2019

ICB0-M08U02IA07

DADES – SQLITE + REALM

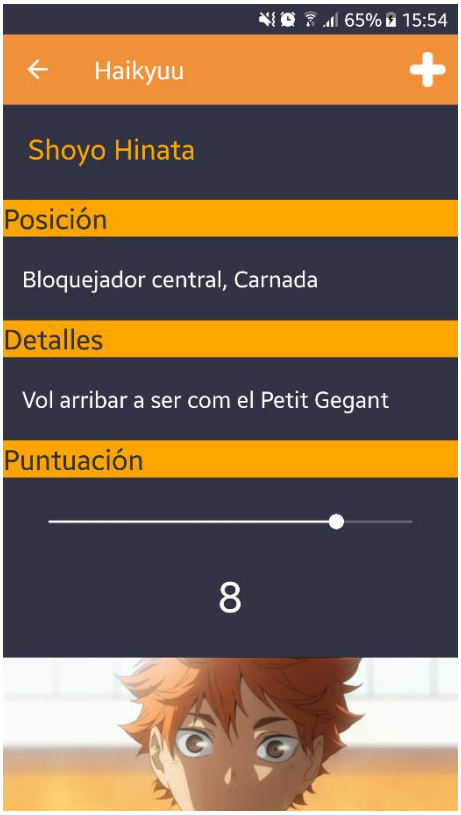
POL BENITO

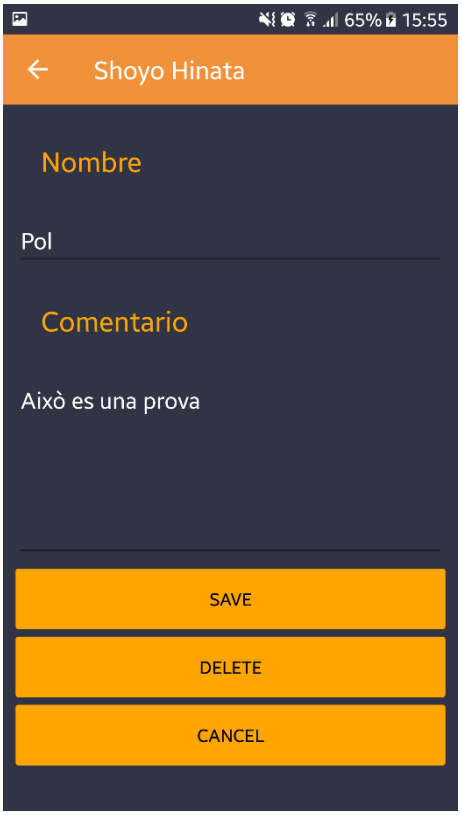
Índice

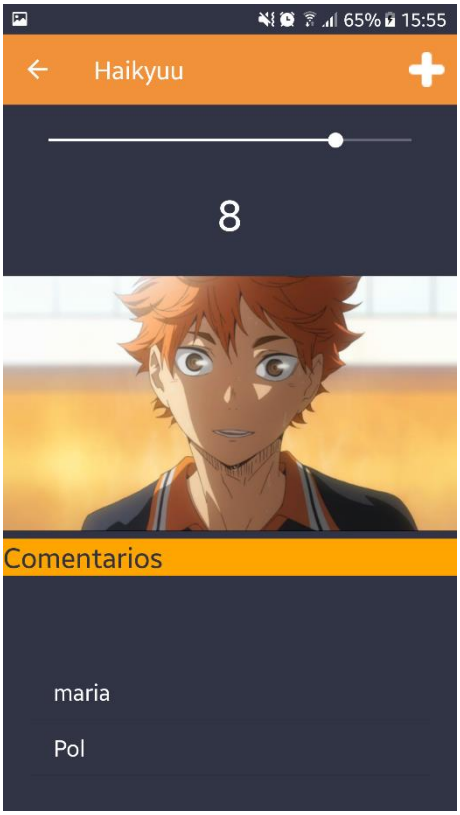
1. Introducción

Esta aplicación será hecha a partir de la aplicación anterior seguida con el tutorial *ICB0-M08U02IA06*. Añadiremos un par de funcionalidades: Una barra de puntuación para puntuar los personajes y un apartado de comentarios que puedes dejar para que cada usuario pueda comentar sobre este.

Aquí a continuación os dejo algunas capturas de cómo quedará la aplicación:







A partir de este ejemplo podemos añadir o modificar cada una de estas pantallas, aunque esto será una guía a través de mi diseño.

Estas funcionalidades nuevas serán hechas a partir de una Base de Datos: una primera aplicación con **SQLite** y otra con **Realm**.

SQLite es una **Base de Datos relacional** con la que empezaremos, ya que, por estadística, la gente utiliza más estas, que son esas que tienen tablas y pueden contener foráneas.

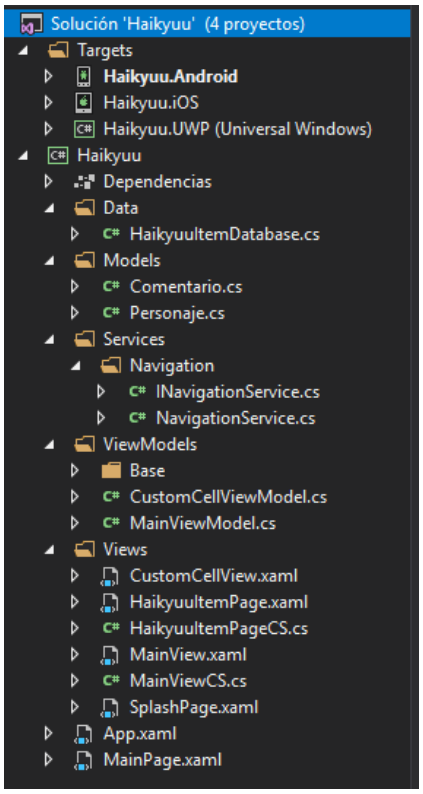
Después seguiremos con **Realm**, algo más difícil de comprender porque no estaremos acostumbrados, pero muy fácil de hacer porque ya habremos hecho la anterior aplicación. Esta **Base de Datos no es relacional**, es decir, no va por tablas. Por eso es un poco confuso al principio cuando intentas entender cómo funciona.

2. SQLite

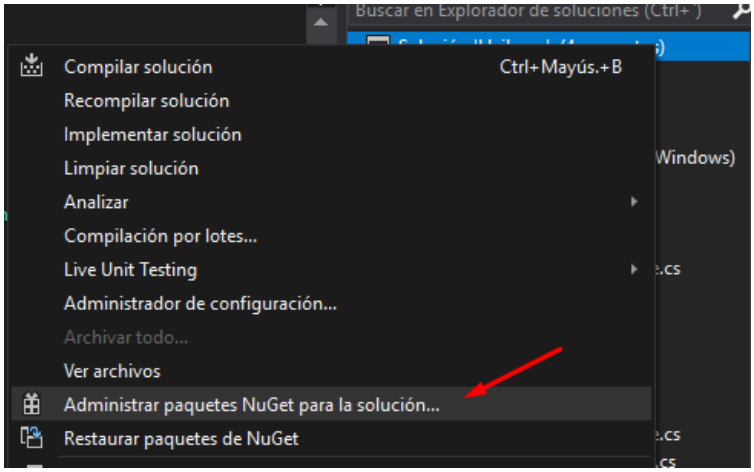
Como he dicho anteriormente, empezaremos a hacer la aplicación **SQLite** ya que será mucho más fácil para empezar.

El seguimiento que haremos es añadir el diseño de cómo queremos hacer la aplicación, después crearemos el fichero para agregar **nuestra base de datos** y creas las **funciones necesarias** y después relacionar estas funciones a nuestro código dentro de cada fichero **.xaml** y **.cs**.

