

# PikLab

## User Guide

---

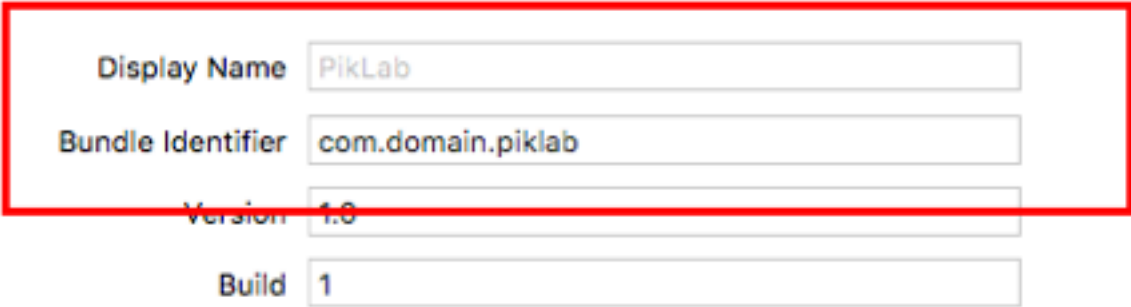
Thanks for purchasing **PikLab** iOS Swift Universal Photo Editor App template, we appreciate your support.

### 1. BASIC SETUP

#### Bundle Identifier & App name

Once you have generated a new **App ID** and **Distribution Provisioning** file from the Apple Developer portal (in order for you to be able to submit the binary of your reskinned version of this app for Review on **iTunes Connect**), you must change the **Bundle Identifier** accordingly to the one you generated in your provisioning profile, and you have also to select the Team accordingly to your own one (the one you've registered in the Apple Developer portal).

Also, you must change the **Display Name** as you wish, so you and your users will see your own app's name underneath the app's icon on the device (see below):



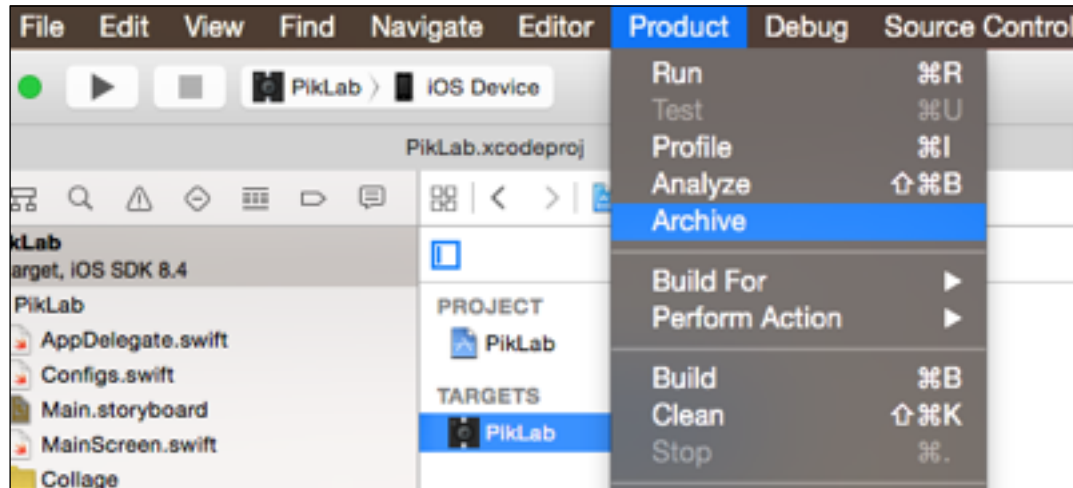
The screenshot shows the 'General' tab of the Xcode project settings. A red rectangle highlights the 'Display Name' and 'Bundle Identifier' fields. The 'Display Name' field contains 'PikLab'. The 'Bundle Identifier' field contains 'com.domain.piklab'. Below these fields, the 'Version' field contains '1.0' and the 'Build' field contains '1'.

Display Name	PikLab
Bundle Identifier	com.domain.piklab
Version	1.0
Build	1

**NOTE:** do not change the **PikLab** folder's name or Project's name, otherwise XCode won't recognize the app and you won't be able to test it on the iOS Simulator nor Archive it for AppStore publishing.

