DESARROLLO DE INTERFACES

TÉCNICO EN DESARROLLO DE APLICACIONES MULTIPLATAFORMA

Desarrollo de interfaces para iOS

Descarga e instalación del entorno Xcode

En este último capítulo, nos vamos a iniciar en el desarrollo para **iOS**, el **sistema operativo** de los **dispositivos** móviles de **Apple**. Se llevará a cabo una revisión de los aspectos clave relativos a la construcción del entorno necesario de desarrollo, se creará una primera aplicación, y, finalmente, analizaremos algunas pautas para el diseño de la interfaz de usuario para las aplicaciones iOS.

En primer lugar, desde el siguiente enlace se accede a la web de desarrollo de Apple en la que se encuentra toda la información oficial necesaria.

La **herramienta esencial** que se necesita para desarrollar en este sistema operativo es **Xcode**, el Entorno de Desarrollo Integrado (**IDE**) que nos proporciona Apple. Desde este IDE es posible **diseñar la interfaz** gráfica, **implementar** la aplicación, **probarla** de forma adecuada y, finalmente, **publicarla** en la App Store. Gracias a sus características, resulta un entorno de desarrollo excelente, puesto que desde una única herramienta será posible abordar todo el ciclo de vida de una aplicación.

Además de Xcode, también será necesario conocer el lenguaje de programación utilizado para el desarrollo de las aplicaciones en iOS, **Swift**.

Por lo tanto, en primer lugar, nos dispondremos a realizar la descarga de Xcode desde la App Store. Es imprescindible tener un equipo con macOS X para poder realizar el desarrollo de una aplicación para iOS.

Xcode

Primeros pasos

Tras realizar la instalación de Xcode en nuestro equipo, iniciamos la aplicación y, en la pantalla de inicio, nos aparecen varias opciones:

- abrir un proyecto anterior,
- crear un playground (permite al desarrollador **implementar** ciertos bloques de código **que no son aplicaciones** como tal),
- crear un nuevo proyecto en Xcode (es decir, crear una aplicación)
- y, finalmente, descargar un proyecto almacenado en un repositorio.

Para crear un nuevo proyecto desde cero, seleccionamos "**Create a new Xcode project**" .

A continuación, la herramienta permite configurar varios aspectos del desarrollo como el **sistema operativo** que se va a utilizar (**iOS, watchOS, tvOS, macOS**) o la **plantilla de diseño** para la aplicación. Como se puede deducir, Xcode permite el desarrollo de aplicaciones para todos los sistemas operativos de Apple, **para cada uno de los dispositivos** de la marca.

Tras la selección del sistema operativo (en nuestro caso iOS) para el desarrollo de aplicaciones móviles y una de las plantillas (plantilla en blanco con una sola pantalla), se pulsa el botón Next.

Configuración de un nuevo proyecto

A continuación, aparece la ventana de configuración del proyecto en la que indicaremos el nombre del mismo. En la lista de valores Team, es posible seleccionar una **cuenta de desarrollador necesaria** para poder **publicar** en la App Store las aplicaciones implementadas.

Otro de los datos clave que se indican en la ventana de configuración es Language, donde se seleccionará **Swift**, el lenguaje más moderno y potente para el desarrollo en Apple. **Objetive-C** es el primer lenguaje que se utilizó.

