


ESCUELA TÉCNICA SUPERIOR DE INGENIEROS INDUSTRIALES Y DE TELECOMUNICACIÓN

Titulación :

INGENIERO TÉCNICO DE TELECOMUNICACIÓN, ESPECIALIDAD EN SONIDO E IMAGEN

Título del proyecto:

# DISEÑO Y DESARROLLO DE UNA APLICACIÓN HÍBRIDA PARA *SMARTPHONE* UTILIZANDO EL FRAMEWORK JQUERY MOBILE Y PHONEGAP

Alumno: Victor Lucian Timoftii

Tutor: Marko Galarza Galarza

Pamplona, 27 de Junio del 2013

ÍNDICE

1. Introducción ..... 4
1.1 El Smartphone ..... 5
1.2 Aplicaciones Móviles ..... 7
1.3 ¿Aplicaciones web, nativas o híbridas? ..... 8
1.3.1 Aplicaciones web (HTML5) ..... 9
1.3.2 Aplicaciones nativas ..... 10
1.3.3 Aplicaciones híbridas ..... 11
1.3.4 Conclusiones ..... 11
1.4 Google play store vs Apple store ..... 12
1.4.1 Google play store ..... 12
1.4.2 Apple store ..... 13
1.4.3 Conclusiones ..... 14
2. Objetivo ..... 15
3. Tecnologías ..... 17
3.1 Framework "jQuery Mobile" ..... 17
3.1.1 ¿Qué es un Framework? ..... 17
3.1.2 Introducción a jQuery Mobile ..... 18
3.1.3 Anatomía de una página ..... 19
3.2 Open-Data Navarra ..... 21
3.3 CSS ..... 23
3.3.1 ¿Qué es CSS? ..... 23
3.3.2 ¿Qué se puede hacer con CSS ..... 23
3.3.3 ¿Cuál es la diferencia entre el CSS y HTML? ..... 24
3.3.4 ¿Qué ventajas tiene el CSS? ..... 25
3.4 HTML5 ..... 25
3.4.1 Introducción ..... 25
3.4.2 Histórico ..... 26
3.4.3 Implementación de HTML5 ..... 27
3.4.4 Estructura y semántica ..... 27
3.4.5 Elementos nuevos en HTML5 ..... 28
3.5 JavaScript ..... 28
3.5.1 ¿Qué es JavaScript? ..... 28
3.5.2 ¿Es Java y JavaScript lo mismo? ..... 29
3.5.3 ¿Qué puede hacer JavaScript? ..... 29
3.5.4 El nombre real de JavaSCript es ECMAScript ..... 30
3.6 PhoneGap ..... 31
3.6.1 ¿Qué es PhoneGap ..... 31

4. Desarrollo de la aplicación ..... 33
4.1 Esquema de la aplicación ..... 33
4.2 Origen de los datos y almacenamiento de los ficheros ..... 36
4.2.1 Los datos y su origen ..... 36
4.2.2 Almacenamiento de la aplicación en la web ..... 37
4.3 Los códigos para importar los ficheros XML y JSON ..... 39
4.3.1 XML ..... 39
4.3.2 JSON ..... 41
4.4 Aspecto visual ..... 43
4.5 Navegación ..... 45
4.6 Mapas y geolocalización ..... 46
4.6.1 Descripción de la página ..... 46
4.6.2 Explicación del código ..... 47
4.7 Implementación de PhoneGap y la utilización de QR-code ..... 51
4.7.1 El papel de PhoneGap en la aplicación ..... 51
4.7.2 QR-code ..... 52
4.8 Elementos adicionales ..... 53
4.8.1 El reloj flash ..... 53
4.8.2 Los logotipos de las secciones ..... 54
4.8.3 El widget de weather ..... 55
4.8.4 iframe de Facebook ..... 56
4.8.5 Animation loop New in town ..... 57
5. Conclusiones ..... 58
6 Posibles Mejoras ..... 59
7 Bibliografía ..... 60

# CAPÍTULO 1. INTRODUCCIÓN

Esta parte constituye una introducción a los nuevos conceptos y tecnologías que se han remarcado en los últimos años, produciendo un crecimiento importante en la venta de los dispositivos que utilizan los nuevos descubrimientos relacionados con IT, ofreciéndonos un modo superior de acceso a las informaciones pero también la posibilidad de interaccionar con las mismas. Otro aumento significativo se observó, a partir de estos logros, en el campo de entretenimiento, el cual hace posible desarrollar una amplia gama de juegos, reproductores de audio y vídeo, y herramientas para leer libros.

## 1.1 El Smartphone

Un Smartphone es un teléfono móvil con un sistema operativo (OS) avanzado, que permite instalar unas aplicaciones complejas y una amplia personalización de la interfaz gráfica de usuario. A diferencia de un teléfono móvil estándar, los teléfonos inteligentes tienen un espacio de almacenamiento interno mucho más grande, y la mayoría tiene la posibilidad de introducción de una tarjeta micro-SD, la cual suele tener entre 2 GB y 32 GB, para ampliar la memoria de almacenamiento de los dispositivos. Un teléfono inteligente está equipado, en la mayoría de los casos, con una pantalla táctil, superior a 3 pulgadas. Además, un smartphone, en general, no tiene un teclado físico. Si es proporcionado por el fabricante, se trata de un formato deslizante y es de tipo QWERTY (hace referencia a las primeras seis letras que aparecen en la esquina superior izquierda de un teclado físico).

Hoy en día un smartphone es considerado un teléfono que ejecuta uno de los siguientes sistemas operativos: Android OS (en aumento, con una gran cantidad de fabricantes que lo utilizan), BlackBerry OS (tan común para personas de negocios como para simples usuarios), Symbian, iOS (software de Apple para iPhone), MeeGo (unión de los sistemas operativos de Maemo de Nokia y Moblin de Intel) y Windows Phone (con poca experiencia pero muy prometedor en el futuro).

La figura 1.1.1 es un diagrama que muestra el análisis de la utilización de los diferentes OS, realizado a mediados del año 2012 por las empresas consultoras y de investigación IT: Gartner e IDC.

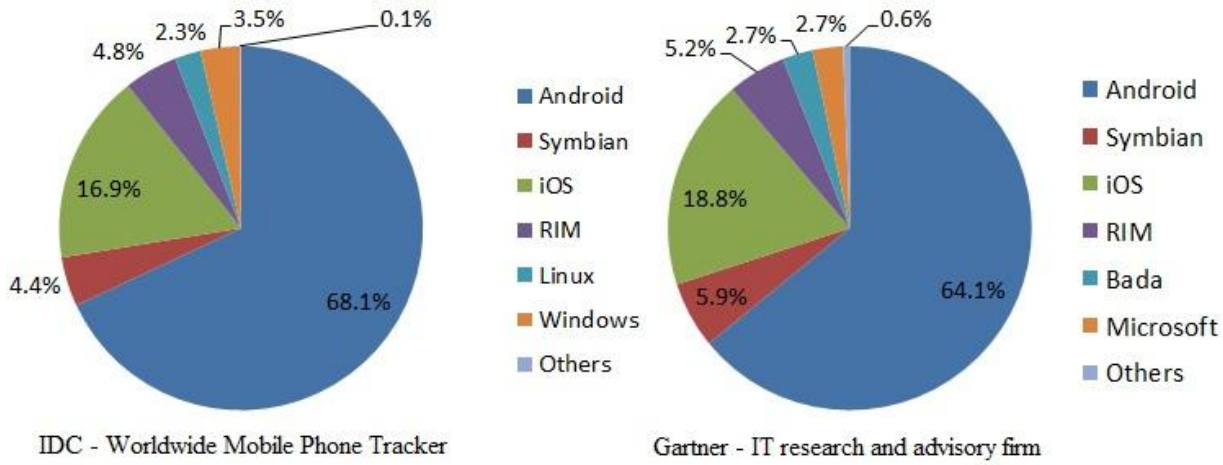



***Figura 1.1.1*** *- diagramas de análisis realizados por Gartner e IDC*

Un teléfono inteligente es solo tan bueno como la cantidad de aplicaciones disponibles para el OS que se está ejecutando.

En términos de hardware, el smartphone viene con un chipset de gran alcance en el cual se incluye el procesador y el chipset gráfico dedicado. Los procesadores integrados en la mayoría de los teléfonos inteligentes tienen uno, dos o cuatro núcleos y son capaces de funcionar en frecuencias que van desde 600 MHz a 1,5 GHz.

Además de las características habituales de un teléfono, como la agenda, la mensajería, la calculadora, los juegos, los teléfonos inteligentes vienen con algunas funciones extra: acceso a internet, capacidad para recibir y enviar e-mails, fax, lectura de documentos (*.doc o *.pdf), navegación GPS, cámara integrada de alta resolución, rápida transferencia de datos a un ordenador, etc. Además, al tener un OS, pueden instalarse fácil las aplicaciones que se necesitan, desde un programa básico de ver fotos hasta un sistema más complejo, como GPS. Así, el smartphone viene a satisfacer las necesidades de los usuarios más exigentes, pudiendo ser personalizado.

La imagen 1.1.2 representa una estadística acerca de cómo los usuarios usan los teléfonos inteligentes en general.

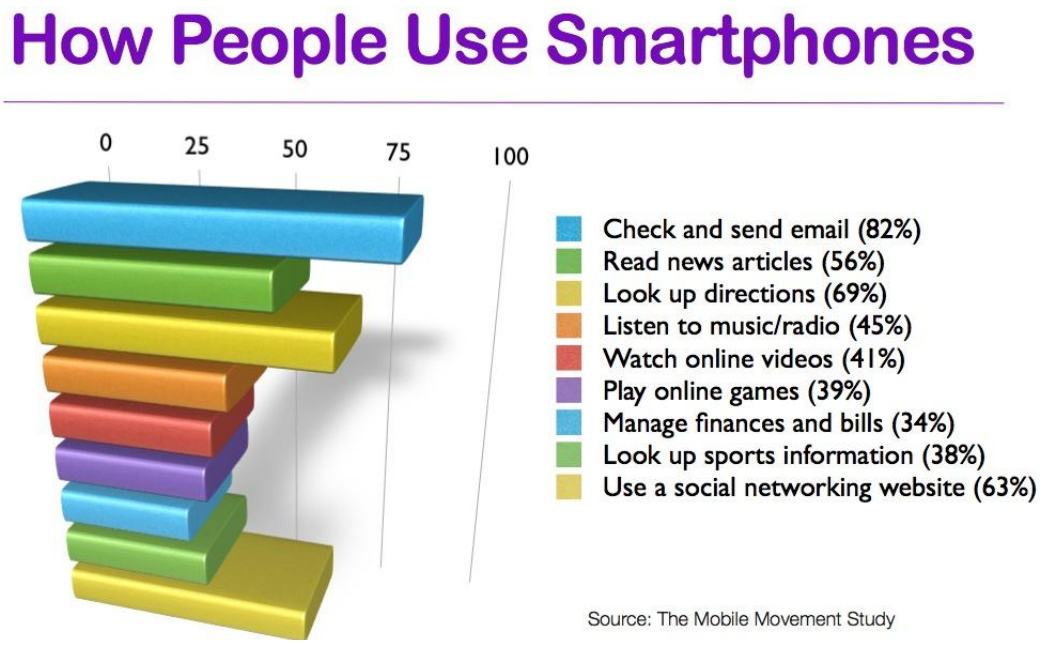



***Figura 1.1.2*** *- diagrama de porcentajes realizado por The Mobile Movement*

El futuro del Smartphone se ve muy prometedor. Al ser dispositivos que poseen la movilidad, la conectividad y la capacidad de programación, se espera que su uso siga creciendo en los próximos años.

## 1.2 Aplicaciones móviles

El conjunto de las aplicaciones móviles nativas es básicamente un software de ordenador, que, después de que se instalen en un sistema operativo determinado cumplen funciones útiles para los usuarios. Para Android OS, estos archivos tienen formato de *.apk, los de iOS de *.ipa y los de Symbian *.sis o *.sisx. Aunque los usuarios tienen acceso directo a la aplicación completa, su única tarea es descargarla e instalarla. Detrás de estos programas se lleva a cabo un trabajo que requiere programas avanzados y una amplia experiencia en el campo. El lenguaje de programación principal que se utiliza para la mayoría de las aplicaciones es Java, y uno de los programas profesionales que apoya al programador es Eclipse. Tanto Java como Eclipse requieren conocimientos avanzados a la hora de usarlos.

El mercado de aplicaciones móviles, con un valor actual de 20 mil millones de dólares, lo que aumentará cinco veces en los próximos tres años, no sólo revolucionando la industria del software, sino que también tiene un impacto económico importante, de acuerdo con un nuevo estudio realizado. Gracias a la última generación de teléfonos inteligentes y tabletas, se da lugar a la creación de un nuevo tipo de aplicaciones avanzadas para desarrolladores de software, proporcionando oportunidades para una nueva generación de pequeñas empresas. Hoy en día las aplicaciones para los sistemas Android y Apple representan el 75% del mercado de las aplicaciones para smartphones.

Del total de las aplicaciones móviles descargadas en 2012, alrededor del 89% son aplicaciones gratuitas, según Gartner. Gartner es una Empresa consultora y de investigación de las tecnologías de información con sede en Stamford, Connecticut, EEUU. Gartner también señala que, a nivel global, las aplicaciones móviles se han descargado 45,6 mil millones veces en 2012. El director de la empresa afirma que "En cuanto a las aplicaciones que compraron los consumidores, el 90%

de ellas costaban menos de 3$ cada una, y las aplicaciones que costaban entre 0.99$ y 2.99$ representaron el 87,5% de la descargas del 2012. En 2013 el porcentaje alcanzará el 96%."

