**Clone YO app with Parse.com Swift and iOS 8.**

In this tutorial, you’ll learn how to make an application like YO (<https://itunes.apple.com/us/app/yo./id834335592?mt=8>) names Clone YO with cloud service Parse.com Swift and iOS 8.

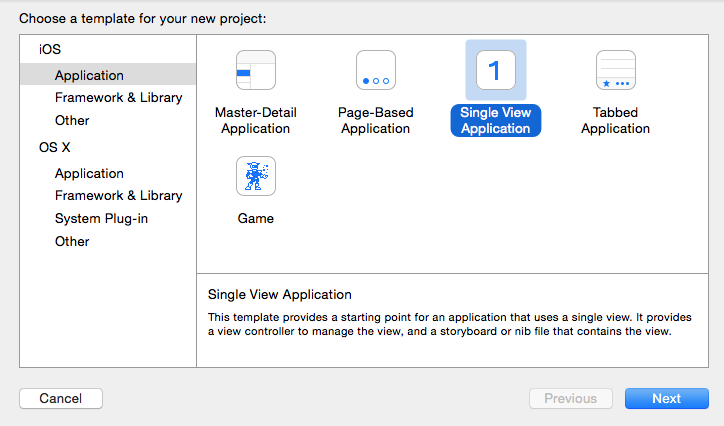
Parse.com is backend cloud service from Facebook.

In the process of going through this tutorial, you’ll get some excellent practice with Swift, new iOS 8 features such as Interactive Push Notification.

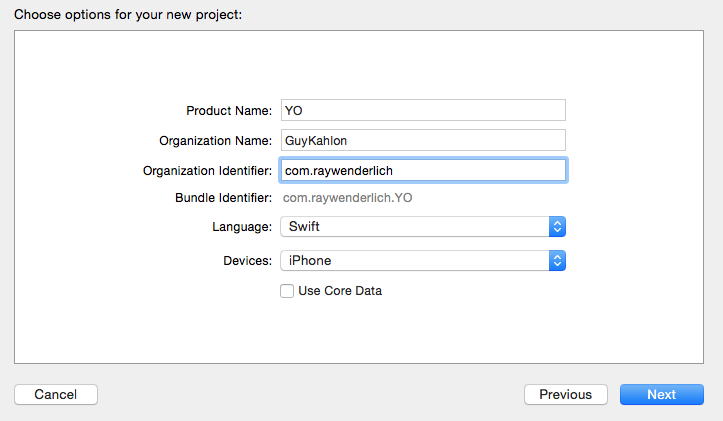
You’ll also learn a lot about social app architecture and best practices.

**Getting Started**

Start up Xcode 6.x, go to ***File\New\Project…***, choose the ***iOS\Application\Single View Application*** template and click ***Next***. Fill out the options as follows:

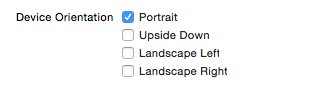


* Product Name: ***CloneYO***
* Language: ***Swift***
* Devices: ***iPhone***



Click ***Next***, choose a folder for your project and click ***Create***.

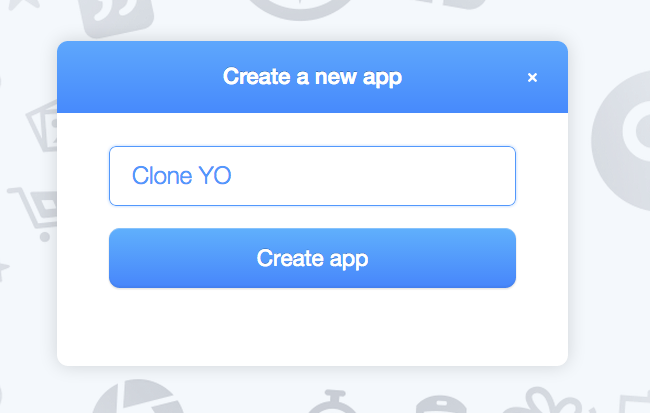
This is a portrait-only application, so open the ***Target Settings*** screen and in the ***General*** tab, make sure only***Portrait*** is checked in the ***Device Orientation*** section:



The start project is ready, before starting the development of your CloneYO app, the first step is to create an app in the Parse.com backend. Every developer and every app requires a unique identifier.

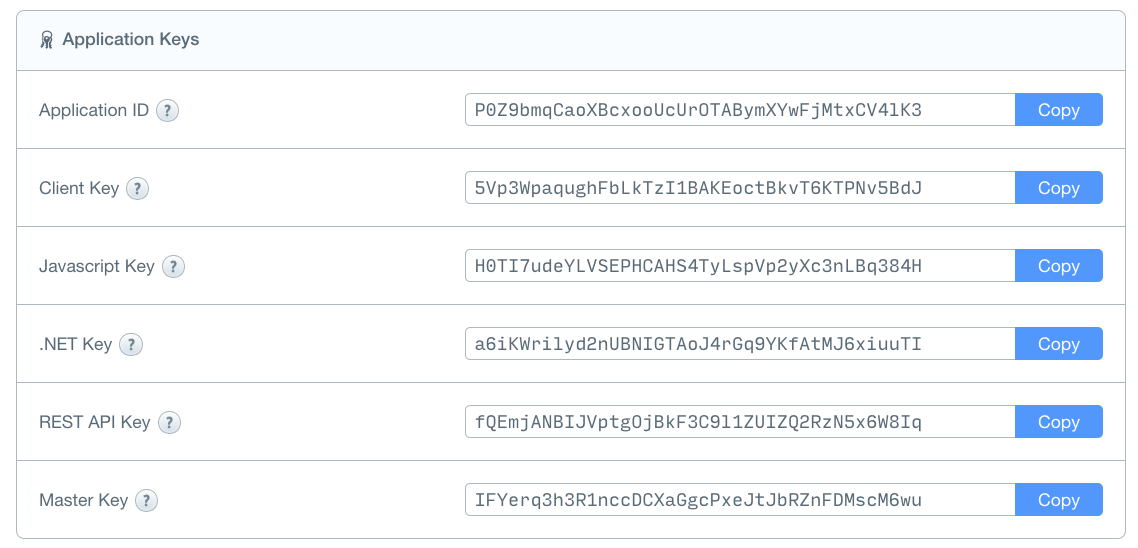
Your first step is to visit [Parse.com](http://parse.com/) and click **Sign Up** to create a new account or login with existing account.

Once the account is created, you will be asked to create your first app. every app you use with the backend must be registered separately. In this case, we call it “Clone YO”



1. Continue according to instructions on the screen.

At the end of the process you should have the following information:

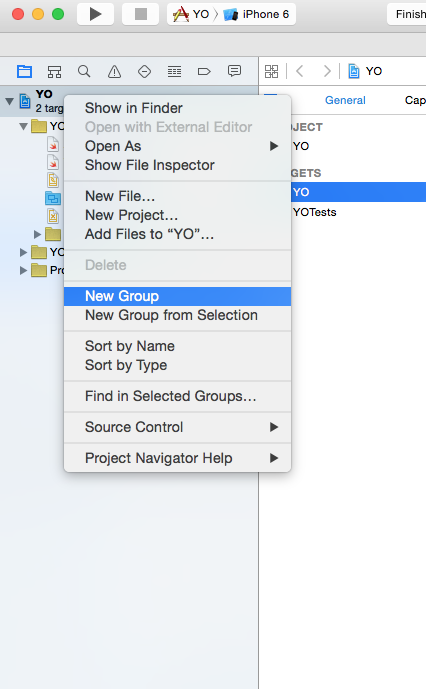


Great, now we have a start project and app in Parse.

