PROYECTO FINAL DE CARRERA

Gestión de Horarios y Accesos de Un Gimnasio

Alumno: Héctor Martínez García
Directora: Sara Blanc Clavero

Ingeniería Técnica en Informática de Sistemas
28 de Septiembre de 2011

Índice

1.- Introducción

En la actualidad nos encontramos en el año 2011, y tras unas décadas de la creación de Internet y su estandarización por todo el mundo, estamos viviendo una nueva revolución. Los dispositivos móviles con conexión a Internet están creciendo a un ritmo frenético. Internet es el lugar donde vamos a buscar nuestra información, y también donde queremos realizar nuestras gestiones cotidianas.

Todo ese auge viene debido al afán de disponer de la información requerida, sea cual sea el sitio en que estemos. Esto mediante un dispositivo Smartphone, Tablet o portátil conectados a Internet resulta de lo más sencillo y práctico. Por ello para los negocios de corte tradicional también es realmente importante disponer de un espacio virtual donde un potencial cliente pueda encontrar toda esa información y que pueda interactuar potenciando la flexibilidad en la gestión la comunicación virtual. Deberemos por lo tanto, mostrar qué somos, quién somos y a qué nos dedicamos.

Una vez el usuario ya ha calmado sus ansias de información, necesita pasar al siguiente paso: La comunicación. La cuestión es bien clara, si yo dispongo de un canal de comunicación (como es internet), por qué debo utilizar otro para comunicarme (teléfono, presencia en el negocio, etc.). Por un motivo estamos hablando de pura usabilidad, por ejemplo a nadie le gustaría tener que bajarse del coche para pagar un peaje. Por otro lado está la cuestión económica, si yo pago una cierta cantidad por tener Internet, voy a intentar sacarle el máximo provecho reduciendo el consumo de otros que me cuestan un dinero. Y por último está el tiempo, ese gran desconocido de muchos de nosotros en esta época, por ejemplo, estoy leyendo una noticia en Internet y recuerdo que no he reservado hora para el médico. Sencillo, me conecto al servicio pertinente y efectúo la reserva. No he tenido que desplazarme a ningún sitio, ni esperar ninguna cola.

Uno de los problemas de este campo es la dificultad de cazar las tendencias a tiempo antes de que alguien ya haya realizado algo similar. Ello es debido a que en el mercado tradicional, la mercadotecnia está basada en costosos y laboriosos estudios de mercado que pueden alargarse meses y años completos. En cambio, en el mercado de Internet este tiempo es mucho más valioso. Sin embargo los usuarios son muy receptivos en este campo, probablemente más que en ningún otro, y uniéndolo a las grandes posibilidades que disponemos, es realmente fácil crear un servicio innovador (aunque no sea a nivel global).

Todas estas novedades que eran impensables hace unos años, hoy por hoy son realidad, y no es sino una tendencia al alza. Numerosos son los estudios que han verificado que en el mundo de los servicios y las compras on-line existe más demanda que oferta. Aquí tenemos un campo suficientemente amplio como para experimentar con infinidad de sectores comerciales. Los límites únicamente los va a poner la propia tecnología y el usuario. Es por tanto la innovación uno de los aspectos más importantes que debemos tener en cuenta en este campo de las TICs.

Para los que hemos tenido el placer de conocer el nacimiento y la expansión de Internet, es una experiencia única que ninguna otra generación posterior tendrá la posibilidad de vivir.

Hemos visto el paso del analógico al digital, de nuestra vida en el mundo a nuestro mundo en Internet, de la consulta de enciclopedias en papel a la consulta de grandes bases de datos online. La información nos ha invadido, ahora no podemos escapar de ella porque está en todas partes. La posibilidad de calmar nuestra curiosidad nos ha vuelto más curiosos todavía, ahora podemos aprender conocimientos sobre campos en los que antes era imposible.

Con todo ello debemos ver a Internet como una herramienta capaz de lograr ayudarnos de forma que antes no podíamos, pero siempre debemos recordar que esto es sólo una extensión de nuestro propio conocimiento.

Por mi parte el consejo que hagamos todos un uso responsable Internet, de forma que todo sean ventajas, y no tengamos que lamentarnos de nada. Ayudar a que Internet sea cada vez más rico en conocimientos aportando nuestro propio conocimiento para que otros puedan aprovecharlo.

Por todos estos motivos y por alguno más que puede que olvide, se pensó en realizar dicho Proyecto Final de Carrera (en adelante PFC) en el ámbito de las Aplicaciones Web. Espero algún día pueda resultar de utilidad para alguien la cantidad ingente de horas que se han dedicado tanto a la elaboración de esta memoria, como al desarrollo de la propia aplicación. Sería sin duda la garantía de que todo el trabajo ha valido la pena.

Agradezco también la elección de este PFC que me ha hecho reconsiderar mi futuro laboral una vez titulado como Ingeniero Técnico en Informática de Sistemas. Dicha carrera va a ser orientada hacia este mundo tan competitivo y en constante evolución. Por ello presento este PFC como el principio de lo que espero que sea una brillante carrera profesional en la que pueda dar solución a multitud de los problemas con los que ahora se encuentran particulares y empresas.

2.- Objetivos

El objetivo principal de este Proyecto Final de Carrera es desarrollar un conjunto de metodologías que permitan desarrollar una aplicación web de gestión de un gimnasio con una serie de particularidades que lo hacen distinto al resto. Entre esas particularidades se encuentra la reserva online a través del sitio web de las distintas actividades ofertadas por el gimnasio, permitiendo un gestión de cobros en concordancia con el consumo efectuado. Otra particularidad a tener en cuenta es la gestión del sistema de accesos a las instalaciones, ya que se realizarán mediante tarjetas RFID para identificar a los distintos socios.

Las pretensiones para llegar a decidir estas u otras características han sido, por una parte, la posibilidad de desarrollar un servicio de forma estándar para que pudiera ser implantado en multitud de lugares diferentes. Por ello se ha procurado que este sistema sea totalmente modular al resto de la aplicación, ya que de esta forma tanto el sistema de reservas, como el de control de accesos, podrían ser implantado en un sitio web ya existente con el menor número de cambios posible. Esto implica que un mismo producto puede ser reutilizado con una tasa de esfuerzo baja, lo que implica aunque el costo de desarrollo sea el mismo, conforme vaya siendo más utilizado su costo será más bajo (tanto de implantación, como de mejora y adaptación). A la larga, debido a su bajo coste, podrá ser implantado por empresas que jamás pensaron que podían permitirse una aplicación de este calibra.

Otro de los motivos para elegir este tipo de gestión de horarios de un gimnasio ha sido la propia comodidad del usuario, ya que debemos romper todas las barreras posibles entre nuestra empresa y el cliente. Así por tanto, si el medio natural donde se desarrolla la gente es Internet, nosotros deberemos ir un paso por delante y tener una presencia en dicho terreno. Y si además de la presencia, el cliente puede tener las mismas posibilidades que otro cliente tiene estando en las mismas instalaciones, mejor todavía. Ese es por tanto el objetivo en ese sentido, intentar que la aplicación web sea lo más parecido posible a la oficina que se puede encontrar en las instalaciones de nuestro gimnasio, y dar al cliente todas las facilidades posibles para informarse y usar nuestros servicios.

Otro factor a resaltar es el tema económico. Esta aplicación pretende ayudar tanto cliente, como al gimnasio a hacer un mejor uso de sus recursos financieros. Por parte del cliente se rompe el modelo clásico de negocio en los gimnasios, este es a grandes rasgos bonos mensuales baratos, pero precios diarios caros. De esta forma se aseguraban el factor psicológico que te hacía adquirir el bono mensual, ya que con el mismo dinero que te costaba ir una semana podías ir todo un mes. Evidentemente, que alguien disponga de un bono mensual no implica que vaya a hacer un uso efectivo de él. Lo que aquí se plantea es romper ese negocio y que el cliente pague por el uso que realmente dé a las instalaciones del gimnasio. Así se evitará pagar ese mes al que no fuimos ningún día, o pagar el bono mensual cuando únicamente fuimos un par de horas, o pagar más cuando se hace un uso intensivo.

Por parte del gimnasio se pueden encontrar ventajas también en el aspecto económico. Una de ellas tiene que ver con la mejor utilización de los recursos materiales de los que disponen las instalaciones. Esto es así porque es muy difícil predecir cuando va a venir más o menos

gente, y a qué actividades mostrarán mayor interés. De esta forma se pretende ir un paso por delante, reservando directamente las actividades que desean los clientes en unas horas y días determinados. Las ventajas a priori son muchas, ya que se puede saber qué salas están ocupadas y cuáles libres, y de esta forma destinar estas a otros usos o actividades. También es posible realizar un mayor aprovechamiento de los profesores que efectúan las distintas actividades, ya que tienen determinadas las horas de inicio y fin de cada actividad. El objetivo por lo tanto es aprovechar al máximo las instalaciones que se tienen, para sacar el máximo beneficio posible.

A decir verdad lo más importante de todo este sistema es la información que el mismo es capaz de generar. Las ventajas para el empresario parecen no tener fin: estadísticas sobre las horas de mayor afluencia, días en los que se efectúan más reservas, profesores que atraen a más gente a sus clases, actividades que mejor consideración tienen entre los alumnos, etc. Toda esta información puede ser reutilizada en beneficio del propio gimnasio y todo su ecosistema (tanto trabajadores, como usuarios, directivos, etc.)

Todo lo comentado hasta aquí vienen a ser las motivaciones generales de la aplicación que aquí se describe, pero profundizando más en el propio proyecto se extraen objetivos más específicos. Uno de ellos es la usabilidad, por lo que se ha hecho un estudio de los avances que ha logrado este campo para lograr acercarse al usuario de forma más clara y transparente, de forma que nuestra aplicación sea fácil de utilizar e intuitiva para los no expertos.

Otro objetivo que se ha tenido en cuenta ha sido la seguridad en Internet. Se ha hecho otro estudio sobre la forma de proteger las aplicaciones web, así como las vulnerabilidades más importantes existentes ahora mismo en internet y la forma de evitar caer en ellas en nuestro código. Este factor es importante para lograr la tan ansiada credibilidad que necesita una aplicación web a los ojos de las personas que lo utilizan y tienen almacenados en ella sus datos personales.

En cuanto a la propia aplicación web, se ha pretendido en todo momento diseñar un sistema sencillo pero altamente configurable, que permita a los administradores cierta flexibilidad a la hora de agregar nuevas noticias, modificar las instalaciones, añadir nuevos usuarios, profesores, etc. Por lo tanto podríamos decir que la aplicación es totalmente autogestionable, ya que una vez implantada en el sistema destino ya no serán necesarios ningún programador y/o diseñador para el mantenimiento de la aplicación.

También se ha tenido especial cuidado en la preparación del servidor, que será el entorno de producción en el cual funcionará nuestra aplicación web. Es por ello que este es un punto en el que debemos poner especial atención tanto en su instalación, como en su configuración. Se tiene especial interés en que sea un sistema plenamente funcional, escalable y seguro en todos los sentidos.

Podríamos resumir por lo tanto que se pretende dar una vuelta de tuerca al modelo de negocio de los gimnasios, haciendo una propuesta de un modelo importado de otros mercados como puede ser el de los hoteles. Para ello se ha pretendido siempre usar tecnologías

novedosas que permitan estar a la vanguardia, y que el paso de los años deteriore lo menos posible la forma en que está pensado todo el sistema.

3.- Conceptos clave

RFID (Radio Frecuency IDentification): Es un sistema para leer y almacenar información en etiquetas RFID. Dichas etiquetas pueden ser de un tamaño tan pequeño que pueden ser implantadas en personas y animales. Para la lectura/escritura no se requiere contacto alguno.

Servidor Web: Es un software que se ejecuta en el lado del servidor, el cual está a la espera de que algún cliente le solicite alguna petición para procesarla, y devolverle el fichero solicitado. También se utiliza para referirse al servidor que ejecuta dicho software.

Aplicación web: Tipo de software que se ejecuta en un servidor web y que permite al usuario interactuar con ella a través de un navegador.

Lector RFID: Dispositivo capaz de leer el contenido de ciertas tarjetas que utilizan tecnología de radiofrecuencia para transmitir los datos. Existen numerosas tecnologías que se diferencian por la frecuencia a la que operan, y que permiten leer a mayor o menor distancia.

Navegador: Aplicación informática que permite consultar e interactuar con documentos de hipertexto servidos por un servidor web. Ejemplos podrían ser Chrome, Safari y Firefox.

Arquitectura cliente / servidor: Es un tipo de computación en el que el cliente genera una petición al servidor, el cual genera una respuesta para éste. Aunque es normal que el cliente y servidor operen a través de una red, es perfectamente compatible que ambos se encuentren en la misma máquina.

URL: Identificador único de un recurso alojado en Internet. Tiene la siguiente forma: PROTOCOLO://MAQUINA:PUERTO/RECURSO. Es normal que se refiera únicamente al protocolo http, pero puede referirse a cualquier otro existente.

Login: Acción de iniciar sesión mediante unas credenciales de acceso que normalmente suele ser un nombre de usuario y una contraseña.

Logout: Es la acción contraria al login, por lo que permite volver al estado de usuario anónimo.

Portal web: Sitio web que aglutina una variedad de servicios web diferentes.

Demonio (Linux/Unix): Es un tipo de software que se caracteriza por ejecutarse sin interacción con el usuario. Su configuración se realiza mediante la carga en el arranque de directivas existentes en un fichero de texto.

Metaetiquetas: En el lenguaje HTML existen este tipo de etiquetas que permiten agregar cierta información sobre el propio documento en el que se encuentran declaradas. Son usadas

por parte de los buscadores para identificar autor, palabras clave, descripción, caducidad de la caché, etc. Se conocen también por su nombre en lengua inglesa “metatags”.

Script: Es un fichero que contiene órdenes que para ser ejecutadas necesitan de un intérprete, por lo que no existe la compilación como en otro tipo de ficheros ejecutables. No existe un lenguaje común en el que deban ser escritas dichas órdenes para ser consideradas un script.

Log: Se refiere a un registro de sucesos (como pueden ser accesos o errores) guardados en un fichero por parte de un demonio. En nuestro caso nos interesará los logs (también llamados registros) generados por el servidor web.

Autenticación: Proceso mediante el cual se comprueba la identidad de una persona a través de unas credenciales de acceso, que normalmente suelen ser un nombre de usuario y una contraseña. Este proceso determina también los privilegios de los usuarios que se han autenticado.

Depuración: Una vez desarrollado un software, es necesario identificar y corregir errores en el código. En inglés se conoce como debugging, ya que es el nombre que aparece en numerosos entornos de desarrollo.

Caché (web): Es una funcionalidad que se sitúa entre un servidor web y un cliente, que permite guardar copias del contenido servido por el servidor web para servirlo a clientes puntales. Las motivaciones principales son reducir la latencia en satisfacer la petición del cliente, y reducir el tráfico de red.

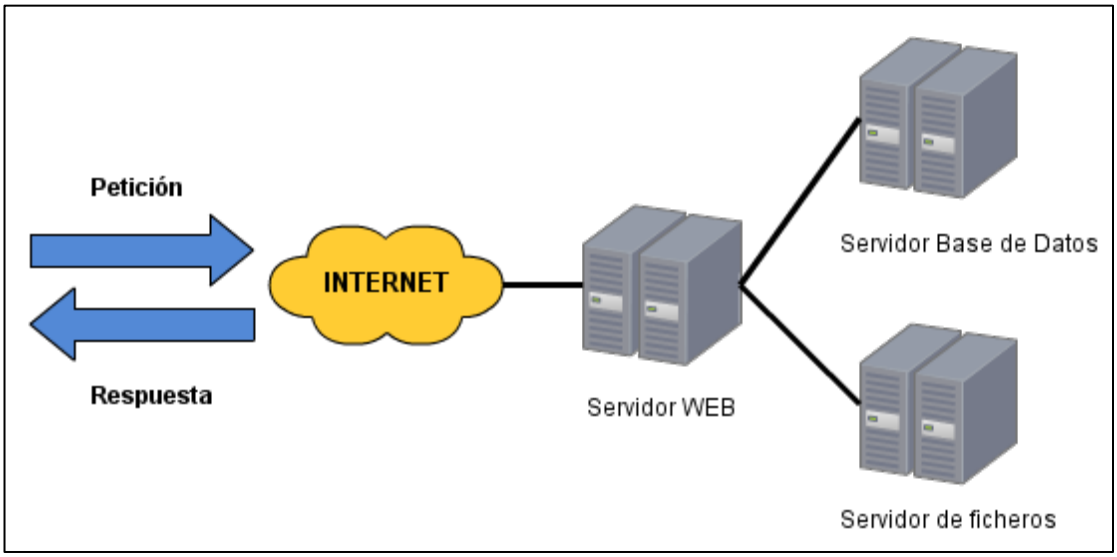
Monitorización: Proceso de supervisión de los servicios que ofrece un equipo servidor. El motivo de esta monitorización es evaluar el funcionamiento y el rendimiento de dichos servicios.

4.- Elementos del sistema

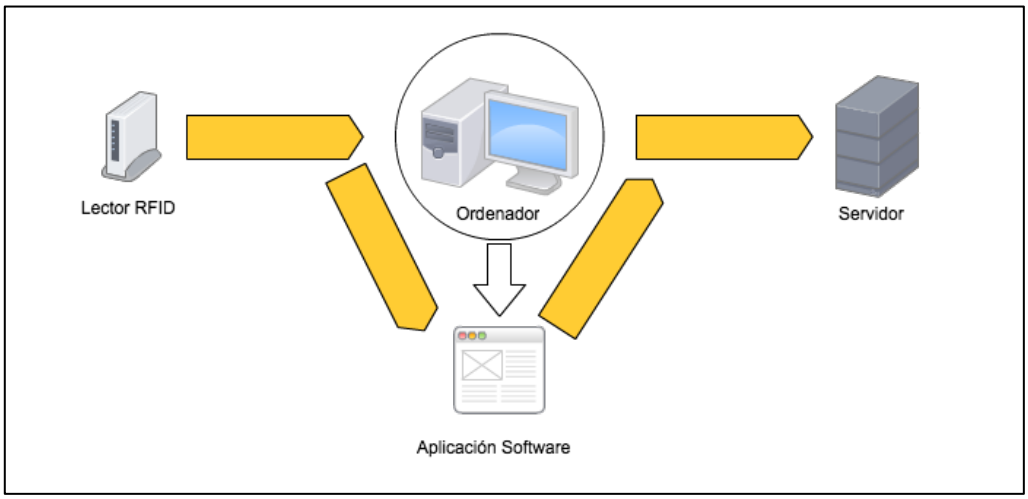
Ahora veremos cómo será el esquema que se utilizará para implementar todo el sistema. Se pretende dar una visión general, puesto que en posteriores apartados se tratará en profundidad cada uno de los apartados.

En primer lugar, y al tratarse de una aplicación web requeriremos de dos elementos bien diferenciados. Por un lado la propia aplicación web y por otro el entorno en el que se ejecutará dicha aplicación. Este entorno será un servidor que deberá ser capaz de ejecutar nuestra aplicación con los requisitos que esta necesite, de forma que se forme un conglomerado seguro, fiable y eficiente.

Inicialmente, para la ejecución de la aplicación web se necesitarán tres servidores distintos, cada uno de ellos dispondrá de una utilidad bien diferenciada. Los tres servidores serán: uno para servir la web, otro para servir la base de datos, y por último otro para servir los distintos ficheros. Estos servidores deberán ser accesibles desde el exterior para que los usuarios dispongan de acceso a los mismos. A continuación se puede ver un diagrama:



Una vez preparado el entorno para ejecutarse la aplicación web, deberemos diseñar la forma en que interactuará el sistema de accesos. Para ello se usará un lector de RFID conectado a un ordenador. Este lector transmitirá vía USB al ordenador los datos de la tarjeta leída. En dicho ordenador habrá una aplicación esperando la lectura de las tarjetas RFID. Dicha aplicación será la encargada de transferir los datos al servidor web, que será el último responsable de tratar los datos para insertarlos en la base de datos y realizar las operaciones convenientes.


Lector RFID
Ordenador
Servidor
Aplicación Software

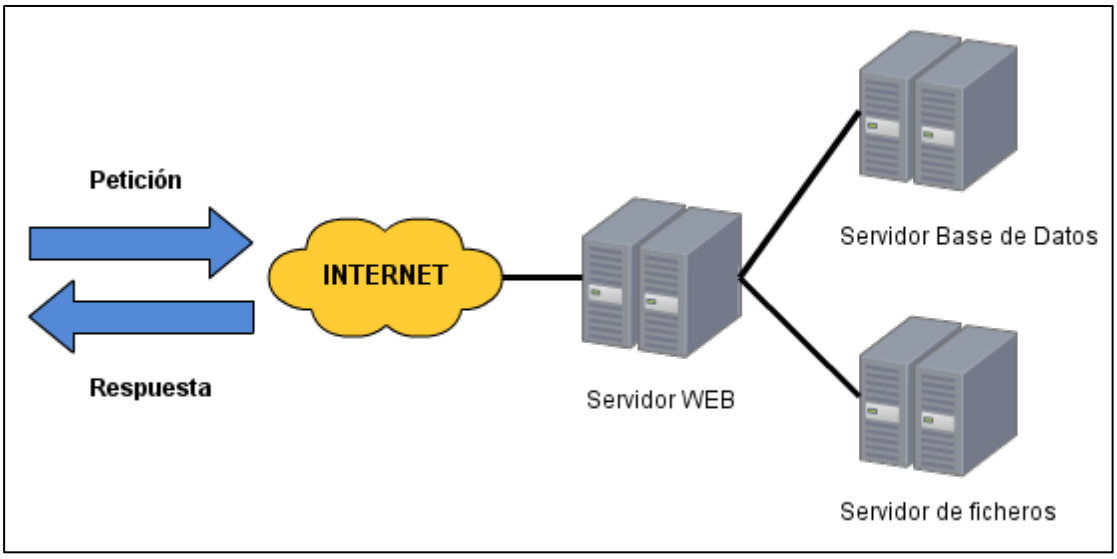
5.- Servidor

En todo el sistema que vamos a desarrollar va a haber un elemento que va a tener una importancia especial. Este elemento va a tener funciones distintas a cualquier otro, esto por un lado significa que habrá que prestar especial atención a su configuración, seguridad e integridad.

Este elemento de vital importancia, y entre las tareas que va a realizar se encuentra todos las operaciones computacionales, la gestión de los datos, almacenamiento de los distintos ficheros, modificación de los datos, etc.

Dado que el servidor realiza funciones completamente diferenciadas, parece conveniente separa dichas funcionalidades en distintos servidores, como pueden ser el servidor web, el servidor de ficheros y el servidor de base de datos.

Por motivos de simplicidad, y dado que el servidor web y el servidor de ficheros comparten algunas funcionalidades, en nuestro caso se van a unificar ambos en uno solo. Esta carencia se va a compensar colocando los ficheros que deseemos compartir en el directorio de publicación del servidor web.



Por lo tanto, a partir de este momento, las dos partes principales son el servidor web y el servidor de base de datos. La primera se encargará de servir las páginas estáticas (HTML), los distintos contenidos alojados, y las páginas dinámicas (PHP, ASP…). Entre los contenidos que podemos incluir en el servidor de ficheros, incluiríamos las imágenes que aparecen en las distintas páginas, documentos PDF disponibles para la descarga, y los vídeos disponibles.

La segunda parte se encargará de servir y almacenar a nuestra aplicación los distintos datos que requiera. Dichos datos serán usados para generar las páginas dinámicas y guardar los todos los datos relacionados con el proceso de negocio. Entre estos datos podríamos incluir los datos de nuestros clientes, usuarios, productos, horarios, reservas, instalaciones, etc.

5.1.- Sistema Operativo

El servidor que pretendemos montar debe tener una serie de características (tanto software como hardware), que permitan realizar todas las tareas requeridas por la aplicación de forma estable y eficiente. Hay que tener en cuenta que estos sistemas deben estar preparados para soportar una gran carga, ya que puede haber gran cantidad de usuarios conectados a nuestro servidor realizando todo tipo de operaciones.

En primer lugar debemos tener un sistema claramente enfocado a trabajar con redes de alta eficiencia, tanto por la cantidad de conexiones que puede llegar a tener que soportar, como por la carga y cantidad de datos de las mismas. Por otro lado el sistema deberá soportar una gran carga computacional para procesar todas las tareas del proceso de negocio.

En el mercado de los sistemas operativos domésticos existe un reducido número de opciones disponibles a la hora de elegir para nuestro equipo personal. Básicamente estas opciones se reducen a 3 solamente: Windows, Linux y Mac OS X.

Es posible utilizar algunos de estos sistemas operativos domésticos para nuestro propósito, por el simple hecho de que no son tan domésticos como parece. En primer lugar vamos a descartar la opción de Windows, ya que el rendimiento y la estabilidad son muy bajos para un entorno de producción a gran escala como va a ser el nuestro. Si necesitamos un servidor con Windows, deberemos usar las versiones desarrolladas especialmente para estos fines. Actualmente estas versiones son todas las ediciones de Windows 2000 excepto la Professional, y todas las ediciones de Windows 2003 y 2008.

Linux y Mac OS X comparten gran cantidad de características como una base Unix, ya que Linux es un clon de este, y Mac OS X es un derivado de BSD. Por lo tanto, aunque hemos clasificado a estos sistemas como domésticos que pueden ser usados con carácter profesional, en realidad sería todo lo contrario, ya que son sistemas profesionales que pueden ser usados como domésticos.

Es cierto que existen ciertos sistemas operativos cuyo uso es prácticamente profesional, entre los que destacan Solaris, BSD (con sus distintas variantes), Novell, QNX y Plan 9. Si pretendemos utilizar alguno de estos sistemas deberemos documentarnos convenientemente sobre las distintas opciones de configuración que deberemos llevar a cabo para disponerlos como servidores de producción.

En el caso que vamos a tratar para el presente proyecto nos interesaremos por un sistema operativo que sea fácilmente entendible por el lector de la presente memoria, por lo que

recurriremos a un sistema operativo "de uso doméstico" para que las operaciones que se describen sean comprendidas con mayor rapidez, por lo que vamos a descartar todas las opciones "profesosionales". Esto reduce las opciones a 3: Windows, Linux y Mac OS X.

De entre ellas nos quedaremos con Linux por ser la más flexible en cuanto a la personalización del mismo, en cuanto a la libertad de elección de la distribución a utilidad, su libertad de uso sin necesidad de pagos, su orientación a servidores, su fiabilidad y estabilidad, la gran cantidad de documentación disponible y su relativamente poca especialización.

En el mundo Linux podemos optar por gran cantidad de distribuciones, desde las que se distribuyen gratuitamente, hasta las que pueden llegar a costar miles de dólares. Entre las opciones que deberíamos tener en cuenta se encuentran Debian, Red Hat Enterprise Linux, Fedora, SuSE, Gentoo, CentOS,…

De entre todas ellas vamos a optar por una gratuita, con un ciclo de desarrollo largo entre versión y versión para asegurar tanto la estabilidad, como los problemas de actualización entre versiones. Los principales son Debian y CentOS (que proviene de las fuentes de Red Hat)

Las imágenes de los sistemas operativos se pueden encontrar aquí:

Debian: http://www.debian.org

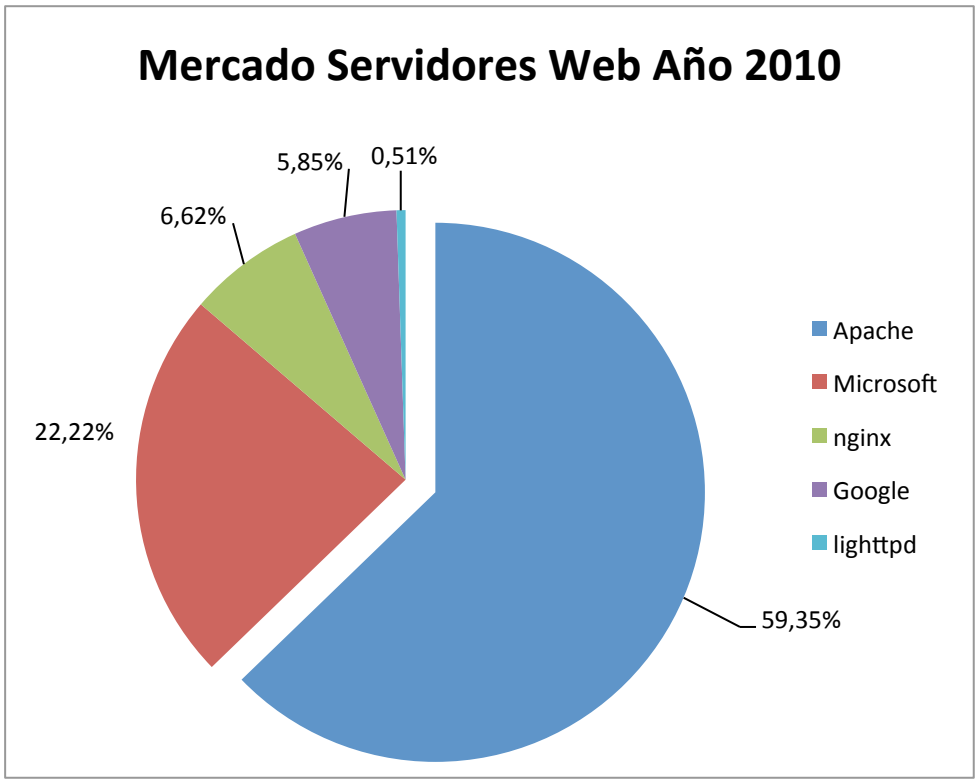
CentOS: http://www.centos.org

5.2.- Servidor Web

De entre todos los procesos que van a llevarse a cabo en el servidor, el más determinante de todo ellos es el servidor web. Esta será la pieza clave de todo nuestro entramado, por lo que deberemos prestar especial atención tanto a la elección, como a la configuración de dicho elemento.

Al igual que en el mercado de los sistemas operativos de servidor, en el caso de los servidores web deberemos ver las opciones de las que podemos disponer. Esta decisión afortunadamente será más sencilla ya que disponemos de pocas opciones, aunque no por ello las opciones deben ser de menor calidad, ya que es un mercado muy maduro.

En primer lugar vamos a comprobar las opciones que nos da el mercado, y las características de cada uno. La compañía *Netcraft* nos ofrece en su web los datos de penetración de mercado de cada servidor web, los cuales se muestran a continuación (concretamente del año 2010) :



Los datos del gráfico hablan por sí solos, ya que la opción mayoritaria agrupa a más de la mitad de los servidores web del mundo. Dicha opción es Apache con su servidor httpd, que logra casi el triple servidores que la segunda opción en el mercado. La segunda opción es el servidor de Microsoft (IIS). El resto de opciones son prácticamente despreciables ya que el porcentaje de ellos es muy bajo.

