UNIVERSIDAD POLITECNICA DE VALENCIA

ESCUELA POLITECNICA SUPERIOR DE GANDIA

LICENCIADO EN COMUNICACIÓN AUDIOVISUAL

UNIVERSIDAD
POLITECNICA
DE VALENCIA

ESCUELA POLITECNICA
SUPERIOR DE GANDIA

"Desarrollo de un lector de noticias RSS multiplataforma"

TRABAJO FINAL DE CARRERA

Autor:
Juan Collado Navarro

Directores:
Francisco de Zulueta Dorado
Daniel Palacio Samitier

GANDIA, 2013

Agradecimientos

En primer lugar, me gustaría dar las gracias a Editorial Prensa Valenciana, S.A. por permitirme utilizar las fuentes RSS del especial de Motor; éstas han sido de gran valor para ilustrar este trabajo escrito y el producto final desarrollado. Y en segundo lugar, agradecer muy especialmente a Óscar García, responsable técnico web y tutor de mis prácticas en la mencionada empresa, por darme la oportunidad de trabajar con él en el desarrollo de varias aplicaciones en HTML5 para móviles. Por último, expresar mi gratitud a los directores de mi PFC, Francisco de Zulueta y Daniel Palacio, por toda la ayuda que han depositado.

Contenidos

Introducción

La penetración de Internet móvil ha empujado a los medios de comunicación a desarrollar aplicaciones para servir sus contenidos en las plataformas móviles más importantes. Como veremos, en el mercado existen varios tipos de aplicaciones para móviles con sus ventajas e inconvenientes, y que se ajustan en menor o mayor medida a determinados casos.

La idea de realizar un proyecto de estas características ha surgido durante un periodo de prácticas de seis meses que realicé en Editorial Prensa Valenciana, S.A. , empresa responsable de un importante periódico de la comunidad, Levante -El Mercantil Valenciano-. En adelante nos referiremos a tal entidad simplemente como Levante EMV.

En esta enriquecedora experiencia, tuve la oportunidad de desarrollar algunas aplicaciones para móviles mediante los estándares web (HTML5, JavaScript y CSS3) que aprovechasen los flujos de información existentes de la versión digital del periódico, levante-emv.com. Entre los productos desarrollados, destacan el mapa fallero de las Fallas 2013 y lector RSS para el canal Motor.

Se da por hecho que el lector cuenta con nociones básicas de estándares web porque esta memoria explicativa no pretende ser un exhaustivo tutorial paso a paso de como desarrollar una aplicación para móviles; ya que esto último excedería el límite de horas del PFC. El cometido de este escrito es presentar un proyecto de carácter interactivo y explicar las decisiones tomadas para llevarlo a cabo, las fases de desarrollo que implica y, en general, algunos de los conocimientos aprendidos durante las prácticas en Levante EMV.

Objetivos

Con este proyecto, mi objetivo principal es diseñar y programar mediante los estándares web (HTML5, JavaScript y CSS3) una aplicación multiplafatorma que interprete contenidos RSS o Atom de un servidor de noticias y demostrar que este tipo de aplicaciones son las idóneas para un medio de comunicación digital.

En concreto, mis propósitos en la realización de este trabajo son:

- Diferenciar los tipos de aplicaciones que hay en el mercado móvil y elegir la más conveniente para un medio de comunicación digital.
- Ahondar en los conocimientos sobre programación web adquiridos en las asignaturas “Comunicación Interactiva” y “Taller de Productos Interactivos”.
- Experimentar las posibilidades que ofrecen los estándares web (HTML5, JavaScript y CSS3) para el desarrollo de aplicaciones interactivas para navegador.
- Familiarizarse con las tecnologías RSS y Atom, formatos de fuente web basado en XML para difundir y compartir información actualizada de *websites* y *blogs*. Los servidores de noticias, entre otros, utilizan estos lenguajes.
- Emplear la API Google Feed para la descarga de medios RSS y Atom; librerías como jQuery y jQuery Mobile para el diseño de la interfaz gráfica; y el framework Adobe Phonegap para la conversión de la aplicación web final en una aplicación híbrida para Android.

Tipos de aplicaciones para móviles

No podemos hablar sobre nuestra aplicación sin antes explicar los distintos tipos de aplicaciones dirigidas a móviles que existen en la actualidad: nativas, web e híbridas.

Aplicaciones nativas

Técnicamente, las aplicaciones nativas son ficheros ejecutables independientes que previa instalación, interactúan directamente con el sistema operativo. Dicho de otro modo, tras su inicialización, la aplicación nativa interactúa con el sistema operativo móvil, sin ningún intermediario o contenedor.

Pueden tener acceso a todos los recursos del teléfono y todas las API's del sistema operativo: cámara, notificaciones, contactos, calendario, geolocalización... Aprovechan todo el potencial del sistema operativo y el dispositivo. Tienen un desempeño veloz, arrancan con rapidez y poseen un control preciso de los gestos táctiles.

Generalmente se distribuyen en las tiendas oficiales de cada plataforma: Google Play, Apple App Store... Avisan cuando hay actualizaciones disponibles. El usuario puede o no actualizar la aplicación.

El código escrito para una plataforma móvil no se puede reutilizar para otra ya que cada sistema operativo necesita un lenguaje de programación diferente (Objective C, Java...). El programador no sólo debe conocer estos lenguajes sino también las librerías propias del dispositivo. Esto hace que el desarrollo y mantenimiento requieran mucho tiempo y recursos.

Aplicaciones web

Técnicamente las aplicaciones web son ficheros que son ejecutados por el navegador nativo del dispositivo, no directamente por el sistema operativo. La aplicación está contenida en un navegador que le sirve de intermediario con el sistema operativo.

El acceso a los recursos del teléfono está limitado. El navegador ya es de por si, una aplicación nativa que puede acceder a algunos recursos del dispositivo como la geolocalización o el almacenamiento, gracias al potencial de HTML5. Con las continuas y prometedoras mejoras del estándar HTML5, los desarrolladores son capaces de crear aplicaciones cada vez más avanzadas. Estas aplicaciones suelen tener un desempeño más lento, tardan más en arrancar y el control no es tan preciso como en una nativa.

No se distribuyen en tiendas de aplicaciones pero pueden ser indexadas por motores de búsqueda. Se actualizan automáticamente y de forma centralizada: todos los dispositivos reciben la misma actualización.

La principal ventaja de estas aplicaciones son el soporte multiplaforma y el desarrollo económico. El empleo únicamente de estándares web -HTML5, CSS3 y Javascript- las convierten en productos poco costosos en tiempo y recursos. Se ejecutan en el navegador de cada dispositivo con un código generalmente común.

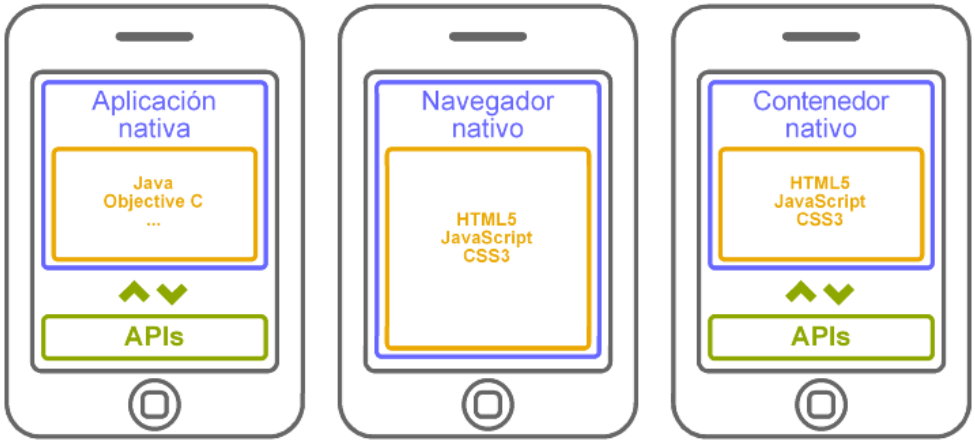
Aplicaciones híbridas

Las aplicaciones híbridas combinan las ventajas del desarrollo nativo con las de la tecnología web. Constan de un contenedor nativo ejecutado por el sistema operativo que interpreta contenido web. Técnicamente, la parte nativa de la aplicación utiliza la API del sistema operativo para crear un motor de representación HTML embebido que sirva de puente entre el lenguaje web y el sistema operativo.

El acceso a las API's del sistema no está limitado y el desarrollador puede acceder a la mayoría de recursos del dispositivo. Estas aplicaciones tienen un mejor rendimiento con respecto a las aplicaciones web y una experiencia de usuario cercana a las aplicaciones nativas.

Se pueden distribuir en tiendas oficiales de aplicaciones. El contenido web (HTML, JavaScript, CSS y archivos multimedia) puede ser cargado desde ficheros locales o desde Internet. En el primer caso, la aplicación puede requerir que el usuario acepte actualizaciones periódicamente pero el rendimiento será mejor. En el segundo caso, la aplicación no necesitaría que el usuario la actualice pero ésta necesitaría siempre acceso a Internet. Se puede combinar ambas posibilidades y hacer que la aplicación cargue solo algunos recursos desde el servidor y otros queden almacenados localmente para un mejor rendimiento.

Como en las aplicaciones web, la principal ventaja de estas aplicaciones son el soporte multiplaforma y el desarrollo económico. Los desarrolladores escriben la mayoría del código de la aplicación en lenguajes web multiplataforma y pueden elegir entre implementar su propio puente para la parte nativa de cada plataforma o aprovechar soluciones ya preparadas, como Phonegap.



De izquierda a derecha: esquema de una aplicación nativa, web e híbrida.

El mejor enfoque

Como hemos visto, el enfoque nativo sobresale por el rendimiento y el acceso a los recursos del dispositivo pero como contrapunto, supone mayores costes de desarrollo y mantenimiento. Por otro lado, el enfoque web es más sencillo de implementar, menos costoso y más fácil de actualizar pero su funcionalidad y experiencia de usuario está más limitada. El enfoque híbrido trata de aprovechar las bondades de los dos enfoques mencionados y en muchas ocasiones, es el idóneo si se quiere desarrollar un producto para diferentes sistemas operativos móviles.

