

# Utilizando SQLite en Swift

iOS Dev Lab FCA  
Cristian Tafolla Rosales

# Base de datos

- Conjunto de datos pertenecientes a un mismo contexto y almacenados sistemáticamente para su uso posterior.



# Base de datos relacional

- Colección de elementos de datos organizados en un conjunto de tablas.
- Nos permite acceder o administrar los datos sin tener que reorganizar las tablas de dicha base.

Visor de datos - Employee ( Dibujo1.dwg )

The screenshot shows a data viewer window titled "Visor de datos - Employee ( Dibujo1.dwg )". The window contains a table with the following columns: Emp\_Id, Last\_Name, First\_Name, Gender, and Title. The data is as follows:

	Emp_Id	Last_Name	First_Name	Gender	Title
▶	1000	Torbati	Yolanda	F	Programmer
	1001	Kleinn	Joel	M	Programmer
	1002	Ginsburg	Laura	F	President
	1003	Cox	Jennifer	F	Programmer
	1005	Ziada	Mauri	M	Product Designer
	1006	Keyser	Cara	F	Account Executive
	1010	Smith	Roxie	M	Programmer
	1011	Nelson	Robert	M	Programmer
	1012	Sachsen	Lars	M	Support Technician
	1013	Shannon	Don	M	Product Designer

Registro 1

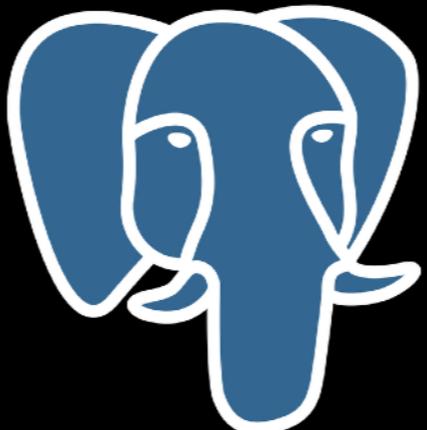
# Hay dos tipos de relaciones:

- Tablas
- De asociación o vinculación



# RDBMS

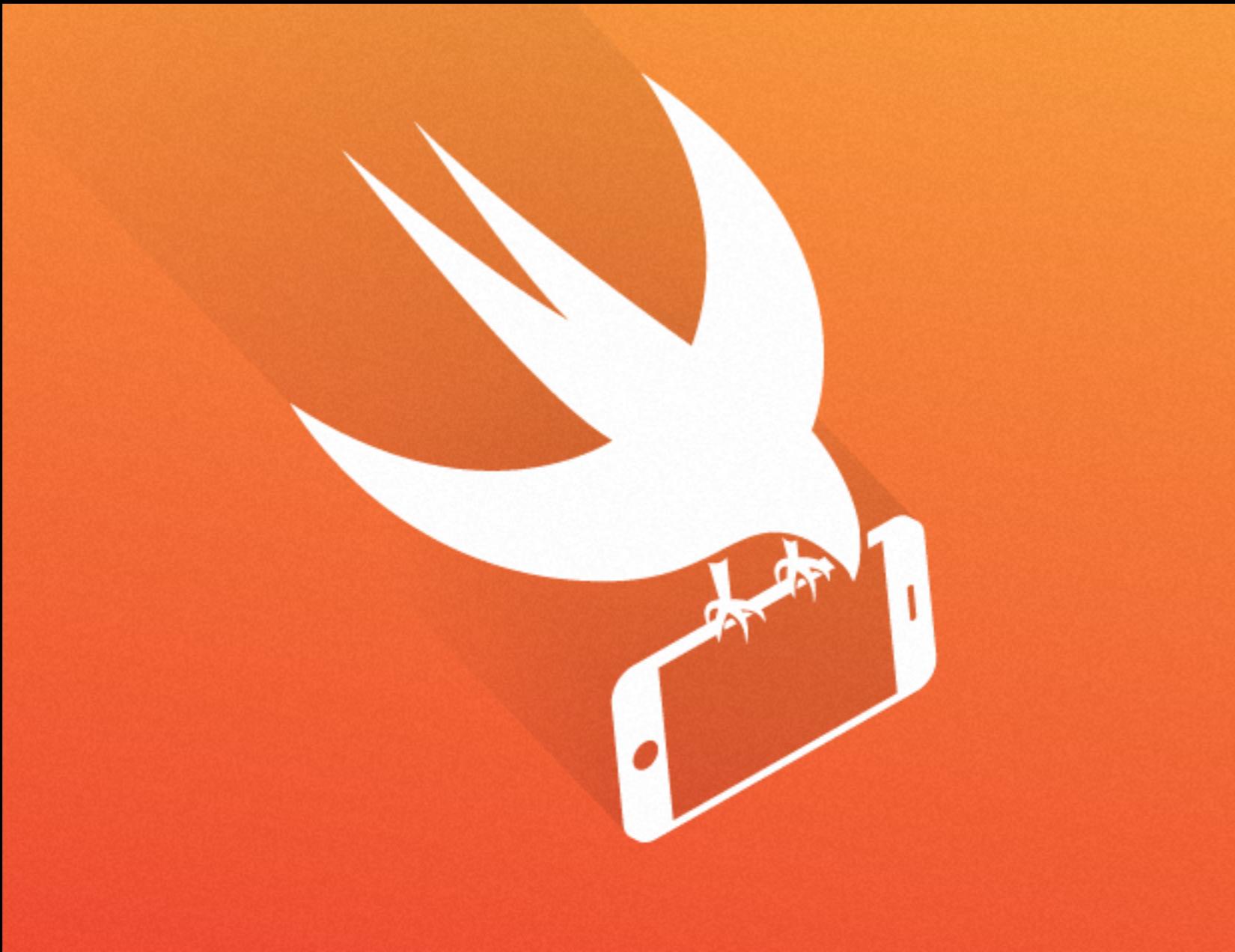
- Software que permite gestionar una base de datos relacional. Por lo general utilizan el Lenguaje de Consultas Estructuradas (SQL).



# SQLite

- Sistema de gestión de bases de datos relacionales (RDBMS), de dominio público.





Manos a la obra



Why GitHub? Enterprise Explore Marketplace Pricing

sqlite.swift



Sign in

Sign up

## Repositories

287

Code



Commits

1K

Issues

682

Marketplace

0

Topics

0

Wikis

27

Users

0

## Languages

Swift

217

Objective-C

23

C

9

Shell

3

C++

1

HTML

1



## Swift

Swift is a modern programming language focused on safety, performance, and expressivity.

[See topic](#)



## 287 repository results

Sort: Best match

### [stephencelis/SQLite.swift](#)



★ 6k

A type-safe, Swift-language layer over SQLite3.

swift

sqlite

MIT license Updated 8 days ago 3 issues need help

### [ryanfowler/SwiftData](#)



★ 518

Simple and Effective SQLite Handling in Swift

MIT license Updated on 18 May 2018

Acceder a Github y buscar “sqlite.swift”

 <b>yopovych</b>	Merge pull request #853 from hellohuanlin/huan_primary_key_with_4_col...	...	Latest commit 5be9fa9 10 days ago
 <a href="#">.github</a>	WS		2 years ago
 <a href="#">Documentation</a>	Updated version in README and Documentation		13 days ago
 <a href="#">SQLite.playground</a>	Connection#pluck can throw		3 years ago
 <a href="#">SQLite.xcodeproj</a>	Updated to Swift 5		18 days ago
 <a href="#">Sources</a>	Merge branch 'master' into huan_primary_key_with_4_columns		12 days ago
 <a href="#">Tests</a>	Merge branch 'master' into huan_primary_key_with_4_columns		12 days ago
 <a href="#">.cocoadocs.yml</a>	Add config file for CocoaDocs		3 years ago
 <a href="#">.gitignore</a>	Absolute path		2 years ago
 <a href="#">.swift-version</a>	Swift 4.2, Xcode 10.1		5 months ago
 <a href="#">.travis.yml</a>	Updated to Swift 5		18 days ago
 <a href="#">CHANGELOG.md</a>	Swift 4.2, Xcode 10.1		5 months ago
 <a href="#">CONTRIBUTING.md</a>	Documentation		15 days ago
 <a href="#">LICENSE.txt</a>	2015: The year of header cleanup		4 years ago
 <a href="#">Makefile</a>	Updated iOS simulator to iPhone XS		18 days ago
 <a href="#">Package.swift</a>	Changed language versions array to contain Ints to try and get CI wor...		15 days ago
 <a href="#">README.md</a>	Updated version in README and Documentation		13 days ago
 <a href="#">SQLite.swift.podspec</a>	Fixed version		13 days ago

Buscar el archivo “SQLite.xcodeproj” y verificar si concuerda con la versión de Swift que utilizamos

Nota: Nosotros estamos  
utilizando la versión 4.2 de  
Swift, así que tendremos que  
hacer un poco más de trabajo

The screenshot shows a GitHub repository page for 'stephencelis / SQLite.swift'. At the top, there's a navigation bar with links like 'Why GitHub?', 'Enterprise', 'Explore', 'Marketplace', and 'Pricing'. On the right side of the header are 'Search', 'Sign in', and 'Sign up' buttons. Below the header, the repository name 'stephencelis / SQLite.swift' is displayed, along with statistics: 'Watch 211', 'Star 6,025', and 'Fork 1,046'. A navigation bar below the repository name includes tabs for 'Code' (which is selected), 'Issues 177', 'Pull requests 26', 'Projects 0', 'Wiki', and 'Insights'. In the center, there's a 'Branch: master' dropdown set to 'SQLite.swift / SQLite.xcodeproj /'. To the right of the branch dropdown are buttons for 'Create new file', 'Find file', and 'History', with 'History' being the active tab and highlighted with a red box. The main content area displays a list of recent commits:

Commit	Message	Date
	sburlewapg Updated to Swift 5	Latest commit 9ae1a0b 18 days ago
..		
	project.xcworkspace Prep for public release to CocoaPods	3 years ago
	xcshareddata/xcschemes Updated project to Swift 5, fixed build warnings	a month ago
	project.pbxproj Updated to Swift 5	18 days ago

At the bottom of the page, there are links for 'Contact GitHub', 'Pricing', 'API', 'Training', 'Blog', and 'About', along with the GitHub logo.

