

Curso iOS

Preparado por: Eduardo Hoyos Loli

RESUMEN

Introducción

El presente manual servirá como material de apoyo para las clases del Curso de iOS, se explicarán a detalle cada tema. El curso tendrá una duración de 15 horas, las cuales serán divididas en 20% teoría y 80% práctica.

Objetivos

- El alumno tendrá la capacidad de conocer cuales son los elementos principales de Xcode.
- El alumno tendrá la habilidad de desarrollar una aplicación desde cero.
- Aprenderá a desarrollador usando el lenguaje de programación Swift 5.
- Conocerá los componentes de la interfaz gráfica de una aplicación
- Conocerá las características del componente de MapKit.
- Aprenderá a consumir servicios SOAP y mostrar la información en la aplicación.
- Se verá la subida de un app a tienda.

ENTORNO XCODE

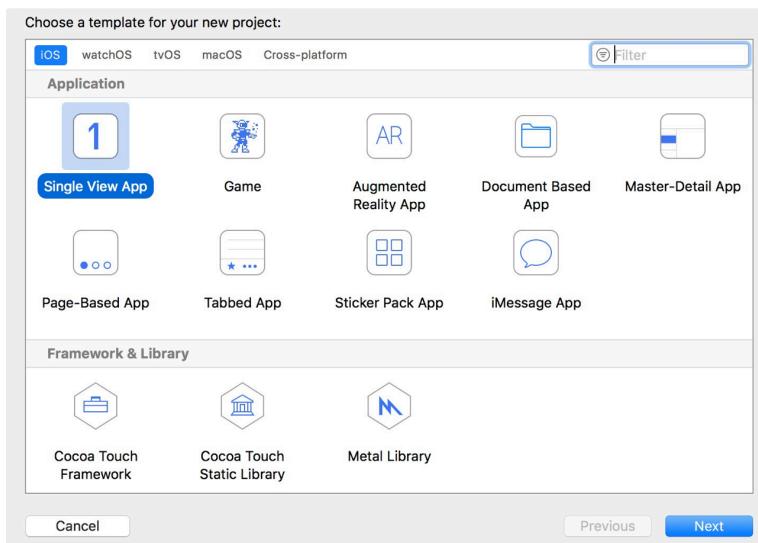
Xcode

Para conocer el entorno de Xcode procederemos a ver las siguientes imágenes.



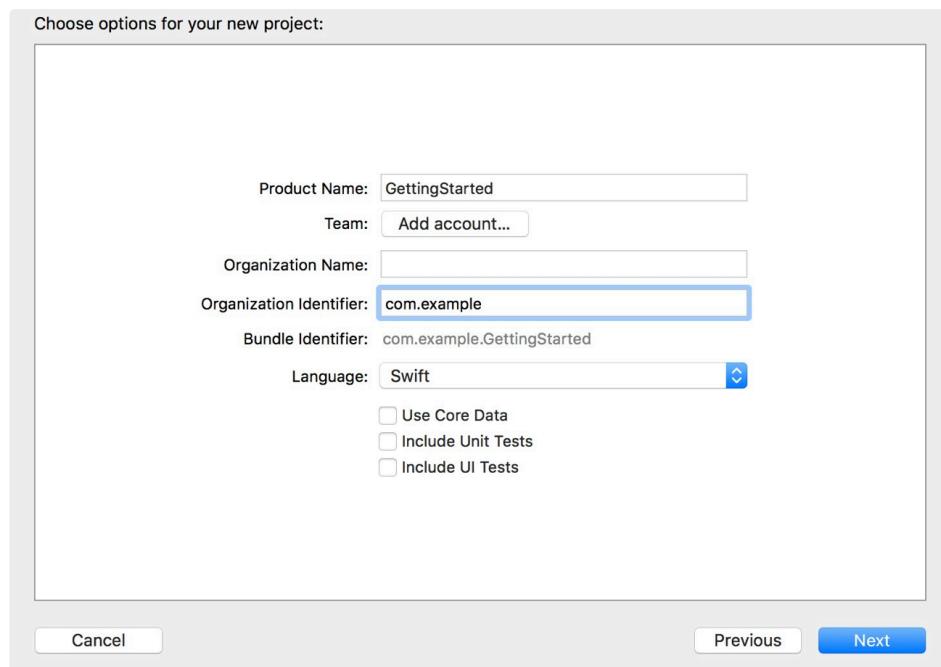
Al abrir Xcode podemos observar la siguiente ventana. Donde nos da la opción de comenzar un proyecto con Playground, Crear un proyecto nuevo o Clonar un proyecto existente.

