USERS

INCLUYE VERSION DIGITAL GRATIS

jQuery Mobile

DISEÑO Y DESARROLLO DE APLICACIONES PARA SMARTPHONES Y TABLETS



Aplicaciones para iOS, Android, BlackBerry y WindowsPhone

HTML5, CSS3 y JavaScript

Compilación con Adobe PhoneGap Build

Instalación de librerías

Creación de temas personalizados con ThemeRoller

Definición de controles para formularios móviles de entrada de datos

por LUIS HERETER / VIVIANA ZANINI

CONOZCA TODAS LAS VENTAJAS DEL USO DE ESTE FRAMEWORK

RU

JQUERY MOBILE

DISEÑO Y DESARROLLO DE APLICACIONES
PARA SMARTPHONES Y TABLETS

por Viviana Zanini y Luis Hereter

RedUSERS



TÍTULO: jQuery Mobile
AUTORES: Viviana Zanini y Luis Hereter
COLECCIÓN: Seriado
FORMATO: 24 x 17 cm
PÁGINAS: 192

Copyright © MMXIV. Es una publicación de Fox Andina en coedición con DÁLAGA S.A. Hecho el depósito que marca la ley 11723. Todos los derechos reservados. Esta publicación no puede ser reproducida ni en todo ni en parte, por ningún medio actual o futuro sin el permiso previo y por escrito de Fox Andina S.A. Su infracción está penada por las leyes 11723 y 25446. La editorial no asume responsabilidad alguna por cualquier consecuencia derivada de la fabricación, funcionamiento y/o utilización de los servicios y productos que se describen y/o analizan. Todas las marcas mencionadas en este libro son propiedad exclusiva de sus respectivos dueños. Impreso en Argentina. Libro de edición argentina. Primera impresión realizada en Sevagraf, Costa Rica 5226, Grand Bourg, Malvinas Argentinas, Pcia. de Buenos Aires en XII, MMXIV.

ISBN 978-987-734-004-4

Hereter, Luis
jQuery Mobile / Luis Hereter y Viviana Zanini - 1a ed. - Ciudad Autónoma de Buenos Aires : Fox Andina; Buenos Aires: Dalaga, 2014.
192 p. ; 24x17 cm. - (Seriado; 14)

ISBN 978-987-734-004-4

1. Informática. I. Zanini, Viviana II. Título

CDD 005.3

Viviana Zanini



Viviana Zanini es analista de Sistemas de Computación y profesora de Informática. Ha realizado diferentes cursos de perfeccionamiento en el área de Programación. Se desempeña como profesora en los niveles primario, secundario y terciario. Tras haber desarrollado su experiencia laboral en empresas, actualmente realiza distintos proyectos de forma independiente.

Ha confeccionado guías de estudio para las diferentes materias que imparte y ha colaborado como autora en los fascículos de la colección *Excel Curso Visual y Práctico*, de esta misma editorial. Además, cuenta con otras dos publicaciones en esta editorial: *Macros en Excel 2013* y *Tablas y Gráficos dinámicos en Excel 2013*.

Agradecimientos

A mi familia, por su confianza y su apoyo incondicional. A Luis, por darme la oportunidad de escribir este libro juntos. Y a todos mis amigos.

Luis Hereter



Luis Hereter es profesor, analista de Sistemas de Computación y desarrollador web. Se desempeña como webmaster desde hace más de 7 años, y ha desarrollado aplicaciones móviles y de escritorio.

Agradecimientos

A mi mujer, Lorena, por estos casi 20 años juntos.
A mi luz absoluta, mi hijo Leandro. A mi sobrino Jerónimo y a mi hermana Sonia. A mi suegra Nanda, por todo su apoyo, y a mis tíos Jorge y Sabina, por estar siempre. A Luis y Claudia. A mis hermanos de la vida: Charli, Maxi y Mingo, y a mis amigos de la Poker League: Ariel, Carina, Alejandro, Santiago, Jesica y Silvana. A Sergio, por los conocimientos compartidos durante tantos años. A todo el equipo de la editorial, por apoyar y creer en este proyecto. Y, por último, un agradecimiento muy especial a mi amiga y coautora de este libro, Viviana, porque sin ella no hubiera sido posible.

Prólogo

Con el correr de los años, la tecnología móvil avanzó de manera muy presurosa. Todos los diseñadores y desarrolladores web nos tuvimos que orientar, tarde o temprano, hacia ese lado. Recuerdo cuando hace muchos años llegó a mí la primera agenda electrónica de bolsillo o PDA (asistente digital personal); primero fue la Palm, hacia fines de los noventa, la que me llamó la atención, y luego, cerca del año 2000, las HPPocket PC, que tenían cargado Windows CE.

Un tiempo después, desarrollar en Java para teléfonos celulares era una moda, hasta que en 2007 Steve Jobs anunció el primer iPhone. Eso marcó un antes y un después en materia de tecnología móvil. Recuerdo sus palabras: "Un iPod, un teléfono móvil y un dispositivo de comunicación... todo en un solo dispositivo". Desde ese momento, todo cambiaría en materia de desarrollo: los que programábamos en Adobe Flash o Adobe Flex tuvimos que volcarnos, con el correr de los años, a HTML5, CSS3 y JavaScript, y ya no solo programábamos para sitios web diseñados para computadoras de escritorio y laptops, sino que también debimos tener presentes los dispositivos móviles.

En ese momento, nació una pregunta que todos nos hacíamos: ¿podemos crear aplicaciones o sitios web para móviles, sin la necesidad de programarlos en el lenguaje nativo de cada sistema operativo? Por suerte, muchos desarrolladores –quizás por necesidad– ya estaban trabajando en ese tema. Desde entonces, comencé a desarrollar en jQuery Mobile y conocí PhoneGap y, más adelante, el servicio de Adobe PhoneGap Build. Todo lo que venía haciendo hasta ese momento ahora estaba en un solo lugar, de una manera simple y sorprendente.

Esperamos que este libro sea un comienzo para todos los estudiantes y desarrolladores que quieran aprender a programar y diseñar aplicaciones, para que no tengan la necesidad de estudiar cada lenguaje en particular. Esperamos, también, que disfruten de este libro como nosotros disfrutamos de hacerlo. Y, como diría el genio de Steve Jobs: "¡Sigamos hambrientos!".

Luis Hereter

El libro de un vistazo

Este libro está orientado a todas aquellas personas que posean conocimientos básicos sobre programación web y quieran aprender a utilizar las herramientas que nos proporciona jQuery Mobile para crear aplicaciones web para dispositivos móviles, y luego distribuirlas en sus respectivas tiendas utilizando Adobe PhoneGap Build.

*01 MOBILE WEB

En este capítulo nos introduciremos en los conceptos básicos sobre las aplicaciones web. Conoceremos los diferentes tipos de aplicaciones web que existen y haremos una concisa introducción sobre los principales sistemas operativos para dispositivos móviles que hay en el mercado. Finalmente, estaremos preparados para empezar a conocer los distintos entornos de desarrollo de jQuery Mobile.

*02 CREAR UNA PÁGINA EN JQUERY MOBILE

En este capítulo aprenderemos a instalar y utilizar el framework de jQuery Mobile. Explicaremos cómo podemos organizar nuestros proyectos empleando Dreamweaver CC. Posteriormente, conoceremos cuál es la estructura de una página básica y sus principales componentes.

*03 BOTONES Y LISTAS

Una vez creada la página básica, comenzaremos a ver otros componentes del framework que podemos agregar para crear la interfaz de la aplicación. En este capítulo aprenderemos a crear y configurar botones. Explicaremos también cómo crear diferentes tipos de listas, de acuerdo a lo que necesitemos en cada uno de nuestros proyectos.

*04 FORMULARIOS

En este capítulo nos dedicaremos a conocer el funcionamiento y la utilidad de los formularios que nos permitirán introducir datos para luego ser procesados en el servidor. Describiremos sus principales atributos y funciones y luego explicaremos cómo utilizarlos e incluirlos dentro de un formulario para el envío de datos.

*05 OTROS COMPONENTES DE LA INTERFAZ DE USUARIO

En este capítulo, a través de diferentes ejemplos, conoceremos y aprenderemos a utilizar componentes que nos van a permitir crear una interfaz que pueda mostrar la información en una cantidad limitada de espacio, como los dispositivos móviles.

*06 INTERFAZ DE USUARIO Y TEMAS

En los capítulos anteriores aprendimos a crear páginas básicas y complejas, empleando los diferentes componentes que nos ofrece el framework. En este capítulo veremos temas relacionados con la disposición del contenido y aprenderemos a aplicar muestras de color a los elementos de la página utilizando los temas de jQuery Mobile y temas personalizados.

*07 PHONEGAP

En este capítulo, vamos a realizar una introducción a PhoneGap y Adobe PhoneGap Build, y veremos cuáles son sus diferencias. Una vez terminada la aplicación web, aprenderemos a compilarla utilizando Adobe PhoneGap Build y, de esta manera, lograr una aplicación que sea compatible con todos los dispositivos sin importar el sistema operativo.

INFORMACIÓN COMPLEMENTARIA

A lo largo de este manual, podrá encontrar una serie de recuadros que le brindarán información complementaria: curiosidades, trucos, ideas y consejos sobre los temas tratados. Para que pueda distinguirlos en forma más sencilla, cada recuadro está identificado con diferentes iconos:


CURIOSIDADES E IDEAS


ATENCIÓN


DATOS ÚTILES Y NOVEDADES


SITIOS WEB

Contenido

Sobre los autores 4

Prólogo 5

El libro de un vistazo 6

Información complementaria 7

Introducción 12

*01 Mobile Web

Mobile web 14

Aplicaciones 16

Tipos de aplicaciones móviles 16

Sistemas operativos para dispositivos móviles 19

Android 19

iOS (iPhone OS) 21

Windows Phone 24

BlackBerry 10 26

Herramientas para crear aplicaciones 27

HTML5 para móviles 27

Estructura de una página HTML5 28

jQuery Mobile y PhoneGap 31

Distintos entornos de desarrollo 32

Resumen 33

Actividades 34



*02 Crear una página en jQuery Mobile

El framework 36



CDN 36

Descargar y trabajar con el archivo comprimido 37

El desarrollo de un proyecto 38

Organización del proyecto 39

Trabajar con Dreamweaver 40

Crear páginas de jQuery Mobile 47

Página básica y multipágina 49

Estructura básica 52

Meta viewport 52

Header bar 53

Footer bar 53

Content section 55

Enlaces 55

Transiciones 58

Tipos de transiciones 58

Resumen 59

Actividades 60



*03

Botones y listas

Botones **62**
Características 62
Iconos 63
Conjunto de iconos o valores para el atributo data-icon 64
Posición del icono 65
Botones con un ancho específico 66
Botones agrupados 66
El botón "volver" 68
Comandos para utilizar funciones del teléfono 68
Listas **69**
Lista interactiva 69
Listas interactivas anidadas 71
Lista numerada 72
Filtrar elementos y agregar placeholder 74
Dividir y autodividir 75
Contador burbuja 78
Agregar imágenes-iconos 79
Navbar **84**
Botón activo 85
Barra de navegación fija 86
Iconos en el navbar 88
Resumen **89**
Actividades **90**



*04

Formularios

Estructura de un formulario **92**

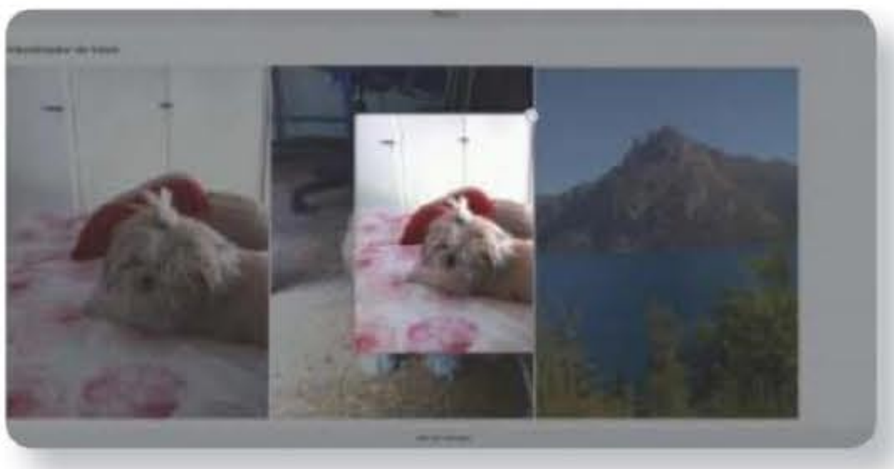


Envío del formulario 92
Deshabilitar elementos 94
Minielementos 95
Field containers 96
Controles de un formulario **96**
Etiqueta (label) 97
Campo de texto (text input) 98
Campo de búsqueda (search input) 102
Slider 103
Interruptor (flip toggle switch) 104
Botón de opción (radio button) 105
Casilla de verificación (checkbox) 107
Menú de selección (select menu) 109
Selector de fechas (datepicker) 114
Archivo (file) 115
Resumen **115**
Actividades **116**

*05

Otros componentes de la interfaz de usuario

Otros componentes UI **118**
Tablas **118**
Crear tablas 119



Componentes desplegables 126
Acordéon (collapsible-set) 127
Pestañas (tabs) 129
Ventanas emergentes (pop-up) 132
Distintos tipos 132
Cerrar un pop-up 142
Posición 143
Transiciones 144
Flechas (arrows) 145
Paneles 146
Resumen 149
Actividades 150

*06 Interfaz de usuario y temas

Grids 152
Agrupar y dividir contenido 156
Temas 158
Temas personalizados mediante ThemeRoller 160
Crear un tema 164
Asignar temas 167
Importar y compartir un tema 169



Resumen 169
Actividades 170

*07 PhoneGap

¿Qué es PhoneGap? 172
Dispositivos compatibles 173
Descargar PhoneGap 173
Adobe PhoneGap Build 174
Cómo crear una cuenta 175
Acceder al servicio de PhoneGap Build 176
Subir un proyecto 178
Descargar la aplicación 179
Trabajar con Dreamweaver CC 180
Cómo comenzar a trabajar 180
Otras opciones: Adobe Edge Code y Bracket 182
Extensiones para PhoneGap 183
Cómo instalar extensiones 183
Extensiones disponibles 185
El emulador Ripple 189
Resumen 191
Actividades 192



Introducción

En los últimos años, el desarrollo de aplicaciones web para dispositivos móviles ha crecido mucho. Por tal motivo, este libro está orientado a usuarios que posean conocimientos previos sobre programación web y deseen aprender a utilizar una herramienta que les permita crear una aplicación web para dispositivos móviles de manera rápida y fácil.

Comenzaremos por explicar los conceptos básicos sobre las aplicaciones web para dispositivos móviles y los diferentes sistemas operativos sobre los cuales se emplean. Haremos una introducción sobre lo que es el framework de jQuery Mobile para, enseguida, aprender cómo trabajar con este framework y cómo incluir en un archivo HTML las librerías de jQuery Mobile. También veremos cómo usar Dreamweaver CC para organizar y administrar los archivos que componen la aplicación.

Luego explicaremos cuál es la estructura de una página en jQuery Mobile y aprenderemos a crear una página básica y multipágina. Posteriormente, explicaremos cómo crear botones y diferentes tipos de listas y formularios para el envío de datos.

En los últimos capítulos avanzaremos un poco más: aprenderemos a crear una interfaz utilizando los componentes de jQuery Mobile, de modo tal que la aplicación web nos permita mostrar la información en una cantidad limitada de espacio, como las pantallas de los smartphone o de las tabletas.

A continuación, explicaremos cómo aplicar muestras de color a la página y a los componentes específicos dentro de la página, empleando los temas de jQuery Mobile. También aprenderemos a crear nuestros propios temas personalizados empleando una herramienta basada en la Web: ThemeRoller.

Finalmente, aprenderemos a compilar nuestra aplicación utilizando PhoneGap Build y, así, lograr una aplicación que sea compatible con todos los dispositivos sin importar el sistema operativo que utilicen.

*01

Mobile web

En este capítulo explicaremos qué son las aplicaciones web y cuáles son las clases que existen. También realizaremos un breve recorrido por la historia de los sistemas operativos para dispositivos móviles: iOS, Android, Windows Phone, BlackBerry 10. Luego conoceremos los distintos entornos de desarrollo de jQuery Mobile, que nos permitirán diseñar y desarrollar nuestras propias aplicaciones móviles.

Por último, conoceremos qué es y cómo funciona PhoneGap.

- Mobile web.....14
- Aplicaciones.....16
- Sistemas operativos para dispositivos móviles.....19
- Herramientas para crear aplicaciones.....27
- jQuery Mobile y PhoneGap.....31
- Distintos entornos de desarrollo.....32
- Resumen.....33
- Actividades.....34

Mobile web

Para comenzar, debemos entender qué es una **mobile web**. Un concepto simple y claro sería decir que una mobile web es aquella web a la cual podemos acceder a través de un dispositivo móvil, sin importar el tipo de dispositivo desde el cual lo hagamos.

Con el correr de los años, cada vez más personas se conectan a páginas web desde sus dispositivos móviles. Un ejemplo típico son las redes sociales, como **Facebook** o **Twitter**, entre otras, en las que la mayoría de los usuarios se conectan por medio de lo que se denomina **aplicaciones nativas** –que veremos más adelante en este mismo capítulo– sin entrar ni siquiera por el explorador del dispositivo.

Desde la salida del **iPhone** (teléfono celular comercializado por **Apple Inc.**) en 2007, y con la evolución en los últimos años de los smartphones (teléfonos inteligentes) y las tabletas, el mundo giró hacia el lado de la tecnología móvil. Ahora, por medio de un dispositivo móvil, podemos acceder en cualquier momento y lugar a nuestro correo, ver nuestra cuenta de Twitter, revisar nuestra cuenta bancaria, y mucho más.



Figura 1. **Twitter** ofrece aplicaciones oficiales para dispositivos con sistemas operativos **Android**, **iOS**, **Windows Phone**, entre otros.

A partir de la evolución de los smartphones, surge la necesidad de programar aplicaciones compatibles para estos dispositivos móviles. Entonces, la diferencia entre Web y mobile web es muy simple: la mobile web, como la Web, corre también en un navegador web

(como **Safari**, **Chrome** y otros), pero lo hace a través de un dispositivo móvil, a diferencia de la Web, que lo hace desde cualquier navegador en una computadora de escritorio o laptop.

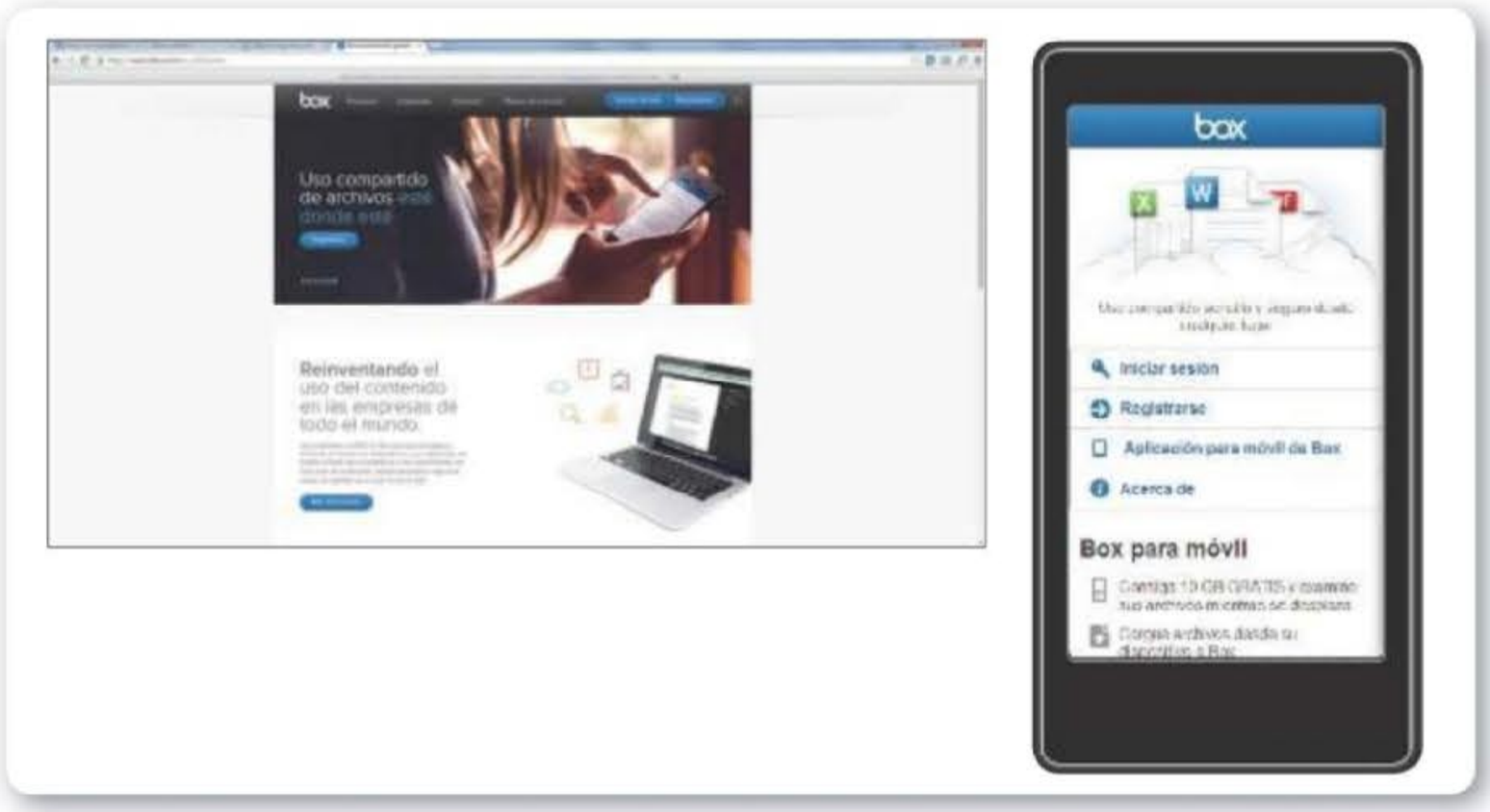


Figura 2. Cómo se ve el sitio web de la empresa **Box** en pantalla completa desde una PC de escritorio con el navegador **Chrome**, y con el mismo navegador adaptado a un dispositivo móvil.

Para los programadores o diseñadores, el desarrollo de la web mobile consiste en plantear un diseño pensado de manera exclusiva para los usuarios de dispositivos móviles, teniendo en cuenta, por ejemplo, las resoluciones de pantalla, el acelerómetro o la geolocalización. Es decir, permitir que se aprovechen al máximo las posibilidades que brindan los dispositivos móviles y así lograr una mejor experiencia de usuario, que no conseguiríamos si solo programáramos o diseñáramos para la Web.



REDUSERS PREMIUM

Para obtener material adicional gratuito, ingrese a la sección **Publicaciones/Libros** dentro de **http://premium.redusers.com**. Allí podrá ver todos nuestros títulos y acceder a contenido extra de cada uno, como los ejemplos utilizados por el autor, apéndices y archivos editables o de código fuente. Entre los complementos de este libro encontrará, además, tutoriales en video para mejorar la comprensión de los conceptos desarrollados en la obra.

Aplicaciones

Una **aplicación**, también llamada **app**, es un software escrito en un lenguaje de programación, que nos permite utilizar una computadora o un dispositivo móvil con un objetivo específico. Existe un gran número de aplicaciones, por ejemplo, reproductor de audio, navegador y aplicaciones de correo electrónico, entre otras.

Como mencionamos antes, gracias al auge de los dispositivos móviles, se han ido creando muchas aplicaciones para ser utilizadas tanto desde un smartphone como de una tableta.



Figura 3. El uso de dispositivos móviles en varios ámbitos de la vida ha promovido la creación de una gran cantidad de aplicaciones.

Tipos de aplicaciones móviles

Existen tres tipos de aplicaciones para dispositivos móviles:

- Aplicaciones nativas (*native apps*).
- Aplicaciones web (*web apps*).
- Aplicaciones híbridas (*hybrid apps*).

APPLE INC.

Apple Computer Inc. es una empresa multinacional estadounidense que fue fundada en el año 1976 por Steve Jobs y Steve Wozniak. En el año 2007, cambia de nombre por **Apple Inc.** Esta empresa se dedica a diseñar y comercializar software (Mac OS X, iOS), computadoras (Macintosh) y dispositivos electrónicos móviles, como el iPhone, el iPod y el iPad.

Aplicaciones nativas

Las **aplicaciones nativas** son aquellas que están desarrolladas para ejecutarse sobre un sistema operativo o plataforma específicos y se desarrollan en el lenguaje nativo de cada dispositivo, de ahí su nombre. Por eso, si desarrollamos una sola aplicación, debemos adaptarla para cada una de las diferentes plataformas; es decir, si desarrollamos una aplicación para **iOS**, no la podríamos instalar en **Android**.

Estas aplicaciones están escritas en un lenguaje de programación propio, lo que permite acceder a todos los recursos del hardware del dispositivo móvil, como la cámara de fotos y el GPS. Por ejemplo, para desarrollar aplicaciones para iPhone, debemos usar el lenguaje **Objective C** o **Swift**, en cambio, si queremos desarrollar una aplicación para Android, deberemos emplear el lenguaje **Java**.

Las aplicaciones nativas se descargan desde la **tienda de aplicaciones** (en inglés, *market place*) desarrollada para cada dispositivo.

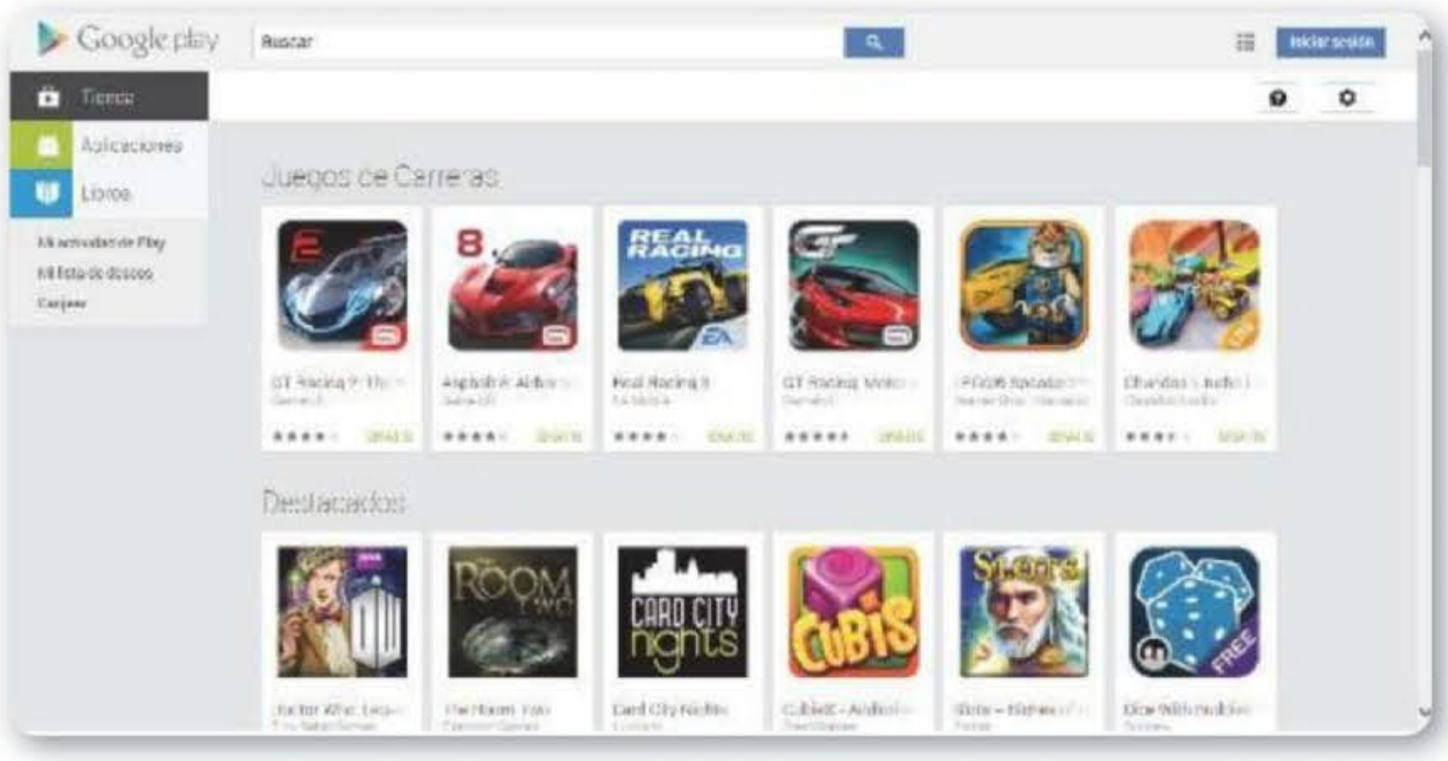


Figura 4. Google Play es la tienda de aplicaciones para usuarios de Android, a la que accedemos desde el dispositivo móvil.

SWIFT

Se trata de un nuevo lenguaje de programación propuesto por Apple en la conferencia anual de desarrolladores en junio de 2014. Mucho más veloz que Objective C, es un lenguaje nativo y sirve tanto para programar en **OSX** (sistema operativo de las computadoras Macintosh, basado en el sistema operativo Unix) como en **iOS**, en el entorno de desarrollo de Apple Xcode.

Aplicaciones web

Cuando hablamos de **aplicaciones web**, nos referimos a un sitio web optimizado para dispositivos móviles. Su principal atributo consiste en que puede visualizarse en cualquier dispositivo; por lo tanto, no tendremos que desarrollar una versión de la aplicación para BlackBerry, Android, iOS o Windows Phone, por ejemplo.

Para desarrollar este tipo de aplicación, lo único que necesitaremos es contar con una conexión a internet y con un dispositivo que posea un navegador que cumpla con los estándares de **HTML5**, ya que este, a diferencia de las versiones anteriores de HTML, permite combinar perfectamente sus nuevas funciones con **JavaScript** y **CSS3**.

Otro punto a favor de las aplicaciones web es que no siempre deben descargarse desde la tienda de aplicaciones. Esto nos evita tener que pasar por los procesos de validación y sus costos; solo basta con tener la URL de la aplicación y crear un acceso directo a ella en el dispositivo.

Otra ventaja para los desarrolladores es que tampoco es necesario que el usuario actualice el producto cada vez que se lanza una revisión o actualización; al tratarse de una aplicación web, basta con actualizarla en nuestro servidor, y esos cambios se van a ver reflejados en el instante en que el usuario la utilice.

Aplicaciones híbridas

Las **aplicaciones híbridas** se escriben en código HTML5, JavaScript y CSS3 para ser ejecutadas en un navegador. Este código se compila en un archivo ejecutable, y se crea una aplicación **multiplataforma** que puede ser distribuida a través de las tiendas de aplicaciones.

Las aplicaciones híbridas, al igual que las nativas, permiten acceder a los recursos del hardware de un dispositivo móvil.

MARKET PLACES

Llamados también **stores** o **tiendas**, son aquellos sitios web donde podemos encontrar distintos tipos de apps discriminadas por categorías (gratuitas o pagas, más descargadas, etcétera). Estos sitios nos permiten personalizar nuestros dispositivos para lograr una mayor experiencia de usuario. En ellos podemos, además, comentar o calificar las aplicaciones y compartir nuestras opiniones con otros usuarios.

La conveniencia de desarrollar una aplicación nativa o una aplicación web dependerá del producto y de nuestra necesidad de venderlo. Una aplicación web, comparada con una nativa, puede hacer absolutamente todo, y, si a eso le agregamos la capacidad de actualizarla y no tener que distribuirla, más los costos y el tiempo de desarrollo, sería una buena elección para tener en cuenta a la hora de programar para móviles.



Figura 5. Apple Store es la tienda virtual de **Apple**, desde la cual se pueden realizar compras online de sus productos y aplicaciones.

Sistemas operativos para dispositivos móviles

Al igual que en las computadoras personales, los sistemas operativos para dispositivos móviles nos permiten administrar los recursos del dispositivo. Además, nos proporcionan una plataforma de software sobre la que se pueden ejecutar las aplicaciones.

Existen varios sistemas operativos para dispositivos móviles. A continuación, describiremos algunos de ellos.

Android

Android es el sistema operativo más usado de los dispositivos móviles. Está basado en **Linux** y fue desarrollado en el año 2004 como plataforma

para cámara de fotos, según reveló Andy Rubin, uno de los creadores del proyecto. En julio de 2005, fue comprado por Google a sus desarrolladores, **Android Inc.**, para lanzar la versión **1.0** en el año 2008.

En abril de 2009 se lanzó la versión **1.5** (Cupcake), mucho más estable que la versión 1.0. A partir de este momento, la mayoría de los fabricantes optaron por equipar sus dispositivos con este sistema operativo.

Las versiones **2.0**, **2.2** o **2.3** trajeron muchas mejoras, como por ejemplo, la incorporación de **Google Maps** y la integración con varias cuentas de correo a la vez desde un mismo teléfono.



Figura 6. Podemos descargar la aplicación **Google Maps** para dispositivos con sistema operativo Android e iOS desde sus respectivas tiendas.

La versión **3.0** (**Honeycomb**) cambió la interfaz, pensada para adaptarse preferentemente a grandes resoluciones de pantalla, ya que está destinada a tabletas de 1280 x 800 pixeles. Así, es más parecido al sistema operativo de una computadora que al de un teléfono. El sistema multitarea fue mejorado y soporta videochat mediante **Gtalk**.

LOLLIPOP

Lollipop (chupetín) es el nombre de la versión 5 de Android, continuando la costumbre de nombrar las versiones como golosinas o postres. Ha salido a la luz en octubre de 2014 junto con el nuevo teléfono inteligente **Nexus 6,** en asociación con Motorola. Este equipo posee una pantalla de 6 pulgadas, superior al iPhone 6 plus y al Galaxy Note 4. Los otros dos dispositivos que tendrán de fábrica el sistema operativo serán la tableta **Nexus 9** y el dispositivo para televisores **Nexus Player**.

La versión **4.0** (**Ice Cream Sandwich**) fue lanzada en octubre de 2011, con una interfaz completamente mejorada y rediseñada. Las versiones **4.1**, **4.2** y **4.3** (**Jellybean**) incluyeron el servicio de asistencia **Google Now**.

En la versión 4.4 (**KitKat**), una de las novedades fue la tecnología **immersive mode**, que oculta automáticamente la interfaz de usuario al leer un libro, ver una película o jugar. También agregó un administrador de impresión compatible con las aplicaciones de Google e integró la herramienta **Quickoffice**.

La última versión al escribir este libro es la **5.0** (**Lollipop**). El principal cambio está en la interfaz de usuario, con el diseño **Material Design**, que unifica la experiencia de usuario sea cual sea el dispositivo y el tamaño de la pantalla. También se pueden observar grandes mejoras en el panel de notificaciones, sonidos, fondos de pantalla y menús.

Ventajas

Al ser un software de código abierto, Android puede ser adaptado a todos los dispositivos y permite también que cualquier desarrollador pueda modificarlo e, incluso, mejorarlo.

Es un sistema operativo completamente personalizable tanto para usuarios como para fabricantes.



Figura 7. Podemos conocer más sobre el sistema operativo Android ingresando al sitio **www.android.com**.

iOS (iPhone OS)

Desarrollado por Apple Inc., este sistema operativo fue presentado en enero de 2007 por **Steve Jobs** y, en junio de ese mismo año, salió al mercado la versión **1.0** junto con el primer iPhone. Incorporaba

IPHONE OS ES EL SISTEMA OPERATIVO DESARROLLADO POR APPLE INC. QUE FUE PRESENTADO EN 2007

funciones revolucionarias para un teléfono en ese tiempo, como: e-mail, calculadora, fotos, música, notas, etcétera. Si bien todo comenzó con el iPhone, el sistema operativo está orientado a otros dispositivos de Apple: iPad, iPod Touch y Apple TV. Está basado en el sistema operativo **Unix** al igual que OSX (sistema operativo de Macintosh).

Con la versión **2.0** del sistema operativo, apareció el **iPhone 3G** que, a diferencia del primer iPhone, se comercializó en todo el mundo y permitía instalar aplicaciones de terceros.

La versión **3.0**, lanzada en junio de 2009 con el **iPhone 3GS**, trajo funciones que habían sido pedidas por los usuarios, como las de "copiar" y "pegar", así como el envío y recepción de mensajes multimedia.

La versión **4.0** por primera vez deja de ofrecer soporte a algunos dispositivos, como el iPhone 2G y el 3G, y les limita el número de funciones. Esta versión sale a la luz en conjunto con el **iPhone 4** y soporta por completo los dispositivos iPhone 3GS y 4 e iPod Touch 3 y 4.

La versión **iOS 5** es lanzada en junio de 2011 junto al **iPhone 4S**. Además de este dispositivo, soporta iPhone 3GS y 4, iPod Touch 3 y 4, iPad 1 y 2, y Apple TV.

El **iPhone 5** y la versión del **iOS 6** llegaron en agosto de 2012.

En junio de 2013 se lanza la versión beta de **iOS7** y, más adelante, en septiembre de ese mismo año, la versión estable junto con sus nuevos teléfonos, el **iPhone 5s** y el **5c**. En esta versión hay un cambio en la interfaz, que no ocurría desde la primera versión. Incluye iconos planos, mayor integración con **iCloud** (sistema de almacenamiento en la nube desarrollado por Apple) y un original diseño que estuvo a cargo de Jonathan Ive. Los dispositivos compatibles son: iPhone 4, 4s y 5,

SMARTPHONE

Los **smartphones**, también llamados *teléfonos inteligentes*, son aquellos dispositivos que incorporan las funciones de un teléfono móvil y además nos permiten comunicarnos a través de Wi-Fi y bluetooth, conectarnos a internet, enviar y recibir mensajes de correo electrónico e instalar aplicaciones (ya sean desarrolladas por el fabricante del smartphone o por un tercero).

iPod Touch de quinta generación, iPad 2 y todos los iPad con pantalla de retina y, por último, el iPad Mini 1 y 2.

La última versión hasta la fecha es la **iOS 8**, que nos trae algunas novedades como, por ejemplo, la posibilidad de organizar y editar las fotos de todos los dispositivos de un mismo usuario, o la posibilidad de agregar teclados desarrollados por terceros. Otra novedad es **iCloud Drive**, que nos permite trabajar directamente desde la nube con presentaciones, imágenes, archivos PDF, entre otros.

Este sistema, además, está orientado a la salud, ya que nos permite tener monitoreado en una sola aplicación todo lo referente al estado cardíaco, las actividades físicas, etcétera.



Figura 8. iCloud, el sistema de almacenamiento en la nube de Apple, permite una fácil sincronización entre los dispositivos **iPhone**, **iPad** y **iPod**.

Ventajas

El usuario siempre está actualizado a la última versión de iOS, ya que Apple envía actualizaciones para casi todos sus dispositivos y no depende del fabricante del equipo como pasa en Android. Además, la tienda de aplicaciones de Apple es una de las más grandes del mundo.

STEVE JOBS

Steve Paul Jobs (1995-2011) fue un famoso empresario estadounidense que revolucionó la industria informática y de telefonía móvil. Fundador y CEO de Apple Inc., entre sus productos más destacados se encuentran las computadoras personales Apple y Macintosh, los dispositivos iPod, iPhone, iPad y los sistemas operativos para computadoras (Mac OS) y dispositivos móviles (iOS).



Figura 9. Podemos conocer las características de los smartphones de Apple ingresando al sitio **www.apple.com/es/iphone**.

Windows Phone

Desarrollado por **Microsoft**, sucesor de la plataforma **Windows Mobile**, este sistema operativo salió al mercado en octubre de 2010 con la versión Windows Phone 7.

En el año 2011, Microsoft firma un acuerdo con **Nokia** para sus teléfonos Nokia Lumia 800, Nokia Lumia 710 y Nokia Lumia 900 y, de esta forma, deja de lado **Symbian**, sistema operativo que venía en los teléfonos Nokia, para adoptar **Windows Phone** como su principal sistema operativo para teléfonos celulares.

En abril de ese año, Microsoft compra la división móvil de Nokia, por lo que pasa a llamarse **Microsoft Mobile** y, así, unifica los teléfonos celulares con su sistema operativo.

EL PRIMER TELÉFONO INTELIGENTE

Simon, el primer smartphone de la historia, fue creado por **IBM** en 1992 y comercializado por **BellSouth**. Su tamaño era de 4,5 pulgadas y tenía una pantalla táctil que funcionaba a través de la presión de un puntero. Además de poder realizar llamadas telefónicas, permitía enviar y recibir correo, enviar fax, entre otras funciones. La desventaja fue que este teléfono podía funcionar únicamente en 190 ciudades de Estados Unidos.

La versión Windows Phone 8 fue lanzada a fines de 2012 y estuvo disponible solo para nuevos dispositivos, ya que actualiza el kernel (núcleo del sistema operativo) con nuevos requerimientos de hardware, lo que hace que los dispositivos anteriores no puedan actualizar de la versión 7.8 a la 8, optando de esta forma por estandarizar todos los sistemas en uno (tabletas, teléfonos, PCs).

Esta versión utiliza una nueva interfaz llamada **Metro** compuesta por mosaicos (*tiles*) totalmente personalizables, que funcionan como enlaces a aplicaciones.

La última versión de Windows Phone es la 8.1. Esta versión incorpora un asistente de voz llamado **Cortana**, que se puede utilizar para acceder a aplicaciones y para realizar búsquedas por internet. Asimismo, trae mejoras en la personalización del sistema y, entre otras novedades, permite que el usuario comparta archivos a través de la nube o de redes sociales.



Figura 10. La interfaz **Metro** de **Windows** apunta a la integración entre los sistemas operativos de escritorio y los sistemas operativos móviles.

Ventajas

Además de ser un sistema operativo simple e intuitivo, Windows Phone es el principal sistema operativo para todos los smartphones de Nokia y cuenta con el respaldo de Microsoft.

Como su tienda de aplicaciones recién comienza, significa una gran ventaja para los desarrolladores a la hora de mostrar sus productos, ya que el mercado todavía se encuentra con muy pocas aplicaciones, y esto permite que sean visibles y se encuentren con mayor facilidad.

BlackBerry 10

Se podría decir que es el sistema operativo para dispositivos móviles más nuevo, comparado con los presentados antes. Fue desarrollado por **RIM** en Canadá y lanzado en mayo de 2012. Está basado en el sistema operativo de tiempo real **QNX**, adquirido por la empresa RIM en el año 2010. Este sistema, como la mayoría, está basado, a su vez, en Unix.

En su lanzamiento también se presentaron dos nuevos teléfonos: el **Z10** y el **Q10**. La principal novedad del Z10 (y la más visible) es que deja de lado el teclado **QWERTY**, característica principal de todos los teléfonos BlackBerry hasta esa fecha. Este fue reemplazado por un teclado táctil que se adapta a la manera de escribir del usuario, haciendo sugerencias personalizadas. Cuenta con una interfaz muy fluida, el navegador permite una mejor experiencia de usuario con HTML5 y soporta completamente **Flash**. Entre otras cosas, también permite al usuario cambiar de aplicación sin la necesidad de pasar primero por la página de inicio. Además cuenta con la tecnología **BlackBerry Balance**, que nos permite separar la parte de uso personal de la parte empresarial, no puede ser copiada, y restringe su acceso para no poner en riesgo la seguridad de la empresa. Por ejemplo, si el usuario cambia de trabajo, la empresa puede optar por borrar la información corporativa, sin afectar los datos personales del usuario y sus aplicaciones.

Ventajas

Es un sistema operativo que no copia a ningún otro, muy intuitivo y seguro. Su centro de distribución de aplicaciones, llamado **BlackBerry World**, cuenta con cientos de aplicaciones para todo tipo de usuario.



Figura 11. Podemos visitar la tienda de **BlackBerry** ingresando al sitio **http://appworld.blackberry.com/webstore**.

Herramientas para crear aplicaciones

HTML5 junto con **CSS3** y **JavaScript** desde hace un tiempo se han convertido en las herramientas de desarrollo más populares. Desde la salida del iPhone y con la prohibición de parte de Apple de permitir correr Adobe Flash en sus dispositivos, son las herramientas por excelencia más usadas por la comunidad de desarrolladores.

Si bien todavía nada puede imitar por completo a Adobe Flash, con HTML5 y CSS3 podemos crear juegos, animaciones, transiciones, y además son compatibles con las últimas versiones de nuestros navegadores, ya sea Mozilla Firefox, Internet Explorer, Safari o Google Chrome.

HTML5 para móviles

HTML es el lenguaje de etiquetas diseñado para estructurar texto con el formato de los documentos de la **World Wide Web**, cuyo estándar es desarrollado por el consorcio **W3C** (World Wide Web Consortium) para normalizar la Web. Este consorcio es, a su vez, el que se encarga de publicar las especificaciones relativas al lenguaje HTML.

Si hablamos de HTML5, este lenguaje es soportado por la mayoría de los dispositivos móviles, por no decir todos, y sus navegadores móviles como, por ejemplo, los de iPhone: Safari, Chrome u Opera.



Figura 12. Portal del consorcio **W3C** (**www.w3.org/TR/html5**), que regula las especificaciones de HTML5.

Estructura de una página HTML5

Para conocer y entender un poco más de la estructura usada por **jQuery Mobile** –entorno de desarrollo del que enseguida comenzaremos a hablar– en sus páginas o documentos, primero necesitamos saber cómo está compuesta la estructura de una página HTML5.

Una página HTML5, a diferencia de sus versiones anteriores, tiene la siguiente estructura:

```
<!DOCTYPE html>
<html lang="es">
<head>
<title>Mi sitio</title>
<meta charset="utf-8" />
</head>
<body>
<header>
<h1>Mi sitio web</h1>
<p>Mi sitio web creado en HTML5</p>
</header>
<nav>
<ul>
<li><a href="#">Menú A</a></li>
<li><a href="#">Menú B</a></li>
<li><a href="#">Menú C</a></li>
</ul>
</nav>
<section>
<article>
<h2>Título del contenido<h2>
<p> Contenido </p>
</article>
</section>
<aside>
<h3>Título de contenido</h3>
<p>contenido</p>
</aside>
<footer >
```

```
<h2>Esto es un pie de página</h2>
</footer>
</body>
</html>
```

El primer elemento que visualizamos en el documento es **<!DOCTYPE html>**, que siempre tiene que ir al comienzo de la página, antes de la etiqueta **<head>**, y es el encargado de decirle al navegador que lo que está por leer es una página HTML5. Se diferencia del conocido **<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">** que se emplea en las versiones anteriores de HTML.

Header

En la etiqueta **<header></header>** se ubica la cabecera de la página; por ejemplo, un logo, una descripción, una imagen de la empresa o un formulario de búsqueda. Podemos tener más de una etiqueta en todo el documento; se puede utilizar, por ejemplo, como título de un artículo.

Nav

La etiqueta **<nav></nav>** es el lugar reservado para el navegador de nuestro sitio, formado por enlaces a determinados lugares de nuestra web o enlaces externos. También hay que tener en cuenta que no todos los links deben estar entre las etiquetas **<nav></nav>**; por ejemplo, los enlaces que se encuentran en el pie de página que ya están contenidos por la etiqueta **<footer></footer>** no deben colocarse entre**<nav></nav>**.

Como en el ejemplo anterior, tenemos que poner los enlaces entre las etiquetas **
**, para crear una lista no ordenada, y cada punto o enlace, entre las etiquetas **
**.

Article

Esta nueva etiqueta de HTML5 representa exclusivamente el contenido de una página web, como ser: comentarios de foros, entradas de blogs, artículos de periódicos o revistas, comentarios enviados por un usuario o cualquier otro contenido estático. Este elemento debería tener un título **<h1></h1>** que podría ir entre etiquetas **<header></header>.**

Section

La etiqueta **<section></section>** agrupa el contenido que se refiere a un mismo tema; por ejemplo, si tenemos un sitio web en donde ofrecemos productos de computación y, a su vez, las distintas formas de pago, una sección serían los productos, y otra, la forma de pago. Otro ejemplo podría ser una web de noticias, en donde una sección sean las noticias internacionales, y otra, las de deportes. Y, a su vez, cada sección puede tener otra sección anidada: siguiendo el ejemplo anterior de la web de noticias, en la sección deportes podríamos incluir subsecciones dedicadas a cada uno de los deportes, como rugby, tenis, fútbol. Es muy importante que tengamos en claro qué vamos a incluir en cada sección, para después poder dividirla en subsecciones.

Aside

La etiqueta **<aside></aside>** se utiliza para todo el contenido tangencial, es decir, aquel que no tiene que ver con el contenido principal; por ejemplo, calendario de eventos, publicidad, enlaces externos, etcétera. Por lo general, estas etiquetas están representadas por barras laterales a la derecha o izquierda de nuestra página web.

Footer

El footer o pie de página, como su nombre lo indica, se ubica en la parte inferior de la página y puede incluir todos los elementos que necesitemos mostrar sin límite de contenido: generalmente, links, publicidad, derechos de autor, mapa del sitio, entre otras opciones.

W3C Y HTML5

W3C es un consorcio formado por distintas organizaciones a nivel mundial que establecen y desarrollan los estándares para el funcionamiento de la Word Wide Web o, simplemente, Web. Esta fue creada el 1 de octubre de 1994 por Tim Berners-Lee, científico británico que estableció la primera comunicación cliente-servidor usando el protocolo HTTP en el año 1989. La sede central del consorcio se encuentra en el MIT, en Estados Unidos. W3C es, a su vez, el encargado de regular los estándares para el lenguaje HTML5.

jQuery Mobile y PhoneGap

jQuery Mobile es un framework o entorno de desarrollo de interfaz de usuario que combina JavaScript, HTML5 y CSS3, para el desarrollo de aplicaciones móviles compatibles con la mayoría de los dispositivos móviles más populares.

Es un sistema multiplataforma, ya que nos permite crear aplicaciones que pueden ser ejecutadas sobre cualquier sistema operativo de dispositivos móviles.

Tal como fue definido en su sitio web **http://jquerymobile.com**, es un sistema de interfaz para todas las plataformas de dispositivos móviles más comunes, construido sobre **jQuery** y **jQuery UI**.

Una de las ventajas de JQuery Mobile es que, como aparece descripto en su sitio web, permite "escribir menos y hacer más". Así, en lugar de escribir aplicaciones para cada dispositivo, jQuery Mobile permite crear una sola aplicación para su sitio web o su aplicación móvil y que funcione en smartphones, tabletas y programas de escritorio.

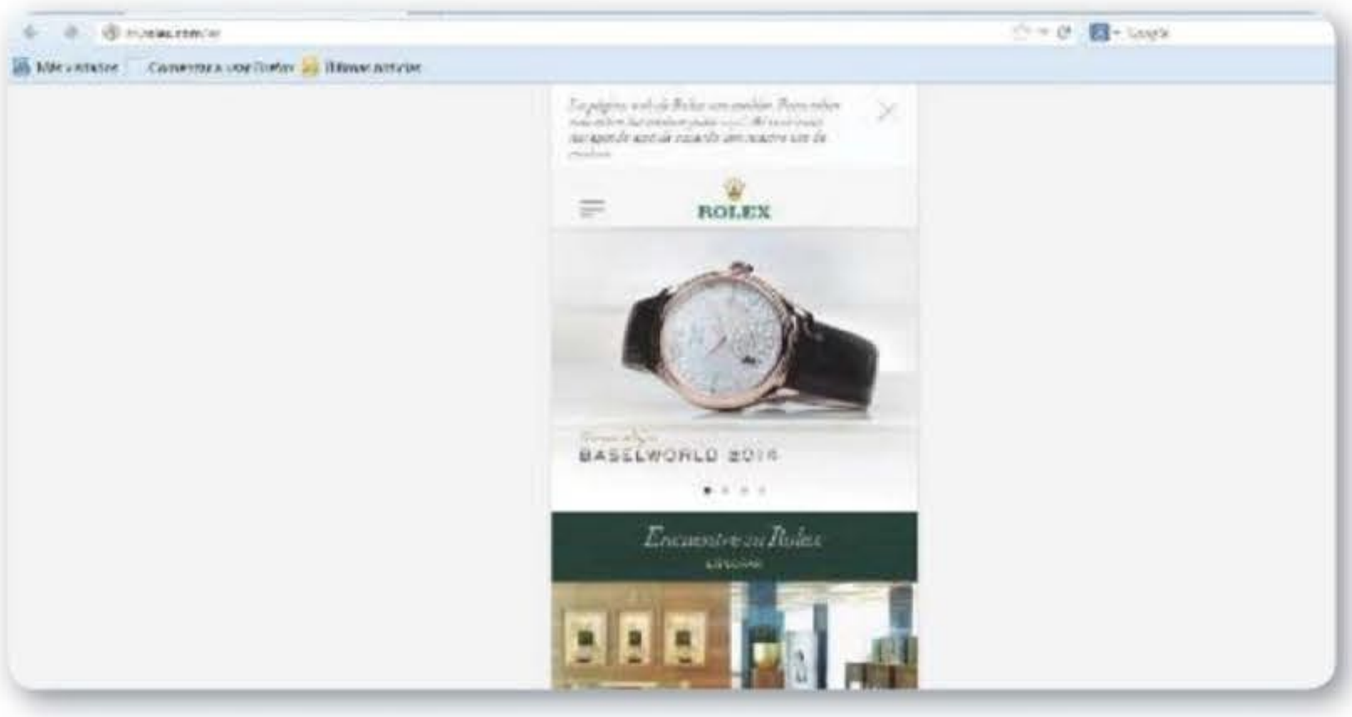


Figura 13. El sitio de **Rolex** (**http://m.rolex.com/en**) es un ejemplo de desarrollo con jQuery Mobile.