Dentro del archivo “SQLite.xcodeproj” iremos a su historial

- Reverted to swift 4.2, will go to 5 in another version update ...  
sburlewapg committed on 4 Apr ✘

Commits on Apr 3, 2019

Swift 5 ...  
sburlewapg committed on 3 Apr ✘

Commits on Mar 30, 2019

Updated project to Swift 5, fixed build warnings  
sburlewapg committed on 30 Mar

Commits on Dec 19, 2018

Swift 4.2, Xcode 10.1 ...  
jberkel committed on 19 Dec 2018 ✓ Verified cf4183f

Commits on Apr 13, 2018

Exclude test on iOS/tvOS 9.x  
jberkel committed on 13 Apr 2018 ✓

Commits on Apr 12, 2018

Fix Swift 4.1 warnings  
jberkel committed on 12 Apr 2018 ✘

Iremos al commit correspondiente a la versión 4.2 de Swift

Why GitHub? Enterprise Explore Marketplace Pricing

Search Sign in Sign up

stephencelis / SQLite.swift Watch 211 Star 6,025 Fork 1,046

Code Issues 177 Pull requests 26 Projects 0 Wiki Insights

**Swift 4.2, Xcode 10.1**

- fix tests running against SQLCipher 4.x

master (#866) 0.12.0 0.11.6

jberkel committed on 19 Dec 2018 Verified 1 parent d13baed commit cf4183f4ef54c5c9e82176cc46451934c8fdec2a

Showing 16 changed files with 83 additions and 73 deletions.

Unified Split

.swift-version

...	...	@@ -1 +1 @@
1	- 4.1	+ 4.2

.travis.yml

...	...	@@ -1,10 +1,11 @@
1	1	language: objective-c

Browse files

Exploraremos los archivos correspondientes a este commit

A type-safe, Swift-language layer over SQLite3.

swift sqlite

807 commits

3 branches

13 releases

54 contributors

MIT

Tree: cf4183f4ef ▾

New pull request

Find File

Clone or download ▾

jberkel Swift 4.2, Xcode 10.1 ...

.github

WS

Documentation

fix markdown table formatting

SQLite.playground

Connection#pluck can throw

SQLite.xcodeproj

Swift 4.2, Xcode 10.1

Sources

Swift 4.2, Xcode 10.1

5 months ago

Tests

Swift 4.2, Xcode 10.1

5 months ago

.cocoadocs.yml

Add config file for CocoaDocs

3 years ago

.gitignore

Absolute path

2 years ago

#### Clone with HTTPS ?

Use Git or checkout with SVN using the web URL.

<https://github.com/stephencelis/SQLite>

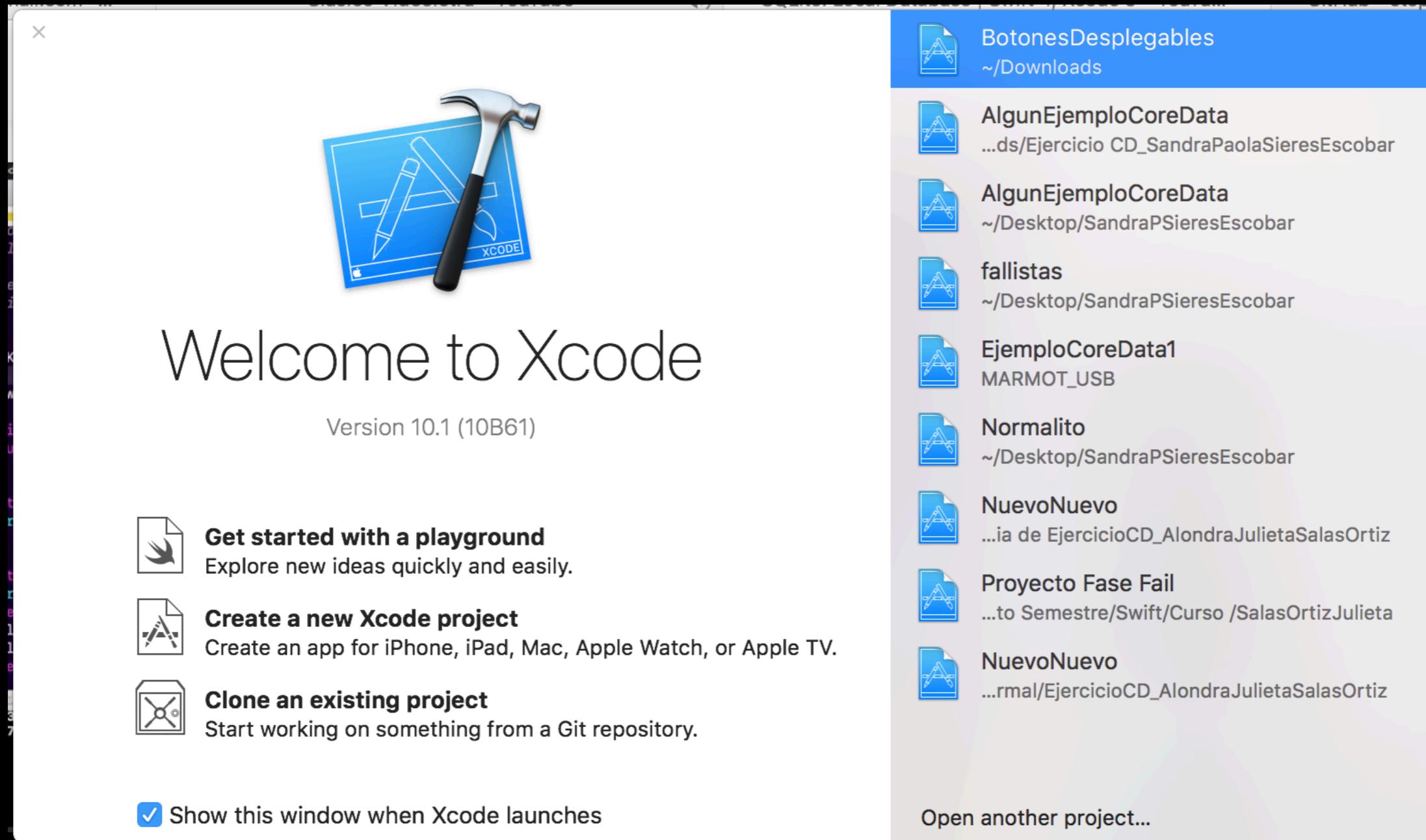


[Open in Desktop](#)

[Download ZIP](#)

