

**UNIVERSIDAD DE JAÉN
ESCUELA POLITÉCNICA SUPERIOR DE JAÉN**



**Minitutorial Swift 4:
Utilización de la herramienta Webkit View**

**Curso 2017-2018
Grado de Ingeniería Informática**

Realizado por:

Nombre: Manuel García López DNI: 77381751E Correo electrónico: mgl00050@red.ujaen.es

Grupo 1 Prácticas: Viernes 8:30-10:30

Fundamento teórico

1-¿Por qué usar la herramienta WebKit View? Para responder a esta pregunta, debemos responder anteriormente a la siguiente pregunta. ¿Que es la herramienta WebKit View?

WebKit View es una herramienta la cual nos permite realizar búsquedas a través de un motor de internet previamente seleccionado haciendo de interfaz entre dicho motor y la aplicación de usuario. Lo cual es muy útil ya que es como tener ese browser instalado sin la necesidad de tenerlo realmente.

Una vez habiendo explicado esto, podemos entender, el porqué del uso de WebKit View. Pero, ¿no hay mas herramientas que nos permitan obtener la misma funcionalidad de una forma más sencilla?

WebKit View es una herramienta perteneciente a Swift 4, esto significa que anteriormente para implementar la misma funcionalidad dentro de una aplicación, se debía realizar de otra forma. Concretamente, existía y aún existe una herramienta llamada WebView que sin embargo ante la aparición de WebKit View, ha quedado en cierto modo obsoleta, además iOS, desaconseja su uso actualmente.

Por lo tanto, WebKit View se nos presenta como el modo más efectivo de incluir una búsqueda web de forma cómoda y eficiente.

2- Si WebKit View ya posee una barra (perteneciente al propio browser) que realiza sus propias búsquedas, ¿Porqué añadir una funcionalidad extra, consistente en una barra en la que poner texto, y un botón de búsqueda? La razón es que la herramienta de WebKit View, no nos proporciona una barra de búsqueda fija, sino simplemente la del visor por defecto, por lo tanto, es interesante de cara a el manejo con el usuario, el hecho de tener una barra fija en la que poder insertar un texto y establecer una búsqueda en todo momento.

Tutorial

En este tutorial, vamos a aprender a utilizar una de las herramientas que iOS pone a nuestra disposición en la biblioteca de herramientas de Swift 4. En este caso usaremos la funcionalidad Webkit View, mediante la cual podremos realizar una búsqueda (en nuestro caso a través de Google) sin salir de la vista actual en la que nos encontramos, y escribiendo solo la ruta, o palabras que escribiríamos de encontrarnos usando el propio motor de búsqueda de Google (o cualquier otro).

Inicialmente, no usaremos la barra de búsqueda cargada por defecto en el motor de búsqueda, sino que usaremos una barra auxiliar que crearemos nosotros, y en la que parsearemos la información escrita por el usuario.

Empecemos por lo básico:

Crear el main Storyboard

1. Abra Xcode, el cual de no estar anclado en la barra inicial del escritorio, podrá encontrar a través del Finder en la sección de aplicaciones.

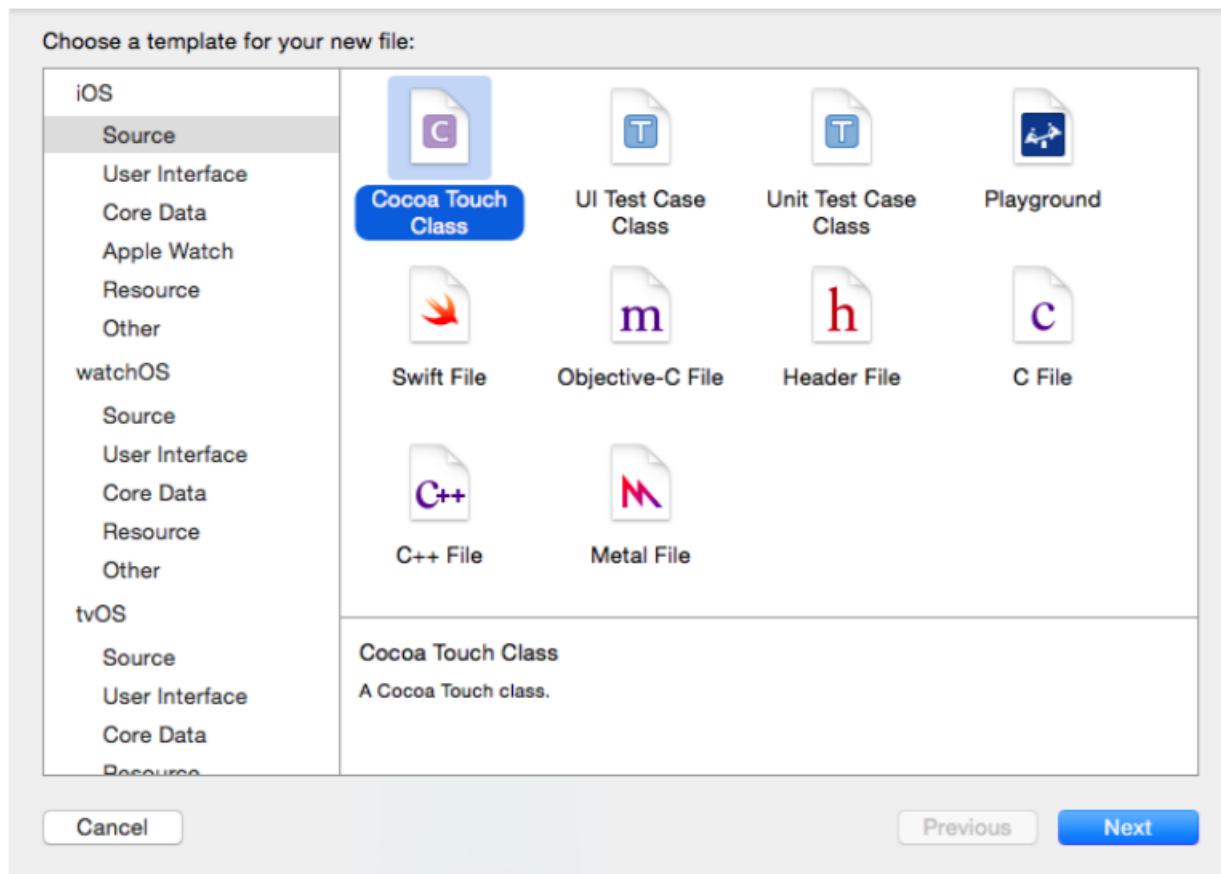
2. Seleccione Open project si desea incrustar la vista del tutorial en un proyecto ya existente (2.1), o bien haga click en nuevo proyecto si desea partir desde cero (2.2)

2.1 Crear una nueva vista en un proyecto ya existente

1. Abra su StoryBoard, y clique en Main.storyboard.
2. Abra la “Object library” en la biblioteca de utilidades, situada en la esquina inferior derecha de Xcode.
3. En la “Object library”, busque en la barra de navegación un objeto denominado “View Controller”.
4. Arrastre dicho “View Controller” y déjelo caer sobre el ‘canvas’ en una sección libre de otras vistas ya existentes. Este controlador de vista será el que accederemos desde otra vista ya existente, y en el que estableceremos la búsqueda web.