EJEMPLOS DE JQUERY MOBILE

En el sitio web de jQuery Mobile, **www.jqmgallery.com**, podemos encontrar una galería que nos muestra varios ejemplos de sitios web creados con jQuery Mobile. Recomendamos acceder desde nuestro dispositivo, así tenemos una mejor experiencia de usuario.

En cuanto a **PhoneGap**, este framework se diferencia de jQuery Mobile porque posee la ventaja de permitirnos compilar la aplicación y empaquetarla; de esta forma, queda lista para ser subida a la tienda de aplicaciones correspondiente. Por esta razón, podemos decir que PhoneGap es un framework para desarrollar aplicaciones móviles híbridas.

Distintos entornos de desarrollo

Al hablar de entornos de desarrollo para trabajar con jQuery Mobile, con un bloc de notas y un navegador en su última versión, como Chrome o Safari, bastaría, ya sea para Mac, Windows o Linux. Pero, si queremos trabajar cómodos, resaltar código y visualizar los errores, deberíamos pensar en otra cosa.

Podemos usar cualquier editor de HTML5 como, por ejemplo, **Notepad++**, **gedit**, **Sublime Text**, **Aptana Studio**, que son de distribución gratuita, u optar por **Adobe Dreamweaver Creative Cloud** o **Microsoft Visual Studio**, que son pagos.



Figura 14. Sitio web de **Adobe Creative Cloud** en español, desde donde podemos descargar aplicaciones y probarlas por 30 días, o bien asociarnos a un plan pago.

Tanto para jQuery Mobile como para PhoneGap, en este libro vamos a utilizar Dreamweaver CC, ya que, desde su versión **CS5.5**, nos permite trabajar y combinar ambos frameworks. En este editor, además, disponemos de una vista "en vivo", por lo que no tendremos la necesidad de volver a cargar el navegador cada vez que realizamos un cambio. Otra ventaja es que nos permite empaquetar directamente nuestra aplicación con PhoneGap sin salir del editor.



Figura 15. Se puede descargar una versión de prueba de **Dreamweaver CC** en **www.adobe.com/es/products/dreamweaver.html**.

RESUMEN

En este primer capítulo nos introdujimos en el mundo de las aplicaciones web. Para ello, comenzamos explicando los diferentes tipos de aplicaciones móviles (nativas, web, híbridas) y sus características más importantes. Después, hicimos un recorrido por los principales sistemas operativos de los dispositivos móviles que existen en el mercado actual. Para finalizar, realizamos una breve descripción de las herramientas que nos van a permitir diseñar nuestras propias aplicaciones hibridas.

Actividades

TEST DE AUTOEVALUACIÓN

1 ¿Cuál es la diferencia entre mobile web y Web?

2 ¿Qué tipos de aplicaciones web existen?

3 ¿Qué es jQuery Mobile?

4 ¿Qué es PhoneGap?

5 ¿Desde qué versión de Adobe Dreamweaver se puede trabajar con PhoneGap y jQuery Mobile?

EJERCICIOS PRÁCTICOS

Para observar sitios desarrollados con jQuery Mobile:

1 Visite el sitio de United Airlines: **http://mobile.united.com**.

2 Visite el sitio de Verizon Wireless: **https://m.verizonwireless.com**.

3 Visite el sitio de Disney World: **http://m.disneyworld.disney.go.com**.

4 Visite el sitio de Macworld: **www.macworld.com**.

5 Ingrese en el sitio **http://notepad-plus-plus.org** y descargue el editor **Notepad++**.

PROFESOR EN LÍNEA

Si tiene alguna consulta técnica relacionada con el contenido, puede contactarse con nuestros expertos: profesor@redusers.com

*02

Crear una página en jQuery Mobile

Describiremos cómo descargar y utilizar el framework de jQuery Mobile. Veremos cómo organizar nuestro proyecto y explicaremos cómo trabajar con Dreamweaver CC para crear nuestras aplicaciones. Posteriormente, vamos a conocer la estructura de una página y aprenderemos a crear una página básica con sus componentes principales. Por último, explicaremos cómo vincular las páginas que componen nuestra aplicación.

- El framework 36
- El desarrollo de un proyecto ... 38
- Página básica y multipágina.... 49
- Estructura básica 52
- Transiciones 58
- Resumen..... 59
- Actividades..... 60

↗↗↗

El framework

Las páginas de jQuery Mobile se construyen sobre la base de HTML5. Para comenzar a trabajar con el framework de jQuery Mobile, tenemos que incluir, en un archivo HTML, dentro de las etiquetas **<head></head>**, las librerías de jQuery Mobile.

Existen dos maneras de comenzar a desarrollar las aplicaciones con jQuery Mobile. Una es por **CDN** (*Content Delivery Network*), y la otra es descargándonos las librerías en un archivo comprimido desde el sitio **http://jquerymobile.com**. En la sección **Download** encontraremos la última versión del framework (**Lasted Version: 1.X.X**), disponible para descargar, además de otros recursos.

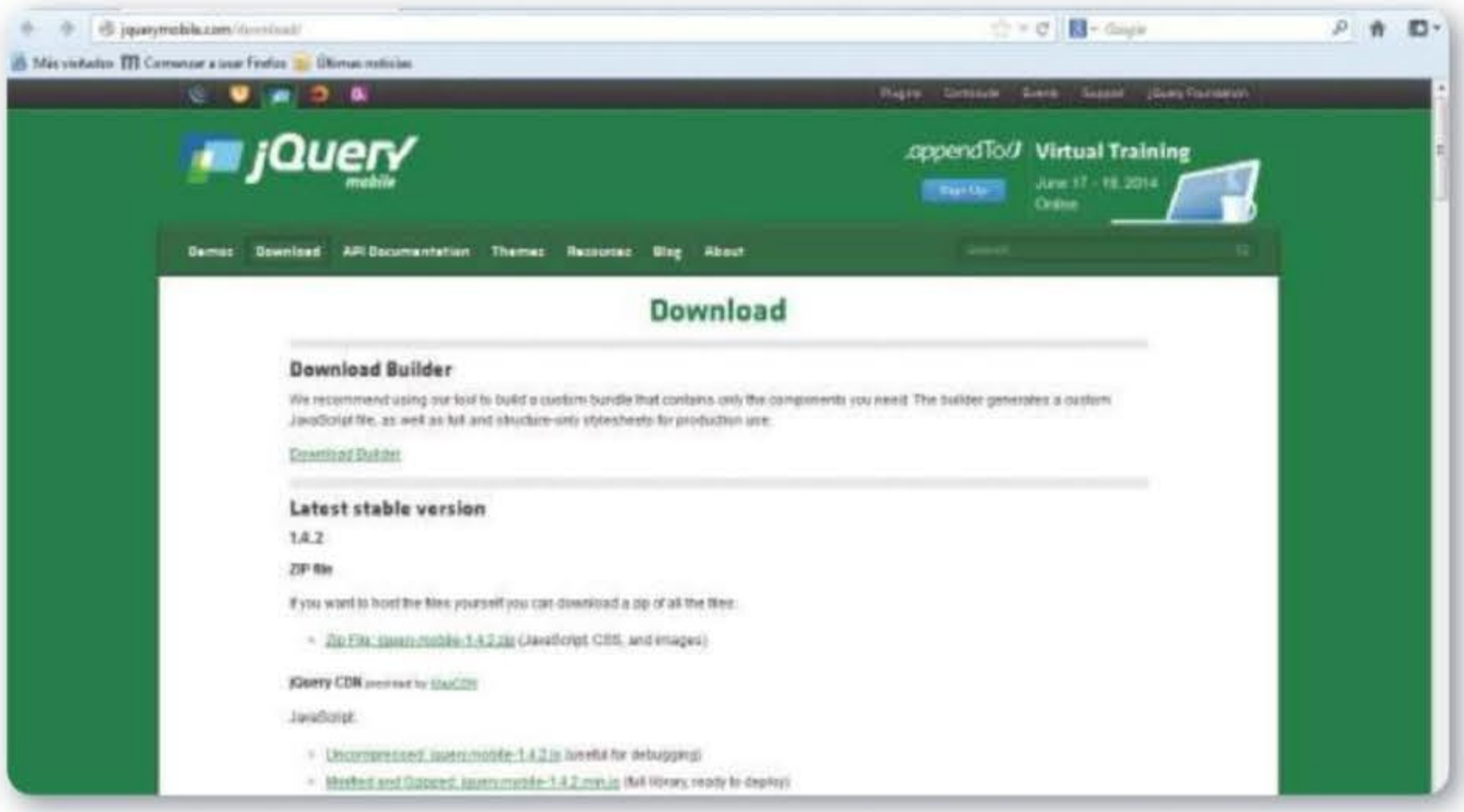


Figura 1. En el sitio web de jQuery Mobile podemos encontrar demos y otros recursos para utilizar en nuestra aplicación.

CDN

CDN consiste en una **red de distribución de contenido**.

Al usar este método, trabajamos con las librerías en un servidor externo alojado en Google, Microsoft o jQuery.

Por lo tanto, para utilizar las librerías de jQuery Mobile a través de estas redes de distribución, tenemos que incluir los enlaces que hacen referencia a ellas, en un archivo HTML5 entre las etiquetas **<head></head>**, ya presentadas en el **Capítulo 1**.

```
<link rel="stylesheet
 href=http://code.jquery.com/mobile/1.3.0/jquery.mobile-1.3.0.min.css />
<script src="http://code.jquery.com/jquery-1.9.1.min.js"></script>
<script src="http://code.jquery.com/mobile/1.4.2/jquery.mobile-1.3.0.min.js"></
      script>
```

De esta forma podremos comenzar a desarrollar la aplicación de manera inmediata, mediante las librerías de jQuery Mobile.

Si bien trabajar usando CDN es una ventaja, ya que alojamos las librerías en otro servidor, dependeremos siempre de que el servidor esté activo, de que tengamos conexión a internet y de que el servidor no falle o caiga inesperadamente.

Descargar y trabajar con el archivo comprimido

Si queremos trabajar de manera independiente en nuestro servidor o entorno de desarrollo, lo que tenemos que hacer es descargar el archivo comprimido o **Zip File**. El nombre del archivo es **jquery.mobile-X.X.X.zip**; donde la letra X indica la versión (por ejemplo, en el momento de escribir este libro, es la 1.4.2.).

Para trabajar localmente, solo tenemos que descomprimir el archivo en el directorio raíz de nuestro proyecto. En cambio, si disponemos de alojamiento web (*web hosting*), lo que deberíamos hacer es descargar el archivo Zip, descomprimirlo y, por medio de un cliente **FTP** (*File Transfer Protocol*) subirlo al servidor y ubicarlo dentro del directorio raíz de nuestro proyecto.

En ambos casos, solo tenemos que llamarlo por medio del siguiente **script** que incluimos entre las etiquetas **<head></head>** del archivo HTML5:

```
<link rel="stylesheet" href="themes/directorio.min.css" />
<script src="jquery-1.x.x.min.js" type="text/javascript"></script>
<script src="jquery.mobile-1.x.min.js" type="text/javascript"></script>
```

Este método es recomendado si vamos a trabajar con PhoneGap para crear una aplicación web, ya que necesitaremos tener estos archivos para poder armar nuestra aplicación.

```
<link rel="stylesheet
 href=http://code.jquery.com/mobile/1.3.0/jquery.mobile-1.3.0.min.css />
<script src="http://code.jquery.com/jquery-1.9.1.min.js"></script>
<script src="http://code.jquery.com/mobile/1.4.2/jquery.mobile-1.3.0.min.js"></
      script>
```

De esta forma podremos comenzar a desarrollar la aplicación de manera inmediata, mediante las librerías de jQuery Mobile.

Si bien trabajar usando CDN es una ventaja, ya que alojamos las librerías en otro servidor, dependeremos siempre de que el servidor esté activo, de que tengamos conexión a internet y de que el servidor no falle o caiga inesperadamente.

Descargar y trabajar con el archivo comprimido

Si queremos trabajar de manera independiente en nuestro servidor o entorno de desarrollo, lo que tenemos que hacer es descargar el archivo comprimido o **Zip File**. El nombre del archivo es **jquery.mobile-X.X.X.zip**; donde la letra X indica la versión (por ejemplo, en el momento de escribir este libro, es la 1.4.2.).

Para trabajar localmente, solo tenemos que descomprimir el archivo en el directorio raíz de nuestro proyecto. En cambio, si disponemos de alojamiento web (*web hosting*), lo que deberíamos hacer es descargar el archivo Zip, descomprimirlo y, por medio de un cliente **FTP** (*File Transfer Protocol*) subirlo al servidor y ubicarlo dentro del directorio raíz de nuestro proyecto.

En ambos casos, solo tenemos que llamarlo por medio del siguiente **script** que incluimos entre las etiquetas **<head></head>** del archivo HTML5:

```
<link rel="stylesheet" href="themes/directorio.min.css" />
<script src="jquery-1.x.x.min.js" type="text/javascript"></script>
<script src="jquery.mobile-1.x.min.js" type="text/javascript"></script>
```

Este método es recomendado si vamos a trabajar con PhoneGap para crear una aplicación web, ya que necesitaremos tener estos archivos para poder armar nuestra aplicación.

Al descomprimir el archivo Zip que descargamos del sitio de jQuery Mobile, vamos a encontrar un archivo con las imágenes (**images**), una carpeta **demos** con ejemplos, y los archivos JavaScript y CSS. Debemos tener presente que eso puede variar de acuerdo con la versión del framework que descarguemos.

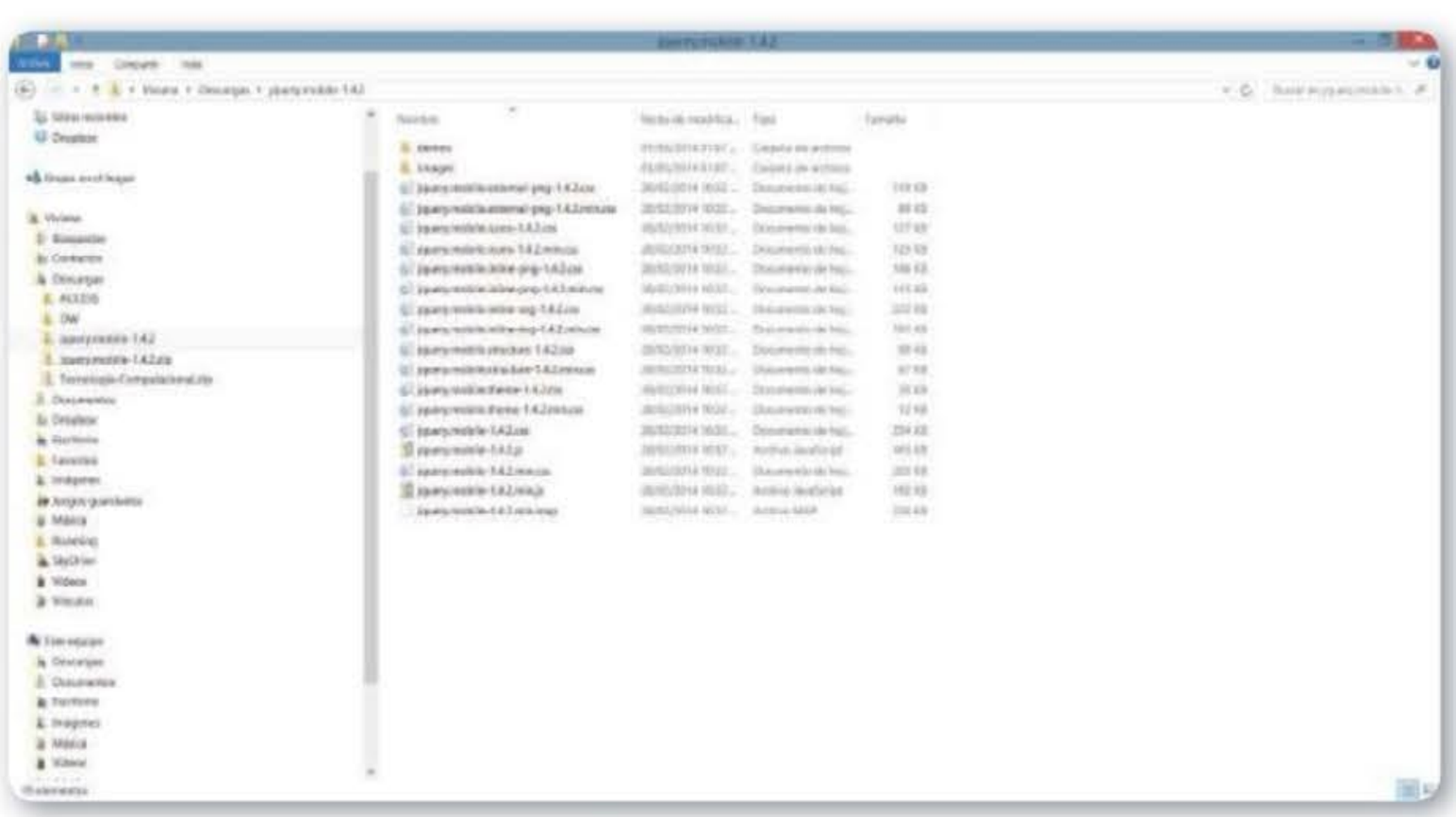


Figura 2. El archivo Zip contiene también las versiones reducidas de los archivos de JavaScript y CSS.

Además de los archivos incluidos en el archivo Zip, para desarrollar una aplicación necesitamos las librerías de jQuery.

Por lo tanto, también debemos descargar o enlazar (linkear) la librería de jQuery, en lo posible en su última versión.

```
<script src="http://code.jquery.com/jquery-1.X.X.min.js"></script>
```



El desarrollo de un proyecto

Para desarrollar un proyecto en jQuery Mobile, lo primero que debemos tener en cuenta es su organización: ya sea para este framework o para cualquier otro, es indispensable ser prolijos y ordenados, aún más cuando se trabaja con otras personas en el mismo proyecto. Empezaremos por ver cómo organizar el editor de código.

Organización del proyecto

Ya sea que utilicemos Dreamweaver o cualquier otro **editor de código**, como Bracket, Sublime text, gedit o Notepad++, para crear nuestra aplicación, lo primero que tenemos que hacer es organizarnos. Para ello, crearemos una estructura jerárquica de carpetas que contendrán todos los componentes que conformarán nuestro proyecto.

Por ejemplo, podemos crear la siguiente estructura de carpetas, como se muestra en la **Figura 3**.



Figura 3. Si bien Dreamweaver nos arma un directorio con el proyecto, siempre es conveniente que nosotros también lo hagamos.

En esta estructura, vemos lo siguiente:

- La carpeta **miProyecto** es el directorio principal. En esta carpeta vamos a guardar el archivo **index.html**. En el caso de trabajar con varios archivos HTML, podríamos crear otro subdirectorio llamado, por ejemplo, **archivos**, para almacenarlos.
- En el directorio **css** vamos a guardar todas las hojas de estilo.
- En el subdirectorio **imagenes** guardaremos todas las imágenes del proyecto. Incluso, para ser más prolijos, podemos añadir otras subcarpetas por cada categoría de imagen.

CSS

Las hojas de estilo en cascada (o **CSS**, Cascading Style Sheets) son un conjunto de instrucciones que definen la apariencia de uno o varios documentos HTML. CSS nos permite determinar colores, tipos de letra, líneas, imágenes de fondo, entre muchas otras opciones. Estas instrucciones que definen la apariencia del código pueden residir en un archivo externo o en alguna parte del documento HTML.

- En el subdirectorio **js** vamos a copiar y guardar todos los archivos con la extensión de JavaScript (.JS). En caso de que decidamos guardar los archivos de jQuery Mobile en nuestro servidor, irían en esta carpeta.

Trabajar con Dreamweaver

Como ya mencionamos, para crear nuestras aplicaciones podemos emplear cualquier editor de código. En este libro, vamos a utilizar Dreamweaver en su versión **CC** (**Creative Cloud 2014**) para realizar nuestros ejemplos. Se puede descargar una versión de prueba por 30 días de **www.adobe.com/la/products/dreamweaver.html**.

Dreamweaver es un potente editor de código HTML que nos permite diseñar, codificar y desarrollar aplicaciones web. Este software está disponible tanto para la plataforma Mac como para Windows.

A partir de la versión CS5.5, nos permite trabajar con jQuery Mobile.

DREAMWEAVER CS5.5 NOS PERMITE TRABAJAR CON JQUERY MOBILE PARA DISEÑAR APLICACIONES WEB

Definir el sitio local

Cuando vamos a trabajar con varios proyectos, es conveniente que definamos en Dreamweaver un sitio, así podremos organizar y administrar todos los archivos que componen la aplicación.

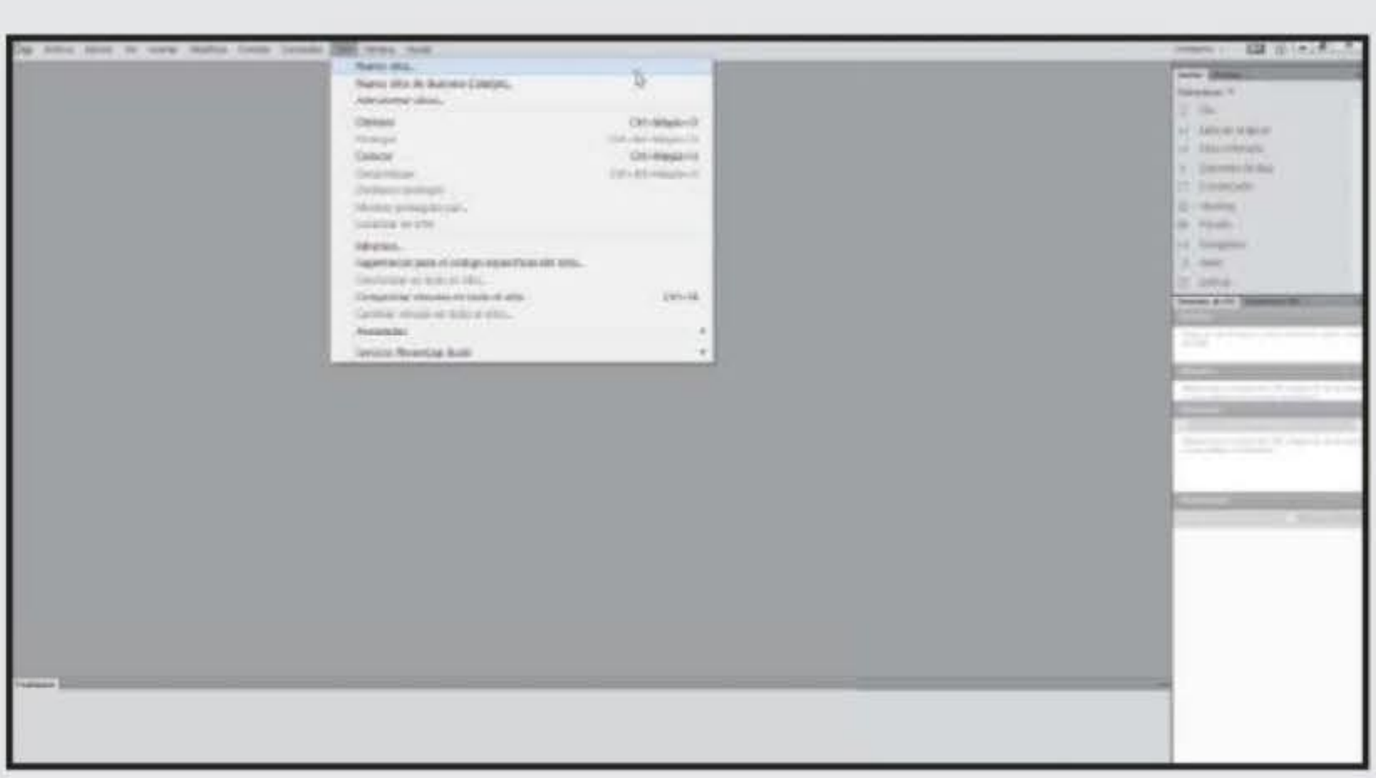
En Dreamweaver podemos crear un **sitio local** o un **sitio remoto**. El sitio local, por lo general, es una carpeta que se encuentra en nuestro disco rígido. En cambio, en un sitio remoto, la carpeta y los archivos de la aplicación se almacenan en un servidor web. El servidor web es el equipo donde se ejecutarán los archivos, y para ello debemos contar con la URL que le corresponde y, de esta manera, trabajaríamos online.

Al crear un sitio local, le estamos indicando a Dreamweaver que la carpeta raíz se encuentra en nuestro equipo y que en esa carpeta se almacenarán y se buscarán todos los archivos relacionados con nuestro proyecto.

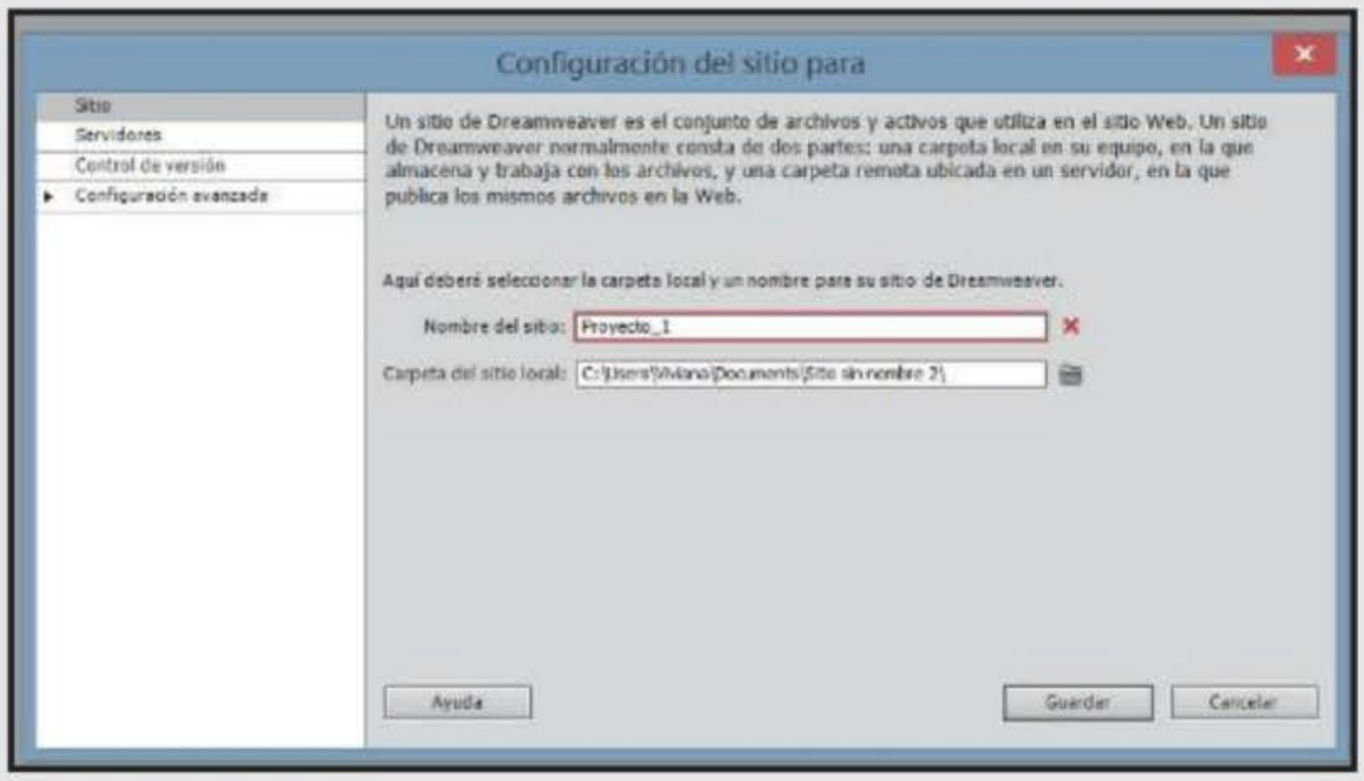
Una vez creada la estructura de carpetas en nuestro disco rígido, que contendrá todos los archivos del proyecto, seguiremos el siguiente **Paso a paso** para definir el sitio.

PAP: DEFINIR UN SITIO EN DREAMWEAVER

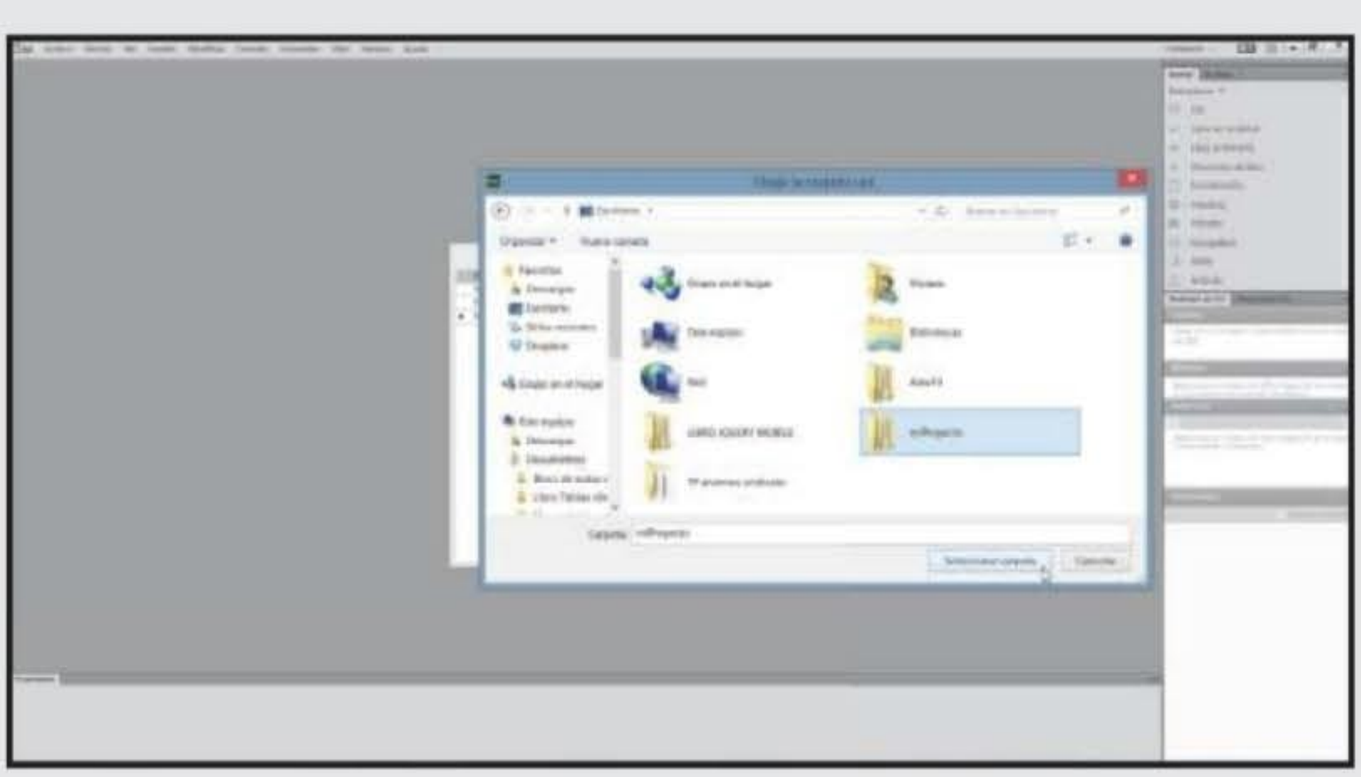
01 Haga clic en el menú `Sitio` y luego elija la opción `Nuevo sitio...`.



02 Se abrirá el cuadro de diálogo `Configuración del sitio para...`. En la opción `Nombre del sitio`, escriba un nombre para el sitio. Como ejemplo, escriba **Proyecto_1**; este nombre no se publicará con el sitio.



03 Haga clic en el icono de carpeta situado al lado del campo `Carpeta del sitio local` y busque la carpeta que creó para guardar los archivos del proyecto. En este caso, `miProyecto`. Luego presione `Seleccionar carpetas`.



04 Para finalizar, presione `Guardar`. En el `Panel de Archivos` aparecerán la carpeta raíz y las subcarpetas del sitio.



Una vez creado el sitio en Dreamweaver, el siguiente paso es crear la/s página/s de nuestra aplicación.

Abrir sitios locales

Cuando queremos trabajar en un proyecto en particular, debemos abrir el sitio del proyecto. Para ello, hacemos clic en el menú **Sitio** y luego seleccionamos la opción **Administrar sitios**. En el cuadro de diálogo que aparece, seleccionamos el nombre de nuestro sitio. Finalmente, presionamos el botón **Listo**.

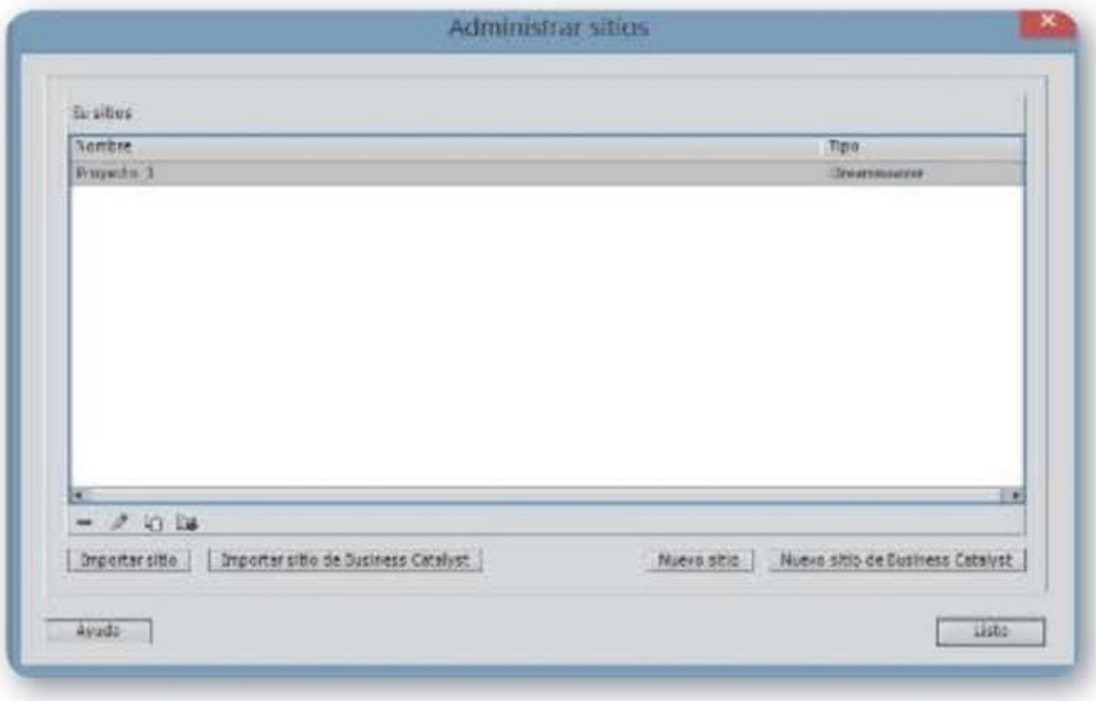


Figura 4. Desde la ventana **`Administrar sitios`** podremos crear un nuevo sitio, exportar o importar un sitio ya existente.

Editar sitios locales

Si, por ejemplo, necesitamos cambiar la ubicación de la carpeta de nuestro sitio, o queremos cambiar el nombre de la carpeta del sitio local, tendremos que editar el sitio que hemos creado en Dreamweaver.

Para editar el sitio, en la barra de menú hacemos clic en **Sitio** y elegimos la opción **Administrar sitios...**. En el cuadro de diálogo **Administrar sitios**, seleccionamos el nombre del sitio que vamos a editar, luego presionamos el botón **Editar el sitio seleccionado actualmente**.

> DREAMWEAVER NOS PERMITE ABRIR, EDITAR Y ELIMINAR SITIOS LOCALES DE NUESTRO PROYECTO

Se abrirá el cuadro de diálogo **Configuración del sitio**, donde podemos modificar la información que queramos del mismo modo como lo realizamos cuando creamos un nuevo sitio. Una vez realizadas las modificaciones, guardamos los cambios presionando el botón **Guardar**. Para finalizar, pulsamos el botón **Listo** del cuadro de diálogo **Administrar sitios**.

Eliminar un sitio

Si, por alguna razón, ya no vamos a necesitar trabajar con un sitio, podremos removerlo de la lista de sitios de Dreamweaver.

Tengamos en cuenta que, cuando quitamos un sitio de la lista, no eliminamos las carpetas y los archivos que forman el sitio, solo borramos la información de su configuración.

Para quitar un sitio, en la barra de menú hacemos clic en **Sitio** y elegimos la opción **Administrar sitios...**. En el cuadro de diálogo **Administrar sitios**, seleccionamos el nombre del sitio que queremos remover y luego presionamos el botón **Eliminar el sitio o los sitios seleccionados actualmente**.

Aparecerá un cuadro de diálogo en el que se nos pide la confirmación para quitar el sitio; presionamos el botón **Sí**. Por último, para finalizar, pulsamos el botón **Listo** del cuadro de diálogo **Administrar sitios**.

Las vistas en Dreamweaver

En Dreamweaver podemos seleccionar distintas vistas para trabajar con un archivo. En la barra de herramientas de **Documento**, encontramos las siguientes opciones:



Figura 5. Estas son las opciones de visualización de la barra de herramientas de **Documento**. También podemos acceder a estas vistas desde el menú **Ver**.

DUPLICAR UN SITIO

Si, por algún motivo, necesitamos crear una copia del sitio en Dreamweaver, podemos ir al menú **Sitio/Administrar sitios...**. En el cuadro de diálogo **Administrar sitios**, seleccionamos el sitio que queremos copiar y, luego, presionamos el botón **Duplicar el sitio o los sitios seleccionados actualmente**.

- **Código**: nos muestra el entorno de programación. Aquí es donde escribimos y editamos el código de nuestra aplicación. Con esta vista, solo visualizaremos el código.
- **Dividir**: esta opción nos divide la pantalla de trabajo en dos paneles; en panel de la izquierda, nos muestra el código y, en el panel de la derecha, la representación visual del código. Cuando realizamos un cambio en alguno de los paneles, este cambio se ve reflejado directamente en el otro panel.
- **Diseño**: esta vista nos muestra la representación visual del código de programación, de manera similar a como la veríamos en cualquier navegador.
- **En vivo**: con esta vista, podemos ver e interactuar con nuestra aplicación como si la viéramos en un navegador o dispositivo móvil. La vista en vivo no es editable. Al seleccionar este modo, podremos activar la vista **Código en vivo**. Podemos acceder también a esta opción presionando la combinación de teclas **ALT** + **F11**.
- **Código en vivo**: con esta vista podemos ver e interactuar con nuestra aplicación como si la viéramos en un navegador o dispositivo móvil. Esta vista, también, divide la pantalla en dos paneles. En el panel de la derecha, muestra el código (resaltado en color amarillo) que un navegador emplearía para ejecutar la aplicación. Cuando interactuamos con los elementos de la aplicación, se resalta el código del elemento que se seleccionó.
- **Vista previa**: al hacer un clic en esta opción, podremos elegir el navegador que tengamos instalado: Firefox, Chrome, Safari o Internet Explorer (incluso, podemos agregar uno nuevo) para visualizar nuestra aplicación en el navegador seleccionado. Para seleccionar esta vista, podemos también presionar la tecla **F12**.

> LA VISTA "DIVIDIR" MUESTRA, EN DOS PANELES, EL CÓDIGO Y SU REPRESENTACIÓN VISUAL

Lo más recomendable es trabajar con la opción **Dividir**. De esta manera, mientras trabajamos en el código podemos ver su diseño, y, haciendo clic en el botón **En vivo**, vamos a ver nuestro documento tal cual lo veríamos en nuestro navegador o dispositivo móvil.

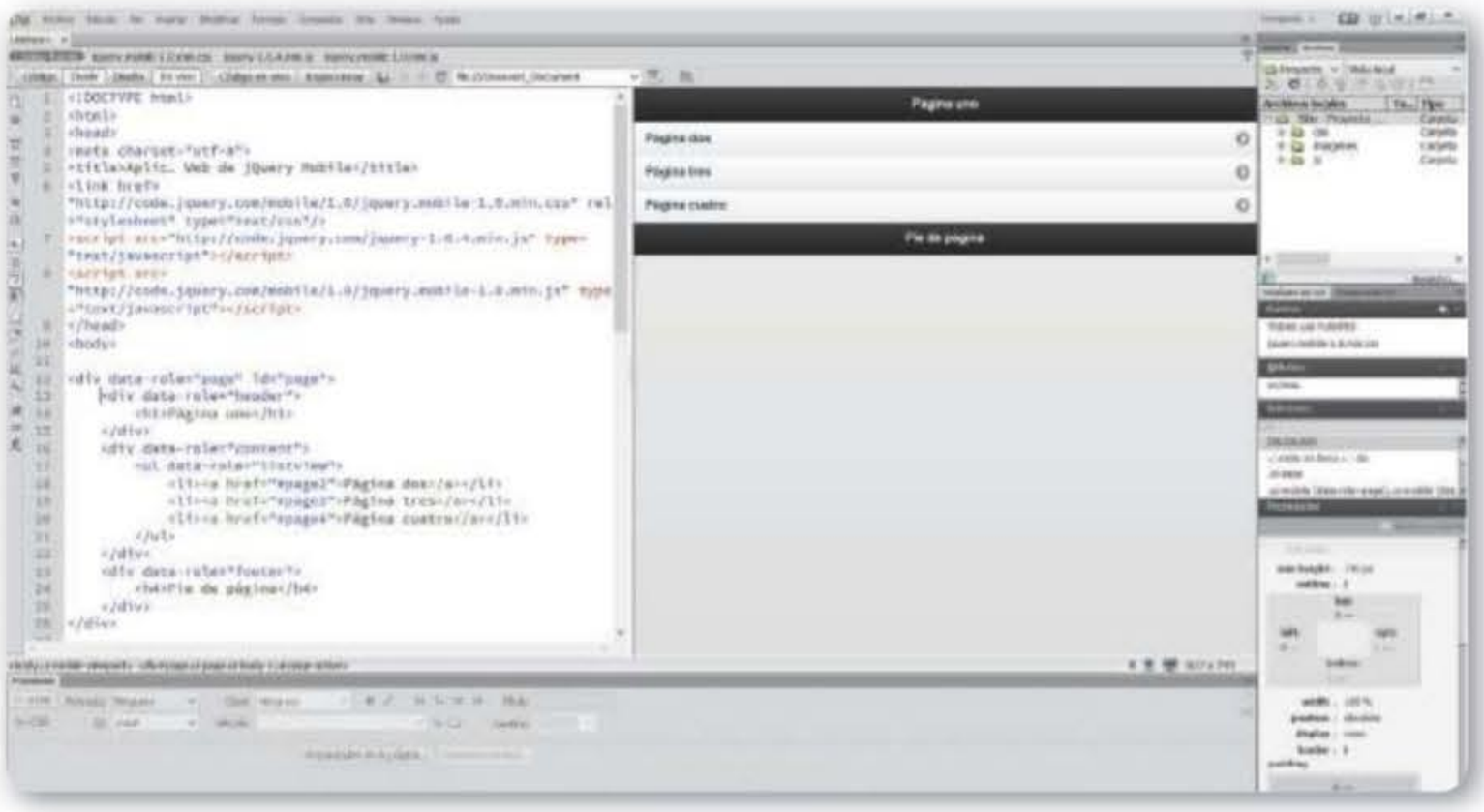


Figura 6. La vista **En vivo** utiliza el motor renderizado WebKit de los navegadores Safari y Chrome.

Mostrar la aplicación en diferentes resoluciones de pantallas

Recordemos siempre que el usuario puede ver o usar nuestra aplicación desde el escritorio o desde cualquier dispositivo móvil, con diferentes tipos de resoluciones. Estas diferencias pueden hacer que nuestros diseños se aprecien bien en un dispositivo y mal en otros; por ello, es conveniente tratar de visualizar nuestra aplicación en distintos tipos de resoluciones, lo que en diseño web se denomina **diseño web adaptable** (o, en inglés, *responsive*), y así poder hallar un diseño que sea correctamente presentado entre las diferentes resoluciones de pantalla.

Dreamweaver CC nos ofrece las herramientas para visualizar nuestra aplicación en diferentes resoluciones de pantalla.

Cuando visualizamos la página, ya sea en la vista **Diseño**, **Dividir** o **En vivo**, a la derecha de la **barra de estado** encontraremos cuatro

TRABAJAR CON OTROS EDITORES

Podemos crear aplicaciones con algún editor de texto (como, por ejemplo, **Notepad++**, **Sublime text** o **gedit**) y, si disponemos de un navegador como Chrome o Safari en su última versión y tenemos instalado **Xampp** o similar, podremos usar nuestra computadora como un servidor web.

botones que nos permiten mostrar nuestra aplicación según un tipo y resolución de pantalla:

- **Tamaño móvil** (480 x 800).
- **Tamaño tableta** (768 x 1024).
- **Tamaño escritorio** (variable).

Crear páginas de jQuery Mobile

Como ya hemos mencionado, las páginas de jQuery Mobile se construyen sobre la base de HTML5.

Podemos crear una página jQuery Mobile manualmente o usar las páginas predefinidas (**plantillas de inicio**) que nos ofrece Dreamweaver.

Usar las plantillas de inicio de Dreamweaver

Para usar una plantilla de inicio de Dreamweaver, hacemos clic en el menú **Archivo** y luego seleccionamos **Nuevo**. En el cuadro de diálogo **Nuevo documento**, elegimos **Plantillas de inicio/Mobile Starters**, donde tendremos tres opciones:

- **jQuery Mobile** (**CDN**): con esta opción usamos las librerías almacenadas en un servidor remoto.
- **jQuery Mobile** (**Local**): con esta opción, las librerías se encuentran ubicadas en nuestro disco.
- **jQuery Mobile con tema** (**Local**): esta opción utiliza archivos CSS ubicados en nuestro disco, separados en componentes de estructura y temas, lo que significa usar una plantilla prediseñada en jQuery Mobile.

En todos los casos, en la opción **Tipo de documento** dejamos la que viene por defecto: **HTML5**. Para finalizar, presionamos el botón **Crear**.

Las plantillas de inicio incluyen archivos de imagen y las librerías de jQuery Mobile. Siempre debemos asegurarnos de que se trata de la última versión disponible de jQuery Mobile, tanto para Dreamweaver CS5.5, CS6 y CC, o bien trabajar directamente por CDN, ya que cuenta con las últimas actualizaciones y correcciones del framework.

Para esto es conveniente ir al sitio de jQuery Mobile y comparar el número de versión de la web (*latest stable version*) con la versión que incluye el Dreamweaver, y, si difieren, borrarla y dejar la más reciente, o sea, la que descargamos del sitio web oficial de jQuery Mobile.

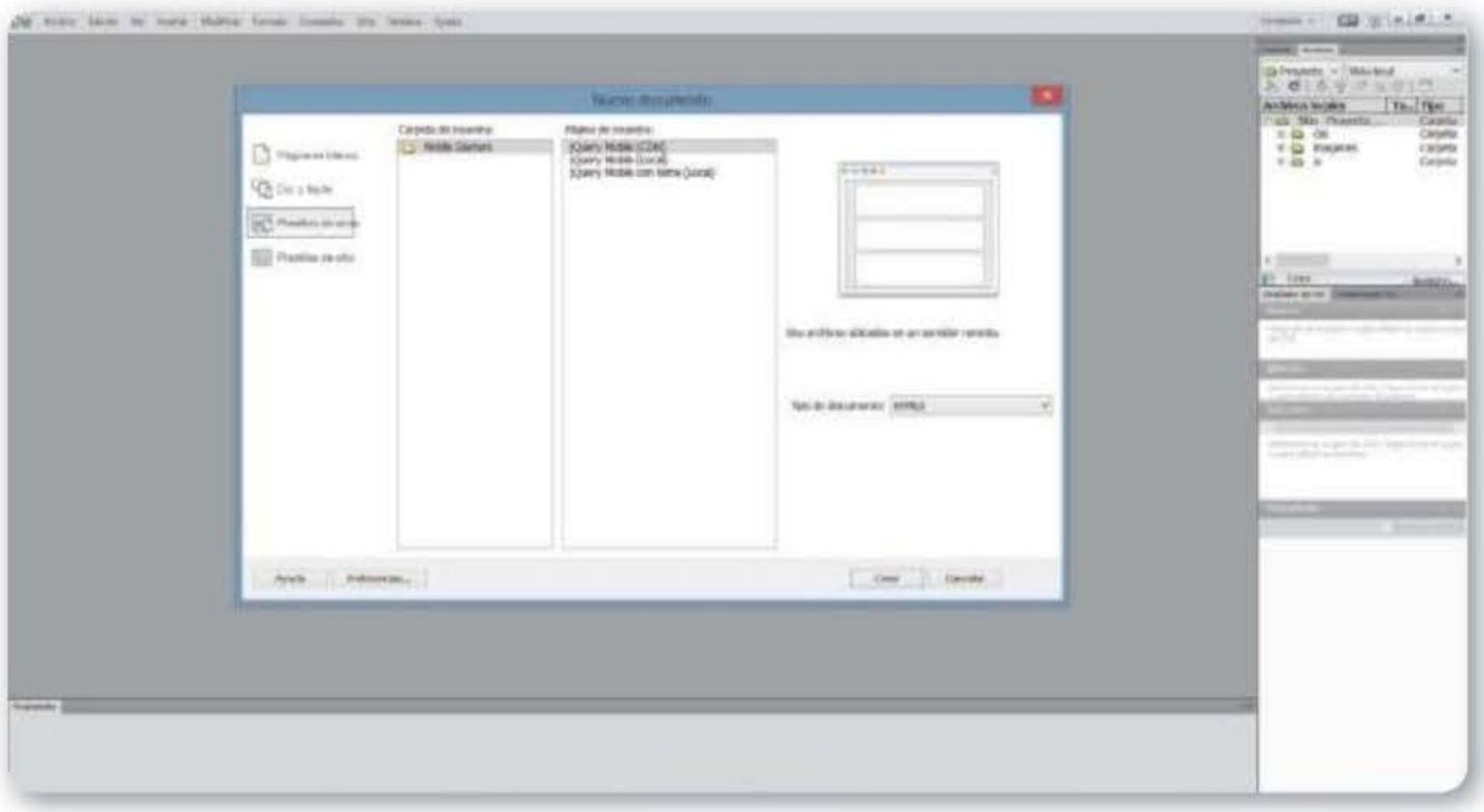


Figura 7. Las plantillas de inicio nos agilizan la tarea a la hora de diseñar nuestra aplicación.

Crear una página manual

Para agregar una página a nuestro proyecto, hacemos clic en el menú **Archivo/Nuevo**. En el cuadro de diálogo **Nuevo documento**, elegimos **Página en blanco**, luego seleccionamos en **Tipo de página: HTML** y, como tipo de documento, **HTML5**. Finalmente, presionamos el botón **Crear**.
Nos aparecerá una página HTML, donde vamos a tener que incluir o referenciar las librerías de jQuery Mobile entre las etiquetas **<head></head>**.

Por ejemplo, si queremos cargar las librerías de jQuery Mobile, jQuery, desde el servidor CDN, tendremos que incluir el siguiente código entre las etiquetas **<head></head>**:

```
<!doctype html>
<html>
```

EDITOR BRACKET

Otra alternativa para crear aplicaciones es usar el editor **Bracket**, de código abierto (**http://brackets.io/**), apto para usar con JavaScript, CSS3 y HTML5, que muestra en tiempo real (usando el navegador Google Chrome) las modificaciones que vamos haciendo en nuestro proyecto, igual a Dreamweaver. Se puede usar tanto en Linux, Mac o Windows.

```
    <head>
        <meta charset="utf-8">
        <link rel="stylesheet" href="http://code.jquery.com/mobile/1.4.2/jquery.
mobile-1.4.2.min.css" />
        <script src="http://code.jquery.com/jquery-1.9.1.min.js"></script>
        <script src="http://code.jquery.com/mobile/1.4.2/jquery.mobile-1.4.2.min.
js"></script>
        <title>Mi primera web jQuery Mobile</title>
    </head>
    <body>
    </body>
    </html>
```

Página básica y multipágina

Una vez que hemos incluido las librerías necesarias en el archivo HTML5, vamos a desarrollar la aplicación web.