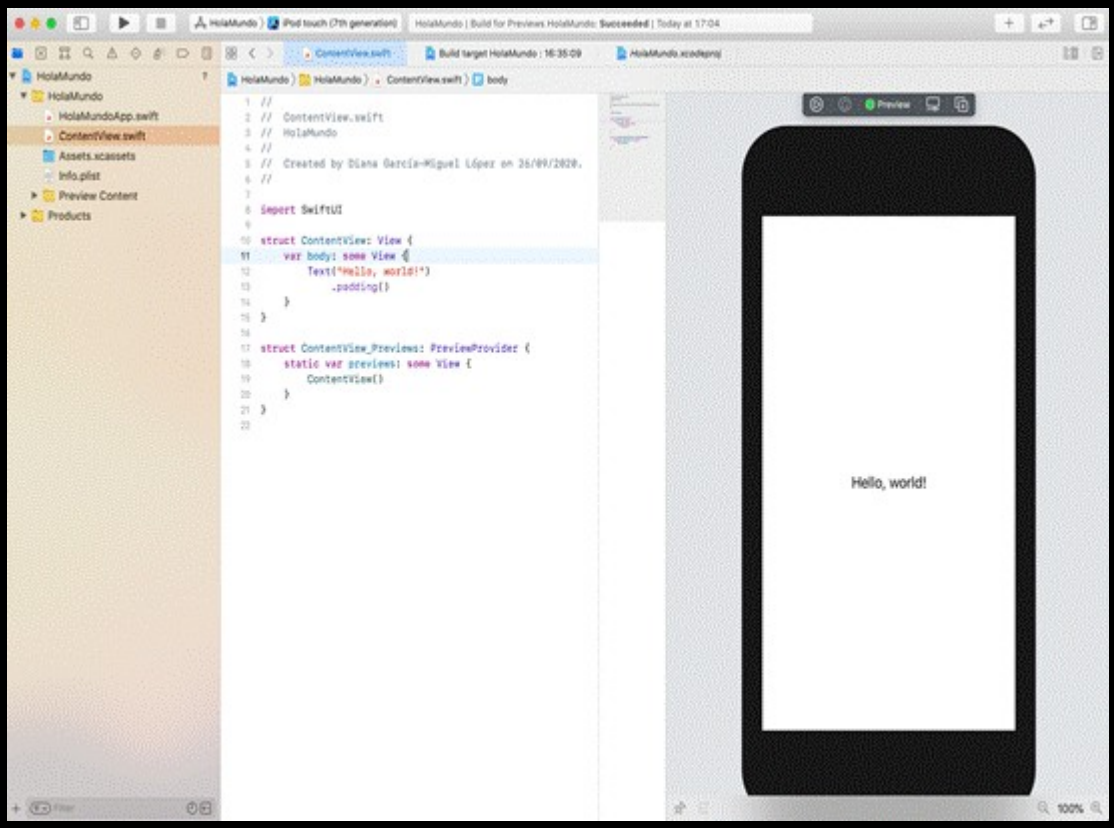
Finalmente, será necesario definir la interfaz de usuario (User Interface), que puede tomar los siguientes valores en función del tipo de aplicación desarrollada:

- Storyboard: tipo de **interfaz** utilizada **hasta 2019**.
- SwitfUI: **nueva librería** que incluye mejoras para el desarrollo de las **interfaces web**, y es la **más utilizada** en la actualidad para nuevos desarrollos.

Tras pulsar el botón Next, seleccionamos el directorio en el que va a estar ubicado el proyecto y, finalmente, se pulsa Create. Por defecto, se creará un nuevo "Hola Mundo".

Análisis del entorno

Menú de configuración (zona superior de Xcode)



Este entorno de diseño permite la creación de aplicaciones de forma sencilla, analizamos cada uno de los apartados de esta herramienta.

En primer lugar, en la siguiente figura, se muestra la zona de la herramienta que permite **ejecutar y detener** la aplicación, así como el **nombre del proyecto** y el tipo de **simulador** escogido para probar la aplicación que se está desarrollando.

En este último menú, mediante una lista de valores, también es posible escoger un **dispositivo físico** para probar la aplicación.

En la barra superior de la herramienta, se indica el **estado actual** de la aplicación, por ejemplo, *Running HelloWorldCode on iPhone 11*, es decir, nos indica que se está ejecutando el proyecto de nombre HelloWorldCode en un simulador de tipo iPhone 11.

Finalmente, en la parte superior, podemos observar el siguiente menú:

Otras funcionalidades + librería

Este menú es uno de los más **importantes**, puesto que será el que nos permita seleccionar **nuevos componentes** para ser colocados sobre la interfaz de la aplicación. El **acceso a la paleta** de componentes se realiza a través del botón con el s**ímbolo "+"**. Es el **botón de librería**.

La ventana de selección de elementos queda distribuida en varios tipos:

- **Componentes** gráficos.
- **Bloques** de código.
- **Imágenes**.
- Paleta de **colores** configurados para el proyecto.