Crear una subclase de UIViewController

1. Seleccione File > New > File .
2. A la izquierda del cuadro de diálogo que aparece, seleccione Source bajo iOS, a continuación, seleccione Cocoa Touch Class.

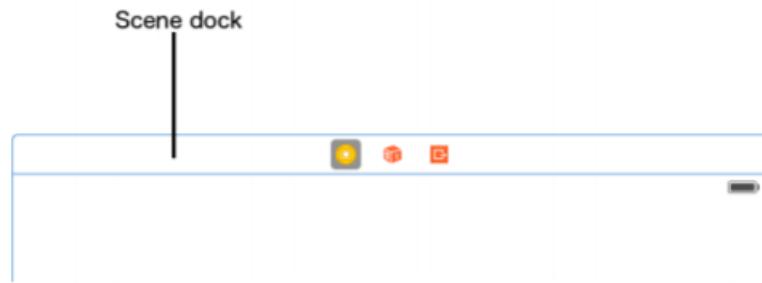


3. A continuación, haga clic en Next.
4. En el campo Class, escriba el nombre que desee.
5. En el campo “Subclass of” marque la opción UIViewController . El título de la clase puede cambiar, es mejor no tocarlo.
6. Asegúrese de que la opción “Also create XIB file” no está seleccionada.
7. Asegúrese de que la opción de Language se establece en Swift.
8. Vuelva a hacer clic en Next.
9. Guarde los valores predeterminados de ubicación para el directorio del proyecto. En la sección de Targets, su aplicación está seleccionada y las pruebas para su aplicación están sin seleccionar.
10. Deje estos valores por defecto, y haga clic en Create.