La estructura básica de una página jQuery Mobile es la siguiente:

```
<div data-role="page" id="pagina1">
        <div data-role="header">
                <h1>TÍTULO</h1>
        </div>
        <div data-role="content">
                <h2>CONTENIDO<h2>
        </div>
```



JQUERY

Es una librería de JavaScript que simplifica la forma de utilizarlo. Es un framework o conjunto de funciones y procedimientos ya desarrollados para ser utilizados de manera más simple. Se puede decir entonces, que jQuery Mobile deriva directamente de jQuery, aunque está adaptado para simplificarnos el desarrollo en dispositivos móviles.

```
        <div data-role="footer">
                <h4>PIE DE PÁGINA</h4>
        </div>
</div>
```

Como podemos observar en el código, la estructura no es más que etiquetas ** ** de HTML con CSS (hojas de estilo). Cada ** ** cumple una función específica combinada con la propiedad **data**.

Dentro de las etiquetas **<body></body>**, dividimos la página en zonas, mediante las etiquetas ** ** y el atributo **data-role**.

> UN DOCUMENTO MULTIPÁGINA ES UN DOCUMENTO CON MÁS DE UNA PÁGINA EN EL ARCHIVO HTML

Cada archivo HTML o documento jQuery Mobile puede contener una o más páginas. Cada página está definida con el atributo **data-role="page"** seguido del **id** (identificador) que la hace única **id="pagina1"**. Entonces, si tenemos más de una página en el archivo HTML, eso es lo que llamaríamos un documento jQuery Mobile **multipágina**. El siguiente código nos muestra una estructura multipágina compuesta por dos páginas: **pagina1** y **pagina2**.

```
<div data-role="page" id="pagina1">
        <div data-role="header">
                <h1>TÍTULO</h1>
        </div>
        <div data-role="content">
                <h2>CONTENIDO<h2>
        </div>
        <div data-role="footer">
                <h4>PIE DE PÁGINA</h4>
```

ATRIBUTO DATA

El framework de jQuery Mobile utiliza las etiquetas HTML y sus atributos. Por ejemplo, usa el prefijo de HTML5 **data-atributo**; estos atributos son opcionales y permiten al framework asignar roles y otras características complementarias a nuestro documento (**DOM-Document Object Model**) a través de jQuery.

```
        </div>
</div>

<div data-role="page" id="pagina2">
        <div data-role="header">
                <h1>TÍTULO</h1>
        </div>
        <div data-role="content">
                <h2>CONTENIDO<h2>
        </div>
        <div data-role="footer">
                <h4>PIE DE PÁGINA</h4>
        </div>
</div>
```

Con el atributo **data-role="page"** indicamos que entre esos **div** HTML de apertura y cierre estamos creando una página jQuery Mobile dentro del documento y que va a tener un identificador o nombre único e irrepetible con el atributo **id**.

En nuestro ejemplo, con la siguiente línea de código estamos llamando a la página **pagina1**:

```
<div data-role="page" id="pagina1">
</div>
```

Y con la siguiente línea de código creamos otra página llamada **pagina2**:

```
<div data-role="page" id="pagina2">
</div>
```

VISUALIZAR EL PROYECTO EN BRACKET

Hacemos clic en el menú **Archivo**, luego seleccionamos **Vista previa en vivo**. Nos va a mostrar, en el navegador Google Chrome (solo admite este navegador), una vista previa en vivo del documento jQuery Mobile. Si modificamos algo, vamos a ver cómo va a ir cambiando la vista en vivo. Esto resulta muy útil a la hora de programar.



Figura 8. Así se vería la estructura de una página básica en una tableta o un teléfono.

Estructura básica

El framework dispone de una estructura básica creada para dispositivos móviles, lista para comenzar un proyecto. Esta puede ser modificada según nuestra necesidad.

Meta viewport

Al disponer de un navegador para acceder a la Web y de numerosos dispositivos móviles con distintas resoluciones de pantalla, nos encontramos con que no es lo mismo realizar una aplicación para un iPhone que para un dispositivo Samsung Galaxy S5 o un iPad.

Por esta razón, necesitaríamos no solo saber la resolución de cada uno de los dispositivos, sino también deberíamos adaptar nuestra aplicación a cada resolución. Es decir, sería necesaria una aplicación para cada dispositivo.

Si queremos que nuestra aplicación se adapte a cada resolución, tenemos que usar la etiqueta **viewport**. En ella se puede indicar el ancho, el nivel de zoom inicial y máximo.

```
<meta name =”viewport” content=”width=device-width,initial-scale=1, maxi-
mum-scale=1” />
```

En el caso del ancho, es siempre conveniente que sea igual al ancho del dispositivo y no ponerle uno específico, a no ser que estemos trabajando una aplicación para un dispositivo en particular. Al limitar el zoom, el usuario no va a poder ampliar el texto. Es muy importante que, además de probarlo con las distintas formas de visualizar de Dreamweaver, lo probemos en diferentes dispositivos con variadas resoluciones de pantalla, para estar seguros de que se vea bien.

Header bar

Es una barra en la parte superior de la aplicación que, por lo general, contiene el título de la página, un logo y, además, botones opcionales situados a la izquierda o derecha del título.

Los encabezados generalmente permanecen fijos en la parte superior en todo momento, en vez de desplazarse por la página.

Para crear un encabezado, utilizamos el atributo **data-role="header"**.

Por ejemplo, las siguientes líneas de código nos permiten crear un **encabezado de página** (**header**) donde vamos a incluir el título "Mi aplicación".

```
<div data-role="header">
        <h1>Mi aplicación<h1>
</div>
```

Footer bar

Es una barra que se ubica en la parte inferior de la aplicación y que, por lo general, contiene los elementos de navegación (botones) o información general, como el copyright o los derechos de autor.



ETIQUETAS H1...H6

HTML nos provee de etiquetas para definir los títulos: estas son **<h1>**, **<h2>**, **<h3>**, **<h4>**, **<h5>** y **<h6>**. La etiqueta **<h1>** es la de mayor importancia y, por lo general, se la emplea para definir los títulos de la página. El resto de etiquetas es de importancia descendiente, hasta llegar a la etiqueta **<h6>**, que nos permite definir los subtítulos de las páginas.

Para crear un **pie de página**, empleamos el atributo **data-role="footer"**. En el siguiente ejemplo, creamos un pie de página donde incluimos un título utilizando las etiquetas de HTML **<h4></h4>**:

```
<div data-role="footer">
        <h4>Pie de Página<h4>
</div>
```

Tanto el encabezado (*header bar*) como el pie de página (*footer bar*) pueden estar fijos, lo que facilita la experiencia de usuario. Por ejemplo, si tenemos muchos datos o no queremos dificultar la visión del contenido, podemos hacer que ambas barras aparezcan al sacar los dedos de la pantalla y desaparezcan mientras estamos desplazándonos a través del contenido. Para esto, disponemos del atributo **data-position= "fixed"**.

Con el atributo **data-position="inline"**, las barras de encabezado y pie de página no quedarían fijas y, si tenemos muchos datos, tendríamos que ir hasta arriba para recién poder ver los botones del header bar.

En el siguiente ejemplo, creamos un pie de página empleando el atributo **data-position="fixed"**:

```
<div data-role="footer" data-position="fixed">
        <h4>Pie de Página<h4>
</div>
```

No es necesario poner **data-position="inline"**, ya que es la posición predeterminada por jQuery.

Otra manera de dejar fijas estas barras es trabajar en pantalla completa con el atributo **data-fullscreen** que, poniéndolo en **"true"**, desaparece al tocar con el dedo la pantalla, y soltando se ve transparente mientras nos desplazamos por el contenido de la página.

```
<div data-role="footer" data-fullscreen="true">
        <h4>Pie de Página<h4>
</div>
```

Este atributo es utilizado generalmente al mostrar una imagen, para que no se vean las barras y así facilitar la visión.

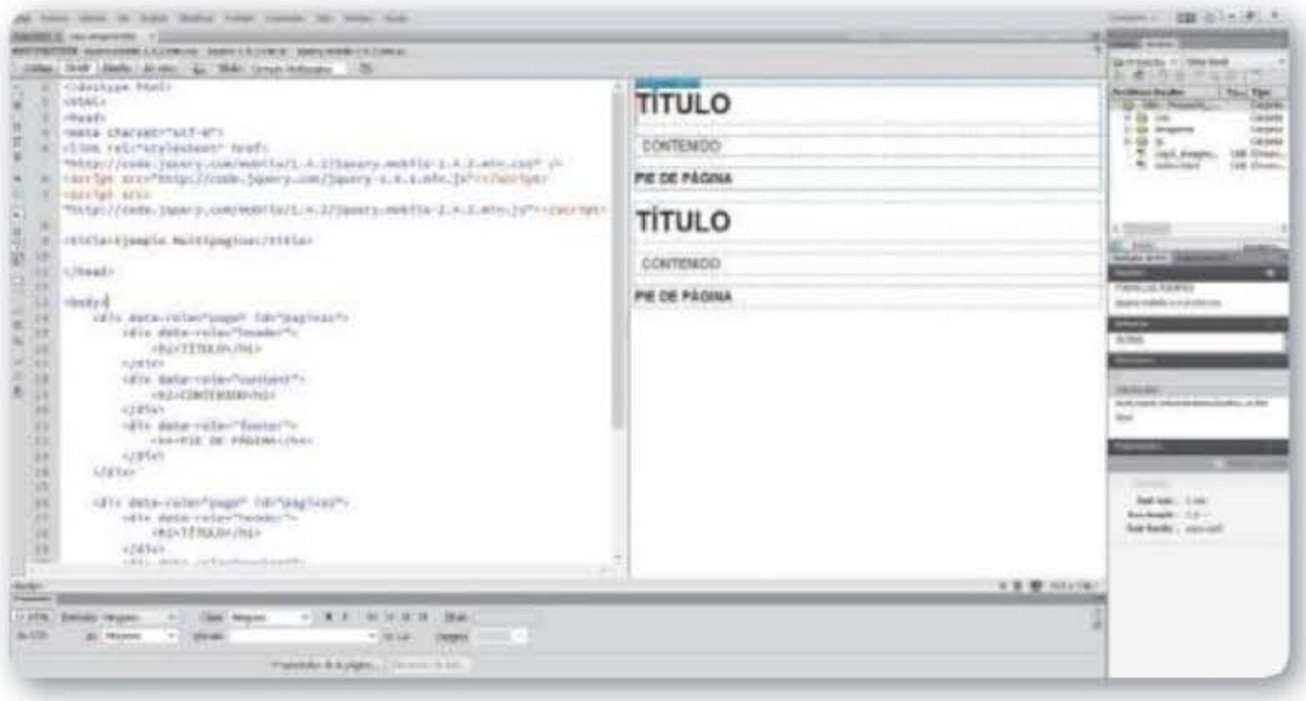


Figura 9. Cada página puede contener un encabezado o pie de página. Ambos son opcionales.

Content section

En la **sección de contenido** (o **content section**) vamos a agregar todo el contenido de la página. Si lo comparamos con una página web estándar, sería lo que va dentro del **cuerpo** (o **body**) en HTML, como ser: botones, información, imágenes, enlaces.

Para crear el área de contenido, usamos el atributo **data-role"content"**.

```
<div data-role="content">
        <h2>CONTENIDO<h2>
</div>
```

Enlaces

Tenemos tres opciones para ir de una página a otra:

- Llamar a una página dentro de un mismo documento jQuery Mobile.
- Llamar a una página externa desde otro documento jQuery Mobile de una misma aplicación.
- Llamar a una página externa alojada en otro servidor.

Para llamar a una página dentro del mismo documento, usamos el símbolo "#", como en el siguiente ejemplo:

```
<a href="#pagina2" data-role="button">Ir a página 2</a>
```

En la etiqueta **<a href>** de HTML colocamos el id de la página a la que queremos ir, anteponiendo "**#**", seguido del atributo **data-role** y pasándole el valor "**button**" (**data-role="button"**), más la leyenda o título del botón (por ejemplo: "ir a página 2").

Entonces, si tuviéramos dos páginas con dos identificadores: **pagina1** y **pagina2**, y desde **pagina1** queremos llamar a **pagina2**, la estructura sería la siguiente:

```
<div data-role="page" id="pagina1">
        <div data-role="header">
                <h1>TÍTULO</h1>
        </div>
        <div data-role="content">
                <h2>CONTENIDO<h2>
                <a href=#pagina2 data-role="button">Ir a página 2</a>
        </div>
        <div data-role="footer">
                <h4>PIE DE PÁGINA</h4>
        </div>
</div>

<div data-role="page" id="pagina2">
        <div data-role="header">
                <h1>TÍTULO</h1>
        </div>
        <div data-role="content">
                <h2>CONTENIDO<h2>
        </div>
        <div data-role="footer">
                <h4>PIE DE PÁGINA</h4>
        </div>
</div>
```

ENLACES EN HTML

Un enlace o link tiene, en HTML, un origen y un destino que apunta a cualquier recurso de la Web: una imagen, un video, otra página, etcétera. En la etiqueta **Destino**, el atributo **href** hace referencia al destino por medio de una URL que puede ser local o externa.

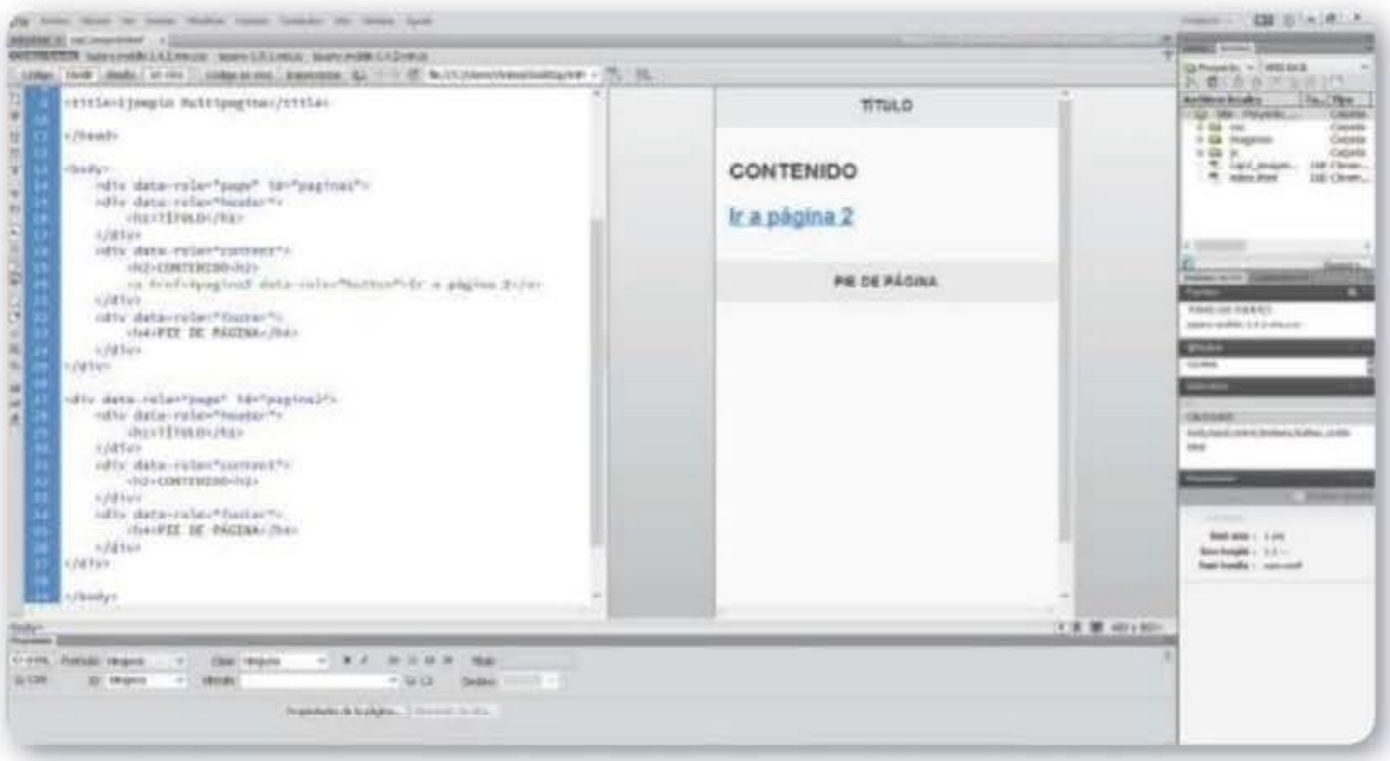


Figura 10. Podemos convertir los enlaces en botones usando los estilos que se incluyen en las hojas de estilos.

Cuando llamamos a una página externa dentro del mismo servidor o dominio (es decir, a otra página o archivo dentro de nuestro proyecto), el framework interpreta que estamos dentro de la misma aplicación y nos agrega una transición como si estuviera en el mismo documento. Vale decir que es lo mismo tener todo en un mismo archivo o documento jQuery Mobile que en varios documentos. Lo más recomendable es tener varios documentos si nuestra aplicación es muy grande, ya que resulta mucho más prolijo.

En el caso de que, por algún motivo, no quisiéramos el efecto de transición, tendríamos que agregarle al link el atributo **"data-ajax="false""** y, al ir a la nueva página, la volvería a cargar como si fuera una página nueva, sin usar ningún efecto de transición, como vemos en el siguiente ejemplo:

```
<div data-role="page" id="pagina1">
        <div data-role="header">
                <h1>TÍTULO</h1>
</div>
<div data-role="content">
                <h2>CONTENIDO<h2>
    <a href="pagina2.html" data-ajax="false">Página 2</a>
</div>
<div data-role="footer">
                <h4>PIE DE PÁGINA</h4>
```

```
        </div>
</div>
```

Por último, para llamar a una página externa alojada en otro servidor, se haría de la misma manera, pero en este caso no tendría transición de ningún tipo.

```
<div data-role="page" id="pagina1">
        <div data-role="header">
                <h1>TÍTULO</h1>
        </div>
        <div data-role="content">
                <h2>CONTENIDO<h2>
            <a href="www.paginaExterna.html"> Página Externa en otro servidor</a>
        </div>
        <div data-role="footer">
                <h4>PIE DE PÁGINA</h4>
        </div>
</div>
```



Transiciones

Al pasar de una página a otra –ya sea desde un mismo documento jQuery Mobile o entre varios–, se produce una transición automática que desliza la página hacia adelante o hacia atrás, arriba o abajo, con un determinado efecto, típico de los dispositivos móviles. Estas transiciones o animaciones pueden cambiar su comportamiento de acuerdo al atributo **"data-transition="""**.

Tipos de transiciones

- **Slide**: es la transición por defecto. Este efecto desliza la página hacia la izquierda.
- **Slideup**: una nueva página aparece desde abajo hasta ocupar toda la pantalla.

- **Slidedown**: una nueva página aparece desde arriba hasta ocupar toda la pantalla.
- **Pop**: una nueva página aparece desde la que fue llamada, como si fuera un zoom que va desde el centro hacia afuera.
- **Fade**: una nueva página aparece sobre la página actual mostrando un efecto de disolvencia.
- **Flip**: hace un efecto 3D como si se diera vuelta la hoja, simulando un giro de 180°, y muestra la nueva página. jQuery Mobile aclara en su página web que esta transición no funciona correctamente en algunos dispositivos (como en la mayoría de las versiones del sistema operativo Android).

Veamos un ejemplo: el siguiente código muestra cómo aplicar un efecto de transición cuando llamamos a la **pagina2.html**.

```
<div data-role="page" id="pagina1">
        <div data-role="header">
                <h1>TÍTULO</h1>
        </div>
        <div data-role="content">
                <h2>CONTENIDO<h2>
            <a href="pagina2.html" data-transition="pop">Página 2</a>
        </div>
        <div data-role="footer">
                <h4>PIE DE PÁGINA</h4>
        </div>
</div>
```

RESUMEN

En este segundo capítulo empezamos explicando cómo descargar el framework de jQuery Mobile e instanciar las librerías en un archivo HTML5. Luego aprendimos a organizar nuestros proyectos, administrando los sitios desde Dreamweaver CC. Conocimos los componentes básicos de una página jQuery Mobile y explicamos cómo usar el atributo **data-role="page"** para crear las diferentes páginas de nuestra aplicación y los atributos que nos permiten estructurar la página: **data-role="header"**, **data-role="content"** y **data-role="footer"**. Finalmente, aprendimos a realizar vínculos internos y externos.

Actividades

TEST DE AUTOEVALUACIÓN

1. ¿Cuáles son las ventajas de trabajar por CDN?
2. ¿Cuáles son las distintas opciones que nos da el framework de jQuery Mobile para dejar fijo un encabezado o un pie de página?
3. ¿Qué es la sección de contenido?
4. ¿Cuál es la forma correcta de llamar a una página dentro de un mismo documento o archivo?
5. Enumere los tipos de transiciones.

EJERCICIOS PRÁCTICOS

1. Abra el editor con el cual eligió trabajar y haga el llamado a las librerías de jQuery Mobile por CDN.
2. Desarrolle una página básica.
3. A la página anterior, agréguele dos páginas más dentro del mismo documento y, como identificador, llámelas "**pagina1**", "**pagina2**" y "**pagina3**", respectivamente.
4. En la sección de contenido de la "**pagina1**", haga un enlace a la "**pagina2**" y otro a la "**pagina3**".
5. En los enlaces de la "**pagina1**", agregue una transición **fade** al link que abre la "**pagina2**" y una transición **slideup** a la que abre la "**pagina3**".



PROFESOR EN LÍNEA

Si tiene alguna consulta técnica relacionada con el contenido, puede contactarse con nuestros expertos: profesor@redusers.com

*03

Botones y listas

A partir de la página base que aprendimos a desarrollar en el Capítulo 2, veremos ahora cómo crearle una interfaz gráfica, empleando los componentes que nos ofrece el framework de jQuery Mobile. Veremos cómo crear y configurar botones, cómo enlazarlos, cambiar el diseño y agruparlos. También explicaremos cómo organizar la información de nuestra aplicación utilizando listas. Analizaremos cada uno de los distintos tipos de listas y sus diversos usos, y aprenderemos a crearlas empleando las características de jQuery Mobile.

- Botones 62
- Listas 69
- Navbar 84
- Resumen 89
- Actividades 90

↗↗↗

Botones

Crear botones con jQuery Mobile es muy simple: se pueden alinear, agregarles iconos o agruparlos a partir de atributos que vienen predeterminados en el framework.

El atributo **data-role** es el encargado de crear el botón pasándole el valor **"button"**. Y, también, contamos con muchas posibilidades en lo que se refiere al diseño.

Para crear un botón, le agregamos el atributo **data-role="button"** a un enlace HTML, como mostramos a continuación:

```
<a href="#" data-role="button">Mi botón</a>
```

Como podemos ver, el **botón de enlace** (*link button*) o –simplemente– **botón** no es ni más ni menos que un simple botón que al hacer clic nos lleva a otra página o a una **dirección URL** (*Uniform Resource Locator*) externa.

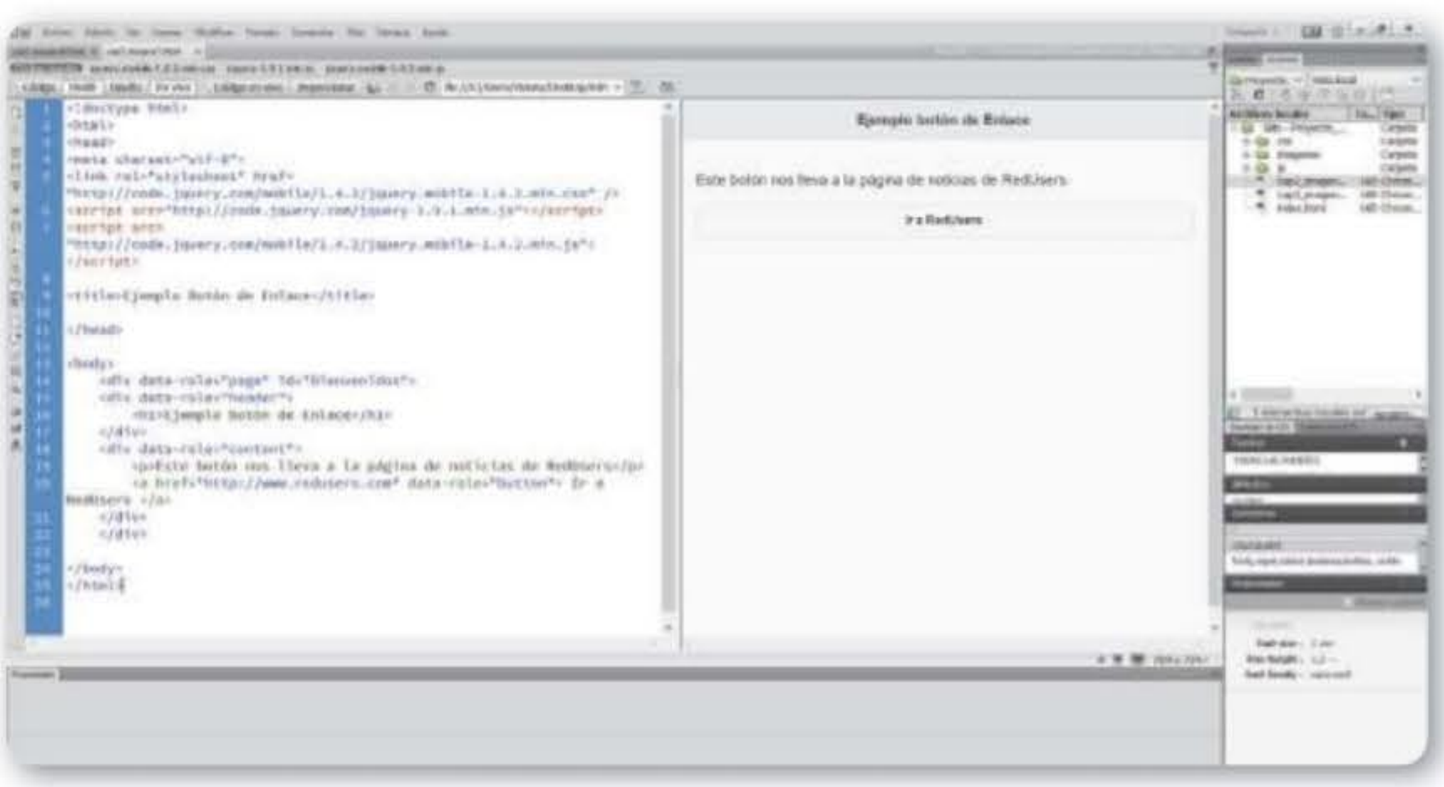


Figura 1. Debemos escribir la dirección URL entre comillas después del atributo **href** de HTML.

Características

Existen una gran cantidad de atributos y clases de jQuery que nos permiten configurar los botones para que tengan la apariencia deseada.

Por ejemplo, si quisiéramos deshabilitar el botón, esto se hace posible agregando la clase **ui-disabled**.

```
<a href="#" data-role="button" class="ui-disabled">Mi botón</a>
```

Para hacer un botón más chico o con dimensiones reducidas, hay que agregar el atributo **data-mini** que acepta dos valores: **true** o **false** (verdadero o falso). Este tipo de botón nos puede servir para espacios reducidos o en donde tenemos mucho contenido, ya que su versión compacta no es tan grande como la versión estándar, y la letra es mucho más pequeña.

```
<a href="#" data-role="button" data-mini="true">Mi botón</a>
```

Los botones tienen un **estilo** creado por CSS. Estos estilos se pueden modificar; por ejemplo, es posible aplicar una gama de colores, como explicaremos en el **Capítulo 6** de este libro.

Iconos

Para resaltar el diseño de los botones, les podemos agregar iconos.

En lo que se refiere a botones con imágenes o con iconos, jQuery Mobile nos ofrece un abanico de posibilidades ya diseñadas por el framework para poder agregarlos.

Este conjunto de iconos es el más utilizado en las aplicaciones móviles ya que, al estar incluido en el framework, nos permite reducir la carga de la página. También podemos incorporar otros iconos si lo creemos necesario. Para poder utilizarlos, tenemos que agregar el atributo **data-icon** a nuestro botón y así obtendremos un botón con imagen.

Por ejemplo, si quisiéramos agregar un icono de borrar, tendríamos que hacer lo siguiente:

POO-CLASES

La **programación orientada a objetos** (**POO**) es un concepto que se basa en la existencia de objetos. Los **objetos** representan cosas, se identifican por sus atributos y sus métodos. A su vez, los objetos se agrupan en **clases** que representan un tipo particular de objetos. Por ejemplo, la clase "aviones" estaría representada por todos los aviones. Cada clase tiene un conjunto de propiedades y métodos comunes a un conjunto de objetos.

```
<a href="#" data-role="button" data-icon="delete">Botón de borrar</a>
```

Y lo mismo si tuviéramos un minibotón:

```
<a href="#" data-role="button" data-mini="true" data-icon="delete">Botón mini
    de borrar</a>
```

Conjunto de iconos o valores para el atributo data-icon

Como mencionamos antes, jQuery Mobile nos ofrece una gran variedad de iconos que podemos utilizar en nuestro botones, como mostramos en la siguiente tabla:

ICONOS DISPONIBLES EN JQUERY MOBILE

▼ ICONO POR MOSTRAR	▼ ATRIBUTO
Flecha a la izquierda	data-icon="arrow-l"
Flecha a la derecha	data-icon="arrow-r"
Flecha arriba	data-icon="arrow-u"
Flecha abajo	data-icon="arrow-d"
Borrar	data-icon="delete"
Más	data-icon="plus"
Menos	data-icon="minus"
Chequeado	data-icon="check"
Configuración	data-icon="gear"
Actualizar	data-icon="refresh"
Adelante	data-icon="forward"
Atrás/Volver	data-icon="back"
Grilla/Tabla de datos	data-icon="grid"
Estrella	data-icon="star"
Alerta/Notificación	data-icon="alert"
Información	data-icon="info"
Inicio	data-icon="home"
Buscar	data-icon="search"

Tabla 1. Valores que puede tomar el atributo **data-icon**.



Figura 2. Por defecto, jQuery Mobile redondea automáticamente los bordes de los botones.

Iconos propios

Posiblemente algunas veces necesitemos agregar iconos personalizados (es decir, que no estén dentro del conjunto de iconos que trae el framework). Para esto, lo que tenemos que hacer es agregar, al comienzo de la página, una etiqueta de estilo **<style></style>** y cambiar la clase **.ui-icon-** ya definida en las hojas de estilo que vienen con jQuery Mobile. Podemos observarlo en el siguiente ejemplo:

```
<style type="text/css">
        .ui-icon-miIcono {
                background: url("1.png") no-repeat rgba(0, 0, 0, 0.4) !important;
}
</style>
```

Luego, para que aparezca en el botón, debemos llamar a la imagen de la siguiente forma:

```
<a href="#" data-role="button" data-icon="miIcono">Botón</a>
```

Posición del icono

Por defecto, la posición del icono es a la izquierda del botón, pero con el atributo **data-iconpos** podemos cambiar su posición, ubicarlo a la derecha, a la izquierda, arriba o abajo, o incluso dejar solo el icono sin texto.

Para ubicar el icono **a la derecha** del texto, le agregamos al atributo **data-iconpos** el valor **"right"** como nos muestra la siguiente línea de código:

```
<a href="#" data-role="button" data-icon="star" data-iconpos="right">Mi botón</a>
```

Si queremos ubicar el icono **arriba del texto**, empleamos el valor **"top"**:

```
<a href="#" data-role="button" data-icon="star" data-iconpos="top">Mi botón</a>
```

En cambio, si queremos ubicar el icono **debajo del texto**, usamos el valor **"botom"**:

```
<a href="#" data-role="button" data-icon="star" data-iconpos="botom">Mi botón</a>
```

Si deseamos que nuestro botón solo tenga un icono, le asignamos al atributo **data-iconpos** el valor **"notext"**:

```
<a href="#" data-role="button" data-icon="star" data-iconpos="notext">Mi botón</a>
```

Botones con un ancho específico

Por defecto, los botones ocupan todo el ancho de la pantalla; para que sean más chicos o se limiten al ancho del texto o del icono, existe el atributo **data-inline**, que acepta los valores **true** o **false** (verdadero o falso).

La siguiente línea de código nos muestra cómo ajustar el tamaño del botón en relación con el texto del botón:

```
<a href="#" data-role="button" data-icon="star" data-inline="true">Mi botón</a>
```

Botones agrupados

En ciertas ocasiones, necesitamos tener agrupados varios botones del mismo tipo. Para crear grupos, se utiliza la etiqueta de HTML ** ** con el atributo **data-role="controlgroup"**: esto creará un marco contenedor, sin espacios y sin sombras entre los botones.

Por defecto, el framework creará un marco donde agrupará los botones en forma vertical, pero esto se puede modificar agregándole

al atributo **data-role="controlgroup"** el atributo **data-type**. Este atributo puede recibir el valor **horizontal** o **vertical**.

Por ejemplo, si queremos agrupar tres botones en forma vertical, escribiremos las siguientes líneas de código:

```
<div data-role="controlgroup">
<a href="#" data-role="button">Botón 1</a>
<a href="#" data-role="button"> Botón 2</a>
<a href="#" data-role="button"> Botón 3</a>
</div>
```

> POR DEFECTO, LOS BOTONES SE AGRUPAN DE FORMA VERTICAL, PERO ESTO PUEDE MODIFICARSE

En cambio, si queremos agrupar los botones en forma horizontal, le tendremos que asignar el valor **"horizontal"** al atributo **data-type**, como vemos a continuación:

```
<div data-role="controlgroup" data-type="horizontal">
<a href="#" data-role="button" >Botón 1</a>
<a href="#" data-role="button" > Botón 2</a>
<a href="#" data-role="button" > Botón 3</a>
</div>
```

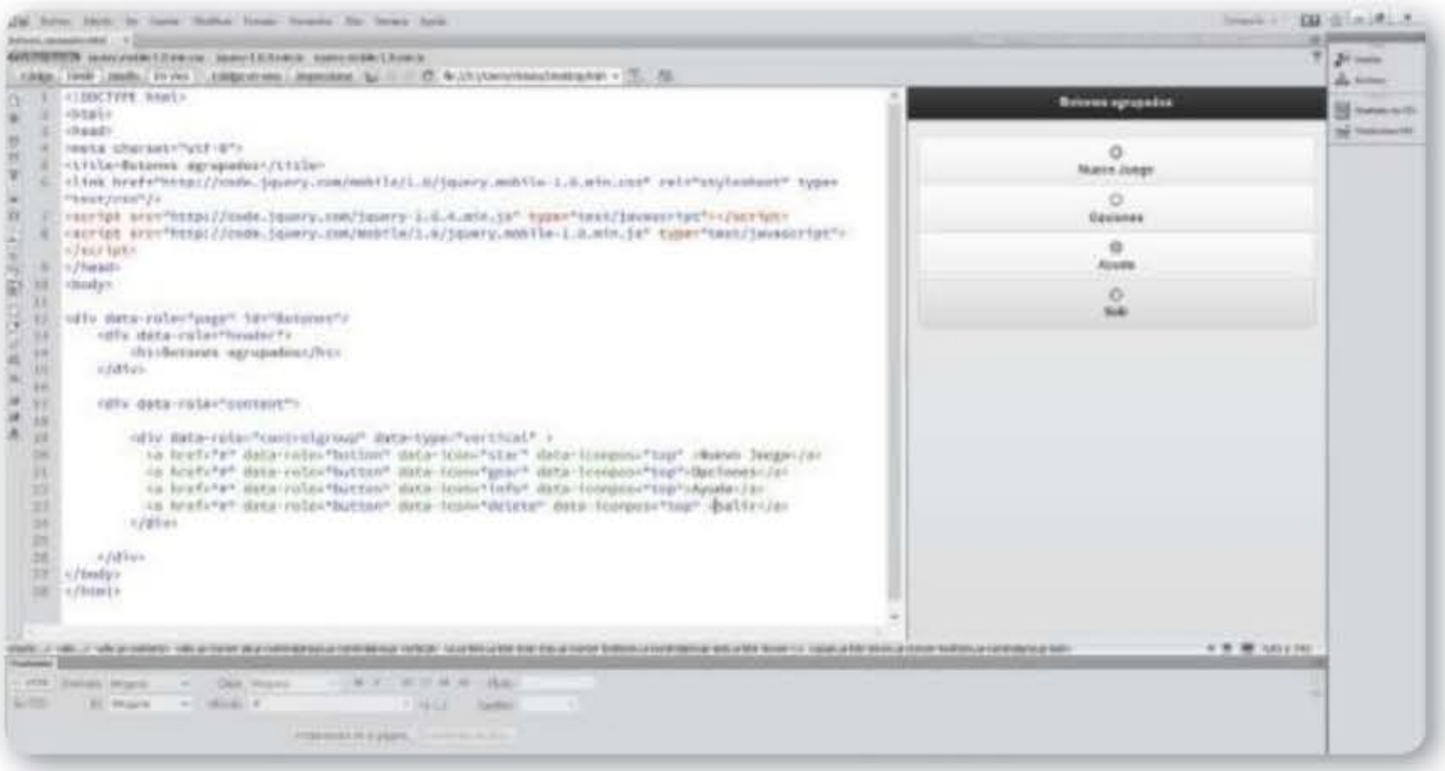


Figura 3. Cuando agrupamos botones, podemos emplear también el atributo **data-iconpos**.

El botón "volver"

Una de las funciones más importantes tanto en la Web como en un dispositivo móvil es siempre tener visible en nuestra aplicación el botón de **volver atrás**, que nos permite volver a la página anterior del historial de navegación. El framework jQuery Mobile no está exento de esta regla –no escrita pero imprescindible– que viene desactivada por defecto. Para activarla, tenemos que escribir el atributo **data-add-back-btn** y pasarle como valor **true**, siempre dentro de los **div** de la cabecera de nuestra página.

> ES MUY IMPORTANTE QUE EL BOTÓN DE VOLVER ATRÁS ESTÉ DISPONIBLE EN NUESTRA APLICACIÓN

Nuestro código quedaría de la siguiente forma:

```
<div data-role="page" data-add-back-btn="true"></div>
```

Comandos para utilizar funciones del teléfono

Existen diferentes comandos o funciones ya definidas por jQuery Mobile, que nos permiten utilizar algunas funciones básicas de nuestro teléfono a través de un botón de enlace.

Además, todos los enlaces estándares de HTML son soportados por el framework, como es el caso del comando **mailto**.

Por ejemplo, si queremos agregar un enlace para llamar por teléfono, tendríamos que escribir el siguiente código:

```
<a href="tel:11111111" data-role="button">Llamar por teléfono</a>
```

MAILTO

El comando **mailto** en HTML es un enlace que nos permite enviar un e-mail desde una página web. Lo que hace es abrir nuestra casilla de correo, escribir el mensaje y el remitente, entre otras cosas. Es un comando que actúa del lado del cliente, es decir, siempre abrirá el gestor de correo que el usuario tenga definido.

También podemos incluir en nuestra aplicación un enlace que nos permita enviar mensajes de texto (o **SMS**, acrónimo de *Short Message Service*, 'servicio de mensajes cortos'). Para ello, emplearíamos el siguiente código:

```
<a href="sms:llllllll" data-role="button">Mandar mensaje de texto</a>
```

Y, por último, el código para enviar un correo electrónico sería:

```
<a href="mailto:miEmail@correo.com" data-role="button">Mandar un email</a>
```



Listas

Una **lista**, en programación, es un conjunto de elementos del mismo tipo ordenados, que puede contener cualquier cosa que necesitemos mostrar.

En las aplicaciones desarrolladas para dispositivos móviles, las listas sirven para organizar el contenido, y en jQuery Mobile disponemos de un conjunto de listas adecuadas a cada necesidad que requiera nuestra aplicación. El framework nos permite crear listas **simples** y listas **interactivas**. Las listas interactivas se diferencian de las listas simples en que las primeras poseen un enlace y son las más empleadas en aplicaciones web.

También, podemos crear listas con iconos, numeradas, con imágenes, y muchas opciones más.

Lista interactiva

Para crear una lista interactiva básica con un enlace, solo tenemos que agregar el atributo **data-role=listview** dentro de la etiqueta de apertura de la lista **ul**.

Este tipo de lista estándar ocupa casi todo el ancho de la pantalla con un icono de flecha a la derecha, que al hacer clic muestra el contenido de la lista con una transición; ese contenido no es ni más ni menos que otra página interna o externa jQuery Mobile.

La sintaxis que define una lista es la siguiente:

```
</div>
        <ul data-role="listview">
      <li><a href="#locales">Locales</a></li>
      <li><a href="#delivery">Delivery</a></li>
      <li><a href="#productos">Productos</a></li>
    </ul>
  </div>
```

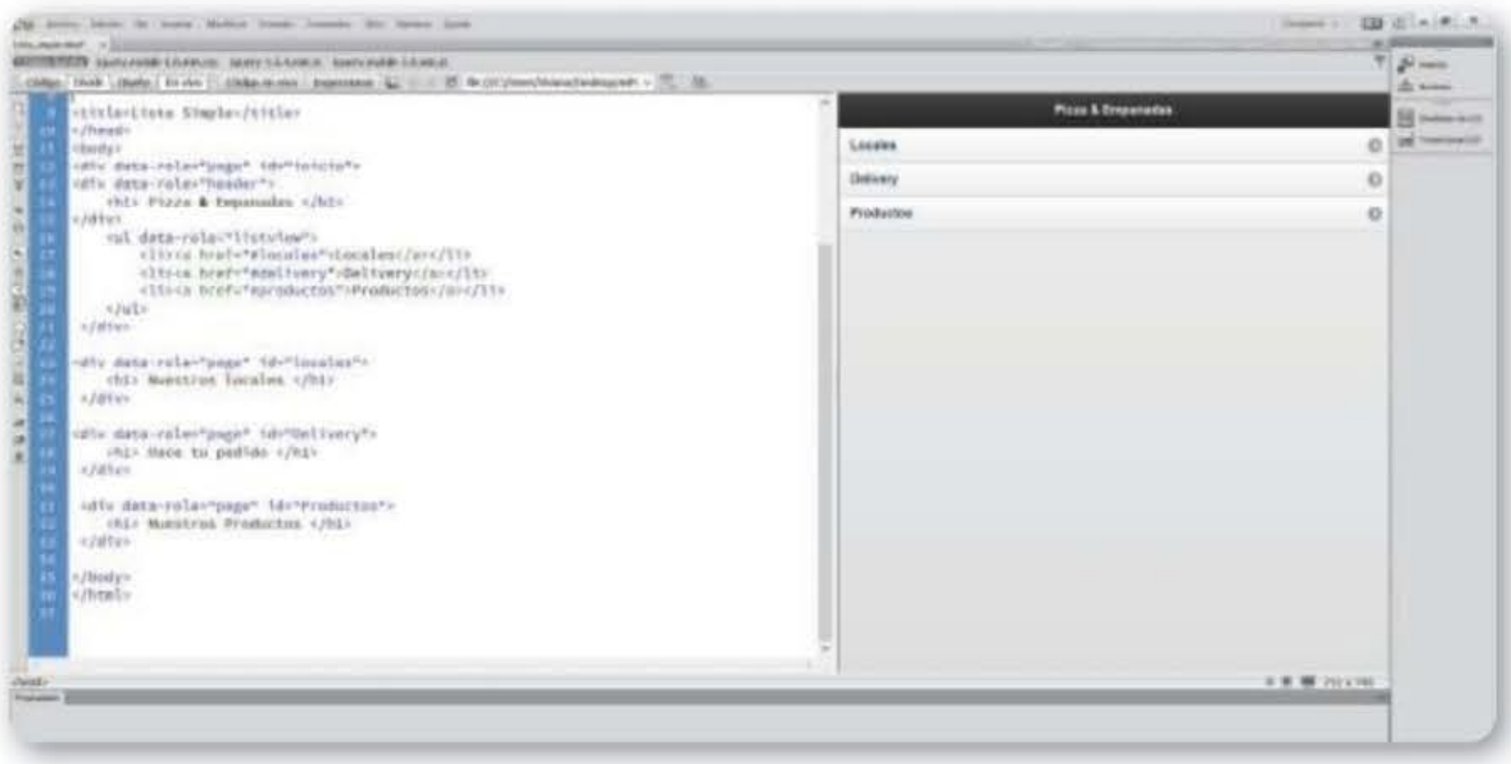


Figura 4. Las listas en pantalla completa (fullscreen) no permiten tener contenido alrededor.

También podemos hacer que nuestra lista se vea con un poco de margen y más redondeada en las puntas. Por defecto, una lista abarca todo el ancho de la pantalla; si, en cambio, queremos que la lista no ocupe todo el ancho de la pantalla, debemos incluir el atributo **data-inset** y asignarle el valor **true**. De esta forma, la lista se verá centrada.

El siguiente código nos muestra cómo usar el atributo **data-inset**:

```
<ul data-role="listview" data-inset="true">
      <li><a href="#locales">Locales</a></li>
      <li><a href="#delivery">Delivery</a></li>
      <li><a href="#productos">Productos</a></li>
</ul>
```

Listas interactivas anidadas

Las **listas anidadas** son aquellas que contienen sublistas: así, cuando seleccionamos un elemento de la lista, nos muestra otra lista y, a su vez, cada elemento de esta sublista puede contener otra lista.

Para anidar una lista debemos escribir la nueva lista dentro de la etiqueta **<li>** de los elementos de la lista principal.

Veamos un ejemplo: en una heladería tenemos un listado de sus productos, los que están clasificados por tipo: helados de agua, helados de crema, helados para celíacos. A su vez, cada tipo de producto contiene una lista de sabores.

El siguiente código nos muestra cómo crear la lista anidada del ejemplo anterior:

```
<ul data-role="listview" data-inset="true">
   <li><h3>Helados de Crema</h3>
      <ul>
        <li><a href="#">Chocolate</a></li>
        <li><a href="#">Dulce de Leche</a></li>
        <li><a href="#">Americana</a></li>
      </ul></li>
   <li><h3>Helados de Agua</h3>
      <ul>
        <li><a href="#">Frutilla</a></li>
        <li><a href="#">Limón</a></li>
        <li><a href="#">Ananá</a></li>
      </ul></li>
   <li><h3>Helados para Celíacos</h3>
      <ul>
        <li><a href="#">Chocolate</a></li>
        <li><a href="#">Dulce de Leche</a></li>
        <li><a href="#">Vainilla</a></li>
      </ul></li>
</ul>
```

Como podemos observar, los elementos principales de la lista que creamos no poseen enlaces, pero esto no implica que no se comporten como una lista interactiva.

Las listas anidadas en jQuery Mobile nos generan automáticamente múltiples páginas sin que las tengamos que crear nosotros mediante el atributo **data-role="page"**.

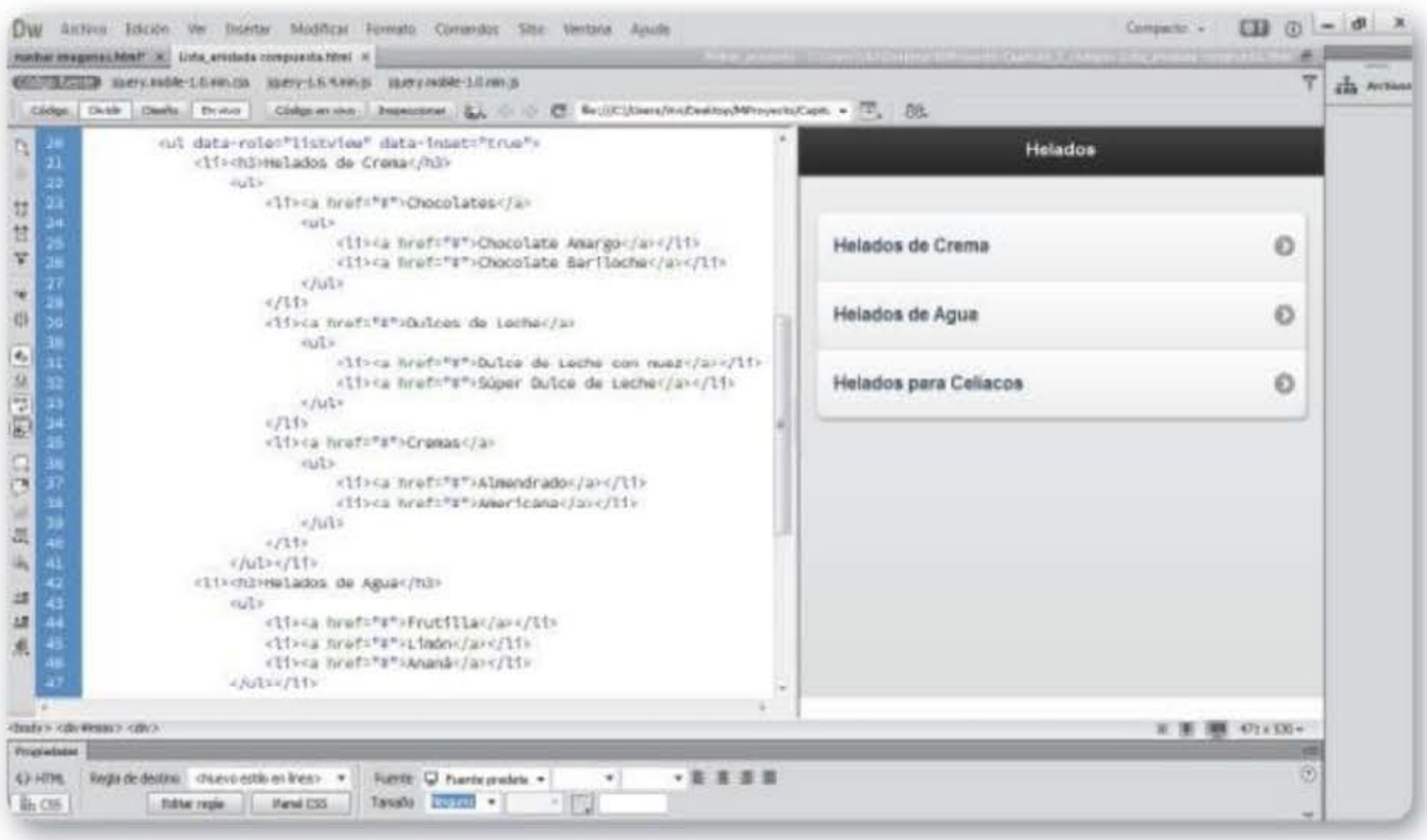


Figura 5. Podemos tener varios niveles en una lista anidada; por ejemplo, dos niveles, como se observa en esta imagen.

Lista numerada

Si, en lugar de crear una lista interactiva, queremos crear una lista simple, ya sea para ordenar una lista o simplemente numerarla visualmente, lo que tenemos que hacer es agregarle la etiqueta HTML **
** en vez de la etiqueta **
**; esto incorporará un número de orden a la lista.

El siguiente código nos muestra un ejemplo de cómo crear una lista numerada con la etiqueta HTML **
**:

ENLACES HTML

Los enlaces en HTML se crean mediante la etiqueta **<a>**. Su atributo más importante es **href**; este se emplea para indicar la URL a la que apunta el enlace. Cuando hacemos clic sobre un enlace, el navegador se dirige a la URL del recurso indicado mediante **href**. Las URL de los enlaces pueden ser absolutas, relativas, internas y externas.

```
<div data-role="content">
    <ol data-role="listview" data-inset="true" >

        <li>Christian</li>
        <li>José</li>
        <li>Lorena</li>
        <li>Leandro</li>
        <li>Luis</li>
        <li>Marcelo</li>
        <li>Rosalba</li>
        <li>Sandra</li>
        <li>Susana</li>
        <li>Viviana</li>

    </ol>
</div>
```



Figura 6. Las listas simples son llamadas también **listas de solo lectura**, ya que se emplean para presentar solamente contenido.

TÍTULOS Y PÁRRAFOS EN UNA LISTA

Dentro de un elemento de la lista, podemos incluir un título y algún párrafo relacionado con el elemento. Para ello, debemos incluir dentro del link del elemento de lista las etiquetas de títulos (headers) y las etiquetas de párrafos **(<pl></p)**.

Filtrar elementos y agregar placeholder

Por lo general, cuando contamos con muchos elementos en una lista, se hace muy tedioso tener que encontrar un elemento en particular, ya que deberíamos recorrer y buscar por toda la lista ese ítem, lo que nos llevaría mucho tiempo. Para que esto no ocurra, necesitamos tener un **filtro** o **buscador** que nos permita escribir el elemento por encontrar y nos lo muestre, sin la necesidad de recorrer uno a uno todos los ítems de la lista.