En septiembre de 2012 Gartner presentó la siguiente tabla, representada en la figura 1.2.1, donde se muestran las descargas de aplicaciones móviles en todo el mundo durante 2010-2016 (En millones de descargas)

| | 2011 | 2012 | 2013 | 2014 | 2015 | 2016 |
|---|---|---|---|---|---|---|
| **Descargas gratuitas** | 22,044 | 40,599 | 73,280 | 119,842 | 188,946 | 287,933 |
| **Descargas pagadas** | 2,893 | 5,018 | 8,142 | 11,853 | 16,430 | 21,672 |
| **Total descargas** | **24,936** | **45,617** | **81,422** | **131,695** | **205,376** | **309,606** |
| Descargas gratuitas % | *88.4%* | *89.0%* | *90.0%* | *91.0%* | *92.0%* | *93.0%* |

***Figura 1.2.1*** *- tabla de descargas realizada por Gartner*

## 1.3 ¿Aplicaciones web, nativas o híbridas ?

Hoy en día, en función de cómo se da el desarrollo de la aplicación, podemos hablar de tres tipos de aplicaciones móviles, que se definen como:

- Aplicaciones web: completamente desarrolladas utilizando HTML5
- Aplicaciones nativas: aquellas que normalmente utilizan el lenguaje Java y están completamente programadas en el entorno de desarrollo específico para cada sistema operativo.
- Aplicaciones híbridas: aplicaciones desarrolladas en parte con el lenguaje web (HTML5) y en parte con el entorno de desarrollo nativo.

### 1.3.1 Aplicaciones web (HTML5)

Al día de hoy, los smartphones vienen con potentes navegadores que soportan múltiples funciones: HTML5 (lenguaje para estructurar y presentar el contenido de la www), CSS3 (utilizado para controlar el estilo y el diseño de páginas web), y JavaScript (lenguaje de programación avanzado, bien conocido por su uso en la construcción de páginas web). Los proveedores de aplicaciones pueden crear páginas web que reconocen el acceso desde móvil y actúan de manera que se pueda ver en una pantalla más pequeña. También se puede diseñar una página web que sea muy similar a una aplicación nativa, aunque no lo sea. Se puede proyectar la aplicación web que no se ve como una aplicación nativa, pero tienen la misma funcionalidad y en ocasiones puede solicitar la instalación en el móvil para lograr que tenga el mismo aspecto. Esta tendencia se ha acelerado en los últimos años.

Para este tipo de aplicaciones el problema que se presenta es que los lenguajes web no tienen todavía compatibilidad con todas las funciones nativas (API), tales como el GPS, acelerómetro, captura de imágenes, agenda de contactos, calendario, etc.

Una habilidad que se destaca sobre las aplicaciones web en comparación con las nativas, es que son multiplataforma, lo que significa que se puede ejecutar en cualquier sistema operativo, sin tener que volver a programar, por lo tanto, esto disminuye los problemas que traen las aplicaciones nativas creadas para un único OS que requieren atención constante, lo que implica un aumento de los costes. Una aplicación que contiene HTML5 funciona generalmente en todos los dispositivos móviles, siempre y cuando sean compatibles con el motor del Safari Webkit (plataforma para aplicaciones que funciona como base para un navegador).
En comparación con las aplicaciones nativas, las aplicaciones web requieren un tiempo invertido relativamente corto, y la programación en HTML5 es mucho más simple lo que hace que se necesiten menos requisitos.

Un punto negativo para las aplicaciones web es la promoción, porque su presencia en los mercados oficiales de los sistemas operativos predominantes como iOS y Android OS, no está admitida. Esto hace que el descubrimiento de las aplicaciones sea más complicado, lo que

supone un esfuerzo mayor en el sector de marketing. Otra desventaja es el rendimiento de las aplicaciones web que sigue siendo mucho menor, porque éstas son mucho más lentas en comparación con las aplicaciones nativas.

### 1.3.2 Aplicaciones nativas

Las aplicaciones nativas son consideradas aplicaciones convencionales móviles. Se instalan directamente en el smartphone. "Viven" en los dispositivos y pueden interactuar con ellos utilizando las características y funciones del teléfono en el que están instaladas, por ejemplo, puede utilizar las funciones de acelerómetro, calendario, libreta de direcciones y muchas otras prestaciones integradas al teléfono inteligente. Este tipo de aplicaciones no requieren una "aplicación contenedor", como las aplicaciones web que necesitan un navegador, esto significa que las nativas prestan la interfaz del sistema operativo del móvil.

Una aplicación nativa es aquella que ejecuta el sistema operativo directamente y suelen desarrollarse con lenguajes de programación que requieren una compilación anterior. La gran desventaja de una aplicación nativa es que si se quiere lograr una aplicación profesional se necesita un nivel de conocimiento de programación y una experiencia avanzados, lo que hace que la implementación sea más costosa. Otro inconveniente sería que las aplicaciones nativas necesitan un desarrollo para cada uno de los sistemas operativos e incluso para versiones diferentes de estos.

La ventaja del desarrollo de este tipo de aplicaciones es que existe un mercado donde se pueden encontrar todas las aplicaciones, lo que supone una mayor difusión de la aplicación, siendo más accesible para los usuarios. Sin embargo, para que esté disponible para todos los mercados, la aplicación debe ser programada para cada plataforma.

Las aplicaciones nativas reciben las notificaciones y las actualizaciones automáticas al instante. Éstas pueden funcionar aunque el teléfono no esté conectado a internet, así los usuarios pueden acceder a la información de la aplicación en cualquier momento.

### 1.3.3 Aplicaciones híbridas

Este tipo de aplicación se utiliza cuando se quiere desarrollar una aplicación móvil muy flexible que combina elementos web y nativos. Del punto de vista técnico, una aplicación híbrida es una aplicación nativa construida con HTML5. Un claro ejemplo podría ser, este mismo proyecto que, trata de convertir una aplicación web en una aplicación híbrida utilizando el framework de PhoneGap.

La aplicación híbrida contará con todos los beneficios de una aplicación nativa, teniendo acceso a todos los APIs, teniendo en cuenta que unas partes de la aplicación serán desarrolladas utilizando tecnologías web.

El problema surge debido a que empresas como Apple no aceptan aplicaciones que intentan infiltrar contenido que no puede ser revisado por ellos mismos.

### 1.3.4 Conclusiones

Toda empresa que quiera desarrollar aplicaciones móviles debe considerar las ventajas y desventajas de los tres tipos de aplicaciones antes de empezar a construir el código. La decisión depende. Si una empresa quiere lanzar y distribuir lo más rápidamente posible una sencilla aplicación para promocionarla, la mejor opción sería hacer una aplicación web. Si la empresa tiene más tiempo, más capital para invertir y desea realizar una aplicación más potente, debe elegir la aplicación nativa.

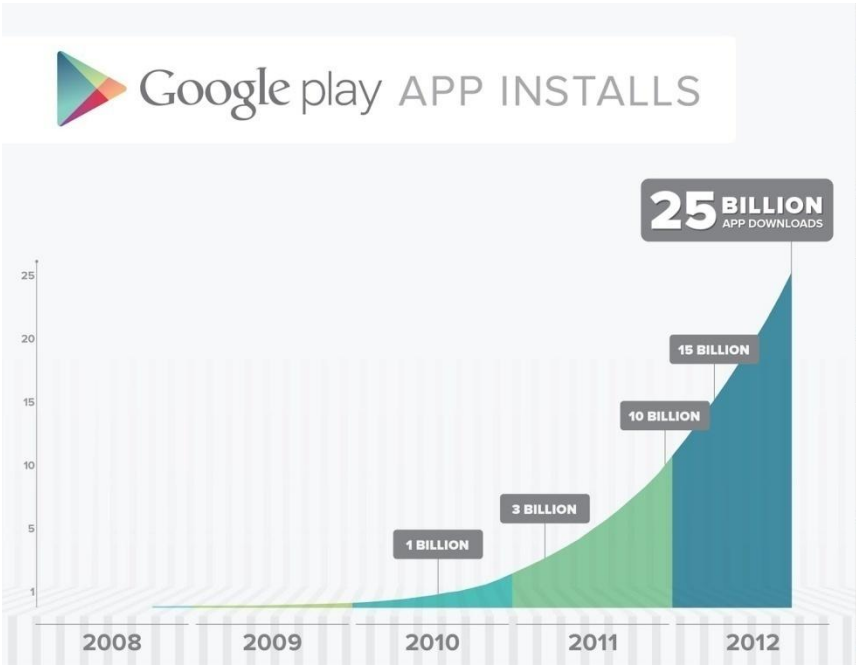
Aunque las aplicaciones híbridas parecen ser las más favorecidas, heredan las desventajas de las dos.

## 1.4 Google play store vs Apple store

### 1.4.1 Google play store

A finales de 2012, Google alcanzó la cifra de 700.000 aplicaciones disponibles en Android Market, ya a principios de marzo de 2013 superó las 800.000. La tienda online antes conocida como Android Market, siempre en competencia con App Store de Apple, parece ahora que la ha superado.

De todas las aplicaciones disponibles en Google Play, las gratuitas se pueden descargar en 190 países, mientras que los usuarios de Android pueden comprar aplicaciones en solo 132 países. El 50% de lo que gana Google Play se debe a la venta de dichas aplicaciones. En total estas aplicaciones se han descargado más de 25 billones de veces.




| Year | Month | Applications available | Downloads to date |
|---|---|---|---|
| 2009 | March | 2,300 | |
| | December | 16,000 | |
| 2010 | March | 30,000 | |
| | April | 38,000 | |
| | August | 80,000 | 1 billion |
| | October | 100,000 | |
| 2011 | May | 200,000 | 3 billion |
| | July | 250,000 | 6 billion |
| | October | 319,000 | |
| | December | 380,297 | 10 billion |
| 2012 | January | 400,000 | |
| | February | 450,000 | |
| | May | 500,000 | |
| | June | 600,000 | 20 billion |
| | September | 675,000 | 25 billion |
| | October | 700,000 | |
| 2013 | February | 800,000 | |

***Figura 1.4.1.1*** *- tabla y gráfico del mercado de Google play obtenido de Wikipedia*

## 1.4.2 Apple Store

En la otra cara de la competencia, es decir, en Apple Store, el número total de aplicaciones en enero de 2013 era de 800.000. Los usuarios de iOS han descargado 30 billones de veces las aplicaciones de Apple Store desde 2008 hasta ahora. Aunque ambas tiendas se pusieron en marcha unos meses de diferencia, una posible justificación de la discrepancia en el rendimiento podría ser debido al hecho de que la tienda virtual, Apple Store, en el momento de lanzamiento, ya tenía una base de usuarios significativa. La figura 1.4.2.1 muestra la evolución de los aplicaciones disponibles en el mercado de Apple.

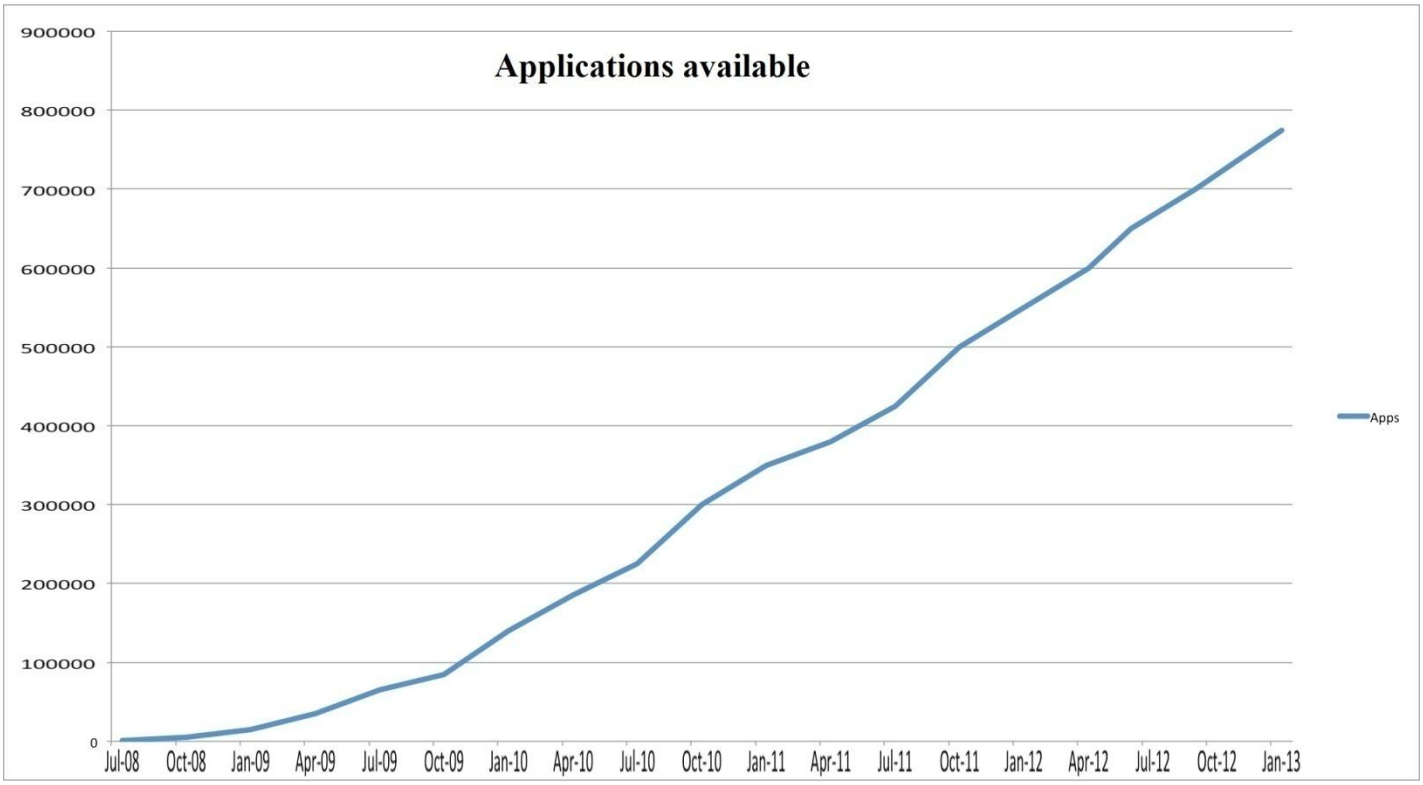



***Figura 1.4.2.1*** *- gráfico de las aplicaciones disponibles en el mercado de Apple. Datos obtenidos de Apple.*

### 1.4.3 Conclusiones

Aunque en este momento la carrera entre Google y Apple parece muy apretada, finalmente una de las dos plataformas prevalecerá al frente. Este no es el caso de Windows Phone, ya que este mismo está muy poco desarrollado respecto a los dos anteriores.