Asigne la clase creada a la nueva escena

1. Abra su storyboard.

2. Seleccione la escena de la web haciendo clic en su scene dock.

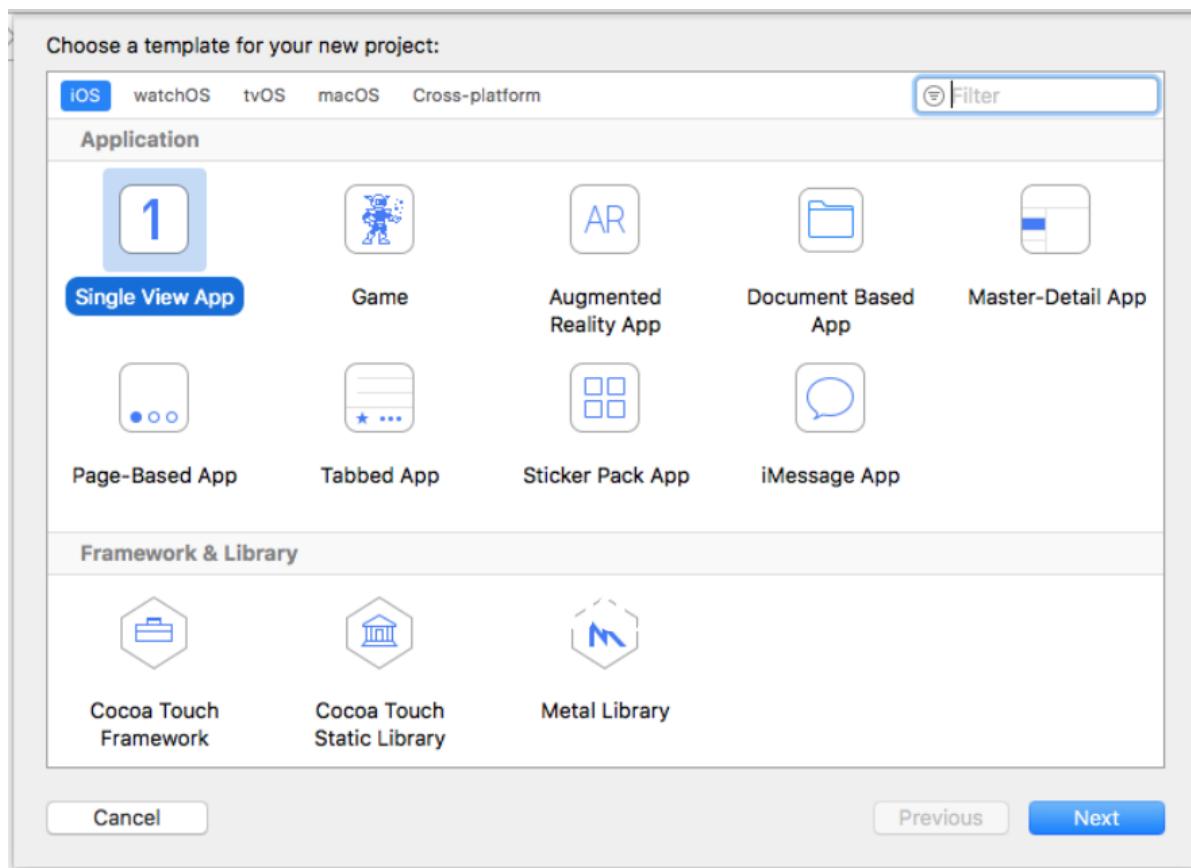


3. Con la escena de la web seleccionado, abra el Identity Inspector.

4. En el Identity Inspector, seleccionamos en el campo Class la clase WebViewController.

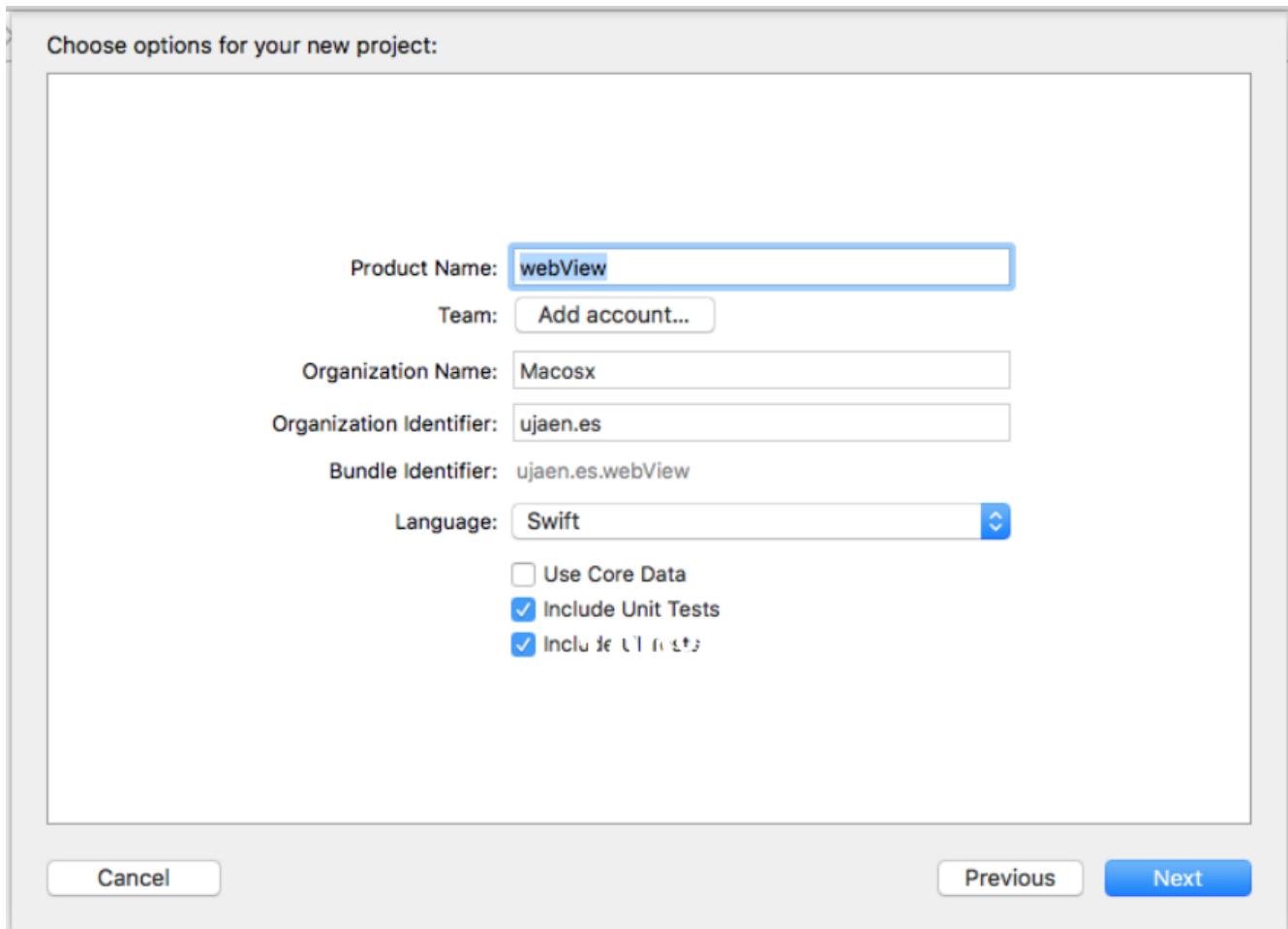
2.2 Crear el proyecto desde cero

1. Seleccione como plantilla la que viene por defecto, asegurese de que sea de tipo Single View App.

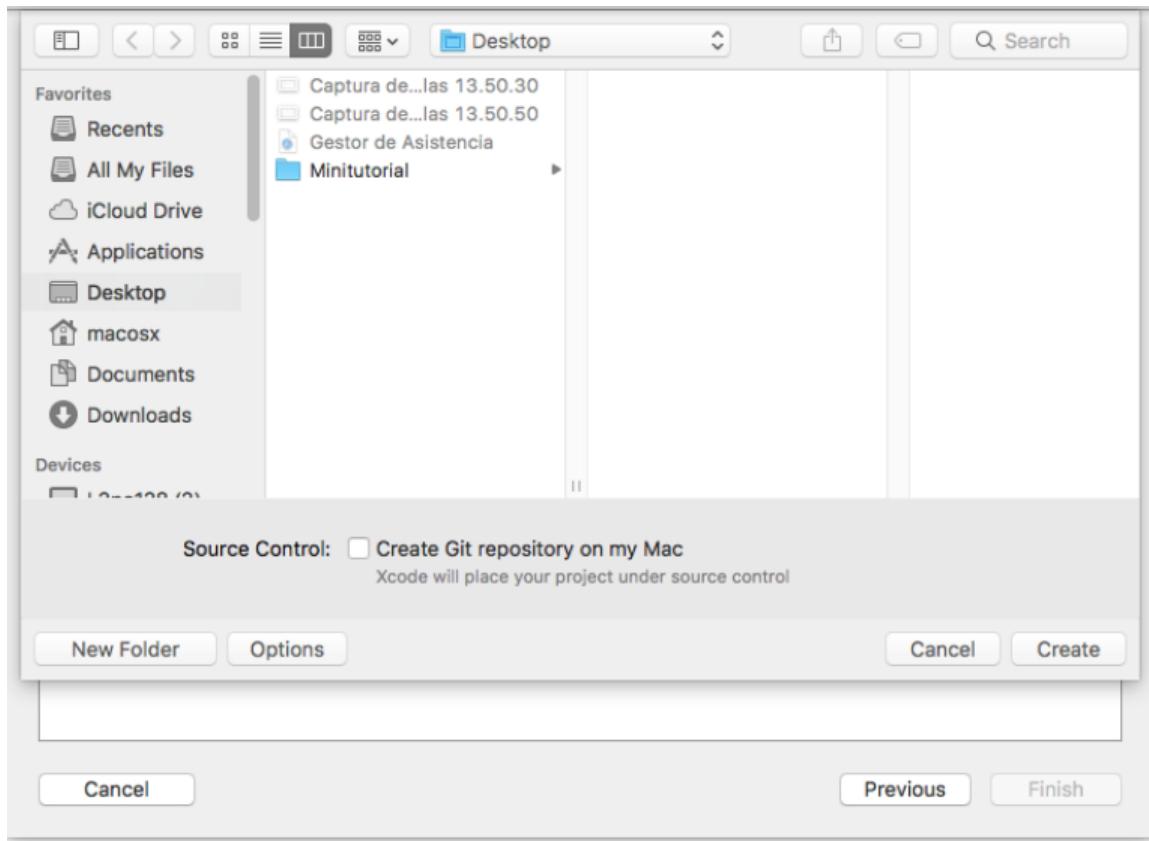


2. A continuación, haga clic en Next.

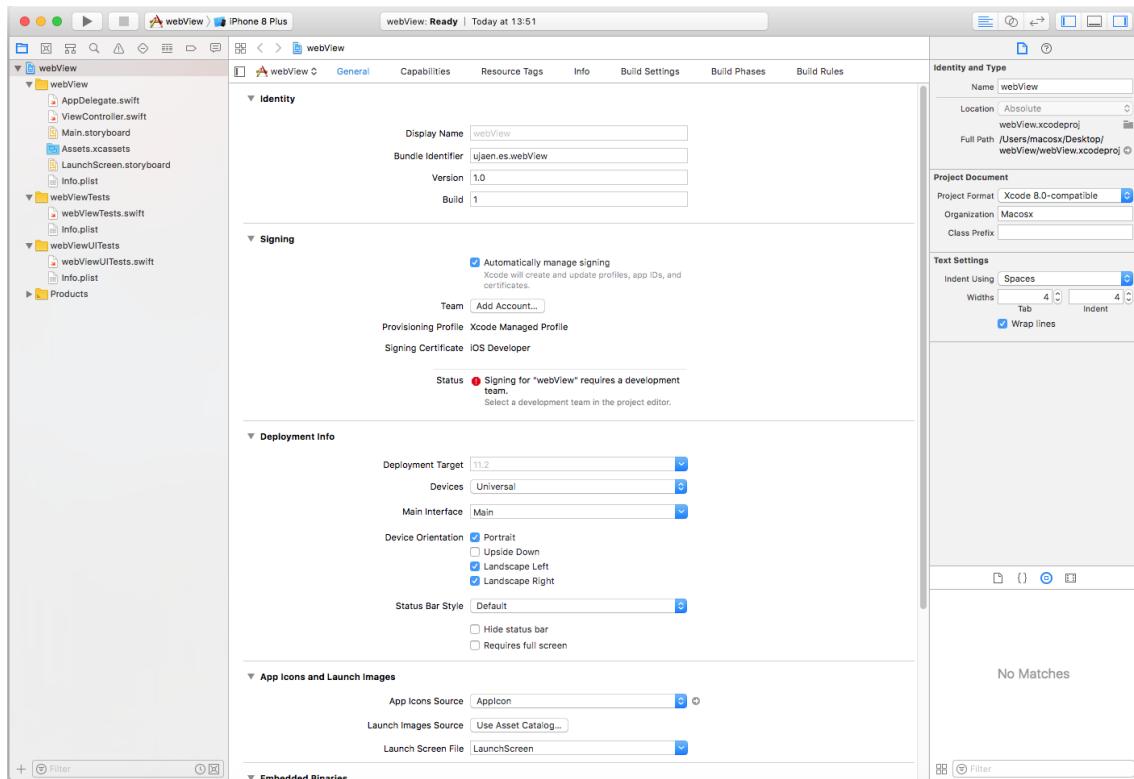
3. Escriba el nombre de la app que desee.
4. Asegurese de que el lenguaje está asignado a Swift.
5. Asegurese de que las unidades de test estan seleccionadas.
6. Vuelva a hacer clic en next.