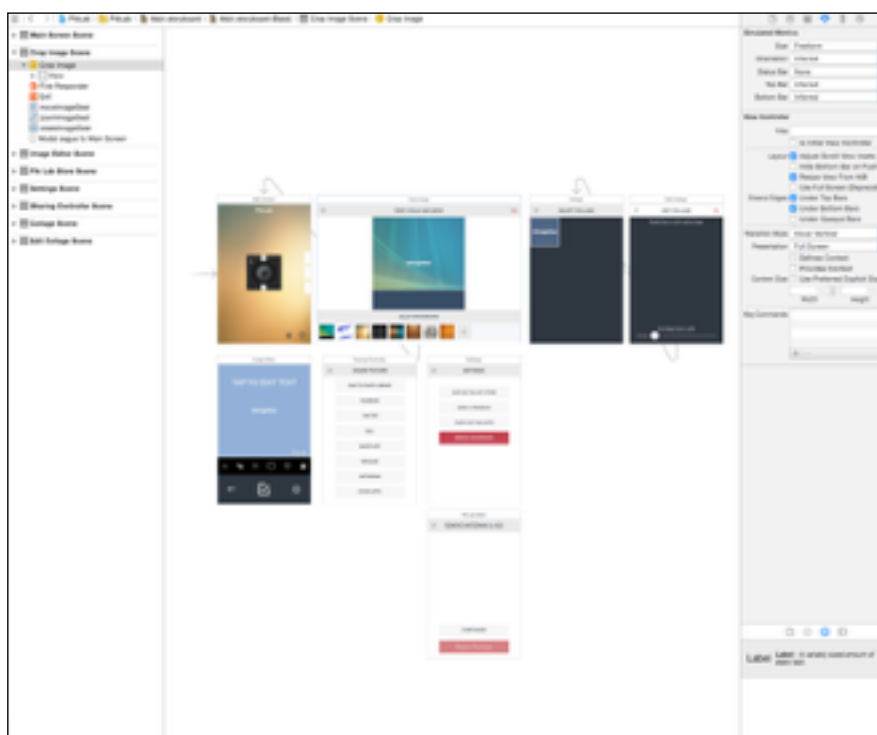
## Archive the App (for submission on the App Store)

In order for you to archive the app and be able to submit it for Review on the iTunes App Store, select **Generic iOS Device** on the device list, then click on **Product -> Archive** to archive the app and launch the **Organizer** once the archiving process will be done (see below):



## 2. CUSTOMIZATION

It's very easy to customize the UI elements in the XCode project, just check Main.storyboard file, select the views on the Controllers and edit them via the right-side Inspector panel. **Images.xcassets** folder contains all the 1x, 2x and 3x graphics, well organised by folders.



## Adding extra Background images for Croplmage class.

You can simply add new buttons that will load background images into Croplmage class by Storyboard by following these steps:

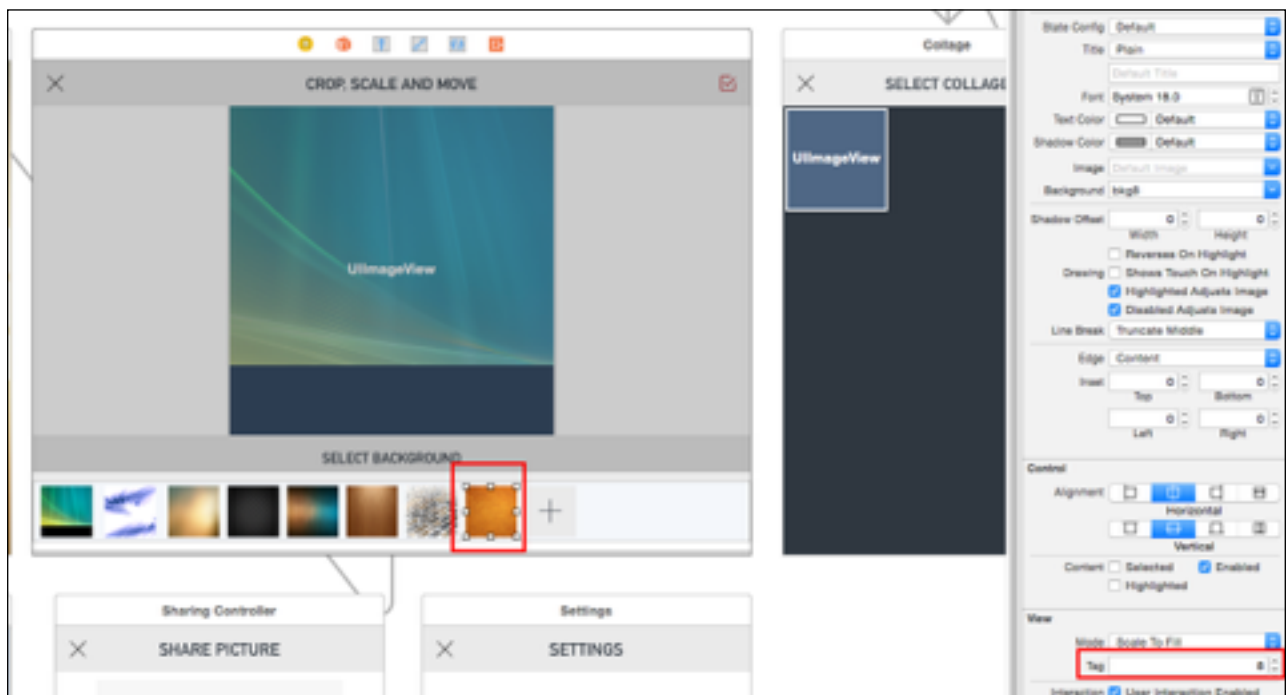
1. Enlarge the **Croplmage Controller** by using the **Size Inspector**



in order to get more space into the bkgScrollView to manually add new bkg buttons.

2. Move the “+” button on the right in order to leave some room for the new button you will add.
3. Hold **alt-mouse button** over the button with **tag 8** and drag it on the right side of it, before the .
4. Change its tag to **9** and assign a new background image to it

Done, the app will recognize the new button and its tag and load its background image when tapped. Please note that if you'll add other bkg buttons, you will have to progressively assign them a tag number (so a new one will be 10, then 11, 12, etc.).



## Adding extra Fonts for the Text Editor.

Follow these steps to add extra custom fonts top **PikLab**:

1. Download your **.ttf** font file from the web and get their PostScript names (we suggest you to download fonts from <http://www.fonts2u.com> since it has PostScript names of all its fonts)

2. Add it to the **fontList** array into **Configs.swift** file.
3. Drag your font file into **Fonts** folder in XCode (you can find it into **Supporting Files** folder).
4. Go into **Info** tab in XCode and expand the **Fonts provided by application** row. Click on Item's + sign to add the complete name (in this example it'll be **Lobster\_1\_4.otf**).  
You're done, now **PikLab** will automatically load this new font into its proper TableView.

General	Capabilities	Info	Build Settings	Build Phases	Build Rules
<b>Properties</b>					
Key	Type	Value			
NSLocationWhenInUseUsageDescription	String				
Bundle identifier	String	com.domain.piklab			
InfoDictionary version	String	6.0			
Main storyboard file base name	String	Main			
Bundle version	String	1			
NSLocationAlwaysUsageDescription	String				
Executable file	String	\$(EXECUTABLE_NAME)			
Application requires iPhone environment	Boolean	YES			
▼ Fonts provided by application	Array	(7 items)			
Item 0	String	CFJacquesParizeau-Regular.ttf			
Item 1	String	HussarGothic.otf			
Item 2	String	Springtime_Blues.ttf			
Item 3	String	LemonMilk.otf			
Item 4	String	BebasNeue.otf			
Item 5	String	Comfortaa_Regular.ttf			
Item 6	String	Lobster_1_4.otf			
► Supported interface orientations	Array	(1 item)			
► Required device capabilities	Array	(1 item)			
Bundle versions string, short	String	1.0			
Bundle creator OS Type code	String	????			
Bundle OS Type code	String	APPL			
Status bar is initially hidden	Boolean	YES			
Localization native development region	String	en			
Bundle name	String	\$(PRODUCT_NAME)			

Please note that not all **.ttf** or **.otf** fonts found on the web are compatible with iOS, so in case you'll get some error from XCode, just delete that font, get a new one and test it to make sure it's compatible. Fortunately the 95% of fonts are compatible with iOS, so issues are just a few, probably you'll never get one.

