

Documentación implementación de SDK en iOS nativo usando SwiftUI

A. Implementación mediante Podfile

1. Flutter Module

Primero descargamos el repositorio del proyecto [laraigo_module_flutter](#), y lo ubicamos al mismo nivel del proyecto ios sobre el cual trabajaremos.

Dentro del proyecto ***laraigo_module_flutter***, en la terminal de nuestro editor ejecutaremos los siguientes comandos.

- a. [flutter clean && flutter pub get](#)
- b. [cd .ios/](#) -> Este comando nos permitirá ingresar al directorio .ios.
- c. [pod install](#) -> Instalaremos todas las dependencias necesarias que consumiremos en nuestro proyecto iOS.

2. iOS Project

Dentro de la carpeta del proyecto comprobaremos si contamos con la carpeta **Podfile**, de no ser el caso ejecutaremos el siguiente comando en consola al nivel de dicho proyecto [pod init](#).

Después de esto, abriremos el archivo PodFile que se generó o en caso de ya tenerlo para su modificación, agregando el siguiente contenido.

```
flutter_application_path = '../laraigo_module_flutter'
load File.join(flutter_application_path, '.ios', 'Flutter', 'podhelper.rb')

target 'Nombre_de_nuestro_proyecto_ios' do
  # Comment the next line if you don't want to use dynamic frameworks
  use_frameworks!

  # Pods for LaraigoSdk
  install_all_flutter_pods(flutter_application_path)
end

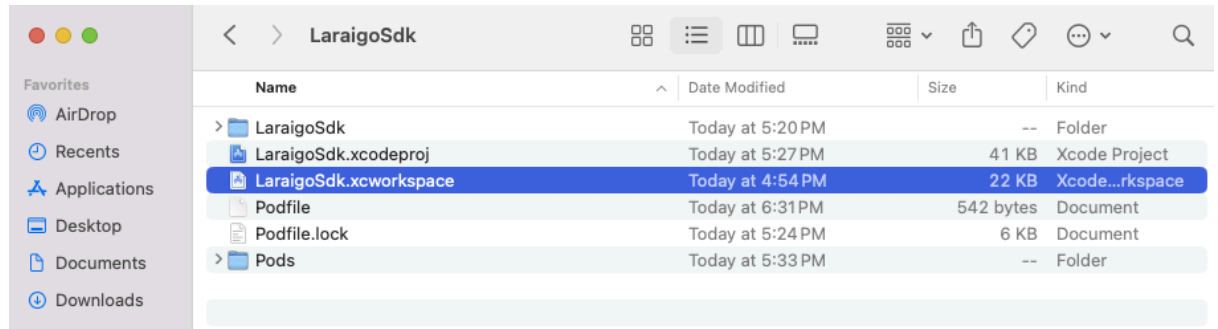
post_install do |installer|
  flutter_post_install(installer) if defined?(flutter_post_install)
end
```

Finalmente volveremos a nuestra consola al mismo nivel donde creamos nuestro Podfile, es decir, dentro del proyecto ios para ejecutar el comando [pod install](#), esto instalará todas las dependencias de nuestro SDK en el proyecto iOS donde se ejecutó.

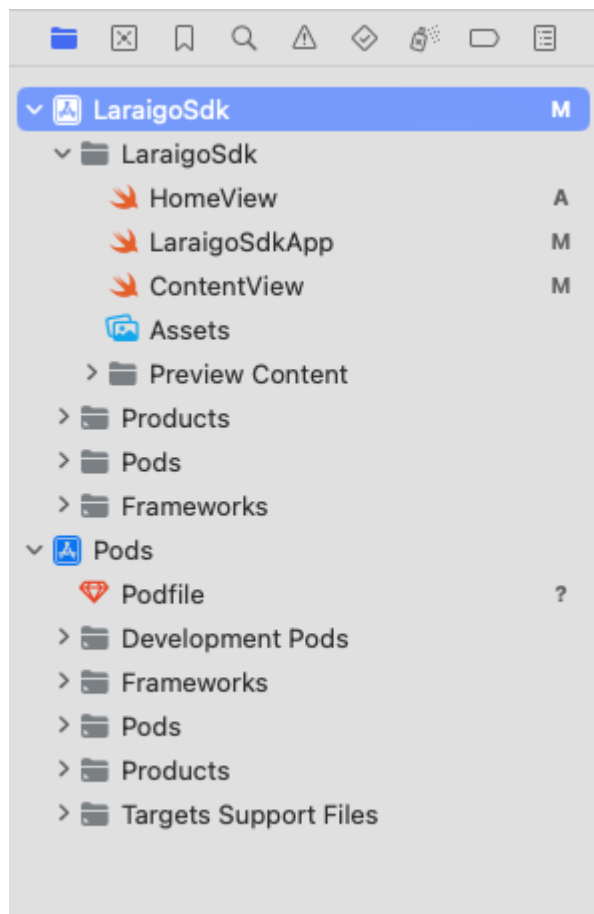
3. iOS Project - *iOSProject.xcworkspace*

