

About my *ToDoList* App:

It's Cross-Platform *IOS/Android* Application with *Xamarin*, Visual Studio and *C#* using *MVVM* Pattern.

The App functionality:

- View a list of Todo Items.
- Add a new Todo item to the list of tasks.
- Mark a Todo item as '*Done*'.
- Delete an existent Todo Item from the list of tasks.

In all cases all the tasks are stored in a local SQLite database.

My programming Main steps:

- Creating Cross-Platform *Xamarin* App with Target platforms (*Android / IOS*) and shared code (use *.Net standard*) and using *git* for version control.
- Adding a new Package "*SQLite-net-pcl*" under my project NuGet node, as a local database where my data will be stored.
- Creating a new folder in the Todo project called "*Data*" representing the connecting data properties and actions in the user interface elements and contains
 - "*TodoItemsDB.cs*" file, which have a declaration of a new class to create the database and to read, write, and delete data from it using asynchronous SQLite.NET APIs.
- Creating a new folder in the Todo project called "*Models*" represents the business entities of the application and contains
 - "*ItemsEntity.cs*" file, which have a declaration of a new class to get and set the data of each Todo Item in the application "*ID*, *Name* and *Done*" attributes.
- Creating a new folder in the Todo project called "*Views*" represents the actual pages of the application, along with all of the elements that make them up, including custom controls and depends on platform APIs to render the application's user interface(UI) and contains :
 - "*ItemManager.xaml*" file: Contains the UI of TodoItem Page.

- “[ItemManager.xaml.cs](#)” file: Contains the event handlers of TodoItem page’s buttons (**Save** , **Delete** and **Cancel**).
- “[ItemManagerCS.cs](#)” file: Contains TodoItem page content layout and binding the page controls to the database.
- “[MyTodoItems.xaml](#)” file: Contains the UI of TodoList Page.
- “[MyTodoItems.xaml.cs](#)” file: Contains the event handlers of TodoList page (**ItemAdded**, **ListItemSelected**).
- “[MyTodoItemsCS.cs](#)” file: Creating the new Listview layout, Control the navigation between the two pages(**MyTodoItems Page** and **ItemManager Page**) and passing the Item information between them.
- Coding in file **App.cs**: which adjusting the application layout and returns a local file path for storing the database.
- Testing my App.
- Checked out my Todo project to GitHub and sent you the link :)

Note:

For more technical details about my code implementation, please open my project solution and check the comment lines in my code.

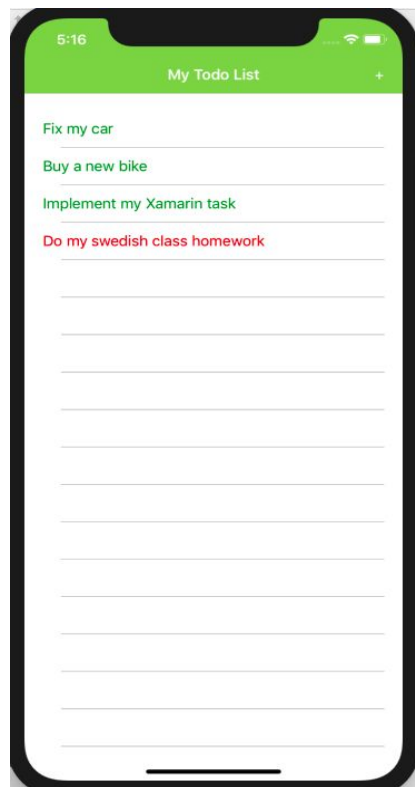
The application User Interface UI:

The application consists of 2 pages, user can navigate from one page to the other one.

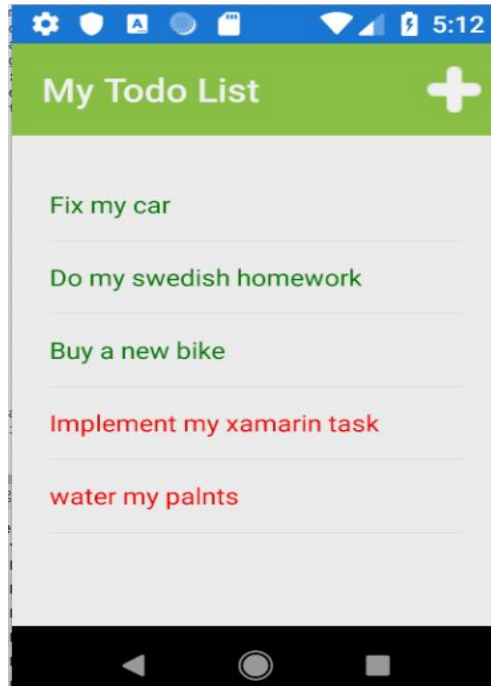
The first page (main page) called:

1. **MyTodoItems** Page:
 - a. **Page Title:** [My Todo List](#)
 - b. **Page Task:**
contains a list of all the Todo items, the Todo items which are Done are written in a [green](#) font color and the others are written in a [red](#) font color.
 - c. **Page controls:**
 - [ListView](#) -->[ViewCell](#) -->[StackLayout](#) (which contains)
 - [Label](#): Bound to Todo Item **Name** saved in the SQLite Database, and it’s Font color is depending on the value of **Done** attribute saved in the SQLite Database.
 - [ToolBarItem](#): in the Top-Right of the page. It’s text is “+” User can click on it to create a new Todo Item, so the application navigates to the other page to let user enter the Item name and status([Done](#) or [not](#)).
 - d. **Page Layout:**

IOS Layout:
iPhoneXR



Android Layout:
Nexus 4



The Second page called:

2. **ItemManager** Page:

a. **Page Title:**

- In case of editing an existing Item it is Binded to the name of the selected Item from the first page.
- In case of adding new item it will be empty.

b. **Page Task:**

User can use this page to

- Add a new Todo item to his/her list
- Edit an existent Todo item (by changing the Item name or status as Done or not)
- Delete an existent Todo item.

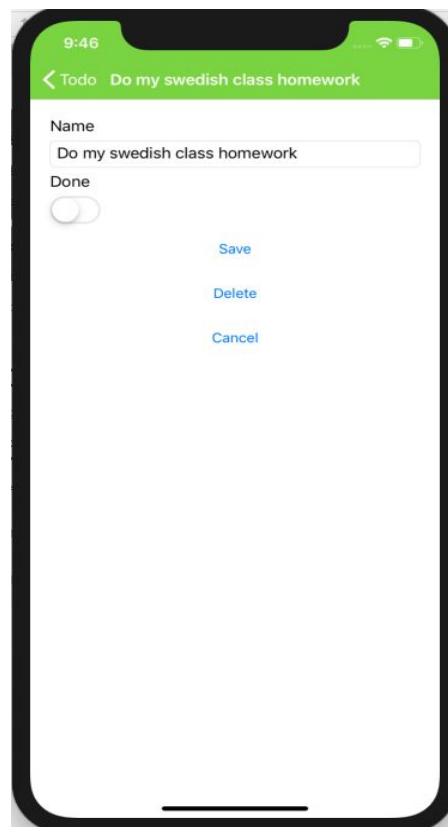
c. **Page controls:**

- **StackLayout** (which contains)
 - **Entry**: where user can enter or edit the Todo Item name, binded to the column **Name** saved in the SQLite Database.
 - **Switch**: where user can *check/uncheck* the Todo Item as **Done** Item or **not**, binded to the column **Done** saved in the SQLite Database (Boolean attribute).
 - **Save Button**: User can click on it to **save** the Todo item changes.

- *Delete Button:* User can click on it to **delete** the Todo item.
- *Cancel Button:* User can click on it to **discard** any Todo item changes.

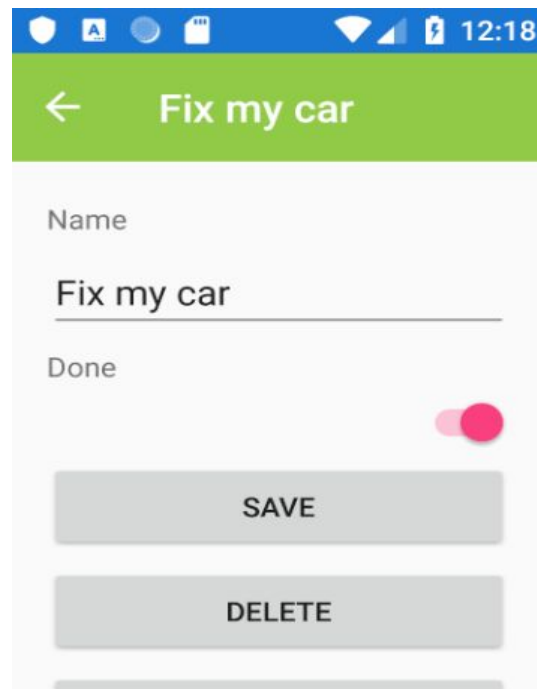
d. Page Layout:

IOS Layout:
iPhoneXR



Android Layout:

Nexus 4



Note:

I took a look in some online samples for inspiration and to learn how to apply MVVM architecture in my project (I enjoyed learning it :))...

Thanks for giving me this opportunity to gain this experience :)