Para esto, jQuery Mobile nos ofrece una solución bastante sencilla: solo tenemos que agregar el atributo **data-filter = "true"** y obtendremos, por encima de la lista, un cuadro de texto que nos va a permitir ingresar la palabra o las iniciales de la palabra que buscamos dentro de la lista.

Por defecto, el **placeholder** (marcador de posición) de la caja de texto del **data-filter** dice: "**Filter ítems...**" lo que se podría traducir por 'Buscar por ítems...'. Para cambiar el placeholder solo tenemos que agregarle el atributo –**placeholder** al atributo **data-filter**, que recibirá por valor el texto que necesitemos mostrar.

A continuación, veremos un ejemplo de cómo usar un filtro. Siguiendo con el ejemplo de la heladería, incluimos un filtro que nos permite buscar un determinado sabor dentro de una lista de helados. Para lograr esto, escribimos el siguiente código:

```
<div data-role="content">

<ul data-role="listview" data-inset="true" data-filter="true" data-filter-
placeholder="¿Qué sabor estás buscando?">

  <li><h3>Helados de Crema</h3>
```

PLACEHOLDER

Placeholder se puede traducir al español como 'marcador de posición'. El placeholder nos sirve para indicarle al usuario, en un campo de texto, una sugerencia sobre lo que queremos que escriba en él. Esa sugerencia o ayuda aparecerá dentro del campo de texto en forma de marca de agua. De esta forma, se ahorra espacio, y, por eso, es recomendable para teléfonos celulares.

```
    <ul>
      <li><a href="#">Chocolate</a></li>
      <li><a href="#">Dulce de Leche</a></li>
      <li><a href="#">Crema Americana</a></li>
    </ul></li>
  <li><h3>Helados de Agua</h3>
    <ul>
      <li><a href="#">Frutilla</a></li>
      <li><a href="#">Limón</a></li>
      <li><a href="#">Ananá</a></li>
    </ul></li>
  <li><h3>Helados para Celíacos</h3>
    <ul>
      <li><a href="#">Chocolate</a></li>
      <li><a href="#">Dulce de Leche</a></li>
      <li><a href="#">Vainilla</a></li>
    </ul></li>

</ul>
</div>
```



Figura 7. Este buscador funciona de manera automática para todos los tipos de listas de jQuery Mobile.

Dividir y autodividir

jQuery Mobile nos da la opción de agrupar una lista por ítems del mismo tipo; a esto se lo llama **dividir** y **autodividir** un **listview**. A continuación, veremos de qué se tratan estas funciones.

Dividir

En algunas ocasiones, tenemos la necesidad no solo de ordenar una lista, sino también de agruparla o dividirla para que visualmente mantenga informado al usuario en cuanto a su contenido. Estos divisores no responden a ninguna acción o evento del usuario, sino que son más bien informativos y podemos llegar a usarlos para agrupar contenido del mismo tipo.

Pasándole el valor **list-divider** al atributo **data-role** dentro de las etiquetas **
**, obtenemos una división que nos permite clasificar el contenido.

En el siguiente código, usamos los separadores de listas para dividir los diferentes tipos de productos que comercializa la heladería: helados y postres.

```
<ul data-role="listview" data-inset="true">
   <li data-role="list-divider">Helados</li>
      <li><a href="#">Cremas</a></li>
      <li><a href="#">Dulces de Leche</a></li>
      <li><a href="#">Chocolates</a></li>
   <li data-role="list-divider">Postres</li>
      <li><a href="#">Barras</a></li>
      <li><a href="#">Tortas</a></li>
</ul>
```



Figura 8. Por defecto, jQuery Mobile le asigna un estilo al ítem que actúa como divisor.

Autodividir

jQuery Mobile nos permite crear de forma automática separadores de lista, usando la primera inicial de cada ítem.

Para dividir o agrupar en forma automática una lista alfabéticamente y que marque las divisiones con la primera letra de los ítems, tenemos que usar el atributo **data-autodividers="true"**.

Un ejemplo típico es una agenda, donde tenemos un listado de personas, a las que podemos agrupar por la inicial de cada nombre, para obtener un listado alfabético, de manera tal que nos resulte más fácil localizar cada uno de ellos.

```
<ul data-role="listview" data-autodividers="true" >

        <li>Agostinetti Melisa</li>
                <li>Amaro Lorena</li>
        <li>Ballesteros Christian</li>
        <li>Bosi Gabriel</li>
        <li>Cianciarullo Rosalba</li>
        <li>Contreras María</li>
        <li>Di Bertoni Mónica</li>
        <li>Heredia Viviana</li>
        <li>Hereter Leandro</li>
        <li>Lovecchio Susana</li>
        <li>Pezzimenti Claudia</li>
        <li>Recalde Damián</li>
        <li>Sauco Ricardo</li>
        <li>Zanini Sandra</li>
        <li>Zanoni Patricia</li>
</ul>
```



COMENTARIOS

A veces es muy útil anotar comentarios en el documento HTML sobre el código que escribimos, ya que nos pueden servir para aclarar o recordar posteriormente qué hicimos. Para escribir comentarios, debemos encerrar el comentario entre los símbolos **<!–** y **–>**. Todo lo que encerremos entre estos dos símbolos no se visualizará en pantalla.

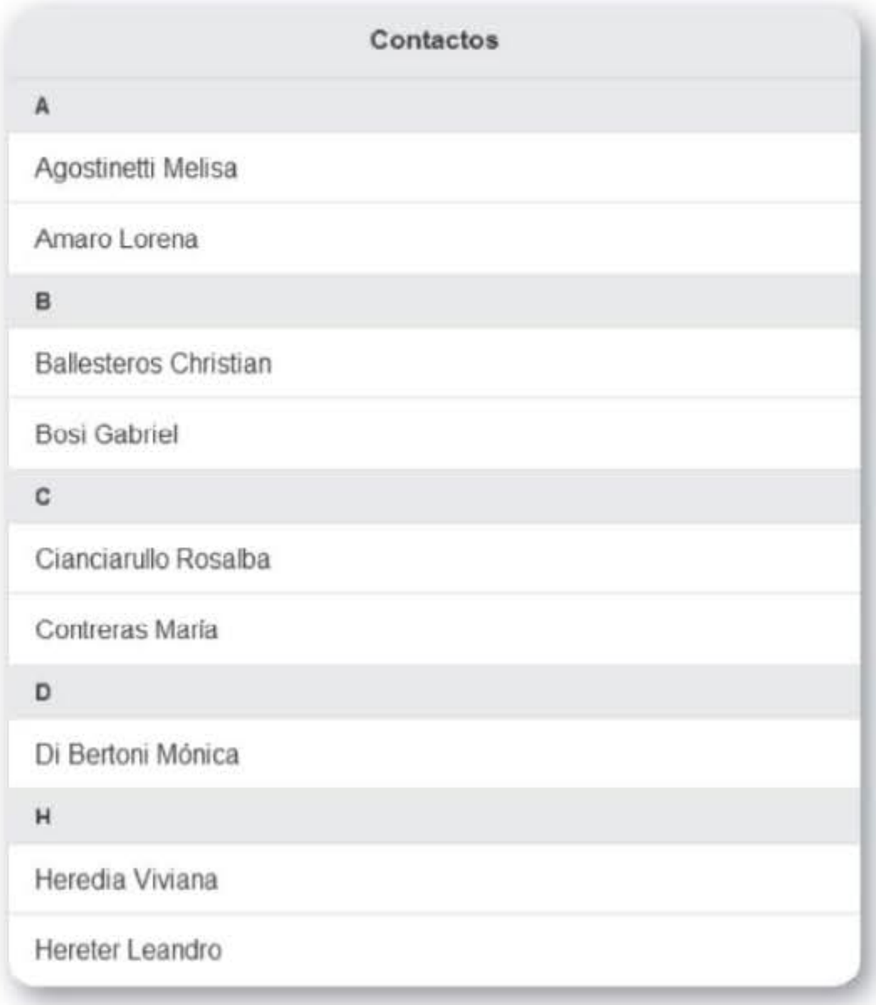


Figura 9. Lista de contactos ordenada alfabéticamente, donde cada letra hace de divisor.

Contador burbuja

Un **contador burbuja** o **count bubbles** es una utilidad del framework, que nos sirve para informarle al usuario la cantidad de ítems que se encuentran dentro de ese elemento de la lista. Esta información se representa por un número dentro de una burbuja o círculo en el extremo derecho de la lista.

Si bien el número que se escribe es estático, usando un poco de programación y conocimientos de jQuery podríamos hacerlo dinámico; así, si el contenido cambia, el número de la burbuja también cambiaría.

Para que cada elemento de la lista nos muestre en su extremo derecho un número, tenemos que agregar la clase **ui-li-count** dentro de las etiquetas **
, en el enlace **href dentro de las etiquetas ****, y colocar manualmente el número por mostrar.

Por ejemplo, tenemos un listado de productos clasificados por tipos, en este caso helados; mostraremos por cada tipo de helado un número que nos indicará cuántos sabores hay por cada uno.

```
<ul data-role="listview" data-inset="true">

    <li><a href="#">Helados de Crema<span class="ui-li-count">25</span></
a></li>
    <li><a href="#">Helados de Agua<span class="ui-li-count">15</span></
```

```
a></li>
        <li><a href="#">Postres Helados<span class="ui-li-count">7</span></a></
li>

    </ul>
```



Figura 10. La etiqueta **** se utiliza para agrupar elementos o añadirles estilo (CSS).

Agregar imágenes-iconos

Para darle un mayor impacto visual a nuestra página, podemos agregar imágenes en miniatura (**thumbnails**) o iconos a los elementos de nuestras listas.

Los **formatos de imagen** que normalmente se utilizan en aplicaciones web son:

- **JPG** o **JPEG** (*Joint Photographic Experts Group*)
- **GIF** (*Graphics Interchange Format*)
- **PNG** (*Portable Network Graphics*)



CELDA DIVIDIDA

Si incluimos dos enlaces dentro de un elemento de la lista, es decir, entre las etiquetas **
**, veremos que la fila del elemento de la lista queda dividida en dos celdas, en donde el primer enlace actuará sobre la celda de la izquierda, y el segundo, en la celda de la derecha.

Recordemos que las imágenes con formato GIF pueden contener hasta 256 colores, y este es el formato ideal para guardar imágenes pequeñas como, por ejemplo, iconos. El formato JPG usa hasta 16 millones de colores; por este motivo, es ideal para imágenes de alta resolución (como, por ejemplo, fotografías); el formato PNG puede contener hasta 281 trillones de colores, que suelen emplearse tanto para iconos como para gráficos.

Agregar imágenes a las listas

El tamaño de las imágenes deben ser de 80 x 80 pixeles; si incluimos una imagen de mayor tamaño, jQuery Mobile automáticamente reducirá su tamaño a un cuadrado de 80 pixeles.

Tengamos en cuenta que, si la imagen no es cuadrada, al cambiar el tamaño quedará desproporcionada.

Para agregar imágenes en los elementos de una lista, debemos incluir dentro del elemento **<li>** la etiqueta **<img>** (*image*) y el atributo **src** (*source*), para indicar la ruta donde se halla nuestra carpeta con las imágenes. La etiqueta **<img>** no tiene cierre.

Las siguientes líneas de código nos muestran cómo agregar imágenes en una lista. En este ejemplo, las imágenes se encuentran en la carpeta **Imágenes**, que se incluyen en la carpeta de nuestro proyecto.

```
<div data-role="page" id="inicio">
   <div data-role="header">
      <h1> Propiedades de las Frutas </h1>
   </div>
   <div data-role="content">
      <ul data-role="listview">
         <li><a href="#banana">
```

EDITAR IMÁGENES

Podemos usar un programa de edición de imagen propietario, como **Photoshop**, **Fireworks,** o gratuito y online, como **Photoshop Express** (**www.photoshop.com/tools?wf=editor**), para convertir archivos de gráficos de un formato a otro y elegir el grado de compresión que deseamos aplicar a las imágenes.

```
            <img src="/imagenes/banana.png">
            <h2> Banana </h2>
                        <li><a href="#durazno">
            <img src="/imagenes/durazno.png">
            <h2> Durazno </h2>
                        </a></li>
        <li><a href="#frutilla">
            <img src="/imagenes/frutilla.png">
            <h2> Frutilla </h2>
                        </a></li>
        <li><a href="#kiwi">
            <img src="/imagenes/kiwi.png">
            <h2> Kiwi </h2>
                        </a></li>
        </a></li>
        <li><a href="#limon">
            <img src="/imagenes/limones.png">
            <h2> Limón </h2>
                        </a></li>
        </ul>
    </div>
    <div data-role="footer">
     <h1> Coma Natural </h1>
    </div>
```



Figura 11. Al añadir una imagen en un elemento de la lista, el alto de la fila se agrandará.

Usar imágenes como iconos en las listas

Podemos utilizar también imágenes como iconos; para ello, el tamaño de estas debe ser de 16 x 16 pixeles.

Para agregar imágenes como iconos, además de pasarle la ruta de la carpeta en donde se encuentran las imágenes, tendremos que agregar la clase que nos provee jQuery Mobile **ui-li-icon**, como se muestra en el siguiente código:

```
<ul data-role="listview">
   <li><a href="#">
      <img src="/imagenes/iconos/banana.png" class="ui-li-icon">
      <h2> Banana </h2>
   <li><a href="#">
      <img src="/imagenes/iconos/durazno.png" class="ui-li-icon">
      <h2> Durazno </h2>
   </li></a>
   <li><a href="#">
      <img src="/imagenes/iconos/frutilla.png"class="ui-li-icon" >
      <h2> Frutilla </h2>
   </a></li>
   <li><a href="#">
      <img src="/imagenes/iconos/kiwi.png"class="ui-li-icon" >
      <h2> Kiwi </h2>
   </a></li>
   </a></li>
   <li><a href="#">
      <img src="/imagenes/iconos/limones.png"class="ui-li-icon" >
      <h2> Limón </h2>
   </a></li>
</ul>
```

NO COLOCAR ICONOS EN LAS LISTAS

Por defecto, las listas no interactivas no traen iconos, pero, para tener la certeza de que no va a colocar ninguno, al atributo **data-icon** le daríamos el valor **false, (data-icon="false")**. De esta manera, nos aseguraríamos de que no va a mostrar ninguno en ese ítem en particular.

En este ejemplo, las imágenes se encuentran dentro de la subcarpeta **Iconos** de la carpeta **Imágenes** del proyecto.



Figura 12. Cuando usemos imágenes como iconos, estos se ubicarán a la derecha de la lista.

Usar los iconos del framework en las listas

Por defecto, como estuvimos viendo, las listas interactivas traen a la derecha un icono con forma de flecha. Sin embargo, podemos cambiarlo por alguno de los iconos disponibles en el framework, que vimos en este capítulo al hablar de botones.

Para sustituir el icono de la flecha, incluimos dentro del elemento **<li>** el atributo **data-icon**, como podemos ver en el siguiente código:

```
<ul data-role="listview">

   <li data-icon="star">
      <a href="#">Nuevo juego</a></li>
   <li data-icon="gear">
   <a href="#">Opciones</a></li>
   <li data-icon="grid">
      <a href="#">Estadísticas</a></li>
   <li data-icon="info">
      <a href="#">Ayuda</a></li>
   <li data-icon="delete">
   <a href="#">Salir</a></li>

</ul>
```



Figura 13. Empleamos un icono de la librería de jQuery Mobile para sustituir la clásica flecha.

Navbar

Otro componente muy útil que podemos utilizar en nuestra aplicación es la **barra de navegación** (**navbar**), ya que permite que el usuario pueda desplazarse por cada página que compone la aplicación.

Lo ideal es ubicar la barra de navegación ya sea en el encabezado (header) o en el pie de página (footer).

Para crear una barra de navegación, utilizaremos el atributo **data-role** y le pasaremos el valor **"navbar"** dentro de una etiqueta ** **:

```
<div data-role="navbar">
```

Dentro de este elemento, incluiremos una lista desordenada (**ul**) donde cada elemento de la lista (**li**) contendrá un link.

Si queremos mostrar una barra de navegación en la parte superior de la página, incluiremos el atributo **data-role="navbar"** dentro del **data- role="header"** y, en caso de querer mostrarlo en el pie de página, lo incluiremos en el **data-role="footer"**.

En el siguiente código de ejemplo, agregamos una barra de navegación dentro de un pie de página (**data-role="footer"**) que nos llevará a cuatro secciones de página: "Nosotros", "Productos", "Locales" y "Delivery".

```
<div data-role="footer" data-position="fixed">

 <div data-role="navbar">
   <ul data-role="listview">
      <li><a href="#nosotros">Nosotros</a></li>
      <li><a href="#productos">Productos</a></li>
      <li><a href="#locales">Locales</a></li>
      <li><a href="#delivery">Delivery</a></li>
   </ul>
 </div>

</div>
```



Figura 14. Cada botón se puede vincular tanto a una página interna como a una página externa.

Botón activo

Para indicarle al usuario que se encuentra en una determinada página (**página activa**), usamos la clase que nos provee jQuery Mobile **ui-btn-active**. De modo tal que, cuando el usuario haga clic en un botón de la barra de navegación, este quedará resaltado por sobre los demás.

Siguiendo con el ejemplo anterior, para indicarle al usuario cuál es la página activa (en este caso, la página llamada "Nosotros"), agregamos la clase **ui-btn-active** al primer elemento de la lista, como se muestra en el siguiente código:

```
<div data-role="footer" data-position="fixed">

 <div data-role="navbar">
  <ul data-role="listview">
     <li><a href="#nosotros" class="ui-btn-active">Nosotros</a></li>
     <li><a href="#productos">Productos</a></li>
     <li><a href="#locales">Locales</a></li>
     <li><a href="#delivery">Delivery</a></li>
  </ul>
 </div>

</div>
```

Barra de navegación fija

Si deseamos que la barra de navegación se mantenga constante en todas las páginas donde la incluimos, vamos a tener que asignarle un nombre al footer mediante la propiedad **data-id**.

El nombre que le daremos al footer debe ser el mismo en todas las páginas. De este modo jQuery Mobile no animará el pie de página.

Además debemos ponerle el atributo **data-position="fixed"**, de modo que el footer se mantenga fijo.

Siguiendo con el ejemplo anterior, el código para la primera página ("Productos") sería:

```
<div data-role="page" id="nosotros">

 <div data-role="header" data-position="fixed">
```

ANCHO DE LOS BOTONES DE NAVBAR

Por defecto, los botones de navbar se alinean en forma horizontal con un tope de cinco elementos. Cada uno de los botones tendrá el mismo ancho; por ejemplo, si navbar tiene tres botones, cada botón ocupará 1/3 de la anchura de la barra de navegación. Si navbar tiene más de cinco botones, los mostrará en varias líneas.

```
        <h1> Heladería & Más </h1>
    </div>

    <div data-role="content">
     <h3>Nosotros</h3>
     <p> Somos una empresa que se dedica a la elaboración y comercialización de
helados y postres helados artesanales. Nuestra trayectoria nos ha colocado entre las
empresas líder en este rubro dentro del país </p>
    </div>

    <div data-role="footer" data-id="menu" data-position="fixed" >
     <div data-role="navbar">
        <ul>
       <li><a href="#nosotros" class="ui-btn-active">Nosotros</a></li>
       <li><a href="#productos">Productos</a></li>
       <li><a href="#locales">Locales</a></li>
       <li><a href="#delivery">Delivery</a></li>
      </ul>
     </div>
    </div>
   </div>
```

Y el código para la segunda página ("Productos") sería:

```
<div data-role="page" id="productos"data-position="fixed" >

 <div data-role="header" data-position="fixed">
  <h1> Heladería & Más </h1>
```

LISTAS EN HTML

La **lista no ordenada** consiste en un conjunto de elementos relacionados entre sí, en los que no se indica un orden o una secuencia determinados. HTML nos provee de las etiquetas **<ul>** para definir la lista, y la etiqueta **<li>** para definir cada uno de los elementos de la lista.

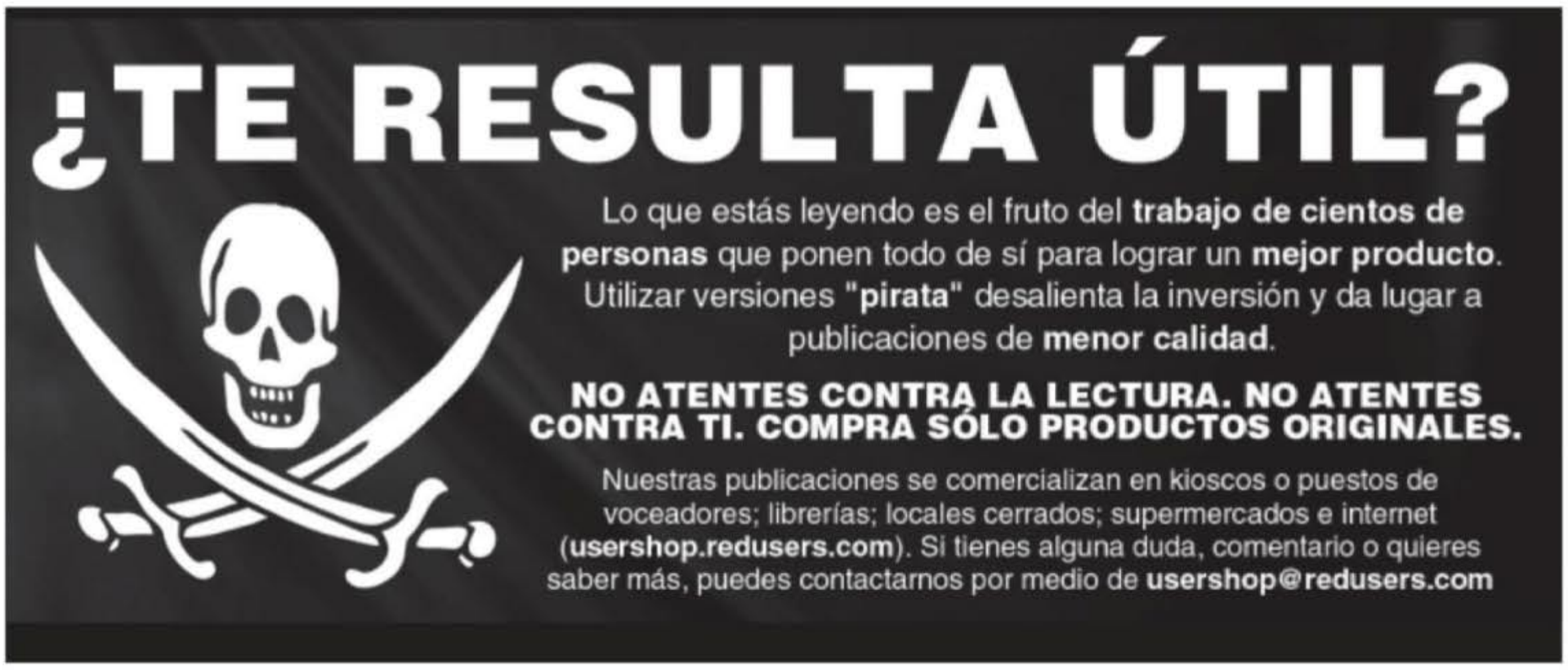
```
</div>

<div data-role="content">
 <h3>Productos</h3>
 <p> Tenemos los mejores productos </p>
</div>

 <div data-role="footer" data-id="menu" data-position="fixed" >
  <div data-role="navbar">
       <ul>
    <li><a href="#nosotros" class="ui-btn-active">Nosotros</a></li>
    <li><a href="#productos">Productos</a></li>
    <li><a href="#locales">Locales</a></li>
    <li><a href="#delivery">Delivery</a></li>
   </ul>
  </div>
 </div>
</div>
```

Iconos en el navbar

Al igual que hicimos con los botones y con las listas, podemos incluir iconos en los elementos del navbar, para ello usamos también la propiedad **data-icon**. En el siguiente código, vemos un ejemplo de navbar con iconos disponibles en el framework:



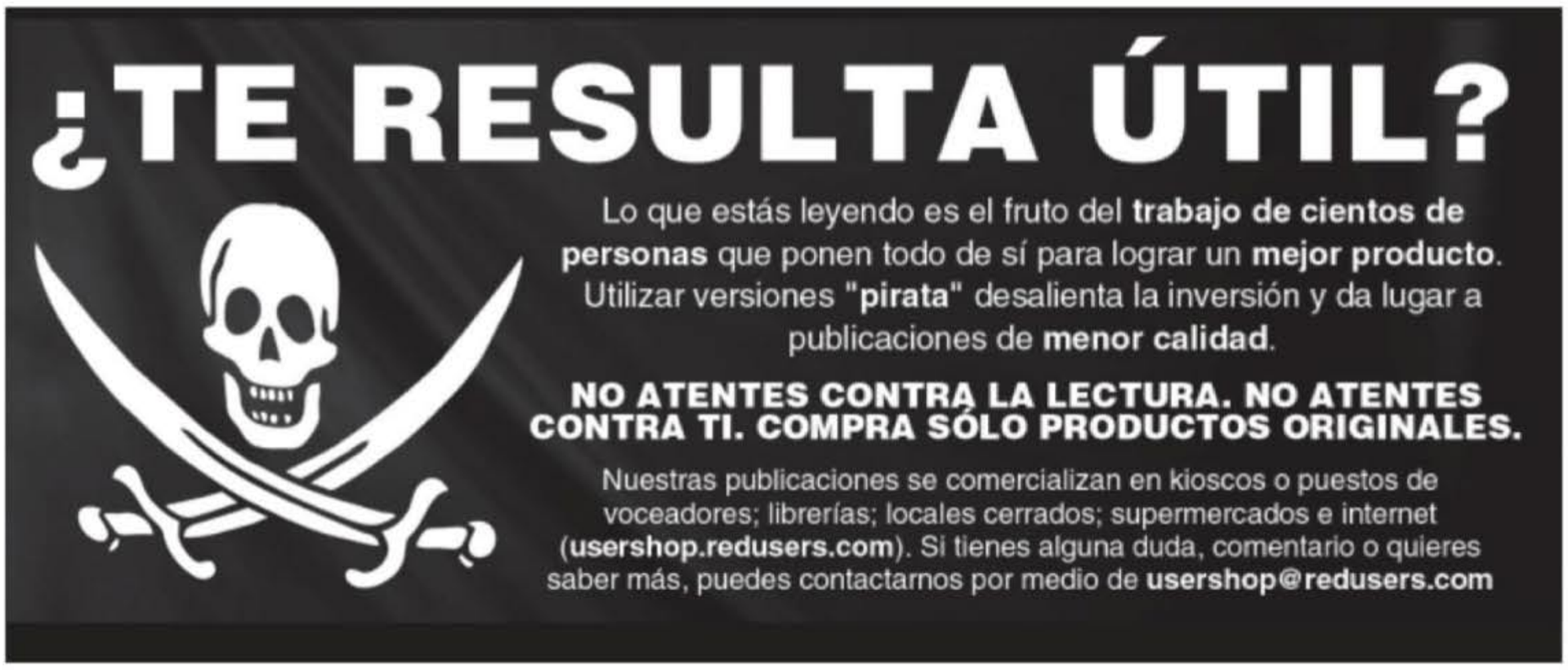
```
</div>

<div data-role="content">
 <h3>Productos</h3>
 <p> Tenemos los mejores productos </p>
</div>

 <div data-role="footer" data-id="menu" data-position="fixed" >
  <div data-role="navbar">
       <ul>
    <li><a href="#nosotros" class="ui-btn-active">Nosotros</a></li>
    <li><a href="#productos">Productos</a></li>
    <li><a href="#locales">Locales</a></li>
    <li><a href="#delivery">Delivery</a></li>
   </ul>
  </div>
 </div>
</div>
```

Iconos en el navbar

Al igual que hicimos con los botones y con las listas, podemos incluir iconos en los elementos del navbar, para ello usamos también la propiedad **data-icon**. En el siguiente código, vemos un ejemplo de navbar con iconos disponibles en el framework:



```
<div data-role="footer" data-position="fixed" >
 <div data-role="navbar">
      <ul>
    <li><a href="#inicio" data-icon="home">Inicio</a></li>
    <li><a href="#albunes" data-icon="grid">Albumes</a></li>
    <li><a href="#buscar" data-icon="search">Buscar</a></li>
    <li><a href="#mas" data-icon="plus">Más</a></li>
     </ul>
   </div>
  </div>
</div>
```



Figura 15. Podemos usar el atributo **`data-icon`** para cambiar la posición del icono.

RESUMEN

JQuery Mobile nos ofrece un conjunto de componentes que nos ayudan a simplificar los requisitos de navegación entre las distintas páginas de nuestra aplicación. En este capítulo, hemos visto cómo mejorar la accesibilidad a las diferentes secciones de nuestra aplicación mediante el uso de botones y barras de navegación. Aprendimos a crear y personalizar estos componentes. Más tarde, analizamos el empleo de listas en una aplicación y, por último, aprendimos a crear cada uno de los tipos de listas para organizar la información dentro de nuestra aplicación.

Actividades

TEST DE AUTOEVALUACIÓN

1. ¿Qué atributo y valor debe llevar un botón para que esté desactivado?
2. ¿Qué valor debe tener el atributo **data-icon** de un botón, para que muestre un icono de borrar?
3. ¿Cómo cambiaría la posición de un icono para que se vea a la derecha del texto?
4. ¿Qué es una lista anidada?
5. ¿Qué etiqueta HTML se usa para enumerar una lista?

EJERCICIOS PRÁCTICOS

1. Diseñe una página que tenga un enlace a otra página y que en el head muestre un botón "volver" para volver a la página de la cual fue llamado.
2. Arme una lista anidada, en donde muestre marcas de automóviles y, en otra lista, sus respectivos modelos.
3. A la lista del punto 2, agréguele iconos de 16 x 16 pixeles.
4. Agréguele también un contador burbuja, que muestre la cantidad de modelos de automóviles.
5. Confeccione una página nueva y agréguele dos botones agrupados; uno que mande mensajes SMS y el otro que llame a un número determinado.



PROFESOR EN LÍNEA

Si tiene alguna consulta técnica relacionada con el contenido, puede contactarse con nuestros expertos: profesor@redusers.com.

*04

Formularios

En los capítulos anteriores aprendimos a crear la estructura de una página en jQuery Mobile, usando sus elementos básicos (botones, listas). En este capítulo, explicaremos el funcionamiento y la utilidad de los formularios.
Estos nos permiten introducir datos para ser procesados en el servidor. También conoceremos y aprenderemos a utilizar los componentes incluidos dentro de un formulario para el envío de datos.

- Estructura de un formulario.... 92
- Controles de un formulario 96
- Resumen..... 115
- Actividades..... 116

Estructura de un formulario

Así como en HTML, en jQuery Mobile también podemos trabajar con **formularios**, y se lo hace de una forma muy sencilla.

Los formularios nos sirven para crear una interacción entre la aplicación web y los usuarios. Es decir, nos permiten recibir datos que serán procesados en nuestro servidor por medio de algún lenguaje de programación, como, por ejemplo, **PHP**.

Al igual que en HTML, la estructura básica de un formulario en jQuery Mobile comienza con la etiqueta de apertura **<form>** y termina con la etiqueta de cierre **</form>**. Con estas etiquetas, el navegador interpretará que estamos frente a un formulario.

Dentro del formulario, deberemos incluir controles para que el usuario pueda interactuar con la aplicación.

Sin los controles, la sintaxis de un formulario vacío se vería como la siguiente:

```
<form action="mi_archivo.php" method="post" >

    <!-- Controles de formulario -->

</form>
```

Envío del formulario

Para el envío de información, tanto en HTML como en jQuery Mobile se utiliza el atributo **method**, y los valores que este puede tomar son dos: **get** o **post**. Otro atributo para tener en cuenta es **action**, que recibe por valor el nombre del archivo que va a procesar los datos ubicados en nuestro servidor.

En el ejemplo que representamos a continuación, definimos un formulario donde le solicitamos al usuario que ingrese su dirección de correo electrónico:

```
<form action="miArchivo.php" method="post">

    <label>Ingrese su correo electrónico: </label>
```

```
        <input type="text" name="correo"/>
        <input type="submit" value="enviar"/>

</form>
```



Figura 1. Cuando se envía un formulario, jQuery Mobile aplica un efecto de transición Ajax.

Observemos con más detalle el ejemplo anterior:

- El atributo **action** –que, en este caso, tiene por valor **miArchivo.php**– indica el archivo donde se enviarán los datos. Puede ser un archivo o una dirección de e-mail.
- El atributo **method** –que, en nuestro ejemplo, tiene por valor **post**– indica la forma en que se enviarán los datos para ser procesados en el archivo **miArchivo.php**. Este atributo puede tomar los valores **get** o **post**.

En el ejemplo, también tenemos las etiquetas **<label></label>**, que solamente muestran el texto **Ingrese su correo electrónico**, y una caja de



PHP

PHP (**Pre-Hypertext Processor**, procesador previo al hipertexto) es un lenguaje de programación empleado para diseñar sitios web dinámicos, con la posibilidad de combinarlo con bases de datos. PHP es un lenguaje que se ejecuta en el servidor; esto quiere decir que el código del programa no va a ser interpretado por el navegador, sino por un servidor web.

texto o text input, donde el usuario va a escribir su correo electrónico. Por último, un botón enviar del tipo **submit**.

De esta forma, el usuario interactúa con la aplicación pasando algún dato –en este ejemplo, su dirección de correo electrónico–, que es procesado por el archivo PHP **miArchivo.php** y que, posteriormente, será enviado por e-mail o guardado en una base de datos.

Deshabilitar elementos

Todos los elementos de un formulario cuentan con los atributos **disabled** o **enabled**. En el caso de que necesitemos deshabilitar un elemento, tenemos que asignarle el valor **true** al atributo **disabled**; de esta forma, el usuario no va a poder ingresar datos en este elemento, tal como lo vemos en el siguiente ejemplo:

```
<form action="index2.php" method="post" >

 <label>Ingrese su nombre</label>
 <input type="text" name="nombre">
 <label>Ingrese su e-mail</label>
 <input type="text" name="email" disabled="true"/>              <input
type="submit" value="enviar">

</form>
```

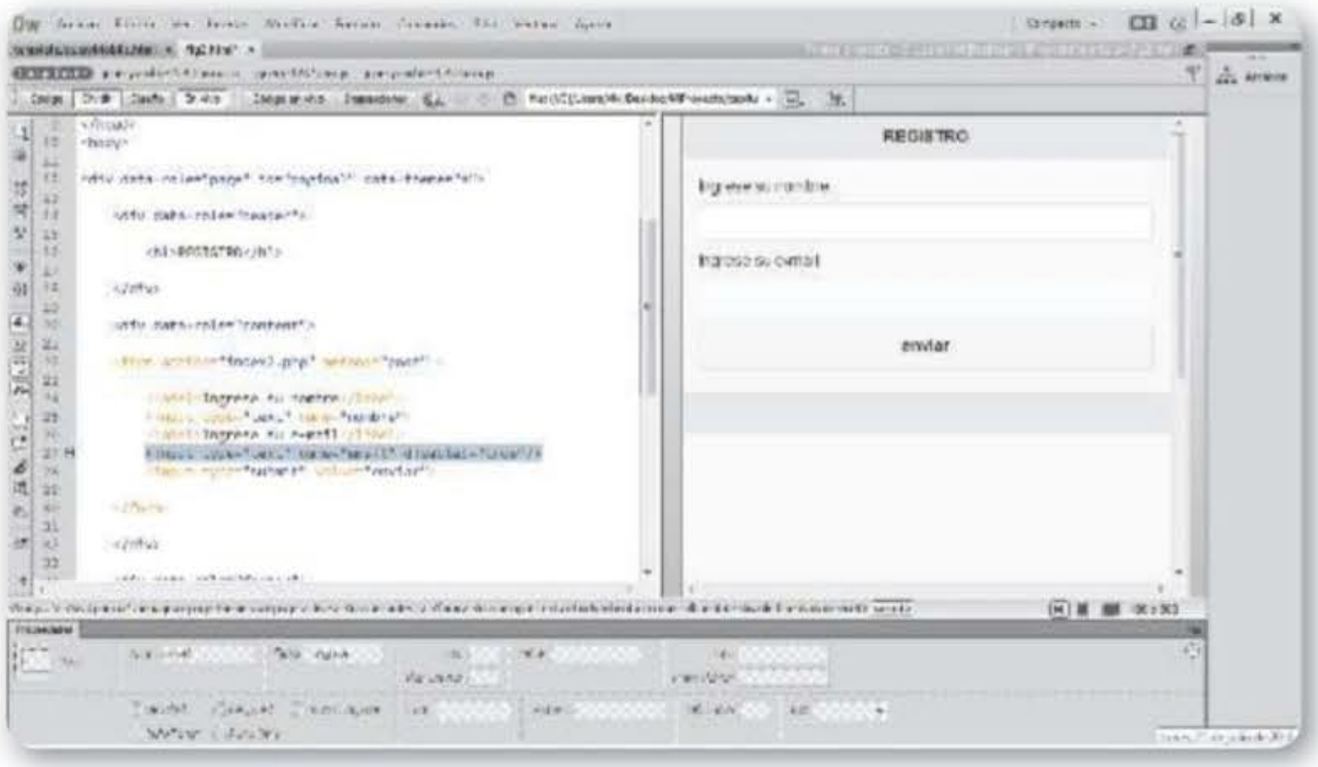


Figura 2. El input del tipo **`text email`** tiene la propiedad **`disabled="true"`**, que le impide al usuario ingresar datos allí.

Minielementos

Para hacer el formulario más compacto y pequeño –ya que generalmente contamos con espacios reducidos en los dispositivos móviles– utilizamos el atributo **data-mini**.

Este atributo admite los valores **true** y **false**. Al establecer el atributo **data-mini="true"**, creamos una versión más chica del componente.

```
<form action="miArchivo.php" method="post">

        <label>Ingrese su correo electrónico: </label>
        <input type="text" name="correo" data-mini="true"/>
        <input type="submit" value="enviar" data-mini="true"/>

</form>
```

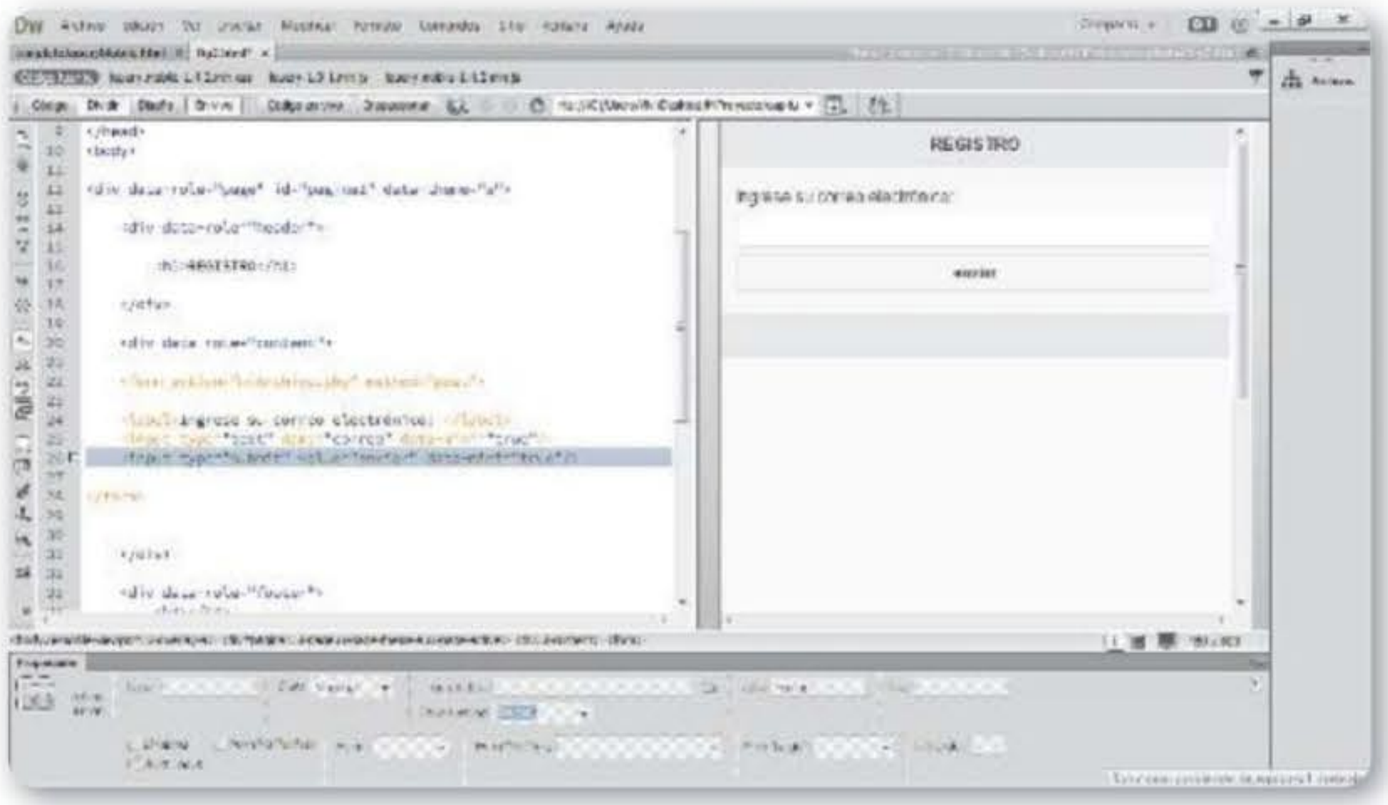


Figura 3. El atributo **`data-mini="true"`** se puede aplicar a todos los controles del formulario.

MÉTODOS GET Y POST

Los métodos **get** y **post** sirven para enviar información desde un formulario para que sea procesada en el servidor. Una de las diferencias que existe en el envío de información es que el método **get** lo hace por URL, es decir, cuando se procesen los datos, se van a poder ver en la URL del navegador; en cambio, si se envían por **post**, no son visibles por el usuario.

Field containers

Otra manera de reducir espacio consiste en agrupar los elementos de un formulario. Esto se logra a través de una etiqueta ** ** con el atributo **data-role**, pasándole el valor **fieldcontain**.

Una característica importante de este atributo es que el diseño del formulario se define automáticamente en función de la anchura del dispositivo. Por ejemplo, si el dispositivo tiene una pantalla estrecha, la etiqueta se colocará en la parte superior del control del formulario, en lugar de ubicarla a su izquierda.

Así, podemos dividir o agrupar visualmente varios elementos del mismo tipo y ganar un poco más de espacio, como se muestra en el siguiente ejemplo:

> ES POSIBLE AGRUPAR LOS ELEMENTOS DEL FORMULARIO PARA ADAPTARLO A CADA TIPO DE DISPOSITIVO

```
<form action="index.php" method="post" >
    <div data-role="fieldcontain">
      <label>Ingrese su nombre</label>
      <input type="text" name="nombre">
      <label>Ingrese su apellido</label>
      <input type="text" name="apellido">
      <label>Ingrese su e-mail</label>
      <input type="text" name="email">
      </div>
      <input type="submit" value="enviar" >
</form>
```

Controles de un formulario

Un formulario contiene –además de los elementos típicos, como párrafos, imágenes y listas– un conjunto de **controles**, llamados también **componentes** o **campos de formulario**, que le van a permitir al usuario ingresar, seleccionar y visualizar información.

jQuery Mobile utiliza los mismos controles para formularios que HTML, pero adaptados a su framework, para que visualmente se adecuen a los dispositivos móviles.

Etiqueta (label)

El control **label** (**etiqueta**) se emplea en un formulario para mostrar información que el usuario no puede modificar, como, por ejemplo, descripciones o mensajes de ayuda, y para identificar otros controles, como pueden ser los cuadros de texto.

```
<form action="miArchivo.php" method="post">

        <label>Esta es una etiqueta o label </label>
<input type="text" name="texto" >

</form>
```



Figura 4. Si necesitamos, por alguna razón, ocultar una etiqueta, podemos aplicarle la clase **`ui-hidden-accesible`**.

La etiqueta **label** tiene el atributo **for**. Este atributo es opcional; cuando se lo agregamos a un control **label**, estamos indicando el identificador (con el atributo **id**) al cual pertenece el título.

```
<label for="nombre">Nombre</label> <br/>
<input type="text" id="nombre" name="nombre" />
```

En el ejemplo anterior, podemos observar cómo le agregamos el atributo **for** al **label** "**nombre**" y, a continuación, agregamos un input del tipo **text** con el atributo **id="nombre"**. De esta forma, estamos indicando que el **label** "**nombre**" pertenece al **text** con el identificador "**nombre**".

Otra opción interesante que cumple la misma función de un **label** dentro de un formulario es el atributo **placeholder**.

El **placeholder** representa un texto corto que muestra una sugerencia o ayuda para que el usuario ingrese información en un cuadro de texto o en un cuadro de búsqueda.

El texto que muestra este atributo se visualiza como una marca de agua dentro del control en el que se aplicó el atributo.

A diferencia de las páginas web convencionales, este atributo se utiliza mucho en dispositivos móviles y siempre tiene que ir sobre el componente que queremos describir.

En el ejemplo, agregamos el atributo **placeholder** a un cuadro de texto:

```
<form action="miArchivo.php" method="post">

    <input type="text" name="texto"
    placeholder="Ingrese su nombre">

</form>
```



Figura 5. Cuando se hace un clic en el cuadro de texto, el texto que se muestra (**placeholder**), en este caso, "Ingrese su nombre", desaparece.

Campo de texto (text input)

Los **campos de texto**, **text input**, o, simplemente, **campos de entrada**, se utilizan para recibir información generada por el usuario.

Como ya hemos mencionado, si solo vamos a mostrar o visualizar texto, usamos el control **label**, pero, si en cambio necesitamos que el usuario ingrese algún dato, debemos usar los campos de texto.

Tipos de text input

Existen distintos tipos de text input; jQuery Mobile soporta todos los tipos de text input de HTML5, como la contraseña, teléfono, e-mail y varios más. Solamente tenemos que agregar al atributo **type** el valor que necesitemos.

Los valores que podemos asignarle al atributo **type** son los siguientes: **text**, **password**, **email**, **tel**, **url**, **search**, **number**, **textarea**, **time**, **date**, **week**, **datatime**, **color**, **range** y **submit**.

```
<div data-role="content">
                <h2>Mi formulario<h2>
      <form action="index2.php" method="post" >

        <label>Contraseña</label>
        <input type="password" name="pass"  >
        <label>Numero</label>
        <input type="number" name="numero"  >
        <label>E-mail</label>
        <input type="email" name="email"  >
        <label>Url</label>
        <input type="url" name="url"  >
        <label>Teléfono</label>
        <input type="tel" name="tel"  >
        <label>Hora</label>
        <input type="time" name="hora"  >
        <label>Fecha</label>
        <input type="date" name="fecha"  >
        <label>Día</label>
        <input type="date" name="dia"  >
        <label>Semana</label>
        <input type="Week" name="semana"  >
        <label>Fecha y hora</label>
        <input type="datetime" name="fechayhora"  >
        <label>Color</label>
        <input type="color" name="color"  >
        <input type="submit" value="enviar" >
      </form>
       </div>
```

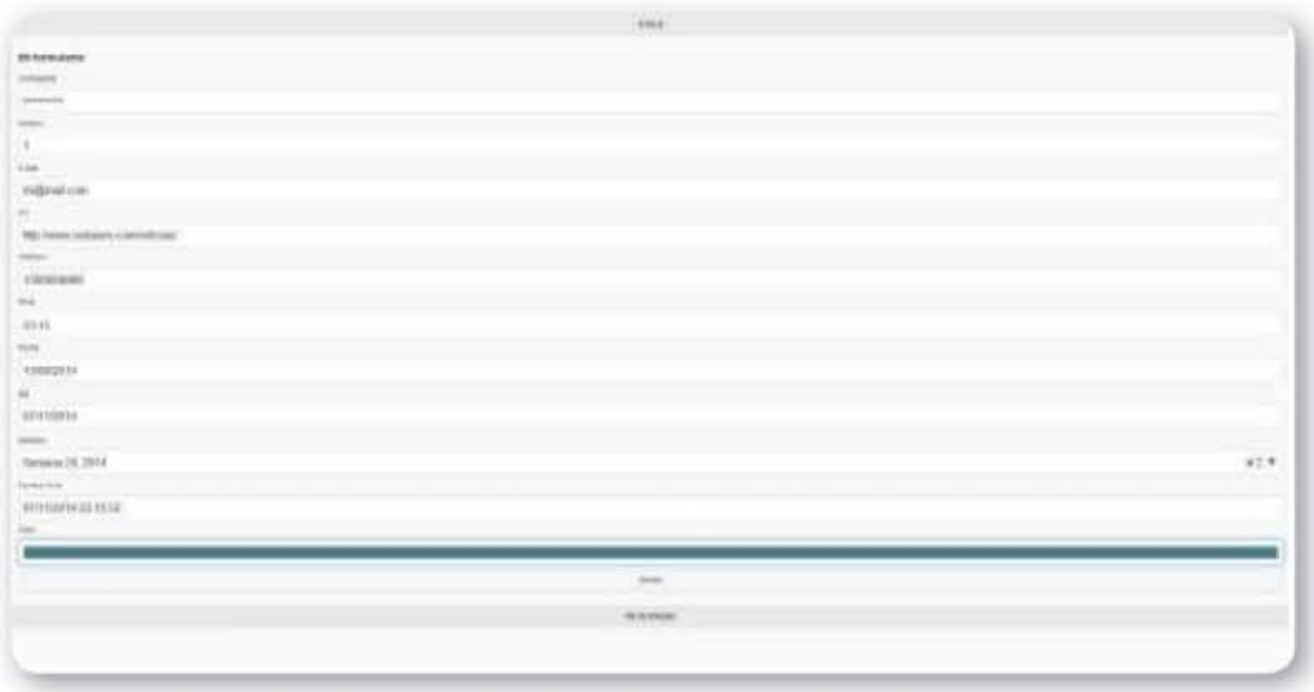


Figura 6. En Dreamweaver los campos de texto se pueden insertar a partir del menú **`Insertar/jQuery Mobile`**.

El control tipo **text** nos permite generar un cuadro de texto de una sola línea. Sus parámetros más importantes son: **name**, que permite establecer el nombre con el que haremos referencia al cuadro de texto en el código, y **maxlength**, que especifica la longitud máxima de caracteres permitidos para el cuadro de texto. Si no se establece este parámetro, se permite cualquier longitud de texto.

```
<label for="nombre">Ingrese su nombre:</label>
<input type="text" name="nombre" maxlength= "25">
```

El control tipo **password** (**contraseña**) es igual al cuadro de texto y se genera agregando el valor **password** al atributo **type**. Los valores que ingresemos en el cuadro de texto se verán con puntos o asteriscos para que no se pueda visualizar el texto escrito en él. Tengamos en cuenta que, en algunos dispositivos, el carácter que se ingresa se muestra durante un segundo antes de que sea cambiado por el carácter asterisco o punto.

```
<label>Ingrese su contraseña</label>
 <input type="password" name="contraseña">
```

En los casos de los tipos **número** (**type="number**) o **número de teléfono** (**type="tel"**), el framework interpreta que se trata de números, entonces nos facilita el ingreso, mostrándonos el teclado numérico y no el estándar.

En el caso del control de tipo **número**, aparte de mostrarnos un teclado simplificado, si tocamos sobre el campo de texto, se despliega una lista ordenada con los números negativos y positivos, para poder seleccionar el que necesitemos sin que tengamos que tipearlo sobre el teclado.

Con los controles del tipo **email** y **url** pasa exactamente lo mismo: el dispositivo nos muestra un teclado con @ y **.com** y otros caracteres especiales, para facilitarnos el ingreso de datos. Además, al definirlo como tipo **email** o **url**, estamos restringiendo el ingreso de datos a un formato válido.

Los controles que son del tipo **fecha** (**type="date"**), **fecha y hora** (**type="datetime"**) y **semana** (**type="week"**) vienen predeterminados directamente, es decir, si tocamos sobre el componente lista, se despliega un calendario para poder seleccionar un día, una fecha o semana, que necesitemos agregar al cuadro de texto. En el caso de un control del tipo **hora** (**type="time"**), al tocar el campo cuadro de texto podemos visualizar el formato de **hh:mm** para poder escribir sobre él, o bien seleccionar en la lista desplegable el número que necesitemos usar.

Para generar un control que nos permita seleccionar colores de una lista, usamos el tipo **color** (**type="color"**).

El tipo **submit** (**type="submit"**) nos crea un botón de envío. Al pulsarlo, todos los datos ingresados en los controles del formulario se enviarán para ser procesados.

