



**PUCE**

Sede  
Ibarra

# APLICACIONES MÓVILES HÍBRIDAS



PRIMERA EDICIÓN

**Pontificia Universidad Católica del Ecuador - Sede Ibarra**

Ibarra: Av. Jorge Guzmán Rueda y Av. Aurelio Espinosa

Pólit. Cdla “La Victoria”

Teléfono: 06 2615 500 / 06 2615 631

Fax: (593)6-2615 446

Apartado Postal 10.01.12

Web Site: [www.pucesi.edu.ec](http://www.pucesi.edu.ec)

Email: [prorect@pucesi.edu.ec](mailto:prorect@pucesi.edu.ec)

**Sello editorial****Centro de publicaciones PUCE**

[www.edipuce.edu.ec](http://www.edipuce.edu.ec)

Quito, Av. 12 de Octubre y Robles

Apartado n.o 17-01-2184

Telf.: (5932) 2991 700

E-mail: [publicaciones@puce.edu.ec](mailto:publicaciones@puce.edu.ec)

**ISBN:**

978-9978-375-54-9

**Título:**

Aplicaciones Móviles Híbridas

**Autores:**

Mgs. Galo Puetate

Mgs. José Luis Ibarra

Escuela de Ingeniería

**Ilustración de portada:**

Autores

**Concepto gráfico y diagramación:**

Oswaldo Portilla Villamagua

**Revisión de estilo y redacción:**

Mgs. Gabriela Garcés

PUCE-Sede Ibarra

**Correcciones generales****Centro de Publicaciones Pontificia Universidad Católica del Ecuador.**

Este libro fue sometido al debido arbitraje y dictamen de pares evaluadores expertos en el área.



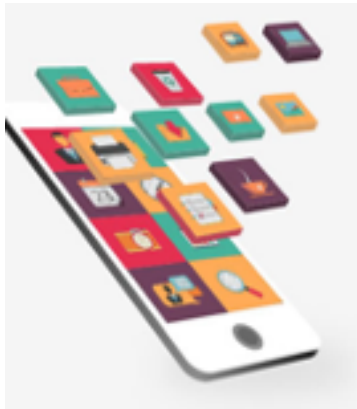
# **APLICACIONES MÓVILES HÍBRIDAS**

**GALO PUETATE**

Escuela de Ingeniería.

**JOSÉ LUIS IBARRA**

Escuela de Ingeniería



## PRÓLOGO

El presente libro tiene como objetivo mostrar al lector ejercicios, herramientas, lenguajes y procedimientos requeridos para el desarrollo de aplicaciones móviles híbridas, para permitir el desarrollo e implementación de una aplicación móvil similar a una generada con un lenguaje de programación nativo.

A su vez en el contenido de este libro se da a conocer las ventajas que se obtiene con este tipo de desarrollo móvil, siendo una de ellas el no requerir aprender nuevos lenguajes de programación, sino en su defecto optimizar y potenciar los conocimientos adquiridos en la programación web, en el uso del desarrollo de aplicaciones móviles, las cuales pueden ser compiladas para diferentes sistemas operativos móviles sin requerir diferente programación como es: Android, IOS, Windows iPhone, entre otros.

## RESUMEN EJECUTIVO

Las tecnologías han impactado de gran manera en las actividades cotidianas del ser humano, en este contexto se ha pasado de los sistemas de escritorio, por las aplicaciones web hasta llegar a las aplicaciones móviles; los dispositivos inteligentes, motivo por el cual nace el libro como material didáctico para brindar a los estudiantes de la Escuela de Ingeniería y profesionales del área de tecnologías, un libro como medio de conocimiento enfocado al desarrollo de aplicaciones móviles híbridas.

## CONTENIDOS

<b>PRÓLOGO</b>	<b>5</b>
<b>RESUMEN EJECUTIVO</b>	<b>6</b>
<b>CONTENIDOS</b>	<b>7</b>
<b>LISTADO DE FIGURAS</b>	<b>9</b>
<b>GLOSARIO DE TÉRMINOS</b>	<b>11</b>
<b>INTRODUCCIÓN</b>	<b>17</b>
<b>APLICACIONES HÍBRIDAS</b>	<b>19</b>
<b>CONOCIMIENTOS PREVIOS</b>	<b>24</b>
Variables	24
Tipos de variables	25
Arreglos	25
Tomar valores por pantalla	26
Confirmación y ejecuta una acción	26
Importar datos del sistema	27
Mostrar y ocultar formularios	28
<b>RESTRICCIÓN DE DATOS EN UN CAMPO DE TEXTO</b>	<b>30</b>
Elementos aleatorios sin repetir	30
Ejecutar una acción al pasar sobre un elemento	32
Información enlazada a un menú de selección	33
<b>EJECUTAR ACCIONES AL CARGARSE EL PROGRAMA</b>	<b>36</b>
Modificar texto	36
Identificar el botón de opción señalado	37
Temporizador	38
<b>SUSTITUCIÓN DE TEXTO EN CADENAS</b>	<b>40</b>
Obtener y mostrar la geolocalización del usuario	40
Visualizar la posición obtenida en el mapa de google	41
Obtener el valor y texto de una opción de selección señalada	43
Agregar contenido HTML mediante JavaScript	43
<b>AGREGAR ESTILOS CASCADING STYLE SHEETS CSS</b>	<b>45</b>
Forma general	45
Por clase (se identifica a una clase por el “.”)	45
Por identificador (se determina un id por el signo “#”)	45
Aplicación de estilos CSS	46
Reglas de hojas de estilo CSS	46
Aplicaciones de hojas de estilo cascada	48
<b>DESARROLLO DE APLICACIONES HÍBRIDAS</b>	<b>51</b>
Navegación entre las páginas	54
Enviando un parámetro	54

Home.html	56
Formularios de aplicaciones móviles	59
home.html Formulario	60
<b>BookDIRECTORYDEMO</b>	<b>62</b>
Configuración del Proyecto de Firebase	62
Almacenamiento persistente en dispositivos móviles	63
Estructura de base de datos	67
Configuración de seguridad	70
<b>CONSTRUCCIÓN DE APLICACIONES MÓVILES</b>	<b>73</b>
<b>HÍBRIDAS</b>	
<b>AGREGANDO APP A LA PLATAFORMA ANDROID</b>	<b>89</b>
<b>PUBLICACIÓN DE LAS APLICACIONES</b>	<b>91</b>
Versión de lanzamiento de app	91
Publicación de Android	92
Google Play Store	95
Actualización de la aplicación	96
Publicación de iOS	96
Construcción del build desde Xcode	99
Crear App desde Itunes connect	101
<b>EPÍLOGO</b>	<b>103</b>
<b>BIBLIOGRAFÍA</b>	<b>104</b>



## LISTADO DE FIGURAS

Fig. 1: Tipos de aplicaciones móviles	20
Fig. 2: Proceso de despliegue	21
Fig. 3: Aplicaciones híbridas	22
Fig. 4: Valores de pantalla	26
Fig. 5: Confirmación de acción código HTML	27
Fig. 6: Importar datos desde sistema	28
Fig. 7: Mostrar formularios	29
Fig. 8: Mostrar formularios JavaScript	29
Fig. 9: Elementos aleatorios	32
Fig. 10: Datos enlazados	35
Fig. 11: Ejecución de acciones de programa	36
Fig. 12: Acciones de identificación con botones	38
Fig. 13: Acciones de temporización HTML	39
Fig. 14: Acciones de temporización JavaScript	39
Fig. 15: Acciones de visualización de posiciones Google Map	42
Fig. 16: Salida de mapa	42
Fig. 17: Aplicación de estilos CSS	46
Fig. 18: Reglas para la aplicación de estilos CSS	47
Fig. 19: Utilización de reglas CSS	48
Fig. 20: Formularios con estilos CSS	49
Fig. 21: Navegación entre páginas	54
Fig. 22: Navegación entre páginas secuencias	55
Fig. 23: Navegación páginas depuración	55
Fig. 24: Formularios app	59
Fig. 25: Formularios app móvil ingreso datos	59
Fig. 26: Creando proyecto en FirebaseConsole	62
Fig. 27: Creando base de datos	63
Fig. 28: Configuración seguridad de base de datos	63
Fig. 29: Estructura de base de datos	68
Fig. 27: Subiendo la base de datos	69
Fig. 31: Importar archivo Datos.Json	69
Fig. 32: Datos.Json	70
Fig. 33: Reglas de seguridad de la aplicación móvil	70
Fig. 34: Configuraciones de seguridad app	71
Fig. 35: Importar archivo estructura Datos.Json	71
Fig. 36: Cargar proyecto	71
Fig. 37: Agregar a proyecto a Firebase	72
Fig. 38: Configuraciones del proyecto en Firebase	72
Fig. 39: Crear App Móvil Híbrida	73
Fig. 40: Configuración de estructura de base de datos app	74
Fig. 41: Proceso para incorporar módulos de AngularFire	74
Fig. 42: Crear categorías app	78
Fig. 43: Crear categorías app	80
Fig. 44: Importar módulos de las categorías de la App	82

Fig. 45: Proceso de carga de datos módulos de la App	83
Fig. 46: Importar módulos de las categorías de la App	84
Fig. 47: Proceso de compilado de la App hibrida	85
Fig. 48: Resultado configuración de colores app móvil	86
Fig. 49: Resultado de aplicar categorías a una aplicación móvil	87
Fig. 50: Despliegue de contenido HTML en aplicación móvil	88
Fig. 51: Instalación de plataforma Cordova a Ionic	89
Fig. 52: Despliegue de la aplicación móvil	90
Fig. 53: Acceder a Google Play Developer Console	95
Fig. 54: Creado proyecto en Xcode	97
Fig. 55: Comprobar credenciales y certificados de la cuenta Apple	98
Fig. 56: Asociar App-ID	98
Fig. 57: Certificado de distribución	99
Fig. 58: Certificado de distribución	99
Fig. 59: Construcción del build	100
Fig. 60: Generando buil para lista de distribución	100
Fig. 61: Crear App desde Itunes connect	101
Fig. 62: Cargado aplicación de Xcode a Itunes connect	101
Fig. 63: Enviando aplicación a Itunes connect	102
Fig. 64: Enviando aplicación a la tienda Apple Store	102

## GLOSARIO DE TÉRMINOS

<b>Android Studio</b>	Android Studio es un nuevo entorno de desarrollo integrado para el sistema operativo Android lanzado por Google, diseñado para ofrecer nuevas herramientas para el desarrollo de aplicaciones y alternativa al entorno Eclipse, hasta ahora el IDE más utilizado.
<b>Angularjs</b>	AngularJS es un framework MVC de JavaScript para el Desarrollo Web Front End que permite crear aplicaciones SPA Single-Page Applications.
<b>Apis</b>	La interfaz de programación de aplicaciones, abreviada como API del inglés: Application Programming Interface, es un conjunto de subrutinas, funciones y procedimientos (o métodos, en la programación orientada a objetos) que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción.
<b>App</b>	Es el nombre usado comúnmente para referirse a las aplicaciones, que surge de acortar el vocablo inglés application. Es una pieza de software que se ejecuta en teléfonos móviles y tabletas y, como te habrás dado cuenta, es el objeto de estudio de este libro. Si aún no entiendes lo que es una app te recomendamos leer este libro con más atención.
<b>App nativa</b>	Aplicación que se desarrolla de forma específica para un determinado sistema operativo: Android, iOS o Windows Phone.
<b>App híbrida</b>	Son aplicaciones desarrolladas usando HTML5, CSS y JavaScript, desplegadas dentro de un contenedor nativo como Phonegap/Cordova el cual brinda acceso a las capacidades del dispositivo de una forma

	totalmente neutral respecto al sistema operativo. Es un modelo neutro respecto a a la plataforma y con portabilidad máxima
<b>App store:</b>	Es el marketplace de aplicaciones para Apple, donde los desarrolladores pueden publicar sus apps y los usuarios pueden descargarlas para sus dispositivos iOS.
<b>Apk:</b>	Un APK es el formato de archivo que debes subir a Google Play para publicar tu aplicación Android. Este formato es una variante JAR de Java y se usa para distribuir e instalar componentes empaquetados para la plataforma Android, tanto smartphones como tablets.
<b>Benchmarking:</b>	El benchmarking es un proceso sistemático para evaluar comparativamente productos, servicios y procesos. En nuestro contexto, es entonces el estudio comparativo y analítico de otras aplicaciones, con el fin de determinar la calidad y características de cada una de ellas, tomándolas como parámetros de referencia.
<b>Compilar</b>	Es la acción de empaquetar un código. El resultado de compilar el código de una aplicación es el archivo final que está listo para ser subido a la tienda.
<b>Contexto uso</b>	Entorno general conformado por la ubicación y espacio físico que rodea al usuario y al dispositivo. El contexto de uso determina, además, la forma en que estos dos componentes se relacionan e interactúan entre sí.
<b>Css</b>	Siglas de Cascading Style Sheets, que en español sería «Hojas de estilo en cascada». Ya sea en archivos separados o dentro del código HTML, este lenguaje determina la apariencia visual de una web o aplicación

	web definiendo, entre otras cosas, los colores y tamaños de fuente.
<b>DOS</b>	Interpreta comandos de DOS en sistemas operativos de Windows.
<b>DP</b>	Corresponde a las siglas de Density-independent pixels o «píxeles independientes de la densidad». Es una unidad de medida empleada por Android que está relacionada con la densidad física de la pantalla. Los DP son unidades relativas a las pantallas de 160 DPI, en las cuales un DP equivale a un píxel.
<b>Escenario</b>	Se refiere a la combinación de contexto de uso y «Persona». Determina la manera como un usuario se relaciona con el móvil en una situación específica.
<b>Firebase</b>	Firebase es la nueva plataforma de desarrollo móvil que permite desarrollar apps multiplataforma (Android, iOS y web)
<b>Google play:</b>	Es el marketplace de aplicaciones para Android, donde los desarrolladores pueden publicar sus apps y los usuarios pueden descargarlas para sus dispositivos Android.
<b>HTML</b>	Corresponde a las siglas de HyperText Markup Language. Es el lenguaje que se utiliza tradicionalmente para construir páginas web y aplicaciones web para móviles. Define la estructura de un documento web basado en una serie de etiquetas.
<b>Ionic</b>	Es un framework para el desarrollo de aplicaciones híbridas, inicialmente pensado para móviles y tablets, aunque ahora también capaz de implementar aplicaciones web e incluso aplicaciones de escritorio multiplataforma

<b>Interfaz UI</b>	La interfaz o User Interface es la capa que existe entre el usuario y el dispositivo, que le permite interactuar con este último. En las aplicaciones se trata del componente gráfico que contiene elementos que producen reacciones al pulsarlos y permiten al usuario realizar tareas, como también, aquellos estáticos sobre los cuales se interpretan los contenidos.
<b>Javascript</b>	Lenguaje de programación utilizado principalmente en proyectos web como sitios o aplicaciones, que muchas veces actúa en conjunto con HTML y CSS para dotarlos de funcionalidad.
<b>Librería</b>	En programación, se llama así al conjunto de código externo que puede aprovecharse para conseguir determinados comportamientos. Tiene relación directa con el lenguaje de programación elegido.
<b>Móvil</b>	También llamado (teléfono) celular en algunos países de América Latina, es un artefacto electrónico de tamaño variable donde funcionan las aplicaciones y estamos casi seguros de que tienes uno en tu mano o bolsillo ahora mismo.
<b>Node.js</b>	Node.js® es un entorno de ejecución para JavaScript construido con el motor de JavaScript V8 de Chrome. Node.js usa un modelo de operaciones E/S sin bloqueo y orientado a eventos, que lo hace liviano y eficiente.
<b>Orientación</b>	Es la manera en que se muestra el contenido en pantalla, dependiendo de la forma en que el usuario sostiene su tableta o teléfono. Puede ser vertical u horizontal.
<b>PhoneGap</b>	Es un framework para el desarrollo de aplicaciones móviles que permite desarrollar aplicaciones para

	dispositivos móviles utilizando herramientas genéricas como JavaScript, HTML5 y CSS3
<b>SDK</b>	El Software Development Kit o «Kit de desarrollo de software» provee a los programadores herramientas necesarias para desarrollar el código de una aplicación. Tanto Android, como iOS y Windows Phone, ofrecen uno diferente.
<b>Sistema O.S</b>	Es el software que contiene cada uno de los teléfonos y sobre el cual se ejecutan las aplicaciones. Las distintas versiones de Android, iOS y Windows Phone, son ejemplos de sistemas operativos.
<b>Tienda</b>	Es el canal de distribución y comercialización de aplicaciones, desde donde pueden descargarse de forma gratuita o paga. Cada uno de los sistemas operativos móviles mencionados en este libro tiene una tienda oficial; sin embargo, en el caso de Android, existen varias opciones alternativas además de Google Play; cómo la tienda de apps de Amazon o Samsung.