5 months ago

# Descargaremos el archivo ZIP



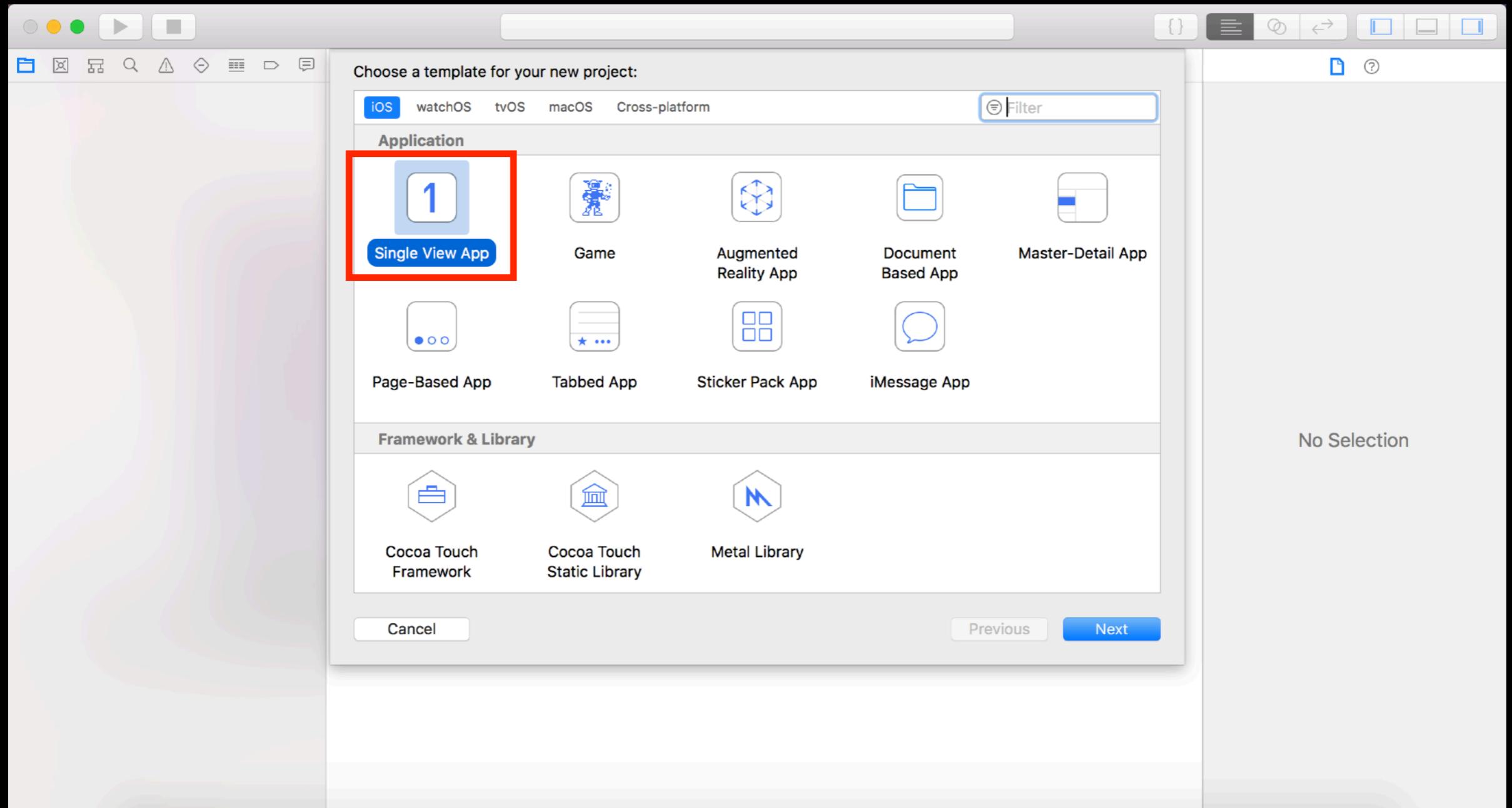
The image shows the Xcode 10.1 welcome screen. On the left, there's a large icon of a hammer and a pencil on a blueprint with the word "XCODE" at the bottom. Below it, the text "Welcome to Xcode" is displayed, followed by "Version 10.1 (10B61)". To the right, there are three main sections: a list of project templates, a checkbox for launching the window again, and a link to open another project.

Show this window when Xcode launches

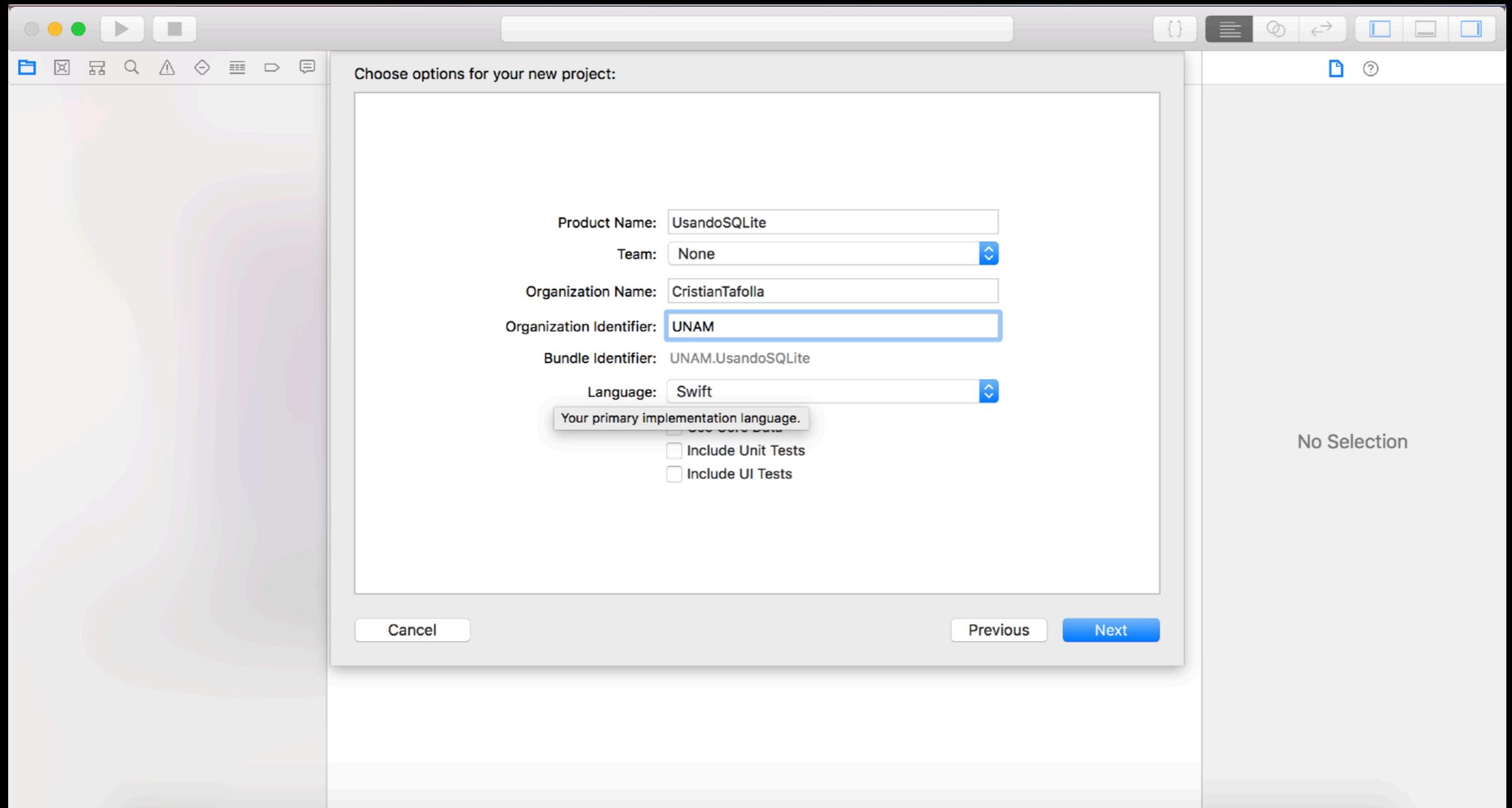
[Open another project...](#)

Project Name	Location
BotonesDesplegables	~/Downloads
AlgunEjemploCoreData	...ds/Ejercicio CD_SandraPaolaSieresEscobar
AlgunEjemploCoreData	~/Desktop/SandraPSieresEscobar
fallistas	~/Desktop/SandraPSieresEscobar
EjemploCoreData1	MARMOT_USB
Normalito	~/Desktop/SandraPSieresEscobar
NuevoNuevo	...ia de EjercicioCD_AlondraJulietaSalasOrtiz
Proyecto Fase Fail	...to Semestre/Swift/Curso /SalasOrtizJulieta
NuevoNuevo	...rmal/EjercicioCD_AlondraJulietaSalasOrtiz