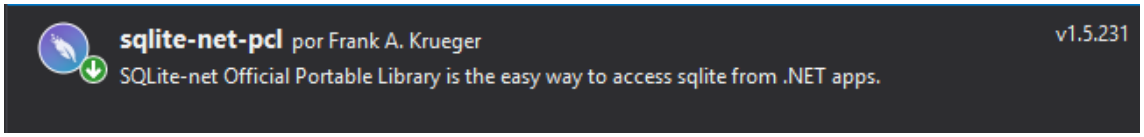
Aquí dejo un esquema de los ficheros que crearemos y utilizaremos:



Como esta aplicación es con SQLite, vamos a descargarnos el paquete de éste para poder utilizarlo. Para ello vamos a la solución del proyecto y botón derecho en **Administrar paquetes NuGet para la solución…**



Dentro, en la pestaña de examinar, buscamos "sqlite-net-pcl" y nos lo instalamos para toda la aplicación:



2.1. HaikyuuItemDatabase.cs

Este es el fichero más importante de toda nuestra aplicación, porque es donde creamos las funciones para utilizar nuestra Base de Datos.

```
1.  using System.Collections.Generic;
2.  using System.Threading.Tasks;
3.  using SQLite;
4.
5.  using Haikyuu.Models;
6.
7.  namespace Haikyuu
8.  {
9.      public class HaikyuuItemDatabase
10.     {
11.         readonly SQLiteAsyncConnection database;
12.
13.         public HaikyuuItemDatabase(string dbPath)
14.         {
15.             database = new SQLiteAsyncConnection(dbPath);
16.             database.CreateTableAsync<Comentario>().Wait();
17.             database.CreateTableAsync<Personaje>().Wait();
18.         }
19.
20.         public Task<List<Comentario>> GetItemsAsync()
21.         {
22.             Task<List<Comentario>> firstList;
23.             firstList = database.Table<Comentario>().ToListAsync();
24.
25.             return firstList;
26.         }
27.
28.         public Task<List<Personaje>> GetItemsAsyncPersonaje()
29.         {
30.             Task<List<Personaje>> firstList;
31.             firstList = database.Table<Personaje>().ToListAsync();
32.
33.             return firstList;
34.         }
35.
36.         public Task<Comentario> GetItemAsync(int id)
37.         {
38.             return database.Table<Comentario>().Where(i => i.id == id).FirstOr
    DefaultAsync();
39.         }
40.
41.         public Task<int> SaveItemAsync(Comentario item)
42.         {
43.             if (item.id != 0)
44.             {
45.                 return database.UpdateAsync(item);
46.             }
47.             else
48.             {
49.                 return database.InsertAsync(item);
50.             }
51.         }
52.
53.         public Task<int> SaveItemAsyncPersonaje(Personaje item)
54.         {
55.             if (item.id != 0)
56.             {
57.                 return database.UpdateAsync(item);
```

```
            }
            else
            {
                return database.InsertAsync(item);
            }
        }

        public Task<int> DeleteItemAsync(Comentario item)
        {
            return database.DeleteAsync(item);
        }
    }
}
```

2.2. Models

Definimos como queremos que tenga cada modelo.

1.2.1 Comentario.cs

```
using SQLite;

namespace Haikyuu.Models
{
    public class Comentario
    {
        [PrimaryKey, AutoIncrement]
        public int id { get; set; }
        public string Name { get; set; }
        public string NameCharacter { get; set; }
        public string Notes { get; set; }
    }
}
```

1.2.2 Personaje.cs

```
using SQLite;

namespace Haikyuu.Models
{
    public class Personaje
    {
        [PrimaryKey, AutoIncrement]
        public int id { get; set; }
        public string Name { get; set; }
        public string Location { get; set; }
        public string Details { get; set; }
        public string numPuntuacion { get; set; }
        public string Image { get; set; }
    }
}
```

2.3. Services

1.2.3 Navigation

1.2.3.1 INavigationService.cs

```
using System;

namespace Haikyuu.Services.Navigation
{
    public interface INavigationService
    {
        void NavigateTo<TDestinationViewModel>(object navigationContext = null
);

        void NavigateTo(Type destinationType, object navigationContext = null)
;

        void NavigateBack();
    }
}
```

1.2.3.2 NavigationService.cs

```
namespace Haikyuu.Services.Navigation
{
    using System;
    using System.Collections.Generic;
    using ViewModels;
    using Views;
    using Xamarin.Forms;

    public class NavigationService : INavigationService
    {
        private IDictionary<Type, Type> viewModelRouting = new Dictionary<Type
, Type>()
        {
            { typeof(MainViewModel),  typeof(MainView) },
            { typeof(CustomCellViewModel), typeof(CustomCellView) }
        };

        public void NavigateTo<TDestinationViewModel>(object navigationContext
 = null)
        {
            Type pageType = viewModelRouting[typeof(TDestinationViewModel)];
            var page = Activator.CreateInstance(pageType, navigationContext) a
s Page;

            if (page != null)
                Application.Current.MainPage.Navigation.PushAsync(page);
        }

        public void NavigateTo(Type destinationType, object navigationContext
= null)
        {
            Type pageType = viewModelRouting[destinationType];
            var page = Activator.CreateInstance(pageType, navigationContext) a
s Page;

            if (page != null)
                Application.Current.MainPage.Navigation.PushAsync(page);
        }

```

```
35.         public void NavigateBack()
36.         {
37.             Application.Current.MainPage.Navigation.PopAsync();
38.         }
39.     }
40. }
```

2.4. ViewModels

1.2.4 Base

1.2.4.1 DelegateCommand.cs

```
1.  using System;
2.  using System.Windows.Input;
3.
4.  namespace Haikyuu.ViewModels.Base
5.  {
6.      /// <summary>
7.      /// This class allows us to delegate command execution to viewmodels.
8.      /// </summary>
9.      public class DelegateCommand : ICommand
10.     {
11.         private readonly Action _execute;
12.         private readonly Func<bool> _canExecute;
13.
14.         /// <summary>
15.         /// Constructor not using canExecute.
16.         /// </summary>
17.         /// <param name="execute"></param>
18.         public DelegateCommand(Action execute) : this(execute, null) { }
19.
20.         /// <summary>
21.         /// Constructor using both execute and canExecute.
22.         /// </summary>
23.         /// <param name="execute"></param>
24.         /// <param name="canExecute"></param>
25.         public DelegateCommand(Action execute, Func<bool> canExecute)
26.         {
27.             _execute = execute;
28.             _canExecute = canExecute;
29.         }
30.
31.         /// <summary>
32.         /// This method is called from XAML to evaluate if the command can be
    executed.
33.         /// </summary>
34.         /// <param name="parameter"></param>
35.         /// <returns></returns>
36.         public bool CanExecute(object parameter)
37.         {
38.             if (_canExecute != null)
39.                 return _canExecute();
40.
41.             return true;
42.         }
43.
44.         /// <summary>
45.         /// This method is called from XAML to execute the command.
46.         /// </summary>
47.         /// <param name="parameter"></param>
48.         public void Execute(object parameter)
49.         {
50.             _execute();
```

```
51.         }
52.
53.         /// <summary>
54.         /// This method allow us to force the execution of CanExecute method t
    o reevaluate execution.
55.         /// </summary>
56.         public void RaiseCanExecuteChanged()
57.         {
58.             var tmpHandle = CanExecuteChanged;
59.             if (tmpHandle != null)
60.                 tmpHandle(this, new EventArgs());
61.         }
62.
63.         /// <summary>
64.         /// This event notify XAML controls using the command to reevaluate th
    e CanExecute of it.
65.         /// </summary>
66.         public event EventHandler CanExecuteChanged;
67.     }
68.
69.     /// <summary>
70.     /// This class allows us to delegate command execution to viewmodels using
     a T type as parameter.
71.     /// </summary>
72.     public class DelegateCommand<T> : ICommand
73.     {
74.         private readonly Action<T> _execute;
75.         private readonly Func<T, bool> _canExecute;
76.
77.         /// <summary>
78.         /// Constructor not using canExecute.
79.         /// </summary>
80.         /// <param name="execute"></param>
81.         public DelegateCommand(Action<T> execute) : this(execute, null) { }
82.
83.         /// <summary>
84.         /// Constructor using both execute and canExecute.
85.         /// </summary>
86.         /// <param name="execute"></param>
87.         /// <param name="canExecute"></param>
88.         public DelegateCommand(Action<T> execute, Func<T, bool> canExecute)
89.         {
90.             _execute = execute;
91.             _canExecute = canExecute;
92.         }
93.
94.         /// <summary>
95.         /// This method is called from XAML to evaluate if the command can be
    executed.
96.         /// </summary>
97.         /// <param name="parameter"></param>
98.         /// <returns></returns>
99.         public bool CanExecute(object parameter)
100.                {
101.                    if (_canExecute != null)
102.                        return _canExecute((T)parameter);
103.
104.                    return true;
105.                }
106.
107.                /// <summary>
108.                /// This method is called from XAML to execute the command.
109.                /// </summary>
110.                /// <param name="parameter"></param>
111.                public void Execute(object parameter)
112.                {
```

```
113.                     _execute((T)parameter);
114.                 }
115.
116.                 /// <summary>
117.                 /// This method allow us to force the execution of CanExecute m
    ethod to reevaluate execution.
118.                 /// </summary>
119.                 public void RaiseCanExecuteChanged()
120.                 {
121.                     var tmpHandle = CanExecuteChanged;
122.                     if (tmpHandle != null)
123.                         tmpHandle(this, new EventArgs());
124.                 }
125.
126.                 /// <summary>
127.                 /// This event notify XAML controls using the command to reeval
    uate the CanExecute of it.
128.                 /// </summary>
129.                 public event EventHandler CanExecuteChanged;
130.             }
131.         }
```

