{1.3}

Aprendiendo Swift 3

El lenguaje abierto de Apple de propósito general

iOS, macOS, tvOS, watchOS, Linux, Android, Windows, FreeBSD, Raspberry Pi...

Preparado para Xcode 8 y Swift Playgrounds para iPad

Julio César Fernández Muñoz

Aprendiendo Swift 3

El lenguaje abierto de Apple de propósito general
iOS, macOS, tvOS, watchOS, Linux, Android, Windows, FreeBSD, Raspberry Pi...

Primera publicación: Noviembre 2016

Publicado por: Gabhel Studios S.L.

Gabhel Studios S.L.
C/Anabel Segura, 10 3º Planta - Edificio Fiteni IX
28108 Alcobendas (Madrid)
España

ISBN-13: 978-84-617-6587-4

www.gabhel.es
www.applecoding.com

Imagen de la portada: Gabhel Studios S.L.
Diseño y maquetación: Marta Cañadas

Aprendiendo Swift 3

El lenguaje abierto de Apple de propósito general

iOS, macOS, tvOS, watchOS, Linux, Android,
Windows, FreeBSD, Raspberry Pi...

Gabhel Studios

Contenido

1 - INTRODUCCIÓN

1.1 Sobre el autor

Julio César Fernández Muñoz

Experto en tecnología y programación, nacido en 1975, ha sido testigo de primer orden de la evolución tecnológica de finales del siglo XX y comienzos del XXI. Emprendedor, co-fundador y director técnico de Gabhel Studios (estudio independiente de creación de apps móviles y videojuegos, así como formación en nuevas tecnologías), desarrollador, consultor de apps y videojuegos, modelos de negocio y conferenciante sobre nuevas tecnologías y el mundo profesional y de consumo. Ingeniero de sonido y vídeo, actor de doblaje, locutor y experto en comunicación a audiencias, editor jefe de la web AppleCoding y responsable de su podcast.

Su padre, profesional pionero del mundo de la tecnología para el sector bancario, le transmitió su pasión por la tecnología. Con solo 8 años su padre trajo a casa el primer ordenador que caería en sus manos, un Sinclair ZX Spectrum de 48K, donde empezó a adentrarse en el mundo tecnológico. Con 10 años, a finales de 1985, estrenó un Amstrad CPC 6128 donde jugando a un videojuego pulsó la tecla ESC sin querer y el juego se paró mostrando un mensaje: *BREAK*. Volvió a pulsar y el ordenador le dio un error: “Break in line 630”, así como un cursor que esperaba para empezar a escribir debajo del mensaje: "Ready". Buscó en el manual y leyó que aquello era un mensaje de bloqueo del código del programa. “Escriba list para ver las líneas del código”. Lo hizo y cuando el código en BASIC detrás del juego que había estado probando empezó a desfilar ante sus ojos, fue el momento en que supo que quería dedicarse a ese apasionante mundo. A crear mediante instrucciones.

Tras estudiar informática durante varios años, compaginado con ser el responsable del mantenimiento de los ordenadores de familiares y amigos, e instalar tantos Windows que no podría ni contarlo, comenzó su carrera profesional como desarrollador de aplicaciones en entornos Oracle Developer con PL/SQL. Más de 10 años de experiencia de carrera profesional donde llegó a convertirse en Director de Equipos de Desarrollo así como Administrador de Base de Datos y Sistemas y responsable de las aplicaciones de gestión para el Personal Aeronáutico del Ministerio de Fomento del Gobierno de España. Fue pionero en Administración Electrónica, Seguridad y firma digital, siendo además experto en consultoría de procesos y adaptación de flujos de trabajo en soluciones a medida de desarrollo.