EXISTEN DIFERENTES TIPOS DE CAMPOS DE TEXTO QUE PODEMOS AGREGAR A LOS FORMULARIOS

Textarea

Si bien el text input y el textarea nos sirven para el ingreso de datos por parte del usuario,

USAR DREAMWEAVER

En lugar de escribir manualmente el código de los controles, podemos insertarlos empleando Dreamweaver. Para ello, debemos hacer clic en el menú **Insertar/jQuery Mobile**. Del submenú que se despliega, seleccionamos el control que deseamos insertar. En forma automática, Dreamweaver nos generará el código del control, que podremos retocar de acuerdo con nuestras necesidades.

la diferencia que existe entre ambos es que **textarea** permite el ingreso de **texto multilínea** en el formulario.

Si tan solo colocamos el atributo **<textarea></textarea>**, va a permitir al usuario ingresar más de una línea de texto.

Un ejemplo clásico es un formulario de consulta, en donde el usuario ingresa en el textarea la consulta, y esta es enviada al servidor para luego ser procesada, como se muestra en el siguiente ejemplo:

```
<div data-role="content">
                    <h2>Mi formulario</h2>
      <form action="index2.php" method="post" >
        <label >Consulta:</label>
        <textarea name="textarea" id="textarea"></textarea>
         <input type="submit" value="enviar" >
      </form>
   </div>
```



Figura 7. El control **textarea** se va expandiendo a medida que vamos ingresando texto en él.

Campo de búsqueda (search input)

Como en HTML5, en jQuery Mobile también disponemos, en el atributo **type**, del valor **search** que nos sirve para ingresar valores de búsqueda. Este cuenta con un icono –ya predefinido– de una lupa en el costado izquierdo. Cuando ingresamos texto, en el costado derecho del cuadro de texto aparece un icono en forma de cruz, que al presionarlo nos permite borrar lo que escribimos.

Podemos utilizar la siguiente sintaxis para crear un campo de búsqueda en nuestros desarrollos:

```
<form action="index2.php" method="post" >
  <label>Buscar:</label>
  <input type="search" name="buscar" id="buscar"> </form>
```



Figura 8. En Dreamweaver podemos insertar un cuadro de búsqueda haciendo clic en **`Insertar/jQuery Mobile/Buscar`**.

Slider

El **slider** es otro valor que recibe el atributo **type** de la etiqueta **input** (**<input type="range">**). Los sliders nos permiten ajustar un valor entre un rango determinado mediante un componente deslizable.

El slider viene acompañado de un text input a la derecha, en el que podemos ingresar un valor que será asociado directamente a la barra deslizante. Lo mismo ocurre si nos deslizamos con la barra: nos mostrará el mismo valor en el text input de la derecha, por lo que siempre estarán sincronizados asegurándonos que tengan el mismo valor para luego ser procesados.

Los atributos que debemos tener en cuenta para configurar el slider son el valor de comienzo del rango, el valor final y el valor de incremento.

Para esto disponemos de los atributos **min** y **max** que serían, respectivamente, el valor inicial y el final. Para el incremento, contamos con el atributo **step**.



ATRIBUTOS COLS Y ROWS

Estos dos atributos son opcionales en el textarea: el atributo **cols** especifica el número de caracteres que el usuario ingresará por línea, y el atributo **rows** especifica el número de líneas visibles: **<textarea cols="40" rows="8" name="textarea" id="textarea"></textarea>**.

Para darle un mejor aspecto, contamos también con el atributo **data-highlight** que, si le pasamos por valor **"true"**, nos pinta en color el valor utilizado en la barra.

La sintaxis para crear un control del tipo slider es la siguiente:

```
<label>Slider: </label>
<input type="range" name="slider" id="slider" max="100" min="00" value="25" step="20" data-highlight="true" />
```

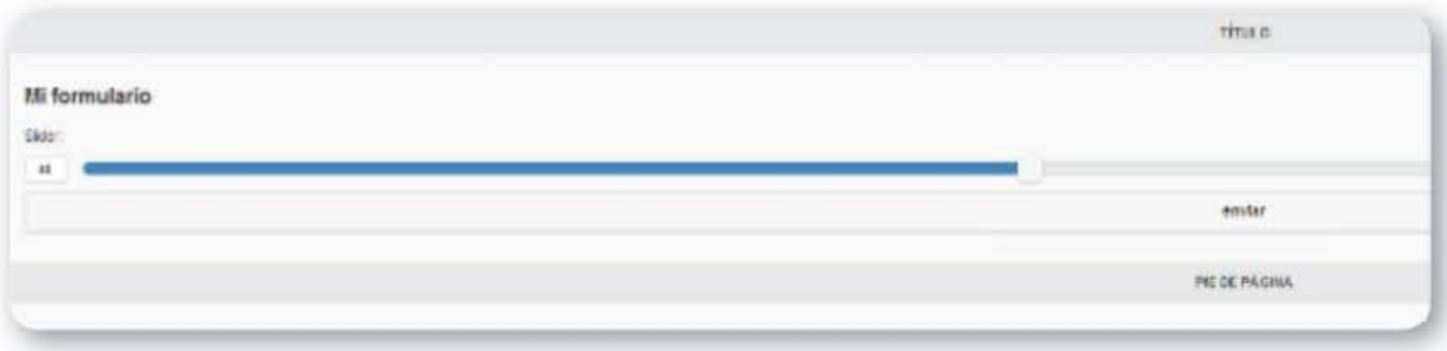


Figura 9. En Dreamweaver insertamos un control del tipo slider haciendo clic en **Insertar/jQuery Mobile/Regulador**.

Interruptor (flip toggle switch)

El botón **interruptor** (*flip toggle switch*) es un control muy usado en los dispositivos móviles; no es más que una barra de deslizamiento que contiene dos valores: **verdadero** (**true**) o **falso** (**false**), a los que también podemos llamar **prendido** (**on**) o **apagado** (**off**).

Este control es muy útil cuando tenemos que darle al usuario la posibilidad de elegir entre dos opciones. Solo permite seleccionar una opción; es decir, en términos informáticos es un control booleano.

Para utilizarlo en nuestro formulario, tenemos que agregar una etiqueta de HTML **<select></select>**, pasarle al atributo **data-role** el valor

DATO LÓGICO O BOOLEANO

El **dato booleano**, también llamado **boolean** o **dato lógico** es el tipo de dato que representa los valores binarios 0 o 1, o verdadero o falso. En informática se lo suele emplear para expresar y usar instrucciones lógicas en los programas. Se denomina así en honor al matemático y lógico inglés George Boole (1815-1864), uno de los fundadores de la lógica matemática.

slider y, en las etiquetas de opciones **<option></option>**, pasarle por valor (value), **on** u **off**. El código que presentamos a continuación muestra un ejemplo de cómo usar el control flip toogle switch:

```
<form action="../mi_archivo.php" method="post">

<label>Desea suscribirse para recibir: </label>
<br><br>

<label>Correo: </label>
  <select name="flip_2" id="flip_2" data-role="slider">
    <option value="off">Si</option>
    <option value="on">No</option>
  </select>

<label>Newsletter: </label>
  <select name="flip_3" id="flip_3" data-role="slider">
    <option value="off">Si</option>
    <option value="on">No</option>
  </select>

</form>
```



Figura 10. En Dreamweaver insertamos el interruptor haciendo clic en **Insertar/ jQuery Mobile/ Interruptor de conmutación de volteo**.

Botón de opción (radio button)

El control **botón de opción** (*radio button*) permite, a diferencia del control anterior (interruptor), tener más opciones: no solamente tendemos verdadero o falso, sino que nos sirve cuando necesitamos que el usuario elija una sola opción entre varias.

Un ejemplo podría ser la elección de pago de un determinado producto, donde las opciones que podría elegir el usuario serían: **Pago en efectivo**, **Pago con tarjeta de crédito** o **Pago con tarjeta de débito**. En este ejemplo, necesitamos una sola opción para después ser procesada. Igual que con el control flip toggle switch, la elección del usuario nos devolvería solamente un valor, pero con este control podría elegir entre varias.

La sintaxis que nos permite incluir este tipo de controles en un formulario es la siguiente:

```
<fieldset data-role="controlgroup" >
  <legend>Titulo del que se visualiza con el grupo:</legend>
      <input name="radio" id="radio-1" type="radio" value="opción1" checked>
      <label for="radio-1">Opción 1</label>
      <input name="radio" id="radio-2" type="radio" value="opción2">
      <label for="radio-2">Opción 2</label>
</fieldset>
```

> EL BOTÓN DE OPCIÓN PERMITE QUE EL USUARIO SELECCIONE UNA OPCIÓN ENTRE VARIAS DISPONIBLES

Como podemos observar en el código anterior, para incluir un radio button en un formulario tenemos que usar primero la etiqueta **<fieldset> </fieldset>** que se usa en HTML para agrupar campos relacionados, y al atributo **data-role** pasarle por valor **controlgroup**, seguido, de manera opcional, por una etiqueta **<legend> </legend>** en el caso de que quisiéramos ponerle un título al grupo de opciones.

Luego, en la etiqueta **<input></input>**, al atributo **type** se le asigna el valor **radio**. Es importante que, a continuación, agreguemos las etiquetas **<label> </label>** con el atributo **for** que deberá ser igual al **id** correspondiente del radio button.

En el ejemplo que detallamos a continuación, se le solicita al usuario que elija una opción de pago:

```
<fieldset data-role="controlgroup" >
  <legend>Elija una opción de pago:</legend>
```

```
        <input name="radio" id="radio-1" type="radio" value="efectivo" checked>
        <label for="radio-1">Pago en efectivo</label>
        <input name="radio" id="radio-2" type="radio" value="credito">
        <label for="radio-2">Pago con Tarjeta de Crédito</label>
        <input name="radio" id="radio-3" type="radio" value="debito">
        <label for="radio-3">Pago con Tarjeta de Débito</label>
</fieldset>
```

Radio button vertical y horizontal

De acuerdo con nuestra necesidad, en el diseño, podemos ubicar el radio button de manera vertical u horizontal; estas dos opciones se encuentran disponibles en el framework.

Para ubicarlo de manera vertical, tenemos que agregarle a la etiqueta **<fieldset></fieldset>** el atributo **data-type** y pasarle por valor **vertical**.
En cambio, si lo que queremos es ubicarlo de manera horizontal, tendríamos que pasarle el valor **horizontal** al atributo **data-type**.
Por defecto, el radio button viene de manera vertical, por lo que, si no le ponemos ningún valor, el framework interpretaría que es vertical.



Figura 11. En Dreamweaver insertamos el control radio button haciendo clic en **`Insertar/jQuery Mobile/Botón de opción`**.

Casilla de verificación (checkbox)

El control **casilla de verificación** (*checkbox*) se utiliza para permitirle al usuario seleccionar una condición sí o no, verdadero o falso; este control, a diferencia del radio button, nos permite hacer múltiples selecciones.

Es posible incluir la cantidad que nuestra aplicación necesite. Por ejemplo, podemos preguntarle al usuario qué tipo de deporte practica, y no necesariamente puede elegir una opción, sino que puede elegir varias.

Para agregar un solo checkbox al formulario, la sintaxis sería:

```
<label><input type="checkbox" name="checkbox-0" /> Acepto el contrato</label>
```

En el código anterior, se le solicita al usuario que acepte el contrato. Este ejemplo es muy usado para aceptar términos y condiciones en sitios web.

El siguiente ejemplo nos muestra cómo establecer un grupo de opciones dentro de un formulario, para permitir que el usuario nos indique cuáles son los deportes que practica.

```
<fieldset data-role="controlgroup" data-type="vertical">
        <legend>Seleccione el o los deportes que practica:</legend>
        <input name="checkbox-1" id="checkbox-1" type="checkbox">
        <label for="checkbox-1">Fútbol</label>
        <input name="checkbox-2" id="checkbox-2" type="checkbox">
        <label for="checkbox-2">Tenis</label>
        <input name="checkbox-3" id="checkbox-3" type="checkbox">
        <label for="checkbox-3">Béisbol</label>
        <input name="checkbox-4" id="checkbox-4" type="checkbox">
        <label for="checkbox-4">Natación</label>
            </fieldset>
```



Figura 12. En Dreamweaver insertamos el control checkbox haciendo clic en **Insertar/jQuery Mobile/Casilla de verificación.**

Checkbox vertical y horizontal

Al igual que con el radio button, el checkbox se puede visualizar de manera vertical u horizontal, de acuerdo con nuestro diseño. Para esto, a la etiqueta **<fieldset></fieldset>** dentro del atributo **data-type** le vamos a pasar el valor **horizontal** o **vertical**.

Menú de selección (select menu)

El **menú de selección** (*select menu*) se utiliza también, como los controles anteriores (radio button y checkbox), para mostrarle contenido al usuario y que este pueda elegir entre una o varias opciones. Al presionar el botón, se despliega una lista con opciones, y al seleccionar un elemento de la lista, esta se contrae.

La sintaxis para crear un menú de selección es la siguiente:

```
<label for="selectmenu" Opciones:</label>

<select name="selectmenu" id="selectmenu">

    <option value="option1">Opción 1</option>
    <option value="option2">Opción 2</option>
    <option value="option3">Opción 3</option>

</select>
```

En el siguiente ejemplo, creamos una lista desplegable para que el usuario seleccione una bebida:

```
<label for="selectmenu" >Bebidas Calientes:</label>

<select name="selectmenu" id="selectmenu">

  <option value="americano">Americano</option>
  <option value="machiatto_c">Machiatto</option>
```

CASILLAS DE VERIFICACIÓN CON DREAMWEAVER

Para agregar casillas de verificación utilizando Dreamweaver, primero seleccionamos en el panel código, donde vamos a ubicar este control; luego, vamos al menú **Insertar/jQuery Mobile** y del submenú seleccionamos **Casilla de verificación**. En el cuadro **Nombre** ubicado dentro del cuadro de diálogo **Casilla de verificación**, escribimos el nombre del grupo (**name**), e indicamos la cantidad de casillas que deseamos y el diseño (horizontal/vertical). Luego presionamos **Aceptar**.

```
<option value="expreso">Espresso</option>
<option value="irlandes">Café Irlandés</option>
<option value="moka">Café Moka</option>
<option value="capuchino">Capuchino</option>
<option value="chocolate">Chocolate</option>
<option value="te">Té</option>

</select>
```



Figura 13. En Dreamweaver insertamos el select menu haciendo clic en **`Insertar/jQuery Mobile/Seleccionar`**.

Personalizar el menú

Otra opción para el select menu es el **menú personalizado** que, a diferencia del menú nativo o básico, cambia el diseño mostrándonos los distintos ítems de manera diferente. Colocando en el atributo **data-native-menu** de la etiqueta **<select></select>** por valor **true**, tendríamos esta opción disponible, como podemos observar en el siguiente código:

```
<label for="selectmenu" >Bebidas Calientes:</label>

<select name="bebidas" id="bebidas" data-native-menu = "true" >
```

```
<option value="americano">Americano</option>
  <option value="machiatto_c">Machiatto</option>
  <option value="expreso">Espresso</option>
  <option value="irlandes">Café Irlandés</option>
  <option value="moka">Café Moka</option>
  <option value="capuchino">Capuchino</option>
  <option value="chocolate">Chocolate</option>
  <option value="te">Té</option>

</select>
```

De forma predeterminada en un select menú, podemos seleccionar un solo elemento de la lista, pero es posible habilitar la selección de varios elementos; para ello a la etiqueta **<select></select>** le agregamos el atributo **multiple** con el valor **multiple**.

Esto nos va a permitir hacer una selección de más de un elemento agregándonos un checkbox al costado derecho de cada ítem de la lista.

Siguiendo con el ejemplo anterior, si queremos que el usuario pueda elegir más de una bebida, el código nos quedaría de la siguiente manera:

> ES POSIBLE PERSONALIZAR EL MENÚ PARA MOSTRAR LOS ELEMENTOS DE FORMA DIFERENTE

```
<label for="selectmenu" >Bebidas Calientes:</label>

<select name="bebidas" id="bebidas" data-native-menu="false" multiple="multiple" >

  <option value="americano">Americano</option>
  <option value="machiatto_c">Machiatto</option>
  <option value="expreso">Espresso</option>
  <option value="irlandes">Café Irlandés</option>
  <option value="moka">Café Moka</option>
  <option value="capuchino">Capuchino</option>
  <option value="chocolate">Chocolate</option>
  <option value="te">Té</option>

</select>
```



Figura 14. Los menús nativos no admiten la selección múltiple, por lo tanto, si necesitamos esta opción, debemos personalizarlos.

Si queremos mostrar una opción, pero necesitamos que esté deshabilitada, lo que tenemos que hacer es agregarle a la etiqueta **<option></option>** el atributo **disabled** y pasarle por valor **true**. De esta manera, veríamos el ítem, pero no podríamos seleccionarlo.

Por ejemplo, supongamos que momentáneamente no vendemos más café irlandés, por lo tanto, el código nos quedaría de la siguiente manera:

```
<label for="selectmenu" >Bebidas Calientes:</label>

<select name="bebidas" id="bebidas" data-native-menu="false" multiple="multiple" >

  <option value="americano">Americano</option>
  <option value="machiatto_c">Machiatto</option>
  <option value="expreso">Espresso</option>
  <option value="irlandes" disabled="true" >Café Irlandés</option>
  <option value="moka">Café Moka</option>
  <option value="capuchino">Capuchino</option>
  <option value="chocolate">Chocolate</option>
  <option value="te">Té</option>

</select>
```

En caso de que necesitemos mostrar un título sobre cada elemento o simplemente agruparlos por categoría, contamos con la etiqueta **<optgroup></optgroup>**, a la que le agregamos el atributo **label** y le pasamos por valor el título de la categoría que necesitemos mostrar.

Por ejemplo, podemos crear un listado de equipos de fútbol y el país al cual pertenecen:

```
<select name="equipos" id="equipos" >
        <option value="">Seleccione un equipo de fútbol ...</option>
        <optgroup label="Inglaterra">
           <option value="manchester">Manchester City</option>
        </optgroup>
        <optgroup label="España">
                <option value="barcelona">Barcelona</option>
        </optgroup>
        <optgroup label="Uruguay">
                <option value="peniarol">Peñarol</option>
        </optgroup>
        <optgroup label="Italia">
                <option value="juventus">Juventus</option>
        </optgroup>
</select>
```



Figura 15. Podemos agrupar opciones que están relacionadas bajo un título en común usando la **<optgroup>**.

Select menu vertical y horizontal

Como en los componentes anteriores, también el select menu puede ser horizontal o vertical; para esto, podemos pasar los valores **horizontal**

o **vertical** en la etiqueta **<fieldset></fieldset>** en el atributo **data-type**, según lo necesitemos en nuestro diseño.

Selector de fechas (datepicker)

El **selector de fechas** (*datepicker*) es un componente que nos permite seleccionar una fecha dentro de un calendario.

Hay dos formas de mostrar el calendario: una es escribiendo la fecha sobre un componente de entrada de texto en donde, al escribir, se desliza un calendario para que podamos elegir la fecha por mostrar, como en el siguiente ejemplo:

```
<input data-role="date" type="text">
```

La otra forma es mostrando el calendario directamente sobre el formulario, como la siguiente:

```
<input data-role="date" data-inline="true" type="text">
```



Para esto, solo tenemos que agregarle al atributo **data-inline** el valor **true**.

Figura 16. El datepicker utiliza la biblioteca de componentes del framework de jQuery (jQuery UI), que le agrega los elementos gráficos.



SELECTOR DE FECHAS

Como bien dice en la página oficial de jQuery Mobile, el datepicker es un componente de jQuery y **no** de jQuery Mobile, pero que, combinado con este, funciona perfectamente. Solo tenemos que llamar a la librería de interfaz de usuario de jQuery (jQuery UI) con su correspondiente estilo.

Archivo (file)

El elemento **file** nos permite subir archivos al servidor, siempre y cuando tengamos los permisos para hacerlo. En la etiqueta **input** solo hay que agregarle al atributo **type** el valor **file**; esto hace que podamos navegar en el directorio del dispositivo y elegir el archivo por subir. Hay que tener en cuenta que no funciona en todos los sistemas, ya que es un componente muy nuevo.

```
<label for="file">File:</label>
<input name="file" id="file" value="" type="file">
```

RESUMEN

En este capítulo, describimos la utilidad de los formularios y sus principales atributos. Conocimos los métodos de envío de formulario (post y get). También explicamos los diferentes controles que podemos incluir en el formulario. Aprendimos a utilizar los principales atributos de los controles que se emplean para ingresar información (text input), para localizar información (search input) y los que permiten seleccionar opciones por parte del usuario (slider, flip toogle switch, radio buttons, checkboxes, select menu, datepicker y file).

Actividades

TEST DE AUTOEVALUACIÓN

1. ¿Cuál es el atributo HTML por el cual el navegador interpreta que estamos frente a un formulario?
2. ¿Qué valor debe tener el atributo **type** para poder escribir una contraseña?
3. ¿Qué atributo, en un slider, nos indica el mínimo y el máximo del control?
4. ¿Se puede seleccionar más de un elemento en un select menu?
5. ¿Para qué se utiliza el atributo **date**?

EJERCICIOS PRÁCTICOS

1. Realice un formulario que tenga tres etiquetas **input**, una para la contraseña, otra para escribir un e-mail y otra para la entrada de números.
2. Al formulario del punto 1 agréguele cinco checkboxes dándole al usuario la opción de seleccionar cuáles son sus colores favoritos.
3. Realice un formulario que tenga un **radio button**, donde le dé al usuario la opción de elegir el sexo (masculino/femenino).
4. Agréguele también al formulario anterior un **flip toggle switch**, que le permita al usuario seleccionar la opción de si estudia o trabaja.
5. Realice un formulario de consulta en donde le pida al usuario el e-mail, la fecha de nacimiento y un breve comentario sobre sus estudios cursados.



PROFESOR EN LÍNEA

Si tiene alguna consulta técnica relacionada con el contenido, puede contactarse con nuestros expertos: profesor@redusers.com.

*05

Otros componentes de la interfaz de usuario

En este capítulo, conoceremos otros componentes que nos ayudarán a crear una buena interfaz de usuario, como las tablas, el acordeón, las pestañas (tabs) y distintos tipos de ventanas emergentes. Estos componentes nos permiten mostrar la información de nuestro sitio web en espacios limitados, como las pantallas de los smartphones o de las tabletas. Por lo tanto, conoceremos sus principales atributos y, a través de diversos ejemplos, aprenderemos a crearlos.

- Otros componentes UI 118
- Tablas 118
- Componentes desplegables ..126
- Pestañas (tabs) 129
- Ventanas emergentes (pop-up).....132
- Paneles.....146
- Resumen.....149
- Actividades.....150

↗↗↗

Otros componentes UI

Así como disponemos de distintos componentes para formularios o, como vimos en capítulos anteriores, listas y botones; también el framework nos ofrece una gran cantidad de otros componentes que no son tan comunes a la hora de armar una aplicación, pero que sí son muy útiles para mejorar aún más la experiencia de usuario.

Tablas

Cuando necesitamos mostrar datos u organizar la información, lo más conveniente es mostrarlos en una tabla.

Una tabla en programación se utiliza para presentar los datos y organizarlos de manera que el usuario pueda ver la información de forma correcta. Está representada por **columnas** y **filas**. La intersección entre una fila y una columna forma una **celda**. Dentro de una celda podemos incluir cualquier elemento que compone la aplicación, como texto, imágenes, listas.

Las tablas son un componente muy útil para presentar información con un formato tabular, como por ejemplo un listado con datos personales, resultados estadísticos, entre otras opciones.

Cuando tenemos una tabla con muchas columnas, se presenta un problema, ya que los datos no se verán en su totalidad en pantallas pequeñas como las de un smartphone o una tablet. Tendremos que desplazarnos por la tabla de izquierda a derecha para poder visualizar el resto de las columnas.

Para solucionar este inconveniente, jQuery Mobile nos ofrece dos opciones para la presentación de tablas haciendo uso del concepto

EFECTO AUDIO

En los componentes desplegables (**collapsible** y **collapsible set**), podemos incluir dos atributos que proporcionan un texto audible para los usuarios que tengan un software lector de pantalla. Estos son **data-collapsed-cue-text** y **data-expanded-cue-tex**, y reciben por valor una cadena de texto.

responsive que, como vimos en el **Capítulo 2**, se refiere a adaptar nuestra web a las distintas resoluciones de pantalla, ya sea que la visualicemos en un smartphone o una tableta. Estas dos opciones son:

- **Modo reestructurado** (*reflow table*): cuando el ancho de la pantalla del dispositivo es demasiado pequeño para mostrar todas las columnas de la tabla, automáticamente cambia el diseño de la tabla, y se reorganizan sus datos en una sola columna; es decir, las filas de la tabla se representan en una columna con sus respectivas etiquetas. Este formato es ideal para datos que no necesitan ser comparados entre sí.
- **Modo conmutación de columnas** (*column toggle table*): este modo trabaja ocultando o mostrando columnas basadas en el ancho de la pantalla y de acuerdo a la prioridad que le asignó el desarrollador. A su vez, también le permite al usuario controlar, en forma manual, las columnas que desea visualizar por medio de un menú emergente.

> JQUERY MOBILE NOS OFRECE DOS FORMATOS PARA LA PRESENTACIÓN DE TABLAS

Crear tablas

Ambos modos de tabla (reflow y column toggle) comienzan con el estándar marcado de HTML.

Para diseñar una tabla, utilizamos las etiquetas **<table></table>**. Estas etiquetas nos indican el comienzo y el fin de la tabla. Dentro de estas, además de definir la estructura de la tabla, debemos agregar el atributo propio de jQuery Mobile **data-role="table"**.

Para definir la estructura de la tabla usaremos las siguientes etiquetas de HTML:

- **<tr></tr>**: utilizamos estas etiquetas para indicar el comienzo y el fin de una fila.
- **<td></td>**: empleamos estas etiquetas para indicar el comienzo y el fin de una celda. Dentro de estas etiquetas, colocaremos el contenido de la tabla.
- **<th></th>**: utilizamos estas etiquetas para indicar las celdas que serán consideradas como **encabezado de la tabla**.

- **<thead></thead>**: estas etiquetas nos permiten crear la zona del encabezado de la tabla. Aquí agrupamos todas las celdas que actuarán como encabezado de la tabla.
- **<tbody></tbody>**: estas etiquetas las utilizamos para agrupar el resto de las filas de la tabla.

Crear una tabla en modo reestructurado (reflow)

Para crear una tabla en este modo, tenemos que agregar, dentro de la etiqueta **<table>**, el atributo **data-mode="reflow"** y la clase **class="ui-responsive"**.

```
<table data-role="table" data-mode="reflow"  class="ui-responsive" id="mi_tabla">
```

En el siguiente ejemplo ilustramos cómo crear una tabla con cinco columnas empleando el modo reflow.

```
<table data-role="table" data-mode="reflow" class="ui-responsive">

   <thead>
      <tr>
         <th>Consola</th>
         <th>Juegos</th>
         <th>Género</th>
         <th>Idioma</th>
         <th>Cantidad de Jugadores</th>
         <th>Precio </th>
       </tr>
   </thead>
   <tbody>
   <tr>
      <td>Nintendo ds</td>
      <td>Mario kart</td>
      <td>Carreras</td>
      <td>Inglés </td>
      <td>1</td>
      <td>$90,00 </td>
```

```
 </tr>
 <tr>
<td>Nintendo ds</td>
<td>Mario party ds</td>
<td>Plataformas</td>
<td>Inglés </td>
<td>1</td>
<td>$85,00 </td>
 </tr>
 <tr>
<td>Nintendo Wii</td>
<td>Deca sports</td>
<td>Deportes</td>
<td>Inglés</td>
<td>4</td>
<td>$120,00 </td>
 </tr>
 <tr>
<td>Nintendo Wii</td>
<td>Mario Galaxy</td>
<td>Aventura</td>
<td>Inglés</td>
<td>1</td>
<td>$130,00 </td>
 </tr>
 <tr>
<td>Play 2</td>
<td>Kung Fu Panda</td>
<td>Aventura </td>
<td>Inglés</td>
<td>4</td>
<td>$25,00 </td>
</tr>
    </tbody>

</table>
```

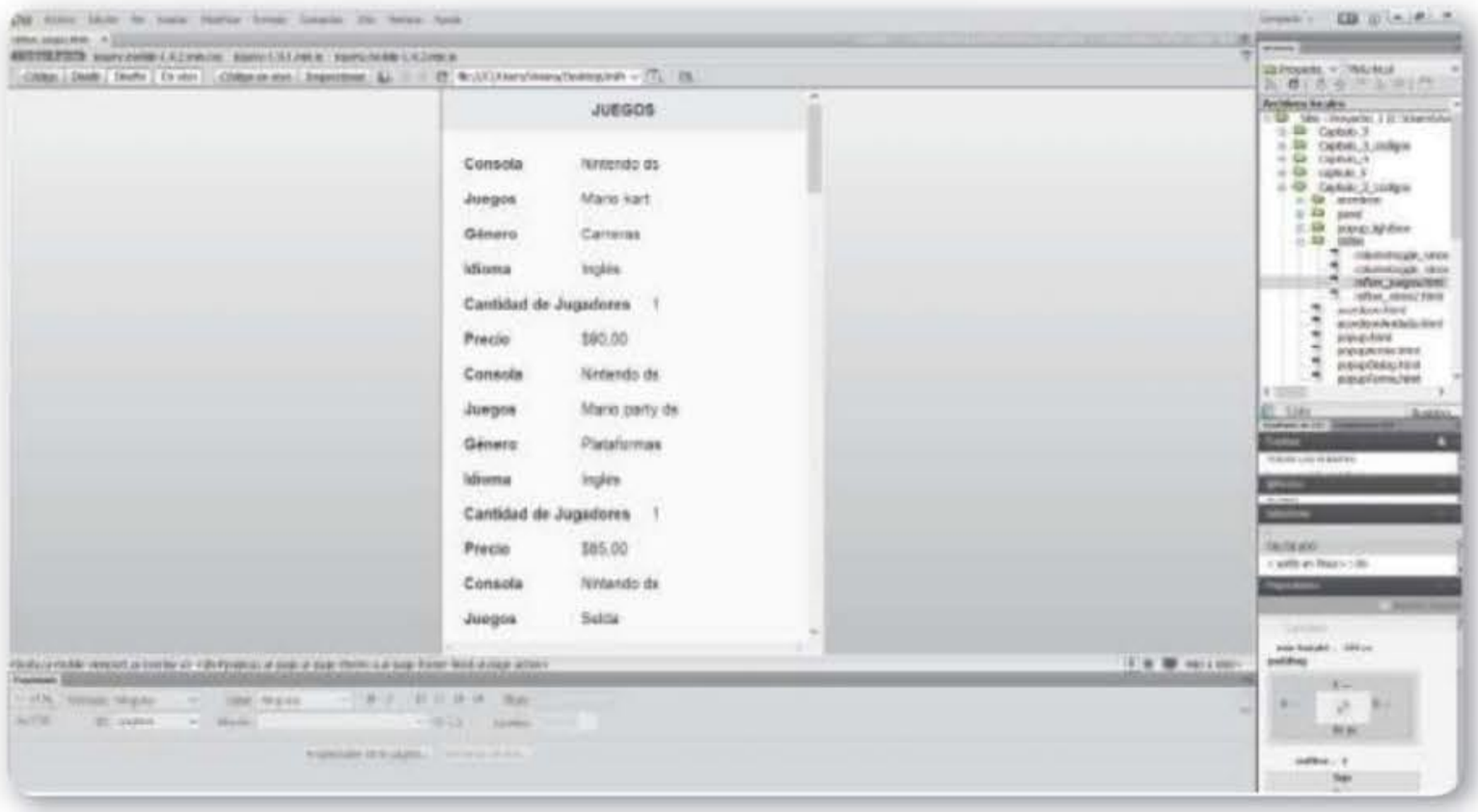


Figura 1. La tabla en modo reestructurado hace que los datos se puedan ver y leer con mayor facilidad.

Crear una tabla en modo conmutación de columnas (column toggle)

Para crear una tabla en este modo, tenemos que agregar, dentro de la etiqueta **<table>**, el atributo **data-mode="columntoggle"** y la clase **class="ui-responsive"**.

```
<table data-role="table" data-mode="columntoggle" class="ui-responsive"
id="mi_tabla">
```

Además, debemos especificar las columnas que deseamos que aparezcan. Para ello, dentro de la etiqueta **<th>**, debemos asignarle prioridades mediante el atributo **data-priority**.

Los valores que puede tomar este atributo son números que van del **1** al **6**. El número 1 es el nivel más alto, y el 6, el nivel más bajo.

Si no se especifica ningún valor de prioridad, entonces, la columna se considera necesaria y siempre se va a mostrar.

En el siguiente ejemplo, establecemos diferentes niveles de prioridad a las columnas de la tabla.

```
<thead>
    <tr>
```

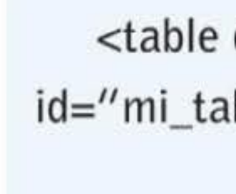
```
    <th data-priority ="1">Año</th>
  <th data-priority>Sucursal</th>
  <th data-priority ="2" >Producto</th>
  <th data-priority ="3">Trimestre 1</th>
  <th data-priority ="4">Trimestre 2</th>
  <th data-priority ="5">Trimestre 3</th>
  <th data-priority ="6">Trimestre 4</th>
  <th >Total</th>
 </tr>
</thead>
```

Cuando el ancho del dispositivo no permita ver la totalidad, jQuery Mobile ocultará las columnas sobre la base de la prioridad que le dio el desarrollador y, a su vez, aparecerá un botón emergente (**Columns...**); al hacer un clic sobre él, se mostrará una lista dinámica de las columnas que se encuentran ocultas. De esta forma, el usuario podrá seleccionar la columna que desea visualizar.

El código completo de la tabla nos quedaría de la siguiente manera:

```
<table data-role="table" data-mode="columntoggle" class="ui-responsive"
id="mi_tabla">

 <thead>
  <tr>
    <th data-priority ="1">Año</th>
     <th data-priority>Sucursal</th>
     <th data-priority ="2" >Producto</th>
     <th data-priority ="3">Trimestre 1</th>
     <th data-priority ="4">Trimestre 2</th>
     <th data-priority ="5">Trimestre 3</th>
```



PRIORIDAD PARA EL USUARIO

Tengamos en cuenta que las columnas que el usuario selecciona del menú emergente adquieren mayor prioridad que las asignadas por el desarrollador, por lo tanto, las columnas seleccionadas manualmente se mantendrán visibles en todos los anchos de pantalla hasta que la página se actualice.

```
        <th data-priority ="6">Trimestre 4</th>
        <th >Total</th>
    </tr>
</thead>

    <tbody>

        <tr>
          <td>2014</td>
          <td>Norte</td>
          <td>Vino Blanco</td>
          <td >21393</td>
          <td >19872</td>
          <td >14465</td>
          <td >11111</td>
          <td >66841</td>
      </tr>
        <tr>
          <td>2014</td>
          <td>Este</td>
          <td>Vino Blanco</td>
          <td >17775</td>
          <td >12258</td>
          <td >21889</td>
          <td >18877</td>
          <td >70799</td>
      </tr>
        <tr>
          <td>2014</td>
          <td>Sur</td>
          <td>Vino Blanco</td>
          <td >13374</td>
          <td >10786</td>
          <td >16439</td>
          <td >17473</td>
          <td >58072</td>
      </tr>
```

```
    </tbody>

</table>
```

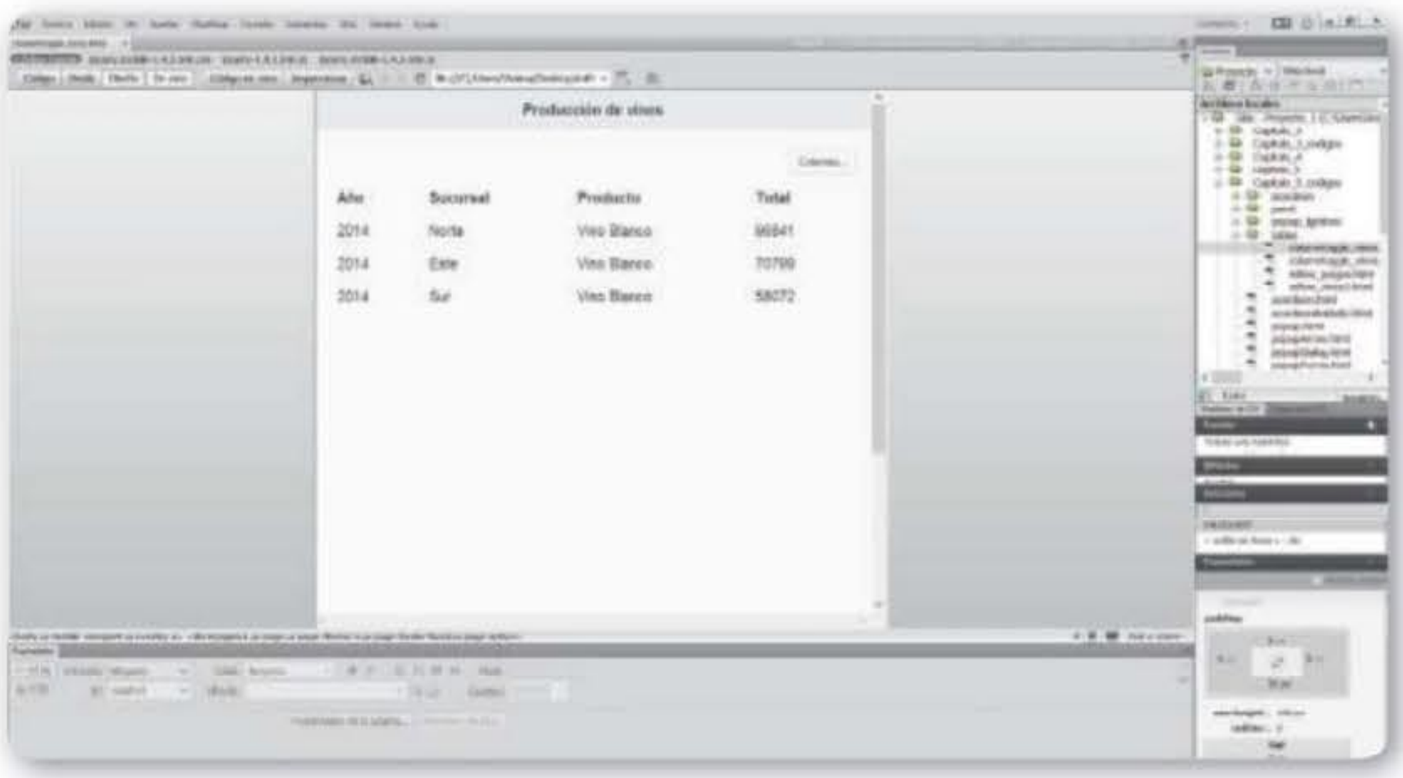


Figura 2. El modo **`column toggle`** es ideal para tablas que necesitan mantener la comparación de datos entre columnas y filas.

Como ya hemos mencionado, jQuery Mobile crea en forma automática el botón emergente y lo ubica en la parte superior derecha de la tabla. Por defecto el título del botón es **Columns...**.

Podemos personalizar el texto del botón utilizando el atributo **data-column-btn-text**. Este atributo tenemos que agregarlo dentro de la etiqueta **<table>**.

Por ejemplo, si queremos que el botón muestre el texto **Más...**, utilizamos la siguiente sintaxis:

```
<table data-role="table" data-mode="columntoggle" class="ui-responsive"
id="mi_tabla" data-column-btn-text="Más..." >
```



ETIQUETA "TFOOT"

La etiqueta **<tfoot>** forma parte del conjunto de etiquetas semánticas de HTML5. Se emplea para definir el pie de una tabla. Agrupa el contenido del cuerpo de la tabla formando así los resúmenes por columna. Esta etiqueta debe colocarse al final de la tabla, antes de la etiqueta **</table>**.



Figura 3. Haciendo un clic sobre la casilla de verificación, seleccionamos la columna por visualizar.

Componentes desplegables

Otra forma de mostrar información en espacios reducidos, como por ejemplo los dispositivos móviles, es mediante los **componentes desplegables** (en inglés, *collapsible*).

Los componentes desplegables poseen en su cabecera un título que, al tocarlo, permite contraer o expandir la información, es decir, muestra u oculta un panel con contenido.

Para crear un componente desplegable, lo que tenemos que hacer es agregarle, a una etiqueta ** **, el atributo **data-role** y asignarle el valor **"collapsible"**. Además tenemos que crear una **barra de título**; para esto, empleamos una etiqueta **head** (puede ser cualquiera de **h1** a **h6**).

Por último, para agregar contenido podemos hacerlo con alguna etiqueta HTML ** ** (párrafo) o cualquier otra, de acuerdo con lo que necesitemos mostrar (una imagen, texto, algún link o botón).

El código para un desplegable básico es el siguiente:

```
<div data-role="collapsible">
   <h4>Título</h4>
   <p>Esto es el contenido de un componente desplegable.</p>
</div>
```



Figura 4. Para especificar el tamaño de los botones podemos usar también el atributo **`data-mini`**.

Por defecto, se agregará a la izquierda del título un icono con el signo **+** (más), para informarle al usuario que debe presionarlo para acceder al contenido. A su vez, al mostrar el contenido, este icono cambia por el de **–** (menos), para informarle al usuario que debe presionarlo para contraer el contenido.

Si queremos que estos iconos se muestren a la derecha, lo que tenemos que hacer es agregarle el atributo **data-iconpos** y asignarle el valor **"right"**.

```
<div data-role="collapsible" data-collapsed="true"
data-iconpos="right">
```

Acordeón (collapsible-set)

Si agrupamos varios componentes desplegables dentro de un **div data-role="collapsible-set"**, estamos creando un **acordeón**.

Al estar compuesto por varios componentes desplegables unidos y relacionados entre sí, cuando abrimos uno, todos los demás se cerrarán, produciendo el efecto clásico de un acordeón.

> LOS ELEMENTOS DESPLEGABLES PUEDEN UNIRSE Y VISUALIZARSE EN FORMA DE ACORDEÓN

Veamos un ejemplo para explorar sus posibilidades:

```
<div data-role="collapsible-set">
 <div data-role="collapsible">
    <h3>Cataratas del Iguazú</h3>
    <p>En la provincia de Misiones, se encuentran las enormes Cataratas del
Iguazú, que además de ser una maravilla de la naturaleza poseen...</p>
    </div>

    <div data-role="collapsible">
      <h3>Bariloche</h3>
      <p>Bariloche ofrece opciones imperdibles: la caminata del Cerro Otto, el
Circuito Chico... </p>
    </div>

    <div data-role="collapsible" data-collapsed="true">
      <h3>El Calafate</h3>
      <p>El Parque Nacional Los Glaciares guarda uno de los tesoros de la Argen-
tina....</p>
    </div>
  </div>
```

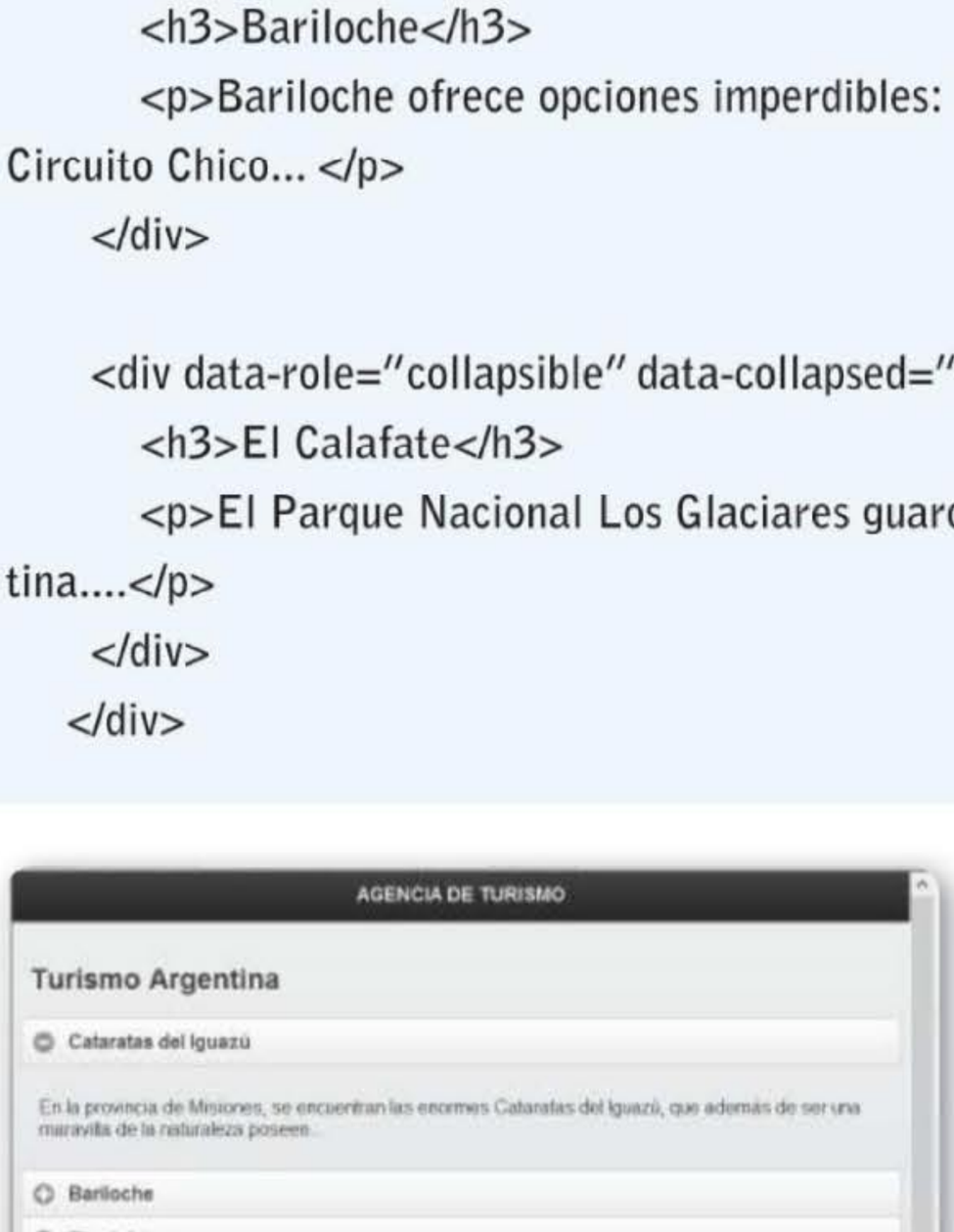


Figura 5. El acordeón también puede combinarse con una lista. De esta forma, al hacer clic sobre un ítem, se abrirá y mostrará más opciones.

Por defecto, el acordeón aparece contraído y muestra solo el título y el signo **+**; pero, si queremos que al abrir la página aparezca directamente el contenido de un componente desplegado, tenemos que agregarle el atributo **data-collapsed** y pasarle el valor **"false"**.

Otro atributo que puede servir para cambiar el aspecto del acordeón es el que saca las puntas redondeadas y hace que ocupe todo el ancho de pantalla. Para esto, al atributo **data-inset** le pasamos por valor **"false"**.

Pestañas (tabs)

Una forma muy útil, que está de moda en la mayoría de los sitios web y nos sirve para organizar la información, son los **tabs** o también llamadas **pestañas**.

Las pestañas son de mucha ayuda para los usuarios, ya que nos permiten, a los programadores, mantener grupos y subgrupos de información de manera organizada, facilitando así una mejor disposición de la información en espacios reducidos.

Para crear este componente, tenemos que utilizar la etiqueta ** ** con el atributo **data-role** y pasarle el valor **"tabs"**.

```
<div data-role="tabs" id="tabs">
```

De esta forma, le estamos diciendo al navegador que el contenido incluido va a ser una pestaña.

Luego, para crear las pestañas, lo que tenemos que hacer es agregarle, al ** ** anterior, otro ** ** que va a tener el atributo **data-role** con el valor **"navbar"** y, dentro de este componente,

BLOQUE DESPLEGABLE EN DREAMWEAVER

Podemos crear un componente desplegable en Dreamweaver ubicando el cursor en la sección contenido (**data-role="content"**). Luego hacemos clic en el menú **Insertar/jQuery Mobile** y seleccionamos **Bloque que puede contraerse**. A diferencia de otros objetos de jQuery Mobile, no aparecerá ningún cuadro de diálogo. Dreamweaver creará automáticamente el código HTML de un bloque desplegable con tres secciones.

incluiremos una lista desordenada (**ul**) donde cada elemento de la lista (**li**) contendrá un link.

Debemos tener presente que cada ítem es una pestaña, y cada pestaña debe tener un enlace a su contenido del tipo **Tab 1**, por ejemplo.

Por defecto, cuando llamamos de una página a otra, se aplica un efecto de transición. Es recomendable desactivar los efectos de transición empleando el atributo **data-ajax="false"**, como vimos en el **Capítulo 2,** porque lo que estamos haciendo es pasar de un enlace a otro cuando tocamos cada pestaña como si fuera un menú de navegación. Entonces, al quitarle la transición, no se notaría ese llamado a otro enlace, sino que parecería que se tratase de un solo objeto o componente.

En el siguiente ejemplo, vemos cómo mostrar productos de una heladería: en la primera pestaña se pueden ver sabores de helados; en la segunda, postres, y en la tercera, salsas; de esta forma tenemos bien definidos los grupos y subgrupos.

El código que nos permite representar el ejemplo es el siguiente:

```
<div data-role="tabs" id="tabs">
                <div data-role="navbar">
        <ul>
         <li><a href="#helados" data-ajax="false">Helados</a></li>
         <li><a href="#postres" data-ajax="false">Postres Helados</a></li>
         <li><a href="#salsas" data-ajax="false">Salsas</a></li>
                </ul>
                </div>
                <div id="helados">
        <h3>Nuestros sabores de helados</h3>

        <ul data-role="listview" data-inset="true">
                <li>Frutilla</li>
                <li>Dulce de Leche</li>
                <li>Chocolate</li>
                <li>Sambayón</li>
        </ul>
       </div>
```

```
<div id="postres">
                    <h3>Nuestros Postres Helados</h3>

    <ul data-role="listview" data-inset="true">
                    <li>Almendrado</li>
        <li>Tricolor</li>
        <li>Selva helada</li>
        <li>Chajá</li>
     </ul>
    </div>

<div id="salsas">
   <h3>Nuestras Salsas</h3>

          <ul data-role="listview" data-inset="true">
      <li>Dulce de leche</li>
      <li>Frutos rojos</li>
      <li>Frambuesa</li>
      <li>Frutilla</li>
    </ul>
</div>
   </div>
```

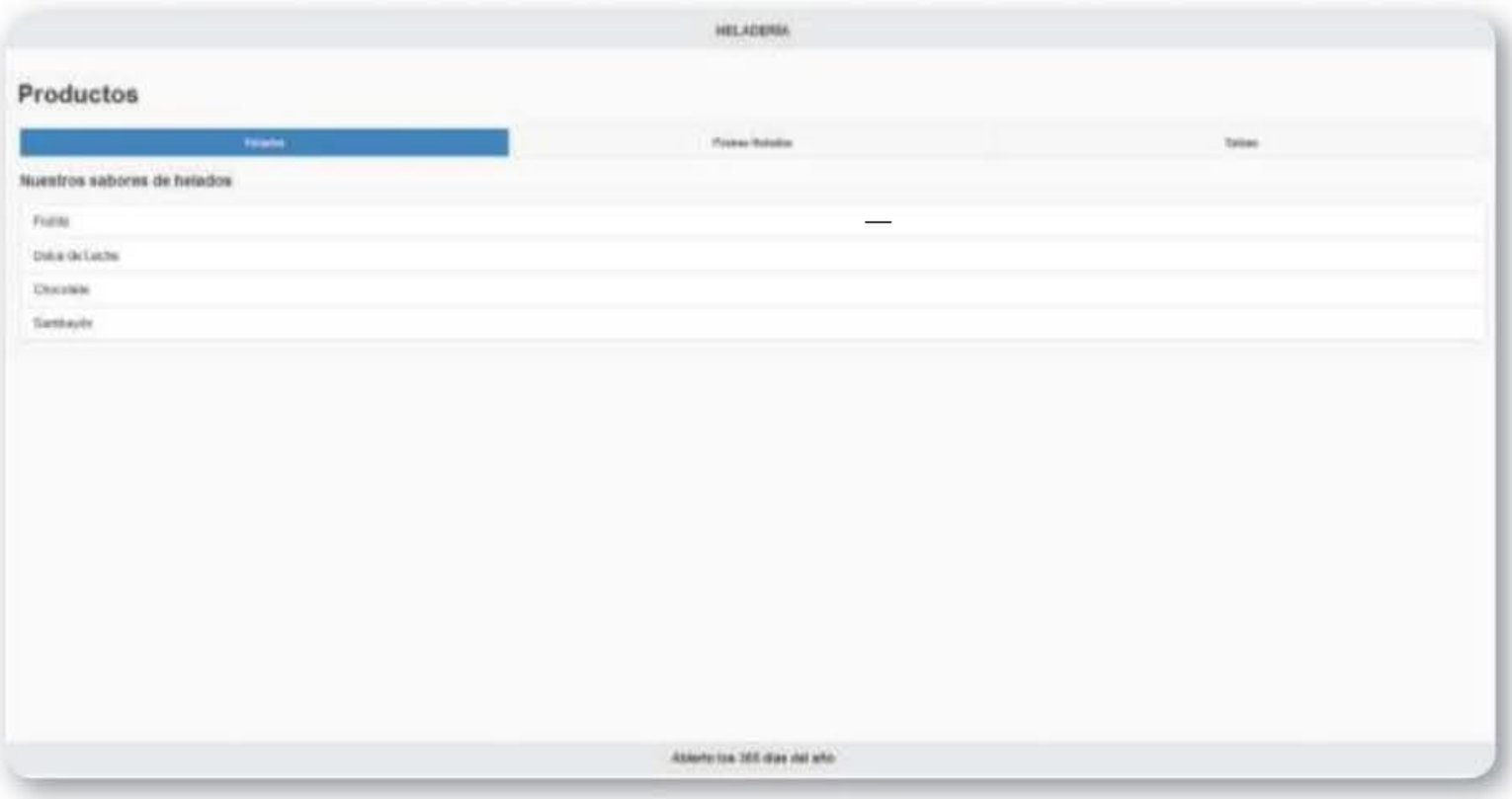


Figura 6. Visualizamos el contenido de cada una de las pestañas cuando tocamos sobre cada una de ellas.