Escenarios para un **desarrollo nativo**:

- Empresas que ya cuentan con un equipo de programadores con conocimientos en varios lenguajes nativos. En este sentido, tendrían escasos costes de inversión por formación o contratación de nuevo personal.
- Aplicaciones dirigidas exclusivamente a un sistema operativo móvil.
- Aplicaciones que requieren funcionalidades nativas que carecen de una sólida implementación en HTML5.
- En aplicaciones complejas como juegos que cuentan con una interfaz de usuario enriquecida y que capacitan al usuario a dar una respuesta en tiempo real. En este aspecto, HTML5 no ha evolucionado lo suficiente para dar una solución equivalente.

Escenarios para un **desarrollo web**:

- Empresas que prefieren una distribución directa de sus aplicaciones sin pasar por la aprobación de las tiendas oficiales.
- Empresas que prefieren actualizar su aplicación de forma centralizada y automatizada.
- Versiones experimentales de aplicaciones cuya calidad en el resultado final puede ser decisiva para decantarse por una versión híbrida en lugar de una nativa.
- Mayor visibilidad y audiencia en la web que en una tienda de aplicaciones debido a los motores de búsqueda.
- Aplicaciones que fundamentalmente se basen en contenidos actualizados desde Internet como las de lectura de noticias.

Escenarios para un **desarrollo híbrido**:

- Empresas que busquen un desarrollo y actualización de su aplicación centralizados.
- Empresas que cuenten con desarrolladores familiarizados con los lenguajes web: HTML, CSS y JavaScript.
- La confianza en HTML5 debido a su creciente compatibilidad y a sus prometedoras funcionalidades. Las principales marcas utilizan WebKit, el motor de renderizado de HTML5 diseñado principalmente por Google y Apple. Optar por HTML5 parece una inversión segura porque probablemente se llegue a convertir en la tecnología estándar para el desarrollo de aplicaciones del lado de cliente.
- Aplicaciones que fundamentalmente se basen en contenidos actualizados desde Internet como las plataformas de compras *online*.

Por tanto, el proceso de elegir un enfoque de desarrollo para una aplicación móvil implica varios parámetros como el presupuesto, el tiempo disponible, el publico objetivo, el conocimiento de los programadores, los equipos necesarios o las características de la propia aplicación.

Considerando que para un medio de comunicación digital, lo importante es llegar al máximo número de usuarios posibles (es decir, a los máximos dispositivos) y que además suelen contar en plantilla conocedora de estándares web, no así de lenguajes de programación nativos; las aplicaciones web e híbridas parecen soluciones acertadas para ellos. En este sentido, puede estar justificado restar algo de fluidez a la aplicación para hacerla llegar a más dispositivos con menos costos y en menos tiempo.

Plan de trabajo

Tiempo necesario

El proceso de realización de este proyecto conlleva varias fases. En la siguiente tabla se recogen cada una de ellas con una breve descripción y una estimación de las horas dedicadas.

Análisis Análisis de las aplicaciones en el mercado que cumplen los mismos o similares objetivos.	5 horas
Wireframes Esquema de la interfaz de usuario: aspecto gráfico y funcional de las pantallas y controles necesarios.	10 horas
Preparación Explicación de las tecnologías necesarias. Elección de las librerías o *frameworks* de JavaScript más adecuados para abordar el proyecto, así como las fuentes RSS que nutrirán la aplicación.	30 horas
Diseño Creación del logotipo y los iconos; y desarrollo de las pantallas de la aplicación mediante JQuery Mobile y hojas de estilos (CSS3).	20 horas
Programación Programación de la aplicación mediante la API Google Feed y las librerías JQuery y JQuery Mobile, entre otras.	90 horas
Pruebas y compatibilidad Pruebas de la aplicación web en varias dimensiones y resoluciones de pantalla (tanto en navegadores de escritorio como de dispositivos móviles).	30 horas
Compilación para Android Adaptación de la aplicación al *framework* Adobe Phonegap para permitir su posterior compilación en una aplicación nativa para Android.	15 horas
Total de horas de trabajo personal	200 horas

Material necesario

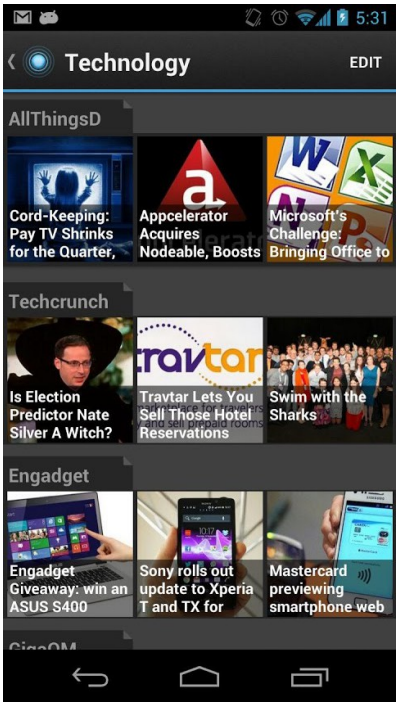
Para la elaboración de este proyecto, se va a utilizar el siguiente material y licencias:

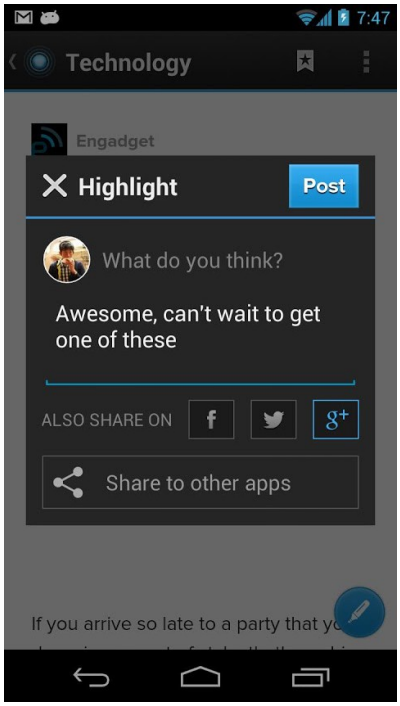
- **Hardware:**
  - Ordenador portátil ASUS U36S con el sistema operativo Windows 7
  - Dispositivo móvil SONY Xperia U con el sistema operativo Android 2.3.7 (para testeo)
  - Tableta ASUS Transformer con el sistema operativo Android 3.2 (para testeo)
  - iPod Touch con el sistema operativo IOS 6.1.3 (para testeo)
- **Sofware comercial:**
  - Adobe Photoshop CS3
  - Adobe Illustrator CS3
- **Software libre:**
  - Sublime Text Editor 2
  - Google Chrome 28
  - Mozilla Firefox 22
  - Windows Internet Explorer 10
  - Apache OpenOffice 3.4.1
- **Librerías JavaScript libres**
  - jQuery 1.9.1
  - jQuery Mobile 1.3.1
  - iScroll 4.2.5
  - Google Feed Api 1
  - Adobe Phonegap 2.9.0
- **Herramientas online**
  - Mockingbird para crear wireframes(https://gomockingbird.com/)
  - Convertidor de iconos (http://www.favico.com/)
  - Simulador de resoluciones de pantalla (http://mattkersley.com/responsive/)
  - Adobe Phonegap Build para compilar para Android (https://build.phonegap.com/)
- **Recursos libres**
  - Fuente tipográfica Open Sans (http://www.google.com/fonts/specimen/Open+Sans)

Fases de desarrollo

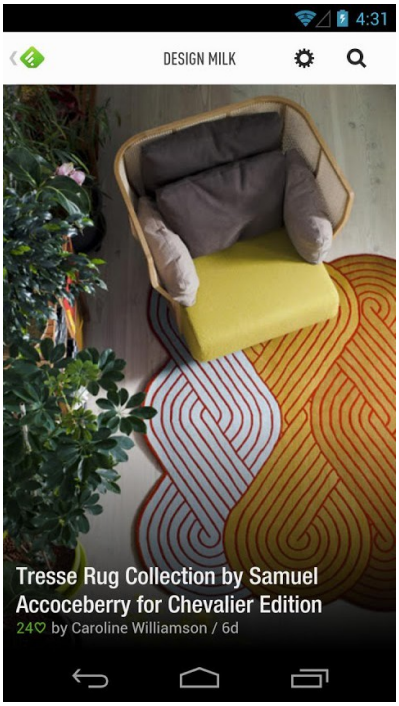
Análisis

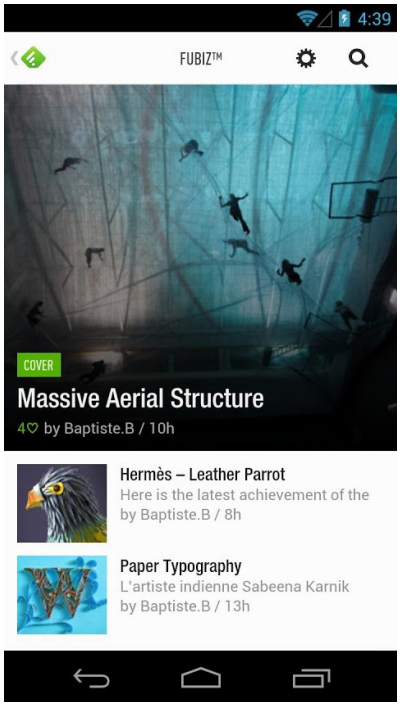
Para poder desarrollar esta aplicación, se debe conocer que aplicaciones ya existen en Google Play, la tienda oficial de aplicaciones de Android, que cumplen con iguales o similares objetivos: hay que considerar lo que hace la competencia.

