Trabajo final de carrera

CalUOC: Calendario para dispositivos móviles vinculado al Campus virtual de la **UOC**.

Juanjo Navalón Flor

9 de junio de 2013

Índice de contenido

1. Introducción

El presente documento pretende describir la planificación del proyecto final de carrera de Ingeniería Técnica de Informática de Sistemas en el área de desarrollo de aplicaciones para dispositivos móviles.

La versión final de la aplicación se encuentra accesible desde Internet para cualquier navegador web en el siguiente enlace:

http://80.36.192.153:8008/~juanjo/tfc/prod/CalUOC.html

2. Descripción del proyecto

El proyecto consistirá en el desarrollo de una aplicación multiplataforma, es decir, capaz de ejecutarse en multitud de sistemas operativos de distintos fabricantes de dispositivos móviles, como son teléfonos inteligentes y tabletas.

La aplicación consistirá en un calendario conectado al Campus de la UOC[1], en el que seleccionando una fecha concreta, podremos visualizar la información relevante que hay en el Campus relacionada con esa fecha.

Para utilizar la aplicación el usuario podrá optar usar el navegador de su dispositivo para acceder al sitio web en el que se encuentre alojada la aplicación, o descargar un programa específico para su sistema operativo.

1 UOC: Universistat Oberta de Catalunya.

3. Justificación de la tecnología elegida

En la última década y especialmente, desde la aparición del teléfono inteligente *iPhone*[2] de Apple a mediados de 2007, seguido inmediatamente de terminales de distintos fabricantes equipados con el sistema operativo *Android*[3] de Google; hemos asistido a una espectacular proliferación de aplicaciones creadas específicamente para estos dispositivos. Esta tendencia se ha mantenido e incluso aumentado con la aparición de los *tablet*[4], que ejecutan sistemas operativos similares, pero disponen de características mejoradas, como son: pantallas de visualización de mayor tamaño, mayores capacidades de proceso y un uso todavía más amplio de la tecnología táctil para interactuar con las aplicaciones y el sistema.

No cabe duda de que a este hecho han contribuido las facilidades ofrecidas por los fabricantes a los programadores; por un lado, poniendo a su disposición las herramientas, documentación y acceso a las funciones propias de estos dispositivos, como son: GPS[5], acelerómetro[6], cámara de fotos, SMS[7], agenda de contactos, etc. y, por otro lado, facilitando la distribución de las aplicaciones a los usuarios finales mediante plataformas de contenidos, como App Store de Apple o Google Play, a cambio de un porcentaje de los beneficios generados por la venta de las aplicaciones o por la publicidad que pueden llevar incrustada.

Desde el punto de vista de los usuarios, con este modelo también disfrutan de grandes ventajas como son: menores precios de adquisición del software, facilidad de acceso e instalación de las aplicaciones y ciertas garantías de que las aplicaciones no contendrán componentes maliciosos como virus, *spyware*[8], etc., ya que los programas son sometidos a

2 *iPhone*. Teléfono inteligente con pantalla táctil fabricado por Apple.
3 *Android*: Sistema Operativo de Google empleado en *smartphones* y *tablets* de diversos fabricantes.
4 *Tablet* o tableta. Computadora portátil de mayor tamaño que un teléfono inteligente con pantalla táctil.
5 Global Positioning System: Sistema de geolocalización incorporado en dispositivos móviles que utiliza la señal de una red de satélites.
6 Acelerómetro: Dispositivo incorporado en terminales móviles y tabletas que permite detectar ciertos cambios en su posición relativa, como girar la pantalla.
7 *Short Message Service*: Servicio de envío y recepción de mensajes cortos de texto empleado en la telefonía móvil desde el estándar GSM.
8 *Spyware*: Programa espía que se instala sin el consentimiento explícito del usuario y recoge información personal que envía al atacante.

ciertos controles de calidad antes de ser publicados en sus "tiendas" de aplicaciones.

Con el tiempo también se ha visto que existen algunos inconvenientes, por ejemplo, para los programadores supone un inconveniente la gran diversidad de sistemas operativos y versiones que continúan apareciendo y que tienen que conocer, si no dominar, si quieren que sus desarrollos tengan la mayor difusión posible. El mantenimiento e implementación de nuevas funcionalidades en el software también se complica debido a que suele ser necesaria la intervención del usuario para instalar las nuevas versiones de la aplicación.

Otro factor negativo a tener en cuenta, causado por las incompatibilidades entre dispositivos, es un posible retroceso a inconvenientes similares a los que tuvieron lugar al final de los años 90, en que parte de los contenidos web solo eran accesibles de manera adecuada utilizando software propietario (como Internet Explorer de Microsoft y la tecnología Flash[9] de Adobe) que no cumplía con los estándares abiertos. Debido a su abrumadora mayor difusión ciertos proveedores de contenidos no consideraban rentable, o no tenían la capacidad suficiente, para adaptar su accesibilidad a usuarios que no disponían o no querían hacer uso de tecnologías propietarias.

La labor de la fundación Mozilla (liberando el código fuente del navegador Firefox) y el W3C, entre otros, contribuyó a volver a la filosofía abierta que sin duda fue la motivadora del auge original de Internet y la WWW, obligando a que las nuevas versiones de otros navegadores se comporten de la forma esperada cuando acceden a sitios web que cumplen con el estándar.

HTML5[10] y CSS3[11], junto con el uso de técnicas de programación web como el diseño adaptativo (*responsive design*[12]) permiten crear contenidos que utilizan las características más avanzadas de las versiones más recientes de los navegadores, como incorporar vídeos,

9 *Flash*: Tecnología software de Adobe para crear y manipular gráficos vectoriales mediante el lenguaje ActionScript. Para visualizar el contenido es necesario el componente software *Flash Player*.

10 *HyperText Markup Language* versión 5: Lenguaje declarativo que especifica el contenido y estructura de una página web.

11 Cascading Style Sheets versión 3: Lenguaje de hojas de estilos usado para describir el aspecto y formato de un documento HTML.

12 *Responsive Design*: Técnicas de diseño y desarrollo web que intenta adaptar el sitio web al entorno del usuario.

animaciones 3D y otros complementos multimedia; a la vez que los navegadores más antiguos o con capacidad limitada pueden seguir accediendo al contenido esencial, eso sí, con menor funcionalidad y vistosidad.

Los dispositivos que son objetivo del presente proyecto (*tablets* y *smart phones*) también disponen de navegadores web completamente funcionales que permiten aprovechar las tecnologías anteriormente comentadas. Pero quizá el interés por parte de los fabricantes de seguir manteniendo sus plataformas de distribución de aplicaciones, junto con los problemas de seguridad que se podrían presentar, hace que los desarrolladores de sistemas operativos para dispositivos móviles tengan ciertas reticencias a la hora de permitir que las aplicaciones ejecutadas en el navegador tengan acceso completo al hardware y funciones propias del dispositivo comentadas en el segundo párrafo de este apartado.

Nuevamente la fundación Mozilla, junto con diversos fabricantes de dispositivos y operadores de telecomunicaciones, con el futuro lanzamiento de el sistema operativo FirefoxOS[13] actualmente en fase de desarrollo, pueden crear las condiciones necesarias para que otros fabricantes incorporen las mayores funcionalidades posibles en los navegadores de los dispositivos móviles.

Por todo lo expuesto anteriormente y porque la aplicación que se pretende diseñar es totalmente dependiente del contenido que se encuentra en la web, en este caso el Campus de la UOC, se ha optado por diseñar una aplicación web que utilice los estándares abiertos. Para ello utilizaremos HTML5, ciertas librerías para el lenguaje JavaScript y el *framework*[14] *jQueryMobile*[15], ya que, todas ellas siguen estos estándares. Finalmente, para convertir la aplicación en una Mobile App que el usuario pueda instalar en su dispositivo, se utilizará la herramienta *PhoneGap*[16].

13 FirefoxOS: Sistema operativo móvil basado en el navegador de código abierto Firefox y con núcleo Linux.
14 *Framework*: Abstracción software que consiste en una plataforma de funcionalidad genérica y modificable utilizada para desarrollar aplicaciones con un patrón de diseño preestablecido.
15 *jQueryMobile*: *Framework* para el desarrollo de aplicaciones y páginas web adaptadas a dispositivos móviles.
16 *PhoneGap*: Herramienta *software* que permite convertir aplicaciones web en aplicaciones nativas para sistemas operativos de *smartphones* y *tablets*.

4. Objetivos del proyecto

Los objetivos que se pretenden alcanzar son los siguientes:

- Estudio de tecnologías web HTML5 y JavaScript[17], uso de *frameworks* y librerías orientadas a la programación web y su utilización para el acceso al API[18] de la UOC.

- Evaluar los distintos modelos y filosofías de desarrollo de software: en cascada, de prototipos, en espiral, por etapas, iterativo, etc. y escoger el más apropiado para este proyecto.

- Implementar una aplicación web accesible desde distintos modelos de dispositivos móviles inteligentes que permita el acceso a cierta información del usuario en el Campus de la UOC.

- Redactar el presente documento-memoria para describir los fundamentos teóricos, justificación y descripción de las distintas fases de desarrollo del proyecto.

- Elaborar una presentación que resuma el documento anterior y describa las funcionalidades y modo de uso de la aplicación desarrollada en el proyecto.

17 *JavaScript*: Lenguaje de programación interpretado concebido inicialmente para su uso en navegadores web.
18 *Application Programming Interface*: Especificación de cómo deben interactuar unos componentes software con otros.

5. Alcance del proyecto

Las funcionalidades que se pretenden alcanzar con la aplicación son las siguientes:

- Mostrar un calendario en el que se puede seleccionar cualquier fecha del semestre en curso.

- Una vez seleccionada la fecha, la aplicación nos permitirá elegir entre: mensajes personales, mensajes recibidos en los buzones de las asignaturas matriculadas o eventos programados en la agenda del Campus.

- La aplicación mostrará los mensajes y eventos relacionados con las opciones seleccionadas, permitiendo volver a cambiar los criterios de selección para reducir o ampliar la cantidad de información obtenida.

- Alojar la aplicación en una web accesible desde Internet para permitir el acceso desde navegadores de Internet disponibles tanto en dispositivos móviles como ordenadores portátiles y de sobremesa.

- Adaptar la aplicación para poder ejecutarla como programa nativo en Android e iOS[19] y evaluar las opciones disponibles para su distribución a los usuarios finales.

- Durante la fase de análisis e implementación se procurará utilizar técnicas y herramientas que permitan el posterior mantenimiento y ampliación de funcionalidades soportadas.

19 iOS: Sistema Operativo empleado en los productos iPhone e iPad de Apple.

6. Planificación

6.1. Tareas principales

- Tarea 1. Definición del proyecto.
- Tarea 2. Planificación del proyecto.
- Tarea 3. Investigación.
- Tarea 4. Análisis y diseño.
- Tarea 5. Implementación y pruebas.
- Tarea 6. Documentación.
- Tarea 7. Presentación.

6.2. Entregas

- PEC 1. Entrega del plan de trabajo. 11 de marzo de 2013 (aproximada).
- PEC 2. Primera entrega de control. 8 de abril de 2013 (aproximada).
- PEC 3. Segunda entrega de control. 20 de mayo de 2013 (aproximada).
- Entrega final. Memoria, presentación y aplicación. 10 de junio de 2013 (inamovible).

7. Conocimientos previos

Para la realización del proyecto se pondrán en práctica, tanto los conocimientos teóricos y prácticos adquiridos a lo largo de la carrera, como la experiencia profesional acumulada a lo largo de los últimos años desarrollando tareas de ingeniería en el sector de las telecomunicaciones.

Para una buena planificación será importante evaluar los conocimientos previos de que ya se dispone, para dedicar la mayor parte del tiempo de investigación a las partes del proyecto para las que la información previa sea más escasa.

- En la fase de planificación se ha empleado una aplicación de software libre llamada GanttProject para la representación de diagramas de Gantt. Ya se han utilizado herramientas similares en el ámbito laboral.

- Para llevar a cabo la investigación a lo largo del proyecto se utilizarán las técnicas de búsqueda de información en Internet, utilizadas en la mayoría de las asignaturas cursadas y en la actividad profesional y personal habitual. También se utilizan los libros enumerados en la bibliografía en formato electrónico y papel.

- En la fase de análisis se hará uso de los conocimientos adquiridos y las herramientas utilizadas en la asignatura Ingeniería del Software.