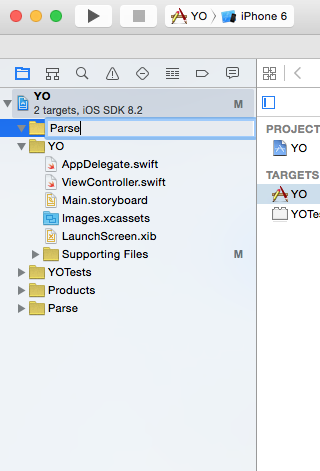
Now we should to integrate parse iOS SDK into our project, download the latest Parse SDK for iOS from <https://parse.com/docs/downloads/>

Unzip the file.

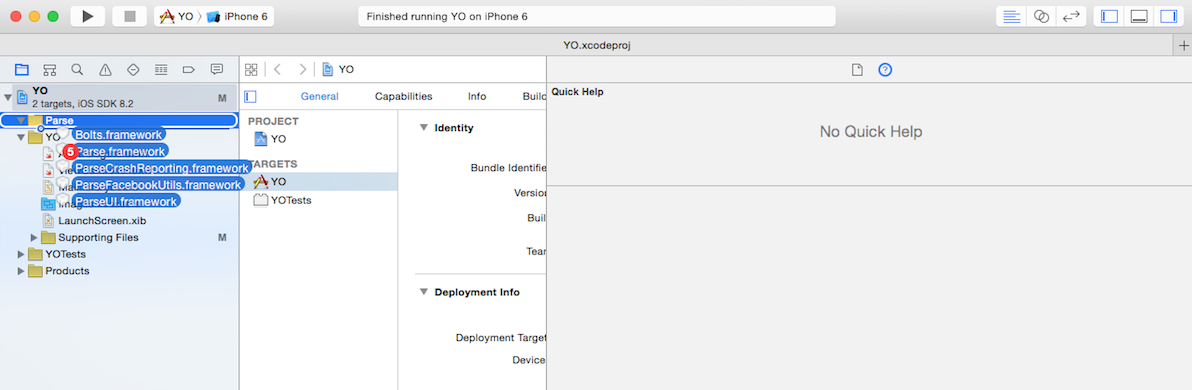
Create new group.



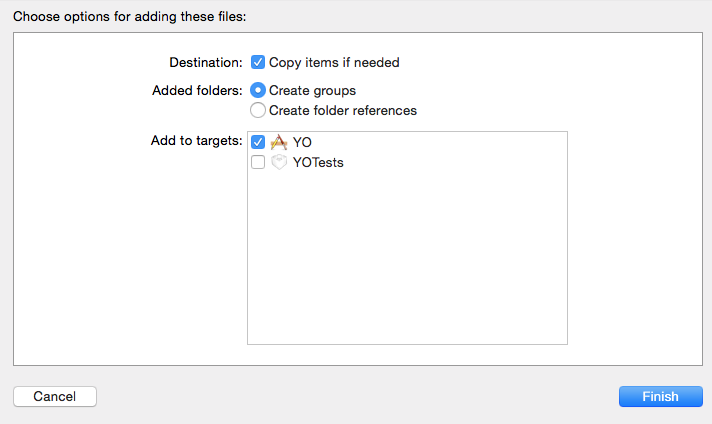
Called “Parse”



Drag and drop all the files from the Parse iOS SDK you unzip earlier.



Don’t forget select “Copy items if needed”.



Parse iOS SDK is Objective C code and our project will written with Swift, So, **how we can use with legacy Objective C code from Swift project?**

All we need is bridging file, that ‘bridge’ between the code is written in Swift code to the Objective C code.

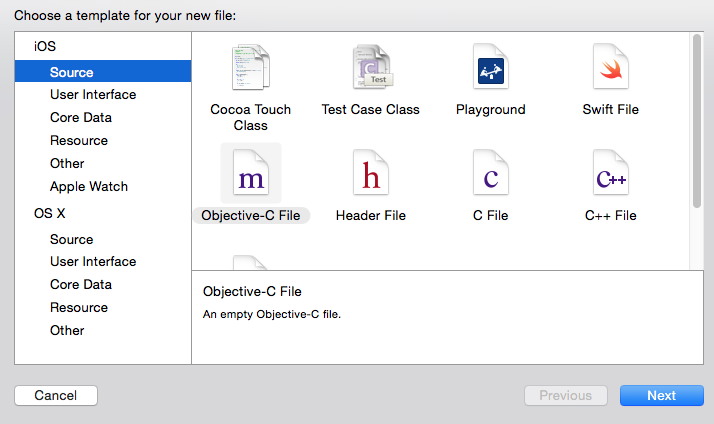
We should to add all the imports (Objective C code) to the bridging file.

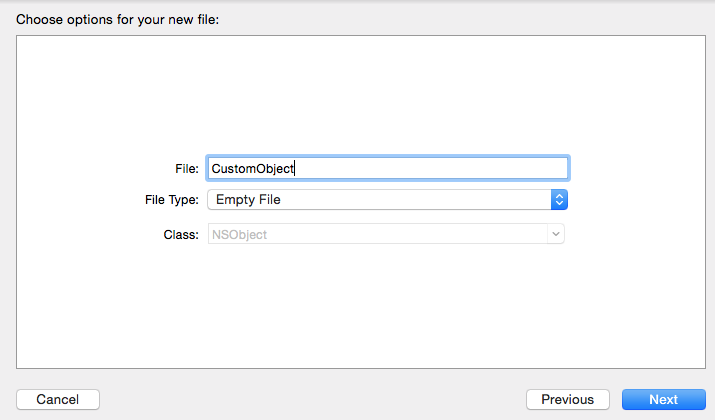
**How we create a bridging file?**

There are several ways to create this bridge file, I would prefer the automatic way.

We create a new Objective .m file calls CustomObject.

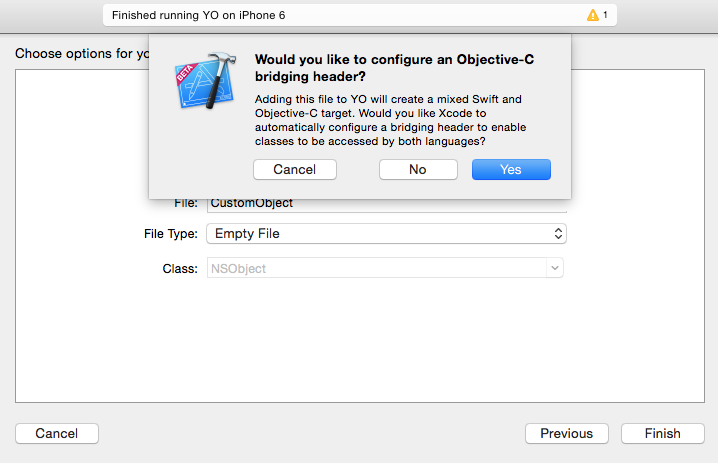
Go to ***File\New\File… (or just use the Xcode keyboard Shortcuts*** ⌘N***)***, choose the ***iOS\Source\Objective-C File***  and click ***Next***. Fill out the options as follows:





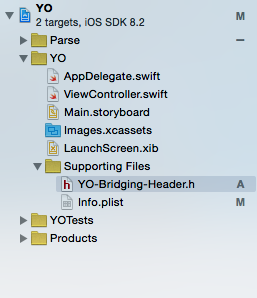
Click next.

Now the Xcode create a bridging file for us.

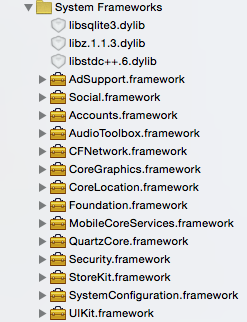


Click **Yes** and now you can remove the CustomObject. (We aren’t really need this file)

Please ensure you have the YO-Bridging-Header.h file in your project.