# CAPÍTULO 2. OBJETIVO

La aplicación **New in Town - Pamplona** ha sido diseñada y desarrollada con el fin de ayudar a los turistas y a los residentes de la ciudad para que puedan conocerla más fácilmente y acceder más rápido a los servicios de emergencia. También facilita el encuentro de la ubicación deseada disponiendo de informaciones desde la distancia y el mapa del trayecto utilizando la geolocalización hasta el número de teléfono y el horario del destino. Utilizando estas funciones, el usuario tiene acceso instantáneo a las informaciones básicas de los principales hospitales, restaurantes, tiendas y puntos culturales.

Estas aplicaciones son posibles gracias a la evolución del teléfono móvil, en concreto a la aparición del smartphone. En pocos años, estos mini-ordenadores han logrado conquistar el mercado de la telefonía móvil, gracias a sus cualidades no vistas hasta ahora. En enero de 2007 Apple presentó el primer terminal iPhone y comenzó a dominar el mercado de los teléfonos inteligentes antes de su lanzamiento en el mercado en junio del mismo año. Este momento marcó el declive de muchos gigantes, de los cuales Microsoft, Palm, RIM (compañía canadiense de dispositivos inalámbricos más conocida como la fabricante del dispositivo de comunicación de mano BlackBerry) y Nokia redujeron enormemente su cuota de mercado en tan sólo cinco años de existencia del iOS pero también del Android. Google ha conseguido poner en marcha su sistema operativo en el año 2008, y su crecimiento ha sido más significativo que la comunidad iOS, sobre todo gracias al hecho de que este sistema operativo se utiliza en terminales mucho más baratos que los dispositivos de Apple. En cinco años, Android OS y iOS superaron las cuotas del mercado de los productores de smartphones, RIM siendo el único que aún mantiene una cuota de mercado significativa.

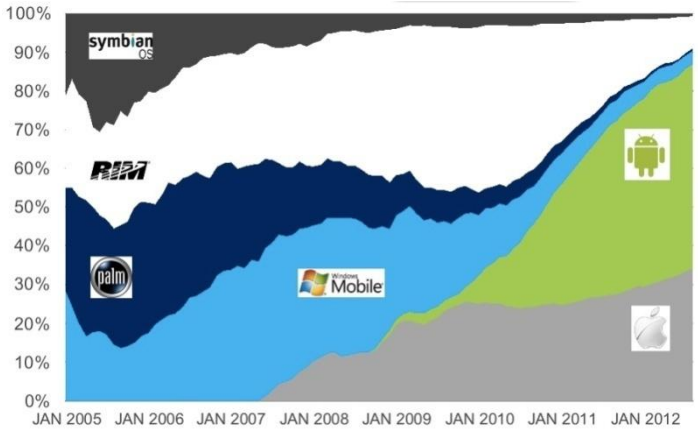



***Figura 2.1*** *- gráfica con la evolución de los sistemas operativos obtenida de [www.teknology.com](http://www.teknology.com)*

Utilizando el servicio de web framework de jQuery Mobile se ha logrado exponer de manera interactiva todos los tipos de datos de Open-Data proporcionados gratuitamente por el Gobierno de Navarra y se ha conseguido crear una aplicación web capaz de ejecutarse en la mayoría de las plataformas móviles y en una gran variedad de navegadores web, tales como: iOs, Android, BlackBerry, WebOS, Symbian, Windows Phone 7, Safari, Opera, Chrome, Firefox.
El modelo Open-Database es una iniciativa destinada a construir bases de datos muy grandes, como las de las grandes empresas, y luego ponerlos a disposición de los usuarios como herramientas de código público y abierto.

El objetivo del proyecto es crear una aplicación semi-nativa que se podrá descargar e instalar en una plataforma determinada. Esto fue posible convirtiendo la aplicación web de jQuery Mobile en un único fichero *.apk usando el servicio de PhoneGap. El usuario podrá descargar el archivo desde una aplicación de mercado online y también a través de un código QR, siempre que esté subida en una página que lo soporte.

Para el desarrollo de la aplicación se ha utilizado jQuery Mobile como web framework porque actualmente es el servicio más avanzado de este tipo. Aunque el tiempo de proceso es superior a otros servicios similares, logramos reducir la desventaja convirtiendo la aplicación web en una aplicación semi-nativa (híbrida) utilizando el framework PhoneGap.

La aplicación **New in Town - Pamplona** intenta unir en un solo medio fácil de utilizar, varios servicios que para un turista o un residente podrían resultar necesarios, tales como: gasolineras, paradas de autobuses y taxis, escuelas, bares y discotecas.

En las páginas siguientes trataré de explicar los términos anteriores para ofrecer una mejor compresión de cómo se concibió y se creó este proyecto.

# CAPÍTULO 3. TECNOLOGÍAS

## 3.1 Framework "jQueryMobile"

El jQuery Mobile es un sistema unificado, con interfaz de usuario, basado en la tecnología HTML5, disponible para todas las plataformas de dispositivos móviles más populares, construido con el sistema sólido jQuery y jQuery UI.

### 3.1.1 ¿Qué es un Framework?

***Figura 3.1.1.1*** *- Logo Framework*

El concepto de framework se utiliza, no sólo para las aplicaciones web, sino también en varias áreas de desarrollo de software. Puede haber frameworks para el desarrollo de aplicaciones médicas, desarrollo de juegos, y para cualquier otro campo que se nos pueda ocurrir. En general, cuando se utiliza el término de framework, nos referimos a una estructura software construida con elementos personalizados e intercambiables para el desarrollo de una aplicación. En otras palabras, un framework puede ser visto como una aplicación genérica incompleta y configurable a la que se le puede añadir las últimas piezas para construir una aplicación determinada. Los principales objetivos de un framework son: acelerar el proceso de desarrollo, la reutilización del código existente y la promoción de unas prácticas de desarrollo, como el uso de patrones. Un framework web, por lo tanto, se puede definir como un conjunto de componentes (por ejemplo clases en java y descriptores y archivos de configuración en XML) que componen un diseño reutilizable para el desarrollo rápido y sencillo de los sistemas web.

## 3.1.2 Introducción a jQueryMobile

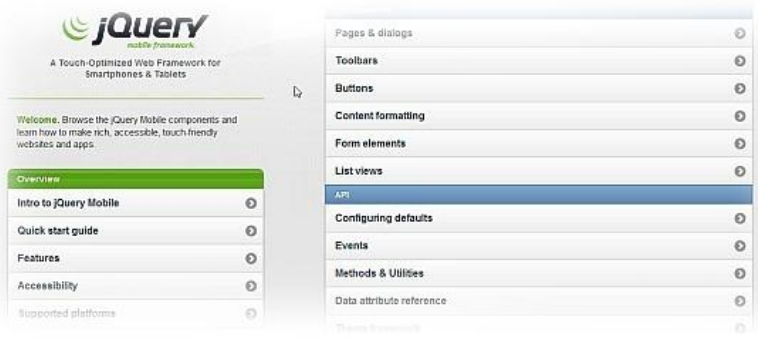



*Figura 3.1.2.1 - Logo jQuery Mobile*

Jquery Mobile es un framework multiplataforma diseñado para simplificar y mejorar el desarrollo de aplicaciones web para móviles mediante la integración de HTML5, CSS3, JavaScript y jQuery en un marco (framework) que no sólo es resistente, también fácil de mantener y muy organizado.
Sin embargo, todavía tiene algunos inconvenientes como el lento funcionamiento y unos errores de estructura.

jQuery Mobile se basa en el núcleo de jQuery existente, lo que significa que si un programador entiende la sintaxis de jQuery, entonces el programador no tendrá problemas para conseguir que el JM funcione. El framework es compatible con la mayoría de aplicaciones para PC y móviles, incluyendo iOS, Android, Blackberry, Palm WebOS, Nokia/Symbian, Windows Mobile, Opera Mobile, Firefox Mobile y todos los navegadores de PC modernos.

El API de jQuery Mobile apoya no sólo los eventos táctiles, que se utilizan para navegar en los smartphones hoy en día, sino también los eventos de ratón normales generalmente asociados con jQuery, por lo que es totalmente compatible con varios navegadores y plataformas. El framework ofrece una variedad de temas de estilo a elegir por el usuario, pero también dispone del servicio jQuery Themeroller con el que puede crear diseños de widgets personalizados.

Personalmente he elegido trabajar con jQuery Mobile, ya que es un framework bastante sencillo, bien estructurado y con muchos ejemplos útiles. Para entender y ver jQuery Mobile en acción es necesario leer la documentación que se encuentra en la página web oficial:

http://view.jquerymobile.com/1.3.1/dist/demos/

Los tutoriales que presenta la página son una documentación y a su vez demostración interactiva, porque mientras se ven los ejemplos se dispone también de la explicación de cómo se han realizado.

Mientras que la oferta de la página no es bastante rica como otros frameworks, nos encontramos con los siguientes ingredientes principales:

- **Páginas** - las aplicaciones web pueden estar compuestas por varias páginas y varios archivos (una página=un archivo) o, puede estar todo el código en un solo archivo (varias páginas en un archivo).
- **Navbar** - varios ejemplos de barras para navegar hacia arriba y/o hacia abajo
- **Botones** - una variedad de botones incluyendo: botones simples, checkboxes, radio buttons
- **Elementos de forma** - líneas, columnas, contenido que se oculta/muestra
- **Listas** - modalidades estéticas para mostrar más artículos
- **Encabezado y pie de página** - una banda estética horizontal al principio y/o al final de la página
- **Áreas y opciones de texto** - diferentes opciones para las regiones donde se introduce el texto

### 3.1.3 Anatomía de una página jQuery

```
1  <!DOCTYPE html>
2  <html>
3     <head>
4        <title>Page Title</title>
5        <meta name="viewport" content="width=device-width, initial-scale=1">
6        <link rel="stylesheet" href="http://code.jquery.com/mobile/1.0b2/jquery.mobile-1.0b2.min.css" />
7        <script type="text/javascript" src="http://code.jquery.com/jquery-1.6.2.min.js"></script>
8        <script type="text/javascript" src="http://code.jquery.com/mobile/1.0b2/jquery.mobile-1.0b2.min.js"></script>
9     </head>
0     <body>
1        ...content goes here...
2     </body>
3  </html>
```

***Figura 3.1.3.1*** *- Código HTML*

Como podemos ver en la figura 3.1.3.1, la página es un simple documento HTML5, solamente incluyendo en las líneas 6-7-8 los archivos de CSS y jQuery de CDN (CDN es un grupo de servidores que guardan los ficheros de jQuery). Si hubiéramos descargado los archivos de otros lugares, solo tendríamos que cambiar las rutas de los archivos que se encuentran en la etiqueta "**src**".

La etiqueta "**meta**" de la línea 5 es muy importante ya que ésta dice a nuestro dispositivo móvil que la anchura de la página debe ser la misma anchura que la pantalla del smartphone y que el navegador debe acercar el texto a 100% (escala normal).

Ahora que tenemos un documento simple, podríamos comenzar a configurar el diseño básico de nuestra aplicación. En jQuery Mobile, las páginas están separadas en el mismo documento HTML5 por las líneas de código "** **", que suponen el comienzo de una nueva página en nuestra aplicación. Por lo tanto en la figura 3.1.3.2 se podría ver en la parte izquierda el código de cómo agregar a la aplicación una página que tiene un encabezado, contenido y pie de página, y en la parte derecha cómo se vería en el dispositivo móvil.

```
1  <!DOCTYPE html>
2  <html>
3      <head>
4          <title>Page Title</title>
5          <meta name="viewport" content="width=device-width, initial-scale=1">
6          <link rel="stylesheet" href="http://code.jquery.com/mobile/1.0b2/jquery.mobile-1.0b2.min.css" />
7          <script type="text/javascript" src="http://code.jquery.com/jquery-1.6.2.min.js"></script>
8          <script type="text/javascript" src="http://code.jquery.com/mobile/1.0b2/jquery.mobile-1.0b2.min.js"></script>
9      </head>
10     <body>
11         <div data-role="page">
12             <div data-role="header">
13                 <h1>Header</h1>
14             </div>
15             <div data-role="content">
16                 <p>Content...</p>
17                 <p>More Content</p>
18             </div>
19             <div data-role="footer">
20                 <h4>Footer</h4>
21             </div>
22         </div>
23     </body>
24 </html>
```

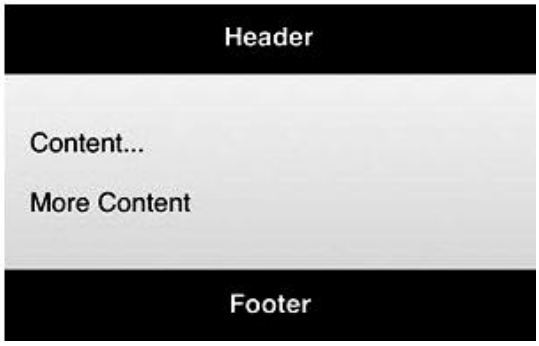



***Figura 3.1.3.2*** *- Código HTML y la aplicación resultante*

He mencionado antes que varias páginas podrían incorporarse en el mismo documento. La figura 3.1.3.3 nos da un ejemplo de cómo se podría conseguir esto.

```
<div data-role="page" id="page1">
  <div data-role="header">
    <h1>Page 1</h1>
  </div>
  <div data-role="content">
    <a href="#page2" data-role="button">Go to Page 2</a>
  </div>
</div>

<div data-role="page" id="page2">
  <div data-role="header">
    <h1>Page 2</h1>
  </div>
  <div data-role="content">
    <a href="#page1" data-role="button">Go to Page 1</a>
  </div>
</div>
```


***Figura 3.1.3.3*** *- Código HTML para varias páginas*

A partir de este simple documento, junto con la documentación que nos ofrece jQuery Mobile y con los tres ingredientes (HTML5, JavaScript y CSS) se podría empezar a desarrollar páginas y aplicaciones más complejas y más atractivas.