Ventanas emergentes (pop-up)

Las **ventanas emergentes** (en inglés, *pop-up*), o también llamadas **cuadros de diálogo**, quizás sean, junto con las listas, los elementos más utilizados en los dispositivos móviles o equipos de escritorio.

Sirven para informar al usuario, alertarlo, ayudarlo o preguntarle sobre alguna decisión, y son ventanas generalmente de menor tamaño que el documento.

También incluyen un botón que le va a permitir al usuario cerrar el cuadro de diálogo para volver a la aplicación.

Distintos tipos

Los pop-up siempre tienen que estar dentro de las etiquetas de contenido ** **.

Tienen varios usos y, de acuerdo con esto, diferentes formas de configurarlos y, por lo tanto, de dividirlos en varios grupos o categorías, que son los siguientes:

- Básico
- Mensaje de ayuda (*tooltip*)
- Visualizador de imágenes (*photo lightbox*)
- Menú
- Formulario
- Cuadro de diálogo

Básico

El pop-up básico solo muestra un mensaje al usuario. Para crear un pop-up básico, tenemos que agregarle a un ** ** el atributo **data-rel** y pasarle el valor **"popup"**, como se puede ver en el siguiente código:

PARA VERSIONES ANTERIORES

En versiones anteriores del framework se utilizaban cuadros de diálogo agregando el atributo **data-rel="dialog"**. En la versión 1.4.2 (actual hasta la fecha de edición de este libro), se unifican los cuadros de diálogos y pop-up, pasando a ser un único grupo "Pop-up".

```
<div data-role="content">
        <h2>POPUP</h2>

    <a href="#popupBasico" data-rel="popup" data-role="button" >Popup básico</a>

    <div data-role="popup" id="popupBasico">
        <p>Esto es un popup básico.</p>
    </div>
</div>
```

Si analizamos el código anterior, vemos que, como ya mencionamos, el pop-up está contenido entre las etiquetas ** ** de un **content**, y generalmente es llamado por un botón o una determinada acción del usuario que debemos programar; en este ejemplo, usamos una etiqueta HTML de enlace ** **, que genera un botón de jQuery Mobile con el atributo **data-role="button"**.

También le decimos que queremos que el contenido del enlace se muestre como una ventana emergente con el código **data-rel="popup"**, y ese enlace es a una página que llamamos **popupBasico**. Dicha página está creada más abajo entre las etiquetas ** ** con el atributo **data-role="popup"** y con un identificador (**id**) que debe tener el mismo nombre de la referencia del enlace anterior (en nuestro caso, **popupBasico**), para que pueda ser llamada.

> EL POP-UP BÁSICO SOLO MUESTRA UN MENSAJE O CONTENIDO AL USUARIO

En resumen, necesitamos: por un lado, un botón o acción del usuario que llame o ejecute el pop-up; y, por otro lado, crear el pop-up con el mensaje o contenido para que sea visualizado por el usuario.

TECLAS RÁPIDAS

En Dreamweaver, para manejarnos con el teclado, podemos emplear la combinación de teclas **ALT** + **I** para abrir el menú **Insertar** y, luego, para abrir el submenú **jQuery Mobile** presionamos la tecla **J**. Pulsamos la tecla **C** para elegir la opción **Cuadrícula de diseño** y la tecla **Q** para seleccionar la opción **Bloque que puede contraerse**.



Figura 7. Al tocar el botón, aparece una ventana emergente con información para el usuario.

Mensaje de ayuda (tooltip)

Un **tooltip** (mensaje de ayuda), o también llamado **etiqueta emergente**, nos sirve para ayudar al usuario mostrando contenido adicional que se visualiza al pasar el cursor sobre un elemento (en el caso de las computadoras de escritorio) o pulsando sobre un icono (en el caso de los dispositivos móviles).

Se crea de la misma forma que un pop-up básico, solo que necesitamos agregarle un icono y estilo con el atributo **class="my-tooltip-btn ui-btn ui-alt-icon ui-nodisc-icon ui-btn-inline ui-icon-info ui-btn-icon-notext"**, que ubica el icono de información que trae el framework, al lado del párrafo.

En el siguiente ejemplo mostramos la sintaxis de un tooltip:

```
<div data-role="content">
        <h2>POPUP</h2>

        <p>Esto es un párrafo con tooltip. <a href="#popup" data-rel="popup"
class="my-tooltip-btn ui-btn ui-alt-icon ui-nodisc-icon ui-btn-inline ui-icon-info ui-
btn-icon-notext" title="Leer más">Leer más</a></p>

        <div data-role="popup" id="popup">
                <p>Este es el contenido del tooltip</p>
        </div>
</div>
```

ETIQUETAS PARA TABLAS

Otra etiqueta que podemos utilizar cuando creamos una tabla en HTML es **<caption>**. Esta etiqueta se emplea para asignarle un nombre general a la tabla, es decir, que defina el título de la tabla. Debe emplearse al comienzo de la tabla, después de la etiqueta **<table>**.



Figura 8. Cambiando el valor de la clase **`ui-icon`**, se le puede asignar otro icono, como por ejemplo: **`class="ui-icon-clock"`**.

Visualizador de imágenes (photo lightbox)

Un **visualizador de imágenes** (en inglés, *photo lightbox*, 'caja de iluminación para imágenes') es una galería con varias imágenes, que, al seleccionar una, la amplía y la ubica en el centro de la pantalla con un fondo gris traslúcido para que se pueda visualizar mejor.

Lo primero que tenemos que tener en cuenta son las imágenes que vamos a utilizar para mostrar. Estas imágenes debemos tenerlas en una carpeta a la que se pueda acceder dentro del proyecto y que además no posean una resolución muy alta, ya que esto nos haría demorar la carga de la página.

Y también debemos considerar que cada una de las imágenes es un pop-up diferente, es decir, que tendríamos un enlace por cada foto, como se puede observar en el siguiente código de ejemplo:

```
<div data-role="content">
        <h2>Visualizador de fotos</h2>

        <a href="#popupIMG_001" data-rel="popup" data-position-
to="window" data-transition="fade"><img class="popphoto" src="img/IMG_001.
JPG" style="width:30%"></a>

        <a href="#popupIMG_002" data-rel="popup" data-position-
to="window" data-transition="fade"><img class="popphoto" src="img/IMG_002.
JPG" style="width:30%"></a>

        <a href="#popupIMG_003" data-rel="popup" data-position-
to="window" data-transition="fade"><img class="popphoto" src="img/IMG_003.
JPG" style="width:30%"></a>

        <div data-role="popup" id="popupIMG_001" data-overlay-
theme="b" >
```

```
                    <a href="#" data-rel="back" class="ui-btn ui-corner-all ui-
shadow ui-btn-a ui-icon-delete ui-btn-icon-notext ui-btn-right">Cerrar</a><img
class="popphoto" src="img/IMG_001.JPG" style="max-height:512px;" >
                    </div>

                    <div data-role="popup" id="popupIMG_002" data-overlay-
theme="b" >
                    <a href="#" data-rel="back" class="ui-btn ui-corner-all ui-
shadow ui-btn-a ui-icon-delete ui-btn-icon-notext ui-btn-right">Cerrar</a><img
class="popphoto" src="img/IMG_002.JPG" style="max-height:512px;" >
                    </div>

                    <div data-role="popup" id="popupIMG_003" data-overlay-
theme="b" >
                    <a href="#" data-rel="back" class="ui-btn ui-corner-all ui-
shadow ui-btn-a ui-icon-delete ui-btn-icon-notext ui-btn-right">Cerrar</a><img
class="popphoto" src="img/IMG_003.JPG" style="max-height:512px;" >
                    </div>

    </div>
```

En este ejemplo, vemos, en primer lugar, que definimos el pop-up con una imagen para que, al tocarla, llame a la página que contendrá la foto más grande, le damos una posición centrada con el atributo **data-position-to="window"** y un efecto de transición con el atributo **data-transition="fade"** (más adelante en este capítulo aprenderemos sobre transiciones y posiciones del pop-up).

Y con respecto a la imagen, le damos un ancho del 30 % del valor original con **style="width:30%"**; de esta manera, no necesitamos recortar

IMÁGENES

Para acceder a una imagen o un archivo desde código HTML, se utiliza el atributo **src**, por ejemplo **src="ruta del archivo"**. Este archivo debe estar en la carpeta del proyecto, junto con el archivo principal (index). En este caso, la ruta sería: **src="miArchivo"**. En cambio, si está en un directorio posterior al principal, la ruta será **src="carpeta/miArchivo"**, y, si está en un directorio anterior al proyecto, **src="../carpeta/miArchivo"**. Es decir con **../** para un directorio anterior y **/** para uno posterior.

la imagen o tener dos, una para la previsualización y otra de tamaño completo, ya que solamente indicándole el porcentaje nos la reduce.

En segundo lugar, al ** ** que nos va a mostrar la imagen en tamaño completo al ser seleccionada, le agregamos **data-role="popup"**, el identificador (**id**) para que sea llamado y el atributo **data-overlay-theme="b"**, que nos va a dar el color gris traslúcido de fondo.

Para terminar, le agregamos en la esquina superior derecha un botón de cerrar (**X**), que nos hace volver a las imágenes previsualizadas.

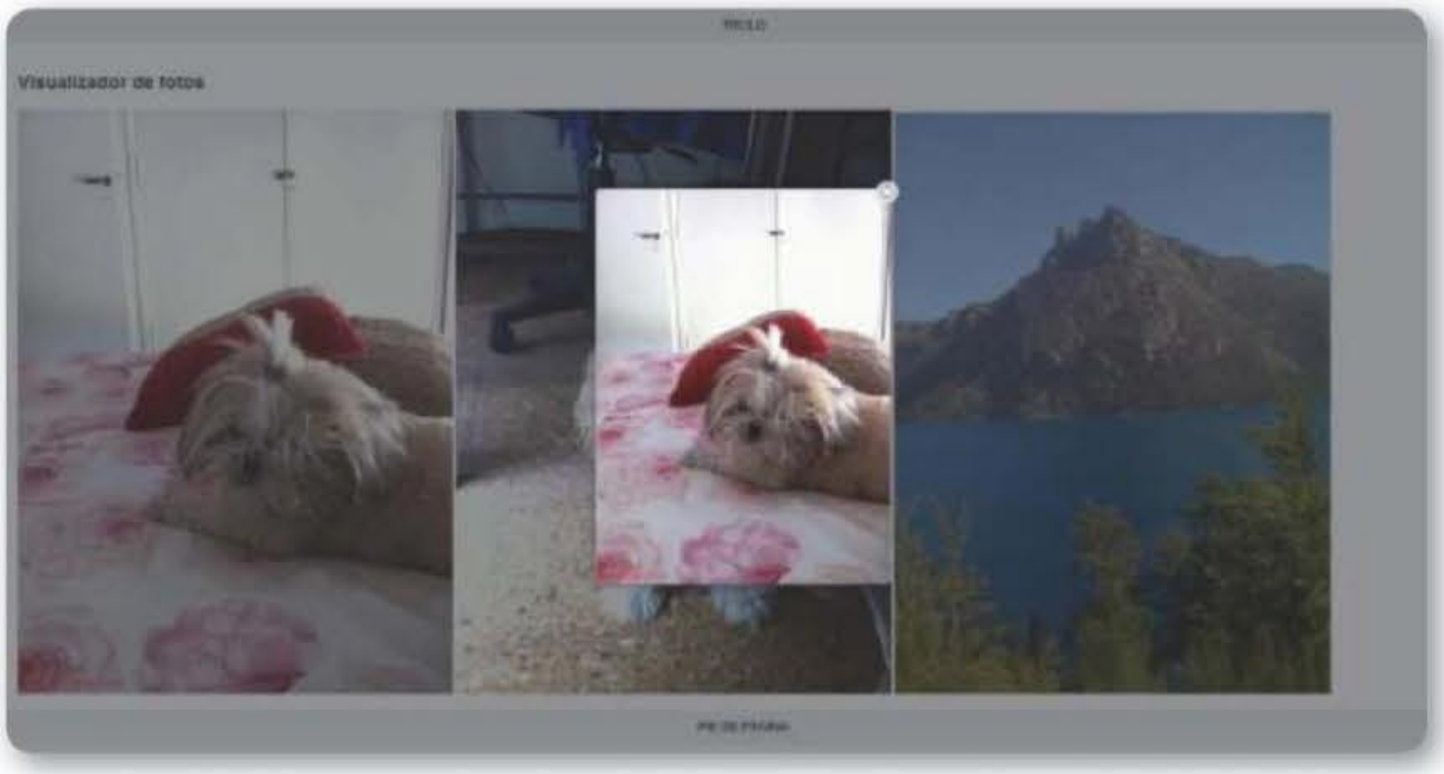


Figura 9. El visualizador de imágenes (**photo lightbox**) es ideal cuando tenemos pocas imágenes para mostrar.

Menú

Otra opción interesante es crear un **menú** mediante pop-up y que en su interior tenga una lista.

Esto es tan simple de resolver que solamente basta con agregarle al ** ** que se va a mostrar (**data-role="popup"**) una lista con sus correspondientes enlaces, tal cual se muestra en el siguiente código de ejemplo:

```
    <a href="#popupMenu" data-rel="popup" data-transition="slideup" class="ui-
btn ui-corner-all ui-shadow ">Ver...</a>

        <div data-role="popup" id="popupMenu" >

            <ul data-role="listview" data-inset="true" style="min-
```

```
width:210px;">
            <li data-role="list-divider">Elija una opción </li>
            <li><a href="#">Editar</a></li>
            <li><a href="#">Copiar</a></li>
            <li><a href="#">Pegar</a></li>
            <li><a href="#">Borrar</a></li>
                </ul>
</div>
```



Figura 10. Con el atributo **`data-shadow="false"`**, podemos quitar la sombra alrededor del pop-up.

Formularios en pop-up

Podemos incluir también dentro de un pop-up un formulario HTML como los que hemos visto en el **Capítulo 4**.

Por ejemplo, si necesitamos realizar una página cuyo acceso se restrinja a usuarios previamente registrados, en ese caso podemos crear un pop-up que, en su interior, contenga un formulario con dos campos de texto (text input), que solicite completar con nombre de usuario y contraseña para entrar, y un botón para ingresar al sistema. Podemos observar este ejemplo en el código que sigue:

```
    <div data-role="content">
        <a href="#popupLogin" data-rel="popup" data-position-to="window"
class="ui-btn ui-corner-all ui-shadow ui-btn-inline ui-icon-check ui-btn-icon-left ui-btn-
```

MENÚ ANIDADO (NESTED MENU)

Si queremos hacer más interesante un menú, podemos también incluir dentro del ** ** del pop-up un acordeón (**collapsible set**) que contenga, a su vez, una lista (**listview**). De esta manera, quedaría formado un **menú anidado**, muy parecido a los que podemos ver en una página web de escritorio.

```
a" data-transition="pop">Ingresar</a>
        <div data-role="popup" id="popupLogin" >
        <form>
          <div style="padding:10px 20px;">
            <h3>Complete la información para ingresar al sistema</h3>
            <label >Usuario:</label>
            <input name="user" id="un" value="" type="text">
            <label>Contraseña:</label>
            <input name="pass" id="pw" value="" type="password">
            <button type="submit"> Ingresar</button>
          </div>
        </form>
    </div>
    </div>
```

Si analizamos el código anterior, podemos observar que, dentro de los ** data-role= "popup"**, se incluye el formulario.



Figura 11. Podemos incluir todos los elementos de los que dispone el framework para formularios de acuerdo a nuestra necesidad.

Cuadros de diálogo

El **cuadro de dialogo** nos sirve para tomar alguna decisión sobre el comportamiento del programa. Se trata de una ventana pop-up que, en su interior, tiene botones y muestra una pregunta o advertencia al usuario sobre una determinada acción que va a realizar el sistema de acuerdo a la elección que este haga, y que nosotros como programadores debemos programar.

Por ejemplo, si tenemos que informarle al usuario que ha actualizado sus datos personales, suponiendo que cambió algo de su información personal, le mostraríamos un cuadro de diálogo, que no es más que una ventana pop-up que se ubica en el centro de la pantalla con el mensaje **Se han actualizado sus datos correctamente**. Esta ventana también tendría un botón **Aceptar** que nos permite cerrarla para volver a la página.



Figura 12. Cuadro de diálogo básico con un botón **Aceptar** e información para el usuario.

Para configurar el cuadro de diálogo, deberemos tener presente que necesitamos agregarle un título, un mensaje y el o los botones con los que contará para poder programar las acciones o decisiones que toma el sistema de acuerdo a la elección del usuario.

El título del cuadro de diálogo se codifica agregándole, al ** **, otro ** ** con el atributo **data-role** y pasándole el valor **"header"**. Recomendamos poner el título entre los atributos **<h1></h1>** para destacarlo del resto, como lo hacemos en HTML.

Después le agregamos otro ** ** para el contenido con la clase **class="ui-content"** y el atributo **role** que le pasamos el valor **"main"**.



WIDGETS

En programación, el término **widget** se emplea para hacer referencia a los componentes gráficos mediante los que el usuario puede interactuar con la aplicación y el sistema operativo. Los widgets pueden tomar la forma de un botón, un cuadro de diálogo, una ventana emergente, un menú desplegable, un acordeón, entre otros.

El mensaje se configura, por ejemplo, con un título del tipo **<h3></h3>** con el mensaje o pregunta para el usuario.

También, de manera optativa, podemos agregarle un comentario que puede ir entre las etiquetas ** **. Y, por último, los botones o el botón del tipo ****.

Todo esto lo podemos apreciar en el siguiente ejemplo, en donde se le pregunta al usuario si desea borrar un determinado contenido:

```
<a href="#popupDialog" data-rel="popup" data-position-to="window" data-
transition="pop" class="ui-btn ui-corner-all ui-shadow ui-btn-inline ui-icon-delete
ui-btn-icon-left ">Borrar registro</a>

<div data-role="popup" id="popupDialog" style="max-width:400px;">

    <div data-role="header" >
        <h1>Eliminar</h1>
    </div>

    <div role="main" class="ui-content">
        <h3 class="ui-title">¿Desea borrar el registro actual?</h3>
        <p>Esta acción no puede volver atrás.</p>
        <a href="#" class="ui-btn ui-corner-all ui-shadow ui-btn-inline "
data-rel="back">Cancelar</a>
        <a href="#" class="ui-btn ui-corner-all ui-shadow ui-btn-inline "
data-rel="back" data-transition="flow">Eliminar</a>
    </div>
</div>
```



Figura 13. Los cuadros de diálogo son llamados así porque permiten una comunicación entre el sistema y el usuario.

Cerrar un pop-up

Por defecto, si tocamos fuera del pop-up, este se cierra. En caso de que necesitemos evitar esto, solo tenemos que agregarle el atributo **data-dismissible** y pasarle el valor **"false"**. De esta forma, nos aseguramos de que el usuario toque los botones de cerrar o cancelar. Entonces, para los botones de cerrar o cancelar, lo que necesitamos es agregarle al botón el atributo **data-rel** y pasarle el valor **"back"**. Este atributo hace volver al estado anterior, o sea, a la página de donde fue llamado.

```
<a href="#" data-rel="back">Cancelar</a>
```

Con respecto al icono de cruz que indica la acción de "cerrar", típico de las ventanas de Windows, también podemos crearlo en jQuery Mobile. Es bastante simple: solo tenemos que agregarle la clase correspondiente, el icono (**ui-icon-**) y la ubicación, ya sea izquierda (**ui-btn-left**) o derecha (**ui-btn-right**).

El siguiente código nos muestra dos botones con la cruz de cerrar, uno a la derecha y otro a la izquierda.

```
    <a href="#popupCerrarDerecha" data-rel="popup" class="ui-btn ui-corner-all ui-
shadow ui-btn-inline">Botón de cerrar a la derecha</a>

    <a href="#popupCerrarIzquierda" data-rel="popup" class="ui-btn ui-corner-all
ui-shadow ui-btn-inline">Botón de cerrar a la izquierda</a>

    <div data-role="popup" id="popupCerrarDerecha" class="ui-content" style="max-
width:280px">
        <a href="#" data-rel="back" class="ui-btn ui-corner-all ui-shadow ui-btn-a ui-
icon-delete ui-btn-icon-notext ui-btn-right">Close</a>
            <p>Este botón tiene el ícono de cerrar a la derecha.</p>
    </div>

    <div data-role="popup" id="popupCerrarIzquierda" class="ui-content"
style="max-width:280px">
            <a href="#" data-rel="back" class="ui-btn ui-corner-all ui-shadow ui-btn-a
ui-icon-delete ui-btn-icon-notext ui-btn-left">Close</a>
            <p>Este botón tiene el ícono de cerrar a la izquierda.</p>

    </div>
```



Figura 14. También podemos agregarle un efecto de transición al pop-up con el atributo **data-transition**.

Posición

La posición por defecto de un pop-up es centrada en el medio del botón o elemento que fue llamado. Si en cambio necesitamos que el pop-up esté centrado en el medio de la pantalla, empleamos el atributo **data-position-to** al que le pasamos por valor **"window"**. Y si queremos que el pop-up se abra sobre el elemento del que fue llamado, al atributo **data-position-to**, le asignamos el valor **"origin"**.

> EL ATRIBUTO DATA-POSITION-TO PERMITE QUE MODIFIQUEMOS LA POSICIÓN DEL POP-UP

También es posible especificar una posición de acuerdo a la ubicación de un elemento HTML; por ejemplo, si queremos que el pop-up se ubique en la posición del header de nuestra página, tenemos que pasarle al atributo **data-position-to** el valor **"#position-header"**.

El siguiente código ilustra el uso del atributo **data-position-to**:

```
<a href="#centrado" class="ui-btn ui-corner-all ui-shadow ui-btn-inline" data-
rel="popup" data-position-to="window">Posición centrada</a>

<a href="#origen" class="ui-btn ui-corner-all ui-shadow ui-btn-inline" data-
rel="popup" data-position-to="origin">Posición sobre el origen</a>

<a href="#selectiva" class="ui-btn ui-corner-all ui-shadow ui-btn-inline" data-
rel="popup" data-position-to="#position-header">Posición selectiva</a>

<div data-role="popup" id="centrado" class="ui-content" data-theme="a">
    <p>Centrado</p>
</div>

<div data-role="popup" id="origen" class="ui-content" data-theme="a">
```

```
        <p>Sobre el origen</p>
</div>

<div data-role="popup" id="selectiva" class="ui-content" data-theme="a">
        <p>Selectiva sobre el header</p>
</div>
```

Transiciones

Las transiciones que soporta el pop-up son las mismas que explicamos en el **Capítulo 2** cuando hablábamos de enlazar una página con otra. Lo que tenemos que tener en cuenta –por recomendación de los autores del framework– es utilizar las transiciones comunes a la mayoría de los dispositivos móviles (pop, fade) o ninguna, lo que es conveniente por motivos de rendimiento, ya que se utilizan muchos recursos, y eso puede provocar que la página no funcione bien o esté lenta.

Por defecto, el pop-up no utiliza ninguna transición, para hacer más rápida su aparición. Pero si queremos agregarle alguna, al botón o enlace desde donde se la va a llamar le agregamos el atributo **data-transition** y le pasamos el valor de la transición que queramos, como por ejemplo **data-transition="fade"**.

En el siguiente código de ejemplo podemos ver todas las transiciones soportadas por el framework:

```
    <a href="#transicion" data-transition="none" class="ui-btn" data-
rel="popup">Sin transición</a>
    <a href="#transicion" data-transition="pop" class="ui-btn"  data-
rel="popup">Pop</a>
    <a href="#transicion" data-transition="fade" class="ui-btn" data-
rel="popup">Fade</a>
    <a href="#transicion" data-transition="flip" class="ui-btn" data-
rel="popup">Flip</a>
    <a href="#transicion" data-transition="turn" class="ui-btn" data-
rel="popup">Turn</a>
    <a href="#transicion" data-transition="flow" class="ui-btn" data-
rel="popup">Flow</a>
    <a href="#transicion" data-transition="slide" class="ui-btn" data-
rel="popup">Slide</a>
```

```
    <a href="#transicion" data-transition="slidefade" class="ui-btn" data-
rel="popup">Slidefade</a>
    <a href="#transicion" data-transition="slideup" class="ui-btn" data-
rel="popup">Slide up</a>
    <a href="#transicion" data-transition="slidedown" class="ui-btn" data-
rel="popup">Slide down</a>

    <div data-role="popup" id="transicion" >
    <p>Ejemplo de transición.</p>
    </div>
```

Flechas (arrows)

Otra opción interesante y vistosa en el diseño de las ventanas emergentes es agregarles una flecha en uno de sus bordes, para hacerle notar al usuario de dónde sale el pop-up o cuál es su origen.

Para lograr esto, le agregamos al **div** del **data-role="popup"** el atributo **"data-arrow"**, al que le asignamos como valor **"true"**, como vemos en el siguiente ejemplo:

```
    <a href="#flecha" class="ui-btn" data-rel="popup">Toque el botón para ver el
pop-up</a>

    <div data-role="popup" id="flecha" data-arrow="true">
        <p>Esto es un pop-up con flecha o arrow.</p>
    </div>
```



Figura 15. Para que la flecha del pop-up apunte hacia abajo, usamos el atributo **data-arrow="b"**.

Paneles

Por lo general, se los emplea para crear barras de navegación, mostrar un formulario o visualizar información adicional. Cuando el usuario presiona, por ejemplo, un botón o un enlace, los paneles se abren, ya sea a la izquierda o a la derecha de la pantalla, superponiéndose a ella o deslizando la página principal. Debemos crear el panel antes o después de las secciones de una página jQuery Mobile: **header**, **content** y **footer**, pero nunca en el medio de ellas. Para crear este componente, usamos el atributo **data-role="panel"**. Además, debemos asignarle un identificador (**id**) que nos servirá para luego hacer el vínculo que nos va abrir el panel.

La sintaxis básica del panel es la siguiente:

```
<div data-role="panel" id="mi_panel">
        <!-- Acá ponemos el contenido del panel-->
</div>
```

Como mencionamos antes, para abrir el panel tenemos que crear un enlace que llame al **id** del panel. Este enlace puede estar en cualquiera de las secciones de la página, es decir, lo podemos incluir ya sea dentro del encabezado o en el pie de página, o dentro de la sección de contenido.

Por ejemplo, si el **id** del panel es **mi_panel**, la sintaxis para abrir el panel mediante un botón es la siguiente:

```
<a href="#mi_panel" data-role="button">Abrir</a>
```

Por defecto, el panel se abrirá en el lado izquierdo de la pantalla. Sin embargo, podemos hacer que el panel aparezca desde el borde derecho de la pantalla; para lograr esto, usamos el atributo **data-position** al que le asignamos el valor **"rigth"**, como mostramos a continuación.

TABS DE JQUERY UI

Podemos crear un control tabs en Dreamweaver ubicando el cursor en la sección contenido (**data-role="content"**). Luego hacemos clic en el menú **Insertar/jQuery UI** y seleccionamos **Tabs**. Automáticamente, Dreamweaver nos genera el código necesario para crear un componente con tres pestañas.

```
<div data-role="panel" id="mi_panel" data-position="rigth">
        <!-- Acá ponemos el contenido del panel-->
</div>
```

Veamos un ejemplo más para explorar sus posibilidades. El siguiente código muestra un panel que presenta un menú que se abre a la izquierda de la pantalla:

```
<body>

<div data-role="page" id="nosotros">

   <div data-role="panel" id="mi_panel">
     <ul data-role="listview">
        <li><a href="#nosotros">Nosotros</a></li>
        <li><a href="productos.html">Productos</a></li>
        <li><a href="locales.html">Locales</a></li>
     </ul>
   </div>

        <div data-role="header">
                    <h1>Heladería Paco</h1>
      <a href="#mi_panel" data-role="button">Abrir</a>
      </div>

      <div data-role="content">
       <h2>El sabor original</h2>
    <p> Contamos con 50 años de experiencia en la fabricación de cremas heladas
utilizando materias primas naturales de altísima calidad.</p>
      </div>

       <div data-role="footer" data-position="fixed">
     <h1> Visítenos los 365 días del año </h1>
    </div>

</div>

</body>
```



Figura 16. En este ejemplo, creamos un menú que llama a páginas externas.

Por defecto, el panel se cierra cuando hacemos clic o tocamos fuera de este, aunque también podemos incluir un botón para cerrar el panel. Para ello, creamos un enlace a la página donde el usuario debe saltar cuando cierre el panel. En este enlace, además, incluiremos el atributo **data-rel="close"**, como mostramos en el siguiente ejemplo:

```
<a href="#pagina1" data-role="button" data-rel="close">Cierra el panel</a>
```

Podemos especificar cómo el contenido del panel debe colocarse de acuerdo al diseño del resto de la página, cuando el usuario se desplaza por ella (scroll). De manera predeterminada, el panel se desplazará con la página, pero su contenido se mantendrá en la parte superior. Para que el contenido del panel se visualice siempre, sin importar lo lejos que se haya desplazado el usuario por la página, agregamos el atributo **data-position-"fixed" ="true"**.

También podemos incluir dentro de una página más de un panel. En el siguiente ejemplo, mostramos una página que da la posibilidad de abrir dos paneles, uno desde el borde izquierdo y otro desde el borde derecho de la pantalla.

```
<div data-role="page" id="pagina1">

 <div data-role="panel" id="panel_izq">
            <h2> Panel izquierdo</h2>
    <p>Este es un ejemplo que abre un panel a la izquierda</p>
     <a href="#pagina1" data-role="button" data-rel="close" >Cerrar</a>
 </div>
```

```
<div data-role="panel" id="panel_der" data-position="right">
            <h2> Panel derecho</h2>
 <p>Este es un ejemplo que abre un panel a la derecha</p>
 <a href="#pagina1" data-role="button" data-rel="close" >Cerrar</a>

</div>

<div data-role="header">
            <h1>Paneles</h1>
    </div>

    <div data-role="content">
            <h2>Este ejemplo permite abrir dos paneles</h2>
   <a href="#panel_izq" data-role="button"> Panel izquierdo</a>
   <a href="#panel_der" data-role="button"> Panel Derecho</a>

    </div>

    <div data-role="footer" data-position="fixed">
            <h4></h4>
    </div>
</div>
```

RESUMEN

En este capítulo, conocimos y aprendimos a utilizar distintos componentes que nos sirven para crear una buena interfaz de usuario. Vimos, primero, los distintos tipos de tablas que podemos crear usando jQuery Mobile. Hemos examinado también los diferentes componentes desplegables que nos permiten visualizar todo el contenido dentro de una única página. Aprendimos a emplear sus principales atributos y, posteriormente, a crear distintos tipos de pop-up que nos permiten mostrar mensajes emergentes para el usuario.

Actividades

TEST DE AUTOEVALUACIÓN

1. ¿Qué etiqueta HTML se utiliza para formar una fila dentro de una etiqueta **table**?
2. ¿Qué valor hay que pasarle al atributo **data-role** para formar un acordeón?
3. ¿Qué valor hay que pasarle al atributo **data-role** para formar una pestaña o tab?
4. ¿Entre qué etiquetas del diseño de una página jQuery Mobile debe ir un pop-up?
5. Cuando utilizamos el atributo **data-role="panel"**, ¿es conveniente asignarle un identificador o id?

EJERCICIOS PRÁCTICOS

1. Realice una tabla con el atributo **<table></table>**, que muestre los datos de una agenda telefónica (nombre, e-mail y número de teléfono).
2. Con los datos del ejercicio anterior, cree un acordeón y muestre los mismos datos, colocando como título el nombre de cada persona.
3. A partir de los ejercicios anteriores (1 y 2), cree un componente pestaña en donde cada una de ellas sea el nombre de cada persona.
4. Al ejercicio anterior, agréguele también un botón sobre cada persona, que, al presionarlo, muestre un pop-up con la foto del contacto.
5. Realice una página que tenga, en la sección contenido, un botón y que, al presionarlo, muestre un panel a la derecha de la pantalla.



PROFESOR EN LÍNEA

Si tiene alguna consulta técnica relacionada con el contenido, puede contactarse con nuestros expertos: profesor@redusers.com

*06

Interfaz de usuario y temas

En este capítulo, veremos temas relacionados con la disposición de contenido; cómo es el uso de grids y cuáles son los estilos propios ya definidos en el framework. Además, aprenderemos cómo cambiar un estilo y personalizarlo usando las herramientas disponibles de jQuery Mobile. Por último, conoceremos ThemeRoller y aprenderemos a crear un tema personalizado, a generar el archivo Zip y, finalmente, a agregar nuestro tema al proyecto.

- Grids 152
- Agrupar y dividir contenido... 156
- Temas 158
- Resumen..... 169
- Actividades..... 170

Grids

Uno de los problemas más comunes cuando programamos para distintos dispositivos es, como ya lo hemos dicho en los **capítulos 2** y **5**, la resolución y la disposición de los distintos elementos o componentes en nuestra aplicación. Para esto, el framework cuenta con el elemento **grid**, que nos permite dividir la pantalla en columnas, y así disponer del espacio y ubicar los elementos de acuerdo a la resolución. Por ejemplo: una lista en un iPad y en un iPhone nos ocuparía toda la pantalla (aunque la resolución fuera distinta); con un grid podemos dividir la pantalla y, de esta forma, aprovechamos más el espacio en la resolución de pantalla del iPad.

Los grids no contienen bordes o color de fondo, son invisibles y no tienen márgenes. Se agregan de forma secuencial, de lado a lado, como bloques, formando una grilla. Cada grid se agrega con la clase **ui-grid-a/b/c/d/e**.

Para esto, lo primero que tenemos que hacer es crear una etiqueta ** ** y agregarle una clase de estilo (**class**) a la que le vamos a pasar como valor **ui-grid-**. Cada grid puede tener hasta cinco columnas (**a-b-c-d-e**), y cada columna se denomina **bloque** (**block**). De esta forma, obtendremos:

- grids de dos columnas, agregándole **a** a la clase **ui-grid-a**;
- grids de tres columnas, agregándole **b** a la clase **ui-grid-b**;
- grids de cuatro columnas, agregándole **c** a la clase **ui-grid-c**;
- grids de cinco columnas, agregándole **d** a la clase **ui-grid-d**.

Una vez que definimos las columnas que va a tener nuestro grid (a, b, c, d), solo nos resta definir los bloques. Como dijimos antes, es un bloque por columna, y se agregan de la misma forma que los grids;

TEMAS EN DREAMWEAVER

Cuando creamos en Dreamweaver una página de jQuery Mobile, podemos cambiar los estilos CSS empleando los temas predefinidos desde el menú **Ventana/Muestras de jQuery Mobile**. Al hacer esto, se abrirá una ventana que muestra una paleta de colores con los temas ordenados alfabéticamente. Desde esta ventana, definimos el estilo de cada componente de la página.

agregamos una etiqueta ** ** por bloque y la clase (**class**) **ui-block-a/b/c/d/c/e**. De acuerdo con esto, nos quedaría de la siguiente forma:

- para crear un solo bloque, le agregamos **a** a la clase **ui-block-a**;
- para crear dos bloques, le agregamos **b** a la clase **ui-block-b**;
- para crear tres bloques, le agregamos **c** a la clase **ui-block-c**;
- para crear cuatro bloques, le agregamos **d** a la clase **ui-block-d**;
- para crear cinco bloques, le agregamos **e** a la clase **ui-block-e**.

En el siguiente código, vemos un grid básico de dos columnas:

```
<div class="ui-grid-a">
  <div class="ui-block-a">Block A</div>
  <div class="ui-block-b">Block B</div>
</div>
```



Figura 1. Grid básico **a** con dos bloques: A y B.

```
<div class="ui-grid-b">
   <div class="ui-block-a">Bloque A</div>
   <div class="ui-block-b">Bloque B</div>
   <div class="ui-block-c">Bloque C</div>
</div>
```



Figura 2. Grid **b** con tres bloques: A, B y C.

En el ejemplo que detallamos a continuación, veremos cómo utilizar un grid de tres columnas y tres filas. En cada bloque colocaremos una imagen:

```
<div data-role="page" id="pagina1" data-theme="a">

    <div data-role="header">
                <h1>Portfolio personal</h1>
    </div>

    <div data-role="content">

     <div class="ui-grid-b">

       <div class="ui-block-a">
    <img src="../img/1.jpg" style="width:80%">
   </div>
       <div class="ui-block-b">
    <img src="../img/2.jpg" style="width:80%">
   </div>
       <div class="ui-block-c">
    <img src="../img/3.jpg" style="width:80%">
   </div>
       <div class="ui-block-a">
    <img src="../img/4.jpg" style="width:80%">
   </div>
       <div class="ui-block-b">
    <img src="../img/5.jpg" style="width:80%">
   </div>
       <div class="ui-block-c">
       <img src="../img/6.jpg" style="width:80%">
   </div>
       <div class="ui-block-a">
    <img src="../img/7.jpg" style="width:80%">
   </div>
       <div class="ui-block-b">
    <img src="../img/8.jpg" style="width:80%">
```

```
    </div>
        <div class="ui-block-c">
     <img src="../img/9.jpg" style="width:80%">
     </div>
    </div>

        </div>

        <div data-role="footer" data-position="fixed">
                <h4>- 2014 -</h4>
        </div>

</div>
```

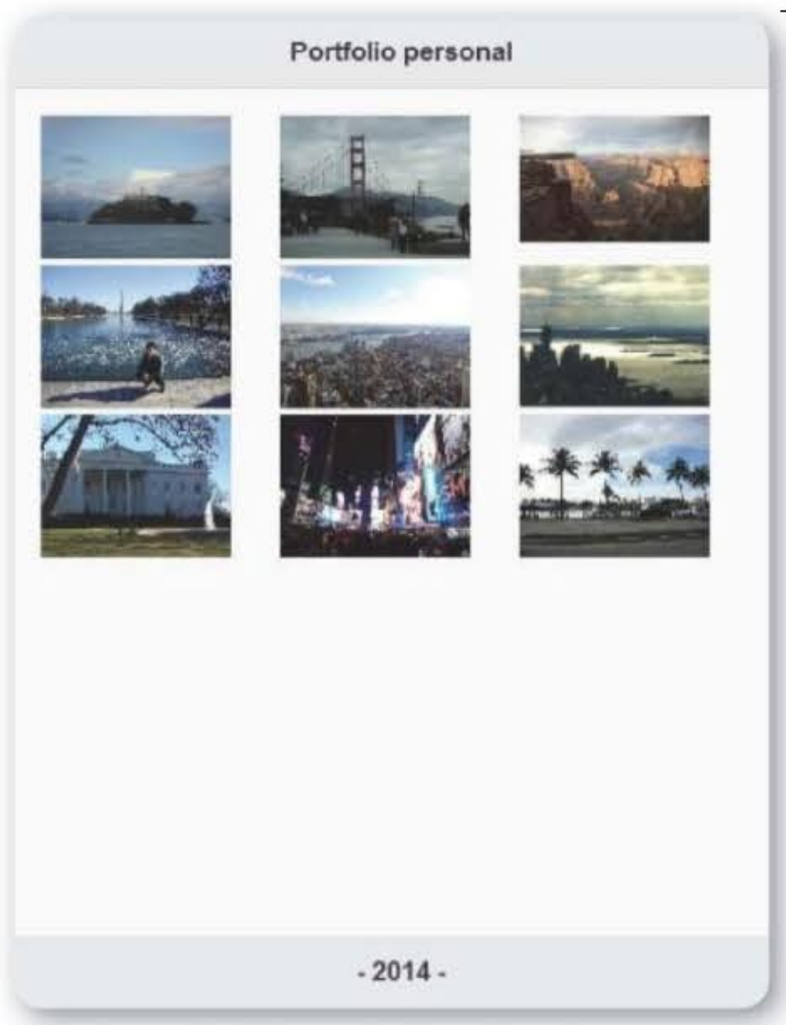


Figura 3. En un grid básico, podemos incluir otros componentes de jQuery como, por ejemplo, botones, listas e imágenes.

GRIDS CON DREAMWEAVER

Para agregar un control grid utilizando Dreamweaver, primero seleccionamos dónde vamos a ubicar este control; luego, vamos al menú **Insertar/jQuery Mobile** y, del submenú, seleccionamos **Cuadrícula de diseño**. En el cuadro de diálogo **Cuadrícula de diseño**, seleccionamos la cantidad de filas y columnas que deseamos que tenga el grid. Luego presionamos **Aceptar**.

Agrupar y dividir contenido

Para agrupar contenido y enriquecer la interfaz de usuario, otra forma que nos ofrece el framework, además de los grids, es agrupar contenido mediante las clases **ui-bar** y **ui-body**.

La clase **ui-bar** nos crea encabezados que abarcan todo el ancho de la página y que nos permiten dividir o separar secciones. Luego, con la clase **ui-body**, podemos agregarle contenido a cada sección. Para todo esto tenemos que incluir un título cualquiera de **<h1></h1>** a **<h6></h6>**, y a ese título le añadimos la clase **ui-bar**.

```
<h3 class="ui-bar ui-bar-a">Encabezado</h3>
```

Es posible crear una sección de noticias destacadas, como podemos ver en el siguiente ejemplo:

```
<h3 class="ui-bar ui-bar-a">Noticias destacadas</h3>
<div class="ui-body">
        <p>Estas son las noticias destacadas...</p>
</div>
```



Figura 4. Es posible añadirle un estilo acorde con nuestra página, con la clase **ui-bar-** (**ui-bar-a**), que toma valores de la **a** a la **z** (**ui-bar-[a-z]**).

Si quisiéramos cambiar un poco el diseño y unificar lo que es la cabecera del contenido, es decir, hacer un recuadro en donde no esté separado el título del contenido, sino que sea uno, solamente tenemos

Agrupar y dividir contenido

Para agrupar contenido y enriquecer la interfaz de usuario, otra forma que nos ofrece el framework, además de los grids, es agrupar contenido mediante las clases **ui-bar** y **ui-body**.

La clase **ui-bar** nos crea encabezados que abarcan todo el ancho de la página y que nos permiten dividir o separar secciones. Luego, con la clase **ui-body**, podemos agregarle contenido a cada sección. Para todo esto tenemos que incluir un título cualquiera de **<h1></h1>** a **<h6></h6>**, y a ese título le añadimos la clase **ui-bar**.

```
<h3 class="ui-bar ui-bar-a">Encabezado</h3>
```

Es posible crear una sección de noticias destacadas, como podemos ver en el siguiente ejemplo:

```
<h3 class="ui-bar ui-bar-a">Noticias destacadas</h3>
<div class="ui-body">
        <p>Estas son las noticias destacadas...</p>
</div>
```



Figura 4. Es posible añadirle un estilo acorde con nuestra página, con la clase **ui-bar-** (**ui-bar-a**), que toma valores de la **a** a la **z** (**ui-bar-[a-z]**).

Si quisiéramos cambiar un poco el diseño y unificar lo que es la cabecera del contenido, es decir, hacer un recuadro en donde no esté separado el título del contenido, sino que sea uno, solamente tenemos

que agregarle un ** ** sin el encabezado. Siguiendo con el ejemplo anterior, nos quedaría de la siguiente forma:

```
<div class="ui-body ui-body-a ui-corner-all">
<h3>Noticias destacadas</h3>
    <p>Estas son las noticias destacadas...</p>
</div>
```



Figura 5. Agrupamos contenido junto con el título del recuadro.

Y para hacer un solo recuadro, como el del ejemplo anterior, pero que resalte el título con otro color distinto al del contenido, tendríamos que hacer lo siguiente:

```
<div class="ui-corner-all custom-corners">
  <div class="ui-bar ui-bar-a">
    <h3> Noticias destacadas </h3>
  </div>
  <div class="ui-body ui-body-a">
    <p> Estas son las noticias destacadas...</p>
  </div>
</div>
```

Entonces se agrega un ** ** general para todo el recuadro:

```
<div class="ui-corner-all custom-corners"> </div>
```

y se separa, en otros dos ** **, el título:

```
<div class="ui-bar ui-bar-a"></div>
```

y el contenido:

```
<div class="ui-body ui-body-a"></div
```

Temas

Para cambiar el aspecto de cada objeto o componente de jQuery Mobile, contamos con una serie de **temas** o **estilos prediseñados**.

A cada tema le corresponde una letra: van desde la **a** hasta la **e**, y podemos agregarlos con el atributo **data-theme** y pasándole el valor **a**, **b**, **c**, **d** y **e**, según necesitemos. Por ejemplo:

```
<div data-role="page" id="pagina" data-theme="c" >
</div>
```

En este caso, el estilo **"c"** va a ser efectivo para toda la página. Si, en cambio, queremos personalizar elementos con diferentes temas, podemos agregarle el atributo **data-theme** a cada objeto en particular. De esta forma, obtendríamos varios estilos en una misma página jQuery Mobile.

En el siguiente ejemplo vemos cómo una misma página tiene varios estilos a la vez:

BORDES REDONDEADOS

Si queremos que nuestra cabecera o título del ejemplo anterior (**<h3 class="ui-bar ui-bar-a">Noticias destacadas</h3>**) tenga los bordes redondeados, debemos agregarle la clase **corner-all**. Lo mismo ocurre para el contenido: si le agregáramos la misma clase y tema, se mostraría redondeado: **class="ui-body ui-body-a ui-corner-all"**.

```
<div data-role="page" id="pagina">
        <div data-role="header" data-theme="a" >
        <h1>Título</h1>
        </div>

    <div data-role="content" data-theme="e" >
                <p>Contenido</p>
                        <a href="" data-role="button" data-theme="b">Botón</a>

        </div>

    <div data-role="footer" data-theme="c" data-position="fixed">
                <h4>Pie de página</h4>
        </div>
</div>
```

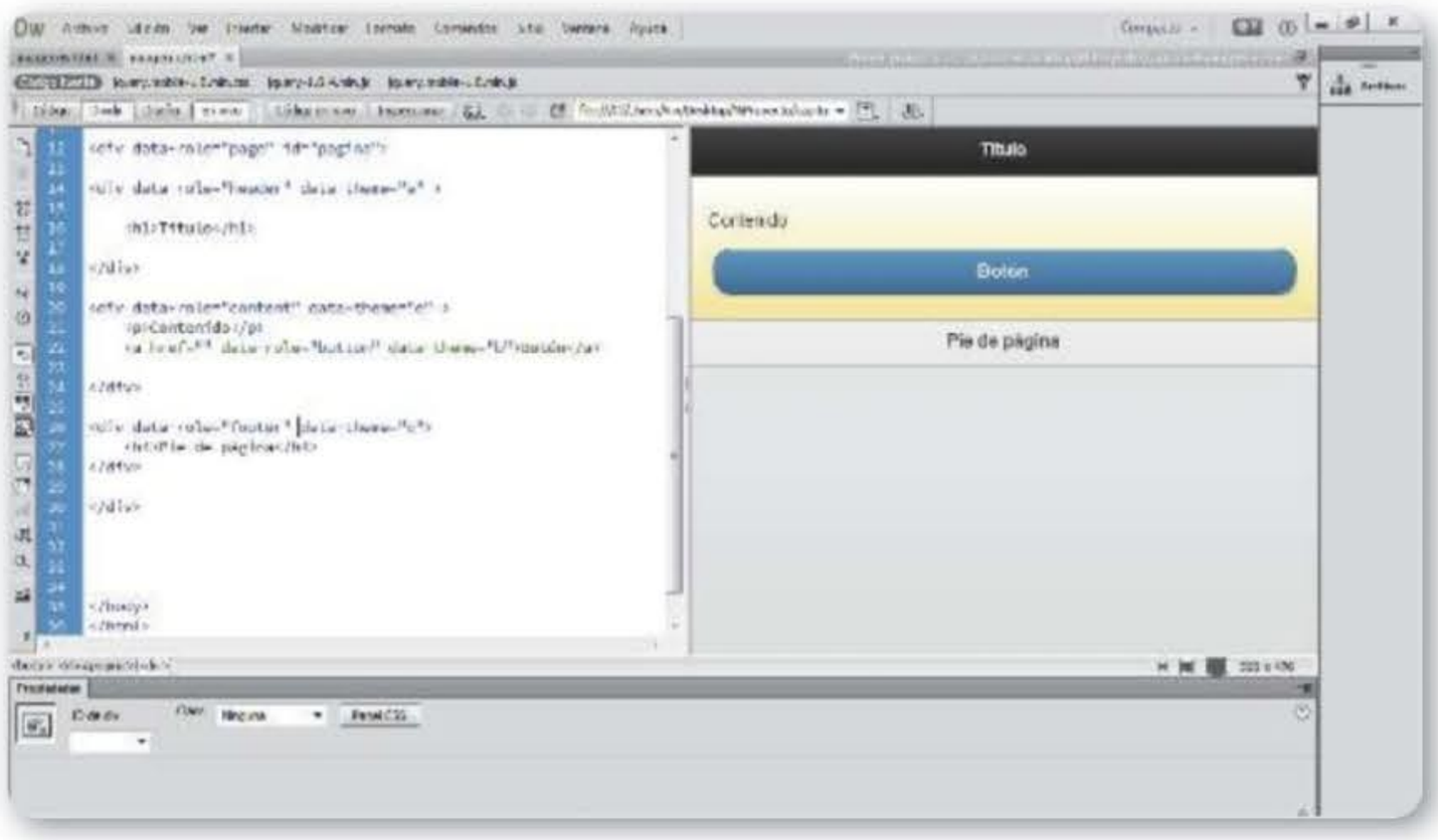


Figura 6. Todos los componentes de jQuery Mobile soportan el atributo **`data-theme`** con sus correspondientes valores.

Son cinco los estilos disponibles en el framework. A cada uno de ellos, le corresponde una letra:

- **a** (negro)
- **b** (azul)
- **c** (gris)
- **d** (gris y blanco)
- **e** (amarillo)

Temas personalizados mediante ThemeRoller

Así como podemos combinar diferentes temas dentro de una misma aplicación, como en el ejemplo anterior, también el framework nos ofrece la posibilidad de crear nuestros propios **temas personalizados**.

Para esto, disponemos de una herramienta llamada **ThemeRoller**, que se encuentra online en la página web de jQuery Mobile, **http://themeroller.jquerymobile.com**.

Es fácil de usar y nos permite crear, en tan solo unos pasos, diferentes temas que nos ayudarán a mejorar la apariencia de nuestra aplicación.

> THEMEROLLER ES UNA HERRAMIENTA QUE NOS PERMITE CREAR TEMAS ONLINE PARA NUESTRA APLICACIÓN

ThemeRoller es una página web muy parecida a un programa de escritorio, es decir, nos vamos a encontrar con la misma funcionalidad y rapidez, pero con la diferencia de que estamos trabajando en línea.

Lo primero que vemos cuando ingresamos a la página web de ThemeRoller es una pantalla de bienvenida, que nos muestra un breve texto con información sobre esta herramienta.

Al cerrar esta ventana, vemos la interfaz principal de ThemeRoller.