Una vez instaladas todas las dependencias dentro del proyecto, iremos al folder del proyecto, como se muestra en la imagen adjunta para ejecutar el archivo seleccionado.

Pasaremos de usar el archivo [.xcodeproj](#) a abrirlo desde [.xcworkspace](#).



Dentro nuestro [.xcworkspace](#) encontraremos la siguiente estructura de archivos:

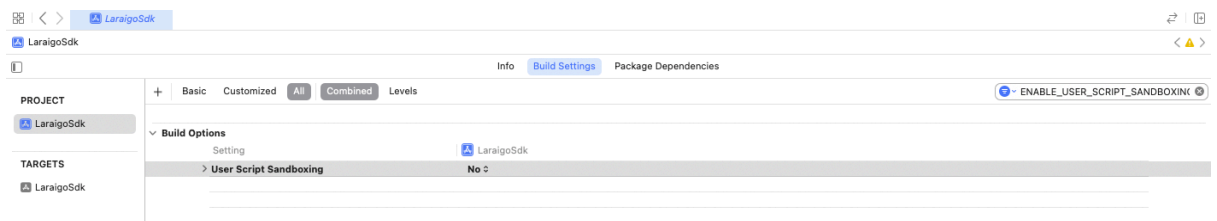


NOTA: El siguiente paso se deberá tomar en cuenta si al momento de compilación salta el siguiente error:

*Sandbox: rsync.samba(94055) deny(1) file-write-create
/Users/user_name/Library/Developer/Xcode/DerivedData/LaraigoSdk-fivnqzjv
oyoqukdroyutbjmprusj/Build/Products/Debug-iphones/Flutter.framework*

Seleccionamos el nombre de nuestro proyecto, en el caso de la imagen referencial es LaraigoSdk, para deshabilitar la opción **ENABLE_USER_SCRIPT_SANDBOXING**: **No**.

Tal cual se muestra en la imagen.



4. Swift Files en iOS Project

Por defecto tendremos los archivos: [ContentView](#) y [LaraigoSdkApp](#) (este nombre es el del proyecto).

Adicionalmente se agregó un archivo llamado [HomeView](#) netamente por motivos visuales de ejecución.

❖ ContentView

Se agrega el siguiente código, netamente para redireccionar la ejecución de nuestro SDK.

```
1 //
2 // ContentView.swift
3 // LaraigoSdk
4 //
5 //
6
7 import SwiftUI
8 import Flutter
9
10 struct ContentView: View {
11     var body: some View {
12         VStack {
13             NavigationView(content: {
14                 NavigationLink(destination: {HomeView()}) { Text("Init App") }
15             })
16         }
17     }
18 }
19
20 #Preview {
21     ContentView()
22 }
```

❖ LaraigoSdkApp

Dentro de nuestro archivo principal tendremos que agregar el siguiente código, este es importante para su ejecución.

```

1 //
2 //  LaraigoSdkApp.swift
3 //  LaraigoSdk
4 //
5 //
6
7 import SwiftUI
8 import Flutter
9 import FlutterPluginRegistrant
10
11 class FlutterDependencies: ObservableObject {
12     let flutterEngine = FlutterEngine(name: "flutter-engine")
13     init() {
14         flutterEngine.run()
15         GeneratedPluginRegistrant.register(with: self.flutterEngine)
16     }
17 }
18
19 @main
20 struct LaraigoSwiftApp: App {
21     @StateObject var flutterDependencies = FlutterDependencies()
22     var body: some Scene {
23         WindowGroup {
24             ContentView().environmentObject(flutterDependencies)
25         }
26     }
27 }
```

❖ HomeView

Finalmente en este archivo se encuentra nuestro código de ejecución en SwiftUI.

