ICB0-M08U02IA07

DADES – SQLITE + REALM

Pol benito

2019

Índice

[1. Introducción 2](#_Toc9000702)

[2. SQLite 4](#_Toc9000703)

[2.1. HaikyuuItemDatabase.cs 6](#_Toc9000704)

[2.2. Models 7](#_Toc9000705)

[2.2.1 Comentario.cs 7](#_Toc9000709)

[2.2.2 Personaje.cs 7](#_Toc9000710)

[2.3. Services 8](#_Toc9000711)

[2.2.3 Navigation 8](#_Toc9000712)

[2.4. ViewModels 9](#_Toc9000713)

[2.2.4 Base 9](#_Toc9000714)

[2.2.5 CustomCellViewModel.cs 15](#_Toc9000715)

[2.2.6 MainViewModel.cs 17](#_Toc9000716)

[2.5. Views 18](#_Toc9000717)

[2.2.7 CustomCellView.xaml 18](#_Toc9000718)

[2.2.8 HaikyuuItemPage.xaml 19](#_Toc9000719)

[2.2.9 MainView.xaml 21](#_Toc9000720)

[2.2.10 SplashPage.xaml.cs 26](#_Toc9000721)

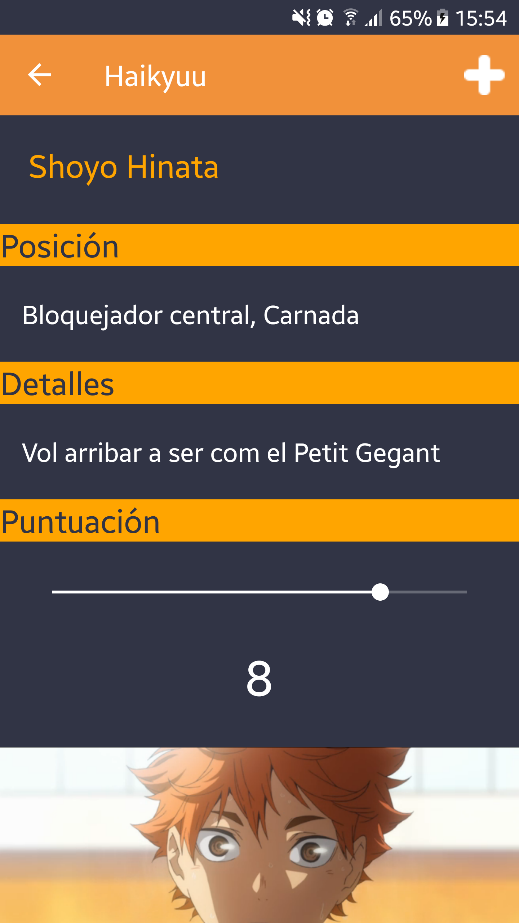
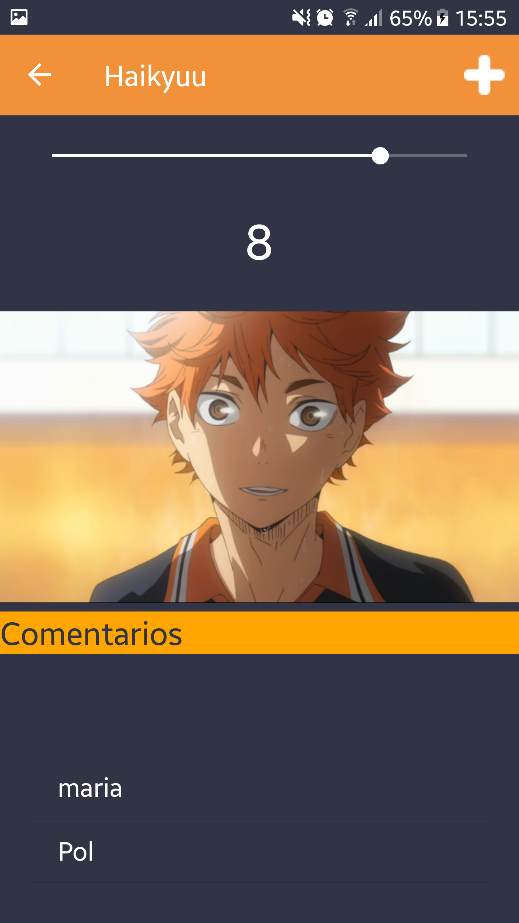
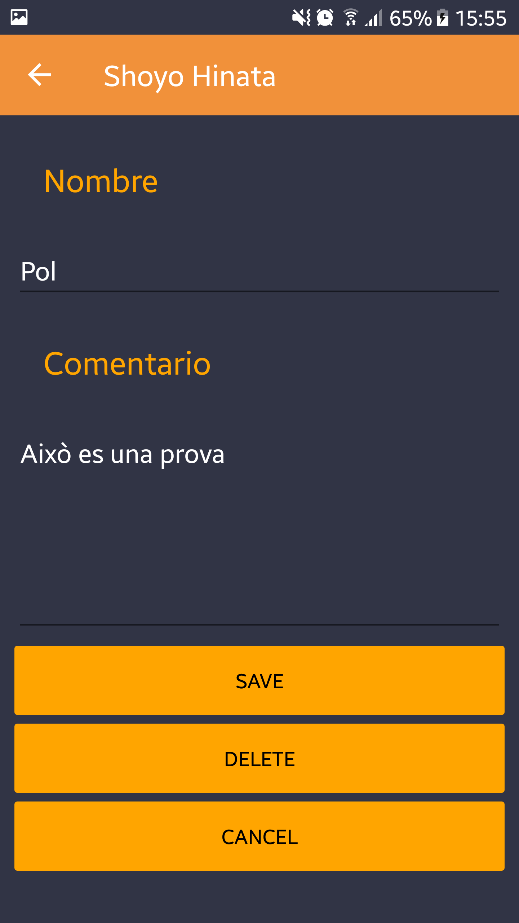
[2.2.11 App.xaml.cs 30](#_Toc9000722)