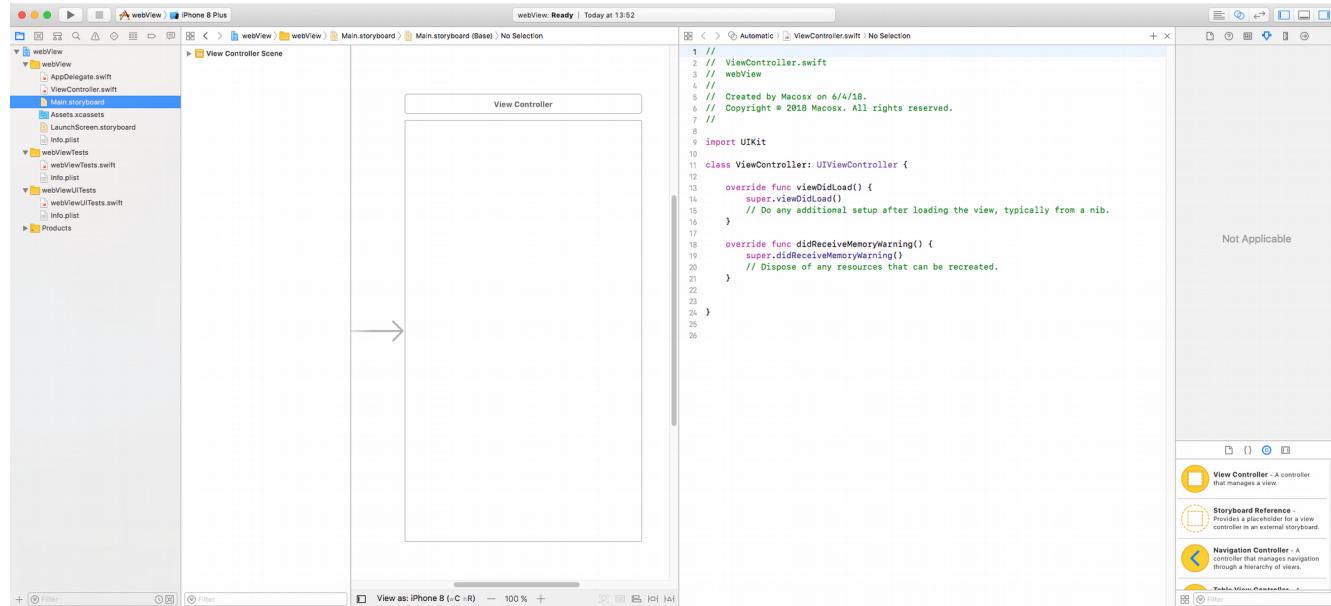
7. Deje los siguientes valores por defecto y haga clic en create.



Al finalizar, su nueva app se visualizará así:

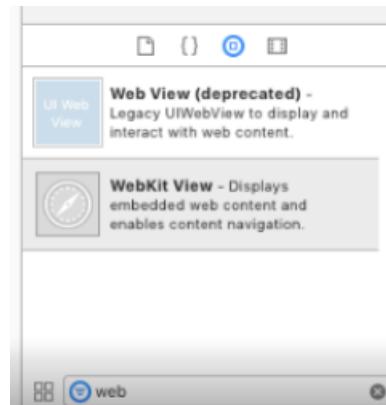


8. Haga clic en main storyboard, y a continuación expanda la pantalla de visión para poder observar también el código del View Controller correspondiente, para ello haga clic en los dos círculos que aparecen en la esquina superior derecha de Xcode.

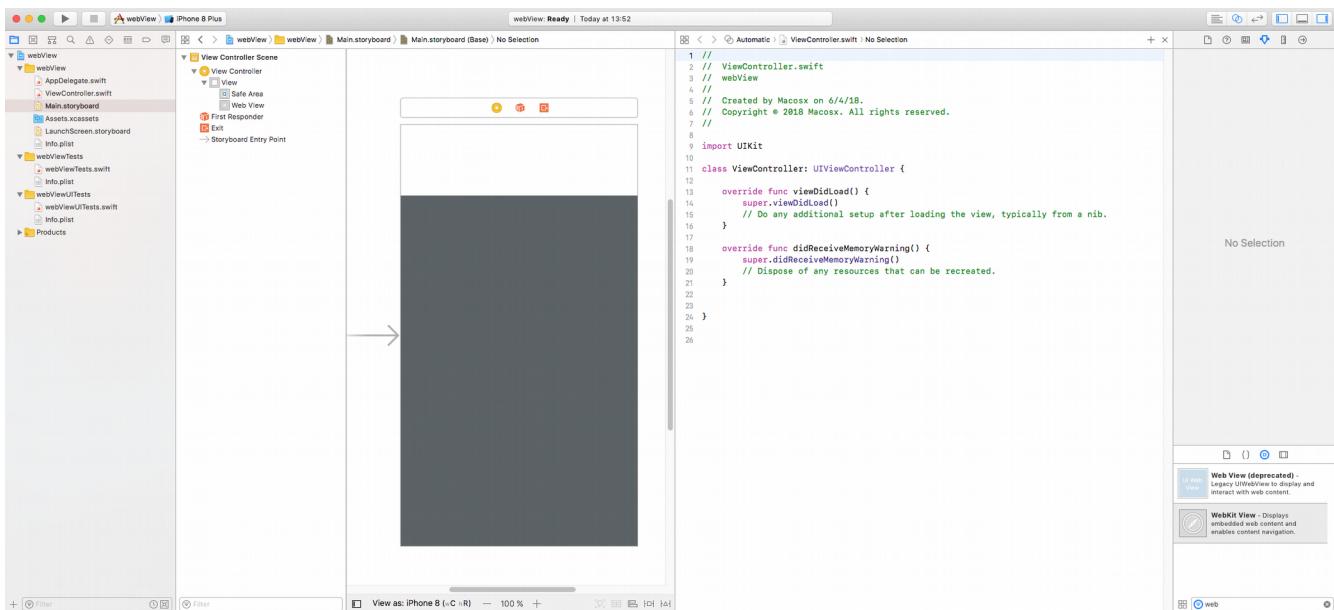


Utilización e implementación de la funcionalidad Webkit View

1. Una vez habiendo seleccionado la vista en la cual va a colocar el objeto WebKit View, dirígase a la barra de navegación situada en la esquina inferior derecha. Una vez ahí, escriba WebKit View o Web View.