- Para la implementación de la aplicación serán de gran utilidad los conocimientos adquiridos en las asignaturas de programación como son Fundamentos de Programación y Programación Orientada a Objetos. Para implementar la aplicación se utilizará el lenguaje JavaScript, que no se ha visto en estas asignaturas, pero tiene grandes similitudes con Java[20] y C[21] que sí se han empleado anteriormente.

20 Java: Lenguaje de programación orientado a objetos, cuyo código generado se compila en bytecodes que son interpretados en una máquina virtual disponible para la mayoría de los sistemas operativos.

21 C: Lenguaje de programación compilado de alto nivel que permite el uso de funciones de bajo nivel cercanas al hardware.

8. Diagrama de Gantt

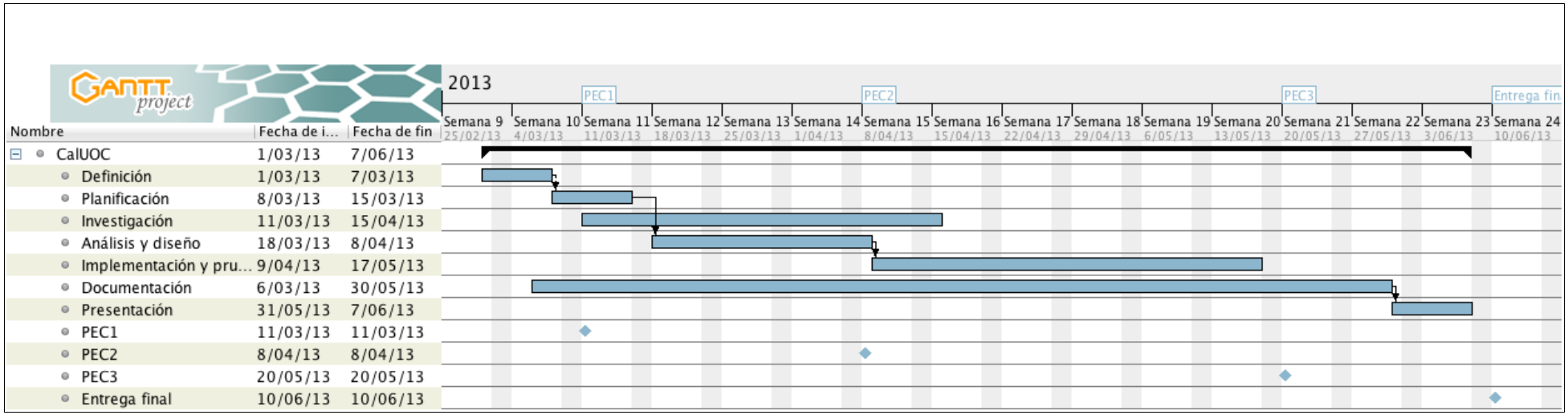

Figura 1. Diagrama de Gantt de la planificación del proyecto.

9. UOC Open API

Una parte importante del proyecto está basada en las funcionalidades que ofrece la Open API de la UOC. La documentación se puede encontrar en el enlace:

http://blogs1.uoc.es/developer.

Para poder utilizar la API es necesario realizar una solicitud de una API KEY en la que se tendrán que indicar ciertos datos personales y una breve descripción del proyecto en el que se va a utilizar la Open API. También será necesario leer y aceptar las condiciones de uso y aspectos legales que atañen a su utilización.

Una vez que la solicitud ha sido aprobada, se recibirá un correo electrónico en el que se indican un identificador "*Client*" y un código "*Secret*".

La descripción completa de la sintaxis de las peticiones y el formato de los datos obtenidos de todos los servicios disponibles actualmente en la Open API:

http://blogs1.uoc.es/developer/documentacio/uoc-public-api.

Algunos de las operaciones RESTful[22] que se utilizarán en este proyecto son:

- URI: /api/v1/user. Para obtener los datos personales del usuario.

- URI: /api/v1/mail/messages/unread. Recupera los mensajes no leídos del buzón personal.

- URI: /api/v1/classrooms. Obtiene las aulas de las asignaturas en la que está matriculado el usuario.

22 RESTful: Librería que cumple los patrones de diseño de la arquitectura REST.

9.1. OAuth 2.0

El proceso de autenticación de la aplicación y del usuario se lleva a cabo mediante los mecanismos descritos el el protocolo abierto OAuth2[23].

La especificación de OAuth2 viene descrita en la RFC6749 de IETF (*The OAuth 2.0 Authorization Framework*) disponible en línea en http://tools.ietf.org/html/rfc6749. En concreto para este proyecto se utiliza el perfil de cliente "*user-agent-based application*" y el mecanismo "*Implicit Grant*" que se muestra en el apartado 4.2 de la RFC[24].

El siguiente diagrama de secuencia muestra el proceso de autorización para el uso del API de Google desde una aplicación JavaScript como la de este proyecto:

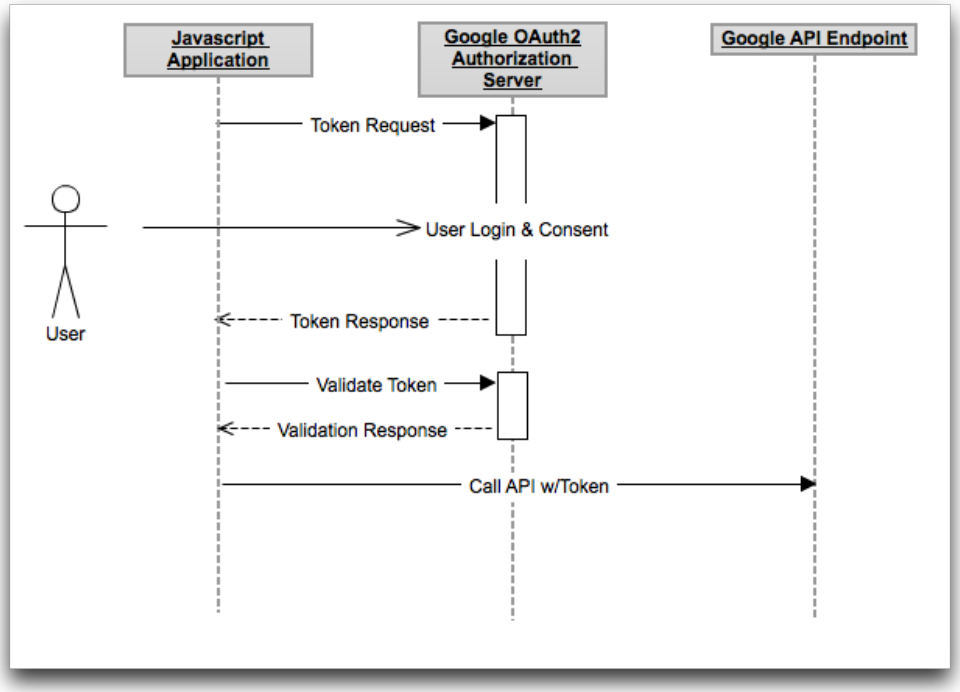

Figura 2. Diagrama de secuencia de acceso al API de Google desde JavaScript.
Referencia: https://developers.google.com/accounts/docs/OAuth2

23 *Open Authorization* versión 2: Protocolo abierto para la autorización segura del uso de una API abierta.
24 *Request for Comments*: Publicación técnica del IETF e ISOC que describe algún estándar concreto empleado en Internet.

A continuación se muestra el ejemplo a la hora de inscribirse un usuario en la web de Udacity, una organización educacional que pone al servicio de la comunidad gran cantidad de cursos mayoritariamente tecnológicos en la modalidad MOOC (*Massive Open Online Course*).

1. Al utilizar el aplicativo "*Sign Up*" para darse de alta, se nos ofrece la posibilidad de hacerlo mediante las redes sociales Facebook o Google+:

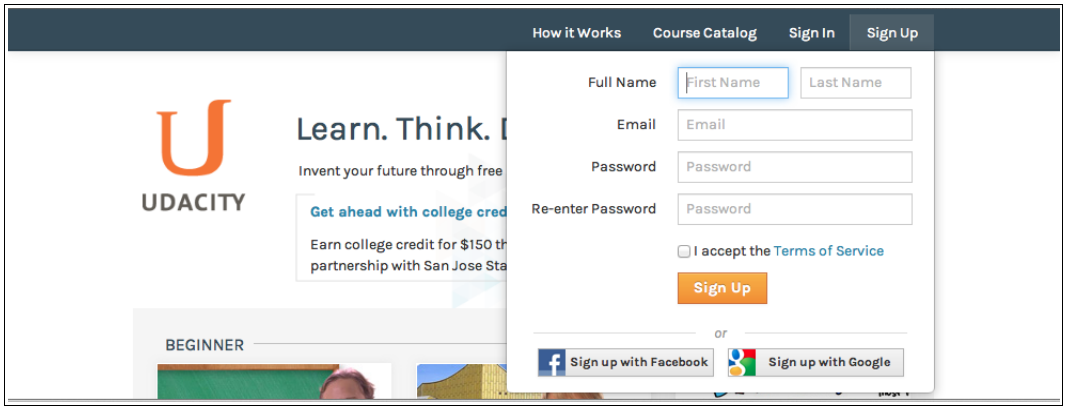

Figura 3. Registro en aplicación web con posibilidad de autenticación delegada.

2. Utilizando, por ejemplo, la opción de Facebook, aparece una nueva ventana del navegador (*pop up*) pidiendo autorización para que Udacity pueda usar los recursos del API de Facebook que se muestran, en este caso: visualizar el perfil público, la lista de amigos y la dirección de correo electrónico:

Figura 4. Solicitud de consentimiento del usuario.

3. Esta ventana proviene directamente, mediante conexión segura HTTPS[25], de los servidores de Facebook y el usuario puede verificarlo visualizando el certificado de seguridad, con lo que se puede tener la seguridad de que los datos de acceso en ningún momento están disponibles para la entidad en la que nos estamos dando de alta, en este caso Udacity.

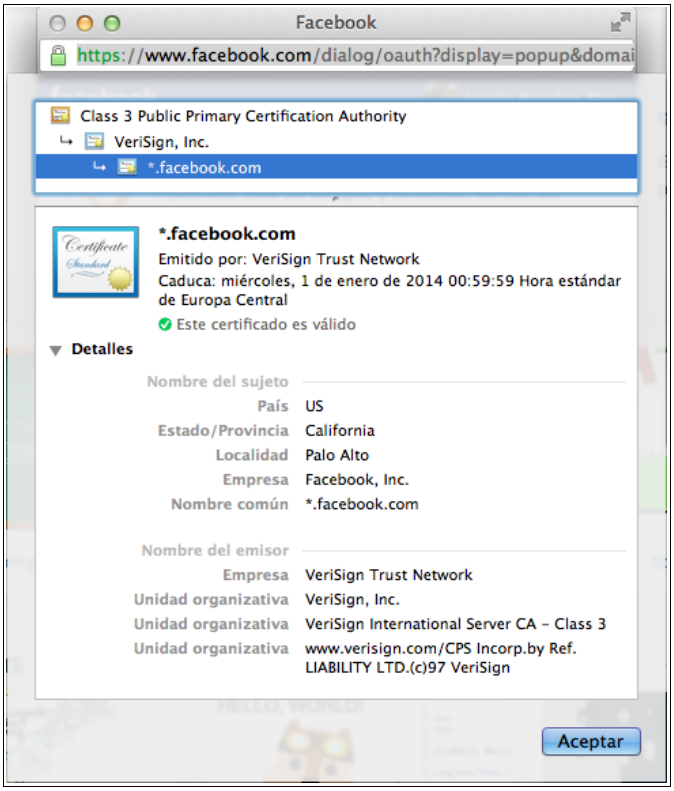

Figura 5. Certificado de seguridad del servidor de autenticación.

Algunas aplicaciones web, además de delegar la autenticación de usuarios en ciertas redes sociales (en este caso *Facebook*), también solicitan permiso para utilizar los recursos de la API de la red social para publicar mensajes, fotografías, realizar cambios de estado, etc.

Otra aplicación interesante es la función, entre otras, que ofrece Google a las aplicaciones autorizadas, de guardar información en el espacio Google Drive del usuario de la aplicación. El proceso de autenticación en este caso se muestra en la figura 2.

25 *Hypertext Transfer Protocol Secure*: Versión segura del protocolo HTTP.