En paralelo a su carrera, tuvo diversos contactos con ordenadores Apple durante su juventud y fue el lanzamiento del iMac en 1998 lo que le hizo interesarse por la compañía de Cupertino y comenzar a seguir su devenir con interés. En 2004 se hizo con su primer dispositivo Apple: un iPod, que provocaba las miradas de atención de la gente y el interés en un momento donde aquellos dispositivos eran casi unos desconocidos en España. No solo por los característicos auriculares blancos, si no que cuando lo sacaba del bolsillo para pasar de canción o subir/bajar el volumen la gente no entendía qué era aquello tan blanco y compacto. En 2007 se enamoró del concepto

del iPhone y adquirió un iPod Touch de primera generación en cuanto se lanzó en España. Siguió y aprendió toda la tecnología relacionada con Apple hasta que en 2010, compaginado con su trabajo, comenzó a colaborar como periodista especializado en tecnología Apple para el medio AppleWeblog del grupo editorial Hipertextual. 18 meses de trabajo intenso con más de 10 artículos semanales, pruebas de productos e invitaciones de la propia a Apple a eventos de prensa exclusivos, que cristalizaron en 2011 cuando decidió, junto a su mujer, fundar lo que hoy se ha convertido en Gabhel Studios.

Y fue cuando Apple presentó Swift en 2014, que Gabhel Studios fundó AppleCoding.com, y se comenzó a fraguar lo que hoy día es el libro que estás leyendo, resultado de años de trabajo y conocimiento, así como meses y meses de desarrollo en proyectos reales que han creado un currículum de enseñanza cercano y ameno del que ya es uno de los lenguajes de programación de mayor repercusión y proyección de futuro. Apple Coding, además, se ha convertido en referencia del mundo del desarrollo, con una enorme cantidad de material didáctico, noticias, análisis y contenidos, una importante presencia en redes sociales y un podcast de gran difusión que ha sido destacado por Apple en varios países en el iTunes Store. Una media de 1.000 visitas al día a la web, cientos de suscripciones al podcast, miles de oyentes semanales del podcast y ser número uno en ventas en las listas generales en varios países hispanos de ediciones anteriores de este libro, avalan el éxito de este trabajo. Trabajo destacado en medios de la reputación de Applesfera.com, quien reseñó “Aprendiendo Swift 2” y que ahora cristaliza aun más con una gran y renovada revisión "Aprendiendo Swift 3".

Hoy, Julio César Fernández, es uno de los más conocidos expertos en entornos Apple, de su desarrollo y del mundo de la tecnología en habla hispana, colaborador habitual de diversos medios del grupo editorial Difoosion u otros como FSGamer (del grupo El Correo), además de formador, conferenciante y podcaster con miles de seguidores que lo escuchan semana a semana para estar al día de todo lo que sucede en el mundo del desarrollo e incluso gente que, sin ser desarrollador, se acerca al podcast para aprender mejor cómo funciona internamente la tecnología que les apasiona.

Tras más de 30 años en el mundo tecnológico, la experiencia y recorrido profesional y personal del autor avalan una obra como esta, primera de un conjunto de libros que verán la luz durante 2016 y 2017.

1.2 Estructura del libro

Bienvenidos a “Aprendiendo Swift 3”, la tercera edición del primero de una serie de libros editados por la web applecoding.com, sobre el desarrollo en entornos Apple y que permite servir de guía de aprendizaje y referencia para todo tipo de perfiles. Desde el profano en el mundo de la programación y que quiere dedicarse a esto o simplemente aprender, pasando por gente de diferente nivel en otros lenguajes, tanto Objective-C como lenguajes de scripting como el famoso Javascript. Incluso gente que ya sepa Swift pero que quiera tener un libro de referencia del lenguaje o actualizarse a la nueva especificación 3 del mismo, que abarca este libro (más concretamente, la versión 3.0 que acompaña a Xcode 8.0).

En el presente libro se abarcará desde la historia del desarrollo en entornos Apple, pasando por una introducción a la programación y la orientación a objetos, el propio lenguaje Swift, las herramientas para trabajar con él y realizar prototipos, así como cada uno de los elementos imprescindibles del lenguaje.

