Creación de Mobile apps para catálogos de empresa

Memoria de Proyecto Final de Grado

Grado de Multimedia

Mención en *«Desarrollo de aplicaciones interactivas»*

Autor: Francisco Olmo Ortiz

Consultor: Eva Casado de Amezua Fernandez-Luanco

Profesor: Carlos Casado Martinez

20 de Junio de 2013

Creación de Mobile apps para catálogos de empresa

Memoria de Proyecto Final de Grado de Multimedia

Consultor: Eva Casado de Amezua Fernandez-Luanco
Profesor: Carlos Casado Martinez

Universitat Oberta de Catalunya (UOC)

Dedicado a la memoria de mi padre, Antonio

Abstract

El proyecto que se presenta en esta memoria, se enmarca en el entorno de las tecnologías basadas en estándares web, con las cuales se pretende desarrollar un producto comercial, apto para poder competir en el mercado de las aplicaciones móviles, demostrando ser definitivamente una seria alternativa en el emergente campo de la programación multiplataforma.

A lo largo del desarrollo del proyecto, hemos adoptado el esquema Modelo Vista Controlador, utilizando las nuevas características de PHP5 que nos introducen en el uso avanzado de la programación orientada a objetos. Esto nos ha permitido usar metodologías de programación cercanas a las habituales en el desarrollo de aplicaciones, pero utilizando las tecnologías a las que estamos habituados como diseñadores de sitios web. A su vez, el uso del framework JQuery Mobile, nos ha facilitado la creación de interfaces de usuario con el *look and feel* de las aplicaciones nativas. Por último, las soluciones de encapsulado como PhoneGap, usada en este proyecto, completan el proceso de creación de aplicaciones nativas a partir de web apps.

Podemos concluir que el mercado actual nos ofrece herramientas suficientes como para considerar el uso de tecnologías web, toda una apuesta de futuro en el desarrollo de aplicaciones móviles. Además el fuerte apoyo de grandes nombres del sector y su rápida evolución, no hacen sino corroborar que crear aplicaciones nativas multiplataforma está en vías de convertirse en toda una realidad.

Palabras clave: memoria, Trabajo Final de Grado, dispositivos móviles, smartphone, tablet, web app, estándares web, aplicaciones nativas, multiplataforma, Android, iPhone, iPad, iOS, Windows Phone, BlackBerry, HTML5, CSS3, JavaScript, AJAX, MySQL, PHP, programación orientada a objetos, JQuery, JQuery Mobile, PhoneGap, QR, Google Maps, catálogo electrónico, publicidad, empresa, pyme.

Agradecimientos

A mi esposa Alexandra, a mi madre Salud y a mis hermanos Juan y Antonio, por apoyarme sin condiciones.

A todos los profesores, tutores y compañer@s que he conocido a lo largo de la carrera que consiguieron crear en mí el sentimiento de pertenecer a una gran comunidad de personas.

Gracias a todos,
Francisco Olmo Ortiz

Índice

1. Introducción

El presente proyecto trata sobre la elaboración de una herramienta software que permitiera la creación y gestión de los contenidos, de apps nativas para iOS y Android a partir de plantillas personalizables, creadas en formato HTML5, las cuales se nutrirían de una base de datos alojada en la nube.

Las aplicaciones web así obtenidas, serían encapsuladas y convertidas en apps nativas, para poder ser descargadas e instaladas en smartphones y tablets.

Su objetivo sería proveer a las empresas de catálogos electrónicos en formato de aplicaciones para dispositivos móviles, ofreciendo un nuevo canal publicitario para sus novedades o productos destacados, en un formato moderno y atractivo y que además les permita gestionar sus contenidos, con la finalidad de estar permanentemente actualizados.

1.1 Público objetivo

El público objetivo del servicio propuesto en este proyecto, son pequeñas empresas que manufacturan productos *(muy especialmente artesanía)*, las cuales no tienen en su gran mayoría, ni conocimientos ni tiempo, para dedicarle a este tipo de cosas. Nosotros les ofrecemos un producto de publicidad electrónica, en un formato que se está imponiendo de forma masiva, como son las aplicaciones para dispositivos móviles.

Estos empresarios, tan solo han de preocuparse de elegir los colores o diseños de su aplicación y de gestionar la información que deseen mostrar, a través de la aplicación web del gestor de contenidos que les ofreceríamos, para que su información esté siempre actualizada; de todo lo demás nos encargamos nosotros.

1.2 Producto resultante

Si bien existen en la web innumerables diseños y plantillas HTML, para este proyecto, se crearía un prototipo propio, ya que éste requiere estar programado con el código AJAX/PHP correspondiente, que se nutra de los contenidos de nuestra base de datos y tendría su estructura establecida en nuestro servicio.

El producto resultante, consistiría en una aplicación web de gestión de contenidos básicos (productos, logo de empresa y datos) y una plantilla HTML preparada con código dinámico PHP para nutrirse de nuestra base de datos alojada en la nube, gestionada por dicho CMS. Se entregaría ya encapsulada como app nativa para Android 4.x, funcionando en un terminal smartphone o tablet.

1.3 Requerimientos técnicos

Finalmente, se exponen los requerimientos en cuanto a equipamiento hardware y herramientas de software, que se estiman necesarios para la realización del presente proyecto.

Hardware:

- Equipo PC de escritorio para desarrollo, con impresora y escáner
- Smartphone y tablet Android y/o iOS para probar la aplicación resultante
- Conexión ADSL para Internet

Software:

- IDE para el desarrollo de las aplicaciones de gestión de contenidos y la web app: Dreamweaver o Aptana. En este caso, se ha usado el entorno Adobe Dreamweaver CS6
- Servidores web Apache+PHP y MySQL instalados, para la ejecución en local de las aplicaciones web
- Herramienta de gestión MySQL Workbench, para la gestión de la base de datos del sistema
- Aplicación web PHPMyAdmin para la gestión de la base de datos del servidor de pruebas remoto de la UOC
- Framework JQuery Mobile para el diseño de las vistas de las web apps
- Framework PhoneGap para convertir en apps nativas, a las web apps creadas por el sistema
- Google Maps para geolocalización sobre el mapa
- Adobe Fireworks CS6 y Adobe Photoshop CS6, como software de tratamiento gráfico para la edición de imágenes
- Aplicaciones ofimáticas Microsoft Office Word y Excel, para la redacción de la documentación del proyecto

2. Descripción

El proyecto integraría la creación de tres módulos, que serían los siguientes:

1. **Módulo de Gestión de clientes**

   Este módulo crea y edita a los clientes del servicio. Es gestionado por la empresa proveedora del servicio *(en adelante, nosotros)*.

   Se trata de una aplicación web programada en PHP, conectada a la base de datos que habremos creado previamente. Nosotros nos encargamos simplemente de dar de alta a los clientes. La aplicación generará automáticamente, una clave de acceso, una carpeta en el servidor y un código QR para cada cliente que demos de alta.

2. **Módulo de Gestión de contenidos**

   Aplicación web a modo de un sencillo gestor de contenidos *(en adelante, CMS)*, programado en PHP y conectado con la misma base de datos descrita en el módulo anterior. Esta aplicación será gestionada por nuestros clientes, mediante el acceso con la contraseña que nosotros le proveemos.

   Una vez dentro, los clientes, gestionarán su catálogo de novedades *(fotografía, nombre, breve descripción y precio)* así como sus datos *(teléfono, dirección, email ...)*

   También desde esta aplicación, los clientes podrán subir su logotipo de empresa y elegir la plantilla o colores que prefieren para su app.

   Además, esta aplicación mostrará en forma de imagen PNG, el código QR que le generamos automáticamente al darlo de alta en el sistema. Este código QR tendrá como finalidad, que lo puedan imprimir o mostrar en su web, con el fin de que los interesados en sus productos accedan sin más a las stores donde podrán descargarse la app, según el sistema operativo de su dispositivo móvil.

   Finalmente, cuando nuestro cliente está conforme con su logo y diseño de plantilla, clicará una opción, que generará en la carpeta del cliente, en el servidor web, un fichero XML el cual contendrá el nombre e identificador de la empresa, así como el identificador de la plantilla de la web app.

   En ese mismo proceso, a nosotros se nos activará en nuestro gestor de clientes, una alarma que nos indica que el cliente ya ha dado su aceptación, para que procedamos a crearle su app y subirla a las stores.

3. Módulo de Creación de la app

Nosotros crearemos unas plantillas, ya preparadas con su código AJAX/PHP, que se nutran de los contenidos incorporados en nuestra base de datos en la nube. Esta aplicación estará diseñada con estándares web: HTML5/CSS3 y se construirá con la ayuda del framework JQuery Mobile.

A partir de la carpeta creada en el servidor web para cada cliente, que contendrá estas plantillas, ya personalizadas con su identificación de cliente correspondiente, nosotros procederemos a encapsular dicha web app con PhoneGap y obtendremos las app nativas para los sistemas operativos que necesitemos.

Finalmente, nosotros publicaremos las apps nativas en sus respectivas stores, para que puedan ser descargadas por todas las personas interesadas en los productos de nuestras empresas clientes.

3. Objetivos

3.1 Principales

- Crear una herramienta de gestión de contenidos para pequeñas empresas manufactureras, con el objetivo de proveer de información a una web app, encapsulada como app nativa, la cual será ofrecida por dicha empresa a sus posibles clientes, como elemento publicitario *(catálogo electrónico)* donde mostrará sus productos.

- Minimizar el tiempo requerido para la puesta en marcha y mantenimiento del sistema, pues la empresa que contrata el servicio tan solo tiene que preocuparse de entrar en el CMS y gestionar sus productos y datos, así como elegir la plantilla que desee. La empresa proveedora del servicio, se encargaría de encapsular la wep app generada por el CMS y publicarla en las stores.

3.2 Secundarios

- Realizar un proyecto que abarque la mayor parte de los temas contenidos en las asignaturas optativas que conforman la mención de "Desarrollo aplicaciones interactivas":

    - ***Programación web avanzada***
      Uso de tecnologías AJAX/JQuery y PHP.

    - ***Aplicaciones rich media***
      Creación de una aplicación interactiva multimedia en formato de web app, usando estándares JavaScript/HTML5/CSS3 para la creación de las interfaces de usuario.

    - ***Uso de bases de datos***
      Creación de una base de datos MySQL, que albergará la información gestionada por los clientes del sistema mediante una aplicación web, programada en PHP.

    - ***Seguridad y calidad en servidores web***
      Aplicación de medidas de seguridad a la hora de programar las entradas y salidas a la base de datos, para evitar ataques como inyección de código y similares. Elección del hosting adecuado para alojar el servicio.

  Ampliar los conocimientos adquiridos a lo largo del grado, mediante la investigación y estudio de las nuevas técnicas de creación de aplicaciones interactivas para dispositivos móviles. De este modo, se cumplen de forma expresa los objetivos y competencias transversales establecidos en el plan docente del TFG.

4. Marco teórico/Escenario

Cuando en el año 2007 Apple comercializó la primera versión de su dispositivo iPhone, tal vez no muchos pudieron vislumbrar la revolución tecnológica que se produciría en los meses siguientes. Fruto de la convergencia tecnológica de la telefonía móvil y la informática, arropadas por una rápida expansión de la líneas ADSL que hacían posible un servicio de Internet de alta velocidad, el mercado de las telecomunicaciones veía nacer una nueva manera de llevar las tecnologías de la información y la comunicación, al gran público. En los meses siguientes, asistimos a toda una sucesión de novedades que venían a ampliar la oferta de dispositivos y sistemas; Android de Google, BlackBerry y Windows de Microsoft, hacen sus apuestas, unos con más fortuna que otros.

Nace al mismo tiempo un nuevo concepto del software, las aplicaciones que se descargan desde las tiendas que cada plataforma pone a disposición de sus usuarios. Suelen ser pequeños programas que realizan tareas muy concretas y cuyas características principales son su bajo consumo de recursos hardware y su pequeño tamaño, al tener que operar en el mundo de la Internet móvil, donde se paga por el tráfico de información consumida.

Este naciente ecosistema multiplataforma y multisistema, plantea un serio reto a los desarrolladores de software por su diversidad de tecnologías y lenguajes de programación. El concepto de "write once, run everywhere" que años atrás Sun Microsystems lanzara como eslogan para su plataforma JAVA, vuelve a aparecer en la mente de todos.

Sin embargo, paralelamente al boom de smartphones y tablets, las tecnologías de Internet también han evolucionado de forma considerable. Nuevos estándares para el diseño y programación de aplicaciones web hacen su aparición, con la intención de entrar rápidamente en juego. Conceptos como HTML5, CSS3 y AJAX, hacen que estas tecnologías puedan representar un papel relevante en el nuevo escenario de las aplicaciones para dispositivos móviles.

El último eslabón de esta cadena de evolución tecnológica, lo representan aplicaciones que son capaces de encapsular todos los ficheros que componen la aplicación web, en una aplicación nativa, la cual usará para el renderizado de sus vistas al navegador interno del sistema operativo, en lugar de la interfaz gráfica propia.

Tenemos así, todos los elementos necesarios para poder obtener aplicaciones nativas multiplataforma, usando tecnologías estándares, sin necesidad de aprender múltiples lenguajes de programación.

5. Metodología

Patrón Modelo Vista Controlador (en adelante MVC)

Dado que este proyecto pretende fundamentalmente el desarrollo de aplicaciones web, aunque finalmente las encapsulemos como aplicaciones nativas, usaremos la metodología MVC por ser la que mejor se adapta a las características de funcionamiento de las aplicaciones interactivas en general.

Este método, separa las capas de presentación de la aplicación, gestión de datos y dominio o negocio, en tres componentes: el modelo, la vista y el controlador. Básicamente su función será:

- **Modelo**: comprende las capas de gestión de datos y el dominio de la aplicación.
- **Vista**: es la capa de presentación, la interfaz con la que interactúa el usuario.
- **Controlador**: responde a las peticiones que realiza el usuario, generalmente mediante acciones y eventos de la interfaz. El controlador se comunica con el modelo y devuelve el resultado solicitado a la vista, pudiendo por tanto, producir cambios en ambos componentes.

En el Anexo 2 de *Código fuente*, se muestra un ejemplo completo de desarrollo, de acuerdo al patrón Modelo Vista Controlador, incluyendo extractos de código fuente que ilustran su funcionamiento.

Diseño centrado en el usuario (en adelante DCU)

El presente proyecto pretende como producto final, la creación de una aplicación interactiva, en forma de app nativa para dispositivos móviles. Esto implica que será la interfaz de usuario el elemento central de su funcionamiento. Por tanto, el diseño de dicha interfaz, para cada uno de los tres módulos a desarrollar, se llevará a cabo bajo la metodología del diseño centrado en el usuario, cuyas etapas fundamentales son:

- **Análisis**. Objetivos de la aplicación, características de los usuarios, requerimientos del proyecto
- **Diseño**. Diseño conceptual *(arquitectura de la información)* y diseño visual
- **Prototipo**. Prototipos de baja y de alta fidelidad. Prototipo funcional
- **Evaluación de la usabilidad.** Testeo y pruebas de la interfaz

En el capítulo 9 de *Proceso de trabajo*, se expone con todo detalle el desarrollo del diseño de las tres aplicaciones del proyecto, de acuerdo a los principios aquí expuestos, de la metodología DCU.

6. Arquitectura de la aplicación

Una vez definida la aplicación que vamos a desarrollar y detallados sus contenidos y las metodologías implicadas, pasamos en este capítulo a describir la arquitectura en la que quedará enmarcada.

