**100 days of Swift (UI)**

**View Controller**

The UIKit means we will have access to Apples UI toolkit, while the UIViewController is Apple’s main screen type, in line **super**.viewDidLoad() we tell Apple’s UI View controller to run its code before ours.

A Bundle is just a directory containing our compiled program and all our assets. UITableViewController is another option for an iOS Swift, which derives from normal UIViewController.

\* To add a new View Controller, follow these steps:

1. Add it from the object library.
2. Associate it to a class on the dropdown *Class* of the *Identity Inspector.*
3. If the view is the first to be shown when the app runs, go to the *Metrics* tab and click the checkbox *Is Initial View Controller.*
4. Add a *Navigation Controller* by going to *Editor -> Embed In -> Navigation Controller.*

To tweak parameters of a table:

1. To modify how many rows we want to show, use fx **override** **func** tableView (numberOfRowsInSection) { … }
2. To modify the content of each cell, use fs **override** **func** tableView (cellForRowAt) { cell.textLabel?.text = pictures[indexPath.row] }

For each new screen, you will be adding a new Cocoa Touch file and follow the steps \*.

**Auto layout**

The easiest way to position effectively an element in the auto layout is the next:

1. Place it and grow it as desired.
2. Select it and go to *Editor -> Resolve Auto Layout Issues -> Reset to Suggested Constraints*

To add a component from the Storyboard to the code, just select it, then with *control* key pressed, toggle its arrow to the associated Cocoa Class.

To center a component using auto constraints, *select it -> control click it to the out-reference area -> select areas (top, center)*

To add a new button action, just *control click it -> put in in controller -> select “Action” connection.* You can then add as many buttons to that action as you want.

**Useful shortcuts**

|  |  |  |  |
| --- | --- | --- | --- |
| **Shortcut** | **Action** | **Shortcut** | **Action** |
| cmd + shift + l | Display object library |  |  |
|  |  |  |  |
|  |  |  |  |

**MVC Architecture**

Let us remember that the **model** contains data, or a blueprint on how to build something, the **view** displays the data from the model which can get its data because the **view** controller controls, shares and modifies, and updates all of that information.