Escuela Politécnica Superior de Jaén

Grado en Ingeniería Informática

UNIVERSIDAD DE JAÉN
Escuela Politécnica Superior de Jaén
Departamento de informática

Trabajo Fin de Grado

DISEÑO DE UN SISTEMA DE GESTIÓN PARA UNA PEÑA DEPORTIVA

Alumno: Luis Javier Díaz Medina

Tutor: Prof. D. José María Serrano Chica

Dpto: Departamento de Informática

Junio, 2017

Universidad de Jaén
Escuela Politécnica Superior de Jaén
Departamento de Informática

Don José María Serrano Chica , tutor del Proyecto Fin de Carrera titulado: DISEÑO DE UN SISTEMA DE GESTIÓN PARA UNA PEÑA DEPORTIVA, que presenta Luis Javier Díaz Medina, autoriza su presentación para defensa y evaluación en la Escuela Politécnica Superior de Jaén.

Jaén, SEPTIEMBRE de 2017

El alumno: Los tutores:

SERRANO CHICA JOSE MARIA - 26490278M

Firmado digitalmente por SERRANO CHICA JOSE MARIA - 26490278M
Nombre de reconocimiento (DN): c=ES, serialNumber=IDCES-26490278M, givenName=JOSE MARIA, sn=SERRANO CHICA, cn=SERRANO CHICA JOSE MARIA - 26490278M
Fecha: 2017.09.06 11:33:12 +02'00'

LUIS JAVIER DÍAZ MEDINA JOSÉ MARÍA SERRANO CHICA

Índice

Índice de Ilustraciones

Índice de Tablas

Agradecimientos

Un largo camino hasta llegar aquí, una etapa universitaria que llega a su fin, quizás más tarde de lo que debería haber llegado, pero al fin y al cabo, lo importante es llegar.

Por este motivo, quiero agradecer a todas esas personas que han estado ahí apoyándome día a día y a las que también hago partícipes en mis éxitos presentes.

En primer lugar, agradecer a mi familia todo el apoyo durante toda la carrera, a mis padres, que siempre confiaron en mi desde el momento cero, a mis tíos y primos que siempre me han recordado la importancia de sacar todo esto adelante, y a mi abuela, que a pesar de su avanzada edad, siempre se ha preocupado por mis estudios.

En segundo lugar, a mis amigos, tanto los de la universidad como a mis amigos de toda la vida. Me llevo grandes recuerdos de mi vida universitaria, y también grandes amigos, con ellos he vivido una de las mejores épocas de mi vida, y por suerte, hoy en día vivo en Madrid con dos de ellos, amistades que durarán toda la vida.

En último lugar, a todos los profesores de la Universidad de Jaén, unos grandes profesionales, y que gracias a ellos, hoy puedo decir que estoy trabajando de lo que me gusta, y con unas expectativas profesionales muy altas. En especial agradecer a José María Serrano, por su dedicación y compresión, y por su entera disposición. Muchos compañeros de clase en Madrid, y es un orgullo ver que todos aspiramos a cosas muy grandes, que siempre habrá jiennenses “dando guerra” por Madrid.

A todos ellos, y a muchas más personas que me gustaría nombrar, sólo les puedo dar palabras de agradecimiento.
Gracias.

1. INTRODUCCIÓN

El mundo está cambiando, al igual que la sociedad cambia sus hábitos, la forma de comunicarse, la forma de organizarse, la forma en la que las personas perciben la información que tenemos alrededor [1]. Vivimos en una época de continuo cambio, sobre todo tecnológico. Hace tan sólo un par de décadas era imposible pensar que en pleno siglo XXI la tecnología avanzara tanto.

Como consecuencia, actualmente vivimos ligados a la tecnología, y es que cualquier persona puede acceder a “casi” toda la información que desee con un par de clicks. Es un hecho que los nacidos en el siglo XXI se les llama “nativos tecnológicos”, y es que, cualquier niño, cada vez con menor edad, es capaz de manipular cualquier tipo de dispositivo. Lo que está claro es una cosa, dependemos de la tecnología en nuestro día a día.

Las nuevas tecnologías, relacionadas con nuestro entorno, están agilizando, optimizando y perfeccionando ya muchas actividades que realizamos en nuestro día a día. La comunicación en la actualidad ha avanzado mucho, y es mucho más rápida que antes, como por ejemplo internet, en el caso de transmitir mensajes, imágenes, videos y todo tipo de documentos desde diferentes partes del mundo y cuya información está disponible las 24 horas del día. Poco queda ya de los envíos de documentos por medio de los servicios convencionales.

La tecnología ha jugado un papel muy importante en el mundo desde que se crea un “algo” innovador que todos queremos tener cuanto antes. Podemos hablar de marketing tecnológico, la sociedad quiere estar a la moda y presumir de tener lo último del mercado. Un ejemplo de esta competitividad por tener lo mejor del mercado son los teléfonos móviles. Todos desean tener el mejor teléfono móvil, y a su vez tener las aplicaciones más novedosas del mercado.

Otro punto a tener muy en cuenta son las redes sociales. Es un medio muy utilizado por todos teniendo un gran impacto en la sociedad, porque han logrado transformar la forma de comunicarnos tanto a nivel personal, como profesional. Las redes sociales se han convertido en un medio de comunicación e información de nuestra sociedad, avisando en tiempo real de lo que sucede en el mundo y los

sucesos que ocurren a nuestro alrededor. Hoy en día cualquier persona tiene facebook, twitter, instagram o cualquier otra red social. Una de las ventajas de esta comunicación virtual, es que te puedes comunicar rápido e instantáneamente con cualquier persona que se encuentre en cualquier punto del planeta, hacer negocios, compartir contenido multimedia, incluso buscar un nuevo empleo.

También es importante, llegados a este punto, hablar de cómo cualquier tipo de organización aprovecha esta tecnología para la gestión de su información. Hoy en día, desde una pequeña organización hasta una gran multinacional implementa sus propios sistemas para gestionar su propio contenido, mejorar su modelo de negocio o darse a conocer al mundo exterior. Cabe destacar la gran cantidad de webs de empresa que se están lanzando en la última década.

La web, a diferencia de un folleto o catálogo, es un soporte vivo, que hay que alimentar con nuevos contenidos y funcionalidades. Esto quiere decir, que si no se hace un mantenimiento adecuado, se irá quedando obsoleta y perderá interés para los usuarios. El problema de muchas organizaciones (organizaciones pequeñas y pymes) es que una vez que han invertido en la creación de una web, no son capaces de poder mantener “vivo” su proyecto en Internet. Una excelente opción para evitar esta situación es utilizar programas de gestión de contenidos, más conocidos como CMS (Content Management System). Estas herramientas facilitan la gestión del contenido de una web. En los últimos años el número de webs que se gestionan por este tipo de sistemas ha crecido de una forma considerable.

Por último, destacar la tendencia actual de desarrollo de aplicaciones para múltiples dispositivos. Se puede comprobar facilmente que cualquier aplicación web ya está también disponible en una multitud de dispositivos que también pueden conectarse a Internet. A este fenómeno se le conoce como *IoT* [2] (Internet of things, Internet de las cosas en español). Es un concepto que se refiere a la interconexión digital de objetos cotidianos con internet.

La ilustración 1. A muestra una descripción gráfica del mundo interconectado:

Ilustración 1: Internet de las cosas

Como graficamente se indica, la tendencia actual es desarrollar aplicaciones que se puedan ejecutar en una amplia gama de dispositivos. Como ejemplo, podemos nombrar la aplicación de Netflix, cuya finalidad es la reprodución de contenido multimedia, generalmente series y películas; está disponible para la web, para dispositivos móviles, tablets, smartTV, incluso también están presentes en videoconsolas como Playstation.

1.1. Motivación

En el punto anterior se ha descrito como la tecnología está inmersa en la sociedad, la tendencia a la web para la distribución de la información, el crecimiento de dispositivos conectados entre sí, y en consecuencia el despegue de las aplicaciones desarrolladas para estos dispositivos.

Resulta muy interesante la idea de poder crear un sistema de gestión para una organización basándonos en la idea de que la tecnología puede favorecer de una manera muy positiva la forma en la que se interactúa con los individuos de dicha organización, y mantenerlos informados día a día, estando presente en ellos a través de los dispositivos que actualmente y día a día usa cualquier usuario.

A nivel personal, creo que es un trabajo ambicioso, el cuál requiere un cuidado extremo, ya que habría que idear un sistema que mantenga los datos actualizados para cada uno de las aplicaciones (para diferentes los dispositivos que se van a usar). Por tanto, mi motivación para la realización de este trabajo también se basa en saber como funciona hoy en día la mayoría de aplicaciones móviles, usando la potencia de los servicios web, y profundizar en un mayor grado el funcionamiento interno de los sistemas de gestión de contenidos.

1.2. Objetivos

En este trabajo se aborda el caso de creación de un sistema de gestión de una pequeña peña deportiva de Jaén, llamada “Viejos Amigos FS”.

Hasta ahora, la gestión de los socios, de aquí en adelante usuarios, se hacían a mano, y la información que se emitía de los servicios, actividades, etc sólo llegaban a través de correo electrónico, cuando el encargado, de aquí en adelante administrador, consideraba que era importante y necesario.

La problemática de este sistema tradicional, principalmente, es que los usuarios no están al tanto de los eventos que transciernen para ellos, no teniendo disponible la información en el día a día, imposibilitando el crecimiento de la peña, que cada vez más se encuentra en decadencia, con una gestión dificil y tediosa.

Por tanto, los objetivos generales de este trabajo se basan en las siguientes puntos:

- Aprovechar las ventajas de utilizar Internet como forma de gestión de la peña, automatizando la mayoría de tareas que actualmente se realizan de forma manual.
- Aprovechar el auge de la utilización de los dispositivos móviles como forma de mantener al usuario informado en el día a día.
- Aprovechar el tirón de las redes sociales para llegar a más gente, darse a conocer en el mundo exterior y actualizar de una forma rápida las noticias relacionadas con la organización.

- Realizar una aplicación web para la gestión de la peña, con la finalidad de automatizar la gestión de ésta, dando solución a la problemática del primer punto.
- Realizar una aplicación móvil para mantener la peña cercana a los usuarios con los dispositivos más usados hoy en día.
- Integrar ambas aplicaciones con redes sociales.

Con todos estos puntos, mi motivación es bastante clara, crear un sistema actual y novedoso, ya que en cualquier tipo de organización, no es difícil ver que cualquier organización, pequeña empresa ya plantea este modelo tecnológico para mejorar su modelo corporativo.

1.3. Estructura del proyecto

En este punto se explicará de manera resumida el desglose de tareas para la realización del proyecto.

Se ha escogido un modelo de proceso bastante utilizado, el prototipado.

El Modelo de prototipos [3], en Ingeniería de software, pertenece a los modelos de desarrollo evolutivo. El prototipo debe ser construido en poco tiempo, usando los programas adecuados y no se debe utilizar muchos recursos.

El diseño rápido se centra en una representación de aquellos aspectos del software que serán visibles para el cliente o el usuario final. Este diseño conduce a la construcción de un prototipo, el cual es evaluado por el cliente para una retroalimentación; gracias a ésta se refinan los requisitos del software que se desarrollará. La interacción ocurre cuando el prototipo se ajusta para satisfacer las necesidades del cliente. Esto permite que al mismo tiempo el desarrollador entienda mejor lo que se debe hacer y el cliente vea resultados a corto plazo.

Las características principales de este modelo de proceso son las siguientes:

- Una de sus principales utilidades es el estudio de la interfaz persona – ordenador, poniéndose en la piel del usuario para predecir el comportamiento de éste con el sistema.

- Por lo general, un prototipo se diferencia del sistema final en que es una solución inacabado y tiene una construcción menos elástica.
- Ocasionalmente los requisitos no están claros. El prototipado evita muchas de las equivocaciones y ambigüedades que pueden existir en estos.
- En otros modelos, el usuario sólo comprueba el funcionamiento del producto en la entrega final de este. Esto puede dar lugar a que al final, las necesidades que tenía el cliente no coincidan con el producto realizado, pudiendo obligar al equipo de desarrollo a afrontar cambios drásticos.
- El código es facilmente reutilizable.

Las etapas del prototipado se pueden resumir en:

- Plan rápido.
- Modelado, diseño rápido.
- Construcción del prototipo.
- Desarrollo, entrega y retroalimentación.
- Comunicación.
- Entrega del desarrollo final.

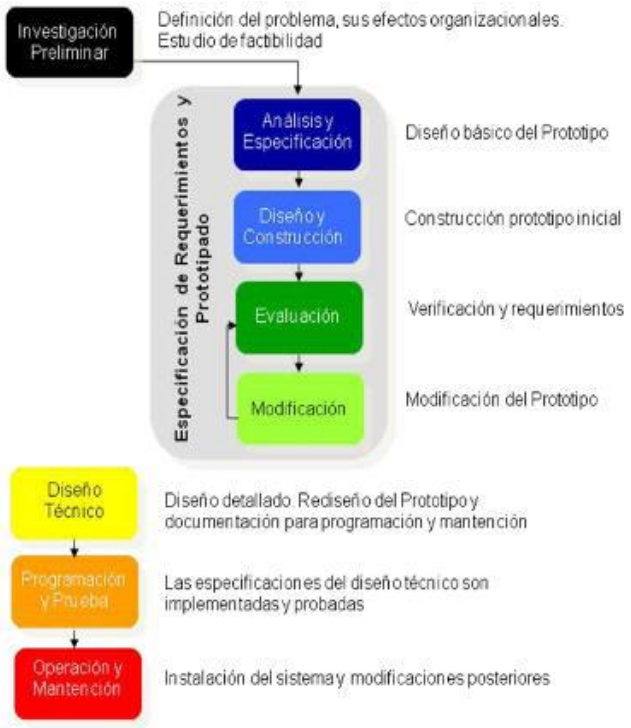


Ilustración 2: Etapas del prototipado

Una vez especificado el modelo a seguir, se procederá a describir la estructura el proyecto capítulo a capítulo.

Durante los siguientes cinco capítulos se detallará, de forma general, la información teórica sobre todas las tecnologías utilizadas para la realización del trabajo, incluyendo la información necesaria para que el lector pueda entender la consecuencia y la manera en la que se ha desarrollado el trabajo.

El capítulo dos está dedicado a los aspectos relacionados con los Servicios Web, definiciones, ventajas e inconvenientes, por qué utilizarlo, etc. También se describirá en este punto la herramienta elegida para el desarrollo del trabajo: XAMPP.

Una vez finalizado el punto de Servicios Web, el lector estará más preparado para leer el siguiente capítulo, conocer el funcionamiento de la arquitectura RestFul, que será utilizada para sincronizar la aplicación Android con los datos alojados en el servidor web.

En el capítulo cuatro, se describirán los aspectos más importantes de Android, la opción más usada hoy en día en el desarrollo de software para dispositivos móviles. Aquí se incluirá un estudio de los puntos más relevantes de esta arquitectura y por qué está triunfando en el mercado actual.

En el capítulo cinco se explicará en qué consiste Joomla!, la solución adoptada para resolver la casuística expuesta en puntos anteriores. Se describirán los puntos fuertes de este sistema de gestión de contenidos, las ventajas e inconvenientes de su utilización, cómo funciona en general. Al tratarse más de una herramienta que de una tecnología, con este capítulo concluiremos el bloque teórico.

Una vez definidos los puntos teóricos más importantes sobre los cuales se desarrollará el trabajo pasamos a describir los capítulos más prácticos de enfoque sobre el proyecto en sí.

En el capítulo siete se desarrollará la parte de análisis, una de las bases sobre las cuales se establece la futura estructura, objetivos del proyecto, y en la cuál se

realizan los diagramas de análisis propios de la ingeniería del software como son los diagramas de tiempo, PERTs o diagrama de costes, entre otros.

Posteriormente, en el capítulo ocho, se abordarán temas del diseño de la aplicación, tales como el diseño de las interfaces que se utilizarán en las aplicaciones, así como el diseño del modelo de datos. Se pretende que el lector, al leer este punto, pueda comprender de una forma más o menos clara cuál es el funcionamiento de la aplicación.

En el último capítulo numerado del trabajo, el capítulo nueve, estará dedicado a los temas de implementación del prototipo, teniendo en cuenta las arquitecturas utilizadas en el desarrollo, así como los aspectos más importantes en el desarrollo del código.

En la parte final se incluirá un capítulo con las conclusiones finales sobre el proyecto, las posibles mejores que se podrían aplicar a corto y largo plazo, y las principales dificultades y problemas surgidas a lo largo de todo el desarrollo de este.

Por último se incluirán los anexos con información necesaria adicional sobre el trabajo. Estos anexo son los siguientes:

- Anexo I: Encuesta.
- Anexo II: Manual de instalación.
- Anexo III: Manual de usuario aplicación web.
- Anexo IV: Manual de usuario aplicación móvil.
- Anexo V: Pruebas servicios REST.

2. SERVICIOS WEB

2.1. Introducción

Existen numerosas definiciones de Servicios Web y esto demuestra, en parte, la gran complejidad de los servicios que se agrupan bajo este término y las implicaciones asociadas a ellos. Hasta ahora la definición más general y convincente es decir que los Servicios Web son el conjunto de aplicaciones o tecnologías con capacidad para interoperar en la Web. Estas tecnologías intercambian datos entre ellas con el fin de ofrecer unos servicios [4].

Un servicio Web es cualquier servicio que:

- Está disponible en Internet o en redes privadas (Intranet).
- Usa un sistema de mensajería XML estandarizado.
- No está supeditado a algún sistema operativo o lenguaje deprogramación.
- Se describe a sí mismo a través de una gramática XML común.
- Tiene la capacidad de ser descubierto a través de un mecanismo simplede búsqueda

Un servicio web es capaz de integrar una comunicación entre un cliente sin que estos dos estén en el mismo ordenador, utilizando protocolos estándar (http, SOAP, WSDL, UDDI).

En la siguiente ilustración se muestran los principales protocolos estándar utilizados en servicios web.

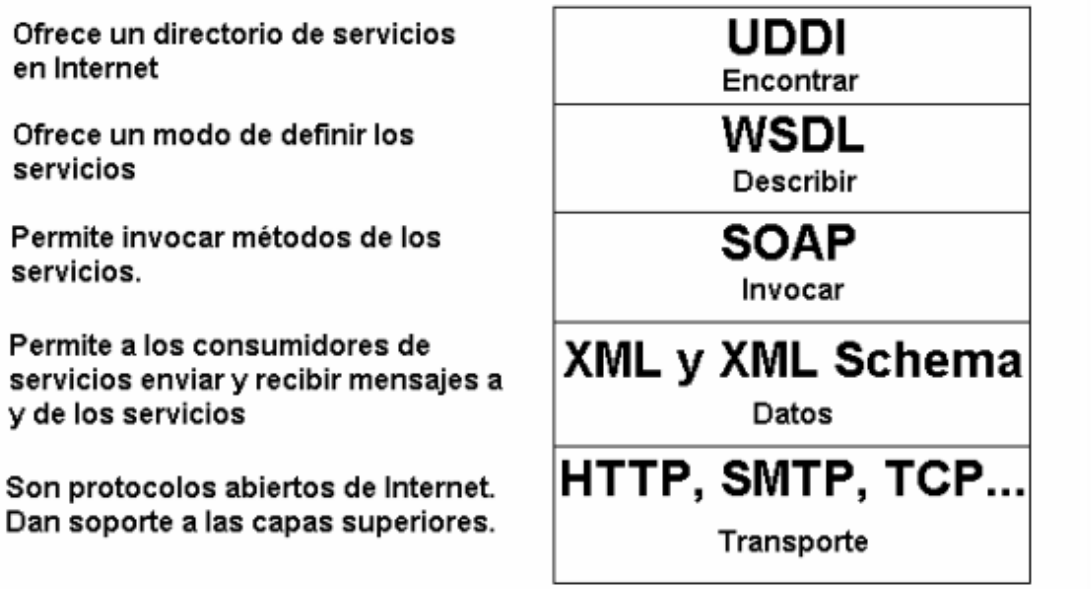


Ilustración 3: Protocolos de los Servicos Web

Este tipo de tecnología tiene una serie de impedimentos. Uno de los más importantes es que no ofrece una interfaz de usuario para el usuario que demanda los servicios, como podría proporcionarlo un servidor de páginas web. Por tanto son programas conectados entre sí pero que no interactúan directamente con el usuario, lo que significa que es tarea del programador el realizar una interfaz de usuario para la conexión entre el usuario y estos programas.

2.2. Estándares

A continuación vamos a centrarnos en el funcionamiento de los diferentes protocolos de estándar mencionados anteriormente:

UDDI: La especificación UDDI (Universal Description, Discovery, and Integration) define un modo de publicar y encontrar información sobre servicios Web.

UDDI tiene dos funciones:

- Es un protocolo basado en SOAP que define cómo se comunican los clientes con los registros UDDI.
- Es un conjunto de registros duplicados globales en particular.

UDDI incluye un esquema XML para mensajes SOAP que define un conjunto de documentos para describir información de empresas y servicios, un conjunto común de API para consultar y publicar información en los directorios y una API para duplicar entradas de directorio entre nodos UDDI iguales [5].

- WSDL: WSDL es el lenguaje propuesto por el W3C para la descripción de Servicios Web y permite describir la interfaz de un servicio web en un formato XML. Una de sus ventajas es que permite separar la descripción abstracta de la funcionalidad ofrecida por un servicio, es decir, de los detalles concretos del mismo, como puede ser el enlace a un protocolo de red o un formato de mensaje concreto que puede ser SOAP, HTTP o MIME.

- SOAP: Abreviación de Simple Object Access Protocol . SOAP es un formato de mensaje XML utilizado en interacciones de servicios web. que se usa para codificar información de los requerimientos de los Web Services y para responder los mensajes antes de enviarlos por la red. Los mensajes SOAP habitualmente se envían sobre HTTP o JMS, pero se pueden utilizar otros protocolos. El uso de SOAP en un servicio web específico se describe mediante la definición WSDL.

-XML: Abreviación de eXtensible Markup Language que fue desarrollado por el Word Wide Web Consortium (W3C), una sociedad mercantil internacional que elabora recomendaciones para la World Wide Web. El XML es una adaptación del SGML(Standard Generalized Markup Language), un lenguaje que permite la organización y el etiquetado de documentos. Esto quiere decir que el XML no es un lenguaje en sí mismo, sino un sistema que permite definir lenguajes de acuerdo a las necesidades. El XHTML, el MathML y el SVG son algunos de los lenguajes que el XML tiene la capacidad de definir.

-HTTP: El Protocolo de Transferencia de HiperTexto (Hypertext Transfer Protocol) es un sencillo protocolo cliente-servidor que articula los intercambios de información entre los clientes Web y los servidores HTTP. HTTP se basa en sencillas operaciones de solicitud/respuesta. Un cliente establece una conexión con un servidor y envía un mensaje con los datos de la solicitud. El servidor responde con un mensaje similar, que contiene el estado de la operación y su posible resultado. Todas las operaciones pueden adjuntar un objeto o recurso sobre el que actúan; cada objeto Web (documento HTML, fichero multimedia o aplicación CGI) es conocido por su URL.

2.3. Ventajas de los Web Services

A continuación se describen las principales ventajas del uso de los web services [6]:

- Aportan interoperabilidad entre aplicaciones de software independientemente de sus propiedades o de las plataformas sobre las que se instalen.
- Los servicios Web fomentan los estándares y protocolos basados en texto, que hacen más fácil acceder a su contenido y entender su funcionamiento.
- Al apoyarse en HTTP, los servicios Web pueden aprovecharse de los sistemas de seguridad firewall sin necesidad de cambiar las reglas de filtrado.

2.4. Desventajas de los Web Services

A continuación se describen las principales desventajas de usar web services:

- Para realizar transacciones no pueden compararse con los estándares abiertos de computación distribuida como CORBA (Common Object Request Broker Architecture).
- Su rendimiento es bajo si se compara con otros modelos de computación distribuida, como RMI(Remote Method Invocation), CORBA, o DCOM (Distributed Component Object Model).
- Al apoyarse en HTTP, pueden esquivar medidas de seguridad basadas en firewall cuyas reglas tratan de bloquear la comunicación entre programas.
- Existe poca información de servicios web para algunos lenguajes de programación

2.5. Motivos para la utilización de Servicios Web

La razón más importante para utilizar servicios web es que utiliza el protocolo HTTP sobre TCP utilizando para ello el puerto 80. Es importante señalar que los servicios web se pueden utilizar sobre cualquier protocolo, sin embargo, TCP es el más común.

Otra razón es que, antes de que existiera SOAP, no había buenas interfaces para realizar el acceso a las funcionalidades de otros ordenadores conectados en red.

La última razón es la gran independencia que los servicios web ofrecen en la relación entre la aplicación que utiliza el servicio web y el propio servicio. Esta independencia hace que los cambios en un sistema no afectan al otro sistema. Esta flexibilidad será cada vez más importante, dado que existe una tendencia a construir grandes aplicaciones a partir de componentes distribuidos más pequeños.

A continuación vamos a hablar sobre la infraestructura elegida para realizar un servicio web. Es una infraestructura que proporciona todas las herramientas y funcionalidades necesarias para realizar una buena comunicación entre la aplicación y los demás sistemas implicados.

En el siguiente capítulo nos centraremos en XAMPP.

2.6. XAMPP

XAMPP, es un servidor de plataforma libre, es un software que integra en una sola aplicación, un servidor web Apache, interpretes de lenguaje de scripts PHP, un servidor de base de datos MySQL, un servidor de FTP FileZilla, el popular administrador de base de datos escrito en PHP, MySQL, entre otros módulos [7].

Permite instalar de forma sencilla Apache en tu propio ordenador, sin importar tu sistema operativo (Linux, Windows, MAC o Solaris).

XAMPP es una herramienta de desarrollo que permite probar el trabajo (páginas web o programación por ejemplo) en tu propio ordenador sin necesidad de tener que acceder a internet.

XAMPP provee de una configuración totalmente funcional desde el momento que lo instalas. Sin embargo, es bueno acotar que la seguridad de datos no es su punto fuerte, por lo cual no es suficientemente seguro para ambientes grandes o de producción.

A continuación se describen las principales características de XAMPP:

- Para Windows existen dos versiones, una con instalador y otra portable (comprimida) para descomprimir y ejecutar.
- Otra característica no menos importante, es que la licencia de esta aplicación es GNU ((General PublicLicense), está orientada principalmente a proteger la libre distribución, modificación y uso de software. Su propósito es declarar que el software cubierto por esta

licencia es software libre y protegerlo de intentos de apropiación que restrinjan esas libertades a los usuarios.)

- La filosofía de XAMPP, como lo indican en su sitio web, es crear una distribución fácil de instalar, de tal manera que los desarrolladores web principiantes cuenten con todo lo necesario ya configurado.
- XAMPP solamente requiere descargar y ejecutar un archivo .zip, .tar, o .exe, con unas pequeñas configuraciones en alguno de sus componentes que el servidor Web necesitará.
- Una de las características sobresalientes de este sistema es que es multiplataforma, es decir, existen versiones para diferentes sistemas operativos, tales como: Microsoft Windows, GNU/Linux, Solaris, y MacOS X. Existen versiones para Linux (testeado para SuSE, RedHat, Mandrake y Debian), Windows (Windows 98, NT, 2000, XP y Vista), MacOS X y Solaris (desarrollada y probada con Solaris 8, probada con Solaris

Xampp es una herramienta muy práctica que nos permite instalar el entorno MySQL, Apache y PHP, suficiente para empezar proyectos web o revisar alguna aplicación localmente. Además trae otros servicios como servidor de correos y servidor FTP.

Una de las ventajas de usar XAMPP es que su instalación es de lo mas sencilla, basta con descargarlo, extraerlo y comenzar a usarlo. En general es bastante fácil la instalacion de apache y php sobre Unix, sobre todo si dispone de un manejador de paquetes.

La mayor ventaja de XAMPP es que es muy fácil de instalar y las configuraciones son mínimas o inexistentes, lo cual nos ahorra bastante tiempo. Sin embargo hay ocasiones en que es mejor dejar atrás la comodidad por las siguientes razones:

- No soporta MySQL desde la consola. Xampp trae PhpMyAdmin para administrar las bases de datos de MySQL, sin embargo para tareas más específicas es mejor utilizar la consola (linea de comandos) y Xampp no la soporta.

- No se pueden actualizar individualmente las versiones de los programas que instala.
- Xampp trae las últimas versiones de las aplicaciones que instala, sin embargo cuando pasa el tiempo y salen nuevas versiones de las mismas, no queda otra salida que reinstalar todo Xampp.
- Dificultad para configurar aplicaciones de terceros.

En la siguiente ilustración se muestra un ejemplo de la interfaz gráfica de configuración de XAMPP:

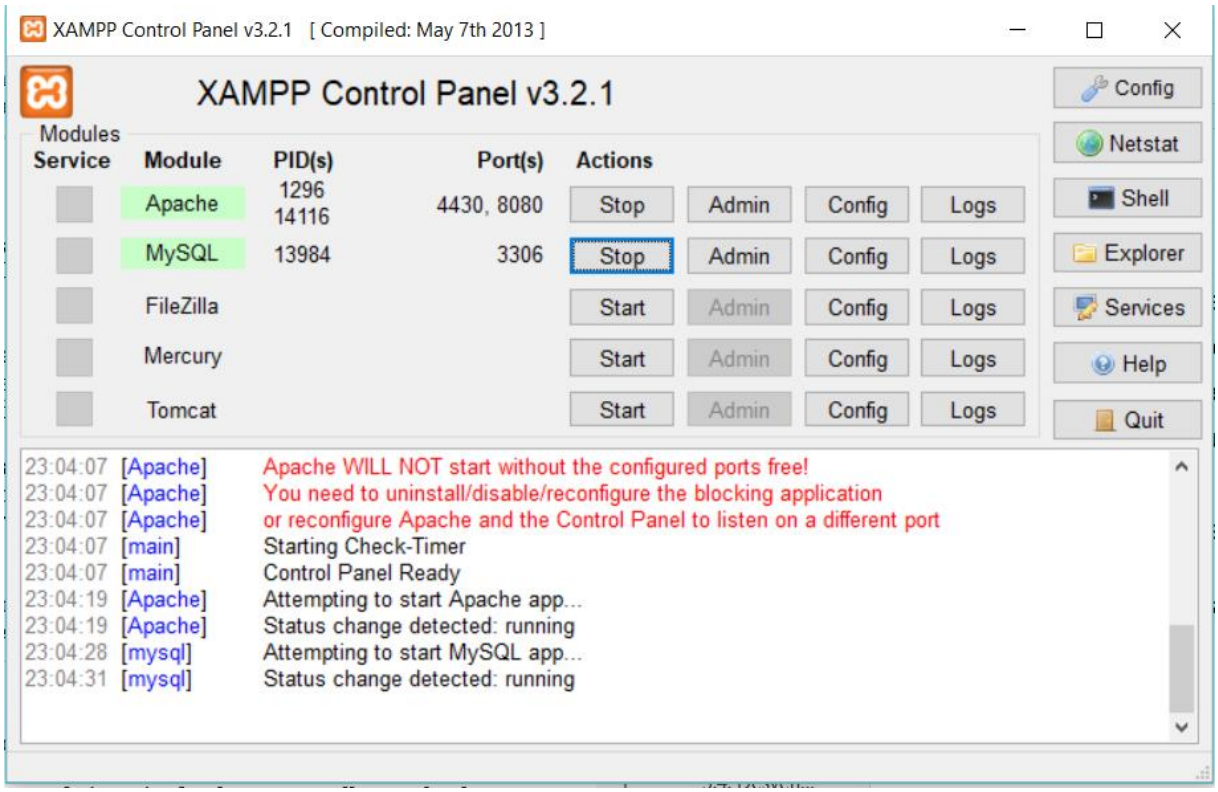


Ilustración 4: Panel de control de XAMPP

2.7. Herramientas de XAMPP

En este punto se describen las principales herramientas en las que se basa XAMPP para ofrecer sus servicios.

2.7.1. Servidor Web Apache

Ilustración 5: Logo de Apache

El servidor HTTP Apache es un servidor web HTTP de código abierto, que implementa el protocolo HTTP/1.1 y la noción de sitio virtual. El servidor Apache se desarrolla dentro del proyecto HTTP Server (httpd) de la Apache Software Foundation. Apache presenta muchas características altamente configurables, bases de datos de autenticación y negociado de contenido [8].

Apache cuenta con una amplia aceptación en la red desde 1996 (es el servidor HTTP más usado). Este alcanzó su máxima cuota de mercado en 2005 siendo el servidor empleado en el 70% de los sitios web en el mundo. La mayoría de las vulnerabilidades de la seguridad descubiertas y resueltas tan sólo pueden ser aprovechadas por usuarios locales y no de manera remota. Sin embargo, algunas se pueden accionar remotamente en ciertas situaciones, o explotar por los usuarios locales malévolos en las disposiciones de recibimiento compartidas que utilizan PHP como módulo de Apache.