Hay que decir que en este análisis únicamente aparecen servidores web genéricos, que aunque no son los únicos que existen en el mercado, encajan perfectamente con nuestros requerimientos. En principio cualquiera de ellos podría servirnos, pero vamos a analizarlos más a fondo en el siguiente apartado.

5.2.1.- Posibles configuraciones

1. Apache

Este servidor web es el servidor web más utilizado desde abril del 1996, y en realidad se refiere a un proyecto llamado “httpd” de la Apache Software Fundation, cuya última versión en el momento de redactar este documento era la 2.2 (aunque existe una versión preliminar de la versión 2.3)

Apache utiliza varias licencias para publicar tanto el software como la documentación del mismo. Actualmente la versión de la licencia es la “Apache License, versión 2.0” que es

compatible con la licencia GPL. Permite ser utilizado para cualquier propósito, distribuirlo, modificarlo, y distribuir dichas modificaciones, pero no obliga a que las versiones derivadas de este software deban ser distribuidas usando la misma licencia.

Este servidor puede ser utilizarlo en diversos sistemas operativos modernos, incluyendo todas las versiones de UNIX (BSD, Solaris, …), Linux, y Windows NT entre otros.

Su arquitectura modular permite el uso de módulos para extender o reducir sus funcionalidades, lo que permite una gran flexibilidad.

Aunque su configuración pueda ser complicada (ya que no existe aplicación gráfica), existen gran cantidad de documentación tanto en los manuales oficiales, como en páginas relacionadas con el software libre.

2. Microsoft

Este servidor es conocido como IIS (Internet Information Server) y actualmente forma parte de la familia de sistemas operativos para servidor de Windows (2000 Server, 2003 y 2008), aunque existe una versión limitada para Windows XP.

Al igual que Apache, el servidor IIS tiene una estructura modular que permite añadirle funcionalidades para procesar documentos dinámicos como ASP, PHP, etc. En cuanto a las opciones que vienen incluidas por defecto en la versión 7.0 se encuentra IIS, Visual Web Developer, SQL Server, .NET Framework y Silverlight Tools.

Su configuración se realiza mediante una aplicación gráfica, por lo que ésta se simplifica mucho con respecto a Apache. A pesar de esto las configuraciones son mucho menos personalizadas, y se dedican a personalizar temas mucho más superfluos.

La configuración típica en cuanto al contenido dinámico es usar el lenguaje ASP o ASP.NET, aunque también puede extenderse mediante extensiones para que sea capaz de procesar ficheros PHP o Perl.

Aunque IIS no tiene un coste implícito ya que se distribuye gratuitamente, si que tenemos que tener en cuenta que habrá que costear el precio de la licencia del sistema operativo Windows Server, cuyo precio empieza en los 999 dólares estadounidenses.

5.2.2.- Servidor Web Apache

En el desarrollo del presente proyecto se va a utilizar el servidor web Apache, por lo que en los siguientes apartados se va a dejar de nombrar el servidor IIS. Por este motivo, todos los ejemplos a partir de este momento se van a realizar utilizando el servidor Apache, concretamente la versión 2.2

Se intentará dar una visión global de todos los aspectos de este servidor, ya que pueden ser de vital importancia a la hora de pulir ciertos detalles que pueden ser importantes para el funcionamiento de nuestra aplicación.

Entre los aspectos que se van a tratar empezaremos en primer lugar la forma de obtener Apache, que como hemos visto en el apartado anterior está disponible bajo la licencia Apache, que permite el uso y distribución libre de la aplicación.

Una vez obtenido, el siguiente paso será conseguir que todo ese entramado de ficheros consiga realizar la función que de él necesitamos. Dicho paso será la instalación, de la cual hay diversas variantes dependiendo del método elegido.

También se va a realizar un análisis de los distintos directorios que forman la instalación de Apache, haciendo especial hincapié en los contenidos y funciones que desarrollan cada uno de ellos.

Si todo ha ido correctamente, deberemos aprender a iniciar y detener Apache para empezar a disfrutar del servidor. Veremos cómo iniciarlo automáticamente para que se arranque al inicio de nuestro servidor.

Un aspecto fundamental serán los ficheros de configuración, donde podremos personalizar nuestra instalación para que funcione acorde a nuestros requisitos, tanto de funcionalidad como de seguridad y personalización.

Como se ha comentado en apartados anteriores, Apache puede extender sus funcionalidades mediante distintos módulos. Estas funcionalidades son las que se verán en este apartado, así como la forma de cargar e instalar los diferentes módulos.

5.2.2.1- Formas de obtener Apache

Para conseguir Apache tenemos varias opciones dependiendo de nuestras necesidades y de los conocimientos que tengamos. Existe desde la versión personalizada para nuestra distribución preparada para funcionar, hasta la versión que incluye únicamente el código fuente que deberemos compilar para lograr que funcione.

En primer lugar vamos a comentar la opción más sencilla de instalar, y por lo tanto más rígida en cuanto a personalización y configuración. Se trata de instalar el paquete que incluye nuestra distribución, que suele encontrarse en los repositorios oficiales. La instalación en este caso es realmente sencilla, ya que podemos tener instalado Apache con un par de clicks o en un único comando.

En mi caso mi máquina corre una versión de Ubuntu, si usa otra distribución mis palabras le resultarán igualmente familiares, pero deberá adaptar el proceso a su distribución. Este es el ejemplo a seguir en Debian y Ubuntu:

```
# sudo apt-get install apache2
```

Hay que decir que en este ejemplo únicamente instalaría el servidor web Apache, y dejaría de lado el servidor de base de datos y el modulo php. Para ello habría que instalar también mysql-server, php5, libapache2-mod-php5, php5-mysql.

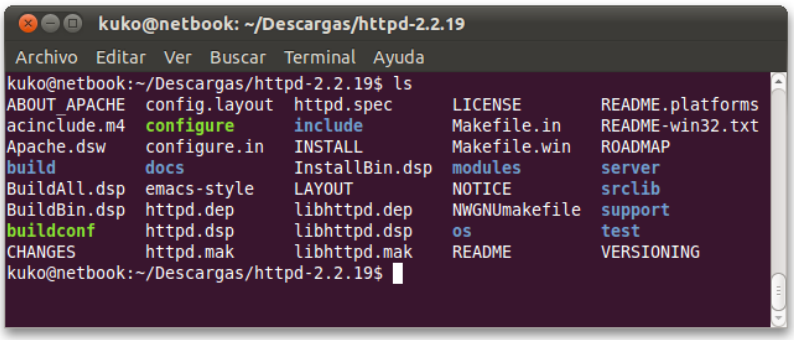
Otra opción sería realizar la instalación a partir de los binarios disponibles en el sitio web de Apache. Desgraciadamente no existe una versión en este formato para sistemas Linux, por lo que no se ajusta a nuestros requisitos. La instalación en este caso consistiría en descargar los binarios de la web de Apache, descomprimir el fichero proveniente de la descarga, y ejecutar el binario.

La opción que nos permite un mayor grado de personalización es compilar Apache a partir del código fuente que podemos encontrar en la web oficial. El lugar concreto en el que se puede encontrar es http://apache.org/download.cgi, donde podremos encontrar las distintas versiones (evitaremos versiones Alpha y Beta). Elegiremos aquellas descargas denominadas con “Unix Source”.

El primer paso será descomprimir todos los ficheros que se encuentran en el comprimido que nos hemos descargado (En nuestro caso con la versión .tar.gz). Para ello desde la consola nos situaremos en el directorio donde se encuentra el comprimible y ejecutamos el siguiente comando (importante sustituir las x por la versión concreta que corresponde con el nombre del fichero):

```
$ tar xzvf httpd-x.x.xx.tar.gz
```

Esto creará un nuevo directorio, en cuyo interior tendremos el contenido del fichero que acabamos de descomprimir. Para acceder deberemos cambiar de directorio con el comando cd y a continuación el nombre del directorio. Si hacemos un listado con el comando ls podemos ver el siguiente resultado:



Cabe destacar sobre todo el script “configure” que será el que usaremos a continuación. Si

deseamos utilizar la configuración por defecto simplemente deberemos ejecutarlo, pero hay que destacar que acepta diversos parámetros que permiten personalizar la instalación en base a nuestras necesidades. Esto resultará familiar para aquellos acostumbrados al mundo Unix / Linux.

El otro script llamado "buildconf" se usará en el caso de que nuestra distribución haya sido descargada de un CVS (sistema de control de versiones). Si no tiene muy claro que es un CVS, probablemente es que no lo haya usado para descargar Apache, aun así si desea más información será interesante buscarla en la documentación oficial de Apache.

Como hemos empezado a comentaren párrafos anteriores, para preparar la compilación con las opciones por defecto, el comando que deberemos ejecutar será el siguiente:

```
$ ./configure
```

El proceso durará un par de minutos, dependiendo de la máquina sobre la que se ejecute. Hay que mencionar algunas opciones de personalización alternativas al comando anterior. Como hemos dicho esta personalización se realiza mediante parámetros. A continuación vamos a mostrar algunos:

Para modificar el directorio donde Apache será instalado, hay que utilizar el parámetro --prefix de la siguiente forma: ./configure --prefix=directorio

Es posible que nos interese desactivar algunas de las características o módulos que están activados por defecto. La lista módulos es la siguiente: mod_actions, mod_alias, mod_asis, mod_auth, mod_autoindex, mod_access, mod_cgi, mod_cgid, mod_charset_lite, mod_dir, mod_env, mod_imagemap, mod_include, mod_log_config, mod_mime, mod_negotiation, mod_setenvif, mod_status, mod_userdir. La forma de desactivarlos será con el modificador --disable-nombreModulo. Por ejemplo: --disable-mod_autoindex.

También es posible indicar el caso contrario, activar un módulo de los que no están activados por defecto. Para lo cual habrá que modificar el parámetro "disable" por "enable". Por ejemplo, para incluir el módulo "expires" añadiremos --enable-expires.

Debido al gran número de posibilidades que existen a la hora de efectuar la configuración, no se van a tener en cuenta todas. En cualquier caso, si todo ha ido correctamente estaremos en condiciones de comenzar con la compilación. Para ello ejecutaremos el comando siguiente:

```
$ make
```

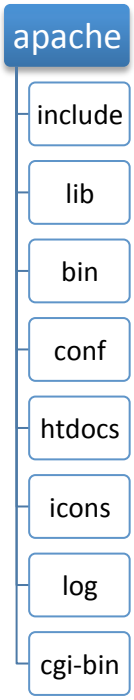
Esto compilará todas las partes de Apache, y ahora sólo quedará ejecutar la instalación. Para ello bastará con ejecutar el comando siguiente:

```
$ make install
```

En este punto ya tendremos Apache instalado y podremos cargar en el navegador la dirección http://localhost para comprobar que todo ha ido correctamente.

5.2.2.2- Estructura de directorios

Apache dispone de una estructura de directorios específica que se encuentra a partir del directorio donde se encuentra instalado. Este directorio puede ser distinto en cada distribución, y en mi caso se encuentra en /etc/apache2. Si usted ha realizado el proceso de compilación puede modificar esta ruta a su voluntad, indicando la ruta en el atributo prefix en el script de "configure".



Aquí puede verse una representación de la estructura típica de directorios que representa Apache una vez instalado a partir del código fuente.

A continuación se hace una breve explicación del contenido y utilidad de cada uno de los directorios.

Algunos de los directorios que se puede encontrar son los siguientes:

- **Include:** Contiene archivos de cabecera que son necesarios si desarrollar aplicaciones web integradas con Apache.
- **Lib:** Contiene las librerías que necesita Apache y sus utilidades para ejecutarse correctamente.

- **Bin:** Contiene todos los programas ejecutables que se distribuyen con Apache, como ab, apachectl, httpd, htdigest…
- **Conf:** Aquí se encuentran los ficheros de configuración de Apache. Estos son los que deberemos modificar para ejecutar Apache a nuestro gusto.
- **Htdocs:** Aquí se encuentra la raíz de documentos que usará Apache como directorio de publicación. Este directorio puede ser modificado en los ficheros que configuración.
- **Icons:** Este directorio contiene distintos iconos que se utilizan para listar directorios en el caso en que no exista un archivo predeterminado.
- **Logs:** Es el lugar donde se encuentran todos los registros de los sucesos que ocurren en el servidor. Puede configurarse a nuestro gusto para decidir qué tipo de información se guardará.
- **Cgi-bin:** Si utilizamos scripts CGI, éste es el lugar donde deberán ubicarse este tipo de aplicaciones. Este directorio puede modificarse también en los ficheros de configuración.

Esta estructura es común a todas las configuraciones instaladas por el modo manual (configuración, compilación e instalación), pero puede verte modificada en los casos en los que la instalación proviene de un paquete personalizado. Aunque la función de los distintos directorios será la misma, lo que cambiará será la ubicación de los mismos.

Por ello si su instalación ha sido realizada desde los repositorios utilizando un paquete específico para su distribución debería informarse de la localización de sus directorios en los documentos distribuidos con su paquete Apache o en la web oficial de su distribución.

5.2.2.3- Iniciar y detener Apache

Una vez Apache ya está correctamente instalado en nuestro sistema, podemos arrancarlo para hacer las primeras pruebas. Por defecto, el servidor se distribuye con una configuración básica que permite que lo utilicemos desde el mismo momento de instalarlo.

Esta configuración no suele ser del agrado de ningún administrador, debido a la carencia de funcionalidades por un lado, y por motivos de seguridad por otro. Por lo tanto habrá multitud de opciones que podremos configurar para adecuarnos a nuestro ecosistema incluyendo todas las opciones posibles tanto de rendimiento, seguridad, directorios, etc.

En el directorio bin comentado en el apartado anterior, se encuentra un binario llamado apachectl, que será el encargado de arrancar Apache. En mi distribución concreta este binario se encuentra en la ruta /usr/sbin, pero suele ser común que esta ruta se encuentre en el PATH del sistema, por lo que probablemente no debamos conocerla, ya que será el sistema el encargado de buscar el ejecutable.

Por lo tanto para iniciar Apache será tan sencillo como ejecutar lo siguiente:

```
#apachectl start
```

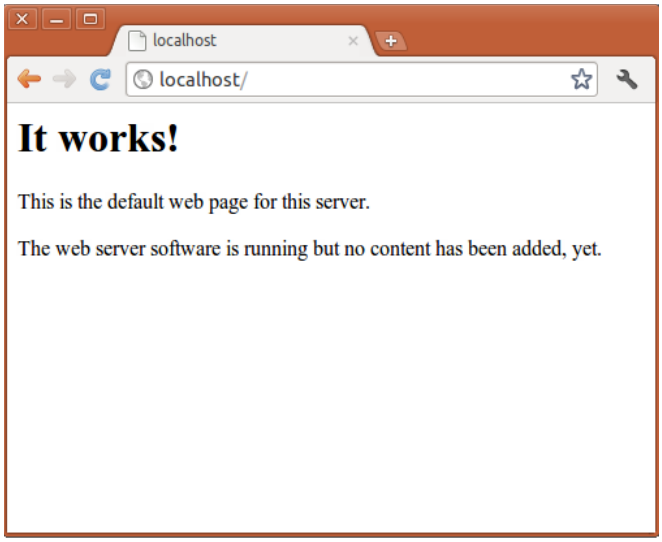
Es posible que si los binarios provienen de una distribución, éstos sean capaces de ejecutarse como servicio, por lo que debería evitarse iniciarlo de la forma que se ha comentado anteriormente. En Debian / Ubuntu, debe ejecutarse el siguiente comando:

```
#service apache2 start
```

En un entorno de producción, Apache debería ejecutarse al iniciar la máquina, ya que en caso de reinicio el tiempo de parada debería ser el mínimo posible. Esto es porque en un reinicio automático no haría falta loguearnos con nuestro usuario y contraseña para que Apache siga funcionando, si no que en el arranque ya comenzaría a funcionar. Otro motivo es que puede que en un entorno de producción deseemos que el servidor no se ejecute un entorno gráfico, por lo que si es incluido en el nivel 3 esto no será necesario.

```
# ln –s /etc/rc.d/init.d/httpd /etc/rc.d/rc3.d/httpd
```

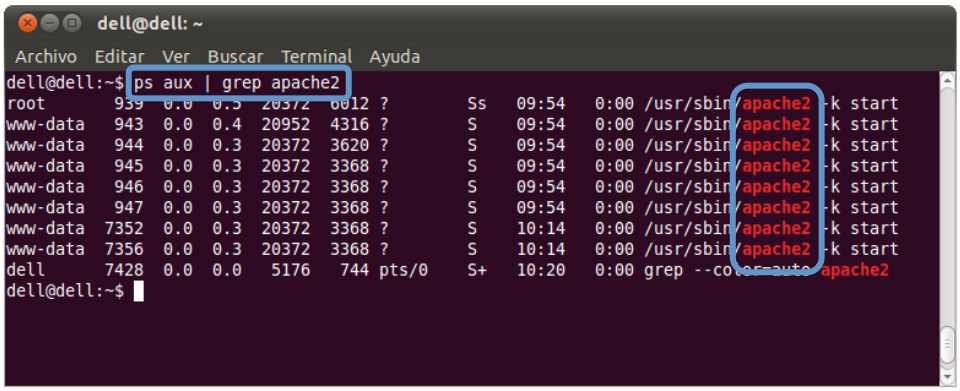
Se puede comprobar una vez iniciado el servidor, si este efectivamente se ha iniciado cargando la URL http://localhost en nuestro navegador. El resultado debería ser similar al siguiente en un sistema recién instalado:



Otra opción para comprobar si todo ha ido correcto es ejecutar el siguiente comando:

```
#ps aux | grep httpd
```

Esto da un listado de los procesos que están actualmente corriendo en el sistema (que es el resultado de la orden "ps") y filtra aquellos que se llamen httpd. En el caso de mi distribución el proceso se llama "apache2". El proceso de filtrado se realiza mediante el comando grep. Aquí se puede ver un ejemplo del resultado que podemos esperar:



De un motivo similar al proceso de inicio de Apache, para detener Apache habrá que tener en cuenta si estamos ejecutándolo mediante un binario, o como un servicio.

Para el caso de que estemos ejecutándolo como un binario, el comando será el siguiente:

```
#apachectl stop
```

Como apunte, decir que en caso de que no nos encontremos en el directorio bin de Apache, deberemos añadir dicho directorio en el PATH del sistema.

Si por el contrario se está ejecutando como un servicio, deberemos hacerlo de la siguiente forma:

```
#service apache2 stop
```

Ya hemos visto los dos comandos básicos de Apache, que son el inicio y la detención del mismo, pero hay un tercer comando que acepta, que es el reinicio. Para ello, simplemente sustituiremos en los comandos anteriores el argumento "stop" por "restart".

Una vez visto que Apache funciona correctamente, podemos comprobar los registros que el propio Apache almacena en los ficheros de log. Podemos encontrar uno que se encarga de los

accesos y otro de los errores. Para ello deberemos conocer cuál es el directorio "log" de nuestra instalación (en mi caso se encuentra en /var/log/apache2). Con estos comandos podemos listar los últimos sucesos de Apache:

```
# tail –f /var/www/apache2/access.log

# tail –f /var/www/apache2/error.log
```

El primer caso sería para los accesos normales, y el segundo para los errores.

5.2.2.4- Ficheros de configuración

La configuración de Apache se realiza mediante edición de ficheros de texto plano, al igual que suele ocurrir con la configuración de la totalidad de los demonios Unix / Linux. Dicha edición de ficheros de texto plano consiste en la colocación de directivas en unos ficheros concretos destinados a la configuración.

Si prestamos atención a la estructura de directorios de Apache que se ha mostrado en apartados anteriores, veremos que existe un directorio llamado "conf". El principal fichero incluido en dicho directorio se llama "httpd.conf", aunque también es posible que se llame también "apache2.conf". A pesar de que estos sean los ficheros principales pueden incluirse numerosos ficheros a la configuración mediante el uso de la directiva `Include nombrefichero`, que permite cargar el fichero adicional que se especifique como parámetro.

En el interior de los ficheros de configuración se encuentran gran cantidad de líneas que comienzan con el símbolo #. Dichas líneas son ignoradas al leer el fichero dado que son comentarios, la gran mayoría de ellas son explicaciones de las directivas o ejemplos de uso que pueden ayudarnos a comprender mejor su funcionamiento.

Es posible que en la modificación de dichos ficheros generemos un error en los mismos, debido por ejemplo al mal uso de las directivas, o a referencias inexistentes. Apache pone a nuestra disposición una herramienta que permite verificar si el fichero de configuración es correcto sin necesidad de iniciar o reiniciar el servidor. Dicha herramienta debe usarse de la siguiente forma:

```
$ apache2ctl configtest
```

Ahora vamos a comentar la forma de modificar la configuración de Apache, para ello se mencionan a continuación las directivas que se deben incluir (o modificar) en los ficheros antes mencionados:

Port: Aquí se indica el número de puerto TCP por el que Apache se pone a la escucha para recibir peticiones. El valor por defecto es 80. Debe tener en cuenta que para habilitar los puertos por debajo del 1024 debe poseer permisos de administrador.

Ejemplo: Port 8080

User y Group: Estas directivas funcionan por separado por están muy relacionadas. La primera indica qué nombre de usuario (UID) debe utilizar. En el segundo caso indica qué nombre de grupo (GID) debe utilizar. Estos valores son usados cuando Apache genera nuevos procesos hijo para atender nuevas peticiones, ya que los UID y GID de estos procesos serán los aquí indicados.

Ejemplo: User Servidor
Group Server

ServerAdmin: En caso de que el servidor genere un error en la página que está mostrando, se mostrará una dirección de correo electrónico por si el usuario desea contactar con el administrador. Dicha dirección de correo electrónico se indica en esta directiva.

Ejemplo: ServerAdmin info@midominio.com

DocumentRoot: Aquí se indicará la ruta del directorio de publicación de Apache de donde se leerán los ficheros que solicite el usuario. Todos los subdirectorios de esta ruta se consideran incluidos. Nunca almacene ficheros comprometidos en esta ruta, puesto que serán accesibles desde el exterior.

Ejemplo: DocumentRoot /var/www/html

UserDir: Configura Apache para que considere un directorio de publicación individual en un subdirectorio concreto del directorio personal de los usuarios. Es común utilizar “public_html”, por lo que los usuarios podrían dejar ahí sus ficheros y verlos a través de Apache desde http://miservidor/~usuario/.

Ejemplo: UserDir public_html

DirectoryIndex: Indica un fichero (o conjunto de ellos) que serán enviados al cliente en caso de que este solicite únicamente una ruta relativa a un directorio y no a un fichero en completo. El

orden en el que se irán buscando dichos ficheros será el mismo en el que estén escritos en la propia directiva DirectoryIndex.

Ejemplo: DirectoryIndex index.html

AccessFileName: Especifica el nombre de un fichero de configuración que podrá establecerse en los propios directorios, y que afectará únicamente al contenido del mismo. Lo común es utilizar el fichero ".htaccess", si desea utilizar otro fichero recuerde protegerlo del exterior mediante la directiva Files que encontrará más abajo.

Ejemplo: AccessFileName .htaccess

Order, Deny, Allow: Estas tres directivas sirven para controlar el acceso al ámbito en el que se encuentran especificados. De esta forma Order especifica el orden en el que serán aplicadas las siguientes directivas. Hay que decir que en el momento exista una coincidencia, denegará o aceptará el acceso a dicho recurso (dependiendo de si la coincidencia es Deny o Allow)

```
Ejemplo: Order allow,deny
         Allow from 192.168.0.100
         Deny from all
```

<Directory *> </Directory>: Permite establecer una serie de directivas específicas para un directorio en particular. Este grupo de directivas se incluyen entre la directiva de apertura y la de cierre. Los directorios a los que se aplica se pueden indicar mediante una expresión regular usando en su lugar la directiva DirectoryMath.

```
Ejemplo: <Directory />
                DirectoryIndex inicio.php
         </Directory>
```

<Files *> </Files>: Permite establecer una serie de directivas específicas para un grupo de ficheros particular. Este grupo de directivas se incluyen entre la directiva de apertura y la de cierre. Los ficheros a los que se aplica se pueden indicar mediante una expresión regular usando en su lugar la directiva FilesMath.

```
Ejemplo:  <Files *.ini>
                Order deny,allow
                Deny from all
          </Files>
```

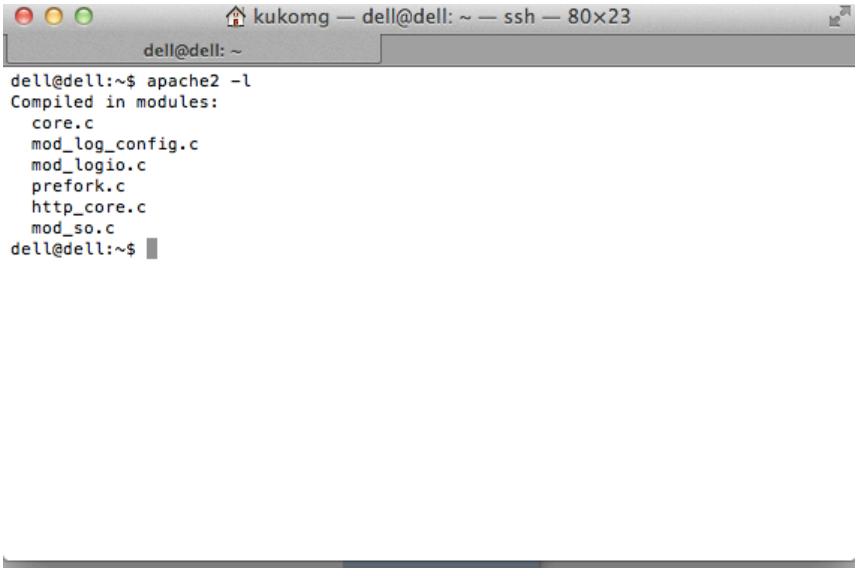
Hay que recordar que después de realizar algún cambio en el fichero de configuración hay que guardar su contenido con el editor, y reiniciar el servidor Apache para que lee la nueva configuración.

5.2.2.5- Módulos Apache

Apache es un servidor completamente modular que permite que el núcleo del servidor únicamente ejecute aquellas tareas más básicas. La ventaja de este tipo de funcionamiento es que se puede extender el funcionamiento del servidor mediante módulos que se pueden cargar, pero también descargar algunos de los que vienen activados por defecto.

Existen dos tipos de módulos para Apache, el primero de ellos se incluye directamente en el proceso de compilación de Apache, y los segundos consisten en una serie de bibliotecas dinámicas que se puede cargar diversos módulos durante el proceso de arranque del servidor.

En primer lugar deberemos conocer cuáles son los módulos que vienen compilados en nuestro Apache, para lo cual deberemos usar el comando "httpd -l" que nos dará el listado que aparece en la siguiente imagen. Notar que como en mi distribución el comando se llama apache2, he utilizado el homólogo "apache2 –l":



Una opción mucho más cómoda es la de permitir el uso de módulos que se cargan como Objetos Dinámicos Compartidos (DSO), para ello es necesario que anteriormente hayamos cargado mod_so. La directiva LoadModule incluida en el fichero de configuración que un módulo sea añadido en el proceso de arranque del servidor.

Módulos de Apache

Core: Contiene las funciones principales de Apache que son incluidas siempre sin necesidad de activar ningún módulo adicional. Este módulo corresponde en realidad con el núcleo del

servidor, por lo que no es posible desactivarlo, aunque sí es posible configurarlo con multitud de directivas. A continuación se muestran algunas de las más relevantes:

`AccessFileName` *`filename`*: Permite distribuir los ficheros de configuración, el cual se especifica con el nombre "filename". Este fichero de configuración será el primero en ser procesado cuando haya una petición. Ejemplo: AccessFileName .htaccess

`AddDefaultCharset On|Off|`*`charset`*: Especifica el conjunto de caracteres con el que se enviará el contenido de respuesta en los casos en los que la respuesta sea del tipo text/pain o text/html. Ejemplo: AddDefaultCharset utf-8

`DefaultType` *`MIME-type`*: El tipo MIME que sea indicado en esta directiva es el que será enviado en el caso de que no se pueda determinar cuál es el tipo MIME del documento que se va a servir. Ejemplo: DefaultType text/plain

`<Directory` *`ruta`*`>…</Directory>`: Permite especificar un ámbito de aplicación a las directivas incluidas en su interior. De esta forma estas directivas únicamente serán aplicables a este directorio. Ejemplo: <Directory /usr/local/htdocs> </Directory>

`DocuementRoot` *`directorio`*: Esta directiva es clave ya que permite especificar la estructura de directorios que será publicada por el servidor Apache. Dicha ruta será la raíz del dominio que gestione el servidor. Ejemplo: DocuementRoot /usr/web

`ErrorDocument código` *`documento`*: Sirve para especificar un documento alternativo que será enviado en caso de que se haya producido algún error en el transcurso de la petición. Es posible especificar documentos distintos para diferentes códigos de error. Ejemplo: ErrorDocument 404 /noexiste.html

`ErrorLog` *`fichero`*: Especifica la ubicación donde el servidor Apache registrará todos los errores que hayan ocurrido en la aplicación. Hay que tener en cuenta que el fichero debe tener permisos suficientes para que el propietario del proceso httpd pueda escribir en él. Es una herramienta básica a la hora de la administración, ya que permite detectar los últimos errores. Ejemplo: ErrorLog /var/log/apache2/error_log

`Include` *`fichero`*: Incluye el contenido del fichero especificado en esta directiva dentro del fichero de configuración del servidor. Es posible indicar un directorio en lugar de un fichero, en cuyo caso Apache incluirá todos los ficheros que en él se encuentren. Ejemplo: Include /usr/local/apache2/conf/ssl.conf

Más información: http://httpd.apache.org/docs/2.0/mod/core.html

Mpm_common: Contiene directivas de multiprocesamiento sobre el número de hilos, parámetros sobre la red, o de configuración de la memoria. En realidad no es una directiva como tal, sino que existen distintas implementaciones en función de la arquitectura en la que estemos trabajando. De este modo existen versiones para BeOS, Netware, OS2, WinNT…

También existen versiones alternativas que evitan el multiprocesamiento. A continuación se muestran algunas de las directivas más relevantes:

`Listen ip:puerto`: Esta directiva sirve para especificar la dirección IP y el puerto en que debe estar escuchando el servidor Apache. Por defecto, el servidor escucha por todas las direcciones IP a las que esté conectado, por lo que es posible dejar sin indicar. Es posible especificar más de una vez esta directiva. Ejemplo: Listen 192.168.0.20:80

`MaxClients numero`: Este número indica el número de procesos hijo que van a ser creados como máximo para atender las peticiones que llegan al servidor. Las peticiones que lleguen después se pondrán en cola. Es una directiva a tener en cuenta para optimizar el servidor. Ejemplo: MaxClients 256

`MaxRequestPerChild numero`: Indica la vida útil de los procesos hijos que procesan las peticiones que llegan, ya que especifica el número máximo de peticiones que procesará antes de morir y ser reemplazado por otro proceso que hará su misma función. En caso de especificar 0, no habrá límite. Ejemplo: MaxRequestPerChild 250

`StartServers numero`: Con esta directiva indicaremos el número máximo de procesos hijo que serán creados cada vez que Apache sea iniciado. Ejemplo: StartServers 5

`User usuario`: En los casos en los que el servidor haya sido iniciado como root, podremos especificar el nombre de usuario del sistema Unix con el que el servidor responderá las peticiones. El usuario debe tener tantos privilegios como se desea que tenga un usuario que provenga del exterior. También es posible especificar el grupo del propietario con la directiva "Group". Ejemplo: User nobody

Más información: http://httpd.apache.org/docs/2.0/mod/mpm_common.html

Mod_access: Este módulo se encarga de implementar la autentificación HTTP básica. En ella podemos filtrar por características que se encuentren en el paquete de petición del cliente. Algunas de estas pueden ser la IP de origen, o nombres de dominio. Se muestran algunas de las directivas más importantes:

`Allow from all | host`: Indica que hosts tienen acceso a un área del servidor. Es posible especificar "all" si queremos incluir a cualquier equipo, o bien una Ip o incluso un rango de estas para restringir su aplicación. Ejemplo: Allow from 192.168.0.100

`Deny from all | host`: Indica que hosts no tienen acceso a un área del servidor. Es posible especificar "all" si queremos incluir a cualquier equipo, o bien una Ip o incluso un rango de estas para restringir su aplicación. Ejemplo: Allow from 192.168.0.100

`Order preferencia`: Establece el orden en el que Allow y Deny son evaluadas. En primer lugar se evalúan las que correspondan a la primera preferencia, después a la segunda, y por último aquellas que no coinciden con ninguno de los dos casos. Ejemplo: Order Deny,Allow

Más información: http://httpd.apache.org/docs/2.0/mod/mod_access.html

Mod_actions: Permite la ejecución de scripts CGI cuando ocurre una petición multimedia. Puede detectarla mediante un tipo MIME o un manejador instalado. Este módulo dispone de dos directivas:

`Action tipo_MIME scipt`**:** Utilizaremos esta directiva para asociar un script a un tipo MIME determinado. Vamos a suponer que cada vez que exista una solicitud de un fichero HTML, deseamos ejecutar un determinado script. Ejemplo: Action text/html /cgi-bin/script

`Script método script`**:** En este caso asociaremos la acción con una solicitud HTTP como pueden ser GET, POST, PUT o DELETE. Ejemplo: Script POST /cgi-bin/script

Más información: http://httpd.apache.org/docs/2.0/mod/mod_actions.html

Mod_alias: Permite direccionar mediante una URL un contenido que se encuentra fuera del árbol de directorio de publicación. También permite añadir un directorio de scripts CGIs y multitud de redirecciones.