## 3.2 Open-Data Navarra

Open-Data son datos que se pueden utilizar libremente, reutilizar y redistribuir por cualquiera, supuestas solo a la obligación de transmisión en las mismas condiciones. Para que un dato sea abierto, es importante que, además de ser accesible, sea reutilizable, lo que permitirá que todo el mundo podría crear servicios derivados de los mismos. El Open-Data está muy vinculado a la exigencia de los ciudadanos de los países democráticos, por lo cual es imprescindible disponer de información correcta.

La definición completa de Open-Data ofrece detalles precisos sobre lo que ésta significa. Resumiendo lo más importante:

- **Disponibilidad y acceso:** los datos deben ser válidos en su conjunto y que tengan un precio de reproducción razonable, de preferencia mediante la descarga desde internet.

Los datos tienen que estar en un formato conveniente y modificable (ficheros *.JSON, *.XML)

- **Reutilización y redistribución:** los datos deben ser proporcionados en términos que permitan la reutilización y la redistribución incluyendo la combinación con otros datos.
- **Participación universal:** todo el mundo debería ser capaz de utilizar, reutilizar y redistribuir - no debe haber discriminación de áreas de investigación, de personas o de grupos.

Es muy importante entender lo que significa Open y por qué la definición debe ser aplicada, y la razón es: la interoperabilidad. La interoperabilidad denota la capacidad de los diversos sistemas y organizaciones para trabajar juntos (inter-operar). En este caso, es la capacidad de inter-operar, o combinar, diferentes conjuntos de datos. Proporcionar una definición clara del término Open, nos asegura que si tenemos dos tipos de datos de dos fuentes diferentes, el utilizador será capaz de combinar, evitando problemas de compatibilidad.

El punto clave es que cuando los datos son abiertos (Open-Data), se centra en datos no personales, es decir, datos que no contienen información sobre individuos específicos. Del mismo modo, ciertos tipos de datos del gobierno podrían estar restringidos, para evitar poner en riesgo la seguridad nacional.

Estados Unidos y Reino Unido son dos ejemplos de gobiernos que han dado pasos firmes en esta dirección. En España la administración pública también ha desarrollado proyectos de este tipo tanto a nivel central como autonómico. Los servicios Open-Data ya operando en España son: País Vasco, Asturias, Canarias, Castilla y León, Cataluña y Navarra. En Andalucía también se está trabajando para ofrecer este servicio.

Pensar en todas las formas de encontrar informaciones sobre el clima, si llueve o a cuantos grados estaremos en los próximos días. Esta información está disponible en la televisión, radio, internet y smartphone y no tenemos que pagar por ella. La razón de que esto sea posible es que el servicio meteorológico nacional está abierto. Lo mismo pasa con el sistemas de transporte. En las estaciones de autobús, de metro, de tren, sabemos exactamente cuándo va a llegar y así podemos

disfrutar del tiempo restante. Nos imaginamos ahora qué pasaría si el mismo sistema se aplicara al tráfico de coches. En las grandes ciudades, solo con disponer de un smartphone o tableta tendrán acceso directo al tráfico de datos que se actualizará al instante. Algunos de estos proyectos ya están operando, y en el futuro cada vez más servicios serán abiertos por el gobierno para ayudar al ciudadano.

Estos datos son generalmente provistos por el gobierno, lo que significa que es una fuente de información en la que se puede confiar. Por lo tanto, ofrece a los usuarios y a las pequeñas empresas la posibilidad de utilizar la información para fines personales pero también para redistribuirla en otras formas. Esto no solo puede ser beneficioso en términos económicos sino también en la transmisión de la información. Otra ventaja de este servicio es que toda la información es gratuita para cualquier usuario, dejando la posibilidad para que todos la puedan utilizar. Por lo tanto, esta iniciativa puede ser vista como un impulso del crecimiento y el empleo.

## 3.3 CSS

### 3.3.1 ¿Qué es CSS ?

CSS significa Cascading Style Sheets. Son unos estilos que se pueden definir tanto en la cabecera de una página web, como en un archivo separado, individual. La segunda opción es la recomendada.

### 3.3.2 ¿Qué se puede hacer con CSS ?

CSS es un lenguaje (style language) que define la forma de salida (layout) para documentos HTML. CSS incluye colores, tipos de letra, imágenes (borders), líneas, altura, anchura, imágenes de fondo y muchas otras opciones.

El HTML muchas veces está mal utilizado para crear el diseño de las páginas web. CSS ofrece más opciones, es más preciso y sofisticado. Además, es compatible con todos los navegadores actuales.

### 3.3.3 ¿Cuál es la diferencia entre el CSS y el HTML ?

HTML se utiliza para estructurar el contenido, mientras que CSS se utiliza para formarlo. En los primeros días de la web, solo HTML se utilizaba para estructurar el texto. El texto se marcaba con la etiqueta "**<hl>**" para el título y "**<p>**" para el párrafo. Cuando empezó a aumentar la popularidad de la web, los diseñadores comenzaron a buscar diferentes posibilidades de agregar el layout de los documentos HTML. Para cumplir estos requisitos, los fabricantes de navegadores han inventado nuevas etiquetas HTML, como "**<font>**" que define el diseño (layout) y no la estructura, con lo que se diferencia de las etiquetas HTML originales.

Esto ha llevado a una situación en la que las etiquetas originales de estructura como "**<table>**" estén utilizadas de forma inadecuada en las páginas de layout. Muchas de las nuevas etiquetas de layout como "**<blink>**" fueron reconocidas sólo por algunos navegadores. Un mensaje común solía aparecer en las páginas web "Se necesita el navegador X para ver esta página". CSS se inventó para remediar esta situación al ofrecer a los diseñadores técnicas más sofisticadas para la edición del layout, aceptadas por todos los navegadores.

Al mismo tiempo, separando los archivos CSS que definen el diseño de la página en sí, facilita enormemente su mantenimiento. Esta separación puede mejorar la accesibilidad de los contenidos, proporcionar más flexibilidad y reducir la carga de mantenimiento de una página web al ofrecer un control sencillo. El CSS también puede reducir la complejidad y la repetición de etiquetas que se utilizan para dar formato a la estructura de contenidos.

### 3.3.3 ¿Qué ventajas tiene el CSS?

El CSS es un elemento revolucionario en el mundo del diseño web. Los beneficios concretos son:

- El control del layout de documentos HTML con solo una página de diseño
- Controlar con más precisión el diseño
- Aplicación de diferentes layouts para diferentes tipos de medios (texto de pantalla, de URL, de impresión)
- Técnicas numerosas y sofisticadas

## 3.4 HTML5


### 3.4.1 Introducción

Los proveedores de navegadores y otros grupos de la industria de world wide web (www) compiten a introducir y aplicar nuevas funcionalidades a un ritmo más rápido que nunca. El navegador fue creado para facilitar la colaboración de investigadores, que podrían así distribuir y establecer vínculos entre los trabajos científicos. No había imágenes, sonidos o animaciones, porque el ancho de banda disponible en aquel momento era insuficiente.

Hoy en día podemos ver videos en alta definición o jugar juegos online, todo a través del navegador, pero la tecnología web sigue siendo la misma. Aún así, la "plataforma web" no fue diseñada desde el principio para que estas cosas sean posibles, pero ha ido evolucionando poco a poco empujada por las contribuciones de las partes interesadas.

### 3.4.2 Histórico

W3C (World Wide Web Consortium - se encarga de hacer la web accesible para todos los usuarios) fue creada en 1994 como un consorcio, para que los miembros de la industria WWW puedan unirse para establecer la normativa juntos, evitando las incoherencias en las implementaciones individuales. HTML4 fue lanzado en diciembre de 1997, pero en la década del 2000 las normas parecía que ya no evolucionaban. Insatisfechos, un grupo dentro de la compañía Apple, la Fundación Mozilla y Opera Software han creado WHATWG, para continuar el desarrollo del estándar HTML y un conjunto de APIs (Application Programming Interface) para las aplicaciones web.

El resultado es lo que hoy conocemos como el HTML5: un conjunto de tecnologías que constituyen la web moderna, permitiendo animaciones, multimedia y aplicaciones más sofisticadas. Básicamente, para los que no saben mucho de WWW, el término de HTML5 significa todo lo que se puede hacer con las tecnologías web modernas, aunque a veces para conseguir cosas avanzadas se utiliza CSS.

El estándar HTML5 en sí, fue publicado por W3C en 2008 y se espera una publicación más avanzada en 2014. WHATWG sigue trabajando para HTML, por lo tanto las especificaciones para el mismo se consideran un "estándar vivo", HTML5 siendo solo un resultado de su evolución.

Los creadores de HTML5 han decidido facilitar el trabajo de los diseñadores, mantener las cosas lo más simples posible:

- para que sea válido, un documento HTML5 necesita un doctype más simple que el de HTML4 o XHTML (**<!DOCTYPE HTML>**)
- algunas etiquetas no son obligatorias
- algunos de los nuevos elementos fueron adoptados siguiendo las prácticas habituales (**<header>**, **<footer>**, **<nav>**)

- algunas reglas son menos estrictas, por ejemplo, los autores podrían omitir las comillas de una etiqueta

HTML5 está diseñado para ser compatible con las versiones anteriores, por lo tanto, cualquier documento válido HTML 4 o XHTML es también valido HTML5, si utiliza el doctype nuevo. Esto es conveniente, porque significa que no hay que olvidar todo lo que sabemos acerca de HTML, y volver a empezar.

### 3.4.3 Implementación de HTML5

Debido a que aún no está finalizado, el HTML5 no está implementado totalmente en ningún navegador. Todos los navegadores modernos, sin embargo, proporcionan soporte para algunas funcionalidades y se pueden utilizar sin problemas, aunque las implementaciones a veces son distintas. Páginas como *www.caniuse.com* pueden ayudar a la decisión de utilizar o no una función particular, indicando si está implementada en los navegadores que se consideren importantes. La página web *www.modernizr.com* es una librería JavaScript que detecta que funciones HTML5 y CSS3 soportan el navegador del usuario, dando a los desarrolladores la capacidad para proporcionar soluciones de seguridad, utilizadas como reservas.

### 3.4.4 Estructura y semántica

Desde cuando apareció el CSS, el papel del HTML se ha limitado a la definición de la estructura de un documento, no a su apariencia visual. HTML5 pone aún más énfasis en la semántica, introduciendo nuevos elementos que permiten a los autores definir con más precisión algunas partes del documento. De esta manera, el HTML5 será mejor comprendido y tratado que los demás, como por ejemplo los motores de búsqueda o programas que ayudan a las personas con deficiencias de visión.

### 3.4.5 Elementos nuevos en HTML5

A continuación, se presenta una breve lista de ejemplos, con algunos elementos nuevos que trae HTML5:

- **<article>, <section>, <nav>, <header>, <footer>, <aside>, <hgroup>:**
  - ayudan a la estructura del documento.
  - **<article>**, por ejemplo, es más genérico que "el artículo de una revista", representa cualquier contenido que forma parte independiente del documento.

- **<details>, <summary>, <menu>, <command>:**
  - son elementos interactivos.
  - **<details>**, por ejemplo, genera un widget que puede mostrar u ocultar la información sobre un producto.

- **<figure>, <figcaption>:**
  - se utilizan para adoptar un gráfico, ilustración, fotografía, etc., a la que se hace referencia en el contenido.

## 3.5 JavaScript

### 3.5.1 ¿Qué es JavaScript?

JavaScript es un lenguaje de programación (scripting), utilizado generalmente para la programación del lado del cliente (client-side): el navegador descarga el código de JavaScript, que se ejecuta e interpreta en el ordenador, no en el servidor. Aunque la programación client-side es el uso más popular de JavaScript, más reciente se puede programar con JS server-side. El JavaScript es un lenguaje de programación simple, diseñado para añadir interactividad a las páginas HTML en las que, normalmente, está incluido.

### 3.5.2 ¿Es Java y JavaScript lo mismo?

¡No! Java y JavaScript son dos lenguajes completamente diferentes tanto en diseño como en concepto, es más que nada un parecido de los nombres. El Java (desarrollado por Sun Microsystems) es un lenguaje más potente y complejo, como C y C++. JavaScript fue desarrollado por primera vez por Netscape, con el nombre de Live Script, que era un lenguaje de programación que extiende las capacidades de HTML.

### 3.5.3 ¿Qué puede hacer JavaScript?

- **JavaScript ofrece a los programadores HTML un lenguaje de programación:**
  - los autores HTML no son programadores, pero JavaScript es un lenguaje de scripting con una sintaxis muy simple.

- **JavaScript puede poner texto dinámico en las páginas HTML:**
  - con el siguiente código, JS puede escribir una variable de texto en una página web:
  ("<**h1**>" + name + "</**h1**>")

- **JavaScript puede reaccionar a los eventos:**
  -JS puede ser configurado para ejecutarse cuando sucede cierta acción, por ejemplo al cargar una página o cuando un usuario hace clic en un elemento HTML.

- **JavaScript puede leer y escribir elementos en HTML:**
  -JS puede leer o modificar el contenido de un elemento HTML

- **JavaScript se puede utilizar para validar los datos:**
  -JS puede ser utilizado para validar un formulario antes de ser enviado al servidor, lo que puede ahorrar al servidor de trabajo extra.

- **JavaScript puede ser utilizado para detectar el navegador de usuario:**
  -JS puede ser usado para determinar el navegador del usuario y en función del navegador va a cargar otra página diseñada específicamente para él.

- **JavaScript se puede utilizar para crear cookies:**
  -JS puede ser utilizado para almacenar y recuperar información de la computadora de un usuario.

Antes de que todo lo de arriba sea posible, el código JS se tiene que insertar en una página web. Para hacer esto existen dos posibilidades:

- se puede escribir el código entre las etiquetas **<script>** y **</script>**, de la siguiente manera:
  (**<script type="text/javascript">//<![CDATA[**
  **// el código JS aquí**
  **//]]></script>**)

- o se puede escribir el código JS en un fichero externo e incluirlo en la página web, con la ayuda de la etiqueta script, de la siguiente manera:
  (**<script type"text/javascript" scr="ruta/al/fichero.js"></script>**)

### 3.5.4 El nombre real de JavaScript es ECMAScript

El nombre oficial de JS es "ECMAScript". El estándar es desarrollado y mantenido por la organización ECMA. ECMA-262 es el estándar oficial JS. La norma se basa en JavaScript (Netscape) y JScript (Microsoft). El lenguaje fue inventado por Brendan Eich en Netscape (con Navigator 2.0), y se ha incorporado en todos los navegadores Netscape y Microsoft desde 1996. La norma fue aprobada como un estándar internacional (ISO / IEC 16262) en 1998. La norma está todavía en desarrollo.