Este código necesita que se implementen principalmente las siguientes líneas.

Las variables de entorno para acceder a las dependencias flutter.

```
@EnvironmentObject var flutterDependencies: FlutterDependencies
@State var flutterBasicChannel : FlutterBasicMessageChannel?
```

Así mismo la implementación de la pantalla donde se renderiza.

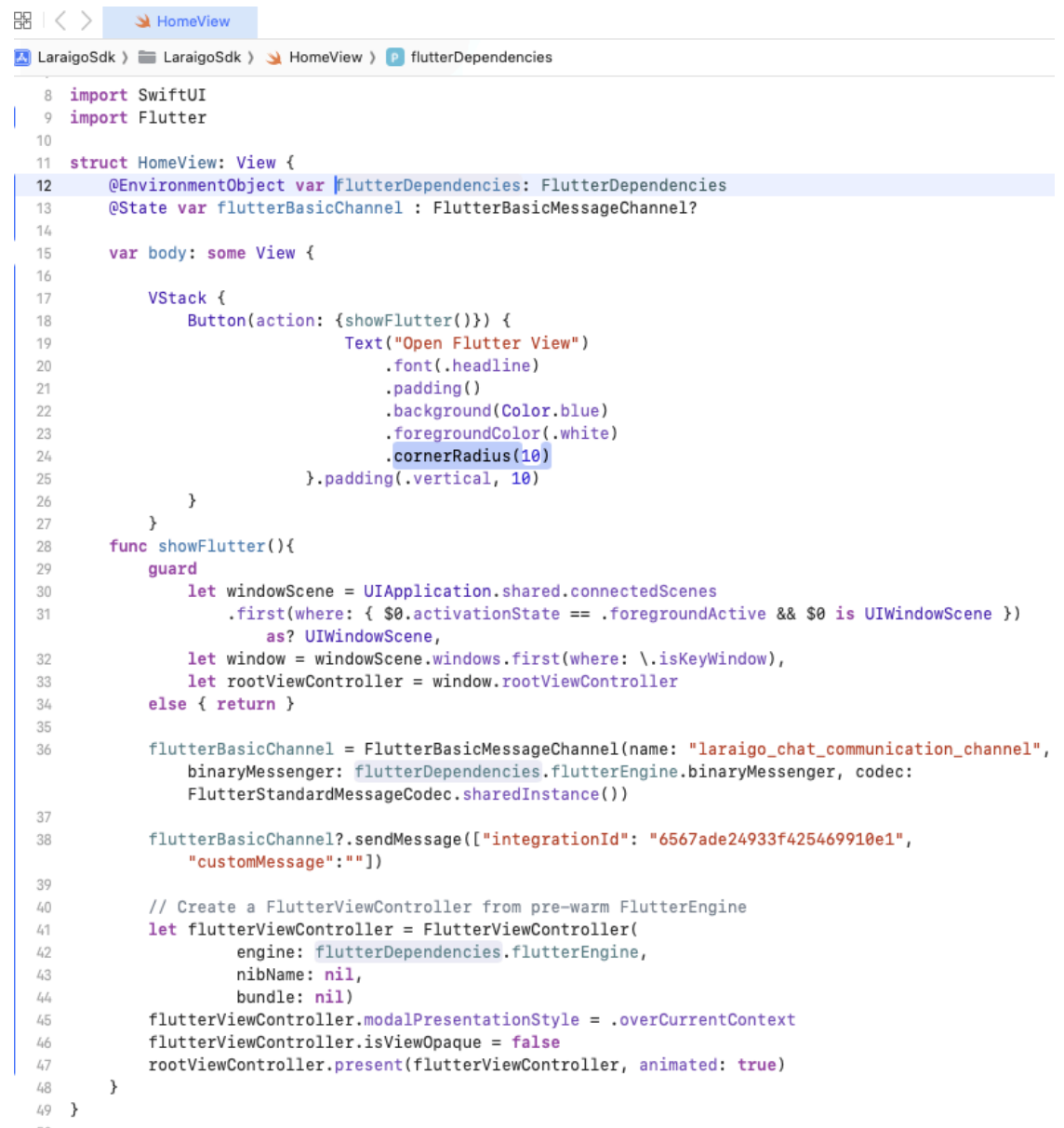
```
let windowScene = UIApplication.shared.connectedScenes
.first(where: { $0.activationState == .foregroundActive && $0 is UIWindowScene })
as? UIWindowScene,
let window = windowScene.windows.first(where: \.isKeyWindow),
let rootViewController = window.rootViewController
```