[3. Realm 31](#_Toc9000723)

# 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.

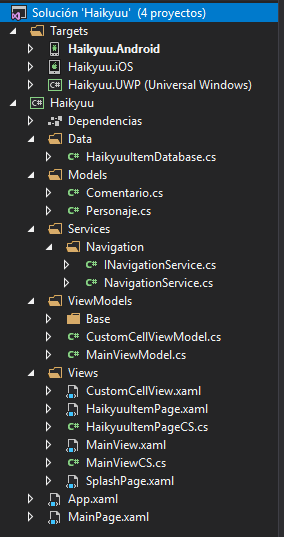
Después seguiremos con **Realm**, algo más difícil de comprender porque no estaremos acostumbrados, pero muy fácil de hacer porque ya habremos hecho la anterior aplicación. Esta **Base de Datos** **no es** **relacional**, es decir, no va por tablas. Por eso es un poco confuso al principio cuando intentas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

# SQLite

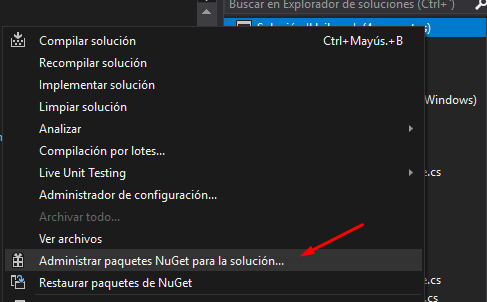
Como he dicho anteriormente, empezaremos a hacer la aplicación **SQLite** ya que será mucho más fácil para empezar.

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**.

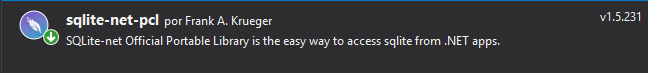
Aquí dejo un esquema de los ficheros que crearemos y utilizaremos:



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:



## 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;
5. **using** Haikyuu.Models;
7. **namespace** Haikyuu
8. {
9. **public** **class** HaikyuuItemDatabase
10. {
11. **readonly** SQLiteAsyncConnection database;
13. **public** HaikyuuItemDatabase(**string** dbPath)
14. {
15. database = **new** SQLiteAsyncConnection(dbPath);
16. database.CreateTableAsync<Comentario>().Wait();
17. database.CreateTableAsync<Personaje>().Wait();
18. }
20. **public** Task<List<Comentario>> GetItemsAsync()
21. {
22. Task<List<Comentario>> firstList;
23. firstList = database.Table<Comentario>().ToListAsync();
25. **return** firstList;
26. }
28. **public** Task<List<Personaje>> GetItemsAsyncPersonaje()
29. {
30. Task<List<Personaje>> firstList;
31. firstList = database.Table<Personaje>().ToListAsync();
33. **return** firstList;
34. }
36. **public** Task<Comentario> GetItemAsync(**int** id)
37. {
38. **return** database.Table<Comentario>().Where(i => i.id == id).FirstOrDefaultAsync();
39. }
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. }
53. **public** Task<**int**> SaveItemAsyncPersonaje(Personaje item)
54. {
55. **if** (item.id != 0)
56. {
57. **return** database.UpdateAsync(item);
58. }
59. **else**
60. {
61. **return** database.InsertAsync(item);
62. }
63. }
65. **public** Task<**int**> DeleteItemAsync(Comentario item)
66. {
67. **return** database.DeleteAsync(item);
68. }
69. }
70. }

## Models

Definimos como queremos que tenga cada modelo.



### Comentario.cs

1. **using** SQLite;
3. **namespace** Haikyuu.Models
4. {
5. **public** **class** Comentario
6. {
7. [PrimaryKey, AutoIncrement]
8. **public** **int** id { **get**; **set**; }
9. **public** **string** Name { **get**; **set**; }
10. **public** **string** NameCharacter { **get**; **set**; }
11. **public** **string** Notes { **get**; **set**; }
12. }
13. }

### Personaje.cs

1. **using** SQLite;
3. **namespace** Haikyuu.Models
4. {
5. **public** **class** Personaje
6. {
7. [PrimaryKey, AutoIncrement]
8. **public** **int** id { **get**; **set**; }
9. **public** **string** Name { **get**; **set**; }
10. **public** **string** Location { **get**; **set**; }
11. **public** **string** Details { **get**; **set**; }
12. **public** **string** numPuntuacion { **get**; **set**; }
13. **public** **string** Image { **get**; **set**; }
14. }
15. }

## Services

### Navigation

#### INavigationService.cs

1. **using** System;
3. **namespace** Haikyuu.Services.Navigation
4. {
5. **public** **interface** INavigationService
6. {
7. **void** NavigateTo<TDestinationViewModel>(**object** navigationContext = **null**);
9. **void** NavigateTo(Type destinationType, **object** navigationContext = **null**);
11. **void** NavigateBack();
12. }
13. }