El segundo botón nos permite comparar el código desarrollado con **versiones previas** que el desarrollador hubiese subido a algún repositorio. Finalmente, el resto de opciones de este menú permiten **personalizar la vista de la interfaz**, al igual que ocurría con Android Studio a través del botón Split.

Zona central de la aplicación

Como se puede observar en la imagen anterior de la interfaz, se muestra la interfaz de la aplicación Xcode que está estructurada en **tres grandes zonas de acción**.

Zona izquierda:

en esta zona, encontramos un completo catálogo de funcionalidades, como son:

- el acceso a los **repositorios**,
- un **buscador**,
- la zona en la que se muestran **advertencias**,
- análisis de **rendimiento**,
- **breakpoints** o puntos de ruptura,
- entre otras.

Una de las funciones más importantes de esta zona es Navigator, accesible a través del icono de carpeta, donde se muestran los **ficheros y directorios** de los que queda compuesto cada proyecto. Si pulsamos sobre el archivo principal de nuestro proyecto (que aparece en primer lugar), se muestra la **configuración principal** del proyecto donde aparecen algunos de los datos principales del mismo.

Zona de desarrollo:

Es la **zona central** de la interfaz de la herramienta y nos muestra el **código** de implementación.

Cuando se selecciona un fichero de un proyecto en desarrollo, esta zona queda subdividida en **dos partes**: una parte para el **código** y otra llamada **Preview**. En esta última, al pulsar el botón **Resume**, se muestra una **previsualización** de la aplicación, tomando como referencia el **simulador** previamente seleccionado.

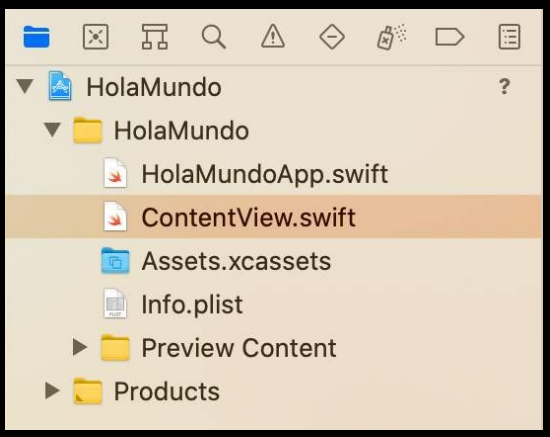
Zona derecha:

Como ocurre en otros entornos de desarrollo, esta zona solo se activa cuando **algún elemento** está **seleccionado**. En función del elemento del diseño escogido, se muestra un menú u otro, permitiendo **modificar las características** de los atributos. Por ejemplo, si se selecciona una etiqueta de texto, será posible modificar el String que se muestra, la fuente, o el tamaño.

Creación de la primera aplicación

En primer lugar, para la **creación del código** de la aplicación, debemos acceder al fichero ContentView.swift.

Ficheros de desarrollo de un proyecto



Para la creación de una primera aplicación, el código utilizado es el siguiente:

1. Se añade la siguiente instrucción que permite importar la librería SwiftUI.

Instrucción librería SwiftUI

```
import SwiftUI
```

2. La función ContentView define el **contenido** de la pantalla. En este caso, se incluye un componente de texto con el contenido “Hola mundo”.

Función ContentView. Define contenido de la pantalla

```
struct ContentView: View {
        var body: some View
        {
                Text ("Hola Mundo")
        }
}
```

3. Finalmente, es necesario introducir el siguiente fragmento de código para que se genere la previsualización que aparece a la derecha de la pantalla en el IDE.

Código lanzador de la previsualización que aparece a la derecha

```
struct ContentView_Previews: PreviewProvider {
        static var previews: some View {
                ContentView()
                }
}
```

Por último, para añadir alguno de los elementos y componentes del menú Librería, accesible a través del **botón '+'** que aparece en el menú superior en la derecha, basta con seleccionar uno y colocarlo en la posición deseada en la interfaz gráfica o en la zona de implementación.