9.2. Arquitectura REST

Representational State Transfer (REST[26]). Se trata, más que de un protocolo o especificación, de un conjunto de pautas de diseño de aplicaciones distribuidas que necesitan comunicarse a través de una red. Originalmente fue propuesta por Roy Fielding y la describe en el documento disponible en línea:

http://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm.

Los elementos en los que se basa son:

- Recursos: Los servicios que se espera obtener del servidor.
- Identificadores de estos recursos: URI, URL, URN[27].
- Representaciones de la información intercambiada: XML[28], JSON[29], etc.
- Metadatos de los recursos y representaciones: Tipo de documento, Fecha de modificación, etc.

Se basa en operaciones cliente-servidor *stateless*. La comunicación es iniciada mediante un comando HTTP desde el cliente para acceder a un "recurso" que se encuentra en el servidor y el servidor está esperando a recibir peticiones de los clientes. Se dice que es stateless, es decir, sin información de estado, porque las peticiones son independientes y no es necesario guardar información entre transacciones, lo que permite una mayor escalabilidad del servidor y tolerancia a fallos.

26 *Representational State Transfer*: Arquitectura de software empleada en sistemas distribuidos en red como la *World Wide Web*.
27 URI, URL, URN: *Uniform Resource Identifier, Locator, Name*.
28 XML: *eXtensible Marckup Language*. Lenguaje superconjunto de HTML utilizado para el intercambio de información entre sistemas heterogéneos.
29 JSON: *JavaScript Object Notation*. Formato ligero para intercambio de datos que utiliza la notación de objetos de JavaScript.

10. Open Web Platform

Es una colección de tecnologías web desarrolladas y mantenidas, entre otras, por las siguientes organizaciones de estandarización:

- El World Wide Web Consortium (W3C) es una comunidad internacional fundada en 1994 por Tim Berners-Lee (creador del primer navegador y servidor web en el CERN), que reúne a las compañías tecnológicas más importantes del mundo. Su motivación, según su declaración de principios, es "guiar la Web hacia su máximo potencial a través del desarrollo de protocolos y pautas que aseguren el crecimiento futuro de la Web. Debajo tratamos importantes aspectos de este objetivo, los cuales promueven la visión del W3C de **Web Única**": http://www.w3c.es/Consorcio/mision.

- IETF: Internet Engineering Task Force. Es una institución abierta a cualquier individuo interesado y sin ánimo de lucro creada en los EEUU en 1986. Es la encargada de gestionar los estándares que definen la familia de protocolos de Internet (TCP/IP) http://www.ietf.org. Sus publicaciones más conocidas son las RFC en las que se basan los estándares definidos por el IETF.

- Ecma International. Es también una organización de estandarización sin ánimo de lucro que se fundó en 1961 como ECMA (European Computer Manufacturers Association) y cambió a su nombre actual en 1994 para enfatizar su ámbito internacional. Surgió por la necesidad de estándares que permitieran a los ordenadores de los distintos fabricantes interactuar entre sí focalizando su actividad en lenguajes de programación. Actualmente su ámbito de actuación es más amplio, pero el aspecto que nos interesa es su especificación del lenguaje ECMAScript. http://www.ecma-international.org/publications/standards/Ecma-402.htm.

A la hora de diseñar una aplicación web es importante mantener separados la estructura, la presentación y la funcionalidad de la aplicación:

10.1. HTML

Es el primer archivo de la aplicación que se carga en el navegador y donde se le indica el resto de archivos necesarios (css para la presentación y js para la funcionalidad) que el navegador tiene que cargar para ejecutar la aplicación. La estructura viene determinada por el uso de etiquetas (tags) como son: <head>, <body>, .

Los navegadores disponen de una estructura de datos llamada DOM[30], donde cargan todos los elementos definidos en el archivo html.

A continuación se muestra una página de ejemplo de una aplicación que utiliza jQuery Mobile como la del presente proyecto:

```
1. <!DOCTYPE html>
2. <html>
3.
4. <head>
5.   <title>Page Title</title>
6.
7.   <meta name="viewport" content="width=device-width, initial-scale=1">
8.
9.   <link rel="stylesheet" href="http://code.jquery.com/mobile/1.3.1/jquery.mobile-1.3.1.min.css" />
10.  <script src="http://code.jquery.com/jquery-1.9.1.min.js"></script>
11.  <script src="http://code.jquery.com/mobile/1.3.1/jquery.mobile-1.3.1.min.js"></script>
12.</head>
13.
14.<body>
15.  <!-- Start of first page -->
16.  <div data-role="page" id="foo">
17.
18.      <div data-role="header">
19.          <h1>Foo</h1>
20.      </div><!-- /header -->
21.
22.      <div data-role="content">
23.          <p>I'm first in the source order so I'm shown as the page.</p>
24.          <p>View internal page called <a href="#bar">bar</a></p>
25.      </div><!-- /content -->
26.
27.      <div data-role="footer">
28.          <h4>Page Footer</h4>
29.      </div><!-- /footer -->
30.
31.  </div><!-- /page -->
32.
33.  <!-- Start of second page -->
34.  <div data-role="page" id="bar">
```

30 Document Object Model: Estructura de datos para representar e interactuar con los elementos de un documento HTML.

```
35.
36.       <div data-role="header">
37.           <h1>Bar</h1>
38.       </div><!-- /header -->
39.
40.       <div data-role="content">
41.           <p>I'm the second in the source order so I'm hidden when the page loads. I'm just shown
   if a link that references my id is beeing clicked.</p>
42.           <p><a href="#foo">Back to foo</a></p>
43.       </div><!-- /content -->
44.
45.       <div data-role="footer">
46.           <h4>Page Footer</h4>
47.       </div><!-- /footer -->
48.   </div><!-- /page -->
49.</body>
50.
51.</html>
```

Para usar jQuery se hará uso intensivo de los atributos data-* en las etiquetas (definidos en la versión 5 de HTML) para hacer referencia a los elementos cuya presentación o comportamiento se pretende modificar. Los atributos data-role son utilizados por jQuery Mobile para que los elementos tengan el comportamiento esperado por el uso de este *framework*.

10.2. CSS

En los archivos con extensión css es donde se definen las características de presentación del documento. En el presente proyecto se utilizará la presentación por defecto de una aplicación jQuery Mobile, pero es posible utilizar este framework con presentaciones personalizadas o diseñar nuevas mediante el uso de ThemeRoller de jQuery:

http://jqueryui.com/themeroller/

Durante la fase de implementación será necesario modificar el archivo css para cambiar el color de los elementos Count Bubble para que aparezcan en rojo el número de mensajes no leídos.

10.3. JavaScript

JavaScript es un lenguaje de programación interpretado un tanto atípico. Se diseñó inicialmente para su uso en el navegador Netscape con el objeto de dotar de cierta inteligencia a las páginas web, que hasta entonces eran estáticas, permitiendo realizar validaciones de formularios, dotar de movimiento a ciertos elementos, interactuar con *applets* de Java, etc. Posteriormente Microsoft implementó una versión llamada JScript en su navegador Internet Explorer y Adobe hizo lo propio para su software Adobe Flash con el nombre de Action Script. Todas estas versiones pueden considerarse dialectos del lenguaje ECMAScript desarrollado en 1996 y adoptado como estándar en 1998 por la organización "*International Organization for Standardization*" como ISO/IEC16262:1998.

Se considera débilmente tipado, ya que no se declara el tipo de variables y el ámbito de las variables es distinto a otros lenguajes como Java o C, ya que las variables son accesibles en toda la función aunque se hayan definido dentro de un bloque delimitado por corchetes. También son accesibles fuera de la función mediante el uso de cierres (*closures*). Esto consiste básicamente en definir una función que devuelve otra función e implícitamente se está retornando la función con todo su entorno.

Es orientado a objetos pero no existe el concepto de clase como en en Java o C++. Su modelo de herencia de objetos proviene del lenguaje Self, y, aunque dispone de varias formas de implementar la herencia, su uso recomendado es a través de prototipos como se explica en el capítulo 5 del libro JavaScript The Good Parts de Douglas Crockford publicado por O'Relly en 2008.

11. Librerías y *frameworks*

11.1. jQuery

jQuery es una librería que proporciona un conjunto de funciones que simplifican el uso de el lenguaje JavaScipt, a la vez que se ocupa de solucionar muchas de las incompatibilidades entre navegadores web: http://api.jquery.com. Algunas de las ventajas que aporta:

- Facilidad para manipular elementos en una página web (DOM), sirve de ayuda en las peticiones Ajax y sus respuestas, y para controlar eventos (zona activa, click, focus).

- Mejorar la usabilidad e interacción con las páginas web dinámicas y crear una mejor experiencia de usuario UX (*User eXperience*).

- Fácil de aprender, no es necesario dominar JavaScript para utilizarlo.

Utiliza selectores al estilo de CSS para consultar y manipular elementos del DOM.

11.2. jQueryMobile

Los mismos desarrolladores de jQuery han creado un framework basado en jQuery y adaptado para su uso en dispositivos móviles. http://api.jquerymobile.com

11.3. Modelo Vista Controlador

Cuando crece la complejidad de las estructuras de datos manejadas por la aplicación es conveniente utilizar un modelo de diseño que sea más óptimo para este cometido que el proporcionan jQuery y jQuery Mobile. Existen multitud de librerías encaminadas a esta función entre las que se encuentras Backbone.js y AngularJS y que pueden coexistir con jQuery Mobile para mantener el aspecto de la aplicación.

11.4. Codiqa

Herramienta de diseño WYSIWYG basada en el *framework* jQuery Mobile:

http://blog.codiqa.com

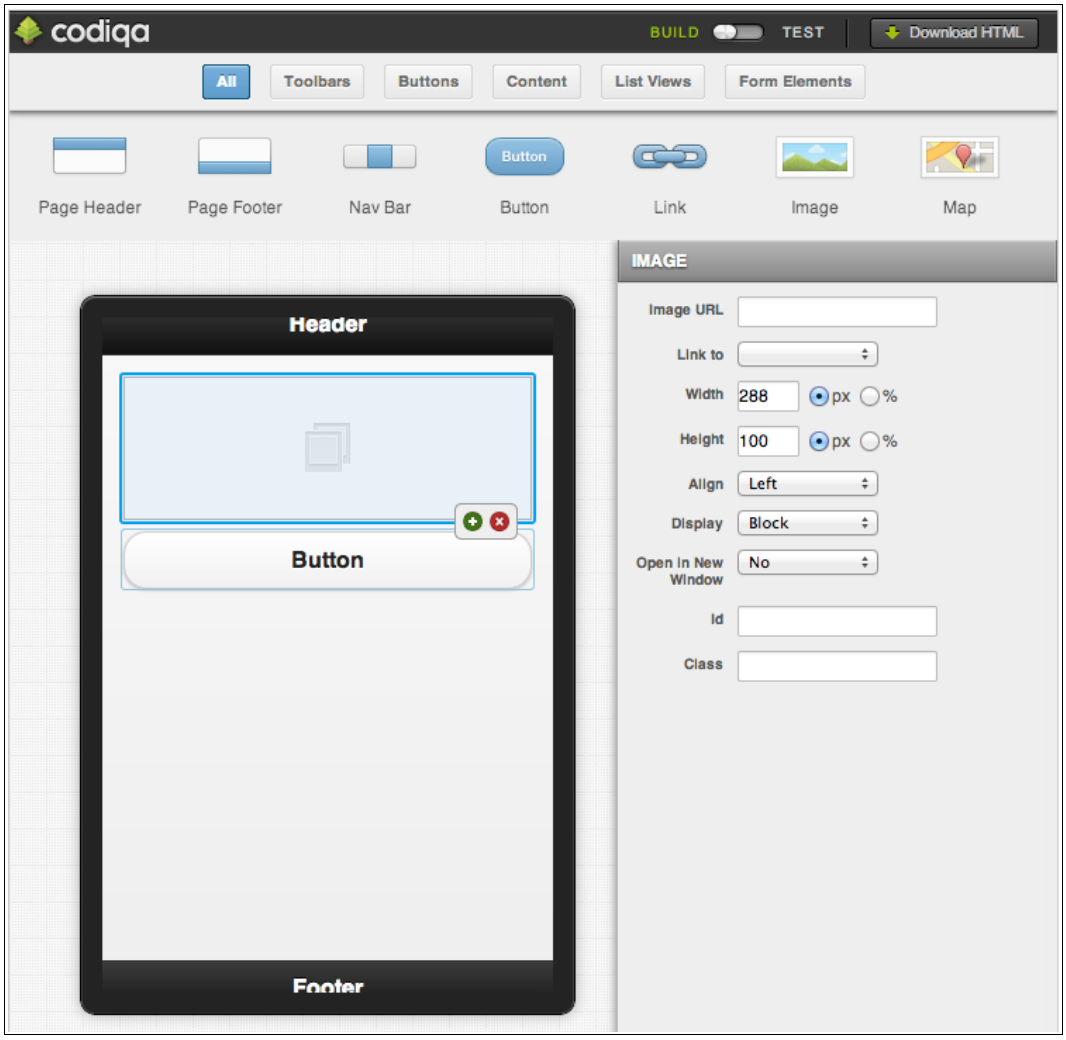

Figura 6. Herramienta *online* codiqa.