Tal como ya hemos comentado anteriormente, se trata de la creación de una aplicación web, la cual finalmente será encapsula y convertida en apps nativas para diferentes plataformas móviles. Por tanto, nos encontramos ante un clásico esquema cliente/servidor. Analicemos sus componentes principales:

- **Servidor web**
  La aplicación encargada de servir las páginas web solicitadas por los clientes. Además, en nuestro caso, nos servirá de alojamiento de las aplicaciones de gestión de clientes y de gestión de contenidos de dichos clientes.

  Para ello, usaremos el servidor web Apache, con el módulo del lenguaje de scripts de servidor, PHP.

- **Servidor de bases de datos**
  Obviamente, la aplicación requerirá de una base de datos donde alojar la información de los clientes, así como los productos y datos que éstos introduzcan en el sistema.

  Hemos elegido el sistema de bases de datos relacionales MySQL, por tratarse de software Open Source y dada su versatilidad y amplia penetración en el mercado actual.

- **Clientes**
  El proyecto comprende la creación de tres módulos independientes, cada uno con una funcionalidad determinada. Usaremos como clientes de las aplicaciones de gestión, cualquier navegador web compatible con las tecnologías HTML5. Para la aplicación nativa que finalmente generaremos, nuestros clientes serán los dispositivos móviles smartphone y tablet.

En la página siguiente, mostramos un esquema gráfico detallado, del funcionamiento de la arquitectura que acabamos de describir.

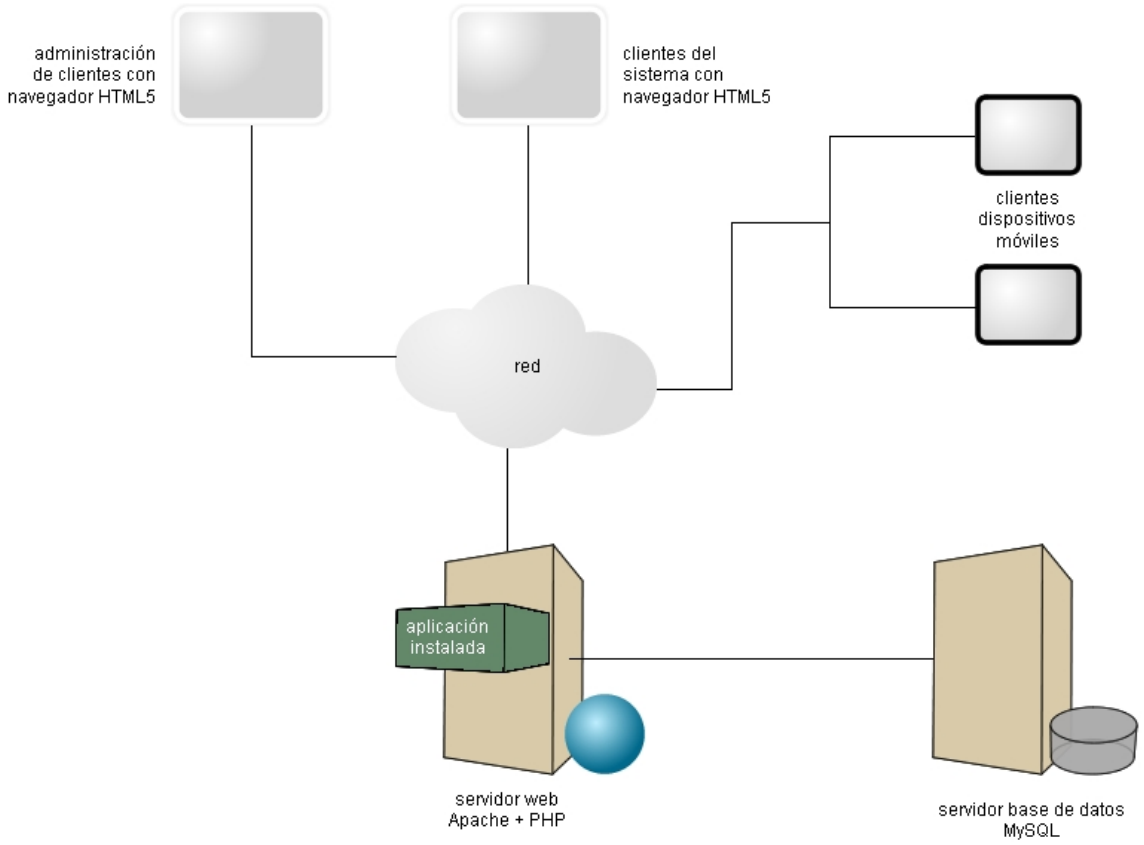

Figura 1: Esquema de la arquitectura del sistema

7. Plataforma de desarrollo

A continuación, se exponen las tecnologías que se emplearán en los distintos procesos de desarrollo del proyecto:

Software:

- Sistema de gestión de bases de datos relacional MySQL, para la creación de la base de datos del sistema y lenguaje de consulta estructurado SQL para la gestión de la base de datos
- Lenguaje de programación del lado del servidor PHP, para el desarrollo del modelo y controlador de las aplicaciones a desarrollar
- JavaScript como lenguaje del lado del cliente, para la creación de la interactividad de las vistas o interfaces de las aplicaciones del proyecto
- Tecnología AJAX para la comunicación asíncrona de las vistas con los scripts PHP del servidor que componen el controlador de la aplicación
- Estándares de diseño y maquetación web HTML5 y CSS3 para la creación de todas las vistas
- Framework JQuery Mobile para facilitar la creación e interactividad de las vistas de la aplicación web final, para los dispositivos móviles
- Framework PhoneGap para la conversión de web apps a aplicaciones nativas multiplataforma
- Google Maps para geolocalización sobre el mapa

Hardware:

- Dispositivos móviles smartphone y tablet, bajo sistema operativo Android 4.x. Se usarán para las pruebas reales de la aplicación

8. Planificación

8.1 Fases de desarrollo del proyecto

Definición formal del proyecto *(27/02/13 - 12/03/13)*

- Análisis
- Definición del proyecto
- Planificación
- Documentación del Trabajo fin de Grado

Hito 1. Entrega de la PEC 1 - 12/03/13

Creación de la aplicación de gestión de clientes *(13/03/13 - 07/04/13)*

- Modelado y creación de la base de datos
- Diseño UML de la lógica de negocio de la aplicación
- Modelo, vista y controlador de la aplicación de gestión de clientes
- Documentación del Trabajo fin de Grado

Hito 2. Entrega de la PEC 2 - 07/04/13

Creación de la aplicación CMS *(08/04/13 - 12/05/13)*

- Diseño de las vistas (interfaz) de la aplicación CMS
- Programación del modelo y controlador
- Reajustes en la base de datos y el diseño UML
- Documentación del Trabajo fin de Grado

Hito 3. Entrega de la PEC 3 - 12/05/13

Creación y encapsulado de la App nativa *(13/05/13 - 20/06/13)*

- Diseño de las vistas (interfaz) de la web app
- Modelo y controlador de la web app
- Programación de la generación de web apps
- Encapsulado de la web app a nativa con PhoneGap
- Pruebas de la app nativa en dispositivos reales
- Pruebas finales del sistema
- Documentación del Trabajo fin de Grado

- ***Hito 4. Entrega final - 20/06/13***.

8.2 Diagrama de Gantt

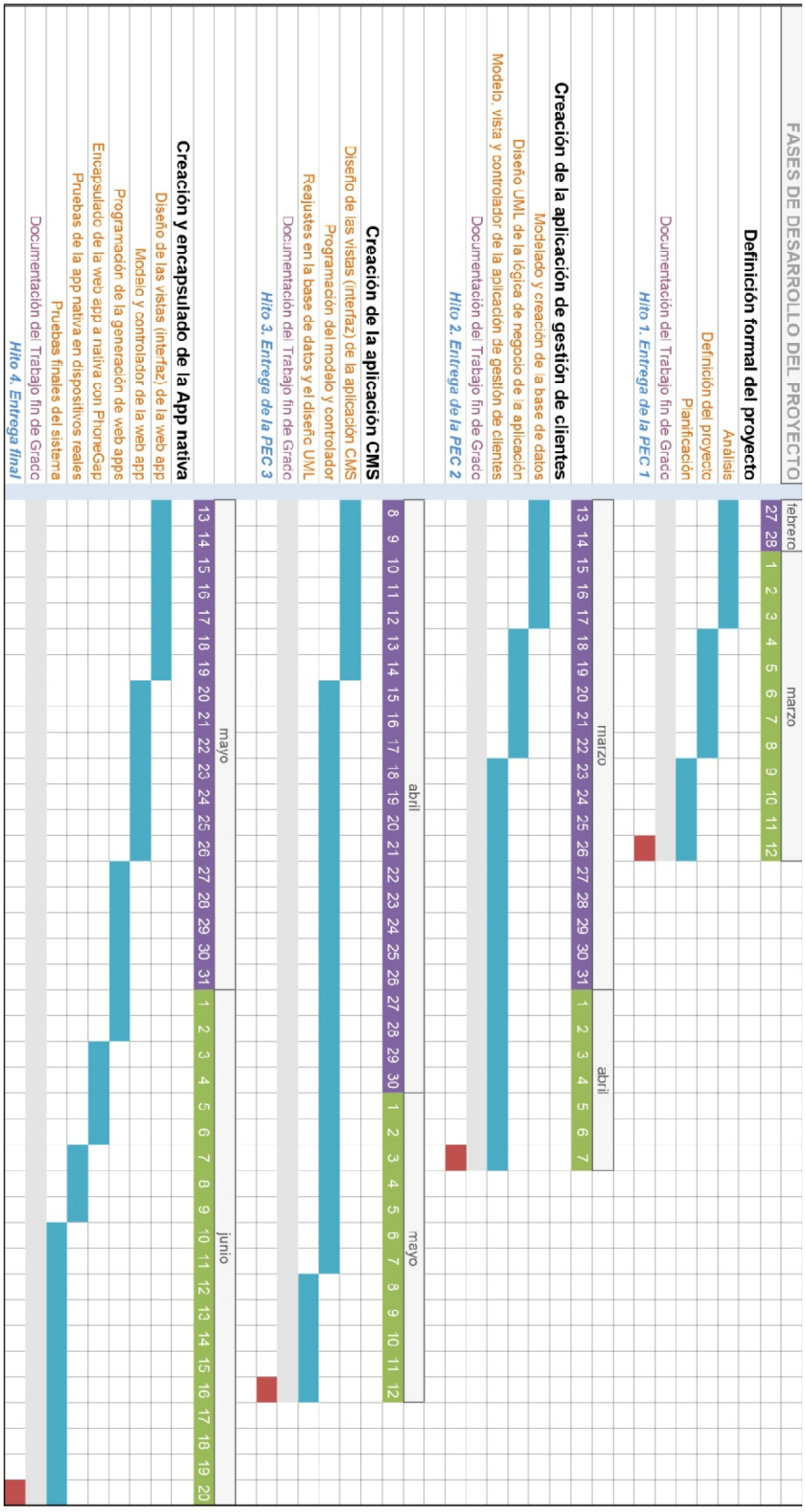

Figura 2: Planificación. Diagrama de Gantt

9. Proceso de trabajo/desarrollo

Detallamos en este capítulo el proceso de desarrollo de cada una de las tres aplicaciones que componen el proyecto. No obstante, como etapa previa, necesitaremos establecer el esquema de funcionamiento de la lógica de negocio de la aplicación, así como el diseño y modelado de la base de datos que soportará todo el sistema.

- **ESQUEMA UML DE LA CAPA DE NEGOCIO**

La capa de negocio o lógica de la aplicación, consiste en los módulos de programación que realizarán todas las peticiones solicitadas por los usuarios del sistema, a través de la interacción con la capa de presentación. Será asimismo, la que se comunicará con la capa de datos, para llevar a cabo la gestión de la base de datos. Ambas capas, la lógica de negocio y la capa de datos, forman parte del modelo, en el esquema modelo, vista, controlador (MVC).

En los códigos fuente de las aplicaciones, se han añadido comentarios sobre la utilidad de cada una de las clases y métodos.

En el siguiente gráfico, mostramos la abstracción de la aplicación, representada por las clases que la componen y las relaciones entre las mismas:

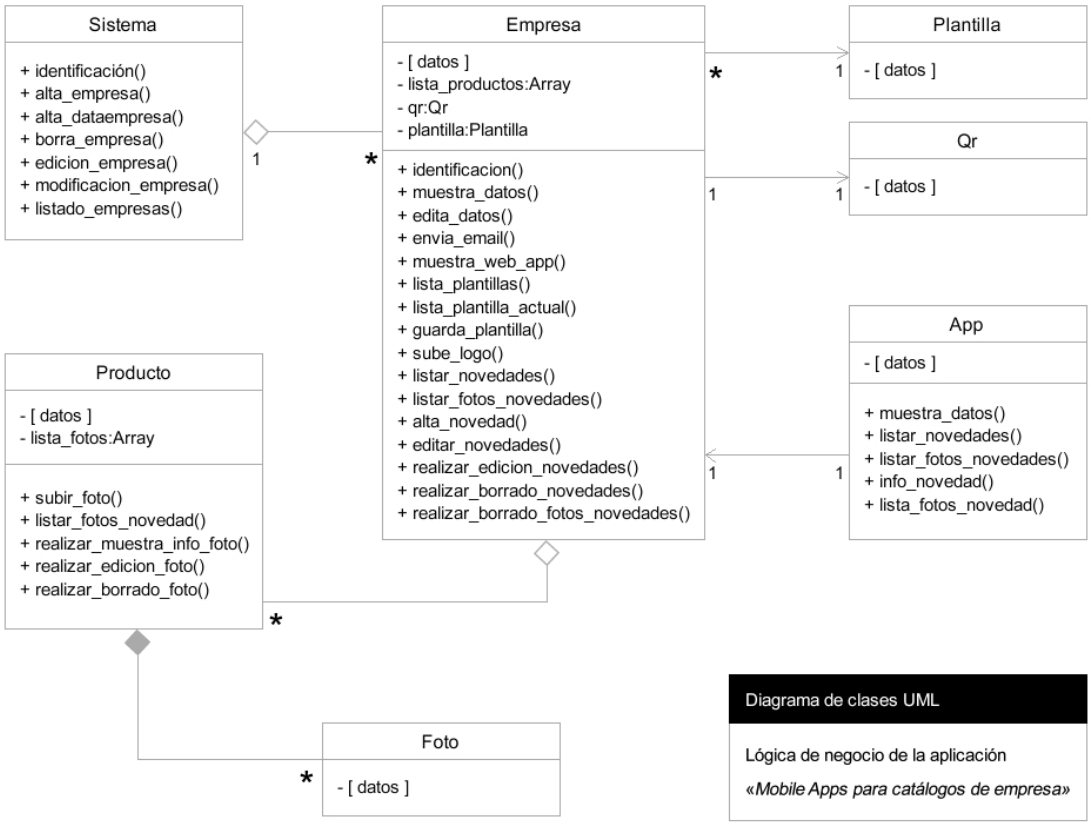

Figura 3: Diagrama de clases UML de la lógica de negocio de la aplicación

- **DISEÑO Y MODELADO DE LA BASE DE DATOS**

  Una vez diseñado el esquema de funcionamiento de la lógica de la aplicación, procedemos al diseño y modelado de la base de datos de nuestro sistema. Como ya hemos indicado anteriormente, hemos optado por usar MySQL como gestor de bases de datos.