Entre las principales ventajas de Apache se pueden enumerar las siguientes:

- Modular.
- Código abierto.
- Multi-plataforma.
- Extensible.
- Popular (es fácil conseguir ayuda/soporte).

En contrapunto, se pueden enumerar las siguientes desventajas:

- Formatos de configuración no estándar.
- No cuenta con una buena administración.
- Falta de integración.

Este es software libre pero es incompatible con la GPL, que es un tipo de licencia de software libre que permite a los usuarios utilizar dicho software, compartirlo e incluso modificarlo.

Apache es usado principalmente para enviar páginas web estáticas y dinámicas en la World Wide Web. Muchas aplicaciones web están diseñadas asumiendo como ambiente de implantación a Apache, o que utilizarán características propias de este servidor web.

Este servidor web se redistribuye como parte de varios paquetes propietarios de software, incluyendo la base de datos Oracle y el IBM WebSphere application server. Mac OS X también integra apache como parte de su propio servidor web y como soporte de su servidor de aplicaciones WebObjects.

Apache es usado para muchas otras tareas donde el contenido necesita ser puesto a disposición en una forma segura y confiable. Un ejemplo es al momento de compartir archivos desde un ordenador personal hacia Internet. Un usuario que tiene Apache instalado en su escritorio puede colocar arbitrariamente archivos en la raíz de documentos de Apache, desde donde pueden ser compartidos.

Los programadores de aplicaciones web a veces utilizan una versión local de Apache con el fin de previsualizar y probar código mientras éste es desarrollado.

2.7.2. Servidor de Base de Datos MySQL

Ilustración 6: Logo de MySQL

MySQL, es un sistema de gestión de base de datos relacional o SGBD. Este gestor de base de datos es multihilo y multiusuario, lo que le permite ser utilizado por varias personas al mismo tiempo, e incluso, realizar varias consultas a la vez, lo que lo hace sumamente versátil [9].

La mayor parte del código se encuentra escrito en lenguaje C/C++ y la sintaxis de su uso es bastante simple, lo que permite crear bases de datos simples o complejas con mucha facilidad.

Además, es compatible con múltiples plataformas informáticas y ofrece una infinidad de aplicaciones que permiten acceder rápidamente a las sentencias del gestor de base de datos.

Como se comenta anteriormente este gestor de base de datos es muy utilizado en desarrollo web, ya que permite a los desarrolladores y diseñadores, realizar cambios en sus sitios de manera simple, con tan sólo cambiar un archivo, evitando tener que modificar todo el código web. Esto se debe a que MySQL, trabaja con un sistema centralizado de gestión de datos, que permite realizar cambios en un solo archivo y que se ejecuta en toda la estructura de datos que se comparte en la red. Además, permite incluir noticias e información rápidamente en un sitio web, utilizando un simple formulario, sin tener que tocar el código del sitio web.

Cuando se combina con PHP, se convierte en una mezcla poderosa, que siempre es tomada en cuenta para realizar aplicaciones del tipo cliente/servidor, que requieran el uso de una base de datos rápida, segura y potente.

MySQL, también ofrece la posibilidad de realizar programas o aplicaciones que requieran acceso a plataformas de base de datos rápidas. Acá tiene un poco de competencia, como PostgreeSQL y otras opciones, pero al ser libre y rápido, siempre va a tener una ventaja frente a sus rivales.

MySQL es muy utilizado en aplicaciones web, como Drupal o Joomla, en distintas plataformas , y por herramientas de seguimiento de errores como Bugzilla.

Su popularidad como aplicación web está muy ligada a PHP, que a menudo aparece en combinación con MySQL.

MySQL es una base de datos muy rápida en la lectura si utiliza el motor no transaccional MyISAM, pero es susceptible de provocar problemas en relación a la integridad cuando se utiliza en entornos con altas tasas de modificación. No obstante, por normal general, las aplicaciones web tienen baja concurrencia en la modificación de datos y en cambio el entorno es intensivo en lectura de datos, lo que hace a MySQL ideal para este tipo de aplicaciones.

Entre las principales características de MySQL es importante nombrar las siguientes:

- Aprovecha la potencia de sistemas multiprocesador, gracias a su implementación multihilo. Soporta gran cantidad de tipos de datos para las columnas.
- Dispone de API's en gran cantidad de lenguajes (C, C++, Java, PHP, etc).
- Gran portabilidad entre sistemas.
- Soporta hasta 32 índices por tabla.
- Gestión de usuarios y passwords, manteniendo un muy buen nivel de seguridad en los datos.
- Condición de open source de MySQL hace que la utilización sea gratuita y se puede modificar con total libertad.
- Se puede descargar su código fuente. Esto ha favorecido muy positivamente en su desarrollo y continuas actualizaciones.

- Es una de las herramientas más utilizadas por los programadores orientados a Internet.
- Infinidad de librerías y otras herramientas que permiten su uso a través de gran cantidad de lenguajes de programación.
- MYSQL, es el manejador de base de datos considerado como el más rápido de Internet.
- Gran rapidez y facilidad de uso.
- Infinidad de librerías y otras herramientas que permiten su uso a través de gran cantidad de lenguajes de programación.
- Fácil instalación y configuración.

2.7.3. PhpMyAdmin

Ilustración 7: Logo de phpMyAdmin

PhpMyAdmin es una herramienta escrita en PHP con la intención de manejar la administración de MySQL a través de páginas web, utilizando para ello Internet [10].

Actualmente esta herramienta permite crear y eliminar Bases de Datos, realizar operaciones CRUD (Create, Read, Update, Delete) sobre las tablas de una base de datos, borrar, editar y añadir campos, ejecutar cualquier sentencia SQL, administrar claves en campos, administrar privilegios, exportar datos en varios formatos y está disponible en 62 idiomas.

Este proyecto se encuentra vigente desde el año 1998, siendo el mejor evaluado en la comunidad de descargas de SourceForge.net como la descarga del mes de diciembre del 2002.

2.7.4. PHP

Ilustración 8: Logo de PHP

PHP es un lenguaje de programación de uso general cuyo código es utilizado para añadir funcionalidad al servidor web, originalmente diseñado para el desarrollo web de contenido dinámico. Ha sido uno de los primeros lenguajes de programación del lado del servidor que se podían incorporar directamente en el documento HTML en lugar de llamar a un archivo externo para el procesamiento de los datos [12].

Es un lenguaje interpretado por un servidor web con un módulo de procesador de PHP que genera la página web resultante. Es usado en la mayoría de los servidores web, así como en casi todos los sistemas operativos y plataformas sin coste adicional.

PHP se considera uno de los lenguajes más flexibles, potentes y de alto rendimiento conocidos hasta el momento.

PHP está enfocado principalmente a la programación de scripts del lado del servidor, por lo que se puede hacer cualquier cosa que pueda hacer otro programa CGI, como recopilar datos de formularios, generar páginas con contenidos dinámicos, o enviar y recibir cookies. Aunque PHP puede hacer mucho más.

Existen principalmente tres campos principales donde se usan scripts de PHP.

- Scripts del lado del servidor. Este es el campo más tradicional y el foco principal. Son necesarias tres cosas para que esto funcione. El analizador de PHP (módulo CGI o servidor), un servidor web y un

navegador web. Es necesario ejecutar el servidor con una instalación de PHP conectada. Se puede acceder al resultado del programa de PHP con un navegador, viendo la página de PHP a través del servidor. Todo esto se puede ejecutar en su máquina si está experimentado con la programación de PHP.

- Scripts desde la línea de comandos. Se puede crear un script de PHP y ejecutarlo sin necesidad de un servidor o navegador. Solamente es necesario el analizador de PHP para utilizarlo de esta manera. Este tipo de uso es ideal para scripts que se ejecuten con regularidad empleando cron (en *nix o Linux) o el Planificador de tareas (en Windows). Estos scripts también pueden usarse para tareas simples de procesamiento de texto.
- Escribir aplicaciones de escritorio. Probablemente PHP no sea el lenguaje más apropiado para crear aplicaciones de escritorio con una interfaz gráfica de usuario, pero si se conoce bien PHP, y se quisiera utilizar algunas características avanzadas de PHP en aplicaciones del lado del cliente, se puede utilizar PHP-GTK para escribir dichos programas. También es posible de esta manera escribir aplicaciones independientes de una plataforma. PHP-GTK es una extensión de PHP, no disponible en la distribución principal.

Las principales ventajas de PHP son:

- Es un lenguaje multiplataforma.
- Completamente orientado al desarrollo de aplicaciones web dinámicas con acceso a información almacenada en una Base de Datos.
- El código fuente escrito en PHP es invisible al navegador y al cliente ya que es el servidor el que se encarga de ejecutar el código y enviar su resultado HTML al navegador. Esto hace que la programación en PHP sea segura y confiable.
- Capacidad de conexión con la mayoría de los motores de base de datos que se utilizan en la actualidad, destaca su conectividad con MySQL y PostgreSQL.
- PHP es que puede funcionar en un servidor Windows y en LINUX.

- Su gran comunidad de PHP hace que el soporte, guías, libros y soluciones de dudas sea mucho mas facil en foros o redes sociales.
- PHP no requiere ningún tipo de licencia.
- Permite las técnicas de programación orientada a objetos.

Como desventajas se pueden enumerar las siguientes:

- El lugar mas seguro para ejecutar una aplicacion es en un servidor propio, por lo cual si un cliente o usuario requiere su codigo en su pc, tendriamos que dejar su codigo, sin manera de ocultarlo, aunque hay muchas aplicaciones que nos ayuda a encriptar el codigo fuente .
- Si no lo configuras correctamente dejas abiertas muchas brechas de seguridad.
- Se necesita instalar un servidor web.

2.7.5. Filezilla

Ilustración 9: Logo de Filezilla

FileZilla es un cliente FTP multiplataforma de código abierto y software libre, licenciado bajo la Licencia Pública General de GNU [11].

Es de los mejores softwares creados para gestionar archivos en el servidor web, subir y bajar los archivos ordenándolos en capetas, te permite renombralos y realizar muchas acciones más.

El programa FileZilla es sencillo para utilizar pero muy potente cliente para la gestión de archivos FTP (File Transfer Protocol) disponible para sistemas operativos Windows, Linux o Mac OS X desde la versión 3.0.0 y gracias al uso de wxWidgets (bibliotecas multiplataforma libres, para el diseño de interfaces gráficas). Es un

software libre bajo Licencia Pública General de GNU, soporta protocolos FTP, SFT, FTPS. En este artículo aprenderás mucho pues irás paso a paso conociendo FileZilla y sus características.

El protocolo FTP es utilizado para la transferencia de archivos entre PC y un servidor web, este protocolo se basa en la arquitectura cliente–servidor. Por ejemplo si te encuentras en un restaurante actuando como cliente (PC), necesitas a un camarero (cliente FTP), para que te traiga o se lleve cosas a la cocina (descargar o subir un archivo del servidor web). Por lo que podemos establecer que la relación es PC – FileZilla – servidor web.

El programa gestor, en este caso FileZilla te permite acceder a otro ordenador que funciona como servidor y gestionar los archivos o entre el servidor y tu PC. En muchas ocasiones te puedes encontrar con gestores de FTP dentro del sistema operativo o navegadores que utilizas, pero FileZilla es un programa muy completo para la realización de la gestión de archivos FTP.

FileZilla te permite visualizar los archivos y carpetas contenidos en un servidor, puedes subir archivos de tu PC al servidor, también descargar archivos desde el servidor a tu PC. Gracias a su sencilla interfaz puedes realizar funciones habituales de Windows como copiar, pegar, arrastrar y soltar para transferir archivos, entre otros.

Por ser un cliente utilizado por varios tipos de usuarios y para funciones que no son tan complejas, no necesitas tener altos grados de conocimiento en su manejo. Para usuarios con más experiencia, cuenta con herramientas de alta funcionalidad y más avanzadas para la gestión de archivos y sus transferencias.

3. REST

En este capítulo se describirán los principales aspectos de la arquitectura REST, usada en este trabajo para comunicar la aplicación móvil desarrollada en Android con el servidor web y los datos de la aplicación web.

3.1. Introducción

REST define un set de principios arquitectónicos por los cuales se diseñan servicios web haciendo foco en los recursos del sistema, incluyendo cómo se accede al estado de dichos recursos y cómo se transfieren por HTTP hacia clientes escritos en diversos lenguajes. REST emergió en los últimos años como el modelo predominante para el diseño de servicios. De hecho, REST logró un impacto tan grande en la web que prácticamente logró desplazar a SOAP y las interfaces basadas en WSDL por tener un estilo bastante más simple de usar [13].

SOAP es el acrónimo de “Simple Object Access Protocol” y es el protocolo que se oculta tras la tecnología que comúnmente denominamos “Web Services” o servicios web. SOAP es un protocolo extraordinariamente complejo pensado para dar soluciones a casi cualquier necesidad en lo que a comunicaciones se refiere, incluyendo aspectos avanzados de seguridad, transaccionalidad, mensajería asegurada y demás. Al final, los servicios web SOAP terminan siendo un monstruo con muchas capacidades pero que en la mayoría de los casos no necesitamos.

3.2. Ventajas

A continuación se describen las principales ventajas del uso de una arquitectura REST:

- Separación entre el cliente y el servidor: el protocolo REST separa totalmente la interfaz de usuario del servidor y el almacenamiento de datos. Eso tiene algunas ventajas cuando se hacen desarrollos. Por ejemplo, mejora la portabilidad de la interfaz a otro tipo de plataformas, aumenta la escalabilidad de los proyectos y permite que los distintos componentes de los desarrollos se puedan evolucionar de forma independiente.

- Visibilidad, fiabilidad y escalabilidad. La separación entre cliente y servidor tiene una ventaja evidente y es que cualquier equipo de desarrollo puede escalar el producto sin excesivos problemas. Se puede migrar a otros servidores o realizar todo tipo de cambios en la base de datos, siempre y cuando los datos de cada una de las peticiones se envíen de forma correcta. Esta separación facilita tener en servidores distintos el front y el back y eso convierte a las aplicaciones en productos más flexibles a la hora de trabajar.
- La API REST siempre es independiente del tipo de plataformas o lenguajes: la API REST siempre se adapta al tipo de sintaxis o plataformas con las que se estén trabajando, lo que ofrece una gran libertad a la hora de cambiar o probar nuevos entornos dentro del desarrollo. Con una API REST se pueden tener servidores PHP, Java, Python o Node.js. Lo único que es indispensable es que las respuestas a las peticiones se hagan siempre en el lenguaje de intercambio de información usado, normalmente XML o JSON.

3.3. Desventajas

Un servicio REST no tiene estado (es stateless), lo que quiere decir que, entre dos llamadas cualesquiera, el servicio pierde todos sus datos. Esto es, que no se puede llamar a un servicio REST y pasarle unos datos (p. ej. un usuario y una contraseña) y esperar que “nos recuerde” en la siguiente petición. De ahí el nombre: el estado lo mantiene el cliente y por lo tanto es el cliente quien debe pasar el estado en cada llamada. Si quiero que un servicio REST me recuerde, debo pasarle quien soy en cada llamada. Eso puede ser un usuario y una contraseña, un token o cualquier otro tipo de credenciales, pero debo pasarlas en cada llamada. Y lo mismo aplica para el resto de información.

El no tener estado es una desventaja clara: tener que pasar el estado en cada llamada es, como mínimo, tedioso, pero la contrapartida es clara: escalabilidad. Para mantener un estado se requiere algún sitio (generalmente memoria) donde guardar todos los estados de todos los clientes. A más clientes, más memoria, hasta que al

final podemos llegar a no poder admitir más clientes, no por falta de CPU, sino de memoria. Es algo parecido a lo que ocurre con la web tradicional (que también es stateless). Sea cual sea la tecnología con la que desarrolles para web, seguro que conoces que puedes crear lo que se llama una “sesión”. Los datos de la sesión se mantienen entre peticiones y son por cada usuario. El clásico ejemplo de sesión puede ser un carrito de la compra, entre las distintas peticiones web, la información del carrito de compra se mantiene.

3.4. Reglas utilizadas en una arquitectura REST

Define una serie de reglas que toda aplicación que pretenda llamarse REST debe cumplir. Como veremos, estas reglas se nos dan ya dadas si vamos a usar el protocolo HTTP, en cualquiera de sus implementaciones [14]:

- Arquitectura cliente-servidor: consiste en una separación clara y concisa entre los dos agentes básicos en un intercambio de información: el cliente y el servidor. Estos dos agentes deben ser independientes entre sí, lo que permite una flexibilidad muy alta en todos los sentidos.
- Cada mensaje HTTP contiene toda la información necesaria para comprender la petición. Como resultado, ni el cliente ni el servidor necesitan recordar ningún estado de las comunicaciones entre mensajes, es decir, si se demanda un recurso a un servidor en el que se le pasa una serie de datos, este servidor no será capaz de recordarlos para futuras peticiones de un cliente, por tanto el estado lo tiene que mantener el cliente y lo mantiene pasando el estado en cada una de las llamadas a un recurso del servidor.
- Stateless: como hemos mencionado arriba, esto significa que nuestro servidor no tiene porqué almacenar datos del cliente para mantener un estado del mismo. Esta limitación es sujeto de mucho debate en la industria, incluso ya empiezan a usarse tecnologías relacionadas que implementan el estado dentro de la arquitectura, como WebSockets. Como sabemos, HTTP también cumple esta norma, por lo que estamos acostumbrados ya a hacer uso de protocolos stateless.

- Cacheable: esta norma implica que el servidor que sirve las peticiones del cliente debe definir algún modo de cachear dichas peticiones, para aumentar el rendimiento, escalabilidad, etc. Una vez más, HTTP implementa esto con la cabecera “Cache-control”, que dispone de varios parámetros para controlar la cacheabilidad de las respuestas.
- Sistema por capas: nuestro sistema no debe forzar al cliente a saber por qué capas se tramita la información, lo que permite que el cliente conserve su independencia con respecto a dichas capas.
- Interfaz uniforme: esta regla simplifica el protocolo y aumenta la estabilidad y rendimiento del sistema. No queremos que la interfaz de comunicación entre un cliente y el servidor dependa del servidor al que estamos haciendo las peticiones, ni mucho menos del cliente, por lo que esta regla nos garantiza que no importa quien haga las peticiones ni quien las reciba, siempre y cuando ambos cumplan una interfaz definida de antemano.
- Operaciones bien definidas: en esta arquitectura se definen una serie de operaciones bien definidas para la consecución de una serie de recursos por parte del servidor y demandado por un cliente. Estas operaciones son las siguientes:

  GET

  POST

  PUT

  DELETE.

- Con frecuencia estas operaciones se equiparan a las operaciones CRUD en bases de datos (Create, Read, Update, Delete) que se requieren para la persistencia de datos, aunque POST no encajaría exactamente en este esquema.
- Con estas simples operaciones se puede realizar cualquier tipo de interactuación entre el cliente y el servidor utilizando REST.

- Sintaxis universal. Se utiliza una sintaxis universal para identificar los recursos. En un sistema REST, cada recurso es direccionable únicamente a través de su URI, la cual es simplemente una es una cadena de caracteres que identifica los recursos de una red de forma unívoca. Una URI se distingue de una URL en que estos últimos hacen referencia a recursos que, de forma general, pueden variar en el tiempo.

4. ANDROID

En el presente capítulo se tratará de forma general los principales rasgos de Android, la tecnología elegida para el desarrollo de la aplicación móvil para este trabajo.

4.1. Introducción

Ilustración 10: ¿Qué es Android?

Android es un sistema operativo de código abierto basado en Linux para dispositivos móviles, como smartphones o tablets. Android fue desarrollado por el Open Handset Alliance, dirigido por Google y otras compañías [18].

Android proporciona un enfoque unificado para el desarrollo software para móviles, lo que se traduce en que los desarrolladores sólo deben centrarse en desarrollar aplicaciones para Android, las cuales deberían ser soportadas en cualquier dispositivo que implementa el sistema operativo.

La primera versión Beta del Android Software Development Kit (SDK) fue lanzada por Google en 2007. No sería hasta septiembre de 2008, cuando Android lanzó su primera versión comercial, Android 1.0.

El 27 de junio de 2012, Google anuncia la próxima versión de Android, Android 4.1 (Jelly Bean). Esta versión presenta grandes cambios a las anteriores, y cuyo

objetivo es centrarse en mejora la interfaz de usuario, en términos de funcionalidad y eficiencia.

El código fuente de Android está disponible bajo licencia Open Source. Google publica la mayoría del código bajo licencia Apache, y el resto, cambios en el kernel de Linux, bajo la GNU General Public License versión 2.

Por lo general, las aplicaciones Android son desarrolladas usando el lenguaje de programación Java, a su vez usando el Android Software Development Kit.

Una vez desarrolladas, las aplicaciones Android pueden ser empaquetadas y vendidas a otros usuarios a través de los conocidos Markets, como el Google Play Store, SlideME, Opera Mobile Store, Mobango, F-droid, o el Amazon Appstore.

Android es usado en cientos de millones de dispositivos en más de 190 países en el mundo. Es el sistema operativo más instalado en plataformas móviles, y su crecimiento es increíble. Cada día, más de un millón de dispositivos Android son activados en el mundo.

La siguiente ilustración muestra los puntos fuertes de Android:

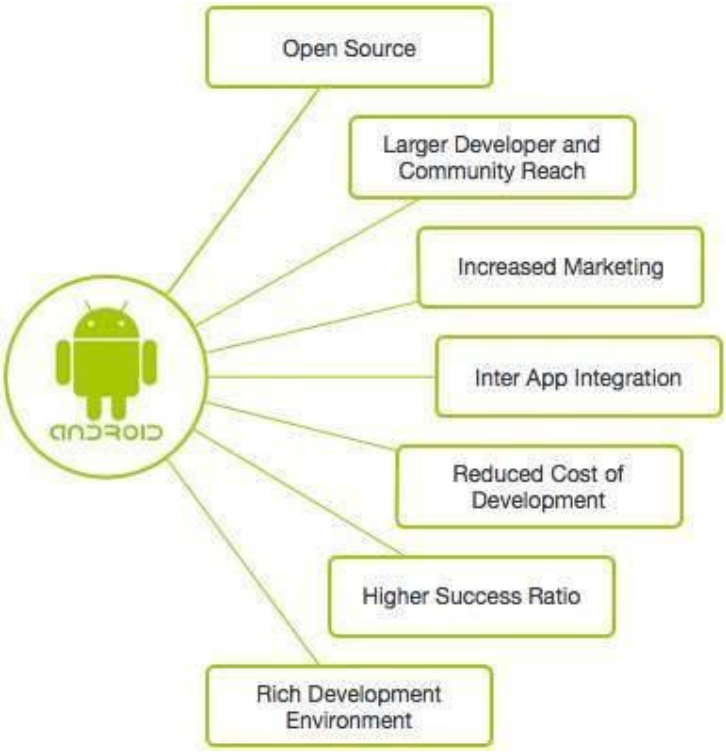


Ilustración 11: ¿Por qué Android?

4.2. Características de Android

Android es un potente sistema operativo que compite directamente con Apple. Android posee una amplia lista de características, las cuáles se listan en la siguiente tabla:

Característica	Descripción
Interfaz Intuitiva	Las pantallas basadas en Android proporcionan una bonita e intuitiva interfaz de usuario.
Conectividad	Grandes opciones de conectividad, tales como GSM/EDGE, IDEN, CDMA, EV-DO, UMTS, Bluetooth, Wi-Fi, LTE, NFC y WiMAX.
Almacenamiento	Android implementa SQLite, una ligera base de datos relacional que permite la persistencia de datos en las aplicaciones.
Soporte Multimedia	H.263, H.264, MPEG-4 SP, AMR, AMR-WB, AAC, HE-AAC, AAC 5.1, MP3, MIDI, Ogg Vorbis, WAV, JPEG, PNG, GIF, y BMP.
Mensajería	SMS y MMS
Navegador Web	Basado en el motor de código abierto WebKit, acoplado con Chrome v8, motor JavaScript y soporte para HTML5 y CSS3.
Multi-touch	Android tiene soporte nativo para el multi-touch, el cual estuvo disponible en primer lugar para el HTC Hero.
Multi-tarea	Android permite que el usuario pueda lanzar varias tareas al mismo tiempo, y que la ejecución de éstas sea en simultáneo.
Multi-Lenguaje	Soporte para una gran cantidad de idiomas.
Tethering	Android permite al teléfono ser usado como un punto de acceso alámbrico o inalámbrico. La mayoría de los dispositivos a partir de la versión 2.2 presentan esta característica.

Tabla 1: Caracaterísticas de Android

4.3. Componentes de Android

Los componentes principales de Android son los siguientes:

- Aplicaciones: las aplicaciones base incluyen aplicaciones como un cliente de correo electrónico, SMS, calendario, mapas (Google Maps), navegador web (Google Chrome en las últimas versiones), contactos, etc. Todas estas aplicaciones están escritas en el lenguaje de programación Java.
- Marco de trabajo de aplicaciones: los desarrolladores tienen acceso completo a las mismas APIs del framework que son utilizados por las aplicaciones base. La arquitectura está pensada y diseñada para la total reutilización de los componentes. Cualquier aplicación puede publicar sus capacidades y cualquier otra aplicación puede luego hacer uso de esas capacidades. Este mismo mecanismo permite que los componentes sean reemplazados por el usuario.
- Bibliotecas: Android incluye un conjunto de bibliotecas del lenguaje C y C++ usadas por varios componentes. Estas características se exponen a los desarrolladores a través del marco de trabajo de aplicaciones Android. Algunas de estas bibliotecas son por ejemplo System C library, bibliotecas de medios, bibliotecas de gráficos 3D, bibliotecas de SQLite, etc.
- Runtime de Android: Android incluye un set de bibliotecas base que proporcionan la mayor parte de las funciones disponibles en las bibliotecas base de Java. La máquina virtual está basada en registros y corre clases compiladas en Java que han sido transformadas al formato .dex por la herramienta “dx”. Desde la versión 5.0 se utiliza ART, que compila totalmente al momento de instalación de la aplicación.
- Núcleo Linux: Android depende de Linux para ofrecer los servicios más primitivos, tales como la seguridad, gestión de memoria, gestión de procesos, pila de red y modelo de controladores. El núcleo también actúa como una capa de abstracción entre hardware y software.

4.4. Arquitectura de Android

Como se ha mencionado en los puntos anteriores, Android es un SO para dispositivos móviles que contiene una pila de software donde se incluye un sistema operativo, middleware y aplicaciones básicas para el usuario.

A continuación se describen las principales capas que conforman una arquitectura Android. Cada una de estas capas utiliza (se apoya) en los servicios ofrecidos por las capas de niveles inferiores, y ofrece a su vez los suyos propios a las capas de nivel superior [17].

La siguiente ilustración resume las capas de la arquitectura Android:

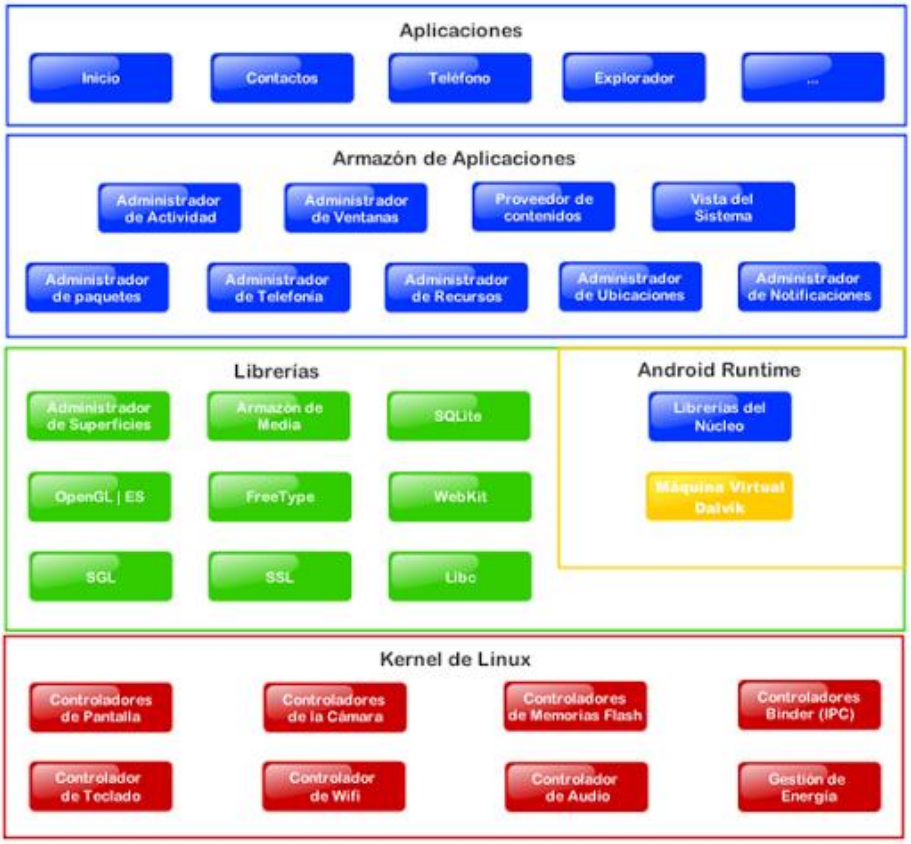


Ilustración 12: Arquitectura de Android

En los siguientes puntos se explicarán las diferentes capas de una arquitectura Android.

4.4.1. Aplicaciones

Esta capa contiene las aplicaciones por defecto de Android, así como las aplicaciones que el usuario va instalando posteriormente, ya sean de terceras personas o de su propio desarrollo. Estas aplicaciones utilizan los servicios, librerías y APIs de capas de niveles inferiores.

4.4.2. Framework de Aplicaciones

Conjunto de herramientas de desarrollo de cualquier aplicación. Toda aplicación desarrollada en Android. Las aplicaciones por defecto de Android, así como las aplicaciones desarrolladas por otras compañías, o incluso las que el propio usuario cree, utilizan el mismo conjunto de API y el mismo framework.

Entre las APIs más importantes, se pueden destacar las siguientes:

- Activity Manager: Conjunto de API que gestiona el ciclo de vida de las aplicaciones en Android.
- Window Manager: Gestiona las ventanas de las aplicaciones. Hace uso de la librería Surface Manager.
- Telephone Manager: Incluye todas las API vinculadas a las funcionalidades propias del teléfono (llamadas, mensajería, etc).
- Content Provider: Permite a cualquier aplicación compartir sus datos con las demás aplicaciones de Android. Por ejemplo, la información de contactos, agenda, mensajes, etc, será accesible para otras aplicaciones.
- View System: Proporciona un gran número de elementos para poder construir interfaces de usuario (GUI), como listas, mosaicos, botones, control de las interfaces mediante el teclado, etc. También incluye algunas vistas estándar para las funcionalidades más frecuentes.

- Location Manager: Posibilita a las aplicaciones la obtención de información de localización y posicionamiento.
- Notification Manager: Mediante el cual las aplicaciones, usando un mismo formato, comunican al usuario eventos que ocurran durante su ejecución.
- XMPP Service: Colección de API para utilizar este protocolo de intercambio de mensajes basado en XML.

4.4.3. Librerías

La siguiente capa hace referencia a las librerías utilizadas por Android. Éstas librerías están escritas en C/C++ y proporcionan a Android la mayor parte de sus capacidades más características. Junto al núcleo basado en Linux, estas librerías constituyen el corazón de Android [15].

Destacamos las siguientes librerías:

- Librería libc: Incluye las cabeceras y funciones del lenguaje C. Todas las demás librerías se definen en este lenguaje.
- Librería Surface Manager: Es la encargada de componer los diferentes elementos de navegación de pantalla. Gestiona las ventanas pertenecientes a las distintas aplicaciones activas en el dispositivo.
- OpenGL/SL y SGL: Representan las librerías gráficas y, por tanto, sustentan la capacidad gráfica de Android. OpenGL/SL maneja gráficos en 3D y permite la aceleración del hardware gráfico 3D (si el dispositivo lo permite). Por otro lado, SGL, proporciona gráficos en 2D, con lo cual es más utilizado en la mayoría de los aplicaciones. Una característica importante de Android es que permite combinar ambos tipos de gráficos, 2D y 3D.
- Librería Media Libraries: Proporciona todos los códecs necesarios para el soporte de contenido multimedia (video, audio, imágenes animadas, etc).
- FreeType: Permite trabajar de forma rápida y sencilla con distintos tipos de fuentes.

- Librería SSL: Posibilidad de utilización del protocolo SSL para el establecimiento de comunicaciones seguras.
- Librería SQLite: Creación y gestión de bases de datos relacionales.
- Librería SQLite: Creación y gestión de bases de datos relacionales.
- Librería WebKit: Proporciona un motor para las aplicaciones de tipo navegador y forma el núcleo del actual navegador incuido por defecto en la plataforma Android. En las últimas versiones del navegador de Android (Chrome) se utiliza el motor Blink, ya que WebKit ha quedado desfasado.