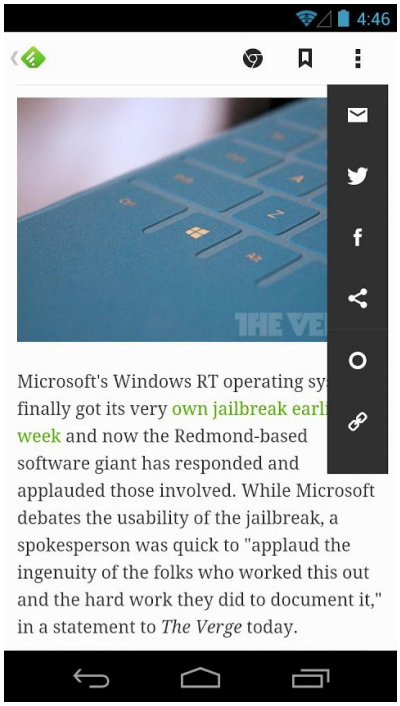




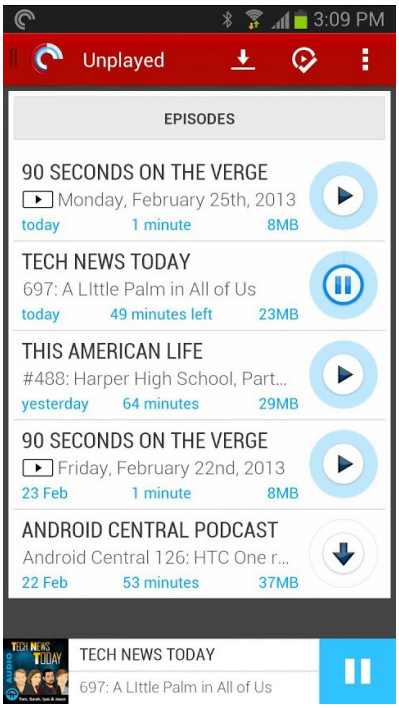
Pulse News es una aplicación para la lectura de noticias con numerosas fuentes, integración con redes sociales y salvado para posterior lectura. También te permite sincronizar con tu cuenta de Google Reader. Lo interesante en esta aplicación es el distinto tratamiento gráfico de las pantallas de los canales con respecto a las pantallas de lectura de noticias; siendo las primeras grises con textos claros, y las segundas blancas con textos oscuros.

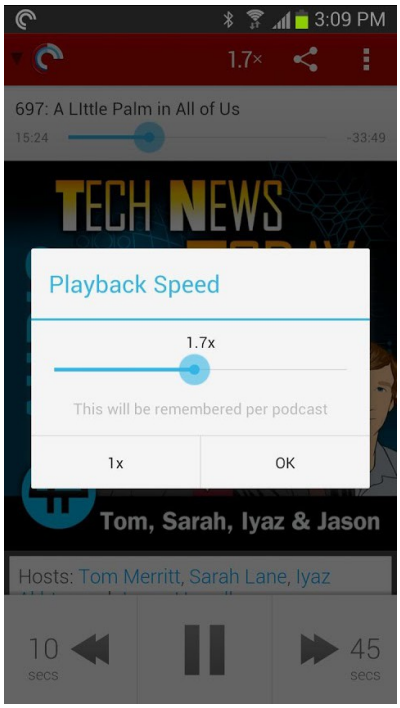






Feedly es un ágil lector de noticias que te permite añadir tus fuentes RSS favoritas de *websites* y *blogs*, y sincronizar con Google Reader. Esta aplicación destaca por su interfaz limpia, clara e intuitiva.





Pocket Casts es una aplicación que te permite reproducir tus podcast favoritos. Los podcasts son información multimedia actualizada y compartida. El apartado visual de esta *app* también destaca por su limpieza y sencillez.

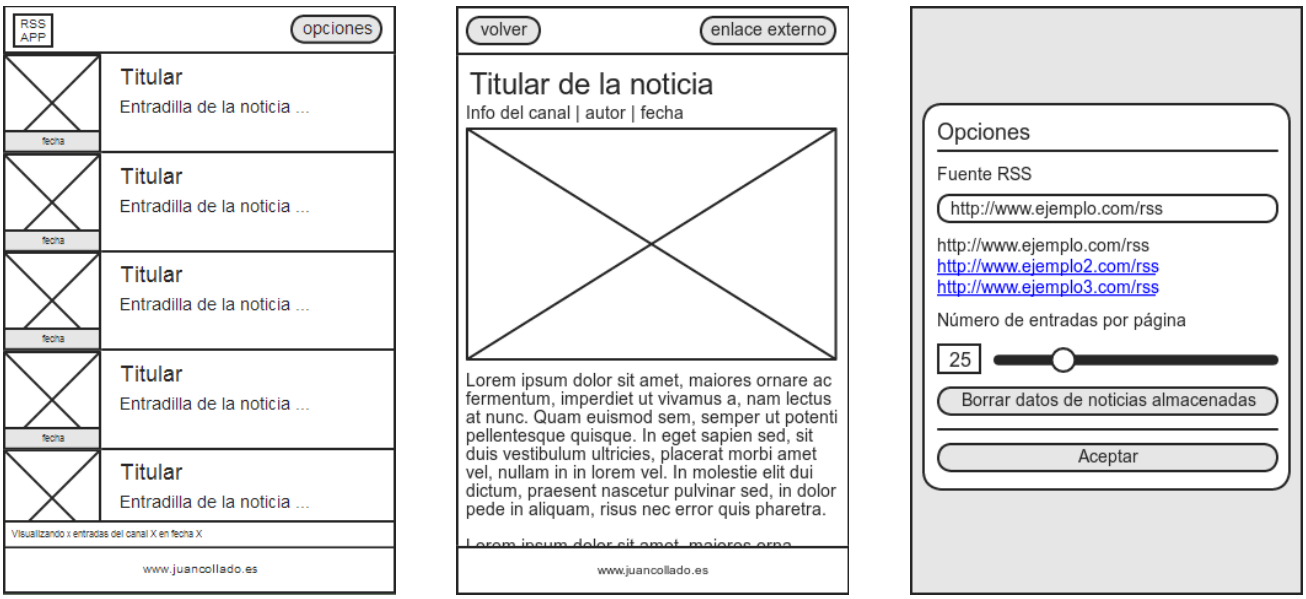
Como hemos visto en los ejemplos mostrados, este tipo de aplicaciones se articulan en varios tipos de pantallas: unas listan fuentes, canales y noticias; otras detallan la noticia u otro contenido seleccionado por el usuario; etc. Además, cuentan con otras pantallas auxiliares para configurar los canales y realizar otros ajustes de la aplicación; y para compartir en redes sociales, etc.

Wireframes

Los *wireframes* son esquemas de las pantallas o páginas que forman la interfaz de una aplicación o sitio web. Son importantes para considerar la funcionalidad, comportamiento y jerarquía de los contenidos, y los elementos que constituyen la interfaz gráfica, sin ser éstos definidos gráficamente de forma precisa. Es decir, los *wireframes* priorizan lo que se hace en la pantalla, no cómo se ve.

El diseño mediante *wireframes* constituye una etapa fundamental en la preproducción de cualquier producto de carácter interactivo en general, y en este proyecto en particular. Es aquí dónde vamos a determinar el flujo de información de la aplicación; el número y tipo de pantallas; así como la posición de los elementos y los principales botones u otros controles.

Básicamente, el **aspecto gráfico** de la aplicación se basará en dos pantallas principales: una para el listado de noticias, y otra para la lectura en detalle de cada noticia. La interfaz gráfica se completará con una tercera pantalla de opciones que permitirá al usuario definir la fuente RSS o Atom que desea cargar, el número de entradas, así como borrar las noticias almacenadas por la aplicación.



De izquierda a derecha: pantalla de listado de noticias, pantalla de noticia detallada y pantalla de opciones o configuración. Estos *wireframes* se han diseñado con la herramienta *online* Mockingbird (https://gomockingbird.com).

El usuario ya está familiarizado con las aplicaciones para *smartphones* y *tablets;* y por tanto, con determinados gestos táctiles y comportamientos para interactuar con éstas. En cuanto al **apartado funcional** de la aplicación, queremos que la aplicación cuente con las siguientes características y controles:

- Ajuste del contenido a las dimensiones y orientación de la pantalla del dispositivo.
- Fecha amigable en el listado de noticias: hace * horas, hace * días, hace * meses...
- Deslizamiento hacia abajo (*swipe down*) para actualizar el listado de noticias.
- Deslizamiento hacia arriba (*swipe up*) para cargar entradas anteriores en el listado de noticias.
- Deslizamiento horizontal (*swipe right* y *swipe left*) en la noticia completa para ver la noticia posterior o anterior (tipo carrusel).
- Pulsación prolongada (*long press*) en la noticia completa para entrar y salir del modo de lectura cómoda.
- Salvado de los ajustes del usuario.
- Almacenamiento de datos (valores y ficheros) en caché para su posterior lectura *offline.*

Preparación

Teniendo en cuenta las características de la aplicación descritas en la sección anterior, ahora es el momento de elegir que librerías de JavaScript son las más adecuadas. Esta etapa es una de las más pesadas porque debemos probar varias soluciones, ver demostraciones, conjugarlas entre sí y valorar el resultado de la sinergia. Metafóricamente hablando podríamos decir que queremos cocinar una tarta, pero antes tendremos que elegir cuidadosamente los ingredientes para que el resultado sea delicioso.

Utilizar JavaScript *puro y duro* multiplicaría exponencialmente el tiempo de desarrollo de nuestra aplicación. Por ello, como librería base, vamos a utilizar **JQuery,** el *framework* de Javascript más utilizado en el desarrollo web. Entre otras bondades, esta librería cuenta con una enorme cantidad de métodos que simplifican el trabajo del programador y resuelve incompatibilidades entre navegadores.