Parse iOS SDK also required list of other frameworks:



To add these frameworks, go to the YO target-> ***General section -> Linked Frameworks and Liberaries.***

Ok, We have start project, app Id in Parse.com, iOS Parse SDK that wrote in Objective C code and we have bridging file to connect between our Swift code and Objective C code.

Open the YO-Bridging-Header.h and add:

#import <Parse/Parse.h>

Now we want to connect our Parse application with our application code.

Open appDelegate.swift and replace with the following method:

func application(application: UIApplication, didFinishLaunchingWithOptions launchOptions: [NSObject: AnyObject]?) -> Bool {

Parse.setApplicationId("P0Z9bmqCaoXBcxooUcUrOTABymXYwFjMtxCV4lK3", clientKey: "5Vp3WpaqughFbLkTzI1BAKEoctBkvT6KTPNv5BdJ")

return true

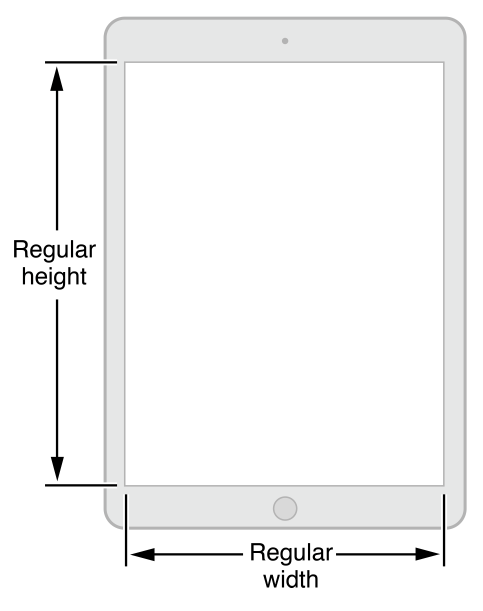
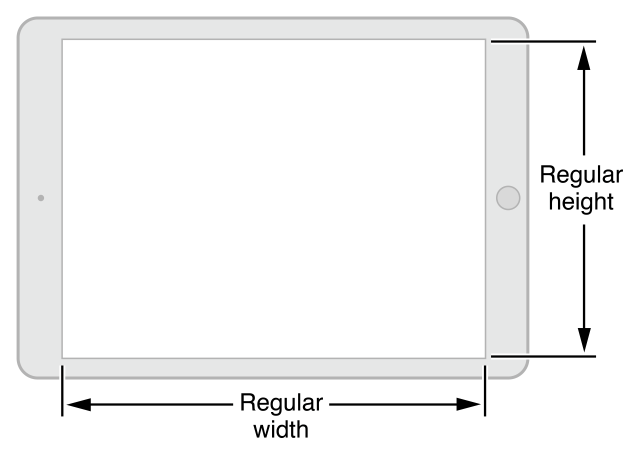
}

Build and run the app to be sure that all works fine.

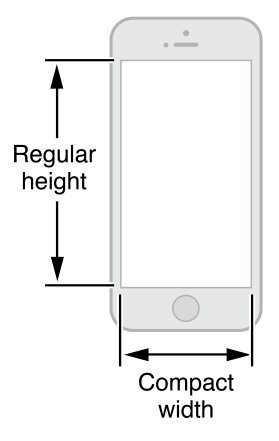
Open the LaunchScreen.xib.

Our application support only iPhone on portrait.

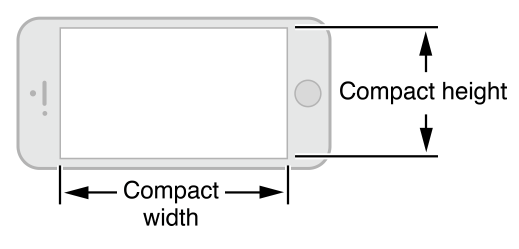
iPad portrait and landscape is **Regular** height and width:



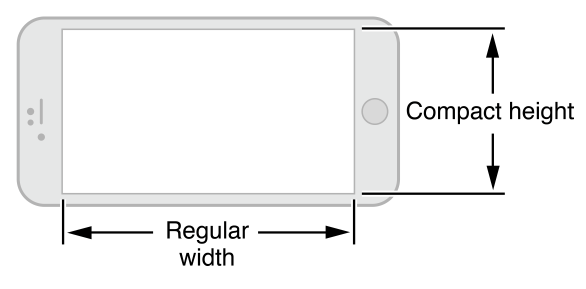
iPhones (include 6 plus) portrait is **Regular** height and **Compact** width:



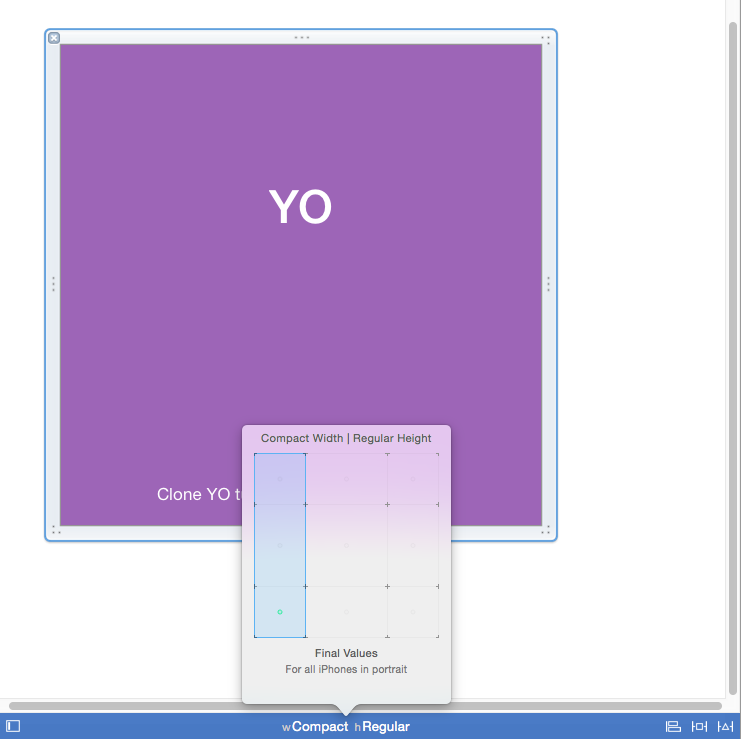
iPhones (not include 6 plus) landscape is **Compact** height and width:



iPhone 6 plus landscape is **Compact** height and **Regular** width:



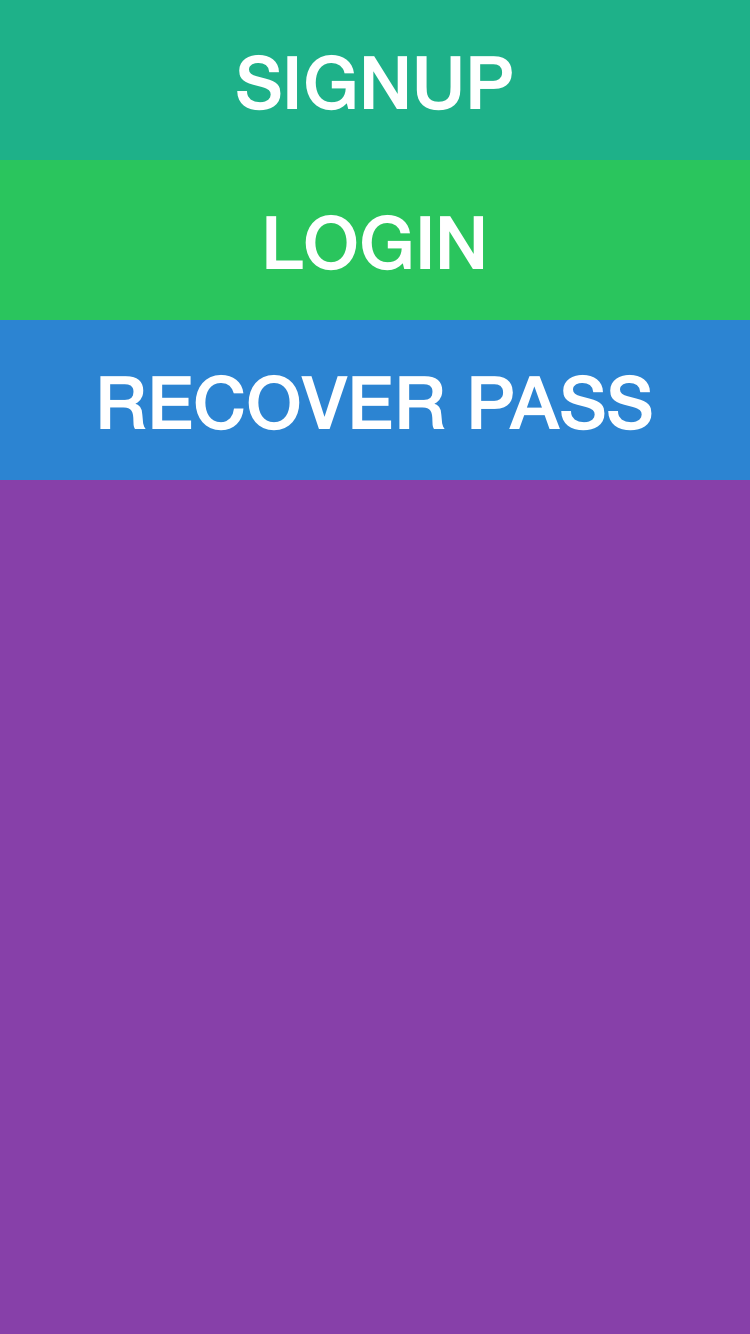
So it’s easy in our case, we should change the size classes to wCompact hRegular.



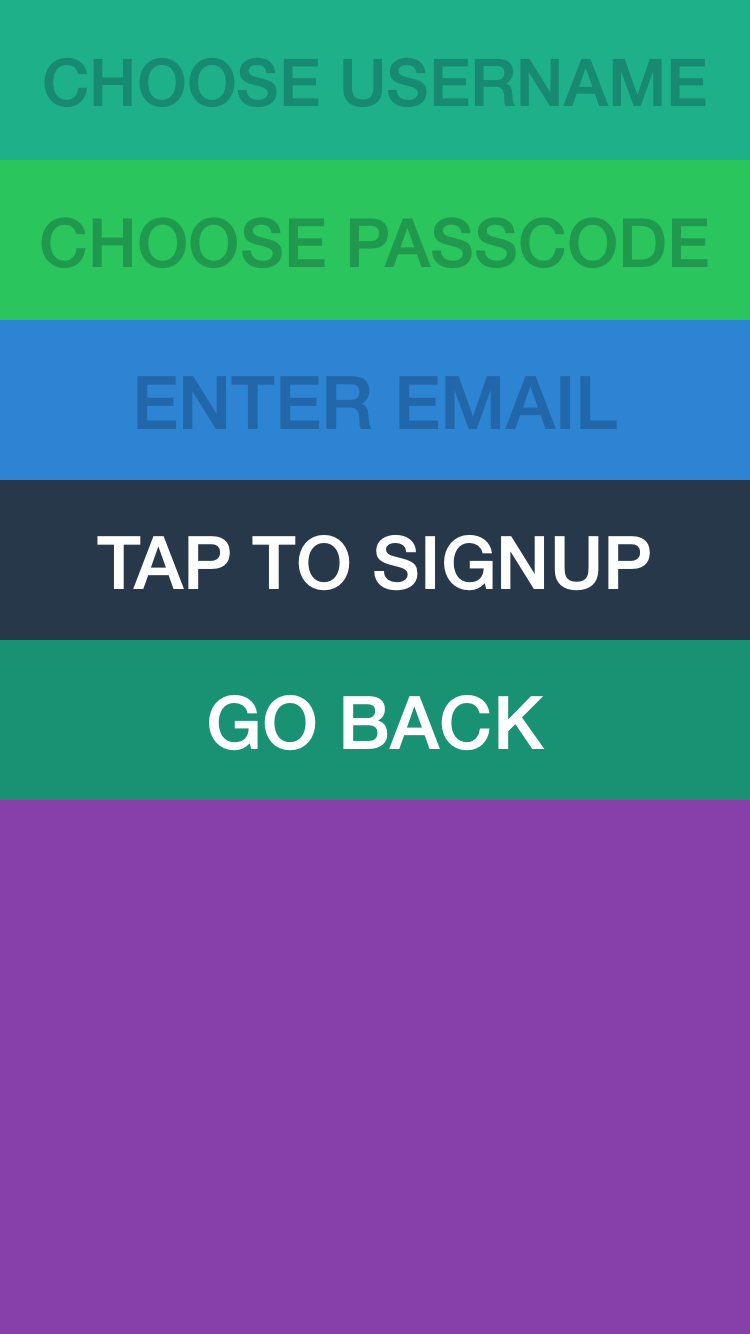
Open the LaunchScreen.xib update the UILable on the bottom screen to “Clone YO tutorial RayWenderlich.com” and Changed the background color to White.

Ok, finally we start to developer our application. If we familiar with the first YO version you know that is vey simple app, in our app we have four screens before the user are logined:

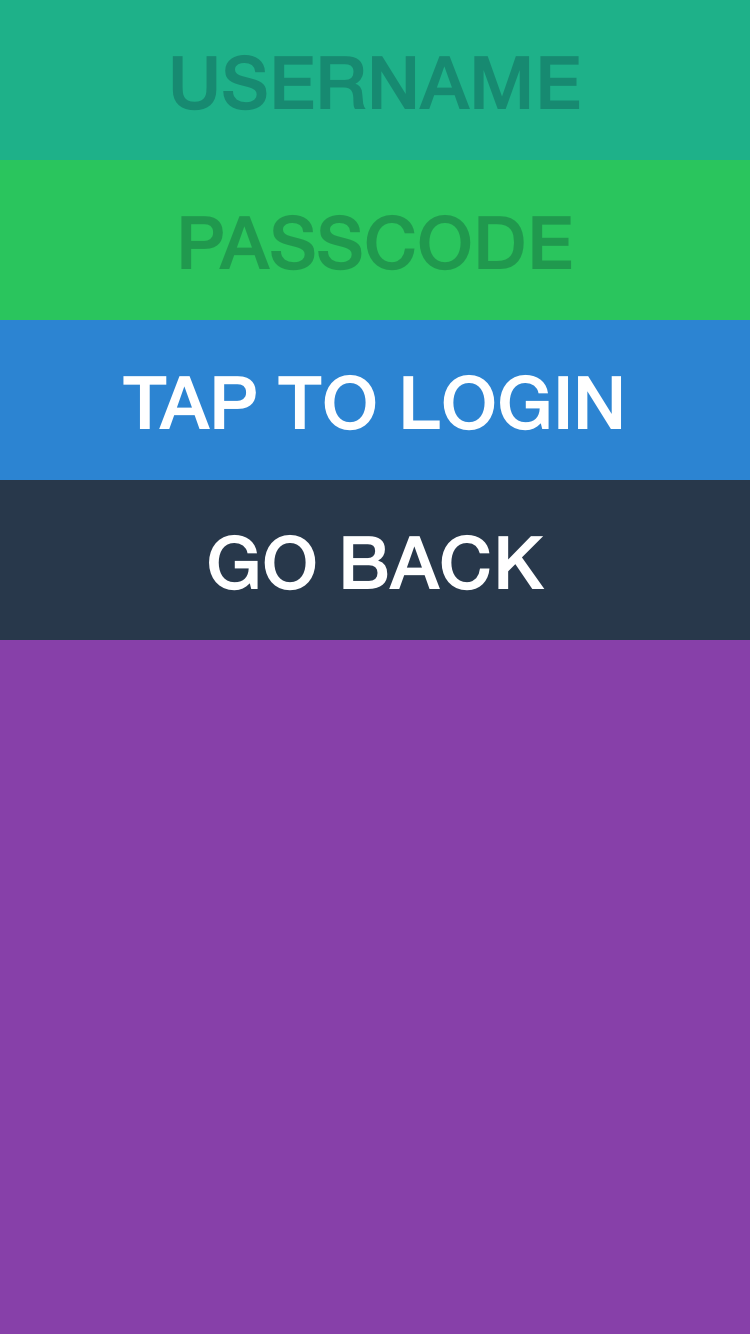
Main:



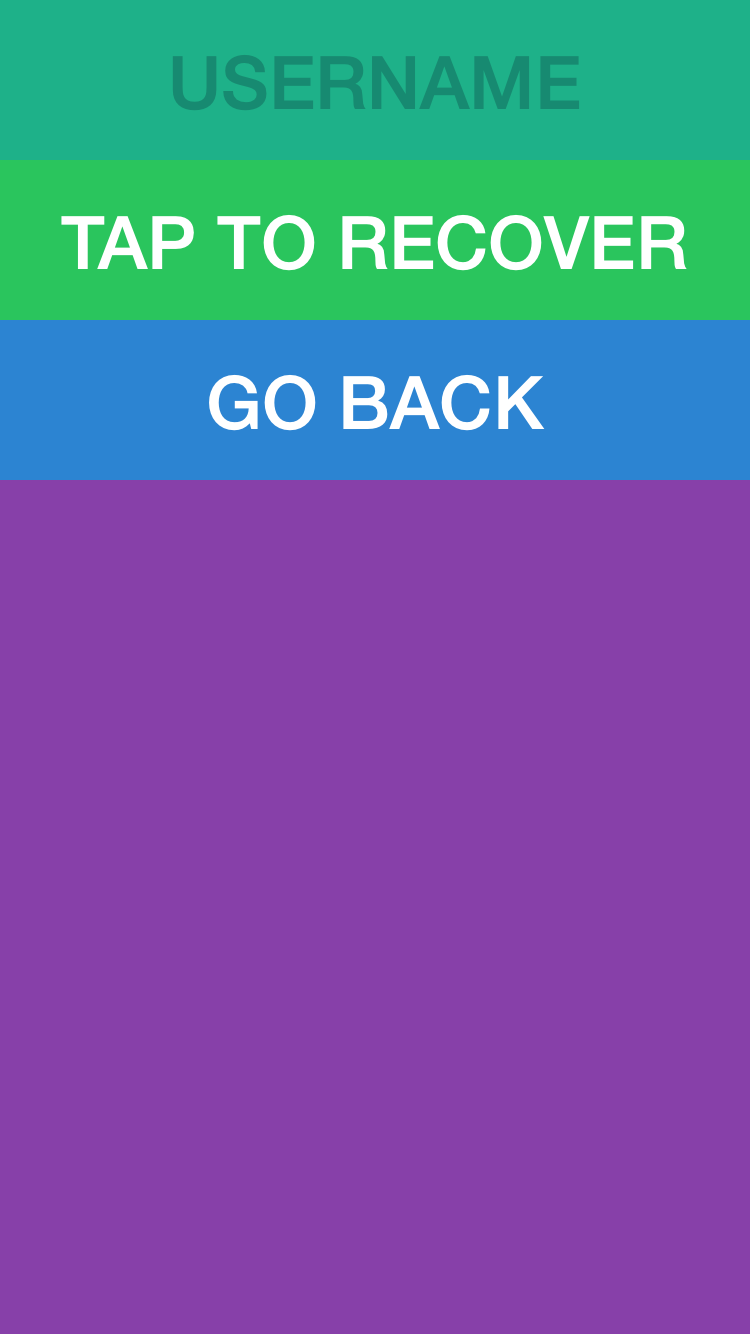
Signup:



Login:



Recover password:



And only one screen after the user login:

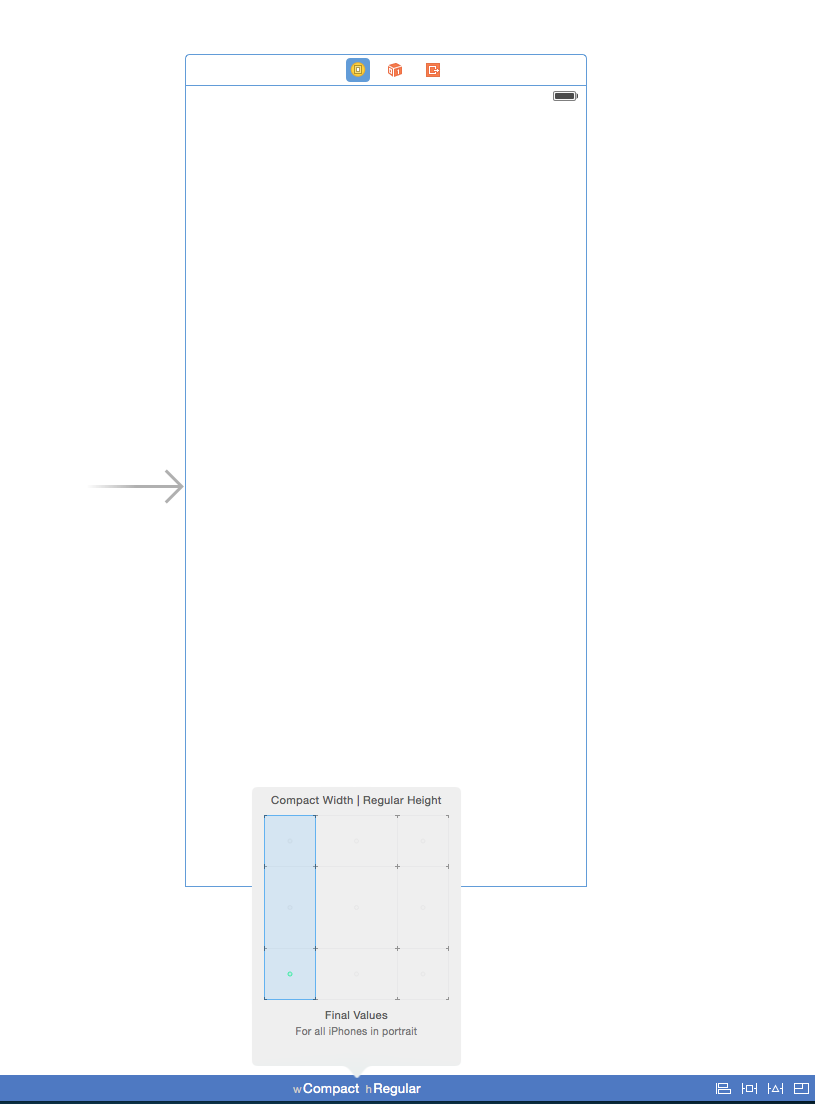


The user registers by name, and sent YO push notification for his friend also by name, simple.