## INTRODUCCIÓN

El desarrollo tecnológico ha marcado la tendencia en cuanto a la forma de que los usuarios puedan acceder a los datos e información de las organizaciones, es así, que en el ámbito de aplicaciones informáticas se ha pasado por las aplicaciones de escritorio, aplicaciones web y a lo que hoy se le denominan aplicaciones para dispositivos móviles (App), que son aplicaciones diseñadas para ejecutarse en tablet, teléfonos o cualquier otro dispositivo móvil. Desde el punto de vista del desarrollo de aplicaciones móviles, es necesario tener conocimientos de programación, manejo de datos, acceso a recursos propios de los dispositivos móviles, además de tener un dominio en distintos framework y lenguajes de desarrollo para aplicaciones móviles.

La finalidad del libro denominado Aplicaciones Móviles Híbridas, es presentar el proceso de desarrollo de aplicaciones híbridas, además de describir los distintos escenarios, o requisitos técnicos necesarios para que los profesionales en formación y los estudiantes de la Escuela de Ingeniería, tengan acceso a material bibliográfico donde puedan desarrollar en el paradigma del diseño y desarrollo de aplicaciones tecnológicas móviles.

El libro se encuentra estructurado en tres secciones claramente definidas, en la primera se abordan aspectos propios de la programación y corresponde a los conocimientos previos que el estudiante deberá conocer antes de adentrarse en el desarrollo de aplicaciones para dispositivos móviles.

La segunda sección del libro, está centrada en sí en el desarrollo de aplicaciones móviles híbridas, que se inicia con la configuración necesaria para crear una aplicación, manejo de parámetros, formularios de las aplicaciones móviles. Configuración de directorios, estructuras de almacenamiento de datos de aplicaciones móviles y aspectos de seguridad.

La tercera sección corresponde al proceso de publicación de las aplicaciones móviles en las diferentes tiendas de app (Google Play, App Store), proceso que debe seguir los aspectos de lanzamiento de versión, actualizaciones y publicación.



## APLICACIONES HÍBRIDAS

En la era de la información e inmediatez en la que nos encontramos las organizaciones dependen en gran medida de las aplicaciones tecnológicas para llevar a cabo tareas cotidianas como enviar mails, revisar estados de cuentas, recibir y enviar informes entre otras actividades que en la actualidad son realizadas a través de aplicaciones para dispositivos móviles.

Las aplicaciones móviles han ido incursionando a partir de inserción de lo que se denomina Smartphone a nuestra cotidianidad, donde cada vez es más frecuente su utilización para comunicarnos y ejecutar tareas de distinta índole. Para entender que es una aplicación móvil, se puede decir que las aplicaciones móviles son programas diseñados para ejecutarse en teléfonos, tablets y otros dispositivos móviles, y que, permiten al usuario realizar actividades profesionales, acceder a servicios, mantenerse informado, entre otras posibilidades y usos, según sea el fin para el cual se desarrollan. (SoftCorp, 2018).

Las aplicaciones móviles incursionaron a mediados del año 2004, cuando el desarrollo de dispositivos móviles (teléfonos) fueron teniendo más capacidad en cuanto a recursos hardware, mejoras significativas en cuanto a la interfaz de usuarios, esto dio como resultado el desarrollo de app móviles tales como: la agenda, juegos como el famoso snake, editores de tonos, herramientas para personalizar el teléfono, entre otras. A medida que los teléfonos fueron teniendo mejores capacidades tecnológicas.

En la actualidad se han desarrollado todo tipo de aplicaciones con distintos fines y objetivos que van más allá de las prestaciones de los primeros teléfonos y en la actualidad se centran en la productividad personal, profesional y empresarial, ya que las apps se integran a los distintos sistemas informáticos de las organizaciones que permiten acceder e interactuar con los datos e información de los usuarios a través de las aplicaciones móviles.

Con el ingreso al mercado de empresas como iPhone, Google, Samsung, Huawei se dio un cambio en el modelo de negocio donde las aplicaciones móviles se hicieron rentables, tanto para los desarrolladores como para el mercado de aplicaciones en las tiendas App Store, Google Play entre otras tiendas.

En la actualidad se han desarrollado tres tipos de aplicaciones las cuales son: app nativa, aplicaciones web y aplicaciones híbridas, son el resultado de la combinación entre las aplicaciones nativas y las webs apps que se desarrollan usando tecnologías web tales como: HTML, CSS, JavaScript que se compila y empaqueta de tal forma que el resultado final es una app para dispositivos móviles.

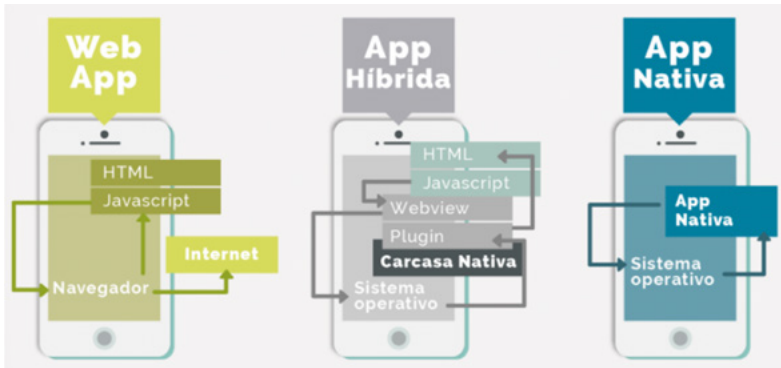


Fig. 1: Tipos de aplicaciones móviles  
Fuente: Gsoft Informática

La fig. 1, muestra las características principales de las aplicaciones híbridas, es la capacidad de estas para ejecutarse en múltiples plataformas de sistemas operativos de los Smartphone tales como: iOS, Android, Windows, entre otras. Es necesario recalcar que toda aplicación indistintamente del tipo sigue un proceso de desarrollo de software que abarca desde la concepción (modelo conceptual de la idea), el análisis, desarrollo y la posterior publicación en las diferentes tiendas de apps.

Las razones principales de porque las aplicaciones móviles híbridas han cobrado mayor desarrollo se resumen a continuación:

- Desarrollo sencillo y económico para las organizaciones.
- El código puede ser reutilizado para ser ejecutado en cualquier plataforma de sistema operativo de los fabricantes de Smartphone.
- No se requiere de complejos procedimientos para la publicación en las tiendas de aplicaciones móviles.

El enfoque híbrido combina desarrollo nativo con tecnología web, donde los desarrolladores escriben gran parte del código de la aplicación en tecnologías web y que mediante un proceso de compilación se puede ejecutar para múltiples plataformas.



Fig. 2: Proceso de despliegue  
Fuente: Gsoft Informática

Por lo tanto, los desarrolladores pueden optar por codificar su propio puente o bien aprovechar soluciones ya construidas, para procesos de despliegue haciendo uso de herramientas tales como; PhoneGap, AngularJs, Ionic, entre otras tecnologías que permiten tener acceso a las distintas funcionalidades propias de los dispositivos que son iguales en todos los sistemas operativos (ver fig 2).

En algunos casos, una solución va a permitir que el desarrollador utilice cualquier conocimiento nativo que pueda tener para adaptar el contenedor nativo a las necesidades únicas de la organización.

La parte web de la aplicación puede ser una página web normal que resida en un servidor o bien un conjunto de archivos HTML, JavaScript, CSS y medios, incorporados en el código de la aplicación y almacenados localmente en el dispositivo. Ambos enfoques presentan ventajas y desventajas.

Comparación de los distintos enfoques a modo de resumen. El enfoque nativo se destaca por su desempeño y acceso de los dispositivos, pero conlleva costos y requiere actualizaciones constantes y por ende mantenimiento y soporte continuo al usuario final. El enfoque web es más simple, menos costoso y más fácil de actualizar, pero actualmente su funcionalidad se limita al conocimiento del desarrollador, así como al nivel de acceso a los sensores y funcionalidades del dispositivo móvil del cliente final, además de las continuas llamadas API nativas.

El enfoque híbrido ofrece un término medio que, en muchas situaciones, constituye lo mejor de ambos enfoques de desarrollo, en especial si el desarrollador desea emplearlo en múltiples sistemas operativos que tenga el usuario final.



Fig. 3: Aplicaciones híbridas  
Fuente: Gsoft Informática

Como se observa en la figura 1, ninguno de los enfoques en sí ofrecen todos los beneficios todo el tiempo; sin embargo, a la hora de elegir el enfoque más adecuado hay que tener en cuenta las necesidades específicas de la organización, y basarse en muchos parámetros, como presupuesto, plazos de entrega, recursos internos, mercado objetivo, funcionalidad requerida de la aplicación, infraestructura de tecnología de Información que disponga la organización, así como el mantenimiento y soporte.

## CONOCIMIENTOS PREVIOS

La imparable expansión de los teléfonos inteligentes o Smartphones ha generado un mapa de oportunidades sin precedentes para el mercado de las aplicaciones móviles. Esta situación ha supuesto todo un revulsivo para desarrolladores y empresas que han visto en este campo un nicho de negocio para crear aplicaciones gratuitas o de pago, de éxito, o con muy pocos recursos.

Pero competir en este mercado emergente no es una tarea fácil y requiere conocimientos de programación. A pesar de que sea complejo no quiere decir que sea imposible, de hecho, no es imprescindible saber programar para crear aplicaciones propias.

En Think Big les vamos a explicar las diferentes herramientas disponibles para crear apps móviles para las distintas plataformas existentes en el mercado sin conocimientos previos de programación.

### Variables

En JavaScript se crean variables mediante la palabra reservada “var” de la siguiente manera:

```
var num1=3;
var num2=4;
var resultado = num1 + num2;
```

Tener en cuenta que:

1. La declaración de las variables es necesario realizarlas una sola vez.
2. Los identificadores o nombre de las variables deben únicamente formarse por letras, números y símbolos de dólar o guión bajo, así como también que el primer caracter que constituye el identificador no puede ser un número.



## Tipos de variables

En JavaScript las variables se crean utilizando la misma palabra reservada “var” dependiendo el tipo de variable únicamente de la forma en la que se le asigna el valor.

```
var num = 3; //variable tipo entero
var sueldo= 375,76; // variable de tipo decimal
var saludo= “Hola mundo”; // variable de tipo cadena de texto
var mensaje= ‘Bienvenido’; //variable de tipo cadena de texto
var estado = true; // tipo de dato booleano que puede ser true o false
```

## Arreglos

Un arreglo es un conjunto de datos finitos que pueden ser del mismo tipo o diferentes.

```
var nom1=“Pedro”;
var nom2=“Juan”;
var nom3=“Lucas”;
var nom4=“Lucia”;
var nombres= [“Pedro”, “Juan”,” Lucas”,” Lucia”];
var nombre_array= [nom1, nom2, nom3, nom4];
```

Para acceder a la información almacena en el arreglo se lo realiza de la siguiente manera:

```
var selección_1 = nombre[0]; //nombre seleccionado Pedro
var selección_2 = nombre_array[3]; // nombre seleccionado Lucas
```

Mostrar valores por pantalla

Se lo realiza mediante la utilización de “alert”.

```
alert(“Su mensaje”);
```

## Tomar valores por pantalla

Para solicitar información al usuario para posteriormente realizar algún tratamiento a esa información recibida se lo realiza mediante la palabra “prompt”.

```
prompt("Mensaje de información");
Sample
var edad= prompt("Ingrese su edad");
if(edad >=18){
    alert("Ud. Es mayor de edad");
} else{
    Alert("Ud. Es menor de edad");
}
```

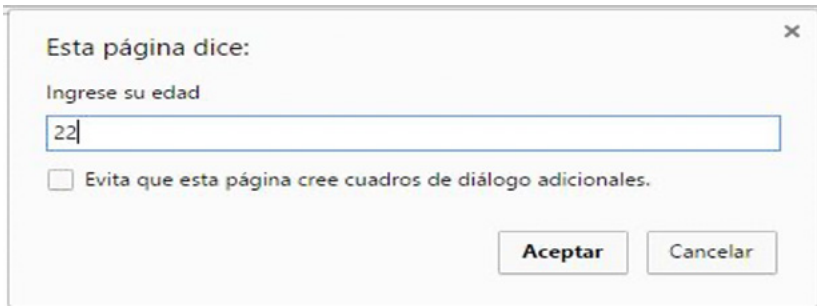


Fig. 4: Valores de pantalla  
Fuente: Los Autores

## Confirmación y ejecuta una acción

Esta función permite que antes de ejecutar una acción el usuario confirme su aceptación para que esta acción se realice. En el siguiente ejemplo se le solicitará al usuario que confirme que está de acuerdo en recibir un saludo de bienvenida.

HTML

```
<input type="button" onClick="confirma();" name="Confirma" value="
Confirmar Saludo">
```

```
<input type="button" onClick="confirma();" name="Confirma" value="
Confirmar Saludo">
```

JavaScript

```
function confirma(){
    if(confirm("Quiere Ud, un saludo")){
        alert("Hola como esta bienvenido");
    }
}
```

### Resultado

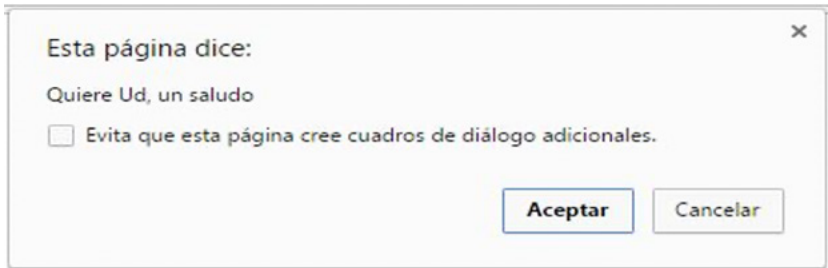


Fig. 5: Confirmación de acción código HTML

Fuente: Los Autores

## Importar datos del sistema

Para mostrar datos del sistema como la hora del sistema se toman los valores del servidor mediante JavaScript y se los imprime en un formulario HTML, de la siguiente manera:

### HTML

```
<body onLoad="mueveReloj();">
<form name="form_reloj">
<input type="text" name="reloj" size="10">
</form>
</body>
```

### JavaScript

```
function mueveReloj(){
    var momentoActual = new Date()
    var hora = momentoActual.getHours()
    var minuto = momentoActual.getMinutes()
    var segundo = momentoActual.getSeconds()
    var horaImprimible = hora + ":" + minuto + ":" + segundo
    document.form_reloj.reloj.value = horaImprimible
}
```

### Resultado

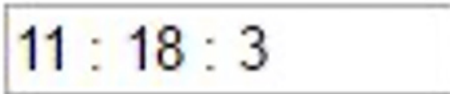


Fig. 6: Importar datos desde sistema

Fuente: Los Autores

## Mostrar y ocultar formularios

Para realizar esta acción es necesario definir el estilo que tendrá el formulario por defecto para posterior a alguna acción este estilo se modifique mediante JavaScript.

## HTML

Aquí hay una corrección que no sé si es procedente

```
<input type="button" name="Registro" onClick="MostrarF();"
id="Registro" value="Registro" />
<form id="f1" style="display:none">
  <input type="text" title="Ingrese solo su nombre" placeholder="
Nombre:" id="Nombre1" required />
  <input type="text" title="Ingrese solo su apellido" placeholder="Ape
llido:" id="Apellido1" required />
  <input type="text" title="Ingrese su edad" placeholder="Edad:"
id="edad1" required />
</form>
```

## JavaScript

```
function MostrarF(){
  document.getElementById("f1").style.display="block";
}
```

## Resultado



Fig. 7: Mostrar formularios  
Fuente: Los Autores

## Resultado



Fig. 8: Mostrar formularios JavaScript  
Fuente: Los Autores

## RESTRICCIÓN DE DATOS EN UN CAMPO DE TEXTO

La restricción de datos a ingresarse en un campo de texto se lo puede hacer mediante JavaScript en el cual definimos el tipo de datos que va a poder ingresar el usuario, e invocamos esta acción JavaScript cada vez que el usuario quiera ingresar un valor mediante el evento HTML “onKeyPress”.

En el siguiente ejemplo se obliga a que únicamente el usuario pueda ingresar números en el campo de texto.