4.4.4. Tiempo de ejecución de Android

Esta capa se encuentra al mismo nivel que la capa de librerías de Android. Esta capa la constituyen las Core Libraries, librerías con multitud de clases Java y la máquina virtual Dalvik. Esta máquina virtual es sustituida más tarde por ART.

4.4.4.1. La máquina virtual Dalvik

Cabe hacer una mención especial sobre la máquina virtual sobre la que correremos la aplicación descrita en este proyecto.

Dalvik es la máquina virtual que utiliza Android. Ha sido diseñada por Dan Bornstein con contribuciones de otros ingenieros de Google.

La Máquina Virtual Dalvik (DVM) permite ejecutar aplicaciones programadas en Java. La mayoría de los programas escritos en Java 5 pueden correr sobre la DVM.

DVM sacrifica la portabilidad que caracteriza a Java para poder crear aplicaciones con un mejor rendimiento y menor consumo de energía, dos características absolutamente importantes en dispositivos móviles, debido a la limitada capacidad de las baterías de estos dispositivos. A su vez, está optimizada para requerir poca memoria y está diseñada para permitir ejecutar varias instancias

de la máquina virtual simultáneamente, delegando en el sistema operativo subyacente el soporte de aislamiento de procesos, gestión de memoria e hilos [16].

Dalvik es nombrada a menudo como una máquina virtual Java, aunque esto no es estrictamente cierto. La herramienta “dx” incluida en el SDK de Android permite transformar las clases Java compiladas al formato de archivos Dex.

Desde la versión 5.0 de Android (Lollipop), Dalvik fue sustituida por ART

4.4.4.2. La máquina virtual ART

Android Runtime (ART) es un entorno de ejecución de aplicaciones utilizado por Android. ART reemplaza a Dalvik, y lleva a cabo la transformación de la aplicación en instrucciones máquina, que luego son ejecutadas por el entorno de ejecución nativo del dispositivo.

Existe una principal diferencia con la máquina virtual Dalvik. Dalvik utiliza “just-in-time” (JIT) para compilar el código cada vez que se inicia una aplicación, en cambio, ART introduce el uso de “ahead-of-time” (AOT), que crea un archivo de compilación posterior a la instalación de la aplicación. Este archivo es utilizado al abrir la aplicación, evitando que la aplicación se compile continuamente cada vez que se ejecuta. Al reducir el número de compilaciones, el uso del procesador del dispositivo móvil se reduce y aumenta la duración de la batería. A su vez, ART mejora el rendimiento, por ejemplo, en la recolección de basura, aplicaciones de depuración y perfilado.

Para mantener la compatibilidad con versiones anteriores, ART utiliza el mismo código de bytes de entrada que Dalvik, suministrado a través de archivos .dex estándar como parte de los archivos APK, mientras que los archivos .odex son reemplazados por archivos de Formato Ejecutable y Enlazable (ELF). Una vez que una aplicación se compila utilizando ART en el dispositivo, ésta es dirigida exclusivamente a partir del ejecutable ELF compilado. Esto requiere de tiempo adicional de compilación cuando se instala la aplicación, y a su vez, las aplicaciones ocupan algo más de espacio para almacenar los compilados.

ART debutó como entorno de ejecución en alternativa a Dalvik en Android 4.4 (KitKat) y reemplaza completamente a Dalvik a partir de la versión 5 de Android (Lollipop).

4.4.5. El núcleo Linux

Android utiliza el núcleo de Linux como capa de abstracción para el hardware disponible en los dispositivos móviles. Esta capa contiene los drivers necesarios para que cualquier componente hardware pueda ser utilizado mediante las llamadas correspondientes. En resumen, se puede decir que el núcleo Linux hace de puente entre el software (Android) y el hardware.

5. CMS: Joomla!

En este capítulo, se describirá que es un CMS, cómo funciona y cuáles son los puntos fuertes por los cuales se ha decidido implementar esta solución. Finalmente se describirá el CMS Joomla!, el elegido de entre los CMS para resolver la problemática descrita en el capítulo uno de este trabajo.

5.1. ¿Qué es un CMS?

Un CMS es un programa que nos permite crear una estructura de soporte para la gestión (administración) de contenidos web, por parte de varios usuarios como pueden ser administradores, editores, participantes, etc.

Consiste en una interfaz que controla una o varias bases de datos donde se aloja el contenido de nuestro sitio web. Además, el CMS permite manejar de manera independiente el contenido y el diseño. También permite una publicación de contenidos ordenada y controlada orientada para distintos usuarios. Traduciendo esto a un lenguaje más llano, un CMS es una herramienta que permite al usuario final crear, publicar y clasificar información en un sitio web.

Ya desde hace unos años se han popularizado los sistemas de gestión de contenidos. Existen tantos cómo tipos de sitios web. Los hay para blogs, noticias, contenidos multimedia, destinados a empresas, magazines, y muchas otras. Tenemos tanto CMS open source desarrollados con licencia gratuita, cómo CMS de pago por los cuales tienes que pagar para obtener sus servicios.

Dentro de los CMS de licencia gratuita, encontramos tres grandes proyectos de los que trataremos en puntos posteriores. Estos CMS son WordPress, Joomla! y Drupal. Estos proyectos desarrollados independientemente son producto de largos años de evolución, y a su vez están respaldados por una gran comunidad de desarrollo que trabajan desinteresadamente para lanzar nuevas actualizaciones que mejoren cada una de las versiones de estos CMS.

Ilustración 13: Principales CMS

Como conclusión a esta introducción, decir que los sistemas de gestión de contenido están en gran auge en la actualidad, dan soporte a una gran variedad de tipo de contenidos en sitios web, y lo más fundamental, proporciona al usuario una herramienta potente para la creación de contenidos webs, incluso a editores inexpertos en edición y programación web.

5.2. Características de un CMS

A continuación se enumeran algunas de las ventajas más características de usar un CMS [19]:

- Facilidad de uso: Los CMS proporcionan al usuario final una interfaz gráfica para trabajar con los objetos que componen el sitio web. Esto hace de estos sistemas herramientas muy útiles para la creación y gestión de dicho contenido, ya que no cargan al usuario con la necesidad de saber medianamente bien el lenguaje web que se va a utilizar. Esta es una de las propiedades por las cuales los CMS están tan extendidos hoy día, debido a la cierta facilidad con la que un usuario puede manejar estos sistemas. Por tanto, no es necesario que el usuario sepa programar para publicar y gestionar contenido web

dinámico o estático. Aclarar también, que hay que tener unos conocimientos mínimos en programación y diseño web para crear un sitio web de calidad. La facilidad de uso de un CMS no quiere decir que un usuario con conocimientos mínimos de informática pueda realizar una página web de calidad.

- SEO e indexación en motores de búsqueda: Los CMS poseen una buena gestión del posicionamiento en buscadores, permitiendo controlar varias características especiales para conseguir que nuestro sitio web pueda aparecer en motores de búsqueda como Google, Yahoo, Bing, etc.
- Personalizable: Los sistemas de gestión de contenido suelen ser sistemas con un alto grado de personalización, el usuario puede personalizar el sitio web tanto como lo desee, desde el diseño de la web, hasta nuevas funcionalidades y opciones, agregando nuevos componentes, editando el diseño de la página, editando los colores, entre otras muchas cosas.
- Escalable: Esta propiedad es uno de los puntos fuertes de los CMS. Los CMS incorporan los llamados plugins o módulos que el usuario puede ir añadiendo en cualquier momento y que pueden aportar una nueva funcionalidad a nuestro sitio web. Estos módulos o plugins pueden ser definidos por el propio usuario (el usuario es el que programa el módulo), comprados (CMS de pago) u obtenidos a través de las páginas webs de los CMS de código abierto, de dónde se pueden obtener módulos o plugins desarrollados por la comunidad de desarrollo de dicho CMS.
- Seguridad: Los CMS proporcionan actualizaciones de seguridad frecuentes, incluyendo protocolos de encriptación de información sensible, seguridad en bases de datos y buen entendimiento con las opciones de seguridad del propio servidor.
- Código abierto: Los CMS más extendidos en la actualidad son de código abierto. Estos CMS cubren las mayorías de funcionalidades y disponen de una gran comunidad de desarrollo y mantenimiento.

- Facilidad de manejo de un gran número de páginas: Proporciona un sistema para distribuir los trabajos de creación, edición, y mantenimiento con permisos de acceso a las diferentes áreas. También proporcionan una gestión de metadatos de cada documento, publicación y caducidad de páginas, versiones, enlaces rotos, etc.
- Páginas interactivas: Los CMS incorporan la posibilidad de realizar páginas dinámicas. Las páginas dinámicas no existen en el servidor tal como se reciben en los navegadores, sino que se generan según las peticiones del usuario. Los CMS conectan con una base de datos que hace de repositorio central de todos los datos de la web.
- Consistencia y cambio de contenido en la web: Si no hay una buena separación entre contenido y presentación, un cambio en el diseño del sitio web podría comprometer el contenido de éste. Los CMS garantizan que se consiga la independencia de presentación (diseño) y contenido. Aplicando esta idea, podemos decir que los CMS garantizan la consistencia del sitio web, aplicando un mismo estilo para las diversas páginas que conforman el sitio web.
- Reutilización de objetos o componentes: Un CMS da la posibilidad de recuperar y reutilizar las páginas, documentos, y en general cualquier objeto publicado o almacenado con anterioridad.
- Control de acceso: El control de acceso no está limitado simplemente al hecho de permitir la entrada o no en la web. Los CMS permiten gestionar los diferentes permisos, de grupos o individuos, a las distintas áreas que conforman el sitio web.
- Soporte: Una de las características que han hecho de los CMS una herramienta tan usada, es el gran soporte que tienen. Como se ha mencionado con anterioridad, la mayoría de CMS disponen de un gran número de colaboradores, desinteresados o no, que mantienen en alza estos sistemas, desarrollando nuevos módulos, componentes o mejorando el propio CMS.

5.3. Estructura y componentes de un CMS

El primer y principal elemento que conforma un CMS es el Web Server. Es un servidor web donde se almacenan los archivos (HTML, PHP, CSS, SQL, JPG, etc.) que componen el sitio y puede.

El CMS también provee la instalación de elementos necesarios para el tratamiento de los archivos como PHP y SQL para su correcto funcionamiento.

Una vez cargado el CMS en el servidor, éste se conecta con una base de datos que almacenará contenidos generados en el back-end y en el front-end.

Ya tenemos el servidor web y la base de datos. El tercer componente que interfiere es la interfaz gráfica que proporciona el CMS. Una vez instalado en el servidor, proporciona al administrador una interfaz preestablecida dónde éste podrá empezar a trabajar en su sitio web.

La interfaz gráfica que representa el área de trabajo del administrador recibe el nombre de back-end.

La interfaz final que visualizarán los usuarios finales recibe el nombre de front-end.

Esta interfaz puede modificarse, con un diseño propio ((Graphic User Interface), estableciendo las características mediante las hojas de estilo CSS para las etiquetas HTML o agregando componentes que modifiquen la estructura visual del sitio) o bien adquiriendo e instalando temas prediseñados por otros usuarios. Existen empresas que se dedican al diseño de temas para los diferentes CMS que existen.

En la siguiente ilustración se muestra de forma gráfica, con explicaciones en texto, los principales componentes de un CMS:

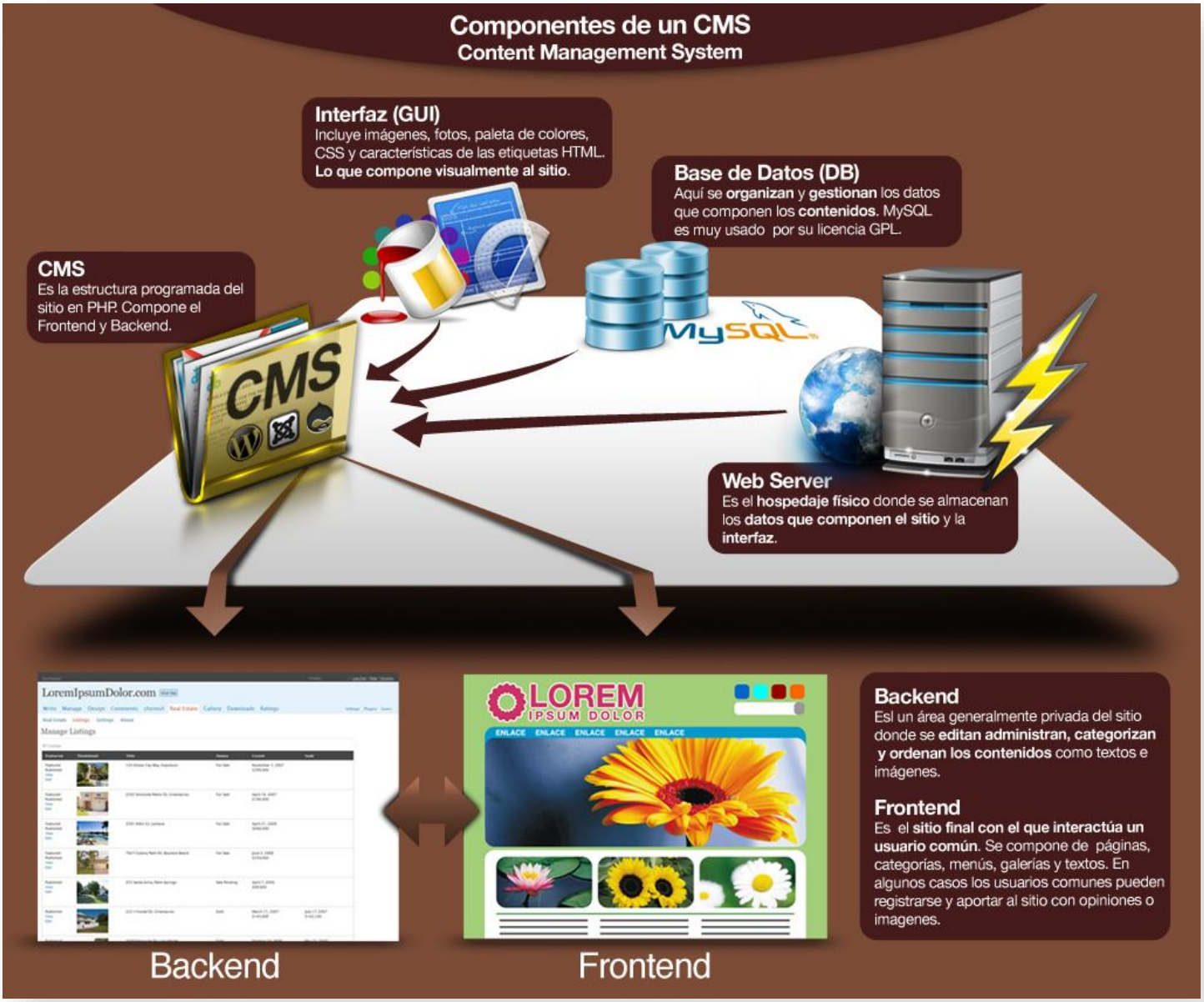


Ilustración 14: Componentes de un CMS

A nivel operativo de trabajo, un CMS dispone de dos partes:

- La parte web pública: Es la página web como tal que ven los usuarios que acceden a ella.

- La parte web privada: Desde cualquier lugar con conexión a internet y un navegador, con un usuario y contraseña podemos acceder a la parte de área de trabajo del gestor de contenidos. Una vez allí, según el perfil que seamos (que se nos tiene permitido hacer) mediante páginas web de uso interno llamadas “maquetadores” podemos actualizar el contenido de la página web pública, modificarlo y corregirlo, crear contenido nuevo, modificar el diseño, etc.

Si tenemos el nivel de permisos adecuado, dispondremos también de un panel de control en el cual se pueden cambiar las configuraciones del sitio web, el diseño, gestionar a los usuarios del sistema, etc.

Los CMS, con la autenticación, también permiten que haya varios administradores del sitio, por ejemplo, que haya un encargado del diseño, otro del contenido, colaboradores, etc.

En la siguiente ilustración se representa de forma gráfica la forma en la que se trabaja con un CMS:

Ilustración 15: Como trabaja un CMS

Un CMS a nivel de su estructura se distingue en 3 capas:

- La capa de la base de datos: La base de datos alberga el contenido que se ha escrito en la web, así como muchos de los parámetros de configuración, categorías, organización, usuarios y contraseñas. Los

sistemas de bases de datos más habituales de los CMS suelen ser MySql o Postgre.

- La capa de programación: Está contenida en los ficheros de la web. Lo que hacen estos ficheros al ejecutarse es solicitar la información que el usuario ha pedido desde el navegador de internet y extraerla para mostrarla al usuario de forma ordenada y estructurada colocándola en los lugares que le corresponde dentro del diseño de la página web. El lenguaje de programación más habitual suele ser PHP.
- La capa de diseño: Reside también en algunos ficheros. Define el diseño de la web, es decir la maquetación sobre la que se insertara el contenido que la programación se encarga de extraer de la base de datos. El lenguaje de programación y maquetación de la web es el HTML y CSS (estos lenguajes son complementados, y cada vez más con JavaScript y AJAX (Dinamismo)).

Cuando vemos la página web en un navegador no se ven estas capas diferenciadas, no vemos la base de datos, ni la programación, ni el código fuente del diseño, lo que vemos es la suma de todas ellas, el resultado final. A esta suma le llamamos la renderización de la página web.

Por la parte incómoda, esta estructura de capas hace que tengamos que para tener una buena copia de seguridad tengamos que tener copia de todas ellas. Por la parte positiva tener esta estructura técnica de capas permite se puede trabajar a la vez independientemente sobre cada una de ellas la programación y el diseño independientemente sin afectar a la otra y por tanto trabajar a la vez y en paralelo.

Por esto, en pocos minutos, instalando o cargando una nueva plantilla de diseño se puede tener una web completamente diferente en diseño, una visualización muy diferente, con el mismo contenido que la web anterior.

5.4. ¿Qué CMS es el más apropiado?

Muchos son los usuarios que se hacen esta pregunta. ¿Qué CMS es el más adecuado para mi aplicación? Cómo se ha mencionado en puntos anteriores, existen unas serie de características o requisitos que se deberían cumplir para una aplicación determinada, y que para un proyecto de envergadura, deberían ser estudiadas en profundidad para determinar cuál es el sistema de gestión de contenidos más adecuado.

Sin embargo, si lo que el usuario desea es una aplicación de una envergadura media – baja, podría seguir estos consejos:

- Drupal es válido para proyectos más bien grandes y comunidades, es más potente y su rendimiento será mayor para proyectos empresariales.
- Joomla generalmente es utilizado en proyectos pequeños, con sitios webs básicos que no son necesariamente blogs.
- WordPress es utilizado para cualquier tipo de proyecto, pero obtiene peores resultados que Drupal en proyectos grandes y requiere más experiencia. Especialización en blogs.

5.5. Joomla. Justificación de la solución

Joomla! es un potente CMS que permite crear sitios web elegantes, dinámicos e interactivos de forma simple. Este gestor de contenidos surge en 2005 como resultado de una división del proyecto Mambo. La primera versión de Joomla integraba el núcleo de Mambo, pero con nuevo software libre y muchos cambios importantes en el código. A partir de esta escisión, muchos colaboradores, comunidades y diseñadores, respaldaron el proyecto, que evolucionó hasta convertirse en lo que es hoy en día, uno de los CMS más usados y conocidos del mercado [20].

En la siguiente ilustración se muestra el logo del CMS Joomla!.

Ilustración 16: Logo de Joomla!

Joomla! es un potente CMS que permite crear sitios web elegantes, dinámicos e interactivos de forma simple. Este gestor de contenidos surge en 2005 como resultado de una división del proyecto Mambo. La primera versión de Joomla integraba el núcleo de Mambo, pero con nuevo software libre y muchos cambios importantes en el código. A partir de esta escisión, muchos colaboradores, comunidades y diseñadores, respaldaron el proyecto, que evolucionó hasta convertirse en lo que es hoy en día, uno de los CMS más usados y conocidos del mercado.

Joomla! es uno de los CMS más utilizados y mejor posicionado del mercado. En principio el proyecto está dirigido a proyectos de pequeña y media envergadura que requieren de forma principal presencia en Internet y comunicación: sitios web corporativos, comunidades de usuarios, tiendas online, etc. Para extraer todo el potencial de Joomla, se requiere cierto conocimiento y experiencia, ya que su máxima versatilidad se obtiene de la integración, adaptación y desarrollo de nuevos módulos.

Algunos datos significativos de Joomla!:

- En 2011, obtuvo una media semanal de descargas de alrededor de 86.500 descargas, el segundo CMS más descargado, después de WordPress.
- Amplia documentación.
- La visión de los usuarios con respecto a este CMS es positiva en casi un 50%.

Joomla! se encuentra liberado bajo una licencia GPL y utiliza PHP como lengujae de programación, MySQL como motor de base de datos, y Apache como servidor web.

También destaca su magnífica comunidad. Fruto de la gran participación de los usuarios, el sistema se encuentra en continua actualización frente a vulnerabilidades, errores, nuevas funcionalidades y extensiones. Dispone de un soporte muy completo a través de webs oficiales, foros y documentación generada.

Otra de las características destacadas es la versatilidad que ofrece el sistema a través de plantillas, extensiones y adaptaciones. Existen cientos de módulos, componentes y plugins que extienden la funcionalidad original del CMS: gestión de archivos, gestión de contactos, sistema de búsqueda, tiendas online, bolsas de trabajo, integración con redes sociales, gestión de noticias, sistemas de encuestas, etc. Estas extensiones se encuentran clasificadas en varias categorías, según su funcionalidad. También dispone de un apartado para visualizar las últimas extensiones subidas al portal o aquellas que han sido actualizadas.

Los puntos destacados de Joomla! son los siguientes:

- Joomla es uno de los CMS de software libre más conocidos del mercado. Destaca especialmente por la fortaleza de su comunidad de desarrollo y por la variedad de extensiones ofrecidas para ampliar la funcionalidad de la aplicación.
- La gestión del proyecto obtiene una puntuación excelente dentro de las áreas de frecuencia de versiones, soporte, control de versiones, gestión de bugs y transparencia de gestión.
- El proceso de instalación obtiene una buena valoración, ya que resulta bastante sencillo. También ofrece la posibilidad de acceder a una demostración online para poder probar y familiarizarse con el producto antes de su instalación.

En contrapuesta, Joomla! también tiene puntos a mejorar, entre los cuales destacan los siguientes:

- La gestión comercial es uno de los puntos más débiles del proyecto, no dispone de programa de partners, ni tampoco se puede acceder desde el portal a servicios de valor añadido, ni dispone de un programa de formación adecuado.
- La información del proyecto sólo se encuentra disponible en inglés.
- La actualización del sistema no está automatizada, lo que puede provocar que el proceso pueda llegar a ser un poco complejo.

Para concluir, se plasmarán las tres razones básicas por las cuales se ha elegido Joomla! para abarcar el problema:

- La primera de ellas es elegir un CMS de fácil aprendizaje.
- La segunda, destacar la gran documentación accesible desde Internet para añadir nuevas funcionalidades al sitio web.
- Gran comunidad de desarrollo, lo que se traduce en gran cantidad de plugins, módulos y plantillas que se pueden utilizar a la hora de la realización de un proyecto.
- En conlusión con el punto anterior, abstrae al desarrollador de elaborar sus propios plugins o extensiones, lo que se traduce en una gran ganancia de tiempo.

6. ANÁLISIS

6.1. Introducción

En los capítulos anteriores se han introducido los aspectos generales, como la motivación o las principales tecnologías y herramientas de forma teórica para desarrollar la idea de este trabajo.

En este capítulo se desarrollarán las ideas referentes al análisis. Como se cita en el capítulo 1, se utilizará la técnica del prototipado. El prototipado es una metodología muy útil para las partes involucradas en el desarrollo de la idea, como lo son los usuarios y el propio equipo de desarrollo.

Así, en los siguientes puntos se explicará detalladamente las técnicas utilizadas para el análisis de requerimientos, cómo lo son las entrevistas y los cuestionarios. También se detallará un diagrama de tiempo (Gantt) más aplicable a la parte de desarrollo, así como un estudio de coste del proyecto.

Llegados a este punto, aclarar que las suposiciones que detallan son puramente inventadas, pero que son basicamente realistas.

6.2. Análisis de requerimientos

El objetivo principal del análisis de requerimientos es reunir el conocimiento y requerimientos necesarios, tanto funcionales como no funcionales con la finalidad de tener una idea clara de los objetivos que el proyecto debe cumplir.

En este proyecto se han utilizado dos técnicas de análisis que se adaptan perfectamente al problema al que se quiere dar solución.

A continuación se expllican las dos técnicas utilizadas:

- Entrevistas: Técnica personal en la que el analista entra en contacto directo con las personas que forman parte de la organización, manteniendo una conversación en la cual el objetivo principal es obtener información acerca del funcionamiento de ésta. El objetivo es

que el propio equipo adquiera una vista objetiva real acerca de la organización.

Para llevar a cabo esta técnica se manera eficiente es recomendable seguir los siguientes puntos:

- Las preguntas han de ser claras y fáciles de responder.
- Trato formal y educado con las personas entrevistadas.
- Se debe promover un ambiente agradable con la finalidad de que el entrevistado conteste con la mayor sinceridad posible.
- No realizar preguntas comprometedoras ni que puedan compremeter a algún otro compañero.
- Utilizar material de grabación si lo consiente la persona entrevistada.

- Cuestionarios: Técnica muy útil y sencilla para obtener información sobre los usuarios. Es muy adecuada si la organización posee un gran número de usuarios. Un aspecto a tener en cuenta es que hay que ser cuiadoso a la hora de redactar el cuestionario, ya que hay que ser conciso y plasmar las preguntas necesarias para poder realizar un análisis posterior más correcto.

Estas dos técnicas han sido escogidas porque, según mi punto de vista, abarcan la problemática principal del problema. Se reallizan entrevista con la directiva de la empresa, de manera más personal, ya que es la directiva la que toma la decisión de implantar la solución tecnológica, y se reparten los cuestionarios a los socios de la peña para tener una visión más global acerca de las tendencias de estos.

6.2.1. Entrevistas

Se llevan a cabo una serie de entrevistas con la directiva de la empresa. En estas entrevistas, el cliente, la peña deportiva Viejos Amigos FS plasma el desea de informatizar la peña, debido a la creciente demanda de abonos anuales y que los usuarios reclaman que no conocen la mayoría de servicios y actividades que van surgiendo.

De estas entrevistas, el equipo puede sacar las siguientes conclusiones:

- El cliente requiere poder gestionar a los socios de forma automática.
- Se habilitará una sección de noticias para informar a los socios sobre noticias de interés, así como de los nuevos retos de la peña.
- Se publicarán los servicios y actividades en la web, y los socios podrán adquirir estos directamente.
- La gestión de los pagos se seguirán haciendo, en la primera versión del producto, por transferencia bancaria.
- Se promoverá la participación activa en las actividades de la peña.
- Se promoverá a la peña en redes sociales.
- Se requiere una aplicación móvil de funciones reducidas cuya funcionalidad irá aumentando con el tiempo.

En este punto, la directiva tiene una idea bien clara de lo que quiere. En el siguiente punto, se realiza un cuestionario que será promulgado a los socios de la peña. Los resultados confirmarán los requerimientos de la aplicación.

6.2.2. Cuestionarios

Una vez realizada la entrevista con la directiva, el objetivo es conocer más acerca de los socios de la peña. Para ello se les pide que realicen una pequeña encuenta de 10 preguntas. (**Ver Anexo I**).

A continuación se detallan los resultados teniendo en cuenta las respuestas de los usuarios a las preguntas del cuestionario:

1. ¿Cuántos años tiene?

A) 15 – 30 años
B) 30 – 40 años
C) 40 – 60 años
D) Más de 60 años

Resultados y conclusión: El 60% de los socios tienen edades comprendidas entre los 15 y los 40 años. Un 30% de ellos tiene entre 40 y 60 años, y sólo un 10% tiene más de 60 años.

De aquí podemos sacar la conclusión de que la mayoría de socios son gente relativamente joven, aunque también hay un porcentaje de personas a las cuales les puede ser difícil adaptarse a las nuevas tecnologías, por lo cual, se deberían de desarrollar interfaces lo más usables posibles.

2. ¿En qué situación laboral se encuentra?

A) Estudiante
B) Trabajando
C) En situación de desempleo
D) Jubilado

Resultados y conclusión: La mayoría de los socios son estudiantes y trabajan, y sólo una mínima parte están jubilados o en situación de desempleo. Podemos sacar la conclusión de que no será difícil una adaptación a la implantación tecnológica.

3. ¿Está familiarizado con la tecnología?

A) Sí
B) No

Resultados y conclusión: Sólo el 2% de los encuestados no está familiarizado con la tecnología, por lo que deducimos que la solución tendrá buena adaptación entre los socios de la peña.

4. ¿Con qué frecuencia se conecta a Internet?

A) Diariamente
B) Una vez a la semana
C) Una vez al mes
D) No sé que es internet

Resultados y conclusión: La mayoría de usuarios eligen la opción A y B, por lo tanto, se podrán aprovechar las ventajas de una aplicación web.

5. ¿Tiene un smartphone o teléfono inteligente?

A) Sí
B) No

Resultados y conclusión: El 90% de los encuentados tienen un smartphone, por tanto sería interesante empezar a desarrollar una aplicación móvil.

6. Seleccione si es usuario de alguna de estas redes sociales.

A) Facebook
B) Twitter
C) Instagram
D) Google +

Resultados y conclusión: La red social más utilizada por los socios es Twitter, con lo cuál, no sería descabellado integrar la API de Twitter en las aplicaciones.

7. ¿Lee usted la prensa digital deportiva?

A) Sí, a diario
B) Alguna que otra vez
C) No

Resultados y conclusión: Ningún encuestado selecciona la opción C. Sería buena idea incluir el bloque de noticias que requiere la directiva.

8. ¿Con qué frecuencia participa en las actividades que promueve la peña desportiva Viejos Amigos FS?

A) Suelo asistir a casi todas las actividades
B) Asistencia media, siete u ocho actividades por año
C) No asisto a ninguna actividad

Resultados y conclusión: Las opciones más frecuentes son la A y la B. Son usuarios participativos en las actividades que promueve la peña.

9. ¿Le gustaría que Viejos Amigos FS tuviera presencia en Internet?

A) Sí
B) No

Resultados y conclusión: El 100% contesta Sí. No hay usuarios reacios a la implantación de la solución.

10. ¿Qué le gustaría poder hacer en el sitio web?

A) Leer noticias acerca de la peña
B) Consultar las nuevas actividades promovidas por la peña
C) Contratar actividades o servicios
D) Poder actualizar las estadísticas de aquellos socios que participan en las actividades deportivas
E) Poder pagar a través de la web

Resultados y conclusión: Pocos usuarios seleccionan la opción E. Podemos suponer que la mayoría de socios no confían en los pagos por internet.

6.3. Requerimientos funcionales

Los requerimientos funcionales son aquellos requisitos necesarios para que la aplicación sea beneficiosa para el cliente.

A continuación se detalla la lista de requisitos funcionales que debe cumplir la aplicación web:

- La aplicación debe ser usable, fácil de usar y de aprender.
- La aplicación contendrá una sección de noticias dónde se informará de noticias de carácter relevante y de los diferentes servicios y actividades que ofrece la peña.
- Los usuarios podrán contratar los servicios/actividades a través de la aplicación web.
- La aplicación debe ser segura.

- Se establecerán tipos de usuario ya que la administración podrá realizar acciones que no podrán realizar los usuarios (socios).
- Los administradores podrán accedere a las fichas (información) de los socios.
- Los administradores podrán confirmar los pagos cuando comprueben que se ha hecho la transferencia bancaria con el pago del servicio/actividad.
- Los administradores podrán crear nuevos servicios/actividades a través de la web.
- El usuario podrá ver sus datos personales.

Recordamos en este punto, que la aplicación móvil contendrá la misma información que la aplicación web, pero la funcionalidad será mucho menor. Sólo tendrá un tipo de usuario (no se distingue entre administradores/socios), con lo cual podemos resumir los siguientes requisitos funcionales:

- El usuario de la aplicación móvil podrá consultar las noticias asociadas a la peña.
- El usuario de la aplicación móvil podrá consultar sus servicios contratados, tanto los confirmados, como los no confirmados.
- El usuario de la apllicación móvil podrá consultar sus datos personales.

6.4. Requerimientos no funcionales

Los requerimientos no funcionales son aquellos que hay que tener en cuenta para que la aplicación pueda llevarse a cabo en todos sus factores y cubriendo sus objetivos.

A continuación se resumen los requerimientos no funcionales de la aplicación web:

- Se incluirá la api de Twitter en la aplicación web.

- Se integrará un módulo de estadísticas para aquellos socios que habitualmente participan en partidos promovidos por la peña.
- Los socios no podrán acceder a la parte de la administración.
- Los socios sólo podrán apuntarse una sóla vez a una actividad ofertada.
- La aplicación mejorará el tiempo de gestión de los administradores.

