ESCUELA TÉCNICA SUPERIOR DE INGENIEROS INDUSTRIALES Y DE TELECOMUNICACIÓN

Titulación:

INGENIERO TÉCNICO DE TELECOMUNICACIÓN, ESPECIALIDAD EN SONIDO E IMAGEN.

Título del proyecto:

DESARROLLO DE UNA GUÍA PARA DISPOSITIVOS MÓVILES DE ESTABLECIMIENTOS PARA CELÍACOS EN LOGROÑO.

Cristina Glera Aransay

Tutor: Marko Galarza Galarza

Pamplona, 27 Junio de 2013

1.Objetivos

El objetivo principal del proyecto es el desarrollo de una aplicación para dispositivos móviles, dicha aplicación consiste en una guía de los establecimientos que disponen productos especializados para celíacos en la ciudad de Logroño.

Se estima que la enfermedad celíaca afecta al 1% de la población, por ello aprovechando el auge de los smartphones se ha pensado crear una aplicación que facilite la vida cotidiana de estas personas.

Debido a que los usuarios serán personas de diferentes edades que pueden estar poco familiarizadas con estas tecnologías se tratará de realizar un diseño sencillo e intuitivo que facilite su uso.

La aplicación será creada mediante tecnologías web y posteriormente se empaquetará para conseguir la aplicación final con alguna característica nativa.

Se estudiarán varias plataformas de desarrollo capaces de realizar este proceso de empaquetamiento y finalmente se elegirá aquella que cubra las necesidades de la aplicación y por ello sea considerada la más apropiada para el proyecto.

A continuación se mostrará detalladamente el proceso de desarrollo de la aplicación, no sin antes explicar algunos conceptos básicos y profundizar en las tecnologías utilizadas, para facilitar la compresión del proceso.

En último lugar se mostrarán algunas mejoras posibles de la aplicación que se podrán realizar en un futuro y las conclusiones obtenidas tras la realización del proyecto.

2.Introducción

Actualmente la tecnología juega un papel fundamental en nuestra sociedad, hasta el punto de que no se podría entender la vida actual sin el uso de ordenadores. La aparición de las nuevas tecnologías en la sociedad supuso la agilización, optimización y el perfeccionamiento de las actividades cotidianas.

Sin duda uno de los campos que más ha notado estas mejoras es el de la comunicación, a través de Internet podemos comunicarnos de forma rápida con cualquier parte del mundo y durante las 24 horas del día.

Es tan importante la presencia de dispositivos electrónicos en nuestra vida, que aunque quizás no nos paramos a pensar en ello, siempre llevamos uno encima. Y es que el actual boom tecnológico es sin duda el mundo de la telefonía móvil e Internet.

Durante mucho tiempo estos dos términos han estado separados, el teléfono móvil se trataba de un aparato que sólo servía para contactar mediante llamadas y mensajes con otras personas. Ahora estos dos mundos van de la mano y el teléfono móvil se ha convertido en un dispositivo indispensable por muchos usuarios para acceder a Internet y realizar numerosas actividades. De modo que el 59 % de la población española ya dispone de un smartphone.

Según el último estudio realizado mediante encuestas a usuarios de Internet, llevado a cabo por AIMC (Asociación para la Investigación de Medios de Comunicación), los dispositivos móviles han incrementado de manera notable su uso para acceder a Internet.

Tanto es así que prácticamente han llegado al mismo nivel que los ordenadores portátiles. El 79 % de los usuarios se conectan a Internet a través de su

smartphone mientras que el 80,95% lo hace desde su ordenador portátil y el 78,8% desde el ordenador de sobremesa.

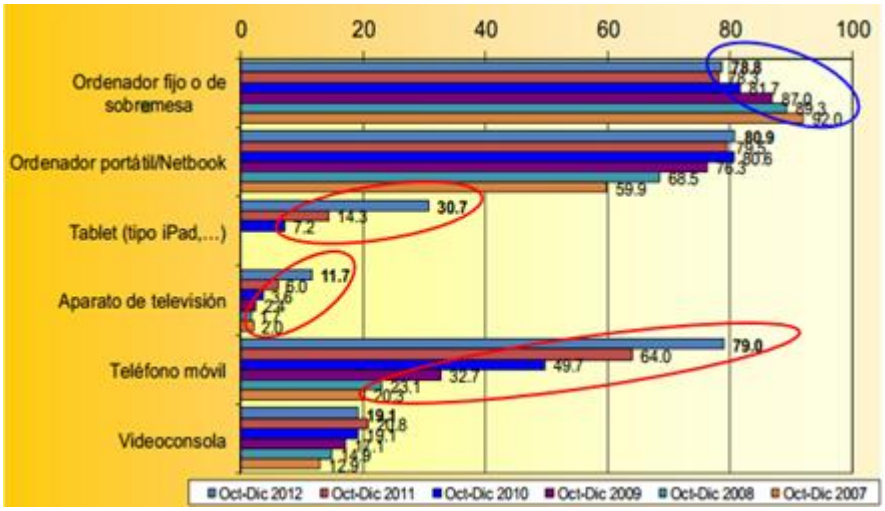

Figura 2.1. Gráfica de los equipos de acceso a Internet.

Otra de las informaciones interesantes que nos aporta este estudio es que solo una minoría de usuarios (27%) cree que el móvil sirve para acceder a Internet de forma puntual frente a un 51% que piensa que sirve para usarlo de manera rutinaria.

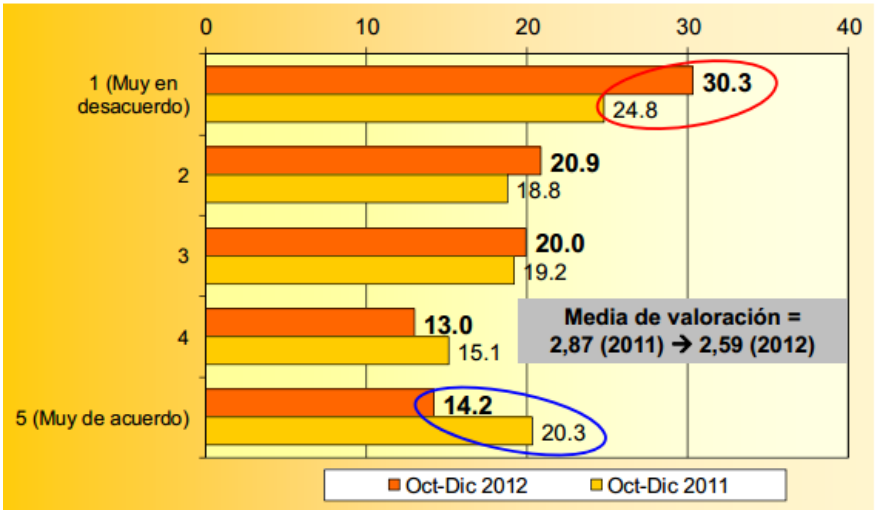

Figura 2.2. Gráfica "Es para usar de forma puntual, para alguna necesidad".

El siguiente gráfico nos muestra la frecuencia con la que los usuarios de smartphones se conectan a Internet en varios países europeos. Como se puede observar todos los países coinciden en que los usuarios se conectan en varias ocasiones a lo largo del día.

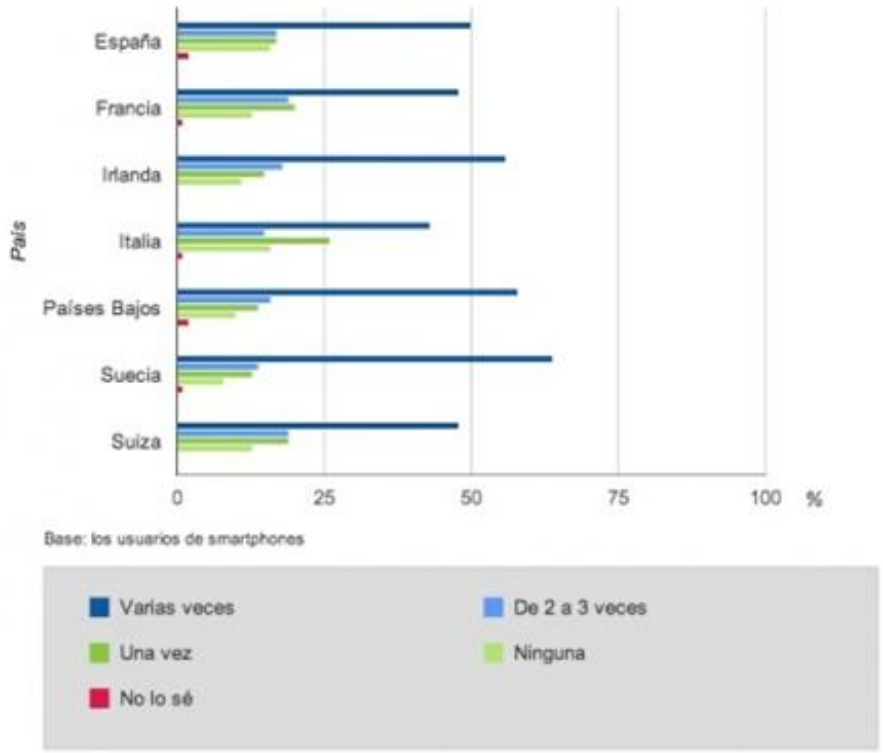

Figura 2.3. Número de sesiones online desde el Smartphone al día.

La utilización de los dispositivos móviles, tanto teléfonos como tabletas, también debe su crecimiento a las aplicaciones. De acuerdo con los resultados de las encuestas anteriores, el 59,1% de los usuarios que acceden a Internet mediante dispositivos móviles, utilizan las aplicaciones varias veces al día.

En cuanto al tipo de aplicaciones más utilizadas depende del dispositivo mediante el cual se accede. Aquellos que usan un smartphone prefieren las de mensajería. Sin embargo en el caso de las tabletas, las aplicaciones más utilizadas son las de correo, seguidas por las de información.

Este elevado uso de los dispositivos móviles para acceder a Internet es sin duda porque ofrecen una serie de prestaciones que no conseguimos con los ordenadores.

Evidentemente una de las grandes ventajas es la movilidad. Su tamaño junto con la autonomía que proporcionan las baterías permite al usuario poder transportarlos. Gracias a esta movilidad, es posible una de las más importantes funcionalidades de los sistemas móviles, el GPS. Utilizado en numerosas aplicaciones, por ejemplo para mostrar lugares cercanos a la localización del usuario o incluso para ayudar a éste a desplazarse.

También ofrecen varias formas para acceder a Internet: la gran mayoría cuentan con conexión wi-fi, pero también con el sistema 3G o 4G. Muchos de los usuarios reconocen que incluso estando en su domicilio utilizan los dispositivos móviles, quizás tenga mucho que ver la posibilidad de conectarse a la red wi-fi de su hogar.

Por otro lado, la constante evolución de los teléfonos móviles ha conseguido que éstos dispongan de funcionalidades que hace tiempo quizás no llegábamos ni a imaginar. Con los nuevos terminales es posible realizar gran variedad de actividades, como puede ser reproducir música o videos, realizar fotos, grabar sonido y videos...

A pesar de todas estas ventajas y posibilidades que nos ofrecen los dispositivos móviles aún no pueden satisfacer todas nuestras necesidades. Existen puntos en los que los ordenadores portátiles o de escritorio siguen siendo muy superiores.

Aunque el reducido tamaño mencionado anteriormente facilita que puedan transportarse también tiene una parte negativa. El tamaño de la pantalla es

pequeño, por lo que en ocasiones resulta incomodo navegar con Internet ya que la gran mayoría de las páginas web no están adaptadas a los móviles.

Una de las principales razones que exponen aquellas personas que no quieren adquirir un smartphone es el elevado precio. Debido a todas las prestaciones que nos aportan los smartphones tienen un precio superior al que nos tenían acostumbrados los teléfonos móviles convencionales.

Otro punto débil de los dispositivos móviles es que aunque tienen una batería que proporciona una autonomía suficiente, debido a todos los recursos del dispositivo y si el usuario utiliza mucho su móvil, la batería apenas tiene una duración de un día entero. En el caso de los móviles convencionales las baterías tienen mayor duración. Es este punto el que produce mayor frustración a los fabricantes de móviles y en el que más se está investigando.

Después de este análisis sobre los dispositivos móviles en la actualidad, es más que evidente la importancia de éstos en la sociedad. Además también hemos visto cómo la mayoría de los usuarios los utilizan para acceder a aplicaciones, por lo que esta aplicación puede ser muy utilizada.

3.Tipos de aplicaciones

3.1. Aplicaciones nativas.

3.1.1. ¿Qué es una aplicación nativa?

Una aplicación nativa es una aplicación que ha sido desarrollada específicamente para ejecutarse en el sistema operativo de un dispositivo y que se instala en dicho dispositivo. Este tipo de aplicaciones son diseñadas de forma específica para cada terminal, es decir, se desarrollan en diferentes lenguajes de programación en función del sistema operativo en el que van a ser utilizadas.

Por ejemplo, aquellas aplicaciones que sean desarrolladas para iOS, es decir para el sistema operativo de iphone e ipad, son diseñadas con los lenguajes: Objetive C. Por otro lado las que son desarrolladas para el sistema operativo de Android utilizan el lenguaje Java.

Sistema operativo	Empresa	Lenguaje de programación
Symbian OS	Symbian Foundation	C++
BlackBerry OS	RIM	Java
iPhone OS	Apple	Objetive C
Windows Phone	Microsoft	C#
Android	Google	Java

Figura 3.1. Principales lenguajes de programación por plataforma.

Las aplicaciones nativas al ser desarrolladas de forma específica con un lenguaje propio para el sistema operativo del terminal, corren de forma más eficiente sobre los dispositivos. Además, por este mismo hecho de un desarrollo específico, estas aplicaciones permiten emplear sensores y elementos del teléfono como por ejemplo, la cámara, el sistema GPS, la agenda u otras herramientas.

El código fuente de estas aplicaciones, escrito en un lenguaje específico para el dispositivo a utilizar, se compila en un ejecutable. Así todos los recursos utilizados por la aplicación como pueden ser imágenes o iconos están incluidos en dicho archivo. Finalmente este archivo compilado está ya preparado para ser distribuido mediante los canales de compra específicos de los dispositivos.

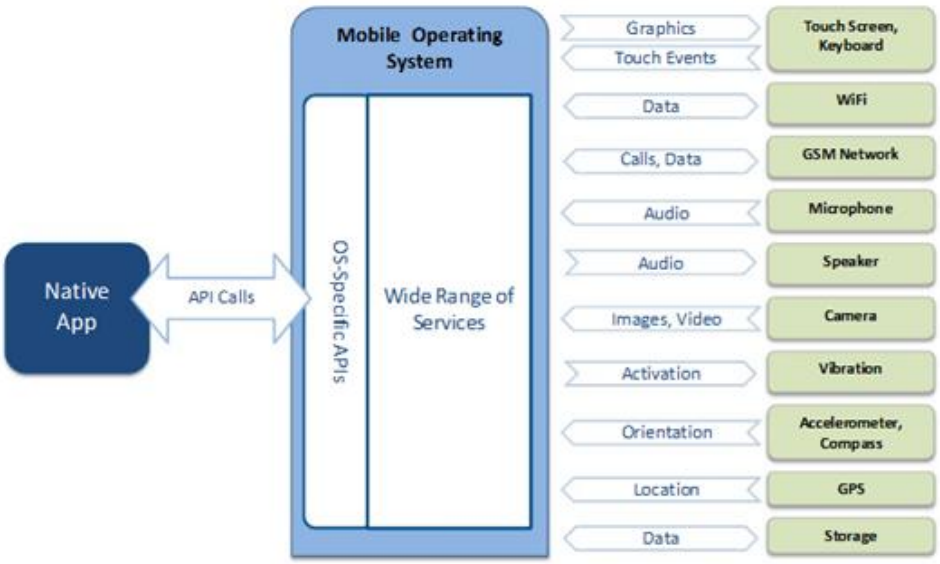

Figura 3.2. Esquema de funcionamiento de una aplicación nativa.

De este modo, antes de comenzar a diseñar una aplicación nativa es necesario conocer en que dispositivos va a ser ejecutada. Para ello, podemos tener en cuenta dos aspectos: la población objetivo y los requisitos técnicos.

- *Población objetivo:* lo que se busca en todo momento es que pueda ser utilizada por el mayor número de usuarios posible. Son muchos los dispositivos con sistemas operativos que soportan Java, por ello la plataforma que nos permite llegar a más usuarios es Java Me. Esta tecnología permite adaptarse a entornos limitados y hace posible la creación de aplicaciones Java que pueden ser ejecutadas en pequeños dispositivos con recursos más limitados.

- *Requisitos técnicos:* si bien en el punto anterior se llegaba a la conclusión de que la tecnología Java era soportada por más sistemas operativos, también cabe destacar que no es la más eficiente a la hora de acceder a todas las posibilidades del terminal. En este aspecto Symbian, que se trata de un sistema operativo formado por la alianza de varias empresas, entre ellas Nokia, proporciona más funcionalidades aunque también es más complejo.

En conclusión, a pesar de ser la más costosa, la mejor solución es crear la aplicación para el mayor número posible de sistemas operativos.

3.1.2. Ventajas y desventajas de las aplicaciones nativas.

Ventajas de las aplicaciones nativas.

- Una de las mayores ventajas es que la aplicación está **instalada en el dispositivo** y se dispone de un acceso directo, de modo que el usuario no tiene que memorizar ninguna dirección.

- Es posible utilizar **los canales de compra de aplicaciones móviles o market places** como el App Store de iOs o el Google Play de Android y así encontrar las aplicaciones más fácilmente que buscando en la web.

- Debido a que son programadas en un lenguaje específico, **las utilidades del dispositivo son accesibles**, como la cámara o el dispositivo GPS.

- Estas aplicaciones pueden ser ejecutadas sin **una conexión a internet**, aunque en ocasiones algunas partes de la aplicación pueden requerir conexión.

- Permiten **notificaciones Push**, éstas consisten en el envío de mensajes al usuario acerca de alguna novedad sobre la aplicación**.**

Desventajas de las aplicaciones nativas.

- El usuario debe **actualizar la aplicación manualmente** desde el canal de compra de su dispositivo.

- Una de las grandes desventajas es tener que desarrollar una **aplicación para cada plataforma.**

- La complejidad de los lenguajes de programación que también conlleva **un mayor tiempo y coste** a la hora de desarrollar dichas aplicaciones.

- El desarrollador se enfrentará a **procesos de validación** en ocasiones complejos a la hora de publicar su aplicación en los distintos market places.

3.2. Aplicaciones web.

3.2.1. ¿Qué es una aplicación web?

Se denomina aplicación web o WebApp, a aquellas aplicaciones que los usuarios pueden utilizar accediendo a un Servidor web a través de Internet o de una intranet mediante un navegador. En otras palabras, es una aplicación software que se codifica en un lenguaje soportado por los navegadores web permitiendo su ejecución en éstos.

Hoy en día estas aplicaciones son populares porque permiten de forma sencilla una comunicación más fluida y dinámica en la computación cliente-servidor, así como a la facilidad para actualizar y mantener dichas aplicaciones web sin distribuir e instalar software específicos a cada usuario. Existen aplicaciones web como los webmails, wikis, weblogs, tiendas en línea e incluso la tan conocida y utilizada Wikipedia.

Sin profundizar mucho, la arquitectura de una aplicación Web normalmente se basa en una estructura que consta de tres-capas. La primera capa la proporciona el navegador Web y un motor capaz de usar alguna tecnología Web dinámica (como por ejemplo PHP) constituye la capa de en medio. La tercera y última capa la constituye una base de datos. De modo que el navegador manda peticiones a la capa de en medio que proporciona información valiéndose de las consultas que realiza a la base de datos.

Se trata de un tipo especial de aplicación cliente/servidor donde tanto el cliente (navegador o explorador), como el servidor (servidor web) y el protocolo mediante el que se comunican (HTTP, FTP…) están estandarizados y no han de

ser creados por el programador. Analicemos los siguientes términos para comprender mejor el funcionamiento:

- *Cliente:* el cliente web es un programa mediante el cual el usuario interacciona para solicitar a un servidor web el envío de los servicios que desea obtener mediante HTTP o FTP.

  Esta parte cliente de las aplicaciones suele estar formada por el código HTML y también por algo de código ejecutable de script del navegador, como por ejemplo JavaScript. Es posible que también se utilicen algunos plug-ins que permiten la visualización de otros contenidos multimedia. En conclusión, la función del cliente web es la interpretación tanto de las páginas HTML como de todos aquellos recursos que son incluidos en éstas, como CSS o Javascript.

- *Servidor:* el servidor web es un programa que permanece constantemente esperando las solicitudes de conexión mediantes por parte de los clientes web y le ofrece los recursos que éste le solita. Esta parte del servidor está formada por diferentes elementos como los documentos HTML, scripts, recursos y algunos documentos adicionales que son ejecutados por el servidor web cuando el navegador del cliente solicita su ejecución.

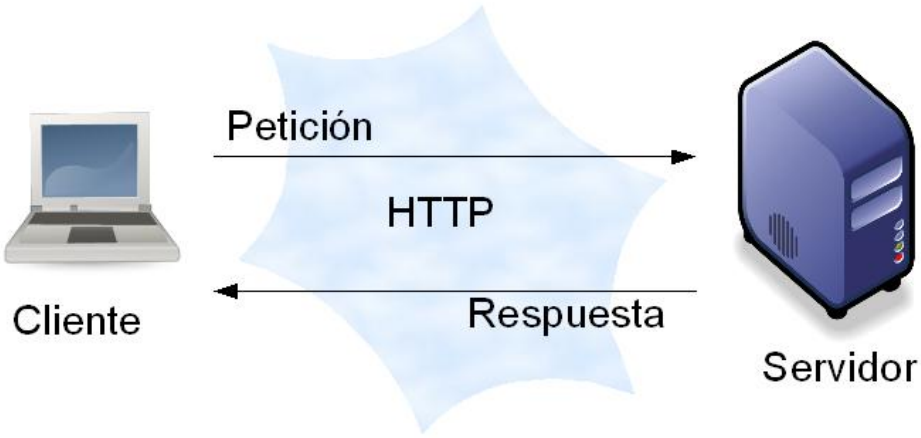

Figura 3.3. Esquema de petición y respuesta mediante HTTP entre cliente y servidor.

En resumen, estas aplicaciones basadas en HTML se ejecutan dentro del navegador del teléfono. Por ejemplo, en el sistema operativo iOS, son ejecutadas en el navegador Safari.