1.2.4.2 DelegateCommandAsync.cs

```
1.  using System;
2.  using System.Threading.Tasks;
3.  using System.Windows.Input;
4.
5.  namespace Haikyuu.ViewModels.Base
6.  {
7.      /// <summary>
8.      ///     This class allows us to delegate command execution to viewmodels.

9.      ///     This version of the command allow to use async/await.
10.     /// </summary>
11.     public class DelegateCommandAsync : ICommand
12.     {
13.         private readonly Func<Task<bool>> _canExecute;
14.         private readonly Func<Task> _execute;
15.
16.         /// <summary>
17.         ///     Constructor not using canExecute.
18.         /// </summary>
19.         /// <param name="execute"></param>
20.         public DelegateCommandAsync(Func<Task> execute) : this(execute, null)

21.         {
22.         }
23.
24.         /// <summary>
25.         ///     Constructor using both execute and canExecute.
26.         /// </summary>
27.         /// <param name="execute"></param>
28.         /// <param name="canExecute"></param>
29.         public DelegateCommandAsync(Func<Task> execute, Func<Task<bool>> canEx
    ecute)
30.         {
31.             _execute = execute;
32.             _canExecute = canExecute;
33.         }
34.
35.         /// <summary>
36.         ///     This method is called from XAML to evaluate if the command can
     be executed.
```

```
37.         /// </summary>
38.         /// <param name="parameter"></param>
39.         /// <returns></returns>
40.         public bool CanExecute(object parameter)
41.         {
42.             if (_canExecute != null)
43.                 return _canExecute().Result;
44.
45.             return true;
46.         }
47.
48.         /// <summary>
49.         ///     This method is called from XAML to execute the command.
50.         /// </summary>
51.         /// <param name="parameter"></param>
52.         public void Execute(object parameter)
53.         {
54.             _execute();
55.         }
56.
57.         /// <summary>
58.         ///     This event notify XAML controls using the command to reevaluat
    e the CanExecute of it.
59.         /// </summary>
60.         public event EventHandler CanExecuteChanged;
61.
62.         /// <summary>
63.         ///     This method allow us to force the execution of CanExecute meth
    od to reevaluate execution.
64.         /// </summary>
65.         public void RaiseCanExecuteChanged()
66.         {
67.             EventHandler tmpHandle = CanExecuteChanged;
68.             if (tmpHandle != null)
69.                 tmpHandle(this, new EventArgs());
70.         }
71.     }
72.
73.     /// <summary>
74.     ///     This class allows us to delegate command execution to viewmodels u
    sing a T type as parameter.
75.     /// </summary>
76.     public class DelegateCommandAsync<T> : ICommand
77.     {
78.         private readonly Func<T, Task<bool>> _canExecute;
79.         private readonly Func<T, Task> _execute;
80.
81.         /// <summary>
82.         ///     Constructor not using canExecute.
83.         /// </summary>
84.         /// <param name="execute"></param>
85.         public DelegateCommandAsync(Func<T, Task> execute) : this(execute, nul
    l)
86.         {
87.         }
88.
89.         /// <summary>
90.         ///     Constructor using both execute and canExecute.
91.         /// </summary>
92.         /// <param name="execute"></param>
93.         /// <param name="canExecute"></param>
94.         public DelegateCommandAsync(Func<T, Task> execute, Func<T, Task<bool>>
     canExecute)
95.         {
96.             _execute = execute;
97.             _canExecute = canExecute;
```

```
98.         }
99.
100.                /// <summary>
101.                ///     This method is called from XAML to evaluate if the comm
    and can be executed.
102.                /// </summary>
103.                /// <param name="parameter"></param>
104.                /// <returns></returns>
105.                public bool CanExecute(object parameter)
106.                {
107.                    if (_canExecute != null)
108.                        return _canExecute((T) parameter).Result;
109.
110.                    return true;
111.                }
112.
113.                /// <summary>
114.                ///     This method is called from XAML to execute the command.

115.                /// </summary>
116.                /// <param name="parameter"></param>
117.                public void Execute(object parameter)
118.                {
119.                    _execute((T) parameter);
120.                }
121.
122.                /// <summary>
123.                ///     This event notify XAML controls using the command to re
    evaluate the CanExecute of it.
124.                /// </summary>
125.                public event EventHandler CanExecuteChanged;
126.
127.                /// <summary>
128.                ///     This method allow us to force the execution of CanExecu
    te method to reevaluate execution.
129.                /// </summary>
130.                public void RaiseCanExecuteChanged()
131.                {
132.                    EventHandler tmpHandle = CanExecuteChanged;
133.                    if (tmpHandle != null)
134.                        tmpHandle(this, new EventArgs());
135.                }
136.            }
137.        }
```

1.2.4.3 ViewModelBase.cs

```
1.  using System.ComponentModel;
2.  using System.Runtime.CompilerServices;
3.
4.  namespace Haikyuu.ViewModels.Base
5.  {
6.      public abstract class ViewModelBase : INotifyPropertyChanged
7.      {
8.          public virtual void OnAppearing(object navigationContext)
9.          {
10.         }
11.
12.         public virtual void OnDisappearing()
13.         {
14.         }
15.
16.         public event PropertyChangedEventHandler PropertyChanged;
17.
18.         public void RaisePropertyChanged([CallerMemberName]string propertyName
    = "")
19.         {
20.             var handler = PropertyChanged;
21.             if (handler != null)
22.                 handler(this, new PropertyChangedEventArgs(propertyName));
23.         }
24.     }
25. }
```

1.2.4.4 ViewModelLocator.cs

```
1.  using Unity;
2.  using Haikyuu.Services.Navigation;
3.
4.  namespace Haikyuu.ViewModels.Base
5.  {
6.      public class ViewModelLocator
7.      {
8.          readonly IUnityContainer _container;
9.
10.         public ViewModelLocator()
11.         {
12.             _container = new UnityContainer();
13.
14.             // ViewModels
15.             _container.RegisterType<CustomCellViewModel>();
16.             _container.RegisterType<MainViewModel>();
17.
18.             // Services
19.             _container.RegisterType<INavigationService, NavigationService>();

20.         }
21.
22.         public CustomCellViewModel CustomCellViewModel
23.         {
24.             get { return _container.Resolve<CustomCellViewModel>(); }
25.         }
26.
27.         public MainViewModel MainViewModel
28.         {
29.             get { return _container.Resolve<MainViewModel>(); }
30.         }
31.     }
```

```
}
```