En cuanto a la aplicación móvil, los requerimientos no funcionales serán:

- Se integra la API de twitter en la aplicación móvil.
- Se guardará la sesión del usuario para mejorar los tiempos de acceso a la aplicación.

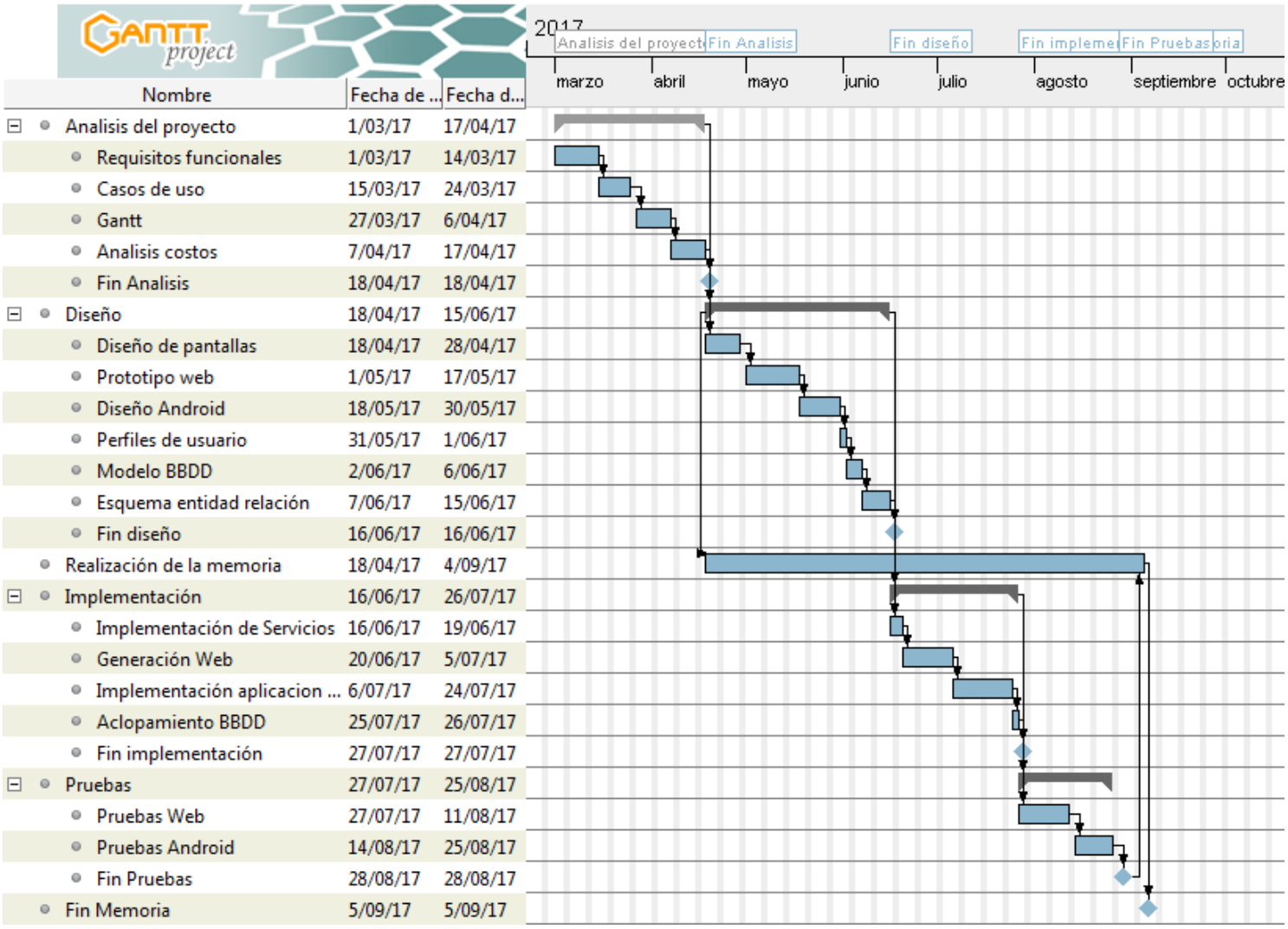
6.5. Usuarios y roles

En este apartado se definen las acciones que pueden llevas a cabo los diferentes tipos de usuarios de la aplicación web.

- Invitado: cualquier persona puede acceder al sitio web y leer las noticias, ver el timeline de twitter, pero no podrá realizar acciones a no ser que se registre. Las acciones que puede realizar son:
  - Leer noticias de la peña deportiva.
  - Registrarse en la web.
- Registrado: Este perfil hace referencia a un socio de la peña. Puede realizar las siguientes acciones:
  - Leer noticias de la peña deportiva.
  - Iniciar sesión.
  - Acceder al detalle de las noticias.
  - Adquirir servicios/actividades.
  - Consultar servicios contratados.
  - Consultar datos personales.
- Administrador: Este perfil hace referencia a los administradores/directivos de la peña. Puede realizar las siguientes acciones:

- Leer noticias de la peña deportiva.
- Iniciar sesión.
- Acceder al detalle de las noticias.
- Adquirir servicios/actividades.
- Consultar servicios contratados.
- Consultar datos personales.
- Confirmar pagos de los socios.
- Añadir nuevos servicios/actividades.
- Ver listado de usuarios.
- Ver ficha de usuarios.

6.6. Planificación temporal



Nombre	Fecha de ...	Fecha d...
Analisis del proyecto	1/03/17	17/04/17
Requisitos funcionales	1/03/17	14/03/17
Casos de uso	15/03/17	24/03/17
Gantt	27/03/17	6/04/17
Analisis costos	7/04/17	17/04/17
Fin Analisis	18/04/17	18/04/17
Diseño	18/04/17	15/06/17
Diseño de pantallas	18/04/17	28/04/17
Prototipo web	1/05/17	17/05/17
Diseño Android	18/05/17	30/05/17
Perfiles de usuario	31/05/17	1/06/17
Modelo BBDD	2/06/17	6/06/17
Esquema entidad relación	7/06/17	15/06/17
Fin diseño	16/06/17	16/06/17
Realización de la memoria	18/04/17	4/09/17
Implementación	16/06/17	26/07/17
Implementación de Servicios	16/06/17	19/06/17
Generación Web	20/06/17	5/07/17
Implementación aplicacion ...	6/07/17	24/07/17
Aclopamiento BBDD	25/07/17	26/07/17
Fin implementación	27/07/17	27/07/17
Pruebas	27/07/17	25/08/17
Pruebas Web	27/07/17	11/08/17
Pruebas Android	14/08/17	25/08/17
Fin Pruebas	28/08/17	28/08/17
Fin Memoria	5/09/17	5/09/17

Ilustración 17: Diagrama de Gantt

En la ilustración 17 se observa la descomposición de tareas llevadas a cabo en este proyecto, así como su planificación temporal.

El proyecto se divide en cuatro fases; en la fase de análisis, la más importante del proyecto, se exponen los requisitos necesarios para el proyecto, su asignación, así como un análisis de tiempo y costo que se necesitarán en las siguientes fases.

El diseño nos da una imagen de cómo va a quedar el proyecto una vez implementado. Es una guía completa de los pasos a seguir para la consecución en el tiempo predicho con los recursos estimados.

La implementación y las pruebas están íntimamente ligadas, para conseguir cumplir los requisitos no funcionales que se le pueden exigir a este tipo de proyectos.

La memoria es el proceso que esta guiado por todas las demás tareas permitiendo así una documentación correcta y precisa.

En la ilustración 18 se muestra un esquema con la jerarquía y secuenciación de tareas mostrando el camino critico que lleva a la consecución del proyecto:

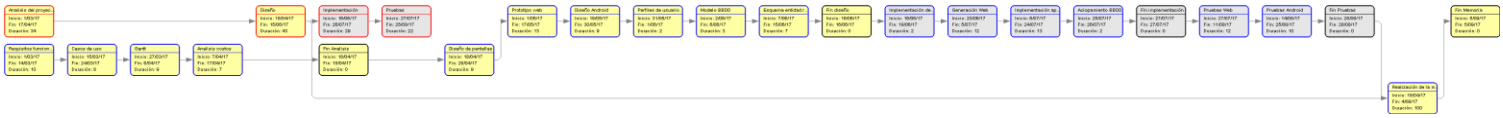


Ilustración 18: PERT

6.7. Planificación de costos

Para la estimación de costes asociados a este proyecto utilizaremos la metodología que nos ofrece el modelo COCOMO. Este modelo, basado en número de líneas de código, esfuerzo por persona y otras variables basadas en conocimientos y experiencia del personal encargado para el proyecto, se dará una aproximación considerablemente buena tanto del esfuerzo de desarrollo como del tiempo necesario de desarrollo.

Al final de este proceso tendremos un esfuerzo promedio de cada mes así como la productividad necesaria para llevarlo a cabo.

COCOMO utiliza para su cometido unas fórmulas que son explicadas a continuación:

- E=(a)(KLDC)b(EAF)
- D=(c)(E)d
- PR=LCD/E
- P=E/D

Siendo “E” el esfuerzo aplicado en hombre mes, “D” el tiempo de desarrollo en meses, PR la productividad, P personal promedio necesario, “KLDC” número de miles de líneas de código estimado para el proyecto, LDC número de líneas de código y EAF el factor de ajuste del esfuerzo que se calcula valorando en la siguiente escala de 15 atributos.

ATRIBUTOS	Valor					
	Muy bajo	Bajo	Nominal	alto	Muy alto	Extra alto
Atributos del software						
Fiabilidad	0,75	0,88	1,00	1,15	1,40	
Tamaño de base de datos		0,94	1,00	1,08	1,16	
Complejidad	0,70	0,85	1,00	1,15	1,30	1,65
Atributos del hardware						
Restricciones del tiempo de ejecución			1,00	1,11	1,30	1,66
Restricciones de memoria virtual			1,00	1,06	1,21	1,56
Volatilidad de la máquina virtual		0,87	1,00	1,15	1,30	
Tiempo de respuesta		0,87	1,00	1,07	1,15	
Atributos de personal						
Capacidad de análisis	1,46	1,19	1,00	0,86	0,71	
Experiencia en la aplicación	1,29	1,13	1,00	0,91	0,82	
Calidad de programadores	1,42	1,17	1,00	0,86	0,70	
Experiencia en la máquina virtual	1,21	1,10	1,00	0,90		
Experiencia en el lenguaje	1,14	1,07	1,00	0,95		
Atributos del proyecto						
Técnicas actualizadas	1,24	1,10	1,00	0,91	0,82	

de programación					
Utilización de herramientas de software	1,24	1,10	1,00	0,91	0,83
Restricciones de tiempo de desarrollo	1,22	1,08	1,00	1,04	1,10

Tabla 2: Atributos COCOMO

Empezamos definiendo el tipo de proyecto, según las especificaciones que nos impone el modelo COCOMO, definiéndolo así como semi-acoplado dado a los siguientes factores:

- Es realizado por una sola persona
- Desarrollo de software con ciertas restricciones
- No se basa en experiencias anteriores del desarrollador

Tipo de proyecto	a	b	c	d
Orgánico	2.4	1.05	2.5	0.38
Semiacoplado	3.0	1.12	2.5	0.35
Empotrado	3.6	1.20	2.5	0.32

Ilustración 19: Proyectos COCOMO

Al elegir el tipo de proyecto semi-acoplado, los valores para las variables “a, b, c, y d” quedan como se observa en la ilustración 19.

Para calcular el factor de ajuste del esfuerzo escogemos:

- Para los atributos del software un valor nominal
- Para los atributos del hardware valor alto
- Para los atributos del personal valor nominal
- Para los atributos del proyecto valor alto

Multiplicando los valores obtenidos mediantes estas asignaciones de la figura 1.5.1 obtenemos un valor de EAF= 1,24.

Aplicando estos resultados las formulas anteriores obtenemos estos resultados:

- E=3*51,12*1,24= 22,56 pers/mes
- D=2,5*22,560,35= 7,44 meses
- PR=6000/22,56= 265 lineas de código/persona mes
- P=22,56/7,44= 3,03 personas

Según estos resultados se necesitan un equipo de 3 personas trabajando alrededor de 7 meses, con un promedio de 221 líneas de código cada persona al mes, pero dado que nuestro plazo era de 9 meses y partíamos de código ya realizado, el proyecto puede ser llevado a cabo por una persona en el tiempo especificado.

Dado que la mitad del trabajo, sobre unos 3 meses y medio, recaería sobre un analista de software y que el salario medio entre los profesionales pertenecientes a este sector es de 2.481€/mes, su salario ascendería a la suma de 8.683,5€

La otra mitad del trabajo, sobre unos 3 meses y medio, recaería sobre un programador software y dado que el salario medio entre los profesionales pertenecientes a este sector es de 2100 €/mes, su salario ascendería a la suma de 7350€

El coste total en euros para llevar a cabo este proyecto sería de 16.033,5€

7. DISEÑO

7.1. Introducción

La finalidad de este capítulo es dar al lector una visión clara sobre las aplicaciones que ocupan este trabajo. Se emplearán técnicas de más a menos abstractas para conseguir dicho fin.

Para el diseño de las aplicaciones se pueden utilizar una amplia gama de técnicas de ingeniería del software, aunque teniendo en cuenta el prototipado, para este trabajo se emplearán sólo tres de ellas.

A continuación se lista el desglose de tareas que se desarrollará en este capitulo. Como nota aclaratoria, decir que la aplicación móvil compartirá el diseño de la base de datos, así como los casos de uso correspondientes ya que la funcionalidad será la misma. Si cambiará el diseño de los storyboard, ya que las tecnologías utilizadas si que difieren en el tema de diseño de la interfaz.

Las técnicas utilizadas para el diseño de las aplicaciones serán:

- Diseño de la base de datos.
- Diseños de la aplicación desglosado en:
  - Casos de uso.
  - Storyboard de la aplicación web.
  - Storyboard de la aplicación móvil.

A través de los casos de uso identificaremos con exactitud las diferentes interacciones de el sistema con el usuario. Con el apoyo de los casos de uso se obtendrá la información necesaria para poder desarrollar los primeros prototipos (storyboard), tanto de la aplicación web como de la aplicación móvil.

7.2. Diseño de la base de datos

En este punto se detallarán los diferentes datos que necesitaremos para desarrollar nuestra aplicación.

Se puede considerar que el diseño de la base de datos es una tarea que nos dará los cimientos de nuestra aplicación, con lo cual si no se hace un buen diseño de ésta, podría causar problemas en un futuro, teniendo que aplicar un esfuerzo considerable para su solución.

Llegados a este punto, cabe destacar que al utlizar la herramiento Joomla, la base de datos contendrá un número elevado de tablas que utiliza la herramienta para su correcto funcionamiento. Nos centraremos unicamente en las entidades relevantes en el desarrollo del flujo de nuestra aplicación. Algunas de estas entidades serán propias de la gestión de Joomla, y otras serán creadas manualmente para que el sistema cumpla con sus objetivos.

Para la realización de este trabajo, se utilizará el modelo entidad-relación. El modelo entidad-relación utiliza el mundo real para representar una serie de objetos, las entidades, las cuales poseen una serie de atributos. También se definen relaciones entre entidades, apoyándose en los atributos de éstas.

A continuación se realizará un estudio de las entidades más importantes que intervendrán en las aplicaciones.

7.2.1. Entidades

Para cada entidad se explicará su definición, la aplicación que tiene dentro del modelo, y los atributos que la definen:

Usuario (Tabla pr_users):

Esta entidad representa a los usuarios de la aplicación. Es creada automaticamente por Joomla y contiene un gran número de atributos utilizados por esta potente herramienta.

#	Nombre	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado	Extra
1	id	int(11)			No	*Ninguna*	AUTO_INCREMENT
2	name	varchar(255)	utf8_general_ci		No		
3	username	varchar(150)	utf8_general_ci		No		
4	email	varchar(100)	utf8_general_ci		No		
5	password	varchar(100)	utf8_general_ci		No		
6	block	tinyint(4)			No	0	
7	sendEmail	tinyint(4)			Sí	0	
8	registerDate	datetime			No	0000-00-00 00:00:00	
9	lastvisitDate	datetime			No	0000-00-00 00:00:00	
10	activation	varchar(100)	utf8_general_ci		No		
11	params	text	utf8_general_ci		No	*Ninguna*	
12	lastResetTime	datetime			No	0000-00-00 00:00:00	
13	resetCount	int(11)			No	0	
14	otpKey	varchar(1000)	utf8_general_ci		No		
15	otep	varchar(1000)	utf8_general_ci		No		
16	requireReset	tinyint(4)			No	0	

Ilustración 20: Tabla Usuario

- id: Identificador único del usuario. Clave primaria de la tabla.
- name: nombre del usuario.
- username: usuario de Joomla.
- email: correo electrónico del usuario.
- password: contraseña del usuario.

Se ha excluido la explicación de los demás atributos ya que son utilizados por el motor de Joomla y son transparentes para nosotros.

Perfil Usuario (Tabla pr_user_profiles):

Esta entidad representa los datos personales del usuario, tales como su dirección, su número de teléfono, su fecha de nacimiento, etc. Está gestionada por Joomla y su funcionamiento es el típico de un diccionario [clave, valor]. Esto quiere decir, que por ejemplo, para la dirección del usuario, key tendrá el valor “dirección” y value tendrá el valor “C/ del Ejemplo nº 1”.

	#	Nombre	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado
☐	1	**user_id**	int(11)			No	*Ninguna*
☐	2	**profile_key**	varchar(100)	utf8_unicode_ci		No	*Ninguna*
☐	3	**profile_value**	text	utf8_unicode_ci		No	*Ninguna*
☐	4	**ordering**	int(11)			No	0

Ilustración 21: Tabla Perfil Usuario

- user_id: clave foránea a la tabla Usuario (pr_users).
- profile_key: clave del diccionario. Tomará valores como por ejemplo dirección, teléfono, año de nacimiento, etc.
- profile_value: valor del diccionario. Tomará el valor concreto para su clave correspondiente.
- ordering: índice de ordenación de las “propiedades” del usuario.

Contenido (Tabla pr_content):

Esta entidad representa el contenido de la aplicación. Está gestionada por Joomla y embarca todo el contenido de la aplicación, tanto noticias como formularios que intervienen en la lógica de la aplicación.

#	Nombre	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado	Extra
1	id	int(10)		UNSIGNED	No	*Ninguna*	AUTO_INCREMENT
2	asset_id	int(10)		UNSIGNED	No	0	
3	title	varchar(255)	utf8_general_ci		No		
4	alias	varchar(255)	utf8_bin		No		
5	introtext	mediumtext	utf8_general_ci		No	*Ninguna*	
6	fulltext	mediumtext	utf8_general_ci		No	*Ninguna*	
7	state	tinyint(3)			No	0	
8	catid	int(10)		UNSIGNED	No	0	
9	created	datetime			No	0000-00-00 00:00:00	
10	created_by	int(10)		UNSIGNED	No	0	
11	created_by_alias	varchar(255)	utf8_general_ci		No		
12	modified	datetime			No	0000-00-00 00:00:00	
13	modified_by	int(10)		UNSIGNED	No	0	
14	checked_out	int(10)		UNSIGNED	No	0	
15	checked_out_time	datetime			No	0000-00-00 00:00:00	
16	publish_up	datetime			No	0000-00-00 00:00:00	
17	publish_down	datetime			No	0000-00-00 00:00:00	
18	images	text	utf8_general_ci		No	*Ninguna*	
19	urls	text	utf8_general_ci		No	*Ninguna*	
20	attribs	varchar(5120)	utf8_general_ci		No	*Ninguna*	
21	version	int(10)		UNSIGNED	No	1	
22	ordering	int(11)			No	0	
23	metakey	text	utf8_general_ci		No	*Ninguna*	
24	metadesc	text	utf8_general_ci		No	*Ninguna*	
25	access	int(10)		UNSIGNED	No	0	

Ilustración 22: Tabla de Contenido

- id: clave primaria de la tabla. Identifica univocamente cada recurso de contenido de la aplicación.
- title: titulo del contenido (articulo, formulario, etc).
- alias: alias identificativo del recurso de contenido.
- introtext: representa el contenido en sí. Puede ser texto plano o código (html, php, css, javascript).
- fulltext: sirve como apoyo a introtext si el contenido de éste es demasiado grande.
- state: estado del contenido. Puede ser publicado o despublicado.

- created: fecha de creación del contenido.
- created_by: clave foránea a la tabla usuario. Identica al creador del recurso de contenido.
- access: controla el acceso al recurso de contenido dependiendo de los roles de los usuarios.

Se han detallado los atributos más importantes obviando algunos de ellos internos de Joomla.

Servicio (Tabla pr_servicios):

Esta entidad representa los servicios y actividades que se ponen a disposición de los usuarios.

	#	Nombre	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado	Extra
☐	1	id	int(11)			No	*Ninguna*	AUTO_INCREMENT
☐	2	nombre	text	utf8_spanish_ci		No	*Ninguna*	
☐	3	precio	text	utf8_spanish_ci		No	*Ninguna*	
☐	4	caducidad	date			Sí	*NULL*	
☐	5	descripcion	text	utf8_spanish_ci		No	*Ninguna*	

Ilustración 23: Tabla de Servicio

- id: clave primaria de la tabla. Representa univocamente cada uno de los servicios/actividades.
- nombre: nombre del servicio/actividad.
- precio: precio del servicio/actividad.
- caducidad: fecha de caducidad del servicio/actividad.
- descripcion: descripción detallada del servicio/actividad.

Pago (Tabla pr_pagos):

Esta entidad representa la contratación de un servicio/actividad por parte de un usuario. Se puede decir que es la entidad que une las entidades Usuario y Servicio.

	#	Nombre	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado	Extra
	1	id_pago	int(11)			No	*Ninguna*	AUTO_INCREMENT
	2	id_user	int(11)			No	*Ninguna*	
	3	id_servicio	int(11)			No	*Ninguna*	
	4	caducidad	date			No	*Ninguna*	
	5	pagado	tinyint(4)			No	*Ninguna*	

Ilustración 24: Tabla de Pago

- id_pago: clave primaria de la tabla. Identifica univocamente cada pago que se realiza en la aplicación.
- id_user: clave foránea a la tabla de usuarios. Identifica al usuario que realiza la contratación.
- id_servicio: clave foránea a la tabla de servicios. Identifica el servicio/actividad que contrata el usuario.
- caducidad: fecha de caducidad del servicio. Se especifica en esta tabla debido a que la caducidad de la cuota anual depende de la fecha en la que se contrate y no de la del servicio en sí.
- pagado: indica si el pago está confirmado o no, es decir, que el administrador ha confirmado en el banco que se ha hecho la transferencia por el valor del servicio/actividad.

Estadísticas (Tabla pr_estadisticas):

Esta entidad representa las estadísticas deportivas de cada usuario. Cuando el usuario se registra en la aplicación, se crea un registro de estadísticas en base de datos que podrá ser actualizado por el propio usuario.

#	Nombre	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado	Extra
1	id_user	int(11)			No	0	
2	goles	int(11)			Sí	*NULL*	
3	faltas	int(11)			Sí	*NULL*	
4	partidos_jugados	int(11)			Sí	*NULL*	
5	minutos	int(11)			Sí	*NULL*	
6	red_cards	int(11)			Sí	*NULL*	
7	yellow_cards	int(11)			Sí	*NULL*	

Ilustración 25: Tabla de Estadisticas

- id_user: clave primaria de la tabla y a su vez clave foránea a la tabla de usuarios. Con esto nos aseguramos de que sólo se cree un registro en base de datos por usuario de la aplicación.
- goles: goles anotados por el usuario.
- faltas: faltas cometidas por el usuario.
- partidos_jugados: partidos disputados por el usuario.
- minutos: minutos disputados por el usuario.
- red_cards: tarjetas rojas recibidas.
- yellow_cards: tarjetas amarillas recibidas.

7.2.2. Modelo Entidad – Relación

En este apartado estudiaremos la base de datos sobre el que las aplicaciones están montadas. En esta base de datos se encuentran los datos de usuarios necesarios para la identificación así como de estadísticas, servicios, perfiles y contenidos para poder hacer gestiones sobre ellos.

Empezaremos con el esquema entidad relación de dicha base de datos y acabaremos con el esquema conceptual modificado.

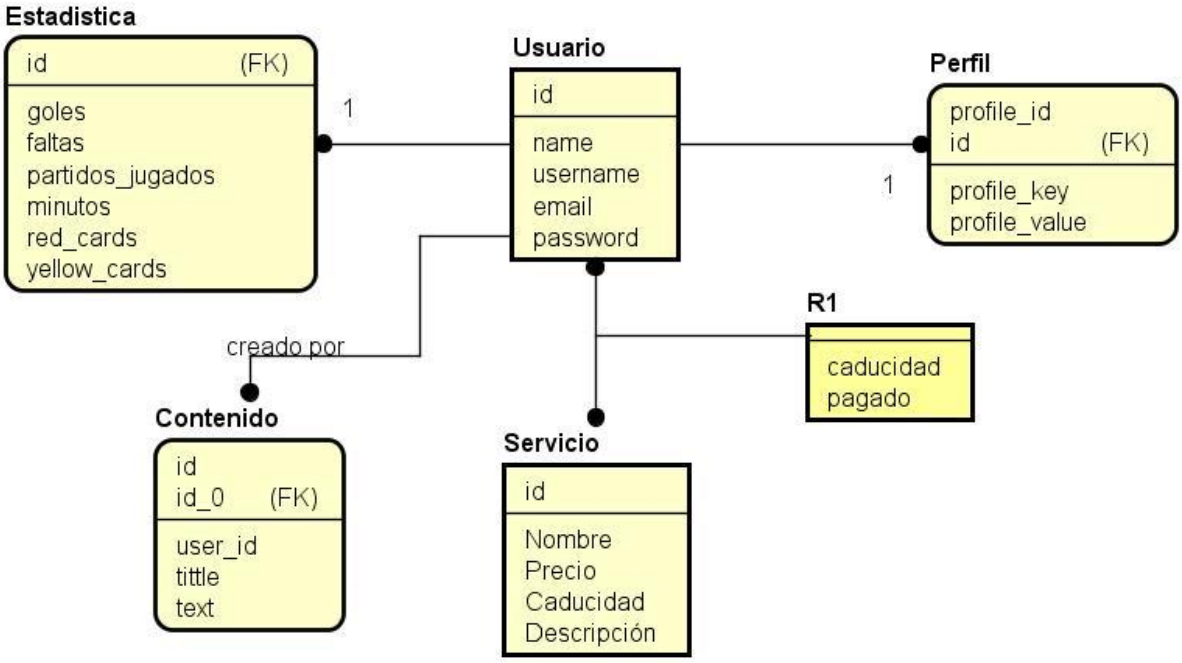


Ilustración 26: Modelo Entidad-Relación

Necesitamos convertir nuestros elementos de información en entidades o relaciones. En nuestro caso, Pago pasará a convertirse en entidad de nuestro esquema conceptual, y R1 se convierte en relaciones que unen la entidad Usuario con Servicio. Así, nuestro modelo E-R se puede ver en la ilustración 26.

A continuación procedemos con esquema conceptual modificado.

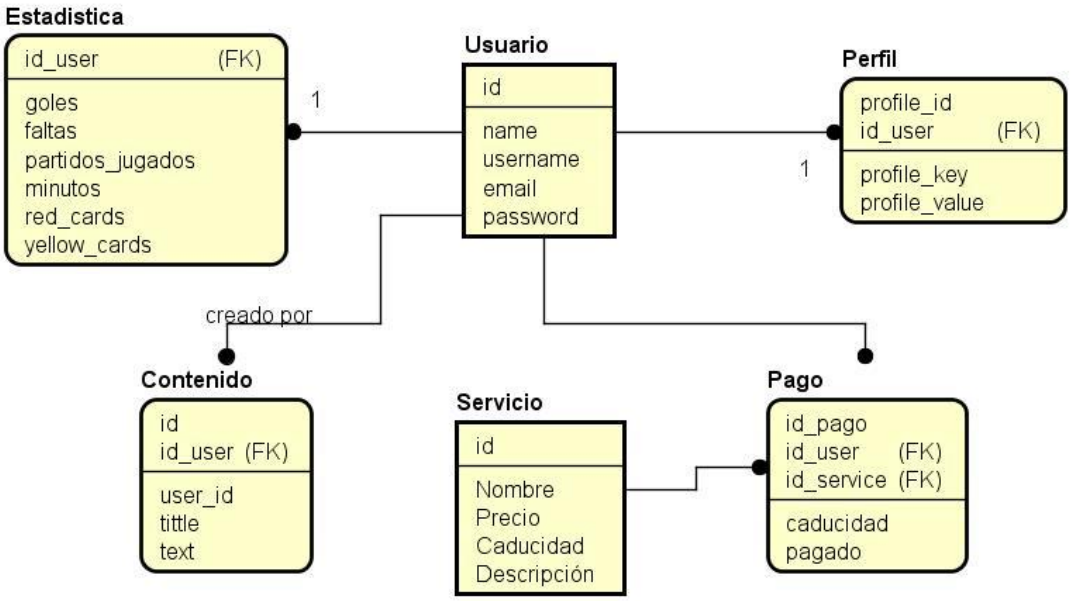


Ilustración 27: Modelo Entidad-Relación Modificado

Para la obtención del Modelo Entidad - Relación Modificado (véase Ilustración 27) a partir del Modelo Entidad-Relación (véase Ilustración 26) se deben hacer los cambios que enunciamos a continuación:

- Eliminar todas las entidades débiles.
- Eliminar las relaciones de muchos a muchos.
- Eliminar las relaciones con atributos existentes en el Esquema Conceptual.

Realizando las modificaciones oportunas, obtendremos una nueva entidad: Pago derivada de la relación muchos a muchos existente en el Modelo E-R. Nuestro Modelo E-R Modificado quedará como muestra la ilustración 27.

7.3. Casos de uso

En ingeniería del software, un caso de uso es una técnica para la captura de requisitos potenciales de un nuevo sistema o una actualización de software. Cada caso de uso proporciona uno o más escenarios que indican cómo debería interactuar el sistema con el usuario o con otro sistema para conseguir un objetivo específico.

Se realizarán tres casos de uso en los que se agrupan las diferentes funcionalidades que proporciona el sistema.

El primer diagrama es el diagrama frontera. Un diagrama frontera es aquel en que se muestran los límites del sistema ofreciendo una visión general de este, y de como este se relaciona con los actores y otros sistemas:

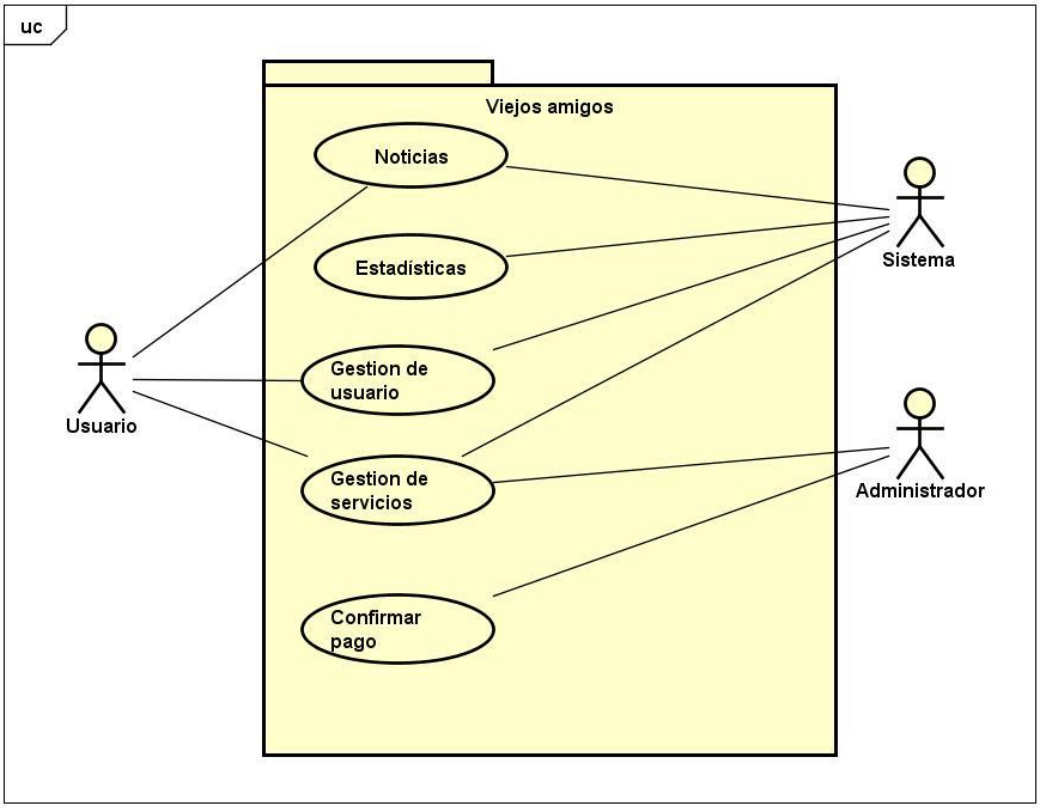


Ilustración 28: Diagrama frontera

Se puede observar en este diagrama concreto en que partes se divide el sistema y cuáles de ellas se relacionan a su vez con usuarios y otros sistemas.