### HTML

```
<body>
<input id="ejemplo" onKeyPress="return justNumbers(event);" va
lue="1"></input>
</body>
```

### JavaScript

```
function justNumbers(e) {
var keynum = window.event ? window.event.keyCode : e.which;
  if ((keynum == 1) || (keynum == 999))
    return true;
  return /^[d/].test(String.fromCharCode(keynum));
}
```

## Elementos aleatorios sin repetir

Para mostrar elementos de un vector de forma aleatoria se lo realiza mediante la función `Math.random()` pero existe la posibilidad de que, si deseamos que selecciones de un vector, 4 elementos estos se repitan, para evitar la selección repetida de elementos de un vector cuando se los ha obtenido de forma aleatoria se lo realiza de la siguiente manera:

## Crear vector

```
var zoo = ['Gato', 'Perro', 'Caballo', 'Ganso', 'Pez', 'Foca', 'Papagayo', 'Coyote', 'Milano', 'Nutria', 'Cotorra', 'Tigre'];
```

## Cantidad de elementos a mostrar

```
var cantidad = 5;
```

## Función que realiza la acción

```
function seleccionar(cantidad, zoo) { // define la cantidad de elementos
a seleccionar y el vector de donde se los seleccionará.
    this.cantidad = cantidad;
    this.zoo = zoo;
    var tamaño = zoo.length; // se determina el tamaño del
vector
    var lote = new Array(); // arreglo donde se almacenar los
elementos ya seleccionados
    var indice = 0;
    do {
        var aleatorio = zoo[parseInt(Math.random()*
tamaño)];
        if(lote.indexOf(aleatorio)!=-1){
            continue;
        }else{
            lote[indice]=aleatorio;
            indice++;
        }
    } while(lote.length < cantidad);

    alert(lote); // alerta mediante la cual se presentará los el
ementos seleccionados de forma aleatoria.
}
```

## Llamado de la función

```
seleccionar(cantidad, zoo);
```

## Resultado función

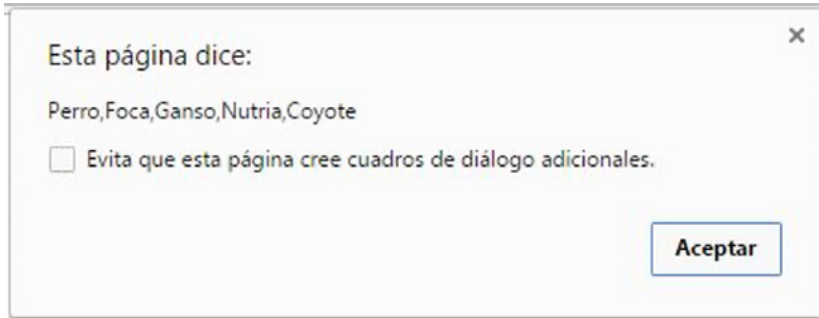


Fig. 9: Elementos aleatorios

Fuente: Los Autores

## Ejecutar una acción al pasar sobre un elemento

Esto se lo puede realizar mediante la función “onmousemove” la cual permite ejecutar una función JavaScript en el momento que el que usuario con el puntero pase por un área definida.

En el siguiente ejemplo se ejecutará un mensaje al pasar con el puntero sobre una imagen.

## HTML

```
<img width="160" height="140" onmousemove="mover()" name="imagen" src="" />
```



## JavaScript

```
function mover() {  
    alert("SOBRE LA IMAGEN");  
}
```

## Información enlazada a un menú de selección

En este apartado se explicará cómo enlazar una información tanto en texto como gráficos a un menú de selección.

## HTML

Formulario donde se define el menú de selección y el área de texto donde se mostrará el texto relacionado al dato señalado en el menú.

```
<form name="formulario">  
    <p>  
        <select name="selectbox" size="1"n onchange="changecon  
tent(this)">  
            <option selected>opción 1</option>  
            <option>opción 2</option>  
            <option>opción 3</option>  
            <option>opción 4</option>  
        </select>  
    <br>
```

```

<textarea rows="8" name="contentbox" cols="35" wrap="virtual" ></
textarea>
</p>
<p>
<!-- <input type="radio" name="radio" id="radio" value="radio"
checked="checked" />
<label for="radio"></label>
</p>
<p>
<input type="radio" name="radio" id="radio" value="radio2" />
<label for="radio2"></label-->
</p>
<p><br>
</p></form>
    
```

Área de imagen definida donde se mostrará la imagen correspondiente al dato señalado en el menú.

```

<img width="160" height="140" name="imagen" src=""/>
    
```

## JavaScript

Se definen los arreglos donde se almacenará las imágenes y la descripción de cada imagen.

```

var d=new Array()
d[0]='img/1.jpg'
d[1]='img/2.jpg'
d[2]='img/3.jpg'
d[3]='img/4.png'
var datos= new Array()
datos[0]="Esta es una descripción de los lenguajes de programación de
última generación";
datos[1]="Esta es una imagen que muestra que el conocimiento del
mundo es para el mundo";
    
```

```
datos[2]="El desconocimiento de la tecnología no justifica el no aprender";  
datos[3]="Deja tu firma en el mundo, plasmando tus ideas de cambio";  
function changecontent(which){  
  imagen.src = d[document.formulario.selectbox.selectedIndex]  
  document.formulario.contentbox.value= datos[document.formulario.selectbox.selectedIndex]  
}
```

## Resultados

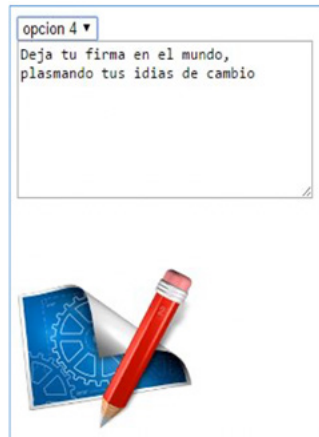
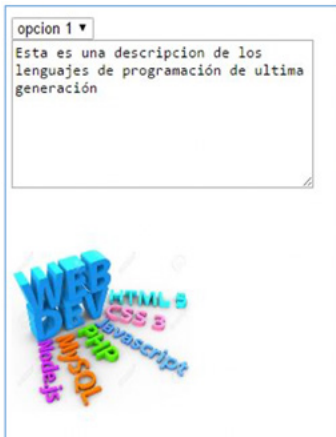


Fig. 10: Datos enlazados  
Fuente: Los Autores

## EJECUTAR ACCIONES AL CARGARSE EL PROGRAMA

Permite ejecutar acciones o funciones específicas al momento que el sistema es ejecutado, ejemplo un mensaje de alerta que el programa está listo para ser usado.

### JavaScript

```
$(document).on('ready',function () {
    alert("Su aplicación movil está lista para ser usada");
})
```

### Resultado

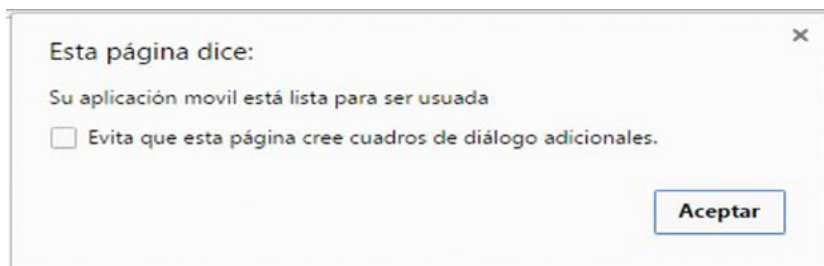


Fig. 11: Ejecución de acciones de programa  
Fuente: Los Autores

### Modificar texto

Se explica cómo modificar el contenido de un texto.

### HTML

```
<div id="texto" >Hola cómo están </div>

<div class="texto2"> Bien gracias</div>

<input type="button" onclick="cambiar();" name="Cambiar" value="Cambiar" />
```

## JavaScript

```
function cambiar () {
    $('#texto').text("esta es una prueba");

    $('#.texto2').html('<strong>Esta es otra prueba</strong>');
}
```

### Identificar el botón de opción señalado

Este apartado muestra cómo identificar de forma instantánea el botón de opción señalado mediante la ejecución de una función JavaScript a través de “onchange”.

## HTML

```
<form name="f1" onchange="javascript:detector()">
<input type="radio" name="cosa" value="1">
<input type="radio" name="cosa" value="2">
<input type="radio" name="cosa" value="3">
<input type="radio" name="cosa" value="4">
</form>
```

## JavaScript

```
function detector() {
    var formulario = document.forms[0];
    for (var i = 0; i < formulario.cosa.length; i++) {
        if (formulario.cosa[i].checked) {
            break;
        }
    }
    alert('el marcado es: ' + formulario.cosa[i].value + '.');
}
```

## Resultado

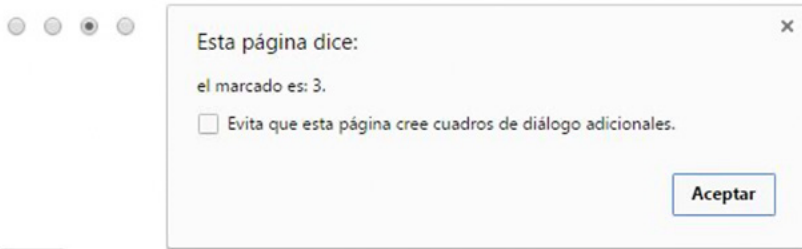


Fig. 12: Acciones de identificación con botones  
Fuente: Los Autores

## Temporizador

Se muestra como realizar un temporizador y como ejecutar una acción cuando este llegue a cero o el valor que se desee.

### HTML

```
<form id="rel" >
<input type="text" id="reloj" disabled="disabled">
</form>
```

### JavaScript

```
function timer()
{
    if(n>=0)
    {
        timeout=setTimeout("timer()",1000);
        document.getElementById('reloj').value=n+"";
    }
    else
    {
        if(cont<5)
        {
            bloqueo();
        }
    }
}
```

```
    alert("la respuesta correcta es: "+resp[cont]);
    document.getElementById('Error').style.display = 'block';
    document.getElementById('res').value= "su nota es: "+nota;
    document.getElementById('rel').style.display = 'none';
        }}
n--;}
```

### Resultado

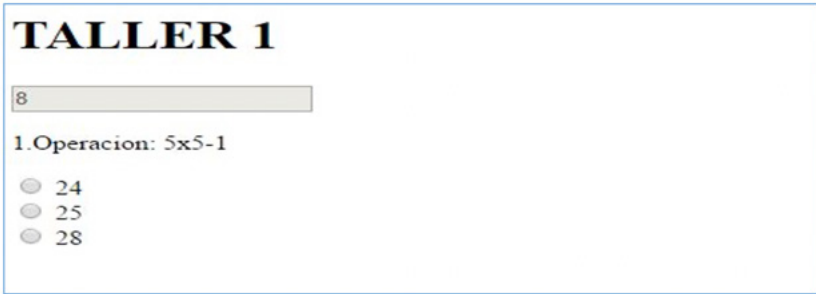


Fig. 13: Acciones de temporización HTML  
Fuente: Los Autores

### Respuesta

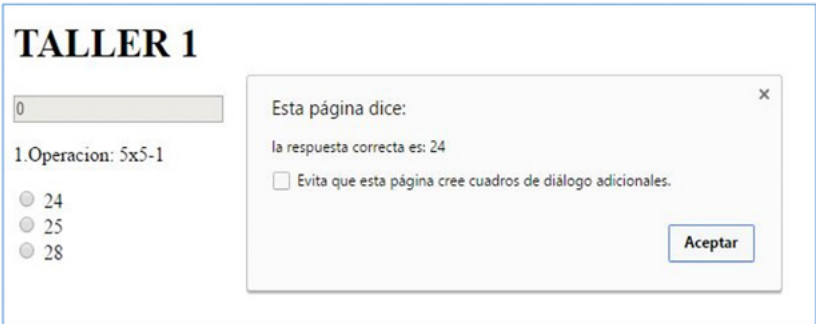


Fig. 14: Acciones de temporización JavaScript  
Fuente: Los Autores

## SUSTITUCIÓN DE TEXTO EN CADENAS

La sustitución de un texto determinado en una cadena es posible realizarlo mediante el uso de la función “replace” de JavaScript.

Ejemplo:

```
var frase = "Son tres mil trescientos treinta y tres con nueve";
frase = frase.replace("tres","dos");
alert(frase);
var frase = "Son tres mil trescientos treinta y tres con nueve";
frase3 = frase.replace(/[aiou]/gi,'e');
alert(frase3);
var frase = "Son tres mil trescientos treinta y tres con nueve";
frase4 = frase.replace(/dos/gi,'nueve');
alert(frase4);
```

## Obtener y mostrar la geolocalización del usuario

Para obtener la geolocalización de un usuario y visualizar en un mapa de google se procede a:

### Llamar a la API de GoogleMaps

```
src="https://maps.googleapis.com/maps/api/js?libraries=adsense">
```

### Activar la posición del usuario

```
var posición;
function getLocation() {
  if (navigator.geolocation) {
    navigator.geolocation.watchPosition(ObtenerPosition);
  } else {
    alert("Tu navegador no soporta la geolocalizacion");
  }
}
function ObtenerPosition(position) {
  posicion = position;
}
```



## Visualizar la posición obtenida en el mapa de google

```

function inicializar_mapa() {
    var mapOptions = {
        center: new google.maps.LatLng(0.35, -78.11667),
        zoom: 15,
        mapTypeId: google.maps.MapTypeId.ROADMAP
    };
    var map = new google.maps.Map(document.getElementById("mapa_div"),
        mapOptions);
    var aux = trayectoriaBusChorlavi();
    var Userpos = new google.maps.LatLng(posicion.coords.latitude, posicion.coords.longitude);
    var marker = new google.maps.Marker({
        position: Userpos,
        map: map,
        title: "Estas Aqui",
        icon: 'https://docs.google.com/uc?authuser=0&id=0BylX-35gpVdO-ZG9rbUZMXzVxNTg&export=download',
        animation: google.maps.Animation.DROP
    });
    for (i = 0; i < aux.length; i += 2) {
        var pos = new google.maps.LatLng(aux[i], aux[i + 1]);
        var imagen = '';
        if (i % 7 === 0 && i !== 0 && i !== aux.length) {
            imagen = 'https://google-maps-icons.googlecode.com/files/bus-tour.png';
        } else if (i === 0) {
            imagen = 'https://docs.google.com/uc?authuser=0&id=0BylX35gpVdO-X2FBUGhlaUFwSG8&export=download';
        } else {
            imagen = 'https://google-maps-icons.googlecode.com/files/bus.png';
        }
        var marker = new google.maps.Marker({
            position: pos,
            map: map,
            title: "Ruta",

```

```

icon: imagen,
animation: google.maps.Animation.DROP
});
}
}
}

```

## Resultado

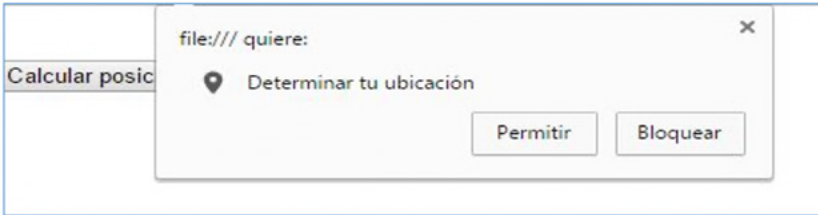


Fig. 15: Acciones de visualización de posiciones Google Map  
Fuente: Los Autores

## Resultado salida mapa

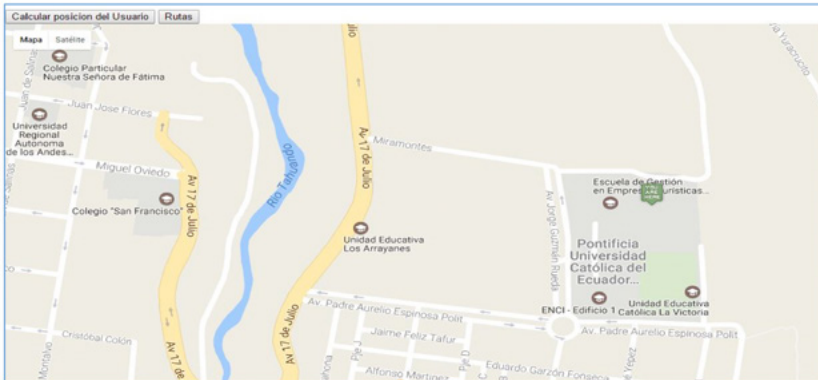


Fig. 16: Salida de mapa  
Fuente: Los Autores

## Obtener el valor y texto de una opción de selección señalada

### HTML

```
<select name="Variedad2" id="Variedad2" onchange='requerimientoN();' >
```

### JavaScript

```
var lista = document.getElementById("Variedad2");  
var indiceSeleccionado = lista.selectedIndex;  
var opcionSeleccionada = lista.options[indiceSeleccionado];  
var textoSeleccionado = opcionSeleccionada.text;  
var valorSeleccionado = opcionSeleccionada.value;
```

## Agregar contenido HTML mediante JavaScript

Este apartado explica cómo ir agregando contenido HTML a un documento mediante código JavaScript.

### HTML

```
<tbody id="elementsList4">  
  <tr>  
    <td colspan="3">Not elements to show</td>  
  </tr>  
</tbody>
```

## JavaScript

### Contenido HTML almacenado en una variable JavaScript

```
var outerHTML3= '';
outerHTML3 += '\n\
    <tr>\n\
        <td>' + n + '</td>\n\
        <td id="datos2" align="center">' + requiereAS + '</\
td>\n\
    </tr>';
```

Se selecciona el elemento HTML y se le agrega la variable JavaScript que contiene el código HTML mediante “innerHTML”.

```
document.querySelector("#elementsList4").innerHTML = outerHT-
ML3;
<head>
  <meta charset="utf-8">
  <title>Ejemplo de enlace a hoja de estilo</title>
  <link rel="stylesheet" href="../varios/htmlcss.css" title="Mi estilo">
</head>
```

## AGREGAR ESTILOS CASCADING STYLE SHEETS CSS

Se puede agregar estilos CSS a nuestro Proyecto móvil de forma general, por clase o por identificadores.