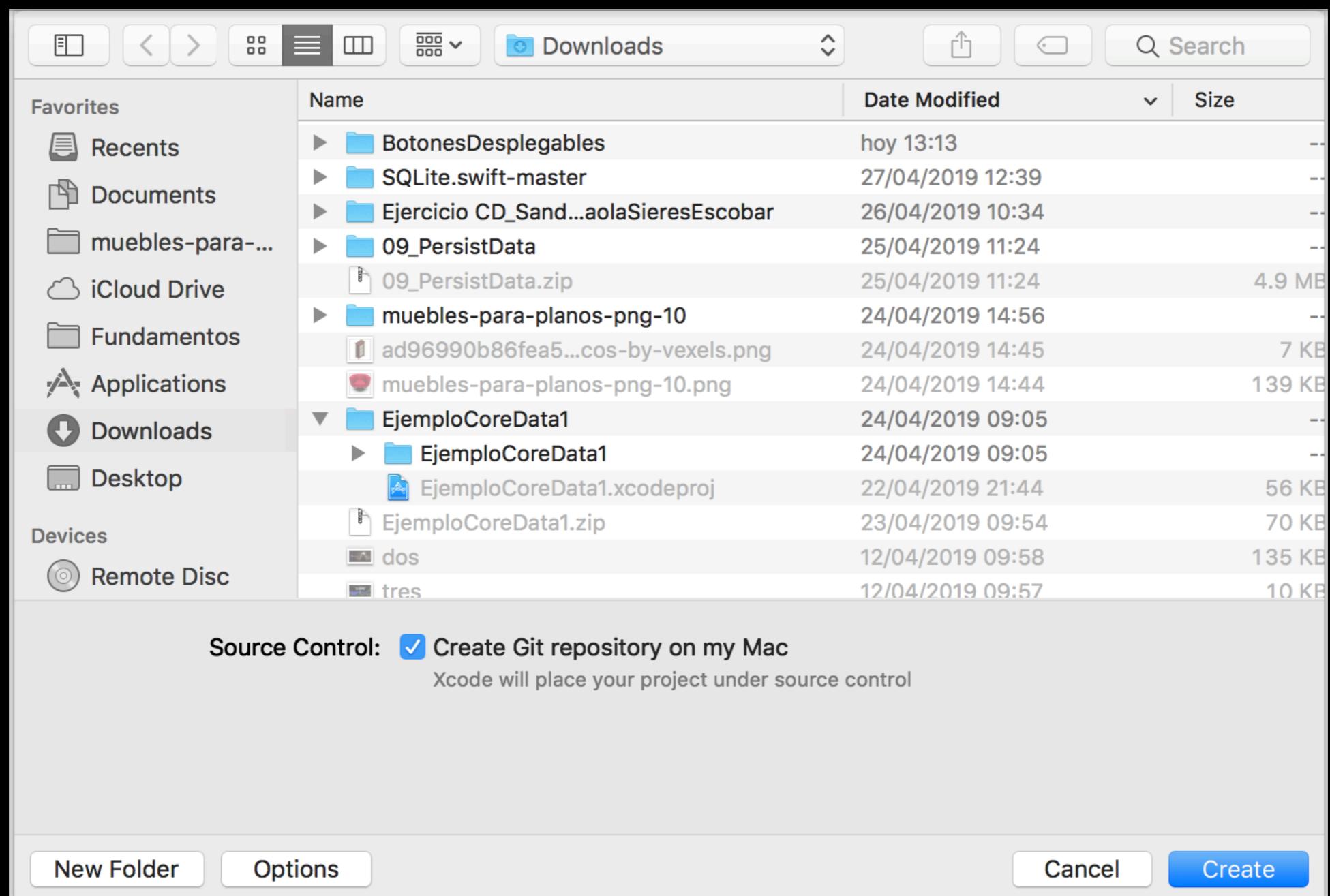
Crearemos un nuevo proyecto en Xcode



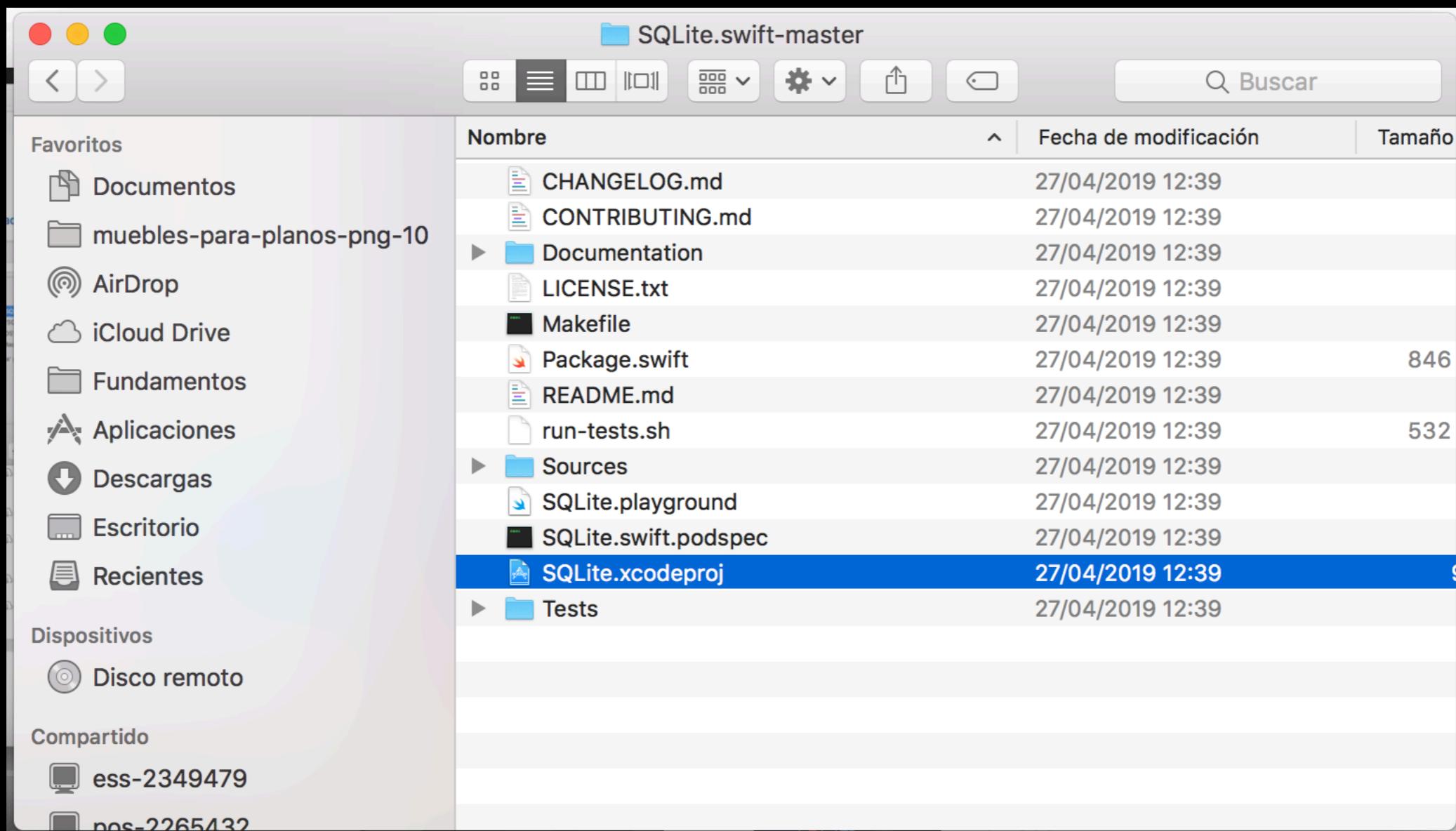
Seleccionamos la opción “Single View App”