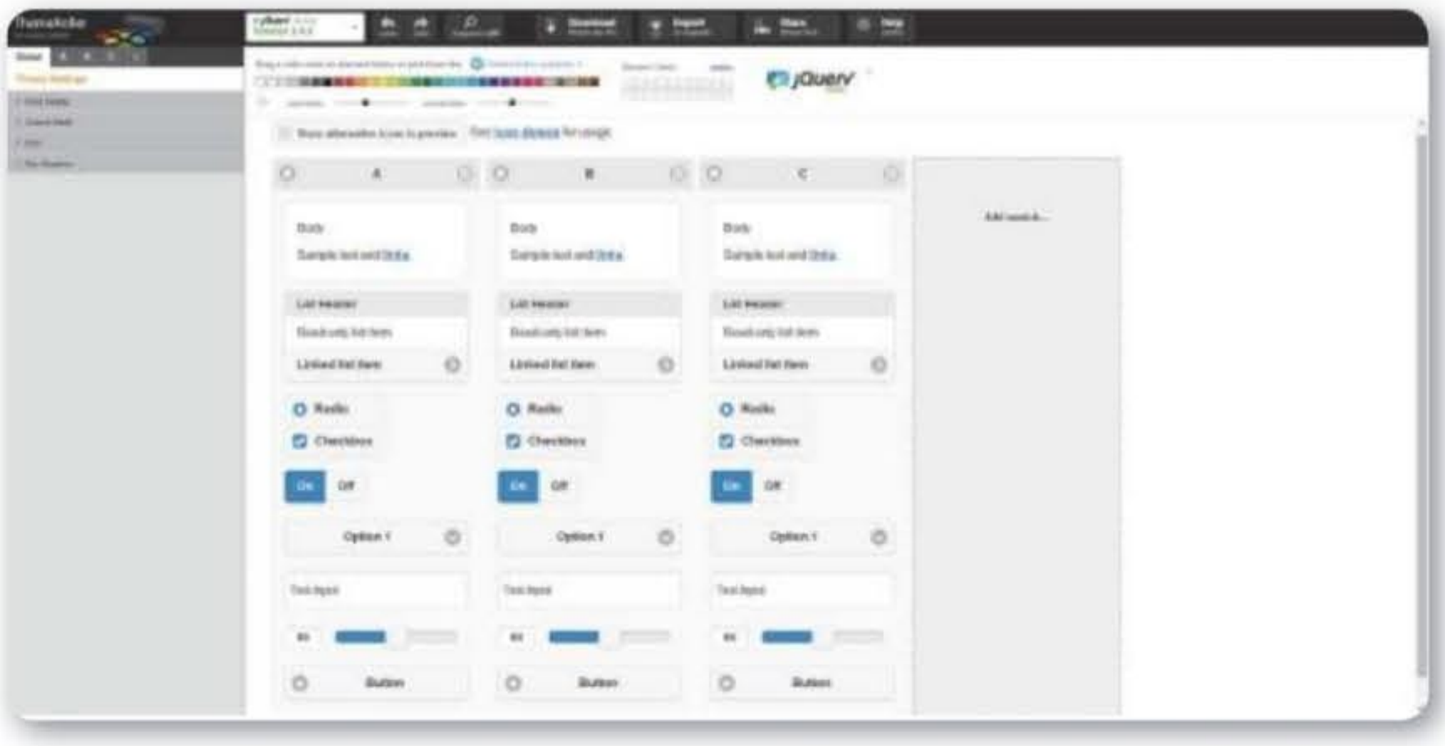


Figura 7. Podemos acceder a la página web de ThemeRoller desde la página de jQuery Mobile, haciendo clic en el menú principal, en la opción **Themes**.

La interfaz de ThemeRoller se encuentra divida en cuatro áreas principales, que veremos en detalle en los próximos apartados:

- Inspector
- Vista previa (*preview*)
- Colores (*colors*)
- Herramientas (*tools*)

Inspector

A la izquierda de la página, justo por debajo del logo de ThemeRoller, se encuentra el panel denominado **inspector**. Este panel contiene un conjunto de fichas: **Global**, **A**, **B**, **C** y **+**. Cada una de estas fichas, a su vez, contiene un menú en forma de acordeón con las opciones de configuración para cada una de las **páginas de muestra** (en inglés, *swatches*).

Si hacemos clic en la ficha **+**, se creará una nueva muestra, a la que se le asignará una letra.



Cuando hacemos clic sobre la ficha **Global**, el menú desplegable nos mostrará las opciones que nos permitirán seleccionar tipo de letra, color de iconos, sombras, bordes de los botones y grupos. Estas opciones se aplicarán a todas las muestras.

Si hacemos un clic en las otras fichas de muestra (**A**, **B** o **C**) –e incluso en las que vayamos agregando–, se nos mostrará un menú con las opciones para configurar cada una de las muestras de forma individual.

Figura 8. En la parte superior, encontramos las opciones para borrar las muestras (**`Delete`**) o para hacer una copia (**`Duplicate`**).

Vista previa (preview)

En el área de **vista previa**, que se encuentra situada en el centro de la pantalla, nos encontramos con tres páginas de muestras, cada una de las cuales contiene componentes de jQuery Mobile (**head**, **footer**, **button**, **collapsible**, entre otros).

Las páginas que vemos aquí son interactivas: si hacemos clic en los componentes, vemos cómo estos responden a la acción que vayamos ejecutando desde los controles del inspector.

Cada una de las muestras en su encabezado tiene una **letra identificadora**, que indica a qué solapa del inspector pertenecen.

Estas páginas están listas para poder comenzar a trabajar en nuestro diseño.

En las páginas de muestra se realiza el diseño para ir probando cómo se verían los distintos componentes a los que les vamos cambiando el estilo.

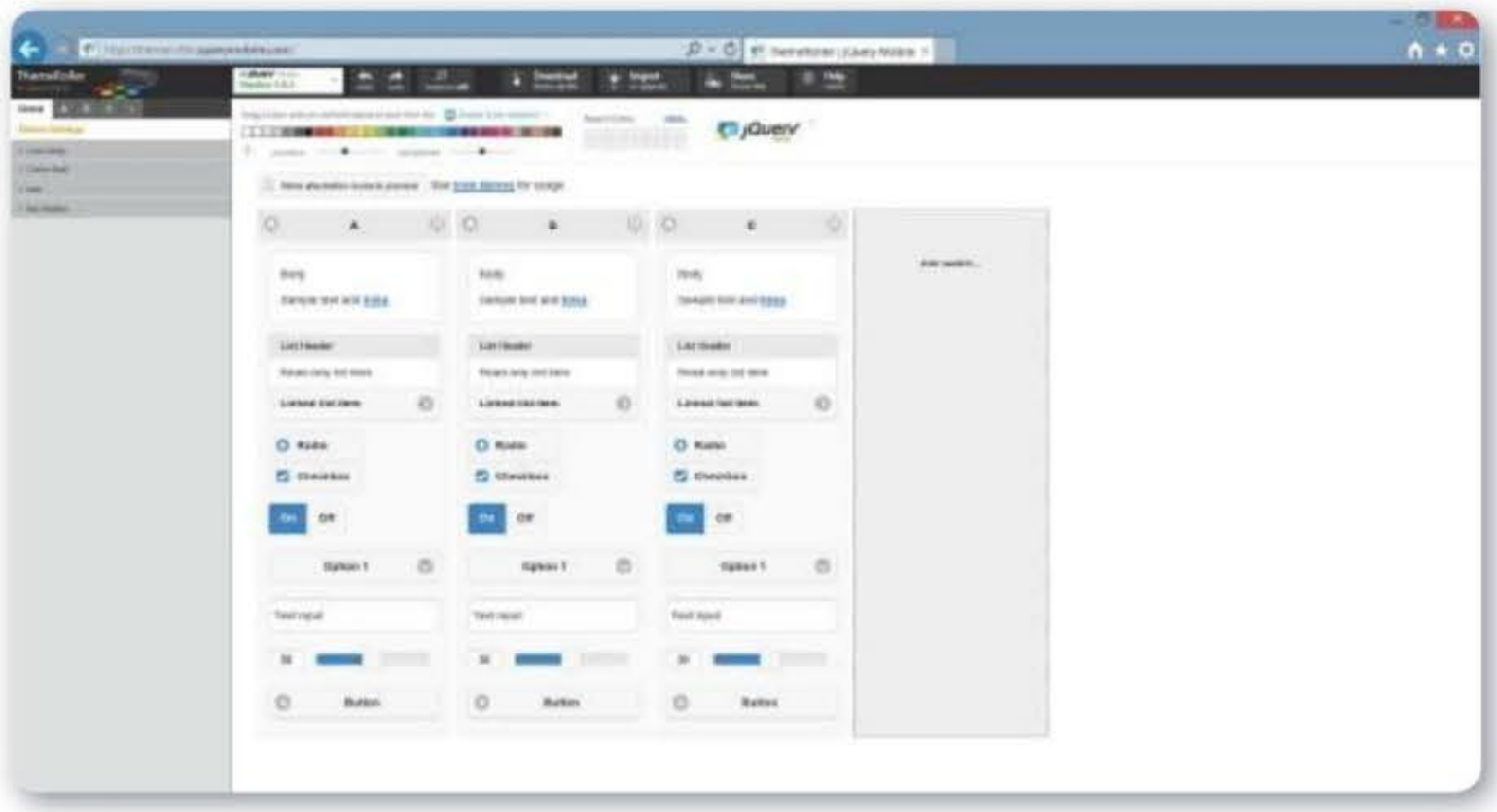


Figura 9. Si agregamos más muestras, las páginas se verán en el panel de vista previa.

MÁS COLORES

También contamos con la posibilidad de agregar colores haciendo clic en la opción **colors...**, que se encuentra sobre la paleta de colores recientes. Se abre un control con el cual podemos personalizar el color. Una vez creado el color, este se mostrará en la paleta de colores recientes.

Colores (colors)

En la parte superior de la página, encontramos una barra que se encuentra divida en dos sectores. En el sector de la izquierda se encuentra la **paleta de colores**, si hacemos clic sobre alguno de estos y, sin soltar el botón del mouse lo arrastramos sobre un componente, este último quedará pintado de aquel color.

A la derecha se encuentra una paleta de colores vacía (**recent colors**), que muestra los colores utilizados recientemente; es decir, a medida que vayamos trabajando, se van a ir ubicando los colores que hayamos elegido. De esta forma, podemos visualizarlos y acceder a ellos de manera rápida.

Debajo de estas paletas, se encuentran dos controles deslizantes con los que podemos modificar el **brillo** (**lightness**) y la **saturación** (**saturation**) de los colores seleccionados.

Herramientas (tools)

En la parte superior de la página de ThemeRoller, se encuentra la **barra de herramientas**, compuesta por ocho botones que, de izquierda a derecha, son los siguientes:

- **jQuery Mobile Version**: nos da la posibilidad de poder trabajar con versiones anteriores del framework. Al hacer clic sobre la que necesitemos armar, vamos a notar cómo cambian las muestras de acuerdo con la versión, agregando o sacando componentes.
- **Undo** o **deshacer**: nos permite deshacer o volver a un estado anterior una acción realizada.
- **Redo** o **rehacer**: nos permite repetir la última acción realizada.
- **Inspector off**: selecciona, por medio de un recuadro de color, cada componente (sobre el que estemos parados) y nos muestra a la izquierda en el menú global las propiedades de ese objeto.



CONFIGURACIÓN DEL MENÚ GLOBAL

Para cambiar valores de colores se utiliza el formato hexadecimal, y el tamaño de los valores se mide en pixeles. Igualmente, cuando seleccionamos una casilla de texto para pasarle un valor, se nos abre una paleta de colores para seleccionar e ir probando el que más nos guste.

- **Download theme zip file**: nos descarga, en formato **.ZIP**, el tema ya listo para agregar a nuestra aplicación.
- **Import or upgrade**: nos permite agregar una muestra al programa para poder modificarla.
- **Share theme link**: nos crea un enlace con todo lo que hayamos hecho en el programa. Es muy útil cuando trabajamos en equipo o queremos mostrar el trabajo antes de crear el estilo definitivo.
- **Help center**: muestra en una ventana la ayuda del programa.

Crear un tema

Como mencionamos antes, crear un tema personalizado en ThemeRoller es muy sencillo: podemos crear uno nuevo o importar uno existente.

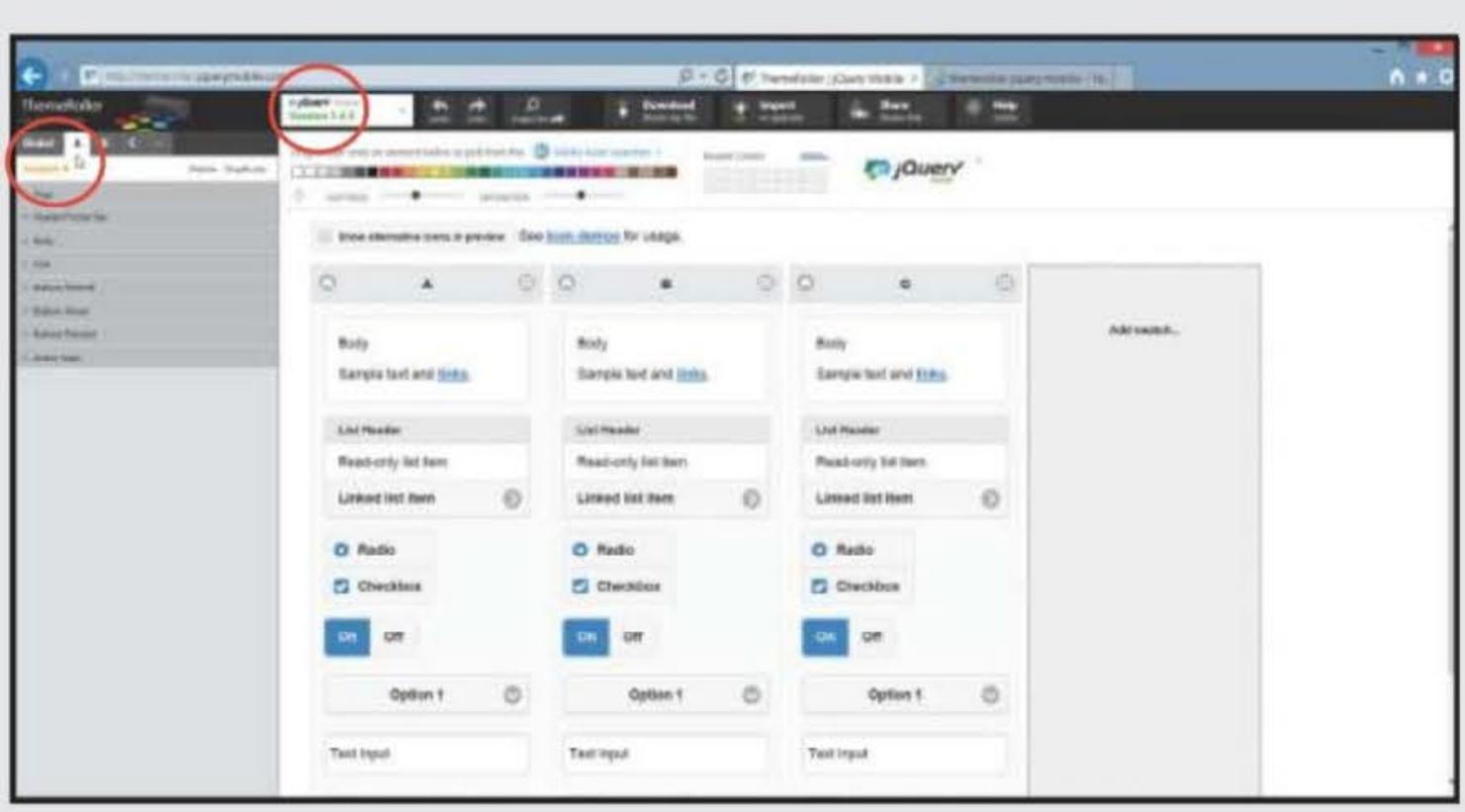
En el siguiente **Paso a paso**, explicaremos mediante un ejemplo cómo crear un tema personalizado compuesto por dos variaciones de colores, una en la gama del azul y la otra en la gama del violeta.

PAP: CREAR UN TEMA PERSONALIZADO

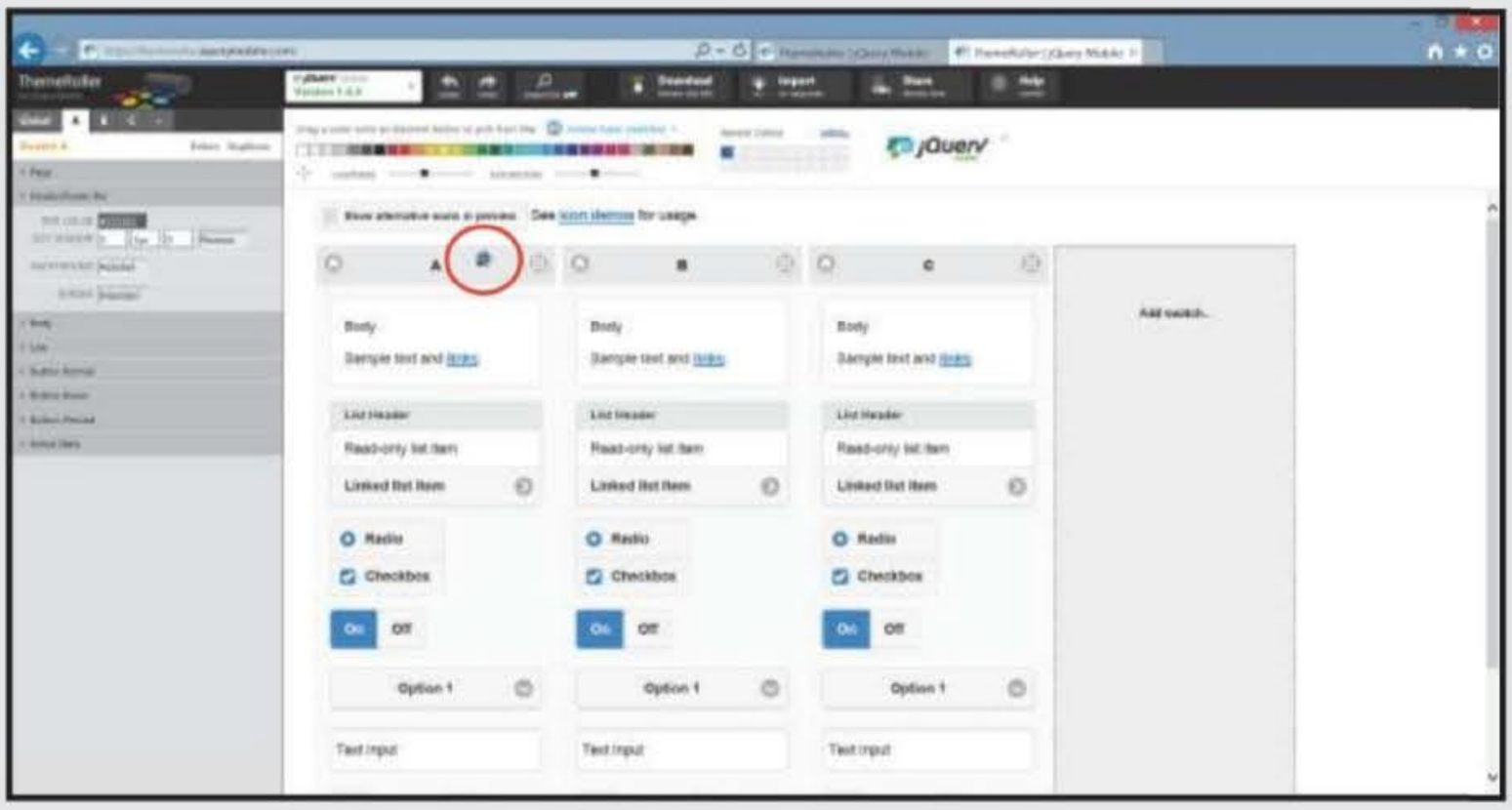
01 Ingrese a la página de ThemeRoller: **http://themeroller.jquerymobile.com**. En la pantalla de bienvenida ("Welcome to ThemeRoller for jQuery Mobile"), haga clic en el botón `Get Rolling`.



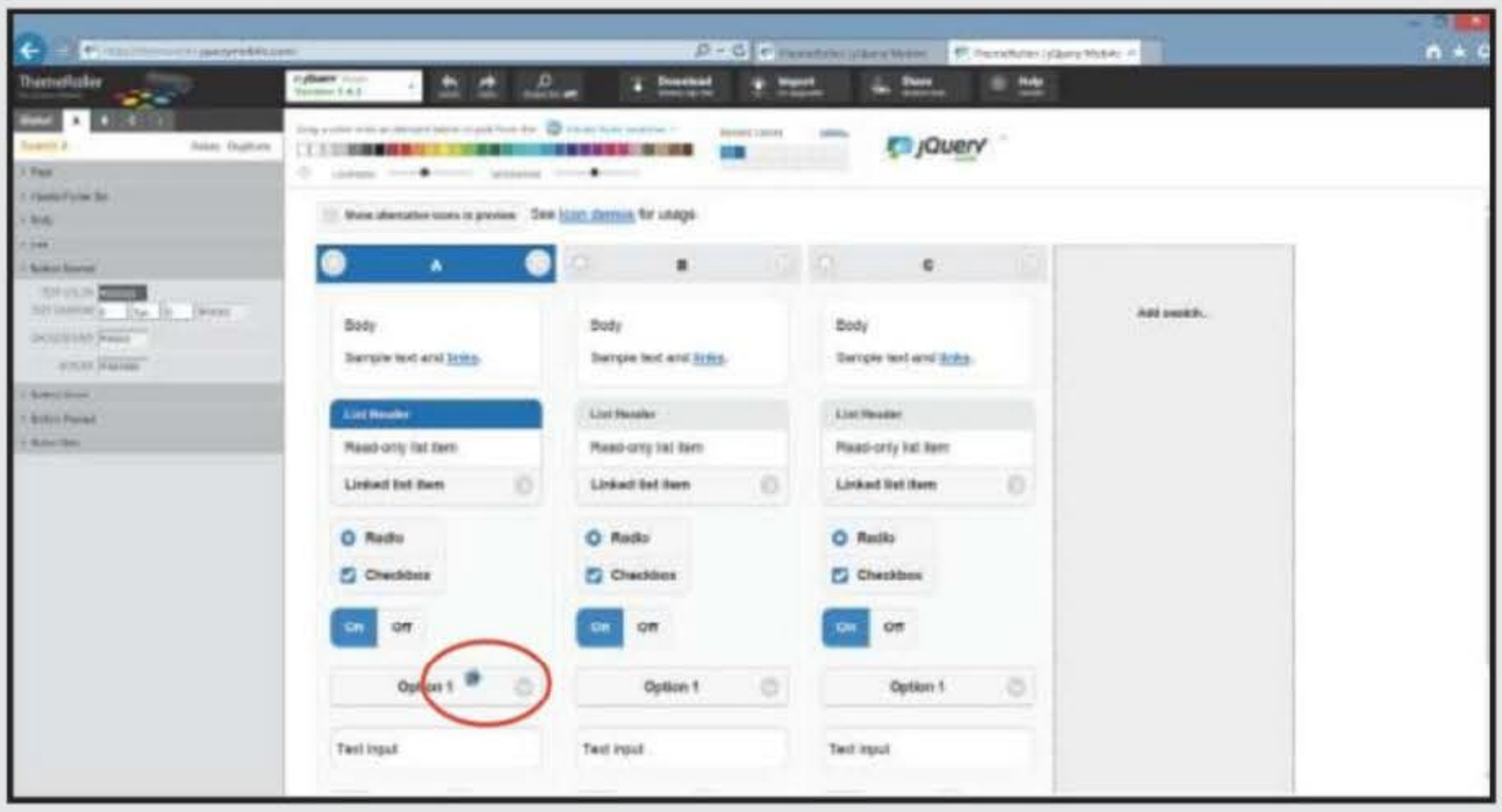
02 En el botón de la barra de herramientas, seleccione la versión de jQuery Mobile: en nuestro caso, la versión 1.4.3. Para crear la primera variación de colores, haga clic en la ficha A del panel `Inspector`.



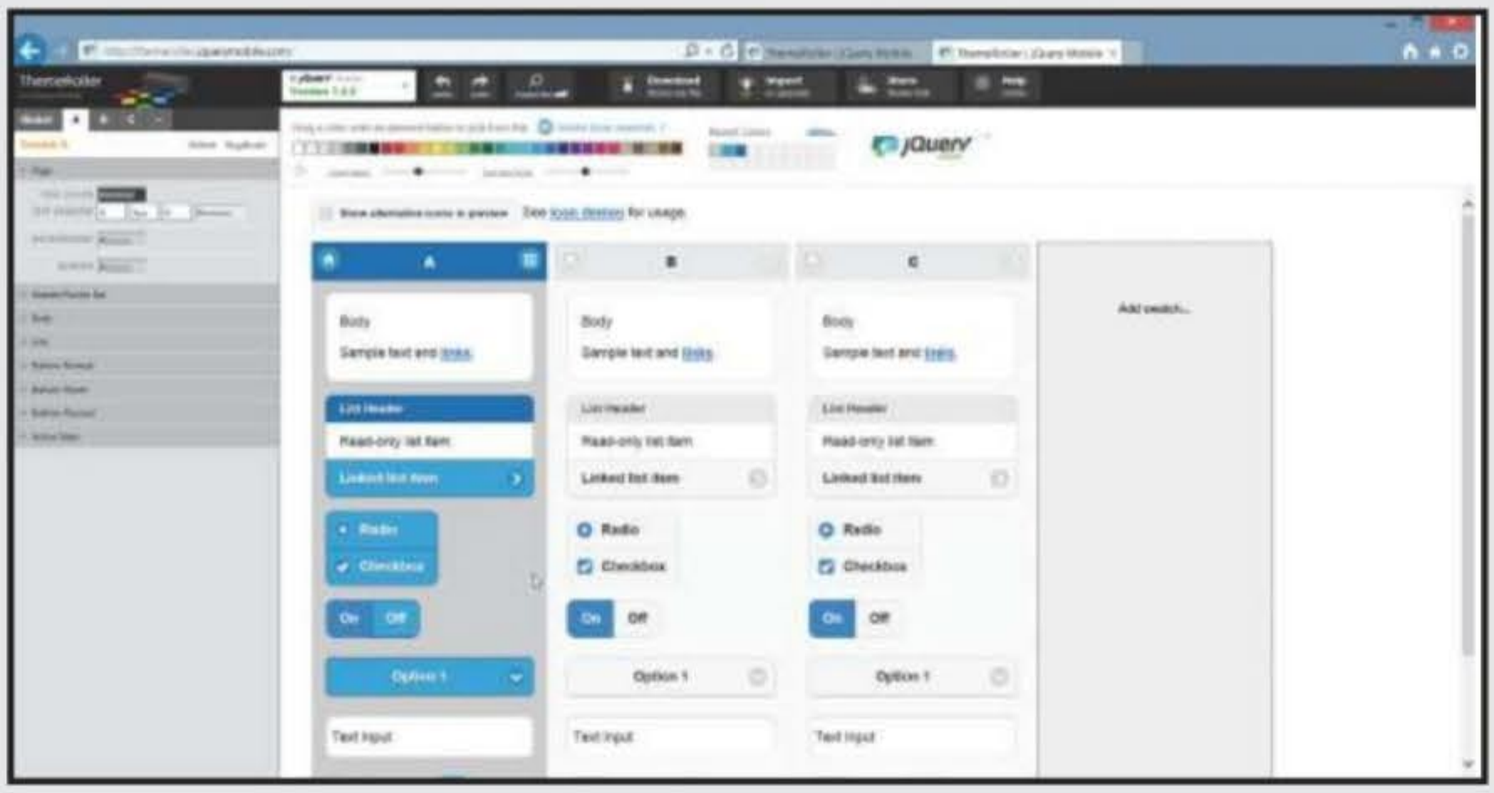
03 Seleccione un color en la paleta de colores (en nuestro caso, azul) y luego arrástrelo sobre el encabezado que se encuentra en el panel de la vista previa.



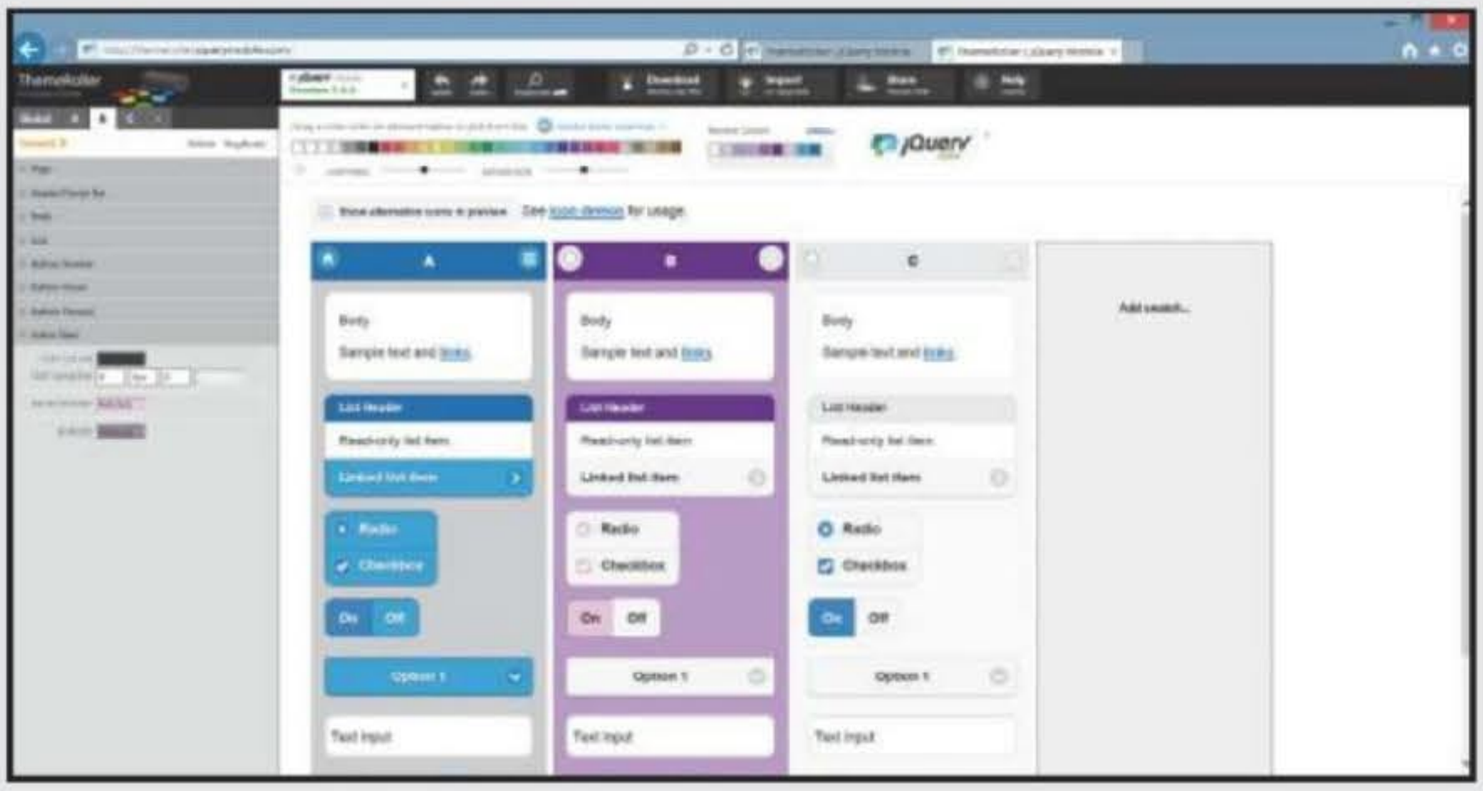
04 Seleccione otro color de la paleta (en nuestro caso, un azul más claro) y arrástrelo hasta el botón que se encuentra en el panel de la vista previa. Continúe de la misma forma con el resto de los componentes.



05 Para cambiar el color de fondo (background), seleccione un color de la paleta de colores y arrástrelo sobre el fondo en la vista previa.



06 Para crear la segunda variación, en el panel `Inspector`, seleccione la ficha B y repita los pasos 3, 4 y 5.



Asignar temas

Para asignar un tema o estilo determinado en nuestro documento jQuery Mobile, lo primero que tenemos que hacer, después de completar el diseño como nos guste en ThemeRoller, es generar el archivo Zip para agregarlo a nuestro proyecto.

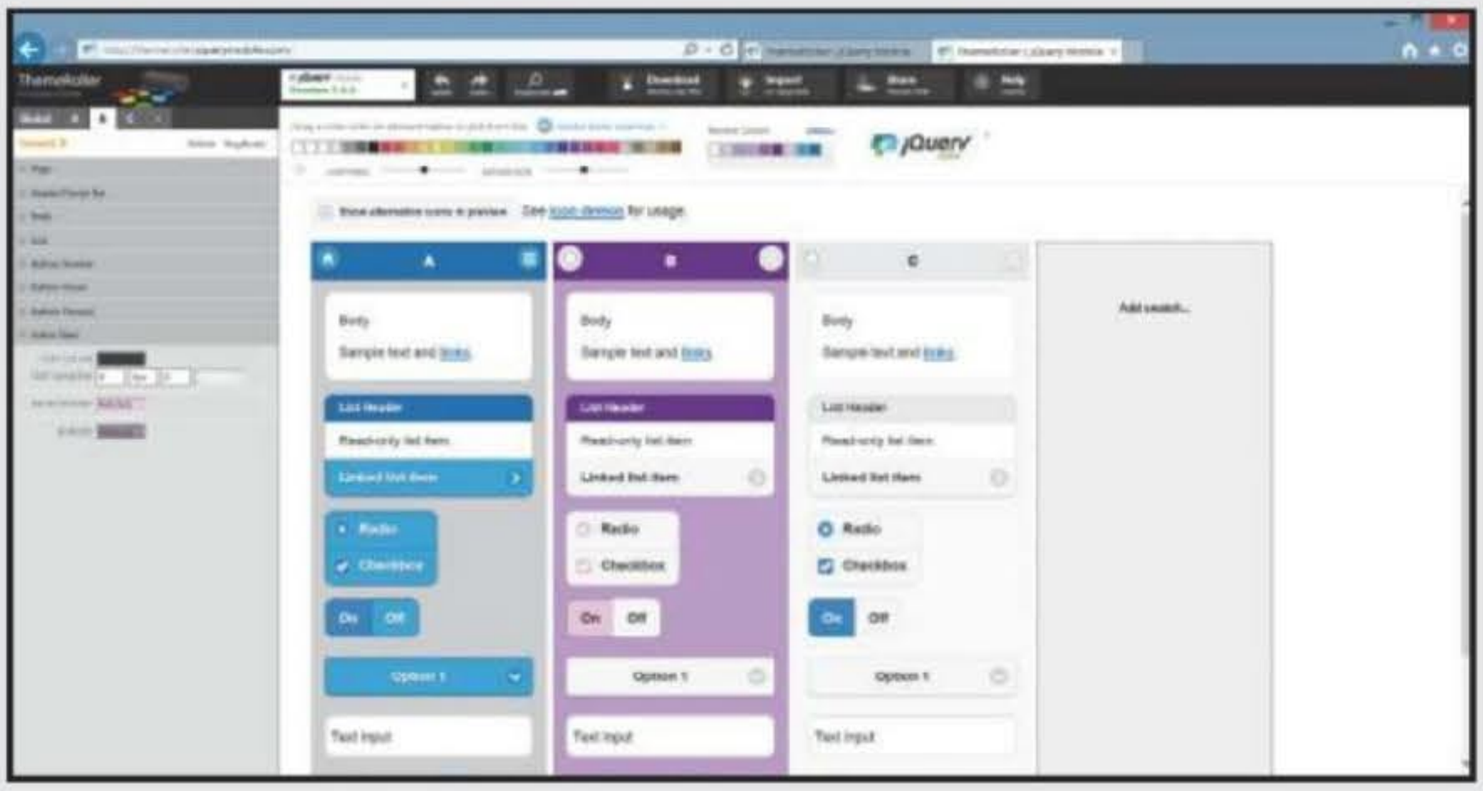
Esto se realiza presionando el botón **Download theme zip file**, de la barra de herramientas de ThemeRoller.

Se va a abrir el cuadro de diálogo **Download theme**; en el campo de texto **Theme Name** deberemos escribir el nombre que le queremos poner al estilo (por ejemplo, **mi_tema**).

ICONOS

Los iconos son imágenes SVG y PNG. Los SVG son soportados en resoluciones estándares y HD (alta definición). Cuando el dispositivo no soporta la extensión SVG, se muestran los iconos PNG. Para incluir un icono, solo tenemos que agregarle al componente el atributo **data-icon** y asignarle el nombre del icono que queremos mostrar.

06 Para crear la segunda variación, en el panel `Inspector`, seleccione la ficha B y repita los pasos 3, 4 y 5.



Asignar temas

Para asignar un tema o estilo determinado en nuestro documento jQuery Mobile, lo primero que tenemos que hacer, después de completar el diseño como nos guste en ThemeRoller, es generar el archivo Zip para agregarlo a nuestro proyecto.

Esto se realiza presionando el botón **Download theme zip file**, de la barra de herramientas de ThemeRoller.

Se va a abrir el cuadro de diálogo **Download theme**; en el campo de texto **Theme Name** deberemos escribir el nombre que le queremos poner al estilo (por ejemplo, **mi_tema**).

ICONOS

Los iconos son imágenes SVG y PNG. Los SVG son soportados en resoluciones estándares y HD (alta definición). Cuando el dispositivo no soporta la extensión SVG, se muestran los iconos PNG. Para incluir un icono, solo tenemos que agregarle al componente el atributo **data-icon** y asignarle el nombre del icono que queremos mostrar.

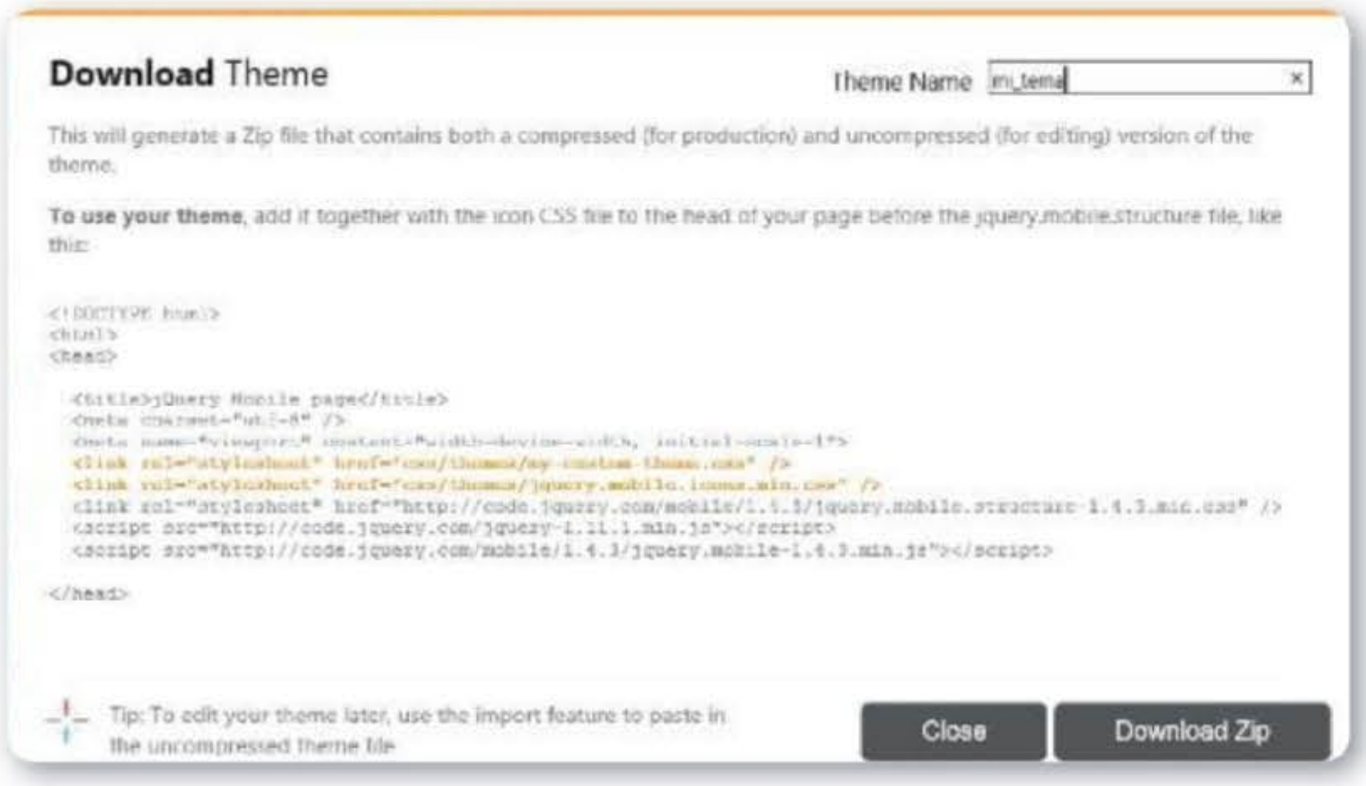


Figura 10. Este cuadro de diálogo también nos muestra un ejemplo de cómo debería incluirse en nuestro código HTML el llamado al estilo.

Por último, debemos presionar el botón **Download Zip** para comenzar la descarga del archivo en nuestra computadora.

El archivo Zip descargado contiene un archivo **index.html** y una carpeta **themes** con los archivos **.CSS** (versión comprimida y sin comprimir, más los **.CSS** de los iconos) junto con otra carpeta con los iconos del estilo, llamada **images**.

Descomprimimos esta carpeta y la copiamos dentro de la carpeta **CSS** de nuestro proyecto. Le cambiamos el nombre a la carpeta; por ejemplo, la denominamos **themes**.

Luego editamos el código de la página, modificando la ruta del link que apunta al estilo creado, como mostramos a continuación:

```
<link rel="stylesheet" href="css/themes/mi_tema.css" />
```

Por lo tanto, el código completo nos quedaría de la siguiente manera:

```
<!DOCTYPE html>
<html>
<!DOCTYPE html>
<html>
<head>
  <title>Mi página jQuery Mobile</title>
  <meta charset="utf-8" />
```

```
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel="stylesheet" href="css/themes/mi_tema.css" />
    <link rel="stylesheet" href="css/themes/jquery.mobile.icons.min.css" />
    <link rel="stylesheet" href="http://code.jquery.com/mobile/1.4.3/jquery.mobile.
structure-1.4.3.min.css" />  <script src="http://code.jquery.com/jquery-1.11.1.min.
js"></script>
    <script src="http://code.jquery.com/mobile/1.4.3/jquery.mobile-1.4.3.min.js"></
script>

  </head>
```

Una vez modificado el código, podemos utilizar el tema creado, simplemente usando el atributo **data-theme** y asignándole una de las letras de las que hemos creado.

Importar y compartir un tema

La herramienta ThemeRoller nos permite, además de crear un tema propio, como ya vimos, importar este tema, ya sea creado por nosotros o por otro desarrollador. Para ello, tenemos que emplear la opción **Import or upgrade** (**Importar)**, que se encuentra en la barra de herramientas.

ThemeRoller también nos permite compartir nuestro tema personalizado con otros desarrolladores. Para ello, en la barra de herramientas, presionamos el botón **Share Theme** y, de esta forma, obtendremos un enlace para enviárselo a la persona con la cual queremos compartir el desarrollo.

RESUMEN

En este capítulo, explicamos el uso del elemento grid, que nos permite dividir la pantalla en columnas para poder organizar el contenido de una página y, mediante ejemplos simples, aprendimos a crearlos. Posteriormente explicamos los aspectos básicos de los temas de jQuery Mobile y vimos las cinco muestras que se incluyen en el framework. Explicamos cómo podemos asignarles los temas a los distintos componentes usando el atributo **data-theme**. Luego aprendimos a personalizar muestras y a crear las propias empleando la herramienta ThemeRoller y, por último, cómo asignar, importar y compartir nuestros temas personalizados.

Actividades

TEST DE AUTOEVALUACIÓN

1. ¿Son invisibles los **grids**?
2. ¿Qué clase es necesaria para crear un grid de tres columnas?
3. ¿Qué otra opción nos ofrece el framework de jQuery Mobile para agrupar contenido?
4. ¿Qué atributo es necesario para cambiar el estilo de un documento jQuery Mobile?
5. ¿Cómo se llama la página a la que podemos acceder para crear un **estilo personalizado**?

EJERCICIOS PRÁCTICOS

1. Realice una página que contenga un grid de tres columnas con dos bloques.
2. Continuando con el ejercicio anterior, agréguele una imagen al segundo bloque de la segunda columna.
3. Realice una página en donde el encabezado tenga tema **c**; el contenido, tema **d**; y el pie de página, tema **e**.
4. Con el programa **ThemeRoller**, desarrolle un tema y aplíquelo a una página estándar.
5. Descargue el tema realizado en el punto 4 y vuélvalo a subir al sitio web de ThemeRoller para volver a editarlo.



PROFESOR EN LÍNEA

Si tiene alguna consulta técnica relacionada con el contenido, puede contactarse con nuestros expertos: profesor@redusers.com

*07

PhoneGap

En este capítulo, conoceremos PhoneGap y Adobe PhoneGap Build, y aprenderemos qué diferencias existen entre ellos. Explicaremos cómo crear una cuenta en Adobe para utilizar el servicio, y cómo compilar una aplicación y descargarla utilizando los servicios de Adobe PhoneGap Build. También explicaremos de qué manera utilizar los distintos editores, Dreamweaver CC, Adobe Edge Code y Bracket, para compilar nuestra aplicación. Por último, conoceremos algunas extensiones disponibles para PhoneGap y el emulador Ripple, que nos permitirá probar nuestras aplicaciones.

- ¿Qué es PhoneGap? 172
- Adobe PhoneGap Build..... 174
- Trabajar con Dreamweaver CC 180
- Extensiones para PhoneGap..... 183
- El emulador Ripple 189
- Resumen..... 191
- Actividades..... 192

¿Qué es PhoneGap?

Como mencionamos en el **Capítulo 1** de este libro, **PhoneGap** es un framework libre y de código abierto (open source) que nos permite hacer aplicaciones para dispositivos móviles utilizando HTML5, JavaScript y CSS3. No es necesario tener conocimientos de cada lenguaje en particular para programar en cada dispositivo (Java, Objective C o C#), sino que el framework compila nuestras aplicaciones y las hace **multiplataforma**. De esta manera, nos ahorramos, como desarrolladores, el trabajo de tener que programarlas para cada dispositivo en particular.

PhoneGap fue desarrollado por la empresa **Nitobi**, que producía aplicaciones para distintos sistemas operativos de dispositivos móviles. Por eso realizaron este framework: para tener solamente una aplicación y que sea compatible con todos los dispositivos sin importar el sistema operativo que tuvieran. Para esto, PhoneGap maneja las **API** (*Application Programming Interface*) del sistema que nos permite, a los desarrolladores, tener acceso a funciones específicas de cada sistema operativo, como el acelerómetro, almacenamiento, cámara de fotos, y muchas más, por medio de JavaScript.

Nitobi fue comprada en octubre de 2011 por **Adobe** y, de esta manera, PhoneGap pasó a formar parte de la suite **Creative Cloud** de Adobe. En el momento de la compra por parte de Adobe, el proyecto fue enviado a la fundación **Apache** con el nombre **Apache Cordova**. Sin embargo, hasta hoy PhoneGap sigue siendo de código abierto y libre. Lo que comercializa Adobe es **Adobe PhoneGap Build** –que también conoceremos en este capítulo– y no PhoneGap, que es libre y gratuito.

API (APPLICATION PROGRAMMING INTERFACE)

Las **API** –o **interfaces de programación de aplicaciones**, en español– son funciones y procedimientos que nos dan acceso al sistema operativo y nos permiten hacer uso de este de forma limitada. En términos de programación, producen una capa de abstracción que nos permite acceder al sistema solo con parámetros de funciones o llamándolas sin necesidad de conocerlas ni de saber cómo fueron programadas. Un ejemplo sencillo y fácil de comprender y que vemos a diario es el uso de Google Maps, en donde, si le pasamos parámetros al servicio, podemos colocar un mapa en nuestra web.

Dispositivos compatibles

Los dispositivos compatibles con PhoneGap son: iPhone, iPhone 3G, 3GS y posteriores, dispositivos con Android, BlackBerry OS 6 o posteriores, BlackBerry 10, Windows Phone 8, Ubuntu y Firefox OS.

En la página web de PhoneGap, en el menú **About** y en la opción **Features**, podemos ver una tabla comparativa con las funciones (APIs) disponibles en los distintos dispositivos móviles.

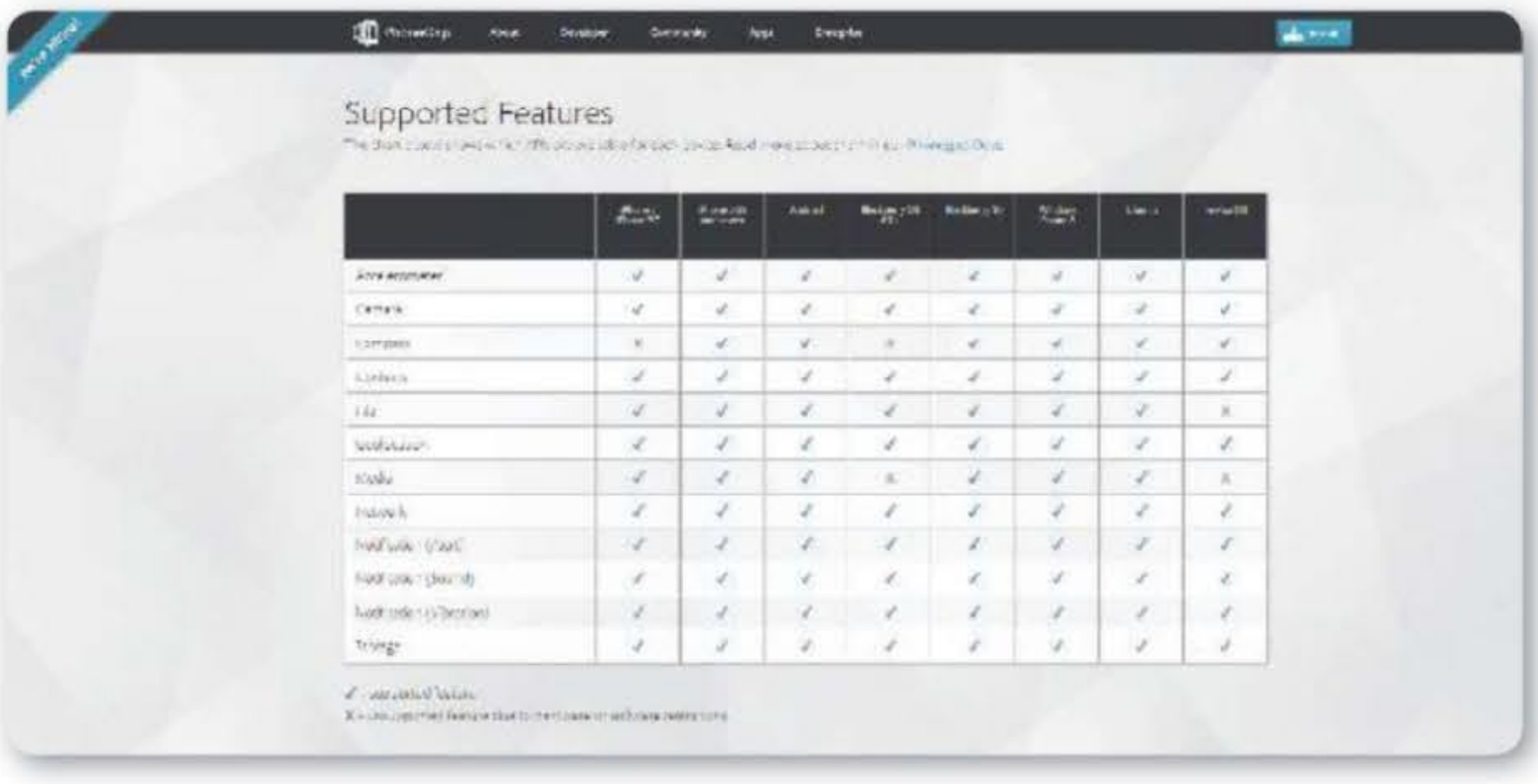


Figura 1. Gráfico comparativo disponible en **http://phonegap.com/about/feature**.

Descargar PhoneGap

Para comenzar a trabajar con PhoneGap, lo primero que tenemos que hacer es descargar el framework directamente desde su sitio web oficial: **http://phonegap.com**.

Una vez que ingresamos a la página, tenemos diferentes opciones para descargar el archivo:

- Hacemos clic sobre el botón **Install**, situado en el menú a la derecha de la barra de menú.
- Hacemos clic en **Developer** y, luego, seleccionamos **Install**.
- Hacemos clic sobre el botón **Install PhoneGap**, que se encuentra ubicado a la izquierda de la página.

Estas formas de enlace nos van a llevar a la página web **http://phonegap.com/install**, en donde encontraremos todas las versiones disponibles del framework hasta la fecha.

Al descomprimir el archivo Zip que descargamos, nos vamos a encontrar con archivos que contienen notas de la versión, y con dos carpetas: una llamada **doc** y otra **lib**. En la carpeta **doc**, disponemos de documentación y ejemplos de cómo utilizar las distintas funciones del framework. En la carpeta **lib**, están las librerías listas para ser utilizadas, divididas por cada sistema operativo (es decir, si vamos a trabajar con Android, tendremos que trabajar solamente con la carpeta llamada **Android**, donde disponemos de la librería para ese sistema operativo).