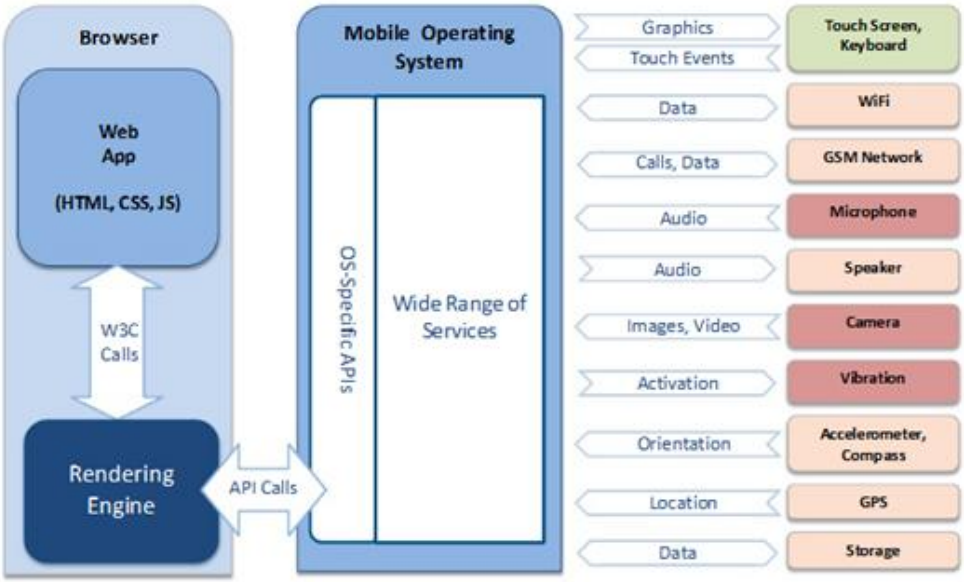

Figura 3.4. Esquema de funcionamiento de una aplicación web.

3.2.2. Ventajas y desventajas de las aplicaciones web.

Ventajas de las aplicaciones web.

- **Fácil diseño**, no es necesario pensar en el diseño de una aplicación móvil, simplemente diseñar para dispositivos con una pantalla más pequeña.

- Requiere **menos complejidad de programación**.

- Son **multidispositivo y multiplataforma**, es decir, funcionan en cualquier dispositivo o sistema operativo, siempre que se disponga de conexión a internet.

- **No ocupa espacio** en el disco duro.

- **Actualizaciones inmediatas**, cuando nos conectamos estamos usando siempre la última versión.

- Los navegadores cada vez ofrecen más y mejores funcionalidades para crear aplicaciones web.

- **Los virus no dañan los datos** ya que éstos están guardados en el servidor de la aplicación, de modo que tampoco se dañan otras aplicaciones.

- Pueden ser **publicadas sin la aprobación** de ningún fabricante.

Desventajas de las aplicaciones web.

- Es **necesaria una conexión a internet** para su ejecución, de modo que si no disponemos o se interrumpe dicha conexión no se puede utilizar.

- **No es posible** publicarlas en los **market places**.

- El acceso a los **recursos del móvil es limitado**, ofrecen menos funcionalidades que las aplicaciones nativas

- El usuario debe recordar la dirección e introducirla en el navegador, haciendo **más difícil acceder a ella.**

- Estas aplicaciones son **más lentas**, ejecutar los HTML e interpretar los JavaScripts es más costoso.

- Solo se encuentra **disponible la última versión**, el usuario no tiene la libertad de elegir la versión que prefiera.

3.3. Aplicaciones híbridas.

3.3.1. ¿Qué es una aplicación híbrida?

Estas aplicaciones son una combinación de los dos tipos de aplicaciones anteriores: aplicaciones nativas y aplicaciones web.

El concepto de aplicación híbrida consiste en una página basada en HTML mediante las herramientas estándar HTML5, CSS3 y JavaScript que posteriormente es envuelta en código nativo, de modo que el paquete resultante puede ser distribuido, al igual que las aplicaciones nativas, a través de los canales de compra de aplicaciones.

Al ser una combinación de dos tipos de aplicaciones se consigue reunir lo mejor de ambos modelos. Como se ha mencionado antes permite el uso de tecnologías multiplataforma como HTML, JavaScript y CSS pero a su vez también permite acceder a una gran parte de las utilidades y sensores del dispositivo. La mayor parte de la infraestructura es tipo web y la comunicación con las herramientas del terminal se lleva a cabo mediante comunicadores como PhoneGap. Estas aplicaciones basadas en web, se ejecutan en el dispositivo y utilizan el motor de navegación del dispositivo pero sin utilizar el navegador en sí mismo.

El proceso de desarrollo para las aplicaciones híbridas es más complicado que para las aplicaciones web. Del mismo modo que para el desarrollo de aplicaciones web, se crean archivo HTML, CSS y JavaScript a ejecutar en un navegador. Pero también al igual que para las aplicaciones nativas, el código generado se compila en un ejecutable. Ambos códigos son compilados y forman un paquete que será compartido mediante los market places.

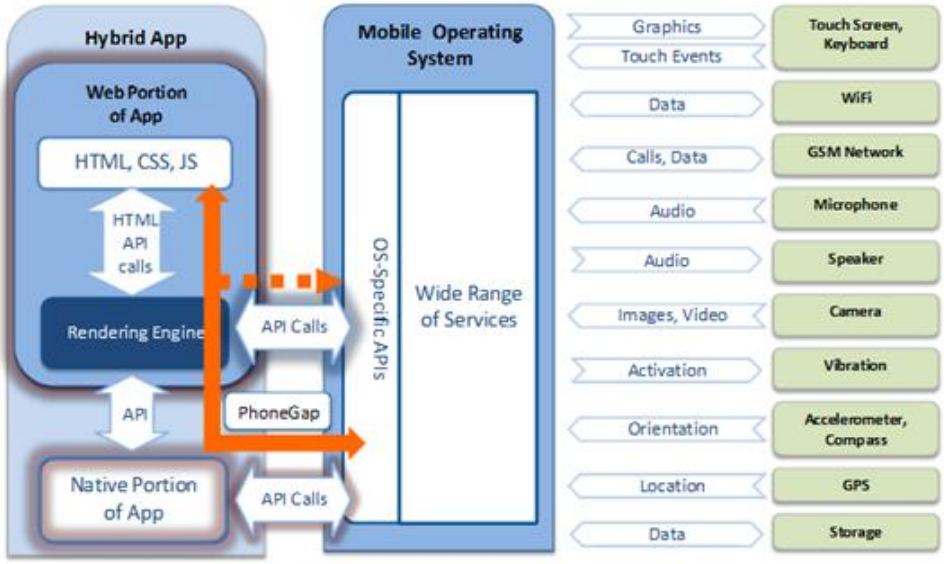

Figura 3.5. Esquema de funcionamiento de las aplicaciones híbridas.

Con este tipo de aplicaciones no existe la necesidad de utilizar un lenguaje específico, como Objetive C o Java, para construir cada aplicación completa. Solo es indispensable que la plataforma contenedor este construida mediante una tecnología específica, lo restante puede ser construido con un lenguaje menos complejo ya mencionado, HTML5.

Una de las aplicaciones híbridas más común es Facebook. Esta aplicación se descarga de los canales de compra de aplicaciones específicos de los dispositivos y tiene características propias de una aplicación nativa.

3.3.2- Ventajas y desventajas de las aplicaciones híbridas.

Ventajas de las aplicaciones híbridas.

- La principal ventaja es que **son multiplataforma**, permiten que el código fuente se pueda ejecutar en diferentes plataformas.

- Estas aplicaciones permiten **acceder a todos los recursos del móvil**, como pueden ser la cámara o el sistema GPS.

- **No es necesaria conexión a internet** para ejecutar la aplicación a excepción de partes concretas de la aplicación que requieran dicha conexión.

- Aúnan lo mejor de los dos tipos de aplicaciones, de modo que se tratan de una **buena alternativa** cuando existe duda en qué tipo de aplicación realizar.

- **No** es necesario utilizar un **lenguaje específico** para crear la aplicación íntegra, solo es necesario para la plataforma contenedor. La mayor parte utiliza HTML5, un leguaje más sencillo y por lo tanto el desarrollo de estas aplicaciones no supone tanto tiempo.

- Es **posible subir** estas aplicaciones **a los market place**, lo cual permite al usuario encontrarlas de una forma más rápida y por ello facilita la distribución de éstas.

- El modelo híbrido permite que las aplicaciones sean **instaladas en el dispositivo** y que el usuario disponga de un acceso directo, facilitando el acceso a ellas.

Desventajas de las aplicaciones híbridas.

- Las **actualizaciones son manuales**, el usuario debe acceder al canal de compra de aplicaciones de las distintas plataformas para conseguir las actualizaciones de la aplicación.

- Al igual que en el caso de las aplicaciones nativas para poder distribuir las híbridas en los market place es necesario pasar unos **procesos de validación** que en ocasiones pueden resultar duros.

- La **parte nativa** que envuelve el código fuente basado en HTML, si debe ser programado **con un lenguaje especifico** para las distintas plataformas.

3.4. Comparación entre las aplicaciones nativas, las aplicaciones web y las aplicaciones híbridas.

Una vez explicados los tres tipos de aplicaciones móviles existentes junto con sus ventajas e inconvenientes, nos enfrentamos al gran dilema: ¿qué tipo de aplicación es mejor?

A través de la red podemos observar gran variedad de opiniones, por un lado se encuentran los grandes defensores de las aplicaciones web, que aseguran que HTML5 será el principal lenguaje en el futuro y dejará a un lado el desarrollo de las aplicaciones nativas. En una posición completamente contraria, se encuentran los detractores de éstas, aquellos que opinan que las aplicaciones nativas aprovechan más las características y recursos que ofrecen los dispositivos móviles. Frente a todas estas opiniones, también existen estudios que afirman que para 2016 más del 50% de las aplicaciones móviles usarán el modelo híbrido. Dejando a un lado todas estas opiniones, vamos a llevar a cabo una comparativa de los tres tipos de aplicaciones.

La principal ventaja de las aplicaciones web es la posibilidad de poder ser ejecutadas en cualquier plataforma, simplemente teniendo en cuenta la compatibilidad con el navegador. Mientras una aplicación web funcionará en la gran mayoría de *smartphones*, una aplicación nativa precisa de un desarrollo específico para cada sistema operativo y en ocasiones incluso para las versiones de éste. Y en el caso de las híbridas también será necesario un lenguaje específico para el desarrollo del "contenedor" nativo que envuelve la parte web de la aplicación.

Por otro lado, el hecho de que tanto las aplicaciones web como las híbridas basen su funcionamiento en tecnología web, conlleva que precisen de una

conexión con los servidores para poder ofrecer los servicios de forma rápida y eficaz. Este punto de la conectividad podemos considerarlo negativo ya que el usuario no podrá hacer uso de las aplicaciones cuando no disponga conexión a internet. Aunque también tiene su parte positiva, ya que no ocupa memoria en el dispositivo y no requiere de un gran procesador para ejecutarlas, todo se lleva a cabo a través del navegador.

Otro punto interesante a analizar es la posibilidad de que las aplicaciones accedan a las funcionalidades que nos ofrecen los móviles. En el caso de la programación basada en web no existe una completa compatibilidad con las oportunidades que nos brindan los móviles de hoy en día, como la cámara, el acelerómetro o el sistema GPS. Si bien es cierto que existe la posibilidad de integrar estas APIs nativas en las aplicaciones basadas en web, también cabe destacar la complejidad que esto conlleva. En ocasiones, es necesario el desarrollo de un framework para acceder a ellas, lo cual complica la programación y puede llegar a ser tan complejo y costoso como desarrollar un código nativo.

Figura 3.6. Características de los distintos tipos de aplicaciones.

Desde el punto de vista de la programación, tanto en las aplicaciones web como en las híbridas no es necesario repetir la mayor parte del desarrollo para cada sistema operativo. Sin embargo si se decide crear aplicaciones nativas compatibles con los distintos sistemas operativos, será necesario realizar el desarrollo para cada uno. También cabe destacar que una vez desarrollada la aplicación no se ha acabado el trabajo, también conlleva un mantenimiento, por lo que todo ésto, aumenta tanto el coste como el tiempo de desarrollo en comparación con los otros dos tipos de aplicaciones.

Otro de los factores importantes a considerar es la consistencia de la aplicación con la interfaz de usuario de cada sistema operativo. Realizar una aplicación compatible con todos los sistemas operativos puede llevarnos a una pérdida en la personalización característica de la interfaz nativa. Es decir, tanto Android

como iOs disponen de un estilo propio de la interfaz nativa, por los que si se crea una aplicación única que pueda ejecutarse en ambos sistemas, se romperá con alguno de los dos estilos propios.

Figura 3.7. Interfaz gráfica de tres sistemas operativos: Android, iOS y Windows Phone.

Otro aspecto a considerar es la forma de distribución de las distintas aplicaciones. En este aspecto las aplicaciones web salen perjudicadas, no es posible subirla en los canales de compra de aplicaciones específicos de cada plataforma. El usuario debe conocer la dirección de la aplicación para poder acceder a ella, por lo que la empresa deberá hacer mayor esfuerzo de marketing para dar a conocer la aplicación.

Las aplicaciones híbridas pueden ser distribuidas mediante estos canales pero algunas no pasan el proceso de aprobación llevado a cabo por dichos canales, dependiendo de la parte que haya sido desarrollada con lenguaje web entre otras cosas. En un principio las aplicaciones nativas si pueden ser subidas en estos market places, pero también deberán pasar unos procesos de evaluación.

Tras realizar la comparativa entre los tres tipos de aplicaciones, hemos observado que los tres presentan tanto ventajas como desventajas en los distintos aspectos estudiados.

En conclusión, no existe una respuesta rotunda a la pregunta que nos hemos formulado al principio (¿Qué aplicación es mejor?). Deberemos elegir el tipo de aplicación dependiendo de nuestras necesidades.

3.5. ¿Por qué una aplicación híbrida?

En el apartado anterior se llegó a la conclusión de que no existe un tipo de aplicación absoluto que sea mejor, a la hora de elegir el tipo de aplicación que vamos a desarrollar debemos tener en cuenta las necesidades que ésta debe cubrir.

Como en este caso la función de la aplicación es facilitar el día a día de las personas que sufren celiaquía, se quiere que todos los usuarios la puedan utilizar independientemente del terminal que dispongan. Esto es posible gracias a la naturaleza multiplataforma y multidispositivo del lenguaje basado en web utilizado para desarrollar el código fuente de la aplicación.

De este modo para realizar estas aplicaciones no es necesario conocer el lenguaje específico de la plataforma. La aplicación se desarrolla con un lenguaje más sencillo, utilizando HTML5, CSS3 para darle personalizar el estilo y JavaScript que nos permite incluir mapas. De forma que llevará menos tiempo el desarrollo de la aplicación y por ello supone menos costes.

Como hemos visto anteriormente, una de las ventajas de las aplicaciones híbridas es la posibilidad de que sean instaladas en el dispositivo y disponer de un acceso directo en él. En este caso resultará muy útil a los usuarios ya que seguramente utilicen la aplicación en varias ocasiones y para ello simplemente tendrán que pulsar el botón de acceso directo, sin necesidad de memorizar la dirección e introducirla en el navegador.

En la aplicación será importante conseguir la ubicación del usuario y mediante JavaScript será posible, lo cual dotará de mayor funcionalidad a la aplicación.

Por otro lado estas aplicaciones permiten al usuario utilizar la aplicación sin necesidad de una conexión a Internet, a excepción de aquellas funciones específicas que requieran Internet.

Si nos interesara distribuir la aplicación mediante las tiendas de aplicaciones de las diferentes plataformas, con el modelo híbrido sería posible, aunque es necesario pasar un proceso de evaluación y que puede llegar a ser muy estricto.

4. HTML5.

HTML es el acrónimo en inglés de HyperText Markup Language, traducido en español como lenguaje de marcado de hipertexto. HTML5 es la quinta revisión del lenguaje HTML, de desarrollo de páginas web. Pero no se trata simplemente de una nueva versión, sino que es el resultado de agrupar diversas especificaciones relacionadas con las tecnologías web:

- HTML4, el estándar en uso desde 1997.
- XHTML1, una adaptación de HTML al lenguaje XML.
- DOM nivel 2, Document Object Model (Modelo de objetos del documento). Se trata de una interfaz de programación de aplicaciones (API), define la estructura lógica de los documentos y permite a los programas cambiar y actualizar dinámicamente los contenidos, la estructura y aspecto de la página.
- CSS nivel 2 (Posteriormente se profundizará en CSS).

Mediante esta versión se intenta solucionar aquellos problemas presentes en versiones anteriores y añadir novedades que se adapten a las necesidades actuales de los desarrolladores. HTML 5 también pretende proporcionar una plataforma mediante la cual desarrollar aplicaciones web cuya ejecución en un navegador no conlleve falta de recursos.

4.1. Mejoras HTML 5.

La aparición de HTML5 ha supuesto las siguientes mejoras:

- *Estructura del cuerpo.* La estructura de la gran mayoría de las páginas web es la misma, contienen una cabecera y un pie, una barra de navegadores, etc. HTML5 tiene en cuenta todas estas partes comunes y crea etiquetas para definirlas de modo que los códigos están mejor estructurados.

- *Etiquetas para contenido específico.* Al igual que explicaba el punto anterior, se crean nuevas etiquetas, pero en este caso se tratan de etiquetas específicas para incorporar contenido multimedia. Es decir una etiqueta para cada tipo de contenido como audio, video, etc.

- *Sintaxis***.** La sintaxis de HTML5 es compatible con HTML4 y XHTML1. Con HTML5 se puede usar sintaxis de XML, también sintaxis de MathML, que es un lenguaje utilizado para describir notaciones matemáticas.

- *APIs.* Se añaden nuevas APIs*,* Interfaz de Programación de Aplicaciones, consiste en un conjunto de funciones y aplicaciones que ofrece una biblioteca para ser usados en otro software. Del mismo modo también se cambian o quedan obsoletas algunas de las ya existentes.

- *Canvas.* Se trata de un nuevo componente que mediante el uso de las funciones de una API, permite dibujar en la página todo tipo de formas, que pueden estar animadas y responder a las interacciones del usuario.

- *Bases de datos locales.* El navegador permite el uso de una base de datos local, utilizada en el modelo cliente/servidor.

- *Aplicaciones web Offline.* Mediante una API es posible construir aplicaciones que funcionen en local y sin requerir una conexión a Internet.

- *Geolocalización.* Se puede localizar geográficamente por medio de un API que lo permita.

- *Drag & Drop* (arrastrar y soltar). Permiten arrastrar los elementos de una página mediante el uso de una API.

- *Eliminación de etiquetas de presentación.* Desaparecen las etiquetas utilizadas para modificar los estilos de la página, ahora esta función solo la realiza CSS.

- *Mejoras en los formularios.* Se añaden mejoras en los formularos como validaciones sin necesidad de utilizar JavaScript obligatoriamente aunque cabe decir que son un poco pobres.

4.2.- Estructura de una página HTML5.

Como se ha explicado en las mejoras, una de las novedades de esta quinta versión es la nueva estructura de las páginas. Se introducen una serie de etiquetas nuevas con cierto sentido semántico, es decir, la etiqueta describe el significado de su contenido.

En la mayoría de las páginas se puede distinguir ciertas partes comunes: una cabecera que suele estar en la parte alta de las webs, un menú o barra de navegación y también suelen tener un pie de página.

En esta versión se han creado etiquetas para definir y describir estas secciones que aparecen comúnmente en la mayoría de las páginas y otras nuevas secciones de las web. De este modo por ejemplo la cabecera será definida con una etiqueta concreta, facilitando a los navegadores el reconocimiento de las partes una web.

Se pretende eliminar el constante uso de las etiquetas * * ya que esto suponía una confusión para los desarrolladores. Con esta nueva versión solo se utiliza la etiqueta * * cuando no existe otra etiqueta más idónea para usar.

Es posible utilizar estas nuevas etiquetas que ofrece HTML5 junto con las del estándar HTML4, exceptuando aquellas que han sido eliminadas como por ejemplo *<font>* o * *, como se verá más adelante, estas funciones las realizará CSS.

A continuación serán explicadas las etiquetas más importantes que ofrece HTML5:

- <header>

La etiqueta <header> </header> describe la cabecera visible del sitio, dentro de esta etiqueta generalmente se encuentra el logotipo, botones de menú y encabezados del sitio.

- <footer>

Con la etiqueta <footer> </footer> nos referimos al pie de página del diseño aunque también puede referirse al de una sección concreta. Tanto esta etuqieta como la de cabecera (header) pueden ser utilizadas varias veces en un mismo diseño.

- <hgroup>

Mediante este elemento se agrupan varias etiquetas de cabeceras (<h1> <h2> <h3> <h4> <h5> <h6>) cuando se van a utilizar títulos, subtítulos. Así se evita romper con el esquema de la página.

- <nav>

Diseñada para colocar los menús de navegación, tanto de la página principal como para un artículo en especial.

- <section>

Define una sección genérica de la página, es decir, una agrupación temática de contenidos, generalmente con una cabecera. Esta etiqueta es equivalente a un DIV con ID= “contenido”.

.

- <article>

Representa una sección de contenido independiente. Esta etiqueta se utiliza para respuestas en un foro, artículos en un blog o comentarios de un artículo, es decir, para cualquier artículo independiente del contenido.

- <aside>

Define una parte de la página que no está directamente relacionada con el contenido de la página. Normalmente se utiliza para representar las barras laterales de los blogs.

- <embed>

Esta etiqueta se utiliza para insertar contenidos que precisan plugins, como por ejemplo Flash. Esta etiqueta es compatible con todos los navegadores.

- <canvas>

Como ya se ha comentado antes, es un elemento complejo que permite dibujar dinámicamente imágenes en una página web, muy utilizado en Google Maps.

En la siguiente figura se puede observar la estructura de las páginas web que ofrece esta nueva versión de HTML mediante el uso de las etiquetas explicadas anteriormente.