`Alias URL-ruta ruta`**:** Asocia una ruta del sistema de ficheros para que sea accesible desde el exterior. Para los usuarios externos será totalmente transparente, ya que no se podrá diferenciar si pertenece al árbol de DocumentRoot o no. Ejemplo: Alias /data/ "/www/datos/". En este ejemplo http://localhost/data corresponderá a la ruta local /www/datos/

`Redirect [código] URL-antigua URL-nueva`**:** Permite redirigir una solicitud de una URL en otra. Adicionalmente permite especificar un código de error que será devuelto en la respuesta del servidor. Es útil si ha movido su web de dominio o ha movido alguna sección de su web. Ejemplo: Redirect /servicios http://url-alternativa/servicios

`ScriptAlias alias ruta`**:** Cuando el destino es una carpeta que va a ser un conjunto de scripts CGI que van a ser ejecutados. Por lo demás funciona igual que Alias. Ejemplo: ScriptAlias /cgi-bin/ /var/www/cgi

Más información: http://httpd.apache.org/docs/2.0/mod/mod_alias.html

Mod_asis: Permite devolver a Apache un fichero sin especificar las cabeceras que le corresponden. El tipo MIME anunciado es httpd/send-as-is. Este módulo no dispone de directivas.

Más información: http://httpd.apache.org/docs/2.0/mod/mod_asis.html

Mod_auth: Usado para crear un sistema de autenticación que permita proteger el acceso a distintos recursos. Los usuarios y grupos se guardan en un fichero de texto plano. Vamos a ver las dos únicas directivas de las que dispone:

`AuthUserFile fichero`**:** Permite el acceso únicamente a los usuarios que están incluidos en el fichero especificado. Para crear el fichero la primera vez ejecutaremos el comando "htpasswd –c fichero usuario", y para las siguientes veces "htpasswd fichero usuario". Suele especificarse dentro de un ámbito específico como puede ser "<Directory>". Ejemplo: AuthUserFile /etc/apache2/usuariosAceptados

`AuthGroupFile fichero`**:** El funcionamiento es muy similar a la directiva anterior, pero especificando el grupo que tiene permitido el acceso. Ejemplo: Ejemplo: AuthUserFile /etc/apache2/gruposAceptados

Más información: http://httpd.apache.org/docs/2.0/mod/mod_auth.html

Mod_auth_anon: A la hora de crear un control de acceso, es posible habilitar un usuario anónimo con sus respectivos datos para permitir acceder a zonas restringidas. Vamos a ver las directivas más importantes:

`Anonymous usuario`**:** Este será el nombre (o nombres) de usuario con el que será posible validarse para obtener los permisos de un usuario anónimo. Ejemplo: Anonymous anonimo invitado

`Anonymous_LogEmail On|Off`**:** Permite registrar la contraseña que ha introducido el usuario para poder loguearse en el caso de usar un nombre de los especificados en la directiva anterior. Esto es así porque está pensado para que la contraseña sea el email del usuario. Ejemplo: Anonymous_LogEmail On

`Anonymous_MustGiveEmail On|Off`**:** Si esta directiva está activada prohibirá que un usuario anónimo se loguee sin contraseña, y lo obligará a especificar su email en el campo de contraseña. Ejemplo Anonymous_MustGiveEmail On

Anonymous_VerifyEmail On|Off: Cuando especificamos que el usuario debe introducir su email en el campo contraseña, activando esta directiva podremos activar la validación de este campo. Básicamente se trata de comprobar que existan los símbolos "@" y ".". Ejemplo: Anonymous_VerifyEmail On

Más información: http://httpd.apache.org/docs/2.0/mod/mod_auth_anon.html

Mod_auth_digest: Este tipo de autentificación es una alternativa a mod_auth, que permite que los datos de acceso sean almacenados en una base de datos.

Más información: http://httpd.apache.org/docs/2.0/mod/mod_auth_digest.html

Mod_autoindex: Cuando no se especifica un fichero dentro del directorio que se solicita y además no existe un documento por defecto (como por ejemplo index.html), este módulo se encarga de generar un listado con los recursos disponibles en ese directorio. El resultado es similar al comando ls de Unix, o dir de Windows. Las directivas de este módulo sirven para modificar ciertos aspectos del listado como pueden ser los iconos o textos alternativos. En lugar de explicar estas directivas, que pueden ser consultadas en la URL que aparece a continuación, se va a mostrar un ejemplo de cómo activar un listado para un directorio concreto.

```
<Directory /var/www/documentos>
     Options + Indexes
</Directory>
```

Más información: http://httpd.apache.org/docs/2.0/mod/mod_autoindex.html

Mod_cache: En ocasiones por cuestión de rendimiento o capacidad del ancho de banda puede ser recomendable guardar ciertos documentos en el disco del servidor, de forma que al solicitar estos recursos puedan ser enviados al cliente con la mayor brevedad posible. Se resumen las directivas más importantes:

`CacheDefaultExpire segundos`**:** Especifica el tiempo predeterminado en segundos que un contenido quedará guardado en la caché por defecto. No afectará a documentos que no dispongan de fecha de última modificación o fecha de creación. Ejemplo: CacheDefaultExpire 86400

`CacheDisable ruta-URL`**:** Impide que todos los documentos que cuelguen de esta ruta sean almacenados en caché. Ejemplo: CacheDisable /documentos

`CacheEnable tipo ruta-URL`**:** Habilita la caché para los documentos de la ruta especificada. Es posible especificar el tipo de manejador que controlará la caché.Ejemplo: CacheEnable mem /manual

`CacheMaxExpire segundos`**:** Especifica el tiempo predeterminado en segundos que un contenido quedará guardado en la caché como máximo. Incluso si el documentos cacheado dispone de su propia fecha de caducidad, se seguirá aplicando esta directiva. Ejemplo: CacheDefaultExpire 86400

Más información: http://httpd.apache.org/docs/2.0/mod/mod_cache.html

Mod_cgi: Permite ejecutar scripts CGI para generar documentos dinámicos o ejecutar diversas acciones en el servidor. CGI es una especificación de una interfaz, y permite usar cualquier

lenguaje siempre y cuando dispongamos del intérprete o compilador necesario. Vamos a resumir a continuación la directiva más importante:

`ScriptLog fichero:` Si especificamos un fichero en esta directiva, obligaremos a que Apache registre todos los errores que sean provocados por la ejecución de scripts CGI. Ejemplo ScriptLog logs/cgi_log

Más información: http://httpd.apache.org/docs/2.0/mod/mod_cgi.html

Mod_cgid: En el caso del módulo anterior, el coste que tiene la creación de nuevos procesos cada vez que se ejecuta un script CGI es altísimo. Este módulo intenta reducir este costo creando un demonio encargado de ejecutar los scripts.

Más información: http://httpd.apache.org/docs/2.0/mod/mod_cgid.html

Mod_charset_lite: Este es un módulo experimental y debe usarse con cautela. Permite especificar una codificación de caracteres para los recursos que sirve el servidor. Estos se pueden especificar en ámbitos <Directory> dentro del fichero de configuración de Apache. Vamos a resumir a continuación la directiva más importante:

`CharsetDefault` *`charset`*`:` Especifica el juego de caracteres con que se devolverá el contenido. Ejemplo: CharsetDefault ISO-8859-1

Más información: http://httpd.apache.org/docs/2.0/mod/mod_charset_lite.html

Mod_dav: Permite añadir soporte al protocolo WebDAV, para crear, modificar, eliminar y copiar recursos en un servidor remoto. Puede ser muy útil para que un editor de un sitio web introduzca contenido en el mismo sin necesidad de tener permisos sobre el código fuente.

`Dav On|Off:` Permite activar o desactivar dicho protocoloto en el ámbito en que sea especificado. Hay que tener especial cuidado con esto último. Ejemplo: Dav On

Más información: http://httpd.apache.org/docs/2.0/mod/mod_dav.html

Mod_dav_fs: Necesita de Mod_dav para funcionar, ya que actúa como soporte de este último para poder proporcionarle acceso al sistema de ficheros.

Más información: http://httpd.apache.org/docs/2.0/mod/mod_dav_fs.html

Mod_deflate: Este módulo proporciona un filtro a la hora de enviar el contenido a los clientes. El objetivo es comprimir todo el contenido antes de enviarlo a través de la red, sacrificando rendimiento del servidor para conseguir un mejor rendimiento de la red. Se puede especificar qué tipos de fichero pueden ser comprimidos y en qué ámbitos.

`SetOutputFilter DEFLATE`: Activa la compresión a todos los documentos que estén contenidos en el ámbito en el que se ha especificado la directiva. Ejemplo: SetOutputFilter DEFLATE

`AddOutputFilterByType DEFLATE tipoMIME`: Restringe la compresión de documentos únicamente a aquellos tipos MIME especificados en esta directiva. Ejemplo: AddOutputFilterByType DEFLATE text/html text/plain text/xml.

Más información: http://httpd.apache.org/docs/2.0/mod/mod_deflate.html

Mod_dir: Por un lado permite redirigir al directorio adecuado cuando se intenta acceder a un recurso directorio en el que no se ha especificado la barra final. Por otro lado permite especificar los ficheros por defecto que se intentarán cargar cuando se especifique la ruta de un directorio.

`DirectoryIndex fichero`: Buscará el fichero (o ficheros) especificado para devolverlo al cliente en el caso de que el recurso que haya solicitado sea un directorio. En caso de indicar varios, se procesarán en el orden en que están escritos. Esta directiva no tiene aplicación en el caso de que Indexes esté activado. Ejemplo: DirectoryIndex index.html

Más información: http://httpd.apache.org/docs/2.0/mod/mod_dir.html

Mod_disk_cache: Este módulo implementa un manejador disco. Necesita de mod_cache para funcionar. Uno de sus usos más importantes es al usar Apache como proxy. Estas son algunas de sus directivas más importantes:

`CacheDirLevels` *`niveles`*: Indica el número de niveles de subdirectorios máximo pueden ser almacenados en la caché. Este número no debe ser mayor que 20. Ejemplo: CacheDirLevels 3

`CacheMinFileSize` *`bytes`*: Indica el número de bytes que debe tener un documento como mínimo para poder ser almacenado en la caché. Ejemplo CacheMinFileSize 64

`CacheRoot ruta`: Especifica la ruta donde serán almacenados en disco los ficheros que permanezcan almacenados en la caché. Ejemplo: CacheRoot /var/cache

`CacheSize KBytes`: Sirve para especificar cuál es el número máximo de KBytes que debe ocupar en disco la caché. Ejemplo: CacheSize 1000000

Más información: http://httpd.apache.org/docs/2.0/mod/mod_disk_cache.html

Mod_dumpio: Para la depuración de errores de entrada / salida se puede habilitar el fichero de registro de errores para que sean almacenados todos los posibles errores que puedan ocurrir. Su configuración es muy sencilla:

`DumpIOInput On|Off`**:** Habilita la depuración de errores de entrada. Ejemplo: DumpIOInput On

`DumpIOOutput On|Off`**:** Habilita la depuración de errores de entrada. Ejemplo: DumpIOOutput On

Más información: http://httpd.apache.org/docs/2.0/mod/mod_dumpio.html

Mod_echo: Este módulo actúa como un servidor de echo. Todo lo que sea tecleado será reenviado.

`ProtocolEcho on|off`**:** Simplemente activa o desactiva el protocolo.

Más información: http://httpd.apache.org/docs/2.0/mod/mod_echo.html

Mod_env: La funcionalidad que implementa este módulo es la de permitir el acceso y la modificación de las variables de entorno. Estas variables serán accesibles por parte de las páginas SSI y los script CGI.

`PassEnv variable [variable]`**:** Especifica las variables que podrán ser accedidas por las páginas SSI y los scripts CGI. Ejemplo: PassEnv LD_LIBRARY_PATH

`SetEnv variable valor`**:** Da un valor a una nueva variable o modifica una ya existente. Ejemplo: SetEnv SPECIAL_PATH /foo/bin

`UnsetEnv variable [variable]`**:** Elimina una o más variables para que no sean accesibles a las páginas SSI ni a los scripts CGI. Ejemplo: UnsetEnv LD_LIBRARY_PATH

Más información: http://httpd.apache.org/docs/2.0/mod/mod_env.html

Mod_expires: Este módulo permite configurar los campos “Expires” y “max-age” de la cabecera HTTP. Esto puede ser útil por ejemplo para ahorrar ancho de banda de nuestra red.

`ExpiresActive On|Off`**:** Habilita las cabeceras “Expires” y “Cache-control” de los documentos servidos. En caso de ser activadas se fijará el criterio elegido en las siguientes directivas. Ejemplo ExpiresActive On

`ExpiresByType tipoMIME <codigo>segundos`**:** Establece un tiempo de expiración para los documentos que pertenecen a un determinado tipo MIME. Esta directiva sobreescribe a ExpiresDefault. Ejemplo: ExpiresByTipe text/html M604800

`ExpiresDefault <codigo>segundos`**:** Establece un tiempo de expiración cualquier tipo de documento. Esta directiva puede ser sobreescrita por ExpiresByType. Ejemplo: ExpiresDefault M604800

En cuanto al código que se ha mencionado en las últimas dos directivas, debe especificarse una "A" para especificar el acceso al documento, y una "M" para la modificación del mismo.

Más información: http://httpd.apache.org/docs/2.0/mod/mod_expires.html

Mod_ext_filter: La utilidad que presta es pasar el cuerpo de la respuesta a través un programa externo antes de que sea entregado al cliente. Los programas no es necesario que estén implementados para funcionar con Apache, ya que está pensado para que los programas externos lean de la entrada estándar y escriban por la salida estándar.

Más información: http://httpd.apache.org/docs/2.0/mod/mod_ext_filter.html

Mod_file_cache: Ofrece almacenamiento en caché para servir ficheros que no necesitan ser atendidos por controladores de contenido especial. Este módulo debe ser utilizado con cuidado.

Más información: http://httpd.apache.org/docs/2.0/mod/mod_file_cache.html

Mod_headers: Nos proporciona instrumentos para controlar y modificar las solicitudes y respuestas HTTP. Es posible tanto añadir, como modificar, como eliminar todas aquellas cabeceras que creamos convenientes.

Más información: http://httpd.apache.org/docs/2.0/mod/mod_headers.html

Mod_include: Este módulo sirve para analizar los documentos de servidor SSI (Server Side Includes). Está implementado como un filtro de salida, ya que la intercepta para procesar los comandos contenidos en dichos ficheros.

Más información: http://httpd.apache.org/docs/2.0/mod/mod_include.html

Mod_info: Permite conocer detalles sobre cómo está configurado el servidor, qué directivas están activadas, qué módulos están instalados. No se encuentra compilado en el servidor por defecto, por lo que deberemos instalarlo por separado.

```
<Location /server-info>
      SetHandler server-info
      Order deny,allow
      Allow from 127.0.0.1
</Location>
```

Este ejemplo permite que a través de la ruta http://localhost/server-info podamos acceder a una página con la información del servidor al completo. Si desea modificar los permisos modifique la línea "Allow from", añadiendo los lugares desde los cuales desea autorizar el acceso a esta información.

Más información: http://httpd.apache.org/docs/2.0/mod/mod_info.html

Mod_isapi: Esta opción únicamente está disponible para la versión para Windows de Apache. Su utilidad es permitir cargar extensiones ISAPI escritos por terceros para añadir nuevas funcionalidades a nuestro servidor.

Más información: http://httpd.apache.org/docs/2.0/mod/mod_isapi.html

Mod_ldap: Permite que el servidor realice conexiones a servidores LDAP, puede utilizarse para verificación de credenciales de acceso, o alojamiento de ficheros. A pesar de que dispone de gran cantidad de directivas aquí no se van a comentar.

Más información: http://httpd.apache.org/docs/2.0/mod/mod_ldap.html

Mod_log_conf: Es el encargado de llevar el registro de las peticiones que registra el servidor web. Permite configurar al detalle cual es el formato que tendrán los registros que queden guardados en el fichero log. Hay que tener especial atención con los permisos que se le aplican a este fichero, ya que puede ser un gran riesgo de seguridad si no es tratado convenientemente.

Más información: http://httpd.apache.org/docs/2.0/mod/mod_log_config.html

Mod_log_forensic: Es un registro similar al anterior, pero mucho más estricto, ya que por cada petición registra antes y después de procesar la solicitud. Existe un script llamado "check_forensic" que permite comprobar si el formato de log configurado es correcto antes de reiniciar el servidor.

Más información: http://httpd.apache.org/docs/2.0/mod/mod_log_forensic.html

Mod_logio: La funcionalidad de este módulo es el registro en el log de las operaciones de entrada y salida. Además registra también el número de bytes enviados / recibidos en cada petición. Este módulo requiere `mod_log_config`.

Más información: http://httpd.apache.org/docs/2.0/mod/mod_logio.html

Mod_mem_cache: Este módulo por el momento es todavía experimental por lo que hay que utilizarlo con cuidado. Su función es proveer un administrador de memoria para los ficheros que se van a almacenar en la caché.

Más información: http://httpd.apache.org/docs/2.0/mod/mod_mem_cache.html

Mod_mime: Sirve para asociar una extensión de fichero a un tipo de fichero, al lenguaje del juego de caracteres y la codificación apropiada. Esta información es utilizada por el cliente para la negociación del tipo de contenido. Por ejemplo lo usaremos para indicar que los ficheros con extensión .gif pertenecen al tipo MIME image/gif.

Más información: http://httpd.apache.org/docs/2.0/mod/mod_mime.html

Mod_mime_magic: En los casos en los que `mod_mime` no pueda concretar cuál es el tipo MIME del fichero que se va a servir, este módulo leerá los primeros bytes del fichero para intentar averiguarlo y que se pueda anunciar correctamente su tipo MIME. Hay que tener en cuenta que si este módulo se ejecuta constantemente el rendimiento del servidor se verá afectado seriamente, por lo que deberemos intentar que el módulo `mod_mime` sea capaz de averiguar el tipo MIME de la mayor cantidad de ficheros posible.

Más información: http://httpd.apache.org/docs/2.0/mod/mod_mime_magic.html

Mod_negotiation: Permite la negociación de contenido, para permitir servir a los clientes el contenido que mejor se adapte a sus necesidades en el caso de haber varios documentos disponibles. Entre los parámetros con los que trabaja este módulo están el tipo MIME, el idioma en el que se encuentran escritos, o la codificación utilizada.

Más información: http://httpd.apache.org/docs/2.0/mod/mod_negotiation.html

Mod_proxy: Con este módulo se puede crear un servidor proxy válido para tratar peticiones HTTP 1.0, HTTP 1.1, HTTPS y FTP. Las funcionalidades del almacenamiento en caché son proporcionadas por `mod_cache`. A través de las directivas de este módulo podemos

configurar opciones como el sitio del que será proxy, los permisos, los tiempos de espera, el máximo número de conexiones, el tamaño de los buffers…

Más información: http://httpd.apache.org/docs/2.0/mod/mod_proxy.html

Mod_proxy_connect: Implementa la petición "CONNECT" de HTTP bajo una conexión a través de un proxy (normalmente bajo conexiones encriptadas SSL). Para que este módulo funcione es necesario que en el servidor se encuentre activado el módulo `mod_proxy`. No dispone de directivas que permitan configurarlo.

Más información: http://httpd.apache.org/docs/2.0/mod/mod_proxy_connect.html

Mod_proxy_ftp: Implementa el soporta para conexiones FTP a través de proxy. Para que este módulo funcione es necesario que en el servidor se encuentre activado el módulo `mod_proxy`. No dispone de directivas que permitan configurarlo.

Más información: http://httpd.apache.org/docs/2.0/mod/mod_proxy_ftp.html

Mod_proxy_http: Implementa el soporta para conexiones FTP a través de proxy. Para que este módulo funcione es necesario que en el servidor se encuentre activado el módulo `mod_proxy`. No dispone de directivas que permitan configurarlo.

Más información: http://httpd.apache.org/docs/2.0/mod/mod_proxy_http.html

Mod_rewrite: Mediante expresiones regulares es capaz de utilizar un motor incorporado para la reescritura de URLs. Las reglas que se pueden crear para reescribir las URLs son muy potentes y permiten una gran cantidad de posibilidades. Es capaz de establecer condiciones en base a las variables de entorno, los encabezados HTTP, y un gran número de parámetros diferentes. Además la reescritura de URLs puede encadenarse hasta que nosotros queramos, unida con un número infinito de reglas, nos da unas posibilidades difíciles de imaginar.

Cuando el servidor recibe una petición, automáticamente esta la pasas al módulo `mod_rewrite`. A partir de este momento, se buscará en las reglas disponibles si existe una expresión regular que coincida con la URL a la que está realizando la petición el usuario. Si al menos una de ellas coincide se le aplicará la regla de reescritura, que dará como resultado una URL nueva que será la que se pasará al motor de Apache para que devuelva el contenido correspondiente al cliente.

Más información: http://httpd.apache.org/docs/2.0/mod/mod_rewrite.html

Mod_setenvif: Permite modificar las variables de entorno en tiempo de ejecución, ofreciendo variables personalizadas para algunas solicitudes HTTP. Este mecanismo funciona aplicando expresiones regulares, de forma que con este criterio podemos establecer el valor de las variables de entorno que creamos convenientes para cada caso particular.

Más información: http://httpd.apache.org/docs/2.0/mod/mod_setenvif.html

Mod_so: Permite cargar código ejecutable y módulos dinámicos en Apache al iniciarlo (o reiniciarlo). En el caso de Unix la extensión de estos ficheros es .so, y en Windows suele ser .mod o .dll. Debe estar seguro de que el módulo que se va a cargar es totalmente compatible con la versión de Apache que está ejecutando.

Por ejemplo, la siguiente directiva podría utilizarse para cargar el módulo mod_actions que se encuentra en la ruta /usr/lib/apache2/modules/ :

```
LoadModule actions_module /usr/lib/apache2/modules/mod_actions.so
```

Más información: http://httpd.apache.org/docs/2.0/mod/mod_so.html

Mod_speling: Este módulo intenta corregir peticiones que llegan al servidor con URLs incorrectas. El caso más sencillo es la equivocación al escribir las letras minúsculas y mayúsculas, pero también es capaz de detectar (y corregir) un error tipográfico.

Vamos a ver dos posibles ejemplos de utilización de este módulo. En el primero la corrección se limita únicamente a los errores de capitalización (minúsculas y mayúsculas), ignorando el resto de posibles errores.

```
CheckCaseOnly on
```

En el segundo caso se activa la totalidad de las opciones del módulo, permitiendo corregir también hasta un error tipográfico.

```
CheckSpelling on
```

Más información: http://httpd.apache.org/docs/2.0/mod/mod_speling.html

Mod_ssl: Permite la utilización del protocolo SSL versión 2 y 3, y el protocolo TLS. Estos son los protocolos que permiten que nuestro tráfico viaje cifrado, de forma que en caso de ser interceptado no sea entendible por nadie (excepto por el emisor y el receptor). Resulta de vital importancia para cualquier sitio donde simplemente se deba introducir un usuario y una contraseña para acceder a un determinado servicio. A pesar de que este módulo añade una carga adicional al servidor es recomendable tanto por cuestiones de seguridad, como por cuestiones de confianza, al mejorar nuestra imagen al exterior.

Más información: http://httpd.apache.org/docs/2.0/mod/mod_ssl.html

Mod_status: Permite monitorizar el uso del servidor Apache en tiempo de ejecución, generando información sobre diversos datos, como el último reinicio, los accesos al servidor, las solicitudes procesadas, el estado de cada hijo (incluyendo CPU, número de solicitudes por segundo, bytes servidos por segundo y bytes por solicitud). A esta información se accede a través de una página HTML que se genera en tiempo real en el servidor en el momento de realizar la petición. Este módulo no está compilando por defecto, por lo que deberemos compilarlo por separado.

```
<Location /server-status>
      SetHandler server-status
      Order deny,allow
      Allow from 127.0.0.1
</Location>
```

Este ejemplo permite que a través de la ruta http://localhost/server-status podamos acceder a una página con la información del servidor al completo. Si desea modificar los permisos modifique la línea “Allow from”, añadiendo los lugares desde los cuales desea autorizar el acceso a esta información.

Más información: http://httpd.apache.org/docs/2.0/mod/mod_status.html

Mod_suexec: Este módulo especificará con qué permisos se ejecutarán los scripts CGIs. Concretamente permite especificar el usario y el grupo propietario con que se ejecutarán los procesos al ser creados. Es una opción importante para la seguridad de nuestro sistema, por lo que debe utilizarse con cuidado.

```
SuexecUserGroup usuario grupo
```

En el ejemplo anterior debe sustituirse el campo “usuario” por el nombre de usuario con el que se ejecutará, y el campo “grupo” para el caso del grupo del propietario del proceso.

Más información: http://httpd.apache.org/docs/2.0/mod/mod_suexec.html

Mod_unique_id: Crea una variable de entorno que proporciona un identificador único para cada solicitud que procesa cada solicitud. Esta variable se llama UNIQUE_ID. El objetivo es que cuando el servidor se ejecute en un entorno de clustering, este identificador único sea conocido por todo el cluster.

Más información: http://httpd.apache.org/docs/2.0/mod/mod_unique_id.html

Mod_userdir: Permite que cada usuario del sistema disponga de un directorio donde publicar sus propios contenidos. Normalmente suele ser accesible a través de http://miweb.com/~user, sustituyendo user por cada uno de los usuarios que dispongan de cuenta de usuario.

Mediante la directiva UserDir se establece el directorio en el cual se almacenarán los documentos a publicar. Éste suele encontrarse en un subdirectorio del home del propio usuario.

Es posible también especificar tanto listas negras, como blancas sobre los usuarios que podrán publicar contenidos a través de su directorio personal.

Más información: http://httpd.apache.org/docs/2.0/mod/mod_userdir.html

Mod_version: Este módulo puede ser realmente útil para usar en pruebas en las que hay que prestar especial atención a las distintas versiones distintas del servidor Apache. Es posible realizar comprobaciones tanto numéricas, como mediante expresiones regulares. El siguiente ejemplo pertenece al primer tipo:

```
<IfVersion 2.1.0>
    # la versión que ejecuta es la 2.1.0
</ IfVersion>
```

Más información: http://httpd.apache.org/docs/2.0/mod/mod_version.html

Mod_vhost_alias: Permite crear de forma dinámica hosts virtuales con criterios para diferenciarlos como la IP o el dominio que aparezca en la cabecerra de la petición HTTP.

Más información: http://httpd.apache.org/docs/2.0/mod/mod_vhost_alias.html

5.2.3.- Trabajando con PHP

En el punto en el que estamos, Apache es capaz de servir perfectamente gran cantidad de contenido a los usuarios que se conecten a nuestro sitio. Pero tenemos el problema de que el contenido es totalmente estático, por lo que habría que modificar las distintas páginas cada vez que hubiera una modificación (algo totalmente inviable), además de necesitar algún mecanismo para capturar los datos que el usuario nos ofrece.

PHP es un lenguaje interpretado que se utiliza para ejecutar scripts en el lado del servidor, sirve perfectamente para todos los cometidos que necesitamos. Normalmente el código PHP suele integrarse con el código HTML, lo que permite que las partes dinámicas sean integradas en nuestras páginas con un esfuerzo sustancialmente pequeño en comparación con la reescritura de todo el código desde cero.

Una de las ventajas que aporta este lenguaje es que es portable a la mayoría de los sistemas operativos , entre ellos Unix, Windows, Mac y Linux. Esto significa que desarrollando únicamente un mismo script podemos hacerlo funcionar en todas aquellas plataformas en las que esté soportado PHP (que no son pocas). Únicamente necesitaremos un intérprete o un módulo para nuestro servidor web.

También es altamente portable a la mayoría de los servidores web, ya que desde PHP 4.0 la interfaz del servidor es abstracta. Esta virtud permite que PHP se integre con multitud de servidores web, entre ellos Apache, IIS, Zeus, etc. Esto es altamente importante si deseamos crear un servidor web embebido para realizar tareas muy concretas, como podrían ser los enrutadores que disponen de su propia interfaz web.