Explicación: En este diagrama intervinenen tres actores: el sistema (entendemos sistema a la propia lógica de la aplicación), el usuario (hace referencia a un usuario registrado en la aplicación) y un administrador (administrador de la aplicación).

Las tareas llevadas a cabo por el usuario son consultar las noticias, gestionar sus propias estadísticas, gestionar su propio perfil y gestionar sus propios servicios/actividades).

El actor Sistema (la propia aplicación) interactúa con todas las tareas, pero se obvian las líneas de aquellas tareas que carecen de lógica, por ejemplo, las estadísticas son propias de los usuarios, demandando sólo al sistema la escritura/lectura de base de datos.

El actor Administrador también es considerado un usuario, pero se obvian las líneas a todas las demás tareas debido a que su principal labor es la gestión de los pagos y de los servicios.

En el siguiente caso de uso se desglosa la tarea “Gestión de usuarios” y se plasma un ejemplo (vease Tabla 4):

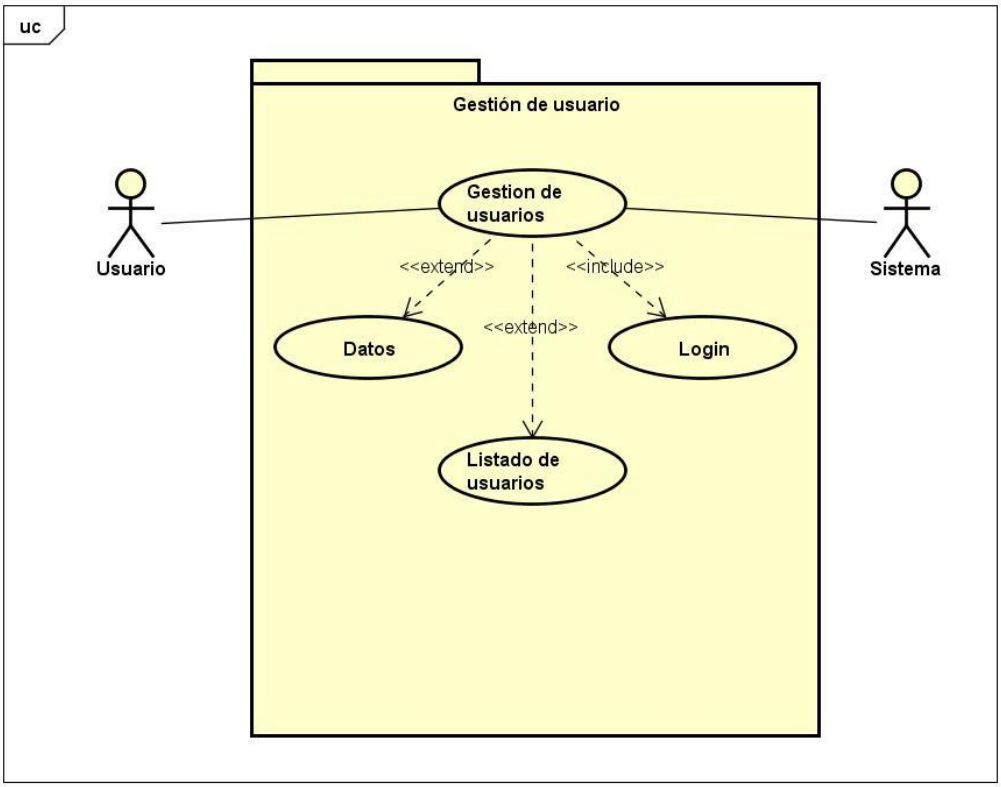


Ilustración 29: Caso de uso Gestión de usuario

Gestión Usuario		
Dependencias	• Entrada al sistema	
Precondición	-El usuario ha accedido al sistema -Conexión con el servidor pre establecida	
Descripción	El sistema debe proporcionar todos los servicios descritos en el caso de uso	
Secuencia normal	Paso	Acción
	1	El usuario inicia sesión en la web
	2	El servidor devuelve sus credenciales
	3	El usuario tiene acceso a los servicios que ofrece la aplicación.
Postcondición	Se almacena la sesión del usuario.	
Excepciones	Paso	Acción
	2	El usuario no puede logarse en el sistema. Es posible recuperar la contraseña o registrarse en la aplicación
Comentarios	Se podrán acceder a las partes de la web dependiendo del rol del usuario logado.	

Tabla 3: Caso de uso Gestión Usuario

En el siguiente caso de uso se desglosa la tarea “Gestión de servicios”. En la Tabla 5 se dará un ejemplo de caso de uso:

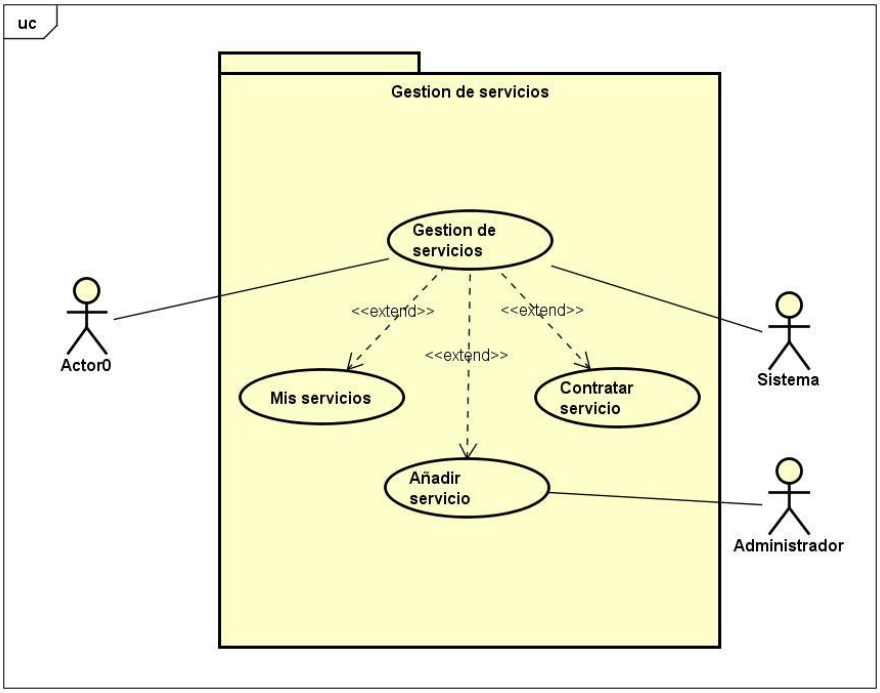


Ilustración 30: Caso de uso Gestión servicios

Gestión Servicios		
Dependencias	• Entrada al sistema	
Precondición	-Conexión con el servidor pre establecida -El usuario ha accedido al sistema	
Descripción	El sistema debe proporcionar todos los servicios descritos en el caso de uso	
Secuencia normal	Paso	Acción
	1	El usuario consulta sus servicios
	2	Sistema hace una consulta a la BBDD y devuelve una lista de servicios.
Postcondición	Se muestra una lista de usuarios	
Excepciones	Paso	Acción
	2	No se puede establecer conexión con la BBDD
Comentarios	El usuario consulta sus servicios contratados.	

Tabla 4: Caso de Gestión Servicios

En estos casos de uso analizamos la relación existente entre los componentes software de la aplicación. Con las tablas podemos deducirla secuencia

de interacciones que se desarrollarán entre el sistema y el actor en respuesta a un evento que inicia el actor principal sobre el propio sistema. Se podrían incluir más casos de uso, pero con estos dos podemos deducir el funcionamiento del sistema.

7.4. Storyboard

Un Storyboard es un conjunto imágenes mostradas en secuencia, con el fin de previzualizar una animación o cualquier otro medio gráfico o interactivo.

Basicamente el stoyboard es un guion gráfico, que nos permite la previsualización de nuestra aplicación antes de que esté terminada.

A continuación se plasmarán los storyboards tanto de la aplicación web como de la aplicación móvil, y daremos por finalizado el capítulo de diseño.

7.4.1. Storyboard de la aplicación web

El prototipo a desarrollar para la aplicación web estará basado en cómo trabaja Joomla. Las diferentes funcionalidades se insertarán en la parte central de la pantalla, a partir de ahora contenedor principal.

Por tanto, el sitio web tendrá el siguiente aspecto:

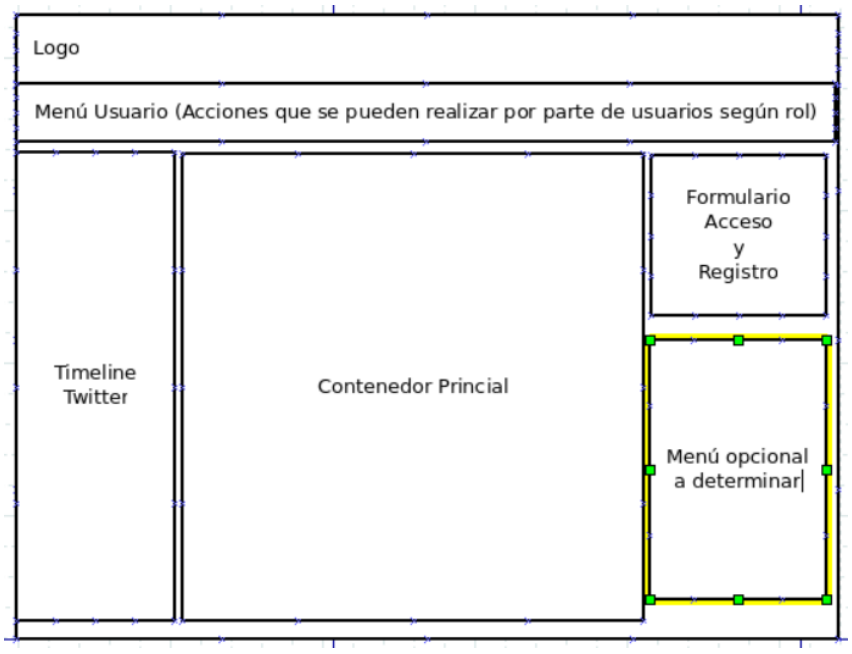


Ilustración 31: Storyboard Web Principal

A partir de este punto, se desarrollarán las diferentes funcionalidades que se incrustarían en el contenedor principal y serán accesibles desde el menú de usuario (Véase Ilustración 31):

- Contratar Servicio:

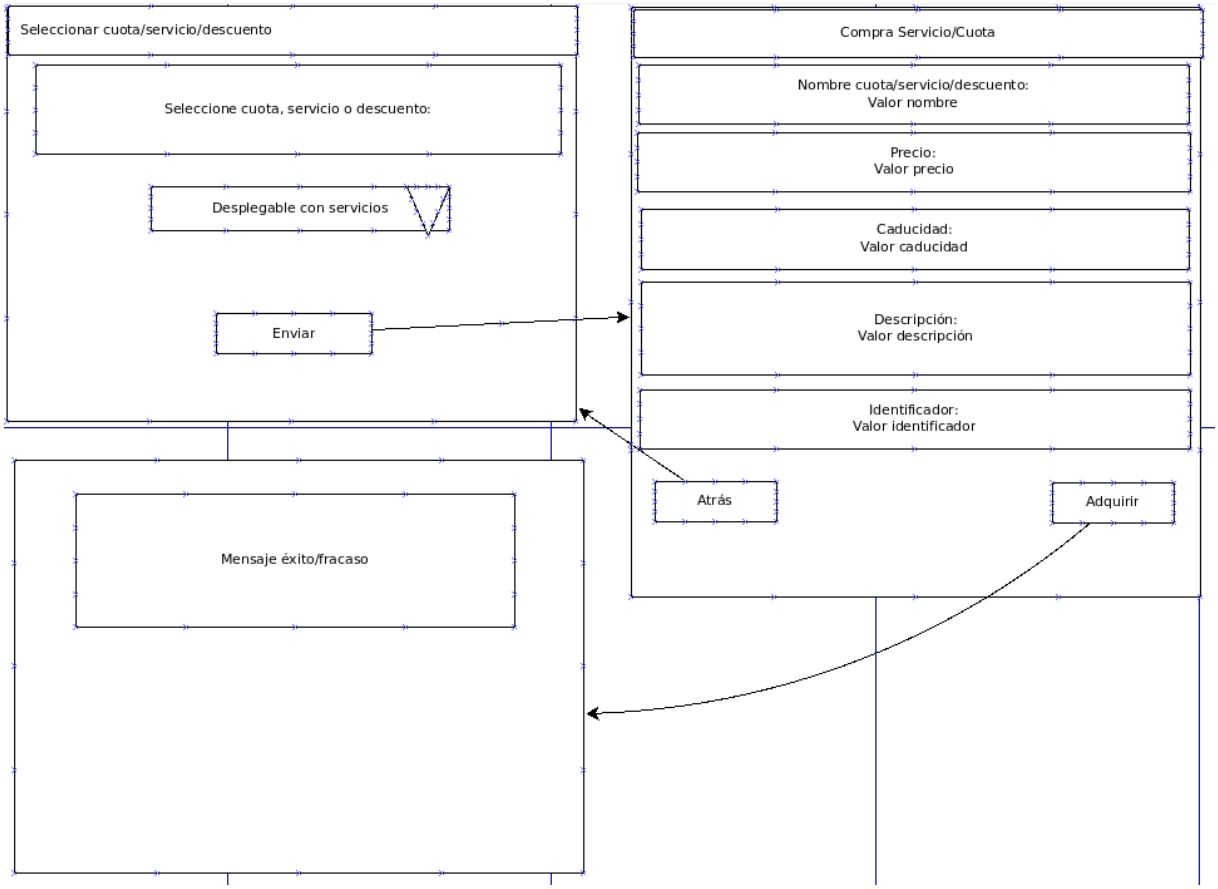


Ilustración 32: Storyboard Web. Contratar servicio

- Estadísticas (aplicable a “Añadir servicio” con sus propios campos):

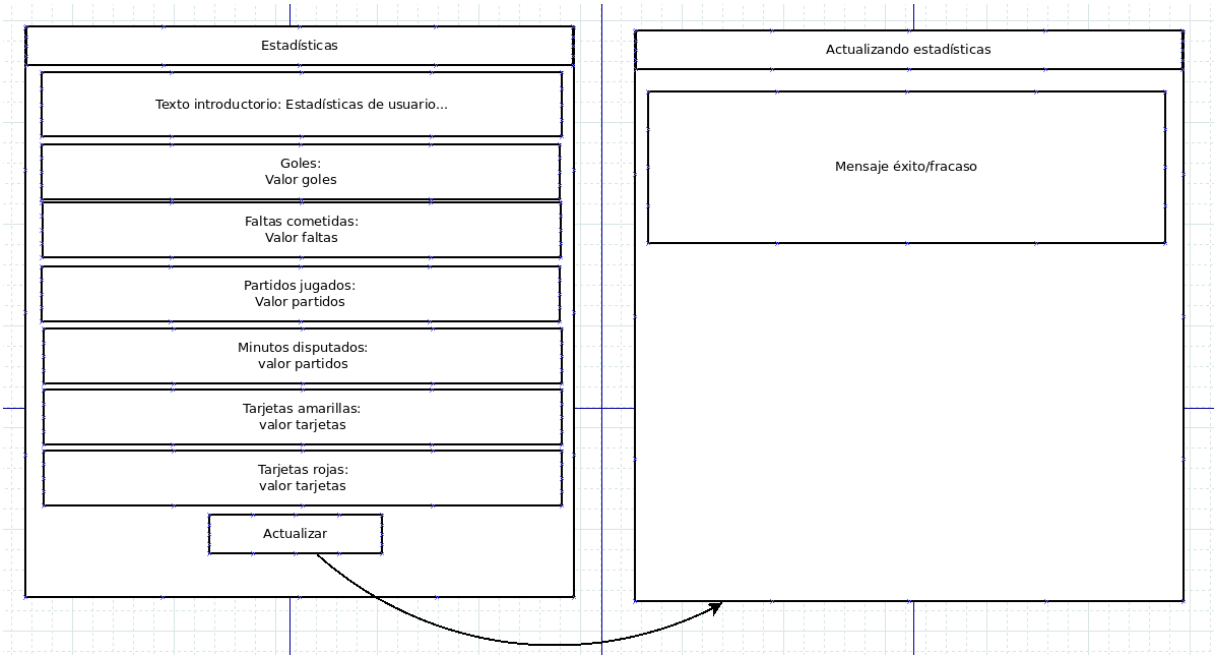


Ilustración 33: Storyboard Web. Estadísticas

- Confirmar servicio:

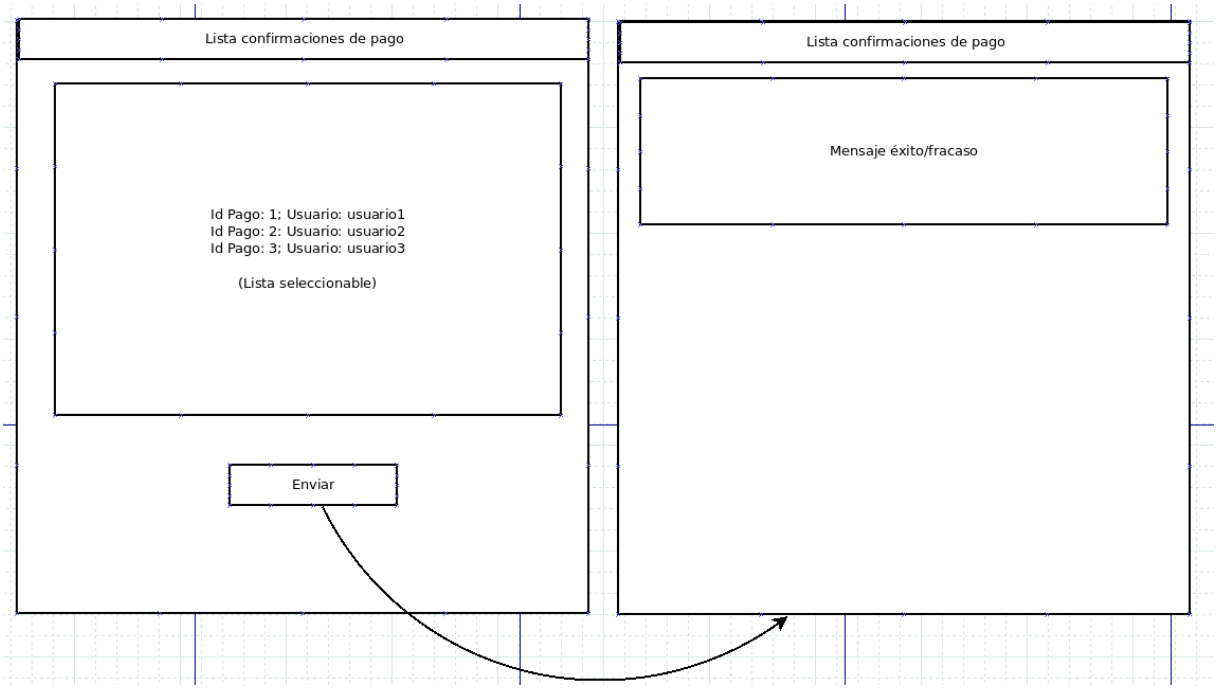


Ilustración 34: Storyboard web. Confirmar pago

- Mis servicios:

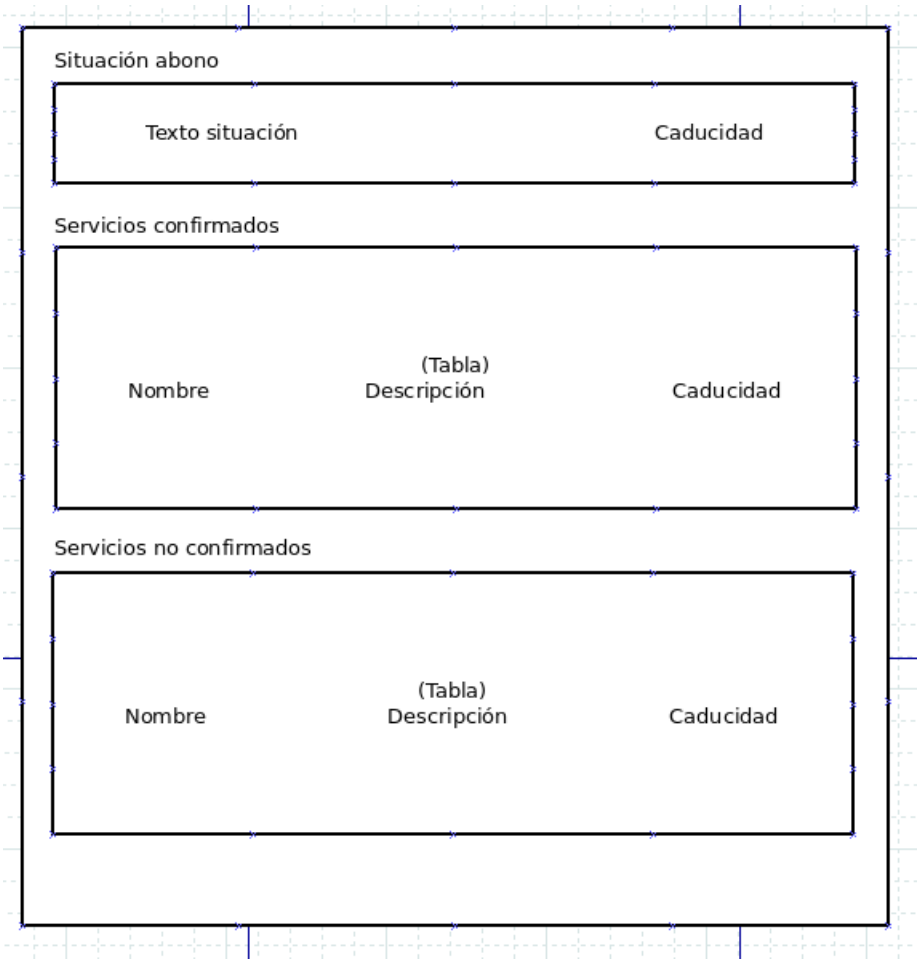


Ilustración 35: Storyboard web. Mis servicios

Para la construcción de estos storyboards se han seguido las siguientes pautas:

- Sencillez de los prototipos. Es simplemente una guía.
- Se han obviado prototipos similares entre pantallas.
- A lo largo de la implementación, por restricciones de diseño, los prototipos pueden cambiar.
- Se ha seguido un diseño lo más usable posible.

7.4.2. Stroryboard aplicación móvil

El storyboard de la aplicación móvil (Android) sería el siguiente:

- Login:

Ilustración 36: Storyboard móvil. Login

- Menú de usuario:

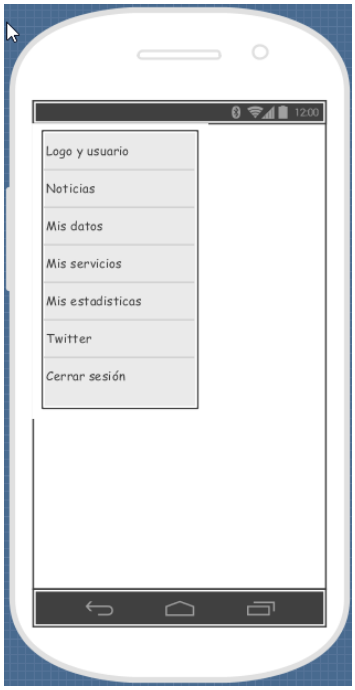


Ilustración 37: Storyboard móvil. Menú usuario

- Noticias (Aplicable a Mis servicios):

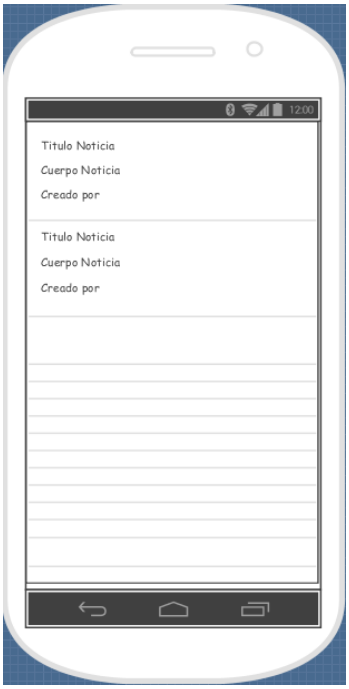


Ilustración 38: Storyboard móvil. Noticias

- Mis estadísticas (aplicable a Mis datos):

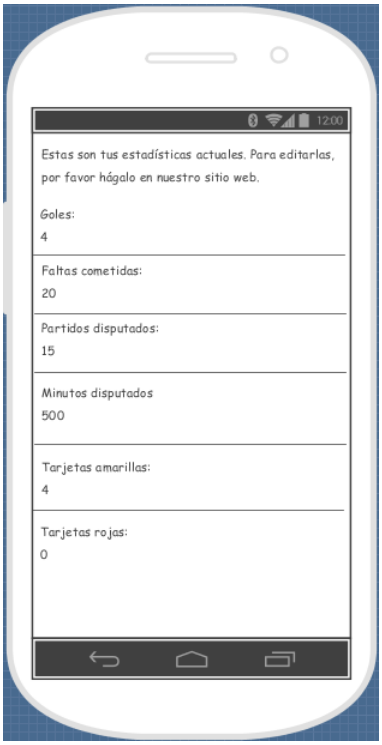


Ilustración 39: Storyboard móvil. Mis estadísticas

Se han tenido en cuenta los siguientes aspectos a la hora del desarrollo de este storyboard:

- Vistas sencillas e intuitivas.
- Desarrollo rápido de los prototipos. Estos prototipos se tomarán como guía inicial.
- Las pantallas pueden cambiar a lo largo de la implementación por restricciones de la tecnología.
- A partir de la pantalla de login, todas las demás vistas son accesibles desde el menú de usuario desde cualquier vista de la aplicación.
- Se han obviado las pantallas cuyo aspecto es similar a otras en las que ya se ha definido su prototipo.

8. IMPLEMENTACIÓN

8.1. Introducción

La implementación es la última parte del ciclo del prototipado en la cual se aplican las fases anteriores de análisis y diseño para obtener algo tangible, construyendo las aplicaciones basándonos en las etapas anteriores.

Este capítulo se dividirá en dos grandes bloques, implementación de la aplicación web e implementación de la aplicación móvil.

8.2. Implementación de la aplicación web

Como se ha comentado en capítulos anteriores, la aplicación web se contruirá a partir del sistema de gestión de contenidos Joomla!. Esta herramienta facilita mucho la etapa de desarrollo, ya que la mayoría de la funcionalidad típica de toda aplicación cómo puede ser el login, el registro, la publicación y gestión de artículos, incluso el diseño principal de los estilos de la aplicación son transparentes para el programador.

En este capítulo nos centraremos en describir los diferentes componentes, plugins y plantillas que se han utilizado para la consecución del proyecto.

8.2.1. Módulos, plugins y extensiones

A continuación se detallan los principales módulos y plugins instalados en nuestra aplicación que intervienen de manera directa con la lógica de ésta, o bien, ayudan a mantener la aplicación web en el tiempo:

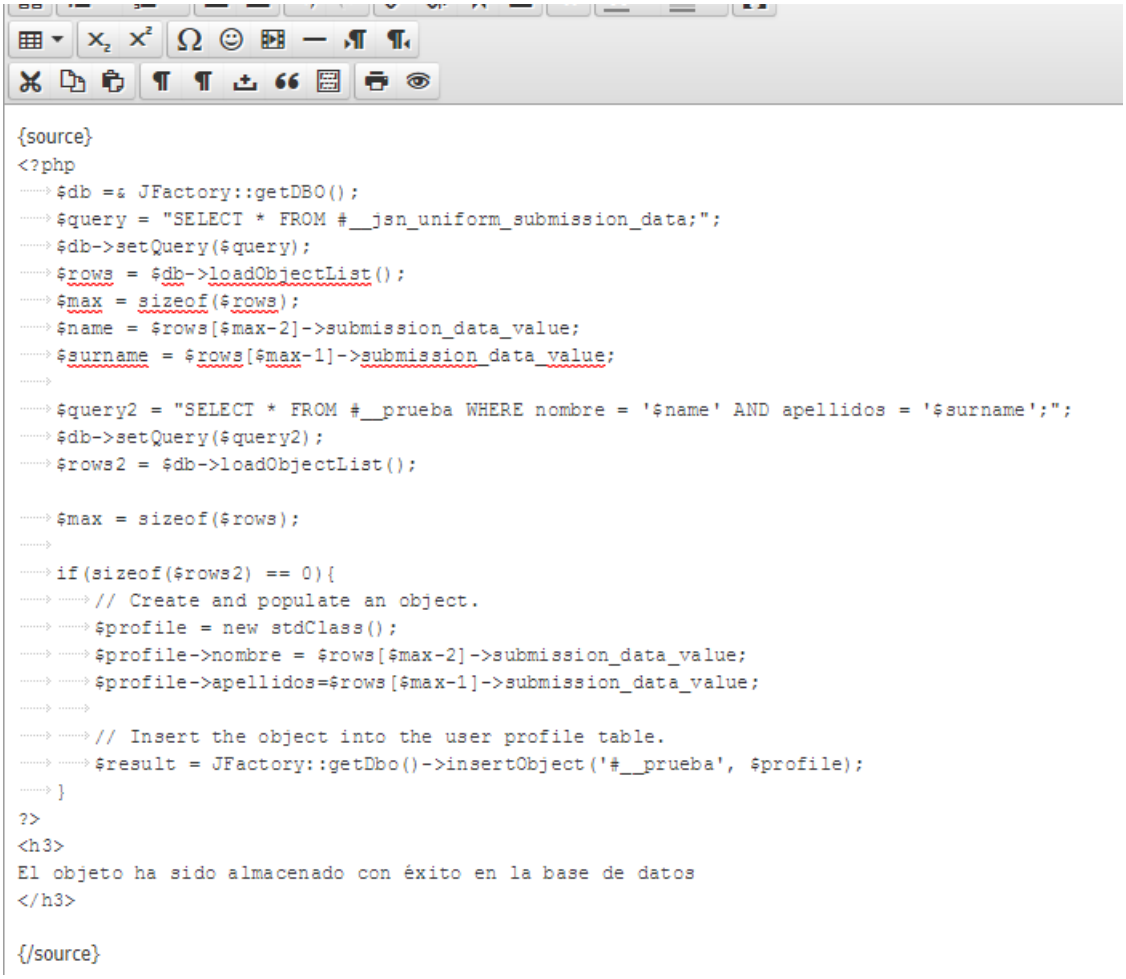
- **Sourcerer**: Podemos decir que esta es una de las extensiones más importantes en este proyecto. Sourcerer es una extensión que permite inyectar código HTML, CSS, JavaScript y PHP en los artículos de Joomla!.

  La mayor parte de los formularios que se muestran al usuario en la

apllicación web han sido creados con esta extensión, y es tan simple como publicar un artículo con el código de los formularios, enlazarlos a un enlace o elemento de menú y publicarlo.

Para utilizar esta extensión, basta con encerrar el código deseado entre las etiquetas {source}{/source}. Cuando el usuario carga un artículo en la aplicación, el motor de Joomla interpreta que el código encerrado entre esas etiquetas no es texto plano, proporcionando el comportamiento deseado.

A continuación se muestra un ejemplo del código de un formulario inyectado en Joomla con la extensión Sourcerer:



```
{source}
<?php
    $db =& JFactory::getDBO();
    $query = "SELECT * FROM #__jsn_uniform_submission_data;";
    $db->setQuery($query);
    $rows = $db->loadObjectList();
    $max = sizeof($rows);
    $name = $rows[$max-2]->submission_data_value;
    $surname = $rows[$max-1]->submission_data_value;

    $query2 = "SELECT * FROM #__prueba WHERE nombre = '$name' AND apellidos = '$surname';";
    $db->setQuery($query2);
    $rows2 = $db->loadObjectList();

    $max = sizeof($rows);

    if(sizeof($rows2) == 0){
        // Create and populate an object.
        $profile = new stdClass();
        $profile->nombre = $rows[$max-2]->submission_data_value;
        $profile->apellidos=$rows[$max-1]->submission_data_value;

        // Insert the object into the user profile table.
        $result = JFactory::getDbo()->insertObject('#__prueba', $profile);
    }
?>
<h3>
El objeto ha sido almacenado con éxito en la base de datos
</h3>

{/source}
```

Ilustración 40: Inyección de código con Sourcerer

- **jBackend Community**: Esta extensión también cobra una gran importancia en el desarrollo de este trabajo. jBackend Community proporciona endpoints predefinidos en nuestra aplicación cuando es instalado en el motor de Joomla. Esto quiere decir que crea una capa capaz de recibir peticiones REST y devolver una respuesta en formato JSON.