11.5. PhoneGap

Es un conjunto de librerías que permiten adaptar una aplicación web diseñada para dispositivos móviles en una aplicación nativa para los sistemas operativos que soporta, entre ellos Android, iOS, Windows Phone, Blackberry OS, etc.

http://docs.phonegap.com/en/2.7.0/index.html

12. Configuración del entorno de pruebas

En una primera fase del desarrollo y gracias al carácter abierto de las tecnologías empleadas, para realizar pruebas solo es necesario disponer de un editor de textos y un navegador web actualizado con soporte para JavaScript.

En este caso, para disponer de un entorno más cómodo y realista para el desarrollo, se ha optado por utilizar un servidor web, lo que además permitirá cargar la aplicación en los navegadores de varios dispositivos móviles.

12.1. Activación del servidor web

Para llevar a cabo las pruebas durante la implementación de la aplicación se ha optado por configurar un servidor web Apache en un ordenador con sistema operativo Mac OS X versión 10.5.8.

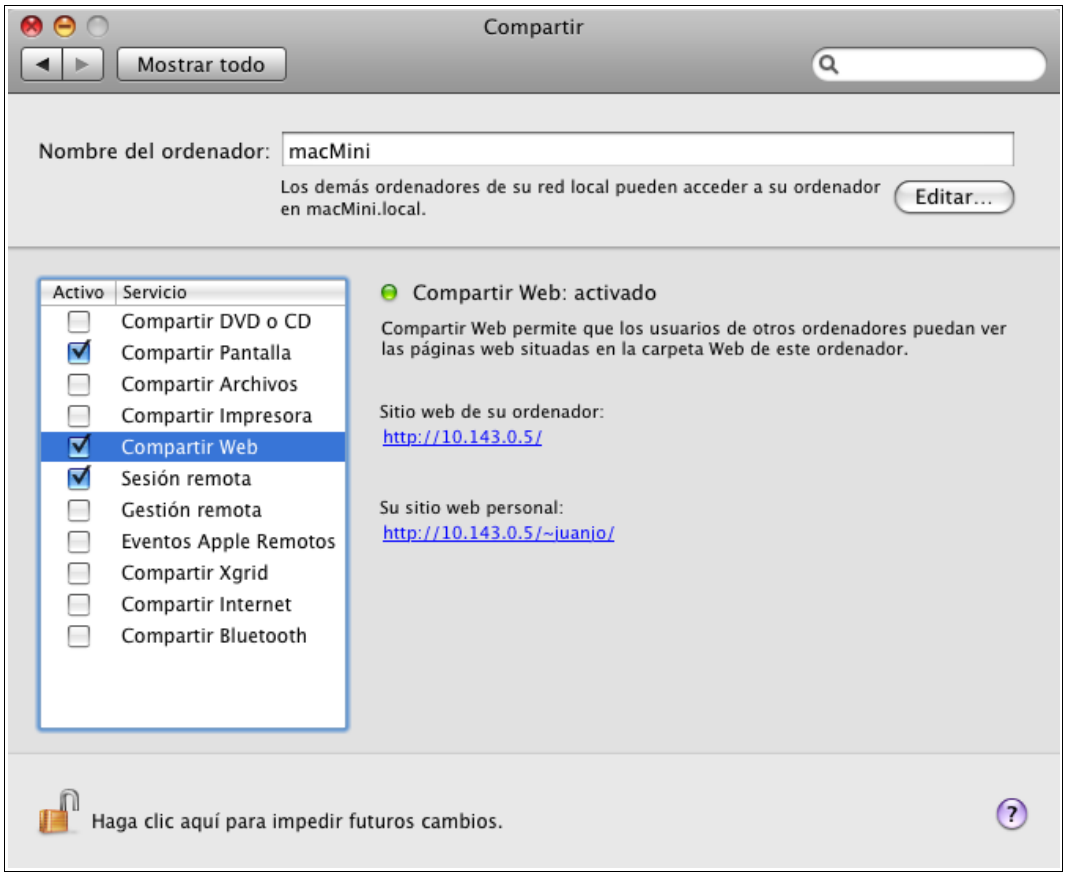

Figura 7. Activación del servidor Apache en Mac OS X.

> **Nota**: En versiones posteriores de OS X la opción de "Compartir Web" no aparece en las preferencias del sistema, pero el servidor Apache sigue estando integrado en el SO. En la web es posible encontrar diversas fuentes para poder activarlo. Por ejemplo: http://osxdaily.com/2012/09/02/start-apache-web-server-mac-os-x

También activamos la función "Sesión remota" para poder subir y editar los archivos remotamente desde Internet.

12.2. Configuración NAT

Aprovechando la circunstancia de disponer de una dirección IP pública fija, se configura el modem/router ADSL para que acepte conexiones desde Internet. Apartado "*Securtiy"*, opción "*Port Forwarding"*. De esta forma no solo será posible el acceso al servidor desde la LAN

doméstica, sino que también será posible acceder desde localizaciones remotas a través de Internet.

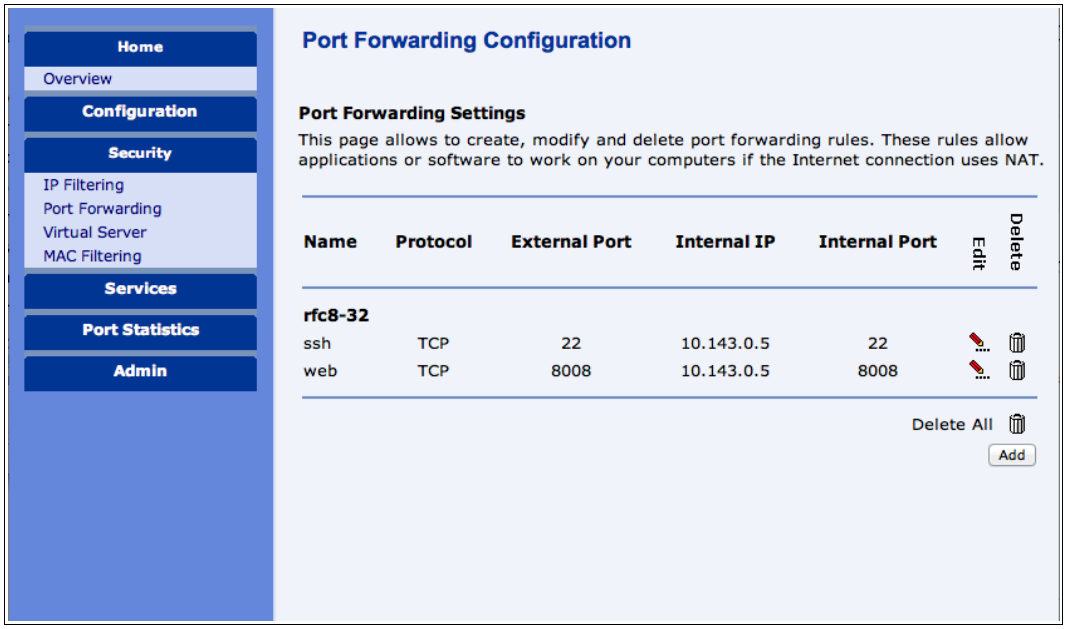

Figura 8. Configuración de puertos en el modem/router ADSL

Configuramos también el puerto TCP 22 para poder editar remotamente los archivos de la aplicación mediante sftp y poder administrar el servidor desde un terminal ssh, por si es necesario reiniciar algún servicio o la máquina entera.

12.3. Editor de código

La elección de un editor para modificar los archivos html, css y js, es una decisión bastante personal y es preferible escoger una herramienta con la que nos sintamos cómodos que otra que disponga de un gran número de funciones que no vamos a utilizar. En este caso se ha optado por el programa TextWrangler, que es la versión gratuita de BBEdit de la compañía BareBones: http://www.barebones.com/products/textwrangler/

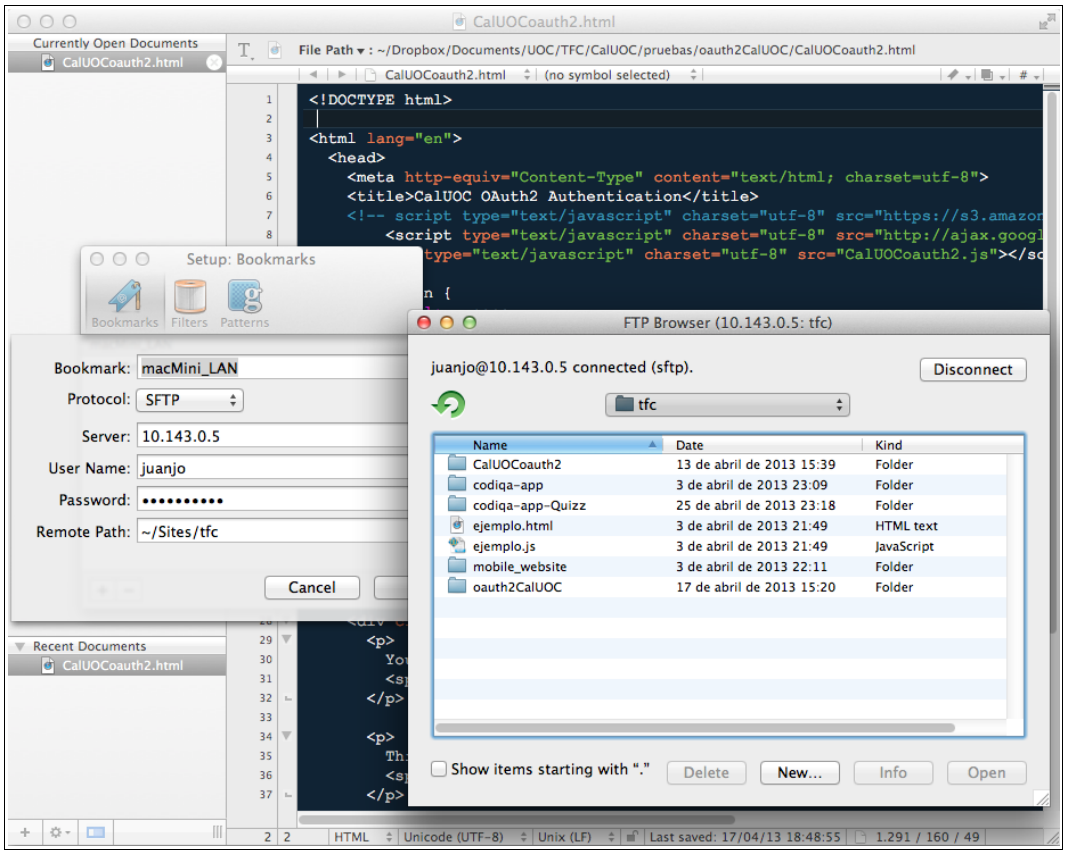

Figura 9. Configuración de *TextWrangler* para editar archivos remotos (sftp).

Las funciones más destacables que se van a utilizar son:

- Posibilidad de editar directamente los ficheros que se encuentran en un servidor remoto. Para usar esta función tenemos que configurar el servidor remoto como se

muestra en la figura a través del menú: File → New → FTP/SFTP Browser.

- Resaltado de sintaxis del código que se está editando. Para esta función basta con salvar el fichero con la extensión correspondiente: html, css o js, en nuestro caso.

- Colapsar/Expandir los bloques de código. Mediante el menú View → Collapse Enclosing Fold, podemos hacer que solo se muestre el nombre de la función en lugar de todo su contenido.

12.4. Configuración de navegadores

- Firefox. En este navegador existe la posibilidad de instalar un plugin que nos ayudará en las tareas de debug de la aplicación. *Plugin* Firebug. http://getfirebug.com/faq/

Figura 10. Depuración con FireBug.