Para comenzar usaremos la opción de Crear un nuevo proyecto. Nos aparecerá la siguiente ventana.



En esta pantalla deberemos elegir el tipo de aplicación que vamos a desarrollar. En nuestro caso será la primera opción donde dice Single View App, o Aplicación de Vista Simple.

Luego, Xcode nos pide que pongamos la información de la aplicación.

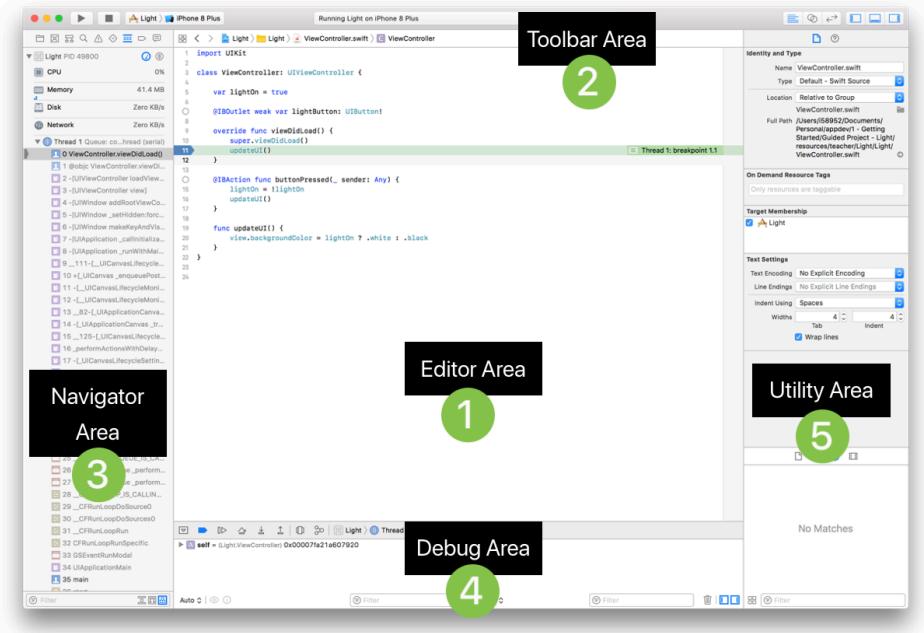


Entre la información que nos pide podemos encontrar cómo se va a llamar nuestra aplicación, cuál es el nombre de la organización desarrolladora, el identificador de la organización, y el lenguaje en el que estará la aplicación.

Luego de ingresar toda la información, revisamos por última vez los campos y le damos click en el botón Siguiente/Next.

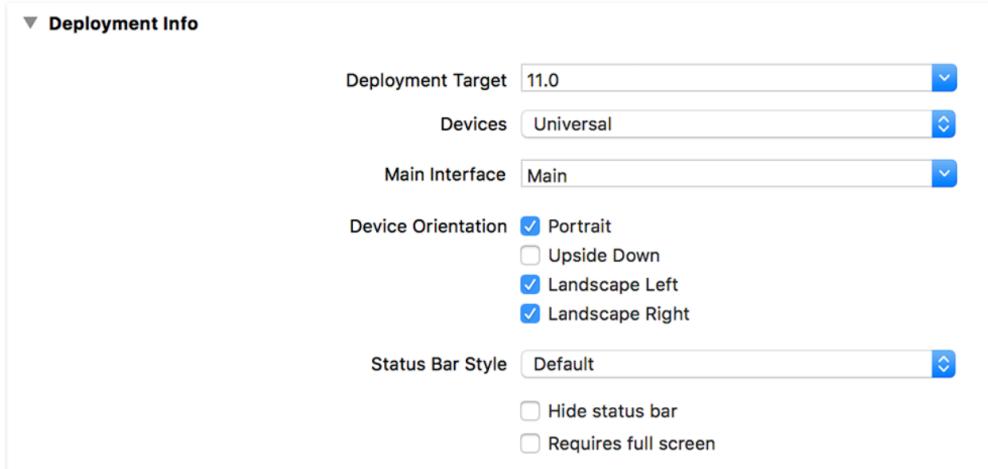
Nuestra aplicación procederá a crearse para ser editada en Xcode. Una vez dentro de Xcode podremos ver las siguientes divisiones en nuestro IDE.