La mayoría de las bases de datos relacionales usadas en la actualidad son totalmente compatibles con PHP. De hecho, existen interfaces nativas para ofrecer un soporte total que permita interactuar con la base de datos que más nos convenga. Podemos optar entre otras por MySQL, Postgres, Oracle y Microsoft SQL Server.

Por otra parte, para los programadores es un lenguaje con una curva de aprendizaje muy suave. El motivo es que está compuesto por instrucciones de otros lenguajes como C, Java y Perl, hasta el punto de que puede resultar confuso disponer de varias instrucciones con un funcionamiento similar. Esto se realizó así para que los desarrolladores que provengan de otros lenguajes pudieran seguir utilizando las instrucciones a las que estaban acostumbrados.

A pesar de ser un lenguaje interpretado (puesto que no necesita compilación), PHP tiene un rendimiento alto en la ejecución de los comandos. Esto es especialmente importante dado que una aplicación compleja puede llegar a ejecutar multitud de instrucciones, y éstas deben

ejecutarse lo más rápidamente posible y con una carga para el servidor tan reducida como sea posible. Esto es crucial para la escalabilidad de los sistemas que desarrollemos.

Las capacidades de comunicación son muy grandes, ya que soporta una gran cantidad de estándares de internet como FTP, IMAP, XML, LDAP, SNMP, etc. Esto provoca que la integración de una aplicación realizada con PHP con otros servicios, pueda ser tan grande como podamos imaginar, ya que todas estas posibilidades vienen incluido en el núcleo del lenguaje.

Cuando nosotros escribimos una aplicación o una web dinámica en PHP, el código fuente nunca es desvelado a los usuarios. Esto es así ya que en el momento en el que el servidor recibe una solicitud, en lugar de devolver el documento al cliente, procesa el documento buscando comandos PHP, a continuación ejecuta los distintos elementos del script y se genera una salida únicamente compuesta por código HTML.

5.2.3.1.- Instalación de PHP

Al igual que en la instalación de Apache, en el caso de la instalación de PHP también tenemos numerosas opciones a tener en cuenta. En primer lugar hay que decir que todo el software se puede descargar desde la web oficial `www.php.net` sin ningún tipo de coste.

Es posible descargar tanto el código fuente sobre el que poder compilarlo, como los binarios directamente listos para ejecutarlos, por lo que deberemos considerar nuestro caso individualmente para comprobar cual es la alternativa que mejor se adapta a nuestra situación.

En el caso de optar por la instalación desde el código fuente, el primer caso será descargar el mismo desde el lugar indicado más arriba. Después de este primer paso, deberemos descomprimir mediante el comando "`tar xvzf php-x.x.x.tar.gz`".

Llegados a este punto deberemos tomar una última decisión sobre si deseamos que PHP se ejecute como un módulo o como un programa CGI.

En la documentación de PHP se aconseja no utilizar la solución CGI, por motivos de rendimiento y por cuestiones de seguridad. Si ejecutamos PHP como CGI debemos tener en cuenta que el rendimiento de los scripts PHP se verá limitado al rendimiento que tengan por definición los scripts CGI, ya que el intérprete de PHP se ejecuta de este modo cada vez cada vez que se solicite un documento PHP. Por otra parte la seguridad podría verse seriamente afectada, por lo que no debería ser usado en un servidor de producción.

Si consideramos la opción de construir PHP como un módulo de Apache, podemos optar por construirla como un módulo estático o dinámico. Ambas son completamente válidas, pero para este proyecto se va a utilizar librerías dinámicas, por lo que este va a ser el caso que se va a explicar con más detenimiento.

Si optamos por construirlo como un módulo estático, el código se almacenará dentro del binario de Apache, por lo que tendremos que recompilarlo añadiendo la opción de incluir el módulo de PHP. Si además deseamos incluir el módulo MySQL para tratar las peticiones a la base de datos este será el momento de hacerlo también.

En el caso de que hayamos optado por un módulo dinámico, una vez ya dispongamos del módulo instalado correctamente, tendremos que decirle a Apache de alguna forma que deseamos activar dicho módulo. Para ello insertaremos la siguiente línea al fichero `httpd.conf`:

```
LoadModule php4_module modules/libphp5.so
```

En este punto ya tendremos activo el módulo, pero deberemos indicar qué tipo de ficheros serán procesados por el motor PHP. En nuestro caso activaremos la extensión “php”, para ello incluiremos las siguientes líneas al fichero de configuración:

```
<Files *.php>
   SetOutputFilter PHP
   SetInputFilter PHP
</Filter>
```

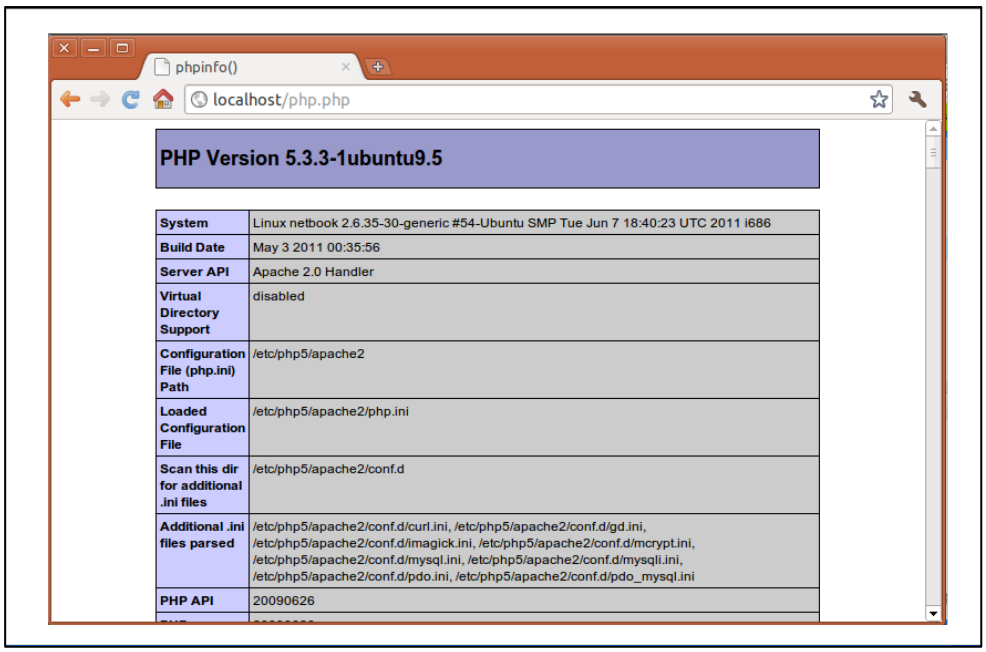
También existe la opción de instalar PHP como un CGI cualquiera, pero entraña ciertos inconvenientes como la limitación de rendimiento a aquella que dispongan los CGIs, ya que este será su funcionamiento. Si se va a procesar un PHP, se ejecutará el intérprete de PHP redireccionándole a su entrada el documento y esperando a su salida la salida de un documento HTML que el cliente ya es capaz de entender. Es cierto que existen soluciones como FastCGI para aumentar el rendimiento de estas aplicaciones, y la solución sería completamente válida para un entorno de desarrollo. Pero nunca para un servidor de producción a no ser que esta sea nuestra última opción por problemas de configuración del servidor, porque sea un servidor compartido, etc.

Una vez la instalación haya finalizado, deberemos comprobar si ha ido todo correctamente. La forma más sencilla es solicitando al servidor una página escrita en PHP, pero primero deberemos crearla. El ejemplo más ilustrativo es que se ejecute la función phpinfo() que aporta información de todo lo referente con PHP. Podríamos usar cualquier función del lenguaje para comprobar que todo ha ido bien, pero de esta forma podremos comprobar si existe algún error.

Para ello iremos al directorio de publicación del servidor web y crearemos un nuevo documento al que podemos llamar "php.php". El contenido de este documento será el siguiente:

```
<?php
     phpinfo();
?>
```

Ahora deberemos cargar el documento que acabamos de crear en nuestro navegador, para ello deberemos cargar la dirección http://localhost/php.php y obtendremos un resultado como el que aparece en la siguiente imagen. Ello será síntoma de que la instalación ha funcionado correctamente, pero además nos permitirá conocer datos sobre la configuración de todo lo relativo con la instalación PHP.



5.2.3.2.- Configurar PHP

Aunque PHP viene con una configuración por defecto, en ocasiones puede ser interesante modificarla según las necesidades de nuestra aplicación. Los motivos de más peso para modificar dicha configuración son los relativos a la seguridad de nuestro sistema, la compatibilidad con el código de las aplicaciones que se ejecuten, o el rendimiento del servidor.

Para la configuración del intérprete de PHP, existe un fichero de texto plano llamado php.ini, que puede encontrarse en el directorio donde está instalado PHP. Al igual que ocurría en Apache, esta es la principal forma de configurar las directivas.

A continuación se repasarán las directivas más importantes que deberemos tener en cuenta:

register_globals: En versiones anteriores de PHP al procesar un formulario, los campos se convertían en variables globales, con el riesgo que puede conllevar si alguna coincide en nombre con alguna registrada con anterioridad. Es aconsejable desactivar esta características, ya que en las nuevas versiones de PHP se aconseja utilizar las variables superglobales $_POST y $_GET.

display_errors: Por defecto cuando en nuestra aplicación se produce un error, el motor PHP lo muestra por la salida estándar (impreso en pantalla). Esta funcionalidad es interesante en un entorno de desarrollo para poder tener una mejor idea del error que ha ocurrido, y poder subsanarlo. Por otra parte, no es aconsejable en un servidor de producción, ya que revelaría detalles importantes del funcionamiento de nuestra aplicación.

log_errors: Es aconsejable que los errores que se producen en nuestras aplicaciones sean registrados en un fichero log para poder ser analizados en un futuro por el administrador o desarrollador del sitio. Esto se hace para redirigir los errores a quien realmente puede sacarles partido, ya que a un usuario poco le importa qué tipo de error interno ha ocurrido.

error_log: Indica un nombre de fichero al que se enviarán los errores generados en caso de tener activada la directiva anterior. También es posible enviarlos al log de registro del sistema especificando “syslog”.

error_reporting: Indica el grado de reporte de los errores que llevará a cabo el motor PHP. Es posible especificar que reporte también las advertencias, ya que por defecto únicamente reporta los errores.

magic_quotes_gpc: Sirve para escapar caracteres especiales en los datos de entrada del usuario. Su objetivo es ofrecer una mayor comodidad al desarrollador a la hora de programar una aplicación, pero su uso es desaconsejable por motivos de compatibilidad. De todas formas si su base de datos y el resto de su aplicación es compatible puede utilizarla.

variables_order = “GPCS”: Al especificarlo de esta forma, PHP no genera la matriz $_ENV que se utiliza para almacenar las variables de entorno. La alternativa es utilizar la función getenv() para recuperar las variables de entorno.

short_open_tag= On: Permite usar en los scripts etiquetas abreviadas <? … ?>. Si el valor es off, la forma <?php … ?> o <script>. Aconsejable dejarlo a off

precision = 10: máximo de posiciones decimales visualizadas por pantalla.

zlib.output_compression = Off: Habilita la compresión mediante la librería zlib de los datos de salida del script.

disable_functions = : Permite deshabilitar las funciones PHP indicadas en esta directiva. Si va a inhabilitar varias, debe separarlas con punto y como ";".

max_execution_time = 30: Fija el máximo de segundos que va a estar ejecutándose un script antes de ser finalizado. Previene errores en el código que puedan provocar bucles infinitos.

log_errors_max_len = 1024 : Especifica el tamaño máximo en bytes del fichero error_log. Puede especificar 0 si no desea límite.

ignore_repeated_errors = Off : Ignora errores repetidos en el fichero de anotaciones.

memory_limit = 8M: Fija el tamaño de memoria máximo que puede utilizar un script durante su ejecución. También evita errores de código.

post_max_size = 8 M : Podemos especificar el tamaño máximo de los datos aceptados por método POST.

default_charset = "iso-8859-1" : Será el código de caracteres enviado por PHP en la cabecera de salida.

default_mimetype = "text/html" : Será el tipo MIME que se enviará por defecto en la cabecera de salida.

file_uploads = On : Indica si se desea habilitar o deshabilitar la subida de ficheros mediante HTTP

upload_tmp_dir = : Directorio donde se guardarán de forma temporal los archivos subido al servidor mediante PHP

upload_max_filesize = 2M : Indica el tamaño máximo de los ficheros que pueden subirse al servidor

extension = : Permite indicar las extensiones que desea que cargue PHP en el proceso de arranque

5.2.4- Monitorización

Uno de los aspectos más importantes una vez tenemos implantado nuestro servidor, es ser capaces de recabar ciertos datos sobre el funcionamiento habitual de todo el sistema para comprobar que no se dan situaciones anómalas. A pesar de que esta situación es deseable en un entorno de producción, esto no significa que no deba usarse en un entorno de desarrollo.

Por un lado será importante en el entorno de desarrollo antes de la publicación de nuestro sitio, para comprobar si los requisitos de nuestra aplicación son aceptables, o si por el contrario deberíamos seguir optimizando nuestra aplicación. También son útiles para conocer

las advertencias y errores que provoca nuestra aplicación, y que de otro modo no seríamos capaces de descubrir. Para ello se aplicarán distintos tests tanto unitarios como peticiones al servidor web para comprobar, por ejemplo, el número de conexiones que es capaz de aceptar el servidor, o el tiempo medio de procesamiento de cada solicitud.

Por otro lado será importante en un entorno de producción, cuando nuestra aplicación ya se encuentre publicada y los usuarios ya hagan un uso continuado de la misma. Aquí son varias las opciones que se nos plantean, entre las cuales se incluyen los registros de accesos y errores del servidor, e informes sobre los usuarios que acceden a nuestra aplicación. Aquí deberemos dejar de realizar pruebas de rendimiento puesto que puede afectar a la calidad de nuestra aplicación y al funcionamiento de la misma, pudiendo provocar fallos y falta de servicio de nuestros clientes. Todas estas pruebas deberán realizarse en un entorno aislado que podríamos considerar como entorno de desarrollo.

Todas estas informaciones son muy importantes para conocer las limitaciones de nuestro servicio (unión de las limitaciones del servidor y de las limitaciones de la aplicación) y el uso real que se le da a nuestro servicio. Esto nos permitirá definir mejor los objetivos que desearemos conseguir, tanto en el momento presente, como en vistas a futuras actualizaciones.

También es importante por razones de marketing y valor de mercado, ya que si conocemos mejor a nuestro usuario medio, podemos conseguir segmentar mejor dependiendo de los clientes a los cuales nos dirigimos, ya que razones culturales, ideológicas o de idiomáticas pueden ser suficientes para que un cliente acabe con una posible relación comercial con nuestra empresa.

5.2.4.1.-Entorno de desarrollo

Modulo mod_info

Este módulo nos ofrece una gran cantidad de datos sobre la configuración actual del servidor, permitiendo acceder a esta información de un modo cómodo a través una página web accesible desde el navegador. Este módulo no está compilado por defecto por defecto en la instalación estándar, por lo que deberemos añadirlo a la compilación o bien instalarlo como módulo dinámico. (estos pasos han sido explicados en apartados anteriores)

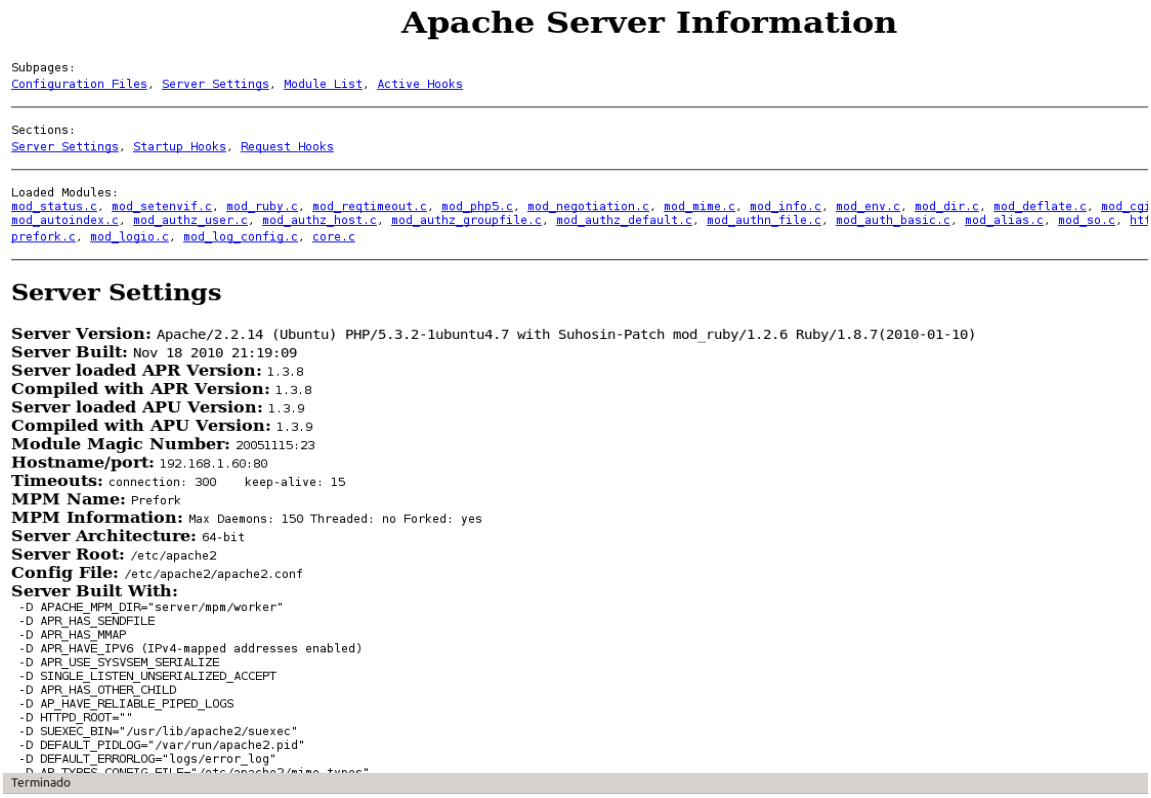
Partiendo de la suposición de que el módulo ya está compilado y configurado, pasemos a la configuración del mismo. Para lo cual deberemos abrir el fichero de configuración y añadir algunas líneas:

```
<Location /server-info>
    SetHandler server-info
    Order deny,allow
    Deny from all
    Allow from 127.0.0.1
</Directory>
```

Estas directivas habilitan la información del servidor en la localización “server-info” del servidor, por lo que la dirección completa a la que deberemos acceder para su consulta será por lo tanto http://localhost/server-info. Por otro lado, se han deshabilitado todos los accesos desde el exterior de la máquina, permitiendo acceder únicamente desde el propio equipo local. Si queremos habilitar más equipos para que puedan acceder a este informe deberemos incluirlos en la lista “Allow from”.

Recuerde reiniciar el servidor para que las modificaciones surjan efecto.

En la imagen siguiente se puede apreciar un ejemplo de un servidor ejecutando las directivas anteriores:



Módulo mod_status

De igual manera que en el caso anterior, el módulo mod_status ofrece una serie de informaciones en la ventana del propio navegador, accesible a través de una URL. Este módulo tampoco está compilado por defecto.

Parte de la información que podemos recabar de este módulo son los momentos de último inicio y reinicio, número de peticiones, número de bytes enviados y recibidos, estado de los procesos hijos, porcentaje de utilización de los recursos hardware del servidor

```
ExtendedStatus on
<Location /server-status>
    SetHandler server-status
    Order deny,allow
    Deny from all
    Allow from 127.0.0.1
</Directory>
```

En el código anterior no se ha comentado el uso de la directiva “ExtendedStatus” nos proporciona un grado de información mayor que el que viene activado por defecto. Si no activáramos esta directiva, algunos de los datos que se han indicado en el párrafo anterior no serían mostrado en nuestro informe.

Esto habilita la información del servidor en la localización “server-status” del servidor, por lo que la dirección completa a la que deberemos acceder será por lo tanto http://localhost/server-status. Por otro lado, se han deshabilitado todos los accesos desde el exterior de la máquina, permitiendo acceder únicamente desde este equipo local. Si queremos habilitar más equipos para que puedan acceder a este informe deberemos incluirlos en la lista “Allow from”.

Volver a recordar que para que los cambios sean efectivos habrá que reiniciar Apache. Esto es un ejemplo de lo que se podrá ver una vez configurado:

Current Time: Monday, 15-Nov-2010 12:41:25 EST
Restart Time: Monday, 15-Nov-2010 12:17:30 EST
Parent Server Generation: 0
Server uptime: 23 minutes 54 seconds
Total accesses: 270 - Total Traffic: 2.9 MB
CPU Usage: u29.46 s2.3 cu0 cs0 - 2.21% CPU load
.188 requests/sec - 2120 B/second - 11.0 kB/request
2 requests currently being processed, 8 idle workers

```
___W_._____.K.....................................................
........................................................................
........................................................................
........................................................................
```

Scoreboard Key:
"_" Waiting for Connection, "S" Starting up, "R" Reading Request,
"W" Sending Reply, "K" Keepalive (read), "D" DNS Lookup,
"C" Closing connection, "L" Logging, "G" Gracefully finishing,
"I" Idle cleanup of worker, "." Open slot with no current process

Srv	PID	Acc	M	CPU	SS	Req	Conn	Child	Slot	Client	VHost	Request
0-0	589	0/31/31	_	3.24	40	200	0.0	0.26	0.26	86.41.207.66	dissociatedpress.net	GET /feed/ HTTP/1.1
1-0	591	0/20/20	_	2.24	37	283	0.0	0.05	0.05	207.46.199.178	dissociatedpress.net	GET /robots.txt HTTP/1.1
2-0	592	0/31/31	_	3.79	72	0	0.0	0.26	0.26	67.195.115.24	dissociatedpress.net	GET /wp-content/themes/twentyten/style.css H
3-0	593	0/21/21	W	3.09	0	0	0.0	0.42	0.42	99.108.115.231	zonker.net	GET /server-status HTTP/1.1
4-0	664	0/3/19	_	0.75	38	200	0.0	0.04	0.23	140.211.169.61	dissociatedpress.net	GET /category/linuxdotcom/feed/ HTTP/1.1
5-0	-	0/0/14	.	2.68	740	0	0.0	0.00	0.19	127.0.0.1	kodos.zonker.net	OPTIONS * HTTP/1.0
6-0	597	0/21/21	_	4.30	72	0	0.0	0.23	0.23	67.195.115.24	dissociatedpress.net	GET /wp-content/plugins/wp-stats/stats-css.css'
7-0	600	0/22/22	_	3.28	10	0	0.0	0.73	0.73	99.108.115.231	dissociatedpress.net	GET /wp-includes/js/comment-reply.js?ver=20
8-0	601	0/20/20	_	4.17	38	462	0.0	0.19	0.19	140.211.169.61	dissociatedpress.net	GET /category/linuxdotcom/feed/ HTTP/1.1
9-0	619	0/30/30	_	1.15	0	0	0.0	0.09	0.09	99.108.115.231	zonker.net	GET /favicon.ico HTTP/1.1
10-0	-	0/0/1	.	0.00	739	0	0.0	0.00	0.00	127.0.0.1	kodos.zonker.net	OPTIONS * HTTP/1.0
11-0	634	12/31/31	K	2.65	3	3	0.0	0.06	0.06	99.108.115.231	dissociatedpress.net	GET /favicon.ico HTTP/1.1

Test ab (apache benchmark)

Uno de los datos fundamentales que debemos tener en cuenta a lo hora de implantar nuestro servicio es conocer cual es la carga máxima que sería capaz de soportar todo nuestro entramado. El objetivo de conocer este dato es, por un lado conocer las posibles limitaciones que puede tener nuestro hardware para tener en cuenta las actualizaciones más convenientes de los componentes del servidor. Por otro lado, sería la posible mejora del rendimiento de nuestra aplicación, ya que es posible que nuestra aplicación no sea todo lo eficiente que debería ser para servir todas las peticiones.

Se ha hablado tanto de una actualización hardware como software, pero evidentemente estos procesos dependerán de los resultados arrojados por dicho test. Estos resultados dependerán enormemente de los requisitos de los que conste nuestra aplicación, ya que no es lo mismo un sitio con alta carga, que uno con baja carga. Ni uno con número de visitas altas, que una con bajas. Por lo tanto, lo importante es definir unos requisitos de disponibilidad y tratar de alcanzarlos, comprobando su aplicación desde esta herramienta.

A pesar de que existen multitud de herramientas, tanto de software libre como privativas, en este proyecto se va a optar por utilizar Apache Benchmark. Dicha herramienta se incluye con la distribución del propio servidor Apache, con la misma licencia de utilización y distribución. Podemos probar el comando escribiendo “ab” en el terminal. En caso de que el sistema no

encuentre el comando, probablemente es que la ruta no se encuentre en el path, por lo que deberemos ejecutarlo colocándonos en el directorio bin de Apache.

Vamos a ver ahora las distintas opciones que podemos manejar con esta herramienta:

Uso: ab [opciones] [http[s]://]hostname[:port]/path

Las opciones son las siguientes:

-n peticiones	Número de peticiones que se van a llevar a cabo
-c peticiones	Número de peticiones que se envían al mismo tiempo
-t tiempo	Número de segundos máximos que se esperarán a las respuestas
-b tamaño	Tamaño en bytes del búffer TCP par enviar / recibir
-p postfile	Fichero que contiene los datos a enviar mediante POST. Recordar usar -T
-u putfile	Fichero que contiene los datos a enviar mediante PUT. Recordar usar -T
-T tipoMIME	Tipo MIME de el contenido que se va a enviar por ejemplo: 'application/x-www-form-urlencoded' Por defecto 'text/plain'
-v verbosity	Muestra una mayor cantidad de información por pantalla
-w	Imprime el resultado en tablas HTML
-i	Use HEAD en lugar de GET (esto es pedir la cabecera, en lugar del documento)
-x atributo	Cadena a insertar en los atributos de la tabla
-y atributo	Cadena a insertar en los atributos de la fila
-z atributo	Cadena a insertar en los atributos de celda o encabezado
-C atributo	Añade una cookie, por ejemplo. 'Apache=1234. (se puede repetir)
-H attribute	Añade una línea de cabecera aleatoria, por ejemplo: 'Accept-Encoding: gzip' Se inserta después de todas las líneas de cabecera normales.
-A attribute	Añade Autenticación Básica WWW, los atributos son usuario y password
-P attribute	Añade Autenticación para el proxy, los atributos son usuario y password
-X proxy:port	Dirección del servidor Proxy y puerto a usar
-V	Únicamente muestra el número de versión
-k	Usa conexiones HTTP 1.1
-d	No muestra porcentajes en la tabla guardada
-S	No muestra estimaciones confidenciales, ni advertencias
-g fichero	La salida se genera en formato gnuplot
-e fichero	La salida se genera en formato CSV
-r	No sale del socket si ha recibido algún error
-f protocolo	Especifica el protocolo SSL/TLS (SSL2, SSL3, TLS1, o ALL)

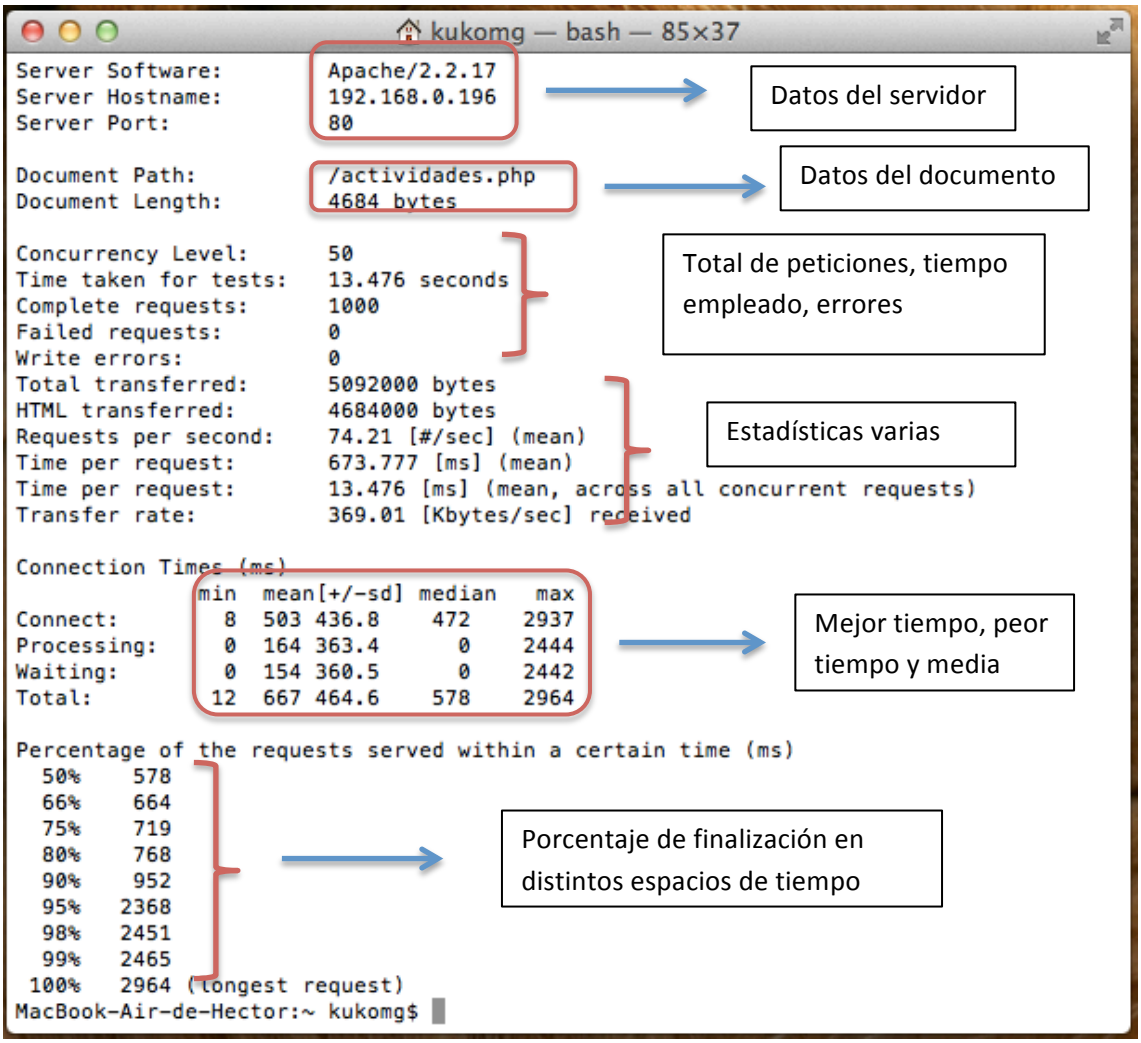
Dada la gran cantidad de opciones que hay vamos a ilustrar mejor esta utilidad con un ejemplo. En nuestro supuesto queremos ver el comportamiento de nuestro servidor realizando 1000 peticiones, 50 de ellas de forma simultánea. Esto significa que si ya hay 50

peticiones procesándose, hasta que no acabe una de ellas no podrá procesarse la siguiente. La dirección que se va a solicitar es http://192.168.0.196/actividades.php. El comando que deberemos usar para el ejemplo será:

```
$ ab -n 1000 -c 50 http://192.168.0.196/actividades.php
```

Hay que decir que la elección del recurso que se ha solicitado en el ejemplo anterior no es una casualidad, ya que contiene una mezcla de código html y php que debe ser ejecutado por el servidor en el momento que se realiza la petición por parte del cliente. En este caso también tiene una carga adicional que debe subsanar el servidor de base de datos, ya que se encarga de mostrar todas las tuplas de la tabla "actividades" por pantalla.