- Chrome en Mac OS X. Inspeccionar elemento.

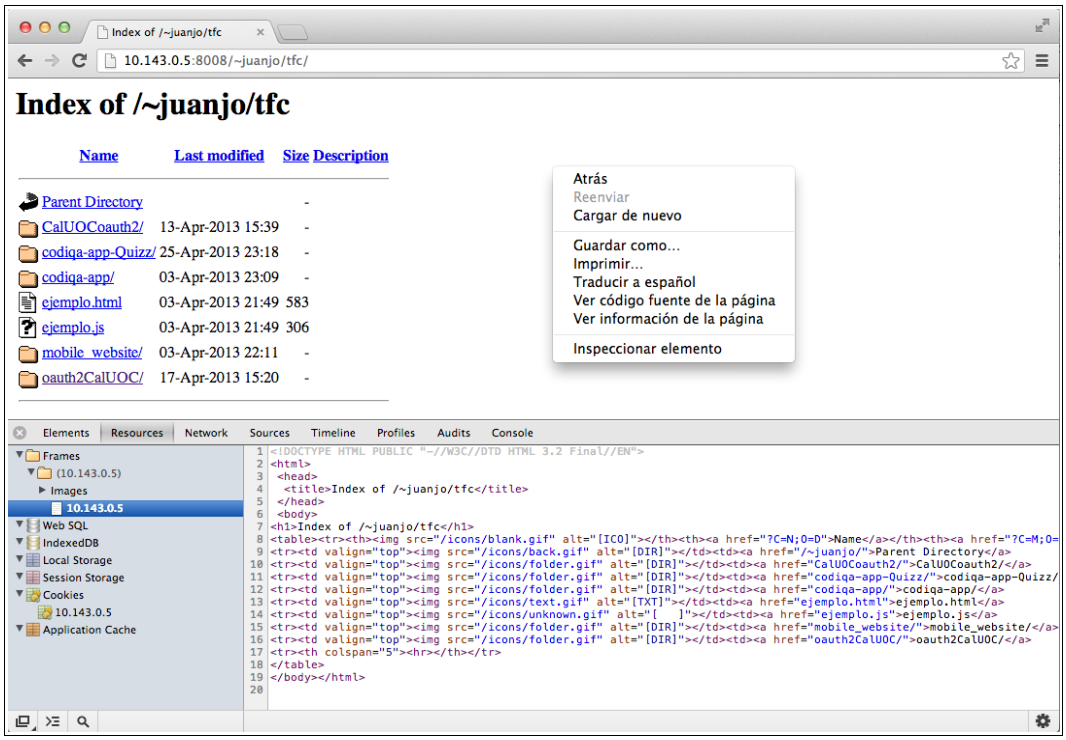

Figura 11. Depuración con inspector de Chrome (Mac OS X).

- Safari en Mac OS X. El navegador de Apple incluye, a partir de la versión 4.0 tanto para Windows como para Mac OS X, la herramienta *Web Inspector* que permite analizar, depurar y optimizar aplicaciones web.

  http://developer.apple.com/library/safari/#documentation/AppleApplications/Conceptual/Safari_Developer_Guide/1Introduction/Introduction.html

  Para poder utilizarlo solo es necesario activar la opción "Mostrar el menú de Desarrollo en la barra de menús" que se encuentra en las opciones generales de las preferencias de la aplicación.

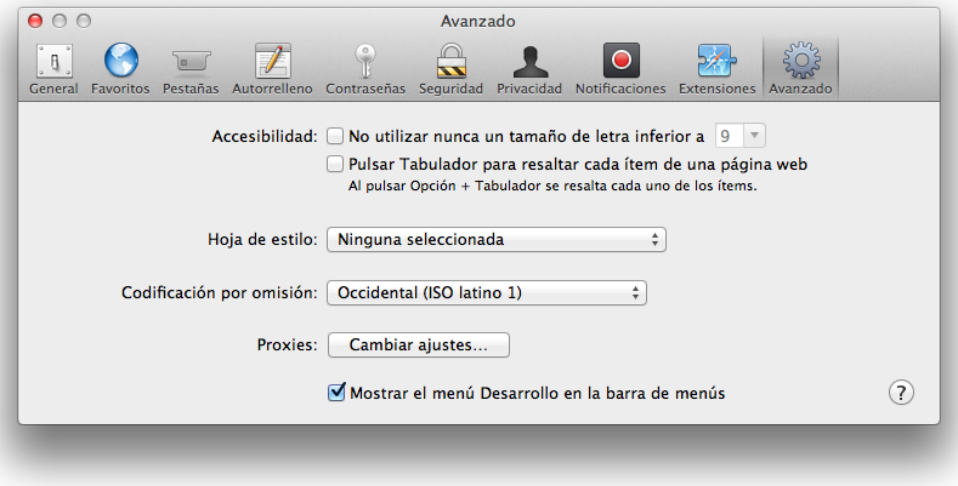

Figura 12. Activación de *Web Inspector* en Safari (Mac OS X).

A partir de este momento se puede acceder fácilmente a la herramienta pulsando con el botón derecho en cualquier elemento de una página web y seleccionando la opción "Inspeccionar elemento" en el menú contextual.

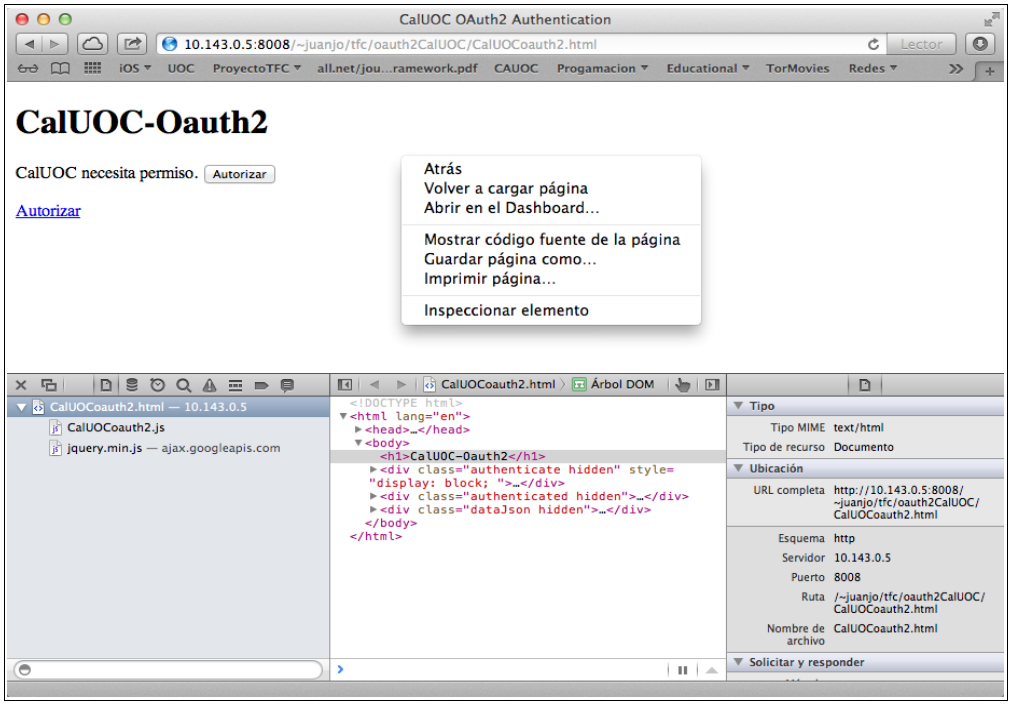

Figura 13. Depuración con inspector de Safari (Mac OS X).

- Chrome en Android (Nexus 7). Además de probar la aplicación como lo haría el usuario final, existe la posibilidad de activar la depuración web desde un ordenador https://developers.google.com/chrome-developer-tools/docs/remote-debugging

Figura 14. Depuración con inspector de Chrome Android (Nexus 7).

- Safari en iOS (iPhone 4). En este caso solo lo utilizaremos para probar el aspecto y funcionamiento.

Figura 15. Visualización en Safari iOS (iPhone 4).

13. Análisis y diseño

En un primer momento y debido a las características del planteamiento de cualquier proyecto final de carrera: plazo limitado en el tiempo, no interacción con el cliente, etc. se optó por un ciclo de vida en cascada. Este modelo permitiría ir avanzando en las fases de desarrollo sin tener que volver a fases anteriores.

Pero ciertas circunstancias inherentes a este proyecto en concreto, como son: el desconocimiento a priori de las funcionalidades disponibles en el API de la UOC, falta de experiencia en el desarrollo de aplicaciones *software*, conveniencia de disponer en un corto plazo de tiempo de prototipos que muestren las funcionalidades que se esperan de la aplicación final y que puedan ser perfeccionados progresivamente conforme avanza el desarrollo del proyecto; han sido determinantes a la hora de decantarse por un ciclo de vida iterativo con prototipos.

En la obra *Engineering Long-Lasting Software*, Armando Fox y David Patterson, con un enfoque eminentemente práctico, describen la gestión de proyectos "*Agile*" para el desarrollo de aplicaciones SaaS (*Software as a Service*) como es el presente proyecto, teniendo en cuenta que la parte correspondiente al servidor ya está implementada en la Open API de la UOC y la aplicación CalUOC solo se encargará del lado del cliente. Las pautas propuestas para el desarrollo de la parte cliente de una aplicación SaaS se tratan en el capítulo 11 del libro.

El tipo de modelo de desarrollo elegido también permitirá el futuro mantenimiento, debido a los posibles cambios que experimente el lado del servidor, y la ampliación de funcionalidades que podrán incorporarse a la aplicación después de que haya concluido el proyecto actual.

La relativa sencillez de las estructuras de los datos que manejará la aplicación nos permite prescindir del diseño de diagrama de clases, ya que todos los datos se almacenarán directamente en el DOM del navegador.

13.1. Descripción funcional general

Cuando la aplicación se pone en funcionamiento, lo primero tendrá que utilizar el identificador "*Client*" en el campo "*client_id*" de un comando GET de HTTP[31] para obtener un *token*[32], una vez que el usuario se ha identificado con las credenciales del Campus de la UOC.

El *token* obtenido tendrá validez temporal, aunque podrá renovarse sin necesidad de que el usuario vuelva a autenticarse en el Campus. La aplicación lo tendrá que incluir en cada una de las peticiones de tipo REST que realice al servidor de recursos donde se encuentra alojada la API de la UOC.

Una vez que se ha obtenido el *token* de acceso al API, la aplicación comenzará a enviar peticiones al servidor de la UOC al que a partir de ahora nos referiremos como servidor OpenAPI. Estas peticiones se realizan mediante comandos GET y POST de HTTP siguiendo el modelo REST en el que está basada la API.

A partir de este momento el usuario podrá volver a la pantalla de búsqueda para limitar la información que aparece en la pantalla de resultados. También se dispondrá de una pantalla de ayuda en la que se podrá volver a solicitar la autorización y la aplicación volverá a solicitar la información al servidor OpenAPI.

31 *Hypertext Transfer Protocol*: Protocolo de nivel de aplicación empleado en las transferencias de datos en la *World Wide Web*.
32 *Token*: Identificador de una sesión de intercambio de mensajes.