El libro se dividirá en varios bloques esenciales que permitan seguirlo de una manera más fácil. Primero, en esta introducción haremos un repaso a tres niveles: este que están leyendo, la historia de los 40 años de desarrollo en entornos Apple y por último sobre la breve historia de Swift y por qué Apple decidió lanzar un nuevo lenguaje cuando ya tenía uno (Objective-C) muy asentado y de larga trayectoria. Además, exploraremos la evolución del lenguaje, cómo ha ido perfeccionándose y por qué, así como entender los naturales pasos que está dando y cuál es su prometedor presente y futuro.

Tras esto, nos adentraremos en Swift, ¿qué es el lenguaje? Para, antes de nada, aprender cómo vamos a trabajar con todos los ejemplos y código del libro: con el uso de los playground, un formato de proyecto básico que nos permitirá ver y evaluar todo lo que hacemos. Sin olvidarnos de la gran novedad de Swift 3: la app para iPad Swift Playground, que aprendemos a cómo manejar y sacarle el mayor partido.

Concluida esta parte y conocidas las herramientas de trabajo, nos remangamos porque comenzamos con el Swift Básico, lo que todo buen programador ha de conocer al dedillo para trabajar con el lenguaje: tipos, operadores, cadenas, colecciones… lo más importante y básico de este, imprescindible para empezar a crear y entender Swift y, lo más importante, para qué y cómo usarlo.

Luego, pasamos al nivel intermedio, donde podremos explotar todos los conceptos básicos aprendidos, conociendo los esquemas del lenguaje, sus estructuras y ahondando un poco más en algunos aspectos que vimos en la parte básica, pero en mayor profundidad para aprovechar mejor el potencial del lenguaje y sus componentes.

Tras superar este capítulo tendremos todas las herramientas en nuestra mano para ser los más preparados desarrolladores en el nuevo lenguaje de Apple, pero eso no es

suficiente porque para ser un Maestro necesitamos ir más allá. Y para eso está el apartado Swift Avanzado donde conoceremos en detalle los elementos más complejos del lenguaje y que nos permitirán sacar el máximo provecho del mismo y entender cómo funciona.

El siguiente capítulo es un obligado: Objective-C sigue formando parte del ecosistema de Apple, y muchos elementos de las librerías que usamos cada día en Cocoa o SpriteKit (para construir apps o juegos, respectivamente), aun siguen programadas en Objective-C. Por lo tanto, es imprescindible que aprendamos cómo interoperar entre los dos lenguajes: cómo mezclar ambos en un solo proyecto, cómo funciona Cocoa… las claves para entender cómo sacar el mejor provecho de ambos.

Tras esto, volvemos al inicio pero ahora con todo lo que ya hemos aprendido. De esta forma, retomamos los playground y aprendemos a cómo sacar el máximo provecho de los mismos y a construir prototipos de apps o de juegos dentro del mismo entorno. También a cómo crear auténticos libros interactivos de código con documentación enriquecida y personalizando la forma en que se muestran nuestros datos.

Por último, las referencias, de consulta imprescindible. Porque es muy complicado tenerlo todo en la cabeza, y esas guías nos ayudan a llegar al grado máximo. Repasamos la fundación del lenguaje para que entendamos su estructura y cómo Swift se construye sobre sí mismo, luego las necesarias referencias y guías de estilo o plantillas y, por supuesto, cómo documentar nuestro código para no tener ningún problema en nuestro camino.

El libro quiere ser una guía de referencia y aprendizaje clara y sencilla, que permita no solo ser un libro de enseñanza si no también de referencia del lenguaje y, por qué no, de la forma de enfrentarnos a los problemas más comunes que se nos pueden presentar con el mismo y cómo solventarlos.

El secreto: explicarlo todo de una manera cercana, evitando los tecnicismos y no dando nada por supuesto. Explicado como te lo explicaría un buen profesor universitario, de una manera cercana para que las explicaciones traspasen la complejidad del tema y lleguen en su contenido de una manera especial y simple de entender: hacer fácil lo difícil.