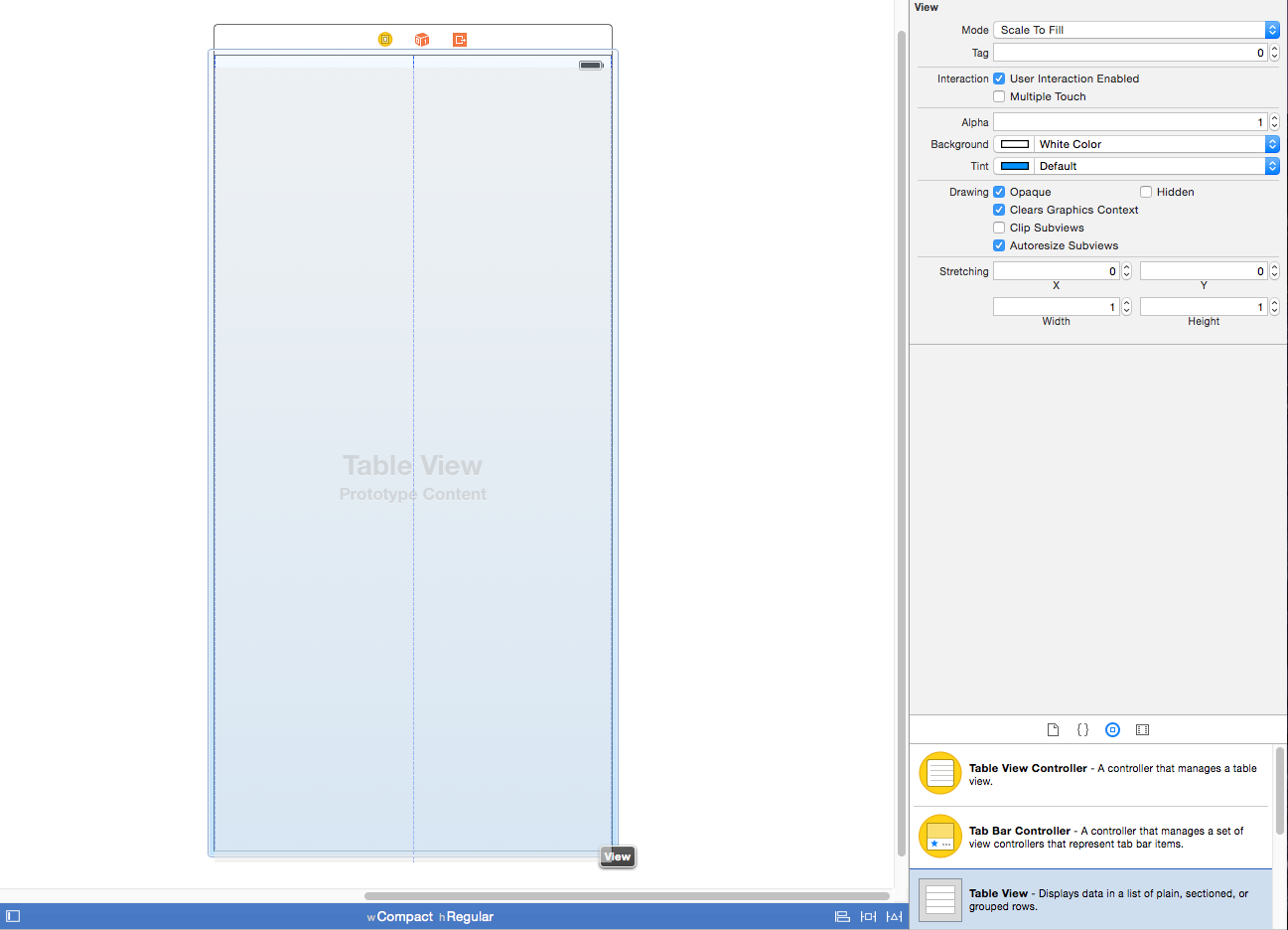
We will create this app with single page. That contain table view.

**The Main screen**

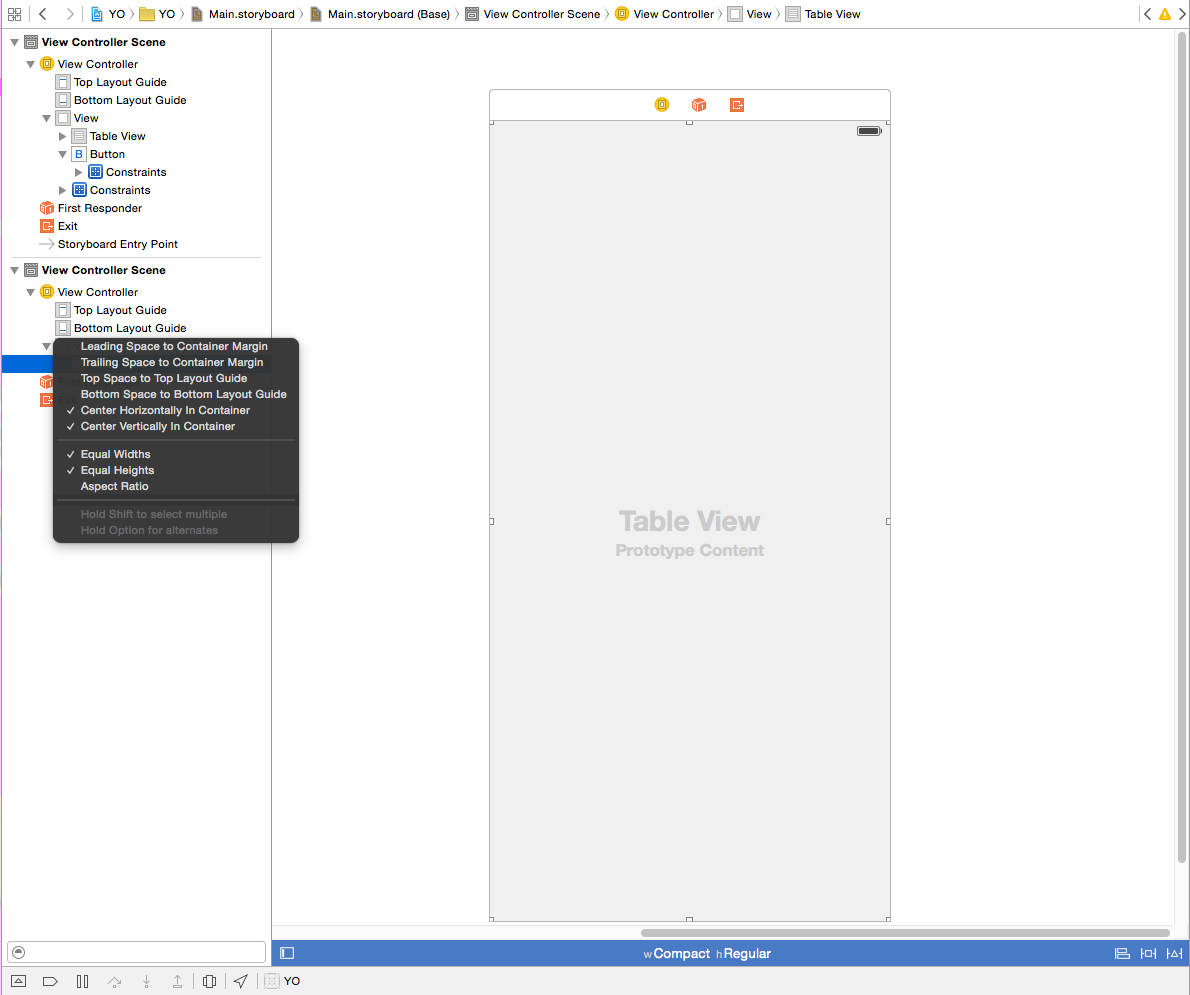
Open the Main.storyboard file, again, we should to change the size classes for iPhone portrait only same as we changed on the LaunchScreen.xib