Para dar alas a la interactividad necesitamos manejar *Asynchronous JavaScript And XML*, o simplemente AJAX. Es una técnica para el desarrollo de aplicaciones web interactivas que te permite cargar en segundo plano datos adicionales que el usuario puede necesitar con posterioridad. De este modo se pueden realizar modificaciones en las páginas sin tener que recargarlas, se reducen considerablemente las peticiones al servidor y, en definitiva, se logra un mejor rendimiento y velocidad en la aplicación.

JQuery cuenta con una extensión muy completa, a la vez que sencilla, dirigida al diseño de aplicaciones para móviles, **JQuery Mobile**. Para ello, esta librería se cimienta sobre la tecnología AJAX permitiendo cargar todas las pantallas de la aplicación en un único documento HTML y mostrándolas a petición del usuario sin necesidad de refrescar la página. Algo parecido pueden hacer otras potentes librerías como Handlebars.js o Backbone.js, pero estás tienen una sintaxis muy diferente a JQuery y usarlas implicarían un tiempo de aprendizaje mayor.

Las fuentes RSS y Atom van codificadas en un formato web basado en el metalenguaje XML y permiten difundir y compartir información actualizada de *websites* y *blogs*. Los periódicos y otros medios digitales

sirven sus noticias y contenidos de este modo. La información contenida en este formato es legible por las máquinas pero no por los usuarios: carece de estilos y necesita ser maquetada previamente.

```
<rss xmlns:content="http://purl.org/rss/1.0/modules/content/" xmlns:wfw="http://wellformedweb.org/CommentAPI/"
xmlns:dc="http://purl.org/dc/elements/1.1/" xmlns:atom="http://www.w3.org/2005/Atom"
xmlns:sy="http://purl.org/rss/1.0/modules/syndication/" xmlns:slash="http://purl.org/rss/1.0/modules/slash/"
xmlns:media="http://search.yahoo.com/mrss/" version="2.0">
        <channel>
                <title>Motor en Levante-emv.com</title>
                <atom:link href="http://motor.levante-emv.com/rss-app/" rel="self" type="application/rss+xml"/>
                <link>http://motor.levante-emv.com</link>
                <description>Sección de motor de levante-emv.com</description>
                <lastBuildDate>Wed, 31 Jul 2013 09:48:43 +0000</lastBuildDate>
                <language>es-ES</language>
                <sy:updatePeriod>hourly</sy:updatePeriod>
                <sy:updateFrequency>1</sy:updateFrequency>
                <generator>http://wordpress.org/?v=3.5.2</generator>
                <item>
                        <title>La línea DS3 de Citroën cambia de marcha</title>
                        <link>http://motor.levante-emv.com/la-linea-ds3-de-citroen-cambia-de-marcha</link>
                        <pubDate>Mon, 29 Jul 2013 07:31:54 +0000</pubDate>
                        <dc:creator>Javier Pinés Gil</dc:creator>
                        <category><![CDATA[ Citroën ]]></category>
                        <category><![CDATA[ Coches ]]></category>
                        <category><![CDATA[ Citroën DS3 ]]></category>
                        <category><![CDATA[ Citroën DS3 Cabrio ]]></category>
                        <category><![CDATA[ DS3 ]]></category>
                        <guid isPermaLink="false">http://motor.levante-emv.com/?p=7523</guid>
                        <description><![CDATA[<p><img width="990" height="577" src="http://motor.levante-
emv.com/wp-content/uploads/2013/07/citroends3cambio.jpg" class="attachment-full wp-post-image"
alt="CitroenDS3Cambio" /></p>Para disfrutar aún más de la experiencia de conducción ● ● ● ]]></description>
                        <media:content url="http://motor.levante-emv.com/wp-
content/uploads/2013/07/citroends3cambio-80x80.jpg" width="80" height="80" medium="image" type="image/jpg"/>
                </item>
                <item> ● ● ● </item>
                <item> ● ● ● </item>
                <item> ● ● ● </item>
        </channel>
</rss>
```

Fuente RSS del especial de Motor de Levante-emv sin estilo, tan solo se muestra la información estructurada con etiquetas especiales en formato XML. Esta información es la que nuestra aplicación debe ser capaz de descargar, interpretar y maquetar para poder mostrarla de un modo legible al usuario.

Para ilustrar este trabajo, y con el consentimiento escrito de Levante EMV, vamos a utilizar su canal **Motor**, un especial dedicado al mundo de los coches y las motos. El documento de autorización se encuentra en los anexos. La aplicación final puede nutrirse de este canal o de cualquier otra fuente RSS o Atom definida por el usuario. Estas son las fuentes RSS del canal Motor de Levante EMV que vendrán precargadas en la *app* para mera demostración.

http://motor.levante-emv.com/rss-app	**Fuente RSS principal que incluye todas las secciones**
http://motor.levante-emv.com/rss-coches/	**Sección para coches**
http://motor.levante-emv.com/rss-motos/	**Sección para motos**
http://motor.levante-emv.com/rss-locales/	**Sección para noticias locales**
http://motor.levante-emv.com/rss-deportes/	**Sección para deportes**
http://motor.levante-emv.com/rss-opinion/	**Sección para opinión**
http://motor.levante-emv.com/rss-reportajes/	**Sección para reportajes**

Debemos determinar que librería utilizar para obtener los datos de servidores de noticias. El gigante Google nos proporciona una sencilla pero potente API, **Google Feed**, con la que descargar cualquier fuente RSS o Atom utilizando sólo JavaScript sin tener que manejar complejas proxies ni lenguajes de servidor.

El resultado que obtendremos con esta API es un objeto JSON con el que fácilmente podremos trabajar. JSON, *JavaScript Object Notation*, es una alternativa al formato XML que destaca por ser más eficiente y sencilla para el manejo de datos. Está especialmente indicado para aplicaciones interactivas del lado del cliente desarrolladas mediante JavaScript y AJAX, nuestro caso.


```
{
        "m":"json",
        "status":{
                "code":200
        },
        "feed":{
                "feedUrl":"http://motor.levante-emv.com/rss-app",
                "title":"Motor en Levante-emv.com",
                "link":"http://motor.levante-emv.com",
                "author":"",
                "description":"Sección de motor de levante-emv.com",
                "type":"rss20",
                "entries":[
                        {
                                "mediaGroups":[
                                        {
                                                "contents":[
                                                        {
                                                                "url":"http://motor.levante-emv.com/wp-
content/uploads/2013/07/citroends3cambio-80x80.jpg",
                                                                "type":"image/jpg",
                                                                "medium":"image",
                                                                "height":80,
                                                                "width":80
                                                        }
                                                ]
                                        }
                                ],
                                "title":"La línea DS3 de Citroën cambia de marcha",
                                "link":"http://motor.levante-emv.com/la-linea-ds3-de-citroen-cambia-de-marcha",
                                "author":"Javier Pinés Gil",
                                "publishedDate":"Mon, 29 Jul 2013 00:31:54 -0700",
                                "contentSnippet":"Para disfrutar aún más de la experiencia de conducción, el
Citroën DS3 incorpora, en el acabado Style, la nueva caja de cambios ...",
                                "content":"<p><img width=\"990\" height=\"577\" src=\"http://motor.levante-
emv.com/wp-content/uploads/2013/07/citroends3cambio.jpg\" alt=\"CitroenDS3Cambio\"></p>Para disfrutar aún más de la
experiencia de conducción  ● ● ● ",
                                "categories":[
                                        "Citroën",
                                        "Coches",
                                        "Citroën DS3",
                                        "Citroën DS3 Cabrio",
                                        "DS3"
                                ]
                        }
                        {...}
                        {...}
                        {...}
                ]
        }
}
```


Objeto JSON que codifica la información anteriormente mostrada . Como hemos dicho, este formato permite el análisis , manipulación y maquetación de los datos de manera más eficiente y sencilla que con el formato XML.

Existen numerosos *plugins* para JQuery -como FeedEk, entre otros- que usan esta API y simplifican al programador la tarea de implementar un lector RSS. Después de haber probado algunos ejemplos con estos *plugins*, hemos preferido acceder directamente a la API de Google con todos sus métodos originales para un control más preciso sobre nuestra aplicación.

HTML5 trae dos importantes mejoras en términos de almacenamiento local en el dispositivo del usuario: L*ocal Storage* y A*ppCache*. Con la primera podremos almacenar datos, muy útil si queremos dar al usuario una experiencia personalizada y guardar ajustes o información de cualquier tipo. Con la segunda podemos almacenar ficheros para que no se tengan que descargar cada vez que el usuario acceda a la aplicación web.

Por último, para conseguir comportamientos gestuales avanzados (*swipe down, swipe up, swipe right y swipe left*) vamos a utilizar **iScroll 4.** Es una librería cada vez más conocida, de poco peso y fácil te utilizar que básicamente te permite crear *scrolls* con soporte táctil dentro de elementos de la pantalla (*divs*). Esto resulta imprescindible porque los navegadores móviles no permiten esta funcionalidad: admiten un *scroll* general para toda la página, pero no para elementos internos.

Diseño

En esta fase, vamos a centrarnos en el **aspecto gráfico** de la aplicación. En primer lugar, y teniendo en cuenta los *wireframes* que hemos definido anteriormente, vamos a utilizar JQuery Mobile para diseñar las tres pantallas del proyecto.