Requisitos

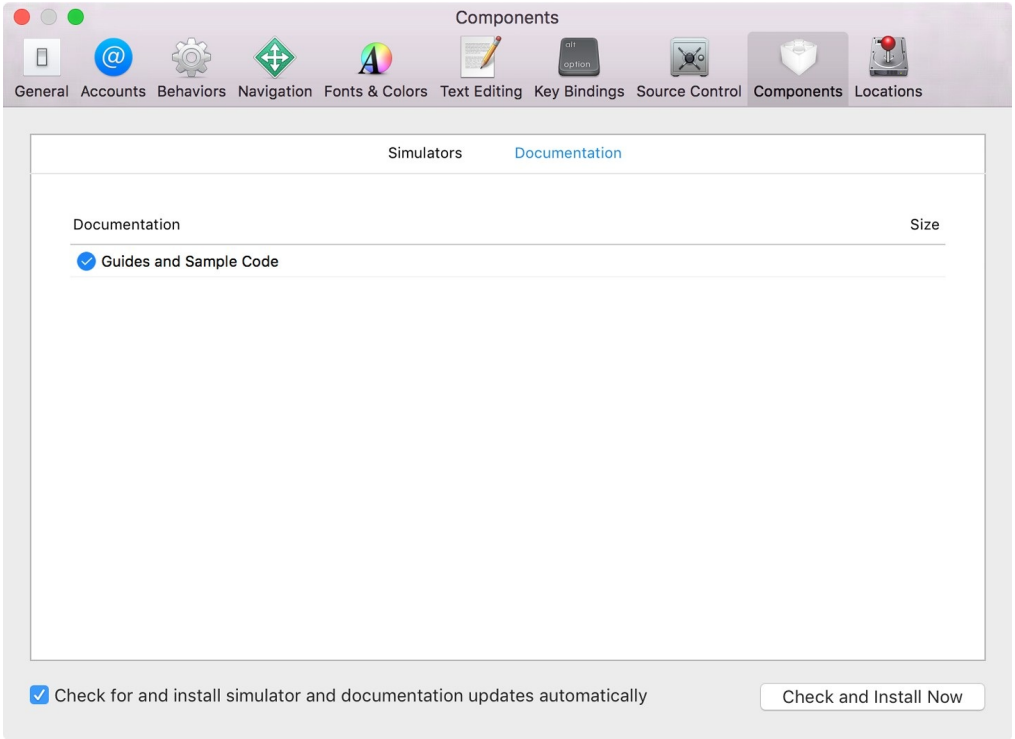
Para trabajar con Swift tenemos varias opciones disponibles y no todas pasan por tener un ordenador Mac de Apple, como la mayoría podría pensar.

La primera y más obvia, no obstante, sí es esa: disponer de un ordenador Mac con el sistema operativo OS X 10.11.5 El Capitan o macOS 10.12 Sierra (o cualquier versión superior, en caso de existir). Lo segundo es tener el software para desarrollo Xcode de la propia Apple, en su versión 8 o superior. El libro cubre la especificación 3 lanzada por Apple el mes de septiembre de 2016.

Como IDE (o entorno de desarrollo interactivo) Xcode ofrece todas las herramientas necesarias para trabajar, desde el editor de código hasta constructores de interfaces, de escenas de juegos, de partículas, control de código… aunque podamos usar alguna herramienta externa de ayuda en algún momento, Xcode será lo único que realmente necesitaremos instalar en nuestro Mac y solo necesitaremos en nuestro devenir (salvo casos puntuales) hacer uso del tipo de proyecto conocido como Playground que veremos en su momento.

La instalación de Xcode no supone mayor desafío para un nuevo usuario que el de acceder a la App Store del Mac, bien en el icono del dock o a través del menú y luego ir a App Store. Ponemos Xcode en el buscador situado en la parte superior derecha y el primer resultado será la herramienta de Apple con su característico martillo. Pulsamos en "Obtener" y se descargará e instalará. El proceso puede ser lento en función de la máquina que tengamos, pero una vez terminado no hay que hacer nada más que abrirlo y comenzar a trabajar.