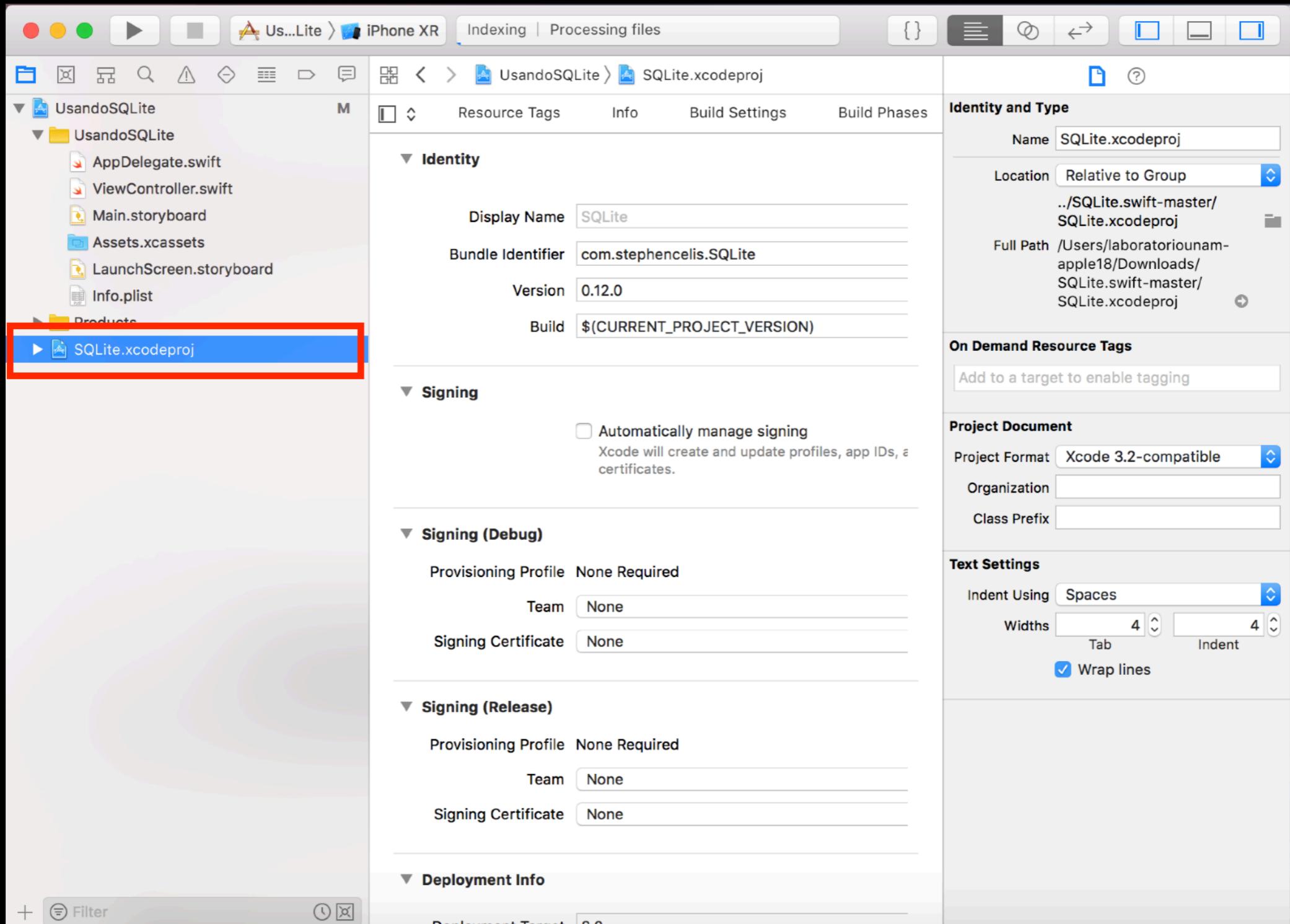
# Agregamos los datos pertinentes



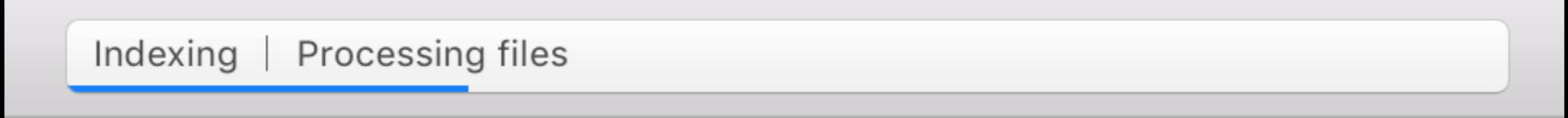
Lo guardamos en la ubicación deseada



Ubicamos el archivo que descargamos desde Github, y dentro de este el correspondiente a “SQLite.xcodeproj”

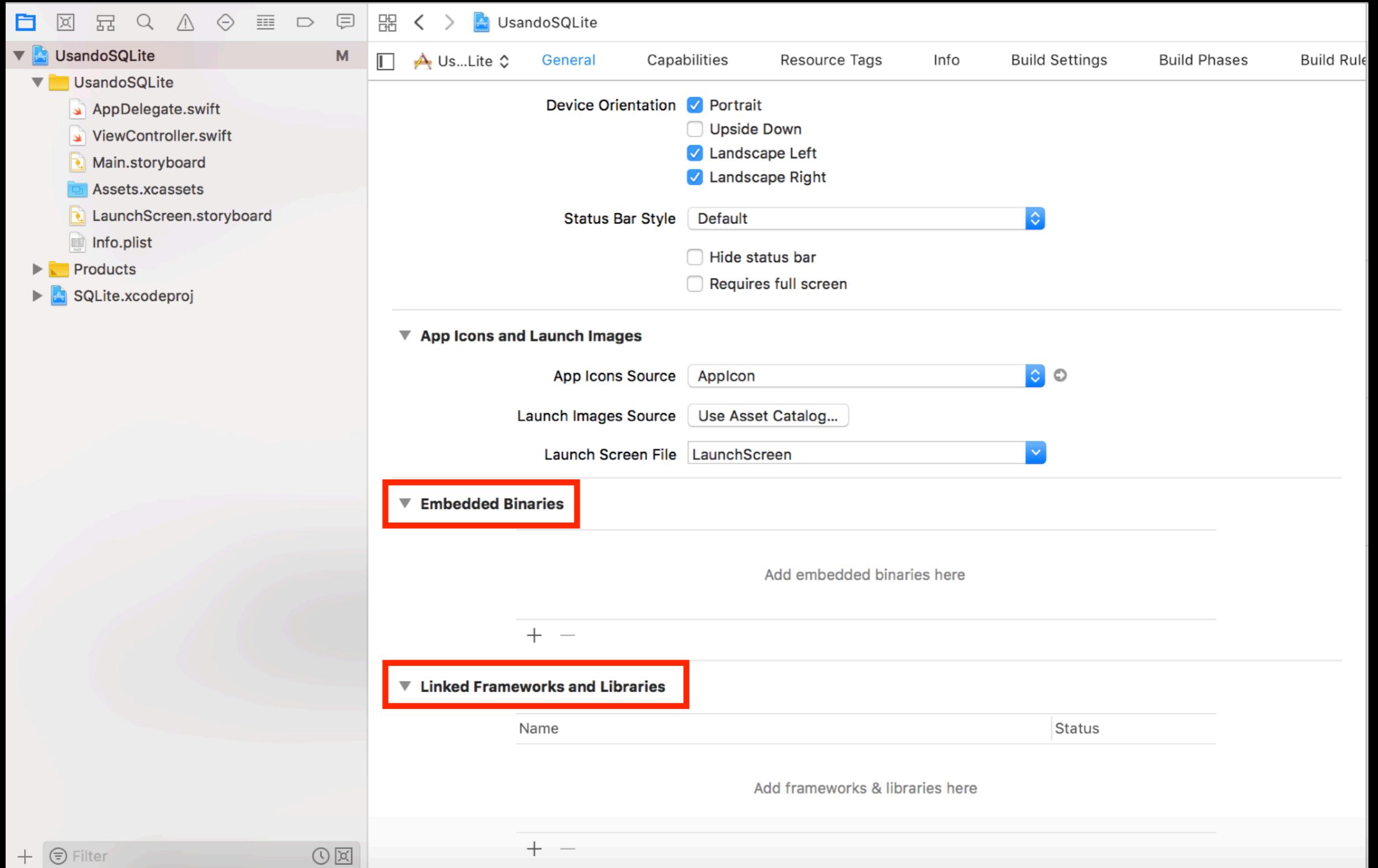


Lo arrastramos dentro del explorador de componentes de nuestro proyecto, en la parte inferior