  Del estudio y análisis de la lógica de la aplicación, concluimos la necesidad de implementar las siguientes tablas y campos, así como sus relaciones:

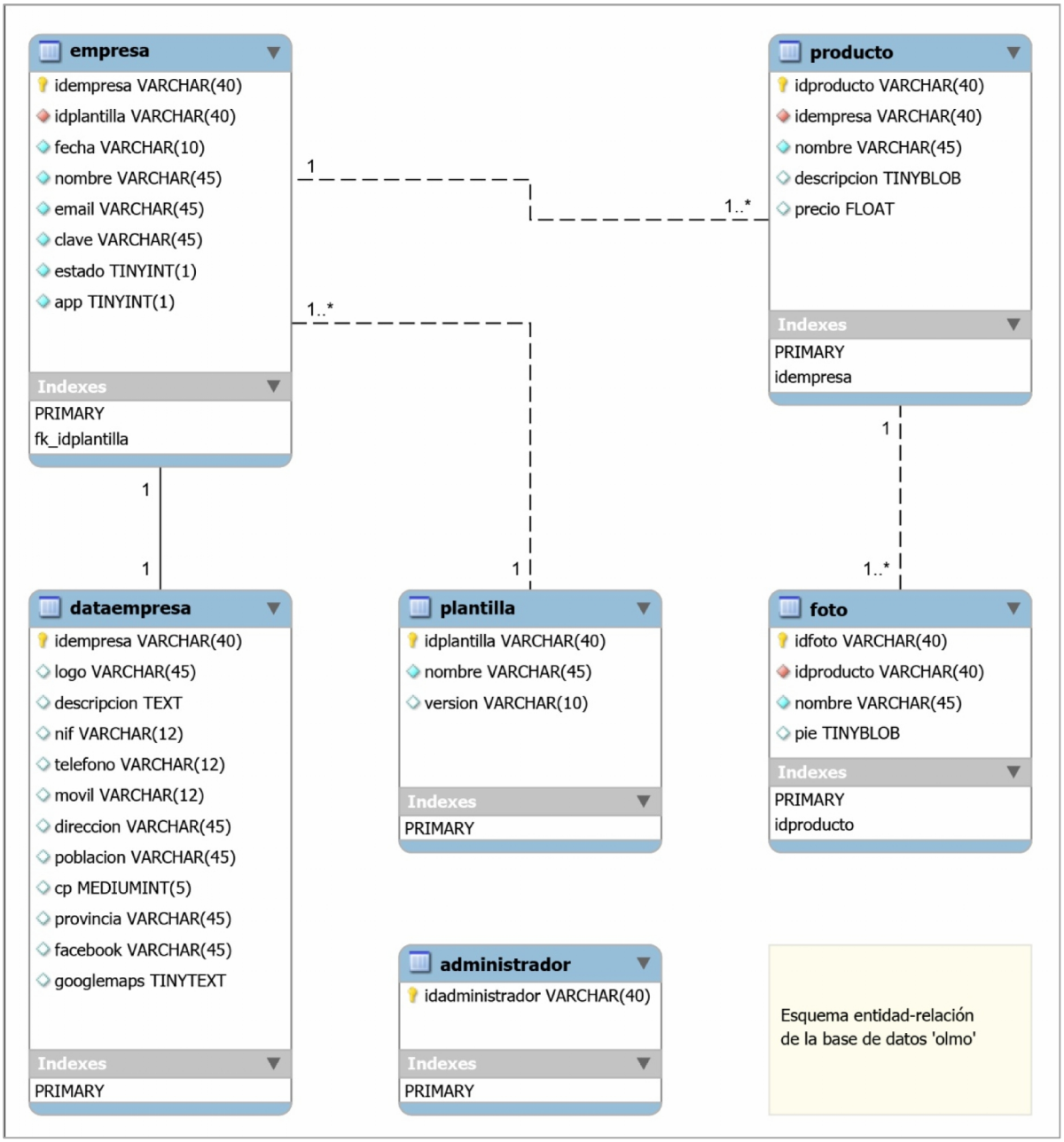

Figura 4: Esquema entidad-relación de la base de datos de la aplicación

De esta forma, tenemos ya creada la capa de datos que nos ha de servir como base sobre la que desarrollaremos toda la aplicación. La programación de las clases establecidas en el esquema UML, así como aquellas que gestionen las entradas y salidas de la base de datos, conformarán el modelo del esquema MVC.

Procedemos ya el desarrollo de los tres módulos de la aplicación. Al tratarse en todos los casos de aplicaciones interactivas, comenzaremos el procedimiento de acuerdo al modelo de diseño centrado en el usuario (DCU). Una vez tengamos claro qué objetivos ha de cumplir cada una de ellas, qué contenidos han de gestionar y cómo han de hacerlo, diseñaremos sus vistas o lo que es lo mismo, la interfaz con la que interactuarán los usuarios.

- **MÓDULO DE GESTIÓN DE CLIENTES**

Diseño de la aplicación. Modelo DCU - Fases

1. **Análisis**

   El objetivo de este primer módulo será gestionar a los clientes de nuestro servicio. El usuario objetivo de la aplicación será el administrador de nuestra empresa, quien se encargará del alta, edición, baja y consultas de clientes. Deberá basarse por tanto, en una interfaz de navegación sencilla y usable que realice las tareas de forma fácil y rápida.

2. **Diseño conceptual. Arquitectura de la información**

   Nos encontramos ante la interfaz de una aplicación software. Como es habitual en este tipo de contexto, usaremos una organización de los contenidos de tipo ambiguo, por tareas.

   En cuanto a la navegación, estableceremos una estructura jerárquica, a partir de un menú principal.

   Recogemos en el siguiente esquema detallado, la organización, navegación y etiquetado de dichas tareas.

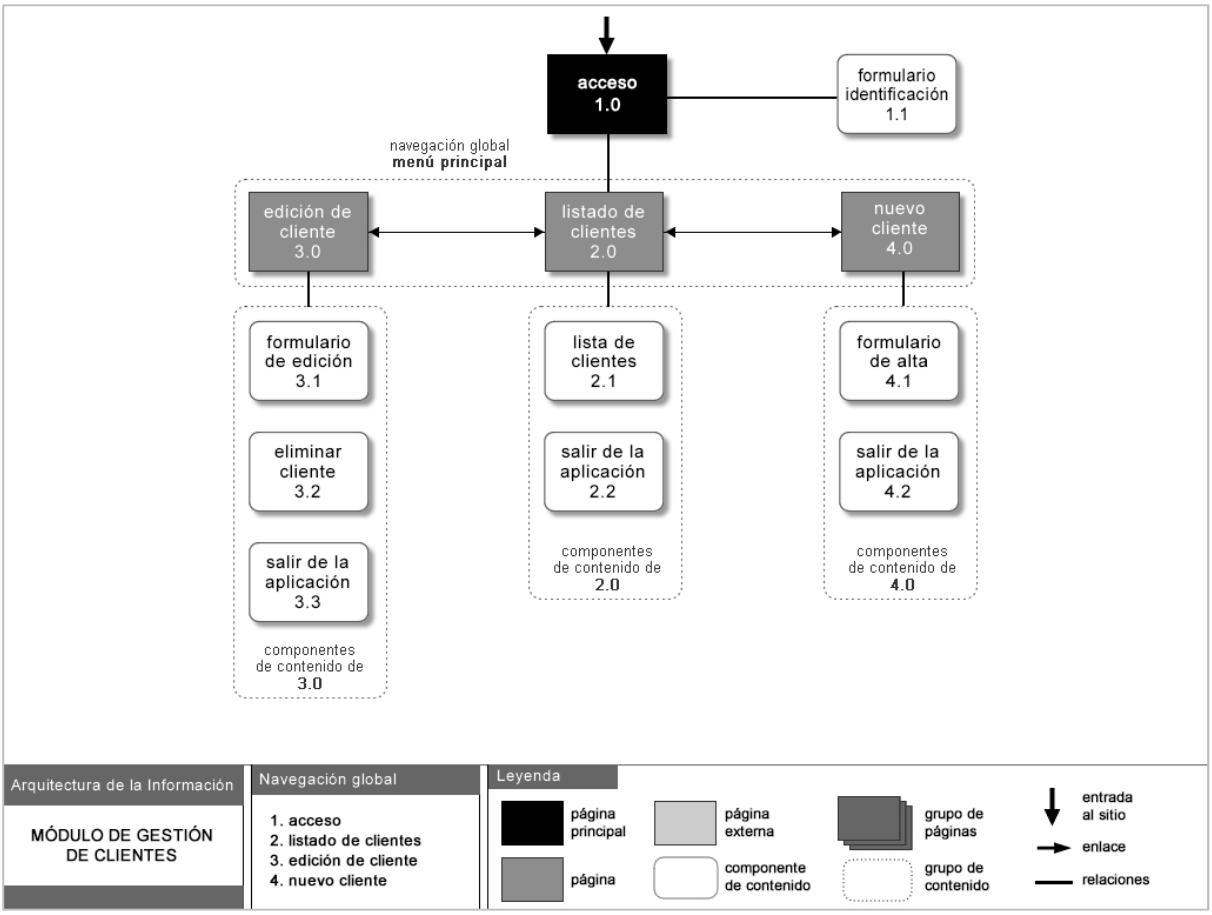

Figura 5: Esquema detallado. Arquitectura de la información de la aplicación

3. Diseño visual. Prototipos

Figura 6: Prototipo de alta fidelidad de la vista de acceso al sistema

Figura 7: Prototipo de alta fidelidad del listado de clientes

Figura 8: Prototipo de alta fidelidad de la vista de alta de clientes

Figura 9: Prototipo de alta fidelidad de la vista de edición de clientes

Esquema modelo, vista, controlador (MVC)

Una vez concluido el diseño visual, procedemos a la creación de los prototipos funcionales en lenguaje HTML y al desarrollo de todos los módulos de programación que componen la aplicación. La siguiente tabla, muestra el conjunto de ficheros que conforman el esquema MVC de dicha aplicación.

Recordemos que tal como se expone en el capítulo 2, la aplicación generará automáticamente, una clave de acceso, una carpeta en el servidor y un código QR, para cada cliente que demos de alta.

MODELO	VISTA	CONTROLADOR
conexion_db.php sistema.php	index.php listado.php alta.php edicion.php	cierra_sesion_administrador.php identificacion.php muestra_alta.php muestra_edicion.php muestra_listado.php realiza_alta.php realiza_borrado.php realiza_edicion.php

- **MÓDULO DE GESTIÓN DE CONTENIDOS**

Diseño de la aplicación. Modelo DCU - Fases

1. Análisis

El módulo de gestión de contenidos o CMS, será la herramienta que usarán las empresas clientes de nuestro servicio, para gestionar sus datos y el catálogo de productos que presentarán a través de la aplicación móvil que el sistema generará automáticamente.

Como ya se adelantó en el capítulo I, nuestro público objetivo serán pequeñas empresas dedicadas especialmente a la artesanía. A partir de este dato, podemos modelar algunos de los perfiles de usuario que habitualmente, se suelen encontrar en este ámbito empresarial:

- ***Personal de administración o secretaría***
  Su labor principal suele ser la gestión de la empresa, contabilidad y demás tareas de administración, por lo que no tendrá mucho tiempo para dedicarle a nuestra aplicación CMS.

- ***Socios, dueños y directivos de la empresa***
  Suelen disponer de más tiempo libre, pero acostumbran a estar muy ocupados con la dirección de la empresa, con lo cual deberemos evitar que la dificultad de uso les haga aburrirse y olvidar la aplicación.

2. Diseño conceptual. Arquitectura de la información

Al igual que en la primera aplicación, nos encontramos ante la interfaz de una aplicación software. Usaremos por tanto una organización de los contenidos de tipo ambiguo, por tareas. En cuanto a la navegación, estableceremos una estructura jerárquica, a partir de un menú principal.

El siguiente esquema detallado muestra la organización, navegación y etiquetado de dichas tareas.

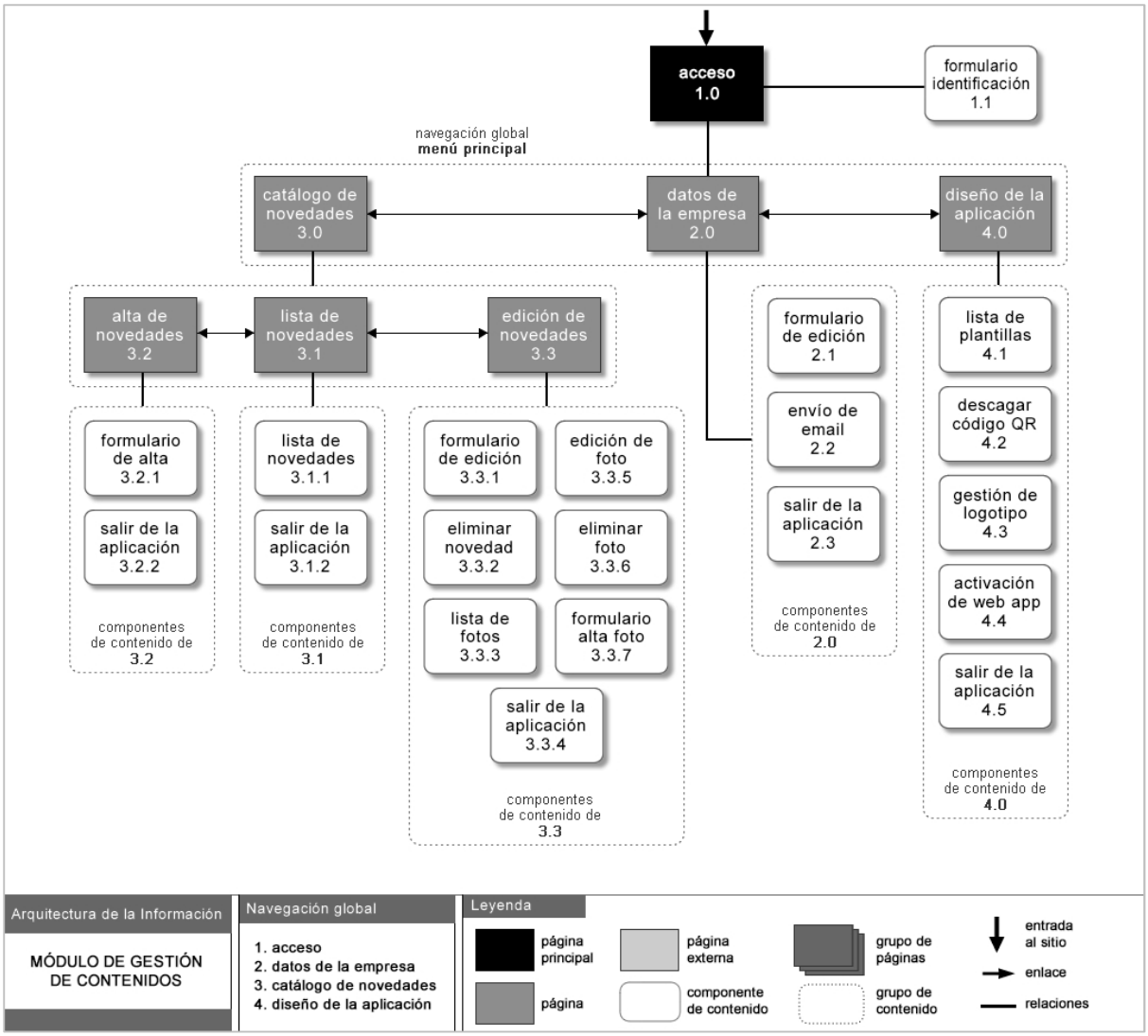

Figura 10: Esquema detallado. Arquitectura de la información de la aplicación

3. Diseño visual. Prototipos

De acuerdo con los perfiles de usuario establecidos en la fase de análisis, el éxito de la aplicación dependerá fundamentalmente de la usabilidad que su interfaz ofrezca, de manera que la curva de aprendizaje quede reducida a la mínima expresión.

Para conseguir este objetivo, se han aplicado principios básicos como la ley de Fitts, de modo que las tareas a realizar estén claramente visibles y accesibles en el mínimo tiempo posible, así como metáforas habituales en las interfaces de aplicaciones informáticas.