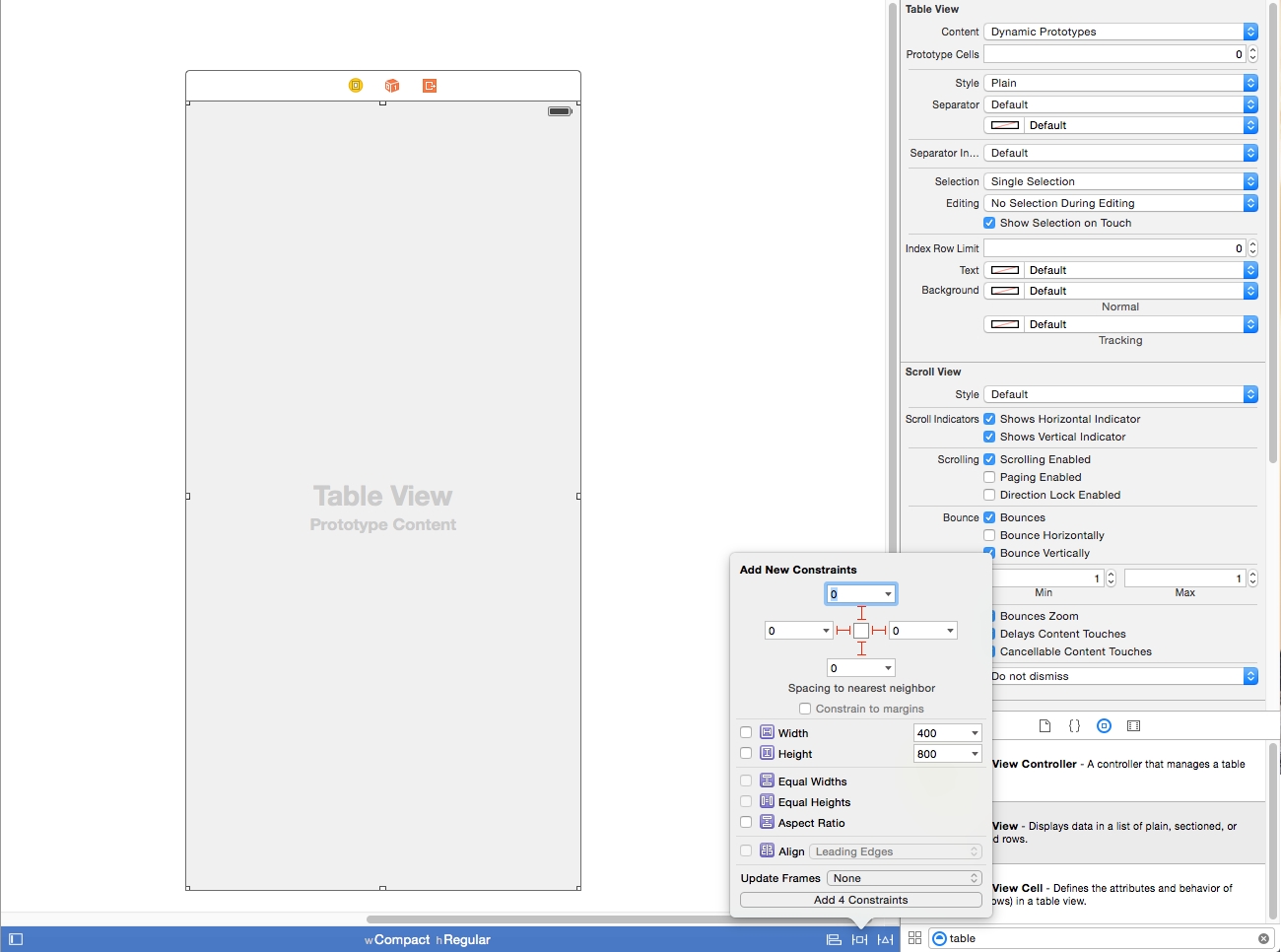
Drag and drop a table view from the object library to the existing view controller:



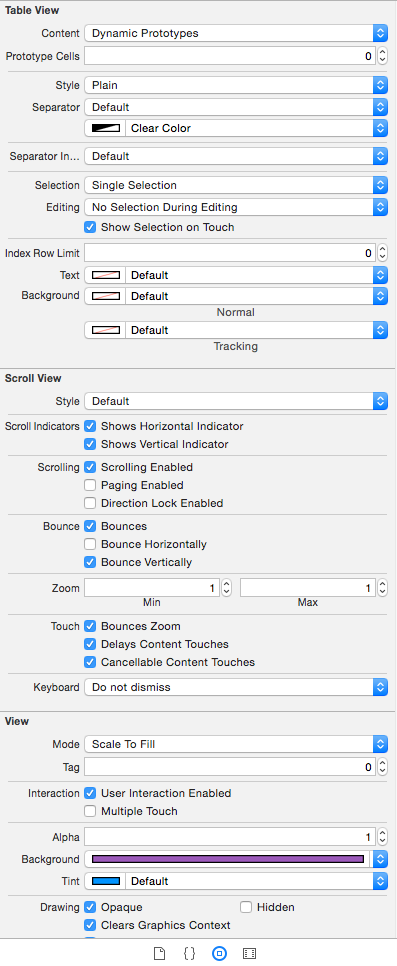
Update the auto layout, We want that the table view will be the same width and the same height as the view controller.



Update the auto layout constrains that the table will be on the all view controller, and change the background color to purple (RGB:135,64,169)

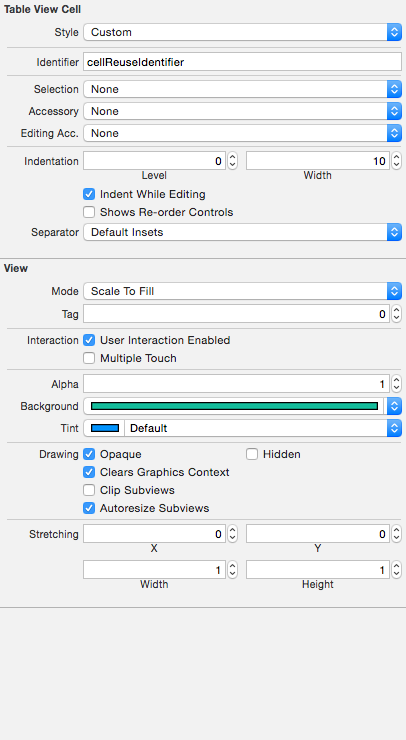


Update on the attributes inspector the background of the table view to be purple (RGB:135,64,169) and the separator to be clear color.



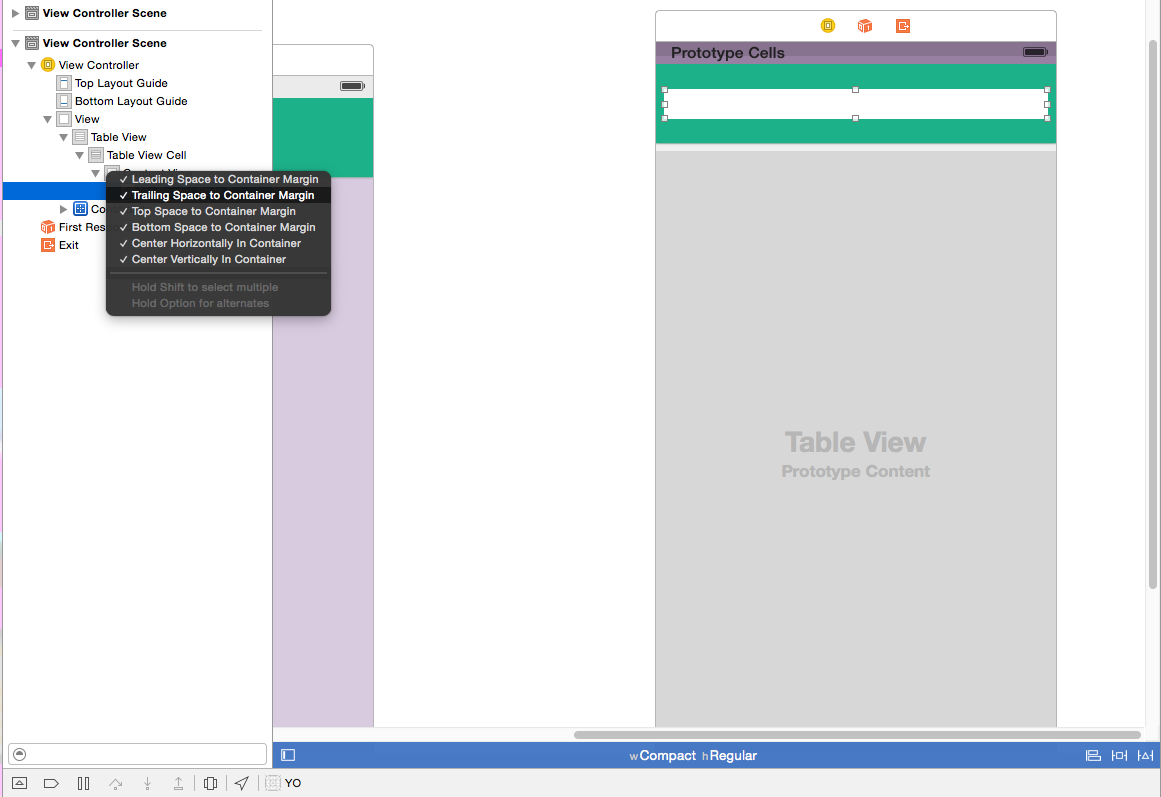
And in the size inspector the **row Height** to be 80.