Una vez ejecutado el comando, la respuesta puede tardar un tiempo, dependiendo de los valores que hayamos introducido. El resultado obtenido es el siguiente:



Ahora vamos a pasar a analizar las distintas informaciones que nos proporciona esta herramienta, ya que a primera vista son un tanto confusas y no deja suficientemente claro qué indica cada valor.

Como algunos datos parecen obvios, no se van a explicar. Estos datos aparecen rotulados en la imagen anterior como “datos del servidor” y “datos del documento”. Del siguiente apartado la mayoría son los datos que hemos introducido nosotros como parámetros al ejecutable, pero aparece el primer dato importante aparece como “Time taken for tests” y nos indica el tiempo máximo que se ha necesitado para llevar a cabo la totalidad de las peticiones.

En el siguiente apartado vemos datos como el total de datos transferidos, el total perteneciente únicamente a documentos HTML, el número de respuestas por segundo, el número de respuestas por segundo cuando están activas todas las conexiones simultáneas, y la tasa de transferencia. Los datos más relevantes de este apartado son el tiempo que tardamos en recibir una respuesta (que nos ayudará a evaluar si debemos optimizar nuestro sistema, tanto en hardware como en software), y el ratio de transferencia ya que nos permite evaluar la configuración TCP/IP y la red en general.

En el apartado de los tiempos de conexión podemos analizar más a fondo el desglose de los tiempos que han tardado en realizarse las distintas fases de la prueba. Estas fases son la conexión, el proceso, y el tiempo de espera. A su vez nos indica el tiempo mínimo y máximo registrado, y la media.

El último apartado indica los porcentajes de finalización en los distintos espacios de tiempo, y puede ser el apartado más difícil de comprender. Básicamente el ejemplo anterior indica que el 50% de las peticiones han acabado en 578ms, el 66% han acabado en 664ms, 75% en 719ms, y así sucesivamente hasta llegar al 100%.

5.2.4.2.-Entorno de producción

Registro de logs

Uno de los datos más valiosos que disponen los administradores de sistemas es la propia información que se va generando mediante el uso continuo de los propios sistemas. Esta información se genera en el transcurso de distintos eventos que van ocurriendo, como peticiones, accesos comunes, errores, peticiones incorrectas, etc.

La primera reflexión a la que podemos llegar es que para que se puedan analizar, primero deberán ser creados, y no sólo eso, si no que deberán estar en un formato adecuado para ser procesado más tarde, y deberá incluir tantos datos como necesitemos para disponer de datos suficientes para el análisis.

La primera directiva que podemos analizar es “LogLevel”, que está incluido en el núcleo del servidor Apache, por lo que no deberemos instalar nada para comenzar a utilizarla. Su

motivación es controlar el nivel de detalle con el que se guardarán los registros Log. Vamos a ver los distintos niveles que podemos configurar:

Nivel	Descripción	Ejemplo
amerg	Emergencia	"Un proceso hijo no puede continuar. Finalizando"
alert	Hay que hace algo ya	"getpwuid: imposible determinar nombre a partir uid"
crit	Errores críticos	"socket: No se encontró un socket adecuado"
error	Errores	"Final prematuro de la cabecera del script"
warn	Advertencias	"El proceso X no ha terminado, enviando SIGHUP"
notice	A tener en cuenta	"httpd: intentando un volcado de memoria en…"
info	Información	"El servidor parece ocupado, quizás desee configurar…"
Debug	Para desarrolladores	"Abriendo el fichero de configuración"

Estos distintos niveles se han extraído de la web oficial de Apache, y puede consultarse en la siguiente URL: http://httpd.apache.org/docs/2.0/es/mod/core.html#loglevel

Ahora que ya se ha explicado la configuración general de logs de Apache, vamos a intentar explicar la forma de crear logs mucho más específicos de forma que podamos añadir a nuestro propio criterio los campos más convenientes para nuestro posterior análisis. Para definir las distintas reglas se utiliza la directiva "LogFormat" cuya sintaxis es la siguiente: `LogFormat format [alias]`

Las opciones que se pueden especificar en el modificador "format" son muy amplias y se especifican en la siguiente tabla:

Nomenclatura	Descripción
%a	Dirección IP del cliente
%A	Dirección IP del servidor
%B	Bytes enviados (sin cabeceras http) 0 si no existe
%b	Bytes enviados (sin cabeceras http) - si no existe
%c	Estado de la conexión: "X" si la conexión fue abortada por el cliente "+" si se utiliza una conexión HTTP 1.1 "-" si se cerró la conexión tras la respuesta
%{cook}C	Contenido de la cookie llamada "cook"
%D	Milisegundos en completar la respuesta
%{var}e	Contenido de la variable de entorno "var"
%f	Nombre del fichero solicitado
%h	Host del cliente (si HostnameLookups está a On)
%H	Protocolo en el que se ha realizado la solicitud
%{IncomingHeader}I	Contenido de la cabecera enviada al servidor
%I	Muestra identidad del cliente (si IdentityCheck está activada)

%m	Método de la solicitud procesada (GET, POST, PUT,…)
%{OutgoingHeader}o	Cabecera que se envía al cliente en la respuesta
%p	Puerto al que se recibió la solicitud
%P	ID del proceso hijo del que sirvió la solicitud
%q	Contenido de la cadena de consulta
%r	Primera línea de la solicitud
%s	Estado devuelto por el servidor
%t	Tiempo de la solicitud (formato CLF)
%{formato}t	Tiempo en el formato especificado (consultar strftime)
%T	Nº de segundos que tarda el servidor en contestar
%u	Nombre usuario en caso de utilizar autenticación HTTP Basic
%U	URL solicitada al servidor
%v	Nombre del host servidor
%V	Nombre del servidor para cada directiva UseCanonicalName
%{cabecera}i	Muestra la cabecera especificada entre corchetes

Una vez tenemos definidas las distintas reglas, podemos aplicarlas a un fichero de log para que empiecen a volcarse de inmediato, o bien podemos dejarlas especificadas para un futuro. La forma de aplicarlas a un fichero log es mediante las directivas “TransferLog” y “CustomLog”

Analizar logs

Hasta ahora se ha profundizado en como recabar la información que deseamos registrar en nuestros fichero log, pero tan importante es registrarlo como posteriormente analizarlo para extraer la máxima cantidad de información posible.

La primera posibilidad es consultar el fichero de log mediante un editor de texto común como puede ser Gedit en el caso de utilizar Gnome, o Kate en el caso de utilizar KDE. Esta posibilidad implica un gran esfuerzo de comprensión por nuestra parte, ya que probablemente tengamos una gran cantidad de información difícil de analizar.

Como se puede apreciar en la captura es muy difícil obtener información del registro, ya que hay demasiada información para que esta nos sea útil. En este caso la imagen del ejemplo es del fichero de accesos (en mi caso /var/log/apache2/access.log). Parece lógico por lo tanto que deberemos encontrar la forma de filtrar tal cantidad de información.

Vamos a suponer que nuestro log dispone de 5 columnas, de modo similar al ejemplo que se muestra en la ilustración anterior. La primera columna es la IP que realiza la petición, la segunda la fecha y hora, el tercero la petición HTTP, el cuarto es el documento que se ha solicitado, y el quinto es la versión del navegador del cliente.

Para filtrar por columna podemos utilizar el siguiente comando (donde deberemos sustituir 1 por el número de columna que deseemos mostrar, y la ruta del fichero por nuestro log):

```
$ cat /var/log/apache2/access.log | awk `{print $1}'
```

También podemos filtrar por dirección IP, por ejemplo mostrando los de nuestra propia red que en este caso es 192.168.0.x. Lo haremos de la siguiente forma:

```
$ cat /var/log/apache2/access.log | grep 192.168.0.
```

Otra posibilidad sería la de filtrar por el navegador que ha realizado la petición, incluso

podríamos afinar más incluyendo la versión (en este ejemplo mostramos únicamente los de Firefox 5):

```
$ cat /var/log/apache2/access.log | grep Firefox/5.0
```

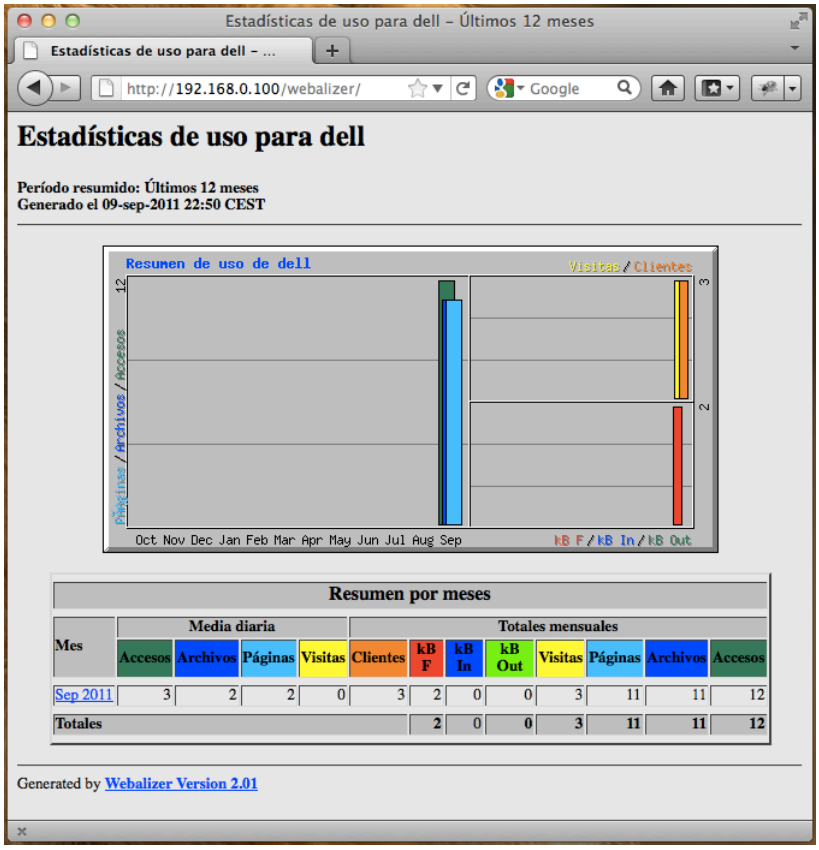
Otra opción más avanzada es utilizar un software especializado y proporcionado por un tercero que nos ofrezca un análisis más detallado. Existen gran cantidad de aplicaciones que podemos utilizar, ya que se basan en un formato común de logs llamado CLF. Entre ellas se encuentran las siguientes: webalizer, WebTrends, Wusage, Wwwstat, Analog, http-analyze, Pwebstats, WebStat Explorer, AcessWatch.

En nuestro caso vamos a analizar la herramienta webalizer. Esta herramienta se distribuye bajo la licencia GPL, por lo que podemos utilizarlo y distribuirlo libremente. Su funcionamiento se basa en el análisis de los logs que Apache va generando continuamente. Su instalación desde una distribución Debian / Ubuntu se realiza de la siguiente forma:

```
# apt-get install webalizer
```

Si tanto Apache, como Webalizer han sido instalado desde los repositorios no tendremos ningún problema en cuanto a configuración, puesto que todo funcionará de forma automática. De no ser así, deberemos modificar el fichero de configuración /etc/webalizer/webalizer.conf, e indicarle la ruta donde se encuentran los logs de Apache.

El siguiente paso será ejecutar webalizer insertando el comando en la consola. Como resultado del comando nos creará un nuevo directorio llamado webalizer en el directorio de publicación de Apache. Deberemos ser cuidadosos y restringir el acceso a este directorio para evitar que toda la información quede visible en el exterior. Ahora podremos acceder desde una navegador y comprobar el resultado de localhost:8888.



En esta primera pantalla pueden verse datos como el total de accesos, clientes, KB transmitidos, páginas vistas, etc. También puede verse dicha información en forma de gráfico, aunque en este ejemplo el fichero de log de Apache no disponía de suficientes datos como para generar estadísticas de interés.

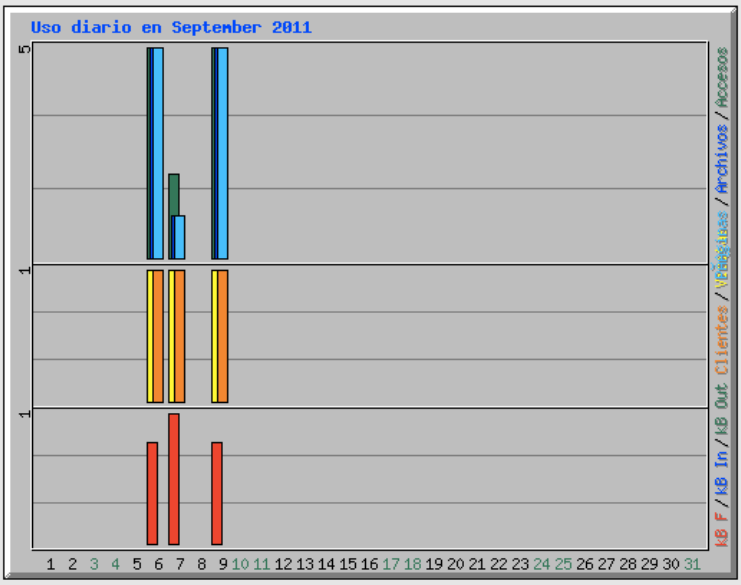
A continuación se muestra un ejemplo de la información que muestra esta herramienta. Dado el gran número de opciones que dispone el programa, por cuestiones de espacio es imposible mostrarlas todas, por lo cual se muestra una pequeña parte:

Estadísticas de uso para dell

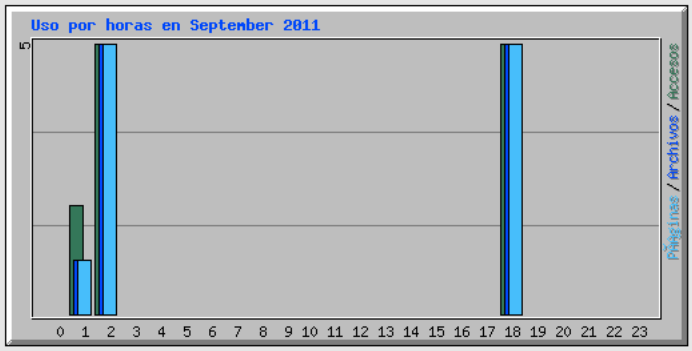
Período resumido: September 2011
Generado el 09-sep-2011 23:12 CEST

[Estadísticas diarias] [Estadísticas por horas] [URLs] [Entrada] [Salida] [Clientes] [Enlaces origen] [Búsqueda] [Navegadores] [Locations]

Estadísticas mensuales de September 2011		
Total Accesos		12
Total Archivos		11
Total Páginas		11
Total Visitas		3
Total kB Files		2
Total kB In		0
Total kB Out		0
Total Clientes		3
Total URLs		2
Total Enlaces origen		1
Total Navegadores		2
	Media	Max
Accesos por Hora	0	5
Accesos por Día	3	5
Archivos por Día	2	5
Páginas por Día	2	5
Visitas por Día	0	1
kB Files per Day	1	1
kB In per Day	0	0
kB Out per Day	0	0
Accesos por código de respuesta		
200 - OK		11
404 - No se encuentra		1



Estadísticas diarias de September 2011															
Día	Accesos		Archivos		Páginas		Visitas		Clientes		kB F		kB In		kB Out	
6	5	41,67%	5	45,45%	5	45,45%	1	33,33%	1	33,33%	1	30,30%	0	0,00%	0	0,00%
7	2	16,67%	1	9,09%	1	9,09%	1	33,33%	1	33,33%	1	39,39%	0	0,00%	0	0,00%
8	0	0,00%	0	0,00%	0	0,00%	0	0,00%	0	0,00%	0	0,00%	0	0,00%	0	0,00%
9	5	41,67%	5	45,45%	5	45,45%	1	33,33%	1	33,33%	1	30,30%	0	0,00%	0	0,00%



Estadísticas por hora de September 2011

Hora	Accesos			Archivos			Páginas			kB F			kB In			kB Out		
	Media	Total		Media	Total		Media	Total		Media	Total		Media	Total		Media	Total	
0	0	0	0,00%	0	0	0,00%	0	0	0,00%	0	0	0,00%	0	0	0,00%	0	0	0,00%
1	0	2	16,67%	0	1	9,09%	0	1	9,09%	0	1	39,39%	0	0	0,00%	0	0	0,00%
2	1	5	41,67%	1	5	45,45%	1	5	45,45%	0	1	30,30%	0	0	0,00%	0	0	0,00%
3	0	0	0,00%	0	0	0,00%	0	0	0,00%	0	0	0,00%	0	0	0,00%	0	0	0,00%
4	0	0	0,00%	0	0	0,00%	0	0	0,00%	0	0	0,00%	0	0	0,00%	0	0	0,00%
5	0	0	0,00%	0	0	0,00%	0	0	0,00%	0	0	0,00%	0	0	0,00%	0	0	0,00%
6	0	0	0,00%	0	0	0,00%	0	0	0,00%	0	0	0,00%	0	0	0,00%	0	0	0,00%
7	0	0	0,00%	0	0	0,00%	0	0	0,00%	0	0	0,00%	0	0	0,00%	0	0	0,00%
8	0	0	0,00%	0	0	0,00%	0	0	0,00%	0	0	0,00%	0	0	0,00%	0	0	0,00%
9	0	0	0,00%	0	0	0,00%	0	0	0,00%	0	0	0,00%	0	0	0,00%	0	0	0,00%
10	0	0	0,00%	0	0	0,00%	0	0	0,00%	0	0	0,00%	0	0	0,00%	0	0	0,00%
11	0	0	0,00%	0	0	0,00%	0	0	0,00%	0	0	0,00%	0	0	0,00%	0	0	0,00%
12	0	0	0,00%	0	0	0,00%	0	0	0,00%	0	0	0,00%	0	0	0,00%	0	0	0,00%

Los 2 primeros de un total de 2 URLs

#	Accesos		kB F		kB In		kB Out		URL
1	10	83,33%	1	60,61%	0	0,00%	0	0,00%	*
2	1	8,33%	0	19,34%	0	0,00%	0	0,00%	/

Los 2 primeros de un total de 2 URLs Por kB F

#	Accesos		kB F		kB In		kB Out		URL
1	10	83,33%	1	60,61%	0	0,00%	0	0,00%	*
2	1	8,33%	0	19,34%	0	0,00%	0	0,00%	/

Los 2 primeros de un total de 2 Total de Páginas de Entrada

#	Accesos		Visitas		URL
1	10	83,33%	2	66,67%	*
2	1	8,33%	1	33,33%	/

Los 2 primeros de un total de 2 Total de Páginas de Salida

#	Accesos		Visitas		URL
1	10	83,33%	1	50,00%	*
2	1	8,33%	1	50,00%	/

Los 3 primeros de un total de 3 clientes

#	Accesos		Archivos		kB F		kB In		kB Out		Visitas		Máquina
1	5	41,67%	5	45,45%	1	30,30%	0	0,00%	0	0,00%	1	33,33%	127.0.0.1
2	5	41,67%	5	45,45%	1	30,30%	0	0,00%	0	0,00%	1	33,33%	Unknown
3	2	16,67%	1	9,09%	1	39,39%	0	0,00%	0	0,00%	1	33,33%	192.168.0.100

Los 3 primeros de un total de 3 clientes Por kB F

#	Accesos		Archivos		kB F		kB In		kB Out		Visitas		Máquina

Mantenimiento logs

No vamos a negar que el uso de logs es un gran avance para el administrador del servidor, pero tenemos que tener en cuenta que no es oro todo lo que reluce y tenemos también algunos inconvenientes que deberemos tener en cuenta.

Uno de ellos es el mantenimiento que deberemos tener de los registros, ya que estos pueden crecer indefinidamente y llegar a ser realmente grandes, pudiendo provocar problemas de espacio en disco, cuotas, o muchos otros. La solución a este problema es la rotación de los registros.

Las dos herramientas más importantes se llaman ***rotatelog*** que viene incluida con la propia distribución de Apache, y ***logrotate*** que estará disponible en nuestro sistema Linux.

A continuación vamos a utilizar un ejemplo de cada herramienta. En primer lugar vamos a ver rotatelog, suponiendo que deseamos rotar los registros cada 24 horas. Para ello utilizaremos la siguiente directiva:

```
TransferLog "| /ruta/de/rotatelogs /var/log/apache2 86400"
```

El primer parámetro se refiere a la ruta donde se encuentra el ejecutable "rotatelogs", que suele estar en el directorio bin de nuestra distribución de Apache, el segundo indica la ruta donde se encuentra el directorio que almacena los logs, y el tercero el tiempo que transcurrirá entre rotaciones expresado en segundos.

Por último vamos a analizar logrotate que debería venir instalada en nuestra distribución, para comprobarlo simplemente ejecutarlo a ver si reconoce el comando (en caso de aparecer ejecutar "apt-get install logrotate").

Esta herramienta puede utilizarse con cualquier demonio que genere logs, por lo que no es exclusivo para Apache. Puede comprobar si existe una configuración específica para nuestro servidor mediante el siguiente comando:

```
$ cat /etc/logrotate.d/apache2
```

En caso de que ya exista una configuración veremos una imagen como la siguiente:

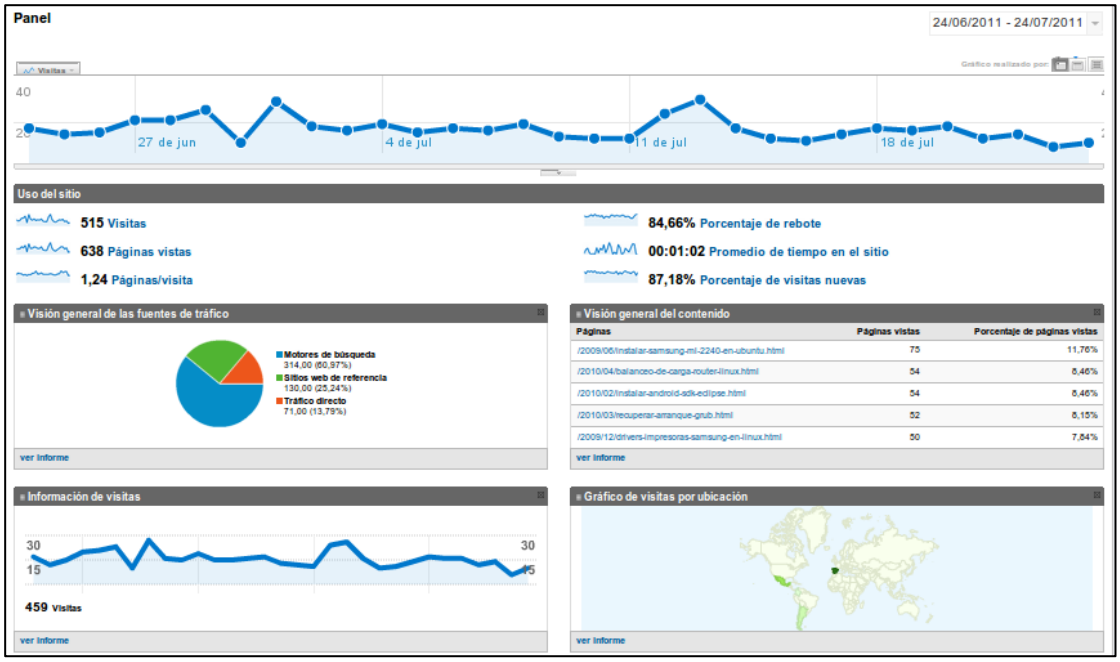
Podemos modificar tanto el lugar donde se encuentran los logs de Apache, como las distintas opciones entre las que se encuentran los permisos, la compresión, el transcurso de tiempo entre rotaciones, etc.

Google Analytics

Google Analytics es una herramienta on-line que nos permite conocer una gran cantidad de datos sobre la audiencia de nuestro sitio web, lo que nos permite conocer la afluencia de tráfico entre otros muchos datos.

Es importante que antes de tratar con un determinado colectivo de potenciales clientes (en adelante target) recolectemos los datos suficientes para intentar ser atractivos a sus ojos. De esta forma intentaremos amoldarnos a sus gustos y preferencias para intentar captar mejor su atención. Todo esto está muy bien pero no serviría de nada si no hacemos un seguimiento a nuestra campaña de marketing.

Ahí es donde entra en juego esta herramienta. En la imagen que se puede ver a continuación, se observa un rápido resumen de algunos datos de una web de ejemplo. En este resumen se pueden ver gráficos que permiten visualizar los datos más importantes como pueden ser el número de visitas de los últimos 30 días o el origen de nuestras visitas.



En el panel superior podemos ver precisamente el número de visitantes únicos diario durante los últimos 30 días (valor que se puede modificar en la esquina superior izquierda). Esto nos permite ver los días de la semana tanto de mayor, como de menor afluencia de visitantes. Es importante a la hora de efectuar campañas de marketing, ya que estamos analizando las conductas de nuestros visitantes.

En el segundo panel se pueden visualizar una serie de valores que referidos al intervalo de tiempo elegido más arriba. El primero es el índice numérico de visitas a nuestro sitio que nos ofrece un valor aproximado de la visibilidad de nuestro sitio. También aparecen las páginas vistas, que a diferencia del valor anterior cuenta el número de visitas a las distintas páginas de nuestro sitio aún siendo realizadas por el mismo visitante. Las páginas por visita nos ofrece una idea del interés que genera nuestra página en los visitantes una vez que estos ya han entrado. Otros valores importantes son el porcentaje de rebotes que son básicamente las personas que han salido de nuestra web sin efectuar ninguna acción, el promedio de tiempo en el sitio y el porcentaje de visitas nuevas.

El tercer panel está separado a su vez en dos, en los que se muestran las fuentes del tráfico y la visión general del contenido. Las fuentes de tráfico se dividen en motores de búsqueda, que son aquellas visitas que provienen de algún buscado, sitios web de referencia, que son aquellas visitas que provienen de un enlace de cualquier página, y tráfico directo que son principalmente aquellos usuarios que escriben la dirección en la barra de direcciones o en marcadores. El apartado de visión general del contenido muestra el top 5 de los documentos más visitado indicando el número de visitas y el porcentaje sobre el total del sitio web.

El último panel también está dividido en el panel de visitas y el gráfico de visitas por ubicación. El primero no nos ofrece una información muy distinta a la del primer panel, salvo por la posibilidad de ver el informe completo. El segundo ofrece un contenido muy importante y es

las visitas por ubicación donde podemos ver cual es el público que más afluencia tiene en nuestro sitio.

A pesar de que nos parezca que de un simple vistazo tenemos todos los datos necesarios, esto es sólo el principio, ya que en cada apartado existe un enlace para ampliar más información y ver un detalle pormenorizado de cada uno de los aspectos que analiza este software. Se van a analizar los apartados que más nos pueden interesar para el transcurso de nuestro proyecto.

5.2.5.- Optimización del servidor Apache

5.2.5.1.- Elección del hardware

CPU

La elección de la CPU depende del tipo de utilización que tenga nuestro servidor Apache, ya que a priori no es una de las decisiones cruciales que afecten al rendimiento en primera instancia. Si nuestra aplicación son mayormente páginas estáticas, la CPU no va a ser un factor determinante. Si por el contrario, nuestra aplicación está compuesta mayormente por páginas dinámicas, si que se necesitará una CPU potente.

Existen CPUs específicas, especialmente diseñadas para su uso en servidores. Dichas CPUs cuentan con mayores tamaños de caché, alto rendimiento, comunicación con otros procesadores, mayor número de núcleos, alta velocidad en los buses, etc. Todo esto hace que estén enfocados para tareas de alto procesamiento. Ejemplos de estas CPUs preparadas específicamente para servidores son Intel Xeon y AMD Opteron.

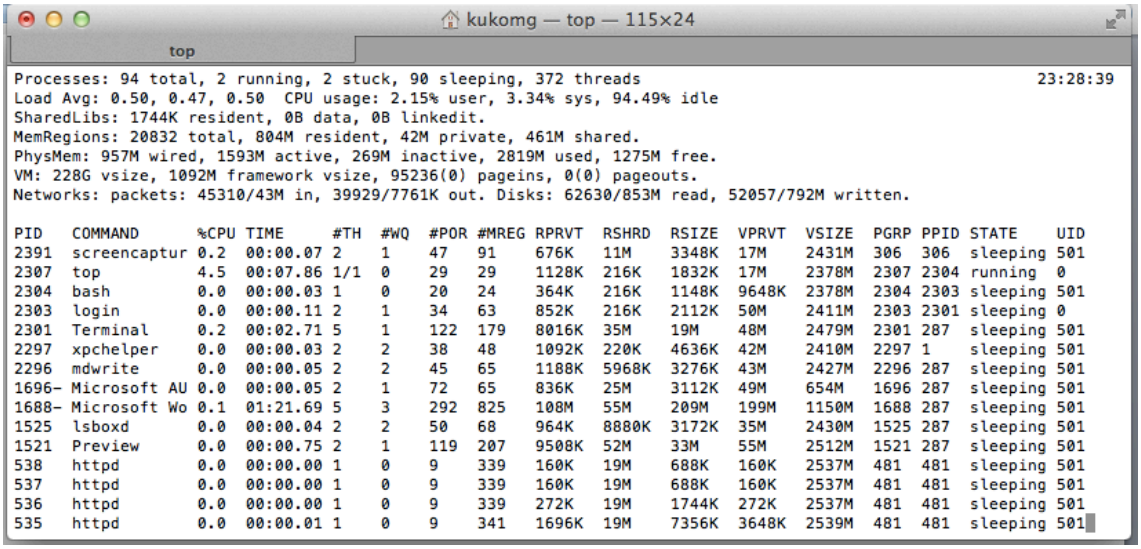
Memoria

A la hora de elegir la memoria RAM debemos tener en cuenta que nuestra necesidad de rendimiento puede venir tanto por falta de velocidad, como por falta de cantidad. Si necesitamos velocidad, deberemos adecuarnos a los buses utilizados por la placa base y por la CPU. Si nuestra necesidad viene por la cantidad, simplemente deberemos instalar más capacidad, ya que las placas bases a actuales suelen aceptar gran cantidad de esta.