#### NavigationService.cs

1. **namespace** Haikyuu.Services.Navigation
2. {
3. **using** System;
4. **using** System.Collections.Generic;
5. **using** ViewModels;
6. **using** Views;
7. **using** Xamarin.Forms;
9. **public** **class** NavigationService : INavigationService
10. {
11. **private** IDictionary<Type, Type> viewModelRouting = **new** Dictionary<Type, Type>()
12. {
13. { **typeof**(MainViewModel),  **typeof**(MainView) },
14. { **typeof**(CustomCellViewModel), **typeof**(CustomCellView) }
15. };
17. **public** **void** NavigateTo<TDestinationViewModel>(**object** navigationContext = **null**)
18. {
19. Type pageType = viewModelRouting[**typeof**(TDestinationViewModel)];
20. var page = Activator.CreateInstance(pageType, navigationContext) **as** Page;
22. **if** (page != **null**)
23. Application.Current.MainPage.Navigation.PushAsync(page);
24. }
26. **public** **void** NavigateTo(Type destinationType, **object** navigationContext = **null**)
27. {
28. Type pageType = viewModelRouting[destinationType];
29. var page = Activator.CreateInstance(pageType, navigationContext) **as** Page;
31. **if** (page != **null**)
32. Application.Current.MainPage.Navigation.PushAsync(page);
33. }
35. **public** **void** NavigateBack()
36. {
37. Application.Current.MainPage.Navigation.PopAsync();
38. }
39. }
40. }

## ViewModels

### Base

#### DelegateCommand.cs

1. **using** System;
2. **using** System.Windows.Input;
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;
14. /// <summary>
15. /// Constructor not using canExecute.
16. /// </summary>
17. /// <param name="execute"></param>
18. **public** DelegateCommand(Action execute) : **this**(execute, **null**) { }
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. }
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();
41. **return** **true**;
42. }
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. }
53. /// <summary>
54. /// This method allow us to force the execution of CanExecute method to reevaluate execution.
55. /// </summary>
56. **public** **void** RaiseCanExecuteChanged()
57. {
58. var tmpHandle = CanExecuteChanged;
59. **if** (tmpHandle != **null**)
60. tmpHandle(**this**, **new** EventArgs());
61. }
63. /// <summary>
64. /// This event notify XAML controls using the command to reevaluate the CanExecute of it.
65. /// </summary>
66. **public** **event** EventHandler CanExecuteChanged;
67. }
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;
77. /// <summary>
78. /// Constructor not using canExecute.
79. /// </summary>
80. /// <param name="execute"></param>
81. **public** DelegateCommand(Action<T> execute) : **this**(execute, **null**) { }
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. }
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);
104. **return** **true**;
105. }
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. }
116. /// <summary>
117. /// This method allow us to force the execution of CanExecute method to reevaluate execution.
118. /// </summary>
119. **public** **void** RaiseCanExecuteChanged()
120. {
121. var tmpHandle = CanExecuteChanged;
122. **if** (tmpHandle != **null**)
123. tmpHandle(**this**, **new** EventArgs());
124. }
126. /// <summary>
127. /// This event notify XAML controls using the command to reevaluate the CanExecute of it.
128. /// </summary>
129. **public** **event** EventHandler CanExecuteChanged;
130. }
131. }

#### DelegateCommandAsync.cs

1. **using** System;
2. **using** System.Threading.Tasks;
3. **using** System.Windows.Input;
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;
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. }
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**>> canExecute)
30. {
31. \_execute = execute;
32. \_canExecute = canExecute;
33. }
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;
45. **return** **true**;
46. }
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. }
57. /// <summary>
58. ///     This event notify XAML controls using the command to reevaluate the CanExecute of it.
59. /// </summary>
60. **public** **event** EventHandler CanExecuteChanged;
62. /// <summary>
63. ///     This method allow us to force the execution of CanExecute method 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. }
73. /// <summary>
74. ///     This class allows us to delegate command execution to viewmodels using 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;
81. /// <summary>
82. ///     Constructor not using canExecute.
83. /// </summary>
84. /// <param name="execute"></param>
85. **public** DelegateCommandAsync(Func<T, Task> execute) : **this**(execute, **null**)
86. {
87. }
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. }
100. /// <summary>
101. ///     This method is called from XAML to evaluate if the command 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;
110. **return** **true**;
111. }
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. }
122. /// <summary>
123. ///     This event notify XAML controls using the command to reevaluate the CanExecute of it.
124. /// </summary>
125. **public** **event** EventHandler CanExecuteChanged;
127. /// <summary>
128. ///     This method allow us to force the execution of CanExecute 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. }

#### ViewModelBase.cs

1. **using** System.ComponentModel;
2. **using** System.Runtime.CompilerServices;
4. **namespace** Haikyuu.ViewModels.Base
5. {
6. **public** **abstract** **class** ViewModelBase : INotifyPropertyChanged
7. {
8. **public** **virtual** **void** OnAppearing(**object** navigationContext)
9. {
10. }
12. **public** **virtual** **void** OnDisappearing()
13. {
14. }
16. **public** **event** PropertyChangedEventHandler PropertyChanged;
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. }

#### ViewModelLocator.cs

1. **using** Unity;
2. **using** Haikyuu.Services.Navigation;
4. **namespace** Haikyuu.ViewModels.Base
5. {
6. **public** **class** ViewModelLocator
7. {
8. **readonly** IUnityContainer \_container;
10. **public** ViewModelLocator()
11. {
12. \_container = **new** UnityContainer();
14. // ViewModels
15. \_container.RegisterType<CustomCellViewModel>();
16. \_container.RegisterType<MainViewModel>();
18. // Services
19. \_container.RegisterType<INavigationService, NavigationService>();
20. }
22. **public** CustomCellViewModel CustomCellViewModel
23. {
24. **get** { **return** \_container.Resolve<CustomCellViewModel>(); }
25. }
27. **public** MainViewModel MainViewModel
28. {
29. **get** { **return** \_container.Resolve<MainViewModel>(); }
30. }
31. }
32. }