# 3.6 PhoneGap

## 3.6.1 ¿Qué es PhoneGap?

PhoneGap es un framework de código abierto (open-source) para la construcción rápida de aplicaciones móviles multiplataforma utilizando HTML5, JavaScript y CSS.

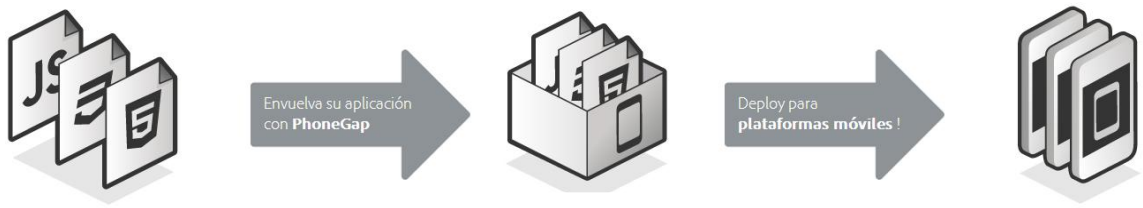



***Figura 3.6.1.1*** *- Logo PhoneGap*

Por ahora, probablemente ya se tiene una buena idea de lo que es PhoneGap, porque durante la memoria hemos visto bastante este término, pero si no es así, aquí se muestra un breve resumen:

PhoneGap nació de iPhoneDevCamp en 2008, donde fue creado simplemente porque no había muchos desarrolladores de Obiective-C (lenguaje de programación orientado a objetos, creado como un superconjunto de C) en comparación con los desarrolladores web.

El reto consistía en establecer un framework que permitiera a los desarrolladores web utilizar HTML, CSS y JavaScript para programar aplicaciones que podrían tomar ventaja de la funcionalidad nativa del dispositivo móvil, como la cámara, almacenamiento y funciones de geolocalización. Inicialmente fue creado para trabajar con el iPhone, pero PhoneGap siguió desarrollándose y en un año ha sido compatible también con Android. Ahora, con casi 4 años, PhoneGap es uno de los software más conocidos para el desarrollo de aplicaciones móviles y es compatible con una gama muy amplia de dispositivos móviles, incluyendo iOS, Android, Blackberry, Symbian, webOS (es un sistema operativo multitarea para sistemas embebidos basados en Linux), WP7 (Windows Phone) y Bada (sistema operativo para teléfonos móviles desarrollado por Samsung). En el momento de la publicación, el framework soportaba los siguientes APIs: acelerómetro, cámara, compás, contactos, archivos, relocalización, media,

network, notificaciones (alerta, sonido, vibraciones) y almacenamiento. Hay compatibilidad total para todas estas características en los dispositivos iOS más recientes y Android. Para más detalles sobre la compatibilidad de BlabkBerry, WP7, Symbian, webOS y los dispositivos Bada compruebe la figura 3.6.1.2 con la tabla oficial de compatibilidad.

| | iPhone / iPhone 3G | iPhone 3GS and newer | Android | Blackberry OS 5.x | Blackberry OS 6.0+ | WebOS | Windows Phone 7 + 8 | Symbian | Bada |
|---|---|---|---|---|---|---|---|---|---|
| Accelerometer | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| Camera | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| Compass | X | ✓ | ✓ | X | X | ✓ | ✓ | X | ✓ |
| Contacts | ✓ | ✓ | ✓ | ✓ | ✓ | X | ✓ | ✓ | ✓ |
| File | ✓ | ✓ | ✓ | ✓ | ✓ | X | ✓ | X | X |
| Geolocation | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| Media | ✓ | ✓ | ✓ | X | X | X | ✓ | X | X |
| Network | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| Notification (Alert) | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| Notification (Sound) | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| Notification (Vibration) | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| Storage | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | X | X |

***Figura 3.6.1.2*** *- tabla oficial de compatibilidad ofrecida por* http://phonegap.com/about/feature/

A pesar de lo que se cree, PhoneGap no es un programa de tipo escribir una vez, desplegar en cualquier sitio (write once, run anywhere), aunque se acerca mucho. Sin duda, es un framework multiplataforma capaz de funcionar con muchos dispositivos compatibles, pero con el fin de implementar con éxito, es probable que necesite probar y modificar su código para cada uno de los dispositivos de destino.

# CAPÍTULO 4. DESARROLLO DE LA APLICACIÓN

## 4.1 Esquema de la aplicación


***Figura 4.1.1*** *- Esquema de la aplicación*

El primer paso que se realizó para iniciar la aplicación ha sido el desarrollo en papel de una esquema de la aplicación web (figura 4.1.1). En concreto, se ha elegido el tipo y el diseño de la aplicación, la organización del contenido y la información que va a contener esto, incluyendo:

-Datos y fotos introducidos manualmente.

-Datos descargados de Open-Data Navarra.

-Datos e informaciones cogidas de las páginas web oficiales de los lugares contenidos en la aplicación.

Después de analizar el esquema anterior, se decidió construir una página de inicio de la aplicación, donde se colocará por categorías, diferentes secciones de centros, lugares, tiendas, para facilitar el trabajo del usuario para encontrar lo que desea. También se ha considerado importante el idioma que se va a implementar y, al ser Pamplona una ciudad turística con visitantes de todos los países, se ha establecido que el inglés será la mejor opción. A continuación se puede observar en la figura 4.1.2 una captura de pantalla de la página de inicio ya terminada, donde encontramos:

- Un header (encabezado de página) con el título de la aplicación.

- Una animación multicolor (la parte azul) que muestra el título de la aplicación y se encuentra en un loop infinito.

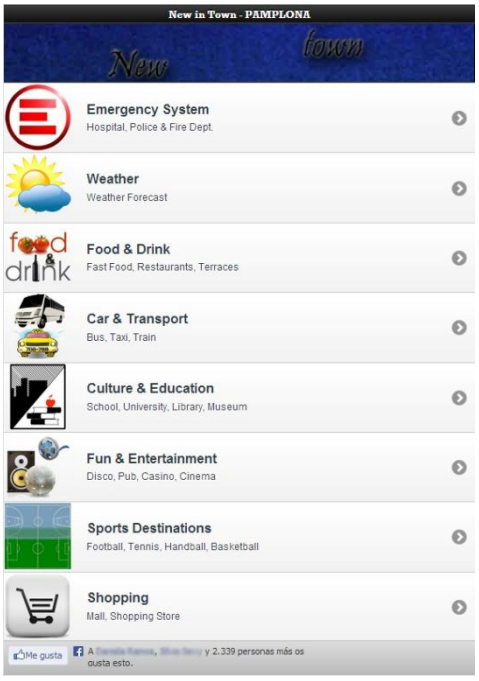



***Figura 4.1.2*** *- página principal de la aplicación*

- Un footer (pie de página) que está formado por un iframe de Facebook.
- Una lista con ocho secciones donde se encuentra organizado todo el contenido de la página:

  - **Emergency System:** incluye la información necesaria para facilitar el acceso a los centros especializados en el caso de emergencias médicas.
  - **Weather:** proporciona acceso a una aplicación flash que nos muestra la predicción del tiempo en Pamplona.
  - **Food & Drink:** ofrece una lista con los mejores restaurantes normales y fast food, terrazas y cafés.
  - **Car & Transport:** sección que ofrece acceso a gasolineras, talleres y estaciones de autobús y taxi más importantes.
  - **Culture & Education:** apartado para conocer la parte cultural junto con los centros educativos de Pamplona**.**
  - **Fun & Entertainment:** zona para ocio y entretenimiento.
  - **Sports Destinations:** sector que localiza los sitios deportivos más conocidos.
  - **Shopping:** parte destinada al sector comercial de la ciudad.

La mejor manera de entender el esquema de navegación de la aplicación web es observar el diagrama de la figura 4.1.3 que se muestra en la siguiente página.

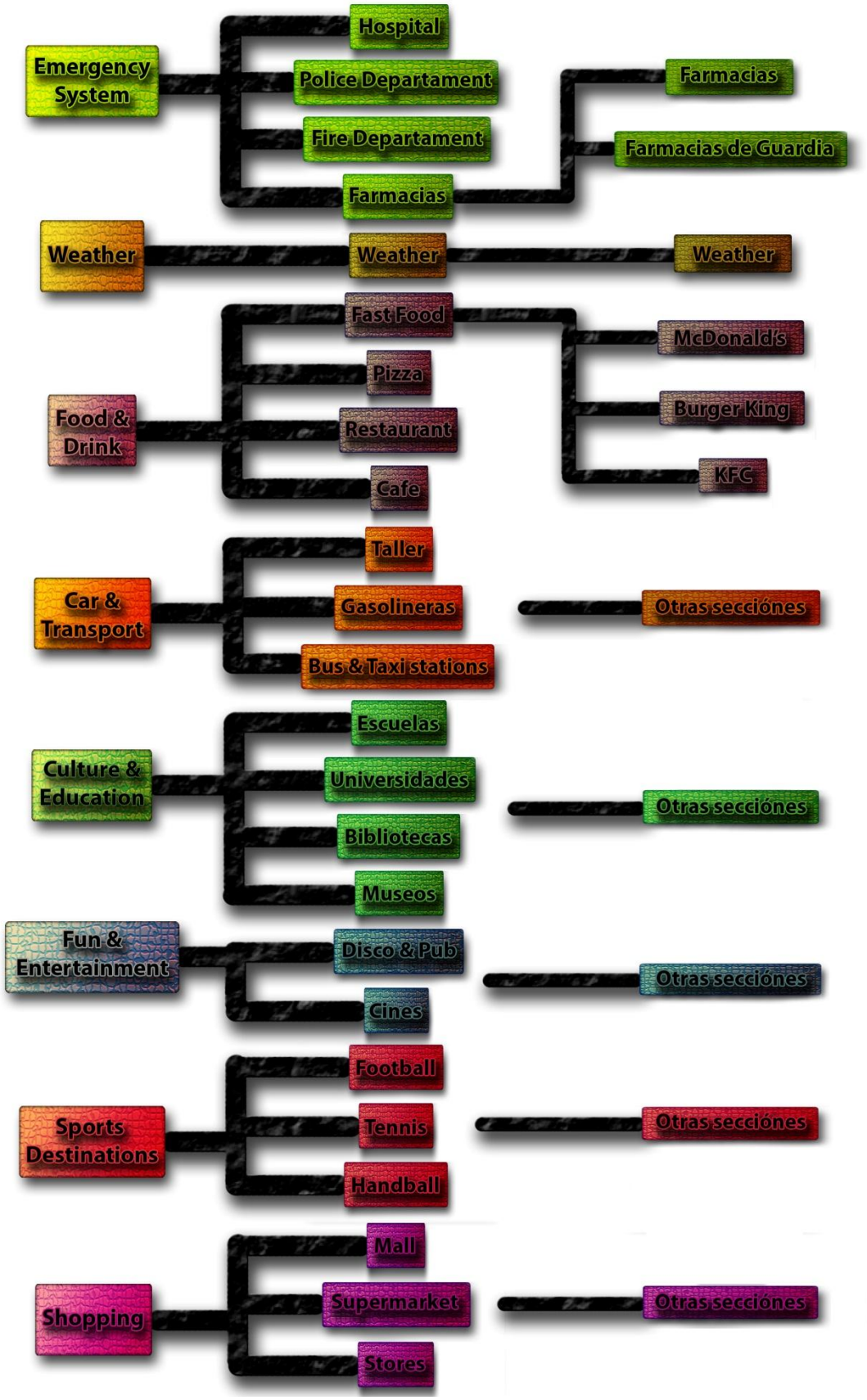



***Figura 4.1.3*** *- Esquema de la aplicación*

En cualquier sección de la aplicación, el usuario dispondrá de un botón para volver a la página principal.

Además se observa que la aplicación está constituida por un máximo de tres niveles, es decir, la navegación es lineal y apenas se tienen que hacer tres clics para llegar a donde desea el usuario lo que hace que sea una navegación sencilla e intuitiva.

## 4.2 Origen de los datos y almacenamiento de los ficheros

### 4.2.1 Los datos y su origen

Lógicamente, lo primero que se necesita en esta aplicación, después de haber terminado el esquema de diseño, son los datos y la información que van a estar disponibles para los usuarios. Para la aplicación **New in town - Pamplona**, se han utilizado dos tipos de datos:

- Datos introducidos manualmente (los datos de la página de McDonald's: la longitud y la latitud, horario etc.)
- Datos abiertos proporcionados por el servicio Open-Data Navarra.

Los datos abiertos son ofrecidos por Open-Data Navarra y se pueden encontrar visitando la página oficial: *http://www.navarra.es/home_es/Open-Data/*. La página dispone de un catálogo amplio de datos distribuidos en cinco páginas de búsqueda. Entre las secciones más importantes podemos encontrar datos sobre:

- **Centros educativos**
- **Tramificación de la red de trafico**
- **Mapa oficial de carreteras de Navarra 2012**
- **Museos de navarra**
- **Bibliotecas públicas**
- **Farmacias de guardia**

La página también dispone de una gran variedad de tipos de datos que se pueden elegir en función del fin que va a tener un archivo particular. A continuación se presenta una lista de las extensiones más importantes y una breve descripción de cada una de ellas:

- **XML:** Es un lenguaje extensible de etiquetas desarrollado por W3C que permite definir lenguajes para diferentes necesidades.
- **CSV:** Los ficheros CSV son un tipo de documento en formato abierto, sencillo para representar datos en formato de tabla.
- **RSS:** Es un formato XML para la distribución de contenidos de páginas web.
- **JSON:** Es un formato ligero para el intercambio de datos basado en la notación literal de objetos de JavaScript.
- **ODS:** Es un contenedor de datos activos, es decir operacionales que ayudan al soporte de decisiones y la operación.
- **KML:** Es una gramática XML y un formato de archivo para la creación de modelos y mapas.