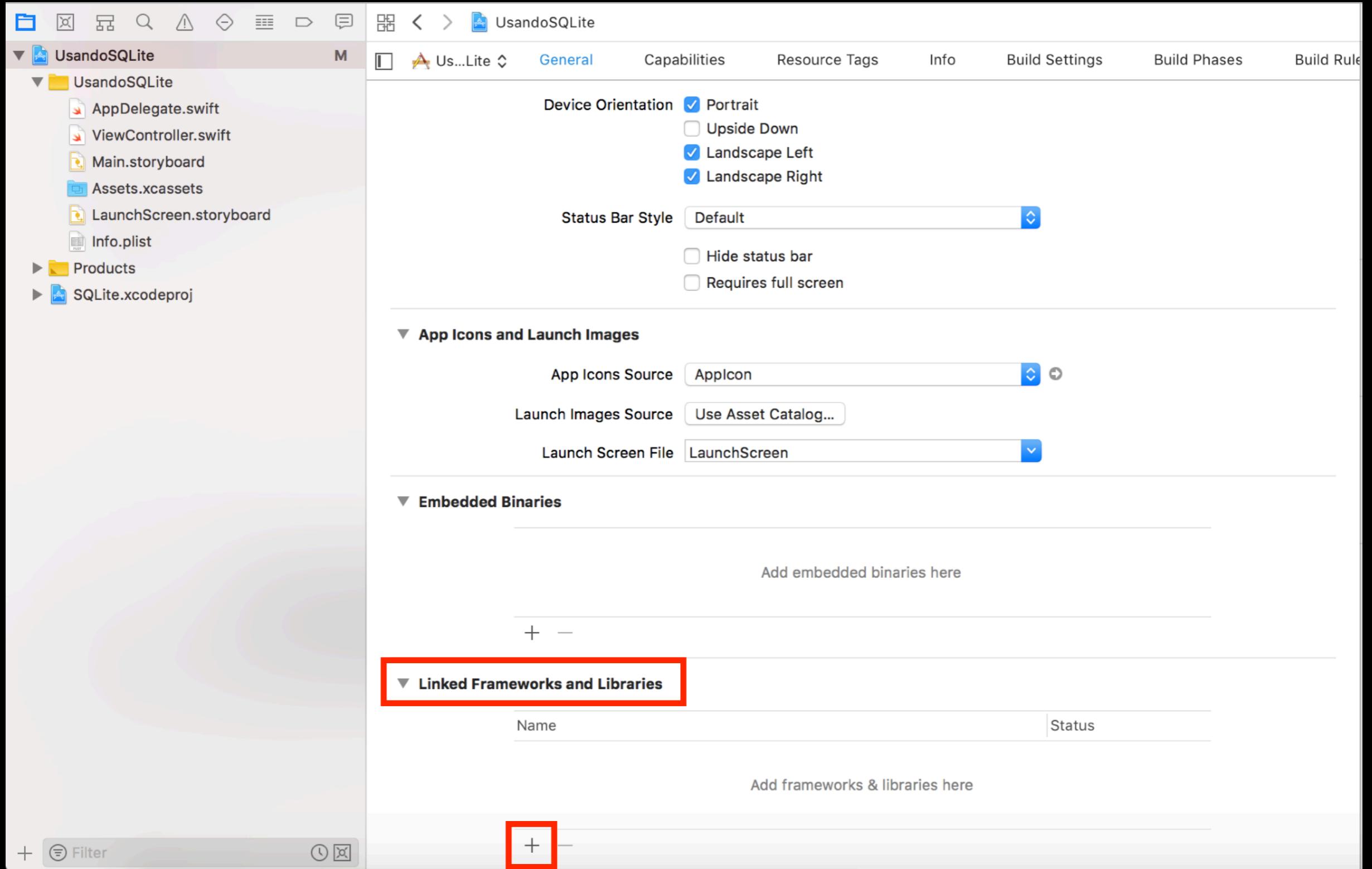


Indexing | Processing files

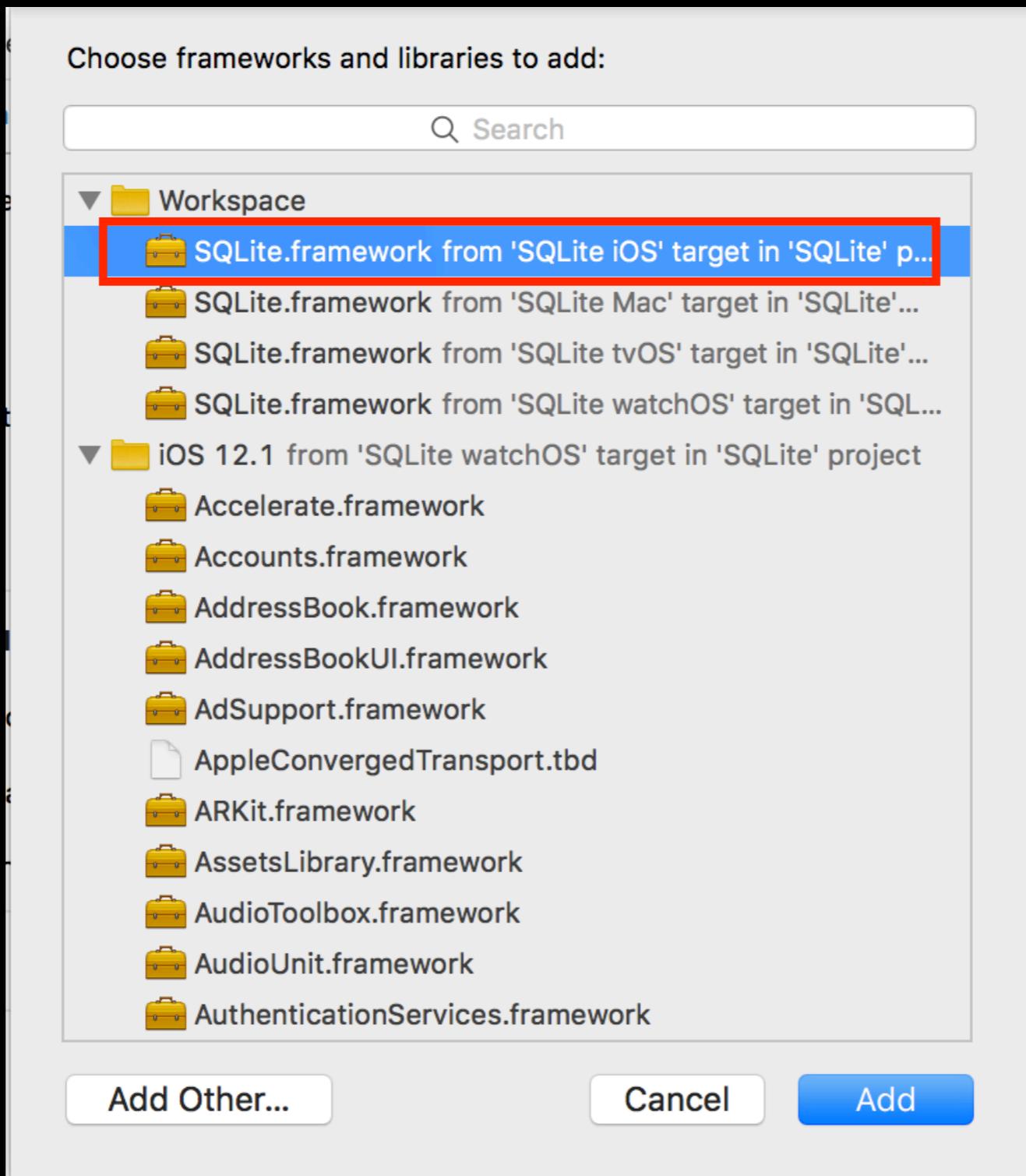
Esperamos a que termine de cargar la barra de estado correspondiente al proyecto (parte superior)



Ubiquemos dos secciones: “Embedded Binaries” y “Linked Frameworks and Libraries”



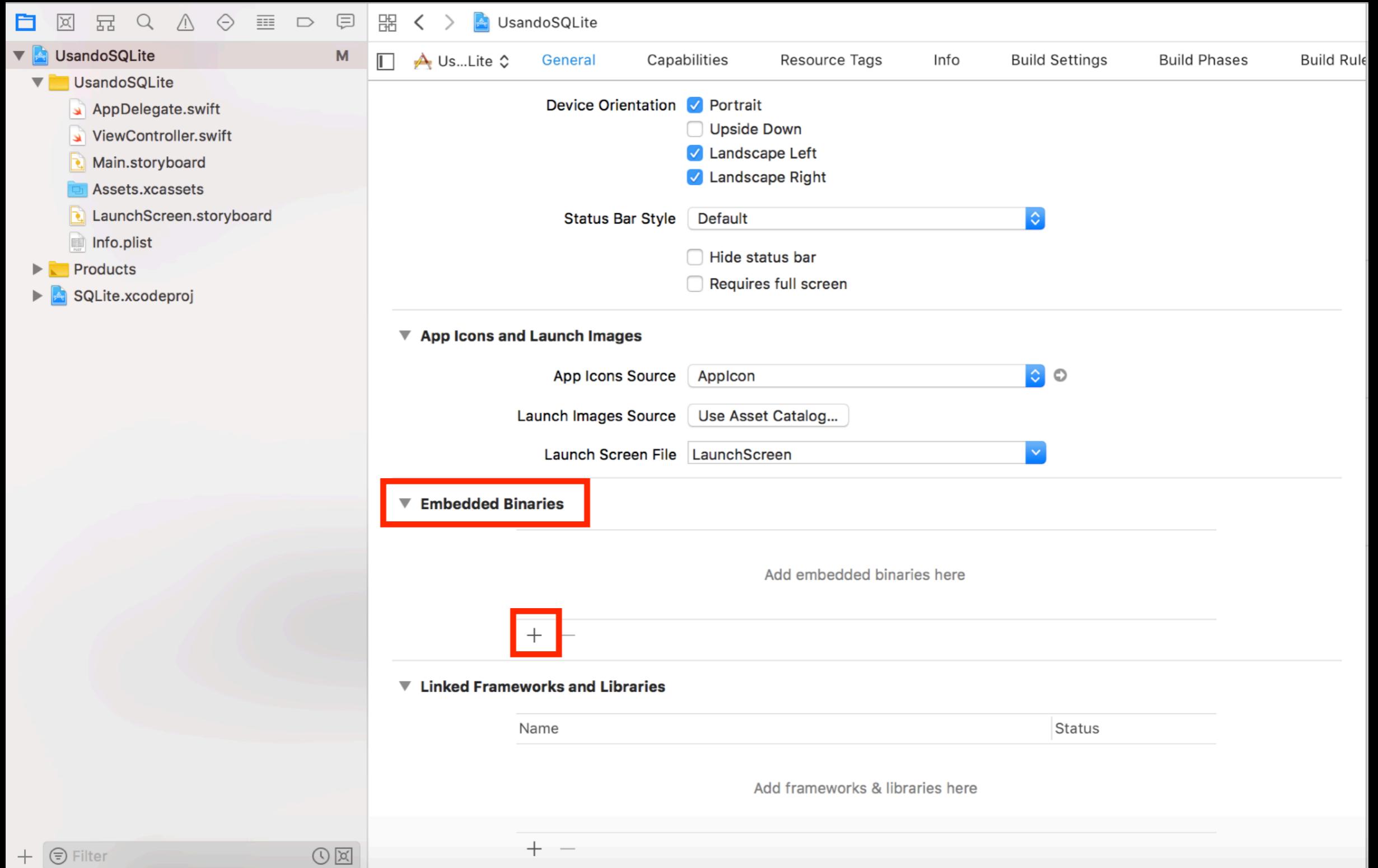
Ubiquemos el signo de más (+) dentro de “Linked Frameworks and Libraries”