Se puede apreciar el resultado obtenido, en los prototipos de alta fidelidad que se muestran a continuación.

Figura 11: Prototipo de alta fidelidad de la vista de acceso de los usuarios del sistema

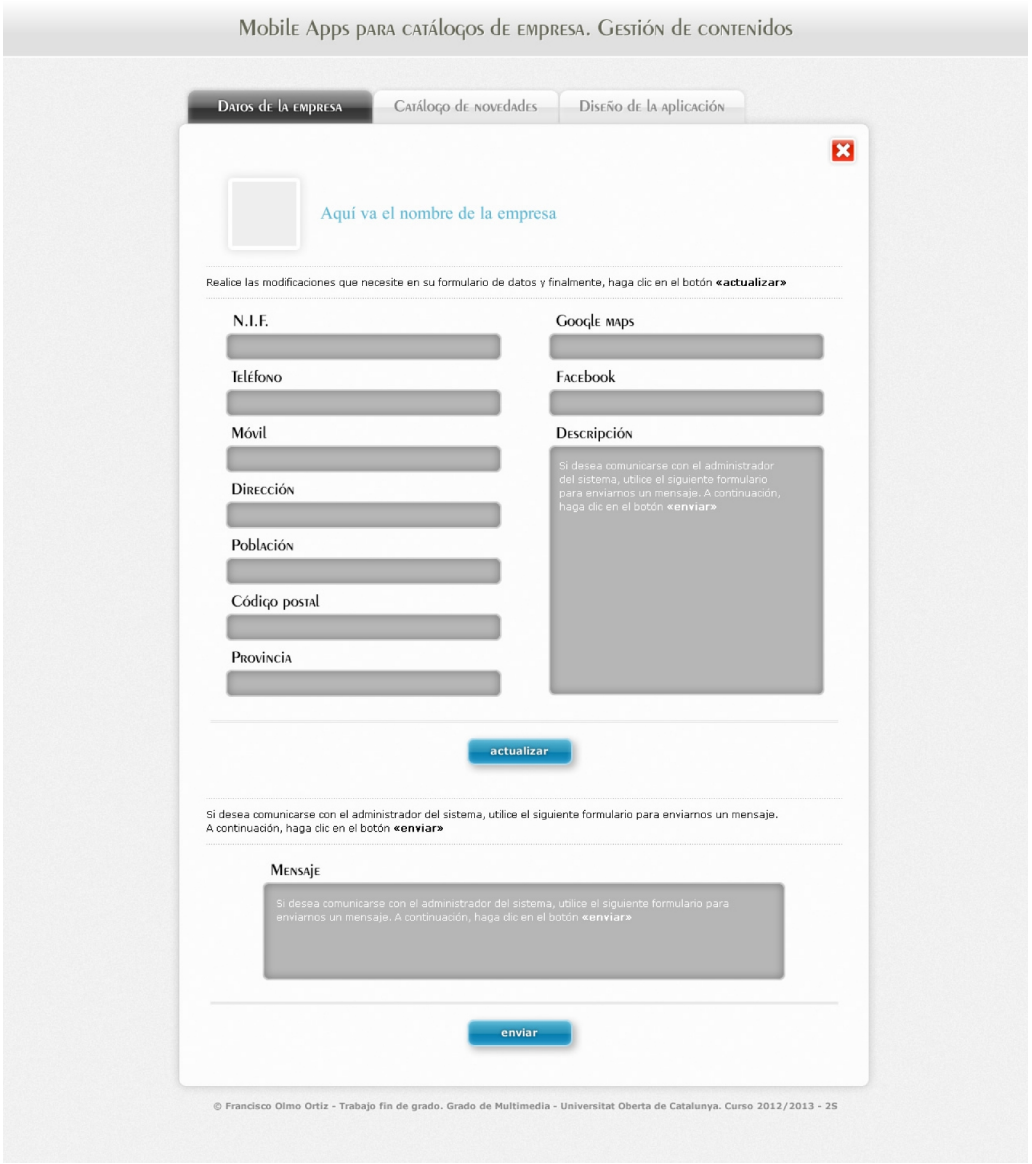

Figura 12: Prototipo de alta fidelidad de la vista de gestión de los datos de la empresa

Figura 13: Prototipo de alta fidelidad de la vista de gestión del diseño que mostrará la web app

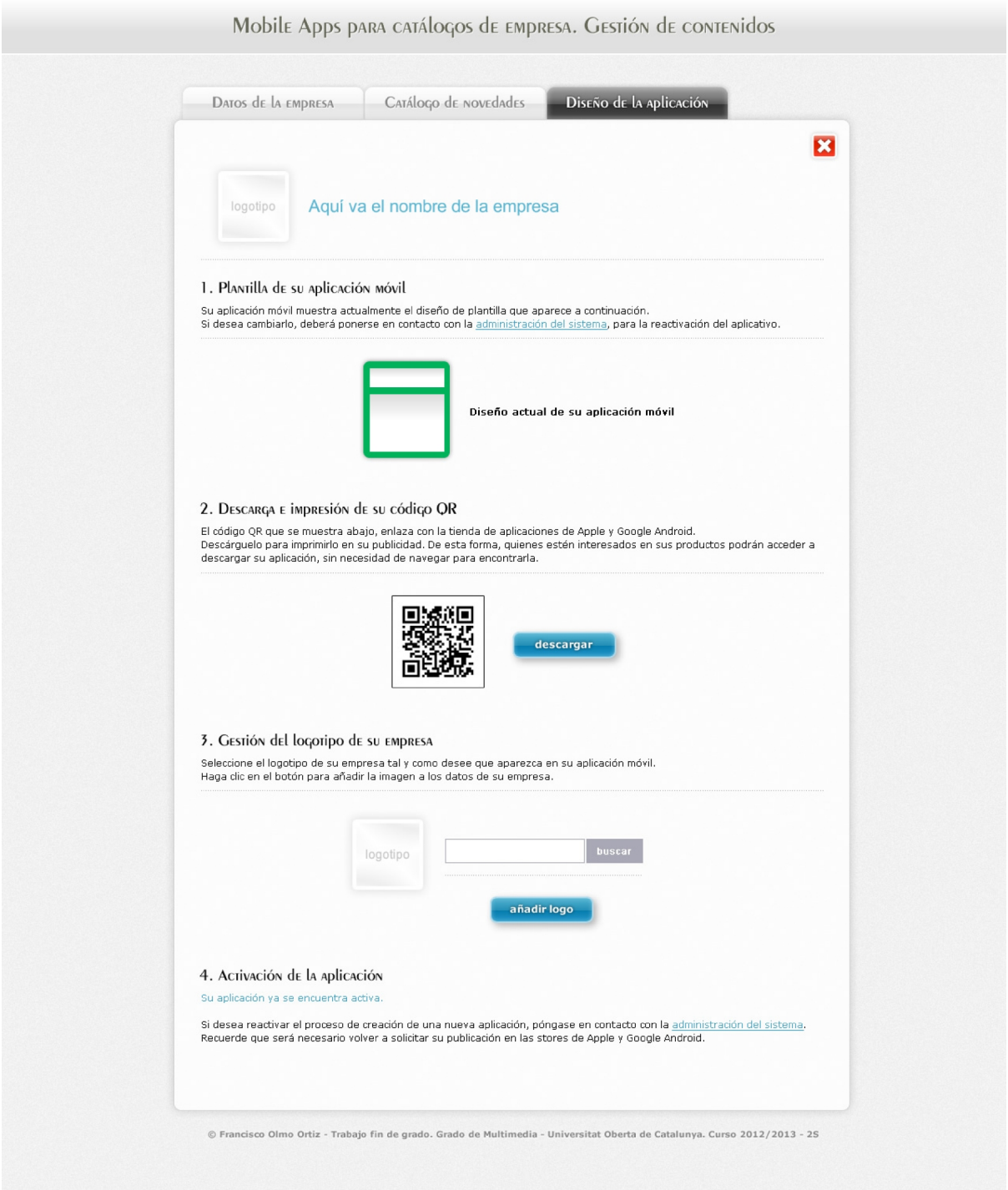

Figura 14: Prototipo de alta fidelidad de la vista de gestión del diseño que muestra la web app activa

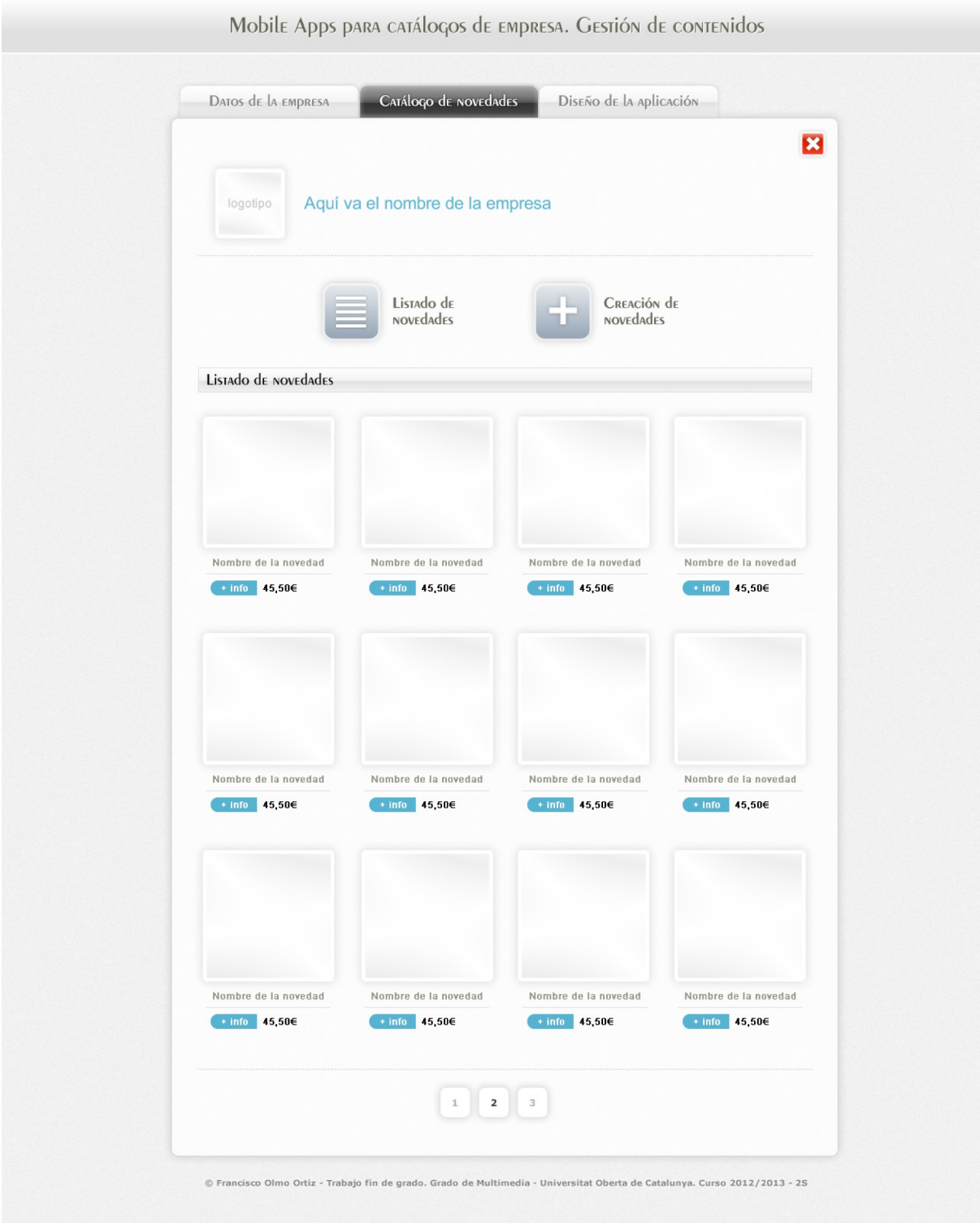

Figura 15: Prototipo de alta fidelidad de la vista de gestión del catálogo de novedades

Figura 16: Prototipo de alta fidelidad de la vista del alta de novedades

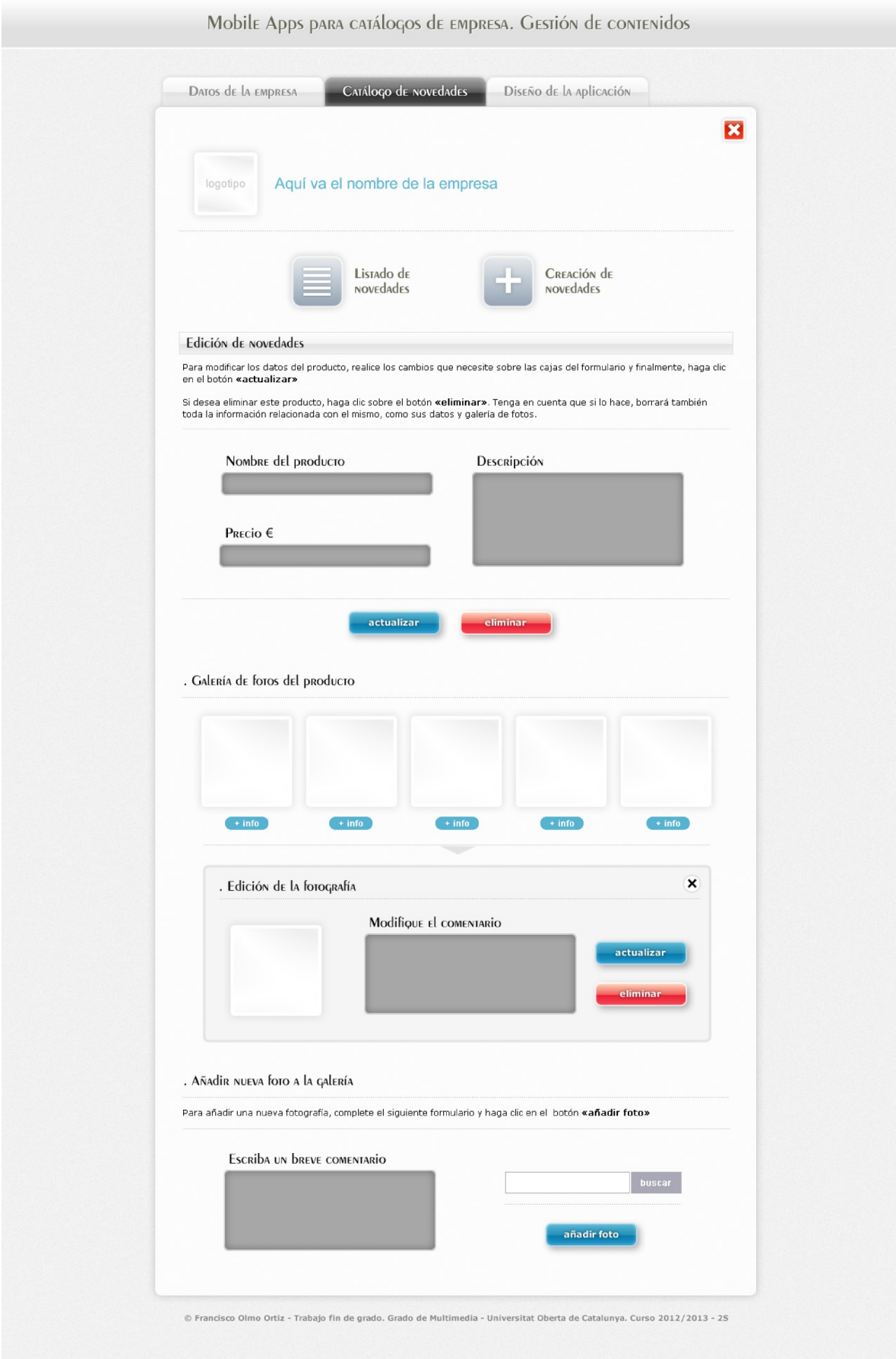

Figura 17: Prototipo de alta fidelidad de la vista de edición de novedades

Esquema modelo, vista, controlador (MVC)

De la misma manera que en la aplicación anterior, procedemos a la creación de los prototipos funcionales en lenguaje HTML y a la creación de todos los módulos de programación que componen la aplicación. La siguiente tabla, muestra el conjunto de ficheros que conforman el esquema MVC de dicha aplicación.