El tipo de memoria actuales más importantes es DDR 3, basado en la tecnología SDRAM. Esta es la mejor combinación para procesadores mutinúcleo (2, 4 u 6 núcleos). El tamaño máximo teórico de fabricación de estos módulos es de 16 GiB.

Es importante el uso de herramientas específicas a la hora de contabilizar la cantidad de memoria que necesitaremos instalar en nuestro servidor. En el caso de sistemas Unix/Linux existen herramientas como “top” que permite analizar el uso de memoria física y de memoria virtual en un momento determinado. La idea es realizar distintas pruebas de rendimiento

efectuando peticiones y midiendo el comportamiento de la memoria del sistema. Un ejemplo de captura de esta herramienta se puede ver en la siguiente imagen:



Disco Duro

El disco duro es el elemento que malgasta más cantidad de tiempo en el acceso a los datos almacenados en él, ya que es el dispositivo más lento. Por ello, es uno de los elementos a los que debemos prestarle más interés a la hora de su elección. Dentro de las tecnologías por las que podemos decantarnos deberemos interesarnos por SATA o SCSI, ya que ofrecen velocidades de transferencia más altas. En cuanto a la tecnología interna más rápida se encuentran los discos SSD que evitan el uso de partes móviles (y por lo tanto la velocidad de acceso es prácticamente instantánea), si no podemos permitirnos esta tecnología (debido a su alto precio y baja capacidad), deberemos tener en cuenta que la velocidad de rotación sea al menos de 10.000 rpm.

Un servidor de hosting suele usar gran cantidad de capacidad en datos almacenados, entre los que se incluye el propio sistema operativo, los distintos servidores, la propia aplicación web, imágenes, videos, infografías, etc. Por ello es importante tenerlo en cuenta para elegir una capacidad adecuada a nuestros requisitos.

Otro aspecto a tener en cuenta viene dado por la capacidad de recuperación de los datos almacenados en los discos duros. Hay que prestar especial atención en este aspecto ya que los discos duros son puntos habituales de fallos dada su gran fragilidad. Lo más habitual es el uso de sistemas RAID que permita la replicación por hardware de toda la información de forma transparente. Existen diversos tipos de RAID que podemos usar, por lo que deberemos elegir uno que nos ofrezca un precio y características adecuadas a nuestras necesidades.

5.3.-Sistema de Base de datos

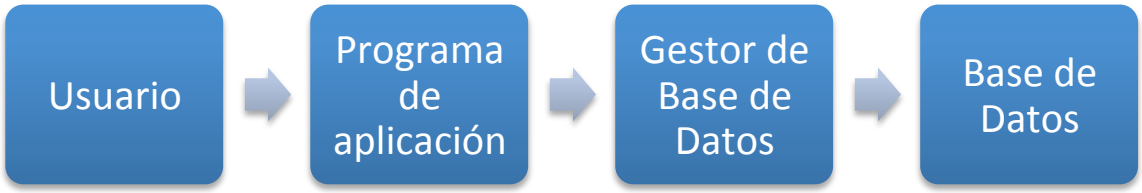
No existe un sistema de base de datos único, ni uno específico que tengamos que usar para nuestro proyecto en particular. Esto quiere decir que en el mercado disponemos de gran cantidad de alternativas que nos pueden servir de plataforma para implementar nuestra base de datos.

Para la gestión de todos los datos que va a necesitar nuestro sistema web vamos a necesitar dos elementos diferenciados. El primero va a ser la propia información que va a ser almacenada en el formato de la base de datos que elijamos. Por otro lado tendremos el SGBD (sistema gestor de base de datos) que será el software encargado de administrar nuestras peticiones y llevarlas a cabo, consultando o modificando los datos almacenados.

A decir verdad esta no es la única forma de almacenar los datos que puede necesitar nuestra aplicación. Otra posibilidad, aunque ya en desuso, sería el almacenamiento en ficheros de texto plano, el problema de esta solución sería la duplicidad de registros.

La ventaja de un SGBD con respecto a otro tipo de soluciones para almacenar grandes cantidades de datos, es que la gestión de los recursos, permisos, concurrencia, etc. es llevada a cabo totalmente por el SGBD.

Para lograr este objetivo, los SGBD proponen una serie de interfaces de programación que permiten al usuario (o software de terceros) realizar todo tipo de operaciones, encargándose de mantener la integridad, confidencialidad y seguridad. Por ello se define la siguiente estructura de acceso a los datos almacenados:



Llegados a este punto y habiendo recalcado la necesidad de utilización de un SGBD, el siguiente paso es ver qué puede ofrecernos el mercado y decidirnos por uno de ellos para la utilización en nuestra aplicación.

La diferenciación más importante que podemos hacer en estos momentos y dado el carácter de este proyecto fin de carrera, es la condición de libre o no libre. Existen numerosos SGBD cuyo uso es completamente libre (aunque algunos demuestran ciertas carencias para usos concretos) como pueden ser MySQL, PostgreSQL, Firebird, SQLite, Apache Derby. En el extremo opuesto y en su condición de no-libres podemos mencionar MySQL, dBase, Informix, Microsoft SQL Server, Oracle y Paradox.

Dada la gran cantidad de documentación disponible para gestionar MySQL, y las facilidades disponibles para su trabajo junto a Apache, en este proyecto se ha optado por este SGBD. Aunque probablemente, si desde un punto de vista objetivo tuviera que escoger uno de la lista anterior elegiría PostgreSQL ya que es dirigido completamente por una comunidad de desarrolladores ajena a cualquier empresa o interés comercial.

A pesar de todo ello, con una aplicación bien desarrollada y estructurada por capas, la adopción de un SGBD u otro implicaría diferencias muy pequeñas. Tanto, que la migración de uno a otro podría significar únicamente el cambio de código de una única u dos funciones en todo la aplicación.

Entre principales características de MySQL está el amplio soporte para distintos lenguajes de programación, alto rendimiento gracias al uso de hilos por parte del kernel, gran escalabilidad gracias a soporte de grandes bases de datos y gran cantidad de comandos para gestionar las bases de datos (chequeo, optimización, reparación…)

5.3.1.- Instalación MySQL

La instalación de MySQL puede de realizarse de diversas formas (como ya se ha visto con Apache y PHP). Una permite la descarga del código fuente para su compilación e instalación, la otra opción es descargarse un paquete preparado para funcionar en nuestra distribución en concreto. La elección entre una u otra queda a nuestro criterio, existiendo los mismos argumentos que en casos anteriores.

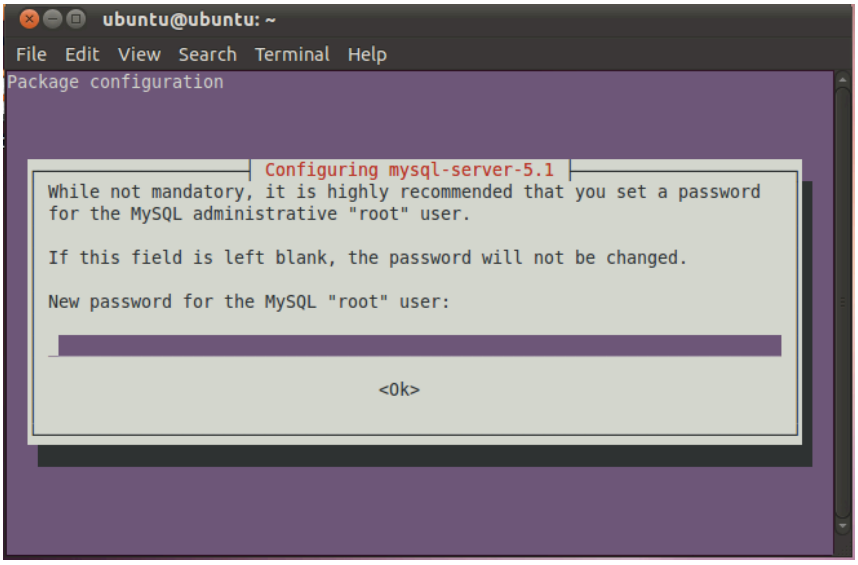
Para realizar una instalación desde el código fuente, el primer caso es descargar la versión desde el sitio web oficial: http//dev.mysql.com/downloads. El fichero descargado es un comprimido tar con la forma mysql-VERSION.tar.gz. Aunque no se ha probado tal configuración se han extraído las siguientes indicaciones del sitio web oficial (http://dev.mysql.com/doc/refman/5.0/es/quick-install.html):

```
$ groupadd mysql
$ useradd -g mysql mysql
$ gunzip < mysql-VERSION.tar.gz | tar -xvf -
$ cd mysql-VERSION
$ ./configure --prefix=/usr/local/mysql
$ make
$ make install
$ cp support-files/my-medium.cnf /etc/my.cnf
$ cd /usr/local/mysql
$ bin/mysql_install_db --user=mysql
$ chown -R root  .
$ chown -R mysql var
$ chgrp -R mysql .
$ bin/mysqld_safe --user=mysql &
```

Para nuestro caso en particular se va a utilizar los paquetes predeterminados de nuestra distribución que se pueden encontrar en los repositorios. Para ello debemos ejecutar el comando:

```
$ sudo apt-get install mysql-server
```

Durante el proceso de instalación se nos solicitará establecer una contraseña para el usuario root de MySQL. Es importante recordar esta contraseña para futuros trabajos de administración:



Todavía queda un paso importante para finalizar la instalación de MySQL, y es instalar el módulo para PHP para que las consultas que realicen dichas aplicaciones ataquen directamente contra el SGBD. Para ello, y suponiendo que no lo hayamos instalado antes, instalaremos el siguiente paquete:

```
$ sudo apt-get install php5-mysql
```

Si has instalado PHP desde las fuentes, asegúrate de haberlo compilado con la opción mysql (--with-mysql). Como último paso queda comprobar si dicho módulo está activado o no, para ello comprobaremos el fichero de configuración de php (en mi caso

/etc/php5/apache2/php.ini) que la línea con el contenido "extensión=mysql.so" no esté comentada con un # al principio.

5.3.2.- Herramienta phpMyAdmin

Para facilitar la gestión de la base de datos vamos a utilizar una herramienta de software libre llamada phpMyAdmin. Dicha herramienta funciona a través de nuestro servidor web, puesto que está escrita en PHP. Algunas de las operaciones más utilizadas son el manejo de la base de datos, tablas, campos, relaciones, índices, usuarios, permisos, etc.

Otra ventaja que podemos obtener mediante la utilización de esta herramienta, es que podemos ejecutar cualquier sentencia SQL directamente en la interfaz. De esta forma podemos comprobar cuáles son sus resultados antes de implementarla en nuestra aplicación.

Su uso es realmente intuitivo, basado en una interfaz web extremadamente sencilla. Además está traducido a una gran cantidad de idiomas entre los que se encuentra el español, catalán, vasco y gallego.

Los requisitos que debemos tener en cuenta para instalar nuestra aplicación son los siguientes:

- PHP versión 5.2 o superior, con soporte de sesión, extensión SPL y soporte JSON.
- MySQL 5.0 o superior.
- Navegador web con cookies activadas.

Además es posible que necesitemos alguna extensión adicional para hacer uso de la totalidad de las funciones de la herramienta. Para ello deberemos consultar la siguiente URL de la web oficial: http://www.phpmyadmin.net/localized_docs/es/Documentation.html#require

5.3.2.1- Instalación

Para la instalación de esta herramienta vamos a explicar dos métodos distintos (ambos muy sencillos):

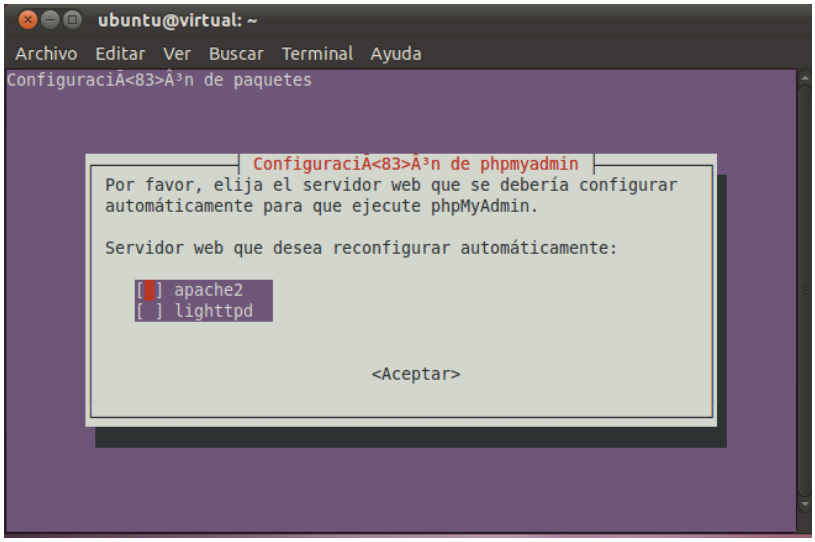
Método 1: Repositorios de nuestra distribución

Las distribuciones Linux están preparadas para su uso como servidor web, por lo que muchas de ellas ya disponen en sus repositorios de todos los paquetes necesarios, tanto para la instalación de los mismos, como para su gestión. Entre ellos se encuentra esta herramienta.

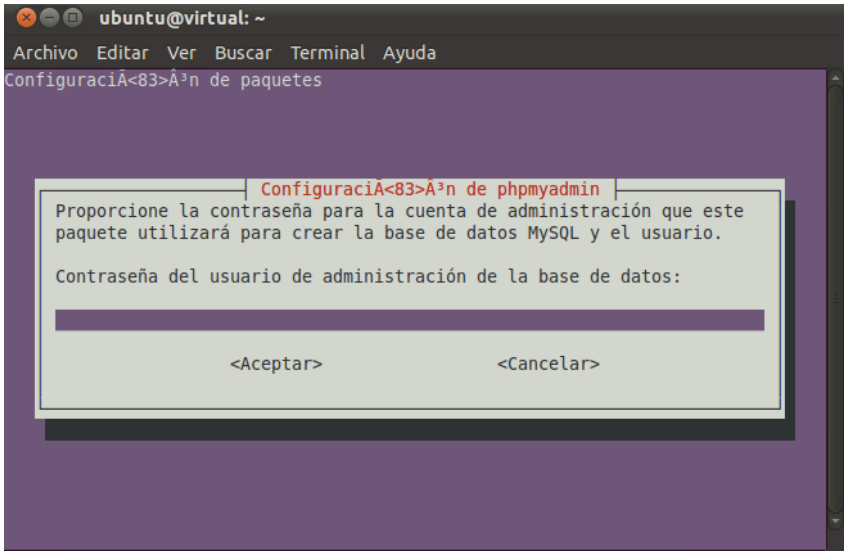
Su instalación es muy sencilla, ya que únicamente deberemos escribir en la terminal el comando de instalación para que esta quede funcionando correctamente. Dicho comando es el siguiente (para Debian/Ubuntu y derivados) :

```
# apt-get install phpmyadmin
```

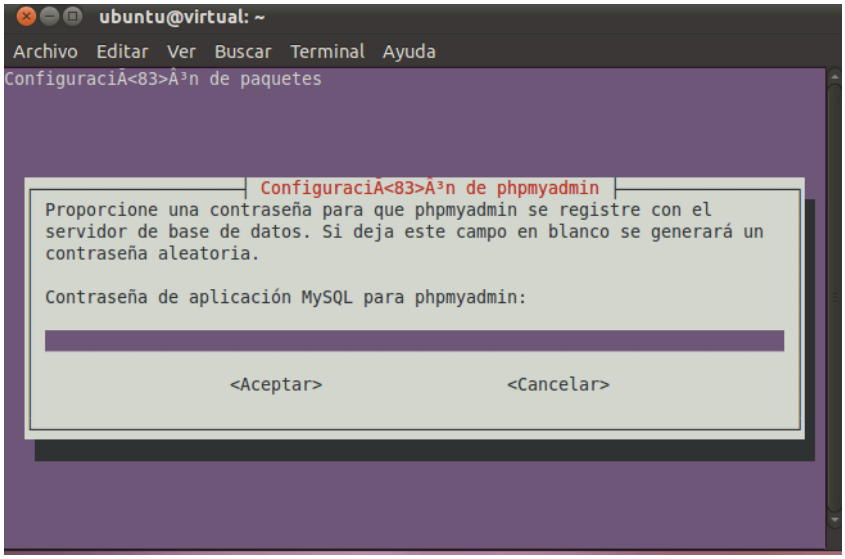
El proceso de instalación se ejecutará ahora realizando un par de preguntas:



La primera intervención será relativa al servidor web con el que desea que se configure phpMyAdmin. En nuestro caso seleccionaremos apache2 y proseguiremos con el resto de la instalación.



Ahora deberemos introducir la contraseña de administración de la base de datos. Esta contraseña fue introducida en el proceso de instalación del servidor MySQL.



Por último deberemos introducir una contraseña para gestionar phpmyadmin en un futuro.

Método 2: A partir de los ficheros oficiales

En la página de Descargas de la web oficial podemos encontrar los paquetes que podemos descargar para instalar phpMyAdmin. Estos paquetes tienen un nombre similar a phpMyAdmin_x.x.x-all-languages.tar.gz (La nomenclatura x.x.x corresponde a la versión). Para ello deberemos acceder a http://www.phpmyadmin.net/home_page/downloads.php.

Una vez descargado el fichero, deberemos copiarlo al directorio de publicación de nuestro servidor Apache. Una vez copiado lo descomprimiremos mediante el siguiente comando:

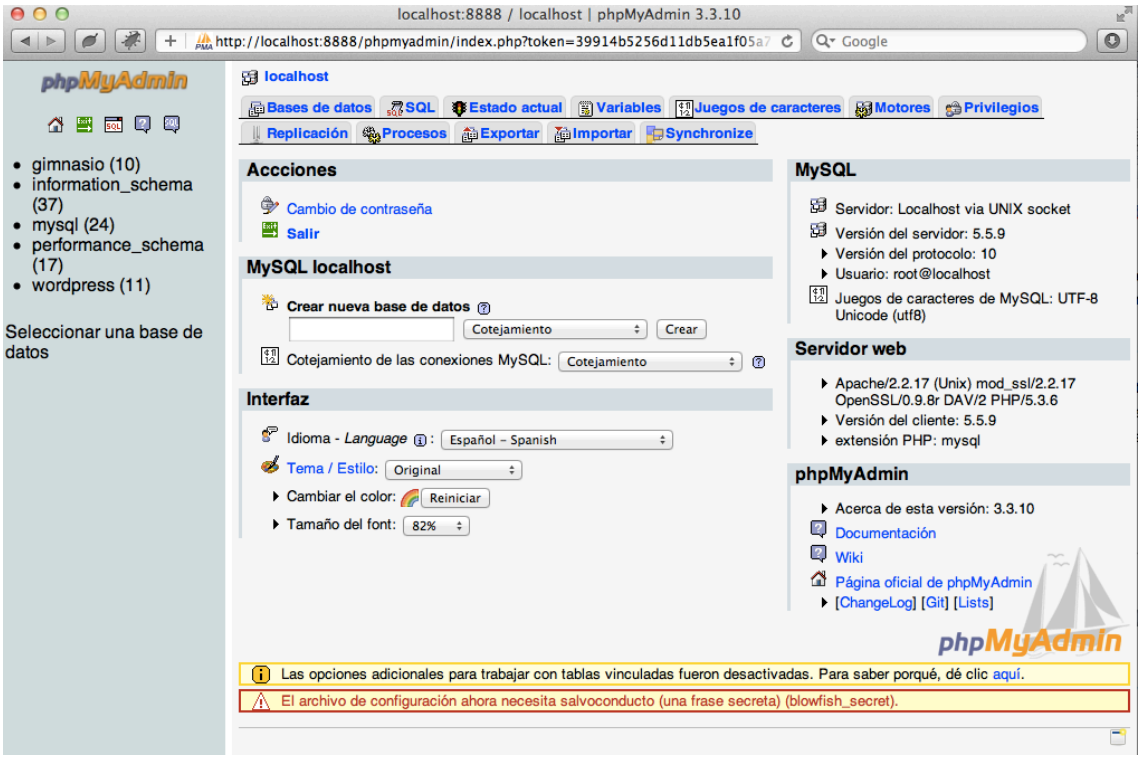
```
$ tar -xzvf phpMyAdmin_x.x.x-all-languages.tar.gz
```

Este proceso puede tardar unos minutos, dependiendo de la velocidad de proceso de nuestro equipo. Una vez descomprimido el fichero, procederemos a eliminarlo, de forma que quede únicamente el directorio que se ha descomprimido.

Una vez descomprimido el directorio, es aconsejable renombrar el nombre del directorio por uno más sencillo. Además si se usa en un servidor de producción deberemos preservar la seguridad de dicho directorio, para que sea accesible únicamente por el personal de administración.

5.3.2.1- Utilización

Ahora es el momento de comenzar a utilizar phpMyAdmin, para ello deberemos iniciar un navegador, y cargar la siguiente dirección: http://localhost/phpmyadmin (en mi caso particular mi servidor web se encuentra en el puerto 8888) :

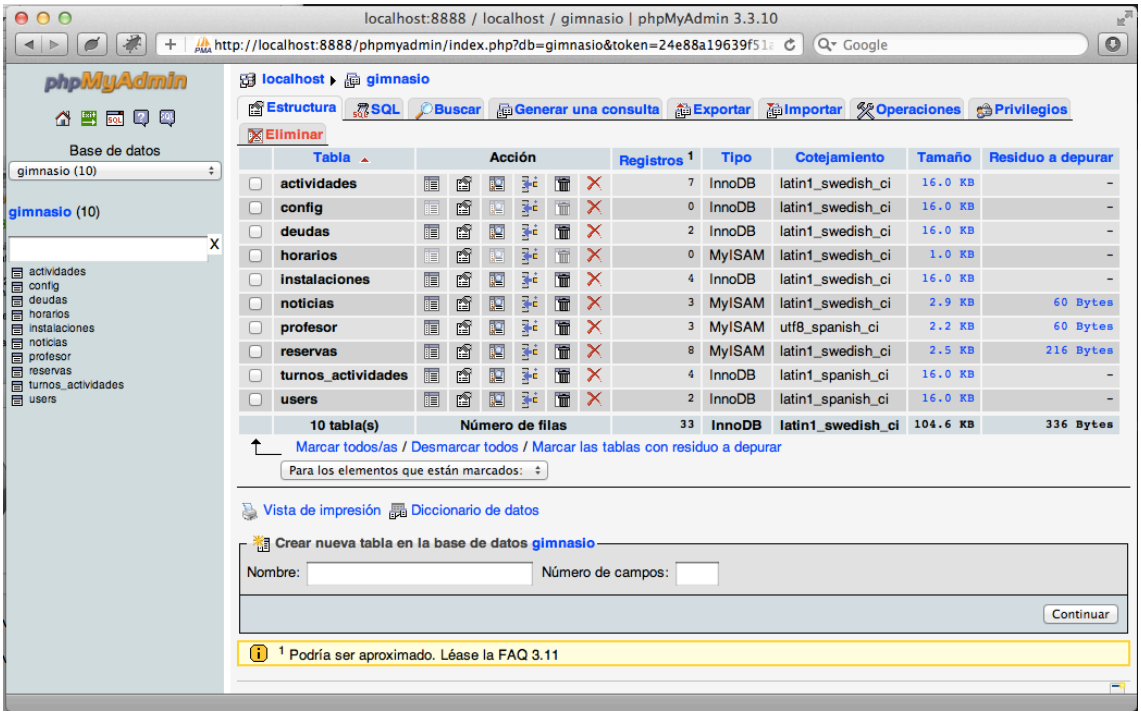


En la parte izquierda de la pantalla aparecerán las distintas tablas de las que dispone el servidor. En este caso podemos apreciar “gimnasio” que es la que usaremos para almacenar los datos de nuestra aplicación. A excepción de “wordpress” que pertenece a una instalación de dicho portal, el resto de tablas pertenece a la propia instalación de mysql, por lo que no deberemos modificar nada si no somos conscientes de lo que hacemos.

Para crear una nueva base de datos deberemos crearla en la parte derecha. Si disponemos de un fichero con sentencias SQL, deberemos hacer click en “Importar” y subir dicho fichero. Para comenzar la configuración de nuestra base de datos, el primer paso es seleccionar la misma en el apartado izquierdo de la pantalla.

En este caso particular, esta base de datos pertenece al servidor de producción de nuestra aplicación web, y ya contiene las tablas convenientemente creadas, con sus respectivos datos almacenados en su interior.

En primer lugar podemos ver que el apartado izquierdo ha cambiado, y en vez de mostrar las diferentes bases de datos, ahora muestran las distintas tablas de la base de datos sobre la que estamos operando. En el menú superior podemos ver una serie de pestañas que sirven para realizar las distintas acciones sobre la base de datos.



En el apartado “SQL” podemos realizar distintas pruebas sobre las consultas que vamos a ejecutar en nuestra aplicación. Por un lado nos muestra si la consulta introducida tiene algún error de sintaxis. En caso de que la sentencia sea correcta, indica el número de filas afectadas, el tiempo que se empleó en realizar la consulta, y por último el resultado de la consulta.

El apartado “Buscar” no tiene mayor explicación que la búsqueda de contenido dentro de la base de datos. Es posible elegir caracteres comodín para personalizar más aún la búsqueda, decidir si se deben buscar al menos una palabra introducida, todas ellas, la frase exacta, o si por el contrario es una expresión regular. Deberemos introducir la tabla en la que buscar. El campo de la tabla es opcional.

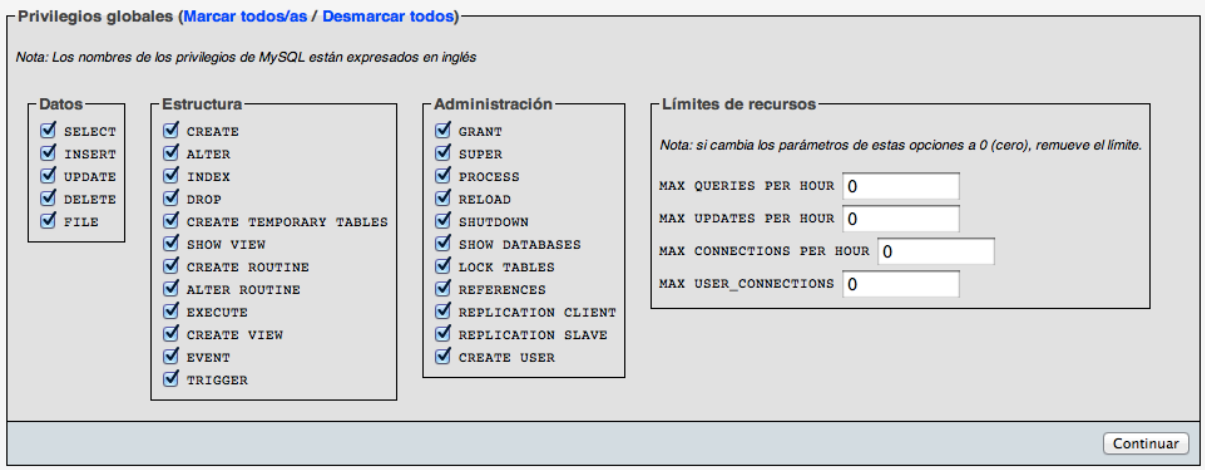
En el apartado “Generar una consulta” podremos realizar una consulta sin necesidad de tener conocimientos de SQL. Para ello hay disponibles una serie de desplegables y cuadros de texto, en los cuales podremos seleccionar los campos a consultar y establecer criterios de búsqueda. Una vez finalizado, nos mostrará el código SQL que correspondería a la sentencia seleccionada.

En el apartado “Exportar” se nos ofrecerá la posibilidad de seleccionar las distintas tablas que deseamos exportar. Los formatos a los que permite exportar son: SQL, LaTeX, PDF, Excel 2000, Word 2000 y CSV. Por un lado podremos seleccionar las partes de la estructura que exportaremos, y por otro los datos. Una vez seleccionado todo, únicamente nos quedará elegir el nombre del fichero y su codificación.

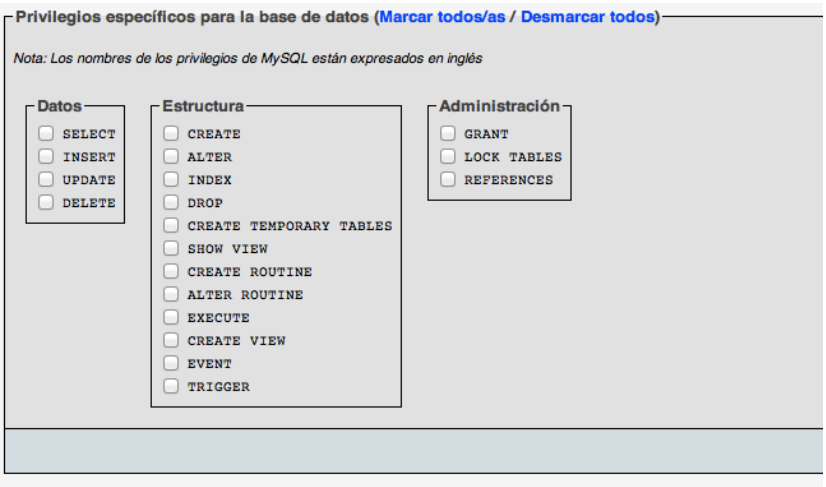
El apartado “Importar” nos permitirá realizar la operación contraria, esto es seleccionar un fichero procedente de una importación. Los formato que acepta son los siguientes: CSV, Open Document Spreadsheet, Excel 97-2007, XML.

En el apartado “Operaciones” podremos crear una nueva tabla dentro de la base de datos, cambiarle el nombre. También podremos copiar la base de datos (únicamente la estructura, estructura y datos, o solamente datos)

Dentro del apartado “Privilegios” podremos gestionar los privilegios que disponen los distintos usuarios. En la primera vista podemos ver todos los usuarios existentes del servidor, junto con los privilegios que éstos tienen. Por un lado podremos agregar nuevos usuarios, y por otro modificar los privilegios de los ya existentes. Existen dos tipos de privilegios, por un lado los globales y por otro los específicos de la base de datos. Aquí se pueden ver los distintos privilegios globales que se pueden asociar:



Y aquí se pueden ver los privilegios específicos para la base de datos seleccionada:



5.4.- Seguridad

Uno de los motivos que generan más escepticismo a la hora de la implantación o utilización de servicios online es la desconfianza que puede generar el uso de datos personales por parte de un tercero, y el desconocimiento del nivel de seguridad que dispone tal servicio.