  Posee fundamentalmente dos endpoints o módulos:

  - API/Módulo de usuario: devuelven datos acerca de los usuarios de Joomla.
  - API/Módulo de contenido: devuelven datos acerca del contenido publicado en Joomla.

  La potencia de este plugin radica en que gestiona de manera eficiente las peticiones, y libera al programador del desarrollo de una compleja API Rest para el control de este tipo de datos.

  Véase **Anexo V** con las pruebas de la API Rest.

  Por dar un ejemplo, si se quiere autenticar a un usuario desde una aplicación externa se lanzaría este tipo de petición:

Example

```
<end-point>?action=post&module=user&resource=login&username=<username>&password=<password>
```

Example (REST format)

```
<end-point>/post/user/login?username=<username>&password=<password>
```

Response

```
{
    "status": "ok",
    "userid": <userid>,
    "username": "<username>"
}
```

Ilustración 41: Ejemplo Login con JBackend Community

- **Akeeba Backup**: Este componente nos permite realizar una copia de seguridad de nuestro sitio Joomla! muy fácilmente. Es muy importante realizar siempre una copia de seguridad antes de realizar cualquier cambio brusco en el sitio, para evitar posibles problemas o pérdidas de información ocasionadas al instalar extensiones dañadas. Podemos decir que es uno de los componentes más importantes a la hora de mantener nuestro sistema ante errores potenciales.

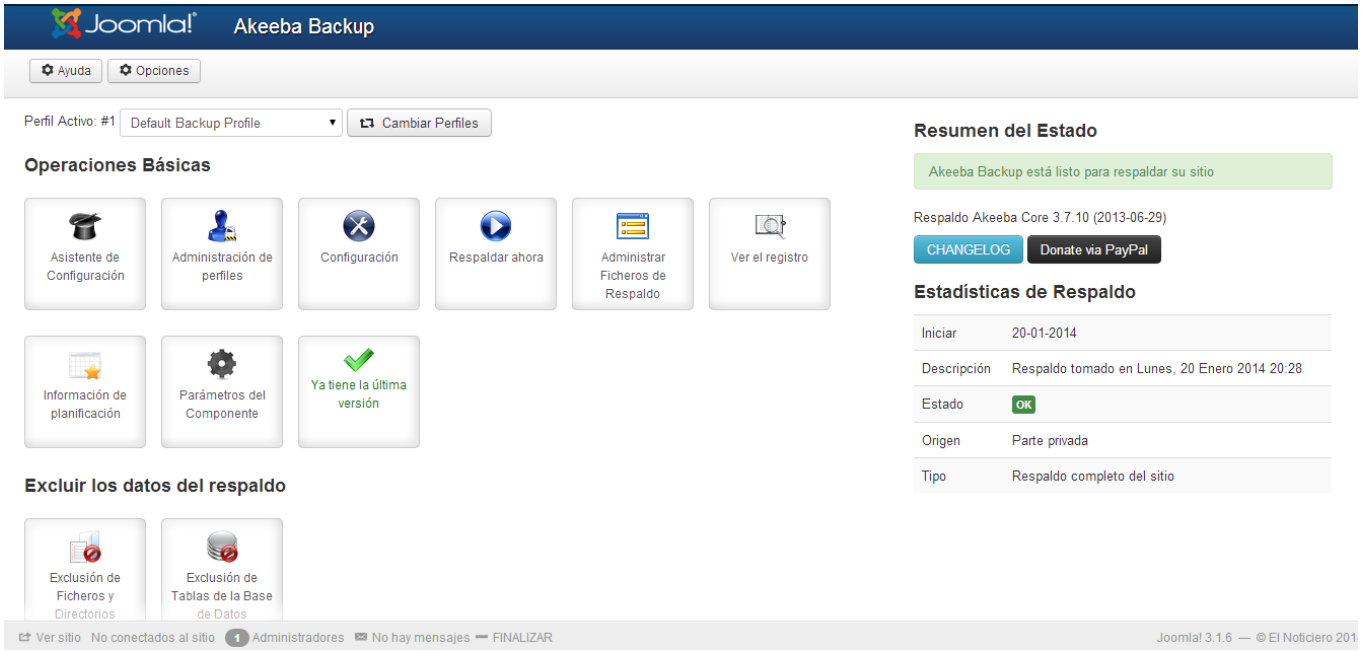


Ilustración 42: Panel de Akeeba Backup

- **BreezingForms**: BreezingForms es una extensión para Joomla! que te permite crear formularios de una manera fácil, rápida y eficaz. Posee dos versiones, una de pago (muy completa) y otra gratis. En la realización de este trabajo se ha usado la versión gratis para crear el formulario de “Añadir Servicio”, obteniendo muy buenos resultados, pudiendo incluso añadir un Captcha para confirmar el formulario. El único problema de la versión gratuita es que sólo permite introducir campos de texto, limitando así las opciones, por lo cuál se han programado los demás formulario con código (Sourcerer).

- **Otros componentes, plugins y extensiones**: llegados a este punto, aclarar que se han utilizado otros componentes propios de Joomla para el desarrollo de este proyecto. Podemos llegar a la conclusión de que estos componentes promueven un desarrollo muy ágil, ya que están optimizados y respaldados por miles de usuarios que los mantienen.

8.2.2. Plantillas

Las plantillas constituyen la forma más rápida de cambiar la apariencia de un sitio web ya que juegan con el contenido HTML, CSS o JavaScript del sitio.

Para este trabajo hemos optado por utilizar la plantilla “protostar”, la plantilla estrella de Joomla!, ya que otorga la suficiente variedad de colocación de módulos.

Existen infinidades de plantillas en la web, con las cuáles se puede cambiar el aspecto de la aplicación sin necesidad de perjudicar el contenido. Otro gran punto a favor de Joomla.

8.2.3. La clase JFactory

En este punto se detalla el funcionamiento de la clase JFactory proporcionada por Joomla, debido a que ha sido muy utilizada en el desarrollo del proyecto, sobre todo a la hora de acceder a la base de datos.

La clase JFactory proporciona el acceso a las partes más importantes de Joomla (la mayoría son singleton). Estos objetos contienen a su vez múltiples objetos, como la aplicación en sí o la configuración global de Joomla, pudiendo editar estas partes desde código.

Para dar un ejemplo de la potencia de esta clase, vemos lo sencillo que es ejecutar una query de base de datos:

```
$db =& JFactory::getDBO();
$query = "SELECT * FROM #__servicios;";
$db->setQuery($query);
$rows = $db->loadObjectList();
```

Ilustración 43: Consulta con JFactory

Aquí podemos ver cómo con sólo cuatro líneas de código obtenemos todos los servicios de base de datos, sin la necesidad de la costosa configuración requerida sin este tipo de clases en PHP.

8.2.4. Software utilizado

Para el desarrollo de la aplicación web se ha utilizado el siguiente software:

- XAMPP: Necesario para disponer de un servidor web y de base de datos.
- Google Chrome: Una vez levantado el servidor XAMPP, podemos acceder al backend de Joomla para empezar a editar y gestionar usuarios, contenido, etc.

8.3. Implementación de la aplicación móvil

En este apartado se desarrollarán los principales puntos que den al lector una idea clara de como se ha desarrollado la aplicación móvil.

8.3.1. Arquitectura del cliente Android

A continuación se dará una visión de la estructura en la que se ha organizado la aplicación Android, explicando el porqué de esta estructura y detallando el acoplamiento entre las diferentes clases:

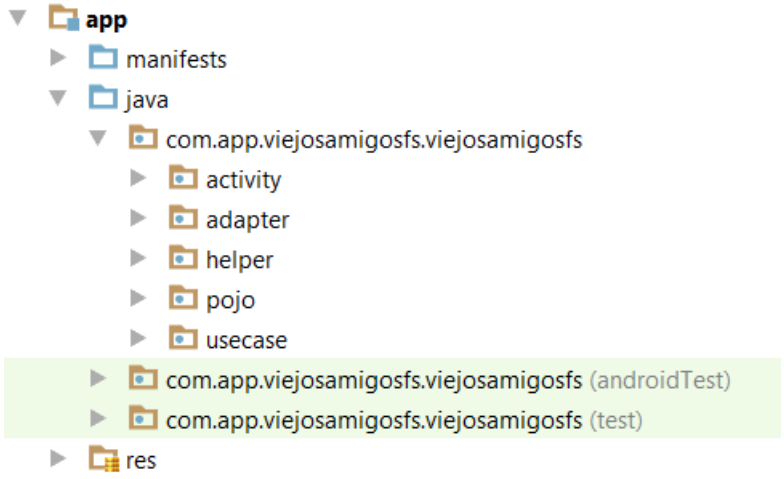


Ilustración 44: Estructura Android

A continuación se explicará el contenido de cada uno de los paquetes:

activity: Este paquete contiene las actividades de la aplicación. Una actividad es un componente básico que permite crear componentes que interaccionan con el usuario. Cada actividad tiene asociada una vista. Esta vista constituye la interfaz de usuario, es decir, las distintas pantallas por las que el usuario navega dentro de la aplicación.

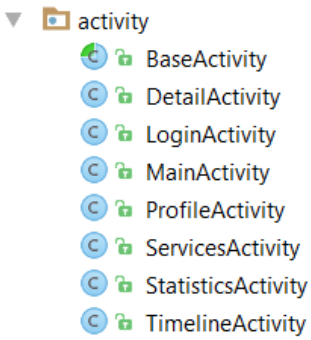


Ilustración 45: Paquete activity

adapter: Este paquete contiene las clases java que actúan de adaptadores. Un adaptador es una clase que ofrece la oportunidad de adaptar el diseño de un ítem determinado de una lista a todo el conjunto de esa lista.

Ilustración 46: Paquete adapter

helper: Este paquete contiene clases que facilitan la lógica de otras clases. En este caso contiene la clase *PreferencesHelper.java* construida con el patrón Singleton para que sólo haya una instancia creada en la vida de la aplicación. Contiene un atributo de tipo SharedPreferences, que es dónde se almacena el identificador y username del usuario logado. Aunque la aplicación no se ejecute, estos dos atributos siguen almacenados en la aplicación. Esto tiene dos motivos principales: almacenar el login del usuario, y poder utilizar dichos atributos en cualquier parte de la aplicación para poder realizar las peticiones REST. Si el usuario cierra sesión, se “limpia” el SharedPreferences.

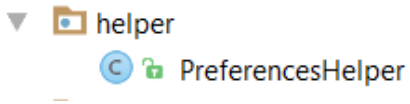


Ilustración 47: Paquete helper

pojo: Este paquete contiene las clases que hacen referencia a los objetos simples que se transfieren entre cliente y servidor. Contienen atributos, constructor por defecto, métodos getters y setters y opcionalmente constructores parametrizados.

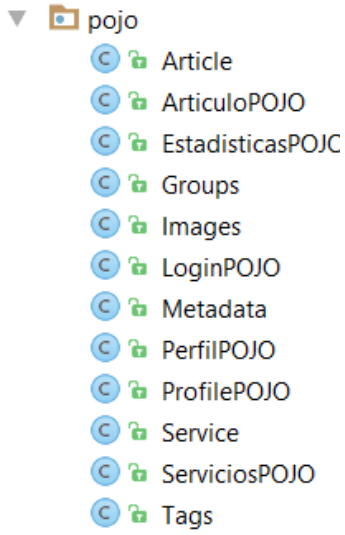


Ilustración 48: paquete pojo

usecase: paquete que contiene las clases que hacen las llamadas a los servicios REST. Todas estas clases heredan de AsyncTask y utilizan la librería OkHTTP para realizar las peticiones.

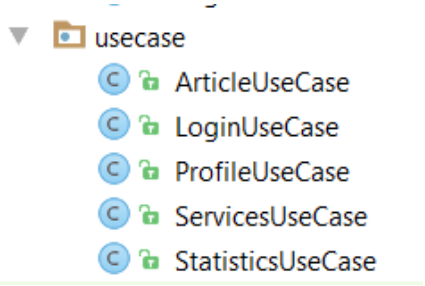


Ilustración 49: Paquete usecase

Una vez identificados los diferentes componentes que definen nuestra arquitectura se explicará como se acomplan entre ellos:

- Las actividades que realizan una petición al servicio REST implementan su Use Case asociado. Esto quiere decir que deberán implementar los métodos onSuccess() y onError() para tratar los datos enviados por el servidor.
- Las respuestas recibidas son en formato String JSON, por lo tanto, por la forma en la que están diseñados los POJOs es muy simple hacer una conversión JSON – Objeto con la librería GSON.
- Las clases que actúan como adaptadores son implementadas en aquellas actividades en las que definen componentes de tipo lista.
- Cobra importancia la clase BaseActivity. Esta clase define el comportamiento del menú de usuario que se define en todas las pantallas (excluyendo el Login). Por tanto las actividades dónde aparece el menú heredan de BaseActivity, aprovechando así la herencia de clases para evitar la reescritura de código.
- El funcionamiento de la aplicación, básicamente, es el de realizar peticiones contra el servidor web, parsear los datos para hacerlos manejables en el contexto Java, y setear el contenido de los diferentes componentes que conforman las interfaces con estos datos.

8.3.2. Librerías y herramientas utilizadas

En este apartado se describirán las principales librerías y herramientas utilizadas en el desarrollo de la aplicación Android:

- OkHTTP: potente librería para realizar peticiones (GET, POST, PUT, DELETE) contra un servicio REST. La gran utilidad de esta librería es que inhibe al desarrollador de programar “a mano” cada una de las tareas asíncronas para la realización de una petición.

- GSON: librería de Google que posibilita la conversión de JSON a objetos y viceversa con un par de líneas de código. La única restricción es que la estructura de la clase ha de estar bien definido para su correcto funcionamiento.
- JSOUP: librería que transforma un texto en formato HTML en texto plano. Debido a que el contenido de los artículos en el servidor contiene html, es necesario "limpiarlo" para una correcta presentación de los datos.
- API Twitter: proporciona los componentes, librerías y autenticaciones necesarias para poder acceder a los datos de Twitter. En este caso se ha usado para mostrar el timeline de la cuenta de twitter de la peña.
- jsonschema2pojo: herramienta online que a partir de un JSON construye los POJOS (clases Java) de manera automática. Es muy configurable.

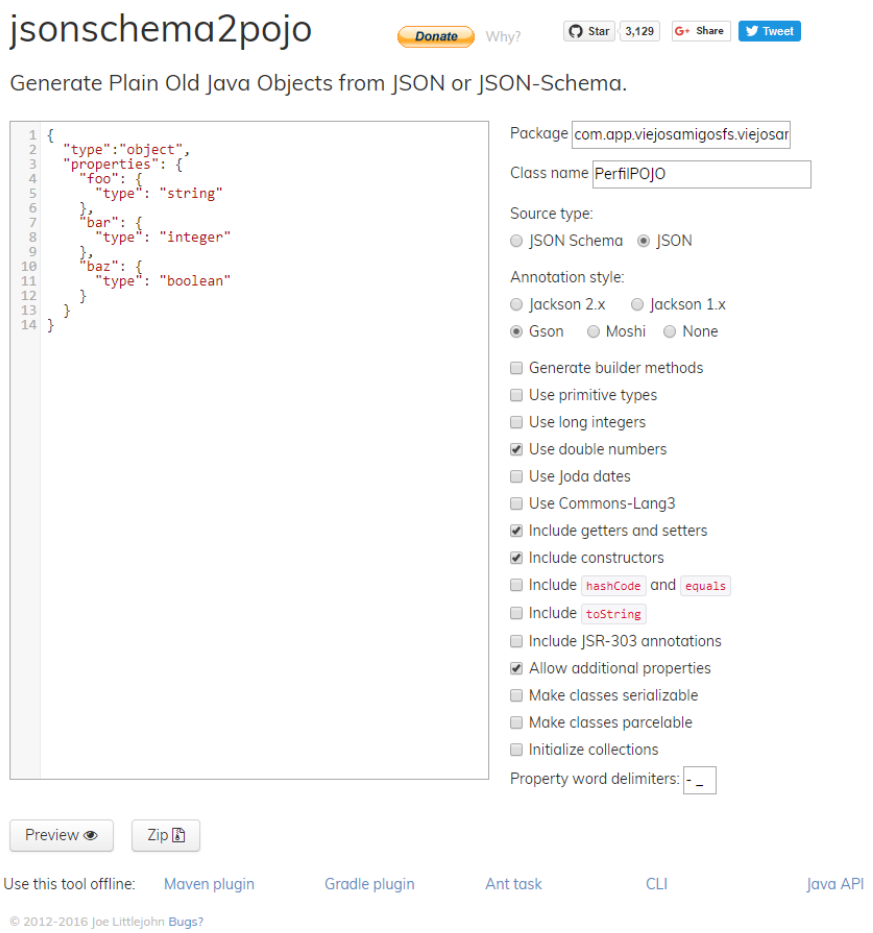


Ilustración 50: jsonschema2pojo

8.3.3. Software utilizado

Para el desarrollo de la aplicación móvil se ha usado el IDE Android Studio (Véase ilustración 51).

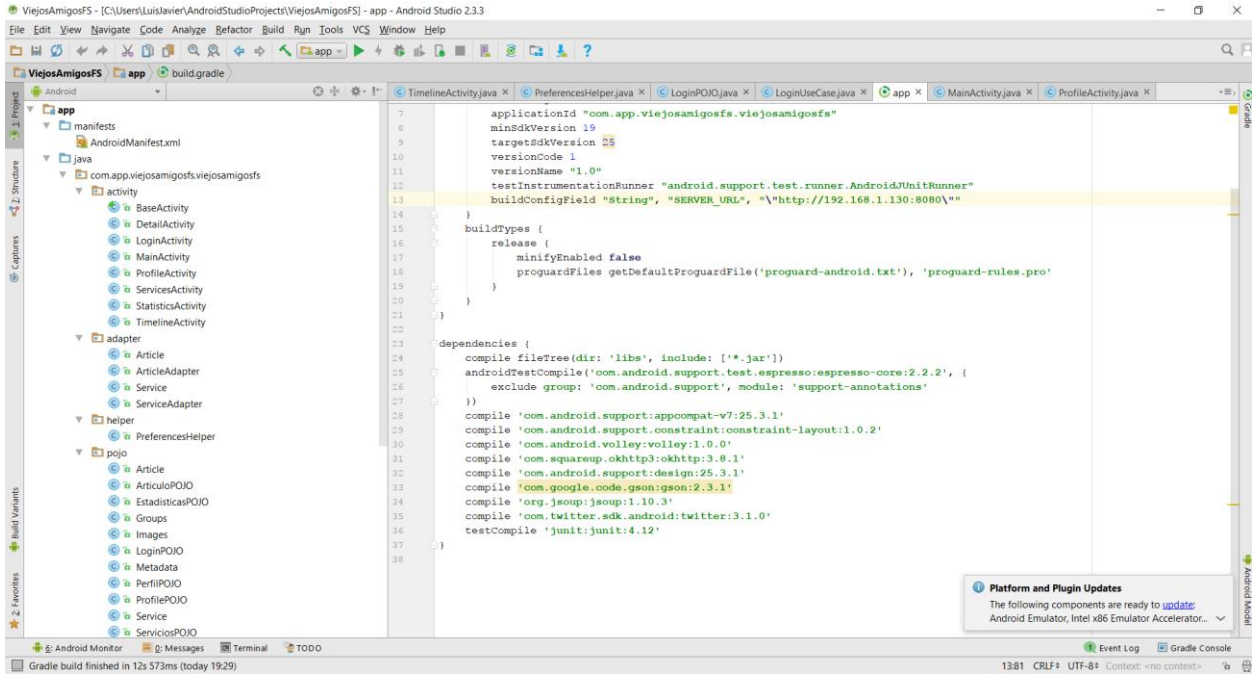


Ilustración 51: IDE AndroidStudio

8.4. Pruebas y validación

Una vez desarrollada ambas aplicaciones entramos en la fase de pruebas y validación de éstas. Es una de las etapas más importantes del desarrollo en la cuál nos podemos dar cuenta de errores potenciales y sobre todo es la fase en la que mejor comprobamos que cumplimos con los requerimientos establecidos en la fase de análisis.

Para agilizar este punto, se han realizado dos videos explicativos donde se puede ver como las apliaciones cumplen con los requermientos establecidos.

Los enlaces a los videos son los siguientes:

- Aplicación web:
  - https://www.youtube.com/watch?v=wzv-yTaL6-c
- Aplicación móvil:
  - https://www.youtube.com/watch?v=F0xAY6I83G0

Entre los errores más puntuales que podemos encontrarnos es una caída del servidor que dejará nuestro sitio web inaccesible, y cuya solución sería volver a arrancar los servicios necesarios.

Si el servidor está caído, la aplicación móvil no podrá cargar el contenido debido a que el servidor es incapaz de responder, devolviendo respuesta con el error TIMEDOUT. Este típico error está controlado en la aplicación Android de modo que si la respuesta recibida no es un OK informe al usuario de que hay un problema de conexión con el servidor.

En la ilustración 52 podemos ver el típico mensaje de error que aparece en la aplicación ante esta situación:

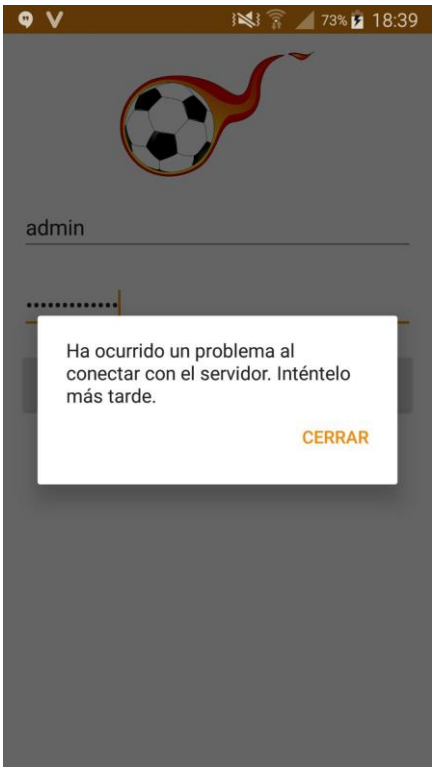


Ilustración 52: Error conexión Servidor

9. CONCLUSIÓN

A lo largo de todo este documento se han descrito las distintas fases por las que ha ido pasando este trabajo. Podemos afirmar que la tecnología ha avanzado de una manera considerable hasta entrar en todos los ámbitos de la sociedad.

Se puede apreciar cómo los sistemas de gestión pueden mejorar de una manera considerablemente buena la organización de una peña deportiva, que ha sido nuestro caso, pero esto es extensible a cualquier tipo de organización, incluso empresarial.

Me llevo muy buenas sensaciones con la realización de este proyecto, ya que he podido afianzar muchos conocimientos tanto teóricos como prácticos. He aprendido cómo funciona a un nivel medio el CMS Joomla y a su vez también he comprendido cómo están programadas las aplicaciones móviles que tan famosas se han vuelto hoy en día. Ambas tecnologías han supuesto un reto para mí, ya que no las había estudiado en profundidad.

También se han conseguido los objetivos redactados en la parte de análisis de requerimientos utilizando la técnica del prototipado, si bien es cierto que puede ser mejorado gracias a la escabilidad que se ha intentado ofrecer.

En cuanto a las dificultades para la realización de este proyecto, me gustaría destacar una en especial. Al principio de empezar con el desarrollo, pensé realizar la instalación del CMS en un hosting gratuito, para dar más realidad a la situación en la que nos queríamos poner.

Nada más instalar veo que apareceren anuncios publicitarios, debido a que es un hosting gratuito, aunque no le doy importancia. Empiezan los problemas. Un día entro en el sitio web para comenzar a desarrollar una nueva funcionalidad y veo que no se muestran los estilos, falta contenido, etc. Al parecer el hosting se había actualizado y se había migrado a una nueva versión de PHP no soportada por la versión de Joomla instalada. Tardé dos días enteros para averiguar el por qué del error, debido a la inexperiencia con estos servidores de hosting.

Tras la solución a este problema, a las pocas semanas, probando los servicios REST desarrollados, aprecio que las respuestas que me devuelve el servidor no son en formato JSON, sino que es un código html. Por lo visto, y después de leer en muchos foros aprecio que en hosting gratuito, se encriptan las respuestas. Esto supuestamente es una “técnica” para que los usuarios paguen la versión PRO.

Al final decidí exportar todo a un servidor XAMPP, y bueno, me quedo con los conocimientos que he aprendido al intentar tener mi propio hosting, tales como la familiarización con el cPanel, el uso de filezilla, y la migración de una aplicación de un servidor a otro, pasando por toda su configuración.

Llegados a este punto sería interesante exponer algunas posibles mejoras que a mi parecer podrían ser interesantes:

- Actualización de la versión de Joomla!.
- Añadir más funcionalidad a la aplicación móvil.
- Creación de un módulo de reserva de instalaciones deportivas.
- Creación de un módulo de ranking de estadísticas, con el objetivo de que las estadísticas de cada usuario sean visibles por las demás.
- Integración de otras redes sociales como Facebook o G+.

Por útlimo, para finalizar, me gustaría expresar la gran conformidad que me llevo terminando este trabajo. Ya voy a hacer dos años en el mundo laboral y me siento muy orgulloso de la preparación que tengo hoy en día, y todo gracias a la Universidad de Jaén.

La cuestión no es saberlo todo, si no tener los medios suficientes para saber donde buscar para saberlo todo.

Bibliografía

[1] "***La tecnología actual en nuestra sociedad***"

https://www.tribunasalamanca.com/noticias/la-tecnologia-actual-en-nuestra-sociedad/

[2] "***Internet de las cosas***"

https://es.wikipedia.org/wiki/Internet_de_las_cosas

[3] "***Modelo de prototipos***"

https://es.wikipedia.org/wiki/Modelo_de_prototipos

[4] "***Servicios Web***"

http://www.hipertexto.info/documentos/serv_web.htm

[5] "***UDDI (Universal Description, Discovery, and Integration)***"

https://www.ibm.com/support/knowledgecenter/es/SS4JE2_7.5.5/org.eclipse.jst.ws.consumption.ui.doc.user/concepts/cuddi.html

[6] "***Web Services***"

https://sistemas3.wordpress.com/2007/06/14/web-services/

[7] "***XAMPP***"

http://myu-charly.blogspot.com.es/

[8] "***Apache Server***"

http://webapache.blogspot.com.es/p/ventajas-y-desventajas.html

[9] "***MySQL***"

https://es.wikipedia.org/wiki/MySQL

[10] "***Bringing MySQL to the web***"

https://www.phpmyadmin.net/

[11] "***FileZilla***"

https://es.wikipedia.org/wiki/FileZilla

[12] "***PHP***"

https://es.wikipedia.org/wiki/PHP

[13] "***API REST: qué es y cuáles son sus ventajas en el desarrollo de proyectos***"

https://bbvaopen4u.com/es/actualidad/api-rest-que-es-y-cuales-son-sus-ventajas-en-el-desarrollo-de-proyectos

[14] "***Servicios web RESTful con HTTP. Parte I: Introducción y bases téoricas***"

http://www.adwe.es/general/colaboraciones/servicios-web-restful-con-http-parte-i-introduccion-y-bases-teoricas

[15] "***Android Runtime***"

https://es.wikipedia.org/wiki/Android_Runtime_%28ART%29

[16] "***Dalvik***"

https://es.wikipedia.org/wiki/Dalvik

[17] "***Arquitectura Android***"

https://sites.google.com/site/swcuc3m/home/android/generalidades/2-2-arquitectura-de-android

[18] "***Android - Overview***"

https://www.tutorialspoint.com/android/android_overview.htm

[19] "***Qué es un CMS y qué ventajas tiene***"

https://www.departamentodeinternet.com/que-es-un-cms-y-que-ventajas-tiene/

[20] "***Joomla: Introducción***"

https://mundo20.wordpress.com/2008/04/25/joomla-introduccion/

Para los apartados de análisis y diseño se ha utilizado documentación propia de las siguientes asignaturas:

- ***Fundamentos de Ingeniería de Software*** (2º curso Grado en Ingeniería Informática, EPS Jaén).
- ***Gestión y Control de Proyectos Informáticos*** (3º curso Grado en Ingeniería Informática, EPS Jaén).

Anexo I: Cuestionario para el análisis de requerimientos.

En el presente anexo se plasma el formulario que se entrega a los usuarios de la peña deportiva Viejos Amigos FS para poder determinar los requerimientos tanto de la aplicación web como de la aplicación móvil.

ENCUESTA VIEJOS AMIGOS FS

Hola socios y socias de la peña. Como siempre es un reto para nosotros, y viendo el auge y el cada vez mayor número de socios que se inscriben cada año, hemos decidido que ha llegado la hora de lanzar un nuevo proyecto para darnos a conocer y estar más cerca de vosotros.

Por este motivo, necesitamos de vuestra participación.

Para obtener una solución más aproximada, por favor, conteste de manera coherente y con la mayor sinceridad posible, sólo le llevará unos cinco minutos.

La encuesta consta de diez preguntas. Por favor rodea con un círculo la opción elegida. En caso de que pueda seleccionar más de una acción será indicado en la pregunta.

11. ¿Cuántos años tiene?

E) 15 – 30 años
F) 30 – 40 años
G) 40 – 60 años
H) Más de 60 años

12. ¿En qué situación laboral se encuentra?

E) Estudiante
F) Trabajando
G) En situación de desempleo
H) Jubilado

13. ¿Está familiarizado con la tecnología?

C) Sí
D) No

14. ¿Con qué frecuencia se conecta a Internet?

E) Diariamente
F) Una vez a la semana
G) Una vez al mes
H) No sé que es internet

15. ¿Tiene un smartphone o teléfono inteligente?

C) Sí
D) No

16. Seleccione si es usuario de alguna de estas redes sociales.

E) Facebook
F) Twitter
G) Instagram
H) Google +

17. ¿Lee usted la prensa digital deportiva?

D) Sí, a diario
E) Alguna que otra vez
F) No

18. ¿Con qué frecuencia participa en las actividades que promueve la peña desportiva Viejos Amigos FS?

D) Suelo asistir a casi todas las actividades
E) Asistencia media, siete u ocho actividades por año
F) No asisto a ninguna actividad

19. ¿Le gustaría que Viejos Amigos FS tuviera presencia en Internet?

C) Sí
D) No

20. ¿Qué le gustaría poder hacer en el sitio web?

F) Leer noticias acerca de la peña
G) Consultar las nuevas actividades promovidas por la peña
H) Contratar actividades o servicios
I) Poder actualizar las estadísticas de aquellos socios que participan en las actividades deportivas
J) Poder pagar a través de la web

¡Muchas gracias por tu participación!

Anexo II: Manual de instalación y configuración

En este anexo se describirán los pasos a seguir para dejar ambas aplicaciones funcionando.