Recordemos que tal como se expone en el capítulo 2, cuando nuestro cliente está conforme con su logo y diseño de plantilla, clicará una opción, que generará en la carpeta del cliente, en el servidor web, un fichero XML el cual contendrá el nombre e identificador de la empresa, así como el identificador de la plantilla de la web app.

MODELO	VISTA	CONTROLADOR
conexion_db.php empresa.php producto.php	alta_novedades.php datos_empresa.php edicion_novedades.php index.php lista_novedades.php web_app.php	cierra_sesion_empresa.php identificacion.php muestra_alta_novedades.php muestra_datos_empresa.php muestra_edicion_novedades.php muestra_lista_novedades.php muestra_web_app.php realiza_activa_app.php realiza_alta_novedades.php realiza_borrado_foto.php realiza_borrado_novedades.php realiza_edicion_empresa.php realiza_edicion_foto.php realiza_edicion_novedades.php realiza_envio_email.php realiza_muestra_info_foto.php realiza_sube_foto.php realiza_sube_logo.php

- **MÓDULO DE LA WEB APP**

Diseño de la aplicación. Modelo DCU - Fases

1. **Análisis**

   En este tercer módulo, el objetivo será la creación de una aplicación instalable en dispositivos móviles que provenga de una web app, la cual será encapsulada con el framework PhoneGap. Esta aplicación, se podrá descargar, bien de las stores de apps para móviles o bien del propio servidor donde está ubicado nuestro sistema y se hará mediante el uso de un código QR.

Recordamos que tal código, fue creado por la aplicación del módulo de gestión de clientes y está disponible para que dichos clientes puedan descargarlo y usarlo impreso en su publicidad.

El público objetivo de esta aplicación, será cualquier persona que pueda estar interesada en la artesanía, en adquirir sus productos, ya sea particular o comerciante de este tipo de artículos. Por tanto, hablamos del público en general, con lo cual debemos pensar que la interfaz que ofrezcamos, tendrá que resultar muy similar en su uso a las aplicaciones que habitualmente solemos encontrar en las stores.

2. Diseño conceptual. Arquitectura de la información

En esta ocasión, nos encontramos ante un escenario prácticamente igual al de una web de contenidos dinámicos. Escogeremos una organización de sus contenidos, de tipo ambiguo por categorías. Así mismo, haremos uso de una estructura de navegación jerárquica, a partir de un menú principal de opciones.

Recogemos en el siguiente esquema detallado, la organización, navegación y etiquetado de dichas categorías.

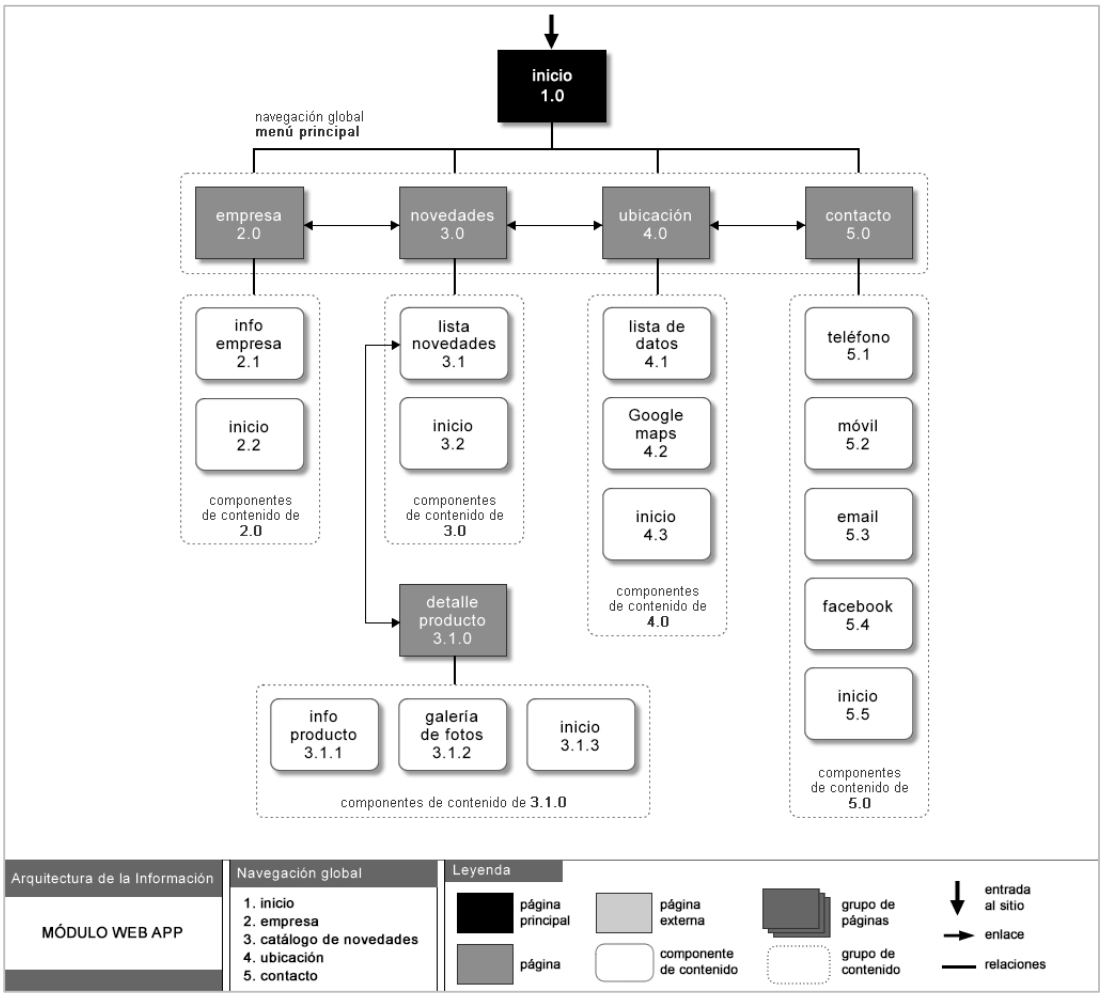

Figura 18: Esquema detallado. Arquitectura de la información de la aplicación

3. Diseño visual. Prototipos

Tal como se ha descrito en el apartado 1 de análisis, nos dirigimos a un público usuario genérico, acostumbrado al uso de aplicaciones nativas en sus smartphones y tablets. Debemos tener en cuenta por tanto, que será necesario dotar a nuestra app del *"look and feel"* de dichas aplicaciones nativas, para lo cual, haremos uso del framework JQuery Mobile que nos proveerá de buena parte de gadgets y animaciones, procedentes sobre todo del entorno Apple iPhone y nos introducirá además, en el ámbito del *responsive design* o diseño adaptativo. Este nuevo concepto, se refiere a la creación de diseños, especialmente web que se adaptan al tamaño de pantalla, del dispositivo que estemos usando para su visualización.

Se han creado para este proyecto, seis opciones de color, recogidas en sendas hojas de estilos CSS. La elección, quedará como ya se ha comentado anteriormente, en manos de la empresa cliente de nuestro servicio, la cual escogerá su plantilla, dentro de la sección "Diseño de la aplicación", descrita en el Módulo de gestión de contenidos. Recordamos también que nuestros clientes, podrán gestionar el logotipo de sus empresas desde la sección comentada.

Se puede apreciar el resultado obtenido, en los prototipos de alta fidelidad que se muestran a continuación.

Figura 19: Prototipos de alta fidelidad de la web app

Figura 20: Prototipos de alta fidelidad de la web app

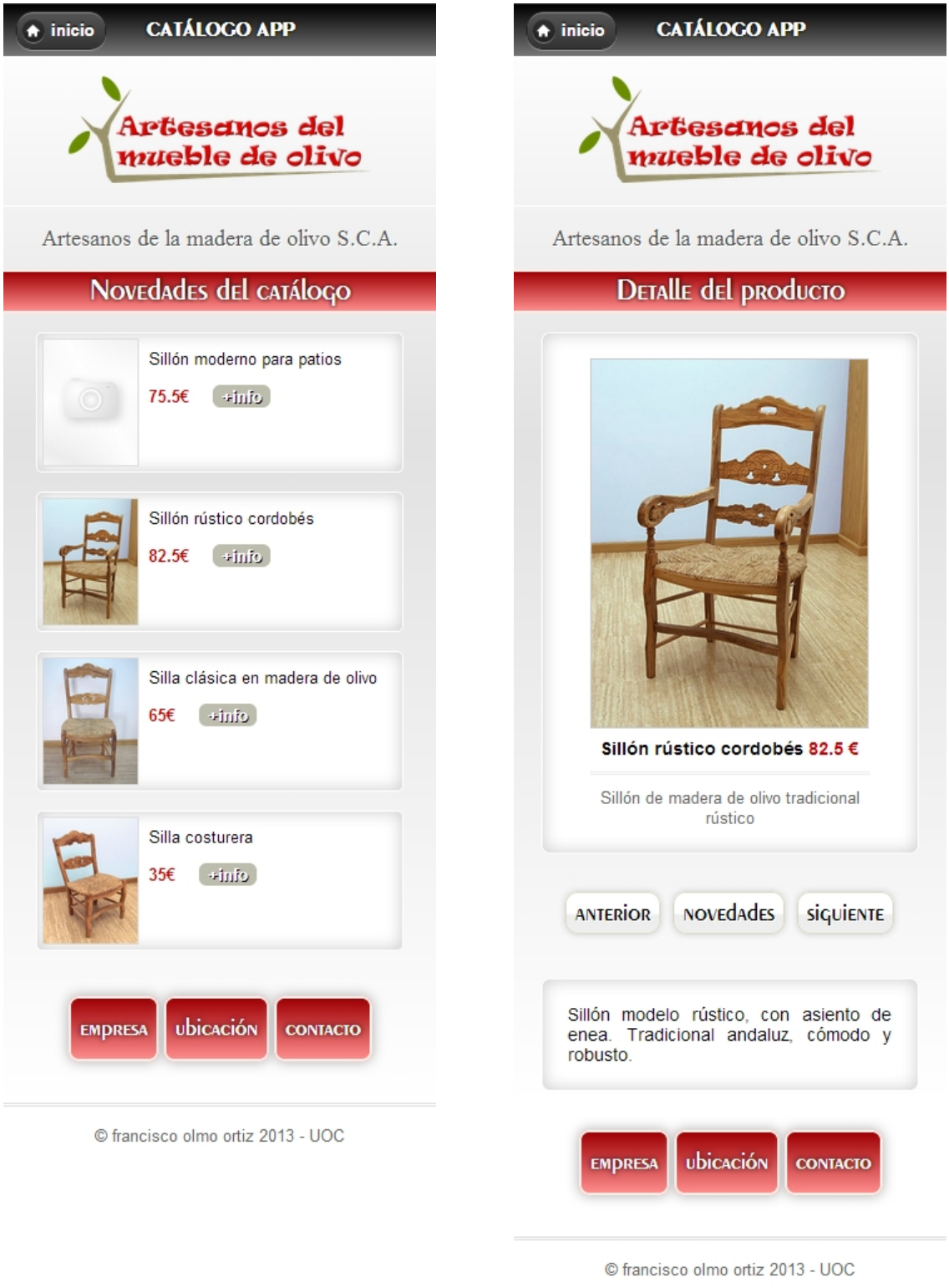

Figura 21: Prototipos de alta fidelidad de la web app

Figura 22: Prototipos de alta fidelidad de la web app

Esquema modelo, vista, controlador (MVC)

Procedemos a la creación de los prototipos funcionales en lenguaje HTML y a la creación de todos los módulos de programación que componen la aplicación.

En este caso, debemos tener en cuenta el hecho de que, ya que nuestra web app será encapsulada como app nativa, no podremos crear documentos en otro formato que no sea HTML, pues solo así, podrán ser interpretados por el navegador que habrá de mostrarlos.

Para conseguir la conexión de la aplicación con los datos de nuestra base de datos remota, haremos uso constante de la tecnología AJAX, mediante la cual, realizaremos llamadas asíncronas a nuestros controladores PHP, los cuales nos devolverán la información que solicitemos.

Los archivos que recogerán el código de las clases del modelo y el controlador, sí estarán programados en lenguaje PHP, ya que estos estarán alojados en el servidor web, al igual que el resto de nuestro sistema.

Mostramos finalmente la tabla que lista el conjunto de ficheros del esquema MVC de nuestra web app que será convertida a aplicación nativa.

MODELO	VISTA	CONTROLADOR
conexion_db.php app.php	index.html Documento JQuery Mobile formando por los siguientes elementos con atributo "data-rol=page": # index # inicio # empresa # contacto # ubicacion # novedades # detalle_producto	muestra_datos_empresa.php muestra_detalle_novedad.php muestra_novedades_empresa.php

Encapsulado de la aplicación - Uso de PhoneGap Build

Una vez hemos concluido la programación de la aplicación y realizadas todas las pruebas pertinentes, como si de una web normal se tratase, llegamos a la fase final del proceso que nos permitirá convertir nuestra web app en una aplicación nativa, haciendo uso del código que hemos programado, el mismo para todos los sistemas móviles en que estemos interesados.

Para conseguir este propósito, hemos optado por hacer uso del framework PhoneGap, el cual es una herramienta que no solo nos ofrece la posibilidad de encapsular web apps en nativas, sino que además provee de una extensa API que permitirá hacer uso de las utilidades hardware que habitualmente incorporan los dispositivos móviles, smartphones y tablets. En el proyecto que hemos desarrollado, no hacemos uso de dichas librerías, pues nuestro objetivo, tal como se expone en la introducción, no es otro que mostrar el proceso que actualmente nos permite obtener una aplicación nativa, haciendo uso de las tecnologías web que ya conocemos y usamos a diario.

PhoneGap, nos ofrece un servicio on-line para realizar el encapsulado de nuestra web app, conocido como PhoneGap Build, el cual simplifica enormemente nuestra labor, al no tener que instalar en nuestro ordenador, los SDK de todas las plataformas para las que estemos interesados en obtener nuestra app nativa.

PhoneGap Build necesita en principio, tan solo dos documentos que habrán de llamarse "index.html" (que contendrá nuestro código) y otro de configuración "config.xml". Si nuestro proyecto consta de otros archivos, scripts, imágenes, hojas CSS u otros, todos deberán ser comprimidos a un solo fichero en formato ZIP. Mostramos un ejemplo del archivo XML de configuración:

```
<?xml version "1.0" encoding="utf-8"?>
<widget xmlns = "http://www.w3.org/ns/widgets"
        xmlns:gap = "http://phonegap.com/ns/1.0"
        id        = "com.phonegap.tfg"
        version   = "1.0.0">

    <name>Folmo App</name>
    <description>
       TFG Grado Multimedia - UOC. App Phonegap para catálogos de productos
    </description>
    <author href="http://www.afosoft.com" email="paco@afosoft.com">
        Francisco Olmo Ortiz
    </author>
</widget>
```

Una vez tenemos preparada nuestra compilación de archivos ZIP, accedemos a la web de PhoneGap Build y nos identificamos si ya estamos registrados o nos subscribimos como usuarios, ya que es requisito indispensable para su uso.

En la siguiente captura de pantalla, se muestra la interfaz de creación de aplicaciones nativas multiplataforma.