Cuando una aplicación debe manejar datos de tanta importancia como pueden ser los datos personales, o la gestión monetaria de cada usuario, es importante tener especial cuidado con el tratamiento de todos ellos. El primer motivo es para mantener la continuidad del negocio a salvo, ya que tendremos graves dificultades para continuar con él si perdemos los datos de todos nuestros clientes. El segundo motivo es puramente legal, ya que en España la Ley Orgánica de Protección de Datos dicta normas muy claras y específicas sobre el manejo de este tipo de datos.

5.4.1.- Conceptos de seguridad web

Vamos a tener en cuenta la forma de elevar el nivel de seguridad de nuestra aplicación. Para ello primero vamos a introducir brevemente la seguridad en las tecnologías de la información:

- **Confidencialidad:** Garantia de que la información va a ser revelada únicamente por los usuarios adecuados.
- **Integridad:** Asegura que la información en tránsito es completamente segura y no puede ser modificada por gente no autorizada.
- **Disponibilidad:** Garantiza que la información debe ser accesible en el momento en que ésta sea solicitada.

5.4.1.1.- Aseguramiento del servidor web

Este apartado es el que más se ha tratado en el tema de la seguridad de sistemas a lo largo de los años, poniendo especial hincapié en servidores Unix / Windows.

La finalidad es conseguir que el servidor web cumpla los requisitos necesarios para hacer correctamente su cometido, pero no sea capaz de realizar nada más. Con esto pretenderemos que nuestro servidor sea capaz de garantizar que ningún usuario pueda tomar el control del mismo.Un ejemplo sería autorizar a usuarios anónimos a visualizar el contenido de la web, pero no le sea posible realizar actividades del estilo de cambiar la configuración de red del servidor, o instalar nuevas aplicaciones en el mismo.

Uno de las mejores prácticas es aislar cada servicio que se preste a los clientes en un equipo diferente, así conseguiremos que un error o bug en un servidor afecte al resto del ecosistema de servicios en la menor medida posible.

En caso de no disponer de recursos suficientes que nos permitan implantar una solución como la comentada anteriormente, una solución es usar un sistema de virtualización que nos permita aislar cada servicio en una máquina virtual diferente.

Otra buena técnica para asegurar el correcto funcionamiento de los servidores y la información contenida en ellos es restringir el acceso a los administradores. Aquí cabría mencionar tanto los accesos que se realizan de forma física como remota. En el primer caso tendría que ver con limitar el acceso al lugar donde está ubicado el servidor y registrar todos y cada uno de los accesos, incluyendo fechas, horas, tareas realizadas, etc. En el segundo caso podría limitarse el número de usuarios que inician sesiones interactivas en el servidor, y siempre usar métodos seguros como SSH o túneles VPN.

5.4.1.2.- Aseguramiento de la información en tránsito

Este es el apartado que parece más susceptible a priori, ya que depende tanto de la infraestructura del servidor, como la del usuario, como de todos aquellos proveedores de servicios por los cuales pasen todos los datos.

Una parte importante son las técnicas criptográficas que sean utilizadas en la conexión que se mantiene entre el cliente y el servidor. De forma estándar se utiliza en protocolo HTTP que funciona por el puerto 80, pero este protocolo no es demasiado seguro cuando se trata de enviar datos personales, contraseñas, cuentas bancarias… Para solucionar este tema, se puede utilizar el protocolo HTTPS que funciona por el puerto 443 que sirve precisamente para asegurar los datos que viajan entre el servidor y el cliente, utilizando para ello protocolos SSL o TLS.

Uno de los ataques más peligrosos son los ataques DoS, ya que permite la inutilización de nuestro servidor sin que podamos poner remedio alguno. El motivo más común de este tipo de ataque son una sobrecarga de peticiones al servidor, realizado desde Internet. La forma de solucionarlo es muy complicada, y suele pasar por utilizar servidores redundantes y sistemas de respaldos.

5.4.1.3.-Aseguramiento del equipo del cliente

Los primeros sitios web aparecieron como simples contenedores de información, cuya única interacción con el cliente era la solicitud de nuevas páginas mediante el uso de los distintos enlaces para solicitar nuevas páginas.

Con el tiempo fueron apareciendo nuevas tecnologías que permitían suplir esta carencia. Entre estas tecnologías se encuentran Java, ActiveX, Flash, plugins, etc.

Las páginas web por sí solas son estáticas y permiten poca interacción entre el cliente y el servidor, pero a menudo se utilizan tecnologías que permiten precisamente aumentar esta interacción. Entre ellas se encuentran Java, ActiveX, Flash, plugins, etc.

A menudo estas tecnologías tienen problemas de seguridad que afectan directamente a los usuarios, por lo que el camino que se está siguiendo es disminuir su utilización como ya ha pasado con Java y ActiveX cuya utilización es prácticamente simbólica y en sitios web antiguos.

Como desarrolladores web, podemos reducir el uso de estas tecnologías, procurando seguir la mayor cantidad de estándares posibles como pueden ser JavaScript, uso del DOM, Ajax, y algunas otras tecnologías más.

Por otro lado podemos extraer mediante JavaScript algunas características del agente del usuario como pueden ser sistema operativo, navegador utilizado, plugins instalados… De esta forma podremos dinamizar en cierta forma el contenido que sirvamos al cliente, para de esta forma evitar posibles fallos de seguridad debidos al uso de versiones con fallos conocidos.

5.4.2.- Usuario y grupo para Apache

Apache dispone de su propia jerarquía de directorios, que comienzan en la raíz que sería ServerRoot, y sobre la cual cuelgan distintos directorios como se han visto anteriormente como son “includes”, “bin”, “htdocs”, “cgi-bin”, etc.

Es importante asignar permisos adecuados a este árbol de directorios para asegurarnos que ente cualquier problema el visitante no disponga de más permisos de los que debería tener para intentar evitarnos los máximos problemas disponibles.

Los dos puntos a tener en cuenta son el usuario que ejecuta el servidor Apache, que será el propietario del proceso, y por lo tanto heredará todos sus permisos. Y por otra parte, los permisos disponibles en el árbol de directorios del servidor (y también fuera de este)

Hay que tener en cuenta también que si disponemos de una persona encargada de editar el contenido, deberá disponer de permisos al directorio de publicación (probablemente llamado “htdocs” o “www”). Los desarrolladores de scripts CGI deberán disponer también de permisos al directorio cgi-bin. Puede ser por lo tanto buena disponer de los distintos directorios en ubicaciones diferentes, evitando que todos estén dispuestos en forma de árbol.

5.4.3.- Permiso de los distintos directorios

Nuestra instalación de Apache es uno de los puntos que deberemos tener en cuenta a la hora de establecer un entorno seguro ante posibles ataques o intrusiones.

La recomendación general es utilizar un usuario y un grupo específico para ser utilizado por Apache. Este usuario/grupo deberá poseer el UID/GID del demonio que ejecuta Apache. Además del ejecutable también es importante tener en cuenta los permisos de los distintos directorios que utiliza Apache, y que han sido vistos en apartados anteriores.

El directorio donde se encuentra instalado Apache debería ser accesible solamente al usuario root, ya que el debe ser el único que debe configurar o controlar el proceso Apache.

Para el directorio de publicación (conocido también como "htdocs") deberemos establecer un usuario o grupo que dispongan de permisos suficientes para modificar su contenido. Este usuario/usuarios estarán destinados a su uso por parte de los webmasters de nuestro sitio. A este directorio también deberá tener permisos de lectura el propio usuario de Apache para que sea capaz de servir el contenido de las peticiones que llegan al servidor.

De forma similar al caso anterior, el directorio donde se encuentran los scripts CGI deberán ser accesibles únicamente a los usuarios encargados de desarrollar los mismos, además del propio usuario de Apache (importante que tenga permisos de ejecución).

En cuanto al directorio donde se encuentran los logs, deberá ser accesible únicamente por el usuario root (con permisos totales 700).

5.4.4.- Replicación y Copias de Seguridad

Mantener la integridad de nuestro sitio web, es una tarea que requiere determinadas tareas de mantenimiento. La opción más segura y efectiva es mantener nuestro sistema web en un clúster de alta disponibilidad que permita balancear la carga de nuestros servidores de forma dinámica. Esta opción tiene el inconveniente de necesitar grandes cantidades de dinero para poder llevarla a cabo.

Otra opción más económica es disponer de un servidor que sea capaz de ejecutar nuestra aplicación en caso de detectar una alerta en un tiempo mínimo, sustituyendo de esta forma a nuestro servidor principal. Esta opción requiere la activación manual de este servidor replicado. Por otra parte es necesario que las bases de datos estén completamente sincronizadas para que no estemos utilizando una versión de esta antigua que puede disponer de datos inconsistentes.

Las copias de seguridad son otro tema imprescindible para mantener la integridad de nuestro sitio ante posibles errores imprevisibles. También hay que tener en cuenta que según la Ley Orgánica de Protección de Datos, este apartado dispone de una específica legislación que deberemos tener en cuenta a la hora de planificar nuestro sistema de copias de seguridad. Los apartados que deberemos respaldar en nuestro caso serán, por un lado el directorio de publicación de Apache donde se incluyen tanto los documentos html, php, imágenes, etc. Por otro lado deberemos respaldar la base de datos completa, ya que realmente es allí donde se encuentran almacenados los datos que utiliza nuestra aplicación.

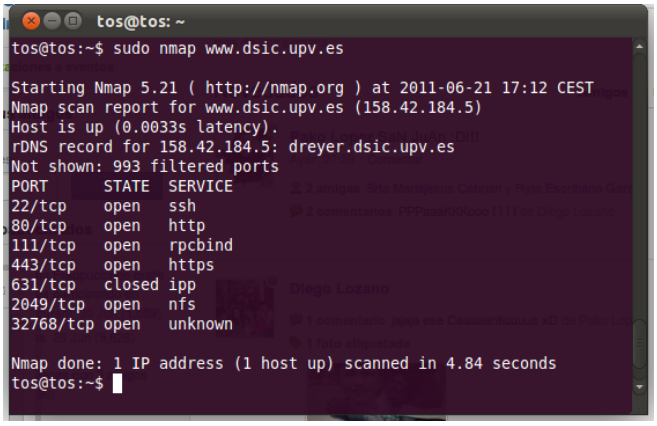
5.4.5.- Escaneo de información

El objetivo de esta operación es realizar un escaneo de puertos de la máquina a analizar para averiguar qué tipo de servicios está ofreciendo.

Nmap (Network Mapper) es una de las herramientas más importante en cuanto a la exploración de redes y auditoría de seguridad. Podemos usarlo libremente gracias a su licencia GPL, estando disponible para prácticamente cualquier plataforma en la dirección (http://nmap.org).

Nuestro objetivo de determinar qué puertos y servicios están activos en las máquinas de una red, lo llevaremos a cabo con esta herramienta, con la que podemos hacer barridos automatizados. A pesar de que existen herramientas gráficas que hacen de front-end para esta aplicación, aquí se va a mostrar únicamente la herramienta en modo consola. Si está interesado en las herramientas gráficas pueden interesarle NmpaFE o Zenmap.

¡ ATENCIÓN DEBE USTED SER ADMINISTRADOR PARA USAR ESTA HERRAMIENTA!



Aquí podemos ver un ejemplo de un escaneo de puertos al servidor web del departamento DSIC de la UPV, se puede apreciar que la orden tiene la forma "`nmap maquina_destino`"

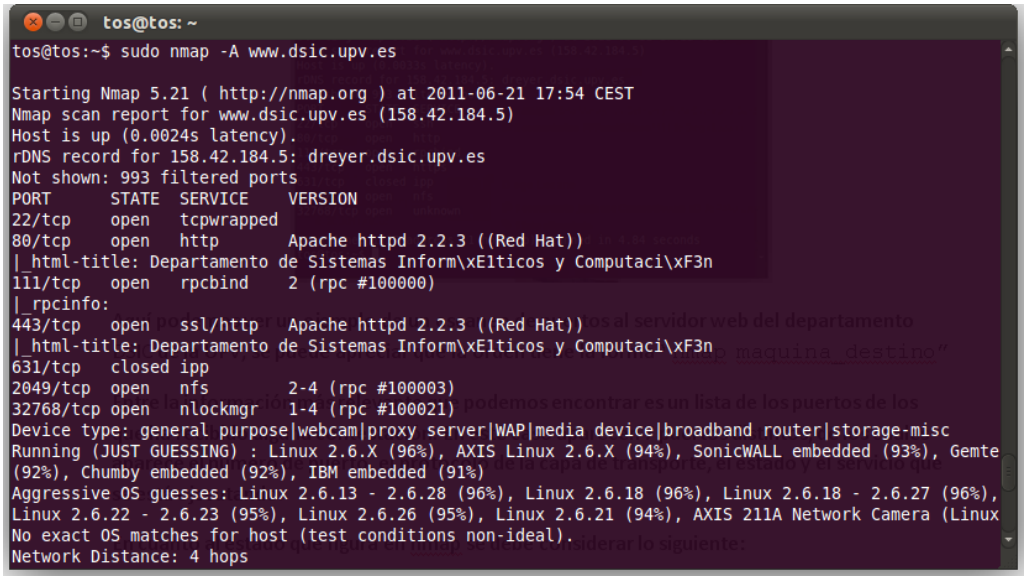
Entre la información más relevante que podemos encontrar es un lista de los puertos de los que ha recibido alguna contestación. En este caso aparecen 7 puertos distintos, de los cuales aparece el número de puerto, el protocolo de la capa de transporte, el estado y el servicio que se está ejecutando.

En cuanto al estado que figura en nmap se debe considerar lo siguiente:

- Open: Se está ofreciendo dicho servicio en el puerto que se menciona
- Close: En el momento del escaneo está cerrado, pero puede ser abierto en un futuro
- Filtered: El paquete no ha llegado al destino, seguramente por algún firewall o router

- No filtered: No está filtrado, pero Nmap no puede saber si está abierto o cerrado
- Open|filtered: No responde a la petición, por lo que no se puede determinar su estado
- Close|filtered: Es posible que el puerto esté cerrado o filtrado

El ejemplo visto anteriormente es el más básico de la potente herramienta Nmap. En este caso vamos a usar la opción "`-A`" para el siguiente ejemplo, que ofrece un ejemplo más detallado.

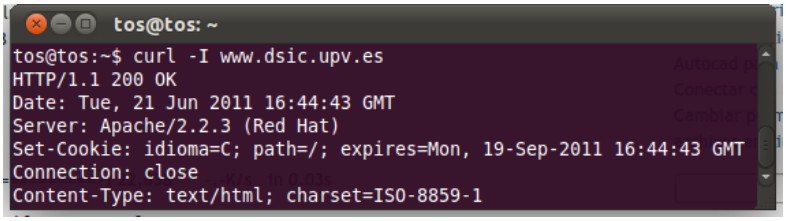


En este caso podemos apreciar que Nmap ha determinado las versiones de los servidores que ofrecen los distintos servidores. En este caso el dato importante es el servidor http y https, que corre Apache 2.2.3 modificado para RedHat.

Luego vemos un dato también muy importante que es el tipo de dispositivo y el sistema operativo que puede ejecutar el servidor destino. En este caso vemos que dice que las condiciones del test no son las ideales, y únicamente puede determinar que se trata de Linux entre la versión 2.6.13 y la 2.6.28

Otra forma de conseguir información sobre un servidor web es solicitar un recurso inexistente, de forma que si el administrador no ha implementado una página de error específica, puede devolvernos un error como el que sigue, en el que aparece la versión del servidor:

Otro lugar dónde suele aparecer más información de nuestro servidor web es en las cabeceras http que el servidor envía al cliente en cada petición. Con el comando "curl –I" podemos recuperar todas esas cabeceras para poderlas analizar:



Ahora vamos a ver cómo podemos reducir estos riesgos, ya que estamos ofreciendo demasiada información que un posible atacante puede usar en nuestra cuenta.

En primer lugar intentaremos reducir el número de servicios prestados por el servidor, como ya se ha comentado anteriormente hay que separar en la medida de lo posible los servicios en diferentes servidores.
Otra opción basada en ofuscar la información se trataría de interponer firewalls entre la red de destino y el servidor. El objetivo de esta configuración es que Nmap detecte que el puerto está filtrado, y tanto los paquetes que envíe como los que reciba pueden ser alterados.

En la siguiente fase vamos a ver las opciones que tenemos para silenciar los errores cuando intentamos acceder a recursos inexistentes en nuestro servidor.

En primer lugar vamos a tratar el error que aparece cuando se intenta acceder a un recurso inexistente. Lo que se va a hacer es sustituir la página de error del servidor por otra que crearemos nosotros.

Aunque el ejemplo que se da aquí es únicamente para ocultar el servidor que estamos utilizando, si hacemos hincapié en la usabilidad para el usuario, ésta página debería tener más funciones como mostrar un mapa del sitio, un campo de buscar o alguna sugerencia para guiar al usuario a encontrar lo que realmente buscaba cuando encontró el error.

En primer lugar tendremos que crear dicha página y copiarla en nuestro directorio de publicación del servidor web (en mi caso /var/www). En este caso se ha llamado “404.html” y tiene el siguiente contenido:

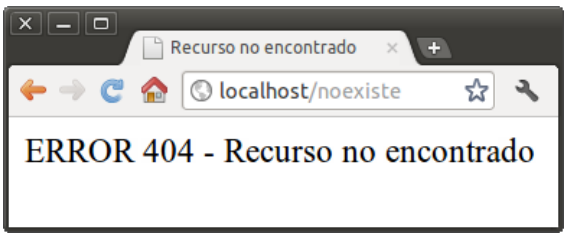
```
<html><head><title>Recurso no encontrado</title></head>
<body>ERROR 404 - Recurso no encontrado</body></html>
```

Ahora hay que configurar el servidor web Apache para indicarle que cuando encuentre un error 404 redirija a esta página que acabamos de mostrar. Para ello deberemos modificar el fichero de configuración (en mi caso /etc/apache2/apache2.conf) , añadiéndole la siguiente línea:

```
ErrorDocument 404 /404.html
```

Hay que recordar que después de realizar cualquier modificación en el fichero de configuración, hay que reiniciar Apache para que estos cambios surjan efecto.

Para comprobar el resultado de la página de error, simplemente deberemos solicitar un recurso inexistente (en este caso la página “noexiste”):

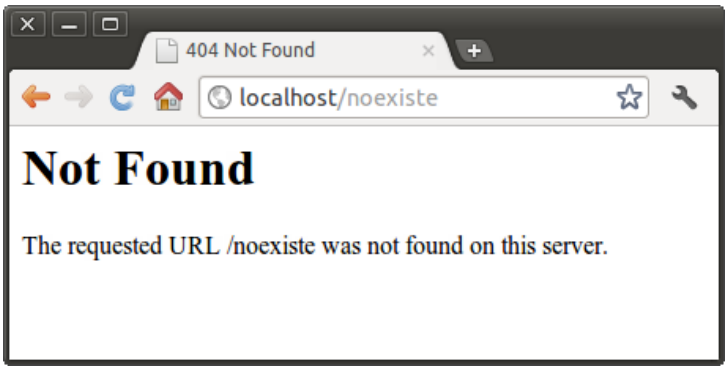


Podemos ver perfectamente como el nombre del servidor y la versión ha desaparecido, siendo reemplazado por la página que hemos creado.

Otra opción alternativa a lo anterior sería usar una directiva de Apache que permite eliminar la firma que aparece en la parte inferior del mensaje de error. Dicha directiva es ServerSignature. Para ello debería establecerse a Off:

```
ServerSignature Off
```

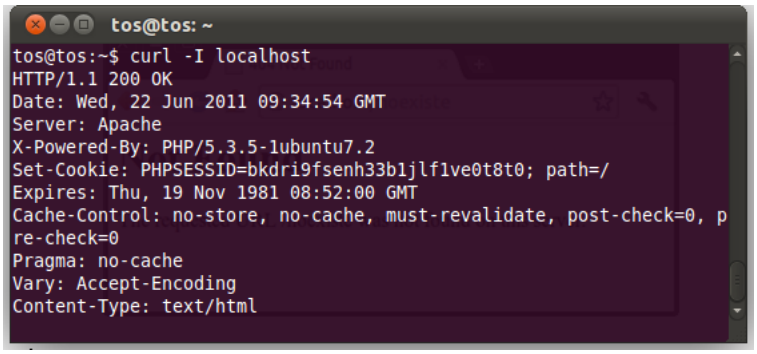
El resultado que podemos ver después de la inclusión de esta directiva es el siguiente:



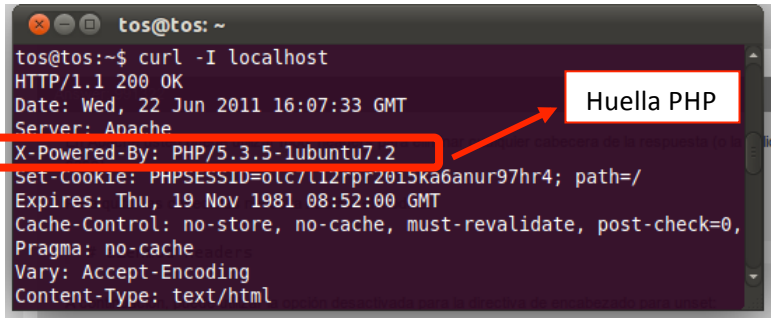
Ahora vamos a eliminar la máxima información posible que aparece en las cabeceras HTTP que el servidor envía al cliente. Para ello vamos a fijarnos en el ejemplo de la siguiente imagen. En primer lugar en el campo "Server" veremos que aparece algo del estilo "Apache/2.2.6", lo que vamos a intentar es eliminar el número de versión de este campo.

Apache dispone de una directiva llamada ServerTokens que permite modificar la información que aparece en el campo Server. A continuación se muestra las diferentes posibilidades de configuración de esta directiva:

ServerTokens Prod → Mostraría Server: Apache
ServerTokens Min → Mostraría Server: Apache/2.2
ServerTokens OS → Mostraría Server: Apache/2.2 (Unix)
ServerTokens Full → Mostraría Server: Apache/2.2 (Unix) PHP/5.0



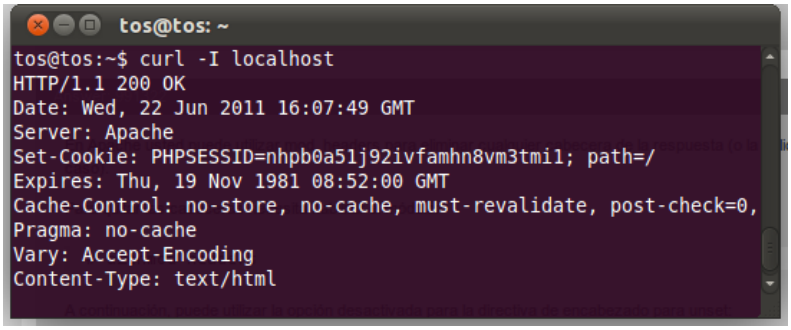
Para seguir analizando las cabeceras HTTP que nos devuelve Apache, vamos a ver un campo llamado "X-Powered-By". En la siguiente imagen se puede ver el ejemplo, en el que se muestra claramente y con todo lujo de detalles la versión de PHP utilizada en el servidor (en este caso PHP/5.3.5-1ubuntu7.2)



Esta parte no corresponde estrictamente a Apache, si no que es una configuración propia de PHP. En primer lugar deberíamos localizar el fichero de configuración de PHP, en mi caso se encontraba en /etc/php5/apache2/php.ini). La directiva a modificar sería `expose_php`, y quedaría de la siguiente forma:

```
expose_php = Off
```

Después de esta modificación habría que reiniciar Apache para hacerse efectivos los cambios (`sudo service apache2 restart`). Una vez reiniciado, podremos realizar otra petición de cabeceras para ver la diferencia:



Hemos dejado para el final la parte más complicada, que corresponde a evitar la identificación del sistema operativo del servidor por parte de Nmap (recordemos `nmap -A`).

6.-Aplicación Web

Una vez que tenemos claro todo el tema del servidor, significará que tenemos preparado el entorno de ejecución para que nuestra aplicación pueda ser ejecutada correctamente. En este apartado empezaremos a definir toda la aplicación web que vamos a desarrollar, teniendo en cuenta todos los aspectos que hemos comentado anteriormente.

Hay que recalcar que la aplicación web no es únicamente la presentación que verá el usuario final (que está compuesta por la parte estática y la dinámica), si no también por toda la lógica que va detrás para crear todos los datos dinámicos y para almacenar los datos.

Empezaremos definiendo los requisitos de la aplicación, esta parte vendrá definida por los clientes a los que va destinada la aplicación. Como en el caso de este proyecto, no hay un cliente fijo y determinado, se hará una previsión de los requisitos informándose a través de personas relacionadas con el mundo del gimnasio.

A partir de todas estas informaciones definiremos de formalmente los requisitos, haciendo una especial diferenciación entre los requisitos de información y los funcionales. Los primeros se refieren a toda la información que se muestran a los distintos usuarios dependiendo de sus roles, y los segundos a los procesos que pueden llevar a cabo, también dependiendo de su rol dentro del sistema.

Una vez definidos todos los requisitos, pasaremos a analizar todos ellos y elaborar los distintos mapas conceptuales y esquemas sobre todos los procesos de nuestra aplicación. Aquí empezaremos a crear los primeros bocetos de la interfaz de usuario, y los diagramas de clase.

A partir de esta fase se analizarán las distintas capas de nuestra aplicación, desde la interfaz de usuario, hasta la parte del almacenamiento de los datos. Se hará hincapié en cada una de ellas mostrando los distintos procesos que se llevan a cabo en cada uno de ellos. También se mostrarán los distintos patrones de diseño utilizados en distintas webs que nos ayudarán a desarrollar un diseño coherente y funcional.

El último apartado será la implementación del sistema, decidiendo las tecnologías más convenientes para el desarrollo de nuestra aplicación. Entre la distintas tecnologías que debemos elegir estarán las de la interfaz, la aplicación y los contenidos.

6.1.- Usabilidad

Partiendo de la base de que los usuarios pueden ser o no expertos en la navegación por Internet, tenemos que pensar en facilitarle el uso de nuestra aplicación.

De esta forma intentaremos aumentar la intuición, permitiendo al usuario una mejor experiencia en nuestra aplicación y transmitir una mayor confianza al usuario.

En primer lugar tenemos que tener en cuenta que el objetivo de nuestro sitio es llamar la atención sobre nuestros servicios, que se pueda encontrar información sobre ellos y que sea sencillo gestionar una reserva. Partiendo de ese punto debemos tener una página de inicio clara que permita localizar al usuario de forma clara dónde se encuentran los distintos apartados de nuestra web.

Existe una rama de investigación centrada en este tema, que intenta averiguar reglas de la conducta humana a la hora de navegar por Internet. La técnica más importante para valorar estas conductas se llama Eyetracking, y se basa en hacer un seguimiento de la posición del ojo a través de la pantalla.

En este caso se hace un muestreo poblacional sobre un conjunto dispar de personas, entre las que se incluyen un amplio rango de edades, sexos, estudios, trabajos, etc. Se somete a este conjunto a la navegación por una serie de páginas web con distintas configuraciones y diseños para comprobar donde se centra la mirada de los navegantes.

Gracias a las conclusiones de este tipo de estudio, se pueden sacar importantes conclusiones a la hora de ponernos a diseñar nuestra página web.

Es importante que distingamos tres tipos de usuarios en nuestra aplicación:

1. Aquellos que entran por primera vez y no tienen muy claro lo que se van a encontrar. Nuestro objetivo para estos usuarios es captar su atención y evitar que se convierta en un rebote abandonando nuestra web. Para ellos deberemos tener preparado datos abundantes, con importante carga visual que les ayude a crear una mejor imagen sobre las instalaciones y los servicios ofertados.

2. Aquellos que vienen buscando algún tipo de información, novedad u oferta en nuestra web, pero no es seguro que contrate nuestros servicios. Para ello deberemos mostrar claramente todos aquellos apartados a los que puede acceder, intentando captar su atención sobre instalaciones y servicios que le puedan ser de interés.

3. Aquellos que vienen directos a hacer una reserva. Este tipo de usuario tiene claro que es lo que busca y necesita perder el menor tiempo posible. Para ello deberemos tener un camino sencillo para que pueda encontrar lo que busca, evitándole clicks innecesarios.

Como el objetivo de este Proyecto Final de Carrera no es hacer un amplio estudio sobre las técnicas de usabilidad, me limitaré a dar una serie de pautas sobre estudios ya realizados.

Los autores *Jakob Nielsen* y *Kara Pernice* destacan aspectos como usar menús visibles mediante imágenes o diferenciados con un color distinto, usar un botón que permita volver a la página inicial, un logotipo en la esquina superior izquierda de todas las páginas, un campo

de búsqueda en la esquina superior derecha, un icono de carro de la compra en la esquina superior derecha, un botón que permita conectar / desconectar nuestra sesión.

La gente por naturaleza ya espera que la información más relevante esté en un determinado lugar, pero con los aspectos antes mencionados lo que se pretende es crear focos de mayor prioridad donde el usuario fije su atención.

Como norma general la zona que se encuentra por debajo de la barra de navegación fijará más la atención que la parte inferior de la página, por lo que deberemos tener en cuenta que el contenido más importante deberá ir en esa zona.

Los aspectos que diferencien los distintos tipos de letra (como pueden ser mayores tamaños, negritas, colores, ...) indicarán que ese texto tiene una mayor importancia. En este aspecto deberemos tener especial cuidado a la hora de evitar llamar en exceso la atención ya que puede ser que consigamos ahuyentar a nuestros visitantes.

También podemos usar aspectos visuales para tratar de dar importancia a un elemento de nuestra web, como incluirlo en algún cuadro sombreado, o coloreado. Es importante no abusar de esta técnica ya que la clave es que las partes importantes se diferencien del resto de la página para tratar de llamar la atención.

De esta forma a pesar de que existan zonas que sean focos de prioridad a priori, podemos hacer determinar cuáles de estos focos serán más importantes para el usuario intentando reducir el tiempo que usará el usuario para “aprender” a usar nuestra web.

A la hora de diseñar nuestro sitio tenemos que tener en cuenta que los usuarios tienen tendencia a buscar primero por la zona derecha y en la superior, por lo que las opciones de navegación más importantes deben estar en ese lugar si queremos facilitar esta labor. Si por el contrario nuestro sitio está basado en la publicidad, sería interesante que estos lugares estuvieran ocupados por publicidad, y obligar al usuario a hacer scroll para encontrar los enlaces de navegabilidad de nuestro sitio.

Muy importante es también diferenciar correctamente aquellos elementos de nuestro sitio que ofrecen algún tipo de interacción, de aquellos que simplemente son meramente decorativos o informativos. De esta manera facilitamos la adaptación del usuario a nuestro web, y conseguiremos por lo tanto que consiga más productividad en su visita.