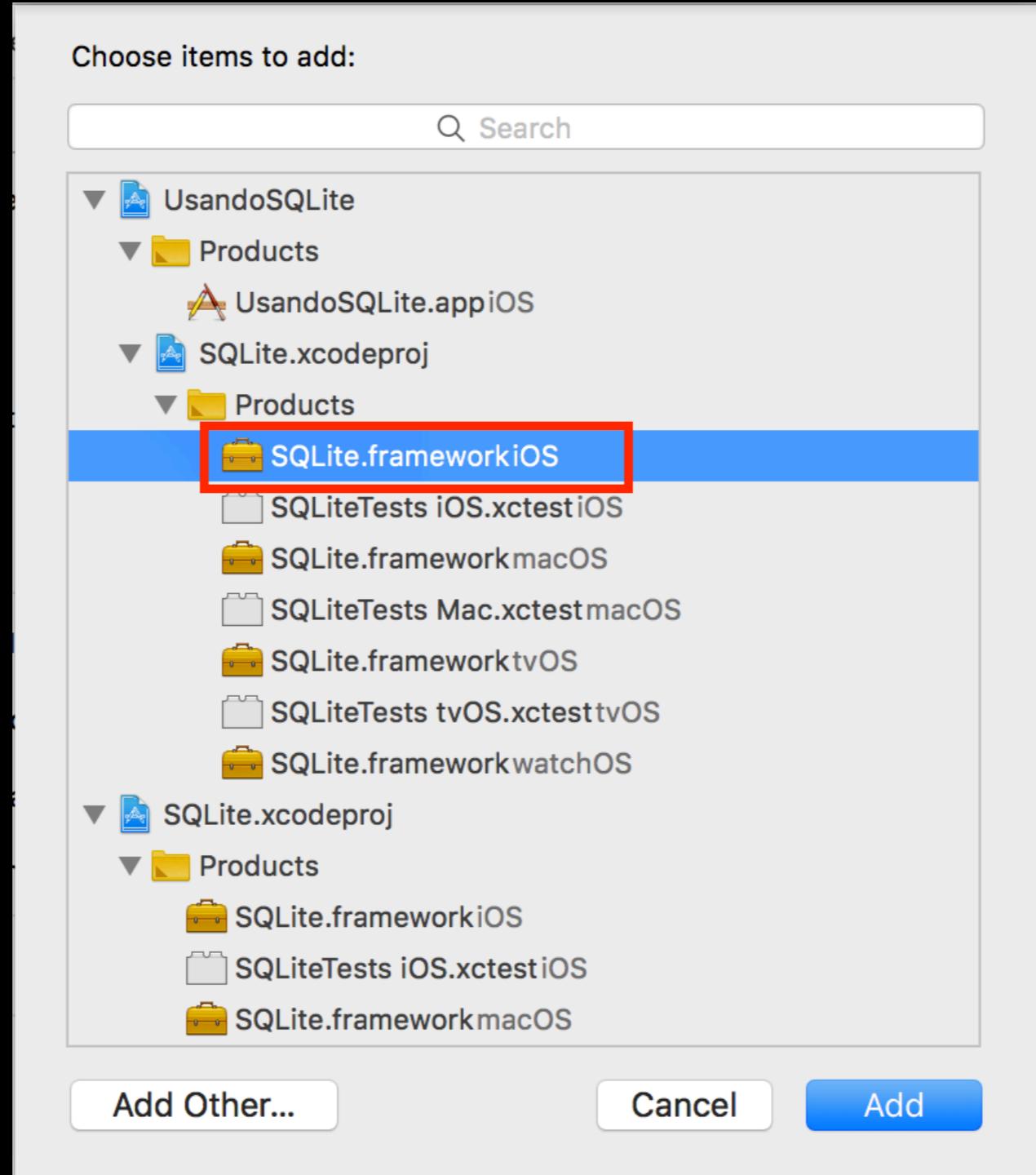
Seleccionemos SQLite.framework correspondiente a los dispositivos iOS

Linked Frameworks and Libraries	
Name	Status
 SQLite.framework	Required ▾
+	-

Veremos que fue agregado a nuestro proyecto dicho framework



Ubiquemos el signo de más (+) dentro de  
“Embedded Binaries”



Seleccionemos el producto correspondiente

The screenshot shows the 'Embedded Binaries' and 'Linked Frameworks and Libraries' sections in Xcode. In the 'Embedded Binaries' section, there is one entry: 'SQLite.framework ...in build/Debug-iphoneos'. Below this section are '+' and '-' buttons. In the 'Linked Frameworks and Libraries' section, there are two entries, both labeled 'SQLite.framework'. The first entry has a status of 'Required ▾' and the second also has 'Required ▾'. Below this section are also '+' and '-' buttons.

Name	Status
SQLite.framework	Required ▾
SQLite.framework	Required ▾

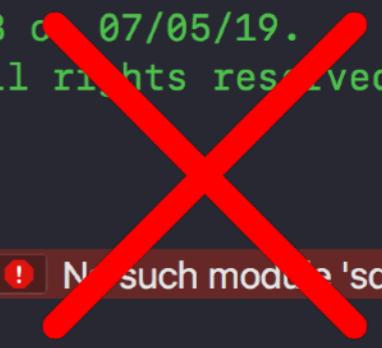
Veremos que el resultado es el siguiente: un componente en la parte superior y dos en la parte inferior



A screenshot of Xcode showing a project named "UsandoSQLite". The file "ViewController.swift" is open. The code includes imports for UIKit and SQLite. A red error message "No such module 'SQLite'" is displayed next to the import statement. The entire screenshot is overlaid with a large green checkmark.

```
1 //  
2 //  ViewController.swift  
3 //  UsandoSQLite  
4 //  
5 //  Created by Laboratorio UNAM-Apple 18 on 07/05/19.  
6 //  Copyright © 2019 CristianTafolla. All rights reserved.  
7 //  
8  
9 import UIKit  
10 import SQLite| ! No such module 'SQLite'  
11  
12 class ViewController: UIViewController {  
13  
14     override func viewDidLoad() {
```

Dentro de ViewController.swift agregaremos el framework SQLite, evitando el autocompetido de SQLite3.



A screenshot of Xcode showing the same project and file setup as the first image. However, the code now shows an attempt to import "sq" instead of "SQLite". A red error message "No such module 'sq'" is displayed. A blue tooltip "Module SQLite3" appears over the partially typed import statement. A large red X is drawn across the entire screenshot.

```
1 //  
2 //  ViewController.swift  
3 //  UsandoSQLite  
4 //  
5 //  Created by Laboratorio UNAM-Apple 18 on 07/05/19.  
6 //  Copyright © 2019 CristianTafolla. All rights reserved.  
7 //  
8  
9 import UIKit  
10 import sq| ! No such module 'sq'  
11     Module SQLite3  
12 class ViewController: UIViewController {  
13  
14     override func viewDidLoad() {
```

The screenshot shows the Xcode interface with the following details:

- Project Navigator:** Shows the project structure with files like AppDelegate.swift, ViewController.swift, Main.storyboard, Assets.xcassets, LaunchScreen.storyboard, and Info.plist.
- Editor:** Displays the content of ViewController.swift. The code includes imports for UIKit and SQLite, and defines a ViewController class that overrides viewDidLoad().
- Identity and Type:** Inspector panel showing the file is named ViewController.swift, is a Default - Swift Source, and is relative to the group.
- On Demand Resource Tags:** Inspector panel showing "Only resources are taggable".
- Target Membership:** Inspector panel showing the file is part of the target UsandoSQLiteSwift4.
- Text Settings:** Inspector panel showing text encoding is "No Explicit Encoding", line endings are "No Explicit Line Endings", and indenting uses spaces with widths of 4 for both Tab and Indent. The "Wrap lines" option is checked.

El error se quitará al compilar el proyecto, aunque no hayamos programado nada aún.

```
13
14    override func viewDidLoad() {
15        super.viewDidLoad()
16
17        do {
18            let documentDirectory = try
19                FileManager.default.url(for: .document
20                Directory, in: .userDomainMask,
21                appropriateFor: nil, create: true)
22        } catch {
23            print(error)
24        }
25    }
26}
```

Vamos a crear un archivo URL para nuestra base de datos. Se utiliza la estructura **try, do-catch**. Esta se utiliza para administrar los errores. Un símil en Java es el do-catch. Cada función que represente un error potencial debe seguir esta estructura.

```
12 class ViewController: UIViewController {
13
14     override func viewDidLoad() {
15         super.viewDidLoad()
16
17         do {
18             let documentDirectory = try
19                 FileManager.default.url(for: .documentDirectory, in: .userDomainMask, appropriateFor: nil, create: true)
20             let fileUrl =
21                 documentDirectory.appendingPathComponent("users").appendingPathExtension("sqlite3")
22             let database = try
23                 Connection(fileUrl.path)
24         } catch {
25             print(error)
26         }
27     }
28 }
```

Después hacemos una base de datos para los usuarios, utilizando la URL de la parte superior. Este tendrá de nombre “`users`” y de extensión “`sqlite3`”, como lo sugiere la documentación de Github.