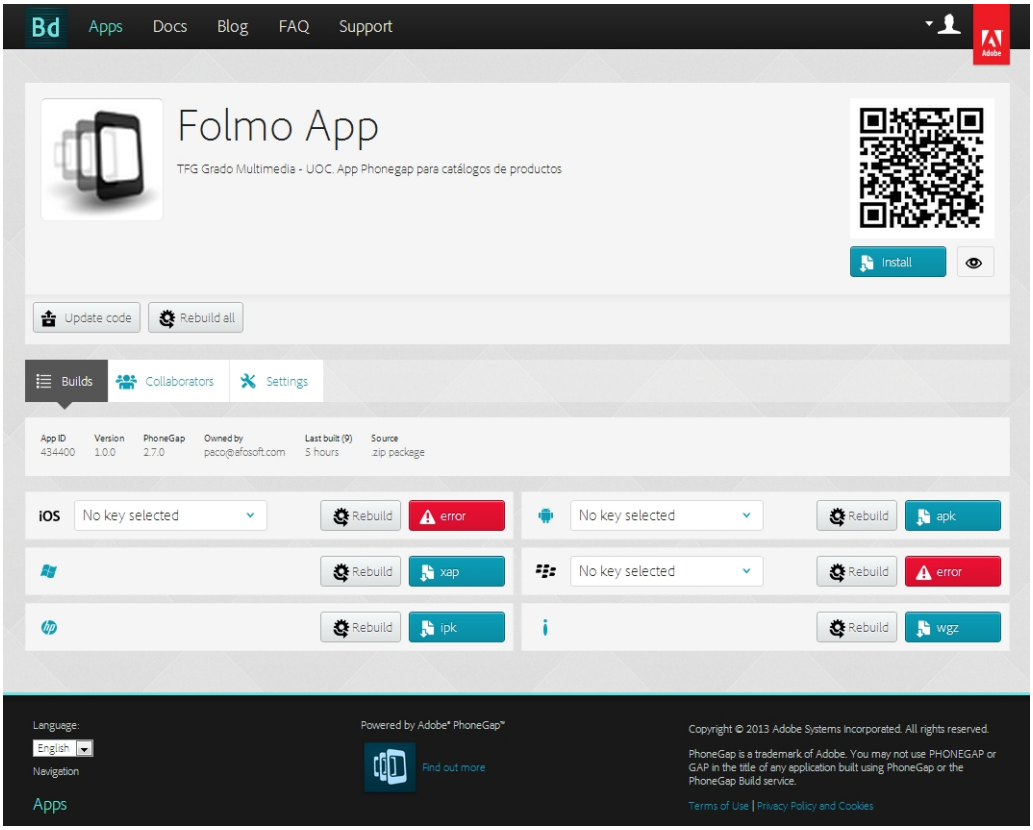

Figura 23: Creación de aplicaciones nativas en PhoneGap Build

Podemos comprobar como PhoneGap Build, ha compilado la web app para los distintos sistemas operativos móviles que soporta. Los botones azules nos indican que las apps nativas, en estas plataformas, están listas para ser descargadas.

Vemos además que para algunos sistemas como iOS nos aparece su botón en rojo, indicando *error*. El motivo es que para poder obtener una aplicación nativa en iOS, Apple exige estar en posesión de los certificados de desarrollo y distribución, lo cual tiene un coste anual que ronda los 100 dólares. Para poder compilar con PhoneGap Build para iOS, tendremos que subir junto con la compilación de ficheros ZIP, ambos certificados.

Indicar por último que tal como se comentó en el capítulo 1 de Introducción, el producto final que obtendremos en este proyecto, será una app nativa para Android 4.x, por lo que procedemos a clicar sobre su botón y descargar la aplicación.

10. Seguridad

Afrontamos el tema de la seguridad desde el ámbito de las aplicaciones web que habitualmente representa un escenario muy problemático, debido a la constante aparición de nuevas amenazas, así como al hecho de que nuestra capacidad de previsión es relativamente reducida, al estar expuestos a la globalidad inherente a la red Internet.

No obstante, vamos a aplicar a nuestro sistema una serie de medidas, más o menos estandarizadas que de alguna manera, suponen un protocolo de seguridad en el desarrollo de aplicaciones y servicios web.

- **Seguridad de la plataforma**

A. Configuración de permisos para los usuarios de las carpetas y ficheros de nuestro sistema, una vez ubicados en el servidor web definitivo, para la fase de producción. Mediante el comando *chmod*, estableceremos a **755** los permisos de todas las carpetas, excepto aquellas que almacenarán ficheros publicados o generados por usuarios públicos, es decir, desde la web, las cuales estarán configuradas a ***777***. En el primer caso, 755, establecemos que ningún usuario desde un navegador web, podrá realizar otra operación que no sea la simple lectura de los archivos. Para el caso de las carpetas 777, tendremos la precaución de no albergar en ellas scripts con código de programación, así como impedir que usuarios públicos puedan hacerlo, usando para ello un fichero de tipo *.htaccess* de Apache. Mostramos el código de configuración del fichero:

```
<Files *.php>
Order Deny,Allow
Deny from all
</Files>

<Files *.html>
Order Deny,Allow
Deny from all
</Files>

<Files *.htm>
Order Deny,Allow
Deny from all
</Files>

<Files *.js>
Order Deny,Allow
Deny from all
</Files>
```

Vemos que se deniega toda actividad a ficheros de tipo PHP, HTML, HTM y JS.

B. Creación de perfiles de usuario distintos, para el propietario o administrador y para los usuarios públicos de la base de datos. Sería la configuración óptima, siempre que el servicio que contratemos para alojar nuestro sistema lo permita. De este modo, creamos un usuario con todos los permisos de gestión de la base de datos, el administrador y un segundo tipo, para los usuarios públicos que solo tendrán permisos para actividades de edición, como insertar, borrar o modificar contenidos en ciertas tablas. Vemos en las siguientes capturas de pantalla, la configuración de ambos perfiles:

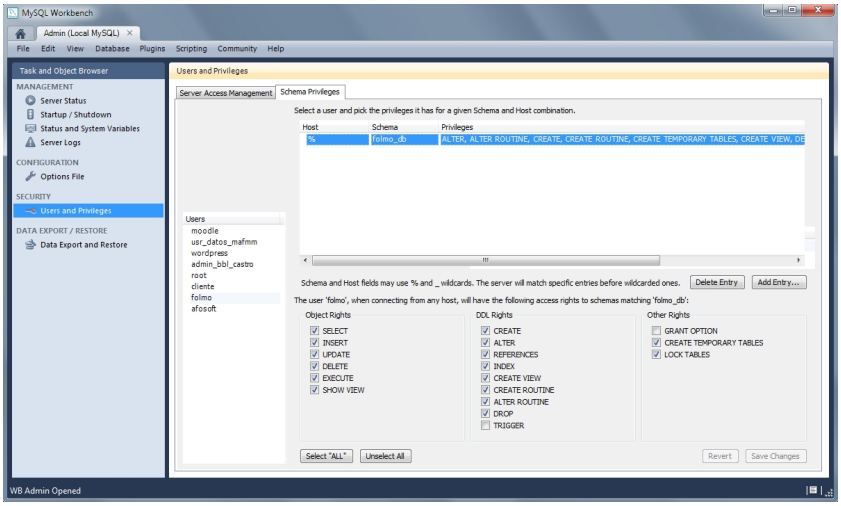

Figura 24: Configuración del usuario administrador ´folmo'

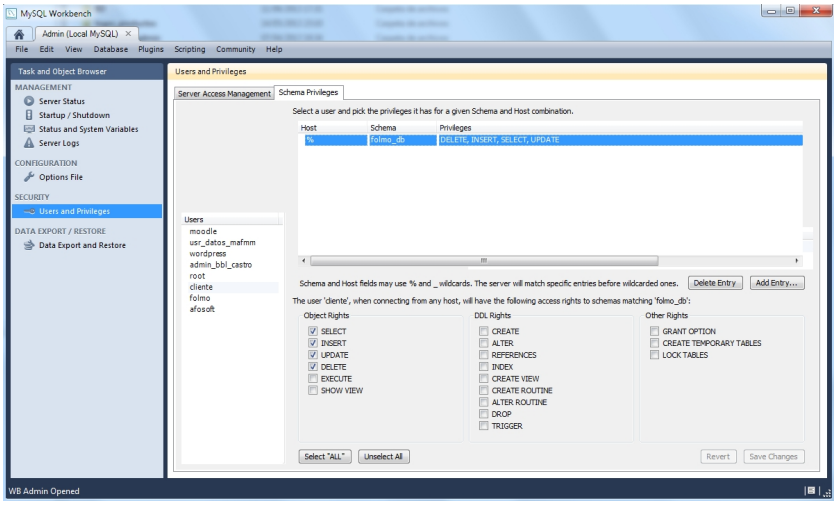

Figura 25: Configuración de los usuarios públicos 'cliente'

- **Seguridad del software**

Nos referimos en este caso a las medidas de seguridad que podemos tomar a la hora de desarrollar nuestro código de programación. Mostramos un listado con dichas medidas, comenzando por la más básica, hasta las más sofisticadas o específicas.

A. Uso del modo **POST** frente al GET, para el envío de formularios HTML al servidor, especialmente aquellos que contengan información que posteriormente vaya a ser mostrada en los navegadores cliente. Es una medida frente ataques del tipo *CSFR (Cross site request forgery)*.

B. Validación correcta de los formularios HTML, de modo que obliguemos a que en sus campos solo se pueda introducir información en el formato especificado. Mostramos algún ejemplo de código, usado en la aplicación:

```
if (formulario.precio.value !="")
{
    if (!isNaN(formulario.precio.value))
    {
            if (formulario.descripcion.value !="")
            {
                    actualizar("form_datos");
            }
            else
            {
                    alert("indique la descripción del producto");
            }
    }
    else
    {
            alert("indique una cantidad para el precio del producto");
    }
}
```

Comprobamos en el script que el campo *precio* sea un dato numérico y que *descripcion* no esté vacío.

C. Escapado correcto de las cadenas de caracteres que llegan al servidor, mediante funciones específicas del lenguaje PHP. Su uso es habitual para evitar ataques de tipo inyección de código SQL y del tipo XSS o *Cross site scripting attacks*. En ambos casos, se inserta código malicioso en los formularios de nuestros documentos HTML. Para el caso de la inyección SQL, el código insertado intenta ejecutar acciones en el servidor sobre nuestra base de datos, hasta incluso llegar a borrarla, realizando una instrucción del tipo *drop database*. Para evitarla, hemos hecho uso de la función PHP *mysql_real_escape_string* que escapará todos los caracteres considerados peligrosos.

En el segundo caso, ataques XSS, el modus operandi es similar, salvo que en esta ocasión, el objetivo es ejecutar código en los navegadores cliente, generalmente mediante JavaScript. Para

evitarlo, usamos la función PHP *htmlentities* que escapa todos los caracteres válidos para la codificación HTML. Mostramos un ejemplo de uso:

```
mysql_real_escape_string(htmlentities($nombre,ENT_QUOTES));
```

D. Codificación de claves de máxima seguridad. Por ejemplo para la contraseña del administrador del sistema. Para ello, se ha usado el comando PHP **md5**, que nos devuelve una cadena encriptada en formato hexadecimal de 32 dígitos.

11. Tests

Tal como ya se ha descrito en el capítulo 9 del proceso de trabajo, los tres módulos que componen el proyecto son aplicaciones interactivas en el entorno web. Por este motivo, han sido desarrolladas de acuerdo al modelo de diseño centrado en el usuario. En este sentido, una de las etapas de esta metodología de trabajo, es el análisis de la usabilidad, cuyos principios o heurísticas se han tenido en cuenta en todo el proceso de desarrollo, como parte inherente al mismo.

Independientemente de los tests de usabilidad, hemos centrado este capítulo de pruebas, en la compatibilidad de la aplicación móvil, con los principales sistemas operativos y versiones de los mismos, con el objetivo de comprobar el grado de cumplimiento de la idea central del proyecto: utilizar las tecnologías de desarrollo web, para la creación de aplicaciones móviles nativas, de acuerdo con el principio de escribir una sola vez el código y ejecutarlo en plataformas distintas. Mostramos a continuación una tabla, con los resultados obtenidos sobre una serie de tests de compatibilidad.

Dispositivos	Compatibilidad de la aplicación	Fluidez en la ejecución	Compatibilidad con Google Maps
HTC Wildfire S Android 2.2.1	✓	X	✓
Samsung Galaxy mini Android 2.3	✓	✓	X
Samsung Galaxy I Android 2.3	✓	✓	X
Samsung Galaxy III Android 4.0	✓	✓	✓
Tablet Woxter 10.1" Android 4.1	✓	✓	✓
iPhone 3GS iOS 6.1.3	✓	✓	✓
iPad pantalla Retina iOS 6.0.1	✓	✓	✓

Óptimo Aceptable No compatible

Podemos comprobar cómo la aplicación desarrollada exclusivamente con tecnologías web, es capaz de ejecutarse en dispositivos bajo sistemas Android e iOS, con el mismo código, sin tener que realizar ningún cambio en el mismo.

Vemos también que se obtienen resultados poco óptimos e incluso negativos, en temas puntuales y con versiones antiguas de Android. Se trata de la fluidez con la que la aplicación realiza las transiciones o animaciones entre pantallas, así como la inserción de Google Maps mediante el uso de la etiqueta HTML ***<iframe>***. Este bug es debido exclusivamente, a la versión del renderizador ***webkit*** incluido en el navegador, de las versiones Android hasta la 2.3 que mostraban una pobre adaptación de los nuevos estilos CSS3. Esta versión, no interpreta aspectos como las sombras tipo *box-shadow*, pero su carencia más acusada es no soportar las animaciones mediante *animation y @keyframes*, por lo que el framework JQuery Mobile, utiliza para suplir esta carencia animaciones JavaScript, ofreciendo una velocidad poco óptima al tratarse de un lenguaje interpretado, frente a la ejecución de CSS3, compilada en lenguaje C++.

Finalmente comentamos que la ejecución en formato de página web no difiere en absoluto, del ofrecido una vez encapsulada la aplicación con PhoneGap, pues en ambos casos, quien se encarga de su visualización es el navegador web del sistema operativo.

Una vez terminado todo el proceso de desarrollo del proyecto y realizados los test y comprobaciones aquí expuestos, entregamos junto a esta memoria, la versión 1.0 de la aplicación, lista para ser instalada y usada, de acuerdo con los requisitos e instrucciones que se indican en los capítulos siguientes.

12. Requisitos de instalación

- **Software**

  Para la implantación y ejecución de las aplicaciones de gestión, deberemos contar con un servicio web, bien sea *hosting* o un sistema virtualizado *cloud*, bajo plataforma Linux, que nos ofrezca instalados, un servidor web Apache 2, soporte PHP 5 y servidor de bases de datos MySQL 5. Para hacer uso de estas aplicaciones, tan solo será necesario que nuestro ordenador cuente con conexión a Internet y un navegador actualizado, compatible con HTML5/CSS3.

  Para la instalación y ejecución de la aplicación nativa, podremos usar cualquier dispositivo móvil Apple iOS o Android preferiblemente con sistema operativo 4.x.

- **Hardware**

  Para las aplicaciones de gestión, bastará con un PC con capacidad para conectarse a Internet y para la aplicación nativa, podremos usar cualquier dispositivo Apple iOS o Android, preferentemente con la versión 4.x del sistema.

- **Formación/Conocimientos**

  En cuanto a la formación que se necesita para el uso de cualquiera de las tres aplicaciones desarrolladas, decir simplemente que no se requieren más conocimientos que los necesarios para el uso básico de utilidades como las que a diario usamos en Internet: Facebook, Twitter, comercio electrónico y similares.