Originalmente se ha pensado que la aplicación contenga sólo tres tipos de datos: XML, JSON y CSV. Luego, cuando el código del proyecto aumentó, por falta de documentación para el formato CSV, se quedaron sólo funciones JavaScript para la lectura e importación de los ficheros XML y JSON.

### 4.2.2 Almacenamiento de la aplicación en la web

Durante todo el proyecto se ha utilizado el término de aplicación web, pero para que una aplicación sea web de verdad debería poder ser visitada desde cualquier ordenador a través de internet. Originalmente, el proyecto ha sido desarrollado en un solo ordenador, guardando todos los ficheros en el disco duro del mismo, pero de esta manera, podríamos acceder a la aplicación únicamente desde dicho ordenador.

Para poder hacer la aplicación disponible para todo el mundo, se ha subido a un servidor de alojamiento gratuito de internet (web hosting). Se ha utilizado la página *http://www.000webhost.com/* y después del registro y de crear la cuenta se ha obtenido el espacio necesario. Es importante especificar que, al ser gratis, sólo se ha obtenido un subdominio, por lo tanto nuestra página quedará de la siguiente forma: *http://www.*ElNombreElegido*.comxa.com/*. En este punto sólo era necesario conectar mediante FTP al servidor recién creado. Para utilizar la función de FTP se ha utilizado el programa Total Commander que tiene un método sencillo para transferir datos. La imagen 4.2.2.1 muestra una captura de pantalla del dicho programa.

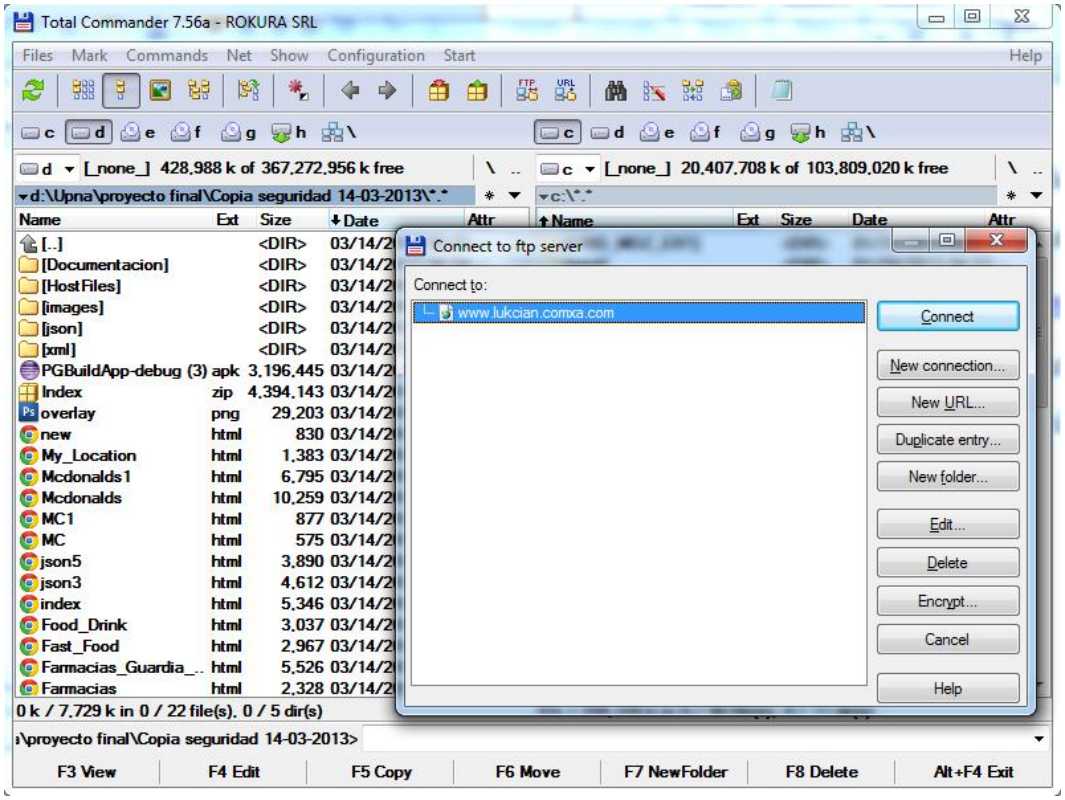



***Figura 4.2.2.1*** *- captura de pantalla del programa Total Commander*

Después de haber subido los ficheros de la aplicación web al espacio otorgado, se puede acceder a la aplicación entera desde cualquier sitio del mundo siempre que haya internet. Cuando se intenta conectar a la aplicación, el servidor junto con el navegador están buscando en la carpeta asignada el fichero index.html, por lo tanto este mismo tiene que existir para que todo funcione bien.

## 4.3 Los códigos para importar los ficheros XML y JSON

### 4.3.1 XML

En el primer caso, se ha intentado publicar los datos que contiene el fichero XML, **Guardias.xml**, descargado de la página Open-Data Navarra y que contiene una lista de farmacias de guardia. La imagen 4.3.1.1 contiene una pequeña parte del fichero, para que podamos observar el estilo XML. Se puede observar que existe una carpeta grande, FARMACIASDEGUARDIA que contiene varias carpetas llamadas FARMACIAGUARDIA. Estas mismas carpetas almacenan una serie de secciones con información sobre cada farmacia.

```
<?xml version="1.0" encoding="UTF-8"?>
<FARMACIASDEGUARDIA>
  <FARMACIAGUARDIA>
    <FECHA>05/11/2012</FECHA>
    <DESDE>9</DESDE>
    <HASTA>9</HASTA>
    <LOCALIDAD>ALTSASU/ALSASUA</LOCALIDAD>
    <GRUPO>Z.B.S. Altsasu/Alsasua</GRUPO>
    <DIRECCION>C/ Alzania, 1</DIRECCION>
    <FARMACIA>LUIS DE REDIN SUBIRA, Mª ROSA</FARMACIA>
    <Cod_FARMACIA>F00159</Cod_FARMACIA>
    <TELEFONO>948 562380</TELEFONO>
  </FARMACIAGUARDIA>
  <FARMACIAGUARDIA>
    <FECHA>05/11/2012</FECHA>
    <DESDE>9</DESDE>
    <HASTA>9</HASTA>
    <LOCALIDAD>AOIZ</LOCALIDAD>
    <GRUPO>Z.B.S. Aoiz - Auritz/Burguete</GRUPO>
    <DIRECCION>C/ Landakoa, 7</DIRECCION>
    <FARMACIA>ROITEGUI AZAGRA,  MARIA CARMEN</FARMACIA>
    <Cod_FARMACIA>F00668</Cod_FARMACIA>
    <TELEFONO>948 334343</TELEFONO>
  </FARMACIAGUARDIA>
```

***Figura 4.3.1.1*** *- Código XML*

Para publicar los datos del fichero .*xml en la página deseada, se necesita colocar un código adecuado en el HTML de dicha página. El código incluirá un script donde comenzará una petición ajax para leer el archivo .*xml. La figura 4.3.1.2 muestra una captura de pantalla con el código utilizado para leer el fichero **Guardias.xml**. La petición Ajax toma en este caso cuatro parámetros:

- **url**: representa la dirección desde la carpeta donde nos encontramos ahora hasta el fichero analizado.
- **dataType**: dice qué tipo de fichero es, en este caso .*xml.
- **success**: en el caso de que exista el fichero, este campo se activará y ejecutará la función que se encuentra aquí.
- **error**: campo opuesto al campo **success**

En el caso de éxito, cuando exista el fichero, se ejecutará la función **data** encontrada en la línea 129, que básicamente, está leyendo cada campo del fichero **Guardias.xml** (fecha, desde, hasta, etc.). En la segunda parte de la función, mediante el comando **append**, se construye el texto que va a aparecer en la pantalla. Es importante observar que, al utilizar **append**, existen dos tipos de código unidos por el símbolo más ( + ):

- el primero, almacena entre comillas ( ' ' **)** el código a ejecutar por el html.
- el otro, sin comillas, busca el texto en el campo indicado del fichero .*xml, y luego lo introduce al código anterior, al de HTML.

La separación hace falta para poder pegar en el código HMTL, los textos que vienen en los campos del fichero .*xml.

```
125 <script>
126         $.ajax({
127     url: 'xml/Guardias.xml',
128     dataType: 'xml',
129     success: function (data) {
130         $(data).find('FARMACIASDEGUARDIA FARMACIAGUARDIA').each(function() {
131             var fecha = $(this).find('FECHA').text();
132             var desde = $(this).find('DESDE').text();
133             var hasta = $(this).find('HASTA').text();
134             var localidad = $(this).find('LOCALIDAD').text();
135             var farmacia = $(this).find('FARMACIA').text();
136
137
138             $('<article class="json"></article>').append(
139
140 '<section class="resultsSource"><h6 class="shareHeader">' +
141                 '<img class="fromUser" src="images/Biblioteca.jpg" width="50" height="50" alt="Biblioteca">' +
142                 '<p> ' + farmacia + '</p>'     +
143                 '</h6></section>' +
144                 '<section class="resultsDescription"><h6>' +
145                 '<a class="titles">Direccion:</a> ' + localidad +
146                 '<p><a class="titles">Teléfono:</a>' + this.TELEFONO + '</p>'  +
147                 '<p><a class="titles">Email:</a> ' + this.CORREO_ELECTRONICO + '</p>'  +
148                 '<p><a class="titles">Fecha:</a> ' + fecha + '</p>'  +
149                 '<p><a class="titles">Hora:</a> ' + ' Desde las ' + desde + ' hasta las ' + hasta + '</p>'  +
150                  '</h6></section>' +
151                 '</br></br>'
152             ).appendTo("#bbb");
153
154         });
155     },
156     error: function() {
157         $('.abc').text('failed to get feed');
158         alert('Failed to get feed');
159     }
160
161 });
162 </script>
```

***Figura 4.3.1.2*** *- código utilizado para leer el fichero* ***Guardias.xml***

A continuación, en la figura 4.3.1.3 se muestra una captura de pantalla de la página resultante del código anterior.


***Figura 4.3.1.3*** *- captura de pantalla de la página "Farmacias de guardia"*

## 4.3.2 JSON

Para la lectura del fichero .*json se ha aprovechado de que en el JQuery existe una función llamada **getJSON** que simplifica las llamadas ajax a un servidor. El formato de esta función es el que se ve en la siguiente figura 4.3.2.1. Esta llamada lo que hace es enviar una petición a una determinada **url** y esperar una respuesta en formato .*json y cuando llega la respuesta se ejecuta una función que procesa la respuesta.

El script empieza con la condición de que el documento esté disponible, luego ejecuta la función **getJSON** que prácticamente lee los datos del fichero **bibliotecas.json**. En la siguiente línea el comando **each** activa la función de adjuntar (**append**) datos para cada biblioteca. Los datos

provienen del fichero .*json donde, parecido al de .*xml, la información viene estructurada en carpetas diferentes para cada biblioteca.

A partir del apartado append, la función será muy parecida a la anterior, publicando en la página HTML las diferentes secciones del fichero .*json. Igual que antes, el código HTML estará situado entre comillas, mientras que el que proviene del fichero de bibliotecas no tendrá comillas. Los dos tipos de texto están unidos por el símbolo más ( + ).

```
<script>
        $(document).ready(function() {
                $.getJSON('json/bibliotecas.json', function(json) {
                $.each(json.Bibliotecas, function () {
                $('<article class="json"></article>').append(

                '<section class="resultsSource"><h6 class="shareHeader">' +
                '<img class="fromUser" src="images/Biblioteca.jpg" width="50" height="50" alt="Biblioteca">' +
                '<p> ' + this.BIBLIOTECA + '</p>'  +
                '</h6></section>' +
                '<section class="resultsDescription"><h6>' +
                '<a class="titles">Direccion:</a> ' +
                 this.POBLACION + ', ' +  this.DIRECCION + ', ' + this.COD_POSTAL +
                '<p><a class="titles">Teléfono:</a>' + this.TELEFONO + '</p>'  +
                '<p><a class="titles">Email:</a> ' + this.CORREO_ELECTRONICO + '</p>'  +
                '<p><a class="titles">Página web:</a> ' + this.PAGINA_WEB + '</p>'  +
                '<a class="titles">Horario:</a><p> ' + this.HORARIO_INVIERNO + '</p>'  +
                '<p> ' + this.HORARIO_VERANO + '</p>'  +
                 '</h6></section>' +
                '</br></br>'

                ).appendTo("#aaa");  /* Appends all ( ... ) to the element with the ID "aaa" */


    });
  });
});
</script>
```

***Figura 4.3.2.1*** *- código utilizado para leer el fichero .*json*

En la figura 4.3.2.2 se puede observar la página obtenida a través del código mostrado anteriormente.


***Figura 4.3.2.2*** *- captura de pantalla de la página "Bibliotecas"*

## 4.4 Aspecto visual

El diseño de una página web se realiza mediante el código HTML, que crea la forma que tendrá el sitio web, ordenando sus diferentes secciones y contenidos de manera estructurada. El aspecto visual, es decir, los colores de los botones, encabezados, tipos de letra y demás se establece utilizando las hojas de estilo CSS.

Además, el framework jQuery Mobile, tiene hojas de estilo predefinidas, ya creadas con diferentes formas y colores. También dispone de listas, botones, encabezados y pies de páginas. Para la webapp se ha utilizado el formato de lista llamado Thumbnails que, según la foto de la derecha, figura 4.4.1, podemos ordenar las secciones deseadas añadiendo en la parte derecha de cada sector una foto representativa. Las fotos se han creado buscando en internet las más adecuadas y a continuación modificadas con Photoshop.


***Figura 4.4.1*** *- Lista Thumbnails*

El tipo de lista es igual en toda la aplicación, pero el tema elegido varía para cada sección. Se han utilizado tres temas de los cinco posibles: blanco, amarillo y negro.