1.2.5 CustomCellViewModel.cs

```
using Haikyuu.Models;
using Haikyuu.ViewModels.Base;
using System.Collections.ObjectModel;

namespace Haikyuu.ViewModels
{
    public class CustomCellViewModel : ViewModelBase
    {
        public ObservableCollection<Personaje> Personajes { get; set; }

        public CustomCellViewModel()
        {
            Personajes = new ObservableCollection<Personaje>
            {
                new Personaje
                {
                    Name = "Shoyo Hinata",
                    Location = "Bloquejador central, Carnada",
                    Details =
                        "Vol arribar a ser com el Petit Gegant",
                    Image =
                        "https://vignette.wikia.nocookie.net/haikyuu/images/c/c2/Rostro_de_Hinata.png/revision/latest?cb=20160210050706&path-prefix=es"
                },
                new Personaje
                {
                    Name = "Tobio Kageyama",
                    Location = "Col·locador",
                    Details =
                        "Juntament amb Hinata, un parella imparable",
                    Image =
                        "https://vignette.wikia.nocookie.net/haikyuu-pedia/images/4/47/Tobio_Kageyama.jpg/revision/latest?cb=20140721131354"
                },
                new Personaje
                {
                    Name = "Daichi Sawamura",
                    Location = "Wing Spikers, Capità",
                    Details =
                        "Especialista en la defensa i una persona necessària per al equip",
                    Image =
                        "https://em.wattpad.com/8bd5055f8ed26c2660a747809f9d943e64c8cfe4/687474703a2f2f36382e6d656469612e74756d626c722e636f6d2f6637636266383238323361303463353334373366656137653033626239393330 2f74756d626c725f696e6c696e655f6e683373646f464d714a3172696e3631382e6a7067?s=fit&h=758&w=1583&q=80"
                },
                new Personaje
                {
                    Name = "Koshi Sugawara",
                    Location = "Segon col·locador, Segon capità",
                    Details =
                        "No serà titular, però es un dels millors en la seva feina",
                    Image =
                        "https://vignette.wikia.nocookie.net/haikyuu/images/8/8d/Sugawara.png/revision/latest?cb=20150821230947&path-prefix=es"
                },
                new Personaje
                {
```

```
53.                     Name = "Asahi Azumane",
54.                     Location = "Estrella",
55.                     Details =
56.                         "A qui pots recórrer quan estàs en problemes",
57.                     Image =
58.                         "https://vignette.wikia.nocookie.net/haikyuu-
    pedia/images/c/c4/Asahi_Azumane.jpg/revision/latest?cb=20140721131410"
59.                 },
60.                 new Personaje
61.                 {
62.                     Name = "Yu Nishinoya",
63.                     Location = "Libero",
64.                     Details =
65.                         "El millor en salvar les pilotes impossibles",
66.                     Image =
67.                         "https://vignette.wikia.nocookie.net/haikyuu/images/8/
    80/Y%C5%AB_Nishinoya.png/revision/latest?cb=20170809205325&path-prefix=es"
68.                 },
69.                 new Personaje
70.                 {
71.                     Name = "Ryunosuke Tanaka",
72.                     Location = "Wing Spikers",
73.                     Details =
74.                         "Vol ser la estrella i té le poder per ser-ho",
75.                     Image =
76.                         "https://vignette.wikia.nocookie.net/haikyuu/images/e/
    e7/RyuTanaka.png/revision/latest?cb=20150107144228&path-prefix=es"
77.                 },
78.                 new Personaje
79.                 {
80.                     Name = "Kei Tsukishima",
81.                     Location = "Bloquejador central",
82.                     Details =
83.                         "No creguis que només bloqueja, també pensa",
84.                     Image =
85.                         "https://vignette.wikia.nocookie.net/haikyuu/images/f/
    f9/Kei_Tsukishima.png/revision/latest?cb=20170822215947&path-prefix=es"
86.                 },
87.                 new Personaje
88.                 {
89.                     Name = "Tadashi Yamaguchi",
90.                     Location = "Pinch Server",
91.                     Details =
92.                         "Encara que pensis que no fa res, ell es qui crea opor
    tunitats",
93.                     Image =
94.                         "https://pbs.twimg.com/media/C2VZ_kGXcAE3oS2.jpg"
95.                 },
96.                 new Personaje
97.                 {
98.                     Name = "Keishin Ukai",
99.                     Location = "Entrenador",
100.                        Details =
101.                            "Antic jugador del Karasuno i el seu avi va ent
    renar al Petit Gegant",
102.                        Image =
103.                            "https://vignette.wikia.nocookie.net/haikyuu/im
    ages/4/44/KeishinUkai.png/revision/latest?cb=20140616134254"
104.                    },
105.                    new Personaje
106.                    {
107.                        Name = "Ittetsu Takeda",
108.                        Location = "Professor",
109.                        Details =
110.                            "Gràcies a ell Karasuno obté la seva força",
111.                        Image =
```

```
                                "https://vignette.wikia.nocookie.net/haikyuu/images/d/da/Takeda-sensei.png/revision/latest?cb=20140616135813"
                        }
                    };
                }
            }
        }
```

1.2.6 MainViewModel.cs

```
using Haikyuu.Services.Navigation;
using Haikyuu.ViewModels.Base;
using System.Windows.Input;

namespace Haikyuu.ViewModels
{
    public class MainViewModel : ViewModelBase
    {
        private ICommand _customCellCommand;

        private INavigationService _navigationService;

        public MainViewModel(INavigationService navigationService)
        {
            _navigationService = navigationService;
        }

        public ICommand CustomCellCommand
        {
            get { return _customCellCommand = _customCellCommand ?? new DelegateCommand(CustomCellCommandExecute); }
        }
        private void CustomCellCommandExecute()
        {
            _navigationService.NavigateTo<CustomCellViewModel>();
        }
    }
}
```

2.5. Views

1.2.7 CustomCellView.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="Haikyuu.Views.CustomCellView"
             Title="Haikyuu">
    <ListView
        ItemsSource="{Binding Personajes}"
        RowHeight="200"
        x:Name="ListaPersonajes">
        <ListView.ItemTemplate>
            <DataTemplate>
                <ViewCell>
                    <StackLayout
                        Orientation="Horizontal">
                        <Image
                            Source="{Binding Image}"
                            Aspect="AspectFill"
                            WidthRequest="380"/>
                    </StackLayout>
                </ViewCell>
            </DataTemplate>
        </ListView.ItemTemplate>
    </ListView>
</ContentPage>
```

1.2.7.1 CustomCellView.xaml.cs

```
using System;
using Haikyuu.Models;
using Haikyuu.ViewModels;
using Xamarin.Forms;

namespace Haikyuu.Views
{
    public partial class CustomCellView : ContentPage
    {
        private object Parameter { get; set; }

        public CustomCellView(object parameter)
        {
            InitializeComponent();

            ListaPersonajes.ItemSelected += ListaPersonajes_ItemSelected;
            BindingContext = App.Locator.CustomCellViewModel;

            Parameter = parameter;
        }

        private void ListaPersonajes_ItemSelected(object sender, SelectedItemC
hangedEventArgs e)
        {
            Navigation.PushAsync(new MainView((Personaje)e.SelectedItem));
        }

        protected override void OnAppearing()
        {
```

```
            var viewModel = BindingContext as CustomCellViewModel;
            if (viewModel != null) viewModel.OnAppearing(Parameter);
        }

        protected override void OnDisappearing()
        {
            var viewModel = BindingContext as CustomCellViewModel;
            if (viewModel != null) viewModel.OnDisappearing();
        }
    }
}
```

1.2.8 HaikyuuItemPage.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="Haikyuu.Views.HaikyuuItemPage"
             BackgroundColor="#313445">
    <StackLayout Margin="10" VerticalOptions="StartAndExpand">
        <Label Text="Nombre" FontSize="Large" TextColor="Orange" Margin="20"/>

        <Entry Text="{Binding Name}" TextColor="White"/>
        <Label Text="Comentario" FontSize="Large" TextColor="Orange" Margin="2
0"/>
        <Entry Text="{Binding Notes}" TextColor="White" HeightRequest="150"/>

        <Button Text="Save" Clicked="OnSaveClicked" BackgroundColor="Orange" T
extColor="Black"/>
        <Button Text="Delete" Clicked="OnDeleteClicked" BackgroundColor="Orang
e" TextColor="Black"/>
        <Button Text="Cancel" Clicked="OnCancelClicked" BackgroundColor="Orang
e" TextColor="Black"/>
    </StackLayout>
</ContentPage>
```