Un detalle, como pequeño consejo personal: una vez abierto es recomendable ir al menú Xcode y luego a Preferences. Pulsamos en Components y es conveniente que verifiquemos que las guías y código de ejemplo (Guides and Sample Code) están instaladas correctamente. No nos vendrá mal porque además aprenderemos a cómo acceder a estas guías y toda la documentación de ayuda que Apple proporciona, en cualquier momento.

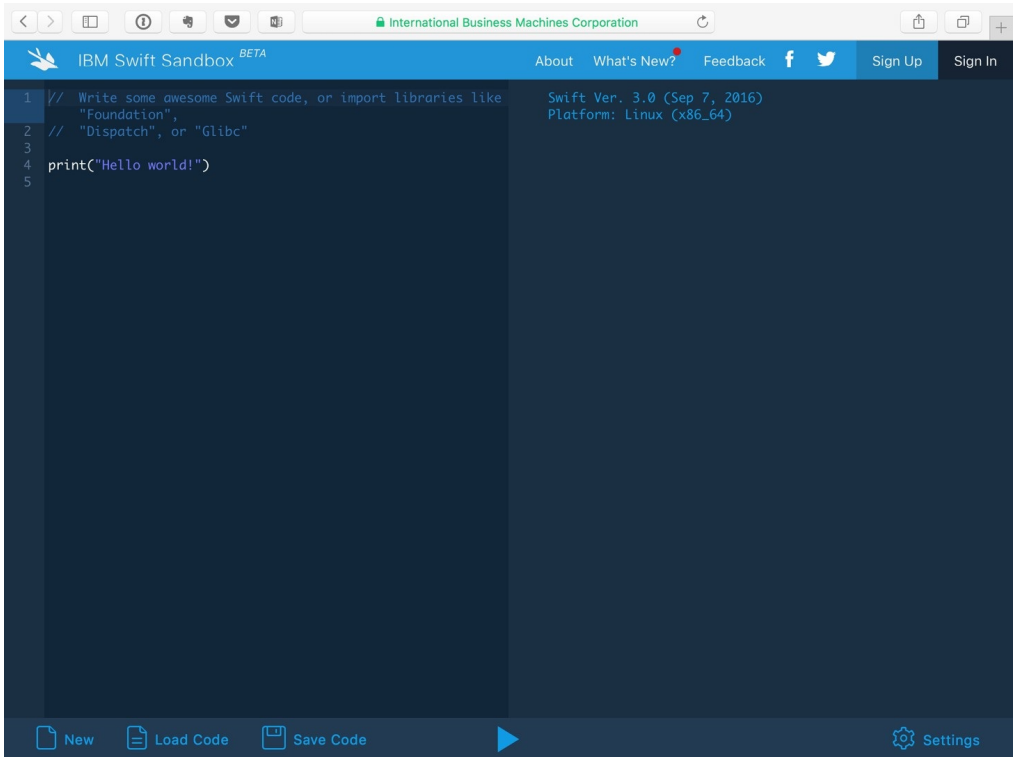


Pero si no disponemos de un ordenador Mac, podemos estudiar Swift y aprenderlo usando cualquier ordenador que utilice el sistema operativo Linux Ubuntu en sus versiones 14.04, 15.10 y 16.04. En dicho caso, es recomendable tener un conocimiento informático más alto para poder configurar Ubuntu y se recomienda usar el editor de

texto enfocado en desarrollo, Atom, de la web GitHub (que además es código abierto y gratuito).

Si estáis interesados, en el último apartado del libro "Referencias", tenéis un capítulo que os guía sobre cómo utilizar Swift en Linux y la estructura de sus proyectos, así como el uso del compildor.

También podemos usar la web con la página: IBM Swift Sandbox, un entorno web parecido al tipo de proyecto playground que usaremos durante la mayor parte del libro (aunque más limitado, obviamente) y que os permitirá trabajar incluso desde un iPad o un iPhone (o cualquier otro dispositivo móvil). Simplemente tenéis que acceder en el navegador a la URL https://swiftlang.ng.bluemix.net/ y comenzar a trabajar.