### Forma general

```
label {  
    font-size: 1.5em;  
    color: gray; }
```

### Por clase (se identifica a una clase por el ".")

```
aviso {  
    padding: 0.5em;  
    border: 1px solid #98be10;  
    background: #f6feda; }
```

### Por identificador (se determina un id por el signo "#")

```
#elegante {  
    width: 90%;  
    padding: 10px;  
    border: 2px dashed #CCC;  
    background: white; }
```

**Ejemplo:** Casilla de selección múltiple de una encuesta aplicando estilos CSS.

## HTML

```

<body>
  <h1>Estilos CSS en checkbox </h1>
  <p>
    <input type="checkbox" id="check1" checked>
    <label for="check1">Desarrollo Web</label>
  </p>
  <p>
    <input type="checkbox" id="check2">
    <label for="check2">Desarrollo Mobile </label>
  </p>
</body>
    
```

### Aplicación de estilos CSS

Se denominan hojas de estilo cascada (CSS), que es un lenguaje que permite dar estilos a los formularios y aplicaciones web, así como para aplicaciones que se ejecutan en dispositivos móviles.

El lenguaje de hojas de estilo CSS, facilita la aplicación de colores, contornos y formas de manera selectiva a los elementos en documentos HTML.



Fig. 17: Aplicación de estilos CSS  
Fuente: Los Autores

### Reglas de hojas de estilo CSS

Las reglas de las hojas de estilo cascada CSS, tienen una estructura lógica de utilización y por lo general tiene una estructura de declaración.

La estructura completa se llama regla predeterminada, pero a menudo se le denomina “Regla”, que es una abreviación.



Fig. 18: Reglas para la aplicación de estilos CSS

Fuente: Los Autores

**Selector.** El elemento HTML en el que comienza la regla. esta selecciona el(los) elemento(s) a dar estilo (en este caso, los elementos p). Para dar estilo a un elemento diferente, solo cambia el selector.

**Declaración.** Una sola regla como `color: red;` especifica a cuál de las propiedades del elemento quieres dar estilo.

**Propiedades.** Maneras en las cuales puedes dar estilo a un elemento HTML. (En este caso, `color` es una propiedad del elemento p.) En CSS, seleccionas que propiedad quieres afectar en tu regla.

**Valor de la propiedad.** A la derecha de la propiedad, después de los dos puntos (:), tenemos el valor de la propiedad, para elegir una de las muchas posibles apariencias para una propiedad determinada (hay muchos valores para `color` además de `red`).

Nota las otras partes importantes de la sintaxis:

Cada una de las reglas (aparte del selector) deben estar encapsuladas entre corchetes (`{}`).

Dentro de cada declaración, debes usar los dos puntos (`:`) para separar la propiedad de su valor.

Dentro de cada regla, debes usar el punto y coma (;) para separar una declaración de la siguiente.

Para modificar varios valores de propiedad a la vez, solo necesitas escribirlos separados por punto y coma (;), así:

```

1 | p {
2 |   color: red;
3 |   width: 500px;
4 |   border: 1px solid black;
5 | }
```

Fig. 19: Utilización de reglas CSS

Fuente: Los Autores

## Aplicaciones de hojas de estilo cascada

A continuación, se describen una serie de estructuras lógicas que permiten la utilización de hojas de estilo.

```

body {
  font-family: "Century Gothic";
  margin: 100px auto;
  width: 450px;
}
h1 {
  margin-bottom: 50px;
}
label {
  font-size: 1.5em;
  color: gray;
}
input[type=checkbox] {
  display:none;
}
input[type=checkbox] + label {
  cursor: pointer;
}
```



```

label:before {
  content: "";
  background: transparent;
  border: 4px solid #88c64b;
  text-shadow: 4px -2px 3px gray;
  border-radius: 25px;
  display: inline-block;
  height: 95px;
  margin-right: 20px;
  text-align: center;
  vertical-align: middle;
  width: 95px;
}
input[type=checkbox]:checked + label:before {
  content: '✓';
  font-size: 85px;
  color: #88c64b;
  font-family: "Times New Roman";
}
    
```

Resultado de aplicación de estilos

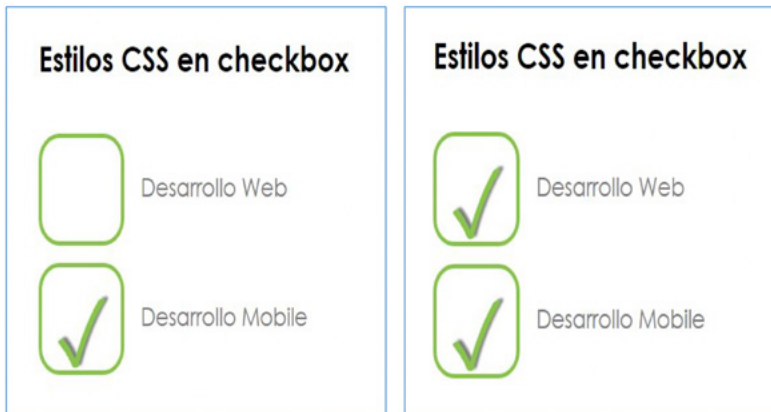


Fig. 20: Formularios con estilos CSS

Fuente: Los Autores

Al tratarse de aplicaciones híbridas que se ejecutan en dispositivos móviles y que se construyen a partir de tecnologías web estándar los ejercicios y manejo de códigos son fundamentales para desarrollar aplicaciones para ejecutarse en teléfonos inteligentes.

## DESARROLLO DE APLICACIONES HÍBRIDAS

Las tecnologías con las que se pueden desarrollar aplicaciones web así como aplicaciones móviles híbridas son variadas por lo que la elección de una determinada herramienta queda a discreción del grado de dominio, experiencia y conocimiento del lector.

“Las tecnologías que se utilizan parten de Node.js son una librería y entorno de ejecución de E/S dirigida por eventos y por lo tanto asíncronica que se ejecuta sobre el intérprete de JavaScript creado por Google V8” (NetConsulting, 2015).

La instalación del framework NodeJs se lo puede realizar desde la página web oficial <https://nodejs.org/en/>, de tener cualquier inconveniente revisar la documentación y soporte que dispone el sitio web en la instalación.

Una vez descargado ingresar a la consola y comandos DOS y ejecutar la siguiente línea de comando.

```
npm -g install ionic cordova
```

Al trabajar con aplicaciones híbridas es necesario tener las plantillas Starter de ionic. Existe plantillas predefinidas en ionic como:

- Blank
- Tabs
- Sidemenu

Dependiendo del contexto y necesidad y objetivo de la aplicación móvil se debe utilizar una determinada plantilla.

Para crear una nueva app en una consola:

```
ionic start [nombre_app] [plantilla]
ionic start MiAplicacion blank
ionic start MiAplicacion tabs
ionic start MiAplicacion sidemenu
```

Una vez creada la app se debe probar que este todo correcto para lo cual se debe ingresar a la carpeta de la app:

```
cd MiAplicacion
```

Luego se debe dirigir al directorio

```
ionic serve
```

Donde se puede visualizar la aplicación que se acaba de generar  
Navegación entre las páginas

Para regenerar una nueva aplicación es necesario ejecutar la siguiente línea de comandos.

```
ionic start demo_navegacion blank
cd demo_navegacion
```

Para probar la aplicación se debe verificar el estado del servicio.

```
ionic serve
```

Para crear una nueva página se debe ejecutar la siguiente línea de comandos:

```
ionic generate page página2
ionis generate page página3
```

Las páginas se pueden agregar en función al tipo de aplicación que se está construyendo, esta página debe incrementarse dentro de la definición del módulo de ionic, dentro d archivo:

```
src/app/app.module.ts
```

```
import { BrowserModule } from '@angular/platform-browser';
import { ErrorHandler, NgModule } from '@angular/core';
import { IonicApp, IonicErrorHandler, IonicModule } from 'ionic-angular';
import { SplashScreen } from '@ionic-native/splash-screen';
import { StatusBar } from '@ionic-native/status-bar';
```

```
import { MyApp } from './app.component';
import { HomePage } from './pages/home/home';
import { Pagina2Page } from './pages/pagina2/pagina2';
import { Pagina3Page } from './pages/pagina3/pagina3';
```

```
@NgModule({
  declarations: [
    MyApp,
    HomePage,
    Pagina2Page,
    Pagina3Page
  ],
  imports: [
    BrowserModule,
    IonicModule.forRoot(MyApp)
  ],
  bootstrap: [IonicApp],
  entryComponents: [
    MyApp,
    HomePage,
    Pagina2Page,
    Pagina3Page
  ],
  providers: [
    StatusBar,
    SplashScreen,
    {provide: ErrorHandler, useClass: IonicErrorHandler}
  ]
})
export class AppModule {}
```

## Navegación entre páginas

Para navegar por las páginas se usa la palabra clave `setRoot` para establecer si una página adquiere el contexto de principal `push` para sobreponer una página en el historial de navegación y mostrar porque está para quitar la página en el tope de navegación. `Navparams` permite enviar un objeto desde una página hacia otro considerando que `navCtrl` es un objeto de `NavController`.

```

this.navCtrl.setRoot('Pagina2Page');
this.navCtrl.push('Pagina3Page');
this.navCtrl.pop();
    
```

## Enviando un parámetro

```

this.navCtrl.setRoot('Pagina2Page', { nombre:'Juan', apellido:'Pérez'
});
this.navCtrl.push('Pagina3Page', { nombre:'Juan', apellido:'Pérez' });
this.navCtrl.pop();
    
```

## Resultado

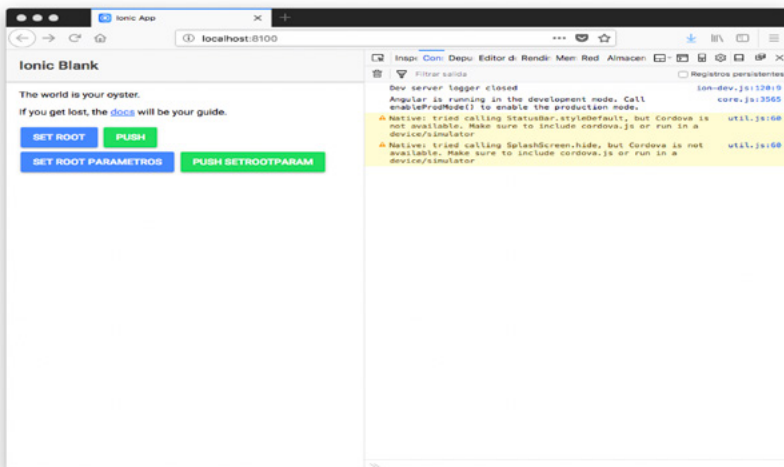


Fig. 21: Navegación entre páginas  
Fuente: Los Autores

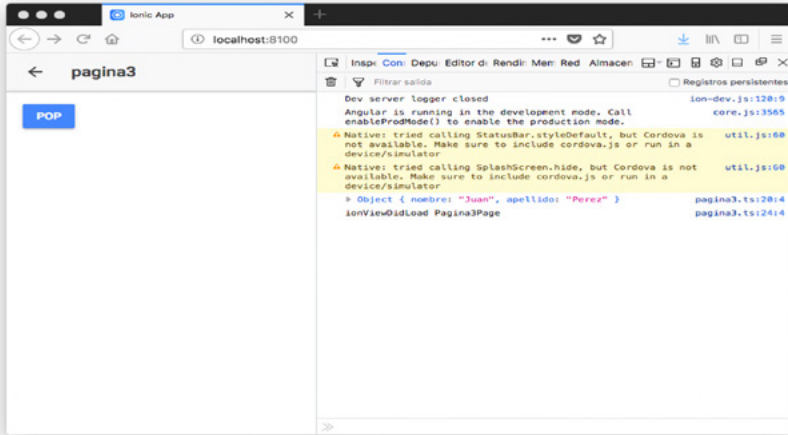


Fig. 22: Navegación entre páginas secuencias  
Fuente: Los Autores

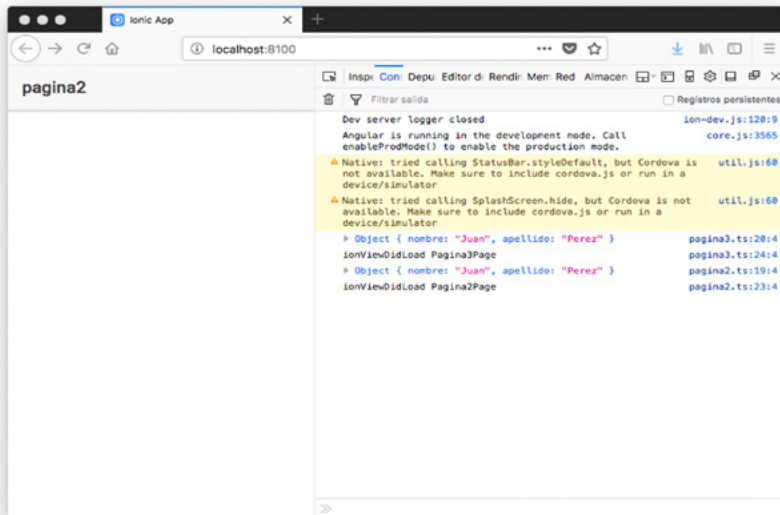


Fig. 23: Navegación páginas depuración  
Fuente: Los Autores

## Home.html

```

<ion-header>
  <ion-navbar>
    <ion-title>
      Ionic Blank
    </ion-title>
  </ion-navbar>
</ion-header>

<ion-content padding>
  The world is your oyster.
  <p>
    If you get lost, the <a href="http://ionicframework.com/docs/v2">-
docs</a> will be your guide.
  </p>
  <button ion-button color='primary' (click)="SetRoot()">
    Set Root
  </button>
  <button ion-button color='secondary' (click)="PushPage()" >
    Push
  </button>
  <br>
  <button ion-button color='primary' (click)="SetRootWParam()">
    Set Root Parametros
  </button>
  <button ion-button color='secondary' (click)="PushPageWParam()"
>
    Push SetRootParam
  </button>
</ion-content>

```



## home.ts

```

import { Component } from '@angular/core';
import { NavController } from 'ionic-angular';
import { Pagina2Page } from '../pagina2/pagina2';
import { Pagina3Page } from '../pagina3/pagina3';
@Component({
  selector: 'page-home',
  templateUrl: 'home.html'
})
export class HomePage {
  constructor(public navCtrl: NavController) {
  }
  SetRoot(){
    this.navCtrl.setRoot(Pagina2Page);
  }
  PushPage(){
    this.navCtrl.push(Pagina3Page);
  }
  SetRootWParam(){
    this.navCtrl.setRoot(Pagina2Page, { nombre: 'Juan', apellido: 'Perez'
  });
  }
  PushPageWParam(){
    this.navCtrl.push(Pagina3Page, { nombre: 'Juan', apellido: 'Perez' });
  }
}

```

## página3.html

```

<ion-header>
  <ion-navbar>
    <ion-title>pagina3</ion-title>
  </ion-navbar>
</ion-header>
<ion-content padding>
  <button color="primary" ion-button (click)="PopPage()">
    Pop
  </button>
</ion-content>

```

## página3.ts

```

import { Component } from '@angular/core';
import { IonicPage, NavController, NavParams } from 'ionic-angular';

/**
 * Generated class for the Pagina3Page page.
 *
 * See https://ionicframework.com/docs/components/#navigation for
 * more info on
 * Ionic pages and navigation.
 */

@Component({
  selector: 'page-pagina3',
  templateUrl: 'pagina3.html',
})
export class Pagina3Page {

  constructor(public navCtrl: NavController, public NavParams: NavParams) {
    console.log(NavParams.data);
  }

  ionViewDidLoad() {
    console.log('ionViewDidLoad Pagina3Page');
  }

  PopPage(){
    this.navCtrl.pop();
  }
}

```

## Formularios de aplicaciones móviles

Para trabajar con formularios de las aplicaciones móviles se debe empezar por crear la aplicación app-Movil-Sample

```
ionic start demo_formulario blank  
cd demo_formulario
```