1.2.8.1 HaikyuuItemPage.xaml.cs

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

using Haikyuu.Models;

namespace Haikyuu.Views
{
    [XamlCompilation(XamlCompilationOptions.Compile)]
    public partial class HaikyuuItemPage : ContentPage
    {
        string nombrePersonaje;

        public HaikyuuItemPage (string _nombrePersonaje)
        {
            InitializeComponent ();

            nombrePersonaje = _nombrePersonaje;

            Title = nombrePersonaje;
        }

        async void OnSaveClicked(object sender, EventArgs e)
        {
            var todoItem = (Comentario)BindingContext;
```

```
26.
27.             todoItem.NameCharacter = nombrePersonaje;
28.
29.             await App.Database.SaveItemAsync(todoItem);
30.             await Navigation.PopAsync();
31.         }
32.
33.         async void OnDeleteClicked(object sender, EventArgs e)
34.         {
35.             var todoItem = (Comentario)BindingContext;
36.
37.             todoItem.NameCharacter = nombrePersonaje;
38.
39.             await App.Database.DeleteItemAsync(todoItem);
40.             await Navigation.PopAsync();
41.         }
42.
43.         async void OnCancelClicked(object sender, EventArgs e)
44.         {
45.             await Navigation.PopAsync();
46.         }
47.     }
48. }
```

1.2.8.2 HaikyuuItemPageCS.cs

```
1.  using Xamarin.Forms;
2.
3.  using Haikyuu.Models;
4.
5.  namespace Haikyuu.Views
6.  {
7.      public class HaikyuuItemPageCS : ContentPage
8.      {
9.          public HaikyuuItemPageCS()
10.         {
11.             Title = "Haikyuu Item";
12.
13.             var nameEntry = new Entry();
14.             nameEntry.SetBinding(Entry.TextProperty, "Name");
15.
16.             var notesEntry = new Entry();
17.             notesEntry.SetBinding(Entry.TextProperty, "Notes");
18.
19.             var saveButton = new Button { Text = "Save" };
20.             saveButton.Clicked += async (sender, e) =>
21.             {
22.                 var comment = (Comentario)BindingContext;
23.                 await App.Database.SaveItemAsync(comment);
24.                 await Navigation.PopAsync();
25.             };
26.
27.             var deleteButton = new Button { Text = "Delete" };
28.             deleteButton.Clicked += async (sender, e) =>
29.             {
30.                 var comment = (Comentario)BindingContext;
31.                 await App.Database.DeleteItemAsync(comment);
32.                 await Navigation.PopAsync();
33.             };
34.
35.             var cancelButton = new Button { Text = "Cancel" };
36.             cancelButton.Clicked += async (sender, e) =>
37.             {
38.                 await Navigation.PopAsync();
```

```
            };

            Content = new StackLayout
            {
                Margin = new Thickness(20),
                VerticalOptions = LayoutOptions.StartAndExpand,
                Children =
                {
                    new Label { Text = "Name" },
                    nameEntry,
                    new Label { Text = "Notes" },
                    notesEntry,
                    saveButton,
                    deleteButton,
                    cancelButton
                }
            };
        }
    }
}
```

1.2.9 MainView.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="Haikyuu.Views.MainView"
             Title="Haikyuu">
    <ContentPage.ToolbarItems>
        <ToolbarItem Text="+" Clicked="OnItemAdded">
            <ToolbarItem.Icon>
                <OnPlatform x:TypeArguments="FileImageSource">
                    <On Platform="Android, UWP" Value="plus.png" />
                </OnPlatform>
            </ToolbarItem.Icon>
        </ToolbarItem>
    </ContentPage.ToolbarItems>
    <ContentPage.Content>
        <ScrollView>
            <StackLayout
                BackgroundColor="#313445">
                <Label x:Name="Name"
                       FontSize="Large"
                       Text="Name"
                       TextColor="Orange"
                       Margin="20"
                       />

                <Label Text="Posición"
                       FontSize="Large"
                       BackgroundColor="Orange"
                       TextColor="#313445"
                       />

                <Label x:Name="Location"
                       FontSize="Medium"
                       Text="Location"
                       TextColor="White"
                       Margin="15"
                       />

                <Label Text="Detalles"
                       FontSize="Large"
                       BackgroundColor="Orange"
```

```
                        TextColor="#313445"
                        />

                <Label x:Name="Details"
                        Text="Details"
                        FontSize="Medium"
                        TextColor="White"
                        Margin="15"
                        />

                <!-- Puntuacion -->
                <Label Text="Puntuación"
                        FontSize="Large"
                        BackgroundColor="Orange"
                        TextColor="#313445"
                        />

                <Slider x:Name="sliderPuntuacion"
                         Margin="20"
                         Maximum="10"
                         Minimum="0"
                         ThumbColor="White"
                         />

                <Label x:Name="numPuntuacion"
                        FontSize="35"
                        TextColor="White"
                        Margin="170, 0, 0, 20"
                        />

                <Image x:Name="Foto"
                        Aspect="AspectFill"
                        WidthRequest="380"
                        MinimumHeightRequest="500"
                        />

                <!--Comentarios-->
                <Label Text="Comentarios"
                        FontSize="Large"
                        BackgroundColor="Orange"
                        TextColor="#313445"
                        />
                <ListView x:Name="listView" Margin="20" ItemSelected="OnListItemSelected">
                    <ListView.ItemTemplate>
                        <DataTemplate>
                            <ViewCell>
                                <StackLayout Margin="20,0,0,0" Orientation="Horizontal" HorizontalOptions="FillAndExpand">
                                    <Label Text="{Binding Name}"
                                           VerticalTextAlignment="Center"
                                           HorizontalOptions="StartAndExpand"

                                           FontSize="Medium"
                                           TextColor="White"
                                           />
                                </StackLayout>
                            </ViewCell>
                        </DataTemplate>
                    </ListView.ItemTemplate>
                </ListView>
                   </StackLayout>
              </ScrollView>
          </ContentPage.Content>
      </ContentPage>
```