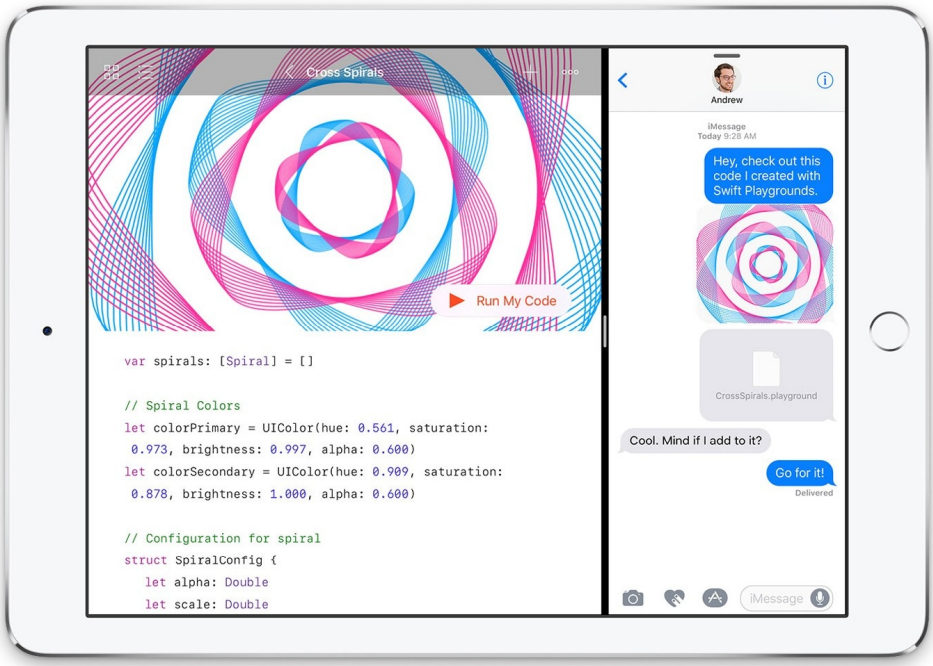


En la parte izquierda se escribe el código. Cuando hayamos terminado y queramos probar, en la parte inferior pulsamos la tecla Play o la combinación CMD+S (CTRL+S si estamos en otro sistema) para ejecutar y vemos qué hace nuestro código. En el desplegable inferior Load Code existen varios interesantes ejemplos de código que podemos probar. Podremos grabar nuestro código, cargarlo y en la parte de Settings podremos incluso cambiar opciones de interfaz o la versión de Swift con que trabajemos (que en principio no es necesario cambiar si esta es Swift 3.0 como en la captura).

Sí debemos tener presente un detalle: es posible que algunos ejemplos de este libro no funcionen al 100% en Linux y en este entorno, debido a que Swift Open Source no incluye las librerías de Cocoa (toda la parte que permite construir apps) por lo que si algún ejemplo hace uso de elementos de Cocoa y UIKit como etiquetas, campos y demás, estos ejemplos darán error en el entorno web de Swift Sandbox. Es importante

que entendamos este detalle. De hecho, una instrucción que repetiremos mucho en algunos ejemplos como `import UIKit`, no funcionará en este entorno web y tendríamos que sustituirla por `import Foundation`.

Por último, y no menos importante, otra forma de usar este libro y aprender Swift (en este caso sin limitación alguna) es con la app para iPad, Swift Playgrounds. Una app que podemos descargar gratuitamente del App Store con nuestro iPad, compatible con todas aquellas tabletas de Apple que cuentan con procesador de 64 bits: iPad Mini Retina 2 o superior, iPad Air 1 y 2, así como los modelos de iPad Pro de 9,7" y 12,9". Swift Playgrounds es una herramienta que Apple promociona como herramienta de aprendizaje del lenguaje enfocada en niños (que lo es), pero en realidad esta app va mucho más allá. Es un Playground completo, con todas las funciones que tiene el entorno con el que vamos a trabajar en Xcode 8. Por lo que, si resulta más cómodo o práctico, puede trabajarse con esta app sin ningún tipo de limitación en cuanto al contenido del libro. Solo hay que abrirla y darle a New para que nos de un entorno vacío donde trabajar en Swift 3.