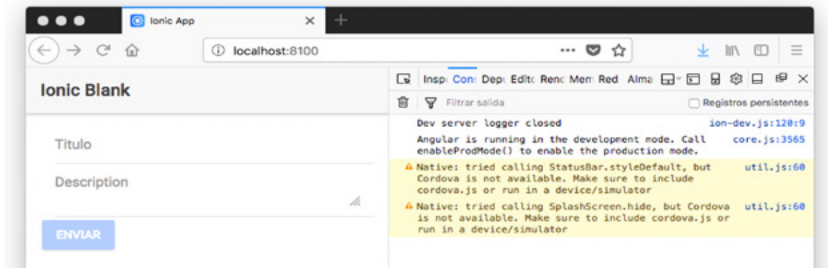


Fig. 24: Formularios app  
Fuente: Los Autores

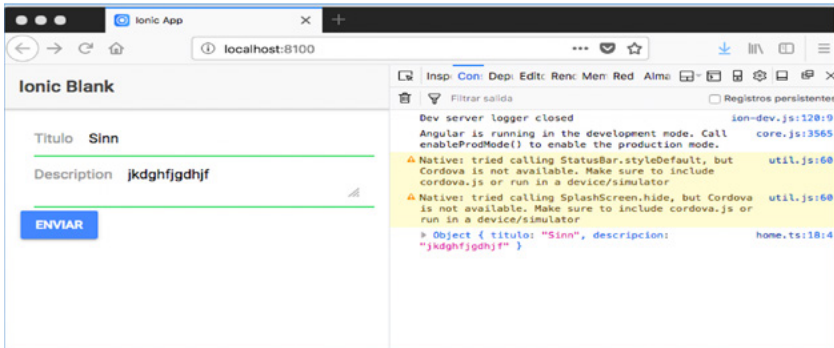


Fig. 25: Formularios app móvil ingreso datos  
Fuente: Los Autores

## home.html Formulario

```
<ion-header>
  <ion-navbar>
    <ion-title>
      Ionic Blank
    </ion-title>
  </ion-navbar>
</ion-header>

<ion-content padding>
  <form [formGroup]="formdata" (ngSubmit)="enviarForm()">
    <ion-item>
      <ion-label>Titulo</ion-label>
      <ion-input type="text" formControlName="titulo"></ion-input>
    </ion-item>
    <ion-item>
      <ion-label>Description</ion-label>
      <ion-textarea formControlName="descripcion"></ion-textarea>
    </ion-item>
    <button ion-button type="submit" [disabled]="!formdata.valid">En-
    viar</button>
  </form>
</ion-content>
```

## home.ts

```
import { Component } from '@angular/core';
import { NavController } from 'ionic-angular';
import { Validators, FormBuilder, FormGroup } from '@angular/forms';

@Component({
  selector: 'page-home',
  templateUrl: 'home.html'
})
export class HomePage {
  private formdata : FormGroup;
  constructor(public navCtrl: NavController,private formBuilder: FormBuilder) {

    this.formdata = this.formBuilder.group({
      titulo: ['', Validators.required],
      descripcion: ['', Validators.required],
    });
  }
  enviarForm(){
    console.log(this.formdata.value)
  }
}
```

## BookDIRECTORYDEMO

El desarrollo de aplicaciones móviles híbridas radica en la tecnología web, así como en los lenguajes de programación y de las tecnologías que se requieran para el desarrollo y empaquetamiento de la aplicación.

A continuación, se detalla las tecnologías para el desarrollo Tecnología:

- Ionic
- Firebase

Descripción: Aplicación de un directorio de negocios en base de datos firebase que se detalla a continuación

### Configuración del Proyecto de Firebase

Crear un nuevo proyecto en Firebase Console

The screenshot shows the 'Agregar un proyecto' (Add a project) dialog in the Firebase Console. It contains the following elements:

- Nombre del proyecto:** A dropdown menu with the value 'firebase-sample-directory'.
- ID del proyecto:** A text input field with the value 'fir-sample-...' and a pencil icon for editing.
- País/Región:** A dropdown menu with the value 'Ecuador'.
- Sugerencia:** A note that says 'Sugerencia: Los proyectos llevarán las apps a distintas plataformas' with a help icon.
- Buttons:** 'CANCELAR' and 'CREAR PROYECTO'.

Fig. 26: Creando proyecto en FirebaseConsole

Fuente: Los Autores

Una vez creado el proyecto, creamos la base de datos necesaria para la gestión de la información, así como de los contenidos de la aplicación.



Fig. 27: Creando base de datos  
Fuente: Los Autores

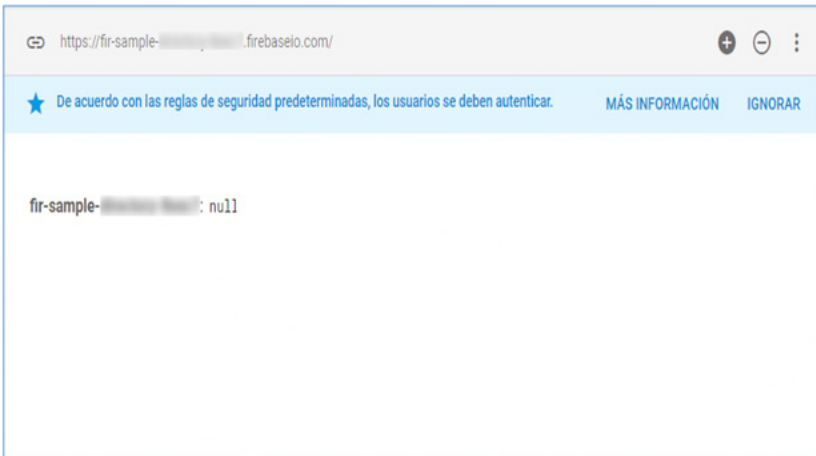


Fig. 28: Configuración seguridad de base de datos  
Fuente: Los Autores

## Almacenamiento persistente en dispositivos móviles

El almacenamiento persistente en dispositivos móviles se lo puede realizar mediante IndexedDb, el cual puede ser usada como una base de datos para

dispositivos móviles, permitiendo de esta manera almacenar múltiples campos a los cuales se puede acceder mediante índices.

Gracias a IndexedDB podemos almacenar todos los datos del usuario de manera local, mostrar el resultado de sus acciones casi al instante y poder mandar los resultados cuando haya una mejor conexión, permitiendo de esta manera trabajar de manera offline. (MDN web docs, 2018).

**IndexedDB está hecho sobre un modelo de base de datos transaccional.** Todo lo que usted haga en IndexedDB siempre ocurre en el contexto de una transacción. La API de IndexedDB provee una gran cantidad de objetos que representan índices, tablas, cursores, y más, pero cada uno de ellos están atados a una transacción particular. Así, usted no puede ejecutar ningún comando o abrir un cursor fuera de una transacción.

Las transacciones tienen un período de vida bien definido. Por esta razón, cualquier intento de utilizar una transacción que ya se ha completado ocasionará excepciones. Igualmente, las transacciones aplican los cambios automáticamente y no se puede hacer commit automáticamente.

Este modelo de transacciones es realmente útil cuando se considera qué podría pasar si un usuario abre dos instancias de una aplicación simultáneamente en dos pestañas. Sin operaciones transaccionales, las dos instancias podrían hacer que los cambios se pisaran o se sobrescribieran entre sí. Si no está familiarizado con el concepto de transacción en una base de datos, lea el artículo correspondiente en la Wikipedia y transacción en la sección de definiciones.

**La API de IndexedDB es mayormente asíncrona.** La API no devuelve datos regresando un valor; en cambio, es necesario pasarle una función como callback. Usted no “guarda” un valor en la base de datos, o “lee” un valor de la misma por medios síncronos. En cambio, usted “solicita” una operación a la base de datos. Un evento del DOM es utilizado para notificarle que la operación terminó, y el tipo de evento recibido le permite saber si ésta falló o si fue exitosa. Esto puede sonar complicado al comienzo, pero hay algunas medidas de sanidad incluidas por defecto. Esto no es muy distinto de cómo funciona XMLHttpRequest.



**IndexedDB usa solicitudes en todos lados.** Las solicitudes (requests) son objetos que reciben los eventos del DOM mencionados previamente. Éstas cuentan con las propiedades `onsuccess` y `onerror`, sobre las cuales se puede aplicar `addEventListener()` y `removeEventListener()`. Estas también tienen las propiedades `readyState`, `result`, y `errorCode` que permiten conocer el estado de la solicitud. La propiedad `result` es particularmente mágica, dado que puede ser muchas cosas distintas, dependiendo de cómo se generó la solicitud (por ejemplo, una instancia de `IDBCursor`, o la llave de un valor recién insertado en la base de datos).

**IndexedDB usa eventos DOM para notificar cuando los resultados están disponibles.** Los eventos del DOM siempre tienen una propiedad `type` (en IndexedDB, ésta es generalmente “`success`” o “`error`”). Los eventos del DOM también tienen una propiedad `target` que dice a dónde apunta el evento. En la mayoría de los casos, el `target` de un evento es el objeto `IDBRequest` que se generó como resultado de una operación sobre la base de datos. Los eventos exitosos no son pasados a los padres y no pueden ser cancelados. Por otro lado, los eventos de error son pasados a los padres del `target` y pueden cancelarse. Esto es importante, dado que los eventos de error cancelan cualquier transacción sobre la que estén corriendo a menos que sean cancelados.

**IndexedDB es orientada a objetos.** IndexedDB no es una base de datos relacional, con tablas, filas y columnas. Esta diferencia fundamental e importante afecta la manera como usted diseña e implementa sus aplicaciones.

En un almacén de datos relacional, usted podría tener una tabla que almacena una colección de filas de datos y columnas de tipos de datos con un nombre. IndexedDB, por otro lado, necesita que usted cree un almacén de objetos para un tipo de datos y sencillamente almacena objetos JavaScript en ese almacén. Cada almacén de objetos puede tener una colección de índices que hacen que la iteración y la búsqueda de elementos sea más eficiente. Si usted no está familiarizado con los sistemas de manejo de datos orientados a objetos.

**IndexedDB no utiliza SQL (Structured Query Language).** En cambio, usa consultas sobre un índice que producen un cursor. Éste puede utilizarse para iterar sobre el conjunto de resultados. Si no está familiarizado con sistemas NoSQL.

**IndexedDB se adhiere a una política de mismo origen.** Un origen es el dominio, protocolo de la capa de aplicación, y el puerto de la URL del documento donde se está ejecutando la aplicación. Cada origen tiene su propio conjunto de bases de datos asociado. Cada base de datos tiene un nombre que la identifica dentro de un origen (MDN web docs, 2018)

### Inicialización de una base de datos IndexedDb

```

window.indexedDB = window.indexedDB || window.mozIndexedDB ||
window.webkitIndexedDB || window.msIndexedDB;
window.IDBTransaction = window.IDBTransaction || window.webkitI
DBTransaction || window.msIDBTransaction;
window.IDBKeyRange = window.IDBKeyRange || window.webkitID
BKeyRange || window.msIDBKeyRange;
if (!window.indexedDB) {
    window.alert("Error de soporte.");
}
    
```

### Abriendo y escritura una base de datos IndexedDb

```

var db;
var request = window.indexedDB.open("newDatabase", 1);
request.onerror = function(event)
{

var active = dataBase.result;

var object = active.createObjectStore('people', { keyPath : 'id', autoIn
crement : true });

object.createIndex('by_name', 'name', { unique : false });
};
    
```

```

request.onsuccess = function(event) {
var active = dataBase.result;
var data = active.transaction(["people"], "readwrite");
var almacen = data.objectStore("people");
var respuesta =almacen.put({
                                nombre :”Juan”,

respuesta.onerror = function (e) {

    alert(respuesta.error.name + '\n\n' + respuesta.error.message);
    };

    data.oncomplete = function (e) {

    };
};

```

## Ventajas de IndexedDB

- Base de datos multiplataforma
- Almacenamiento sin conexión

## Estructura de base de datos

La estructura de la base de datos debe estar organizada con base a los requerimientos necesarios para la transacción de la información, datos y contenidos que va a utilizar y consumir la aplicación móvil.

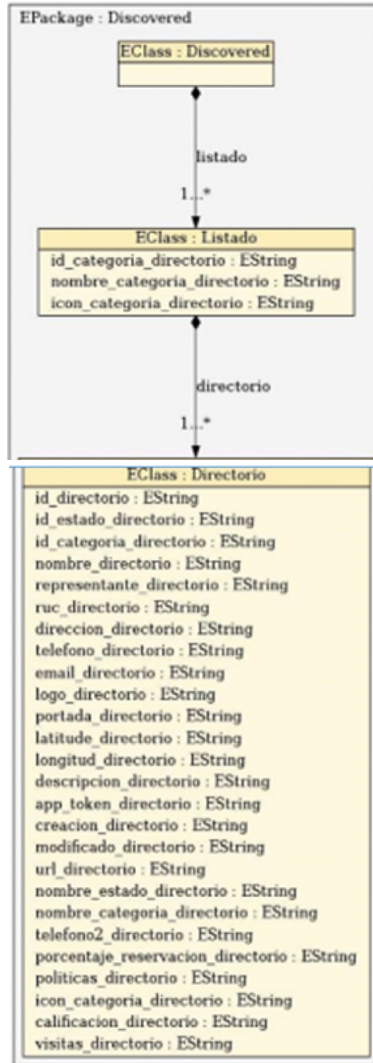


Fig. 29: Estructura de base de datos

Fuente: Los Autores

Una vez definida la base de datos de la aplicación se debe subir el archivo denominado para el ejemplo <directorio.json> que contendrá la información de prueba del proyecto.

Para ello es necesario importar dicho archivo como se muestra en la siguiente figura:

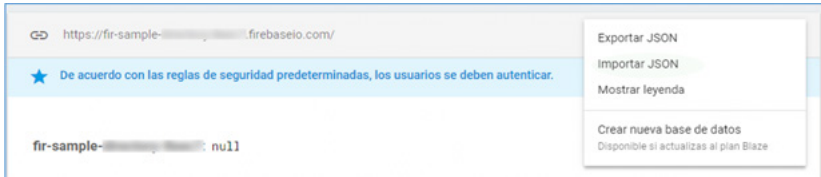


Fig. 27: Subiendo la base de datos  
Fuente: Los Autores

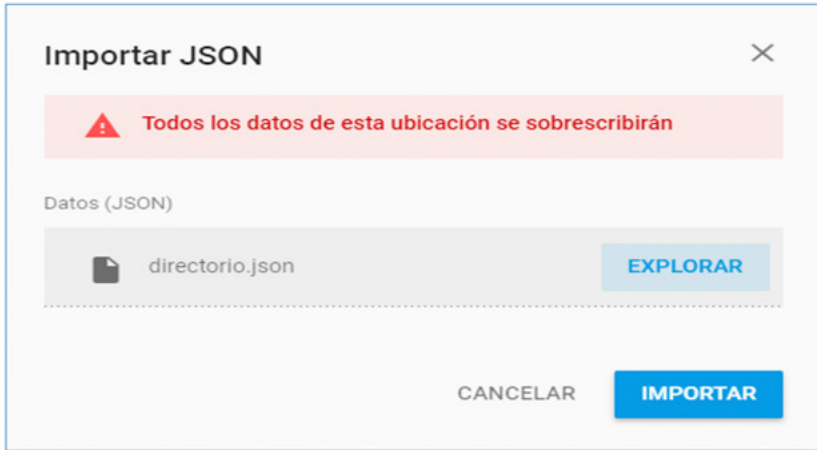


Fig. 31: Importar archivo Datos.Json  
Fuente: Los Autores

Una vez subida la base de datos en la herramienta Firebase, debe quedar la base de datos de la siguiente manera:



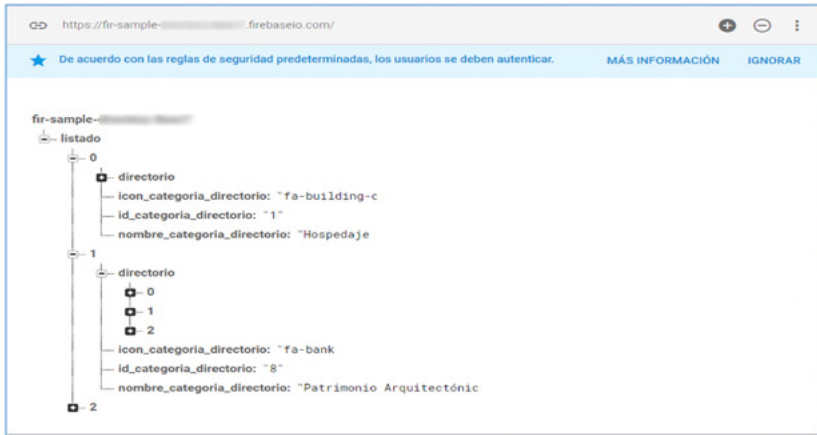


Fig. 32: Datos.Json

Fuente: Los Autores

## Configuración de seguridad

La seguridad es un aspecto importante en el desarrollo de aplicaciones móviles por lo tanto las acciones y configuraciones que se deben realizar son:

Ingresar al menú lateral y dar click en la opción "Reglas"

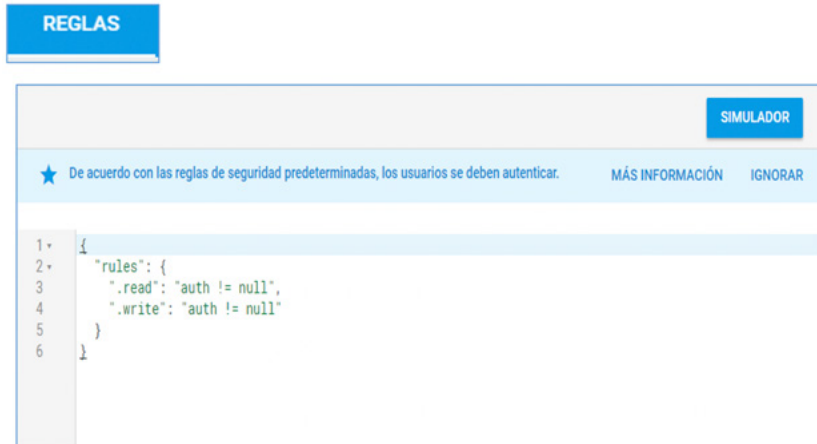


Fig. 33: Reglas de seguridad de la aplicación móvil

Fuente: Los Autores

Esta sección corresponde al nivel de seguridad que se le debe asignar a la aplicación móvil sobre la manipulación de los datos e información de la BDD, por motivos de demostración se dejará abierta, sin embargo, este aspecto debe ser configurado según seguridades y las consideraciones del desarrollador.