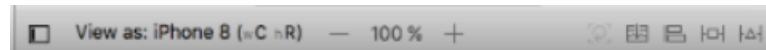


2. Al empezar a escribir en la barra de navegación del object utility, aparecerán dos opciones, Web View y WebKit View. La primera opción está obsoleta, y se desaconseja su uso, por lo que procederemos a usar la segunda. Arrastramos el objeto de tipo WebKit View a nuestra vista deseada del storyboard, y procedemos a posicionarlo donde creamos conveniente.



3. En nuestro caso, procedemos a dejar un espacio de margen, en el cual aparecerá el botón de búsqueda, y la barra de texto.

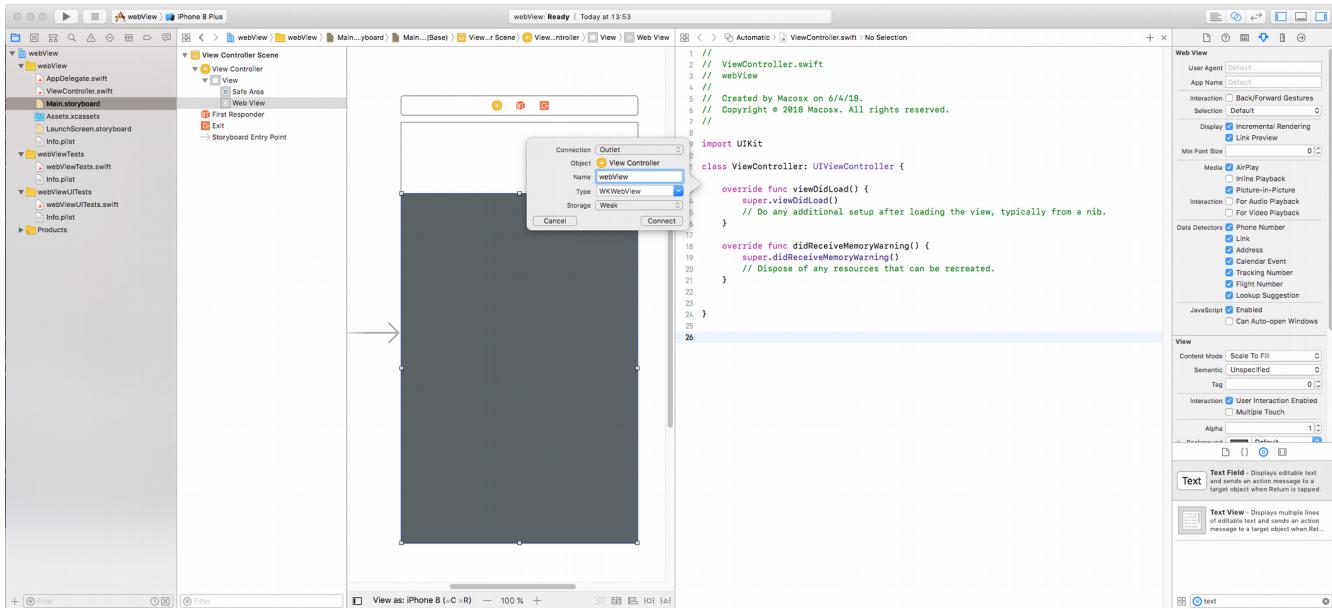
Nota: En este punto del tutorial, se obvió accidentalmente el hecho de hacer la vista de la aplicación de tipo responsive, es decir, que se reajustase a los diferentes formatos de los dispositivos que usan iOS. Para hacer esto, o bien seleccionamos antes de colocar cada elemento, los stack views correspondientes, o bien, una vez colocados todos los elementos en la vista, los vamos embebiendo en distintos stack views a través de la funcionalidad correspondiente del storyboard, a continuación añadiríamos las restricciones oportunas.



Ambas opciones, las podemos encontrar al pie de nuestro storyboard (primera y cuarta opcion correspondientemente de la imagen mostrada).

4. Enlazamos el WebKit View con nuestro código haciendo clic en la zona del canvas donde hemos colocado el objeto, y arrastramos con clic derecho hasta el código.

5. Le damos un nombre a la variable de unión, conservando los valores que ya vienen por defecto.



6. A continuación, haremos un import de nuestro webkit, poniendo: `import WebKit`

7. El botón debe haber quedado similar a lo siguiente:

`@IBOutlet weak var webView: WKWebView!`

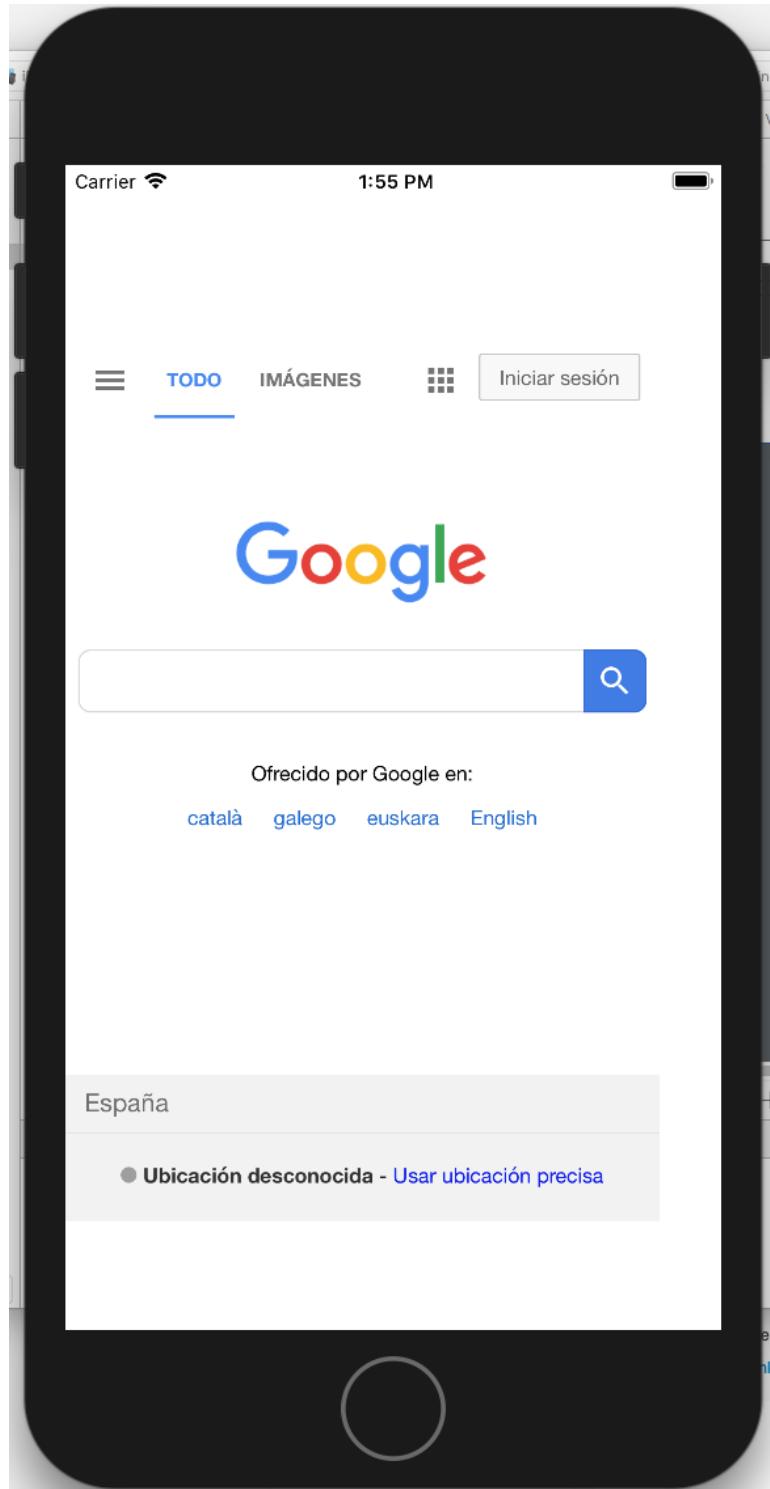
8. Añadiremos una pequeña funcionalidad de prueba, pues aún no disponemos de ninguna barra de búsqueda adicional en nuestra vista. Para hacer esto, en la función `viewDidLoad` colocaremos los siguientes:

```

override func viewDidLoad() {
    super.viewDidLoad()
    let url = URL(string: "https://www.google.com")
    let request = URLRequest(url: url!)
    webView.load(request)
}