Finalmente las llamadas al SDK de Laraigo.

```
flutterBasicChannel = FlutterBasicMessageChannel(name:
"laraigo_chat_communication_channel", binaryMessenger:
flutterDependencies.flutterEngine.binaryMessenger, codec:
FlutterStandardMessageCodec.sharedInstance())

flutterBasicChannel?.sendMessage(["integrationId": "6567ade24933f425469910e1",
"customMessage":""])

// Create a FlutterViewController from pre-warm FlutterEngine
let flutterViewController = FlutterViewController(
    engine: flutterDependencies.flutterEngine,
    nibName: nil,
    bundle: nil)
flutterViewController.modalPresentationStyle = .overCurrentContext
flutterViewController.isViewOpaque = false
rootViewController.present(flutterViewController, animated: true)
```



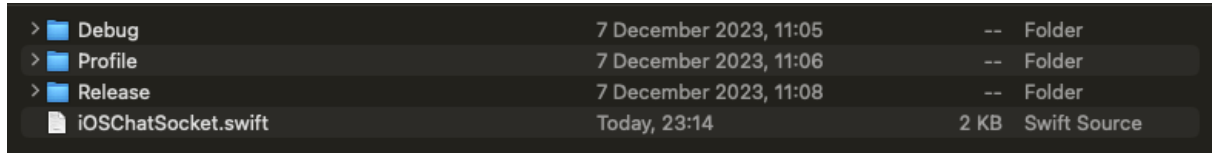
```

8 import SwiftUI
9 import Flutter
10
11 struct HomeView: View {
12     @EnvironmentObject var flutterDependencies: FlutterDependencies
13     @State var flutterBasicChannel : FlutterBasicMessageChannel?
14
15     var body: some View {
16
17         VStack {
18             Button(action: {showFlutter()}) {
19                 Text("Open Flutter View")
20                     .font(.headline)
21                     .padding()
22                     .background(Color.blue)
23                     .foregroundColor(.white)
24                     .cornerRadius(10)
25             }.padding(.vertical, 10)
26         }
27     }
28     func showFlutter(){
29         guard
30             let windowScene = UIApplication.shared.connectedScenes
31                 .first(where: { $0.activationState == .foregroundActive && $0 is UIWindowScene })
32                 as? UIWindowScene,
33             let window = windowScene.windows.first(where: \.isKeyWindow),
34             let rootViewController = window.rootViewController
35         else { return }
36
37         flutterBasicChannel = FlutterBasicMessageChannel(name: "laraigo_chat_communication_channel",
38             binaryMessenger: flutterDependencies.flutterEngine.binaryMessenger, codec:
39             FlutterStandardMessageCodec.sharedInstance())
40
41         flutterBasicChannel?.sendMessage(["integrationId": "6567ade24933f425469910e1",
42             "customMessage":""])
43
44         // Create a FlutterViewController from pre-warm FlutterEngine
45         let flutterViewController = FlutterViewController(
46             engine: flutterDependencies.flutterEngine,
47             nibName: nil,
48             bundle: nil)
49         flutterViewController.modalPresentationStyle = .overCurrentContext
50         flutterViewController.isViewOpaque = false
51         rootViewController.present(flutterViewController, animated: true)
52     }
53 }
```

- Estos son todos los pasos para ejecutar el proyecto, es preferible ejecutarlo en un dispositivo físico dado que en el emulador suele fallar.