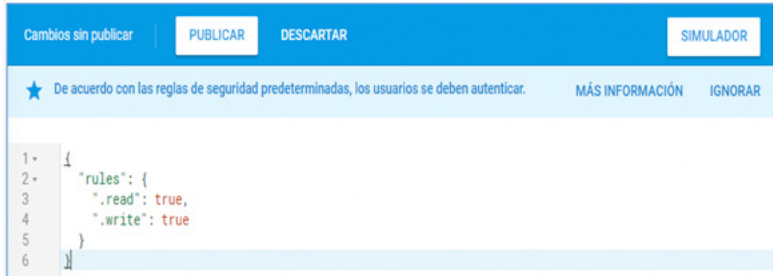


Fig. 34: Configuraciones de seguridad app  
Fuente: Los Autores

El resultado de las configuraciones debe quedar como se muestra en la siguiente figura:

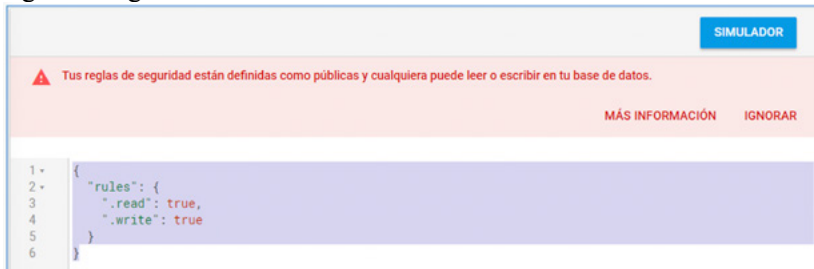


Fig. 35: Importar archivo estructura Datos.Json  
Fuente: Los Autores

Una vez realizado las configuraciones de seguridad así el aspecto de base de datos se debe hacer click en **“Project Overview”**

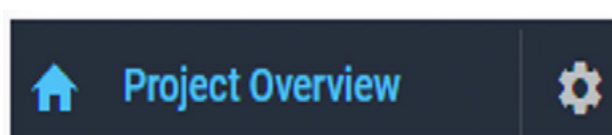


Fig. 36: Cargar proyecto  
Fuente: Los Autores

Una vez ingresado en el proyecto nuevo se debe dar click en “Agregar Firebase a la aplicación web”



Fig. 37: Agregara proyecto a Firebase  
Fuente: Los Autores

Una vez agregada la aplicación se mostrará las configuraciones que se deben incluir en la aplicación ionic <Aplicación Móvil>



Fig. 38: Configuraciones del proyecto en Firebase  
Fuente: Los Autores

Por cuestiones de configuraciones se han borrado las claves debido a que es información privada y por lo tanto no es posible indicar en este apartado, sin embargo, los datos ocultos es la información de configuraciones, acceso a aplicaciones, puertos y demás aspectos de acceso a servidores.



## CONSTRUCCIÓN DE APLICACIONES MÓVILES HÍBRIDAS

Para el desarrollo de las aplicaciones móviles híbridas se debe ingresar al framework Ionic y digitar el siguiente comando

```
ionic start BookDemoDirectory sidemenu
ionic serve
```

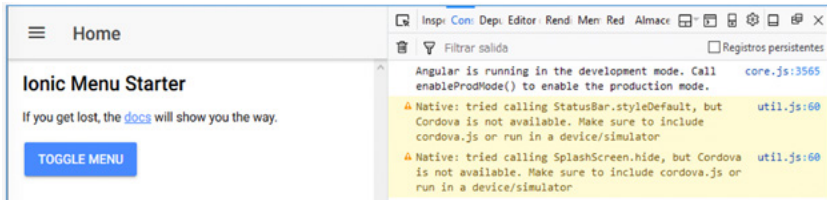


Fig. 39: Crear App Móvil Híbrida

Fuente: Los Autores

Una vez iniciado nuestro proyecto de aplicaciones móviles se deben agregar las librerías de Firebase a nuestra aplicación dentro del proyecto de Ionic. Para ello se debe ejecutar la siguiente línea de comando:

```
npm install angularfire2 firebase
```

Ahora bien, en el archivo

```
src/app/app.module.ts
```

Se debe importar la librería de AngularFire2 además de la configuración de la base de datos de firebase vista en la sección de **<Estructura de base de datos>**

```

1 import { BrowserModule } from '@angular/platform-browser';
2 import { ErrorHandler, NgModule } from '@angular/core';
3 import { IonicApp, IonicErrorHandler, IonicModule } from 'ionic-angular';
4
5 import { MyApp } from './app.component';
6 import { HomePage } from '../pages/home/home';
7 import { ListPage } from '../pages/list/list';
8
9 import { StatusBar } from '@ionic-native/status-bar';
10 import { SplashScreen } from '@ionic-native/splash-screen';
11
12 // Modulos de Angular Firebase
13 import { AngularFireModule } from 'angularfire2';
14 import { AngularFireDatabaseModule } from 'angularfire2/database';
15
16 // Configuración de Base Firebase
17 export const firebaseConfig = {
18   apiKey: "AIza...",
19   authDomain: "fir-sample-...firebaseapp.com",
20   databaseURL: "https://fir-sample-...firebaseio.com",
21   projectId: "fir-sample-...",
22   storageBucket: "fir-sample-...appspot.com",
23   messagingSenderId: "4...1"
24 };

```

Fig. 40: Configuración de estructura de base de datos app

Fuente: Los Autores

Una vez importadas las librerías se deben inicializar los módulos de AngularFire que se detallan a continuación:

```

5 @NgModule({
6   declarations: [
7     MyApp,
8     HomePage,
9     ListPage
10  ],
11   imports: [
12     BrowserModule,
13     IonicModule.forRoot(MyApp),
14     AngularFireModule.initializeApp(firebaseConfig),
15     AngularFireDatabaseModule
16  ],
17   bootstrap: [IonicApp],

```

Fig. 41: Proceso para incorporar módulos de AngularFire

Fuente: Los Autores

Ahora se debe crear las interfaces de la aplicación móvil que se va a desarrollar en este contexto se debe ingresar al archivo:

```
src/pages/home/home.html
```

Es necesario eliminar el contenido de dicho archivo `< src/pages/home/home.html >`, con la finalidad de tener una estructura adecuada para el manejo de los datos e información de la aplicación.

```
<ion-header>
  <ion-navbar>
    <button ion-button menuToggle>
      <ion-icon name="menu"></ion-icon>
    </button>
    <ion-title>Home</ion-title>
  </ion-navbar>
</ion-header>

<ion-content padding>
  <h3>Ionic Menu Starter</h3>

  <p>
    If you get lost, the <a href="http://ionicframework.com/docs/v2">-
    docs</a> will show you the way.
  </p>

  <button ion-button secondary menuToggle>Toggle Menu</button>
</ion-content>
```

Una vez eliminado el archivo `< src/pages/home/home.html >`, se lo debe reemplazar por una lista como se muestra:

```

<ion-content padding>
  <ion-list>
    <ion-item *ngFor="let categoria of listado | async">
      {{categoria.nombre_categoria_directorio}}
    </ion-item>
  </ion-list>
</ion-content>

```

Además, para el manejo de las categorías para el ejemplo se aconseja utilizar un arreglo denominado <listado> para las categorías de la base de datos. Para ello se debe ingresar al archivo.

```
src/pages/home/home.ts
```

En el archivo se debe agregar el módulo <Angular-fire>

```
import { AngularFireDatabase, AngularFireList } from 'angularfire2/
database';
```

Creamos dos variables para la clase HomePage

```
listadoRef: AngularFireList<any>;
listado:Observable<any[]>;
```

Estas variables permitirán navegar por los datos del listado y agregamos la definición del angularfiredatabase al constructor.

```

export class HomePage {
  listadoRef: AngularFireList<any>;
  listado: Observable<any[]>;
  constructor(public navCtrl: NavController,
    public afDatabase: AngularFireDatabase
  ) {

  }

}

```

Lo que resta es importar los datos de la base de datos de firebase

```

this.listadoRef = this.afDatabase.list('/listado');
this.listado = this.listadoRef.valueChanges();

```

Código Completo de Home.ts

```

import { Component } from '@angular/core';
import { NavController } from 'ionic-angular';
import { AngularFireDatabase, AngularFireList } from 'angularfire2/
database';
import { Observable } from 'rxjs/Observable';

@Component({
  selector: 'page-home',
  templateUrl: 'home.html'
})
export class HomePage {
  listadoRef: AngularFireList<any>;
  listado: Observable<any[]>;
  constructor(public navCtrl: NavController,
    public afDatabase: AngularFireDatabase
  ) { this.listadoRef = this.afDatabase.list('/listado');
    this.listado = this.listadoRef.valueChanges();
  }
}

```

Si todas las configuraciones y cambios se han hecho de manera correcta en el visor de prueba se debe mostrar las categorías que tiene la aplicación móvil como se muestra en la siguiente figura.

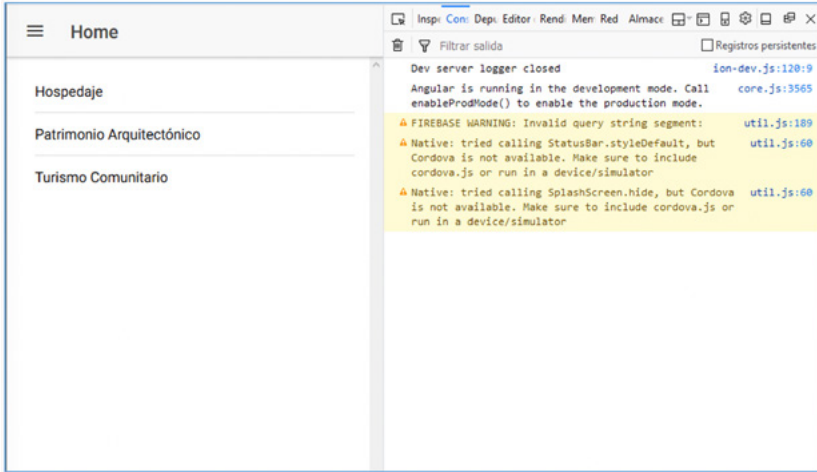


Fig. 42: Crear categorías app  
Fuente: Los Autores

Ahora que ya se han cargado las categorías del ejemplo se debe agregar funcionalidades a la aplicación móvil, para ello se procede a crear dos páginas, una para el detalle de la categoría y otra para el registro del directorio, para ello se debe ejecutar la siguiente línea de comando.

```
ionic generate categoriadetalle
```

```
ionic generate directoriorecord
```

Luego se deben importar las páginas al archivo `src/app/app.module.ts` y se agrega el módulo a la aplicación.

```
import { CategoriadetallePage } from '../pages/categoriadetalle/cate
goriadetalle';
import { DirectoriorecordPage } from '../pages/directoriorecord/direc
toriorecord';
```

```
@NgModule({
  declarations: [
    MyApp,
    HomePage,
    ListPage,
    CategoriadetallePage,
    DirectoriorecordPage
  ],
  imports: [
    BrowserModule,
    IonicModule.forRoot(MyApp),
    AngularFireModule.initializeApp(firebaseConfig),
    AngularFireDatabaseModule
  ],
  bootstrap: [IonicApp],
  entryComponents: [
    MyApp,
    HomePage,
    ListPage,
    CategoriadetallePage,
    DirectoriorecordPage
  ], providers: [ StatusBar,
    SplashScreen,
    {provide: ErrorHandler, useClass: IonicErrorHandler}
  ])export class AppModule {}
```

Una vez realizada la codificación requerida en el archivo Home.html se realiza la definición de la lista que se agregó mediante un evento click.

```
<ion-item *ngFor="let categoria of listado | async" (click)="ItemCate
goriaClick(categoria)">
```

Ahora en el archivo Home.ts se debe agregar el módulo CategoriadetallePage

```
import { CategoriadetallePage } from './categoriadetalle/categoriadetalle';
```

Se asocia y crea el evento click

```
ItemCategoriaClick(item){
    console.log(item);
    this.navCtrl.push(CategoriadetallePage,item);
}
```

Esto permite abrir la página categoriadetalle enviado el directorio de la categoría correspondiente a nuestra aplicación móvil como se muestra en la siguiente figura:

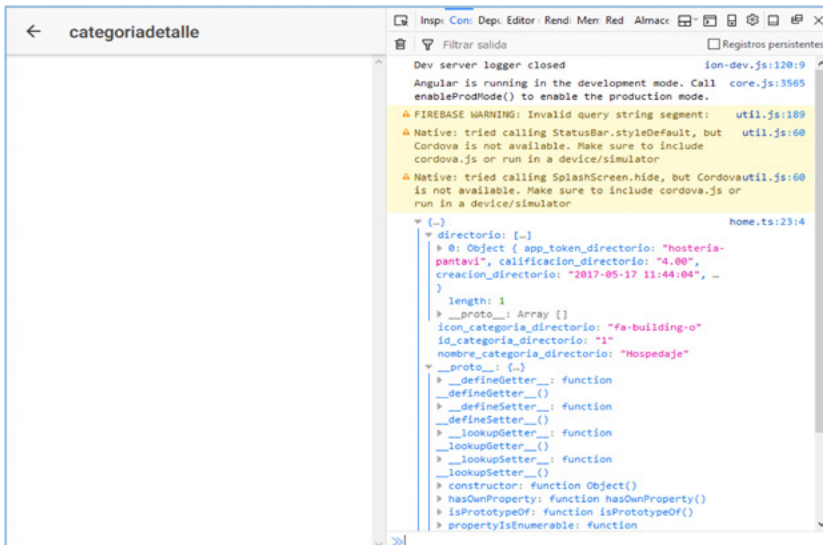


Fig. 43: Crear categorías app

Fuente: Los Autores



```
src/pages/categoriadetalle/categoriadetalle.htm
```

En el archivo se debe agregar una lista similar a la que se agregó al archivo **Home.html** descrito en la sección anterior.

```
<ion-list>
```

```
<ion-item *ngFor="let directorio of listado" (click)="ItemClick(directorio)">
  {{directorio.nombre_directorio}}
</ion-item>
</ion-list>
```

Esta codificación se la debe realizar sin sincronización debido a que no hay datos aún almacenados y asociados.