13. Instrucciones de instalación/implantación

A. Instalación de la app nativa

Tan solo será necesario, enfocar con un dispositivo móvil Apple iOS o Android 4.x, el código QR que la aplicación generó para cada cliente de nuestro sistema y que podrá encontrarse impreso en cualquier publicidad de dicha empresa, o bien desde su página web, mediante un enlace de descarga directa de la aplicación. También, a través de la Apple Store o la Play Store de Google.

B. Instalación de las aplicaciones de gestión

- **Base de datos del sistema**

En la subcarpeta del proyecto, de nombre *base_de_datos*, encontramos el fichero *folmo_bd_uoc.sql* que contiene todas las instrucciones necesarias para crear la base de datos de nuestra aplicación.

En primer lugar, crearemos en el servidor del servicio *hosting* o *cloud* que hayamos contratado, una base de datos de nombre ***olmo***. A continuación, crearemos un usuario para la misma, con los siguientes datos:

- nombre de usuario: ***olmo***
- contraseña: ***FcKL7200***

Si el servicio contratado ya nos dio una base de datos configurada, tendremos que editar el fichero *folmo_bd_uoc.sql*, realizando un operación de *buscar/reemplazar* de la cadena *olmo*, por el nombre de la base de datos que nos haya asignado el servidor.

A continuación, procederemos a importar este fichero SQL desde el servidor remoto *(se incluyen algunos datos de prueba)*. En caso de haber tenido que cambiar el nombre de la base de datos o los parámetros de conexión, tendremos que hacerlo también en nuestra aplicación. Para ello, tan solo será necesario editar el fichero *conexion_db.php* de la carpeta modelo y cambiar los datos que allí aparecen, por los que nos haya suministrado la empresa que nos provea el servicio de servidor.

- **Ficheros de la aplicación**

Accederemos al espacio de la carpeta web de nuestro servidor remoto, a través de cualquier cliente FTP y directamente en su directorio raíz, copiaremos todos los ficheros que nos encontramos dentro de la carpeta *aplicacion* del proyecto.

No necesitaremos más configuración, que asignar permisos mediante *chmod*, de nivel 777 a la carpeta *biblioteca* y asegurarnos de que la misma, contiene al fichero *.htaccess*.

Podemos ya acceder a nuestra aplicación, desde el navegador, en la URL *path/ app_administrador*.

14. Instrucciones de uso

En el presente capítulo, procederemos a explicar el funcionamiento de cada una de las tres aplicaciones que componen el proyecto desarrollado. Para una mejor comprensión de la información que se ofrece a continuación, les remitimos al capítulo 9, donde mostramos capturas de todas las pantallas del sistema, con objeto de no repetir de nuevo las mismas imágenes.

A. Aplicación de gestión de clientes

Accedemos a la aplicación a través de la URL: *http://multimedia.uoc.edu/~olmo/app_administrador/*, donde nos aparecerá el formulario de identificación del administrador. Introducimos la contraseña, *rrswindle*. Vemos de entrada, el listado de empresas que actualmente están de alta en el sistema. Se nos ofrece como información, el nombre de la empresa y su fecha de alta, así como una indicación de color que nos informa si el cliente, desde su gestor de contenidos, activó ya la orden para que procedamos a generar su app nativa.

En la parte superior, tenemos el menú principal, con tres opciones, para el ***listado de empresas***, ***crear nueva empresa*** y ***salir de la aplicación***.
Si clicamos sobre el nombre de una empresa del listado, se nos mostrará la pantalla de *edición de cliente*. En ella aparecen las siguientes opciones:

- **Actualizar**. Guardará en base de datos los cambios que escribamos sobre el formulario, en los campos de nombre y e-mail de la empresa
- **Eliminar**. Borrará a la empresa de la base de datos. Esto solo será posible si el cliente aún no introdujo productos ni activó la orden de generado de su app nativa
- **Estado del cliente**. Nos permite activar/desactivar al cliente en el sistema

Si optamos por dar de alta a un nuevo cliente, la aplicación nos ofrece un formulario donde introduciremos el nombre y e-mail de la empresa. Al clicar sobre el botón *dar de alta*, la aplicación generará un identificador de empresa, así como una contraseña de acceso al gestor de contenidos. Así mismo, se creará una carpeta, dentro de la carpeta *empresas*, cuyo nombre será el identificador de la empresa dada de alta. Por último, se creará un código QR en formato PNG, también de nombre el mismo identificador de la empresa, en la carpeta *biblioteca/qr*. Dicho código enlazará con la aplicación nativa que generaremos para el cliente, que se denominará ***catalogoapp.apk*** y que colocaremos para su descarga, en la carpeta *path/empresas/idempresa*.

La última opción, *salir de la aplicación*, borrará la sesión de usuario y nos devolverá al formulario de identificación.

B. Aplicación de gestión de contenidos

Los clientes del sistema, accederán a la aplicación a través de la URL: http://multimedia.uoc.edu/~olmo/ donde se identificarán con su email y la contraseña suministrada por el administrador de la empresa. Dado que esta aplicación es más compleja que la primera, haremos una descripción de las opciones que nos ofrece cada pantalla:

- **Datos de la empresa**

Disponemos de un amplio formulario desde donde introducir y actualizar nuestros datos. Para el caso de la inserción de Google Maps, utilizaremos la dirección URL completa que nos ofrece este servicio. El usuario de la aplicación cuenta también, con otro formulario para enviar mensajes mediante correo electrónico, al administrador del sistema. Para que esta utilidad esté funcional, habrá que indicar en el fichero de configuración *php.ini*, los datos del servidor *SMTP* adecuados.

- **Diseño de la aplicación**

Esta pantalla, nos ofrecerá dos posibles vistas, antes y después de que el usuario haya pulsado la opción de *activar la aplicación*. Antes de hacerlo, se nos muestra una lista con todas las plantillas disponibles, para que escojamos la que deseemos. Una vez hemos ordenado activar la aplicación, tan solo podremos ver la plantilla que elegimos, pues la app nativa ya existirá con este diseño.

Contamos además con una opción para descargar la imagen PNG del código QR que se generó cuando se dio de alta a la empresa, la cual podremos utilizar impresa en nuestra publicidad.

Una tercera utilidad de esta pantalla nos permite gestionar el logotipo de la empresa, tantas veces como necesitemos, pues recordemos que la aplicación móvil leerá esta información del servidor remoto cada vez que se ejecute.

Por último, la cuarta opción será la ya comentada que nos permitirá activar la aplicación móvil, lo cual quedará automáticamente reflejado en la aplicación del administrador que procederá a encapsular la web app, en formato de aplicación nativa para descargar e instalar. Indicar que en este proceso, se generará un fichero XML que contendrá información del identificador y nombre de la empresa, así como de la plantilla escogida. Este fichero quedará grabado en la carpeta de la empresa que se generó junto a su QR, al ser creada desde la aplicación de gestión de clientes. Servirá para que al ejecutarse al aplicación móvil, muestre ya el nombre de la empresa y su plantilla y sobre todo, busque en el servidor remoto, los datos de dicha empresa.

- **Listado de novedades**

Se muestra el listado de novedades de la empresa, de las cuales podemos ver una foto, si existe, así como el nombre, precio y enlace para ampliar su información. Tenemos en el menú secundario de esta pantalla, la opción de dar de alta un nuevo producto como novedad del catálogo.

- **Creación de novedades**

La creación de novedades del catálogo, la haremos en dos pasos. En primer lugar la daremos de alta con sus datos básicos: nombre, precio y descripción, desde esta pantalla. Una vez creada, se nos mostrará ya en el listado, lógicamente aún sin foto, desde donde podremos clicar en su botón *+info*, para pasar a la pantalla de edición.

- **Edición de novedades**

Un primera sección de esta pantalla, nos permite modificar los datos del producto, así como eliminarlo del catálogo. A continuación, contamos con la gestión de la galería de fotos de dicho producto; se nos muestra un listado con todas sus fotos las cuales van acompañadas de un botón *+info* que nos permite abrir una herramienta desde donde podemos modificar el texto de su pie de foto, así como eliminar dicha foto de la galería. Por último, tenemos un formulario para agregar nuevas fotografías a la galería, acompañadas de su texto al pie.

En todas las pantallas, contamos con un botón en la esquina superior derecha, para abandonar la sesión de usuario y salir de la aplicación.

C. Aplicación móvil nativa

La web app encapsulada con PhoneGap, ofrecerá a los usuari@s que la instalen, las siguientes pantallas y opciones:

- **Carga inicial de datos**

La aplicación lee su fichero *startup.xml*, configura el nombre de la empresa y la plantilla elegida y carga los datos de la empresa del servidor remoto, a excepción de los relativos al catálogo de novedades. Clicamos en el botón *comenzar* para entrar al menú principal.

- **Menú principal de la aplicación**

Se nos ofrecen botones de acceso a cada una de las cuatro secciones de la aplicación.

- **La empresa**

Mostramos el texto de información de la empresa.

- **Novedades del catálogo**

Vemos una lista con todas las novedades del catálogo. De cada una de ellas, veremos una foto en miniatura, el nombre del producto y su precio. Seleccionamos el que deseemos consultar y pasamos a su pantalla de detalle.

- **Detalle del producto**

La aplicación nos muestra la información del producto elegido: nombre, precio y descripción, así como una galería de fotos con sus textos al pie. Si existe más de una foto, tenemos dos botones, *anterior* y *siguiente*, para cambiarlas. Por último, el botón *novedades* nos devolverá al listado de novedades.

- **Contacto**

Tenemos aquí opciones para realizar una llamada al teléfono y al móvil de la empresa, enviarle un e-mail, así como un enlace a su web en *Facebook*.

- **Ubicación**

Por último, la sección de *Ubicación* nos ofrece información acerca de dónde podemos encontrar a la empresa. Nos muestra también un mapa de posición a través del servicio *Google Maps*.

15. Proyección a futuro

El producto obtenido como resultado del presente proyecto, no es sino una base sobre la que evolucionar todo un sistema de generación automática de aplicaciones móviles, desarrolladas con tecnologías web. Nuevos servicios y utilidades, se irían añadiendo progresivamente, para hacerla más atractiva y por tanto, más competitiva.

Mostramos a continuación, una serie de ideas que en conjunto, podrían suponer la siguiente versión de nuestro sistema:

- Ampliación del soporte a dispositivos móviles basados en Windows Phone y BlackBerry
- Uso de la API de PhoneGap para combinar la geolocalización con el posicionamiento sobre mapas de Google Maps
- Captura de fotos desde la cámara del dispositivo móvil mediante la API PhoneGap y subida al servidor, para usar en el catálogo de productos
- Aplicación del *responsive design* para crear versiones de la aplicación específicas para smartphone y tablet
- Aumentar las posibilidades de gestión del diseño de la aplicación, más allá de la actual elección del color base
- Añadir al módulo de gestión del catálogo de novedades, la posibilidad de crear varios catálogos que pueden albergar tipos distintos de productos
- Crear una aplicación de comercio electrónico que permita comprar los productos a través de la aplicación

Todas estas propuestas, son tan solo una muestra del amplio abanico de posibilidades que comienzan a ofrecer las aplicaciones basadas en plataformas móviles. No obstante, la velocidad en su evolución y la constante aparición de nuevas tecnologías, nos hace reflexionar sobre la conveniencia, más que nunca, de apostar por los sistemas de desarrollo basados en estándares web, frente al enorme esfuerzo que representa el intentar dominar la programación de aplicaciones, para un espectro tan grande de sistemas propietarios, como ya existen en el mercado y que todo parece indicar, irá en aumento a corto plazo.

16. Comercialización del producto

El presente proyecto no incluye capítulos o anexos relativos a la valoración de un producto o su comercialización, ya que el objetivo final, no es la concreción de tal producto o servicio preparado para su venta, sino el desarrollo de una metodología de trabajo y el uso de unas determinadas tecnologías que hagan posible, crear aplicaciones móviles de calidad profesional que puedan competir en el mercado actual.

De esta manera, podemos llegar a la conclusión de que la creación de aplicaciones nativas mediante el uso de estándares web, supone una opción muy válida como método de desarrollo multiplataforma. Tal vez esto aún no sea posible para todos los tipos de aplicaciones, pero sí que lo es para casos como el planteado en este proyecto: aplicaciones para uso publicitario o comercial de pequeñas empresas.

El hándicap que supone usar como renderizador al navegador del sistema operativo móvil, por ser significativamente más lento que los métodos nativos, va paliándose poco a poco, sirva como ejemplo este gráfico de la fundación Mozilla, donde se demuestra la velocidad a la que están evolucionando las tecnologías web, en este caso mediante su nuevo optimizador de JavaScript *OdinMonkey*.

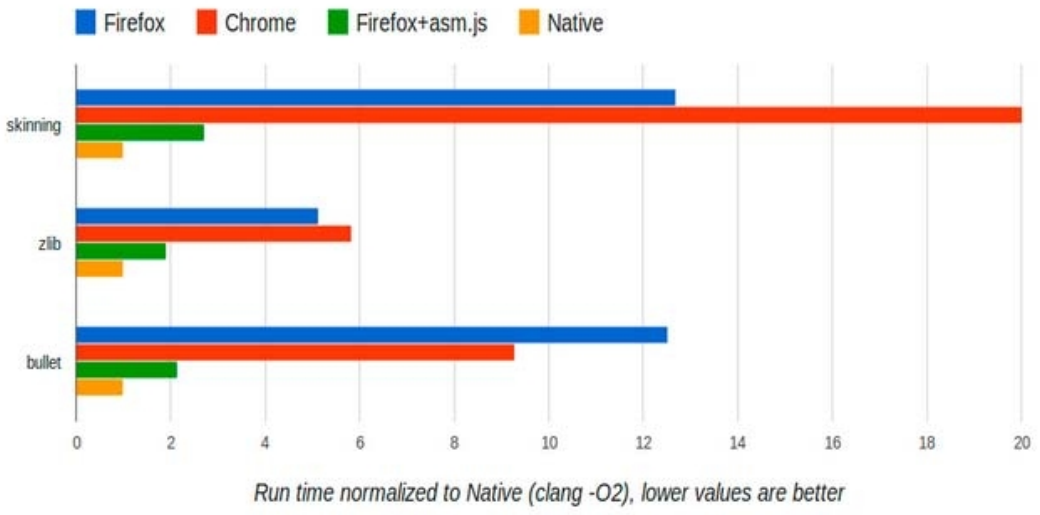

Figura 26: En verde, rendimiento del optimizador JavaScript OdinMonkey

Obviamente el aspecto comercial o económico del uso de sistemas de desarrollo multiplataforma, será uno de los más beneficiados, pues es fácil imaginar la reducción de costes que supone para las empresas de software, no tener equipos de programadores especializados en cada una de las plataformas móviles del mercado. Además muchas de estas plataformas tienen una vida corta y todo el capital invertido en dominarlas, quedará definitivamente perdido cuando ya no existan.

Anexo 1. Entregables del proyecto

El material entregado se ha organizado en tres carpetas, para una mejor estructuración de sus contenidos. Se exponen a continuación, acompañados de una breve descripción.