1. Editor Area: Es la sección donde podremos editar el código de nuestra nueva aplicación.
2. Toolbar Area: En esta sección podremos ver cuando nuestra aplicación se está compilando, podremos elegir el simulador o equipo iPhone en el que deseemos que nuestra aplicación compile, entre otros.
3. Navigator Area: En esta sección podremos ver cuál es la estructura de carpetas y archivos que tendrá nuestra aplicación.
4. Debug Area: Cuando nuestra aplicación esté corriendo o compilando, podremos ver cuál es el comportamiento de las librerías o código que hayamos dejado para poder tracear cada paso que hace nuestra aplicación.
5. Utility Area: En esta sección podremos ver las opciones de los objetos que tengamos en selección en nuestro proyecto, desde la información del archivo hasta diseño de algunos componentes.



Secciones de Xcode

Luego en el detalle del proyecto podremos ver el deployment target, la versión de iOS mínima en la que queramos que nuestra aplicación compile. Devices, aquí podremos elegir entre iPhone, iPad o Universal que involucra a los dos dispositivos. Device Orientation, la manera en la que nuestra app podrá presentarse en el dispositivo.



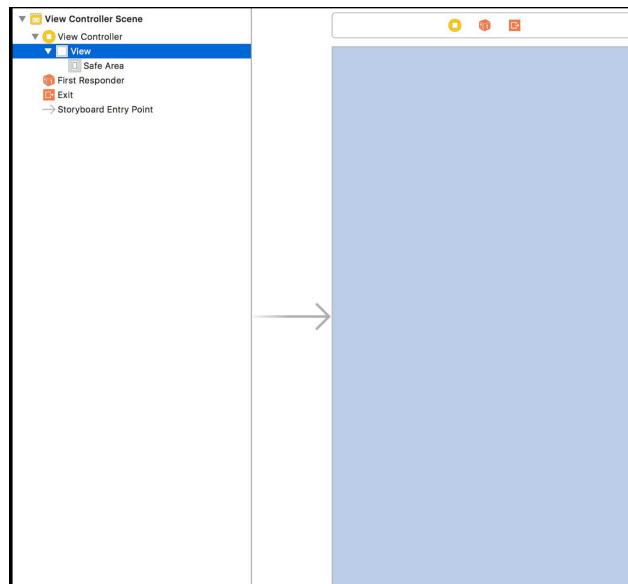
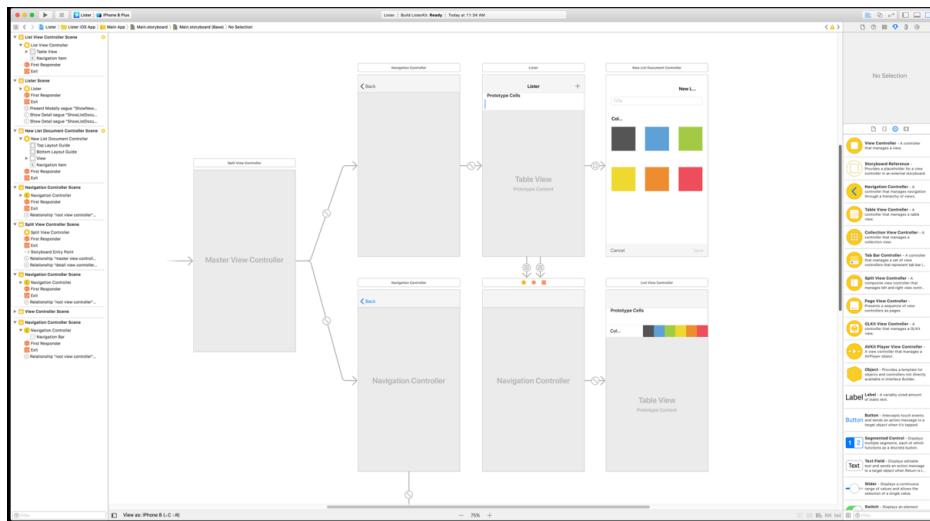
STORYBOARDS