En el archivo

```
src/pages/categoriadetalle/categoriadetalle.ts
```

Se debe importar el módulo de **directoriorecord**

```
import { DirectoriorecordPage } from '../directoriorecord/directoriorecord';
```

Luego se debe declarar una variable de almacenamiento para el listado del parámetro de nuestra aplicación

```
listado:Observable<any[]>;
```

Luego se debe crear un constructor donde se va a recibir los datos que se deben agregar en el evento click

```

constructor(public navCtrl: NavController, public navParams: NavPa
rams) {
  this.listado = navParams.data.directorio;
  console.log(navParams.data.directorio);
} ItemClick(item){ console.log(item); this.navCtrl.push(Directorio
recordPage,item);
}

```

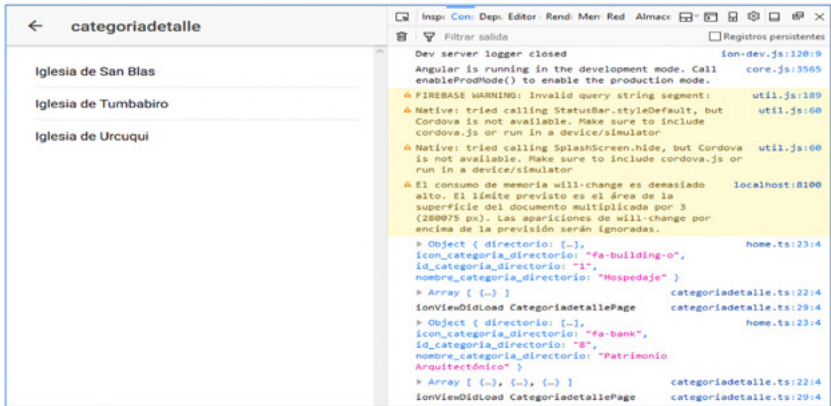


Fig. 44: Importar módulos de las categorías de la App  
Fuente: Los Autores

Ahora de igual forma se debe recibir la información en el módulo **paginarecord**

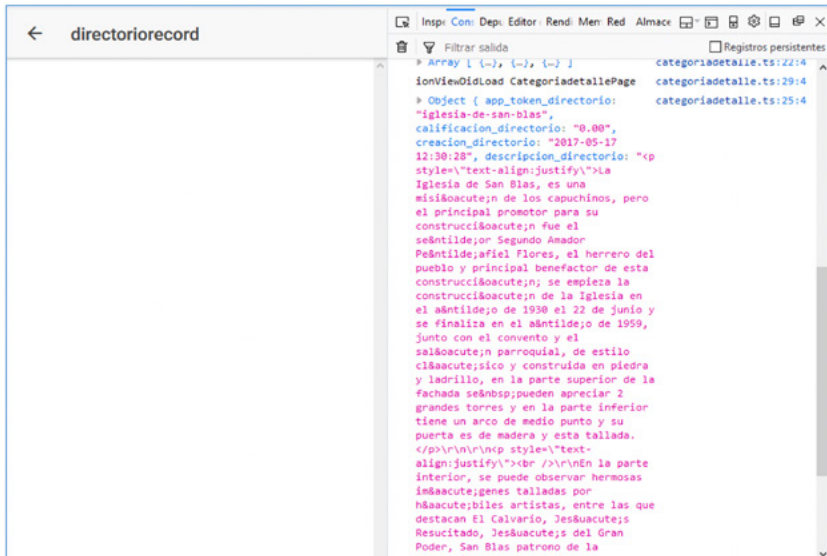


Fig. 45: Proceso de carga de datos módulos de la App

Fuente: Los Autores

Para la página `src/pages/directoriorerecord/directoriorerecord.html` se debe asignar el nombre del directorio en la barra de título

```
<ion-header>
  <ion-navbar>
    <ion-title>{{nombre_directorio}}</ion-title>
  </ion-navbar>
</ion-header>
```

En `src/pages/directoriorerecord/directoriorerecord.ts` se agrega la variable para el nombre del directorio y otra para recibir la información desde `catgoriadetalle`.

```
local_directorio:any;
nombre_directorio:string="";
```

Además, se debe crear el constructor donde se recibe la información requerida para la categoría detalle

```

this.local_directorio = navParams.data;
this.nombre_directorio = this.local_directorio.nombre_directorio;
    
```

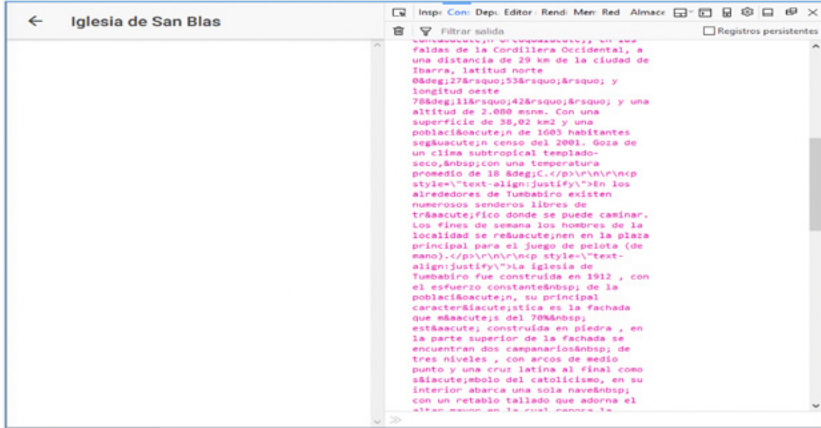


Fig. 46: Importar módulos de las categorías de la App  
 Fuente: Los Autores

En la sección anterior se ha detallado el procedimiento, configuraciones y codificación necesaria para el consumo de la información, además de una vista rápida del acceso y manejo de los datos ahora vamos a proceder a darle estilos a la aplicación, colores y forma:

El resultado de la codificación de los estilos debe quedar de la siguiente forma:

```
$colors: (  
  primary: #488aff,  
  secondary: #32db64,  
  danger: #f53d3d,  
  light: #f4f4f4,  
  dark: #222  
$toolbar-background: color($colors,primary);
```

De forma visual los resultados deben ser los siguientes:

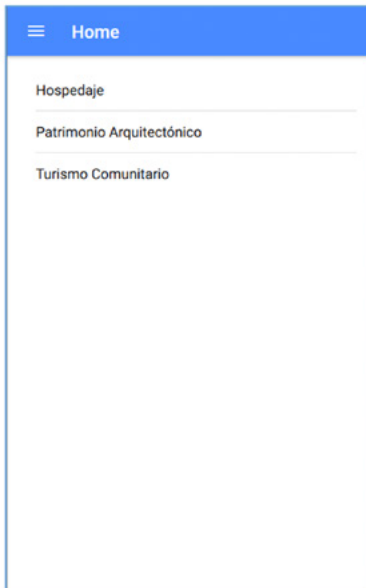


Fig. 47: Proceso de compilado de la App híbrida  
Fuente: Los Autores

Se puede cambiar el color primary a uno diferente. Por ejemplo:  
# 9e0c97

```

Colors: (
  primary: #9e0c97,
  secondary: #32db64,
  danger: #f53d3d,
  light: #f4f4f4,
  dark: #222
);
    
```

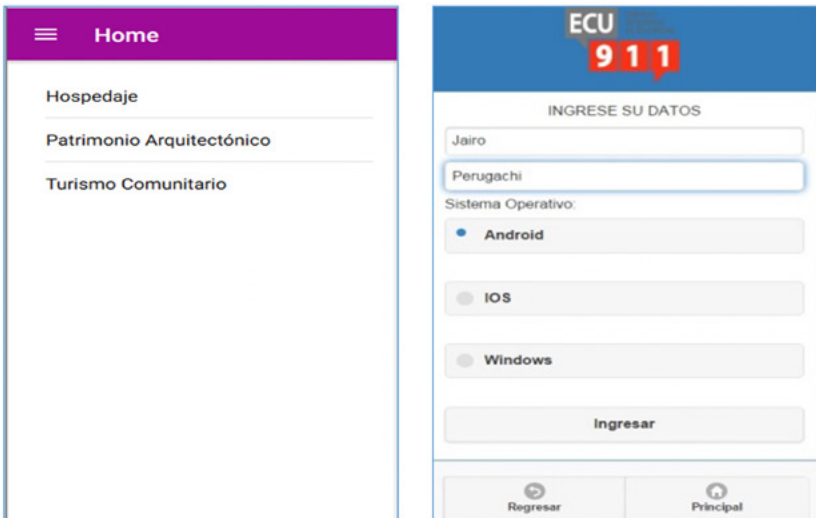


Fig. 48: Resultado configuración de colores app móvil  
Fuente: Los Autores

En categoriadetalle se debe agregar el nombre de la categoría en la cabecera de la página, para eso al igual que con directoriorecord se debe crear una variable y le asignamos;  
Categoriadetalle.html

```

<ion-header>
  <ion-navbar>
    <ion-title>{{nombre_categoria}}</ion-title>
  </ion-navbar>
</ion-header>

```

En el archive Categoriadetalle.ts

```

nombre_categoria:String="";
constructor(public navCtrl: NavController, public navParams: NavPa
rams) {
  this.listado = navParams.data.directorio;
  this.nombre_categoria = navParams.data.nombre_categoria_direc
torio;
  console.log(navParams.data.directorio);
}

```

El resultado de esta asignación se neutra en la siguiente figura:

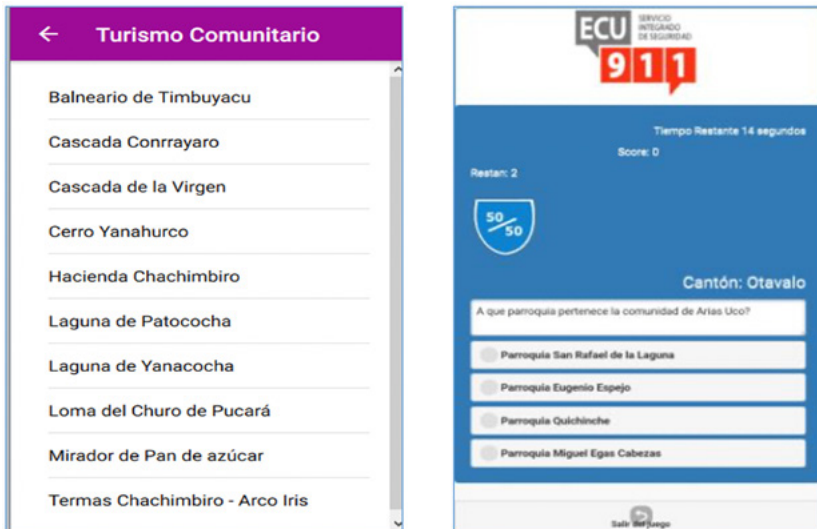


Fig. 49: Resultado de aplicar categorías a una aplicación móvil  
Fuente: Los Autores

Ahora se procede a poner estilo al directoriorecord, haciendo el uso de cards de ionic. Siguiendo la guía modificamos directoriorecord

```

<ion-content class="card-background-page">
  <ion-card>
    
    <div class="card-title">{{local_directorio.nombre_directorio}}</div>
    <div [innerHTML]="local_directorio.descripcion_directorio"></div>
  </ion-card>
</ion-content>

```

Vale la pena indicar que [innerHTML] permite desplegar información que contiene este formato HTML. El resultado de dichas operaciones sería:



Fig. 50: Despliegue de contenido HTML en aplicación móvil  
 Fuente: Los Autores



## AGREGANDO APP A LA PLATAFORMA ANDROID

Es necesario que esté el SDK de Android para esto se puede utilizar el instalador de Android Studio.

Es necesario digitar la siguiente línea de comando:

```
ionic cordova platform add Android
```

```
Installing "cordova-plugin-ionic-keyboard" for android
Adding cordova-plugin-ionic-keyboard to package.json

Saved plugin info for "cordova-plugin-ionic-keyboard" to config.xml

--save flag or autosave detected

Saving android@~6.3.0 into config.xml file ...

> ionic cordova resources android --force
✓ Collecting resource configuration and source images - done!
✓ Filtering out image resources that do not need regeneration - done!
✓ Uploading source images to prepare for transformations - done!
✓ Generating platform resources: 18 / 18 complete - done!
✓ Modifying config.xml to add new image resources - done!
```

Fig. 51: Instalación de plataforma Cordova a Ionic  
Fuente: Los Autores

Finalizada la instalación de la plataforma Android en la aplicación Ionic. Se procede a iniciar la aplicación para ello se debe ejecutar el siguiente comando en el emulador de Android Studio como se muestra en la siguiente figura:

```
ionic cordova run Android
```

El resultado debe quedar como se aprecia en la siguiente figura:



Fig. 52: Despliegue de la aplicación móvil  
Fuente: Los Autores

## PUBLICACIÓN DE LAS APLICACIONES

Una vez que se ha desarrollado la aplicación móvil y que esta se encuentre en funcionamiento, el siguiente paso es publicarla en las distintas tiendas oficiales como Google Play y App Store, entre otras, en este punto se debe tomar en cuenta que el proceso de publicación es la etapa final del desarrollo y por lo tanto la aplicación ya ha pasado la etapa de pruebas.

Las pruebas de la aplicación son aconsejables realizarlas en entornos reales mas no en los simuladores propios de las herramientas de desarrollo debido a que, los sistemas operativos de los teléfonos móviles tienen una versión específica de compilación, y por lo tanto este aspecto se debe tomar en cuenta a la hora de realizar la publicación de la aplicación móvil para los usuarios finales.

### Versión de lanzamiento de app

En el ámbito de tecnología todas las aplicaciones se manejan a partir de una versión, que es la combinación de cada uno de los artefactos compilados necesarios para lanzar la aplicación móvil, así como una actualización de la versión ya publicada para los usuarios.

Para ello primero necesitamos generar una versión de lanzamiento de la aplicación móvil dirigida a cada una de las plataformas de sistemas operativos O.S para los cuales se va a implementar.

Antes de empezar se debe asegurar de tener todos los recursos de diseño requeridos, así como de verificar que la aplicación cumpla con cada una de las políticas de publicación de cada una de las tiendas con el fin de que la app no sea rechazada o bloqueada en el proceso.

```
$ ionic cordova plugin rm cordova-plugin-console
```

**Nota:** El proceso de publicación en cada una de las tiendas Google Play y App Store, entre otras, requiere de una cuenta en la tienda, así como el pago de licencias y demás requisitos propios de cada una de estas.

Para conocer cada uno de los requerimientos ingrese a las siguientes direcciones electrónicas y verifique los aspectos requeridos:

<p>Google Play</p> 	<p><a href="https://play.google.com/apps/publish/signup/">https://play.google.com/apps/publish/signup/</a></p>
<p>Apple Store</p> 	<p><a href="https://developer.apple.com/">https://developer.apple.com/</a></p>

## Publicación de Android

Para generar una compilación de lanzamiento de la aplicación móvil para la plataforma de dispositivos Android, se debe ejecutar el siguiente comando desde la consola de cordova Cli.

```
$ ionic cordova build --release android
```

Este procedimiento generará una versión de lanzamiento basada en la configuración del archivo <connfig.xml>. se debe tomar en cuenta que la aplicación móvil tendrá valores preestablecidos por defecto en este archivo, de ser necesario usted puede personalizar la forma y procedimiento de cómo se compila la aplicación, para ello usted puede editar las configuraciones del archivo <connfig.xml>, consulte la documentación del archivo <connfig.xml> de ser necesario.

A continuación, encontraremos el archivo <APK> sin firmar, se lo encuentra dentro de la plataforma con la siguiente ruta:

```
platforms/android/build/outputs/apk.
```

Para el ejemplo el nombre de la aplicación móvil es: BookDIRECTORYDEMO, el cual debe modificarse en la siguiente ruta:

```
platforms/android/build/outputs/apk/ BookDIRECTORYDEMO-release-unsigned.apk.
```

Ahora, se procede a firmar la APK, a la vez ejecute las utilidades de alineación para la optimización de la aplicación en la tienda. Ahora se debe tener la clave para firmar la aplicación, para ello se debe realizar lo siguiente:

Se debe generar una clave privada para ello se ejecuta el comando keytool que por defecto viene con el JDK.

```
$ keytool -genkey -v -keystore my-release-key.keystore -alias alias_name -keyalg RSA -keysize 2048 -validity 10000
```

Lo primero que se debe hacer es crear una contraseña para el almacén de claves, además se debe responder a una serie de preguntas propias de la plataforma, al final de este procedimiento el archivo debe quedar.

```
my-release-key.keystore
```

**Nota:** El archivo creado debe guardarse en un directorio seguro ya que si se elimina o pierde usted no podrá enviar actualizaciones de la aplicación móvil

Para el proceso de firmar la aplicación se debe ejecutar la herramienta <jarsigner>, que viene incluida en el JDK.

```
$ jarsigner -verbose -sigalg SHA1withRSA -digestalg SHA1 -keystore
my-release-key.keystore HelloWorld-release-unsigned.apk alias_name
```

Finalmente se debe ejecutar la herramienta de alineación del <zip> como un procedimiento necesario para optimizar la aplicación APK. Para realizar este procedimiento es necesario la herramienta < zipalign >.

Herramienta que se la puede encontrar en la ruta:

```
/path/to/Android/sdk/build-tools/Version/zipalign.
```

En el sistema operativo OS X con Android Studio instalado, zipalign está en la siguiente ruta:

```
~/Library/Android/sdk/build-tools/VERSION/zipalign:
```

El comando a ejecutar para realizar este procedimiento es el siguiente:

```
$ zipalign -v 4 BookDIRECTORYDEMO-release-unsigned.apk Book-
DIRECTORYDEMO.apk
```

Después de realizar el proceso de firma de la aplicación se tiene como resultado el binario de lanzamiento final llamado BookDIRECTORYDEMO, que se encuentra listo para subirlo en la tienda de Google Play. Nota: Hay otras formas de firmar una APK, consulte la documentación sobre los métodos y procedimientos de firma de aplicaciones para OS Android < firma de aplicaciones de Android >

## Google Play Store

Una vez que se tiene una versión de lanzamiento lista para ser publicada en Google Play Store, se procede a crear una lista de Play Store y se procede a cargar la aplicación móvil APK. Para empezar, se debe ingresar a la consola de desarrollador de Google Play Store y crear una cuenta de desarrollador. Tome en cuenta que se debe pagar el costo de licenciamiento de publicación que está en \$ 25,00 dólares para Android.



Fig. 53: Acceder a Google Play Developer Console

Fuente: Los Autores

Una vez que se tenga la cuenta de desarrollador, se puede continuar con la publicación en la tienda de Android en Google Play.

- Nueva aplicación Google Play.