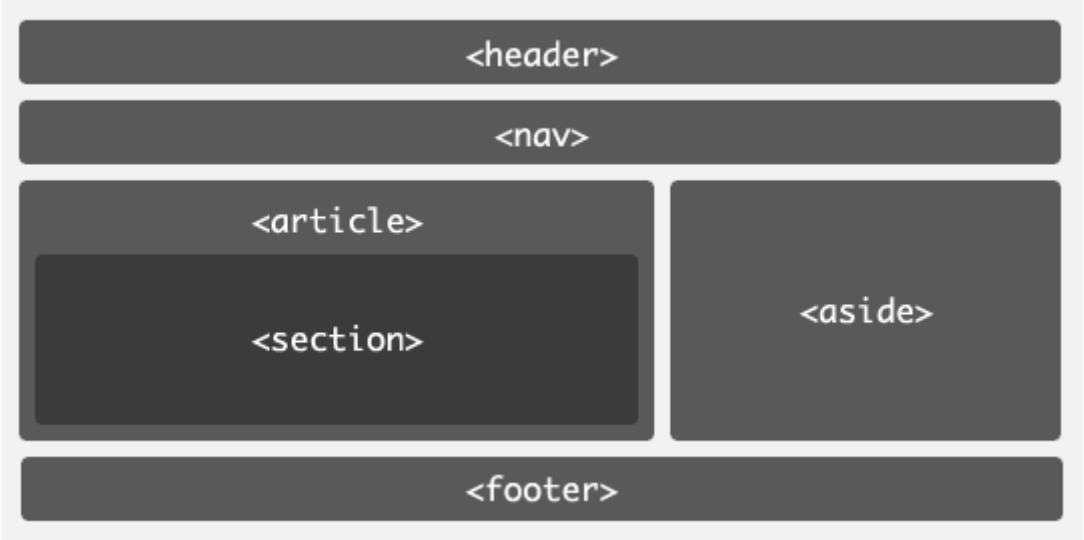

Figura 4.1. Estructura básica de un documento HTML5.

A continuación se muestra un ejemplo de código en HTML5 mostrando la estructura básica:

```
<!DOCTYPE html>

<html lang="es">

<head>
<title>Titulo de la web</title>
<meta charset="utf-8" />
<link rel="stylesheet" href="estilos.css" />
<link rel="shortcut icon" href="/favicon.ico" />
<link rel="alternate" type="application/rss+xml"/>
</head>

<body>
    <header>
       <h1>Mi sitio web</h1>
       <p>Mi sitio web creado en html5</p>
    </header>
    <section>
       <article>
           <h2>Titilo de contenido<h2>
           <p> Contenido (ademas de imagenes, citas, videos etc.) </p>
       </article>
    </section>
    <aside>
       <h3>Titulo de contenido</h3>
           <p>contenido</p>
    </aside>
    <footer>
        Creado por mi
    </footer>
</body>
</html>
```

Figura 4.2. Ejemplo de un código básico de HTML 5.

5. CSS3

5.1. ¿Qué es CSS3?

Las hojas de estilo en cascada (Cascading Style Sheets o CSS) son las que nos permiten definir las reglas para controlar el aspecto y la presentación en diferentes dispositivos, ya sean pantallas de equipos de escritorio, portátiles, móviles, impresoras o cualquier dispositivo capaz de mostrar contenido web. Es decir, CSS es un lenguaje para definir el estilo o la apariencia de las páginas web, escritas con HTML y XHTML.

Fue creado para separar los contenidos y su presentación, y es imprescindible para crear páginas web complejas. La posibilidad de separar estas dos partes proporciona numerosas ventajas, debido a que obliga a crear documentos HTML/XHTML bien definidos y con significado completo. También mejora la accesibilidad del documento, facilita su mantenimiento y hace posible que un único documento se visualice en varios dispositivos.

Cuando se desarrolla una página web, en un primer lugar se crean los contenidos mediante el lenguaje HTML, explicado en el punto anterior, para crear los contenidos, es decir, se diseña la función que lleva cada elemento de la página: párrafo, titular, texto destacado, tabla, menú...

Después de crear el contenido de cada elemento, mediante el lenguaje CSS se define el diseño. Las hojas de estilo permiten modificar el color, el tipo de letra, la posición de los elementos, separaciones entre elementos y otras muchas posibilidades para personalizar cada elemento.

5.2. Evolución CSS3.

Antes de la adopción de CSS, los desarrolladores tenían que definir el estilo de los elementos dentro de las etiquetas HTML de la página. En el siguiente código ejemplo se puede observar esta forma de proporcionar estilo a la página sin la utilización de hojas de estilo CSS.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
<title>Ejemplo de estilos sin CSS</title>
</head>
<body>
<h1><font color="red" face="Arial" size="5">Titular de la página</font></h1>
<p><font color="gray" face="Verdana" size="2">Un párrafo de texto
largo.</font></p>
</body>
</html>
```

Figura 5.1. Ejemplo de página HTML con estilos definidos sin CSS.

Definir el estilo de las páginas sin el uso de CSS conlleva un gran problema, se debe insertar tantas etiquetas que definan los elementos, como elementos contenga la página. Es decir, si la página contiene un número elevado de elementos habrá que definir muchas etiquetas. Además los diseños no suelen ser fijos, es frecuente actualizar el diseño cada cierto tiempo y con esta forma es necesario modificar muchas etiquetas.

Así la aparición de CSS supuso lo siguiente:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
<title>Ejemplo de estilos con CSS</title>
<style type="text/css">
h1 { color: red; font-family: Arial; font-size: large; }
p { color: gray; font-family: Verdana; font-size: medium; }
</style>
</head>
<body>
<h1>Titular de la página</h1>
<p>Un párrafo de texto no muy largo.</p>
</body>
</html>
```

Figura 5.2. Ejemplo de página HTML con estilos definidos mediante CSS.

Mediante esta forma de definir estilos no importa el número de elementos que contenga la página. Todos aquellos elementos que compartan etiqueta, tendrán el mismo diseño. Es decir, no importa el número elementos <h1> que existan en la página, usando CSS todos tendrán el mismo aspecto.

EL auge de las hojas de estilo surgió cuando Internet comenzaba a cobrar mucha importancia y el lenguaje HTML empezaba a tomar protagonismo. Resultaba difícil desarrollar páginas con el mismo aspecto en los diferentes navegadores, debido a la guerra entre navegadores y la ausencia de un estándar para crear estilos.

La primera especificación oficial de CSS fue CSS1 en 1996, que significó un avance considerable a la hora de diseñar páginas web, proporcionando mayor

control de los elementos de la página. Mediante este lenguaje se conseguían los primeros estilos, era posible modificar propiedades como el tipo letra, los colores de texto, alineación de texto, imágenes de fondo, bordes y relleno.

Pero aún quedaron muchas otras cosas que los diseñadores deseaban hacer y que CSS no permitía. Por ello en 1998, CSS2 incorporó algunas novedades, como el posicionamiento absoluto y relativo de los elementos fijos, los tipos de medios y el z-index. Y más allá de CSS2.1 creado para corregir algunos errores y eliminar funcionalidades que no eran soportadas por algunos navegadores. A partir del año 2005 se comenzó a definir CSS3, aportando todavía más control sobre los elementos de la página.

Esta última versión proporciona una gran variedad de opciones para satisfacer las necesidades actuales de los diseñadores web. Funciones de sombreado, de movimiento y transformación entre otras. De este modo CSS3 se ha convertido en el estándar para crear estilo a los documentos.

5.3. Mejoras CSS3.

Como ya se ha explicado, esta última versión supuso la aparición de muchas opciones para dar estilo. A continuación se van a analizar algunas mejoras incluidas en CSS:

- *Bordes.* Incluye nuevas propiedades para trabajar con los bordes, pudiendo modificar el tamaño de éstos, la curvatura e incluso añadir sombras. También se dispone de una mayor gama de colores.

- *Fondos.* Hasta ahora cuando se insertaba una imagen para decorar un fondo, la imagen se repetía rellenando todo el fondo. Con esta versión ya podemos controlar cuantos píxeles ocupa una imagen evitando que se repita. También es posible superponer distintos fondos mediante capas.

- *Color.* Se amplía la gama de colores y además se permite la modificación de la opacidad de los elementos.

- *Fuentes.* Permite utilizar fuentes externas e incluir sombras.

- *Degradados.* Esta nueva versión incluye nuevas propiedades para crear distintos tipo de degradados: lineales, radiales, lineales de repetición y radiales de repetición.

- *Otras mejoras.* Se incluye el uso de Media Queries, utilizadas para desarrollar diseños adaptables a los tamaños de los distintos dispositivos. También es posible modificar el tamaño de la interfaz, la creación de

múltiples columnas de texto, animaciones CSS3 y propiedades orientadas a la lectura automática de páginas web.

6. Estudio y elección de los frameworks para desarrollar aplicaciones híbridas.

Debido a la gran expansión de los dispositivos móviles, las aplicaciones han cobrado mucha importancia, bien sean juegos, aplicaciones útiles en la vida cotidiana o para empresas.

Esto ha conllevado a la creación de una gran variedad de plataformas como Android, iOS, Blackberry, Windows Phone, etc. Esta variedad de plataformas obliga al desarrollador a aprender distintos lenguajes y frameworks de desarrollo.

Además para crear una aplicación que se pueda ejecutar en varios dispositivos se requiere un desarrollo distinto para cada plataforma, aumentando el tiempo de desarrollo considerablemente.

Para solventar este problema surgen herramientas y frameworks de desarrollo multiplataforma capaces de empaquetar aplicaciones web y convertirlas en híbridas o nativas, de modo que con un único código o un código base sea posible desarrollar una aplicación compatible con varias plataformas.

Antes de profundizar en el estudio es importante definir framework. Un framework consiste en una estructura software formada por componentes personalizables para la creación de una aplicación, que permite reutilizar código y acelera el proceso de desarrollo.

A continuación se explican dos frameworks capaces de realizar este empaquetado, PhoneGap y Titanium Appcelerator. Como veremos el primero de ellos crea aplicaciones híbridas y el segundo tanto nativas como híbridas.

6.1. PhoneGap.

PhoneGap es un framework de desarrollo de aplicaciones para dispositivos móviles multiplataforma. Permite crear aplicaciones con aspecto nativo que se pueden desarrollar en diferentes plataformas móviles: iOS, Android, Blackberry, Windows Phone, Web Os, Symbiam y Bada.

La gran ventaja de este framework es la posibilidad de desarrollar estas aplicaciones multiplataforma con un único código base utilizando las tecnologías web: HTML5, CSS3 y JavaScript.

Phonegap trata de solventar el problema de decidir la plataforma con la que trabajar cuando se comienza a crear una aplicación móvil. Que además también supone elegir el lenguaje de programación, ya que cada plataforma utiliza un lenguaje.

En un principio fue desarrollado por Nitobi y su uso era gratuito. En 2011 Adobe anunciaba la adquisición de Nitobi, por lo tanto PhoneGap pasaba al control de Adobe. De modo que ha sido integrado en las últimas versiones de Dreamweaver. Se armó un gran revuelo por el temor de que se abandonara la gratuidad, pero el código fue entregado a la Fundación Apache y así PhoneGap continuaba siendo un software libre.

Actualmente este proyecto en la Fundación Apache recibe el nombre de “Apache Cordova” aunque se sigue manteniendo PhoneGap como una especie de marca comercial y por ello se sigue conociendo al framework como PhoneGap.

Dependiendo de la plataforma se usa un sistema distinto. En el caso de Android se utiliza Eclipse, disponible en Windows, Mac y Linux y una plantilla específica proporcionada por PhoneGap. Para iOS se usa Xcode, que solo está disponible en Mac y otra plantilla específica. Para el desarrollo en la plataforma de Blackberry no hay un entorno específico, se usa Java SDK y Blackberry SDK. También es necesario instalar los SDKs de cada plataforma. Siendo SDK el conjunto de herramientas, librerías y compiladores que permiten desarrollar aplicaciones en un sistema concreto.

PhoneGap no se trata de un IDE, es decir, un entorno de programación como Eclipse o Xcode, no dispone de constructor de interfaz gráfica, ni de editor de código, tampoco de simulador.

Es posible encontrarlo en formato de plugin para utilizarlo en varios programas, como por ejemplo para Eclipse o como plantilla para Xcode en Mac. Y es compatible con frameworks de desarrollo web móvil como por ejemplo JQuery Mobile, Sencha Touch, jQToucho y muchos más.

Phonegap permite convertir casi todos los códigos basados en tecnologías web (HTML, CSS y JavaScript) en código con apariencia de nativo, preparado para compilar en el SDK. Evitando al programador el largo proceso de aprendizaje de lenguajes específicos de programación y facilitando el desarrollo y mantenimiento de la aplicación.

Con apariencia nativa nos referimos a que en realidad no se trata de una aplicación nativa. Es decir, PhoneGap empaqueta las aplicaciones web dentro de una aplicación nativa**,** de cualquiera de las plataformas soportadas (Android, iOs, Blackberry, Windows Phone, Web Os, Symbian), de modo que parece una aplicación nativa, pero no es así, se trata de una aplicación híbrida.

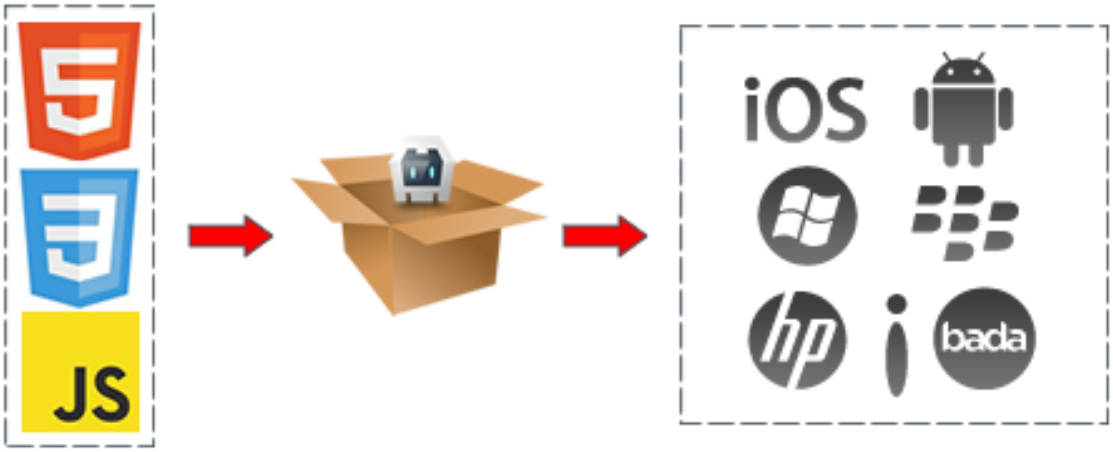

Figura 6.1. Esquema del empaquetamiento de las aplicaciones PhoneGap.

La interfaz de usuario con PhoneGap consiste en una vista de navegador web pero sin la barra de url. Ocupa el 100% del ancho y del alto del dispositivo. Se trata de la misma vista web que en el sistema nativo.

Figura 6.2. Interfaz de usuario con PhoneGap.

PhoneGap consiste en un conjunto de APIs (interfaz de programación de aplicaciones), basadas en JavaScript que permiten al desarrollador acceder a los elementos nativos del dispositivo, como la cámara, el acerelómetro, los contactos, el GPS, etc.

El uso de estas librerías se combina con un framework de desarrollo web móvil como puede ser jQuery Mobile, Dojo Mobile o Sencha Touch, de este modo es posible desarrollar aplicaciones que accedan a partes nativas del dispositivo utilizando únicamente HTML, CSS y JavaScript.

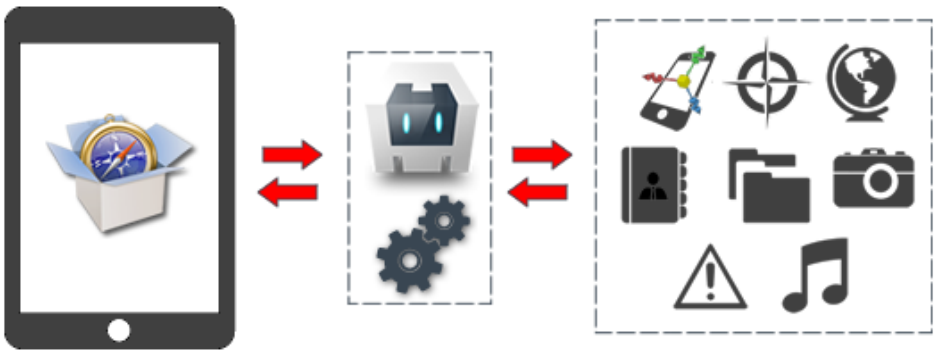

Figura 6.3. Esquema del uso de las APIs de PhoneGap para acceder a elementos del dispositivo.

	iOS iPhone / iPhone 3G	iOS iPhone 3GS and newer	Android	OS 4.6-4.7	OS 5.x	OS 6.0+	WebOS	Symbian	Bada
ACCELEROMETER	✓	✓	✓	✗	✓	✓	✓	✓	✓
CAMERA	✓	✓	✓	✗	✓	✓	✓	✓	✓
COMPASS	✗	✓	✓	✗	✗	✗	✗	✗	✓
CONTACTS	✓	✓	✓	✗	✓	✓	✗	✓	✓
FILE	✓	✓	✓	✗	✓	✓	✗	✗	✗
GEOLOCATION	✓	✓	✓	✓	✓	✓	✓	✓	✓
MEDIA	✓	✓	✓	✗	✗	✗	✗	✗	✗
NETWORK	✓	✓	✓	✓	✓	✓	✓	✓	✓
NOTIFICATION (ALERT)	✓	✓	✓	✓	✓	✓	✓	✓	✓
NOTIFICATION (SOUND)	✓	✓	✓	✓	✓	✓	✓	✓	✓
NOTIFICATION (VIBRATION)	✓	✓	✓	✓	✓	✓	✓	✓	✓
STORAGE	✓	✓	✓	✗	⚠	✓	✓	✓	✗

Figura 6.4. APIs disponibles para cada dispositivo.

Estas APIs permiten desarrollar la aplicación sin utilizar ningún lenguaje específico de cada plataforma como Java o Objetive C. Se crean con tecnologías web alojadas en la propia aplicación a nivel local en la mayoría de los casos.

También cuenta con una serie de librerías desarrolladas en código nativo, es decir, en el código específico de cada plataforma. Éstas actúan como un "puente" entre JavaScript y cada una de las plataformas nativas. Lo cual reduce las limitaciones que podían existir y ayuda a los desarrolladores a llegar más lejos en sus diseños.

Entonces lo que se obtiene al descargar el paquete de la página de PhoneGap es un archivo comprimido que contiene una carpeta para cada sistema operativo.

En cada una de estas carpetas se encuentra una librería JavaScript y otra en código nativo específico de la plataforma.

Por ejemplo en el caso de Android, se encontrará una librería JavaScript para desarrollar las aplicaciones web que permite el acceso al hardware del dispositivo a través de las APIs basadas en los estándares de HTML5 de la W3C (organismo que construye estándares para la web).

Y una librería en Java para permitir el uso de algunas APIs que no era posible acceder mediante JavaScript y también actúa como “puente” entre el código JavaScript y las plataformas nativas, haciendo posible acceder a elementos del dispositivo como la cámara, los contactos, el acelerómetro, las redes, los eventos, etc.

6.1.1. Otra alternativa: PhoneGap Build.

Quizás lo que hemos visto hasta ahora puede resultar complicado. Es cierto que una vez realizado el proceso para una plataforma, el resto son parecidos. Pero hay que tener en cuenta algunas especificaciones concretas de cada plataforma, también es necesario obtener los SDKs de cada plataforma e instalarlos y conseguir los entornos de desarrollo. Todo esto puede llevar un tiempo además del espacio en la memoria que ocuparán todos los sistemas que tenemos que instalar en nuestro ordenador.

Por otro lado como hemos visto antes para desarrollar iOS es necesario disponer de un ordenador con su sistema operativo Apple. En el que caso de Windows Phone ocurre lo mismo solo es posible desarrollar estas aplicaciones en el sistema operativo de Windows.

Para intentar mejorar el proyecto y evitar estas dificultades Nitobi comenzó a pensar una alternativa, consistente en un compilador en la nube y cuando Adobe adquirió Nitobi fue mejorado. Este compilador en la nube se denomina PhoneGap Build, no forma parte del framework anterior pero como veremos es una herramienta muy útil.

En la propia página de PhoneGap lo explican de la siguiente forma: *"Simplemente sube tus archivos HTML, CSS y JavaScript a la nube del servicio Adobe PhoneGap Build nosotros hacemos el trabajo de compilación por ti".*

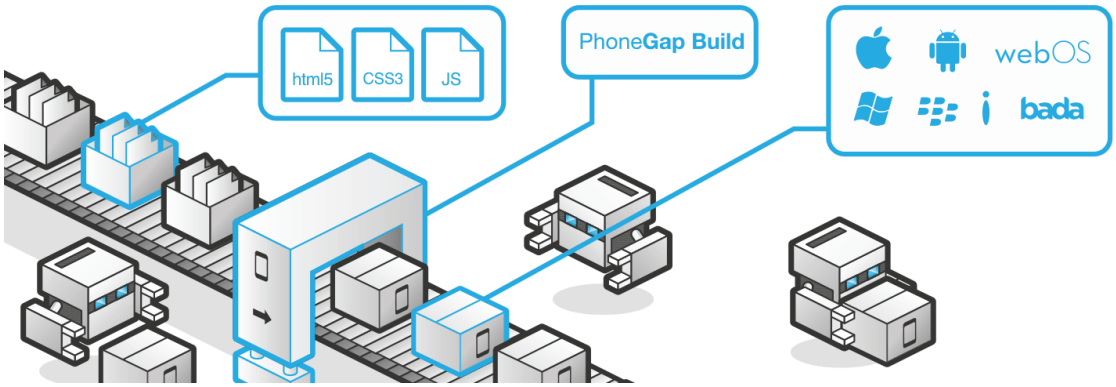

Figura 6.5. Esquema del proceso de compilación de Adobe PhoneGap Build.

De modo que con un único código fuente que se debe desarrollar mediante el uso de algún framework de desarrollo web móvil como jQuery Mobile o Sencha podemos obtener las aplicaciones compatibles con cada sistema.

Mientras que el framework PhoneGap permite desarrollar aplicaciones para sietes plataformas, esta herramienta es posible en seis plataformas iOS, Android, Windows Phone, Blackberry, webOS y Symbian. Esta compilación aún no está disponible para Bada.

Para poder obtener el paquete de instalación de iOS es imprescindible obtener un código de desarrollador que proporciona Apple, sin él no es posible compilar para esta plataforma.

Se puede utilizar esta herramienta de forma gratuita, aunque solo es posible compilar una aplicación. Pero pagando una cuota de 9,99 $ al mes, permite desarrollar 25 aplicaciones.