Las tres pantallas de la *app* diseñadas con JQuery Mobile. De momento no hemos introducido ninguna línea de JavaScript, tan solo algo de HTML y CSS. No obstante, las pantallas ya son funcionales y podemos navegar entre ellas a través de los botones de sus cabeceras gracias a este completo *framework*.

```
<!-- feed page -->
<div data-role="page" id="home">
        <div data-role="header">
                <a href="#settings" data-rel="dialog" data-icon="gear" class="ui-btn-right" data-
iconpos="notext">Opciones</a>
                <h1>RSS App</h1>
        </div>
        <div data-role="content">
                <ul data-role="listview">
                        <li data-role="list-divider" style=" font-size: 12px; ">Visualizando x entradas en el canal
x en fecha x</li>
                        <li> ● ● ● </li>
                        <li> ● ● ● </li>
                         ● ● ●
                </ul>
        </div>
        <div data-role="footer" data-position="fixed">
                <h4>www.juancollado.es</h4>
        </div>
</div>

<!-- post page -->
<div data-role="page" id="post">
        <div data-role="header">
                <a href="#home" data-icon="arrow-l" data-iconpos="notext">Volver</a>
                <h1> </h1>
                <a href="#" data-icon="forward" data-iconpos="notext">Enlace</a>
        </div>
        <div data-role="content">
        <header>
        <h2 class='title'>Titular de la noticia</h2>
        <div class='group'><span>Nombre del canal</span> | <span class='author'>Nombre y apellido del autor</span>
| <span class='date'>Domingo, 21 de julio de 2013</span></div>
        </header>
        <section class='content'>
        <p><img src="design.png"></p>Lorem ipsum  ● ● ●
        </section>
        </div>
        <div data-role="footer" data-position="fixed">
                <h4>www.juancollado.es</h4>
        </div>
</div>

<!-- settings page -->
<div data-role="page" id="settings">
    <div data-role="header">
        <h1>Opciones</h1>
    </div>
    <div data-role="content">
        <label for="rss">Fuente RSS:</label>
        <input type="search" name="rss" id="rss" value="http://www.ejemplo.com/rss"  />
        <ul style="list-style-type: none;">
            <li><a>http://www.ejemplo.com/rss</a></li>
            <li><a>http://www.ejemplo2.com/rss</a></li>
            <li><a>http://www.ejemplo3.com/rss</a></li>
        </ul>
        <label for="slider">Numero de entradas por p&aacute;gina</label>
        <br>
        <input type="range" name="entriesPerPage" id="entriesPerPage" value="25" min="5" max="50" step="5" data-
highlight="true"/>
        <a id="reset" data-role="button" data-mini="true">Borrar cach&eacute;</a>
        <a href="#home" data-transition="pop" id="ok" data-role="button" data-rel="back" data-theme="a" data-
mini="true">Aceptar</a>
    </div>
</div>
```

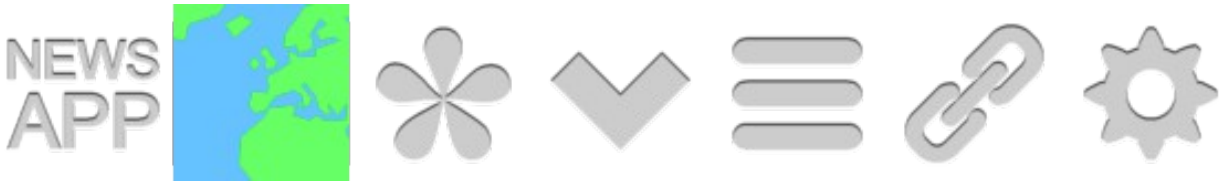
Vista del código de las tres pantallas implementadas con JQuery Mobile dentro de la etiqueta *body* del documento principal "index.html".

En segundo lugar, vamos a diseñar un logotipo para nuestra aplicación. Hemos decidido cambiar el nombre de RSS App por News App por ser éste último más descriptivo y no discriminar a las fuentes Atom, también adecuadas para esta *app*. Aparte del *favicon* (icono de favoritos asociado a la página por el navegador web), vamos a diseñar varias versiones del logo con diferentes dimensiones para los dispositivos de Apple. Estos serán los iconos que se utilicen cuando se cree un acceso directo a la aplicación web en la *home screen* del dispositivo móvil.

De izquierda a derecha: logo para iPhone/iPod Touch (57 x 57 pixeles), para iPhone/iPod Touch en alta resolución (114 x 114 pixeles), para iPad (72 x 72 pixeles) y para iPad en alta resolución (144 x 144 pixeles).

Los elementos de la interfaz y los iconos de JQuery Mobile están bien pero queremos que nuestra aplicación esté más personalizada: por último, vamos a crear nuestro propio *sprite* de iconos, vamos a modificar las reglas CSS que emplea esta librería y vamos a añadir nuevos elementos a la interfaz con nuevos estilos.

El uso de un *sprite* es una práctica muy recomendada para cargar en un solo fichero y de una sola vez todos los recursos gráficos que necesita la aplicación (en lugar de utilizar un fichero por cada recurso gráfico). Este es el *sprite* que va a emplear nuestra *app*:



De izquierda a derecha: logotipo de la aplicación, miniatura por defecto de las noticias, cargador para el listado, flecha para el listado, botón para volver al listado de noticias, botón de enlace externo a la noticia original y botón para abrir la ventana de opciones.

Ahora vamos con la hoja de estilos “index.css” que el lector puede encontrar en la subcarpeta “css”. Para una mejor comprensión, hemos organizado la hoja de estilos en diferentes secciones o módulos:

```
/*general selectors for all pages*/
@font-face {
  font-family: 'Open Sans';
  font-style: normal;
  font-weight: 400;
  src: local('Open Sans'), local('OpenSans'), url(../fonts/OpenSans.woff) format('woff');
}
body {...}
h1, h2, h3, h4, h5, h6, p {...}

ul.navbar {...}

.left {...}
.right {...}
.center {...}

.button, .logo {...}
.logo {...}
.button:hover {...}

.list {...}
.link {...}
.adjustments {...}

.publi {...}
.publi a {...}
```

Selectores generales: aquí definimos la fuente tipográfica que utilizaremos para los principales textos en pantalla (Open Sans), los selectores para "body" y los encabezados (h1, h2 ...), y las clases para el diseño de las barras de la cabecera (logos, botones...) y del pie de las pantallas (publicidad...).

```
/*feed page*/
.marquee{...}
#status{...}
#status .entries, #status .title {...}
#status .date {...}

#listMask {...}
#listScroll {...}
ul#listFeed {...}
ul#listFeed li{...}
ul#listFeed li:hover {...}
ul#listFeed li a {...}
ul#listFeed li a img{...}
ul#listFeed li a h3 {...}
ul#listFeed li a p {...}
ul#listFeed li a span.friendlyDate {...}
.noimage, .preload{...}

...
```

Selectores de ID y clase para los elementos internos de la pantalla del listado de noticias: máscara y área de *scroll* del listado; titular, entradilla, imagen y fecha de las noticias; indicadores de arrastrar y soltar; y marquesina informativa.

```
/*post page*/
#carouselMask {...}
#carouselScroll {...}
.articleMask {...}
.articleScroll{...}

h2.title {...}
.group {...}
.author {...}
.date {...}
.content {...}
```

```
.content p {...}
.content p img {...}
```

Selectores de ID y de clase para los elementos internos de la pantalla de detalle de la noticia seleccionada: máscaras y áreas de *scroll* para el carrusel de noticias y para el artículo completo; así como titular, canal, autor, fecha y contenido de la noticia.

```
/*settings page*/
.window {...}
.window h1 {...}
.window label {...}
```

Selectores de ID y de clase para los elementos internos de la pantalla de configuración: aspecto (fondo y borde) de la ventana y de las etiquetas que encabezan cada una de las opciones.

```
/*JQuery Mobile selectors*/
.ui-footer, .ui-header {height: 60px; background: rgba(0, 0, 0, 0.8)!important; border: none;}
.ui-loader-default {opacity: 0.6;}
.ui-btn-inner{padding-right: .6em!important;}
.ui-icon {display:none;}
```

Selectores propios de JQuery Mobile modificados: tamaño, fondo y borde de las barras de cabecera y de pie; opacidad de la imagen del cargador; y supresión del icono del *select* con las fuentes RSS de ejemplo.

```
/*media queries*/
@media (max-width:350px){
    ul.navbar {margin: 10px 5px 10px 5px;}
    .left {margin-right: 0px;}
    .right {margin-left: 0px;}
}
@media (max-height:319px){
    #listMask {bottom: 20px;}
    .marquee {bottom: 0px;}
    #carouselMask {bottom: 0px;}
    .ui-footer {display: none;}
}
@media (min-width:800px){
    .content p{
        max-width: 800px;
        overflow: hidden;
    }
}
```

Reglas CSS condicionadas por la resolución de la pantalla para un diseño web adaptativo (*Media queries*). En dispositivos con una anchura de pantalla inferior a 320 píxeles, se reducen los márgenes de la barra de de la cabecera. En dispositivos con una altura de pantalla inferior a 320 píxeles, se omite la barra publicitaria del pie. En dispositivos con una anchura de pantalla superior a 800 píxeles (tabletas), se limita la anchura que ocupan las imágenes de las noticias detalladas a 800 píxeles.

Programación

Ahora que ya tenemos todos los elementos con sus estilos, es el momento de hacer que funcionen entre sí y permitir al usuario interactuar con ellos: vamos a tratar el **aspecto funcional** de nuestra *app*. La fase

de programación es la que más tiempo requiere porque se debe conocer bien no solo el lenguaje Javascript, sino también las librerías, *frameworks* y APIs que se van a utilizar.