Luego, procederá hacer clic en el botón para editar la lista de la tienda donde se subirá la APK, recuerde que se debe completar la información requerida sobre la aplicación. Cuando se han terminado de completar los pasos para la versión de lanzamiento hay que esperar a que el equipo técnico de paso al lanzamiento, tarea que puede tardar 24 horas.

## Actualización de la aplicación

Como se explicó en los apartados anteriores correspondientes a publicación de la aplicación se lanza una versión de APK; sin embargo, es muy común que se continúe desarrollando dicha aplicación con nuevas características, funcionalidades u optimización de la misma, por lo que se hace necesario actualizar la versión lanzada de forma periódica.

Para que Google Play Store, acepte la actualización de la versión es necesario que se ingrese al archivo `< config.xml>`, donde se deberá incrementar el valor de la versión y proceder a reconstruir la aplicación para su lanzamiento.

## Publicación de iOS

Este apartado proporciona el proceso y procedimiento para la publicación de la aplicación en la tienda de Apple Store App Store. Si bien la tarea de lanzar una aplicación en Google Play Store no es compleja, el proceso en App Store, requiere de una serie de procesos que deben cumplirse.

### Requerimientos

- En primer lugar, se debe inscribir en el <Programa de desarrolladores de Apple>
- Tener la cuenta de desarrollador con Apple.
- Tener instalado la plataforma Xcode, con las herramientas de desarrollo de aplicaciones de Apple

### Preparación

- Tener los íconos y Splash screens necesarios para la aplicación.
- Se debe crear un build para la producción de la aplicación.

Una vez cumplidos los requisitos descritos en el apartado anterior se debe proceder a crear el build del proyecto.



```
ionic cordova build ios --prod --release
```

Ahora se debe tener un proyecto Xcode, en una carpeta **platforms/ios/**

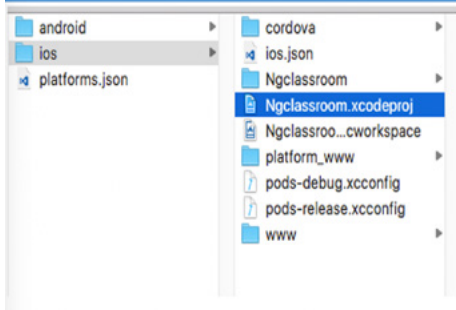


Fig. 54: Creado proyecto en Xcode  
Fuente: Los Autores

A continuación, se procede a abrir el proyecto con el nombre que le hayas asignado en el paso anterior. **Nombre\_del\_proyecto.xcodeproject** que lo debes ejecutar en la plataforma **Xcode**

Asegúrate de que las credenciales y certificados estén ejecutados de forma correcta, es un proceso clave a la hora de subir la aplicación a la tienda de Apple.

Para verificar que todo esté funcionando de forma adecuada se debe ingresar desde el panel de menú de **Xcode->Preferences.**

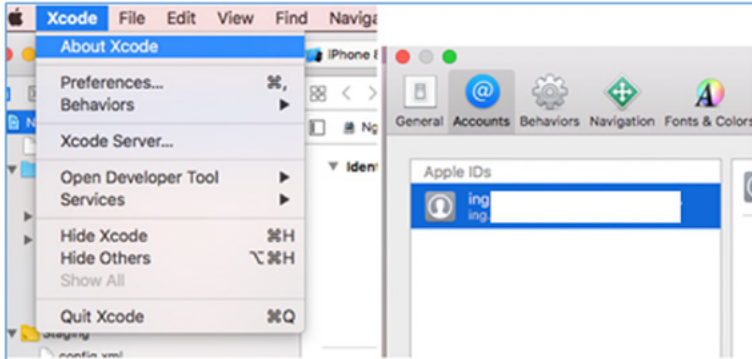


Fig. 55: Comprobar credenciales y certificados de la cuenta Apple  
Fuente: Los Autores

Y se debe proceder a configurar las credenciales a través de Xcode, verificando que el certificado se encuentre asociado a la cuenta de Apple

En este punto es necesario asociar tres aspectos a la cuenta Apple, se requiere una **App-ID** que se asocia a la aplicación móvil y que debe ser la misma que la que se configura en el archivo **config.xml** de la Aplicación en Ionic

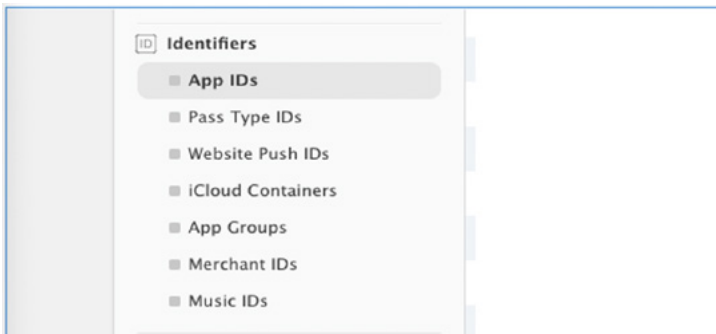


Fig. 56: Asociar App-ID  
Fuente: Los Autores

Se requiere el certificado de distribución de la aplicación móvil que se debe seleccionar de la App Store.

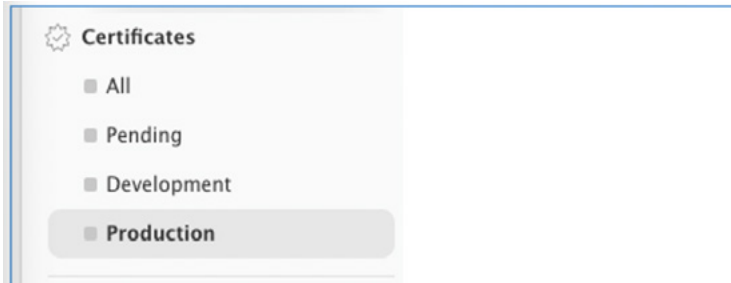


Fig. 57: Certificado de distribución  
Fuente: Los Autores

También se necesita una provisión profile asociada a las dos anteriores y a la App en **Xcode**.

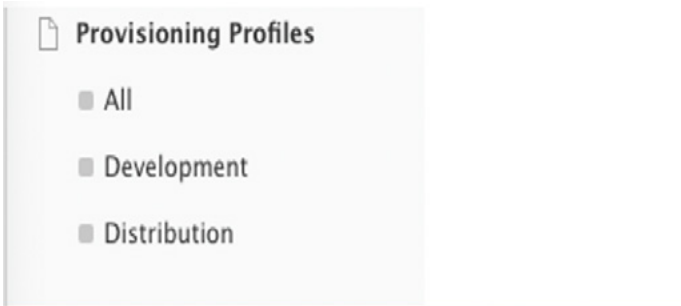


Fig. 58: Certificado de distribución  
Fuente: Los Autores

### Construcción del build desde Xcode.

Una vez que todo se encuentre asociado se debe ingresar a la plataforma desde Xcode a **Product->Archie**

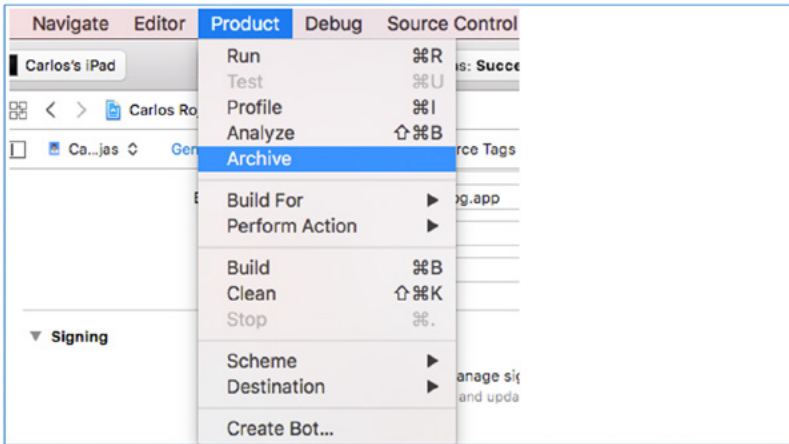


Fig. 59: Construcción del build  
Fuente: Los Autores

Se genera el **build** de la aplicación y debe aparecer una ventana con la lista de distribución.

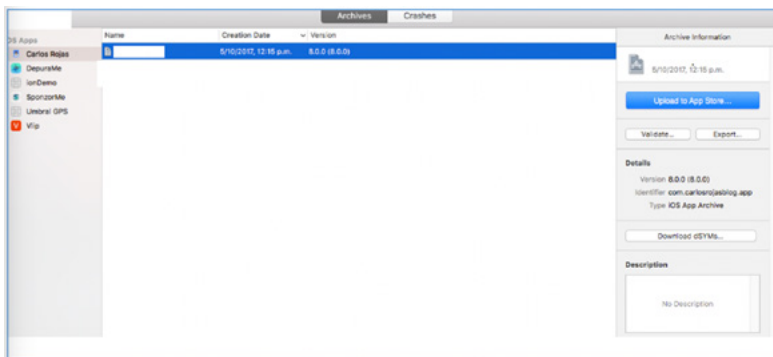


Fig. 60: Generando build para lista de distribución  
Fuente: Los Autores

Para terminar, se debe crear una ficha de la App Móvil, esto se lo puede realizar desde **itunes connet**

## Crear App desde Itunes connect

Para realizar este proceso se debe ingresar a la tienda de aplicaciones de Apple, e ingresar a **itunes connect** e ingresar a **Mis Apps**.

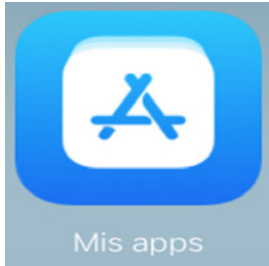


Fig. 61: Crear App desde Itunes connect  
Fuente: Los Autores

En esta sección se debe colocar toda la información de la aplicación móvil, es decir la descripción relevante de la misma.

Una vez realizado todo el procedimiento descrito en los párrafos anteriores se debe regresar a la plataforma de **Xcode**, y enviar la aplicación a **Itunes Connect**.

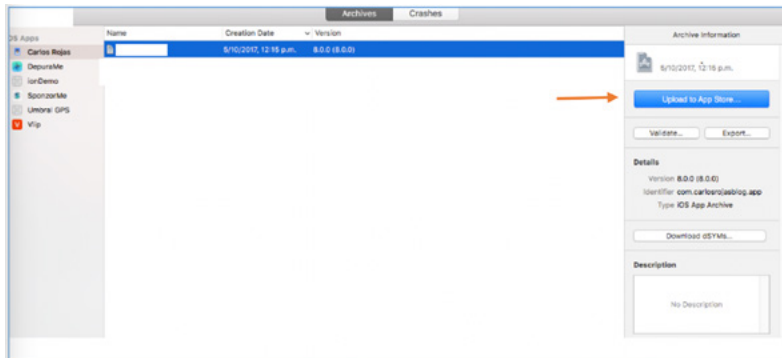


Fig. 62: Cargado aplicación de Xcode a Itunes connect  
Fuente: Los Autores

Ahora se debe enviar la aplicación generada en **Xcode**, y enviar el archivo a **Upload to App Store**.

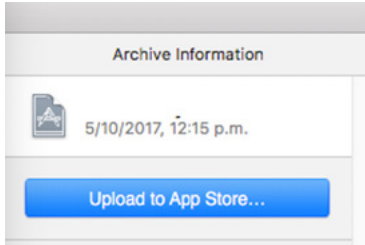


Fig. 63: Enviando aplicación a Itunes connect  
Fuente: Los Autores

Finalmente, se verifica que la aplicación móvil se cargue a la tienda de App Store de forma adecuada.

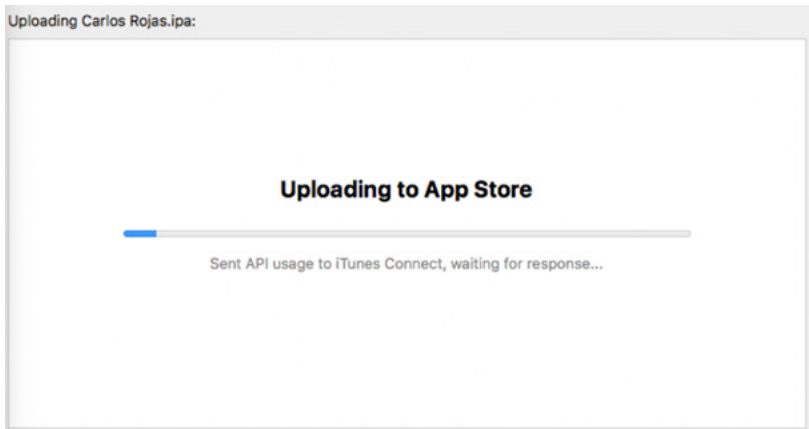


Fig. 64: Enviando aplicación a la tienda Apple Store  
Fuente: Los Autores

Ahora solo resta esperar a que la aplicación móvil se termine de subir a la tienda de Apple Store, y esperar entre 2 y 4 semanas para empezar el proceso de revisión de las políticas y requerimientos propios de la tienda.

## EPÍLOGO

El presente libro presenta ejercicios de manera secuencial que van desde nivel básico a intermedio, los cuales permitirán al usuario ejercitar, validar o aumentar sus conocimientos referentes programación, los cuales serán usados para el desarrollo de aplicaciones móviles.

La optimización de los conocimientos referentes a programación web, se ven aplicados en el desarrollo móvil, evidenciándose a través de los ejercicios planteados como ejemplo y guía, así como también al permitir que la aplicación desarrollada con estos lenguajes sea compilada para los diferentes sistemas operativos móviles.

El desarrollo de aplicaciones móviles híbridas presenta las mismas características y potencialidades que una aplicación móvil que se ha desarrollado con un lenguaje de programación nativo al sistema operativo a utilizar, siendo esta una gran ventaja para la persona que desea incursionar en el desarrollo de aplicaciones móviles y ya tiene una experiencia en el desarrollo de software web.

## BIBLIOGRAFÍA

Android, «Developers Android,» [En línea] <http://developer.android.com/intl/es/sdk/index.html>. [Último acceso: 2014]

Blanco, P. (2012) «Metodología de desarrollo ágil para sistemas móviles, Introducción al desarrollo con Android y el iPhone,» Recuperado de: Available: [http://www.adamwesterski.com/wp-content/files/docsCursos/Agile\\_doc\\_TemasAnv.pdf](http://www.adamwesterski.com/wp-content/files/docsCursos/Agile_doc_TemasAnv.pdf).

Goodman, D.; Morrison, M.; Novitski, P.; Rayl, T. G.: “JavaScript Bible” John Wiley & Sons, 2010.

Hennig, N. (2014). Selecting and evaluating the Best Mobile Apps for Library Services. Library Technology Reports.

MH Riley Ltd. (2020). Spending Tracker (2.3.1) [Aplicación móvil]. Google Play. [https://play.google.com/store/apps/details?id=com.mhriley.spendingtracker&hl=en\\_US](https://play.google.com/store/apps/details?id=com.mhriley.spendingtracker&hl=en_US)

Martínez, L. (2010). “Aplicaciones para dispositivos móviles”. 2010. [Online]. Disponible en: <https://riunet.upv.es/bitstream/handle/10251/11538/Memoria.pdf?sequence=1>. [Último acceso: 28 enero 2018]

Mejía Armijo & Escares Venegas, D. Desarrollo HTML5 para iPad y tabletas, fecha de recuperación 07 de noviembre del 2017, <http://habitatweb.mx/desarrollo-html5-para-ipad-y-tabletas>.

Morillo, J. (2012). “Introducción a los dispositivos móviles”, 2012. [Online]. Disponible en: [http://www.exabyteinformatica.com/uoc/Informatica/Tecnologia\\_y\\_desarrollo\\_en\\_disp](http://www.exabyteinformatica.com/uoc/Informatica/Tecnologia_y_desarrollo_en_disp).

MDN web docs. (20 de mayo de 2018). MDN web docs. Obtenido de <https://developer.mozilla.org/es/docs/IndexedDB-840092-dup>  
NetConsulting. (2015). Node.js: ¿Qué es y para que sirve NodeJS? Obtenido de <https://www.netconsulting.es/blog/nodejs/>

Ramírez, R. (2008). El teléfono móvil y la vida cotidiana [versión



electrónica]. Universidad Autónoma de Barcelona.

SoftCorp. (2018). Definición y cómo funcionan las aplicaciones móviles. Obtenido de <https://servisoftcorp.com/nosotros>

Vázquez, E., & Sevillano, G. M. L. (Eds.). (2015). Dispositivos digitales móviles en educación: el aprendizaje ubicuo. Retrieved from <https://ebookcentral.proquest.com>

Vique, R. (2012). Métodos para el desarrollo de aplicaciones móviles, Cataluña,

World, P. (2014). Sistemas operativos móviles: Comunicación en tiempo real,» [En línea].

ISBN: 978-9978-375-54-9



Pontificia Universidad  
Católica del Ecuador

| Sede  
Ibarra

 **Publicaciones** Centro de  
PONTIFICIA UNIVERSIDAD CATÓLICA DEL ECUADOR