Ampliación de contenido Swift. Manual de uso

https://www.apple.com/la/swift/

libro de Swift:

https://books.apple.com/us/book-series/swift-programming-series/id888896989

Guía del desarrollador de Swift

https://developer.apple.com/swift/

Apple, desde su sitio web, incorpora un amplio catálogo tanto de aplicaciones como de información sobre el lenguaje de programación que se utiliza para el desarrollo de aplicaciones tanto para iOS, iWatch o macOS, entre otras. Como vemos, aquí está la página donde tenemos todo lo relativo a Switch. Si vamos bajando hacia la parte de abajo, tenemos herramientas que han sido desarrolladas utilizando este nuevo lenguaje de programación que, como nos cuentan, se trata de una potente herramienta de desarrollo. Además, nos indica que los docentes están incluyendo Switch en sus planes de estudios.

Podemos ver que esto está desarrollado íntegramente utilizando Switch y el entorno de desarrollo propio de Apple, que ya hemos visto, Xcode. La parte de aquí abajo, que es la que nos queríamos centrar, incorpora los tres elementos principales para comenzar a trabajar con Switch. En este tema, en este capítulo, hemos creado unos primeros pasos para acercarnos en qué consiste, qué entornos de desarrollo necesitamos, qué lenguajes de programación para desarrollar con iOS.

Evidentemente, la creación de este tipo de aplicaciones requiere de tiempo. Aun así, si nos convertimos en expertos desarrolladores de este tipo de aplicaciones, dentro del entorno profesional, eso va a ser un punto a nuestro favor. Para empezar, centrándonos en estos tres elementos que van a ser, en primer lugar, la obra de referencia de Switch Programming Language. Esta obra, este iBooks, permite descargarse gratis desde este enlace. Vamos a necesitar de un dispositivo iOS para poder leer, pero como digo, este libro, vamos a pinchar aquí y vamos a ver la descarga.

Se trata de un libro muy completo, está en inglés, pero es muy completo y nos habla de todas las estructuras de control, todo lo que necesitamos saber para desarrollar utilizando Switch. Como vemos, en este caso, se abriría la aplicación de Switch y si accedemos aquí, vemos una vista previa. En este caso, como lo tengo descargado en el equipo, pulsaríamos en leer y ya nos aparecería todo el libro. De forma gratuita, tenemos un manual completo de un lenguaje de desarrollo. Es muy aconsejable revisar esta aplicación.

Además, tenemos en el siguiente punto Xcode. Como hemos visto, Xcode es el entorno de desarrollo. Para descargarlo, basta con pulsar a esta opción de aquí, descargar Xcode desde la App Store y accederíamos a ello. Finalmente, aquí tenemos el icono que nos habla de Switch. En este caso, si seleccionamos el enlace, nos describe las diferentes pautas. Es un resumen del libro que hablábamos en el primero de los enlaces.

Os aconsejamos revisar todo esto si de verdad queréis comenzar el desarrollo en interfaces multiplataforma, porque se trata de una aplicación de todo lo que se ha visto a lo largo de esta asignatura con las particularidades propias de cada lenguaje de programación y cada entorno de desarrollo, pero la base principal va a ser la misma, la creación de eventos, el tratamiento y captura de estos y la inserción de diferentes componentes empaquetados que nos permite la reutilización de los mismos para crear aplicaciones e interfaces adaptadas a cualquier tipo de entorno y de casuística.

Swift. Primeros pasos

Swift es el lenguaje de programación de Apple para crear aplicaciones para sus dispositivos (iOS, watchOS...). En estos apartados, veremos algunos de los puntos claves de desarrollo de este lenguaje que, al igual que en el resto de lenguajes, presenta su propia sintaxis y reglas.

Creación de variables

Una variable es un elemento utilizado para almacenar un valor que podrá ser modificado a lo largo de la ejecución, es decir, podrá tomar otros valores.

Pero se debe tener presente que si en primer lugar se **crea una variable** con una cadena de texto y luego se modifica su valor a un entero, el sistema nos devolverá un error, puesto que la variable será del tipo que se declaró la primera vez, y ya **no será posible cambiar el tipo de dato interno**. Por tanto, será necesario crear una nueva variable.