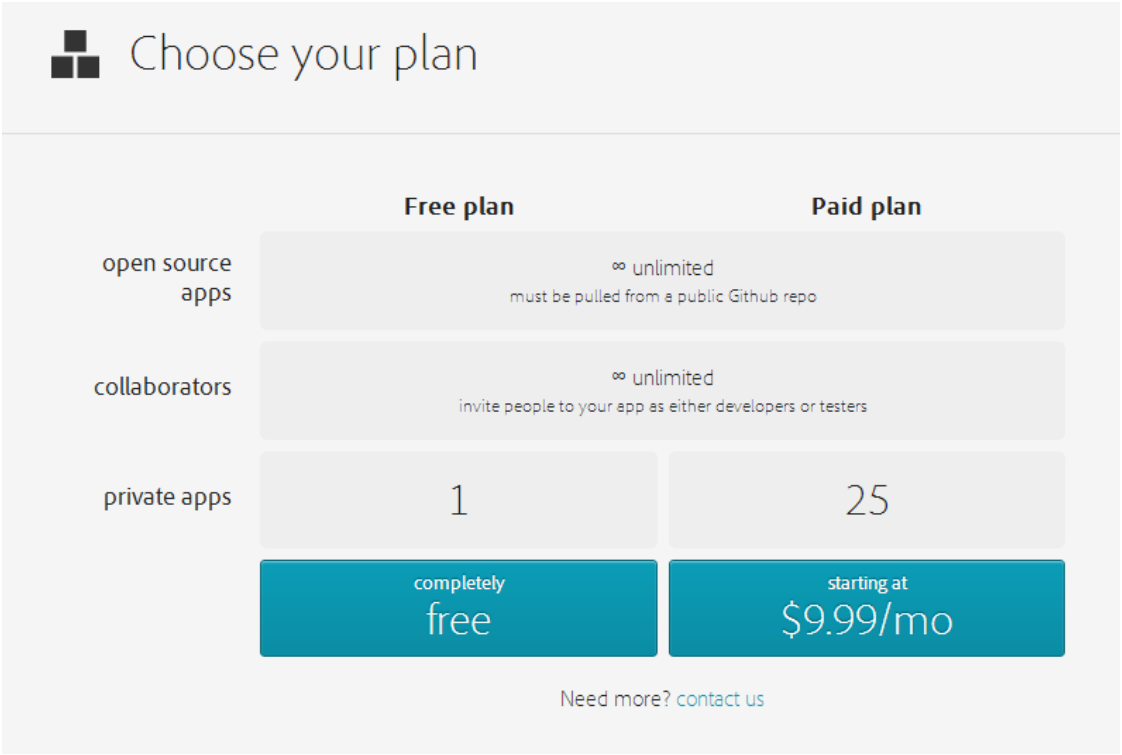

Figura 6.6. Dos planes posibles para comenzar a usar PhoneGap Build.

Como explicaban los propios creadores de PhoneGap simplemente es necesario subir la carpeta en la que estén guardados todos los archivos HTML 5, CSS 3 y JavaScript, eso sí debe tener extensión .zip y esperar a que esta herramienta haga el resto. Cabe destacar que dentro de esa carpeta se debe incluir un archivo index.html. También es posible personalizar el icono y nombre que obtendrá la aplicación cuando se instale en el dispositivo.

Cuando se termina el proceso, si se ha obtenido algún error se notifica y se especifica el error. Si no, simplemente tenemos que descargar los paquetes de instalación. Mediante una página de acceso público se puede compartir las aplicaciones y también ofrece un código QR facilitando el acceso a esos enlaces, de modo que es posible utilizar el lector QR del dispositivo para descargar el paquete de instalación y posteriormente instalarlo en el dispositivo.

Además el compilador en la nube se encargará de versionar las aplicaciones cada vez que realicemos un mantenimiento del código.

6.1.2. Ventajas y desventajas de PhoneGap.

Ventajas de PhoneGap.

- Es gratuito.

- Es multiplataforma ya que corre dentro de un navegador web. Compatible con varias plataformas iOS, Android, Blackberry, Windows Phone, webOS, Symbian y Bada.

- Es fácil de desarrollar y ofrece muchas posibilidades a aquellos que conocen bien los lenguajes basados en tecnologías web (HTML, CSS y JavaScript).

- Existe mucha documentación acerca de PhoneGap e inluso la propia web proporciona muchos ejemplos.

- La alternativa que ofrece mediante el compilador en la nube es una forma muy sencilla que permite mejorar muchas de las desventajas de PhoneGap.

Desventajas de PhoneGap.

- En algunos casos es necesario usar el sistema operativo de la plataforma. Por ejemplo empaquetar aplicaciones Windows Phone solo es posible con el sistema operativo Windows. Lo mismo ocurre con iOS, es necesario usar un Mac.

- Dependiendo de la plataforma se necesita un sistema diferente, para Android se requiere el uso de Eclipse y para iOS el uso de Xcode.

- Es necesario el uso de frameworks de desarrollo web móviles como por ejemplo jQuery Mobile, Sencha Touch. En sí se trata de una aplicación web por lo que el aspecto de la aplicación depende del framework web utilizado.

- Se desarrolla una aplicación híbrida, por lo que el rendimiento no es como el de una aplicación nativa.

- En caso de compilar para iOS es necesario introducir un código de desarrollador dado por la empresa Apple.

6.2. Titanium Appcelerator.

Se trata de una plataforma creada por la empresa Appcelerator en el año 2008 y mediante la cual se han sido desarrolladas más de 50.000 aplicaciones.

Permite desarrollar aplicaciones nativas para varios dispositivos móviles como Android, iOS, Blackberry y Windows Phone. También permite desarrollar aplicaciones híbridas basadas en tecnologías web como JavaScript, HTML, CSS y PHP, Ruby y Phyton.

Al igual que PhoneGap, mediante el uso de lenguaje web, proporciona una alternativa a los lenguajes nativos. Pero la diferencia es que con Titanium se puede generar una aplicación nativa y no un empaquetado que se ejecute en el navegador.

Para programar proporciona un IDE (entorno de desarrollo integrado) basado en eclipse, Titanium Studio, mediante el cual se puede crear proyectos y editar archivos JavaScript u otros recursos. Las aplicaciones nativas resultantes funcionan igual que si fueran programadas con el lenguaje propio de cada plataforma como Java en el caso de Android o Objetive C en el caso de iOS.

Titanium facilita el desarrollo de aplicaciones móviles multiplataforma con un único código base, con el 60% y 90% de código reutilizable entre plataformas. Permite este desarrollo de forma rápida, probar y subir las aplicaciones a los canales de compra de aplicaciones para poder distribuirlas.

Es posible instalar Titanium Studio en Windows, Mac y Linux, pero con algunas limitaciones. Las aplicaciones iOS solo es posible desarrollarlas en el sistema

operativo de Apple ya que las herramientas necesarias como Xcode solo es posible instalarlas en su propio sistema operativo. Lo mismo ocurre con Windows Phone solo es posible su desarrollo en el sistema operativo de Windows.

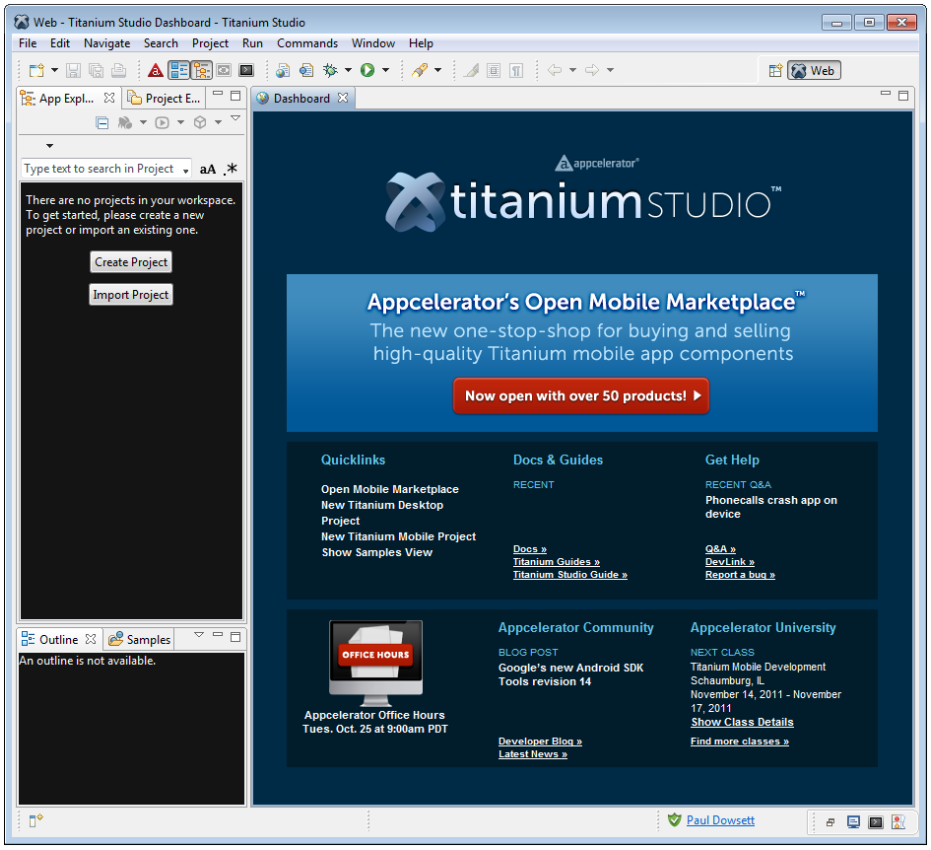

Figura 6.7. Imagen de Titanium Studio IDE.

Como se ha mencionado antes se mantiene un código base entre las distintas plataformas, simplemente es necesario añadir algunas llamadas a las APIs específicas de la plataforma para la que se diseña. Es decir, se desarrolla un código común al que posteriormente se le añaden funciones propias de la plataforma, de modo que se simplifica el desarrollo y el mantenimiento de aplicaciones móviles.

El desarrollo de la aplicación se hace íntegramente en JavaScript por lo que todos los controles tienen que ser creados manualmente y para ello cuenta con una librería JavaScript que permite acceder a los controles del sistema. Así todos los controles (botones, listas, menús) son nativos, lo que hace que sean más rápidos. Otra diferencia con PhoneGap es que no hay DOM, lo que hace que no sea posible utilizar librerías de jQuery, porque con Appcelerator se trabaja con JavaScript puro, no dentro de un documento HTML.

Para desarrollar iOS es necesario utilizar Xcode, un entorno de desarrollo oficial de Apple. Mediante Titanium Studio se genera un proyecto Xcode con el JavaScript transformado y todas las librerías utilizadas. Una vez creada la aplicación en Xcode es posible probarla en el simulador del propio Titanium Studio. El proyecto Xcode generado se puede abrir en Xcode para configurar algunos aspectos de la aplicación como el logo pero no se puede editar el JavaScript desde Xcode, para ello se debe utilizar Titanium Studio.

En el caso de la plataforma Android, para usar el simulador y empaquetar la aplicación solo es necesario el SDK de Android.

El lenguaje JavaScript utilizado para desarrollar la aplicación no se ejecuta en el navegador del dispositivo como ocurre con otros frameworks como jQuery o Sencha. El código se ejecuta en un motor JavaScript que tienen todos los dispositivos, en el caso de Android y Blackberry es Mozilla Rhino y en iOS JavaScriptCore.

De modo que el código JavaScript se traduce al lenguaje nativo que utiliza la plataforma para la que se está diseñando y posteriormente es compilado a código nativo. Así es posible diseñar aplicaciones más rapidas y eficientes que con otros frameworks de JavaScript.

Figura 6.8. Arquitectura de Titanium.

Quizás algo que ha sido desarrollado para iOS no funciona en Android o viceversa. Las librerías JavaScript son diferentes de modo que hay que informarse de lo que es posible hacer con cada plataforma.

Se ha observado anteriormente como PhoneGap contaba con una librería JavaScript para acceder a los elementos del dispositivo. Sin embargo Appcelerator precisa también de librerías para manejar los controles nativos y su distribución en la pantalla, lo que complica el desarrollo.

6.2.1. Ventajas y desventajas de Titanium Appcelerator.

Ventajas de Titanium Appcelerator.

- Desarrolla aplicaciones móviles multiplataforma (iOS, Android, Blackberry y Windows Phone). Y también de escritorio (Windows, Mac y Linux).

- El aspecto y los controles son nativos, por lo que se obtiene mejor rendimiento.

- Gratis, soporte de pago.

- Reutilización del 60-90% de código en varias plataformas.

- Reduce los costes del desarrollo.

- El proceso de desarrollo es más rápido que con lenguajes nativos.

- La comunidad está en constante crecimiento.

Desventajas de Titanium Appcelerator.

- Los componentes visuales y los controles se definen manualmente mediante JavaScript.

- Para empaquetar aplicaciones iOS es necesario usar un Mac con Xcode instalado.

- Las librerías JavaScript son diferentes dependiendo de la plataforma.

- Escasa compatibilidad entre los sistemas operativos.

6.3. Elección del framework de desarrollo.

Una vez estudiado ambos frameworks se va a explicar que opción se ha elegido para realizar el proyecto.

Ambos frameworks presentaban un problema de compatiblidad entre sistemas operativos, es decir, solo es posible desarrollar en iOS dentro del sistema operativo de Apple. También requieren instalar los SDK necesarios de la plataforma y programas específicos, aumentando el tiempo de desarrollo ya que es necesario aprender a utilizar varios programas.

En primer lugar se decidió descartar Titanium Appcelerator por el inconveniente de precisar instalar varios programas y porque los controles son nativos, lo cual ofrece mayor rendimiento pero hay que crearlos manualmente mediante JavaScript. También porque era más difícil encontrar información y ejemplos actualizados sobre este framework.

A diferencia de Titanium, PhoneGap permite el uso de un framework de desarrollo web móvil como jQuery Mobile para realizar el código principal y posteriormente compilarlo. Pero sigue siendo necesario instalar otros programas.

Todos estos problemas de instalar varios programas y especificaciones concretas para cada plataforma no están presentes en la alternativa que ofrece PhoneGap (PhoneGap Build). Esta opción de compilador en la nube resulta atractiva ya que simplemente hay que realizar una aplicación web y luego subirla para que el compilador realice el trabajo necesario para convertirla en híbrida.

PhoneGap Build, el compilador en la nube de PhoneGap, nos permite realizar aplicaciones para iOS, Android, Windows Phone, Blackberry, webOS y Symbian.

El inconveniente es que para iOS es necesario un código de desarrollador que proporciona la empresa Apple. Este inconveniente no es suficiente para descartar esta opción ya que en todo lo estudiado se encuentran dificultades a la hora de desarrollar aplicaciones para iOS. Entonces se ha pensado que como para crear la aplicación híbrida primero es necesario crear una aplicación web, para la plataforma iOS se ofrecerá la opción de la aplicación web y para el resto de plataformas la aplicación híbrida.

Así que finalmente se ha elegido utilizar PhoneGap Build, ya que resulta mucho más fácil que el resto de opciones y además como se ha visto es compatible con varias plataformas.

De modo que primero se creará una aplicación web con un framework de desarrollo web móvil, jQuery Mobile, que será la tecnología más utilizada en el proyecto y explicada en el siguiente apartado. Y posteriormente siguiendo las instrucciones ofrecidas por PhoneGap se compilará dicha aplicación web y se obtendrá la aplicación híbrida final.

7. Tecnologías

Después de haber explicado algunos conceptos básicos para poder comprender mejor el desarrollo de la aplicación, se van a mostrar las tecnologías empleadas a lo largo del proyecto. Posteriormente se verá donde han sido utilizadas dichas tecnologías cuando se profundice en el desarrollo de la aplicación. Así las tecnologías utilizadas en la creación de la aplicación son las siguientes:

-HTML5
-CSS3
-JavaScript
-Ajax
-PHP
-Jquery Mobile
-Theme Roller
-Adobe Dreamweaver
-Phonegap.
-Adobe Photoshop
-XAMPP
-Google Map
-Herramientas Google Chrome

- **HTML5 y CSS3.**

En el apartado de conceptos básicos ya se ha explicado ambas tecnologías.
A modo de resumen se puede decir que mediante HTML5 se crea la estructura de la web, el contenido en forma de texto o se añaden objetos para complementar el texto. Mientras que CSS3 se encarga de la parte visual, haciendo más atractiva la web ya que mediante estas hojas de estilo se modifican los textos, las formas, colores, etc.

Además cabe destacar que también mejora la usabilidad y accesibilidad de nuestra aplicación ya que mediante un único CSS definimos todos los elementos y ésta puede cargar más rápido.

- **JavaScript.**

JavaScript es un lenguaje de programación interpretado, por este motivo no requiere compilar los programas antes de ejecutarlos. Esto quiere decir que los programas desarrollados con este lenguaje se pueden ejecutar en el navegador sin necesidad de utilizar ningún programa intermedio.

Este lenguaje es utilizado principalmente para el desarrollo de páginas web dinámicas. Siendo éstas aquellas páginas que no permanecen fijas, es decir, que presentan efectos como animaciones, funciones que se activan al accionar un botón, mensajes de alerta al usuario o texto que aparece y desaparece.

Por otro lado y debido a que se tiende a pensar que JavaScript y Java son el mismo lenguaje, es necesario aclarar que no son lo mismo, se tratan de dos

lenguajes diferentes tanto en concepto como en diseño. Java es un lenguaje de programación más complejo.

Se trata de un lenguaje que proporciona muchas posibilidades, desde la creación de pequeños scripts al desarrollo de grandes programas. Permite al programador acceder a todos los elementos presentes en la página y así crear acciones con ellos.

- **AJAX**

Ajax no se define como una tecnología en sí misma, sino como una unión de varias tecnologías independientes. Se trata de una técnica de desarrollo web para crear aplicaciones interactivas mediante la combinación de varias tecnologías ya existentes.

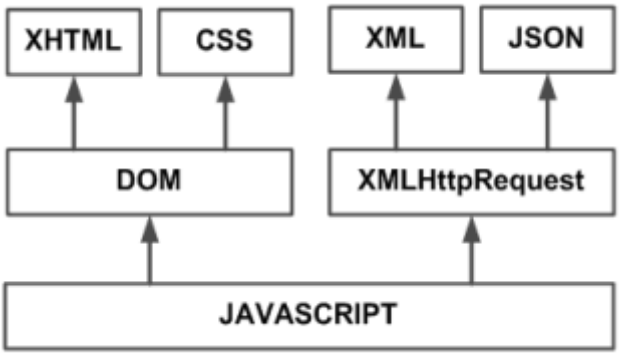

Figura 7.1. Tecnologías agrupadas bajo el concepto de AJAX

Dejando a un lado la explicación técnica, AJAX permite desarrollar aplicaciones capaces de responder a las interacciones del cliente sin refrescar la página. Es decir, lo que hace es actualizar una parte de la página sin volver a cargar todo el código o complementos de la página.

- **PHP**

PHP (*Hypertext Preprocessor*) es un lenguaje de programación del lado del servidor gratuito, es decir, un lenguaje que se ejecuta en el servidor web. Es rápido, independiente de la plataforma y con una gran librería de funciones.

Se trata de un lenguaje popular, existen un gran número de páginas y portales web creados con PHP. Este código puede ser Incrustado en HTML y es posible combinar código PHP con código HTML en un único archivo, siguiendo unas reglas.

PHP es utilizado para generar páginas web dinámicas, aquellas cuyo contenido no es el mismo siempre. Por ejemplo, los contenidos pueden cambiar en base a los cambios que haya en una base de datos, de búsquedas o aportaciones de los usuarios.

- **Jquery Mobile**

El framework utilizado para desarrollar el código base de la aplicación es jQuery Mobile. Permite que las aplicaciones creadas tengan la misma apariencia independientemente del terminal mediante el que se acceda.

No se trata de un framework desarrollado desde cero, se puede entender como un pluging para jQuery ya que se basa en dicho framework JavaScript. Del mismo modo que Jquery, esta versión móvil también se basa en el lema “Write Less, Do More”.

No solo consiste en un framework para desarrollar código JavaScript que pueda ser ejecutado en varios navegadores, la novedad es que presente varias herramientas que facilitan el proceso de desarrollar aplicaciones web.

Las características básicas de jQuery Mobile son las siguientes:

- Es simple de utilizar, desarrollando poco código es posible crear aplicación muy usables y atractivas.
- Jquery Mobile hace uso de las últimas tecnologías web: HTML5, CSS3 y JavaScript.
- Presenta herramientas CSS que permiten que por ejemplo los formularios o las listas se estilicen de forma automática.
- Está preparado para los dispositivos táctiles.
- Presenta muchos automatismos, por ejemplo siempre que pueda realiza conexiones Ajax de forma automática y las transiciones entre páginas también son automáticas.
- Es un framework con un tamaño relativamente pequeño, aproximadamente unos 12Kb de las bibliotecas JavaScript, 6Kb de CSS y algunas imágenes.
- También proporciona una serie de temas para personalizar la aplicación.
- Compatible con varios sistemas operativos:

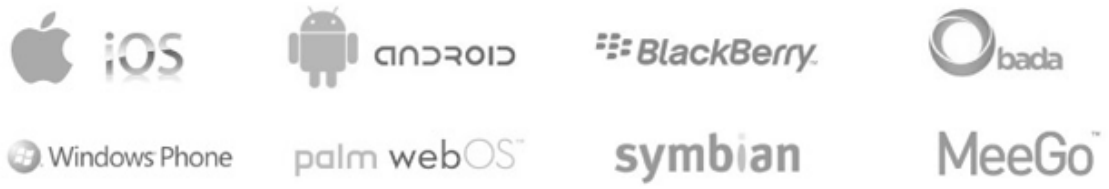

Figura 7.2. Sistemas operativos soportados por jQuery Mobile.

- **PhoneGap.**

Se trata de un framework capaz de desarrollar aplicaciones híbridas con un único código base creado mediante HTML, CSS y JavaScript. Anteriormente ya se ha explicado más profundamente en qué consiste este framework y justificado su elección.

Mediante jQuery Mobile se desarrollará la aplicación web y posteriormente PhoneGap servirá para convertirla en híbrida. Como ya se ha mencionado en el apartado 6 se ha utilizado el compilador en la nube, PhoneGap Build. Este compilador lo que hace es empaquetar la aplicación web dentro de una aplicación nativa, de las siguientes plataformas Android, iOS, Blackberry, Windows Phone, Web OS y Symbian.

- **ThemeRoller.**

Es una herramienta que se encuentra en *jQuery UI* (User Interfaz, interfaz de usuario) para modificar el estilo de la interfaz.