## Adding custom colors.

You can edit/add your own UIColors by just editing the colorList array into **Configs.swift** file:

```
// ARRAY OF COLORS ( You can edit, add or remove custom Colors here)
let colorList = [
    UIColor.whiteColor(),
    UIColor.blackColor(),
    UIColor(red: 204.0/255.0, green: 208.0/255.0, blue: 217.0/255.0, alpha: 1.0),
    UIColor(red: 101.0/255.0, green: 109.0/255.0, blue: 120.0/255.0, alpha: 1.0),
    UIColor(red: 236.0/255.0, green: 136.0/255.0, blue: 192.0/255.0, alpha: 1.0),
    UIColor(red: 172.0/255.0, green: 146.0/255.0, blue: 237.0/255.0, alpha: 1.0),
    UIColor(red: 93.0/255.0, green: 155.0/255.0, blue: 236.0/255.0, alpha: 1.0),
    UIColor(red: 79.0/255.0, green: 192.0/255.0, blue: 232.0/255.0, alpha: 1.0),
    UIColor(red: 72.0/255.0, green: 207.0/255.0, blue: 174.0/255.0, alpha: 1.0),
    UIColor(red: 160.0/255.0, green: 212.0/255.0, blue: 104.0/255.0, alpha: 1.0),
    UIColor(red: 253.0/255.0, green: 207.0/255.0, blue: 85.0/255.0, alpha: 1.0),
    UIColor(red: 251.0/255.0, green: 110.0/255.0, blue: 82.0/255.0, alpha: 1.0),
    UIColor(red: 237.0/255.0, green: 85.0/255.0, blue: 100.0/255.0, alpha: 1.0),

    // You can add custom UIColors here...
]
```

All colors have RGBA values (Red, Green, Blue & Alpha), so you can make new colors by getting RGB values from Photoshop color picker, for example, and add/edit the lines of code above as you wish. Those colors are the ones that appear into Text and Borders Tools.

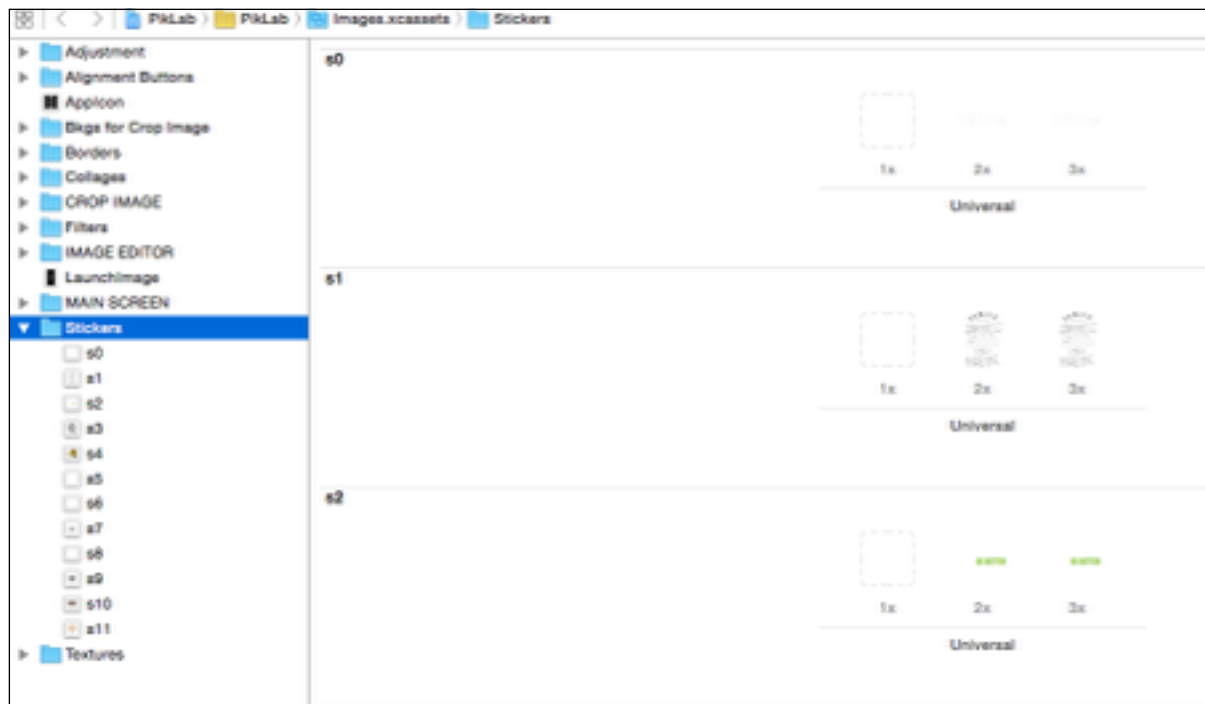
## Adding Stickers, Textures & Borders.