1.2.9.1 MainView.xaml.cs

```
1.  using Xamarin.Forms;
2.  using Xamarin.Forms.Xaml;
3.  using Haikyuu.Models;
4.  using System;
5.  using System.Collections.Generic;
6.  using System.Threading.Tasks;
7.  using System.Linq;
8.
9.  namespace Haikyuu.Views
10. {
11.     [XamlCompilation(XamlCompilationOptions.Compile)]
12.     public partial class MainView : ContentPage
13.     {
14.         public MainView(Personaje parameter)
15.         {
16.             InitializeComponent();
17.
18.             Name.Text = parameter.Name;
19.             Location.Text = parameter.Location;
20.             Details.Text = parameter.Details;
21.             numPuntuacion.Text = parameter.numPuntuacion;
22.             Foto.Source = parameter.Image;
23.
24.             obtenirPuntuacio();
25.
26.             sliderPuntuacion.ValueChanged += SliderPuntuacion_ValueChanged;
27.
28.             BindingContext = App.Locator.MainViewModel;
29.         }
30.
31.         private void obtenirPuntuacio()
32.         {
33.             List<Personaje> personaje;
34.
35.             personaje = App.Database.GetItemsAsyncPersonaje().Result;
36.             personaje = personaje.Where(r => r.Name == Name.Text).ToList();
37.
38.             Personaje psj = personaje.FirstOrDefault();
39.
40.             numPuntuacion.Text = psj.numPuntuacion.ToString();
41.             sliderPuntuacion.Value = Double.Parse(psj.numPuntuacion);
42.         }
43.
44.         private void SliderPuntuacion_ValueChanged(object sender, ValueChanged
    EventArgs e)
45.         {
46.             numPuntuacion.Text = (Convert.ToInt32(sliderPuntuacion.Value)).ToS
    tring();
47.
48.             List<Personaje> listaPersonaje;
49.
50.             listaPersonaje = App.Database.GetItemsAsyncPersonaje().Result;
51.
52.             Personaje personaje = listaPersonaje.Where(a => a.Name == Name.Tex
    t).FirstOrDefault();
53.
54.             personaje.numPuntuacion = numPuntuacion.Text.ToString();
55.
56.             App.Database.SaveItemAsyncPersonaje(personaje);
57.         }
58.
59.         protected override async void OnAppearing()
60.         {
```

```
61.             base.OnAppearing();
62.
63.             // Reset the 'resume' id, since we just want to re-start here
64.             ((App)App.Current).ResumeAtTodoId = -1;
65.             //listView.ItemsSource = await App.Database.GetItemsAsync();
66.
67.             List<Comentario> listaComentarios;
68.
69.             listaComentarios = App.Database.GetItemsAsync().Result;
70.
71.             listaComentarios = listaComentarios.Where( a => a.NameCharacter ==
    Name.Text ).ToList();
72.
73.             listView.ItemsSource = listaComentarios;
74.
75.
76.             List<Personaje> listaPersonaje;
77.
78.             listaPersonaje = App.Database.GetItemsAsyncPersonaje().Result;
79.
80.             Personaje personaje = listaPersonaje.Where(a => a.Name == Name.Tex
   t).FirstOrDefault();
81.
82.             numPuntuacion.Text = personaje.numPuntuacion.ToString();
83.         }
84.
85.         async void OnItemAdded(object sender, EventArgs e)
86.         {
87.             await Navigation.PushAsync(new HaikyuuItemPage(Name.Text)
88.             {
89.                 BindingContext = new Comentario()
90.             });
91.         }
92.
93.         async void OnListItemSelected(object sender, SelectedItemChangedEventA
   rgs e)
94.         {
95.             ///((App)App.Current).ResumeAtTodoId = (e.SelectedItem as TodoItem)
   .ID;
96.             //Debug.WriteLine("setting ResumeAtTodoId = " + (e.SelectedItem as
    TodoItem).ID
97.
98.             if (e.SelectedItem != null)
99.             {
100.                         await Navigation.PushAsync(new HaikyuuItemPage(Name.Tex
    t)
101.                         {
102.                             BindingContext = e.SelectedItem as Comentario
103.                         });
104.                     }
105.                 }
106.             }
107.         }
```

1.2.9.2 MainViewCS.cs

```
1.  using Haikyuu.Models;
2.  using System.Collections.Generic;
3.  using System.Linq;
4.  using Xamarin.Forms;
5.
6.  namespace Haikyuu.Views
7.  {
8.      public class MainViewCS : ContentPage
9.      {
10.         ListView listView;
11.
12.         public MainViewCS()
13.         {
14.             Title = "Haikyuu";
15.
16.             var toolbarItem = new ToolbarItem
17.             {
18.                 Text = "+",
19.                 Icon = Device.RuntimePlatform == Device.iOS ? null : "plus.png
    "
20.             };
21.             toolbarItem.Clicked += async (sender, e) =>
22.             {
23.                 await Navigation.PushAsync(new HaikyuuItemPageCS
24.                 {
25.                     BindingContext = new Comentario()
26.                 });
27.             };
28.             ToolbarItems.Add(toolbarItem);
29.
30.             listView = new ListView
31.             {
32.                 Margin = new Thickness(10),
33.                 ItemTemplate = new DataTemplate(() =>
34.                 {
35.                     var label = new Label
36.                     {
37.                         VerticalTextAlignment = TextAlignment.Center,
38.                         HorizontalOptions = LayoutOptions.StartAndExpand
39.                     };
40.                     label.SetBinding(Label.TextProperty, "Name");
41.
42.                     var stackLayout = new StackLayout
43.                     {
44.                         Margin = new Thickness(10, 0, 0, 0),
45.                         Orientation = StackOrientation.Horizontal,
46.                         HorizontalOptions = LayoutOptions.FillAndExpand,
47.                         Children = { label }
48.                     };
49.
50.                     return new ViewCell { View = stackLayout };
51.                 })
52.             };
53.             listView.ItemSelected += async (sender, e) =>
54.             {
55.                 //((App)App.Current).ResumeAtTodoId = (e.SelectedItem as TodoI
    tem).ID;
56.                 //Debug.WriteLine("setting ResumeAtTodoId = " + (e.SelectedIte
    m as TodoItem).ID);
57.
58.                 if (e.SelectedItem != null)
59.                 {
60.                     await Navigation.PushAsync(new HaikyuuItemPageCS
```

```
                    {
                        BindingContext = e.SelectedItem as Comentario
                    });
                }
            };

            Content = listView;
        }

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

            // Reset the 'resume' id, since we just want to re-start here
            ((App)App.Current).ResumeAtTodoId = -1;
            listView.ItemsSource = await App.Database.GetItemsAsync();
        }
    }
}
```