Antes de empezar con el lenguaje, en la parte dedicada a los Playgrounds, hay un capítulo completo dedicado a cómo funciona Swift Playgrounds y cómo usarlo con el libro: una forma diferente y práctica de aprender el lenguaje sin limitación alguna ya que todas las librerías que forman parte de Cocoa están presentes en esta app de iPad y también aquellas que nos permiten hacer juegos como SpriteKit o SceneKit (las lecciones para niños están hechas en SceneKit, por ejemplo). Y lo mejor de todo es que a través de iCloud Drive, todo el código que uséis estará disponible en todos vuestros dispositivos y ordenadores. Todo lo que se cree en Xcode podrá verse en Swift Playgrounds en el iPad y viceversa. Podéis empezar a trabajar con el iPad, pasar al Xcode, volver al iPad… vuestros proyectos estarán en la nube y replicados en cada dispositivo. Una nueva ventaja y añadido para trabajar con el libro y aprender Swift

como un experto.

Por lo tanto, ya no hay excusa: podemos usar no solo un Mac, si no la web, Linux o la app para iPad Swift Playgrounds. Así que empezamos con la faena. Antes de nada agradecerte que hayas decidido aprender con este libro que se ha hecho con mucho esfuerzo, durante muchos meses, reflejando la experiencia real en este nuevo lenguaje y en proyectos de apps y juegos. Un libro que supone un enorme trabajo de recopilación, aprendizaje, pruebas, prácticas, proyectos y mucha ilusión.

Gracias y esperamos que cuando llegues al final sientas como Neo al abrir los ojos, mirar a Morfeo y decir aquello de: “Ya sé Kung-Fú”. Ahora, demuéstralo.

1.3 Apple y el desarrollo, 40 años de historia

El desarrollo es el componente imprescindible de cualquier dispositivo electrónico. Absolutamente todos los aparatos electrónicos que manejamos tienen una premisa: electricidad que pasa o no por un sitio, que combinado crea un conjunto de funciones o instrucciones que desencadenan en que ese “algo” haga “cosas”. Un televisor muestra la imagen porque sus circuitos están diseñados con una serie de instrucciones para generar el resultado que vemos. Al igual que una nevera, un microondas o un coche tienen instrucciones que les dictan qué hacer cuando se invocan determinadas acciones del mismo. La pantalla de nuestro smartphone no es más que una combinación de puntos o píxeles que se colorean en función de unas instrucciones. Y dichas instrucciones se componen, en su forma más básica, de unos y ceros: pasa o no pasa la electricidad. Lo que dicta si pasa o no es lo que se define como programación.

Los ordenadores como tal, son máquinas que tienen una función más amplia. Son dispositivos programables multifunción (el propio vocablo ordenadores implica la necesidad de indicar órdenes al mismo para que funcionen). Y si usamos la expresión anglosajona computadoras, vemos claramente como el propio nombre indica su función principal: computar o calcular. Son como una navaja suiza que puede ser usada para multitud de cosas, con la ventaja que podemos crear nuestras propias funciones para dicha navaja. Y las herramientas para hacer esto son los lenguajes de programación. Curiosamente, el lenguaje de programación que nos permite programar a su vez está hecho en otro lenguaje (o incluso a veces en el mismo). Y todo hasta llegar a los unos y ceros, es programación de instrucciones para permitir “hacer cosas”.

Cada nivel que recorremos es lo que se denomina una capa de abstracción: un lenguaje convierte sus instrucciones en otro lenguaje más simple como el código máquina (que es el que usan los procesadores del ordenador) y ese código máquina es traducido a unos y ceros por dicho procesador. Si es un lenguaje de script, como el famoso javascript en una página web, ese lenguaje se interpreta por el navegador y se convierte en instrucciones del lenguaje sobre el que está programado, que a su vez se convierte en código máquina y luego en unos y ceros. Al final, todo acaba siendo pasa o no pasa la luz: 1 o 0.