Now, drag and drop table view cell from the object library, Change the background color to be green (RGB:30,177,137) only for convenience, The selection to be None And don’t forget to set identifier, we use with ‘cellReuseIdentifier’ string.

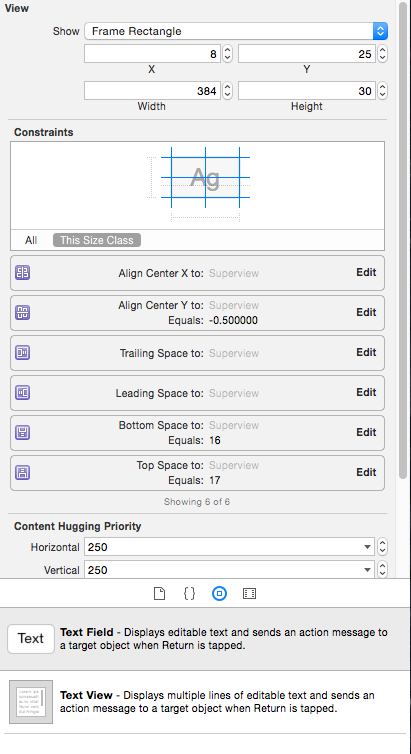


And then drag and drop UITextField to the cell, and No then change on the attribute inspector the background color to be White and the Border style to be without border.

Update the auto layout constraints:



Change on the size inspector all the “magic number” to 0:



And now you should to get something like that:

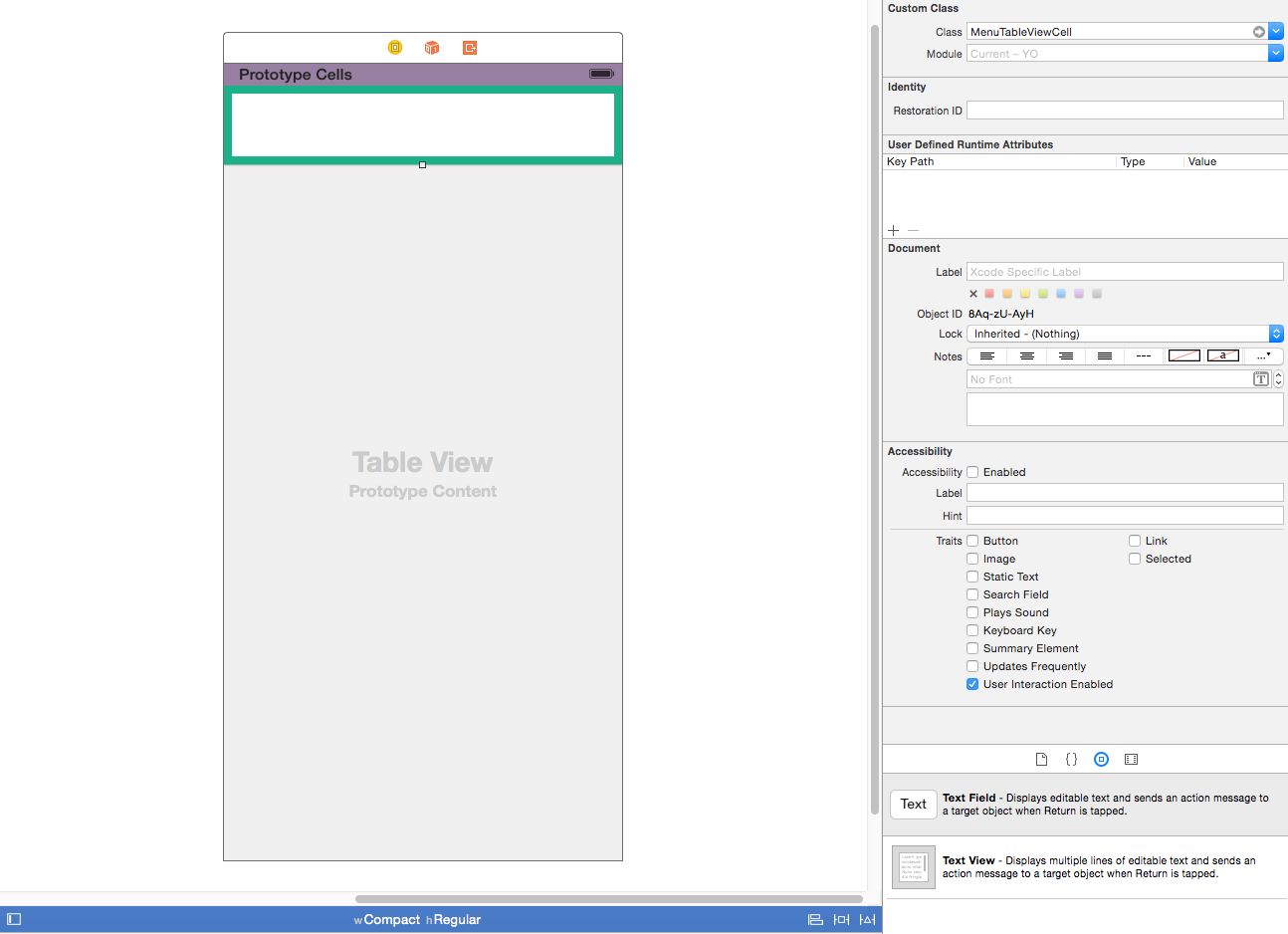


Now we should to create a Swift class for our custom cell, open the ViewController.swift file and create on the top a new swift class that inheritance from UITableViewCell.

class MenuTableViewCell: UITableViewCell{

}

Now open again and Main.storyboard file and update your cell to be a class of MenuTableViewCell on the identify inspector.



Ok, Now we create a IBAutlet for our UITextFiled.

class MenuTableViewCell: UITableViewCell{

@IBOutlet weak var titleTextField: UITextField!

}

Now lets create an IBAutlet also for the tableview, And we also use with a Swift fetchers calls Properties observers and set the viewController as a data source and delegate for the table View.

@IBOutlet weak var tableView: UITableView!{

didSet{

tableView.dataSource = self

tableView.delegate = self

}

}

We will see more things to do with the Property observers when we will use with the model, But for more details you also can read in Apple website: <https://developer.apple.com/library/prerelease/ios/documentation/Swift/Conceptual/Swift_Programming_Language/Properties.html>

Ok, Now our controller is a delegate and data source for the table view but we never no declared it as such.

In swift is very common to