1.2.10 SplashPage.xaml.cs

```
using Haikyuu.Models;
using System.Collections.Generic;

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

namespace Haikyuu.Views
{
    [XamlCompilation(XamlCompilationOptions.Compile)]
    public partial class SplashPage : ContentPage
    {
        Image splashImage;
        public SplashPage ()
        {
            InitializeComponent();


            List<Personaje> listaPersonaje;

            listaPersonaje = App.Database.GetItemsAsyncPersonaje().Result;

            if (listaPersonaje.Count == 0)
            {
                Personaje personaje1 = new Personaje();

                personaje1.Name = "Shoyo Hinata";
                personaje1.Location = "Bloquejador central, Carnada";
                personaje1.Details = "Vol arribar a ser com el Petit Gegant";

                personaje1.numPuntuacion = "5";
                personaje1.Image = "https://vignette.wikia.nocookie.net/haikyuu/images/c/c2/Rostro_de_Hinata.png/revision/latest?cb=20160210050706&path-prefix=es";

                App.Database.SaveItemAsyncPersonaje(personaje1);


                Personaje personaje2 = new Personaje();

                personaje2.Name = "Tobio Kageyama";
                personaje2.Location = "Col·locador";
```

```
39.                 personaje2.Details = "Juntament amb Hinata, un parella imparab
    le";
40.                 personaje2.numPuntuacion = "5";
41.                 personaje2.Image = "https://vignette.wikia.nocookie.net/haikyu
    u-pedia/images/4/47/Tobio_Kageyama.jpg/revision/latest?cb=20140721131354";
42.
43.                 App.Database.SaveItemAsyncPersonaje(personaje2);
44.
45.
46.                 Personaje personaje3 = new Personaje();
47.
48.                 personaje3.Name = "Daichi Sawamura";
49.                 personaje3.Location = "Wing Spikers, Capità";
50.                 personaje3.Details = "Especialista en la defensa i una persona
     necessària per al equip";
51.                 personaje3.numPuntuacion = "5";
52.                 personaje3.Image = "https://em.wattpad.com/8bd5055f8ed26c2660a
    747809f9d943e64c8cfe4/687474703a2f2f36382e6d656469612e74756d626c722e636f6d2f66
    3763626638323832336130346335333437336665613765303362623939333302f74756d626c725f
    696e6c696e655f6e683373646f464d714a3172696e3631382e6a7067?s=fit&h=758&w=1583&q=
    80";
53.
54.                 App.Database.SaveItemAsyncPersonaje(personaje3);
55.
56.
57.                 Personaje personaje4 = new Personaje();
58.
59.                 personaje4.Name = "Koshi Sugawara";
60.                 personaje4.Location = "Segon col·locador, Segon capità";
61.                 personaje4.Details = "No serà titular, però es un dels millors
     en la seva feina";
62.                 personaje4.numPuntuacion = "5";
63.                 personaje4.Image = "https://vignette.wikia.nocookie.net/haikyu
    u/images/8/8d/Sugawara.png/revision/latest?cb=20150821230947&path-
    prefix=es";
64.
65.                 App.Database.SaveItemAsyncPersonaje(personaje4);
66.
67.
68.                 Personaje personaje5 = new Personaje();
69.
70.                 personaje5.Name = "Asahi Azumane";
71.                 personaje5.Location = "Estrella";
72.                 personaje5.Details = "A qui pots recórrer quan estàs en proble
    mes";
73.                 personaje5.numPuntuacion = "5";
74.                 personaje5.Image = "https://vignette.wikia.nocookie.net/haikyu
    u-pedia/images/c/c4/Asahi_Azumane.jpg/revision/latest?cb=20140721131410";
75.
76.                 App.Database.SaveItemAsyncPersonaje(personaje5);
77.
78.
79.                 Personaje personaje6 = new Personaje();
80.
81.                 personaje6.Name = "Yu Nishinoya";
82.                 personaje6.Location = "Libero";
83.                 personaje6.Details = "El millor en salvar les pilotes impossib
    les";
84.                 personaje6.numPuntuacion = "5";
85.                 personaje6.Image = "https://vignette.wikia.nocookie.net/haikyu
    u/images/8/80/Y%C5%AB_Nishinoya.png/revision/latest?cb=20170809205325&path-
    prefix=es";
86.
87.                 App.Database.SaveItemAsyncPersonaje(personaje6);
88.
89.
```

```
90.                 Personaje personaje7 = new Personaje();
91.
92.                 personaje7.Name = "Ryunosuke Tanaka";
93.                 personaje7.Location = "Wing Spikers";
94.                 personaje7.Details = "Vol ser la estrella i té le poder per ser-ho";
95.                 personaje7.numPuntuacion = "5";
96.                 personaje7.Image = "https://vignette.wikia.nocookie.net/haikyuu/images/e/e7/RyuTanaka.png/revision/latest?cb=20150107144228&path-prefix=es";
97.
98.                 App.Database.SaveItemAsyncPersonaje(personaje7);
99.
100.
101.                     Personaje personaje8 = new Personaje();
102.
103.                     personaje8.Name = "Kei Tsukishima";
104.                     personaje8.Location = "Bloquejador central";
105.                     personaje8.Details = "No creguis que només bloqueja, també pensa";
106.                     personaje8.numPuntuacion = "5";
107.                     personaje8.Image = "https://vignette.wikia.nocookie.net/haikyuu/images/f/f9/Kei_Tsukishima.png/revision/latest?cb=20170822215947&path-prefix=es";
108.
109.                     App.Database.SaveItemAsyncPersonaje(personaje8);
110.
111.
112.                     Personaje personaje9 = new Personaje();
113.
114.                     personaje9.Name = "Tadashi Yamaguchi";
115.                     personaje9.Location = "Pinch Server";
116.                     personaje9.Details = "Encara que pensis que no fa res, ell es qui crea oportunitats";
117.                     personaje9.numPuntuacion = "5";
118.                     personaje9.Image = "https://pbs.twimg.com/media/C2VZ_kGXcAE3oS2.jpg";
119.
120.                     App.Database.SaveItemAsyncPersonaje(personaje9);
121.
122.
123.                     Personaje personaje10 = new Personaje();
124.
125.                     personaje10.Name = "Keishin Ukai";
126.                     personaje10.Location = "Entrenador";
127.                     personaje10.Details = "Antic jugador del Karasuno i el seu avi va entrenar al Petit Gegant";
128.                     personaje10.numPuntuacion = "5";
129.                     personaje10.Image = "https://vignette.wikia.nocookie.net/haikyuu/images/4/44/KeishinUkai.png/revision/latest?cb=20140616134254";
130.
131.                     App.Database.SaveItemAsyncPersonaje(personaje10);
132.
133.                     Personaje personaje11 = new Personaje();
134.
135.                     personaje11.Name = "Ittetsu Takeda";
136.                     personaje11.Location = "Professor";
137.                     personaje11.Details = "Gràcies a ell Karasuno obté la seva força";
138.                     personaje11.numPuntuacion = "5";
139.                     personaje11.Image = "https://vignette.wikia.nocookie.net/haikyuu/images/d/da/Takeda-sensei.png/revision/latest?cb=20140616135813";
140.
141.                     App.Database.SaveItemAsyncPersonaje(personaje11);
142.                 }
143.
```

```
144.
145.
146.                     NavigationPage.SetHasNavigationBar(this, false);
147.
148.                     var sub = new AbsoluteLayout();
149.                     splashImage = new Image
150.                     {
151.                         Source = "splashHaikyuu.jpg",
152.                         Aspect = Aspect.Fill,
153.                         WidthRequest = 380
154.                     };
155.
156.                     AbsoluteLayout.SetLayoutFlags(splashImage,
157.                         AbsoluteLayoutFlags.PositionProportional);
158.                     AbsoluteLayout.SetLayoutBounds(splashImage,
159.                         new Rectangle(0.5, 0.5, AbsoluteLayout.AutoSize, Absolu
    teLayout.AutoSize));
160.
161.                     sub.Children.Add(splashImage);
162.
163.                     this.BackgroundColor = Color.FromHex("#F1913A");
164.                     this.Content = sub;
165.                 }
166.
167.                 protected override async void OnAppearing()
168.                 {
169.                     base.OnAppearing();
170.
171.                     await splashImage.ScaleTo(1, 2000);
172.                     //await splashImage.ScaleTo(0.9, 1500, Easing.Linear);
173.                     //await splashImage.ScaleTo(150, 1200, Easing.Linear);
174.                     Application.Current.MainPage = new NavigationPage(new Custo
    mCellView(null));
175.                 }
176.
177.             }
178.         }
```

1.2.11 App.xaml.cs

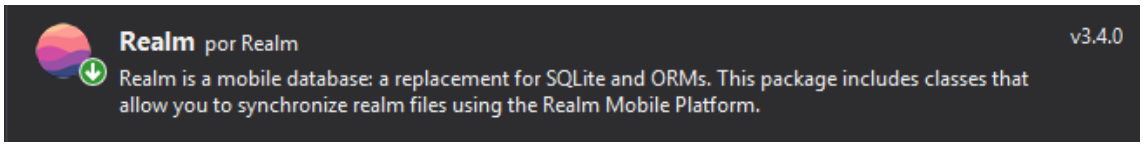
```
1.  using Haikyuu.Views;
2.  using Haikyuu.ViewModels.Base;
3.  using System;
4.  using System.IO;
5.  using Xamarin.Forms;
6.  using Xamarin.Forms.Xaml;
7.
8.  [assembly: XamlCompilation(XamlCompilationOptions.Compile)]
9.  namespace Haikyuu
10. {
11.     public partial class App : Application
12.     {
13.         private static ViewModelLocator _locator;
14.
15.         public static ViewModelLocator Locator
16.         {
17.             get { return _locator = _locator ?? new ViewModelLocator(); }
18.         }
19.
20.         static HaikyuuItemDatabase database;
21.
22.
23.         public App()
24.         {
25.             InitializeComponent();
26.
27.             Resources = new ResourceDictionary();
28.             Resources.Add("primaryGreen", Color.FromHex("91CA47"));
29.             Resources.Add("primaryDarkGreen", Color.FromHex("6FA22E"));
30.
31.             var nav = new NavigationPage(new SplashPage());
32.             nav.BarBackgroundColor = (Color)App.Current.Resources["primaryGree
    n"];
33.             nav.BarTextColor = Color.White;
34.
35.             MainPage = nav;
36.
37.             //MainPage = new NavigationPage(new SplashPage());
38.         }
39.
40.         public static HaikyuuItemDatabase Database
41.         {
42.             get
43.             {
44.                 if (database == null)
45.                 {
46.                     database = new HaikyuuItemDatabase(Path.Combine(Environmen
    t.GetFolderPath(Environment.SpecialFolder.LocalApplicationData), "HaikyuuSQLit
    e.db3"));
47.                 }
48.                 return database;
49.             }
50.         }
51.
52.         public int ResumeAtTodoId { get; set; }
53.
54.         protected override void OnStart()
55.         {
56.             // Handle when your app starts
57.         }
58.
59.         protected override void OnSleep()
60.         {
```

```
61.             // Handle when your app sleeps
62.         }
63.
64.         protected override void OnResume()
65.         {
66.             // Handle when your app resumes
67.         }
68.     }
69. }
```

3. Realm

Ahora en **Realm** vamos a cambiar el fichero de la carpeta **Data**, nos bajaremos la nueva **referencia** y cambiaremos el **nombre** de las **funciones**.