Para ello permite ajustar y definir colores, tipografías, botones, etc. Una vez modificadas las propiedades deseadas, esta herramienta nos permite descargar la plantilla que hemos creado, obteniendo un archivo CSS que introduciéndolo en nuestro código nos proporciona el tema creado.

- **Adobe Dreamweaver.**

Se trata de un software de diseño web, desarrollado para la creación y la edición de sitios web HTML y aplicaciones para dispositivos móviles.

Es el programa de desarrollo y diseño web más utilizado, debido a las funcionalidades que ofrece y a la posibilidad de integrar otras herramientas como Adobe Flash. Además permite la conexión a un servidor o a una base de datos.

- **Adobe Photoshop.**

Consiste en un programa para la creación, edición y retoque de imágenes, desarrollado por la compañía Adobe Systems. En un principio fue creado para la plataforma Apple pero posteriormente se desarrolló para Windows.

Sin duda se trata del programa de edición de fotografía más popular, ello se debe a las numerosas posibilidades de retoque y modificación de fotografías que ofrece.

- **XAMPP**

Es un servidor independiente de plataforma, formado por el servidor web Apache, la base de datos MySQL, y los intérpretes para lenguajes PHP y Perl.

XAMPP nos ofrece la posibilidad de crear un servidor local e independiente en nuestro ordenador con el que probar la aplicación antes de ser subida a un servidor web.

- **Herramientas de Google Chrome.**

- Resize Window.

Mediante esta herramienta se modifica el tamaño de la ventana obteniendo el de los diferentes dispositivos, de modo que nos facilita el trabajo a la hora de visualizar nuestro diseño.

-Lat.-Log

Se trata de una herramienta de Google Maps mediante la cual se obtienen las coordenadas de latitud y longitud de una posición concreta, utilizadas para la creación de mapas.

- **Google Maps**

Google Maps es un servicio de Google que ofrece imágenes vía satélite de todo el planeta. Proporciona mapas desplazables de todo el mundo y otras posibilidades como la obtención de rutas o añadir marcadores en los mapas. Se trata de una herramienta muy útil de Google que tendrá mucha importancia en la aplicación.

8. Desarrollo de la aplicación.

Una vez visto todos los conceptos básicos y tecnologías utilizadas, en este apartado se explicará de forma detallada cómo se ha desarrollado cada parte de la aplicación y dónde han sido utilizadas las tecnologías.

Antes de empezar a desarrollar la aplicación se ha organizado toda la información obtenida y se han estructurado los contenidos de forma que se facilite la navegación al usuario y sea fácil e intuitivo encontrar la información deseada.

También será importante conseguir un aspecto visual atractivo y a la vez sencillo, de modo que el usuario navegue por la aplicación sin que ésta le resulte recargada o pesada y manteniendo una estructura constante para no despistar al cliente.

Una vez creada la aplicación mediante las tecnologías web, se compilará para obtener una aplicación híbrida, más adelante se detallará este proceso.

8.1. Estructura de la aplicación.

8.1.1. Esquema de la aplicación.

Para un correcto desarrollo de la aplicación lo primero que se ha tenido en cuenta es la organización de la misma. Se han organizado los contenidos teniendo en cuenta las necesidades del usuario y de este modo facilitando su uso.

La página de inicio (index.html) está compuesta por cuatro botones:

- El primer botón, dónde comer, que servirá para acceder a los diferentes establecimientos en los que podrán comer aquellos que deban llevar una dieta libre de gluten.

- De la misma forma que el primer botón, encontramos el botón dónde comprar. Pero en este caso, encontraremos establecimientos en los que se podrá comprar productos sin gluten.

- Mediante el botón de recetas se accederá a una página que nos muestre varias recetas, explicadas mediante vídeos o por escrito.

- Y por último el botón de contacto, el cual nos dirige a la página de información de la Asociación de Celíacos de La Rioja, con un mapa que muestra su ubicación. Y a un pequeño formulario, para rellenar en el caso de que conozcamos algún establecimiento que pueda ser incluido en la aplicación.

A su vez tanto la página dónde comer, como la de dónde comprar, están distribuidas del mismo modo. Ambas se componen por botones que indican el tipo de establecimiento (panaderías, restaurantes, etc). Tras haber pulsado, por ejemplo, el botón de panaderías se accederá a una lista de las panaderías disponible.

A través del diagrama de la *figura 8.1* se entenderá mejor el esquema organizativo de la aplicación.

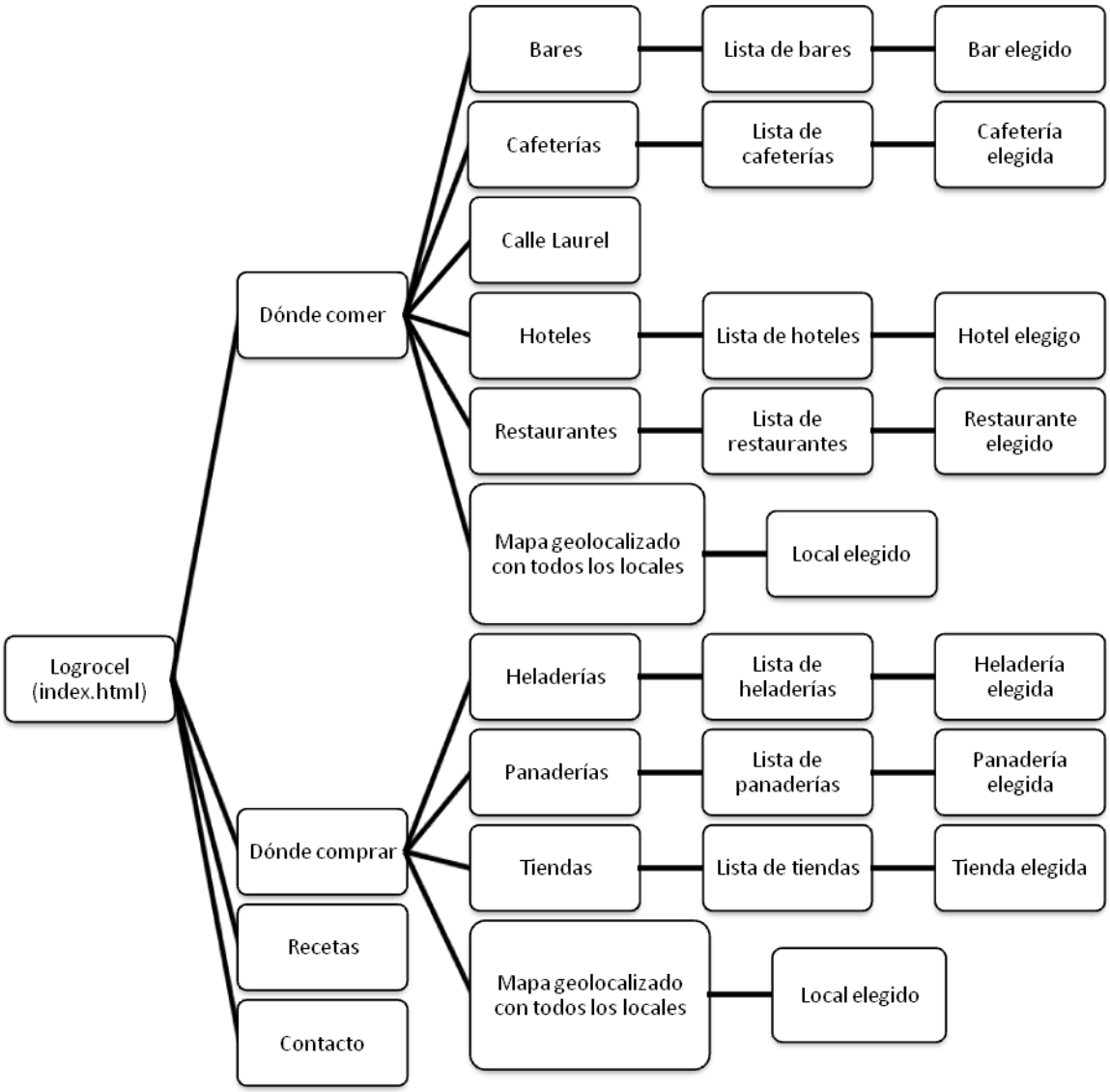

Figura 8.1. Esquema organizativo de la aplicación.

Mediante el esquema anterior podemos observar que la aplicación se organiza de una forma simple, de modo que la navegación es intuitiva y facilita la interacción del usuario. También cabe destacar que no son necesarios más de tres *clicks* para llegar a la información deseada.

8.1.2. Página de inicio.

Antes de explicar la página base, se va a observar la estructura de la página de inicio, la primera página con la que el usuario se va a encontrar.

En esta primera página ya se ve reflejada la temática presente en el diseño de la aplicación y que permanecerá constante durante toda la aplicación.

En el fondo está formado por una imagen de un racimo de uvas editada con Photoshop para resaltar más el color morado, las gotas sobre las uvas y conseguir algún reflejo verde que haga más atractivo el diseño. Cabe destacar que se ha elegido el racimo de uvas debido a que es un símbolo de la comunidad de La Rioja.

En la parte superior se muestra el logo de la aplicación, que consiste en el nombre la aplicación, Logrocel, sustituyendo una “o” por el símbolo de producto sin gluten, un espiga de trigo tachada.

En cuanto al contenido de la página, está formada por una lista personalizada con cuatro botones, que muestran unos iconos sencillos y representativos del contenido que podemos encontrar tras pulsar cada botón.

Los botones también han sido creados mediante el shoftware de edición de imágenes, Photoshop. Primero se ha elegido un tipo de botón de los que ya tiene diseñados Photoshop y posteriormente se ha personalizado añadiendo el icono específico y el color deseado.

Se ha tratado de representar dichos botones con iconos sencillos de comprender por cualquier usuario y para no crear ningún tipo de duda también se ha añadido un texto debajo que explica a que se refiere cada botón.

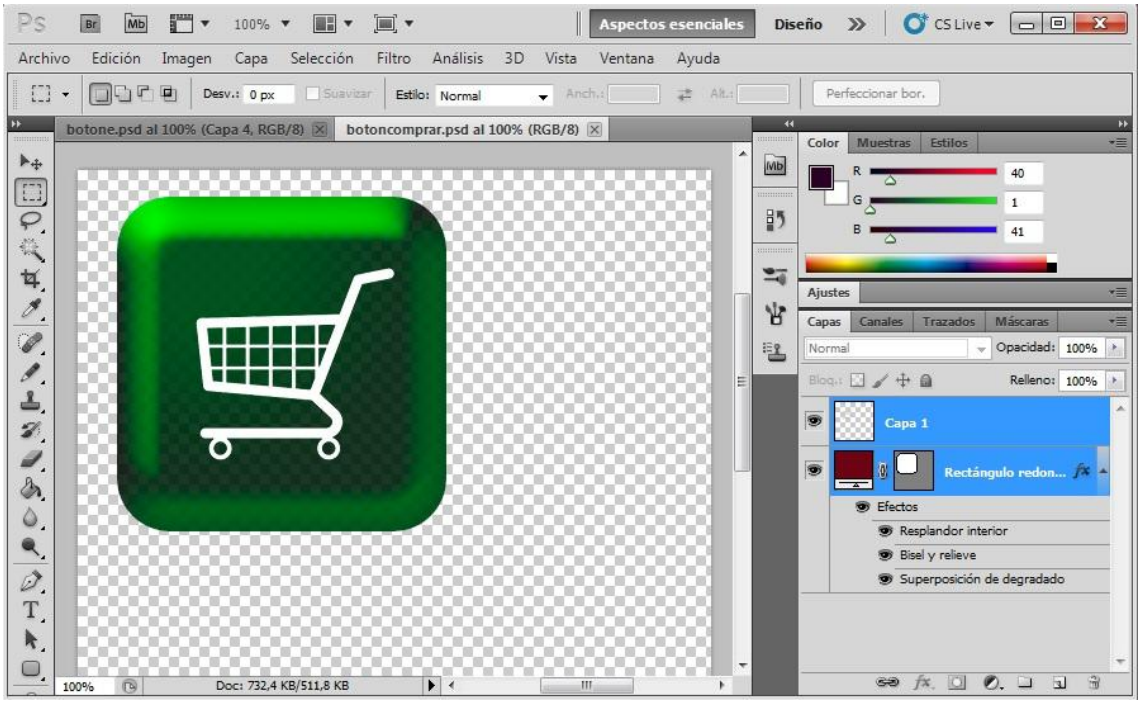

Figura 8.2. Botón "Dónde comprar" creado en photoshop.

Figura 8.3. Página de inicio.

8.1.3. Estructura de la página base.

Después de organizar la estructura de la aplicación, otra parte importante es crear una página base. Una vez diseñada la página base, a la hora de crear el resto de páginas solo variaremos el contenido. De este modo se mantendrá el diseño de forma uniforme durante toda la aplicación.

A excepción de la página de inicio, todas la páginas están creadas sobre dicha página base. Ésta simplemente consta de una cabecera y una barra de navegación en el pie de página, variando el contenido de una página a otra. La temática del diseño sigue siendo las uvas presentes en la primera página, y por lo tanto el color morado está bastante presente.

En primer lugar, la cabecera contiene en la parte izquierda un botón para volver a la página anterior con el logo de la aplicación y una flecha que facilita la compresión de la utilidad del botón. En la parte derecha contiene un botón de inicio representado con una casa.

El pie de página como se observa en la *figura 8.4,* está formado por una barra de navegación, con cuatro botones y sus iconos. Estos cuatro botones son los mismos que los de página principal, de modo que el usuario tiene la posibilidad de dirigirse a las páginas que proporciona el menú principal sin la necesidad de volver a la página principal. Aunque mediante el botón de inicio de la cabecera siempre que se quiera se podrá acceder a la página de inicio.

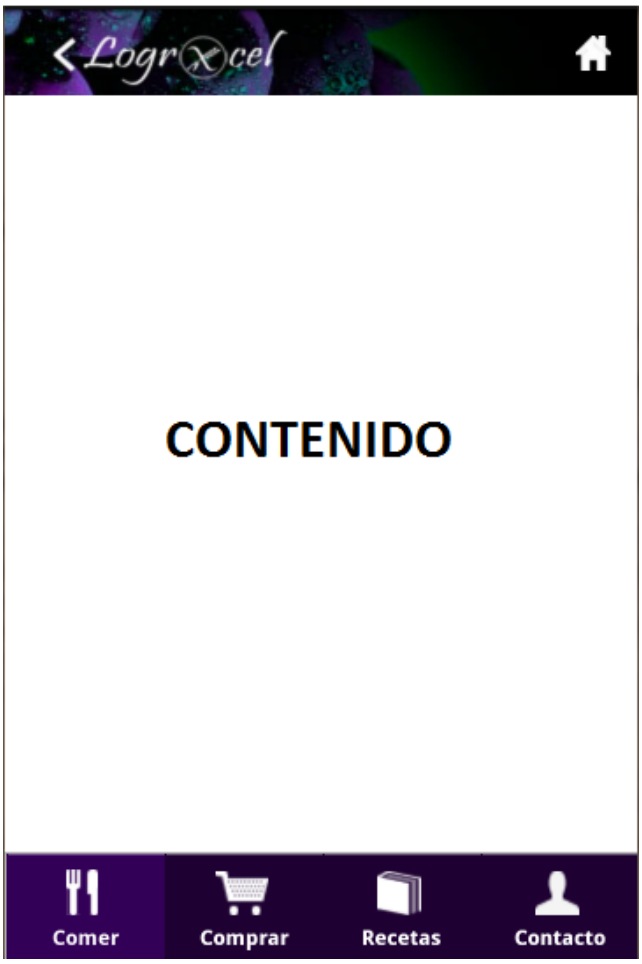

Figura 8.4. Estructura de la página base.

8.2. Aspecto visual.

Mediante HTML 5 se obtiene la forma de la aplicación y las hojas de estilo (CSS) crean el estilo y el aspecto de aquello que se ha creado con HTML.

Ya se ha mencionado que jQuery Mobile cuenta con hojas de estilo que facilitan el desarrollo para móviles y además con una herramienta que permite modificar el estilo del interfaz.

Esta herramienta denominada ThemeRoller permite crear de forma sencilla distintos temas para nuestra aplicación, simplemente modificando manualmente algunas características como el color y las sombras de los botones, colores en los enlaces, etc. Tras conseguir el diseño deseado, ThemeRoller crea una hoja de estilo lista para incluir en el código.

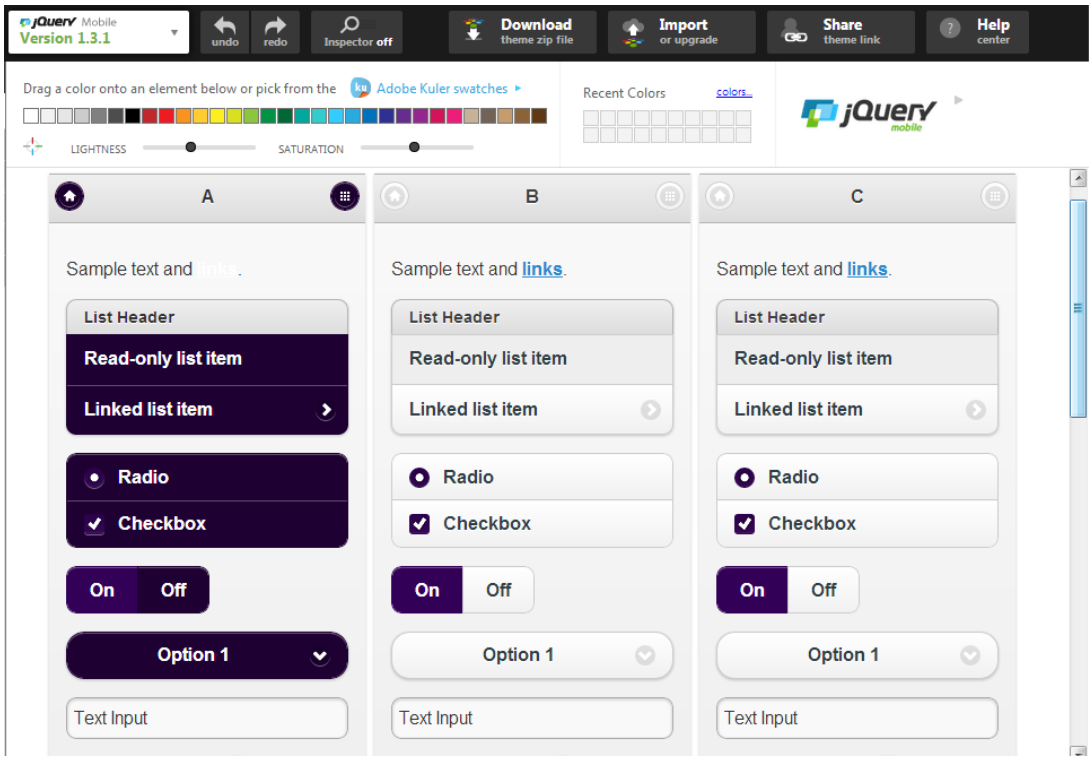

Figure 8.5. Temas diseñados con ThemeRoller.

Además de estos CSS también se han desarrollado dos hojas de estilo para realizar un diseño aún más personalizado y atractivo.

Una de ellas define el estilo de la página de inicio y la otra el diseño de la página base, de los distintos menús y de las páginas de información.

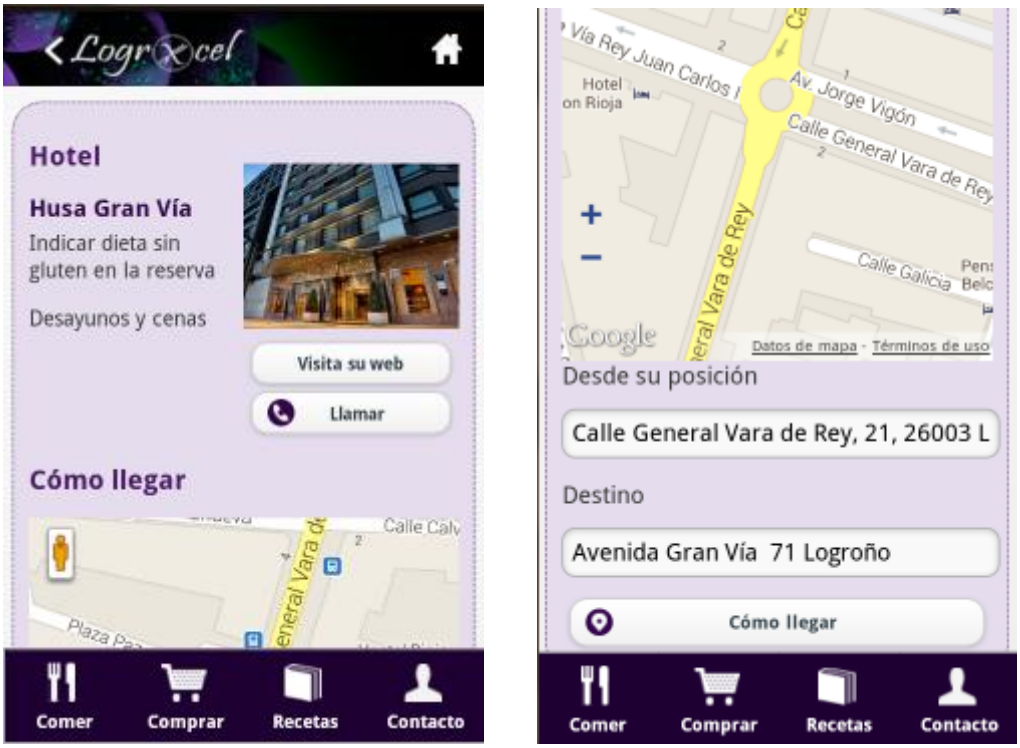

Figura 8.6. Diseño de las páginas de los establecimientos.