INTERFACE BUILDER

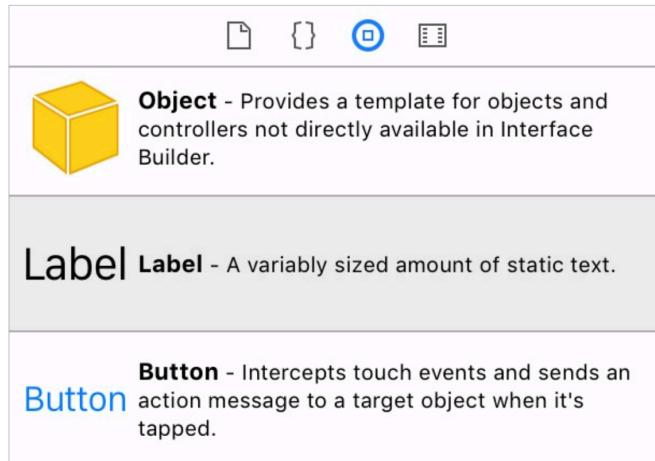
El Interface builder se abre cada vez que seleccionas un archivo .xib o un .storyboard del navegador de proyecto.

Un archivo XIB contiene la interfaz de usuario para un singular elemento, ya sea un vista full-screen, una celda, algún control, algún control UI customizable. Los archivos XIBs se usaron con mayor intensidad antes de los storyboards, ahora se usan para ciertos casos pero ahora nos concentraremos en los storyboards.

En contraste con un archivo XIB, un archivo storyboard incluye varias piezas de la interface, definiendo el layout de una o varias vistas de nuestra aplicación. Como desarrollador, encontrarás que la habilidad de ver multiples ventanas al mismo tiempo ayudará a entender el flujo de tu aplicación.

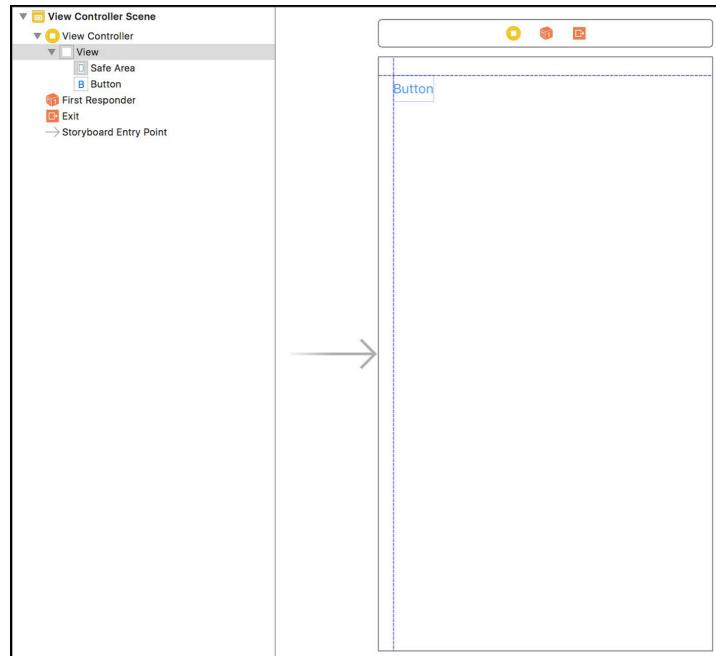


Indagando por nuestra vista que nos ha dejado un archivo storyboard vamos a ver que tenemos un librera de objetos.



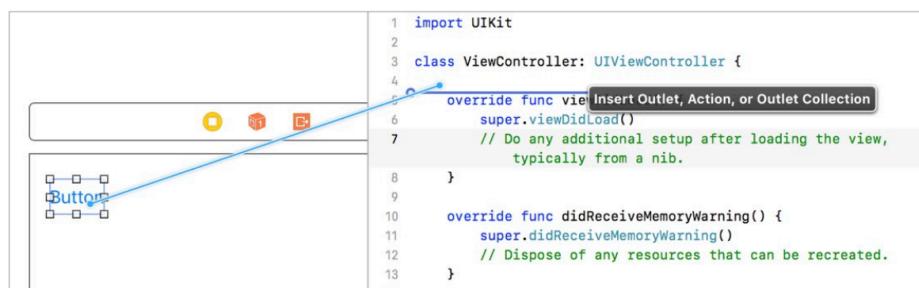
En la cuál podremos encontrar objetos tales como Labels, botones, switches, vistas controladoras, tablas, colecciones, entre otras.