Para agregar el **paquete de Realm** vamos al mismo sitio de antes que en el SQLite y buscamos **"Realm"**:



Pero antes de esto, tenemos dos opciones, quitar las referencias del SQLite y añadir estas, o agregar las nuevas. Aunque también podemos recuperar la aplicación del tutorial anterior y seguir con Realm.

Aquí dejo el código del fichero en cuestión:

```
1.  using System.Collections.Generic;
2.  using System.Threading.Tasks;
3.  using Realms;
4.  using Haikyuu.Models;
5.  using System;
6.  using System.Collections.ObjectModel;
7.  using System.Linq;
8.
9.  namespace Haikyuu
10. {
11.     public class HaikyuuItemDatabase
12.     {
13.         private Realm _realm;
14.
15.         public HaikyuuItemDatabase()
16.         {
17.             try
18.             {
19.                 _realm = Realm.GetInstance();
20.                 AddDataDB();
21.             }
22.             catch (Exception ex)
23.             {
24.
```

```
            }
        }

        private void AddDataDB()
        {
            if (GetAll().Count == 0)
            {
                var chars = new ObservableCollection<Personaje>
                {
                    new Personaje
                    {
                        Name = "Shoyo Hinata",
                        Location = "Bloquejador central, Carnada",
                        Details = "Vol arribar a ser com el Petit Gegant",
                        numPuntuacion = "5",
                        Image = "https://vignette.wikia.nocookie.net/haikyuu/images/c/c2/Rostro_de_Hinata.png/revision/latest?cb=20160210050706&path-prefix=es"
                    },
                    new Personaje
                    {
                        Name = "Tobio Kageyama",
                        Location = "Col·locador",
                        Details = "Juntament amb Hinata, un parella imparable",
                        numPuntuacion = "5",
                        Image = "https://vignette.wikia.nocookie.net/haikyuu-pedia/images/4/47/Tobio_Kageyama.jpg/revision/latest?cb=20140721131354"
                    },
                    new Personaje
                    {
                        Name = "Daichi Sawamura",
                        Location = "Wing Spikers, Capità",
                        Details = "Especialista en la defensa i una persona necessària per al equip",
                        numPuntuacion = "5",
                        Image = "https://em.wattpad.com/8bd5055f8ed26c2660a747809f9d943e64c8cfe4/687474703a2f2f36382e6d656469612e74756d626c722e636f6d2f6637636266383238323361303463353334373666561376530336262393933302f74756d626c725f696e6c696e655f6e683373646f464d714a3172696e3631382e6a7067?s=fit&h=758&w=1583&q=80"
                    },
                    new Personaje
                    {
                        Name = "Koshi Sugawara",
                        Location = "Segon col·locador, Segon capità",
                        Details = "No serà titular, però es un dels millors en la seva feina",
                        numPuntuacion = "5",
                        Image = "https://vignette.wikia.nocookie.net/haikyuu/images/8/8d/Sugawara.png/revision/latest?cb=20150821230947&path-prefix=es"
                    },
                    new Personaje
                    {
                        Name = "Asahi Azumane",
                        Location = "Estrella",
                        Details = "A qui pots recórrer quan estàs en problemes",
                        numPuntuacion = "5",
                        Image = "https://vignette.wikia.nocookie.net/haikyuu-pedia/images/c/c4/Asahi_Azumane.jpg/revision/latest?cb=20140721131410"
                    },
                    new Personaje
                    {
                        Name = "Yu Nishinoya",
                        Location = "Libero",
```

```
78.                         Details = "El millor en salvar les pilotes impossibles
    ",
79.                         numPuntuacion = "5",
80.                         Image = "https://vignette.wikia.nocookie.net/haikyuu/i
    mages/8/80/Y%C5%AB_Nishinoya.png/revision/latest?cb=20170809205325&path-
    prefix=es"
81.                     },
82.                     new Personaje
83.                     {
84.                         Name = "Ryunosuke Tanaka",
85.                         Location = "Wing Spikers",
86.                         Details = "Vol ser la estrella i té le poder per ser-
    ho",
87.                         numPuntuacion = "5",
88.                         Image = "https://vignette.wikia.nocookie.net/haikyuu/i
    mages/e/e7/RyuTanaka.png/revision/latest?cb=20150107144228&path-prefix=es"
89.                     },
90.                     new Personaje
91.                     {
92.                         Name = "Kei Tsukishima",
93.                         Location = "Bloquejador central",
94.                         Details = "No creguis que només bloqueja, també pensa"
    ,
95.                         numPuntuacion = "5",
96.                         Image = "https://vignette.wikia.nocookie.net/haikyuu/i
    mages/f/f9/Kei_Tsukishima.png/revision/latest?cb=20170822215947&path-
    prefix=es"
97.                     },
98.                     new Personaje
99.                     {
100.                            Name = "Tadashi Yamaguchi",
101.                            Location = "Pinch Server",
102.                            Details = "Encara que pensis que no fa res, ell
     es qui crea oportunitats",
103.                            numPuntuacion = "5",
104.                            Image = "https://pbs.twimg.com/media/C2VZ_kGXcA
    E3oS2.jpg"
105.                        },
106.                        new Personaje
107.                        {
108.                            Name = "Keishin Ukai",
109.                            Location = "Entrenador",
110.                            Details = "Antic jugador del Karasuno i el seu
    avi va entrenar al Petit Gegant",
111.                            numPuntuacion = "5",
112.                            Image = "https://vignette.wikia.nocookie.net/ha
    ikyuu/images/4/44/KeishinUkai.png/revision/latest?cb=20140616134254"
113.                        },
114.                        new Personaje
115.                        {
116.                            Name = "Ittetsu Takeda",
117.                            Location = "Professor",
118.                            Details = "Gràcies a ell Karasuno obté la seva
    força",
119.                            numPuntuacion = "5",
120.                            Image = "https://vignette.wikia.nocookie.net/ha
    ikyuu/images/d/da/Takeda-sensei.png/revision/latest?cb=20140616135813"
121.                        },
122.                    };
123.
124.                    foreach (var item in chars)
125.                    {
126.                        SaveItem(item);
127.                    }
128.                }
129.            }
```

```
130.
131.                 public List<Personaje> GetAll()
132.                 {
133.                     return new List<Personaje>(_realm.All<Personaje>());
134.                 }
135.
136.                 public void SaveItem(Personaje item)
137.                 {
138.                     item.id = _realm.All<Personaje>().Count() + 1;
139.
140.                     _realm.Write(() => _realm.Add(item));
141.                 }
142.
143.                 public void UpdateItem(Personaje item, int rating)
144.                 {
145.                     _realm.Write(() =>
146.                     {
147.                         item.numPuntuacion = rating.ToString();
148.                         _realm.Add(item, true);
149.                     });
150.                 }
151.             }
152.         }
```

Ahora, con estas dos aplicaciones, ya tenemos una mínima idea de cómo funciona el SQLite y el Realm.

A partir de aquí, cada uno es libre de usar este código para futuros proyectos y una guía para empezar con Xamarin usando una Base de Datos.