***Figura 4.4.2*** *- Temas disponibles*

La aplicación **New in town - Pamplona** también dispone de hojas de estilo creadas manualmente. A continuación, en la figura 4.4.3, se presenta una parte del código utilizado para los estilos globales de escritura, para publicar la lista .*json.

```
/* Estilos globales de escritura
-------------------------------*/

        article.fbResults h6{
            font-weight:normal; /* el header h6 pone todo el text en tipo BOLD. asi quitamos el B(
            de article.fbResults */
        }
        article.fbResults h6 a.fromName, article.fbResults h6 .shareName{
            font-weight:bold; /* Ponemos solo el nombre en bold */
        }
        article.fbResults a{
            text-decoration:none; /*para quitar el underline effect */
            color=#395A93;
        }
        article.fbResults span.shareCaption, article.fbResults span.shareDescription {
            color:gray;
        }
```

***Figura 4.4.3*** *- Código CSS*

Se ha utilizado CSS personalizado para determinar el aspecto de las listas importadas de los ficheros .*json y .*xml. También se ha diseñado manualmente la página para McDonald's.


***Figura 4.4.4*** *- Página de McDonald's*

## 4.5 Navegación

La navegación a través de la webapp se puede considerar muy simple y lineal. Ya hemos visto anteriormente como es el esquema o diagrama de la aplicación. Solo basta con hacer click en la sección que nos interesa y se activa el salto al siguiente nivel, hasta encontrar la página deseada. Una navegación simple podría ser considerada una ventaja para una aplicación informativa, porque el objetivo de la webapp es llegar rápido y simple a los datos buscados. Además, el usuario dispone en cada nivel de la aplicación de un botón de vuelta a la página inicial.


El jQuery Mobile incluye un conjunto de efectos de transición de paginas basados en CSS, pero no se utilizó ninguna para mejorar en tiempo de respuesta.

## 4.6 Mapas y geolocalización

Esta es una de las herramientas más útiles y más avanzadas que posee la aplicación **New in town - Pamplona**.

### 4.6.1 Descripción de la página

La página consta de dos secciones: una estática y otra dinámica. La estática representa la primera mitad de la página e incluye el logotipo y la información sobre la empresa McDonald's.

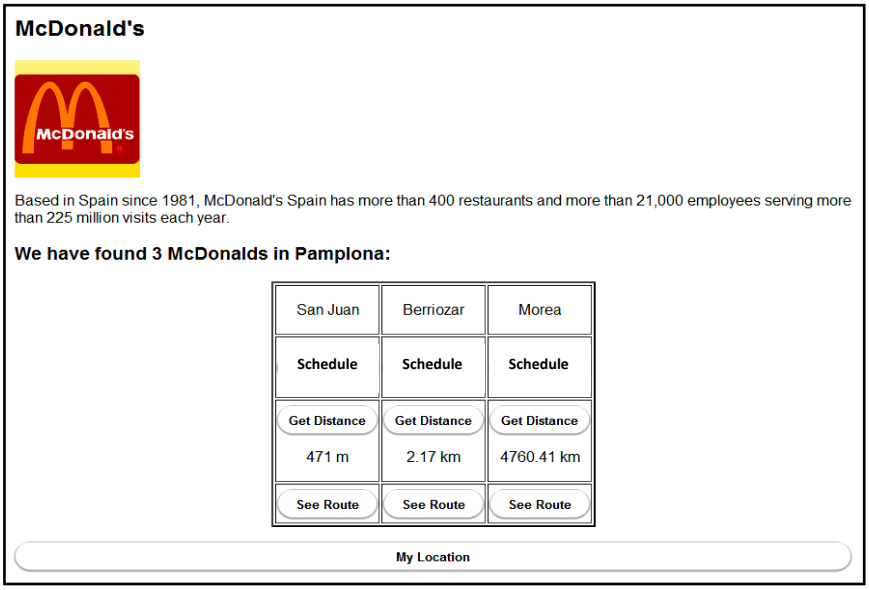



***Figura 4.6.1.1*** *- Pagina McDonald's*

En la segunda mitad, la dinámica, encontramos cuatro secciones:

- **Schedule** (horario) - aquí encontramos el programa de apertura de la empresa.
- **Get distance** (calcular distancia) - en esta sección se puede ver la distancia exacta desde nuestra posición hasta el punto de destino deseado, es decir, la distancia hasta cada uno de los tres McDonald's de la ciudad.
- **See Route** (ver ruta) - este botón nos lleva a una página donde nos permite ver la ruta que se debe seguir desde la posición donde se encuentra el smartphone hasta donde se quiere llegar, o sea, hasta cualquiera de los tres McDonald's de la ciudad.
- **My Location** (mi localización) - nos muestra a través de Google maps la geolocalización del dispositivo móvil.

## 4.6.2 Explicación del código

Al ser una página que realiza funciones más avanzadas que el resto del proyecto, el código también presenta una dificultad mayor que el resto de las funciones vistas a lo largo de la memoria. A continuación se presentara una breve descripción de cómo y qué realiza el código.

- La sección de **body** empieza con cargar la función getLocation, para calcular y guardar los valores de la latitud y longitud.

```
39  <body onload="getLocation()">
```

***Figura 4.6.2.1*** *- Código "onload"*

- Luego se encuentra un script que, en la primera parte asigna a varias variables la longitud y latitud de los tres McDonald's. En la segunda parte se localiza la función de **getLocation** que guarda en dos variables la ubicación geográfica dispositivo en el que se ejecuta el código. Es importante especificar que **getLocation** es una función JavaScript y que la ejecuta el navegador. Dependiendo del dispositivo y del navegador que se utiliza, el tiempo entre hacer la llamada y recibir los resultados, puede variar, por lo tanto, las otras funciones que utilizan dichos resultados se tienen que ejecutar después de ésta misma.

```
44      <script type="text/javascript">
45          var lat1= "42.81071";  // McDonalds San Juan
46          var lon1= "-1.66282";
47          var lat2= "42.83425";  // McDonalds Berriozar
48          var lon2= "-1.66616";
49          var lat3= "42.78626";  // McDonalds Morea
50          var lon3= "-1.63198";
51          var latitude = null;
52          var longitude = null;
53
54          function getLocation() {
55              if (navigator.geolocation) {
56                  navigator.geolocation.getCurrentPosition(function(position){
57                  latitude = position.coords.latitude;
58                  longitude = position.coords.longitude;
59                  });
60              }else {
61                  alert("Geolocation API is not supported in your browser.");
62              }
63              }
64      </script>
```

***Figura 4.6.2.2*** *- Script getLocation*

A continuación tenemos tres funciones **See_Route_** que al ejecutarlas, cada una de ellas abre una nueva ventana que muestra en Google maps la ruta desde la ubicación actual del dispositivo hasta el destino. Las variables **latitude**, **longitude**, **lat_** y **lon_** son las que se han calculado en la parte anterior. Las tres funciones no se activan automáticamente, sino que se ejecutan al pulsar uno de los tres botones **See Route**.

```
75      function See_Route_1(){
76          location.href=('https://maps.google.com/maps?saddr=' + latitude + ',' + longitude + '&daddr=' +
            lat1 + ',' + lon1 + '&z=15&output=embed');
77              // daddr is the location of the first McDonald's: latitude=42.81071 & longitude=-1.66282
78      }
79      function See_Route_2(){
80          location.href=('https://maps.google.com/maps?saddr=' + latitude + ',' + longitude + '&daddr=' +
            lat2 + ',' + lon2 + '&z=15&output=embed');
81              // daddr is the location of the 2 McDonald's: latitude=42.83425 & longitude=-1.66616
82      }
83      function See_Route_3(){
84          location.href=('https://maps.google.com/maps?saddr=' + latitude + ',' + longitude + '&daddr=' +
            lat3 + ',' + lon3 + '&z=15&output=embed');
85              // daddr is the location of the 3 McDonald's: latitude=42.78626 & longitude=-1.63198
86      }
```

***Figura 4.6.2.3*** *- Funciones See_Route*

Un ejemplo de cómo se verá la ruta se podría ver en la siguiente imagen, figura 4.6.2.4, donde **A** representa la localización del móvil y el punto **B** el destino.

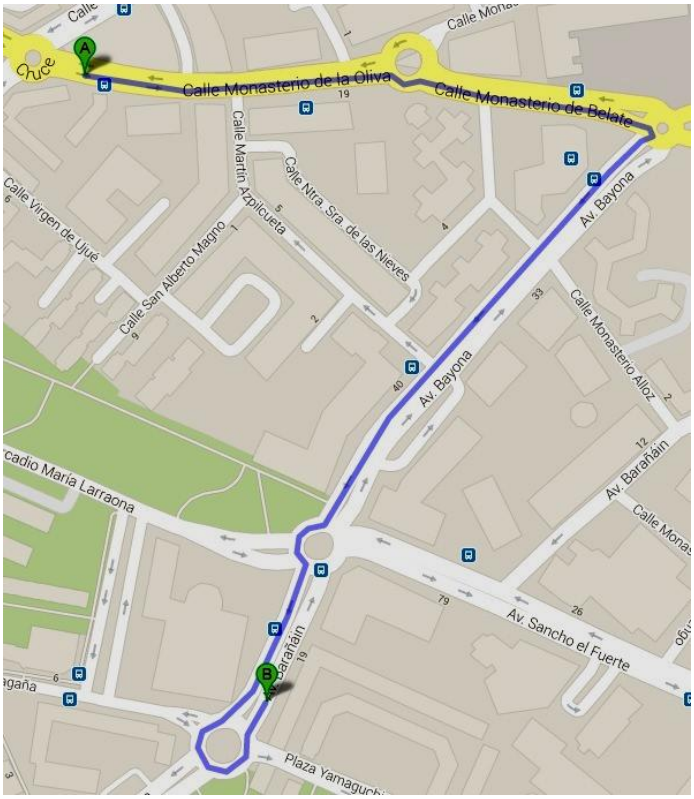



***Figura 4.6.2.4*** *- Captura pantalla ruta*

En la siguiente parte del código se encuentra la función **My_Location** que hace básicamente lo mismo que las anteriores, **See_Route**, pero en este caso nos enseña en un mapa de Google solo la ubicación geográfica del dispositivo. Igual que antes tenemos que activar un botón para ejecutar esta función.

```
88      function My_Location(){
89          location.href=('https://maps.google.com/maps?saddr=' + latitude + ',' + longitude +
            '&z=17&output=embed');
90              // it only shows my location
91      }
```

***Figura 4.6.2.5*** *- Función My_Location*

- La próxima sección, figura 4.6.2.6, incluye dos funciones que muestran con un retardo la distancia entre dos puntos calculada por la función **distance**. El resultado se verá en el cuadrado de **Get Distance** de la tabla que incluye la página web. La primera parte añade para los primeros dos casos (**San Juan** y **Berriozar**) un retardo de 3 segundos. Como se comentó antes, el retardo es necesario porque algunos dispositivos cargan más lento el código, y por lo tanto ejecuta más lento la función **getLocation** y de esta manera dará un resultado mal calculado. Un ejemplo del caso que calcula mal la distancia se puede observar en el tercer campo, en el de **Morea**, donde se ha puesto un retardo de 0.05 segundos, simulando de esta manera el caso cuando esta función intenta ejecutarse antes de que tenga los valores de **latitude** y **longitude** proporcionados por la función **getLocation**. Al intentar ejecutar antes la función **distance**, ésta misma produce un resultado mal calculado: 4760.41 Km. Para corregir el error, se pulsa el botón **Get Distance**, que vuelve a ejecutar la función **distance**, que ahora tiene los valores bien cargados y ofrece un resultado correcto. Esta simulación se ha hecho porque se ha tenido problemas con esta parte del código y se tardó un cierto tiempo en corregirlos, por lo tanto, es importante tenerlo en cuenta. También sirve para observar e intentar evitar este tipo de errores.

```
121     setTimeout(function(){
122         $("#Get_Distance_1").html(distance(lat1,lon1,latitude,longitude));
123         $("#Get_Distance_2").html(distance(lat2,lon2,latitude,longitude));
124     }, 3000);
125     //Obligamos a un delay de 3 segundos para dar tiempo a ejecutarse a la funcion "getLocation"
126     setTimeout(function(){
127         $("#Get_Distance_3").html(distance(lat3,lon3,latitude,longitude));
128     }, 50);
129     //simulando un delay mas corto 0,05 segundos
130     //el browser no tiene tiempo para ejecutar la funcion "getlocation" por lo tanto las variables
        "longitude" y "latitude" no se actualizan correctamente y el resultado de la funcion "distance"
        sera mal calculado
```

***Figura 4.6.2.6*** *- Funciones de retardo*

- En el siguiente apartado se presenta la función de **distance**, que calcula la distancia entre dos puntos geográficos. Se introducen cuatro valores, dos de longitud y dos de latitud, uno para cada uno de los dos puntos. Luego a través de una fórmula matemática se obtiene la distancia en Km o en m en función de la necesidad.

```
132     function distance(lat1,lon1,lat2,lon2) {
133         var R = 6371; // km (change this constant to get miles)
134         var dLat = (lat2-lat1) * Math.PI / 180;
135         var dLon = (lon2-lon1) * Math.PI / 180;
136         var a = Math.sin(dLat/2) * Math.sin(dLat/2) +
137             Math.cos(lat1 * Math.PI / 180 ) * Math.cos(lat2 * Math.PI / 180 ) *
138             Math.sin(dLon/2) * Math.sin(dLon/2);
139         var c = 2 * Math.atan2(Math.sqrt(a), Math.sqrt(1-a));
140         var d = R * c;
141         if (d>1) return Math.round(d*100)/100+" km";
142         else if (d<=1) return Math.round(d*1000)+" m";
143     return d;
144     }
```

***Figura 4.6.2.7*** *- Función distance*

- Después de haber creado bien las funciones solo nos queda la parte de asignarlas a los botones (figura 4.6.2.8). El comando para crear botones, **button**, junto con el comando **onclick**, se crean los botones y se programa que al pulsarlos se activen las funciones correspondientes:
    - a los botones **Get Distance** las funciones **Get_Distance_(1,2,3)**
    - a los botones **See Route** las funciones **See_Route_(1,2,3)**
    - al botón **My Location** la función **My_Location**

```
213 <td> <button onclick="Get_Distance_1()" data-role="button" data-mini="true">Get Distance</button>
214 <p  id='Get_Distance_1' style="text-align:center">
215 <script>
216 document.write('calculating');
217 </script></p>
218 </td>
252 <td>
253 <button onclick="See_Route_2()" data-role="button" data-mini="true">See Route</button>
254 </td>
271         <p><button onclick="My_Location()" data-role="button" data-mini="true">My Location</button></p>
```

***Figura 4.6.2.8*** *- Código botones*

# 4.7 Implementación de PhoneGap y la utilización de QR-code

## 4.7.1 El papel de PhoneGap en la aplicación

Como se ha comentado antes, el objetivo final de esta aplicación ha sido desarrollar una aplicación híbrida. Hasta ahora se ha visto la composición de la aplicación web. El siguiente paso será convertirla en una aplicación híbrida, es decir, una aplicación .*apk que se puede descargar e instalar en un sistema operativo, en este caso Android. Para conseguir esta transformación se ha utilizado el programa online Adobe PhoneGap Build.