Como ya hemos comentado anteriormente, vamos a utilizar la API Google Feed y las librerías JQuery, JQuery Mobile e iScroll 4. Podemos encontrar el código JavaScript en el fichero "index.js" dentro de la subcarpeta "js". Este fichero también se ha organizado en diferentes módulos para facilitar su entendimiento:

```
//GLOBAL VARS
var online = navigator.onLine;//acceso a la red (verdadero o falso)
var entriesMax = 50;//número máximo de entradas que se pueden cargar desde la fuente
var entriesTotal;//número total de entradas que se han cargado con éxito desde la fuente
var entriesPerPage = 25;//número de entradas que se visualizan por pantalla
var feedUrl = "http://motor.levante-emv.com/rss-app"; //fuente RSS o Atom
var feedResult;//objeto json que almacena la información obtenida de la fuente
var feedDate;//fecha en que se ha descargado la información de la fuente
var postId = 0; //entrada actual seleccionada, por defecto es la primera
var listScroll = null;//objeto iScroll para el listado de noticias
var carouselScroll = null;//objeto iScroll horizontal para el carrusel de noticias detallads
var articleScroll = null;//objeto iScroll vertical para la noticia detallada en pantalla
var noScreenModes = false;//por defecto, el usuario puede entrar en modo de lectura cómoda en la noticia detallada
var windowWidth;//anchura de la pantalla del dispositivo
var windowHeight;//altura de la pantalla del dispositivo
```

Variables globales disponibles para cualquier función.

```
//INIT FUNCTIONS
$(document).ready(function(e) {
    //feed page
    $("#feed").bind("pageshow", function(){...});
    $('ul#listFeed').delegate("a","click",function(){...});
    $('#status').bind('marquee', function(){...});
    //settings page
    $("#settings").bind("pageshow", function(){...});
    $("#reset").click(function(e) {...});
    $('#samples').change(function() {...});
    $("#settings").bind("pagehide", function(){...});
    //post page
    $("#carouselMask").on("taphold", function(event){...});

    windowWidth = $(window).width();
    windowHeight = $(window).height();

    if(typeof(Storage)!=="undefined"){
        if (window.localStorage.getItem("feedUrl") == null){
            window.localStorage.setItem("entriesPerPage", entriesPerPage);
            window.localStorage.setItem("feedUrl", feedUrl);
        }else{
            entriesPerPage = parseInt(window.localStorage.getItem("entriesPerPage"));
            feedUrl = window.localStorage.getItem("feedUrl");
            feedResult = JSON.parse(window.localStorage.getItem("feedResult"));
            entriesTotal = parseInt(window.localStorage.getItem("entriesTotal"));
            feedDate = new Date(parseInt(window.localStorage.getItem("feedDate")));
        }
    }else{
        $("#reset").addClass("ui-disabled");
    }

    getData();

});
```

Funciones de inicialización: este módulo asocia diversos eventos a las pantallas y a los elementos que las integran, una

vez que todos ellos han sido cargados correctamente. Además se configuran las variables con las dimensiones de la pantalla (windowWidth y windowHeight) y las relacionadas con la fuente (entriestTotal, entriesPerPage, feedUrl, feedResult y feedDate). Éstas últimas se actualizan con los datos almacenados localmente, si los hubiera, gracias a la *interface* de HTML5 *Local Storage*. Finalmente se hace una llamada a la función principal del siguiente módulo, el de datos.

```
//DATA FUNCTIONS
function getData(){
    if (online) {
        try{getDataOnline();}catch(err){getDataOffline();}
    }else{
        getDataOffline();//we try to get cache data
    }
}
function getDataOnline(){...}
function getDataOffline(){...}
```

Funciones de obtención de datos: este módulo considera el valor *boolean* (verdadero o false) de la variable "online". Si existe conexión a Internet, siempre se descargan del servidor las últimas noticias de la fuente RSS o Atom especificada. En caso de estar *offline*, se obtienen las noticias de la copia almacenada en local, si la hubiera. Finalmente, se llama al siguiente módulo que imprime la información por pantalla.

```
//FEED FUNCTIONS
function loadlistFeed(){...}
function loadcarouselFeed(){...}
```

Funciones para imprimir la información por pantalla: este módulo esta formado únicamente por dos funciones, una encargada de dibujar el listado de noticias para la primera pantalla y otra encargada de dibujar el carrusel de noticias detalladas para la segunda pantalla. Cada función tiene una función compañera en el siguiente módulo , el que crea los *scrolls* necesarios para cada caso.

```
//SCROLL FUNCTIONS
function loadListScroll() {...}
function loadCarouselScroll() {...}
function loadArticleScroll() {...}
```

Funciones para crear *scrolls* con soporte táctil: este módulo está compuesto por tres funciones que asocian a las tres variables globales del principio tres objetos iScroll diferentes con soporte táctil: uno para la primera pantalla de la *app* y los otros dos para la segunda. El primero permite un deslizamiento vertical en el listado de noticias; el segundo, un deslizamiento horizontal en el carrusel de noticias detalladas ; y el último, un deslizamiento vertical por el cuerpo de la noticia detallada.

```
//DATE FUNCTIONS
function getDateHour(){...}
function getMonthName(monthNumber){...}
function dateToSpanish(dateEn){...}
function rssDateToObject(date){...}
function friendlyDate(date){...}
```

Funciones para fecha: este módulo incluye cinco funciones que formatean fechas. La primera crea un objeto de fecha en el momento que es llamada y devuelve una cadena con la hora y fecha completas. La segunda convierte el número de mes en el nombre del mes. La tercera recibe una cadena de fecha en inglés y devuelve una cadena de fecha en español. La cuarta recibe una cadena de fecha en ingles y devuelve un objeto de fecha. La quinta recibe un objeto de fecha y devuelve una fecha amigable (hace * años, hace *meses, hace * días, hace * horas, hace * minutos, hace un momento).

```
//TEXT FUNCTIONS
function textCleaner(text){...}
function textFormatter(text){...}
```

Funciones para texto: este módulo cuenta con dos funciones para formatear el texto de la noticia detallada. La primera elimina o modifica etiquetas multimedia (enlaces a galerías de imágenes y vídeos) que pueda haber en el cuerpo de la noticia. La segunda formatea texto plano adecuadamente en párrafos.

```
//ASPECT FUNCTION
$(window).resize(function() {...});
```

Función para aspecto: función que detecta un cambio en las dimensiones u orientación en la pantalla y recalcula el tamaño del carrusel de noticias y todos sus elementos internos en la segunda pantalla de la *app.*

En este apartado, existe otro fichero importante para nuestra aplicación web, "manifest.appcache". Este documento especifica los recursos que el navegador debe almacenar en caché aprovechando la *interface* de HTML5 *AppCache* que hemos comentado anteriormente.

```
CACHE MANIFEST
# v1 2013-09-01
CACHE:
index.html
favicon.ico
touch-icon-ipad.png
touch-icon-ipad-retina.png
touch-icon-iphone.png
touch-icon-iphone-retina.png
css/index.css
css/jquery.mobile-1.3.0.min.css
fonts/OpenSans.woff
img/ajax-loader.gif
img/sprite.png
img/sprite@2x.png
https://www.google.com/jsapi
js/jquery-1.9.1.min.js
js/jquery.mobile-1.3.1.min.js
js/iscroll.js
js/index.js
# Defines resources to be cached.
NETWORK:
*
# Defines resources that will not be cached
FALLBACK:
/ index.html
# Defines resources to be used if non-cached
```

El archivo de *manifest* de caché para esta *app* que define los recursos que deben almacenarse localmente, los que deben descargarse del servidor y las páginas que han de ser llamadas en caso de que un recurso no sea accesible.

Pruebas y compatibilidad

Ahora llegamos a una etapa tediosa: debemos probar el producto en los máximos soportes posibles y realizar algunos ajustes en el código para maximizar la compatibilidad y el buen rendimiento. En concreto, tendremos que probar la aplicación web no solo en varias dimensiones y resoluciones de pantalla, sino también en diferentes navegadores de escritorio y de dispositivos móviles.

En la siguiente tabla aparecen reflejados los navegadores más comunes en los que se ha ejecutado nuestra aplicación web y una valoración de la experiencia. Afortunadamente, como se puede apreciar, la aplicación ha tenido un buen desempeño y apenas se han tenido que realizar modificaciones.

MICROSOFT INTERNET EXPLORER(v.10)	✔	• Buen funcionamiento
MOZILLA FIREFOX (v.22)	✔	• Buen funcionamiento
GOOGLE CHROME (v.27)	✔	• **Funcionamiento óptimo**
APPLE SAFARI for IOS	✔	• **Funcionamiento óptimo**
ANDROID DEFAULT BROWSER	✔	• Buen funcionamiento
GOOGLE CHROME for ANDROID	✔	• **Funcionamiento óptimo**

Navegador de escritorio Navegador móvil

Para disfrutar de la mejor experiencia usando esta aplicación en un dispositivo móvil u ordenador, recomendamos utilizar navegadores modernos y actualizados, especialmente Google Chrome o Apple Safari.

Compilación para Android

En esta última etapa, tenemos que adaptar nuestra aplicación web, que funciona únicamente para navegador, a una aplicación híbrida que pueda funcionar de manera autónoma en dispositivos móviles. Para ello, debemos realizar algunas modificaciones.

Lo primero y más importante, es incluir la llamada a la librería Adobe Phonegap dentro del "head" del documento "index.html". En esa misma sección, se deben eliminar las llamadas al *favicon* y los iconos especiales para dispositivos de Appe; y sustituir estos archivos por un único icono en formato PNG. En la etiqueta "html" del documento se debe eliminar la llamada al fichero *manifest* de *AppCache,* y también este mismo fichero, puesto que ya no va a hacer falta.

```
<!DOCTYPE html>
<html manifest="manifest.appcache">
    <head>
        <title>Juan Collado - News App</title>
        <meta charset="utf-8">
        <meta name="viewport" content="width=device-width, initial-scale=1">
        <link href="css/jquery.mobile-1.3.0.min.css" rel="stylesheet" type="text/css" />
        <link href="css/index.css" rel="stylesheet" type="text/css" />

        <link rel="shortcut icon" href="favicon.ico" type="image/x-icon">
        <link rel="apple-touch-icon" href="touch-icon-iphone.png" /><!--57x57-->
        <link rel="apple-touch-icon" sizes="72x72" href="touch-icon-ipad.png" />
        <link rel="apple-touch-icon" sizes="114x114" href="touch-icon-iphone-retina.png" />
        <link rel="apple-touch-icon" sizes="144x144" href="touch-icon-ipad-retina.png" />

        <script type="text/javascript" src="https://www.google.com/jsapi"></script>
        <script type="text/javascript" src="js/jquery-1.9.1.min.js"></script>
        <script type="text/javascript" src="js/jquery.mobile-1.3.1.min.js"></script>
```

```
        <script type="text/javascript" src="js/iscroll.js"></script>
        <script type="text/javascript" src="js/index.js"></script>
    </head>
    ...
```

Primeras líneas del documento principal “index.html” para la aplicación web.

```
<!DOCTYPE html>
<html>
    <head>
        <title>Juan Collado - News App</title>
        <meta charset="utf-8">
        <meta name="viewport" content="width=device-width, initial-scale=1">
        <link href="css/jquery.mobile-1.3.0.min.css" rel="stylesheet" type="text/css" />
        <link href="css/index.css" rel="stylesheet" type="text/css" />

        <script type="text/javascript" src="https://www.google.com/jsapi"></script>
        <script type="text/javascript" src="js/jquery-1.9.1.min.js"></script>
        <script type="text/javascript" src="js/jquery.mobile-1.3.1.min.js"></script>
        <script type="text/javascript" src="js/iscroll.js"></script>
        <script type="text/javascript" src="phonegap.js"></script>
        <script type="text/javascript" src="js/index.js"></script>
    </head>
    ...
```

Primera líneas del documento principal “index.html” para la aplicación híbrida.