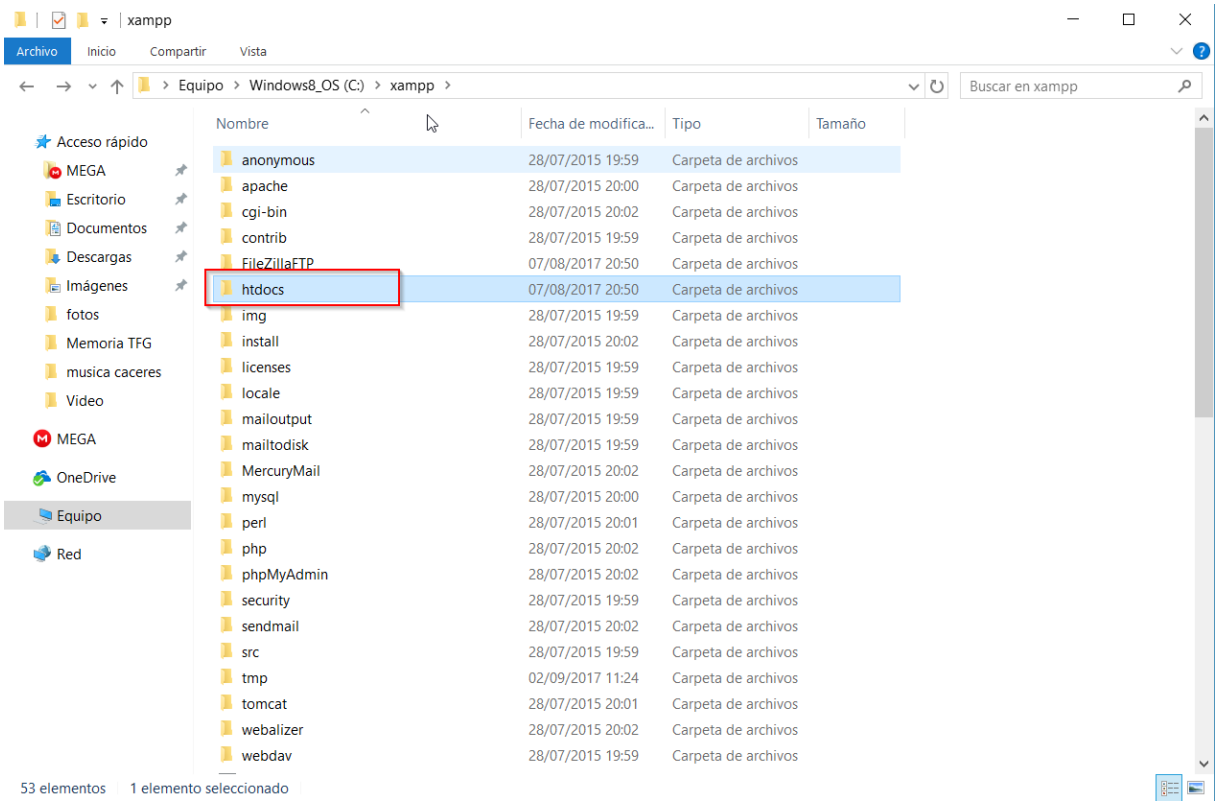
Instalación y configuración de la aplicación web

Para el correcto funcionamiento de la aplicación web seguir los siguientes pasos:

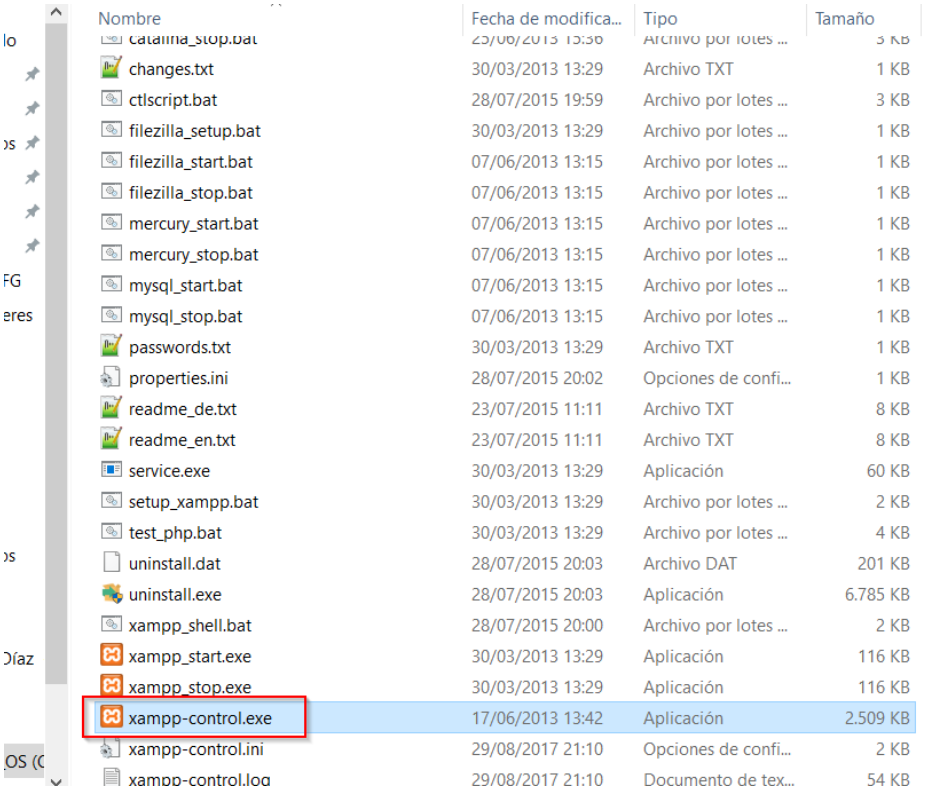
1. Descargar e instalar XAMPP. Como se detalla en puntos anteriores de este trabajo esto es necesario para, como mínimo levantar el servidor Apache y el servidor de base de datos de MySQL. Si no sabe como instalar XAMPP clicke en este enlace y siga los pasos establecidos:

http://es.wikihow.com/instalar-XAMPP-en-Windows

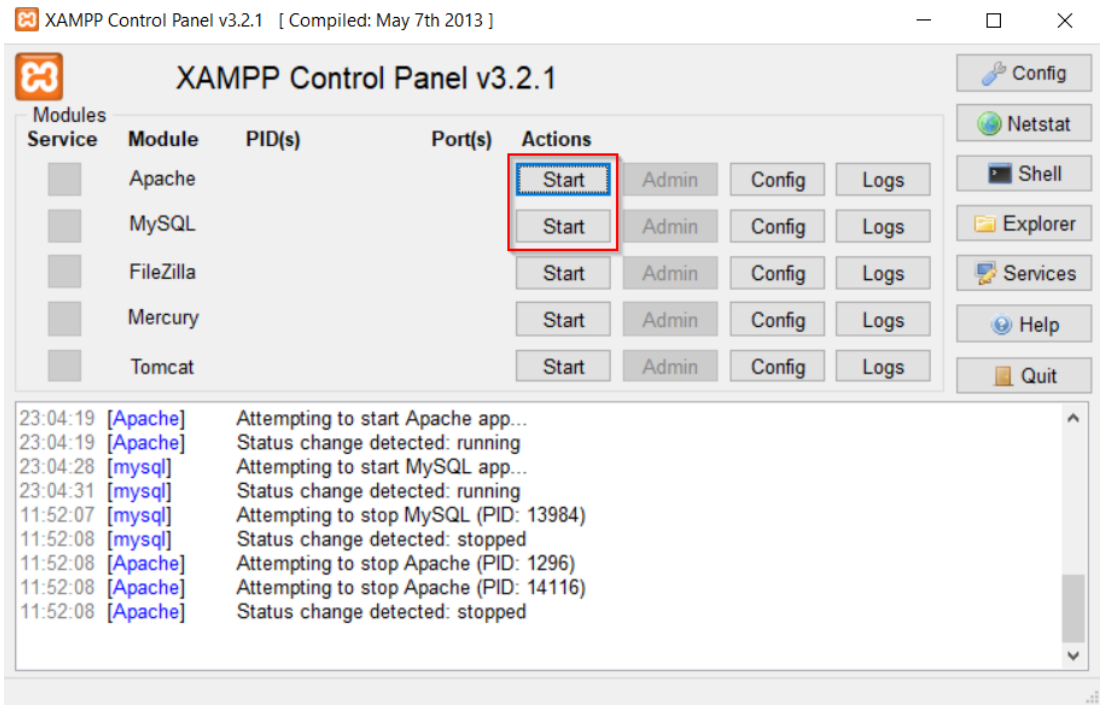
2. Extraer el archivo htdocs.zip contenido en el directorio de este trabajo y copiarlo en la carpeta htdocs dónde haya instalado XAMPP:



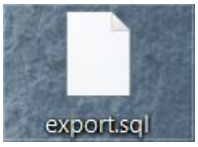
3. El siguiente paso es levantar el servicio web y el servicio de base de datos. Entrar al directorio de instalación de XAMPP y ejecutar **xampp-control.exe**:



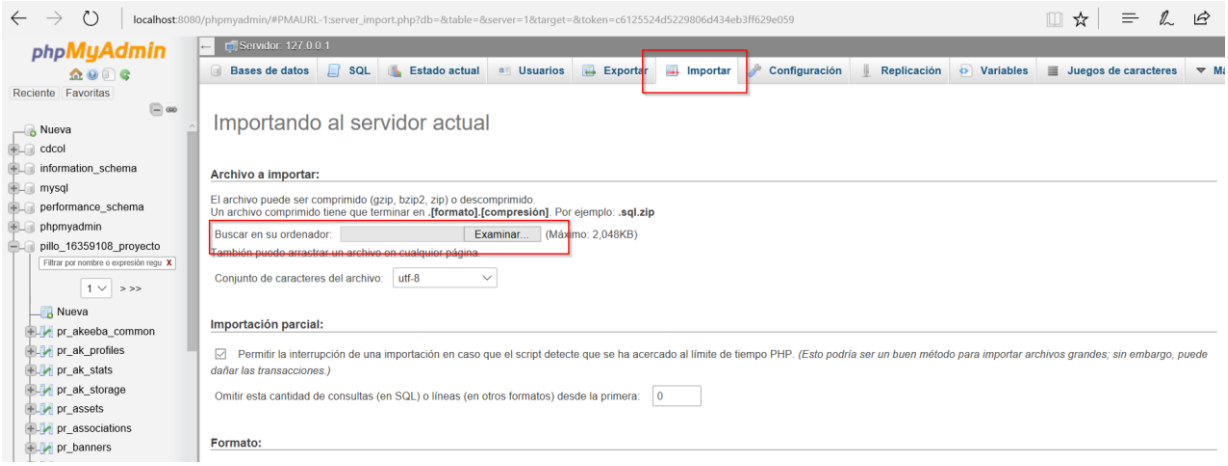
4. Arrancar los módulos de Apache y MySQL (como mínimo):



5. El siguiente paso es importar la base de datos con los datos ya usados. Esto contiene tanto el conjunto de tablas con sus propiedades como los datos actuales. Para ello, copiar el archivo export.sql contenido en el dvd entregado en algún directorio localizable de su equipo.



6. Para importar la base de datos debemos entrar en phpMyAdmin en la siguiente dirección: http://localhost:8080/phpmyadmin (Cuidado con los puertos).
7. Importar la base de datos. Una vez abierta la consola de phpMyAdmin, clickar en Importar y luego seleccionar el archivo export.sql y clicar en Aceptar. Con esto ya tendremos la base de datos configurada.



8. Por último, y como punto muy importante, tenemos que abrir el servidor para que pueda ser visto en red local. Esto es necesario para que la aplicación Android pueda realizar peticiones al servidor. Para ello seguir el siguiente enlace, el cuál viene explicado a la perfección y en el cuál me basé para realizar este procedimiento:

http://web.tecnodus.com/desarrollo-web/apache/abrir-acceso-por-red-a-xampp

Instalación y configuración de la aplicación Android

La instalación de la aplicación Android es más sencilla y sólo hay que seguir unos cuantos puntos básicos:

9. Copiar el archivo viejosamigosfs.apk contenido en el DVD de este trabajo a algún directorio de su dispositivo móvil. (Nota: el desarrollo está implementado para dispositivos con una versión Android 4 o superior, no funcionando en dispositivos con una versión más anterior a esta). Con ésto ya estaría instalada la aplicación.

10. **Nota:** El desarrollo está implementado en local, por tanto, el endpoint es estático en red local. La aplicación está preparada para que funcione con la siguiente dirección IP: **192.168.1.130:8080.** Este end-point está configurado en la aplicación en el archivo de configuración de graddle y se podría cambiar (aunque habría que tocar código). Para solucionar esto, lo más fácil sería cambiar manualmente la dirección IP del equipo que hará de servidor a la especificada anteriormente.

Anexo III: Manual de usuario de la aplicación web

En este anexo se ofrece al lector una guía básica de las acciones que puede realizar en la aplicación web Viejos Amigos FS. Está estructurada por funcionalidad y en cada una de estas funcionalidades se explicará que tipo de usuario tiene acceso a ellas.

Entrar en la aplicación web

1. Acceder a la siguiente dirección: http://localhost:8080

Este acceso es universal, por lo cual, cualquier usuario que acceda podrá ver el contenido principal de la página: noticias, timeline de twitter, etc.

Iniciar sesión

1. Un usuario registrado puede iniciar sesión introduciendo su usuario y su contraseña en el módulo “Formulario de acceso”:

Una vez registrado, dependiendo de los permisos del usuario registrado (usuario, administrador), aparecerán en el menú de la cabecera las acciones permitidas para dicho usuario.

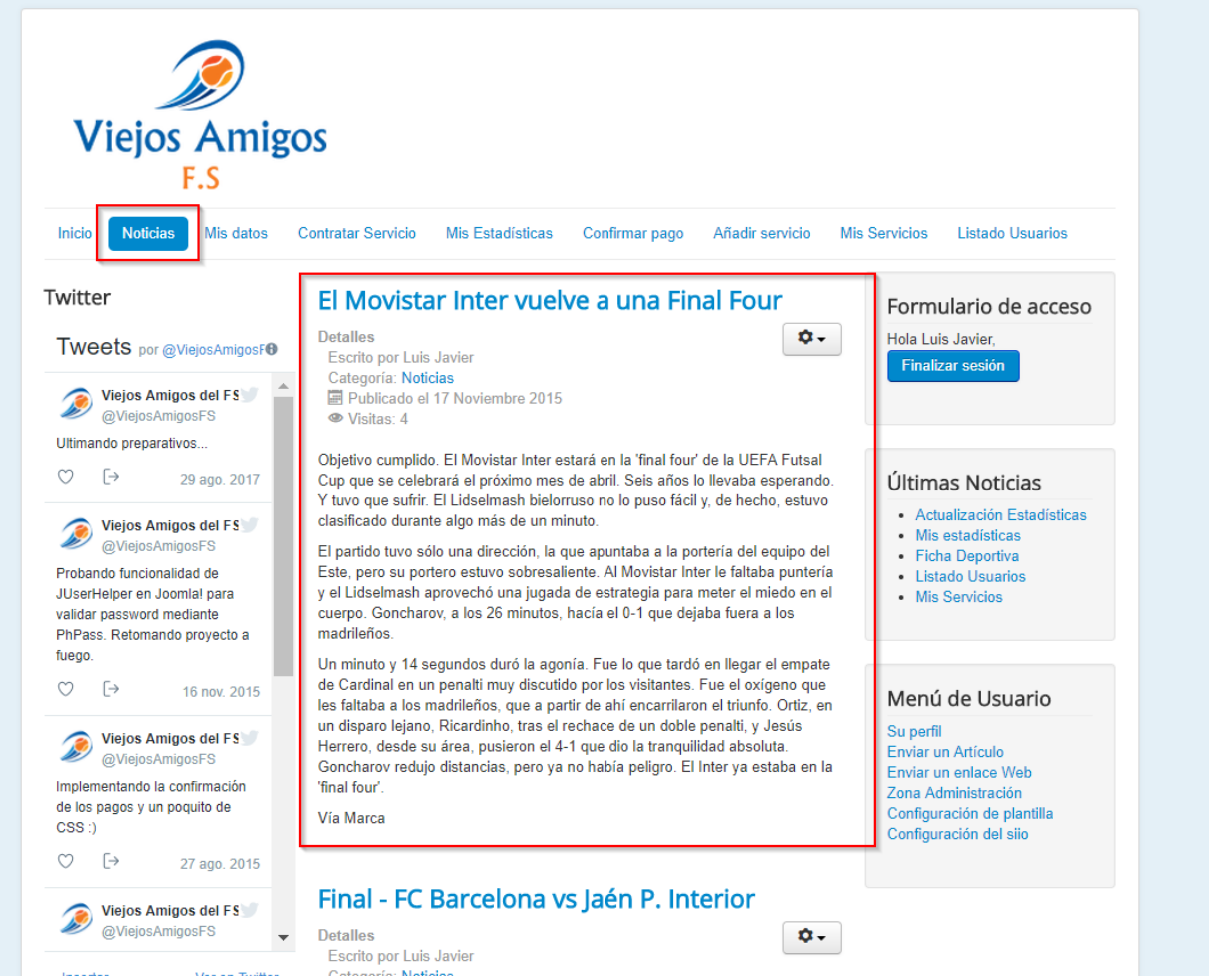
2. Para un usuario de tipo usuario las acciones permitidas son:

3. Para un usuario de tipo administrador las acciones disponibles son:

4. Desde este menú se tendrán acceso a las diferentes funcionalidades de la aplicación web que serán detallados en las siguientes actividades. Aclarar que dichas funcionalidades sólo aparecerán si el usuario registrado tiene los permisos pertinentes para ello.

Leer las noticias de la web

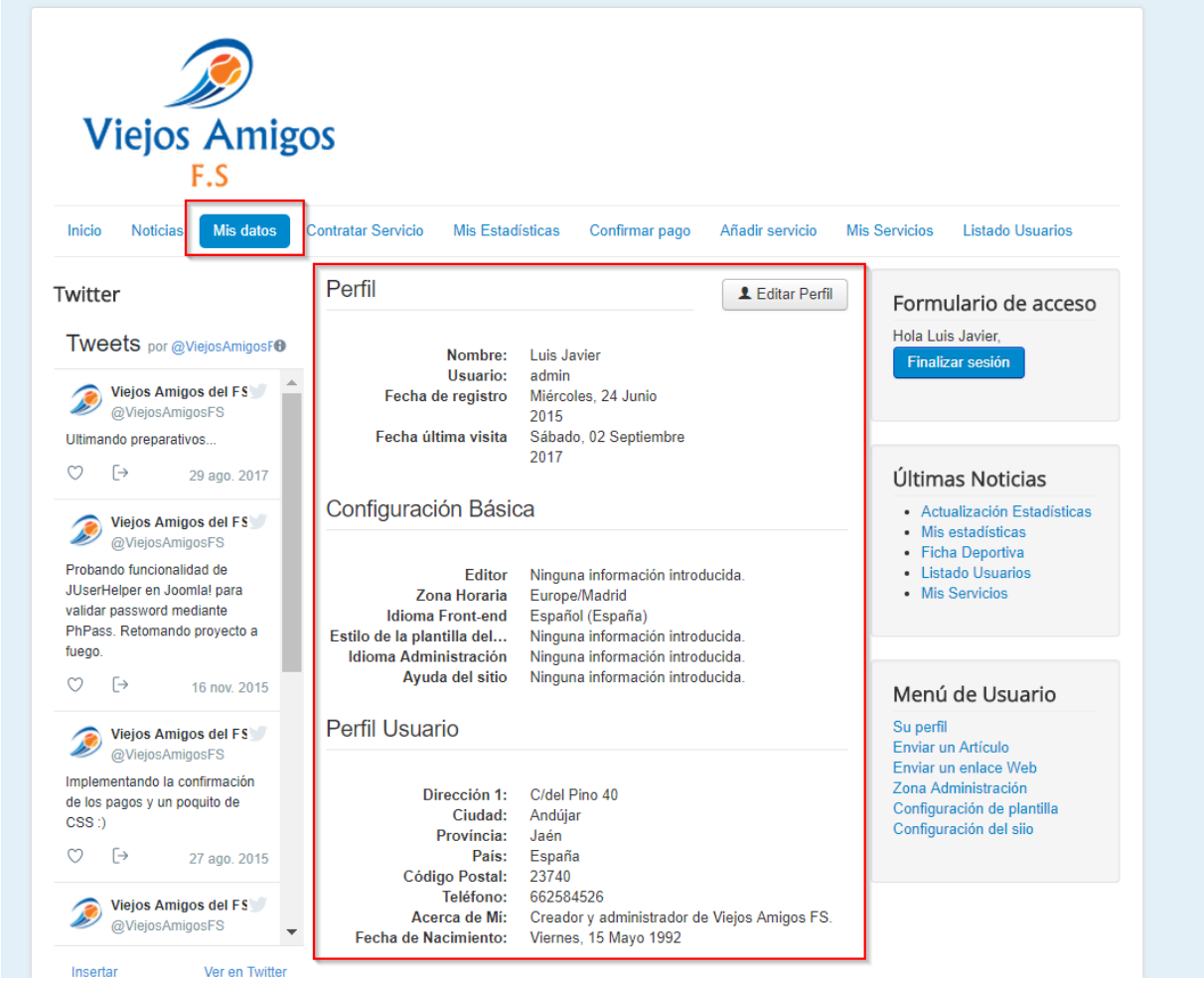
1. Clicar en el elemento de menú "**Noticias**":



2. Aparecerán las diferentes noticias (al igual que en el apartado Inicio) con información adicional de quién la creó y cuando se creó, etc.

Consultar datos del usuario registrado

1. Clicar en el elemento de menú “**Mis datos**”:

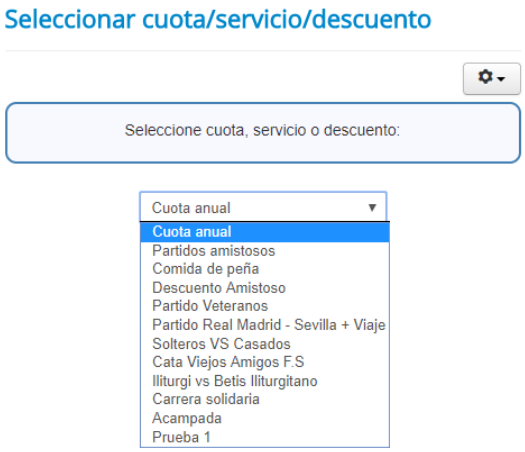


Aparecerá la información referente al usuario registrado.

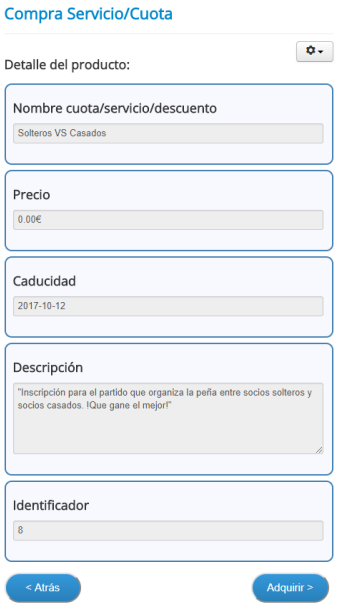
Contratar servicio o actividad

1. Clicar en el elmento de menú “Contratar servicio”:

2. Seleccionar el servicio que se desea contratar en el desplegable y pulsar “Aceptar”:

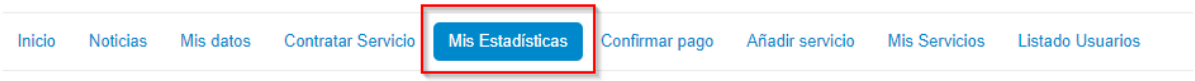


3. Aparecerá la información de dicho servicio y clicar en “Adquirir” para suscribirte a dicho servicio. Aparecerá un mensaje informando si todo ha ido bien o si el usuario ya tenía el servicio contratado:

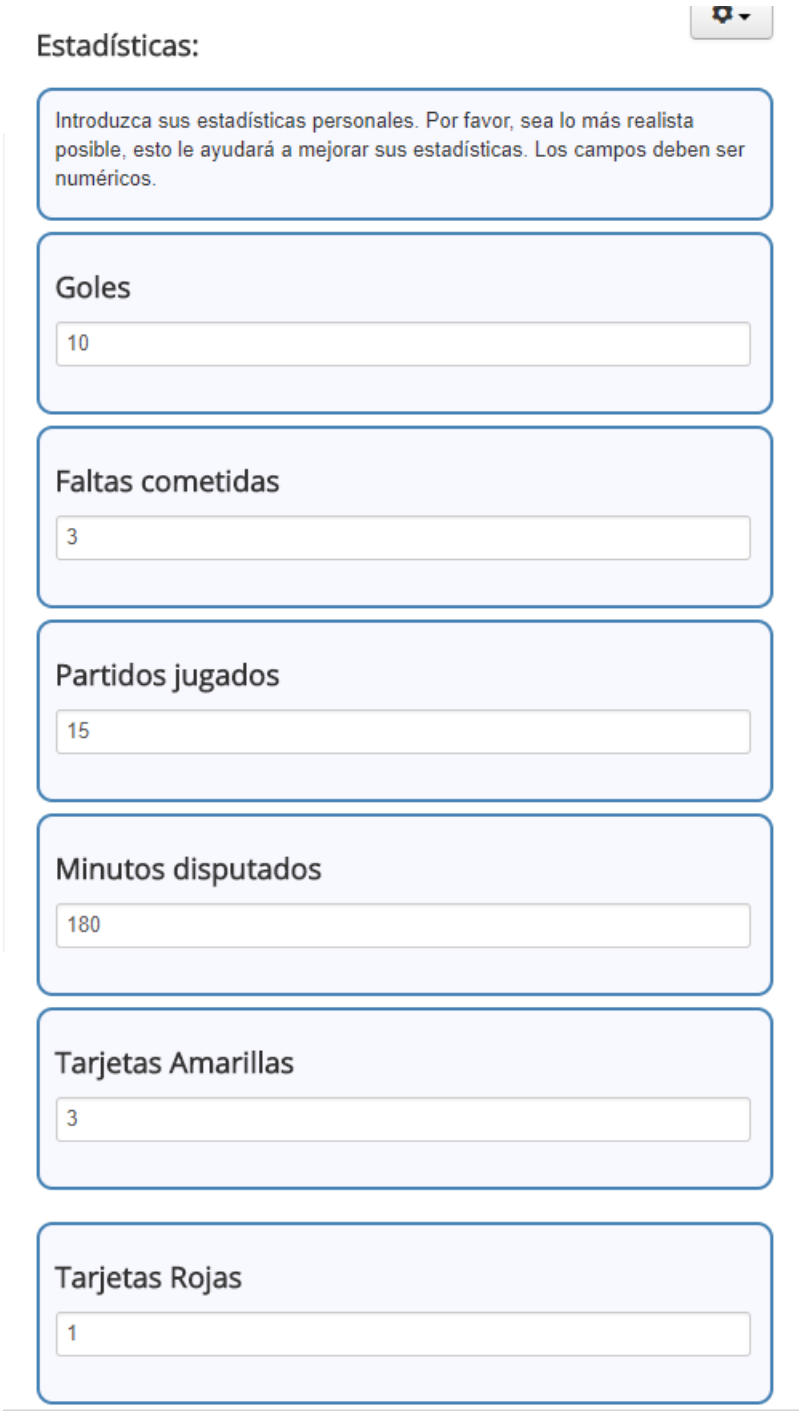


Consultar estadísticas del usuario

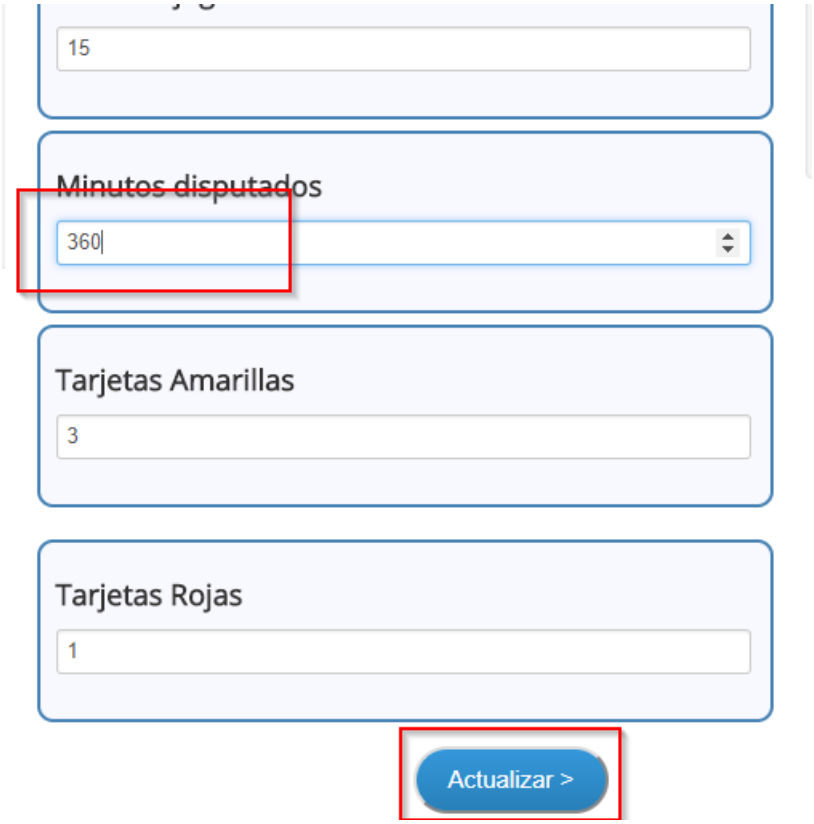
1. Clicar en el elemento de menú “Mis estadísticas”:



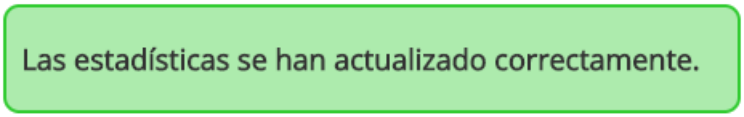
2. Aparecerá el módulo de estadísticas:



3. El usuario podrá modificar sus estadísticas y actualizarlas pulsando en el botón “Actualizar”:



Se mostrará un mensaje de confirmación confirmando que la actualización se ha llevado a cabo correctamente:



Consultar los servicios contratados

1. Clicar en el elemento de menú “**Mis servicios**”:

2. Aparecerán los servicios contratados por el usuario divididos en tres bloques:

a) Situación del abono anual.
b) Servicios confirmados (Un administrador ha validado el pago).
c) Servicios no confirmados.