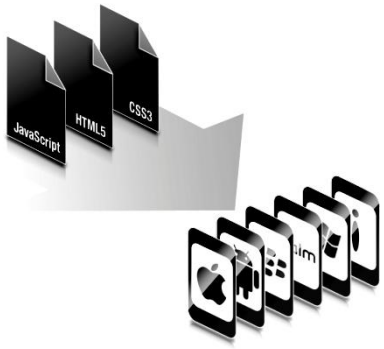



***Figura 4.7.1.1*** *- Logo PhoneGap*

Ya se ha explicado antes que PhoneGap crea aplicaciones multiplataforma para móviles a partir de los ficheros HTML, CSS y JavaScript. El gran problema que se ha tenido con la conversión utilizando PhoneGap ha sido la falta de documentación, es decir, cómo hacer exactamente para pasar de una aplicación web, que es básicamente una página web construida por varios ficheros .*html, a una aplicación semi-nativa. Después de haber entrado en el programa online PhoneGap Build y después de registrarse utilizando el Github, nos daba la posibilidad de subir un archivo .*zip, pero sin especificar qué hay que colocar dentro. Tras varios intentos y buscar información adicional en internet, se ha arreglado el problema. La solución ha sido crear una carpeta con todos los ficheros de la aplicación web, es decir, todas las páginas HTML (incluyendo index.html), fotos y datos adicionales utilizados en la aplicación. Después de haber hecho esto se utilizó el programa Winzip para comprimir la carpeta de antes en un archivo index.zip. Si no se incluyera la página principal de la aplicación con el nombre de index.html, la conversión no funcionaría correctamente. A partir de este paso todo resultó más fácil. Al introducir el archivo .*zip, el PhoneGap empezó automáticamente la conversión y al final enseñó para qué sistemas operativos se ha conseguido la transformación. La foto 4.7.1.2 muestra en azul los OS que han tenido éxito y con rojo los que no.


***Figura 4.7.1.2*** *- OS que han tenido éxito*

### 4.7.2 QR-code

El QR code (quick response code) es un módulo útil para almacenar información en una matriz de puntos o un código de barras. Se caracteriza por los tres cuadrados que se encuentran en las esquinas y que permiten detectar la posición del código al lector.

El PhoneGap ofrece la posibilidad de descargar la aplicación recién creada a través de un código QR que él mismo te lo genera. La figura 4.7.2.1 muestra el código para la aplicación **New in town - Pamplona** y representa básicamente un url que guarda la aplicación. Al intentar leerlo con un QR lector se activará automáticamente la descarga de la aplicación híbrida.

***Figura 4.7.2.1*** *- QR code*

La utilización de este código es una manera más fácil de transmitir informaciones sin la necesidad de copiarlas. En el caso de este proyecto, se puede transmitir la aplicación a través del código, sin que la mandemos por bluetooth o por correo.

## 4.8 Elementos adicionales

### 4.8.1 El reloj flash

La animación que enseña la hora y se encuentra debajo de cada título es un JavaScript que se ha obtenido de la librería ofrecida en el portal: *http://www.clocklink.com/*. El sitio web dispone de una variedad de relojes que se pueden integrar en una página web personal. Además de enseñar la hora, el reloj da un aspecto atractivo a la aplicación, eliminando de esta forma el aspecto estático. Tanto el código como el reloj generado se pueden ver en la figura 4.8.1.1.

```
26  <script src="http://www.clocklink.com/embed.js"></script><script type="text/javascript" language=
    "JavaScript">obj=new Object;obj.clockfile="5012-black.swf";obj.TimeZone="GMT0100";obj.width=91;
    obj.height=30;obj.wmode="transparent";showClock(obj);</script>
```


***Figura 4.8.1.1*** *- Reloj flash, código y animación resultante*

Las características más importantes del código son:

- **src** - representa la página fuente, de donde se ha copiado el reloj.
- **obj.clockfile** - a este campo se le asigna el modelo del flash que se desea. En este caso es "**5012-black.swf**".
- **obj.TimeZone** - aqui se especifica la zona temporal. Para Pamplona, GMT= +1.
- **obj.width** y **obj.height** - establecen las dimensiones del reloj.
- **obj.wmode** - el campo decide qué tipo de transiciones habrá entre cifras.

### 4.8.2 Los logotipos de las secciones

Los dibujos pequeños y representativos que acompañan a las secciones de cada lista de la aplicación web, son imágenes obtenidas de internet y luego editadas y modeladas en Photoshop. Se ha cambiado el fondo y las dimensiones para que coincida con el tema de la página y con las dimensiones de las listas. La foto 4.8.2.1 enseña una captura de pantalla del proceso de edición.


***Figura 4.8.2.1*** *- Captura pantalla Photoshop*

A continuación se presenta una serie de logotipos utilizados en la aplicación.


***Figura 4.8.2.2*** *- Logotipos de la aplicación*

### 4.8.3 El widget de Weather

En la página principal de la aplicación se puede observar una sección de weather que representa el sector de la predicción meteorológica. Al seleccionar el botón, la aplicación nos lleva a otra página, donde encontramos un widget de AccuWeather que nos predice el tiempo. El widget es una pequeña aplicación o programa, usualmente presentado en archivos o pequeños ficheros que son ejecutables por un motor de widgets. Entre sus objetivos están: dar acceso fácil a funciones frecuentemente usadas y proveer información visual.


***Figura 4.8.3.1*** *- Widget Weather*

La forma de construcción de la página es similar a la del reloj. Para obtener el código hace falta entrar en la página http://www.widgetbox.com/ y escribir en campo de búsqueda: **AccuWeather.**

Después de encontrar el widget, podemos personalizar unas características y luego, la página nos proporciona el código a integrar.

### 4.8.4 iframe de Facebook

Un iFrame sirve para crear un espacio dentro de una página web donde se puede introducir otra página web. Es un cuadrado cuyas dimensiones las debe especificar el desarrollador en la etiqueta **iframe**. La página web que se introduce tendrá sus propios contenidos y estilos, independientes del contexto donde estará mostrado. Si tuviera scripts o aplicaciones se ejecutarían de manera autónoma como si fuera una página web normal.

En este proyecto se ha utilizado una sección **iframe** en el pie de la página principal. La página introducida en el cuadrado es un perfil de **Facebook**, al que la gente le puede dar o no **Me gusta**. La foto 4.8.4.1 presenta una captura de pantalla del dicho **iframe**.


***Figura 4.8.4.1*** *- iframe Facebook*

Es importante especificar que la aplicación **New in town - Pamplona** no tiene perfil en Facebook y se ha utilizado el perfil de la UPNA para el **iframe**. Si la aplicación llegara al mercado, sería fácil cambiar el perfil y, de esta manera, se conseguiría aumentar la publicidad de la aplicación.

En la foto 4.8.4.2 se puede observar el código utilizado para la creación del **iframe**.

```
173  <iframe class="likeShare" height="30" src=
     "http://www.facebook.com/plugins/like.php?href=http://www.facebook.com/upnauniversidad?rf=110928785597767"
     ></iframe>
```

***Figura 4.8.4.2*** *- Código iframe*

El contenido de la etiqueta "**src**" se puede construir fácilmente accediendo a la página de plugins de Facebook *https://developers.facebook.com/docs/plugins/* y eligiendo los parámetros deseados. Después se generará el código y se copiará en la sección de arriba.

## 4.8.5 Animación loop New in Town


***Figura 4.8.5.2*** *- Animación página principal*

La animación que se localiza en la parte de arriba de la página principal (figura 4.8.5.2) es básicamente una imagen muy larga (figura 4.8.5.3) que se cambia constantemente de posición hacia arriba. Dispone de una máscara colocada encima de esta imagen que muestra sólo un cuadrado de la misma. De esta manera, al girarse infinitamente, da la sensación de una animación que nunca termina de moverse.

El código de la animación se puede observar abajo en la figura 4.8.5.1. Como se ha especificado antes, cada 50 milésimas de segundo la imagen sube un paso y se repite la misma acción hasta que el valor de la variable **current** llega a cero. Después vuelve a tener el mismo valor que al principio y la aplicación empieza de nuevo. Todo el proceso se repite constantemente.

```
<script type="text/javascript" charset="utf-8">
        var scrollSpeed = 50;       // Speed in milliseconds
        var step = 1;               // How many pixels to move per step
        var current = 0;            // The current pixel row
        var imageHeight = 4300;     // Background image height
        var headerHeight = 70;      // How tall the header is.

        //The pixel row where to start a new loop
        var restartPosition = -(imageHeight - headerHeight); //-4250

        function scrollBg(){
            //Go to next pixel row.
            current -= step;
            //If at the end of the image, then go to the top.
            if (current == restartPosition){
                current = 0;
            }
            //Set the CSS of the header.
            $('#header').css("background-position","0 "+current+"px");
            }
        //Calls the scrolling function repeatedly
        var init = setInterval("scrollBg()", scrollSpeed);
</script>
```

***Figura 4.8.5.1*** *- Código de la animación*

***Figura 4.8.5.3*** *- Imagen entera de la animación*

# CAPÍTULO 5. CONCLUSIONES

La primera conclusión a la que se ha llegado tras realizar este proyecto es que el jQuery Mobile es una excelente solución para crear una web específica para móviles de manera que permita a los usuarios de las plataformas más extendidas acceder a la información de una forma más rápida y cómoda. Aunque a veces tardaba más en cargar las páginas o las transiciones de las mismas en Android producían problemas, JQM es un proyecto estable en donde se están invirtiendo considerables esfuerzos para continuar el desarrollo. Una ventaja es el hecho de que el framework es bastante popular, lo cual significa que existe un gran número de páginas web de ayuda, complementos y tutoriales.

Después de su lanzamiento se esperaba algo mejor del servicio de datos abiertos de Navarra (Open-Data Navarra). Aunque la idea de utilizar datos abiertos fue un gran éxito en países como Alemania o Inglaterra, incluso en algunas regiones de España, la de Navarra muestra grandes problemas y no ha llegado a la altura de las expectativas. Se han encontrado varios datos que no estaban actualizados o estaban incompletos. Sin embargo, el sitio web muestra potencial y si se invierte tiempo y dinero podría llegar a ser considerada una fuente confiable.

Por último hay que señalar que, si bien hemos conseguido crear una aplicación .*apk, no tiene el mismo nivel que las de Google Play o de Apple Store. La razón es que es lenta y bastante básica. Esto se debe a que se ha tenido poco tiempo y experiencia para desarrollar la aplicación. También se tiene que mencionar que para crear una aplicación avanzada nativa hace falta tiempo y un equipo con experiencia en programación Java, JavaScript, HTML etc.

# CAPÍTULO 6. POSIBLES MEJORAS

Una posible mejora sería desarrollar la aplicación desde el principio como una aplicación nativa, es decir, programarla con el lenguaje Java y utilizando un programa específico para aplicaciones Android. De esta manera, se conseguiría que la aplicación tenga un aspecto adecuado y también mejoraría el tiempo de respuesta, o sea, funcionaría más rápido. Una consecuencia de esta misma mejora sería la posibilidad de subir la aplicación al mercado de Google (Google play) con la oportunidad de venderla.

Otra mejora para la aplicación sería la introducción de un formulario dirigido al público. Serviría para recibir comentarios y demandas. Utilizando esta opción, los usuarios podrían ayudar al progreso de la aplicación sugiriendo nuevos locales para colocar en la lista de la misma. Por ejemplo si un turista se encuentra en un bar que no sale en la aplicación, podría mandar un anuncio para señalar con el fin de actualizar la lista de bares junto con el indicado.

Incorporar informaciones en tiempo real del tráfico facilitaría el transporte de los ciudadanos y turistas. De esta manera, las personas que están esperando su transporte podrían aprovechar el tiempo restante de otra forma. Realizar este apartado aportaría un gran avance para la aplicación. Sólo se podría implementar esta opción si se tuviera acceso a informaciones del tráfico que se actualizan constantemente.

Otras ideas que podrían mejorar la aplicación:

- distribuir el código QR en estaciones y locales públicos para aumentar la publicidad.
- ofrecer acceso a guías turísticos online que podrían ofrecer informaciones y ayuda.
- proporcionar un teléfono de atención al cliente para los casos de emergencia con línea directa a policía o a hospital.

# CAPÍTULO 7. BIBLIOGRAFÍA

www.javascriptya.com.ar/ - Documentación sobre jQuery Mobile

www.jquerymobile.com - Documentación sobre jQuery Mobile

www.google.es - Motor de búsqueda

www.wikipedia.com - Web con varias informaciones y explicaciones

es.answers.yahoo.com - Web para preguntas y dudas

www.desarrolloweb.com - Web sobre programación web

Libro de la biblioteca: "**HTML, XHTML, and CSS Bible**" de *Bryan Pfaffenberger, Steven M. Schafer, Chuck White and Bill Karow*

Libro de la biblioteca: "**Build your own WEB SITE THE RIGHT WAT using HTML & CSS**" de *Ian Lloyd*

Libro de la biblioteca: "**JavaScript - The definitive guide**" de *David Flanagan*