13.2. Diagrama de secuencia

Para el análisis necesario para el desarrollo de la aplicación se ha realizado el siguiente diagrama de secuencia:

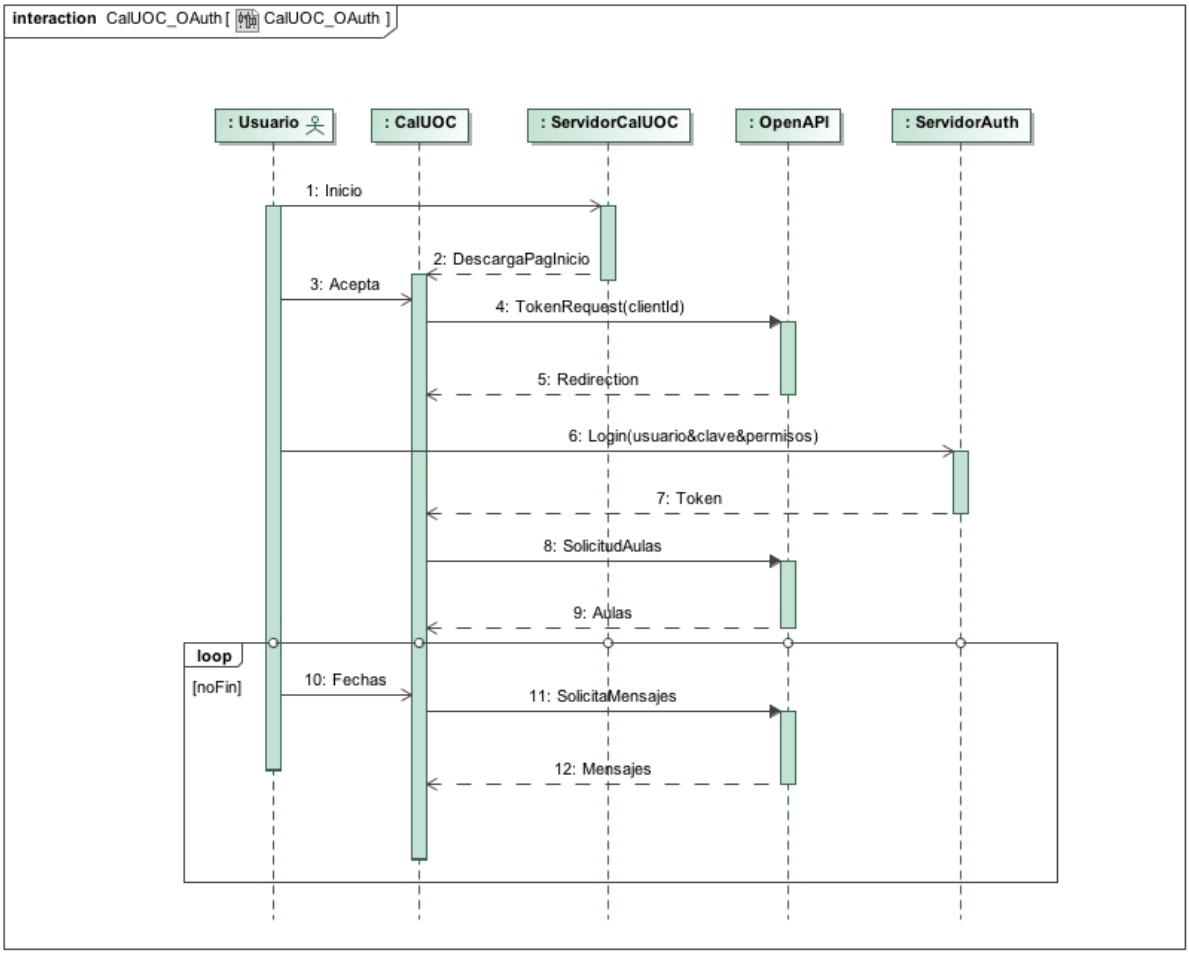

Figura 16. Diagrama de secuencia.

13.3. Descripción del diagrama de secuencia

A continuación se describen los mensajes representados en el diagrama de secuencia:

1. El proceso se inicia cuando el usuario accede a la página web alojada en el servidor de pruebas para obtener la aplicación. En este caso, mediante una entrada en los marcadores del navegador, un acceso directo, o tecleando directamente en la barra de direcciones la URL correspondiente al servidor web de pruebas:

   http://80.36.192.153:8008/~juanjo/tfc/CalUOC.

2. En el navegador aparece una advertencia indicando que el usuario tendrá que validarse en el servidor de la UOC para dar permiso a la aplicación a descargar información del Campus.

3. El usuario tendrá tendrá que pulsar el botón Autorizar para dar su consentimiento y continuar.

4. En ese momento la aplicación envía una petición al servidor de pruebas donde está alojada la Open API de la UOC, en este caso la URL que se utilizará es: http://denver.uoc.es:8080/webapps/uocapi/oauth/authorize. Junto con la petición se enviarán como parámetros del comando GET de HTTP: el identificador de cliente, los permisos que necesita la aplicación para acceder al Campus (READ, READ_MAIL y READ_BOARD), y la url a la que se volverá una vez finalizado el proceso de autenticación.

5. El servidor del API devolverá a la aplicación CalUOC una orden de re-dirección hacia el servidor de autenticación de la UOC: https://cv.uoc.edu/webapps/cas/login.

6. En este momento al usuario se le presenta una página web mediante conexión segura

en la que se le solicita su usuario y contraseña del Campus y que confirme los permisos de lectura mencionados anteriormente.

7. Si el proceso de autenticación tiene éxito, el servidor de autenticación de la UOC devuelve a la aplicación CalUOC el *token* que será necesario para acceder a los servicios del API.

8. La aplicación CalUOC solicita el listado de aulas correspondientes a las asignaturas en las que el usuario está matriculado. A partir de este momento todas las peticiones se realizan enviando el *token* como parámetro de los comandos GET de HTTP. Las peticiones se realizan mediante AJAX para no tener que cargar nuevamente la aplicación en el navegador del usuario.

9. El servidor del API devuelve el listado de aulas en formato JSON.

10. La aplicación configura las opciones en el menú que ofrece el usuario a partir de la información que ha recibido, modificando el DOM mediante funciones jQuery, y solicita al usuario un rango de fechas y ofrece la posibilidad de seleccionar el correo personal, tablones y foros de las aulas, tutoría, etc.

11. La aplicación CalUOC envía las peticiones AJAX necesarias para obtener la información solicitada por el usuario.

12. Con la información recibida se vuelve a modificar la información presentada al usuario y se le permite obtener correos a partir de los asuntos que se muestran, volver a realizar la consulta desde el principio, o finalizar la sesión. Para realizar nuevas consultas ya no será necesario solicitar el *token* ni las aulas, ya que esta información no cambia a lo largo de la sesión.

13.4. Descripción de pantallas

En esta fase del análisis del proyecto podríamos haber optado por utilizar una herramienta de dibujo para representar los scketches o mock-ups de las pantallas en que consistirá la aplicación, pero en este caso nos decantamos por utilizar el primer prototipo creado con la herramienta codiqa.

La utilización de la herramienta codiqa anteriormente descrita nos permitirá disponer de un prototipo totalmente funcional desde el inicio del desarrollo de la aplicación. Evidentemente, en una primera fase, el prototipo no será capaz de conectarse a la Open API, ni obtener información del Campus, pero servirá para tener una idea de las pantallas, las transiciones entre ellas y la disposición de los elementos básicos: botones, cabeceras, listas, etc.

El servicio codiqa nos permite compartir un enlace para mostrar el prototipo a un posible cliente o colaborador. Pero en nuestro caso optamos por descargar los archivos html, js y css y copiarlos en el servidor web que hemos configurado para realizar las pruebas.

Una vez que tenemos instalado el prototipo en el servidor web, ya podemos comprobar su funcionamiento cargando el siguiente enlace en el navegador de un dispositivo móvil o un ordenador de sobremesa:

http://80.36.192.153:8008/~juanjo/tfc/desa/CalUOC.html

A continuación se muestran las pantallas del prototipo en un iPhone 4:

- Pantalla de **inicio**: En la primera pantalla que aparece una imagen con el logotipo de la UOC que contiene un enlace para acceder a la versión móvil del Campus. Debajo hay un botón que en la aplicación final nos permitirá hacer login en el servidor de autorización de la UOC, pero en esta primera versión solo tiene la función de pasar a la siguiente pantalla.

Figura 17. Pantalla inicio.

- Pantalla de **selección**: En la siguiente pantalla que aparece tenemos un campo de entrada tipo fecha, un menú *collapsible* con las opciones, Correo, Aulas y Agenda. Dentro de la opción Aulas aparecerán las asignaturas y, para cada asignatura, una opción Tablón y otra Foro cada una con un *Count Bubble* que mostrará el número de mensajes no leídos.

Figura 18. Pantalla selección.

Figura 19. Pantalla selección (fecha).

- Pantalla de **resultados**: Una vez que hemos seleccionado una opción en la pantalla anterior llegamos a los resultados que se muestran en una *Collapsible List*, cuyos elementos tiene por título la fecha y el asunto y al expandirse muestran el contenido del mensaje.

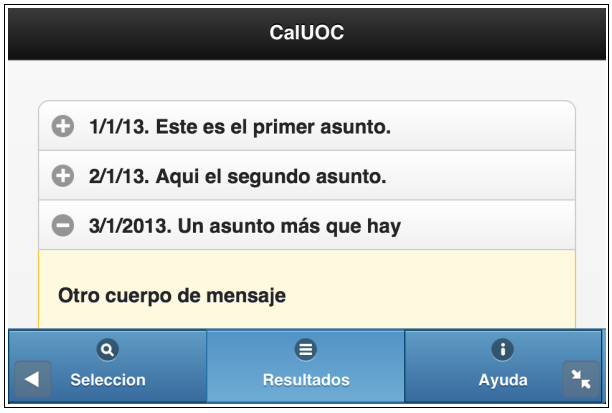

Figura 20. Pantalla resultados.

- Pantalla de **ayuda**: En cualquier momento podremos acceder a la pantalla de ayuda, donde se mostrarán unas breves indicaciones para usar la aplicación y un enlace para volver a autenticarnos y volver a cargar todos los datos del Campus. A esta pantalla también se accederá en caso de error, por ejemplo, un fallo de red mientras se están cargando los datos.

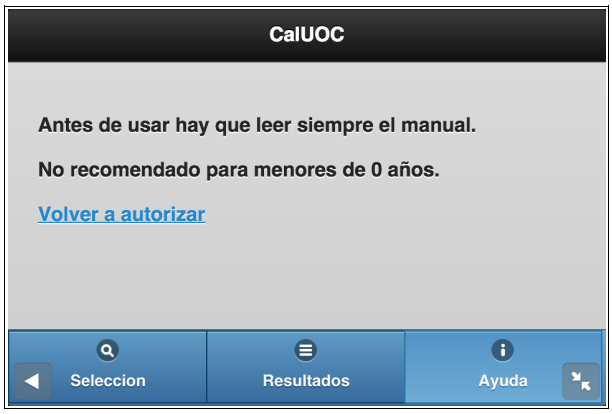

Figura 21. Pantalla ayuda.

14. Implementación y pruebas

La versión final de la aplicación está accesible en:

http://80.36.192.153:8008/~juanjo/tfc/prod/CalUOC.html

Para programar la aplicación partimos del prototipo obtenido con codiqa, básicamente los archivos html y css, los cuales tendremos que modificar, y creamos los archivos de código JavaScript CalUOCoauth.js (para el proceso de autorización) y CalUOC.js (para el resto de la aplicación). Los archivos se irán modificando en el servidor web de pruebas para poder realizar pruebas cada vez que se modifica algo.

En la mayoría de los lenguajes de programación, como C o Java, el ámbito (*scope*) de las variables es de bloque, es decir, todas las variables definidas dentro de un bloque no son visibles fuera de él, en cambio en JavaScript el ámbito de los parámetros y las variables es de función, no son visibles fuera de la función pero sí lo son en cualquier sitio dentro de la función. Normalmente en otros lenguajes se recomienda que las variables se declaren justo antes de ser utilizadas, pero en JavaScript es mejor declararlas al principio.

14.1. Autorización

El proceso de autorización en el servidor de la UOC está implementado en el archivo CalUOCoauth.js disponible en el código fuente.

Primero se definen como variables globales las direcciones de los servidores de autenticación y del API, así como el código de cliente obtenido por correo electrónico para la aplicación, ya que estos datos serán necesarios en las distintas funciones de la aplicación.

El resto de código en CalUOCoauth.js solo se ejecutará una vez que esté cargada toda la

aplicación en el navegador, para ello utilizaremos la función de jQuery $(function () {}.

A continuación tenemos que distinguir si se ha accedido a la página directamente o mediante una redirección desde el servidor de autenticación, en cuyo caso, sabemos que ya estamos autenticados y se puede continuar con el resto de la ejecución. Para esto pasamos como parámetro la URL de la página a la función extractToken que se muestra a continuación:

```
1. //Extraer el token del url devuelto por el servidor de autenticacion
2. var extractToken = function(hash) {
3.   var match = hash.match(/access_token=([\w-]+)/);
4.   return !!match && match[1];
5. };
```

En el caso de que la función no devuelva el token como resultado, todavía no estamos autenticados y hay que modificar el atributo href del botón de la página inicial para que nos dirija al servidor de autenticación con la instrucción de jQuery:

```
$('#autoriza').attr("href", authUrl);
```

En el caso de que la función sí sea capaz de extraer el token de la dirección de la página podremos continuar con la ejecución llamando a la función obtenerDatos() que se encuentra implementada en el archivo CalUOC.js. El token obtenido se incorporará en las cabeceras de todas las peticiones AJAX que se realicen a partir de ahora:

```
$.ajaxSetup({headers: {"Authorization": "Bearer " + token}});
```

El problema con el que nos encontramos en este punto, era que el servidor no devolvía el token en la URL aunque introdujéramos correctamente el usuario y contraseña del Campus de la UOC. Usando la herramienta Firebug del navegador Firefox observamos que se obtiene el error:

> "XMLHttpRequest cannot load http://denver.uoc.es:8080/webapps/uocapi/oauth/token.