## Usage

```
import SQLite

let db = try Connection("path/to/db.sqlite3")
```

```
13  
14     var database: Connection!  
15  
16     override func viewDidLoad() {  
17         super.viewDidLoad()  
18  
19         do {  
20             let documentDirectory = try  
21                 FileManager.default.url(for: .documentDirect  
ory, in: .userDomainMask, appropriateFor:  
22                 nil, create: true)  
23             let fileUrl =  
24                 documentDirectory.appendingPathComponent("us  
ers").appendingPathExtension("sqlite3")  
25             let database = try Connection(fileUrl.path)  
26             self.database = database  
27         } catch {  
28             print(error)  
29         }
```

Probamos la conexión con la base de datos.

```
13
14  var database: Connection!
15
16  override func viewDidLoad() {
17      super.viewDidLoad()
18
19      do {
20          let documentDirectory = try
21              FileManager.default.url(for: .documentDirectory, in: .userDomainMask, appropriateFor: nil, create: true)
22          let fileUrl =
23              documentDirectory.appendingPathComponent("users").appendingPathExtension("sqlite3")
24          let database = try Connection(fileUrl.path)
25
26          self.database = database
27
28      } catch {
29          print(error)
30      }
31  }
32
33  // MARK: - Database
34
35  var database: Connection!
```

Crear una variable global, la cual se denomina database, esta será de tipo Connection!

```
13  
14     var database: Connection!  
15  
16     override func viewDidLoad() {  
17         super.viewDidLoad()  
18  
19         do {  
20             let documentDirectory = try  
21                 FileManager.default.url(for: .documentDirect  
ory, in: .userDomainMask, appropriateFor:  
22                 nil, create: true)  
23             let fileUrl =  
24                 documentDirectory.appendingPathComponent("us  
ers").appendingPathExtension("sqlite3")  
25             let database = try Connection(fileUrl.path)  
26             self.database = database  
27         } catch {  
28             print(error)
```

Posteriormente indicamos que será igual a la base de datos creada en la carga de nuestra aplicación.

```
15  
16  let usersTable = Table("users")  
17  let id = Expression<Int>("id")  
18  let name = Expression<String>("name")  
19  let email = Expression<String>("email")|  
20
```

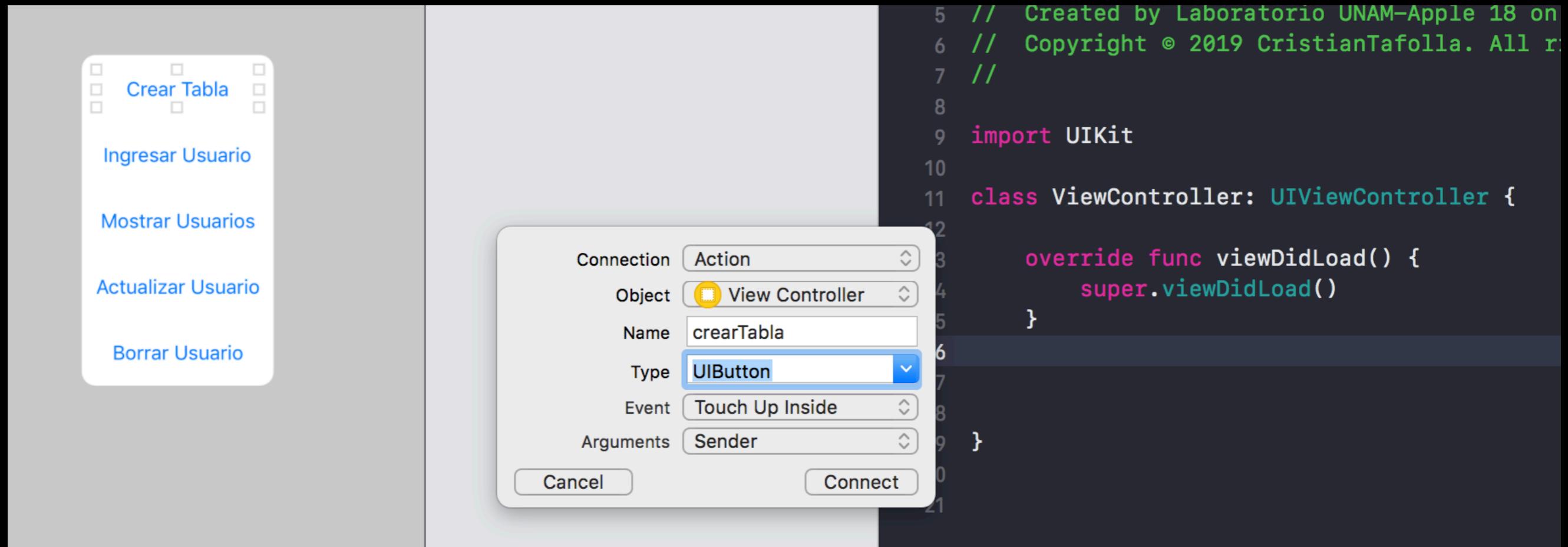
Declaro una constante correspondiente a la tabla de los usuarios.

```
15  
16     let usersTable = Table("users")  
17     let id = Expression<Int>("id")  
18     let name = Expression<String>("name")  
19     let email = Expression<String>("email")|  
20
```

Después creamos los campos de nuestra BD, indicando que son expresiones, entre picoparéntesis ( < > ) el tipo de dicho campo y posteriormente el tipo de dicho campo.

# Tipos de dato

- Programación: Característica que define lo que un dato puede retener.
- Base de datos: Características que definen qué dato puede o no entrar en nuestro caso a una propiedad o atributo.



Dentro del Main.storyboard habrá que crear 5 botones como lo indica la diapositiva

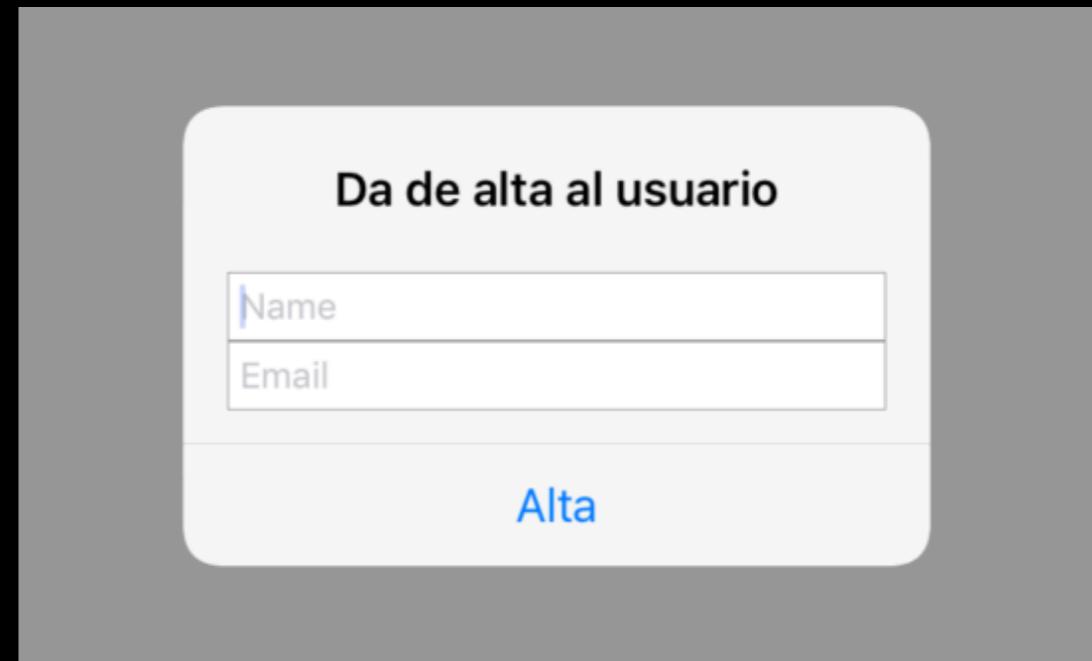


```
11 class ViewController: UIViewController {  
12  
13     override func viewDidLoad() {  
14         super.viewDidLoad()  
15     }  
16  
17     @IBAction func crearTabla(_ sender: UIButton) {  
18         print("Tabla creada")  
19     }  
20  
21     @IBAction func ingresarUsuario(_ sender: UIButton) {  
22         print("Usuario dado de alta")  
23     }  
24  
25     @IBAction func mostrarUsuarios(_ sender: UIButton) {  
26         print("Mostrando usuarios")  
27     }  
28  
29     @IBAction func actualizarUsuario(_ sender: UIButton) {  
30         print("Actualizando usuario")  
31     }  
32  
33     @IBAction func borrarUsuario(_ sender: UIButton) {  
34         print("Eliminando usuario")  
35     }  
36  
37 }  
38  
39 }
```

A cada uno de los botones hay que agregar un “Action”, con un breve mensaje

```
● @IBAction func ingresarUsuario(_ sender: UIButton) {  
56     print("Usuario registrado")  
57     let alert = UIAlertController(title: "Da de alta al usuario", message: nil,  
58                                   preferredStyle: .alert)  
59     alert.addTextField{ (tf) in tf.placeholder = "Name" }  
60     alert.addTextField{ (tf) in tf.placeholder = "Email" }  
61     let action = UIAlertAction(title: "Alta", style: .default){ (_) in  
62         guard let name = alert.textFields?.first?.text,  
63               let email = alert.textFields?.last?.text  
64             else { return }  
65         print(name)|  
66         print(email)  
67     }  
68     alert.addAction(action)  
69     present(alert, animated: true, completion: nil)  
70 }
```

Dentro del “Action” ingresarUsuario, habrá que agregar este segmento de código...



...para que aparezca el siguiente recuadro emergente.

```
① @IBAction func crearTabla(_ sender: UIButton) {  
38     print("Tabla creada")  
39  
40     let createTable = self.usersTable.create{ (table) in  
41         table.column(self.id, primaryKey: true)  
42         table.column(self.name)  
43         table.column(self.email, unique: true)  
44     }  
45  
46     do{  
47         try self.database.run(createTable)  
48         print("La tabla se creó con éxito")  
49     } catch {  
50         print(error)  
51     }  
52 }
```

Dentro del “Action” crearTabla, habrá que agregar este segmento de código para crear la tabla con la estructura especificada al inicio.