Lo siguiente que debemos hacer es crear e Incorporar el fichero “config.xml” con las características y ajustes de la aplicación. En él, se especifican el nombre, la descripción, la ruta del icono y la versión de la *app*. Además en este documento de configuración también se indican la versión de la librería de Phonegap que deseamos y las APIs (acelerómetro, datos del dispositivo, red, agenda, geolocalización...) que necesitará nuestro proyecto.

```
<?xml version="1.0" encoding="UTF-8"?>
<widget xmlns     = "http://www.w3.org/ns/widgets"
        xmlns:gap = "http://phonegap.com/ns/1.0"
        id        = "com.collado.newsapp"
        version   = "1.1.0">
    <name>News App</name>
    <description>News Reader Application - Juan Collado's PFC</description>
    <author href="http://www.juancollado.es/" email="info@juancollado.es">Juan Collado</author>

    <feature name="http://api.phonegap.com/1.0/device" />
    <feature name="http://api.phonegap.com/1.0/network"/>

    <preference name="phonegap-version" value="2.9.0" />
    <preference name="orientation"      value="default" />
    <preference name="target-device"    value="universal" />
    <preference name="fullscreen"       value="false" />

    <icon src="icon.png" />
</widget>
```

Fichero “config.xml” para la aplicación híbrida.

Finalmente debemos hacer cambios en el código JavaScript y utilizar algunos importantes métodos de este *framework* de Adobe. Hemos dividido el módulo de inicialización -primer módulo- de la aplicación en dos funciones. La primera se ejecuta cuando el documento, con todos sus elementos, se ha cargado correctamente; y la segunda cuando el dispositivo y sus APIs están preparadas para ser utilizadas. Dicho

de otro modo, la primera controla la disponibilidad del contenido y la segunda la disponibilidad del contenedor.

```
//INIT FUNCTIONS
function onDocumentReady() {
    ...
    document.addEventListener("deviceready", onDeviceReady, false);
}
function onDeviceReady() {
    windowWidth = $(window).width();
    windowHeight = $(window).height();

    if(typeof(Storage)!=="undefined"){
        if (window.localStorage.getItem("feedUrl") == null){
            window.localStorage.setItem("entriesPerPage", entriesPerPage);
            window.localStorage.setItem("feedUrl", feedUrl);
        }else{
            entriesPerPage = parseInt(window.localStorage.getItem("entriesPerPage"));
            feedUrl = window.localStorage.getItem("feedUrl");
            feedResult = JSON.parse(window.localStorage.getItem("feedResult"));
            entriesTotal = parseInt(window.localStorage.getItem("entriesTotal"));
            feedDate = new Date(parseInt(window.localStorage.getItem("feedDate")));
        }
    }else{
        $("#reset").addClass("ui-disabled");
    }

    if (navigator.connection.type == Connection.NONE){online  = false;}else{online = true;}
    document.addEventListener("online", function(){online = true;}, false);
    document.addEventListener("offline", function (){online = false;}, false);

    getData();
}
```

Funciones de inicialización (aplicación híbrida): la última sentencia de la función “onDocumentReady” crea un evento que detecta cuando el dispositivo con sus API's está listo para ser usado. Cuando esto ocurre, la función “onDeviceReady” realiza varias operaciones: cálculo de las dimensiones de la pantalla; creación o recuperación de valores mediante Local Storage; detección del acceso a la red...

Una vez realizados estos cambios, ya podemos compilar la aplicación para múltiples plataformas; proceso que llevaremos a cabo con Adobe Phonegap Build. Ésta es una potente herramienta de Adobe que te permite compilar en la nube proyectos desarrollados con Phonegap. Gracias a Phonegap Build, el proceso de compilación para múltiples plataformas no lo tendremos que realizar nosotros y por ello, ahorraremos un valioso tiempo y recursos.

Aunque Phonegap Build te permite compilar para los sistemas operativos móviles más conocidos, únicamente vamos hacerlo para Android por ser uno de los más importantes y porque, a diferencia de iOS o Blackberry, no requiere ninguna licencia de desarrollador.

Resultado final

Como se puede ver en las imágenes de abajo, el proyecto terminado destaca por su limpieza y sencillez. Se ha pretendido que la interfaz fuera lo más intuitiva posible para el usuario, y esto se ha logrado dando soporte a los gestos táctiles que se suelen usar en estos productos.

Accediendo a las siguientes direcciones de Internet, podrá ver la aplicación web en funcionamiento desde el navegador de su *smartphone* (disponible para todos los navegadores modernos) o descargar la aplicación híbrida para un dispositivo Android.

http://www.juancollado.es/pfc/	**Enlace en Internet a la aplicación web**
http://www.juancollado.es/pfc/hybrid.apk	**Enlace en Internet a la aplicación híbrida (para Android)**

Estas son las pantallas finales de la aplicación con una breve explicación de su funcionamiento.

Pantalla principal con el listado de noticias

El usuario puede deslizar el contenido del listado hacia abajo para actualizar las noticias (*swipe down*).

El usuario puede deslizar el contenido del listado hacia arriba para cargar noticias anteriores (*swipe up*).

Pantalla para el visionado en detalle de la noticia seleccionada.

El usuario puede arrastrar una noticia horizontalmente (tipo carrusel) para cargar la siguiente o la anterior (*swipe right* y *swipe left*).

El usuario puede pulsar durante dos segundos (*long press*) sobre la noticia para entrar en el modo de lectura cómoda. Pulsando nuevamente dos segundos se recupera el modo de lectura normal.

Mediante la ventana de opciones, el usuario puede definir la fuente RSS o Atom que desea, así como el número de entradas que por defecto la aplicación debe cargar. A mayor número de entradas, mayores tiempos de carga y procesamiento (menor rendimiento). Además, tiene la opción de borrar los datos almacenados por la aplicación.

Conclusiones

El lector ha sido testigo de las etapas necesarias para desarrollar un producto para móviles desde la elección del tipo de aplicación hasta la compilación para Android, pasando por el análisis de la competencia, la preproducción mediante wireframes, la presentación de las tecnologías necesarias, la elección de las librerías de JavaScript más adecuadas, el diseño y programación del propio proyecto, las pruebas de compatibilidad y las modificaciones pertinentes para convertir la aplicación web en híbrida.

HTML5 está abriendo un nuevo mundo de posibilidades en el diseño de aplicaciones del lado de cliente. Ahora cualquiera con conocimientos en lenguajes web puede adentrarse en la aventura de desarrollar para móviles. Esta tecnología, avalada por los gigantes Google y Apple y estandarizada por el Consorcio World Wide Web (W3C) tiene un futuro claramente prometedor.

Solo me queda animar al lector a inspeccionar el código fuente completo tanto de la aplicación web como de la híbrida que encontrará en dos carpetas anexas a este documento. Con las explicaciones que se han dado y con conocimientos básicos de los estándares web, no le será difícil entender como funciona este proyecto.

Bibliografía

Libros

FIRTMAN, MAXIMILIANO (2012). *jQuery Mobile, Aplicaciones HTML5 para móviles.* Madrid: Anaya Multimedia

MELONI, JULIE C.(2012). *Html5, Css3 y Javascript*. Madrid: Anaya Multimedia

SAWYER MCFARLAND, DAVID (2012). *JavaScript y jQuery*. Madrid: Anaya Multimedia

IBM CORPORATION (2012). *Native, web or hybrid mobile-app development*. USA: WebSphere

Citación de páginas web

Tipos de aplicaciones

RUIZ VEGA, CAROLINA. *Aplicaciones híbridas todoterreno*. 9 de diciembre de 2012. <http://www.elfinancierocr.com/tecnologia/aplicaciones-hibridas-todo-terreno_0_204579575.html>(julio de 2013).

Mercado, competencia

GOOGLE INC. *Aplicaciones*. <http://play.google.com/>(julio de 2013).

Wireframes

FUNDACIÓN WIKIMEDIA, INC. *Wireframe (Diseño web)*. 11 de marzo de 2013. <http://es.wikipedia.org/wiki/Wireframe_(Dise%C3%B1o_web)>(julio de 2013).

HERNÁNDEZ, MAURICIO. 10 herramientas gratis para realizar wireframes online. 18 de julio de 2010. <http://www.uxabilidad.com/usabilidad/10-herramientas-gratis-para-realizar-wireframes-online.html>(julio de 2013).

Información de tecnologías y librerías

FUNDACIÓN WIKIMEDIA, INC. *RSS*. 9 de julio de 2013. <http://es.wikipedia.org/wiki/RSS>(julio de 2013).

FUNDACIÓN WIKIMEDIA, INC. *RSS*. 9 de marzo de 2013. <http://es.wikipedia.org/wiki/Atom_(formato_de_redifusi%C3%B3n)>(julio de 2013).

FUNDACIÓN WIKIMEDIA, INC. *FeedSync*. 15 de marzo de 2013. <http://en.wikipedia.org/wiki/FeedSync>(julio de 2013).

FUNDACIÓN WIKIMEDIA, INC. *JSON*. 27 de julio de 2013. <http://es.wikipedia.org/wiki/JSON>(julio de 2013).