### CustomCellViewModel.cs

1. **using** Haikyuu.Models;
2. **using** Haikyuu.ViewModels.Base;
3. **using** System.Collections.ObjectModel;
5. **namespace** Haikyuu.ViewModels
6. {
7. **public** **class** CustomCellViewModel : ViewModelBase
8. {
9. **public** ObservableCollection<Personaje> Personajes { **get**; **set**; }
11. **public** CustomCellViewModel()
12. {
13. Personajes = **new** ObservableCollection<Personaje>
14. {
15. **new** Personaje
16. {
17. Name = "Shoyo Hinata",
18. Location = "Bloquejador central, Carnada",
19. Details =
20. "Vol arribar a ser com el Petit Gegant",
21. Image =
22. "https://vignette.wikia.nocookie.net/haikyuu/images/c/c2/Rostro\_de\_Hinata.png/revision/latest?cb=20160210050706&path-prefix=es"
23. },
24. **new** Personaje
25. {
26. Name = "Tobio Kageyama",
27. Location = "Col·locador",
28. Details =
29. "Juntament amb Hinata, un parella imparable",
30. Image =
31. "https://vignette.wikia.nocookie.net/haikyuu-pedia/images/4/47/Tobio\_Kageyama.jpg/revision/latest?cb=20140721131354"
32. },
33. **new** Personaje
34. {
35. Name = "Daichi Sawamura",
36. Location = "Wing Spikers, Capità",
37. Details =
38. "Especialista en la defensa i una persona necessària per al equip",
39. Image =
40. "https://em.wattpad.com/8bd5055f8ed26c2660a747809f9d943e64c8cfe4/687474703a2f2f36382e6d656469612e74756d626c722e636f6d2f66376362663832383233613034633533343733666561376530336262393933302f74756d626c725f696e6c696e655f6e683373646f464d714a3172696e3631382e6a7067?s=fit&h=758&w=1583&q=80"
41. },
42. **new** Personaje
43. {
44. Name = "Koshi Sugawara",
45. Location = "Segon col·locador, Segon capità",
46. Details =
47. "No serà titular, però es un dels millors en la seva feina",
48. Image =
49. "https://vignette.wikia.nocookie.net/haikyuu/images/8/8d/Sugawara.png/revision/latest?cb=20150821230947&path-prefix=es"
50. },
51. **new** Personaje
52. {
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 oportunitats",
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 entrenar al Petit Gegant",
102. Image =
103. "https://vignette.wikia.nocookie.net/haikyuu/images/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 =
112. "https://vignette.wikia.nocookie.net/haikyuu/images/d/da/Takeda-sensei.png/revision/latest?cb=20140616135813"
113. }
114. };
115. }
116. }
117. }

### MainViewModel.cs

1. **using** Haikyuu.Services.Navigation;
2. **using** Haikyuu.ViewModels.Base;
3. **using** System.Windows.Input;
5. **namespace** Haikyuu.ViewModels
6. {
7. **public** **class** MainViewModel : ViewModelBase
8. {
9. **private** ICommand \_customCellCommand;
11. **private** INavigationService \_navigationService;
13. **public** MainViewModel(INavigationService navigationService)
14. {
15. \_navigationService = navigationService;
16. }
18. **public** ICommand CustomCellCommand
19. {
20. **get** { **return** \_customCellCommand = \_customCellCommand ?? **new** DelegateCommand(CustomCellCommandExecute); }
21. }
22. **private** **void** CustomCellCommandExecute()
23. {
24. \_navigationService.NavigateTo<CustomCellViewModel>();
25. }
26. }
27. }

## Views

### CustomCellView.xaml

1. <?xml version="1.0" encoding="utf-8" ?>
2. <ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
3. xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
4. x:Class="Haikyuu.Views.CustomCellView"
5. Title="Haikyuu">
6. <ListView
7. ItemsSource="{Binding Personajes}"
8. RowHeight="200"
9. x:Name="ListaPersonajes">
10. <ListView.ItemTemplate>
11. <DataTemplate>
12. <ViewCell>
13. <StackLayout
14. Orientation="Horizontal">
15. <Image
16. Source="{Binding Image}"
17. Aspect="AspectFill"
18. WidthRequest="380"/>
19. </StackLayout>
20. </ViewCell>
21. </DataTemplate>
22. </ListView.ItemTemplate>
23. </ListView>
24. </ContentPage>

#### CustomCellView.xaml.cs

1. **using** System;
2. **using** Haikyuu.Models;
3. **using** Haikyuu.ViewModels;
4. **using** Xamarin.Forms;
6. **namespace** Haikyuu.Views
7. {
8. **public** partial **class** CustomCellView : ContentPage
9. {
10. **private** **object** Parameter { **get**; **set**; }
12. **public** CustomCellView(**object** parameter)
13. {
14. InitializeComponent();
16. ListaPersonajes.ItemSelected += ListaPersonajes\_ItemSelected;
17. BindingContext = App.Locator.CustomCellViewModel;
19. Parameter = parameter;
20. }
22. **private** **void** ListaPersonajes\_ItemSelected(**object** sender, SelectedItemChangedEventArgs e)
23. {
24. Navigation.PushAsync(**new** MainView((Personaje)e.SelectedItem));
25. }
27. **protected** **override** **void** OnAppearing()
28. {
29. var viewModel = BindingContext **as** CustomCellViewModel;
30. **if** (viewModel != **null**) viewModel.OnAppearing(Parameter);
31. }
33. **protected** **override** **void** OnDisappearing()
34. {
35. var viewModel = BindingContext **as** CustomCellViewModel;
36. **if** (viewModel != **null**) viewModel.OnDisappearing();
37. }
38. }
39. }