```

Como podremos intuir, ahora, en nuestro WebKit View, aparecerá la página principal de Google. Hagamos una pausa en nuestro Tutorial y probemos que esto, es así. Para ello lanzemos la aplicación, nos deberá salir algo como lo siguiente:

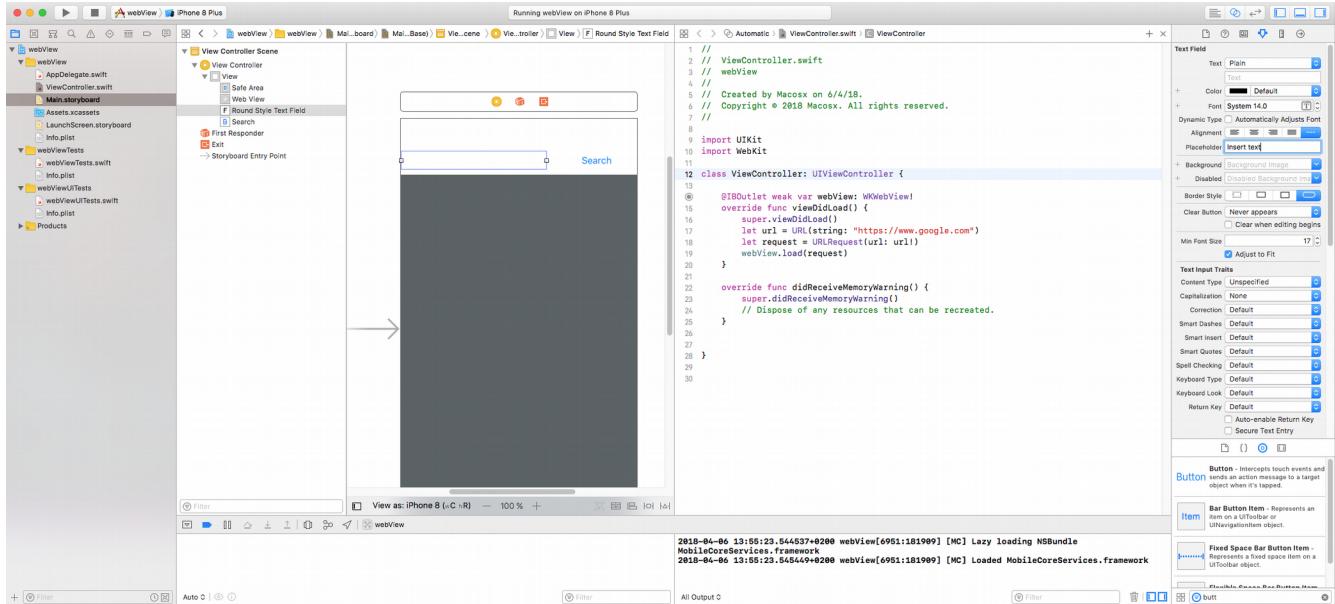


Como se puede apreciar, el WebKit View, no se acomoda del todo a el tamaño de nuestra pantalla, esto se debe a que como se ha indicado anteriormente, no se ha hecho uso del stack view, por lo que este fallo de visualización, se corregiría con el uso de un objeto de tipo stack view.

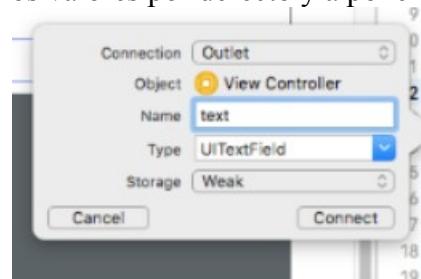
Utilización de WebKit View junto con una barra externa de búsqueda

WebKit View está muy bien, pero lo que realmente queremos, es no tener que depender de la barra del buscador correspondiente para hacer uso WebKit View. Para solventar esto, colocaremos tanto un campo en el poder escribir texto, como un botón con el que poder mandar a buscar dicho texto.

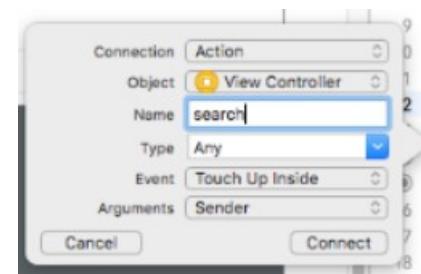
1. Buscamos dos objetos, uno de tipo TextField y otro de tipo Button. Arrastramos ambos, y los soltamos en la vista de nuestro canvas donde ya se encontraba el storyboard.



2. Una vez colocados ambos botones procedemos a su enlace con nuestro código. En caso del Text Field, procederemos a dejar todos los valores por defecto y a ponerle un nombre a la variable:



En el caso del botón, cambiaremos el tipo de conexión, de su valor por defecto a Action. Pondremos un nombre para la variable y dejaremos el resto de opciones por defecto.