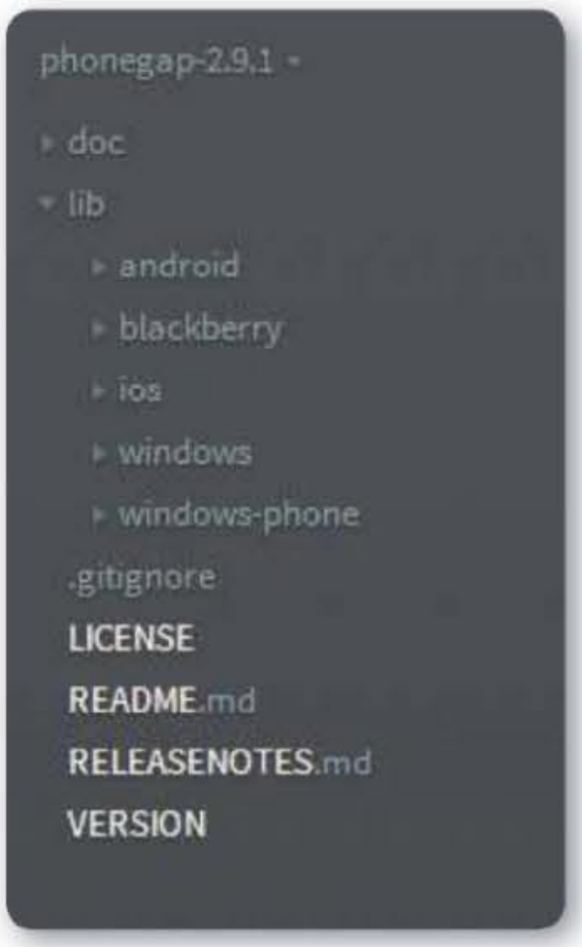


Figura 2. Directorio del Zip descargado del sitio web de PhoneGap con su última versión, la 2.9.1, visto desde **Adobe Edge Code**.

Adobe PhoneGap Build

Adobe PhoneGap Build es un servicio creado por Adobe en la nube, que sirve para compilar nuestras aplicaciones realizadas con JavaScript, CSS3 y HTML5, y descargarlas listas para ser distribuidas en las distintas tiendas de los diferentes sistemas operativos móviles. Es decir que podemos subir nuestra aplicación a un servidor web de Adobe y descargarla en nuestra computadora sin necesidad de instalar ningún software adicional para realizar el proceso de compilación y empaquetado; solamente tenemos que subir nuestro proyecto al servidor mediante la página web de Adobe PhoneGap Build.

También contamos con la posibilidad de compilar nuestra aplicación sin utilizar el servicio de Adobe, pero, para esto, tendríamos que

emplear las herramientas nativas de desarrollo de cada sistema operativo con su correspondiente kit de desarrollo de software (**SDK**). Por ejemplo, en iOS necesitamos utilizar xCode o, en el caso de Android, podríamos utilizar Eclipse.

Adobe PhoneGap Build funciona de una manera sencilla: con solo tres pasos podemos crear una aplicación móvil. Primero, subimos nuestra aplicación ya desarrollada; luego, se compila y se crea el paquete en la nube; y, por último, la descargamos para el sistema operativo que necesitemos, ya lista para ser subida a la tienda de aplicaciones correspondiente.

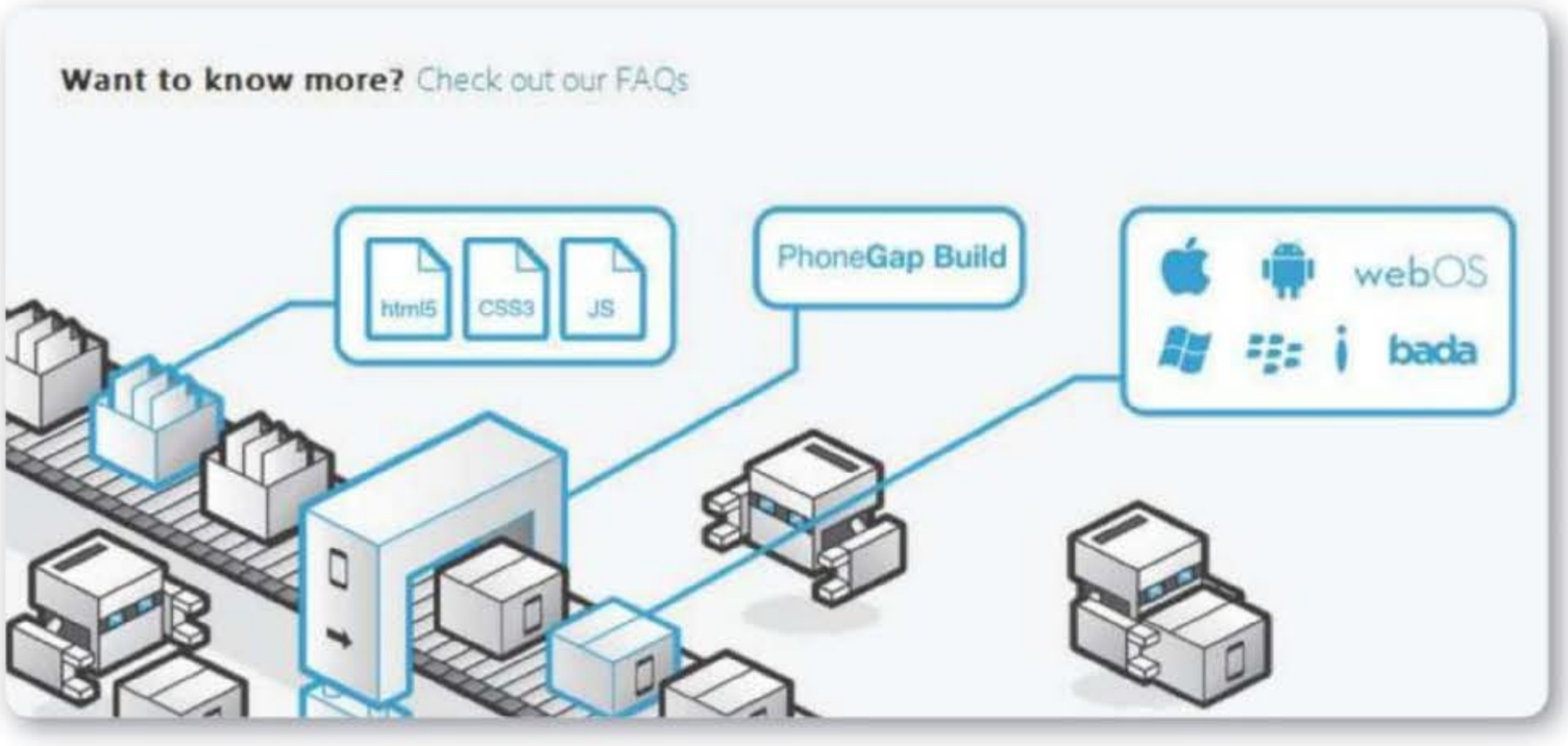


Figura 3. En el sitio web de **Adobe PhoneGap Build**, se puede observar cómo es el proceso de empaquetado una vez que subimos nuestra aplicación a la Web.

Cómo crear una cuenta

Para poder acceder al servicio de Adobe PhoneGap Build, necesitamos tener una **cuenta de usuario** (**ID**) de Adobe o ser un usuario registrado

SDK (SOFTWARE DEVELOPMENT KIT)

SDK (en español, **kit de desarrollo de software**) es un conjunto de herramientas que nos sirve para crear aplicaciones de manera simple para un sistema o plataforma en particular. La mayoría de los SDK es gratuita y se distribuye libremente por la Web. Un ejemplo para dispositivos móviles es el SDK para Android, disponible en su página web para desarrolladores: **http://developer.android.com/sdk/index.html**.

de **GitHub**. Esto se implementó a partir de mayo de 2014, cuando, por mayor seguridad, decidieron que solo se aceptarían usuarios vinculados con estos dos métodos de registro.

En el caso de que no estemos registrados en ninguna de las dos opciones anteriores, podemos crearnos una cuenta de Adobe directamente de la página de Adobe PhoneGap Build (**https://build.phonegap.com**). Para esto, cuando accedemos a la página, en la barra de menú, hacemos clic en **Sign in**. En la pantalla que nos permite iniciar sesión, hacemos clic en **Sign in with Adobe ID** y, luego, en el botón **Get an Adobe ID** ('obtener un ID de Adobe'). En la pantalla de registro, deberemos completar los campos que nos solicitan para así obtener el nombre de usuario y la contraseña, que nos permitirán acceder al servicio de Adobe PhoneGap Build.



Figura 4. Para crear una cuenta de Adobe, debemos ingresar todos los datos que nos piden en el formulario de registro.

Acceder al servicio de PhoneGap Build

Una vez que estamos registrados en Adobe o en GitHub, ingresamos al sitio web de Adobe PhoneGap y, en el apartado **Package mobile apps in the cloud**, que se encuentra en el centro de la página web, hacemos clic en el botón **Try it now!**.

También podemos acceder al servicio de PhoneGap Build directamente desde la página web de Adobe PhoneGap Build, haciendo clic en el botón central de la página, **Get started!** ('comenzar').

Una vez que tenemos nuestro usuario para poder utilizar el servicio, debemos elegir a continuación qué plan utilizar. El plan libre (en inglés, **free plan**) nos permite, al igual que en los otros dos planes **paid plan** (plan pago) o **Adobe Creative Cloud Membership** (como miembro de la suite de Adobe Creative Cloud), tener aplicaciones ilimitadas desde un repositorio de GitHub, y, también, acceso ilimitado para compartir nuestra aplicación con desarrolladores y probadores (en inglés, **testers**). La única diferencia radica en que el plan libre solo puede tener una aplicación privada (a diferencia de los otros dos planes, que pueden tener hasta un máximo de 25). Una **aplicación privada** es aquella en la que el código fuente de nuestra aplicación se encuentra alojado en un servidor al que solo nosotros tenemos acceso con nuestro identificador y contraseña de Adobe. Así, se diferencia de un repositorio como GitHub, en el que pueden acceder los desarrolladores para probar y aportar comentarios.

> PARA ACCEDER A PHONEGAP BUILD DEBEMOS ELEGIR, PRIMERO, EL PLAN QUE UTILIZAREMOS

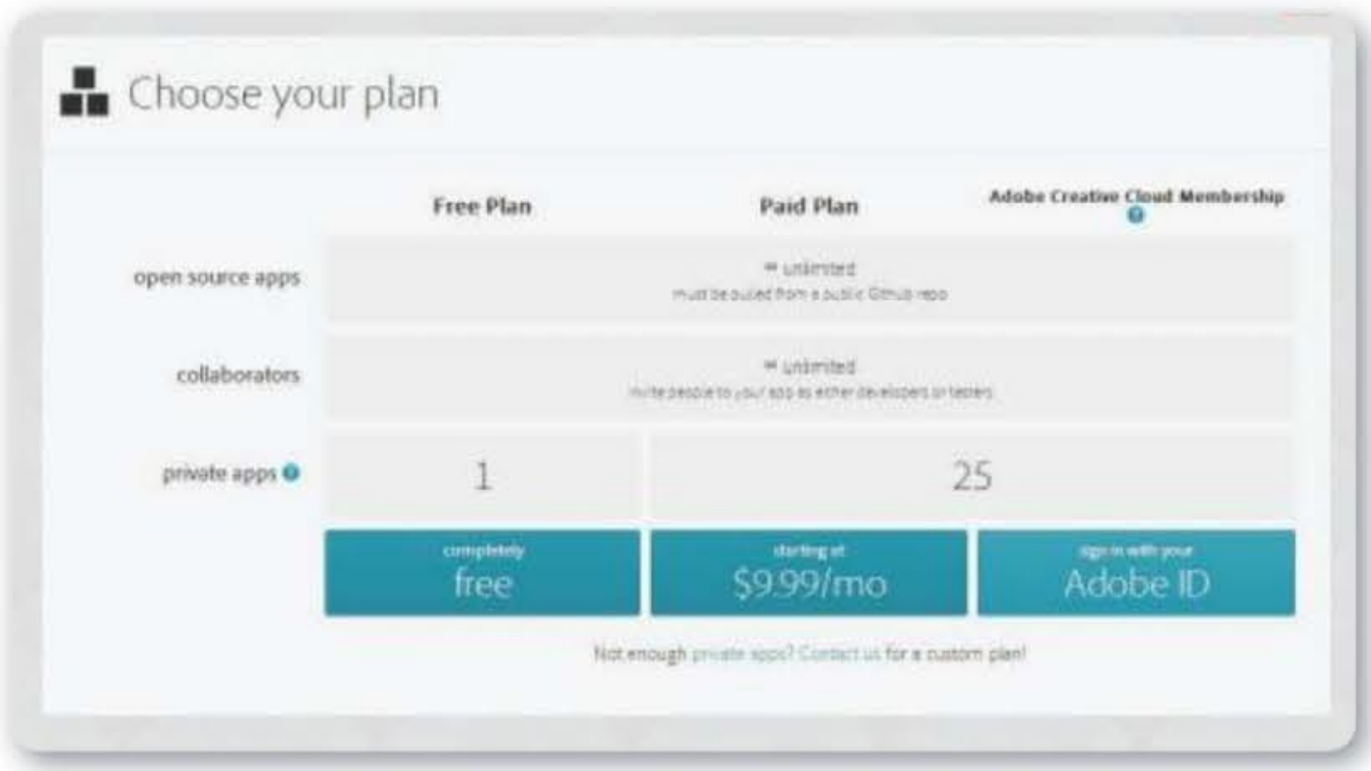


Figura 5. Planes y costos para utilizar el servicio de Adobe PhoneGap Build.

Una vez que hemos seleccionado el tipo de plan por utilizar, ingresamos como usuarios ya registrados, haciendo clic en el botón **Sign in** de la barra de menú principal del sitio. En la pantalla de ingreso que aparece, seleccionamos la cuenta con la cual queremos ingresar, ya sea de Adobe o de GitHub. Una vez que completamos los datos, ya podremos comenzar a subir nuestro proyecto.

Subir un proyecto

Cuando terminamos de realizar nuestro proyecto y ya decidimos con qué plan vamos a acceder a compilar nuestra aplicación con el servicio Adobe PhoneGap Build, lo primero que tenemos que hacer es ingresar al sitio web.

Una vez que ingresamos al sitio web, hacemos clic en el botón **+ new app** ('agregar nueva aplicación'). Se nos va a abrir una página en donde nos encontraremos con dos pestañas de opciones: subir nuestra aplicación **pública** (**open-source**) o **privada** (**private**). Recordemos que eso depende mucho del plan que hayamos adquirido; si es libre, solo vamos a poder tener una aplicación privada e infinitas públicas. En el caso de que elijamos subir una aplicación pública (open source), a diferencia de la privada, tenemos que hacerlo sí o sí mediante la URL de nuestra aplicación en el repositorio de GitHub; en cambio, la privada también nos da la opción de hacerlo mediante un archivo Zip por medio de un clic en el botón **upload a .zip file** ('subir archivo Zip').

> A LA HORA DE AGREGAR NUESTRO PROYECTO, PODEMOS ELEGIR HACERLO EN FORMA PÚBLICA O PRIVADA

En todos los casos, una vez que subimos nuestra aplicación, se nos va a abrir una página en donde ingresamos su nombre (en **Enter your app description**) y, al costado derecho de la página, vamos a visualizar toda la descripción de nuestra aplicación con nuestra dirección de e-mail, ID y formato.

Es muy importante que nos fijemos que esté todo bien y, una vez que hayamos revisado esto, ya podemos hacer clic en el botón **Ready to build** ('listo para construir'). En unos segundos, tendremos nuestra aplicación terminada y lista para ser descargada.

GITHUB

GitHub es un servidor web que nos permite alojar nuestros proyectos y compartirlos con varios desarrolladores que estén trabajando en el mismo proyecto o no. Además, nos permite ver los cambios producidos; es lo que, en programación, se llama **control de versiones**. Para utilizarlo, solo tenemos que registrarnos en su sitio web: **http://github.com**.

Descargar la aplicación

A la hora de descargar la aplicación, debemos tener presente que, en ciertas tiendas, nos van a exigir un ID (identificador) para poder descargarla: es el caso de **Apple** que, para obtenerla, nos exige que estemos registrados como desarrolladores (y esto tiene un costo anual). Sin ese identificador como desarrolladores, no se nos va a permitir descargar la aplicación, y el botón de descarga va a ser de color rojo, como indicador de que hay un problema o error. En cambio, en otras –como, por ejemplo, Android– sí vamos a poder descargarla, aunque también tenemos la opción de agregarle un identificador como desarrolladores.

Haciendo clic sobre los iconos de cada sistema (iOS, Android, Windows Phone, BlackBerry, HP y Symbian), descargamos la aplicación que se necesite, lista para subir a la tienda correspondiente.

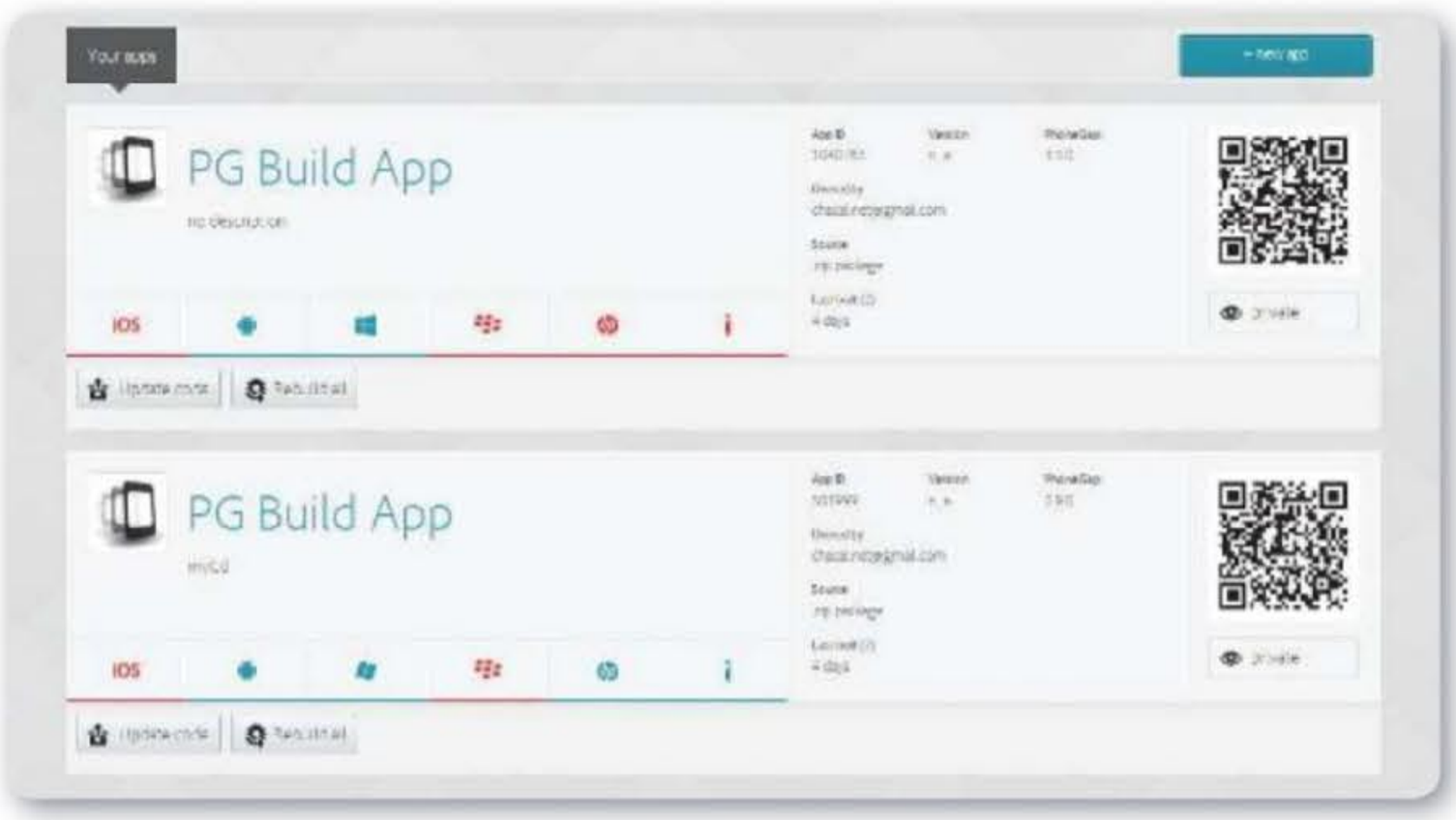


Figura 6. Aplicación terminada y lista para ser descargada desde Adobe PhoneGap Build.

LIBRERÍAS NECESARIAS

Antes de comprimir y subir nuestra aplicación al servicio de Adobe PhoneGap Build, debemos considerar que es indispensable que este tenga todas las librerías, tanto de jQuery Mobile como de PhoneGap, como así también las imágenes, archivos de JavaScript que usemos y todos los archivos adicionales que hacen al proyecto.

Trabajar con Dreamweaver CC

Otra opción para trabajar con PhoneGap y compilar nuestra aplicación es hacerlo directamente con Dreamweaver CC, que nos facilita su uso sin salir del entorno de desarrollo. Lo único que debemos tener es nuestro ID de la cuenta de Creative Cloud e instalar la extensión disponible desde el sitio web de Adobe.

Una vez que instalamos el complemento, tenemos que ir al menú en la opción **Sitio** y luego hacer clic en **Servicio PhoneGap Build**. A partir de ahí, tendremos dos opciones:

- **Servicio PhoneGap Build**: aquí vamos a tener que iniciar sesión con nuestro ID de Adobe y contraseña. Una vez que nos acreditamos como usuarios, podremos crear un proyecto nuevo o trabajar en la lista de proyectos en los que estuvimos trabajando con anterioridad.
- **Configuración de PhoneGap Build**: aquí vamos a configurar el proyecto con la ubicación de sus respectivos SDK.



Figura 7. Podemos introducir nuestra clave de desarrolladores para compilar nuestra aplicación en el servicio Adobe PhoneGap Build.

Cómo comenzar a trabajar

Para comenzar a trabajar con PhoneGap desde Dreamweaver CC, y teniendo todo configurado con la cuenta de Adobe PhoneGap Build, comenzaremos por crear el enlace a la librería de PhoneGap, como se muestra en el siguiente código:

```
<script type="text/javascript" charset="UTF-8" src="phonegap.js"></script>
```

Si bien el archivo **phonegap.js** no se encuentra físicamente en nuestro proyecto, cuando compilamos la aplicación (es decir, cuando la subimos a la página web de Adobe PhoneGap Build), este es agregado directamente al proyecto por el servicio de Adobe. De esta forma, no tendremos que preocuparnos por buscar la última versión.

Otra cosa para tener en cuenta cuando programamos en PhoneGap es que, al ejecutarse la aplicación, estén cargadas las librerías con sus procedimientos y funciones; en el caso contrario, si se llama a una función que escribimos en PhoneGap y no está todavía cargada la librería, no se va a poder ejecutar. Para esto, lo que tenemos que hacer es agregar un script de JavaScript que, primero, espera a que cargue la librería de PhoneGap y, luego, hace que se ejecuten las funciones desarrolladas. Este script utiliza un **listener** ('escuchador', en español) propio de PhoneGap, asociado al evento **deviceready** (que significa, en español, 'cuando el dispositivo está listo'), como podemos observar en el siguiente ejemplo:

```
<script>
        document.addEventListener ("deviceready", iniciarApp, false);

        function iniciarApp(){

        }
</script>
```

Ese listener **deviceready** espera a que cargue la librería y, una vez que está cargada, llama a la función **iniciarApp**. Ahí es donde debemos crear



PHONEGAP BUILD DESDE DREAMWEAVER

Cuando creamos un proyecto nuevo con el servicio de Adobe PhoneGap Build desde Dreamweaver, el sistema nos crea tres archivos: **config.xml**, **index.html** y **Project Settings**. En el archivo **config.xml**, vamos a sustituir los valores que trae por defecto por los de nuestro proyecto, tales como: versión, nombre y descripción, sitio web del autor y nombre. Una vez que cambiamos esos valores, debemos volver a subir la aplicación (clic en el botón **Reconstruir**) para que los cambios surtan efecto.

las funciones o hacer un llamado a estas. Un error muy común es poner directamente las funciones y no esperar la carga de la librería.

Otras opciones: Adobe Edge Code y Bracket

Otra opción para trabajar directamente con el servicio de Adobe PhoneGap Build, es utilizar otro editor, como puede ser **Adobe Edge Code** o **Bracket** (este último, de código abierto). Necesitamos, para ambos editores, bajarnos el complemento (plugin) correspondiente de PhoneGap, desde el **Gestor de extensiones** (**Extension Manager**), e instalarlo.

Una vez que lo instalamos, nos va a aparecer un nuevo botón con el logo de PhoneGap en el costado derecho; allí nos vamos a poder registrar para acceder directamente al servicio.



Figura 8. Desde el editor **Bracket** se puede descargar el complemento (plugin) para trabajar directamente con Adobe PhoneGap Build.

EVENT LISTENER

Un **event listener** –también llamado **escuchador de eventos**, en español– es un objeto de programación que se utiliza para detectar el lanzamiento de un determinado evento. Por ejemplo: si un usuario o el sistema realiza una determinada acción o evento (como tocar la pantalla o un botón), este es registrado por el escuchador de eventos y, a partir de ahí, la aplicación realiza una determinada acción de acuerdo al flujo del programa, previamente programada por nosotros, los desarrolladores.

Extensiones para PhoneGap

Desde la **versión 3** de PhoneGap, como bien dice en su página oficial, la arquitectura del framework cambió, y las APIs que manejan el acelerómetro, las notificaciones y los contactos –entre otras– son manejadas ahora como extensiones (plugins) que el desarrollador deberá incluir si es que desea utilizarlas en su aplicación. Esto es un cambio significativo, ya que ahora tenemos que instalar dichas extensiones en el archivo **config.xml**.

Estas extensiones pueden ser propias del framework o de terceros; debido a esto, hay una gran cantidad de plugins que nos facilitan mucho la tarea de programar para los distintos dispositivos móviles. Incluso, podemos tener nuestro propio repositorio de extensiones, y subir y compartir con la comunidad de desarrolladores las que hayamos creado.

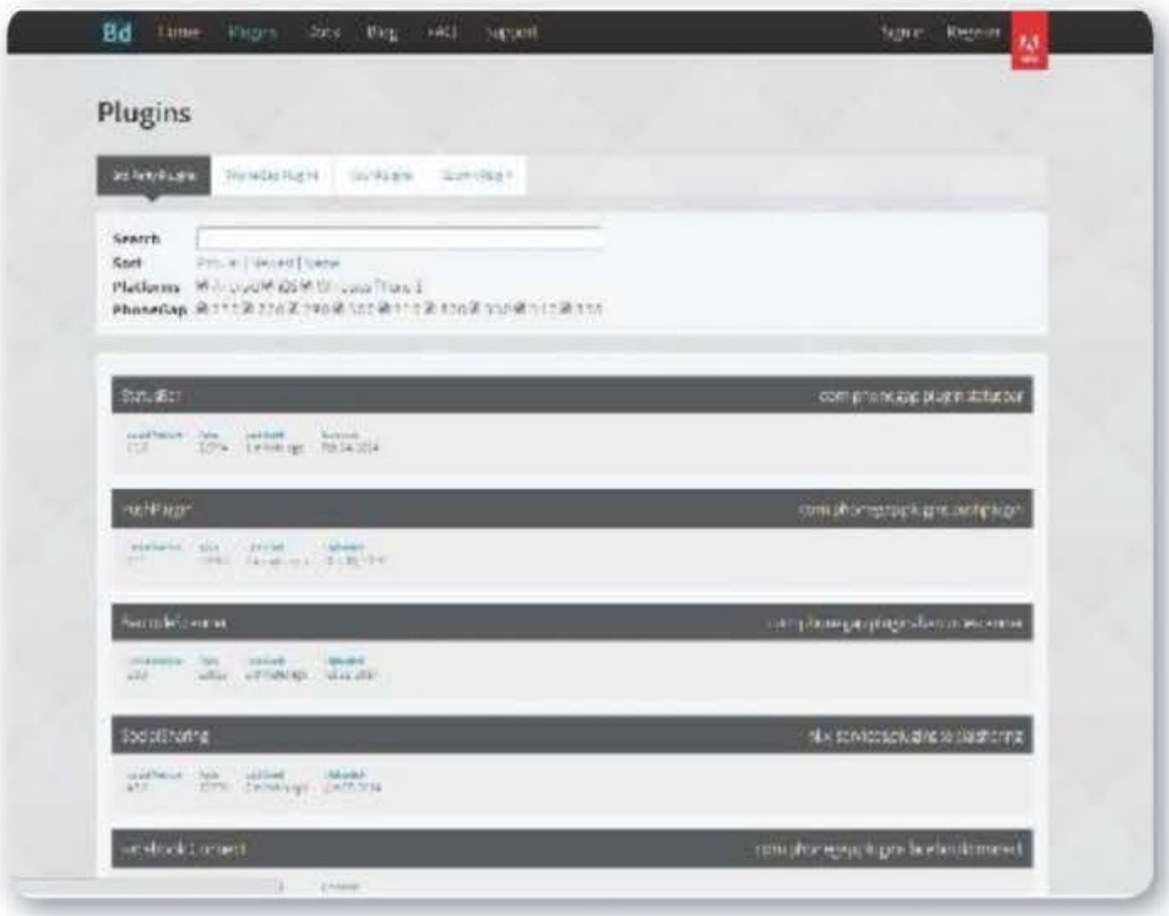


Figura 9. En la página web de Adobe PhoneGap Build, encontramos en el menú **Plugins** una lista con todas las extensiones disponibles.

Cómo instalar extensiones

Para instalar las extensiones tenemos que, simplemente, agregar unas pocas líneas de código al archivo **config.xml** que fue creado cuando armamos el proyecto en Adobe Dreamweaver CC y, luego, volver a reconstruir la aplicación, así surte efecto el cambio.

Por ejemplo: si queremos agregar la extensión para utilizar el acelerómetro, primero tenemos que buscar la extensión en la página de Adobe PhoneGap Build (**https://build.phonegap.com/plugins**), en el menú **PhoneGap Plugins** y, una vez allí, hacer clic sobre **Device motion**, que es la extensión que debemos agregar.

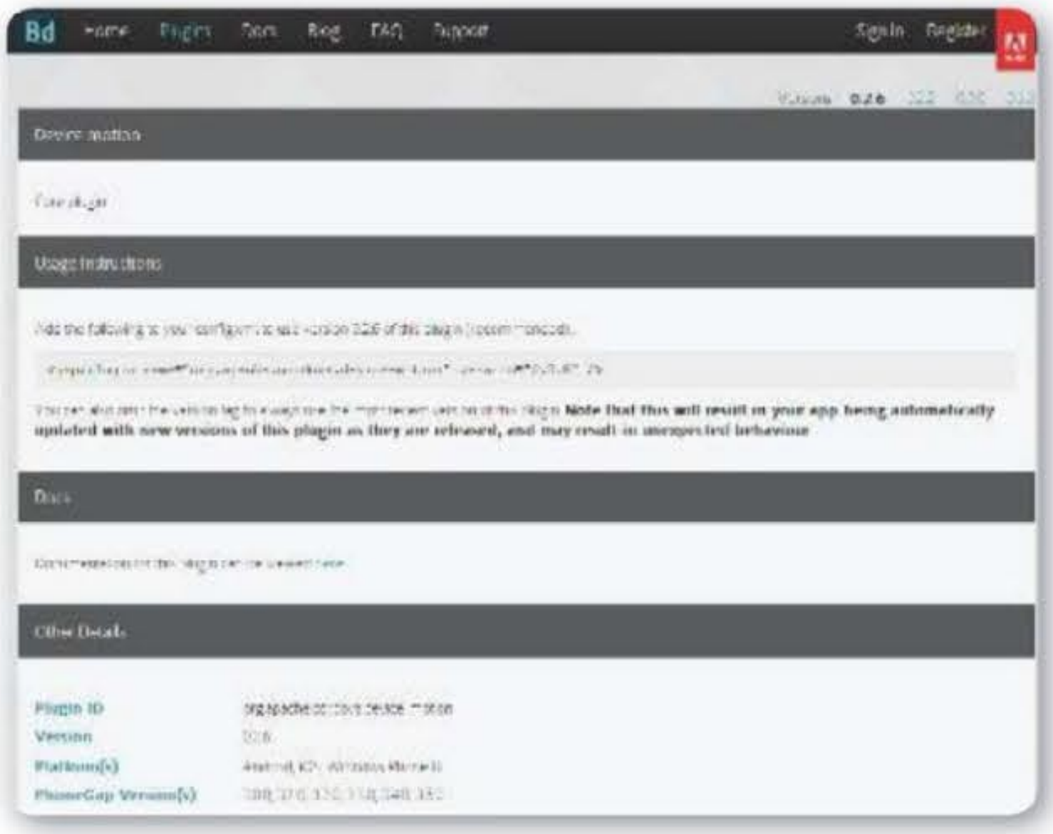


Figura 10. Al seleccionar una extensión, podemos observar sus características y, también, ver y descargar versiones anteriores.

En esa página accederemos a toda la información acerca de la extensión, su documentación y, además, instrucciones para agregarla al proyecto con el código que debemos añadir en el archivo **config.xlm**, como podemos observar en el siguiente código:

```
<gap:plugin name="org.apache.cordova.device-motion" version="0.2.6" />
```

Este puede ser utilizado con la versión estable y recomendada por el framework –en este ejemplo, la versión 0.2.6–, o bien utilizar la última

PLUGIN (COMPLEMENTO)

Un **plugin** es un complemento de un determinado programa, que le aporta una función nueva específica, es decir, le añade características o propiedades adicionales. Por lo general, son programados por desarrolladores externos a los que hicieron la aplicación.

versión, quitándole el número, como se ve en el próximo ejemplo, o usar alguna versión anterior haciendo clic en el costado superior derecho de la página en **Versions** (versiones).

```
<gap:plugin name="org.apache.cordova.device-motion"/>
```



Figura 11. Así quedaría nuestro archivo **`config.xml`** después de que le agreguemos la extensión (plugin) que maneja el acelerómetro.

Extensiones disponibles

En lo que respecta a extensiones propias de PhoneGap (y no de terceros), hasta la edición de este libro contamos con dieciocho. A continuación, las describiremos brevemente:

- **Battery** (batería): nos proporciona datos sobre la batería, en especial, cuando la carga está baja o el dispositivo está desenchufado o enchufado. Para agregar esta extensión, tenemos que incluir el siguiente código en el archivo **config.xml**:

```
<gap:plugin name="org.apache.cordova.battery-status" />
```

- **Camera** (cámara): selecciona imágenes de la biblioteca o abre la aplicación de cámara para capturar una fotografía. Para agregarla, tenemos que incluir este código en el archivo **config.xml**:

```
<gap:plugin name="org.apache.cordova.camera" />
```

- **Capture** (captura): nos proporciona acceso al audio, video e imágenes del dispositivo. Para agregar esta extensión, tenemos que incluir el siguiente código en el archivo **config.xml**:

```
<gap:plugin name="org.apache.cordova.media-capture"/>
```

- **Console** (consola): añade funciones adicionales al depurador del sistema, para evitar que nuestra aplicación tenga errores. Para agregarla, tenemos que incluir este código en el archivo **config.xml**:

```
<gap:plugin name="org.apache.cordova.console" />
```

- **Contacts** (contacto): esta extensión nos permite acceder a la base de datos de contactos del dispositivo. Para agregar esta extensión, tenemos que incluir el siguiente código en el archivo **config.xml**:

```
<gap:plugin name="org.apache.cordova.contacts" />
```

- **Device** (dispositivo): describe en forma general el hardware y software del dispositivo. Para agregar esta extensión, tenemos que incluir el siguiente código en el archivo **config.xml**:

```
<gap:plugin name="org.apache.cordova.device" />
```

- **Device motion** (movimiento del dispositivo): más conocido como acelerómetro, registra el cambio de movimiento del dispositivo en tres dimensiones: X, Y, Z. Para agregar esta extensión, tenemos que incluir el siguiente código en el archivo **config.xml**:

```
<gap:plugin name="org.apache.cordova.device-motion"/>
```

- **Device orientation** (orientación del dispositivo): esta extensión nos proporciona acceso a la brújula del dispositivo; esta mide en grados, y el norte es el grado cero. Para agregar esta extensión, tenemos que incluir el siguiente código en el archivo **config.xml**:

```
<gap:plugin name="org.apache.cordova.device-orientation" version="0.3.5" />
```

- **File** (archivos): nos permite acceder a los archivos del sistema (en inglés, *file system*). Para agregar esta extensión, tenemos que incluir el siguiente código en el archivo **config.xml**:

```
<gap:plugin name="org.apache.cordova.file" />
```

- **File transfer** (transferencia de archivos): nos permite cargar y descargar archivos. Para agregar esta extensión, tenemos que incluir el siguiente código en el archivo **config.xml**:

```
<gap:plugin name="org.apache.cordova.file-transfer"/>
```

- **Geolocation** (geolocalización): es uno de los más usados y nos proporciona datos sobre la ubicación del dispositivo por GPS o por la dirección IP de este. Para agregar esta extensión, tenemos que incluir el siguiente código en el archivo **config.xml**:

```
<gap:plugin name="org.apache.cordova.geolocation" />
```

- **Globalization** (globalización): nos proporciona información sobre la configuración regional y zona horaria del dispositivo. Para agregar esta extensión, tenemos que incluir el siguiente código en el archivo **config.xml**:

```
<gap:plugin name="org.apache.cordova.globalization" />
```

- **Inappbrowser** (navegador en la aplicación): nos muestra un navegador dentro de nuestra aplicación. Para agregar esta extensión, tenemos que incluir el siguiente código en el archivo **config.xml**:

```
<gap:plugin name="org.apache.cordova.inappbrowser" />
```

- **Media** (media): nos permite grabar y reproducir audio. Para agregar esta extensión, tenemos que incluir el siguiente código en el archivo **config.xml**:

```
<gap:plugin name="org.apache.cordova.media" />
```

- **Network information** (información sobre la conexión): nos brinda información sobre la red Wi-Fi del dispositivo, o nos indica si este está conectado a Internet. Para agregar esta extensión, tenemos que incluir el siguiente código en el archivo **config.xml**:

```
<gap:plugin name="org.apache.cordova.network-information" version="0.2.7" />
```

- **Notification** (notificaciones): esta extensión nos permite acceder a la configuración de cuadros de diálogo (notificaciones) nativas del sistema operativo. Para agregar esta extensión, tenemos que incluir el siguiente código en el archivo **config.xml**:

```
<gap:plugin name="org.apache.cordova.dialogs" />
```

- **Splashscreen** (pantalla de inicio): nos permite mostrar y ocultar la pantalla de bienvenida al inicio de la aplicación. Para agregar esta extensión, tenemos que incluir el siguiente código en el archivo **config.xml**:

```
<gap:plugin name="org.apache.cordova.dialogs" />
```

- **Vibration** (vibración): nos permite hacer vibrar el dispositivo. Para agregar esta extensión, tenemos que incluir el siguiente código en el archivo **config.xml**:

```
<gap:plugin name="org.apache.cordova.vibration" />
```



Figura 12. En la página web de Adobe PhoneGap Build podemos encontrar todas las extensiones disponibles.

El emulador Ripple

Para trabajar con dispositivos móviles, siempre es interesante probar nuestra aplicación en un dispositivo real y no en un navegador, como veníamos haciendo hasta ahora; a esto lo llamaríamos **prueba en vivo**. Si bien podemos conectar algún dispositivo con Android, iOS o algún otro sistema operativo, probablemente nunca vamos a tener todos los dispositivos para hacer un test de cómo se vería nuestra aplicación en diferentes resoluciones y sistemas. Sin embargo, contamos con una opción interesante que es un emulador de código abierto, que funciona con el navegador Chrome y se llama **Ripple** (en inglés, **Ripple emulator**).

Cuando utilizamos PhoneGap, es recomendable usar un emulador para poder probar funciones específicas de sistema y del dispositivo. Un emulador es un programa o software que emula o imita la función de un programa o sistema, en una plataforma distinta de la que fue creado. Un ejemplo típico es el de algunos videojuegos que solo funcionan en una plataforma en particular, como los juegos de **PlayStation**: si quisiéramos jugar en nuestra computadora, deberíamos usar un emulador, ya que es una PC y no un reproductor de juegos para PlayStation. O bien, si quisiéramos emular la vieja Commodore 64 (computadora de 8 bits del año 1982), tendríamos que instalarnos en nuestra computadora, por ejemplo, el emulador **Frodo**.



Figura 13. Para utilizar el emulador **Ripple**, una vez instalada la extensión, debemos habilitarlo desde el botón **Enable**.

En lo que respecta a los dispositivos móviles, existen varios emuladores, como el ya mencionado Ripple. Para instalar este emulador, tenemos que buscarlo en la tienda de Google (**Google Web Store**) como una extensión o plugin para Google Chrome. Para utilizarlo, debemos hacerlo desde el navegador mediante un clic en el botón de Ripple ubicado a la derecha de la pantalla y otro en la opción **Enable** (habilitar).

Una vez que lo habilitamos, se nos va a abrir una ventana donde tenemos, entre otras opciones, un botón **Mobile Web (default)** (web móvil, por defecto). Al hacer clic sobre este, nos va a dirigir a la misma página web que estábamos viendo antes de habilitar el emulador Ripple, pero esta vez en formato de dispositivo móvil, con la forma y entorno propios de Ripple.

Entre las opciones que disponemos sobre los paneles flotantes tenemos:

- **Devices** (dispositivos): es una lista desplegable que nos ofrece una gran cantidad de dispositivos móviles para emular (iPhone, BlackBerry, HTC, entre otros). También contamos con la opción de **orientation** (orientación), donde podemos elegir la orientación del equipo, ya sea vertical u horizontal.
- **Platform** (plataforma): aquí podemos cambiar de plataforma de prueba (PhoneGap, BlackBerry, WebWorks y Mobile Web). Por defecto usamos **Mobile Web**, pero en caso de querer cambiarla tenemos que hacer clic en el botón **Change Platform** (cambiar plataforma).
- **Information** (información): en este panel se encuentra toda la información del dispositivo que estamos emulando (plataforma, sistema operativo, resolución de pantalla, tipo de dispositivo, entre otras opciones).
- **Accelerometer** (acelerómetro): esta quizá sea una de las opciones más interesantes a la hora de desarrollar, ya que nos permite emular la movilidad de nuestro dispositivo para saber cómo se comporta



GEOLOCALIZACIÓN (GEO LOCATION)

Es la posibilidad de saber en dónde se encuentra ubicado físicamente un objeto. Puede expresarse con dos coordenadas, latitud y longitud, sobre la ubicación en un mapa. Es muy utilizado en tecnología móvil para determinar la ubicación de dispositivos. Los datos sobre localización pueden ser por **GPS** (**Global Position System**) o pueden obtenerse por medio de la red (utilizando la IP del dispositivo conectado).

nuestro programa de acuerdo con esos cambios de movimiento. Solamente haciendo clic sobre la figura del dispositivo que se encuentra en este panel y sin soltar el mouse cuando movamos el dispositivo, este se irá girando e irán cambiando las coordenadas que figuran debajo de él. Incluso, si hacemos clic en el botón **shake** (agite), el dispositivo se va a mover como si lo estuviéramos agitando.

- **Geo Location** (geolocalización): desde este panel, es posible pasar las coordenadas con las que queremos ubicar el dispositivo en un determinado lugar, o bien ubicarlo en el mapa que se encuentra en la parte inferior del panel.



Figura 14. Con el emulador Ripple podemos cambiar la configuración del dispositivo que emulamos, y hasta es posible, entre otras cosas, utilizar el acelerómetro.

RESUMEN

En este capítulo, aprendimos a crear aplicaciones para dispositivos móviles multiplataforma empleando Adobe PhoneGap Build. Explicamos las diferencias que existen entre PhoneGap y Adobe PhoneGap Build, y cómo descargar las librerías de PhoneGap. A continuación, vimos cómo compilar una aplicación y descargarla utilizando los servicios de Adobe PhoneGap Build, como también de Adobe Dreamweaver CC, Adobe Edge Code y Bracket. Además, aprendimos a instalar extensiones de Adobe PhoneGap Build y de PhoneGap, y, por último, a utilizar un emulador como Ripple.

Actividades

TEST DE AUTOEVALUACIÓN

1. ¿Es necesario saber programar en cada lenguaje nativo para desarrollar una aplicación en PhoneGap?
2. Al descargar PhoneGap desde su sitio web oficial, ¿es necesario trabajar con todas las librerías de todos los sistemas operativos disponibles?
3. ¿Qué diferencia hay entre PhoneGap y Adobe PhoneGap Build?
4. El servicio de Adobe PhoneGap Build, ¿es gratuito?
5. ¿Cuántas aplicaciones públicas puedo subir a la página web de Adobe PhoneGap Build?

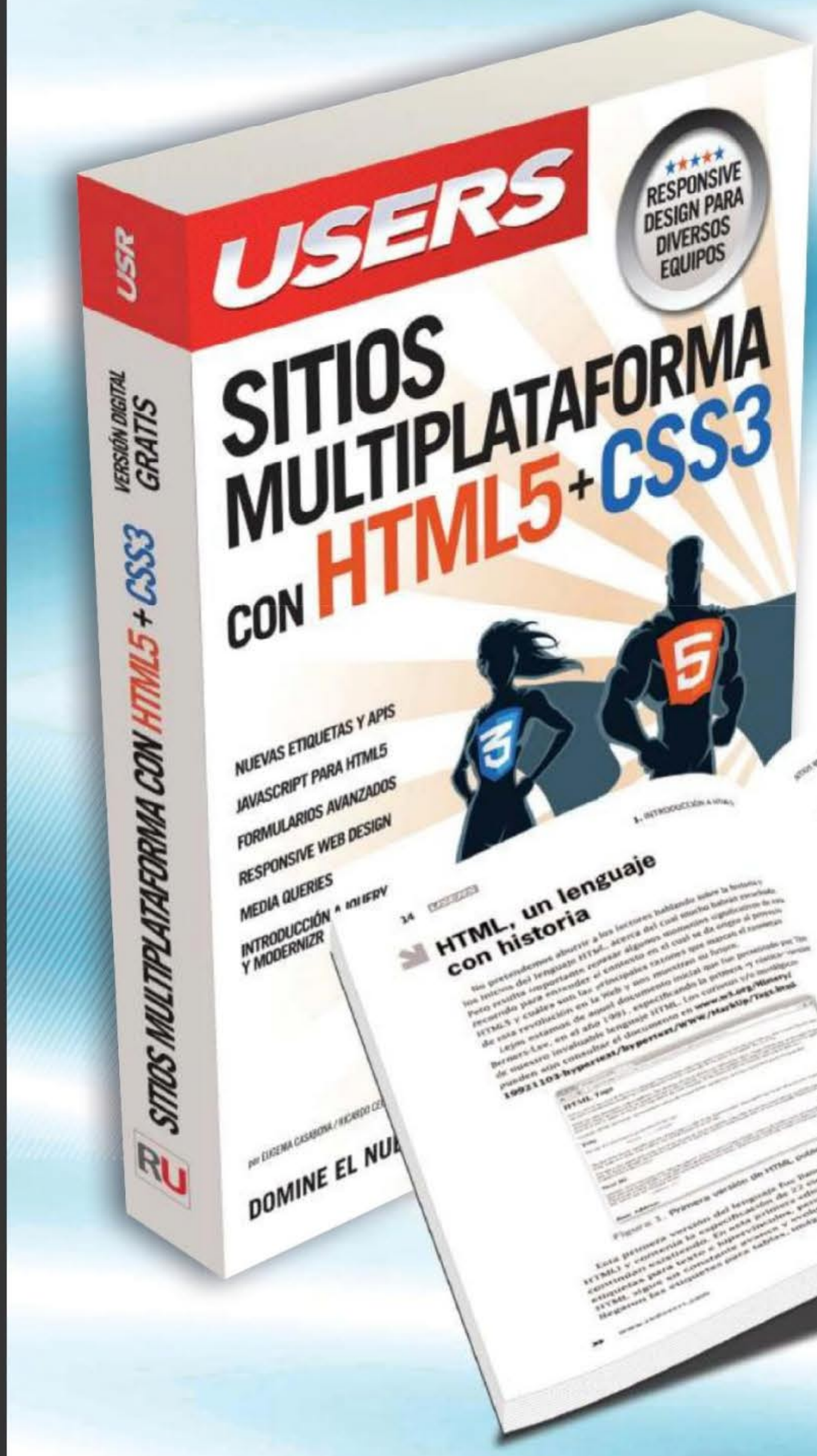
EJERCICIOS PRÁCTICOS

1. Realice una página en jQuery Mobile que tenga dos botones: uno de ellos que llame a otra página y el otro que cierre la aplicación.
2. Comprima los archivos generados en el ejercicio anterior en formato .ZIP.
3. Regístrese en el sitio web de Adobe PhoneGap Build para poder acceder al servicio.
4. A partir del archivo Zip del ejercicio 2, súbalo a la página web de Adobe PhoneGap Build para compilar la aplicación con su usuario creado en el ejercicio 3.
5. Descargue la aplicación generada en el ejercicio 4 para el sistema operativo Android.

PROFESOR EN LÍNEA

Si tiene alguna consulta técnica relacionada con el contenido, puede contactarse con nuestros expertos: profesor@redusers.com.

CONÉCTESE CON LOS MEJORES LIBROS DE COMPUTACIÓN



La potencia de HTML5, CSS3 y JavaScript permite realizar sitios interactivos, de alto impacto visual y excelente performance.

» DESARROLLO / DISEÑO WEB
» 352 PÁGINAS
» ISBN 978-987-1949-45-8

LLEGAMOS A TODO EL MUNDO VÍA OCA* Y DHL**

MÁS INFORMACIÓN / CONTÁCTENOS

usershop.redusers.com +54 (011) 4110-8700 usershop@redusers.com

*SÓLO VÁLIDO EN LA REPÚBLICA ARGENTINA // **VÁLIDO EN TODO EL MUNDO EXCEPTO ARGENTINA

USERS

jQuery Mobile

Este libro está dirigido a todos aquellos que quieran incursionar en el desarrollo de aplicaciones para dispositivos móviles, sin la necesidad de contar con conocimientos acerca del lenguaje nativo de cada sistema operativo. A lo largo de la obra conoceremos los conceptos básicos de programación en jQuery Mobile, que nos permitirán desarrollar nuestras propias aplicaciones, para luego generar una aplicación multiplataforma con Adobe PhoneGap Build. A través de explicaciones sencillas y ejemplos variados, los lectores aprenderán desde la instalación de las librerías de jQuery Mobile hasta la creación completa de una aplicación web híbrida.

> **Es posible crear una aplicación o sitio web para móviles sin la necesidad de programar en el lenguaje nativo de cada sistema operativo.**

* EN ESTE LIBRO APRENDERÁ:

- **Mobile web:** conceptos básicos sobre las aplicaciones web. Tipos de aplicaciones web, principales sistemas operativos para dispositivos móviles y entornos de desarrollo de jQuery Mobile.
- **Creación de una página en jQuery Mobile:** instalación y uso del framework. Organización de los proyectos con Dreamweaver CC. Estructura de una página básica y principales componentes.
- **Botones, listas y formularios:** configuración de botones y diferentes tipos de listas. Funcionamiento y utilidad de los formularios para introducción de datos.
- **Otros componentes de la interfaz de usuario:** creación de una interfaz que permita mostrar la información en una cantidad limitada de espacio.
- **Temas:** aplicación de muestras de color a los elementos de la página mediante temas personalizados o preestablecidos.
- **PhoneGap:** introducción a PhoneGap y Adobe PhoneGap Build. Compilación de la aplicación con Adobe PhoneGap Build para lograr su compatibilidad con todos los dispositivos móviles.





SOBRE LOS AUTORES

Luis Hereter es analista de sistemas de computación, profesor de Sistemas y desarrollador de aplicaciones móviles y de escritorio.

Viviana Zanini es analista de sistemas de computación y profesora de Informática. Ha sido autora de varias publicaciones en esta misma editorial.

NIVEL DE USUARIO

Intermedio

CATEGORÍA

Desarrollo / Internet / Mobile

ISBN: 978-987-734-004-4

REDUSERS.com
En nuestro sitio podrá encontrar noticias relacionadas y también participar de la comunidad de tecnología más importante de América Latina.

PROFESOR EN LÍNEA
Ante cualquier consulta técnica relacionada con el libro, puede contactarse con nuestros expertos: profesor@redusers.com.