### HaikyuuItemPage.xaml

1. <?xml version="1.0" encoding="utf-8" ?>
2. <ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
3. xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
4. x:Class="Haikyuu.Views.HaikyuuItemPage"
5. BackgroundColor="#313445">
6. <StackLayout Margin="10" VerticalOptions="StartAndExpand">
7. <Label Text="Nombre" FontSize="Large" TextColor="Orange" Margin="20"/>
8. <Entry Text="{Binding Name}" TextColor="White"/>
9. <Label Text="Comentario" FontSize="Large" TextColor="Orange" Margin="20"/>
10. <Entry Text="{Binding Notes}" TextColor="White" HeightRequest="150"/>
11. <Button Text="Save" Clicked="OnSaveClicked" BackgroundColor="Orange" TextColor="Black"/>
12. <Button Text="Delete" Clicked="OnDeleteClicked" BackgroundColor="Orange" TextColor="Black"/>
13. <Button Text="Cancel" Clicked="OnCancelClicked" BackgroundColor="Orange" TextColor="Black"/>
14. </StackLayout>
15. </ContentPage>

#### HaikyuuItemPage.xaml.cs

1. **using** System;
2. **using** Xamarin.Forms;
3. **using** Xamarin.Forms.Xaml;
5. **using** Haikyuu.Models;
7. **namespace** Haikyuu.Views
8. {
9. [XamlCompilation(XamlCompilationOptions.Compile)]
10. **public** partial **class** HaikyuuItemPage : ContentPage
11. {
12. **string** nombrePersonaje;
14. **public** HaikyuuItemPage (**string** \_nombrePersonaje)
15. {
16. InitializeComponent ();
18. nombrePersonaje = \_nombrePersonaje;
20. Title = nombrePersonaje;
21. }
23. async **void** OnSaveClicked(**object** sender, EventArgs e)
24. {
25. var todoItem = (Comentario)BindingContext;
27. todoItem.NameCharacter = nombrePersonaje;
29. await App.Database.SaveItemAsync(todoItem);
30. await Navigation.PopAsync();
31. }
33. async **void** OnDeleteClicked(**object** sender, EventArgs e)
34. {
35. var todoItem = (Comentario)BindingContext;
37. todoItem.NameCharacter = nombrePersonaje;
39. await App.Database.DeleteItemAsync(todoItem);
40. await Navigation.PopAsync();
41. }
43. async **void** OnCancelClicked(**object** sender, EventArgs e)
44. {
45. await Navigation.PopAsync();
46. }
47. }
48. }

#### HaikyuuItemPageCS.cs

1. **using** Xamarin.Forms;
3. **using** Haikyuu.Models;
5. **namespace** Haikyuu.Views
6. {
7. **public** **class** HaikyuuItemPageCS : ContentPage
8. {
9. **public** HaikyuuItemPageCS()
10. {
11. Title = "Haikyuu Item";
13. var nameEntry = **new** Entry();
14. nameEntry.SetBinding(Entry.TextProperty, "Name");
16. var notesEntry = **new** Entry();
17. notesEntry.SetBinding(Entry.TextProperty, "Notes");
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. };
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. };
35. var cancelButton = **new** Button { Text = "Cancel" };
36. cancelButton.Clicked += async (sender, e) =>
37. {
38. await Navigation.PopAsync();
39. };
41. Content = **new** StackLayout
42. {
43. Margin = **new** Thickness(20),
44. VerticalOptions = LayoutOptions.StartAndExpand,
45. Children =
46. {
47. **new** Label { Text = "Name" },
48. nameEntry,
49. **new** Label { Text = "Notes" },
50. notesEntry,
51. saveButton,
52. deleteButton,
53. cancelButton
54. }
55. };
56. }
57. }
58. }

### MainView.xaml

1. <?xml version="1.0" encoding="utf-8" ?>
2. <ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
3. xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
4. x:Class="Haikyuu.Views.MainView"
5. Title="Haikyuu">
6. <ContentPage.ToolbarItems>
7. <ToolbarItem Text="+" Clicked="OnItemAdded">
8. <ToolbarItem.Icon>
9. <OnPlatform x:TypeArguments="FileImageSource">
10. <On Platform="Android, UWP" Value="plus.png" />
11. </OnPlatform>
12. </ToolbarItem.Icon>
13. </ToolbarItem>
14. </ContentPage.ToolbarItems>
15. <ContentPage.Content>
16. <ScrollView>
17. <StackLayout
18. BackgroundColor="#313445">
19. <Label x:Name="Name"
20. FontSize="Large"
21. Text="Name"
22. TextColor="Orange"
23. Margin="20"
24. />
26. <Label Text="Posición"
27. FontSize="Large"
28. BackgroundColor="Orange"
29. TextColor="#313445"
30. />
32. <Label x:Name="Location"
33. FontSize="Medium"
34. Text="Location"
35. TextColor="White"
36. Margin="15"
37. />
39. <Label Text="Detalles"
40. FontSize="Large"
41. BackgroundColor="Orange"
42. TextColor="#313445"
43. />
45. <Label x:Name="Details"
46. Text="Details"
47. FontSize="Medium"
48. TextColor="White"
49. Margin="15"
50. />
52. <!-- Puntuacion -->
53. <Label Text="Puntuación"
54. FontSize="Large"
55. BackgroundColor="Orange"
56. TextColor="#313445"
57. />
59. <Slider x:Name="sliderPuntuacion"
60. Margin="20"
61. Maximum="10"
62. Minimum="0"
63. ThumbColor="White"
64. />
66. <Label x:Name="numPuntuacion"
67. FontSize="35"
68. TextColor="White"
69. Margin="170, 0, 0, 20"
70. />
72. <Image x:Name="Foto"
73. Aspect="AspectFill"
74. WidthRequest="380"
75. MinimumHeightRequest="500"
76. />
78. <!--Comentarios-->
79. <Label Text="Comentarios"
80. FontSize="Large"
81. BackgroundColor="Orange"
82. TextColor="#313445"
83. />
84. <ListView x:Name="listView" Margin="20" ItemSelected="OnListItemSelected">
85. <ListView.ItemTemplate>
86. <DataTemplate>
87. <ViewCell>
88. <StackLayout Margin="20,0,0,0" Orientation="Horizontal" HorizontalOptions="FillAndExpand">
89. <Label Text="{Binding Name}"
90. VerticalTextAlignment="Center"
91. HorizontalOptions="StartAndExpand"
92. FontSize="Medium"
93. TextColor="White"
94. />
95. </StackLayout>
96. </ViewCell>
97. </DataTemplate>
98. </ListView.ItemTemplate>
99. </ListView>
100. </StackLayout>
101. </ScrollView>
102. </ContentPage.Content>
103. </ContentPage>