Dejando a un lado el estilo de la página base explicado en el apartado 8.1.3, en la *figura 8.7* se observa el código CSS que define el estilo de las páginas de los establecimientos. Se ha personalizado el tipo, el color, el tamaño y los márgenes del texto, incluso se han añadido sombras. El tamaño de las imágenes es reescalable y se han incluido iconos en los botones que jQuery Mobile no incluye en su librería.

```
/*_____ Comer-comprar_____*/

/* Información establecimientos */
h1{
   font-size:20px;
   color: #4c1867;
   text-align:center;
   text-shadow: 0.1em 0.1em 0.05em #CCC;
   padding-bottom:10px;
}
h2 { font-size:18px; margin:15px auto 15px auto; color: #4c1867;
}
h3 {
   font-size:16px;
   margin:0 auto 5px auto;
   color: #33054b;
   font-family: Arial, Helvetica, sans-serif;
}

/* Imagen */
figure { margin:30px auto 5px auto;
 }
figure img{
   width:100%;
 }

 /* Enlaces información establecimiento*/
.enlace_small {font-size: 11px; color:#cccccc;  /* Tamaño letra*/
}
.enlace_medium {font-size: 12px; color:#cccccc;
}
```

```
.ul-icon-maps {
background: #33054b url(images/maps.png) no-repeat; /* Icono*/
}
.ul-icon-tel{
background: #33054b url(images/phone.png) no-repeat;
}
.ul-icon-comer {
background: #33054b url(images/comer.png) no-repeat;
}

.lineainformacion{
    border:1px solid #e5ddee ;
    margin-top:2px;
    margin-bottom:10px;
    padding:0;
   }

 /*Recuadro que rodea la información*/
#recuadro  {

   padding: 10px;
   border: 1px dotted #33054b;
   border-radius: 20px;
   -webkit-border-radius: 20px;
   -moz-border-radius: 20px;
   background: #e5ddee;
   margin: 10px 2% 10px 2%;
}
```

Figura 8.7. Fragmento de código CSS de los establecimientos.

La página de recetas cuenta con un diseño diferente, siempre siguiendo la línea de las demás páginas.

Se han definido dos fondos diferentes que hacen más atractiva la página, también se han añadido dos vídeos y una imagen que se adaptan al tamaño del dispositivo. Y del mismo modo que en el diseño anterior se ha personalizado el estilo del texto y se han definido márgenes.

Figura 8.8. Página de las recetas.

```
/*_____ Recetas_____*/

/*Imagen*/
#recetas img{
    float: right;
    margin-left:9px;
    border: 3px solid rgba(40,1,41,0.9);}

/*Clase para centrar objetos*/
.centrar{
    text-align:center;
    }

/*Fondo de los videos*/
.recuadro_video {
    margin: 0px 5px 0px 5px;
    padding: 5px 15px 10px 15px;
    border: 1px solid #33054b;
    border-radius: 15px;
    -webkit-border-radius:10px;
    -moz-border-radius: 10px;
    background: rgba(40,1,41,0.9);
    font-size:10px;
    text-align:center;
    }

/*Texto recetas video*/
.recuadro_video li{
    color:#fff;
    font-size:16px ;
    text-align:center;
    font-weight:bold;
    margin-bottom:10px;
    margin-top:30px;
    }
.recuadro_video p{
     color:#FFF;
     marging-bottom:50px;
    }

/*Añadir video*/
iframe{
    height: 300px;
    width:90%;
    }

/*Fondo receta escrita*/
.recuadro_receta {
    margin: 10px auto 5px auto;
    padding: 10px;
    border: none;
    background: #fff url(images/papel.jpg) repeat right 0px;
    }

/*Texto recetas*/
.recuadro_receta li{
    font-size:12px;

    }
h4{
    color: #33054b;
    font-family: Arial, Helvetica, sans-serif;
    }
#recetas p{
    font-size:15px ;
    text-align:center;
    color:#fff;
    margin-bottom:10px;
    margin-top:10px;
    }
#recetas h6{
    font-size:25px ;
    margin-bottom:20px;
    margin-top:10px;
    margin-left:20px;
    }
```

Figura 8.9. Código CSS de la página de recetas.

Por último el diseño de la página de contacto está dividido en dos, una primera parte con información de la Asociación de Celíacos de La Rioja con el mismo diseño que las páginas de los establecimientos. Y una segunda parte que contiene un formulario de contacto.

El formulario se define mediante los estilos propios de CSS pero para personalizarlo más se ha añadido un fondo y modificado los márgenes de modo que el aspecto sea más espaciado y más claro.

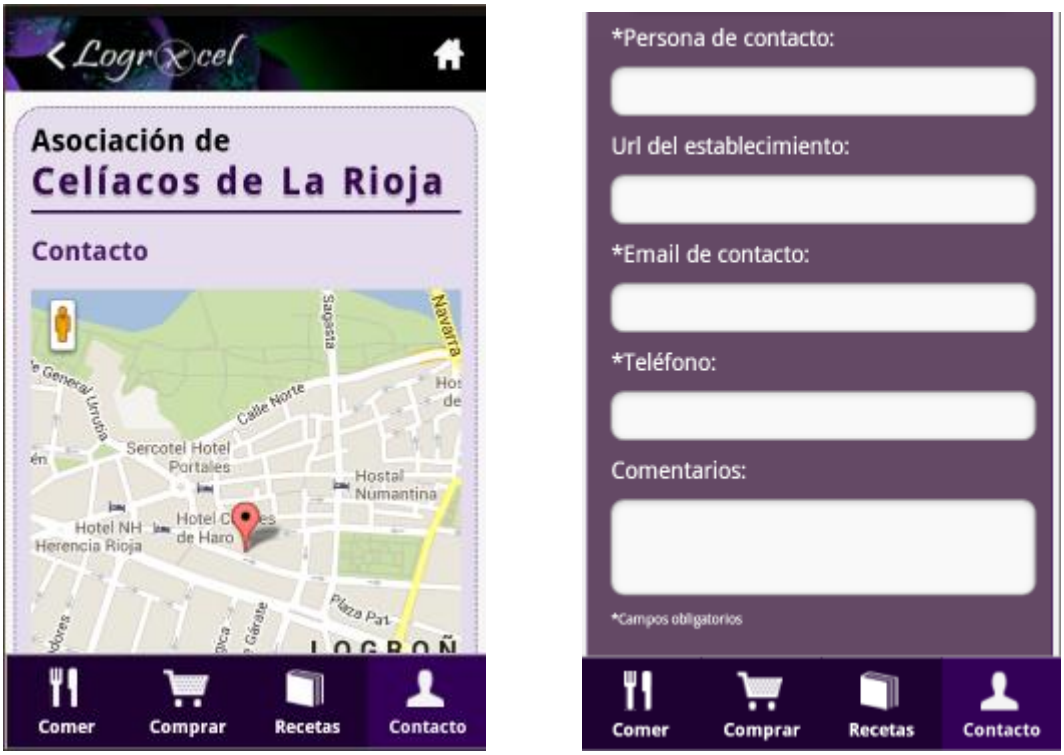

Figura 8.10. Diseño de la página de contacto con la Asociación.

```css
/*_____ Contacto_____ */

/*texto */
h5{
    color: #000;
    font-family: Arial, Helvetica, sans-serif;
    font-size:20px;
    margin:0;
    padding:0;
}
h6{
     text-shadow: 0 1px 0 rgba(255, 255, 255, .4), 0px 2px 0 rgba(0, 0, 0, .3);
    text-align: left;
    color: #33054b;
    margin: 0 0 5px 0;
    letter-spacing: 2px;
    font-family: Arial, Helvetica, sans-serif;
    position: relative;
    font-size:25px;
}
.linea1{
     border:1px solid #33054b ;
     margin-top:2px;
     margin-bottom:10px;
     padding:0;

    }
/*Fondo del formulario */
.recuadro_formulario  {
    margin: 0px 5px 5px 5px;
    padding: 15px;
    border: 1px solid #33054b;
    border-radius: 15px;
    -webkit-border-radius:10px;
    -moz-border-radius: 10px;
    background: rgba(40,1,41,0.7);
    }

/*Texto*/
.recuadro_formulario li{
    font-size:16px ;
    color:#fff;
    text-align:left;
    font-weight:bold;
    margin-bottom:10px;
    margin-top:10px;
    margin-left:10px;
    }
.recuadro_formulario p{
    font-color:#000;
    margin-bottom:10px;
    margin-top:10px;
    }
.enunciado {
    color:#fff;
    font-size:15px ;
    margin-bottom:10px;
    margin-top:10px;
    }
#contacto .ui-grid-a p{
    margin-left:10px;
    margin-top:10px;
    text-align:center;
    }
 .campos {
    color:#000;
    margin-bottom:10px;
    margin-top:10px;
    margin-left:0px;
    margin-rigth:5px;
    }
.obligatorios p{
    font-size:9px;
    color:#FFFFFF;
    }
```

Figura 8.11. Código CSS de la página de contacto con la Asociación.

8.3. Funcionalidades.

Después de analizar la estructura y el diseño de la aplicación se va a explicar las diferentes funcionalidades que la forman:

- *Menús y navegación.* Se mostrará la utilización de los diferentes recursos que proporciona jQuery Mobile mediante el uso de las tecnologías web, HTML5 y CSS3.

- *Localización.* Una de las partes fundamentales de la aplicación es la geolocalización de cada establecimiento, mediante un JavaScript se localizará al usuario y se trazará la ruta hasta el establecimiento elegido.

- *Multimedia.* Con la ayuda de Youtube se insertarán de forma fácil vídeos en la aplicación.

- *Formulario*. En la página de contacto se podrá rellenar un formulario para informar sobre nuevos establecimientos que puedan ser añadidos en la aplicación.

8.3.1. Menús y navegación.

Una de las automatizaciones que incluye jQuery Mobile es la navegación mediante Ajax. Debido a que ocasionaba problemas de navegación y en muchas ocasiones surgían errores, se ha deshabilitado Ajax mediante el siguiente Script.

```
<script>
    $(document).ready(function() {
    // Deshabilitar Ajax
    $.mobile.ajaxLinksEnabled = false;
    $.mobile.ajaxFormsEnabled = false;
    $.mobile.ajaxEnabled = false;
    });
</script>
```

Figura 8.12. Script para deshabilitar Ajax.

Como ya se ha visto la página base está formada por una cabecera que consta de dos botones o navegadores y un pie de página con una barra de navegadores.

La cabecera simplemente consta de dos botones y una imagen en el fondo acorde con la temática escogida para la aplicación que personaliza más la aplicación. Ambos botones han sido creados con Photoshop y han sido incluidos en la cabecera mediante CSS.

```
<header>
 <a class="back" href="#" data-rel="back"  data-iconpos="notext"> </a>
 <a class="inicio" href="index.html"  data-iconpos="notext"> </a>
</header>
```

Figura 8.13. Código HTML de los botones de navegación de la cabecera.

Figura 8.14. Navegador en la cabecera

Figura 8.15. Barra de navegación en el pie.

Glyphish es una página web que proporciona numerosos iconos para incluir en una barra de navegación. Dentro de las posibilidades que ofrece esta página se han buscado aquellos que resulten apropiados para cada función y mediante CSS se han incorporado en la barra de navegación.

```
 <!--</Barra de navegacion inferior>-->
<div data-role="footer" data-position="fixed" class="nav-glyphish" >
    <div data-role="navbar" class="nav-glyphish" >
    <ul>
        <li><a href="comer.html" id="comer"  data-icon="custom" class="ui-btn-active">Comer</a></li>

        <li><a href="comprar.html" id="comprar" data-icon="custom">Comprar</a></li>
        <li><a href="recetas.html" id="recetas" data-icon="custom">Recetas</a></li>
        <li><a href="contacto.html" id="contacto" data-icon="custom">Contacto</a></li>

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

Figura 8.16. Código HTML de la barra de navegación inferior.

Tanto la página “Dónde comer” como la de “Dónde comprar” están formadas por un menú que consta de varias imágenes, éstas han sido creados mediante Photoshop. De modo que este menú está formado por varios enlaces con forma de imagen.

```
      <div class="enlaces">

        <a href="heladeriassingluten.html"><img border="0" src="images/comprar/heladerias.png" width="140" height="140" alt=
"Heladerías"></a>
    <a  href="panaderiassingluten.html"><img src="images/comprar/panaderias.png" width="140" height="140" border="0" alt="Panaderías"
 ></a>

   <a href="tiendassingluten.html"><img border="0" src="images/comprar/tiendas.png" width="140" height="140" alt="Tiendas"></a>

     <a href="comprarcercanos.html"><img border="0" src="images/comer/cercanos.png" width="140" height="140" alt="Heladerías"></a>


       </div>
```

Figura 8.17. Código HTML del menú de la página “Dónde comprar”.

Figura 8.18. Menú de la página "Dónde comprar".

Para seleccionar el establecimiento concreto, es decir, para elegir un bar en concreto dentro del apartado de los bares se muestra una lista.

Mediante jQuery Mobile es muy fácil conseguir listas muy vistosas con muy poco código. Se han creado listas con foto, listas con un pequeño texto de descripción y listas simples. A todas ellas mediante CSS se les ha eliminado la flecha que pone por defecto jQuery Mobile y se ha añadido una un poco más elegante. También se ha añadido una barra de filtrado que no es no lo mismo que una barra de búsqueda pero también puede ayudar al usuario en la búsqueda.

```
<ul  data-role="listview" data-inset="true" class="mainmen" data-filter="true"  data-theme="b"  attributo data-filter-placeholder=
"¿Qué heladería buscas?">

        <li><a href="donatella.html"><h3>Donatella </h3> </a> </li>
       <li><a href="santangelos.html"><h3>Santangelo´s </h3> </a> </li>
       <li><a href="smooy.html"><h3>Smöoy</h3> </a> </li>
        <li><a href="veneciana.html"><h3>Veneciana </h3> </a> </li>
  </ul>
```

Figura 8.19. Código HTML de una lista simple.

```
<ul  data-role="listview" data-inset="true" class="mainmen" data-filter="true"  data-theme="b"  attributo data-filter-placeholder=
"¿Qué bar buscas?">

            <li><a href="larueda.html" ><h3>La Rueda</h3> <p>Pinchos </p></a> </li>
            <li><a href="tiotito.html"><h3>Tio Tito </h3> <p>Bocadillos y cerveza sin gluten. </p></a> </li>
            <li><a href="lafundacion.html" ><h3>La Fundación </h3> <p>Cerveza sin gluten </p></a> </li>
            <li><a href="bocaboca.html" ><h3>Boca Boca </h3> <p>Pinchos, bocadillos y cerveza. </p></a> </li>
        </ul>
```

Figura 8.20. Código HTML de una lista con una pequeña descripción.

```
<ul  data-role="listview" data-inset="true" class="mainmen" data-filter="true"  data-theme="b"  attributo data-filter-placeholder=
"¿Qué hotel buscas?">

            <li><a href="husagranvia.html" ><img src="images/comer/hoteles/husagranvia.jpg" width="96" height="96"><br/><h3>Husa Gran
Vía</h3></a></li>
            <li><a href="trypbracos.html" ><img src="images/comer/hoteles/trypbracos.jpg" width="96" height="96"> <br/><h3>Tryp Bracos
</h3></a></li>
            <li><a href="nhherencia.html" ><img src="images/comer/hoteles/nhherencia.jpg" width="96" height="96"><br/><h3>NH Herencia
</h3></a></li>
            <li><a href="portales.html" ><img src="images/comer/hoteles/hotelportales.jpg" width="96" height="96"><br/><h3>Portales</
h3></a></li>
        </ul>
```

Figura 8.21. Código HTML de una lista con una pequeña descripción.

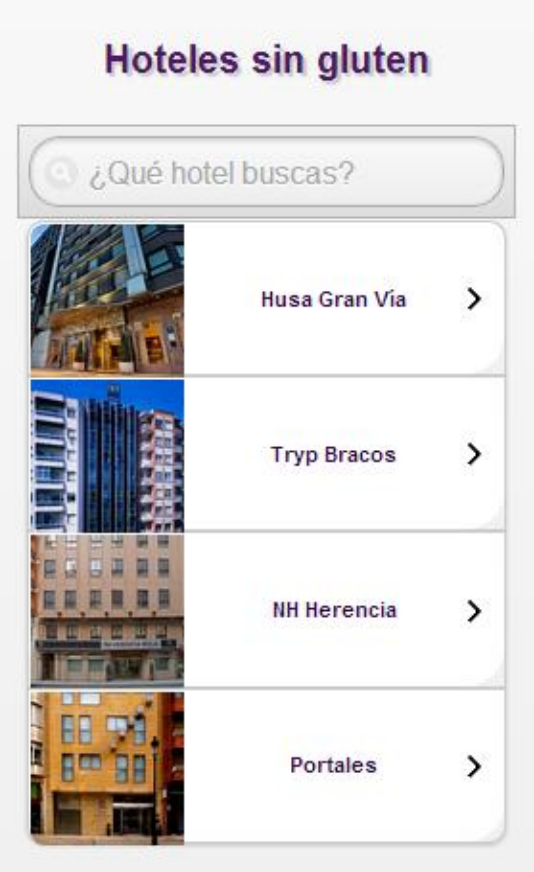

Figura 8.22. Lista con foto

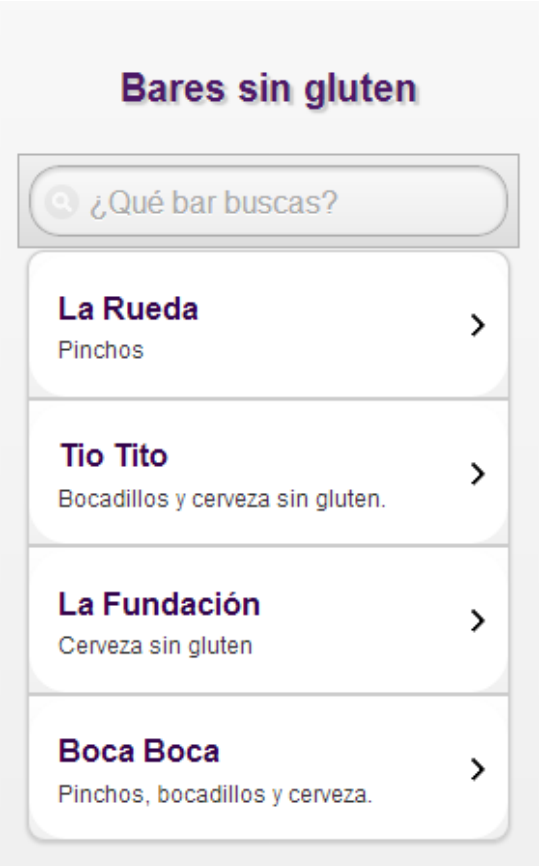

Figura 8.23. Lista con descripción.

Figura 8.24. Lista simple.

8.3.2. Localización.

Dentro de este apartado de localización se van a explicar los diferentes mapas que aparecen en la aplicación.

Se han realizado tres tipos de mapas: un mapa en el que mediante un marcador se indica una posición concreta definida por sus coordenadas, otro que localiza la posición del dispositivo y ofrece al usuario unas pautas para dirigirse hasta el establecimiento. Y por último un mapa con varios marcadores que indican la posición de varios locales y a su vez indica la posición del usuario en ese mismo mapa.

En la página de oficial de jQuery Mobile encontramos varios ejemplos de mapas que utilizan las bibliotecas de Google Maps. Los dos primeros mapas han sido generados a partir de estos ejemplos.

En primer lugar se explicará la localización más simple, mediante este mapa se muestra la ubicación de la Asociación de Celíacos de La Rioja indicada con un marcador. Simplemente se deben añadir unos scripts y descargar unos plugins, que serán explicados posteriormente. También debemos añadir en un script las coordenadas del establecimiento para que el mapa se centre en estas y se añada un marcador en dicha dirección.

```
<script type="text/javascript">
            var mobileDemo = { 'center': '42.4648,-2.4493', 'zoom': 15 };
            ////////////////////////////////////////////////////////////

            $('#contactoasociacion').live('pageinit', function() {
                demo.add('basic_map', function() {
                    $('#map_canvas').gmap({'center': mobileDemo.center, 'zoom':
mobileDemo.zoom, 'zoomControl':true,'streetViewControl':true,'disableDefaultUI':true,
'callback': function() {
                        var self = this;
                        self.addMarker({'position': this.get('map').getCenter() }). °
                    }});
                }).load('basic_map');
            });

            $('#contactoasociacion').live('pageshow', function() {
                demo.add('basic_map', function() { $('#map_canvas').gmap('refresh');
}).load('basic_map');
            });