Para crear variables se utiliza la palabra var y, a continuación, se indica el nombre el nombre de la misma. Para la construcción del nombre, cada vez es más común observar ciertas convenciones sobre cómo deben ser definidos los nombres. Uno de los más utilizados es CamelCast, este indica que se tiene que diferenciar cada una de las palabras con una letra mayúscula, **empezando siempre con una letra minúscula**. Por ejemplo, un nombre válido sería primeraVariable. A continuación, se indica el valor de dicha variable.

Creación de variables

```
var primeraVariable="Hola Mundo"
```

Ahora bien, la creación de las **instancias** en Swift se realiza a través de struct, que equivalente a una **class de JavaScript**.

Por ejemplo, si queremos definir un **nuevo objeto** que contiene una cadena de texto, quedaría de la forma:

Creación de clases

```
struct nuevoDato {
var primeraVariable: String
}
```

Para la posterior instanciación de un objeto de la clase nuevo dato, se utilizará:

```
var nuevoDato1 = nuevoDato(primeraVariable :"Hola")
```

Comentarios

El uso de comentarios en programación no es un hecho trivial, es clave para construir programas que resulten más fáciles de mantener en el futuro.

/* */

(Barra inclinada adelante, asterisco de comienzo, asterisco de fin y barra inclinada adelante): Estos permiten insertar entre los valores de apertura y cierre todas las **líneas** de comentarios que sean necesarias. Este tipo de delimitadores son útiles cuando existen varias líneas de comentarios o se quiere comentar algún fragmento de código.

```
/*
Aquí
hay
varias lineas
*/
```

//

Este tipo de delimitadores se utilizan, habitualmente, para comentar **una línea** completa.

```
// Esto es un comentario de una línea
```

Botones

Los botones son elementos muy importantes en el desarrollo de interfaces, ya que permiten al usuario tener una interacción directa con la aplicación. Para añadir uno de estos elementos, basta con introducir el siguiente **fragmento de código** o seleccionarlo de la paleta de componentes.

Creación de un botón

```
Button(action: signIn) {
        Text("Sign In")
}
```

También es posible **desplegar la librería** a través del botón '+' y desde componentes gráficos, seleccionar el elemento, en este caso, un Button. Si pulsamos sobre él y lo **arrastramos** hasta la zona de desarrollo, se **añade el código**.

- En Action se indica qué va a hacer el botón cuando este sea pulsado,
- y en Content, el **texto** que aparecerá en el botón.

Inserción de botón en código

```
struct ContentView: View {
        var body: some View {
                Button(action:/*Action*/){
                        // Content
                }
}
```

Pautas para la creación de una interfaz en iOS

De nada sirve que el funcionamiento interno de una aplicación sea extraordinario, si el diseño tanto en aspecto como interacción de la interfaz de la aplicación no es bueno.

El desarrollo de las interfaces para aplicaciones en iOS se caracteriza por presentar diseños limpios y **minimalistas** que incluyen todo lo necesario, pero **sin sobrecargar** la interfaz.

Las características de diseño que diferencia este sistema de otras plataformas son:

- Claridad:
    - implica que el **texto** contenido en la aplicación debe ser **legible** en todos los tamaños,
    - los **iconos** deben ser **precisos**, es decir, no utilizan imágenes recargadas.
    - Además, la **distribución** en el espacio de los componentes, la **paleta de color** seleccionada, las **fuentes y los gráficos**, entre otros, resaltan el **contenido importante** de la aplicación y aportan interactividad al usuario.
- Adaptación al contenido:

    la **navegación** en la aplicación debe realizarse a través de un **movimiento fluido** que permite a los usuarios comprender e interactuar con el sistema con movimientos naturales. Por otro lado, el **contenido** debe ocupar **toda la pantalla** del dispositivo.
- Profundidad:

    la aplicación se implementa a través de varias **capas visuales** cuya transición entre capas aporta **sensación de movimiento** sobre la jerarquía de la aplicación, navegando “en profundidad” por el diseño.