Origin http://localhost:8008 is not allowed by Access-Control-Allow-Origin.”.

El problema se debe a que todos los navegadores actuales implementan una medida de seguridad llamada “Same Orign Policy”, que impide que el código JavaScript de una página web pueda obtener datos, mediante peticiones AJAX, de un servidor alojado en un dominio diferente al dominio del que procede el código que realiza la petición. Esto se hace para evitar posibles ataques mediante la inyección de código con XSS (cross-site scripting).

Para resolverlo nos ponemos en contacto con el servicio técnico de la Open API de la UOC en la dirección de correo openapi@uoc.edu proporcionada por el consultor. La respuesta obtenida fue se activó la funcionalidad “Cross-Origin Resource Sharing” (CORS), para permitir que el código JavaScript, descargado desde el servidor doméstico de desarrollo, fuera capaz de realizar peticiones AJAX al servidor OpenAPI.

http://www.w3.org/wiki/CORS

14.2. Obtención de datos

Algo a tener en cuenta es que las funciones empleadas para hacer llamadas AJAX no están pensadas para devolver valores. A pesar de la característica de JavaScript que permite que las variables de una función sean accesibles desde las funciones interiores, en caso de una función AJAX como puede ser getJSON las modificaciones de las variables no se producen en el momento esperado, ya que la ejecución continúa mientras la función espera la respuesta del servidor.

En el caso de este proyecto esta circunstancia nos impedía obtener los mensajes en los tablones de cada aula, ya que para esto antes teníamos que obtener el identificativo de cada aula. Para solucionarlo hubo que deshabilitar el funcionamiento asíncrono por defecto de las peticiones AJAX con la siguiente instrucción:

```
$.ajaxSetup({ async: false });
```

Con esta configuración podemos observar en el debugger del navegador Chrome que, a partir de la obtención de las aulas, las peticiones AJAX se realizan de modo asíncrono, es decir, no se realiza una nueva petición hasta que se ha obtenido la respuesta anterior.

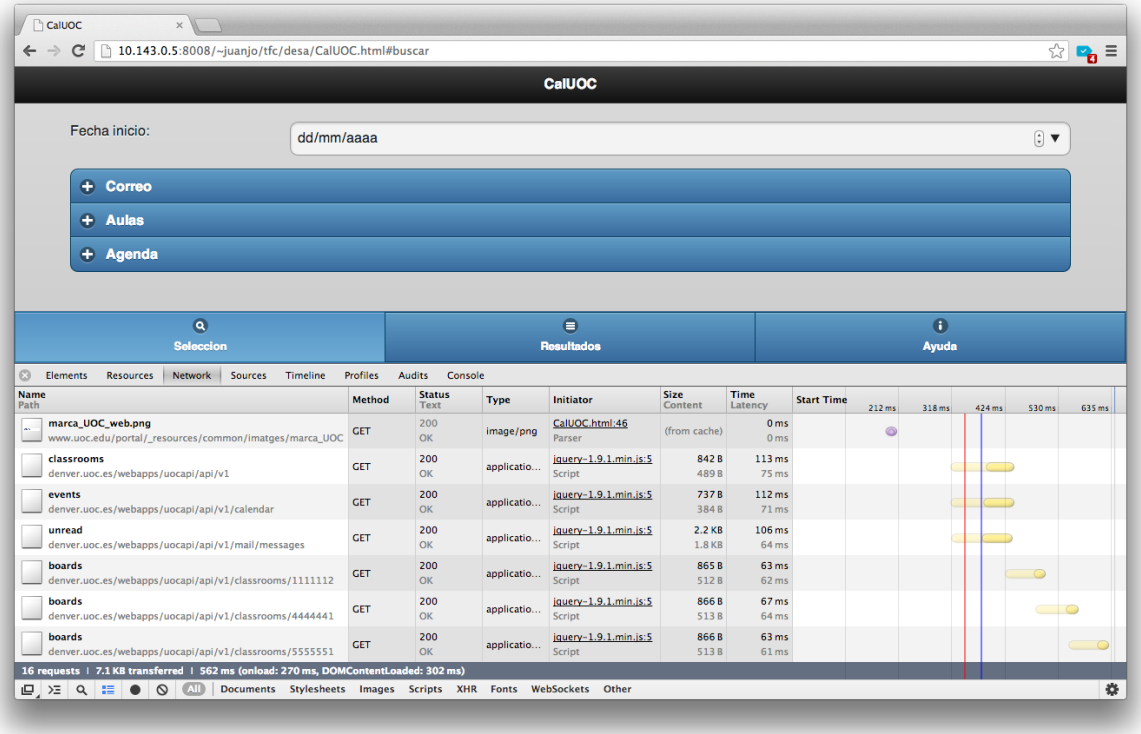

Figura 22. Diagrama de tiempos de carga de las peticiones AJAX.

Para la resolución de este problema fue de gran ayuda la consulta resuelta en la página StackOverflow:

http://stackoverflow.com/questions/5868096/how-to-wait-for-jquery-ajax-response-before-choosing-condition/5868124#5868124

Una vez obtenidos los datos la pantalla de selección que aparece con el menú Aulas desplegado quedaría así en un tablet Nexus 7:

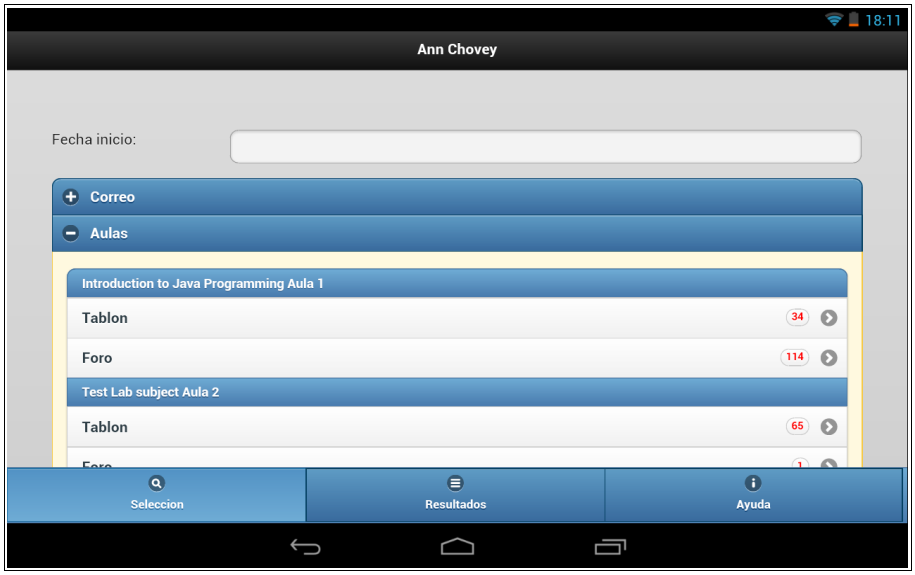

Figura 23. Pantalla selección. Versión final.

En el encabezado aparece el nombre del usuario. En este caso es un nombre ficticio, ya que la Open API a la que accedemos trabaja sobre un entorno de pruebas.

Figura 24. Pantalla selección con calendario. Versión final.

14.3. Transición entre pantallas

Una vez que se ha cargado la aplicación en el navegador, la función de jQuery que hemos utilizado anteriormente ya no resulta útil, porque debido al patrón de diseño de jQuery Mobile ya no se cargan nuevas páginas, si no que se muestra la sección de la página que se encuentra entre los tags   con la propiedad data-role='page' e identificador el de la página que queremos mostrar.

En su lugar tendremos que detectar ciertos eventos definidos por la librería jQuery Mobile, por ejemplo, para acceder a la pantalla de resultados detectamos cada uno de los eventos que nos conducen a esta página y hacemos visible la información solicitada y ocultamos el resto, por ejemplo, cuando se pulsa en la opción de correo de entrada en la pantalla de selección, mostramos la lista mensajes y ocultamos el resto:

```
1. //Transicion desde correo->entrada a resultados
2. $(document).on('vclick', '#corrIn', function(){
3.          $('#citas').hide();
4.          $('#menAulas').hide();
5.          $('#mensajes').show();
6.          $('#resulHead').text('Correo');
7. });
```

Para realizar operaciones antes de cargar una página habrá que observar el evento pagebeforeshow.

Cuando accedemos a la pantalla de resultados desde el apartado aulas, la versión final tendrá el siguiente aspecto:

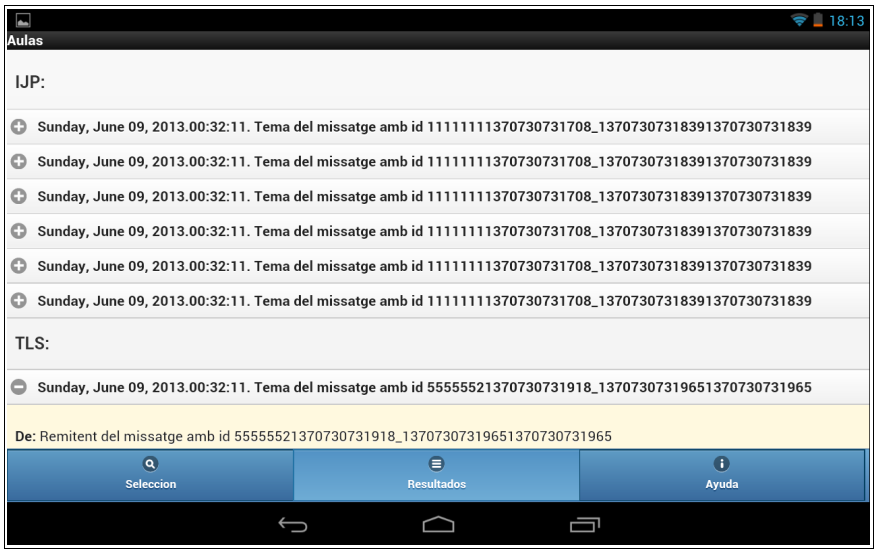

Figura 25. Pantalla resultados. Aulas. Versión final.

En la cabecera aparece el nombre de la información que hemos seleccionado para llegar a esta pantalla. En el centro de la pantalla se muestra una lista de mensajes por cada aula (separadas por el nombre corto de la asignatura) y si pulsamos sobre un mensaje aparece la información del remitente y el cuerpo del mensaje.

En el caso de llegar a la pantalla de resultados desde la opción Agenda:

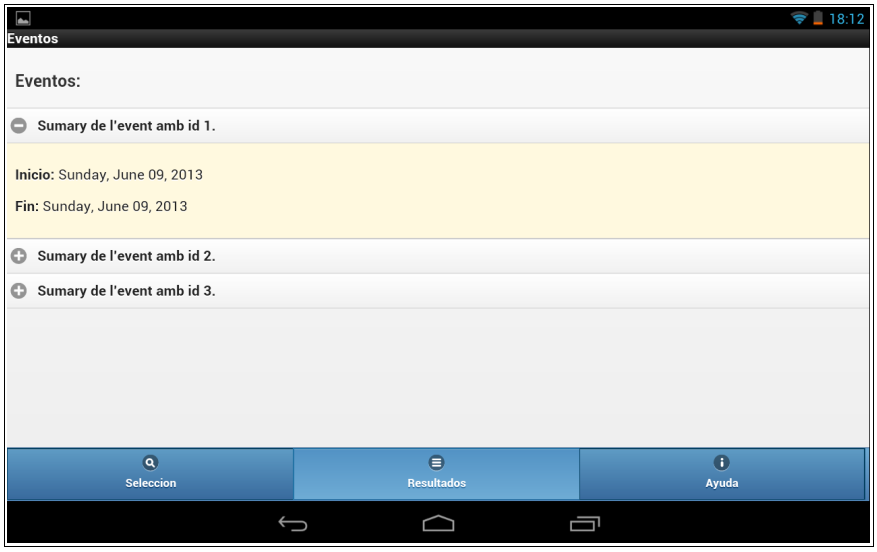

Figura 26. Pantalla resultados. Eventos. Versión final.