#### 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;
9. **namespace** Haikyuu.Views
10. {
11. [XamlCompilation(XamlCompilationOptions.Compile)]
12. **public** partial **class** MainView : ContentPage
13. {
14. **public** MainView(Personaje parameter)
15. {
16. InitializeComponent();
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;
24. obtenirPuntuacio();
26. sliderPuntuacion.ValueChanged += SliderPuntuacion\_ValueChanged;
28. BindingContext = App.Locator.MainViewModel;
29. }
31. **private** **void** obtenirPuntuacio()
32. {
33. List<Personaje> personaje;
35. personaje = App.Database.GetItemsAsyncPersonaje().Result;
36. personaje = personaje.Where(r => r.Name == Name.Text).ToList();
38. Personaje psj = personaje.FirstOrDefault();
40. numPuntuacion.Text = psj.numPuntuacion.ToString();
41. sliderPuntuacion.Value = Double.Parse(psj.numPuntuacion);
42. }
44. **private** **void** SliderPuntuacion\_ValueChanged(**object** sender, ValueChangedEventArgs e)
45. {
46. numPuntuacion.Text = (Convert.ToInt32(sliderPuntuacion.Value)).ToString();
48. List<Personaje> listaPersonaje;
50. listaPersonaje = App.Database.GetItemsAsyncPersonaje().Result;
52. Personaje personaje = listaPersonaje.Where(a => a.Name == Name.Text).FirstOrDefault();
54. personaje.numPuntuacion = numPuntuacion.Text.ToString();
56. App.Database.SaveItemAsyncPersonaje(personaje);
57. }
59. **protected** **override** async **void** OnAppearing()
60. {
61. **base**.OnAppearing();
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();
67. List<Comentario> listaComentarios;
69. listaComentarios = App.Database.GetItemsAsync().Result;
71. listaComentarios = listaComentarios.Where( a => a.NameCharacter == Name.Text ).ToList();
73. listView.ItemsSource = listaComentarios;

76. List<Personaje> listaPersonaje;
78. listaPersonaje = App.Database.GetItemsAsyncPersonaje().Result;
80. Personaje personaje = listaPersonaje.Where(a => a.Name == Name.Text).FirstOrDefault();
82. numPuntuacion.Text = personaje.numPuntuacion.ToString();
83. }
85. async **void** OnItemAdded(**object** sender, EventArgs e)
86. {
87. await Navigation.PushAsync(**new** HaikyuuItemPage(Name.Text)
88. {
89. BindingContext = **new** Comentario()
90. });
91. }
93. async **void** OnListItemSelected(**object** sender, SelectedItemChangedEventArgs e)
94. {
95. //((App)App.Current).ResumeAtTodoId = (e.SelectedItem as TodoItem).ID;
96. //Debug.WriteLine("setting ResumeAtTodoId = " + (e.SelectedItem as TodoItem).ID
98. **if** (e.SelectedItem != **null**)
99. {
100. await Navigation.PushAsync(**new** HaikyuuItemPage(Name.Text)
101. {
102. BindingContext = e.SelectedItem **as** Comentario
103. });
104. }
105. }
106. }
107. }

#### MainViewCS.cs

1. **using** Haikyuu.Models;
2. **using** System.Collections.Generic;
3. **using** System.Linq;
4. **using** Xamarin.Forms;
6. **namespace** Haikyuu.Views
7. {
8. **public** **class** MainViewCS : ContentPage
9. {
10. ListView listView;
12. **public** MainViewCS()
13. {
14. Title = "Haikyuu";
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);
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");
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. };
50. **return** **new** ViewCell { View = stackLayout };
51. })
52. };
53. listView.ItemSelected += async (sender, e) =>
54. {
55. //((App)App.Current).ResumeAtTodoId = (e.SelectedItem as TodoItem).ID;
56. //Debug.WriteLine("setting ResumeAtTodoId = " + (e.SelectedItem as TodoItem).ID);
58. **if** (e.SelectedItem != **null**)
59. {
60. await Navigation.PushAsync(**new** HaikyuuItemPageCS
61. {
62. BindingContext = e.SelectedItem **as** Comentario
63. });
64. }
65. };
67. Content = listView;
68. }
70. **protected** **override** async **void** OnAppearing()
71. {
72. **base**.OnAppearing();
74. // Reset the 'resume' id, since we just want to re-start here
75. ((App)App.Current).ResumeAtTodoId = -1;
76. listView.ItemsSource = await App.Database.GetItemsAsync();
77. }
78. }
79. }