</script>
```

Figura 8.25. Script modificado para realizar el mapa de contacto.

En la anterior figura se observa las coordenadas añadidas para posicionar nuestra ubicación deseada, así como el zoom deseado. Estas coordenadas han sido obtenidas mediante la herramienta *lat.-long* de Google Maps que nos muestra las coordenadas de un punto concreto. También han sido añadidos los controles de zoom y *Google Street.*

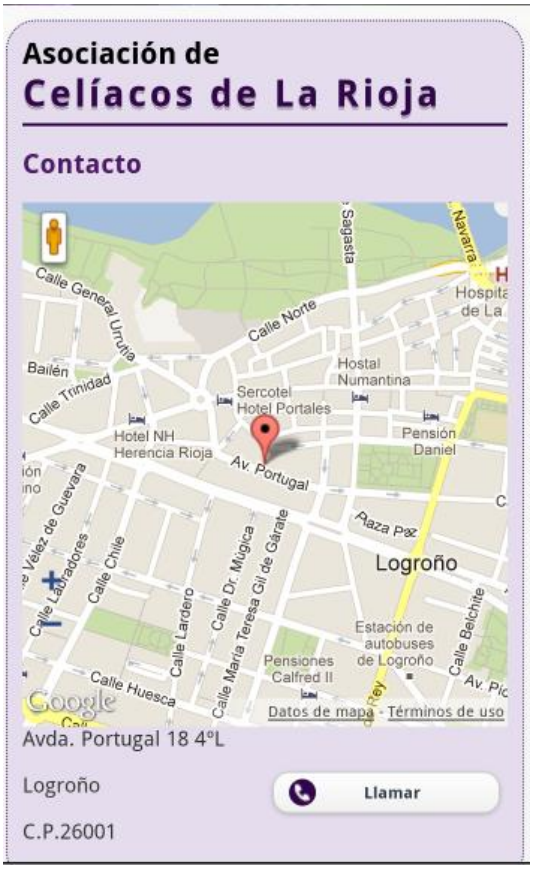

Figura 8.26. Mapa simple con marcador.

Por otro lado, tenemos una de las partes más útiles de la aplicación. Mediante un script creado por Google será posible localizar la posición del dispositivo, siempre y cuando éste admita ser localizado. Y una vez localizado el dispositivo, si el usuario pulsa el botón *"Cómo llegar"*, se trazará la ruta a realizar para llegar al establecimiento elegido.

Esta localización depende del dispositivo y del navegador. El trabajo de obtener la localización lo realizará el navegador y a pesar de que la mayoría soportan HTML5 y jQuery Mobile, existe la posibilidad de que haya algunos problemas de compatibilidad.

Se ha probado en Google Chrome, Safari, Mozilla y Opera. En Google Chrome y en Safari se han obteniedo resultados satisfactorios y ningún problema de

compatibilidad. Sin embargo en Mozilla y Opera en ocasiones han surgido errores.

En la *figura 8.27* vemos que tras aceptar compartir la ubicación, en la casilla "Desde su posición" se añade la localización del dispositivo y aparece en el mapa.

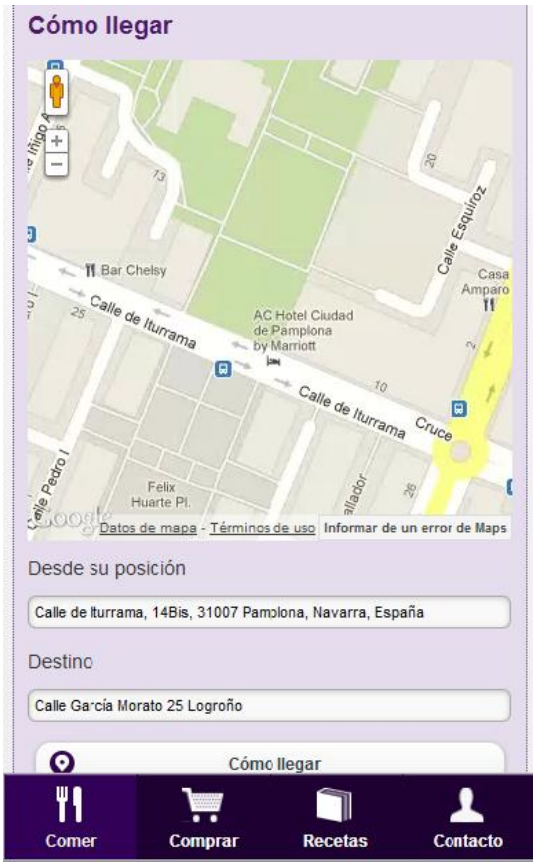

Figura 8.27. Ejemplo de localización del dispositivo.

En la siguiente figura, *figura 8.28*, se observa cómo tras pulsar "Cómo llegar" aparece trazada la ruta en el mapa y una serie de indicaciones a seguir para llegar al establecimiento elegido.

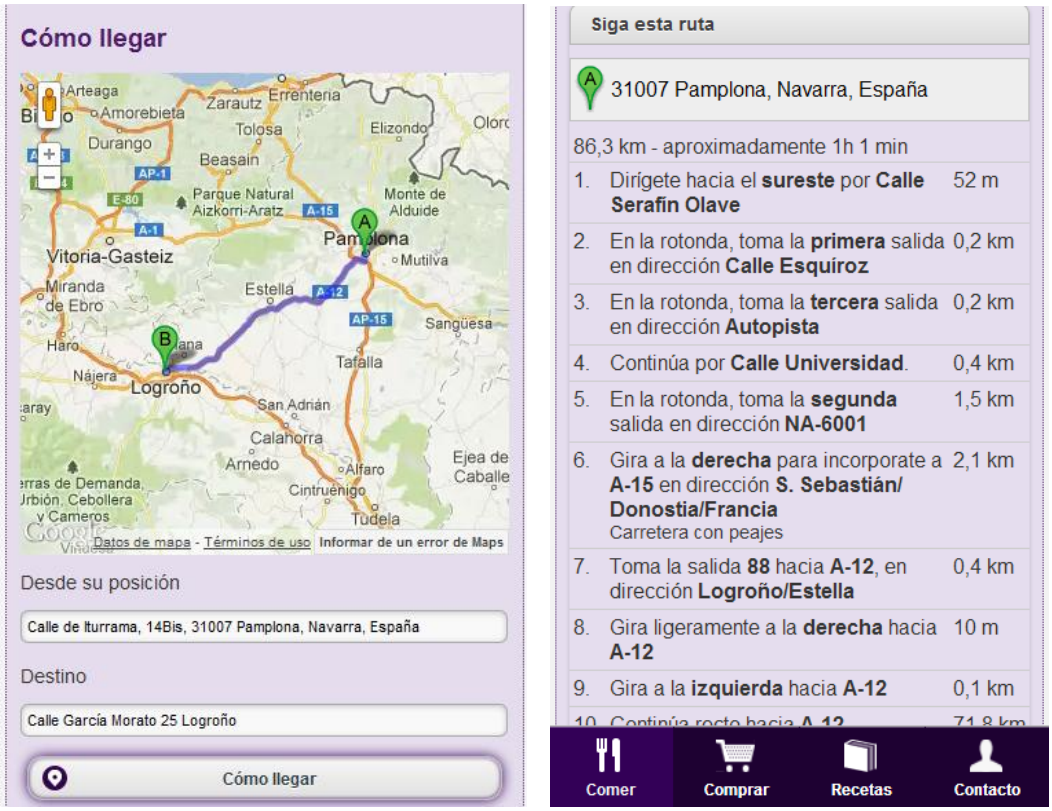

Figura 8.28. Ejemplo de ruta trazada e indicaciones a seguir.

Para poder utilizar esto dos servicios ofrecidos por Google es necesario añadir unos scripts en el código y descargar unos plugins Jquery ui, disponibles en la página web oficial. Estos plugins son los siguientes:

- *jquery.ui.map.js*: que muestra el map.
- *jquery.ui.map.service.js*: que permite aplicar el servicio de Google Maps.
- *jquery.ui.map.extensions.js*: para poder introducir los marcadores.
- *demo.js*: que hace posible que funcionen los tres anteriores.
- *maps.google.com/maps/api/js?sensor=true:* activa el sensor de geolocalización.

En el mapa simple, explicado anteriormente y mostrado en la figura 8.26 no es necesario activar el sensor de geolocalización.

Por último una vez añadidos estos scripts, es necesario un último script que contienen los datos más específicos.

```
<script type="text/javascript">
var mobileDemo = { 'center': '42.4687, -2.4627', 'zoom': 17 };

    $('#bocaboca').live('pageinit', function() {
            demo.add ('directions_map', function() {
                $('#map_canvas_1').gmap({'center': mobileDemo.center, 'zoom': mobileDemo.zoom, 'zoomControl':true,'streetViewControl':true, 'd
isableDefaultUI':true, 'callback': function() {
                    var self = this;
                    self.set('getCurrentPosition', function() {
                        self.refresh ();
                        self.getCurrentPosition( function(position, status) {
                            if ( status === 'OK' ) {
                                var latlng = new google.maps.LatLng(position.coords.latitude, position.coords.longitude)
                                self.get ('map').panTo(latlng);
                                self.search({ 'location': latlng }, function(results, status) {
                                    if ( status === 'OK' ) {
                                        $('#from').val(results[0].formatted_address);
                                    }
                                });
                            } else {
                                alert ('No es posible obtener su posición');
                            }
                        });
                    });
                    $('#submit').click(function () {
                        self.displayDirections ({ 'origin': $('#from').val(), 'destination': $('#to').val(), 'travelMode': google.maps.DirectionsTravelMode.D
RIVING }, { 'panel': document.getElementById('directions')}, function(response, status) {
                            ( status === 'OK' ) ? $('#results').show() : $('#results').hide();
                        });
                        return false;
                    });
                });
            }).load('directions_map');
        });

        $('#bocaboca').live('pageshow', function() {
            demo.add ('directions_map', $('#map_canvas_1').gmap('get', 'getCurrentPosition')).load('directions_map');
        });

</script>
```

Figura 8.29. Script del mapa que traza la ruta a seguir.

El primer cambio a realizar en el script son las coordenadas, de modo que en el caso de que el usuario no permita la geolocalización o surja error localizando al dispositivo, el mapa se inicie en esas coordenadas. Así añadiremos las coordenadas de cada establecimiento. Además también se han añadido los controles de zoom y Google Street para ofrecer más facilidades al usuario.

En el caso de que no se pueda acceder a los servicios de de geolocalización se le informará al usuario de ello mediante un mensaje de alerta.

Figura 8.30. Mensaje para indicar que no ha sido posible obtener la ubicación del usuario.

El tercer tipo de mapa no se trata de uno de los ejemplos que componen la página oficial de jQuery Mobile. Para crear este mapa se ha utilizado las librerías de Google Maps basadas en la API de Google Maps JavaScript v3.

En este caso se ha realizado dos mapas de este tipo, uno que contiene todos los establecimientos que se encuentran dentro de la clasificación “Dónde comer” y otro para aquellos que están dentro de “Dónde comprar”.

En primer lugar se inicia un mapa que contiene todas las posiciones de los locales indicadas con marcadores personalizados y tras aceptar compartir la ubicación, un marcador circular muestra la posición del usuario y además el mapa se centra en esa posición. De modo que el usuario puede observar aquellos locales que están más cercanos.

Figura 8.31. Mapa inicial sin ubicación.

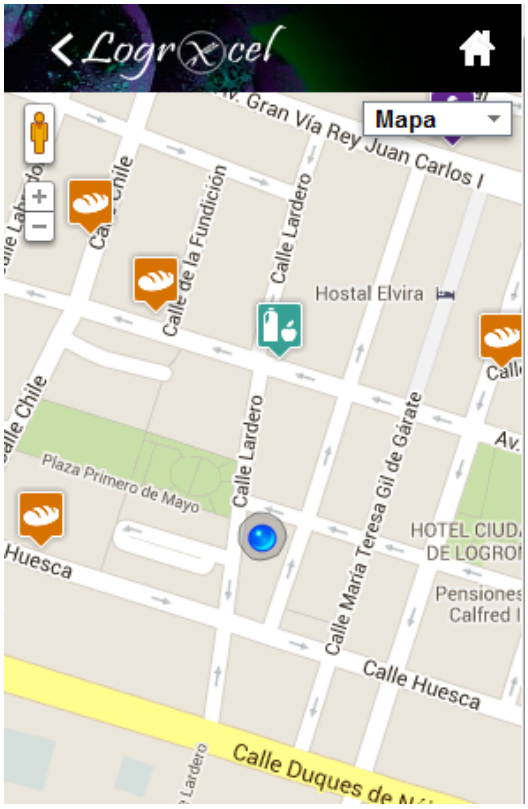

Figura 8.32. Mapa tras compartir ubicación.

Además cada marcador al ser pulsado muestra una ventana de información, ésta indica el nombre del establecimiento y un enlace con la ruta a la página de dicho establecimiento.

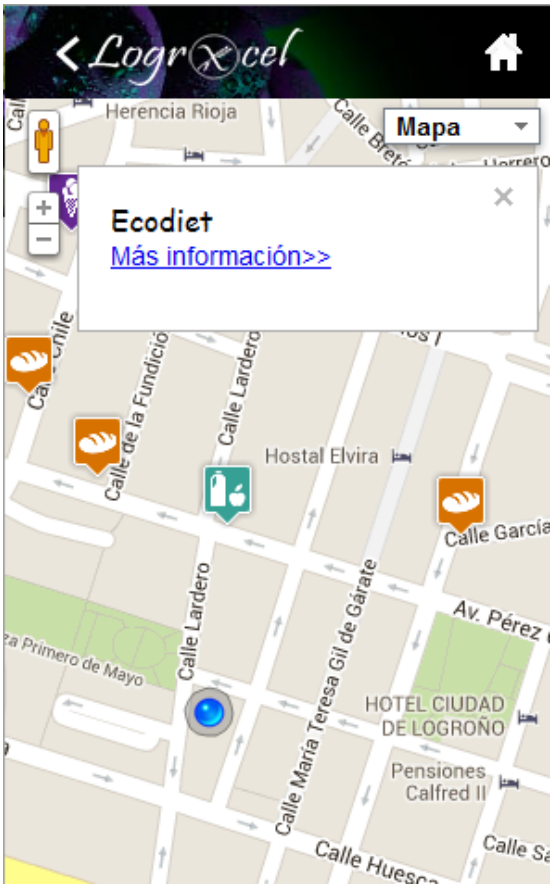

Figura 8.33. Ventana de información obtenida tras pulsar un marcador.

```
<script>

var popup;
var map, GeoMarker;

 function initialize() {

  var mapOptions = {
    zoom : 15,
    center : new google.maps.LatLng(42.4621, -2.4516),
    mapTypeId: google.maps.MapTypeId.ROADMAP
  };

   map = new google.maps.Map(document.getElementById('map_canvas'),
      mapOptions);

   GeoMarker = new GeolocationMarker(map);
   GeoMarker.setCircleOptions({fillColor: '#808080'});

   google.maps.event.addListenerOnce(GeoMarker, 'position_changed', function() {
     map.setCenter (this.getPosition());
     map.setZoom(16);

   GeoMarker.setMap(map);

   });



    var n=1;
    var locales = [
    //                          HELADERÍAS
    {'position':new google.maps.LatLng(42.4664,-2.4474)
        , 'info': 'Heladeria Smöoy'  + "<p> <a href=\"smooy.html\">Más información>></a></p>"      , 'icon': 'images/marcadores/icecream.p
ng'  },
    {'position':new google.maps.LatLng(42.4646,-2.4531)
        , 'info': 'Heladeria Santagelo´s'  + "<p> <a href=\"santangelos.html\">Más información>></a></p>"      , 'icon': 'images/marcadores/i
cecream.png'  },
        {'position':new google.maps.LatLng(42.4651,-2.4558)
        , 'info': 'Heladeria Donatella'  + "<p> <a href=\"donatella.html\">Más información>></a></p>"      , 'icon': 'images/marcadores/icecre
am.png'  },
        {'position':new google.maps.LatLng(42.4640,-2.4492)
        , 'info': 'Heladeria Veneciana'  + "<p> <a href=\"veneciana.html\">Más información>></a></p>"      , 'icon': 'images/marcadores/icecr
eam.png'  },
```

```
        //                          PANADERÍAS
        {'position':new google.maps.LatLng(42.4625,-2.4528)
        , 'info': 'Panaderia El Horno'  + "<p> <a href=\"elhorno.html\">Más información>></a></p>"    , 'icon': 'images/marcadores/bread.p
ng' },
        {'position':new google.maps.LatLng(42.4604,-2.4542)
        , 'info': 'Panaderia Panishop'  + "<p> <a href=\"panishop.html\">Más información>></a></p>"    , 'icon': 'images/marcadores/bread
.png' },
        {'position':new google.maps.LatLng(42.4632,-2.4536)
        , 'info': 'Panaderia Viena'  + "<p> <a href=\"viena.html\">Más información>></a></p>"    , 'icon': 'images/marcadores/bread.png'  },
        {'position':new google.maps.LatLng(42.4620,-2.4486)
        , 'info': 'Alimentación De Torre'  + "<p> <a href=\"detorre.html\">Más información>></a></p>"    , 'icon': 'images/marcadores/bread
.png' },

        //                          TIENDAS
        {'position':new google.maps.LatLng(42.4658,-2.4309)
        , 'info': 'Santiveri'  + "<p> <a href=\"santiveri.html\">Más información>></a></p>"    , 'icon': 'images/marcadores/grocery.png'  },
        {'position':new google.maps.LatLng(42.4621,-2.4513)
        , 'info': 'Ecodiet'  + "<p> <a href=\"ecodiet.html\">Más información>></a></p>"    , 'icon': 'images/marcadores/grocery.png'  }

        ]


      var n=1;
     for(var i in locales){
        var note = locales[i]['info']
    var marker = new MarkerWithLabel({
      position : locales [i]['position'],
      map: map,
        icon:locales[i]['icon']
      });


      // Agregar ventana de información con evento
      mostrarMensaje( marker,locales[i]['info'] );
    }

    function mostrarMensaje(marker, msg) {
      // Crear ventana de información.
      var infowindow = new google.maps.InfoWindow({
        content  : msg
      });

      // Crear evento para mostrar la ventana al dar click
      google.maps.event.addListener(marker, 'click', function(){
        infowindow.  open (map, marker);
      });


    }

  // Crear evento para mostrar mensaje de error
   google.maps.  event.addListener (GeoMarker, 'geolocation_error', function(e) {
    alert ('No es posible obtener su posición.');
   });

  }

  google.maps.  event.addDomListener  (window, 'load', initialize);
</script>
```

Figura 8.34. Script principal del mapa con todos los locales.

Primero se inicializa el mapa centrado con las coordenadas que se indican en el primer recuadro y posteriormente al detectar un cambio de posición debido a la geolocalización, se añade un marcador en esa posición, se centra el mapa también con dicha posición obtenida y se reduce el zoom.

Tanto para añadir los marcadores como las ventanas de información se ha utilizado un bucle *for* que facilita el proceso. Se han creado unas variables que

contienen la información (coordenada, icono, nombre, enlace hacia la página del local) y con este bucle se van asignado a cada marcador.

También es necesario añadir menciones a otros scripts:

- *markerwithlabel.js:* hace posible añadir marcadores.
- *geolocationmarker-compiled.js*: permite añadir el marcador con geolocalización.
- *maps.googleapis.com/maps/api/js?sensor=true*: permite activar el sensor de geolocalización.

En el caso de que ocurra un error de geolocalización y no sea posible conseguir la ubicación del dispositivo se mostrará un mensaje de alerta igual que se ha mostrado en el caso anterior.

8.3.3. Vídeos.

En el apartado de recetas se han añadido unos videos con la ayuda de la plataforma Youtube, se trata de un par de vídeos explicativos sobre cómo realizar algunas recetas sin gluten.

Gracias a Youtube es posible introducir estos videos en la aplicación, ya que esta plataforma proporciona un código para facilitar el trabajo, de modo que simplemente es necesario introducir ese código en donde se quiere mostrar el video.

En la siguiente figura se observa el uso de la etiqueta <iframe> de Html5 para añadir vídeos.

```
<iframe class="youtube-player"  src="http://www.youtube.com/embed/-a-nbH24ed4" frameborder="0" allowfullscreen>
</iframe>
```

Figura 8.35. Código para insertar un vídeo de Youtube.

El alto y ancho del vídeo se ha definido en el archivo CSS, se ha fijado un alto de 300 píxeles y un ancho proporcional de 90%. De este modo, el ancho es reescalable dependiendo de la anchura del dispositivo con el que se acceda a la aplicación.

```
iframe{
    height: 300px;
    width:90%;

    }
```

Figura 8.36. Tamaño del vídeo definido en CSS.

Además, también es posible la reproducción en pantalla completa.

Figura 8.37. Uno de los vídeos insertados en la aplicación.

8.3.4. Formulario.

En la página de contacto se ha introducido un formulario que permitirá al usuario informar de algún posible establecimiento que también sea apto para celíacos y pueda ser añadido en la aplicación.

El formulario contiene un campo desplegable con tres opciones a escoger y el resto son campos que el usuario debe rellenar. En la parte inferior del formulario hay dos botones, uno para borrar todos los datos y otro para enviar el formulario.

Figura 8.38. Formulario de contacto.

Algunos campos del formulario se deben rellenar obligatoriamente para que el formulario se envíe, es decir, no se puede dejar dichos campos en blanco. Esto se consigue mediante el atributo “required” que comprueba si el campo ha sido rellenado sin necesidad de añadir ningún código JavaScript. El único inconveniente es que este atributo no es compatible con todos los navegadores.

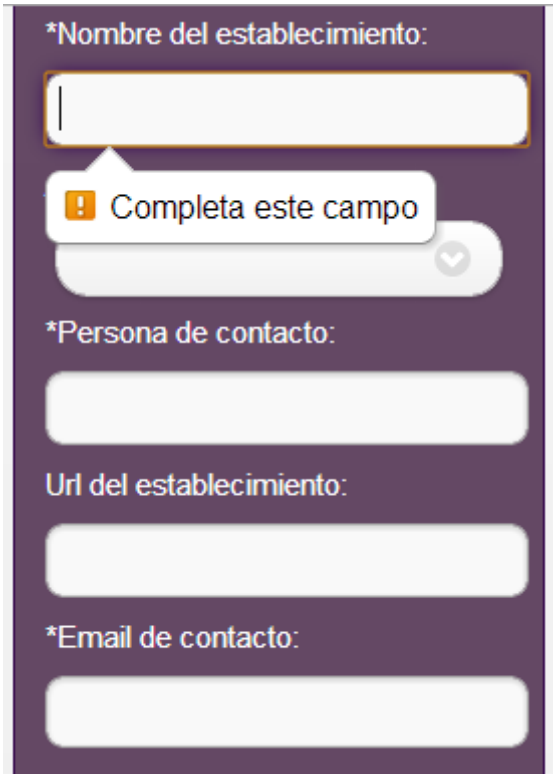

Figura 8.39. Comprobación de campos sin rellenar.

Para enviar el formulario por correo es necesario PHP. En un principio se decidió utilizar PhpMailer, una clase de PHP que permite enviar correos y además también hace posible otras tareas como enviar archivos adjuntos.

Se desarrolló un archivo PHP con esta clase y en el servidor local (Xampp) funcionaba correctamente. Pero en el servidor web salía continuamente un error que por mucho que se intentó solucionar no fue posible por lo que se decidió pasar a utilizar la clase mail().

Si bien es cierto que mail() no permite realizar algunas tareas en este caso no supone un inconveniente puesto que solo es necesario que se envíe un correo electrónico con los datos del formulario rellenado.

Los datos del formulario son enviados al archivo PHP mediante el método "post" definido en la etiqueta "form". Este archivo PHP estará subido en el servidor FTP para que sea posible el envío de correos.

```php
<?php

//Información obtenida del formulario.
$nombre = $_POST['nombre'];
$actividad = $_POST['actividad'];
$persona = $_POST['persona'];
$url = $_POST['url'];
$emailfrom = $_POST['mail'];
$telefono = $_POST['telefono'];
$mensaje = $_POST['mensaje'];


//Para el envío en formato HTML
$header .= "Mime-Version: 1.0 \r\n";
$header .= "Content-Type: text/plain";

//Dirección del remitente
$header = 'From: '. $emailfrom ."\r\n";
$header .= "X-Mailer: PHP/" . phpversion() . " \r\n";


//Escribir el mensaje.
$cuerpo = "Formulario Enviado \r\n";
$cuerpo .= "Nombre del establecimiento: " . $nombre. "\r\n";
$cuerpo .= "Actividad del establecimiento: " . $actividad . "\r\n";
$cuerpo .= "Persona de contacto: " . $persona. "\r\n";
$cuerpo .= "Url del establecimiento: " . $url . "\r\n";
$cuerpo .= "Email: " . $emailfrom . "\r\n";
$cuerpo .= "Telefono: " . $telefono . "\r\n";
$cuerpo .= "Consulta: " . "" . "\r\n";
$cuerpo .= "". $mensaje. "\r\n";

//Dirección del destinatario.
$para = 'logrocel@gmail.com';

//Asunto del correo.
$asunto = 'Formulario Logrocel';

//Enviar el mensaje.
if (mail($para, $asunto, $cuerpo, $header))
{
echo 'MENSAJE ENVIADO CORRECTAMENTE';
} else {
echo 'Error enviando: ' . $mail->ErrorInfo;
};

?>
```

Figura 8.40. Código PHP para enviar el correo con los datos.

Este código lo primero que hace es recoger los datos e introducirlos en unas variables que luego utilizará para completar los parámetros necesarios para enviar el correo. La clase mail() es que la que permite el envío y consta de cuatro parámetros:

- Información del destinatario, los datos serán enviados al dueño de la aplicación (logrocel@gmail.com).
- Asunto del correo que en este caso se trata de "Formulario Logrocel".

- El contenido del mensaje, es decir, todos los datos rellenados por el usuario.
- La cabecera que contiene información de las versiones y el email del remitente.

Finalmente tras rellenarse el formulario y ejecutarse correctamente este código se recibe el siguiente correo:

Figura 8.41. Correo enviado con los datos del formulario.

Después de enviar el formulario el usuario recibirá un mensaje en la pantalla que le informará si el formulario ha sido enviado correctamente o por el contrario ha surgido un error en el envío. Para conseguir que este mensaje se muestre en la misma página y no redirija al usuario al archivo PHP se ha enviado el formulario mediante Ajax. De este modo el formulario es enviado sin recargar la página.