Comenzaremos arrastrando un botón de la librera de objetos hacia la vista por defecto que tenemos dentro de nuestro storyboard.



OUTLETS AND ACTIONS

Outlet son las definiciones de nuestro objetos en nuestro código. Cuando estemos en el storyboard vamos a seleccionar la vista haciendo click en de la vista. Luego daremos click en el botón para abrir el Assistant Editor. Entonces creamos nuestro primer botón en la vista, vamos a realizar lo siguiente. Para crear un outlet vamos a presionar el botón de Control y Click al objeto que tenemos en la vista, en este caso un botón, luego arrastramos hasta el Assistant Editor que contiene la definición de la clase ViewController. Cuando el cursor llegue al archivo verás una línea azul.

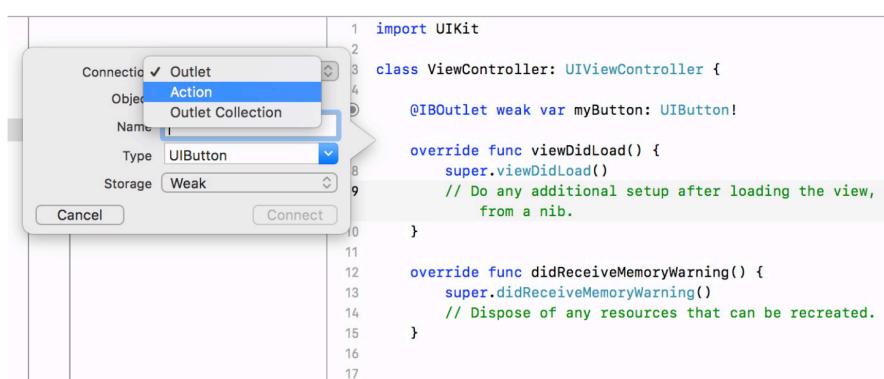


Cuando soltemos el click el diálogo “Outlets and Actions” aparecerá, con Outlet como Connection Type seleccionado. En el campo Name, vamos a especificar un nombre de variable para nuestro botón, en este caso le pondremos “myButton”. Damos click en Connect para terminar la creación del outlet, generando una línea de código que define al outlet.



`@IBOutlet weak var myButton: UIButton!`

Para crear una acción vamos a realizar el mismo procedimiento, en lugar de Outlet pondremos Action.



Cuando demos click en Connect, veremos ahora que nos creó una función la cuál dentro pondremos la lógica que queremos hacer cuando se haga tap en el botón.

```
①     @IBAction func buttonPressed(_ sender: Any) {  
②  
③ }
```

Comenzaremos escribiendo dentro de la función la siguiente línea.

```
print("The button was pressed")
```

Cuando corramos la aplicación y demos Tap en el botón podremos ver en el Debug Area que se imprime lo siguiente.



CONSTANTS

Cuando deseemos crear una variables que no cambie durante su ciclo de vida.

```
let name = "John"
```

El código define a una constante llamada **name** y le asigna el valor “**John**” a la constante. Si quieres acceder a ese valor en futuras líneas después, puedes usar **name** para referenciar directamente a la constante.

```
let name = "John"  
print(name)
```

Ya que **name** es una constante, no le puedes asignar un nuevo valor después de haberla asignado. Por ejemplo, el siguiente código no compilará.

```
let name = "John"  
name = "James"
```

VARIABLES

Cuando quieras nombrar una variables que va a cambiar durante el ciclo de vida de la aplicación, usarás una variable.

Se define variables usando el keyword **var**.

```
var age = 29  
print(age)
```

Ahora ya que **age** es una variable se le puede asignar un nuevo valor en futuras líneas de código.

```
var age = 29  
age = 30  
print(age)
```

El código habrá impreso 30 en la consola.