### SplashPage.xaml.cs

1. **using** Haikyuu.Models;
2. **using** System.Collections.Generic;
4. **using** Xamarin.Forms;
5. **using** Xamarin.Forms.Xaml;
7. **namespace** Haikyuu.Views
8. {
9. [XamlCompilation(XamlCompilationOptions.Compile)]
10. **public** partial **class** SplashPage : ContentPage
11. {
12. Image splashImage;
13. **public** SplashPage ()
14. {
15. InitializeComponent();

18. List<Personaje> listaPersonaje;
20. listaPersonaje = App.Database.GetItemsAsyncPersonaje().Result;
22. **if** (listaPersonaje.Count == 0)
23. {
24. Personaje personaje1 = **new** Personaje();
26. personaje1.Name = "Shoyo Hinata";
27. personaje1.Location = "Bloquejador central, Carnada";
28. personaje1.Details = "Vol arribar a ser com el Petit Gegant";
29. personaje1.numPuntuacion = "5";
30. personaje1.Image = "https://vignette.wikia.nocookie.net/haikyuu/images/c/c2/Rostro\_de\_Hinata.png/revision/latest?cb=20160210050706&path-prefix=es";
32. App.Database.SaveItemAsyncPersonaje(personaje1);

35. Personaje personaje2 = **new** Personaje();
37. personaje2.Name = "Tobio Kageyama";
38. personaje2.Location = "Col·locador";
39. personaje2.Details = "Juntament amb Hinata, un parella imparable";
40. personaje2.numPuntuacion = "5";
41. personaje2.Image = "https://vignette.wikia.nocookie.net/haikyuu-pedia/images/4/47/Tobio\_Kageyama.jpg/revision/latest?cb=20140721131354";
43. App.Database.SaveItemAsyncPersonaje(personaje2);

46. Personaje personaje3 = **new** Personaje();
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/8bd5055f8ed26c2660a747809f9d943e64c8cfe4/687474703a2f2f36382e6d656469612e74756d626c722e636f6d2f66376362663832383233613034633533343733666561376530336262393933302f74756d626c725f696e6c696e655f6e683373646f464d714a3172696e3631382e6a7067?s=fit&h=758&w=1583&q=80";
54. App.Database.SaveItemAsyncPersonaje(personaje3);

57. Personaje personaje4 = **new** Personaje();
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/haikyuu/images/8/8d/Sugawara.png/revision/latest?cb=20150821230947&path-prefix=es";
65. App.Database.SaveItemAsyncPersonaje(personaje4);

68. Personaje personaje5 = **new** Personaje();
70. personaje5.Name = "Asahi Azumane";
71. personaje5.Location = "Estrella";
72. personaje5.Details = "A qui pots recórrer quan estàs en problemes";
73. personaje5.numPuntuacion = "5";
74. personaje5.Image = "https://vignette.wikia.nocookie.net/haikyuu-pedia/images/c/c4/Asahi\_Azumane.jpg/revision/latest?cb=20140721131410";
76. App.Database.SaveItemAsyncPersonaje(personaje5);

79. Personaje personaje6 = **new** Personaje();
81. personaje6.Name = "Yu Nishinoya";
82. personaje6.Location = "Libero";
83. personaje6.Details = "El millor en salvar les pilotes impossibles";
84. personaje6.numPuntuacion = "5";
85. personaje6.Image = "https://vignette.wikia.nocookie.net/haikyuu/images/8/80/Y%C5%AB\_Nishinoya.png/revision/latest?cb=20170809205325&path-prefix=es";
87. App.Database.SaveItemAsyncPersonaje(personaje6);

90. Personaje personaje7 = **new** Personaje();
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";
98. App.Database.SaveItemAsyncPersonaje(personaje7);

101. Personaje personaje8 = **new** Personaje();
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";
109. App.Database.SaveItemAsyncPersonaje(personaje8);

112. Personaje personaje9 = **new** Personaje();
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";
120. App.Database.SaveItemAsyncPersonaje(personaje9);

123. Personaje personaje10 = **new** Personaje();
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";
131. App.Database.SaveItemAsyncPersonaje(personaje10);
133. Personaje personaje11 = **new** Personaje();
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";
141. App.Database.SaveItemAsyncPersonaje(personaje11);
142. }


146. NavigationPage.SetHasNavigationBar(**this**, **false**);
148. var sub = **new** AbsoluteLayout();
149. splashImage = **new** Image
150. {
151. Source = "splashHaikyuu.jpg",
152. Aspect = Aspect.Fill,
153. WidthRequest = 380
154. };
156. AbsoluteLayout.SetLayoutFlags(splashImage,
157. AbsoluteLayoutFlags.PositionProportional);
158. AbsoluteLayout.SetLayoutBounds(splashImage,
159. **new** Rectangle(0.5, 0.5, AbsoluteLayout.AutoSize, AbsoluteLayout.AutoSize));
161. sub.Children.Add(splashImage);
163. **this**.BackgroundColor = Color.FromHex("#F1913A");
164. **this**.Content = sub;
165. }
167. **protected** **override** async **void** OnAppearing()
168. {
169. **base**.OnAppearing();
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** CustomCellView(**null**));
175. }
177. }
178. }