B. Implementación mediante Files Dependencias SDK

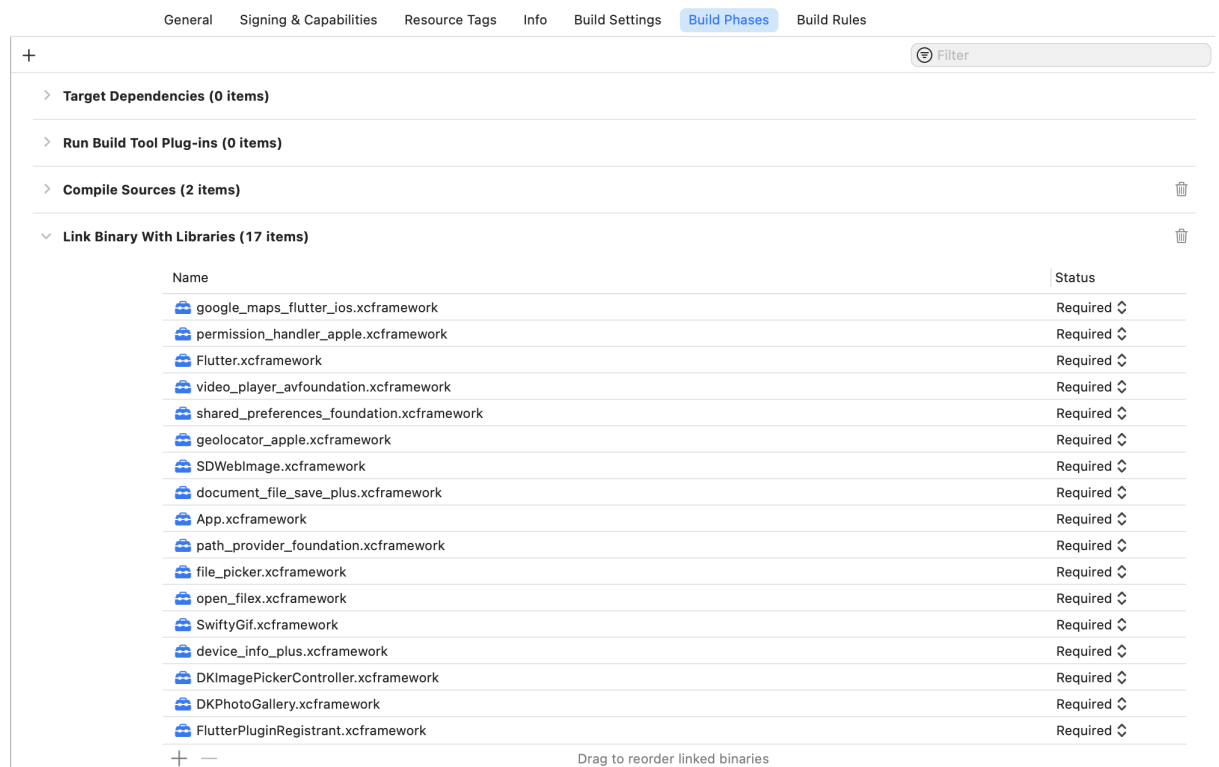
- Primero descargue el repositorio desde el repositorio de github [laraigo_chat_module_ios](#).
- Después de la descarga tendrás cuatro carpetas, un archivo iOS y también una carpeta de assets (ignorar), esta puede descargarse donde se requiera o guste.



- En segundo lugar, irás a la raíz de tu proyecto. En este caso usaré un proyecto vacío.

Vaya a **Runner > Build Phases** y agregue estas dependencias en **Build Phases > Link Binary With Libraries**

Arrastre y suelte los frameworks para la misma compilación como se muestra en la imagen a continuación en Vincular binario con bibliotecas.