```
<script language="javascript">
$(document).ready(function() {

    // Interceptar el evento submit
    $('#form, #contactof').submit(function() {
         // Enviar el formulario usando AJAX
        $.ajax({
            type: 'POST',
            url: $(this).attr('action'),
            data: $(this).serialize(),
            success: function(data) {
                $('#result').html(data);

            }        })

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

Figura 8.42. Script para usar Ajax.

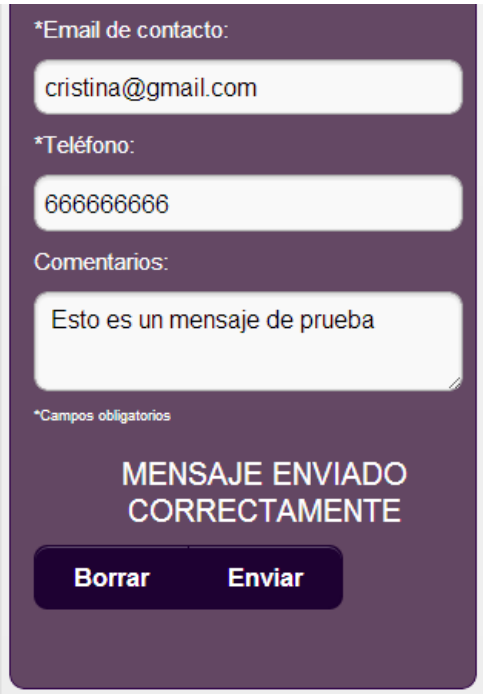

Figura 8.43. Mensaje de confirmación de envío del formulario.

8.4. Servidor local y web para test.

En el apartado de tecnologías se ha explicado en qué consiste Xampp, una plataforma que permite crear de forma rápida y sencilla un servidor local en el ordenador de modo que sea posible probar la aplicación en dicho servidor sin necesidad de subirla al servidor web.

Quizás lo más complicado sea instalar Xampp pero en la red existen numerosos tutoriales que facilitan esta tarea. A la hora de utilizarlo no tiene complicación, los archivos de la aplicación se deben introducir en la carpeta “htdocs” que se genera tras la instalación.

Para poner el servidor en funcionamiento se debe ejecutar Apache y MySQl en el panel de control. Una vez en funcionamiento y con todos los archivos en la carpeta “htdocs” simplemente hay que añadir en el navegador la siguiente ruta “localhost/nombre de la carpeta”.

Figura 8.44. Aplicación ejecutada desde el servidor local.

Se ha utilizado este servidor local para testear la aplicación durante el proceso de desarrollo antes de subirla a un servidor web.

La aplicación también ha sido subida a un servidor web, aunque el objetivo es realizar una aplicación híbrida y por ello no sería necesario realizar este paso. Pero se ha decidido subir la aplicación al servidor para probarla antes de compilar y además ofrecer otra alternativa a la aplicación híbrida.

Para subirla a un servidor web es necesario disponer de un dominio y un host para alojar la web. Se ha decidido utilizar www.webhost000.com que es un servidor web gratuito que permite almacenar 1500 Mb de datos y también permite el uso de PHP.

Tras rellenar todos los datos necesarios en la web para obtener un dominio, nos ofrece una dirección url que será necesaria para acceder a la aplicación web. En este caso la dirección es http://logrocel.net23.net/ .

Dreamweaver facilita el proceso de subir la aplicación al servidor web, es posible configurar este shoftware de modo que se conecte con el servidor y así subir los archivos al servidor directamente desde Dreamweaver. En la *figura 8.45* se puede observar en la esquina inferior derecha la conexión con el servidor web.

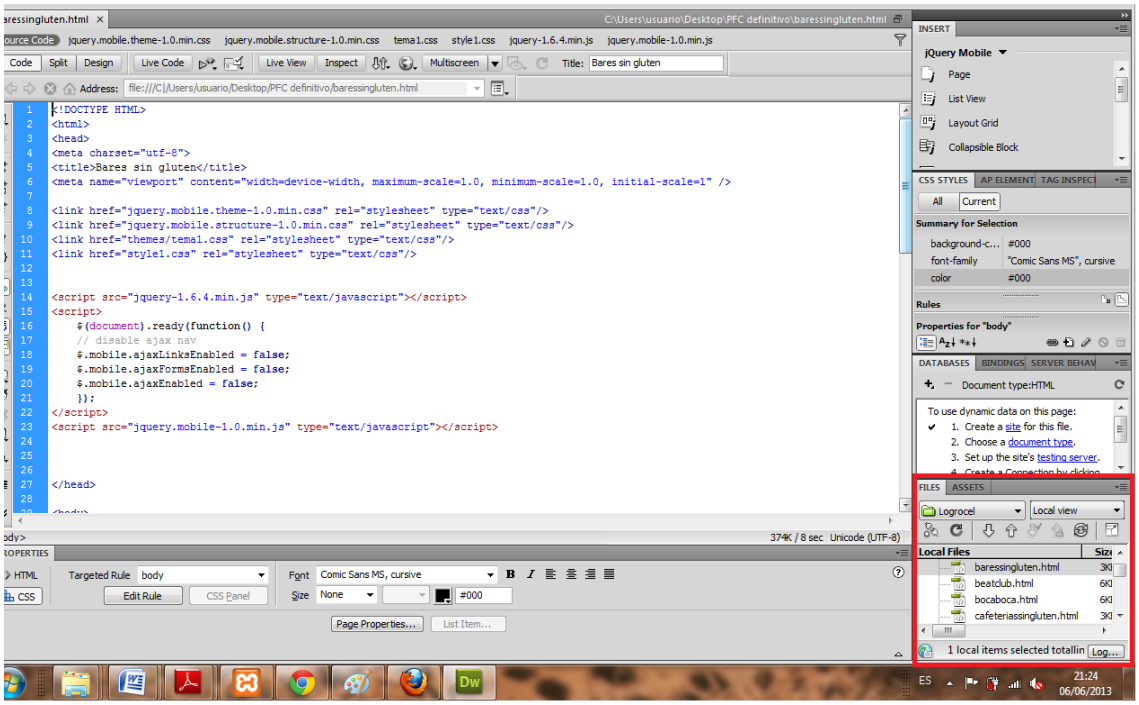

Figura 8.45. Conexión al servidor web mediante Dreamweaver.

8.5. PhoneGap Build.

En el apartado de estudio de los frameworks ya se ha explicado el servicio de compilador en la nube que ofrece PhoneGap. Este compilador lo que hace es convertir la aplicaciones basadas tecnologías web (HTML, CSS y JavaScript) en aplicaciones híbridas.

Con el desarrollo anterior se ha conseguido una aplicación web y ahora mediante el compilador PhoneGapBuild se obtendrá la aplicación híbrida. Este compilador empaqueta las aplicaciones web dentro de una aplicación nativa de cualquiera de las plataformas soportadas (Android, iOs, Blackberry, Windows Phone, Web Os, Symbian).

Este servicio es gratuito aunque solo es posible realizar una aplicación en caso de querer realizar más aplicaciones será necesario paga una cuota mensual. En este caso con la cuenta gratuita será suficiente. Además permite registrarnos mediante una cuenta de Adobe.

Una vez registrados solo hay que subir el código al compilador. Para subir el código es necesario cumplir con unas especificaciones. Se debe comprimir la carpeta que contiene todo los archivos CSS, HTML y JavaScript y además dentro de esa carpeta debe encontrarse un archivo “index.html” que será la página de inicio de la aplicación.

Una vez subida la carpeta, el compilador realiza su trabajo y poco a poco se va compilando para cada plataforma, el proceso no tarda mucho tiempo. En el caso de que surja algún error PhoneGapBuild indica que ha habido un error y especifica en qué consiste el error.

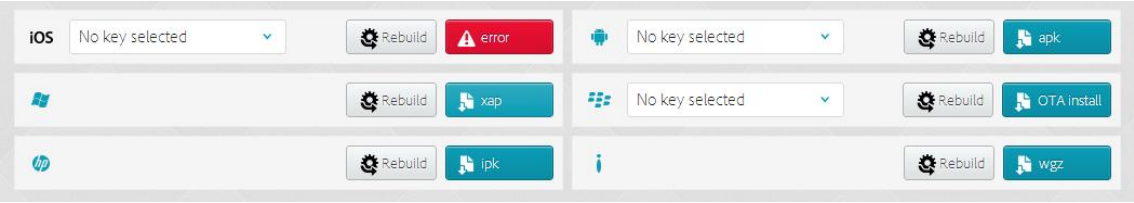

Figura 8.46. Resultado de compilar la aplicación en PhoneGapBuild.

En la *figura 8.46* se observa como indica error en la plataforma iOS y nos especifica que falta la clave. Como ya se explicó cuando se estudió PhoneGapBuild, para desarrollar en esta plataforma es necesario disponer de un código de desarrollador que otorga la empresa Apple.

Es posible descargar la aplicación resultante pulsando los botones azules de cada plataforma que vemos en la *figura 8.46.* Pero además PhoneGapBuild también proporciona un código QR que facilita el enlace de descarga de la aplicación, así se puede utilizar el lector de código QR del dispositivo y descargarla directamente en el dispositivo.

Figura 8.47. Código QR para descargar la aplicación.

Las aplicaciones creadas mediante este servicio se pueden configurar a través de un archivo “config.xml”. Este archivo permite especificar algunos datos,

como el autor, la descripción de la aplicación, habilitar funcionalidades del dispositivo, etc.

No es necesario crear este archivo ya que la aplicación funciona igual pero se ha desarrollado el archivo "config.xml" para añadir alguna especificación interesante. Se ha añadido una pequeña descripción de la aplicación, el nombre del autor, un icono personalizado y la versión de PhoneGap. De modo que queda de la siguiente forma:

Figura 8.48. Personalización de la aplicación.

Así cuando se instala la aplicación en el dispositivo aparece el icono de la aplicación y el nombre de ésta.

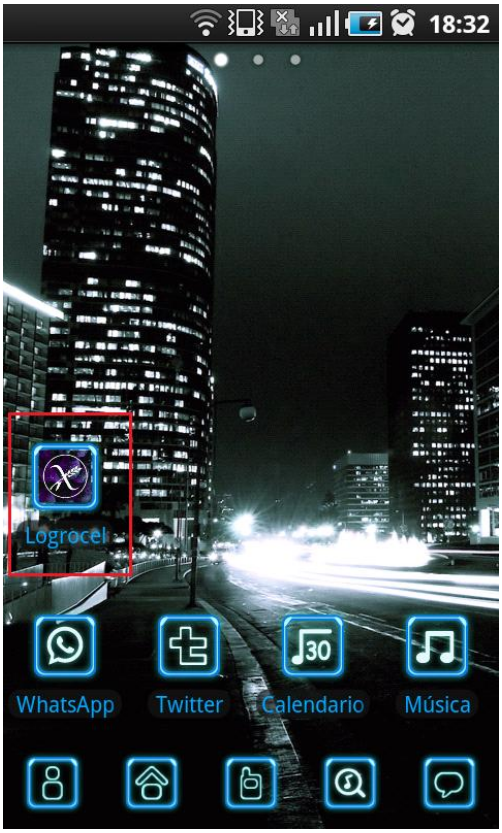

Figura 8.49. Icono de la aplicación en la pantalla del dispositivo.

8.6. Usabilidad móvil.

La usabilidad se refiere a la capacidad de un software de ser comprendido, aprendido, usado y ser atractivo para el usuario, en condiciones específicas de uso (ISO/IEC 9126).

La usabilidad no sólo depende del producto sino también del usuario y por ello un producto no es usable en sí mismo sino que tiene la capacidad de ser usado en un determinado entorno y por unos determinados usuarios.

Se trata de tener en cuenta al usuario durante todo el desarrollo de la aplicación y facilitarle al máximo su uso ofreciendo una navegación sencilla e intuitiva. Es posible conseguir esto diseñando páginas claras y fáciles de comprender, que tras realizar un primer vistazo resulten atractivas.

A continuación se va a mostrar aquellos aspectos que se han tenido en cuenta para hacer la aplicación más usable:

1. Se ha conseguido que el usuario pueda navegar por la aplicación con autonomía, no es necesario seguir unos pasos específicos.

2. En todo momento el color del texto predomina frente a los colores de fondo sin dificultar su compresión y del mismo modo el tamaño es suficientemente grande.

3. Los colores se han utilizado con precaución, teniendo en cuenta que hay muchas personas que tienen dificultades para distinguir colores, por ello no se han mezclado colores que puedan confundirles y se han especificado con texto los botones.

4. El diseño permanece constante y uniforme durante toda la aplicación sin despistar al usuario.

5. El usuario siempre podrá volver hacia atrás o regresar al menú inicial y deshacer las acciones que realice.

6. Los botones y las listas son grandes ya que la mayoría de usuarios utiliza sus dedos para interactuar con la aplicación.

7. Los botones se han localizado de forma coherente y ordenada. El botón de volver se encuentra en la parte izquierda con su correspondiente flecha orientada a la izquierda. Por otro lado, la barra de navegación inferior mantiene el orden del menú principal. Y los contenidos de todas las listas están ordenados de forma alfabética.

Figura 8.50. Botón "volver" colocado en la parte superior izquierda.

Figura 8.51. Lista en orden alfabético.

8. El formulario es sencillo, sin demasiados campos e incorpora un menú desplegable que se selecciona de forma táctil.

9. Los dos mapas que incluyen todos los locales permite al usuario acceder a toda la información de forma rápida, ahorrándole tiempo de búsqueda.

10. Se han incluido botones que permiten al usuario llamar por teléfono directamente al local sin necesidad de salir de la aplicación.

Figura 8.52. Botón para llamar directamente.

11. La información se muestra de forma ordenada y esquemática de modo que no se sobrecarga al usuario de información.

12. El logotipo de la aplicación es sencillo y fácil de comprender, y al igual que en la mayoría de los casos, está colocado en la parte superior izquierda.

13. La distancia hasta el objetivo no debe ser larga ya que dificulta el uso y aumenta el tiempo de búsqueda, en este caso el camino más largo se realiza con tres clicks.

14. Debido a que la aplicación es muy simple e intuitiva, puede ser utilizada desde un primer momento, no requiere ningún aprendizaje previo.

8.7. Google Play

Una vez obtenida la aplicación híbrida sólo queda distribuir dicha aplicación mediante las tiendas de aplicaciones. En este caso se ha distribuido en el canal de la plataforma Android, Google Play.

Al intentar subir el archivo *apk* Google Play informaba de que la aplicación no estaba firmada. Para poder distribuir una aplicación es necesario firmarla como una medida de seguridad para que sólo nosotros podamos modificar o actualizar la aplicación.

PhoneGapBuild nos permite firmar la aplicación pero antes debemos crear un KeyStore o lo que es lo mismo un almacén de claves.

Para ello se ha instalado JDK, el kit de desarrollo de Java y mediante la consola de Windows y el comando keytool se ha generado el KeyStore.

La línea de comandos completa es la que aparece en la *figura 8.53.* Donde my-*release-key* es el nombre del almacén que estamos creando, *alias_name* es el alias que posteriormente nos pedirá PhoneGapBuild y *–validity 10000* es el tiempo que será válido nuestro almacén en días.

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

figura 8.53. Línea de comandos para crear un KeyStore.

Una vez creado el almacén de claves simplemente se debe seguir las indicaciones de PhoneGapBuild y rellenar todos los campos requeridos.

Para poder subir aplicaciones a Google Play es necesario tener una cuenta de desarrollador que se obtiene registrándote como tal y pagando 25$. Por último, una vez subido el archivo de nuestra aplicación solo es necesario cumplir una serie de requisitos para poder publicar finalmente la aplicación.

- La aplicación no debe ocupar más de 50 MB.
- Indicar el título de la aplicación.
- Añadir una descripción de la aplicación.
- Mostrar al menos dos capturas de pantalla que sirvan como una vista previa de la aplicación y que cumplan con los requisitos que pide Google Play.
- Del mismo modo que las capturas de pantalla es necesario proporcionar un icono que cumpla las características que indica Google Play.
- Indicar el idioma y el país al que va destinada la aplicación.
- Especificar el tipo de aplicación, categoría y clasificación del contenido.
- Y por último aceptar algunas políticas de privacidad.

Una vez completada toda la información nuestra aplicación ya está publicada en Google Play y preparada para ser distribuida en este canal.

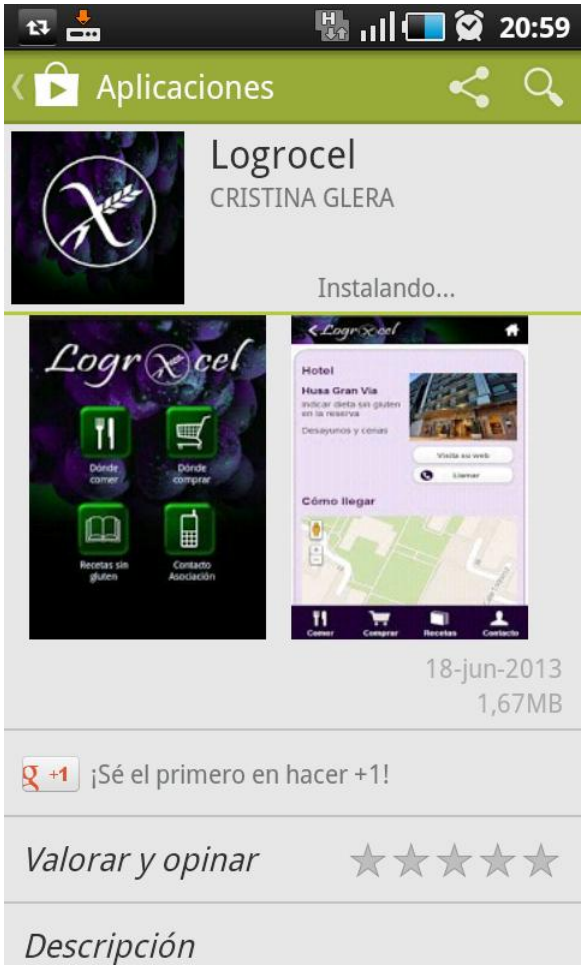

Figura 8.54. Aplicación publicada en la tienda de aplicaciones Google Play.

9. Conclusiones.

- Mediante el uso de HTML5, un lenguaje sin mucha complejidad, se han conseguido los objetivos planteados, desarrollar una aplicación híbrida que pueda ser instalada en el dispositivo.

- Jquery Mobile permite desarrollar aplicaciones sin mucha complejidad de código, con poco código se ha conseguido una aplicación atractiva y útil.

- La alternativa del compilador en la nube que ofrece PhoneGap ha permitido acercar la aplicación al mundo nativo sin necesidad de aprender los lenguajes específicos de cada plataforma.

- Mediante jQuery Mobile y HTML5 se ha conseguido desarrollar una aplicación web compatible con todos los navegadores. Cabe destacar que existen algunas diferencias entre navegadores, como por ejemplo algunos márgenes del diseño.

- Si bien es cierto que las aplicaciones creadas con jQuery Mobile pueden ser un poco lentas, tras convertirla en híbrida se ha mejorado este aspecto.

- Esta forma de desarrollar aplicaciones híbridas con el compilador en la nube permite obtener dos tipos de aplicaciones, la aplicación web inicial y la aplicación híbrida obtenida tras la compilación. De este modo se podrán ofrecer las dos alternativas.

- A pesar de las dificultades encontradas al principio para subir la aplicación a Google Play, hemos visto como también es posible distribuir las aplicaciones híbridas en las tiendas de aplicaciones.

10. Líneas futuras.

A continuación se van a exponer algunas de las posibles mejoras que se podrán realizar en un futuro.

- Cada vez son más lo establecimientos que ofrecen productos sin gluten por ello una de las posibles mejoras a realizar es añadir más establecimientos.

- También podríamos plantearnos ampliar la aplicación ofreciendo información no sólo de Logroño sino también de toda La Rioja.

- En lugar de utilizar PhoneGapBuild se podría utilizar PhoneGap o Titanium Appcelerator de modo que conseguiríamos una aplicación híbrida con más características nativas que podrían mejorar el rendimiento de la aplicación.

- Otro aspecto a mejorar en la aplicación es la geolocalización, ofreciendo mejor servicio para encontrar los establecimientos cercanos. Por ejemplo realizando una base de datos con todas las localizaciones y tras obtener la localización del usuario indicarle cuales son los más cercanos a su posición.

- Se podría tratar de publicar la aplicación en los canales de distribución de aplicaciones de otras plataformas además de Android.

11. Bibliografía.

- BUSCADOR DE GOOGLE.

-INTRODUCCIÓN A CSS – Javier Eguíluz Pérez, 2009.

-FUNDAMENTOS DE JQUERY – Rebeca Murphey, 2011.

- ADOBE.COM – Videotutoriales.

-JQUERYMOBILE.COM – Página oficial de Jquery Mobile.

-PHONEGAP.COM – Página oficial de PhoneGap.

-APPCELERATOR.COM – Página oficial de Appcelerator.

-PHP.NET – Página oficial de PHP.

- GLYPHISH.COM – Iconos para aplicaciones.

-MANUALES HTML5, PHP Y JAVASCRIPT – Varios autores.

- DEVELOPER.ANDROID.COM - Página oficial de Android.