1. Carpeta documentación

Contiene los archivos:

- **folmo_memoria_tfg.pdf** - Documento de la memoria del proyecto
- **folmo_ct.pdf** - Documento de la autoevaluación de competencias transversales

2. Carpeta presentaciones

Contiene los archivos:

- **folmo_video_tribunal.mp4** - Documento en vídeo de la presentación para el tribunal
- **folmo_presentacion.pdf** - Documento de la presentación del proyecto

3. Carpeta proyecto

Contiene los archivos del proyecto y está estructurada a su vez en tres subcarpetas:

3.1. Carpeta **aplicacion**

Contiene toda la estructura de las tres aplicaciones web que componen el proyecto. El contenido de esta carpeta, deberá ser copiado tal cual se entrega, en el servidor remoto donde vaya a ser instalado el sistema, siguiendo las instrucciones indicadas en el capítulo 14 de esta memoria. Su contenido es el siguiente:

- **app_administrador** - Carpeta con los archivos de la aplicación de gestión de clientes
- **app_cms** - Carpeta con los archivos de la aplicación de gestión de contenidos
- **app_movil** - Carpeta con los archivos de la aplicación web móvil
- **biblioteca** - Carpeta donde se almacenarán las fotografías, logotipos, plantillas y códigos QR, creados por la aplicación, o subidos por los clientes
- **empresas** - Carpeta que contendrá a su vez, todas las carpetas de los clientes, generadas automáticamente por la aplicación de gestión de clientes, cuando son dados de alta
- **fuentes** - Carpeta con las fuentes tipográficas no estándares, usadas por las aplicaciones
- **imagenes** - Carpeta con las imágenes usadas por las aplicaciones
- **modelo** - Carpeta con los archivos del componente modelo de la aplicación
- **index.htm** - Archivo htm de inicio de la aplicación. Redirige al formulario de identificación de la aplicación de gestión de contenidos

3.2. Carpeta **app_movil**

- **catalogoapp.apk** - Archivo *APK*, con la aplicación móvil creada con PhoneGap Build. Instalable en dispositivos Android 4.x

3.3. Carpeta **base_de_datos**

- **esquema_bd_folmo.pdf** - Documento con el esquema gráfico de la base de datos
- **folmo_bd_uoc.sql** - Archivo *SQL* con la estructura y datos, para instalar en servidor MySQL

Anexo 2. Código fuente (extractos)

Como ya se comentó en el capítulo 5 de Metodología, nos hemos basado en el patrón *Modelo Vista Controlador (MVC)* para el desarrollo de las tres aplicaciones que componen el proyecto. A continuación, vamos a exponer una secuencia, extraída del código de programación de la app móvil que ilustre el funcionamiento general que se ha seguido para el desarrollo de todos los módulos, es decir, haremos un recorrido desde la solicitud que se hace a través de la vista, pasando por el controlador que la recoge y terminando en el modelo. A su vez, el modelo responderá al controlador y éste enviará finalmente la respuesta a la solicitud que envió la vista.

Paso 1. Se realiza una petición desde la vista

En este primer bloque, se muestra el código HTML de la vista, perteneciente al archivo *index.html*. Se ha marcado en rojo el elemento *<a>* que contiene la llamada al controlador mediante la función JavaScript *lista_novedades()*:

```
<!--SECCION DE INICIO-->
<div data-role="page" id="inicio" data-theme="none">
        <div class="cabecera" data-role="header">
          <h1>CAT&Aacute;LOGO APP</h1>
        </div>
        <div>
                <p class="logo_empresa"></p>
         <p class="nombre_empresa"> </p>
         <ul id="menu_principal">
          <li><a href="#empresa" data-transition="slide">empresa</a></li>
          <li><a href="javascript:lista_novedades();" data-transition = "slide"> novedades </a>
           </li>
          <li><a href="#ubicacion" data-transition="slide">ubicaci&oacute;n</a></li>
          <li><a href="#contacto" data-transition="slide">contacto</a></li>
         </ul>
        </div>
        <div class="pie" data-role="none">
                <p>&copy; francisco olmo ortiz 2013 - UOC</p>
        </div>
</div>
```

El segundo bloque, muestra el código de la función *lista_novedades()*, contenida en el archivo *index.js* que contiene el código JavaScript de la vista. Se muestran comentarios en rojo, con el flujo del proceso de envío de la petición AJAX y el procesado del resultado obtenido.

```
//código que solicita el listado de productos de la empresa
function lista_novedades()
{
	var cargador = creaAjax();//creamos un objeto AJAX

	if (cargador)
	{
		//formamos la cadena de variables que pasaremos via POST
		var query_string = "id_empresa=" + id_empresa;

		//llamamos a la funcion de respuesta del objeto AJAX
		cargador.onreadystatechange = cargaDatos;

		//realizamos la petición AJAX al controlador
		cargador.open("POST", ruta +
"app_movil/controlador/muestra_novedades_empresa.php", true);
		cargador.setRequestHeader("Content-Type","application/x-www-form-urlencoded");
		cargador.send(query_string);//enviamos las variables via POST
	}
	//función de respuesta del objeto AJAX
	function cargaDatos()
	{
		if(cargador.readyState == 4)
		{
			//si obtenemos una respuesta correcta del servidor
			if (cargador.status == 200)
			{
				var cadenaRespuesta = cargador.responseText;

				//insertamos el código HTML devuelto por el controlador
				document.getElementById("bloque_listado").innerHTML =
cadenaRespuesta;
				//navegamos hacia la pantalla de novedades
				location.href = "#novedades";
			}
			else if (cargador.status == 404)
			{
				alert("ERROR EN SU CONSULTA");
			}
		}
	}
}
```

Paso 2. El controlador envía la petición al modelo y recoge su respuesta

El controlador, en este caso el archivo *muestra_novedades_empresa.php*, recoge la variable *id_empresa* que le hemos pasado via POST y hace una llamada a la función estática *novedades_empresa()* de la clase abstracta *MuestraNovedadesEmpresa* del controlador, la cual

realizará la petición indicada en la función *listar_novedades()* del modelo. Una vez ha obtenido respuesta de éste, forma la cadena *$cadena_respuesta* y la envía devuelta a la vista que la solicitó, conteniendo ya el código HTML formateado.

```
<?php
abstract class MuestraNovedadesEmpresa
{
	public static function novedades_empresa($id_empresa)
	{
		//se agrega la clase del modelo
		require_once("../../modelo/app.php");
		//se crea un objeto del modelo
		$mi_app = new App();
		//se ejecuta la función que nos devuelve las novedades de esta empresa
		$resultado = $mi_app->listar_novedades($id_empresa);

		if($resultado)
		{
			$productos = array();
			//metemos la lista de productos obtenida en un array
			while($fila = mysql_fetch_array($resultado))
			{
				$productos[] = $fila;
			}
			//pedimos ahora la lista de fotos de los productos de la empresa
			$resultado_2 = $mi_app->listar_fotos_novedades($id_empresa);
			if($resultado_2)
			{
				$fotos = array();
				//metemos la lista de fotos obtenida en un array
				while($fila = mysql_fetch_array($resultado_2))
				{
					$fotos[] = $fila;
				}

				$cadena_respuesta = "";
				//vamos formando la cadena HTML que devolveremos a la vista
				for($i=0; $i<count($productos); $i++)
				{
					//comprobamos para cada producto, si tiene foto
					for($j=0; $j<count($fotos); $j++)
					{
						if($fotos[$j][0] == $productos[$i][0])
						{
							$nombre_foto_producto = $fotos[$j][1];
							break;
						}
						else
```

```
                                        {
                                                $nombre_foto_producto = "no";
                                        }
                                }
                                $cadena_respuesta .=
                                "<li class='item_novedad' onclick='info_producto(".
htmlentities("'". $productos[$i][0] ."'",ENT_QUOTES) .");'>
                                        <ul>
                                                <li class='item_novedad_foto'><img src=
'http://multimedia.uoc.edu/~olmo/biblioteca/fotos/". $nombre_foto_producto .".jpg' width='75'
height='100' alt='foto producto'></li>
                                                <li class='item_novedad_info'>
                                                        <p
class='item_novedad_info_nombre'>". $productos[$i][1] ."</p>
                                                        <p
class='item_novedad_info_precio'>". $productos[$i][2] ."&#8364; <span>+info</span></p>
                                                </li>
                                        </ul>
                                        <br class='limpia_flota' />
                                </li>";
                        }
                }
                //mostramos la cadena formada con el código HTML
                echo($cadena_respuesta);
        }
        else
        {
                echo("Error en la carga de datos");
        }
    }
}
//Llamamos a la función estática de la clase abstracta, con la variable pasada via POST
MuestraNovedadesEmpresa::novedades_empresa($_POST['id_empresa']);
?>
```

Paso 3. El modelo realiza la consulta solicitada a la base de datos

Las funciones *listar_novedades()* y *listar_fotos_novedades()* del modelo, construyen las sentencias SQL que serán enviadas a la base de datos, a través de la función *realizar_query_db()* de la clase *ConexionDB* del modelo. Mostramos ambas funciones o métodos.

```
//muestra la lista de productos de la empresa
    public function listar_novedades($id_empresa)
    {
        try
        {
```

```
                        $mi_conexion = new ConexionDB();
                        $mi_conexion->conectar_db();
                        //construye la sentencia SQL para enviar a la BD
                        $cadena = "SELECT idproducto,nombre,precio FROM producto WHERE
idempresa='".$id_empresa."' ORDER BY nombre";
                        $mi_resultado = $mi_conexion->realizar_query_db($cadena);

                        if(mysql_num_rows($mi_resultado) !=0)
                        {
                                return($mi_resultado); //devuelve el resource MySQL con los datos
                        }
                        else
                        {
                                return(false);
                        }
                }
                catch(Exception $e)
                {
                        echo($e->getMessage());
                }
        }

        //muestra las fotos de la lista de productos de la empresa
        public function listar_fotos_novedades()
        {
                try
                {
                        $mi_conexion = new ConexionDB();
                        $mi_conexion->conectar_db();
                        //construye la sentencia SQL para enviar a la BD
                        $cadena = "SELECT idproducto,nombre FROM foto GROUP BY idproducto";
                        $mi_resultado = $mi_conexion->realizar_query_db($cadena);

                        if(mysql_num_rows($mi_resultado) !=0)
                        {
                                return($mi_resultado);//devuelve el resource MySQL con los datos
                        }
                        else
                        {
                                return(false);
                        }
                }
                catch(Exception $e)
                {
                        echo($e->getMessage());
                }
        }
```

Anexo 3. Librerías/Código externo utilizado

- **PhoneGap**

  Propiedad de *Adobe Systems Inc* y cedido a la *Apache Software Foundation* (ASF) bajo el nombre de *Apache Cordova* Se trata del framework con el que se ha encapsulado la web app desarrollada en el proyecto, para obtener la aplicación nativa *catalogoapp.apk*, instalable en dispositivos Android 4.x. Si bien PhoneGap pone a disposición de los desarrolladores, una completa API con la que acceder a las utilidades hardware de los dispositivos móviles, en esta ocasión se ha usado exclusivamente con la finalidad de compilar la aplicación web. Por ello, se optó por utilizar la herramienta de compilación on-line PhoneGap Build. Su funcionamiento puede ser consultado en la página 41 del presente documento. Se accede a este servicio en la URL *https://build.phonegap.com/*

- **JQuery Mobile**

  De la *jQuery Foundation*, este framework está creado sobre las librerías *JQuery* y *JQuery UI*. Proporciona al desarrollador web una completa plataforma con la que crear aplicaciones, de diseño y comportamiento igual a las nativas, especialmente a las que encontramos en el mercado de iPhone. Está basado completamente en HTML5 y CSS3, es decir, estándares o futuros estándares web, motivo por el cual, se optó por adoptarlo para este proyecto, frente a otras opciones. Su implicación, ha consistido en su uso en el diseño de la vista del módulo web app, básicamente utilizando la adaptación que el framework hace de los elementos HTML, a través de atributos tipo *data-role*.

  En concreto, se han usado los *data-role='page'* como atributos de elementos *div*, convirtiéndolos en páginas distintas, cuyo nombre sería su atributo *id*. De esta manera, conseguimos que toda la aplicación web, esté contenida en un solo documento HTML. Otros usos de estos atributos han sido, *data-transition='slide'* y *data-direction'='reverse'* para establecer las transiciones entre páginas, así como *data-role='button' y data-icon='home'* para convertir algunos enlaces en botones con iconos.
  Puede descargarse en *http://jquerymobile.com/*

- **phpqrcode**

  Se trata de una librería de código abierto *(LGPL)* para la generación de códigos QR, basada en la librería *libqrencode* escrita en lenguaje C por Kentaro Fukuchi. Proporciona una API para crear imágenes de código QR en formato PNG, gracias a la librería GD2 y está implementada totalmente en PHP, sin dependencias externas. Puede descargarse en *http://phpqrcode.sourceforge.net/*

  En la función *alta_empresa()* de la clase *Sistema (sistema.php)*, se muestra la forma en que se ha usado en este proyecto.

Anexo 4. Bibliografía

1. Material docente de la UOC. Grado de Multimedia

- Gil de la Iglesia, D. y Berni Millet, P. (2010). Laboratorio de PHP y MySQL. Barcelona: UOC
- Gil de la Iglesia, D. y Berni Millet, P. (2010). Diseño de bases de datos. Barcelona: UOC
- Moncho Mas, V. (2010). Orientación a objetos en JavaScript. Barcelona: UOC
- Sánchez Cano, J. (2011). AJAX. Barcelona: UOC
- Monjo Palau, T. (2011). Diseño de interfaces multimedia. Barcelona: UOC
- Fuenmayor López, D. y González Sancho, M. (2012). Aplicaciones Rich Media. Barcelona: UOC
- Arnedo Moreno, J. y Riera i Terrén, D. (2010). Diseño y programación orientada a objetos. Barcelona: UOC

2. Libros, guías y manuales

- Charte Ojeda, F. (2007). Ajax. Madrid: Ediciones ANAYA Multimedia
- Stark, J. (2010). Building iPhone Apps with HTML, CSS and JavaScript: O'Reilly
- Myer, T. (2012). Phonegap. Madrid: Ediciones ANAYA Multimedia
- David, M. (2011). HTML5. Madrid: Ediciones ANAYA Multimedia

3. Documentación electrónica. Internet

- PhoneGap Build: https://build.phonegap.com/docs
- JQuery Mobile: http://jquerymobile.com/demos/1.2.0/
- PHP: http://php.net/
- Android: http://developer.android.com/index.html
- W3Schools: http://www.w3schools.com/

Anexo 5. Vita

Francisco Olmo Ortiz

Nace en Castro del Río, Córdoba, un 17 de abril de 1966. A los 18 años, cuando termina la Educación Secundaria, recibe junto a su hermano Antonio, un Home Computer Philips MSX VG-8020, de manos de su padre Antonio. Empieza aquí su aventura informática que pasará poco a poco de ser su principal afición, a convertirse en su vida profesional. Comienza en estos primeros tiempos, haciendo incursión en la programación Basic, mientras aprende y disfruta de máquinas legendarias como los Commodore 64 y 128, donde aprende conceptos como el de *sprite* que sigue usándose a día de hoy.

Es a principios de los 90, en la época dorada del Amiga de Commodore, cuando descubre la informática multimedia, aprendiendo programación de juegos, digitalización de vídeo y sonido y modelado y animación 3D. A partir de aquí, comienza a asumir retos profesionales, para terminar formando en 1998 junto a su hermano Antonio, la empresa AFOSoft Multimedia, al principio dedicada a trabajos esporádicos. A partir de 2001, despega su actividad en el desarrollo de aplicaciones multimedia, creando interactivos en CD y software para la formación.

Con la llegada de Internet, toda su actividad va pasando a este nuevo medio, para convertirse en desarrolladores de aplicaciones web, campus virtuales para formación vial, diseño web y comercio electrónico. En la actualidad se halla inmerso en la adaptación de sus conocimientos a las plataformas móviles que comienzan ya a dominar el mundo de la informática.

Está vinculado a la UOC desde el año 2000, cursando al principio asignaturas del Graduado Multimedia que le resultaban de interés para su trabajo. En 2010, con la creación de la titulación oficial del Grado en Multimedia, decide completar sus estudios y hacerse con esta interesante titulación.