Puedes asignar constantes y variables desde otras constantes y variables. Esta funcionalidad es útil cuando quieras copiar un valor a una o más variables.

```
let defaultScore = 100
var playerOneScore = defaultScore
var playerTwoScore = defaultScore

print(playerOneScore)
print(playerTwoScore)

playerOneScore = 200
print(playerOneScore)
```

Console Output:

```
100
100
200
```

TIPOS DE VARIABLE

Entre los tipos de variable más comunes tenemos a los siguientes.

Type name	Symbol	Purpose	Example
Integer	Int	Represents whole numbers, or integers.	4
Double	Double	Represents numbers requiring decimal points.	13.45
Boolean	Bool	Represents true or false values.	true
String	String	Represents text.	"Once upon a time..."

CONTROL FLOWS

Dentro de esta sección podremos encontrar controles tales como IF, FOR, SWITCH.

Para comenzar tenemos al control IF, el cual nos indica “Si la condición es verdad, entonces corre el bloque de código”. Si la condición no fuera verdad, el programa saltará el bloque de código.

```
let temperature = 100
if temperature >= 100 {
    print("The water is boiling.")
}
```

Console Output:

```
The water is boiling.
```

Para el caso de IF-ELSE, cuando se añade el ELSE a la sentencia IF, podremos especificar un bloque de código a ejecutarse cuando la condición no sea verdad.

```
let temperature = 100
if temperature >= 100 {
    print("The water is boiling.")
} else {
    print("The water is not boiling.")
}
```

Puede darse el caso de IF-ELSE anidados. Y se pueden usar cuantas veces sea necesario para el programa.

```
var finishPosition = 2

if finishPosition == 1 {
    print("Congratulations, you won the gold medal!")
} else if finishPosition == 2 {
    print("You came in second place, you won a silver medal!")
} else {
    print("You did not win a gold or silver medal.")
}
```

Boolean

Puedes asignar los resultados de una operación lógica a una constante o variable **Bool** para que se verifique o se acceda después. Los valores de **Bool** solo pueden ser **true** o **false**.

```
let number = 1000
let isSmallNumber = number < 10
// number is not less than 10, so isSmallNumber is assigned
// a `false` value
```

También es posible invertir el valor **Bool** usando el operador lógico NOT, el cual se representa con **!**.

```
var isSnowing = false

if !isSnowing {
    print("It is not snowing.")
}
```

Console Output:

```
It is not snowing.
```

De la misma manera, puedes usar el operador lógico AND, representado por **&&**, para verificar dos o más condiciones son **true**.

```
let temperature = 70

if temperature >= 65 && temperature <= 75 {
    print("The temperature is just right.")
} else if temperature < 65 {
    print("It is too cold.")
} else {
    print("It is too hot.")
}
```

Console Output:

```
The temperature is just right.
```

Tienes incluso otra opción, el operador lógico OR, representado por **||**, para verificar si una o dos condiciones son **true**.

```
var isPluggedIn = false
var hasBatteryPower = true

if isPluggedIn || hasBatteryPower {
    print("You can use your laptop.")
} else {
    print("⚠️")
}
```

Console Output:

```
You can use your laptop.
```

Sentencia SWITCH

Una vez visto las sentencias IF-ELSE. La sentencia Switch tiene la misma idea, lo diferente es la sintaxis.

```
let numberOfWheels = 2
switch numberOfWheels {
case 1:
    print("Unicycle")
case 2:
    print("Bicycle")
case 3:
    print("Tricycle")
case 4:
    print("Quadcycle")
default:
    print("That's a lot of wheels!")
}
```

Cuando se trabaja con números, se pueden usar intervalos para verificar si la condición esta incluida dentro de un rango.

```
switch distance {  
    case 0...9:  
        print("Your destination is close.")  
    case 10...99:  
        print("Your destination is a medium distance from here.")  
    case 100...999:  
        print("Your destination is far from here.")  
    default:  
        print("Are you sure you want to travel this far?")  
}
```

Operador Ternario

Conocido entre los diferentes lenguajes de programación, se puede tener la siguiente forma. Dado que son muchas líneas de código para una condición básica.

```
var largest: Int  
  
let a = 15  
let b = 4  
  
if a > b {  
    largest = a  
} else {  
    largest = b  
}
```

Podríamos usar la siguiente forma. Donde se cumple la misma condición y se elige el resultado de forma directa hacia la variable **largest**.

```
var largest: Int  
let a = 15  
let b = 4  
  
largest = a > b ? a : b
```