En cuanto a los criterios de diseño establecidos por Apple para el diseño de las aplicaciones y garantizar de esta forma el impacto de sus desarrollos en el mercado, son:

- **Integridad estética**.
- **Consistencia**.
- **Manipulación** directa.
- **Retroalimentación**.
- **Metáforas**.
- **Control** de usuario.

Aspectos importantes de las interfaces en iOS

El nuevo sistema operativo **iOS 14** trae novedades muy importantes respecto al diseño de interfaces móviles. Desde septiembre de 2020 el diseño de la **pantalla de inicio**, los **mapas** y **mensajes** han sufrido un rediseño.

A partir de esta nueva versión la **pantalla de inicio puede personalizarse** con un mayor grado de detalle:

- se podrán **agrupar las aplicaciones** de distintas pantallas,
- o **paginar en App Library**.
- En la parte **superior izquierda** estará una agrupación de **sugerencias**,
- y en la **parte superior** derecha estarán las de **uso más reciente**.

Además, cualquier aplicación podrá ocupar más espacio en la pantalla mediante la creación de **Widgets del tamaño deseado** por el usuario.

Y por último otra de las novedades más importantes y que se deberá de tener en cuenta a la hora de diseñar interfaces para iOS 14, es que:

desde ahora se puede utilizar el **modo "**Picture in picture**":**

es decir, la posibilidad de tener **abierta una pantalla**, por ejemplo, reproducir un vídeo, mientras que se está **utilizando otra aplicación**.

Colores del sistema

iOS proporciona una gama de colores que se adaptan al diseño garantizando aspectos claves relativos a la **accesibilidad**, como son el **aumento del contraste** y la **disminución de la transparencia**. La elección de color de iOS permite que estos aseguren la satisfacción en el usuario, tanto cuando son mostrados de forma individual como combinándolos.

El **cuidado diseño en las interfaces** de las aplicaciones de Apple que garantizan la usabilidad de la misma son una de sus características de identidad. Por ello, desde el sitio oficial, indican los siguientes pilares clave para la elección del color:

- Usar el color con prudencia. Es aconsejable utilizarlo **solo** para resaltar **determinados elementos** sobre los que queremos llamar la atención del usuario.
- Utilizar colores complementarios en el diseño completo de la aplicación. Es decir, los colores escogidos deben funcionar bien de forma **independiente y combinados**.
- Seleccionar una paleta de colores concreta y limitada. Además, es aconsejable que esta selección vaya en consonancia con el diseño del lago de la aplicación.
- Escoger un color característico para toda la aplicación. Aunque se seleccione más de un color para el diseño de la aplicación, es conveniente que siempre exista **uno principal** que identifique a la aplicación. Por ejemplo, la app del BBVA basa su paleta de color en el azul, o la aplicación general Notes, escoge el color amarillo como central.
- Proporcionar dos colores para garantizar que la aplicación se vea bien en el entorno oscuro y el claro, que son los dos modos disponibles en iOS.

Los dispositivos iOS permiten seleccionar dos tipos de interfaces: una **oscura** y otra **clara**, por lo tanto, en función de cual se seleccione, será conveniente escoger unos colores u otros.

Por ejemplo, en la siguiente imagen, se muestran algunos colores en su versión clara y oscura:


Default
Accessible
Light
Dark
Name
API
R 0
G 64
B 221
R 64
G 156
B 255
Blue
systemBlue
R 36
G 138
B 61
R 48
G 219
B 91
Green
systemGreen
R 54
G 52
B 163
R 125
G 122
B 255
Indigo
systemIndigo
R 201
G 52
B 0
R 255
G 179
B 64
Orange
systemOrange
R 211
G 15
B 69
R 255
G 100
B 130
Pink
systemPink
R 137
G 68
B 171
R 218
G 143
B 255
Purple
systemPurple

Ejemplo de creación de un proyecto desde cero con Xcode

Se va a desarrollar una nueva aplicación básica utilizando la IDE Xcode y el lenguaje de programación Swift. Se muestra una cadena de texto con la frase “Soy tu primera aplicación”. La previsualización será como se muestra en la siguiente imagen:

En primer lugar, se va a crear un proyecto nuevo desde File, New, Project.