Situación Abono Anual

Situación	Caducidad
Su abono anual está registrado y podrá adquirir nuestros servicios. !Gracias por confiar en nosotros!	2017-12-11

Servicios confirmados

Nombre	Descripción	Caducidad
Comida de peña	Comida ofrecida por la peña. El socio pagará una pequeña parte de esta.	2017-09-05
Partidos amistosos	Se podrán ver los partidos amistosos de los clubs	2017-09-29
Cata Viejos Amigos F.S	Gran cata el 9 de noviembre, ¡ven a pasar un buen rato con nosotros!. Habrá cerveza y mucho Aquarius	2017-11-09
Partido Veteranos	Partido que se disputará entre los socios más antiguos de la peña.	2017-10-29
Carrera solidaria	Carrera a favor de la promoción del deporte.	2017-09-30
Acampada	Acampada para día de convivencia.	2017-11-04
Prueba 1	Pruebas	2017-10-20

Servicios NO confirmados

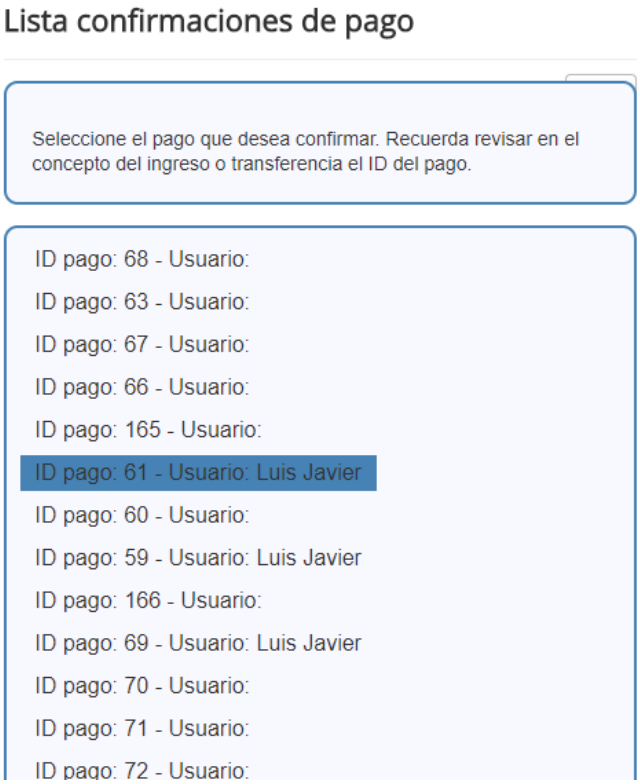
Nombre	Descripción	Caducidad
Partido Real Madrid - Sevilla + Viaje	La peña realizará en diciembre un viaje a Madrid para ver al Real Madrid contra el Sevilla. El precio incluye viaje en autobús. Más información próximamente.	2017-11-14

Confirma pago [Administrador]

Esta acción debe llevarla a cabo un administrador. Cuando el administrador compruebe que el usuario ha realizado el pago del servicio/actividad, puede confirmarlo llevando a cabo los siguientes pasos:

1. Clicar en el elemento de menú "**Confirma pago**":

2. Aparecerá una lista de pagos, en la cual, el administrador debe seleccionar el pago correspondiente.

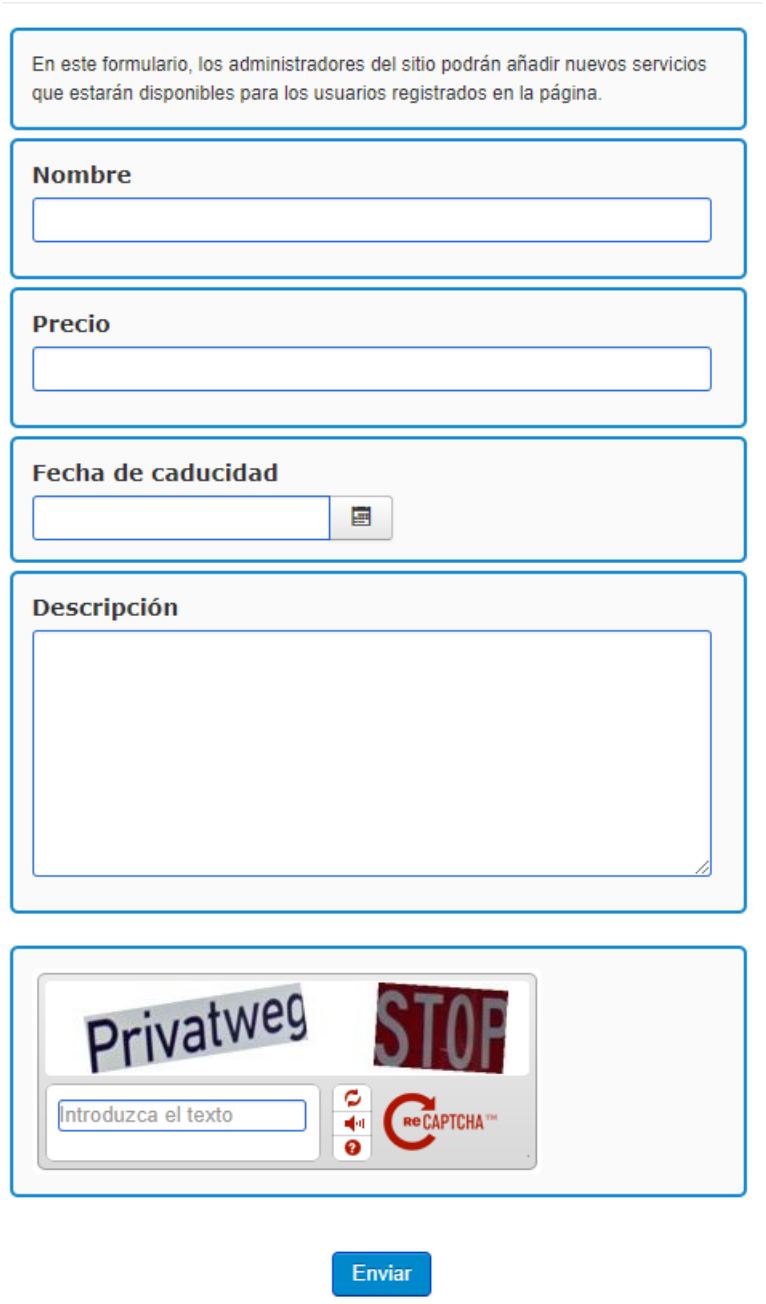


3. Hacer click en el botón "Enviar":

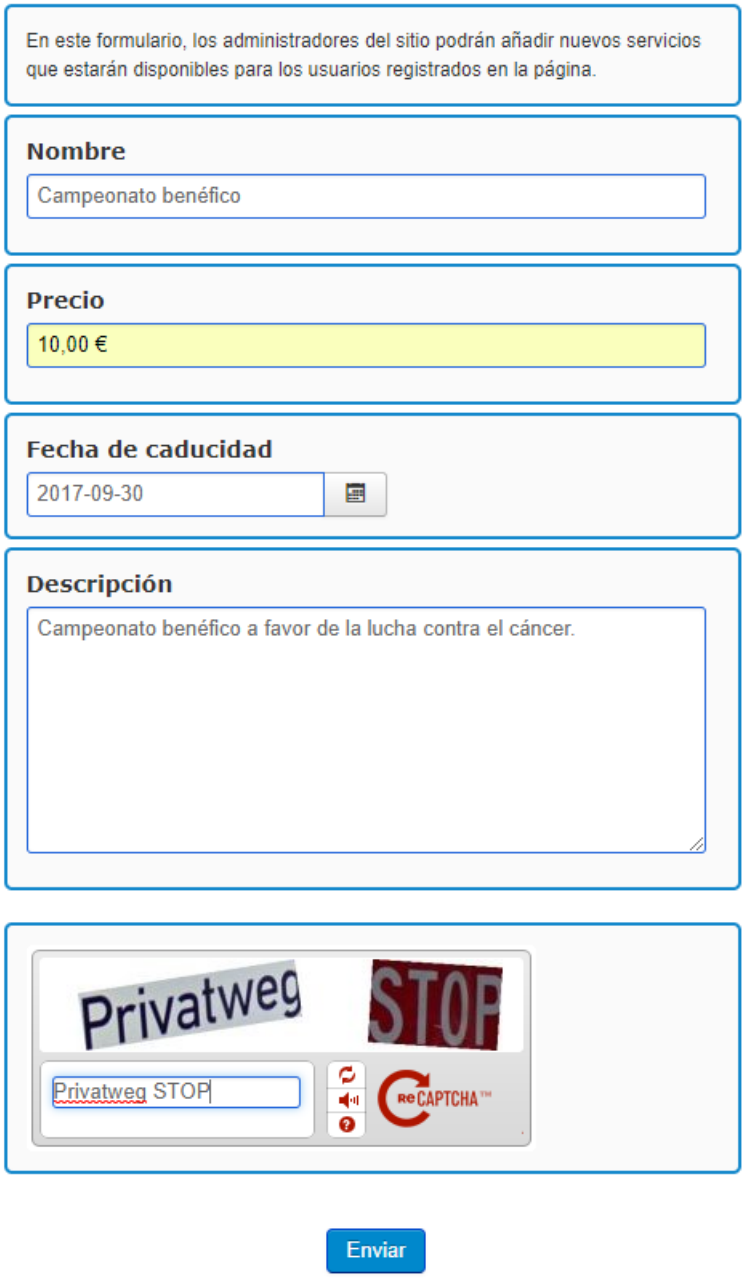
Añadir un nuevo servicio [Administrador]

1. Hacer click en el elemento de menú “Añadir servicio”:

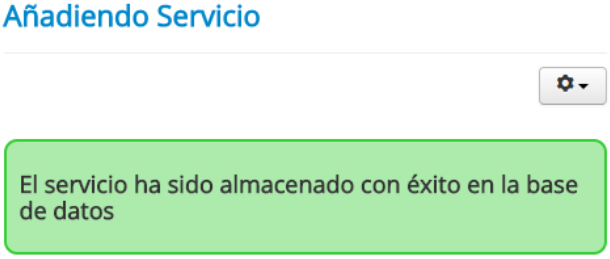
2. Aparecerá un formulario a cumplimentar para añadir el nuevo servicio, en el cuál se han de informar las características del servicio:



3. Cumplimentar formulario y hacer click en "**Enviar**":

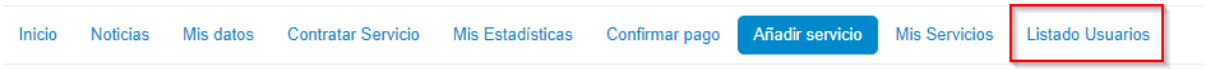


4. Al hacer clic en Enviar se almacenará el nuevo servicio:



Ver listado de usuarios [Administrador]

1. Hacer click en el elemento de menú "**Listado de usuarios**":



2. Aparecerá un listado de usuarios:

3. Si el usuario pulsa en "**Ver Ficha**" sobre cualquier usuario podrá ver la ficha de éste:

Consideraciones finales

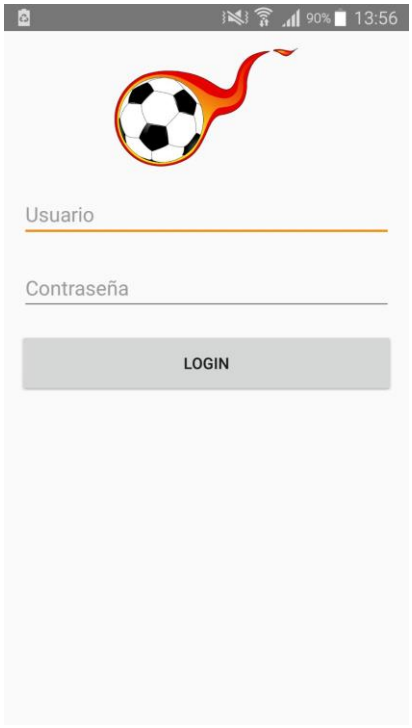
Para terminar con este manual, comentar que se ha obviado parte de la funcionalidad propia de la administración de Joomla!. Debido a la extensión de documento, este manual se centra en la lógica propia del desarrollo del proyecto.

Anexo IV: Manual de usuario de la aplicación móvil

En este anexo se ofrece al lector una guía básica de las acciones que puede realizar en la aplicación móvil Viejos Amigos FS. En la aplicación móvil no se distingue entre perfiles de usuario, y para poder acceder el usuario necesita estar registrado previamente en la web. Se hará la distinción entre las difrentes vistas o pantallas de la aplicación.

Iniciar sesión

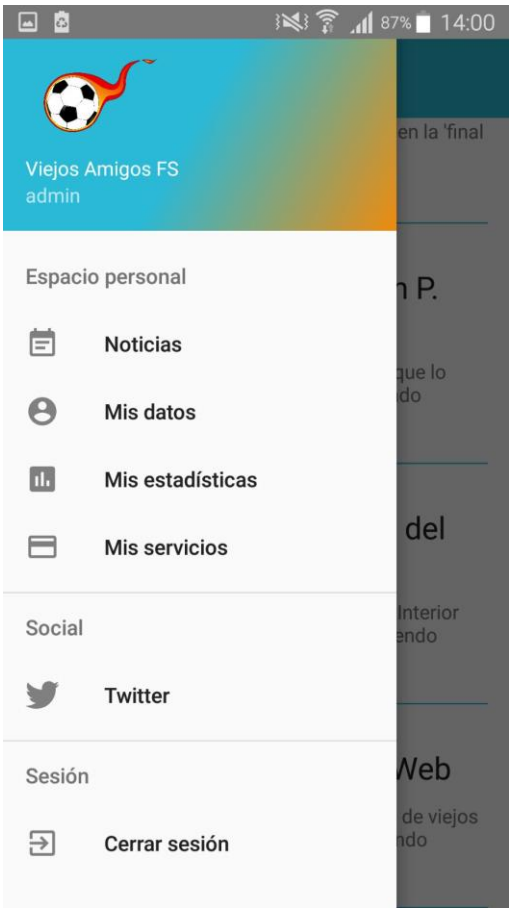
1. Cuando el usuario accede a la aplicación, aparecerá la siguiente pantalla:



2. El usuario introduce su usuario y su contraseña y pulsa “LOGIN”. Si el usuario y contraseña es introducido tendrá acceso a la aplicación. Por defecto se entra en la vista de “Noticias”.

Menú de usuario

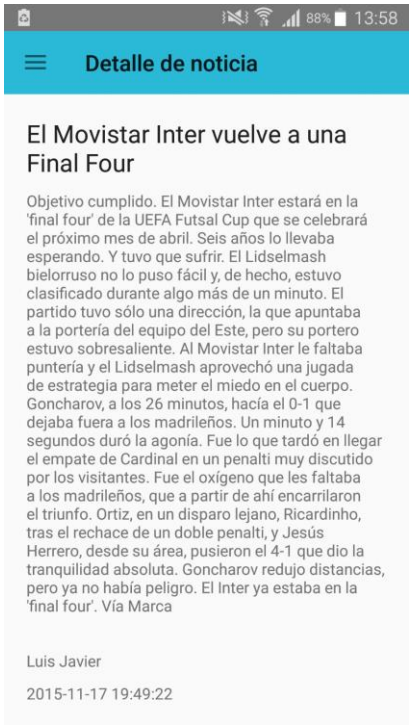
Este menú es accesible desde todas las demás vistas a partir del Login, y desde éste se puede acceder desde cualquier pantalla a las demás pantallas:



Noticias

1. Pulsar sobre “Noticias” en el menú de usuario. Aparecerá una pantalla en la cual se muestran las noticias disponibles:

2. Para ver el detalle completo de una noticia, pulsar sobre ella. Aparecerá la pantalla de “Detalle de noticia”:

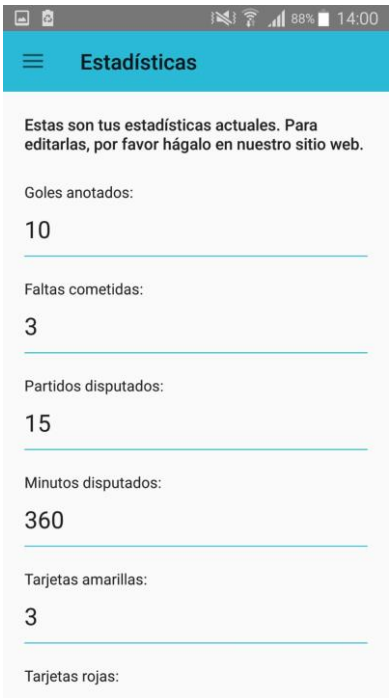


Mis datos

1. Pulsar sobre “Mis datos” en el menú de usuario. Aparecerá una pantalla en la cual se muestran los datos del usuario:

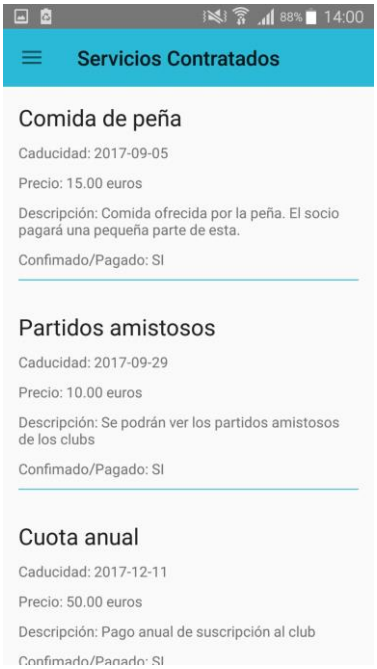
Mies estadísticas

1. Pulsar sobre “Mis estadísticas” en el menú de usuario. Aparecerá una pantalla en la cual se muestran las estadísticas asociadas al usuario:



Mis servicios

1. Pulsar sobre “Mis servicios” en el menú de usuario. Aparecerá una pantalla en la cual se muestran los servicios contratados por el usuario:

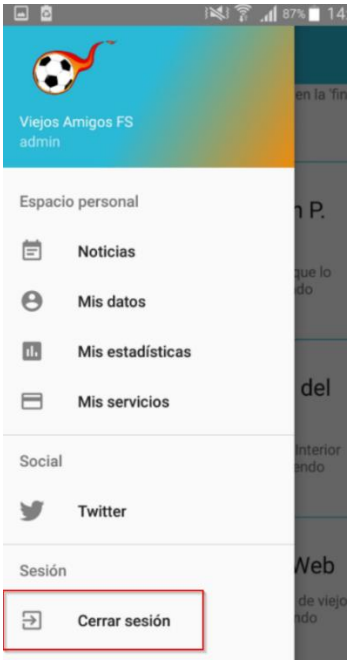


Twitter

1. Pulsar sobre “Twitter” en el menú de usuario. Aparecerá una pantalla en la cual se muestran un timeline con los tweets publicacos en la cuenta de la peña:

Cerrar sesión

1. Pulsar sobre “Cerrar Sesión” en el menú de usuario. Se cerrará la sesión de usuario y automáticamente aparecerá la pantalla de Login para volver a introducir los datos:



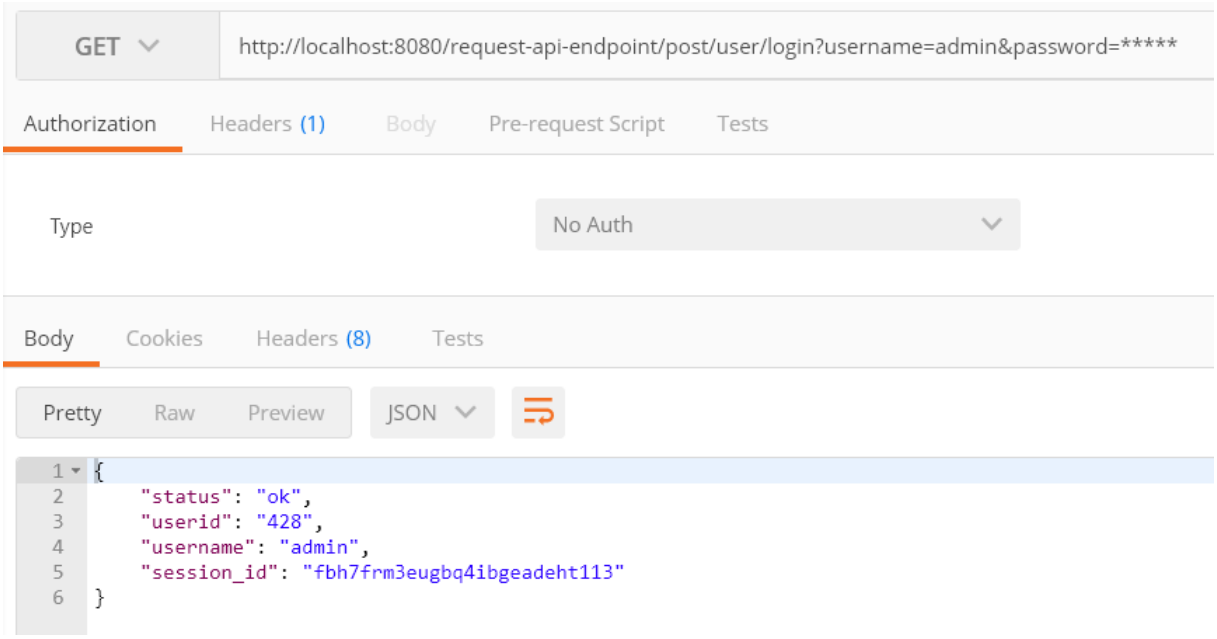
Anexo V: Pruebas servicios REST

A continuación se detallarán las diferentes peticiones y sus correspondientes respuestas que se hacen desde la aplicación Android.

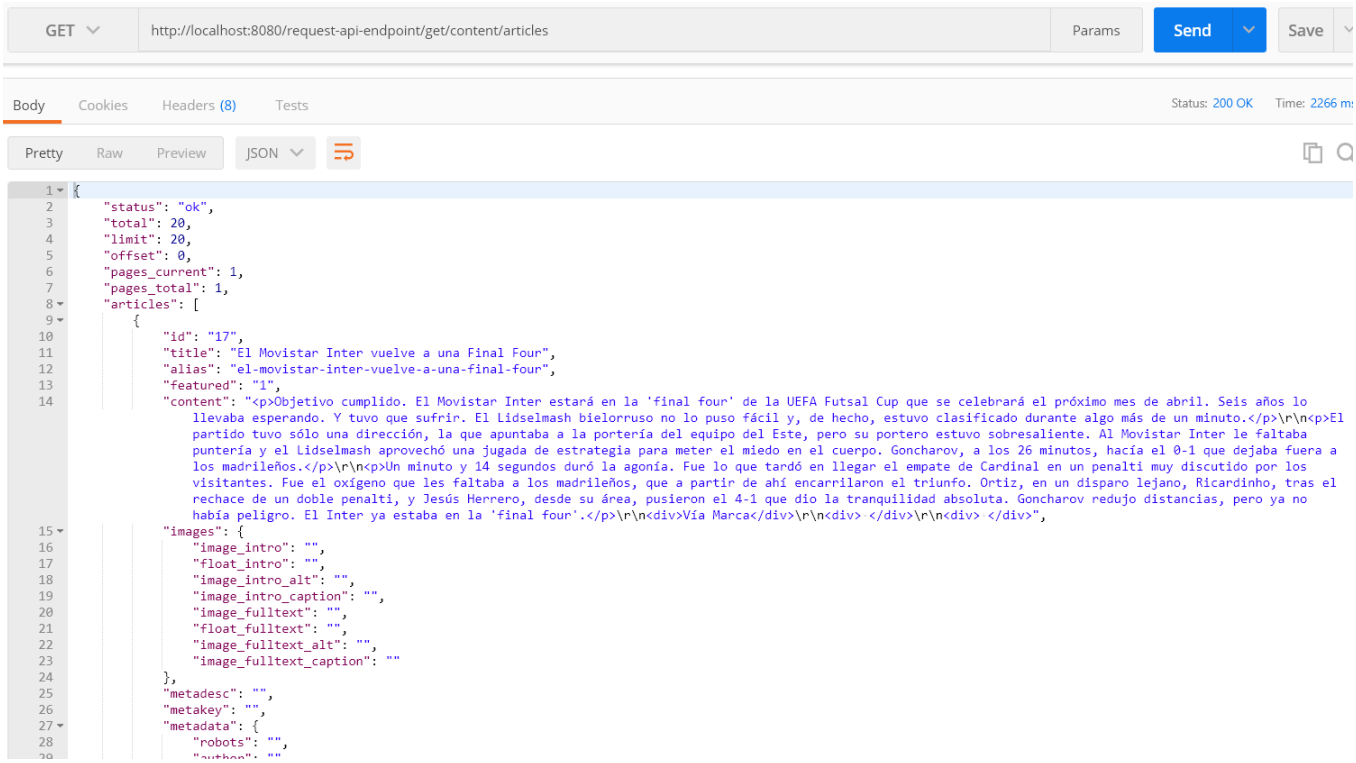
Para ello hemos usado la extensión de Google Chrome POSTMAN. POSTMAN es un sencillo cliente REST que lanza peticiones (GET, POST, PUT, DELETE) y muestra en un campo de salida la respuesta obtenida.

Las pruebas se realizarán para el usuario administrador, cuyo ID dentro del sistema es el 428.

Login:

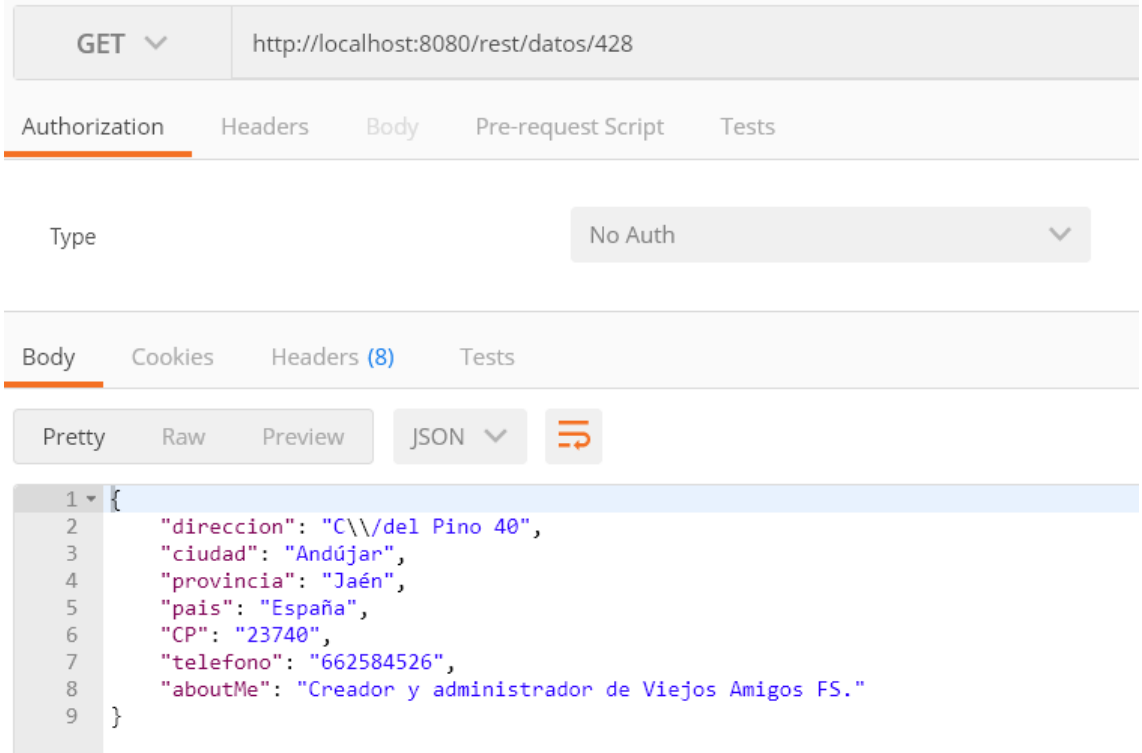


Noticias:



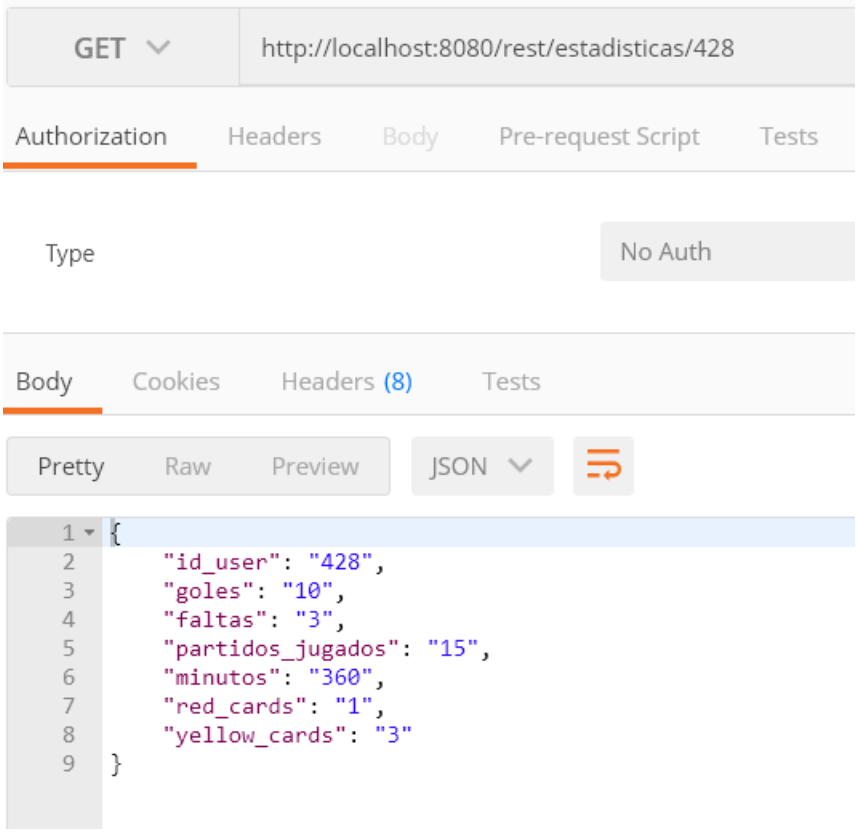

```
{
    "status": "ok",
    "total": 20,
    "limit": 20,
    "offset": 0,
    "pages_current": 1,
    "pages_total": 1,
    "articles": [
        {
            "id": "17",
            "title": "El Movistar Inter vuelve a una Final Four",
            "alias": "el-movistar-inter-vuelve-a-una-final-four",
            "featured": "1",
            "content": "<p>Objetivo cumplido. El Movistar Inter estará en la 'final four' de la UEFA Futsal Cup que se celebrará el próximo mes de abril. Seis años lo llevaba esperando. Y tuvo que sufrir. El Lidselmash bielorruso no lo puso fácil y, de hecho, estuvo clasificado durante algo más de un minuto.</p>\r\n<p>El partido tuvo sólo una dirección, la que apuntaba a la portería del equipo del Este, pero su portero estuvo sobresaliente. Al Movistar Inter le faltaba puntería y el Lidselmash aprovechó una jugada de estrategia para meter el miedo en el cuerpo. Goncharov, a los 26 minutos, hacía el 0-1 que dejaba fuera a los madrileños.</p>\r\n<p>Un minuto y 14 segundos duró la agonía. Fue lo que tardó en llegar el empate de Cardinal en un penalti muy discutido por los visitantes. Fue el oxígeno que les faltaba a los madrileños, que a partir de ahí encarrilaron el triunfo. Ortiz, en un disparo lejano, Ricardinho, tras el rechace de un doble penalti, y Jesús Herrero, desde su área, pusieron el 4-1 que dio la tranquilidad absoluta. Goncharov redujo distancias, pero ya no había peligro. El Inter ya estaba en la 'final four'.</p>\r\n<div>Vía Marca</div>\r\n<div> </div>\r\n<div> </div>",
            "images": {
                "image_intro": "",
                "float_intro": "",
                "image_intro_alt": "",
                "image_intro_caption": "",
                "image_fulltext": "",
                "float_fulltext": "",
                "image_fulltext_alt": "",
                "image_fulltext_caption": ""
            },
            "metadesc": "",
            "metakey": "",
            "metadata": {
                "robots": "",
```


Mis datos:

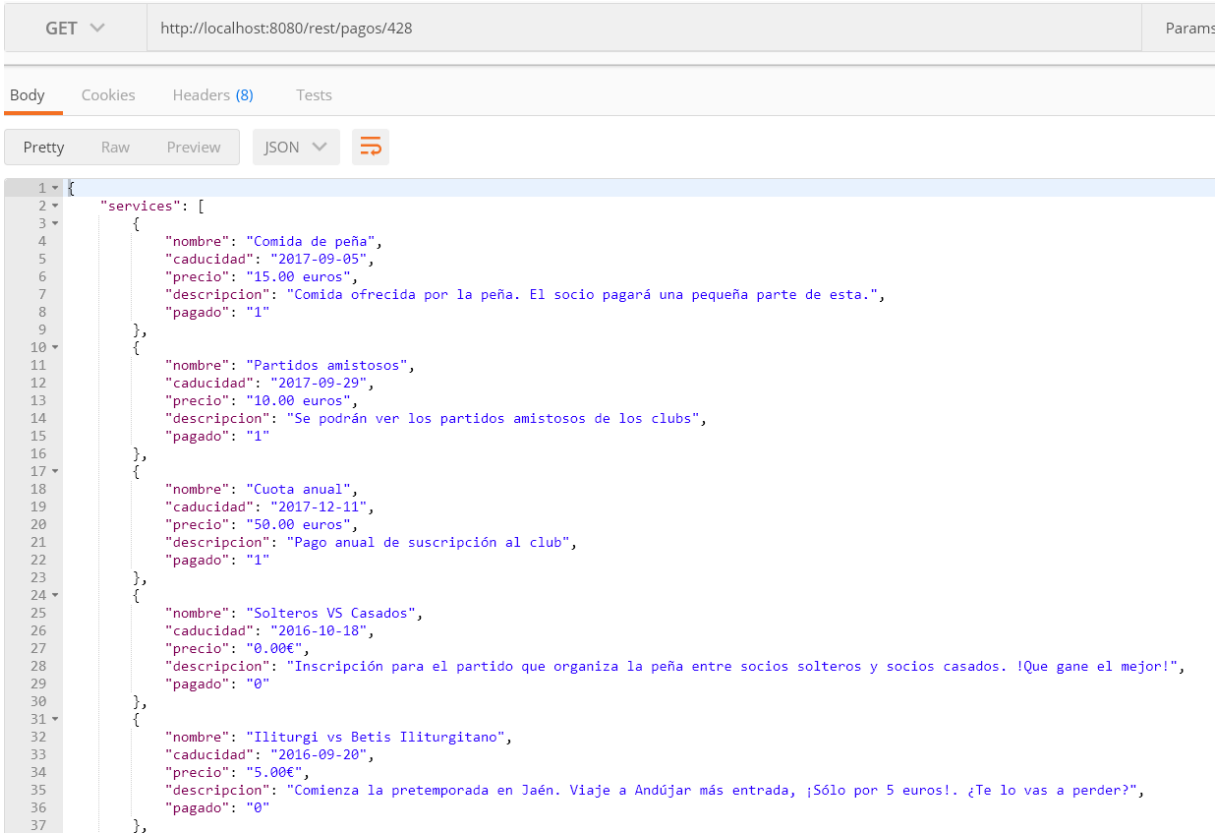



```
{
    "direccion": "C\\/del Pino 40",
    "ciudad": "Andújar",
    "provincia": "Jaén",
    "pais": "España",
    "CP": "23740",
    "telefono": "662584526",
    "aboutMe": "Creador y administrador de Viejos Amigos FS."
}
```

Mis estadísticas:



Mis servicios:




```
{
    "services": [
        {
            "nombre": "Comida de peña",
            "caducidad": "2017-09-05",
            "precio": "15.00 euros",
            "descripcion": "Comida ofrecida por la peña. El socio pagará una pequeña parte de esta.",
            "pagado": "1"
        },
        {
            "nombre": "Partidos amistosos",
            "caducidad": "2017-09-29",
            "precio": "10.00 euros",
            "descripcion": "Se podrán ver los partidos amistosos de los clubs",
            "pagado": "1"
        },
        {
            "nombre": "Cuota anual",
            "caducidad": "2017-12-11",
            "precio": "50.00 euros",
            "descripcion": "Pago anual de suscripción al club",
            "pagado": "1"
        },
        {
            "nombre": "Solteros VS Casados",
            "caducidad": "2016-10-18",
            "precio": "0.00€",
            "descripcion": "Inscripción para el partido que organiza la peña entre socios solteros y socios casados. !Que gane el mejor!",
            "pagado": "0"
        },
        {
            "nombre": "Iliturgi vs Betis Iliturgitano",
            "caducidad": "2016-09-20",
            "precio": "5.00€",
            "descripcion": "Comienza la pretemporada en Jaén. Viaje a Andújar más entrada, ¡Sólo por 5 euros!. ¿Te lo vas a perder?",
            "pagado": "0"
        },
```