Follow these simple steps to add your own stickers to PikLab:

1. Design your **.png** stickers with a size of **512x512px** and transparent background.
2. Add them into **Images.xcassets/Stickers** folder, make sure you place them into 3x and 2x boxes (see below):
3. Go into **ImageEditor.swift** class and look for **setupStickersTool()** method, you have to edit the following line of code:

```
// Loop for creating buttons
for itemCount = 0; itemCount < 11+1; ++itemCount {
```

PikLab has 11 stickers (plus one that is blank), so you have to just change the **11** value into the total amount of stickers you have (excluding the blank one, which is **s0** image in Images.xcassets), and you'll be done, PikLab will load all your stickers.



**NOTE:** For **Textures** and **Borders**, repeat the steps above, of course you'll have to add first your own textures and borders .png images into their relative folders in **Images.xcassets**, then look for the following methods into **ImageEditor.swift** file:

***setupTexturesTool()***  
***setupBordersTool()***

Change their values that are before **+1** (like you did for Stickers) and you'll be done.

## Adding Filters.

PikLab works with Apple **CIFilters**, you can find a reference about them here: <https://developer.apple.com/library/mac/documentation/GraphicsImaging/Reference/CoreImageFilterReference/Reference/reference.html>

If you want to add some extra filter, you can copy some of the existing ones and just edit them, here's a step-by-step example using the customizable **CIColorMonochrome** Core Image filter from iOS:

1. Here are the 2 arrays that store all the PikLab's filters available with their CI names and the relative custom names we gave them:

```

// Filters List & Names -----
let filtersList = [
    "None",           //0
    "CIVignette",     //1
    "CIPhotoEffectInstant", //2
    "CIPhotoEffectProcess", //3
    "CIPhotoEffectTransfer", //4
    "CISepiaTone",     //5
    "CIPhotoEffectChrome", //6
    "CIPhotoEffectFade", //7
    "CIPhotoEffectTonal", //8
    "CIPhotoEffectNoir", //9
    "CIMaximumComponent", //10
    "CIMinimumComponent", //11
    "CIDotScreen",        //12
    "CIColorMonochrome",  //13
    "CIColorMonochrome",  //14
    "CIColorMonochrome",  //15
    "CIColorPosterize",    //16
    "CISharpenLuminance",  //17
    "CIGammaAdjust",       //18
    "CIHueAdjust",         //19
    "CIHueAdjust",         //20
    "CIHueAdjust",         //21
    "CISepiaTone",         //22

    // Add here new CIFilters...
]

let filterNamesList = [
    "None",           //0
    "Vignette",       //1
    "Imagine",        //2
    "Retro",          //3
    "Chic",           //4
    "Sepia",          //5
    "Light",          //6
    "Aqua",           //7
    "Tonal",          //8
    "Noir",           //9
    "Darken",         //10
    "Rude",           //11
    "Dotted",         //12
    "Orange",         //13
    "Reddy",          //14
    "Canary",         //15
    "Poster",         //16
    "Sharp",          //17
    "Boris",          //18
    "Marylin",        //19
    "Fantik",         //20
    "Ipse",           //21
    "Vintiq",         //22

    //Set new Filter Names here...
]
//-----

```

2. You can now copy:

```
"CIColorMonochrome",
```

from **filtersList** array and paste it right below "*CiSepiaTone*", so it should look like this:

```
"CiSepiaTone",          //22  
"CIColorMonochrome", //23
```

3. Create now a new custom name for it, we'll use **Brown** for example, so add it right below

```
"Vintiq", //22
```

into **filterNamesList** array, and it should look like this:

```
"Vintiq", //22  
"Brown", //23
```

So Brown filter will be our **23th** new filter, based on CIColorMonochrome.