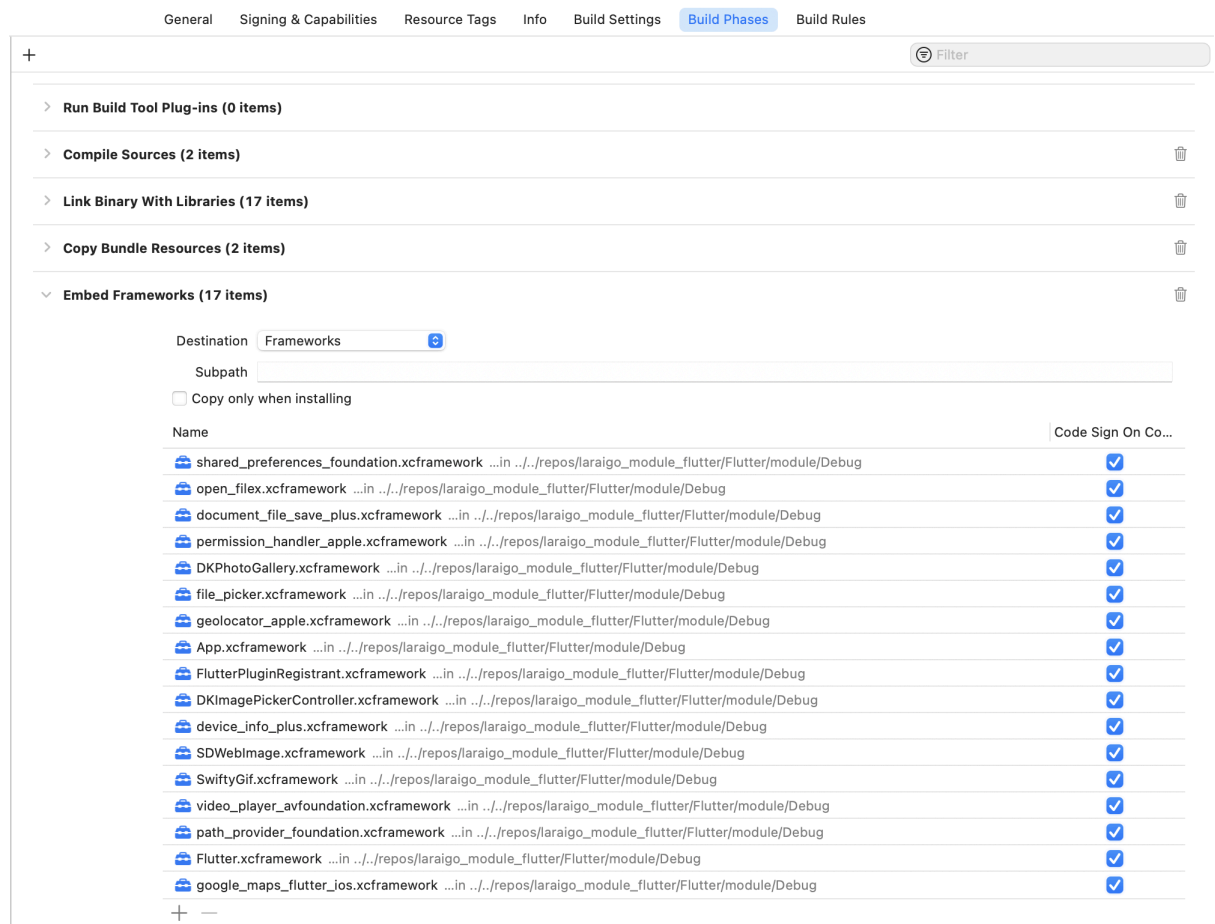


- Luego irás a la raíz de tu proyecto. En este caso usaré un proyecto vacío.

Vaya a **Runner > Build Phases** y agregue estas dependencias en **Build Phases > Embed Frameworks**

Arrastre y suelte los frameworks para la misma compilación como se muestra en la imagen a continuación en Embed Frameworks.

Si no tiene la ubicación de Embed Frameworks, genere una nueva fase de compilación con ese nombre en **Runner > Build Phases > + > New Copy Files Phase**.



- Finalmente seguir la guía desde el paso **número 4** de la primera **implementación A**.