Para acceder a las funcionalidades del repositorio o base de datos. Los respectrivos comandos de cada botón. (ActulizarCommand esta demás).
2. Primer constructor sin recibir un parámetro, por ende los que se digite será la información de una nueva víctima.
3. Recibiendo un parámetro, que al modificar o eliminar afectaran a la víctima antes seleccionada.
4. Funcionalidad del comando GuardarComand
5. Funcionalidad del comando EliminarCommand

```
using BatmanCrud.Model;
using BatmanCrud.Services;
using System;
using System.Collections.Generic;
using System.Text;
using Xamarin.Forms;

namespace BatmanCrud.ViewModel
{
   public class NuevaVictimaViewModel
    {
        Repository Repositorio { get; set; }
        VictimaModel VictimaSeleccionada;
        public Command GuardarCommand { get; set; }
        public Command ActualizarCommand { get; set; }
        public Command EliminarCommand { get; set; }
        public string NombreF { get; set; }
        public int DocumentoF { get; set; }
        public bool EsMayorF { get; set; }

        public NuevaVictimaViewModel()
        {
            VictimaSeleccionada = null;
            Repositorio = new Repository();
            GuardarCommand = new Command(GuardarCambios);
            EliminarCommand = new Command(EliminarVictima);
        }

        public NuevaVictimaViewModel(VictimaModel VictimaParametro): this()
        {
            VictimaSeleccionada = VictimaParametro;
            if (VictimaParametro != null)
            {
                NombreF = VictimaParametro.Nombre;
                DocumentoF = VictimaParametro.Documento;
                EsMayorF = VictimaParametro.EsMayor;
            }
        }
```

```
async void GuardarCambios()
{
    if (VictimaSeleccionada != null)
    {
        VictimaSeleccionada.Nombre = NombreF;
        VictimaSeleccionada.Documento = DocumentoF;
        VictimaSeleccionada.EsMayor = EsMayorF;

        var RespuestaDb = await Repositorio.ActulizarVictimaAsync(VictimaSeleccionada);
        if (RespuestaDb == 1)
        {
            await App.Current.MainPage.DisplayAlert("Informacion", "Registro actualizado con exito", "ok");
            await App.Current.MainPage.Navigation.PushAsync(new InicioVictimaPage());
        }
    }
    else
    {
        var VictimaNueva = new VictimaModel();
        VictimaNueva.Nombre = NombreF;
        VictimaNueva.Documento = DocumentoF;
        VictimaNueva.EsMayor = EsMayorF;
        var ResultadoDb = await Repositorio.CrearVictimaAsync(VictimaNueva);
        if (ResultadoDb == 1)
        {
            await App.Current.MainPage.DisplayAlert("Informacion", "Registro de Victima con exito", "ok");
            await App.Current.MainPage.Navigation.PushAsync(new InicioVictimaPage());
        }
    }
}

async void EliminarVictima()
{
    var RespuestaDb = await Repositorio.EliminarVictimaAsync(VictimaSeleccionada);
    if (RespuestaDb == 1)
    {
        await App.Current.MainPage.DisplayAlert("Informacion", "Eliminacion de Victima con exito", "ok");
        await App.Current.MainPage.Navigation.PushAsync(new InicioVictimaPage());
    }
}
```

Por ultimo y por lógica en el archivo App.Xaml.cs

```
public partial class App : Application
{
    public App()
    {
        InitializeComponent();

        MainPage = new NavigationPage( new InicioVictimaPage());
    }
```