4. Now copy the whole **case 15** (see below):

```
case 15:  
    var color:UIColor = UIColor(red: 241/255.0, green: 247/255.0, blue:  
71/255.0, alpha: 1.0)  
    filter.setValue(CIColor(color: color), forKey: kCIInputColorKey)  
    filter.setValue(0.8, forKey: kCIInputIntensityKey)  
    break
```

And paste it right below the case 22, it should look like this:

```
case 22: //Vintage  
    filter.setValue(0.5, forKey:"inputIntensity")  
    break  
  
case 23: // Canary  
    var color:UIColor = UIColor(red: 241/255.0, green: 247/255.0, blue:  
71/255.0, alpha: 1.0)  
    filter.setValue(CIColor(color: color), forKey: kCIInputColorKey)  
    filter.setValue(0.8, forKey: kCIInputIntensityKey)  
    break
```



5. You can now edit the RGBA and intensity values and create your Brown filter. The brown color **RGB** values are 171-113-39, and let's set its intensity as 0.9. So here's the complete new **case 23**:

```
case 23: // Brown
    var color:UIColor = UIColor(red: 171/255.0, green: 113/255.0, blue:
39/255.0, alpha: 1.0)
    filter.setValue(CIColor(color: color), forKey: kCIInputColorKey)
    filter.setValue(0.8, forKey: kCIInputIntensityKey)
    break
```

Your new Brown filter is now ready to be used!

## Rate App on the AppStore feature.

PikLab Settings Controller has a button that points to your app's page in iTunes Store, so you just have to head over **Configs.swift** file again and replace the existing App ID with your own one:

```
// Replace the red string below with your App ID (you can grab it by
clicking on More -> About This App in your iTunes Connect page). This
is needed for the Rate-App feature
let APP_ID = "957290825"
```

Its green comment explains what to do, like all the other comments in **Configs.swift** file.

## Sharing messages.

Still into Configs.swift file, you can change the red strings below as you wish and they'll appear on almost all the sharing features available such as Facebook, Twitter, Mail and Message.

```
// You can edit the sharing message and subject here:
let subjectText = "My Picture"
let messageText = "Hi there, check this pic out, I've made it with \
(APP_NAME)"
```

## AdMob Interstitial Unit ID

Configs.swift file stores an AdMob ID for showing a test Interstitial ad, so you'll have to just create an Interstitial Unit ID on [www.apps.admob.com](http://www.apps.admob.com) and use it to replace the existing one:

```
// Replace the red string below with the Interstitial Unit ID you will  
create on http://apps.admob.com  
let ADMOB_UNIT_ID = "ca-app-pub-9733347540588953/7805958028"
```

Once you'll Submit your app for Review, you'll have to check the **IDFA** Options as follows:

### Advertising Identifier

Does this app use the Advertising Identifier (IDFA)? ☒ Yes ☐ No

The **Advertising Identifier (IDFA)** is a unique ID for each iOS device and is the only way to offer targeted ads. Users can choose to limit ad targeting on their iOS device.

If your app is using the Advertising Identifier, check your code—including any third-party code—before you submit it to make sure that your app uses the Advertising Identifier only for the purposes listed below and respects the Limit Ad Tracking setting. If you include third-party code in your app, you are responsible for the behavior of such code, so be sure to check with your third-party provider to confirm compliance with the usage limitations of the Advertising Identifier and the Limit Ad Tracking setting.

This app uses the Advertising Identifier to (select all that apply):

- ☒ Serve advertisements within the app
- ☐ Attribute this app installation to a previously served advertisement
- ☐ Attribute this app installation to a previously served advertisement

If you think you have another acceptable use for the Advertising Identifier, [contact us](#).

Limit Ad Tracking setting in iOS

- ☒ I, Anne Johnson, confirm that this app, and any third party that interfaces with this app, uses the Advertising Identifier checks and honors a user's Limit Ad Tracking setting in iOS and, when it is enabled by a user, this app does not use Advertising Identifier, and any information obtained through the use of the Advertising Identifier, in any way other than for "Limited Advertising Purposes" as defined in the [iOS Developer Program License Agreement](#).

**Good luck with PikLab, and don't forget to rate it on your Downloads page on CodeCanyon!**



<http://fvimagination.com>