FUNDACIÓN WIKIMEDIA, INC. *AJAX*. 25 de junio de 2013.
<http://es.wikipedia.org/wiki/AJAX >(julio de 2013).

FUNDACIÓN WIKIMEDIA, INC. *Favicon*. 18 de agosto de 2013.
<http://es.wikipedia.org/wiki/Favicon>(agosto de 2013).

RAHIM. *Building a mobile web site *without* Sencha Touch 2: tools to consider*. 17 de diciembre de 2011.
<http://blog.encona.com/2011/12/building-a-mobile-web-site-without-sencha-touch-2-tools-to-consider/ >(julio de 2013).

JQUERY RAIN. *20+ Best jQuery Rss feed reader tutorial & plugin with examples*. 9 de julio de 2012.
<http://www.jqueryrain.com/2012/07/best-jquery-rss-feed-reader-tutorial-plugin-examples/>(julio de 2013).

Diseño

VILLALOBOS, RAY. *Beginner guide to a mobile app using the jQuery Mobile JavaScript Framework (with video)*. 26 de julio de 2012.
<http://ht.ly/cM7QS>(julio de 2013).

APPLE INC. *Specifying a Webpage Icon for Web Clip.* 1 de mayo de 2013.
<https://developer.apple.com/library/ios/documentation/AppleApplications/Reference/SafariWebContent/ConfiguringWebApplications/ConfiguringWebApplications.html>(agosto de 2013).

APPLE INC. *App Icons*. 19 de septiembre de 2012.
<https://developer.apple.com/library/ios/documentation/UserExperience/Conceptual/MobileHIG/IconsImages/IconsImages.html#//apple_ref/doc/uid/TP40006556-CH14>(agosto de 2013).

FUNDACIÓN WIKIMEDIA, INC. *Media query*. 6 de agosto de 2013.
<http://es.wikipedia.org/wiki/Media_query>(agosto de 2013).

Documentación de librerías

THE JQUERY FOUNDATION. *JQuery Api documentation*.
<http://api.jquery.com/>(agosto de 2013).

THE JQUERY FOUNDATION. *JQuery Mobile Api Documentation*.
<http://jquerymobile.com/>(agosto de 2013).

GOOGLE INC. *Google Feed API Documentation*. 31 de enero de 2012.
<https://developers.google.com/feed/v1/?hl=es-ES>(agosto de 2013).

SPINELLI'S, MATEO. *iScroll documentation*. 10 de marzo de 2011.
<http://cubiq.org/iscroll-4>(agosto de 2013).

ADOBE SYSTEMS INC. *Phonegap API Documentation*. 3.0.0.
<http://docs.phonegap.com/en/3.0.0/index.html>(agosto de 2013).

Tutoriales completos

UNYELIOGLU, KONUR. *Use jQuery Mobile to Build a Native Android News Reader App*. 2 de febrero de 2011.
<http://mobile.tutsplus.com/tutorials/mobile-web-apps/jquery_android/>(agosto de 2013).

WAY, JEFFREY. *How to Build an RSS Reader with jQuery Mobile*. 17 de Octubre de 2011.
<http://net.tutsplus.com/tutorials/javascript-ajax/how-to-build-an-rss-reader-with-jquery-mobile-2/>(agosto de 2013).

VOHRA, DEEPAK. *Retrieving RSS/Atom Feeds with the Google AJAX Feed API.* 7 de septiembre de 2007. <http://www.theregister.co.uk/2007/09/07/rss_atom_feeds/>(agosto de 2013).

Selectores y métodos de JQuery

JQUERY BY EXAMPLE. *jQuery - bind() vs live() vs delegate() methods*. 17 de agosto de 2010. <http://jquerybyexample.blogspot.com/2010/08/bind-vs-live-vs-delegate-function.html>(agosto de 2013).

BELMONTE RUIZ, FRANCISCO. *Curso jQuery. Capítulo 7: Selectores de filtro de hijos*. 7 de diciembre de 2010. <http://parasitovirtual.wordpress.com/2010/12/07/curso-jquery-capitulo-7-selectores-de-filtro-de-hijos/ >(agosto de 2013).

W3SCHOOLS.COM. *jQuery Selectors.* <http://www.w3schools.com/jquery/jquery_ref_selectors.asp?output=print>(agosto de 2013).

Miniaturas de las noticias

RSS ADVISORY BOARD. *RSS 2.0 Specification*. 30 de marzo de 2009. <http://www.rssboard.org/rss-specification#optionalChannelElements>(agosto de 2013).

STACK EXCHANGE INC. *GoogleFeed retrieving Images*. 13 de septiembre de 2012. <http://stackoverflow.com/questions/12404416/googlefeed-retrieving-images>(agosto de 2013).

VAGABUNDIA BLOG. Json: *Detectar la primera imagen de cada entrada*. 3 de mayo de 2011. <http://vagabundia.blogspot.com/2011/05/json-detectar-la-primera-imagen-de-cada.html>(agosto de 2013).

VAGABUNDIA BLOG. *Usar JSON para mostrar las entradas del blog*. 16 de diciembre de 2010. <http://vagabundia.blogspot.com/2010/12/usar-json-para-mostrar-las-entradas-del.html>(agosto de 2013).

Almacenamiento de datos

PILGRIM, MARK. Nº7. *The past, present & future of local storage for web applications*. 19 de junio de 2013. <http://diveintohtml5.info/storage.html>(agosto de 2013).

HEILMANN, CHRISTIAN. *Wrapping Things Nicely with HTML5 Local Storage*. 6 de diciembre de 2010. <http://24ways.org/2010/html5-local-storage/>(agosto de 2013).

W3SCHOOLS.COM. *HTML5 Web Storage.* <http://www.w3schools.com/html/html5_webstorage.asp>(agosto de 2013).

JSON ORG. *JSON in JavaScript*. <http://www.json.org/js.html>(agosto de 2013).

STACK EXCHANGE INC. *Intrepreting/Parsing JSON data with jQuery getJSON*. 1 de enero de 2010. <http://stackoverflow.com/questions/1987682/intrepreting-parsing-json-data-with-jquery-getjson>(agosto de 2013).

Almacenamiento de ficheros

MOZILLA DEVELOPER NETWORK. *Usando caché de aplicaciones*. 15 de julio de 2013. <https://developer.mozilla.org/es/docs/Recursos_offline_en_firefox>(agosto de 2013).

W3SCHOOLS.COM. *HTML5 Application Cache.* <http://www.w3schools.com/html/html5_app_cache.asp>(agosto de 2013).

TORRES, GISELA. *HTML 5 Application Cache: Aplicaciones Offline*. 7 de abril de 2013. <http://www.returngis.net/2013/04/html-5-application-cache-aplicaciones-offline/>(agosto de 2013).

BIDELMAN, ERIC. *Guía para principiantes sobre el uso de la caché de aplicaciones*. 18 de junio de 2010. <http://www.html5rocks.com/es/tutorials/appcache/beginner/>(agosto de 2013).

Marquesina

DASGUPTA, SMARAJIT. *Scrolling text content using jQuery (and not using marquee)*. 6 de enero de 2011. <http://blog.codez.in/scrolling-text-content-using-jquery-and-not-using-marquee/jquery/2011/01/06>(agosto de 2013)

Tamaño y resolución de pantalla de móviles

ONBILE. *Mobile screen sizes*. 9 de mayo de 2010. <http://www.onbile.com/info/mobile-screen-sizes/>(agosto de 2013).

RIEGER, BRYAN. *Effective Design for Multiple Screen Sizes*. 15 de enero de 2009. <http://mobiforge.com/designing/story/effective-design-multiple-screen-sizes>(agosto de 2013).

Phonegap: comunidades y ejemplos

CHARLAND, ANDRE. *PhoneGap Build Community.* <http://community.phonegap.com/nitobi>(agosto de 2013).

PHONEGAP SPAIN. *Comunidad de PhoneGap y PhoneGap Build en España.* <http://www.phonegapspain.com/>(agosto de 2013).

DESARROLLOWEB.COM. *Aplicaciones moviles de la mano de PhoneGap (manual)*. 8 de noviembre de 2012. <http://www.desarrolloweb.com/manuales/aplicaciones-moviles-phonegap.html>(agosto de 2013).

DUNNE, SHAUN. *PhoneGap From Scratch: Device APIs*. 16 de enero de 2012. <http://mobile.tutsplus.com/tutorials/phonegap/phonegap-from-scratch-device-apis/>(agosto de 2013).

CAMDEN, RAYMOND. *PhoneGap Online/Offline Tip*. 24 de mayo de 2013. <http://www.raymondcamden.com/index.cfm/2013/5/24/PhoneGap-OnlineOffline-Tip>(agosto de 2013).

STACK EXCHANGE INC. *PhoneGap: Opening external URL's in Safari*. 20 de abril de 2012. <http://stackoverflow.com/questions/10244965/phonegap-opening-external-urls-in-safari>(agosto de 2013).

Anexos

AUTORIZACIÓN PARA EL USO DE DATOS DE LA EMPRESA

Yo D. Miguel Miró Oriola, con D.N.I 45,633,308-C, en nombre y representación como administrador de la empresa EDITORIAL PRENSA VALENCIANA, S.A., Sociedad Unipersonal con CIF nº A-46229290, sociedad domiciliada en Valencia, calle Traginers, 7, autorizo a D. Juan Miguel Collado Navarro, con D.N.I. nº: 22594272-S al uso de los datos de los RSS de Motor (http://motor.levante-emv.com), gestionados por EDITORIAL PRENSA VALENCIANA. S.A., Sociedad Unipersonal, a través de su aplicación alojada en la web de EDITORIAL PRENSA VALENCIANA, S.A., Sociedad Unipersonal, para su proyecto Final de Carrera. Su uso se limitará a la consulta interna por parte de la Universidad Politécnica de Valencia.

En Valencia, a 14 de junio de 2013.

Firma

www.levante-emv.com
Levante
EL MERCANTIL VALENCIANO