Para calcular las fechas y horas en formato local a partir de la información extraída del Campus se han implementado las funciones calcFecha, calcHora:

```
1. var calcFecha = function(fecha){
2.         var fechaHora = new Date(fecha.replace('CEST', ''));
3.         return fechaHora.toLocaleDateString();
4. };
5. var calcHora = function(fecha){
6.         var fechaHora = new Date(fecha.replace('CEST', ''));
7.         return fechaHora.toLocaleTimeString();
8. };
```

Durante las pruebas detectamos que las aulas obtenidas no se muestran en el formato adecuado debido a que las aulas se añaden después de que la lista haya sido creada y jQuery Mobile no detecta esta modificación. Para solucionarlo nuevamente StackOverflow nos sirve de ayuda para buscar la solución en la documentación de jQuery Mobile:

```
$('#aulas').append(items.join('')).listview('refresh');
```

14.4. Aplicación Android

Para convertir la aplicación web en una aplicación nativa para Android necesitamos instalar el entorno de desarrollo Eclipse y el SDK de Google para desarrollo para Android. En el siguiente enlace podemos descargar un paquete software que contiene ambas herramientas:

http://developer.android.com/sdk/index.html

Siguiendo las instrucciones de la documentación de PhoneGap que podemos encontrar en el siguiente enlace:

http://docs.phonegap.com/en/2.8.0/guide_getting-started_android_index.md.html#Getting%20Started%20with%20Android

Vemos que para crear el proyecto tenemos que usar el siguiente comando desde el subdirectorio bin de la carpeta android que se ha creado al descargar y descomprimir la aplicación:

```
./create <project_folder_path> <package_name> <project_name>
```

En la aplicación Eclipse creamos un nuevo proyecto a partir de código existente indicando la misma ruta que hemos utilizado en el comando anterior en <project_folder_path>.

Dentro del nuevo proyecto en Eclipse podemos copiar nuestra aplicación web bajo la carpeta www, en el apartado assets como se muestra en la figura siguiente:

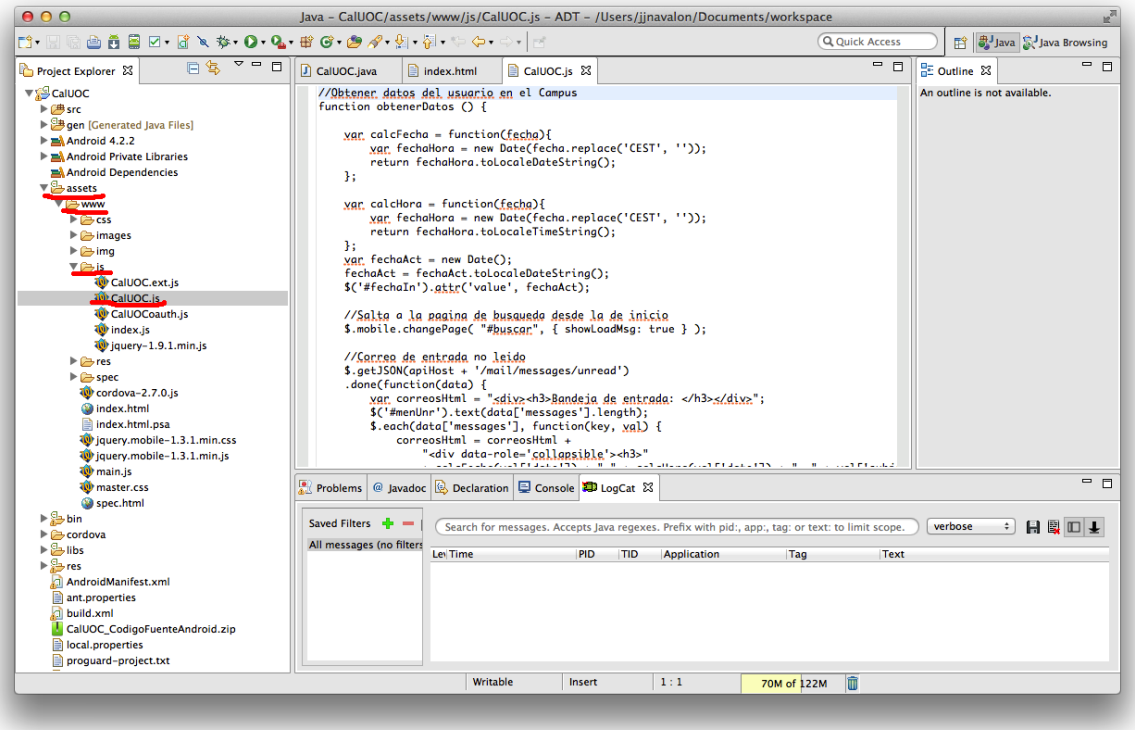

Figura 27. Proyecto PhoneGap en Eclipse.

Para poder probar la aplicación en un dispositivo real, como el Nexus 7 que hemos estado utilizando para las pruebas de la aplicación web, primero es necesario activar el modo desarrollador, para esto tendremos que pulsar varias veces sobre la opción que podemos encontrar en la siguiente ruta del dispositivo:

Ajustes → Sistema → Información del tablet → Número de compilación

A continuación, en el nuevo menú "Opciones de desarrollo" que aparece en el apartado Sistema de los ajustes, habrá que habilitar la opción "Depuración USB".

Desde Eclipse ya se puede ejecutar el proyecto como Aplicación Android y automáticamente instalará y ejecutará la aplicación en el dispositivo que habremos conectado por USB.

En la carpeta bin del proyecto podemos encontrar el archivo ejecutable CalUOC.apk.

15. Conclusiones

La decisión entre a la hora de decantarnos entre una aplicación web o una aplicación nativa no es fácil y hay que evaluar muchos factores como ya se ha comentado en el apartado 3 de justificación de la tecnología elegida.

En este caso la decisión de desarrollar una aplicación web está justificada por la relativa sencillez de las estructuras de datos que tiene que manejar; la alta dependencia de la información que hay en el Campus, solo accesible a través de Internet; y el hecho de no utilizar elementos hardware del dispositivo.

Si, a pesar de necesitar acceder a estas características propias del terminal, seguimos optando por el diseño web, tenemos la posibilidad de añadirle posteriormente las funciones específicas dependientes del hardware mediante la herramienta PhoneGap.

Un motivo suficiente para tener que usar una aplicación nativa podría ser si queremos incorporar notificaciones push que avisen al usuario de los cambios producidos en el Campus. Aunque existen librerías que lo permiten, para el usuario es más cómodo instalar una aplicación, que tener una ventana del navegador siempre abierta con la aplicación.

16. Bibliografía

Fox A, Patterson D (2012). *Engineering Long-Lasting Software*. Strawberry Canyon LLC.

Freeman E, Robson E (2011). *Head First HTML5 Programming*. O'Relly Media, Inc.

Crokford D (2008). *JavaScript: The Good Parts*. O'Reilly.

Castledine E, Sharkie C (2012). *jQuery: Novice to Ninja*. SitePoint Pty. Ltd.

Dutson P (2013). *Sams Teach Yourself jQuery Mobile in 24 Hours*. Sams Publishing.

Shotts K (2013). *PhoneGap 2.x Mobile Application Delvelopment*. Packt Publishing.

17. Referencias en línea

Gantt Project: http://www.ganttproject.biz

UOC Open API: http://blogs1.uoc.es/developer

The OAuth 2.0 Authorization Framework: http://tools.ietf.org/html/rfc6749

CORS: http://www.w3.org/wiki/CORS

Oauth2 Google: https://developers.google.com/accounts/docs/OAuth2#clientside

REST: http://www.w3.org/TR/ws-arch/#relwwwrest

W3School: http://www.w3schools.com

HTML: http://www.w3.org/TR/html51/

CSS: http://www.w3.org/TR/CSS/

jQuery: http://api.jquery.com

jQueryMobile: http://api.jquerymobile.com

PhoneGap: http://docs.phonegap.com/en/2.7.0/index.html

Codiqa:http://blog.codiqa.com

Apache en OSX: http://osxdaily.com/2012/09/02/start-apache-web-server-mac-os-x

TextWrangler: http://www.barebones.com/products/textwrangler/

Firebug: http://getfirebug.com/faq/

http://developer.apple.com/library/safari/#documentation/AppleApplications/Conceptual/Safari_Developer_Guide/1Introduction/Introduction.html

https://developers.google.com/chrome-developer-tools/docs/remote-debugging

Índice de ilustraciones

18. Glosario alfabético de términos

A lo largo del documento se han definido, en notas al pié de página, algunos términos especializados cuando aparecieron por primera vez en el texto. A continuación se muestra una lista de estos términos, esta vez por orden alfabético. Estas definiciones sintéticas se han obtenido con la ayuda de Wikipedia. http://www.wikipedia.org.

Acelerómetro: Dispositivo incorporado en terminales móviles y tabletas que permite detectar ciertos cambios en su posición relativa, como girar la pantalla.

Android: Sistema Operativo de Google empleado en smartphones y tablets de diversos fabricantes.

API: Application Programming Interface: Especificación de cómo deben interactuar unos componentes software con otros.

C: Lenguaje de programación compilado de alto nivel que permite el uso de funciones de bajo nivel cercanas al hardware.

CSS3: Cascading Style Sheets versión 3: Lenguaje de hojas de estilos usado para describir el aspecto y formato de un documento HTML.

DOM: Document Object Model: Estructura de datos para representar e interactuar con los elementos de un documento HTML.

FirefoxOS: Sistema operativo móvil basado en el navegador de código abierto Firefox y con núcleo Linux.

Flash: Tecnología software de Adobe para crear y manipular gráficos vectoriales mediante el lenguaje ActionScript. Para visualizar el contenido es necesario el componente software Flash Player.

Framework: Abstracción software que consiste en una plataforma de funcionalidad genérica y modificable utilizada para desarrollar aplicaciones con un patrón de diseño preestablecido.

GPS: Global Positioning System: Sistema de geolocalización incorporado en dispositivos móviles que utiliza la señal de una red de satélites.

HTML5: HyperText Markup Language versión 5: Lenguaje declarativo que especifica el contenido y estructura de una página web.

HTTP: Hypertext Transfer Protocol: Protocolo de nivel de aplicación empleado en las transferencias de datos en la World Wide Web.

HTTPS: Hypertext Transfer Protocol Secure: Versión segura del protocolo HTTP.

iOS: Sistema Operativo empleado en los productos iPhone e iPad de Apple.

iPhone. Teléfono inteligente con pantalla táctil fabricado por Apple.

Java: Lenguaje de programación orientado a objetos, cuyo código generado se compila en bytecodes que son interpretados en una máquina virtual disponible para la mayoría de los sistemas operativos.

JavaScript: Lenguaje de programación interpretado concebido inicialmente para su uso en navegadores web.

jQueryMobile: Framework para el desarrollo de aplicaciones y páginas web adaptadas a dispositivos móviles.

JSON: JavaScript Object Notation. Formato ligero para intercambio de datos que utiliza la notación de objetos de JavaScript.

OAuth2: Open Authorization versión 2: Protocolo abierto para la autorización segura del uso de una API abierta.

PhoneGap: Herramienta software que permite convertir aplicaciones web en aplicaciones nativas para sistemas operativos de smartphones y tablets.

Responsive Design: Técnicas de diseño y desarrollo web que intenta adaptar el sitio web al entorno del usuario.

REST: Representational State Transfer: Arquitectura de software empleada en sistemas distribuidos en red como la World Wide Web.

RESTful: Librería que cumple los patrones de diseño de la arquitectura REST.

RFC: Request for Comments: Publicación técnica del IETF e ISOC que describe algún estándar concreto empleado en Internet.

SMS: Short Message Service: Servicio de envío y recepción de mensajes cortos de texto empleado en la telefonía móvil desde el estándar GSM.

Spyware: Programa espía que se instala sin el consentimiento explícito del usuario y recoge información personal que envía al atacante.

Tablet o tableta. Computadora portátil de mayor tamaño que un teléfono inteligente con pantalla táctil.

Token: Identificador de una sesión de intercambio de mensajes.

UOC: Universistat Oberta de Catalunya.

URI, URL, URN: Uniform Resource Identifier, Locator, Name.

XML: eXtensible Marckup Language. Lenguaje superconjunto de HTML utilizado para el intercambio de información entre sistemas heterogéneos.