Ahora se selecciona la opción deseada para crear una nueva aplicación. Recordamos que sea en **iOS**, puesto que estamos desarrollando una **aplicación móvil**.

A continuación, en la ventana de configuración, se seleccionan los diferentes parámetros en base al diseño de la nueva aplicación. Ya hemos creado el proyecto, así que para **comenzar a programar**, basta con acceder al fichero **ContentView.swift**.

En el fichero **Content.View.swift** se añade la siguiente instrucción para importar la librería SwiftUI.

```
import SwiftUI
```

La **función clave** de todo desarrollo es ContentView, puesto que será esta la que **defina el comportamiento y los elementos** mostrados en la pantalla.

Si se desean **implementar otras funciones**, se pueden realizar externamente y ser **invocadas desde esta función**:

```
struct ContentView
```

Tal y como se indica en el enunciado, se añade una nueva etiqueta de texto.

```
Text ("Soy tu primera aplicación").padding();
```

El código completo se muestra a continuación:

```
import SwiftUI

struct ContentView: View {
    var body: some View {
        Text("Soy tu primera aplicación").padding()
    }
}

struct ContentView_Previews: PreviewProvider {
    static var previews: some View {
        ContentView()
    }
}
```

A tener en cuenta al diseñar una interfaz para una aplicación iOS

Hemos introducido brevemente el funcionamiento de Swift y el entorno de desarrollo Xcode. Existe mucha información en la red, pero se aconseja, sobre todo, acceder al siguiente enlace, a través de un sistema de MAC OS X. Se trata de un **manual completo de uso para Swift** desarrollado por Apple.

https://books.apple.com/es/book-series/swift-programming- series/id888896989

Se pide diseñar la interfaz para una nueva aplicación iOS que cumpla los criterios de diseño utilizando una herramienta de **prototipado** o diseño gráfico. En este caso, se solicita el diseño de una aplicación para visualizar una galería de imágenes.

La programación de la funcionalidad es clave, pero el diseño de la interfaz también juega un papel fundamental. En el caso de este tipo de dispositivos, se aconseja seguir los criterios establecidos por Apple para el diseño de las aplicaciones y garantizar, de esta forma, el impacto de sus aplicaciones en el mercado.

En esta interfaz, se muestra el resultado de una aplicación para iOS creada con Xcode y lenguaje de programación Swift. Contiene las imágenes publicadas por los usuarios de la aplicación que han asistido a un determinado evento.

Gracias al uso de una interfaz sencilla e intuitiva, el resultado es lo suficientemente accesible para garantizar su éxito en el mercado y cumplir los requisitos de prototipo de las aplicaciones iOS.

Conclusión

Hemos realizado una breve introducción al desarrollo en iOS, centrándonos en el entorno de desarrollo y en los primeros pasos a utilizar con el lenguaje de programación Swift.

La herramienta utilizada para el desarrollo de este tipo de aplicaciones es **Xcode**, el Entorno de Desarrollo Integrado (IDE) que nos proporciona Apple. Desde este IDE, es posible diseñar la interfaz gráfica, implementar la aplicación, probarla de forma adecuada y, finalmente, publicarla en la App Store. Gracias a sus características, resulta un entorno de desarrollo excelente, puesto que desde una única herramienta será posible abordar todo el ciclo de vida de una aplicación.

También, hemos aprendido que el desarrollo de las interfaces de las aplicaciones en iOS se caracteriza por presentar **diseños limpios y minimalistas** que incluyen todo lo necesario, pero sin sobrecargar la interfaz. Las características de diseño que diferencia este sistema de otras plataformas son:

- **claridad,**
- **adaptación al contenido**
- **y profundidad**.

(CAP nemotécnica)

En cuanto a los criterios de diseño establecidos por Apple para el diseño de las aplicaciones y garantizar el impacto de sus desarrollos en el mercado son:

- integridad estética,
- consistencia,
- manipulación directa,
- retroalimentación,
- metáforas
- y control de usuario.

https://www.apple.com/es/swift/

https://developer.apple.com/swift/