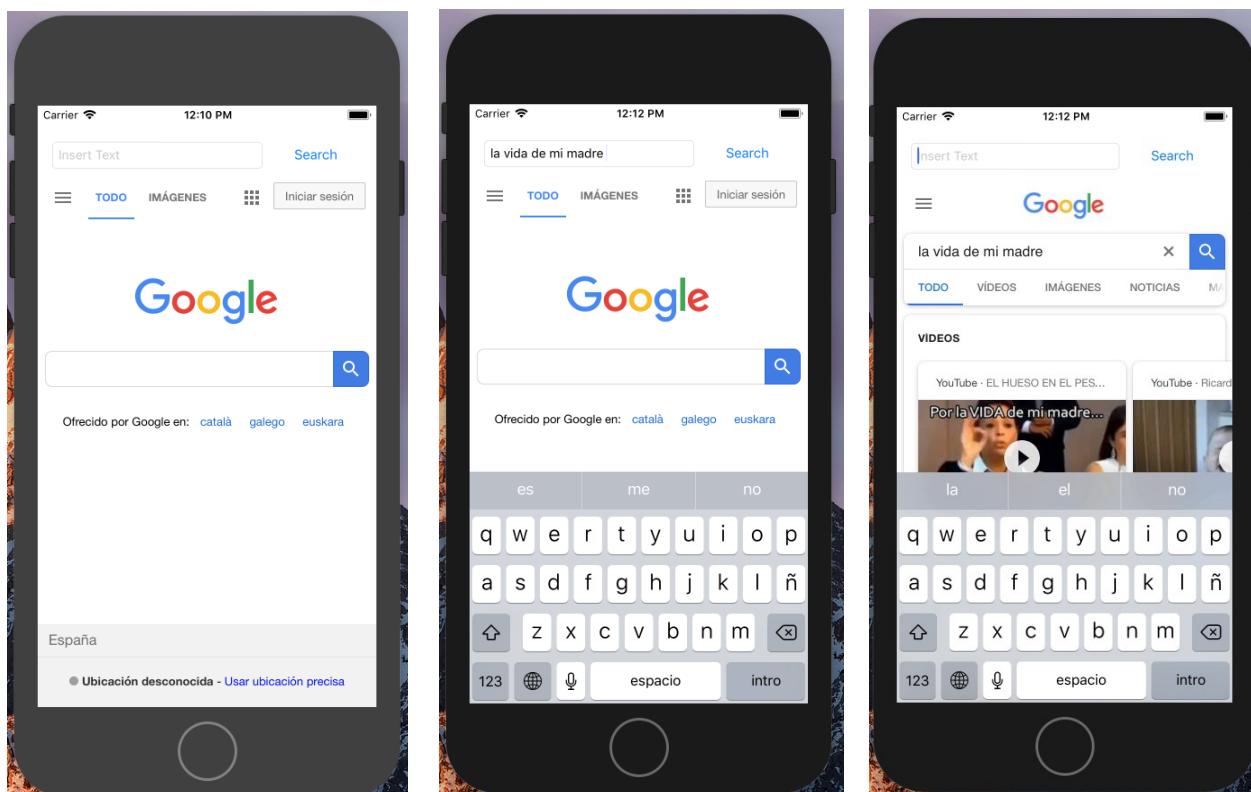
3. En la función de nuestro botón, estableceremos una variable txt en la que guardaremos el valor de el text field a cada momento que se establezca una acción en dicho botón. Comprobaremos así mismo, si este campo está vacío o no en el momento de la búsqueda. En caso de estarlo no hará nada, en caso de no estarlo, sobreescibirá una variable de clase llamada urlText, y la enviará al viewDidLoad(). El código quedaría de la siguiente manera:

```
14     var urlText: String? = "https://www.google.com"
15
16     @IBAction func search(_ sender: Any) {
17         let txt = text.text
18         if txt != nil {
19             urlText = txt!
20             viewDidLoad()
21             text.text = ""
22         }
23     }
24     @IBOutlet weak var text: UITextField!
25     @IBOutlet weak var webView: WKWebView!
26     override func viewDidLoad() {
27         super.viewDidLoad()
28         let url = URL(string: urlText!)
29         let request = URLRequest(url: url!)
30         webView.load(request)
31     }

```

Notese que la función viewDidLoad(), en la línea 28 hemos sustituido la url de Google por nuestra variable urlText.

Hagamos un alto de nuevo en el tutorial, y veamos nuestro progreso, para ello lanzaremos de nuevo nuestra aplicación.



Parseo y mejora de la búsqueda

Probamos a realizar una cuantas búsquedas en base al código que tenemos, y vemos que podemos introducir tanto urls como palabras, todo funciona aparentemente bien, hasta que introducimos una frase, en dicho momento, nuestra aplicación crashea debido a los huecos en blanco que hemos introducido. Esto es debido a que Google, parsea ciertos caracteres por seguridad, por lo que para ejecutar una búsqueda más robusta y eficiente, deberemos introducir los caracteres, tal y como lo haría Google. Para ello, haremos una serie de ligeras modificaciones en nuestro código.

1. Para realizar esta tarea nos apoyaremos en una variable auxiliar, y en una colección de tipo array, para lo cual, incluiremos import Foundation, y definiremos la nueva variable como definitiveText.
2. En la función search, crearemos un array que filtre el contenido de txt eliminando los espacios en blanco, y luego, introduciremos cada uno de esos elementos en definitiveText, separados por un +, esto es debido a que Google parsea de ese modo los espacios en blanco.
3. Finalmente, en urlText, introduciremos la url que Google utiliza de forma estándar en sus búsquedas, seguida de nuestra variable definitiveText. La url de Google es la siguiente:

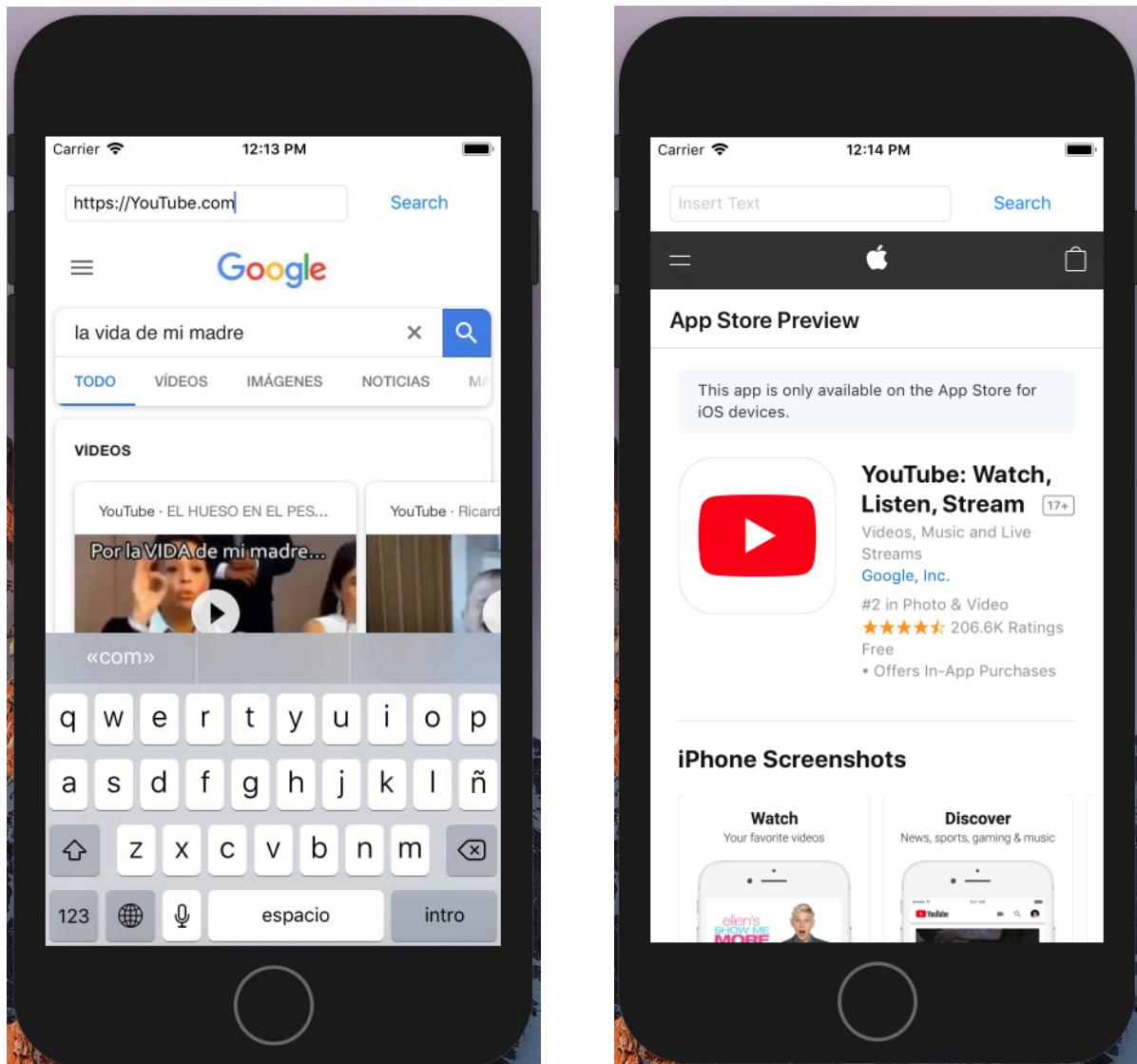
<https://www.google.es/search?q=>

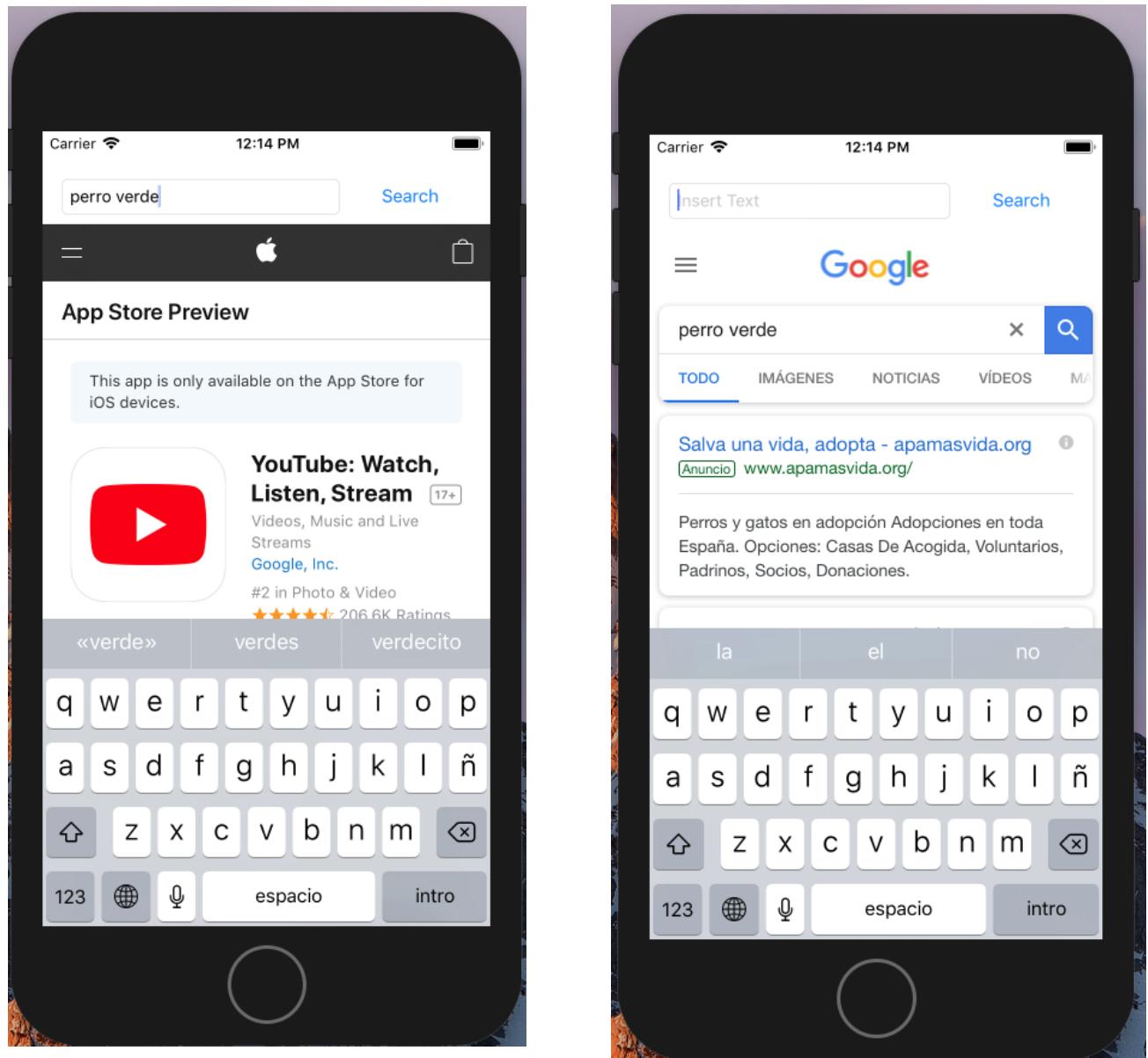
Finalmente el código quedaría de una forma similar a la siguiente:

```
9 import UIKit
10 import WebKit
11 import Foundation
12
13 class ViewController: UIViewController {
14
15     var urlText: String? = "https://www.google.com"
16     var definitiveText: String? = ""
17
18     @IBOutlet weak var webView: WKWebView!
19     @IBOutlet weak var text: UITextField!
20     @IBAction func search(_ sender: Any) {
21         let txt = text.text
22         let array : [String] = (txt?.components(separatedBy: " "))!
23         for elements in array{
24             definitiveText = definitiveText!+"+"+elements
25         }
26         if txt != nil {
27             urlText = "https://www.google.es/search?q="+definitiveText!;
28             viewDidLoad()
29             text.text = ""
30         }
31         definitiveText = ""
32     }
33     override func viewDidLoad() {
34         super.viewDidLoad()
35         if urlText != nil {
36             let url = URL(string: urlText!)
37             let request = URLRequest(url: url!)
38             webView.load(request)
39         }
40     }
41 }
```

Al igual que hemos parseado los espacios en blanco, podríamos parsear ciertos caracteres que google también parse como el carácter “?”, sin embargo, y aunque nosotros no lo hagamos, la aplicación no crasheará al introducir este tipo de caracteres, ya que Google lo hace igualmente de forma interna al mandarle nosotros la petición de búsqueda.

Llegados a este punto, es el momento de testear nuestra aplicación final. El resultado, es que podemos introducir frases, palabras y rutas, de tal forma que obtendremos la búsqueda que queremos.





De esta forma damos por concluido nuestro minitutorial.