Habría que remontarse hasta el año 1843 para rememorar la primera aparición de un análisis y un algoritmo de programación como tal, creado por una matemática, que históricamente es considerada la primera programadora de la historia: Augusta Ada Byron, hija del famoso escritor Lord Byron y más conocida como la Condesa de Lovelace. Ella fue la primera persona que creó un algoritmo de programación cuyo fin era calcular los números de Bernoulli. Algoritmo que nunca llegó a probarse pues la máquina en que debía ser ejecutado, la máquina analítica de Charles Babbage, no llegó a construirse. Pero Lady Lovelace tuvo una gran influencia con su trabajo en los años posteriores, dado que en este se sugerían interesantes conceptos analíticos y de algoritmos matemáticos aplicables a la programación, así como el uso de tarjetas perforadas como método de entrada de estas instrucciones.

No sería hasta años después cuando el matemático Alan Turing, uno de los padres de las denominadas ciencias de la computación (ciencias que estudian los algoritmos que dan instrucciones a una máquina para funcionar, base de todos los lenguajes de programación) definiera los conceptos del algoritmo gracias a la máquina de Turing. Una máquina que a través de una cinta recibía instrucciones de manera secuencial y que luego devolvía en esa misma cinta los resultados. Fueron los conceptos de Turing los que dieron lugar a los primeros procesadores que a su vez vivieron una revolución en su concepto con la llegada de los microprocesadores, donde se sustituían válvulas de vacío por transistores.

Los ordenadores, que durante la década de los años 60 empezaron a crecer como industria, enfocada en las grandes empresas o instituciones académicas y de investigación, se convirtieron en herramientas imprescindibles y a su vez, de alguna forma se supuso que esa sería su mayor evolución: una herramienta para empresas o instituciones de enseñanza. No sería hasta la llegada de los años 70 cuando se introdujo el concepto de la microinformática o la informática enfocada al mercado de consumo. De la mano de algunos visionarios como Steve Jobs o Bill Gates, llegó el siguiente paso en la evolución de la informática: llevar los ordenadores al consumo doméstico y poner un ordenador en cada casa de cada persona.

Algo que, como se puede imaginar, no todos entendieron pues, ¿para qué quería una persona tener un ordenador y para qué iba a usarlo? La pertinente pregunta sería: ¿por qué tenían ese pensamiento que a día de hoy nos parece absurdo? Porque se pensaba que los usos que los ordenadores tenían no eran aplicables a la casa de nadie. Lo que no veían es que la programación sería la herramienta para crear otros usos que hasta entonces ni siquiera se habían planteado como las hojas de cálculo, el diseño, la autoedición, procesamiento de textos, fotografía, videojuegos… cada cosa llegaría cuando los ordenadores permitieran poco a poco ampliar sus posibles funciones hasta el punto que si hoy miramos nuestro móvil, veremos que este sustituye en un solo dispositivo una enorme cantidad de aparatos electrónicos que se usaban hace años como: videocámaras, reproductores portátiles de música, grabadoras de sonido, cámaras de fotos, consolas de videojuegos, televisión, radio, teléfono e incluso un cuaderno de anotación, bloc de bocetos de pintura o sobres para enviar cartas…

Los primeros años hacia el Apple I (Integer BASIC)

El 1 de abril de 1976, Steve Jobs, Steve Wozniak y Ronald Wayne fundaron Apple Computers. Por un motivo principal: el desarrollo.

Steven Paul Jobs, nació el 24 de febrero de 1955. Un visionario que con su especial forma de ver las cosas, su obstinación y amor por el más profundo y pequeño detalle, estaría llamado a redefinir varias industrias culturales y de consumo, como el cine, la música, los videojuegos y la tecnología en sí. Alguien que, guste más o menos, ha ayudado a definir el mundo en que vivimos. Y una de las personas directamente responsables del profundo cambio que el desarrollo ha sufrido en los últimos 40 años,