### App.xaml.cs

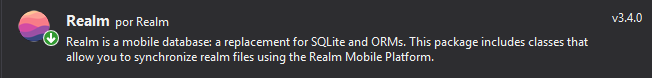
1. **using** Haikyuu.Views;
2. **using** Haikyuu.ViewModels.Base;
3. **using** System;
4. **using** System.IO;
5. **using** Xamarin.Forms;
6. **using** Xamarin.Forms.Xaml;
8. [assembly: XamlCompilation(XamlCompilationOptions.Compile)]
9. **namespace** Haikyuu
10. {
11. **public** partial **class** App : Application
12. {
13. **private** **static** ViewModelLocator \_locator;
15. **public** **static** ViewModelLocator Locator
16. {
17. **get** { **return** \_locator = \_locator ?? **new** ViewModelLocator(); }
18. }
20. **static** HaikyuuItemDatabase database;

23. **public** App()
24. {
25. InitializeComponent();
27. Resources = **new** ResourceDictionary();
28. Resources.Add("primaryGreen", Color.FromHex("91CA47"));
29. Resources.Add("primaryDarkGreen", Color.FromHex("6FA22E"));
31. var nav = **new** NavigationPage(**new** SplashPage());
32. nav.BarBackgroundColor = (Color)App.Current.Resources["primaryGreen"];
33. nav.BarTextColor = Color.White;
35. MainPage = nav;
37. //MainPage = new NavigationPage(new SplashPage());
38. }
40. **public** **static** HaikyuuItemDatabase Database
41. {
42. **get**
43. {
44. **if** (database == **null**)
45. {
46. database = **new** HaikyuuItemDatabase(Path.Combine(Environment.GetFolderPath(Environment.SpecialFolder.LocalApplicationData), "HaikyuuSQLite.db3"));
47. }
48. **return** database;
49. }
50. }
52. **public** **int** ResumeAtTodoId { **get**; **set**; }
54. **protected** **override** **void** OnStart()
55. {
56. // Handle when your app starts
57. }
59. **protected** **override** **void** OnSleep()
60. {
61. // Handle when your app sleeps
62. }
64. **protected** **override** **void** OnResume()
65. {
66. // Handle when your app resumes
67. }
68. }
69. }

# 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;
9. **namespace** Haikyuu
10. {
11. **public** **class** HaikyuuItemDatabase
12. {
13. **private** Realm \_realm;
15. **public** HaikyuuItemDatabase()
16. {
17. **try**
18. {
19. \_realm = Realm.GetInstance();
20. AddDataDB();
21. }
22. **catch** (Exception ex)
23. {
25. }
26. }
28. **private** **void** AddDataDB()
29. {
30. **if** (GetAll().Count == 0)
31. {
32. var chars = **new** ObservableCollection<Personaje>
33. {
34. **new** Personaje
35. {
36. Name = "Shoyo Hinata",
37. Location = "Bloquejador central, Carnada",
38. Details = "Vol arribar a ser com el Petit Gegant",
39. numPuntuacion = "5",
40. Image = "https://vignette.wikia.nocookie.net/haikyuu/images/c/c2/Rostro\_de\_Hinata.png/revision/latest?cb=20160210050706&path-prefix=es"
41. },
42. **new** Personaje
43. {
44. Name = "Tobio Kageyama",
45. Location = "Col·locador",
46. Details = "Juntament amb Hinata, un parella imparable",
47. numPuntuacion = "5",
48. Image = "https://vignette.wikia.nocookie.net/haikyuu-pedia/images/4/47/Tobio\_Kageyama.jpg/revision/latest?cb=20140721131354"
49. },
50. **new** Personaje
51. {
52. Name = "Daichi Sawamura",
53. Location = "Wing Spikers, Capità",
54. Details = "Especialista en la defensa i una persona necessària per al equip",
55. numPuntuacion = "5",
56. Image = "https://em.wattpad.com/8bd5055f8ed26c2660a747809f9d943e64c8cfe4/687474703a2f2f36382e6d656469612e74756d626c722e636f6d2f66376362663832383233613034633533343733666561376530336262393933302f74756d626c725f696e6c696e655f6e683373646f464d714a3172696e3631382e6a7067?s=fit&h=758&w=1583&q=80"
57. },
58. **new** Personaje
59. {
60. Name = "Koshi Sugawara",
61. Location = "Segon col·locador, Segon capità",
62. Details = "No serà titular, però es un dels millors en la seva feina",
63. numPuntuacion = "5",
64. Image = "https://vignette.wikia.nocookie.net/haikyuu/images/8/8d/Sugawara.png/revision/latest?cb=20150821230947&path-prefix=es"
65. },
66. **new** Personaje
67. {
68. Name = "Asahi Azumane",
69. Location = "Estrella",
70. Details = "A qui pots recórrer quan estàs en problemes",
71. numPuntuacion = "5",
72. Image = "https://vignette.wikia.nocookie.net/haikyuu-pedia/images/c/c4/Asahi\_Azumane.jpg/revision/latest?cb=20140721131410"
73. },
74. **new** Personaje
75. {
76. Name = "Yu Nishinoya",
77. Location = "Libero",
78. Details = "El millor en salvar les pilotes impossibles",
79. numPuntuacion = "5",
80. Image = "https://vignette.wikia.nocookie.net/haikyuu/images/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/images/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/images/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\_kGXcAE3oS2.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/haikyuu/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/haikyuu/images/d/da/Takeda-sensei.png/revision/latest?cb=20140616135813"
121. },
122. };
124. **foreach** (var item **in** chars)
125. {
126. SaveItem(item);
127. }
128. }
129. }
131. **public** List<Personaje> GetAll()
132. {
133. **return** **new** List<Personaje>(\_realm.All<Personaje>());
134. }
136. **public** **void** SaveItem(Personaje item)
137. {
138. item.id = \_realm.All<Personaje>().Count() + 1;
140. \_realm.Write(() => \_realm.Add(item));
141. }
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**.