6.1.1.- El espacio en pantalla

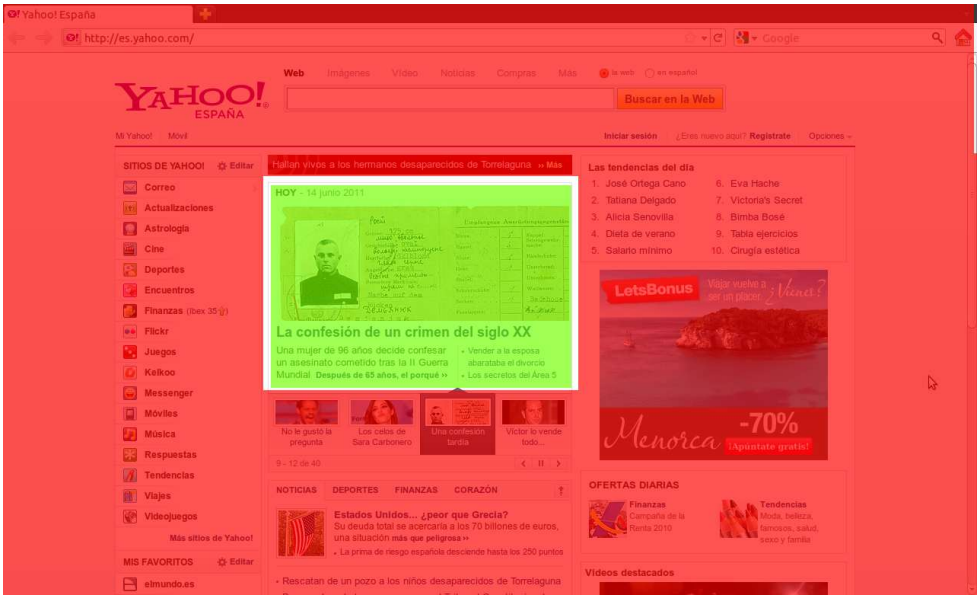
En pantalla hay gran cantidad de elementos imprescindibles que aportan cierto valor a nuestro sitio, pero sin duda, el elemento más importante de la pantalla debe ser la información, por lo que el resto deberemos intentar evitarlo en la medida de lo posible. Intentando que visualmente también sea el elemento más importante de la pantalla.

Cierto es que estos elementos que hemos considerado como de segunda, son imprescindibles para la propia usabilidad de la página, ya que ayudan a encontrar la propia información entre

todas las páginas de la web. Por ejemplo, sin un logotipo nuestra página perdería totalmente la identidad y confundiría al usuario sobre dónde está exactamente, sin enlaces no podríamos navegar a otros contenidos relacionados o a otros espacios de la web, sin publicidad probablemente no podríamos rentabilidad nuestro sitio, o sin espacios en blanco no podríamos estructurar de forma clara la información.

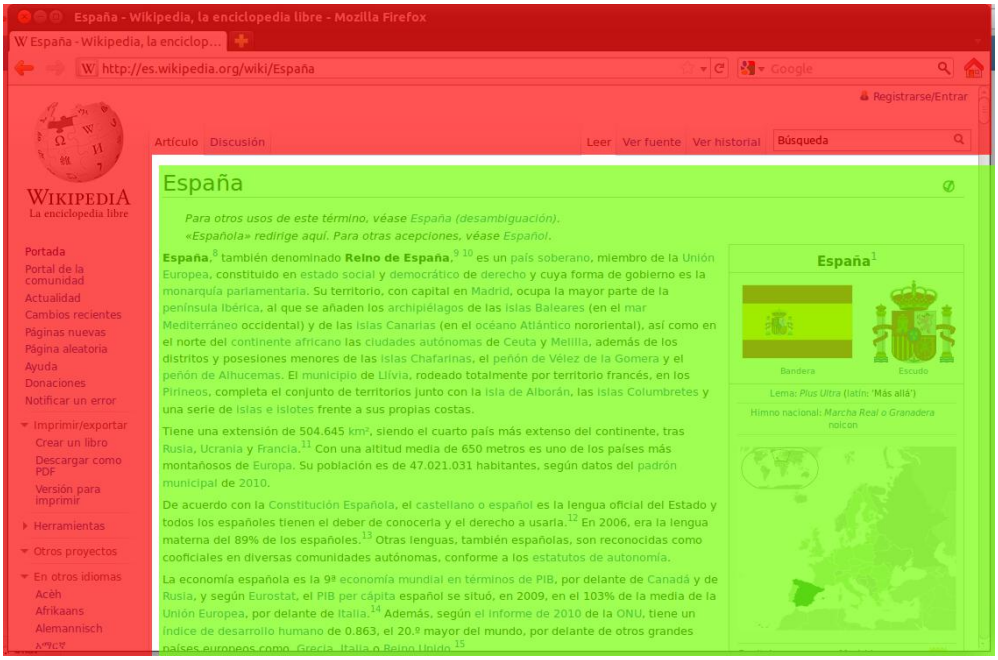
Como muchas otras cosas, se trata de buscar un término medio que permita aprovechar las ventajas, sin caer en los problemas que puede traernos los excesos. Por un lado, si intentamos que todo sea información los usuarios se sentirán muy incómodos en nuestro sitio, ya que le será muy difícil encontrar la información que necesita. Por otro lado, si procuramos que el contenido sea secundario, estaremos sentenciando las futuras visitas de los usuarios, ya que si no encuentran contenido útil no tendrá sentido que vuelvan a nuestra página.

A continuación vamos a ver 2 páginas opuestas. En primer lugar se trata de unas capturas de pantalla de Yahoo! España (Junio de 2011) y Wikipedia (Junio de 2011). Se pretende representar en color verde la proporción de espacio que corresponde a la información propiamente dicha, y en color rojo el resto de elementos.



Como podemos ver en este ejemplo, el espacio desperdiciado es muy grande, tanto que para el usuario resulta casi ridículo, ya que el espacio que tiene para poder visualizar la información que viene buscando es minúsculo. El tamaño que ocupa la información es aproximadamente un 10% del total de espacio en pantalla.

Ahora vamos a ver el siguiente ejemplo:



En esta captura podemos decir que el espacio de la pantalla si que está realmente utilizado por información de interés para el usuario, ya que es lo que venía buscando. Aquí la proporción de color verde es muy superior a la roja, y podríamos decir que la experiencia para el usuario ha sido más gratificante que en caso anterior.

6.1.2.- Zonas visuales

En extensión al apartado anterior, otro punto realmente importante y que cabe mencionar son las distintas zonas en las que se divide el espacio en pantalla. Con el establecimiento de dichas zonas visuales que compondrán la visualización de nuestro sitio se pretenderá que la estructura de dicho sitio sea lo más cohesionada posible.

Es importante agrupar visualmente todo aquello que tenga una clara relación lógica, para que dicha relación sea también visual. De este modo los contenidos tienen entre sí una relación lógica, por lo que deberemos agruparlos para que pertenezcan a un mismo bloque visual. De este modo añadiremos claridad a las acciones que puede realizar el usuario en nuestro sitio. Otros ejemplos pueden ser el menú navegacional, o enlaces relacionados unos con otros.

De esta forma pretenderemos añadir una mayor interactividad de cara al usuario, intentando que cuando el usuario quiera realizar una acción determinada, se dirija al bloque indicado. Esto mejorará el proceso de aprendizaje de nuestra aplicación, consiguiendo que de un vistazo se consiga averiguar el contenido de cada bloque. Así conseguiremos reducir el tiempo que invierte el usuario en averiguar el lugar donde se encuentra aquello que está buscando.

La siguiente imagen ilustra un ejemplo de una página perteneciente al departamento DSIC de la UPV (http://www.dsic.upv.es), y mostrar cuáles son las zonas visuales de las que dispone, y cómo están delimitadas:

Esta imagen pretende ser solamente un ejemplo orientativo, ya que no pretende ser representativo dado que dependiendo del sitio web dichas zonas pueden variar, tanto por defecto como por exceso. Algunas de las zonas que pueden encontrarse además de las ya mencionadas pueden ser formularios de introducción de datos, pie de página o zona de operaciones.

La zona institucional ofrece información sobre la propia compañía, el más importante es el logotipo de la institución de forma que el usuario reconozca el lugar donde se encuentra. Además en este caso ofrece más datos sobre la propia institución (nombre, dirección, teléfono, fax, etc.)

El menú navegacional ofrece una barra con distintos enlaces en los que se distribuyen los distintos lugares a los que se puede navegar. Dicha barra es horizontal y está delimitado tanto por arriba, como por abajo, por un pequeño margen de un color más claro.

La zona de información se presenta en el centro de la pantalla, alineada a la parte derecha de la página. En este caso, la información que se representa son las últimas noticias de la institución ordenadas en orden cronológico. El margen en este caso es más grande, sirviéndose del color de fondo para delimitar el espacio.

6.1.3.- Creación de contenidos

Si hay algo que hay que tener claro en cuanto al usuario, es que va a leer lo mínimo que sea posible para recabar la información que está buscando.

Partiendo de esta premisa debemos simplificar la tarea de "ojeamiento" de nuestro sitio, para hacer más rápida la extracción de esta información, evitando hacer perder tiempo a nuestros usuarios.

Para ello hay que intentar en la medida de lo posible las palabras innecesarias. Conseguiremos reducir el nivel de ruido de la página, realzaremos el contenido que realmente sí vale la pena, y conseguiremos que el usuario vea más información de un solo vistazo sin tener que desplazar la ventana del navegador.

Otra facilidad es la de eliminar las instrucciones para realizar una acción. Tenemos que pensar que si requiere mucha explicación el usuario se puede agobiar, y probablemente la mejor opción sea hacer que la tarea sea más fácil de realizar.

Aunque esto no siempre será posible, ya que puede no quedar suficientemente claro qué se espera que haga el usuario, deberemos intentarlo siempre que podamos. Si realmente son necesarias las instrucciones, deberemos reducirlas a su mínima expresión, haciéndolas tan cortas como sea posible.

Una opción para instrucciones que no es imposible acortar sería incluir una imagen de un interrogante, en la que puedan hacer click aquellos usuarios que necesiten ayuda, o ofrecer las instrucciones paso por paso cuando el usuario pueda necesitarlas y nunca todas al mismo tiempo.

6.1.4.- Controles para la web

En una aplicación web las acciones que el usuario realiza son una parte importante para la interacción con el usuario, por ello debemos tener especial cuidado con la forma en que ofrecemos los servicios.

Hay que hacer una distinción entre la idea que se tiene como diseñador del uso que se le va a dar a nuestra aplicación, y otra muy distinta que es el uso que realmente se le va a dar por parte del usuario.

Uno de los factores que pueden afectar en cuanto a esta diferencia, es que el usuario realmente no entienda la forma en que le ofrecemos realizar diferentes acciones.

Un ejemplo muy ilustrativo serían estos 3 ejemplos de botones, que proporcionarían acceso a una misma acción.

Aquí se aprecia claramente cómo al usuario le puede ser más sencillo entender unos botones que otros con respecto al texto que coloquemos, cuando en realidad se trata de la misma acción.

Sin necesidad de rizar tanto el rizo, el usuario puede que ni siquiera sea capaz de distinguir qué elementos de la página nos ofrecen una acción o una navegabilidad y cuáles no.

En la imagen podemos ver como aparecen marcados con números varios textos en color azul. Aquí el usuario puede tener dudas sobre cuáles son realmente enlaces y cuáles no. En este ejemplo sólo son enlaces los números 4 y 5, el resto son cabeceras o resaltados.

6.1.5.- Legibilidad del contenido

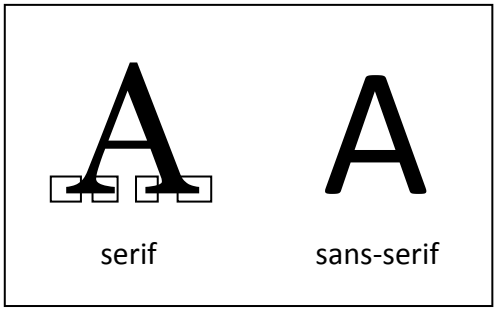
Uno de los aspectos más importantes a la hora de ofrecer un contenido a nuestros usuarios, es ofrecérselos en un formato lo más coherente posible, con el objetivo de reducir al mínimo el esfuerzo que debe hacer el usuario a la hora de leer nuestro contenido.

El tipo de letra es uno de los factores más importantes para aumentar la legibilidad del contenido de nuestro sitio web. Aquí tenemos que tener en cuenta es que nuestro website puede ser accedido a través de diversos medios, aquí vamos a tener los medios visuales como son la pantalla y el impreso. Hay que matizar que para cada caso puede ser más interesante usar una tipografía u otra, ya que unas mejoran la lectura más que otras.

En el caso de la legibilidad en la pantalla se intentará utilizar un tipo de letra cuyo tamaño sea adecuado para lectura en pantalla y sea suficientemente regular. El tipo de letra más adecuado son las llamadas “sans-serif”, estas carecen de los pequeños remates que se encuentran en las terminaciones de las letras. Un ejemplo claro son Verdana, Arial e Impact. Como se puede ver este tipo de tipografías son muy regulares, que es precisamente lo que pretendíamos. En cuanto al tamaño, Verdana sería una opción muy válida, ya que su espaciado es aceptable para la lectura en pantalla. A continuación podemos ver unos ejemplos de fuentes sans-serif:

Si por el contrario, deseamos una correcta legibilidad en papel, deberemos utilizar un tipo de letra que nos permita mantener una mejor guía visual a lo largo del texto. Un ejemplo de ello, son las fuentes “serif”, que poseen unos trazos horizontales o diagonales en las terminaciones de las letras que pretenden crear una línea horizontal por la que desplazar la vista con mayor facilidad. Entre los ejemplos más representativos son Times New Roman y Georgia. Como se puede ver las terminaciones de este tipo de letras permiten mantener mejor la guía de la línea que estamos leyendo en ese momento, evitando un esfuerzo al ojo que permite leer de forma más cómoda y durante más tiempo, ya que evita la monotonía. A continuación se pueden ver unos ejempos de fuentes “serif”:

Para acabar de profundizar en el tema de las tipografías se muestra a continuación un ejemplo de ambos tipos de tipografías. Si desea profundizar más en el estudio de las fuentes, consulte el estándar DIN 16518.



Un tema relacionado con las tipografías tiene que ver con la plataforma sobre la que está accediendo el navegador. Ahora mismo podríamos decir que las plataformas más utilizadas para acceso a nuestros sitios son Windows, MacOS X, Linux para ordenadores personales, y Android, iOS y Blackberry para dispositivos portátiles.

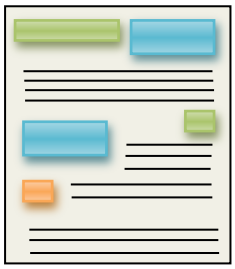
Debemos tener en cuenta que no todos estos sistemas operativos vienen por defecto con las mismas fuentes instaladas, por lo que es posible detectar incompatibilidades en este aspecto. Existe un proyecto llamado "Core fonts for the Web" que pretendía precisamente crear un conjunto de tipos de letra que fuera capaz de reproducirse de forma estándar, y fuera capaz de reproducirse en todo tipo de plataformas. Sería interesante usar estas tipografías, bien como opción principal, o bien como secundaria en caso de que la primera no se encuentre instalada en el sistema. Dichas fuentes son: Andale Mono, Arial, Arial Black, Comic Sans MS, Courier New, Georgia, Impact, Times New Roman, Trebuchet MS, Verdana y Webdings.

En la siguiente tabla se presenta a modo de resumen una clasificación sobre las tipografías en relación con las diferentes plataformas en las que se encuentran. Un color verde indica que se encuentra en prácticamente la totalidad de los equipos con ese sistema, y amarillo que no lo son tanto. Indiferentemente de su color, todas son aceptadas como seguras para la web.

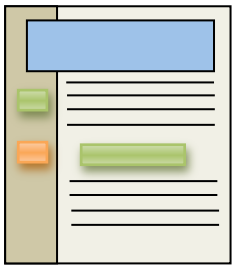
Fuente	Windows 9x / XP	Win Vista / 7	Mac OS X	Linux / Unix
Cambria				
Constantia				
Times New Roman				
Times				
Georgia				
Andale Mono				
Arial				
Arial Black				

Calibri				
Candara				
Century Gothic				
Corbel				
Helvetica				
Impact				
Trebuchet MS				
Verdana				
Comic Sans MS				
Consolas				
Courier New				
Courier				

Para hacer atractiva la lectura del contenido que se ofrece en el sitio web, es importante prestar especial interés al contraste visual que se crea entre los diferentes elementos. Entre estos elementos se encuentran los bloques de texto, imágenes, menús, enlaces y títulos. Si pusiéramos en nuestro sitio únicamente un texto con suma monotonía visual, sería fácil que la mayoría de los usuarios rechazara leer nuestro sitio. En primer lugar, porque es necesario que el usuario sea capaz de diferenciar las distintas partes del contenido para facilitarle así el acceso a la parte de la información de la que realmente está interesado.
Por ello es importante también no abusar de colores y resaltados, ya que el usuario no podría diferenciar las partes importantes, al igual que ocurría en el caso anterior.
Los consejos más habituales suelen ser diseñar sitios sencillos visualmente hablando, con contenido bien organizado, separando los bloques visualmente, seguir un esquema repetitivo en la maquetación, y resaltando únicamente las partes realmente importantes.



Contenido mal organizado



Contenido bien organizado

Otra parte importante para la legibilidad del contenido son todos aquellos aspectos relacionados con las alineaciones y los márgenes. Los márgenes añaden en mayor o menor medida comodidad al leer, ya que ayudan a separar el cuerpo del texto de la zona que lo rodea. Los bloques de texto pueden clasificarse en alineados a izquierda, alineados a derecha, centrado, o justificado.

Texto justificado: El texto justificado se encuentra alineado tanto a la izquierda como a la derecha creando un bloque de texto homogéneo. Se busca la máxima simetría y formalidad. Su funcionamiento se basa en modificar la separación entre palabras para crear esta estructura visual. Es muy difícil conseguir la justificación en resoluciones bajas como puede ser un cuadro de texto pequeño, o un web diseñado para móviles, y además se pierde claridad en la lectura en pantalla.

El texto justificado se encuentra alineado tanto a la izquierda como a la derecha creando un bloque de texto homogéneo. Se busca la máxima simetría y formalidad. Su funcionamiento se basa en modificar la separación entre palabras para crear esta estructura visual.

Texto alineado a izquierda: El texto alineado a la izquierda es la opción que aporta más legibilidad a una página web, dado que el margen izquierdo es constante en todo el bloque de texto. Se puede apreciar que al alinear al margen izquierdo, el derecho pierde su justificación.

El texto alineado a la izquierda es la opción que aporta más legibilidad a una página web, dado que el margen izquierdo es constante en todo el bloque de texto. Se puede apreciar que al alinear al margen izquierdo, el derecho pierde su justificación.

Texto alineado a la derecha: El contenido en este tipo de bloques aparece alineado al margen derecho. Como se puede apreciar este tipo de bloque aporta poca claridad en su lectura. Esto se debe porque en la escritura occidental, las frases se leen desde la izquierda hacia la derecha, lo que provoca confusión al intentar buscar el inicio de cada nueva línea.

El contenido en este tipo de bloques aparece alineado al margen derecho. Como se puede apreciar este tipo de bloque aporta poca claridad en su lectura.

Texto centrado: Este tipo de texto hace un reparto igualitario del espacio destinado al margen izquierdo como al margen derecho. Esto provoca que el texto aparezca en el centro del bloque de contención. La guía visual también se pierde al ser confuso encontrar el principio de cada nueva línea.

Este tipo de texto hace un reparto igualitario del espacio destinado al margen izquierdo como al margen derecho. Esto provoca que el texto aparezca en el centro del bloque de contención. La guía visual también se pierde al ser confuso encontrar el principio de cada nueva línea.

La longitud de cada línea también es otro factor importante que deberemos tener en cuenta a la hora de ofrecer una mejor legibilidad a los futuros usuarios del sitio web.

A una distancia normal de lectura, el ojo es capaz de observar de forma correcta unos 8 centímetros de ancho, por lo que evitaremos superar demasiado esa distancia. Esta regla no sólo es aplicable a los medios digitales, sino también a medios tradicionales como el papel, por lo que deberá tenerse en cuenta para las hojas de estilo que se apliquen para imprimir las distintas páginas web de las que se componga nuestro sitio.

Hay diversas teorías sobre cuál es el número de palabras adecuado que debe haber en cada línea, pero a grandes rasgos indican que deben estar entre 12 y 20 palabras. Esto es tan relativo porque hay muchos otros factores que influyen en el resultado como por ejemplo el interlineado, ya que cuanto mayor sea mayor legibilidad aportará.

6.1.6.- Ubicación del usuario

Tenemos que prever que el usuario no siempre va a seguir el camino que nosotros hemos previsto, ya sea por haber accedido a través de un buscador, o porque la tenga guardada en marcadores, o porque alguien la comparta a través de las redes sociales o blog. Por ello es posible que el usuario en ocasiones se pueda necesitar saber dónde se encuentra, para lo cual daremos soluciones en este apartado.

Es importante que la web tenga un logotipo fácilmente reconocible o muestre claramente el nombre de la web en la que nos encontramos, para así facilitar esta labor a los nuevos usuarios.

Para aquellos que ya estén nuestro sitio, deberemos ofrecerles una navegación constante, manteniendo una serie de elementos comunes a todas las páginas. Esto ayuda a dar coherencia e indicarle al usuario que realmente se continúa en la misma página, y es fácilmente implementable mediante una zona del sitio que sea común a todas páginas. Las páginas de inicio pueden ser una excepción ya que el cometido no es el mismo que el resto de páginas, y aquellas en las que aparezcan formularios pueden ser otra excepción, evitando mostrar elementos innecesarios para intentar evitar distracciones innecesarias.

Además de que sepa que se encuentra en nuestro sitio web, deberemos indicarle en qué lugar de él se encuentra. Para ello habrá que darle alguna indicación resaltada de la sección donde se encuentra en un lugar claro donde el usuario pueda verlo.

6.1.7.- Accesibilidad de los usuarios con discapacidades visuales

Independientemente de que nuestra web esté enfocada específicamente a un público mayoritariamente con discapacidades visuales, debemos intentar ofrecer a este colectivo una experiencia tan satisfactoria como a cualquier otro usuario.

Una discapacidad visual puede definirse según la OMS como “cualquier restricción o carencia (resultado de una deficiencia) de la capacidad de realizar una actividad en la misma forma o grado que se considera normal para un ser humano”.

En este caso nos centraremos sobretodo en las visuales, y debemos tener en cuenta que una discapacidad visual no es únicamente una ceguera, sino que puede ser cualquier tipo de limitación que le impida ver de forma correcta.

Uno de los aspectos más importantes es conseguir que el diseño sea lo más claro y ligero posible, evitando para el usuario todo tipo de confusiones. A decir verdad este esta recomendación es para el diseño de la web en general, ya que los usuarios “videntes” actúan fijándose únicamente en los aspectos que creen más importantes.

Para usuarios totalmente invidentes existen sintetizadores que son capaces de leer el contenido de la página web en la que está el usuario. Por ello es importante mostrar contenido adicional en formato texto para todas las imágenes de nuestro sitio, al menos aquellas que sean suficientemente importantes como para aportar algo al usuario. Esto se realiza mediante etiquetas alt y title de los elementos <img>.

Es conveniente también no redactar textos excesivamente largos puesto que dificulta la lectura, ya que para un usuario común es normal extraer información sin leer todo el contenido, pero para un usuario con deficiencia visual esto es tremendamente complicado. También lo será si debe esperar a que el sintetizador lea la parte en la que está interesada, que probablemente esté al final de todo el texto.

Para facilitar la lectura de las páginas con gran cantidad de textos, se recomienda utilizar encabezados h1, h2, … para separar el texto en las distintas partes. Esto facilita el reconocimiento de estos textos por un sintetizador, ya que es posible que nombrara los distintos encabezados al usuario hasta que este escogiera uno de ellos para su lectura.

En el caso de los usuarios invidentes, la solución pasa por ofrecer un contenido que pueda ser entendible por un ordenador, ya que probablemente nuestro sitio web sea procesado por un sintetizador capaz de leer el contenido para que sea entendible por el usuario.

Para los usuarios con menor grado de discapacidad visual, el propio navegador es capaz de ampliar ciertas partes de la página mediante una combinación de teclas. Para facilitar la experiencia de esta utilidad, debe evitarse codificar todos los tamaños absolutos. Esto es aplicable al fichero CSS, y debe configurarse en relación a un porcentaje. Si estamos hablando de un tamaño de contenedor, se establecerá un porcentaje sobre el propio contenedor en el que se encuentra. En caso de que estemos hablando de tamaño de fuente, los estableceremos en relación al tamaño de fuente predeterminada.

6.2.- Requisitos de la aplicación

Cuando vamos a definir la especificación de requisitos de nuestra aplicación deberemos tener en cuenta todos aquellos objetivos para los cuales va a ser desarrollada. Ver cuáles son las funciones que debe cumplir, los requisitos a tener en cuenta, y las características que deberá tener.

El objetivo de la especificación de requisitos es ajustarse a las necesidades que el cliente espera de la aplicación, y en caso de no ajustarse a ellos, poder rectificar a tiempo antes de empezar a desarrollar la aplicación.

6.2.1.- Captura de requisitos

Parece lógico que el primer paso deber ser capturar las necesidades del cliente para continuar todo el proceso a partir de este punto. En este caso el problema es que este documento es parte de un proyecto académico en el cuál no hay un cliente propiamente dicho. Lo que se ha hecho es proponer un modelo de negocio diferenciado de los ya existentes, y empezar a especificar los requisitos poniéndose siempre en el lugar del cliente.

Para paliar este hecho, a pesar de no tener un cliente propiamente dicho, habría que realizar una captura de requisitos. Deberemos en primer paso identificar todas aquellas personas relacionadas con el mundo de los gimnasios (que es el que corresponde a este PFC) que nos puedan aportar información sobre los procesos que se llevan a cabo en este tipo de organizaciones.

Para la captura de requisitos existen diversas técnicas que nos permiten ir recopilando la mayor cantidad posible de información, procesos que se desarrollan, opiniones, mejoras posibles, y en general todos aquellos datos que puedan ayudarlos a modelar nuestro sistema de información.

Una de estas técnicas que podemos utilizar en este proceso es realizar una serie de entrevistas con algunos trabajadores o directores de algún gimnasio al que tengamos acceso. Esta entrevista debe estar totalmente preparada de antemano, teniendo en cuenta el tipo de cargo que dispone en la organización, preparando las preguntas que van a realizarse (dependiendo del cargo de la persona entrevistada), intentando acotar las respuestas para que el entrevistado no se vaya por las ramas y documentar todos los resultados para extraer posteriormente la mayor cantidad posible de información.

Podemos hacer una plantilla para ir completando los distintos datos que tenemos que tomar del entrevistado, veremos un ejemplo en la siguiente tabla:

Pregunta	Respuesta
Nombre y Apellidos	
Cargo que ocupa en el centro	
Edad	
¿Qué tareas realiza habitualmente en su trabajo?	
¿Cuáles de ellas se realizan con un equipo informático y cuáles no?	
¿Qué software utiliza para la realización de dichas tareas?	
¿Cómo establece comunicación la empresa con usted?	
¿Cómo establece usted comunicación con los usuarios?	
¿Qué tareas realizan habitualmente los	

clientes?	
¿Cuáles de ellas se realizan habitualmente con el uso de un equipo informático?	
¿Qué datos de los clientes se guardan en el gimnasio?	
¿Dispone el gimnasio de una página web? Y en su caso indicarla.	
¿La información que dispone es de utilidad para alguien?	
¿Qué tareas piensa que pueden ser realizadas informáticamente?	
¿Qué tareas piensa que podrían ser eficientes?	

Como se puede apreciar el vocabulario utilizado no es nada técnico, intentando de este modo que el entrevistado pueda entender mejor las preguntas, y responder con mayor facilidad, ya que es muy probable que no comprenda este tipo de lenguaje.

Debemos ser nosotros por lo tanto los que debamos realizar el esfuerzo en todo el proceso de comunicación, intentando traducir a lenguaje técnico todo aquello que nos intenta transmitir el entrevistado. A partir de todos estos datos tendremos que ser capaces de especificar los requisitos en un lenguaje formal y previamente fijado, ya que serán los documentos que usará el equipo de desarrollo en el capítulo de la implementación.

Otro lugar de donde podemos recolectar información son las web de los gimnasios que podamos encontrar por Internet. De este modo podemos establecer pautas sobre la información que estos muestran y el tipo de acciones que permiten realizar.

6.2.2.- Especificación de requisitos

A partir de este momento deberemos especificar los requisitos a partir de los datos recolectados en el apartado anterior, tanto de las entrevistas que hayamos realizado a trabajadores, como a clientes, como a la investigación realizada a través de las distintas webs.

En primer lugar, debemos tener el cuenta que la aplicación debe ser una plataforma que ofrezca información sobre las instalaciones con las que cuenta el gimnasio, intentando ofrecer una visión muy gráfica y llamativa. Además, debe ofrecerse información sobre los servicios que

se ofrecen, como por ejemplo las actividades, a qué horarios se realizan, la localización del local, etc.

Como se pretende que sea una web funcional que permita a los usuarios algo más que ver información, se deberán implementar funcionalidades que permitan a los clientes del gimnasio efectuar una serie de tareas desde el mismo sitio web como la gestión de reservas, información de nuestra cuenta de usuario, etc.

Por otra parte se realizará un sistema de geolocalización que permita a la aplicación conocer la ubicación del usuario desde donde está consultando nuestra web, para poder ofrecerle un servicio que le permita conocer cuál sería la mejor ruta que podrían seguir para llegar hasta nuestras instalaciones.

6.2.2.1.- Requisitos de información

Uno de los aspectos más importantes de todo sitio Web son las propias informaciones que se debe ofrecer al cliente, ya que tenemos que tener en cuenta que en un primer vistazo este va a ser nuestro activo más importante.

Los usuarios que lleguen a nuestro sitio Web van a buscar contenido original y de calidad, el cual le pueda ser de utilidad. Por una parte incentivaremos a los nuevos usuarios para "cazarlos" en su primera visita, y por otro lograremos que vuelvan a nuestro sitio una y otra vez.

Hemos hablado de lo importante que es el contenido para los usuarios, pero también hay que tener en cuenta que debe ser atractivo para los buscadores. Esto influye por un lado en que deben ser originales, ya que los algoritmos de los distintos buscadores penalizan el contenido copiado. Por otro lado influye en la forma que estructuramos los contenidos, por ejemplo creando títulos relevantes.

Uno de los puntos importantes de nuestra aplicación es la identificación y realización de todos los contenidos informativos de los que dispondrá nuestra aplicación web. Para ello deberemos analizar los requisitos de contenido impuestos por el cliente, y extraer de aquí una visión global sobre los aspectos a desarrollar en nuestra aplicación.

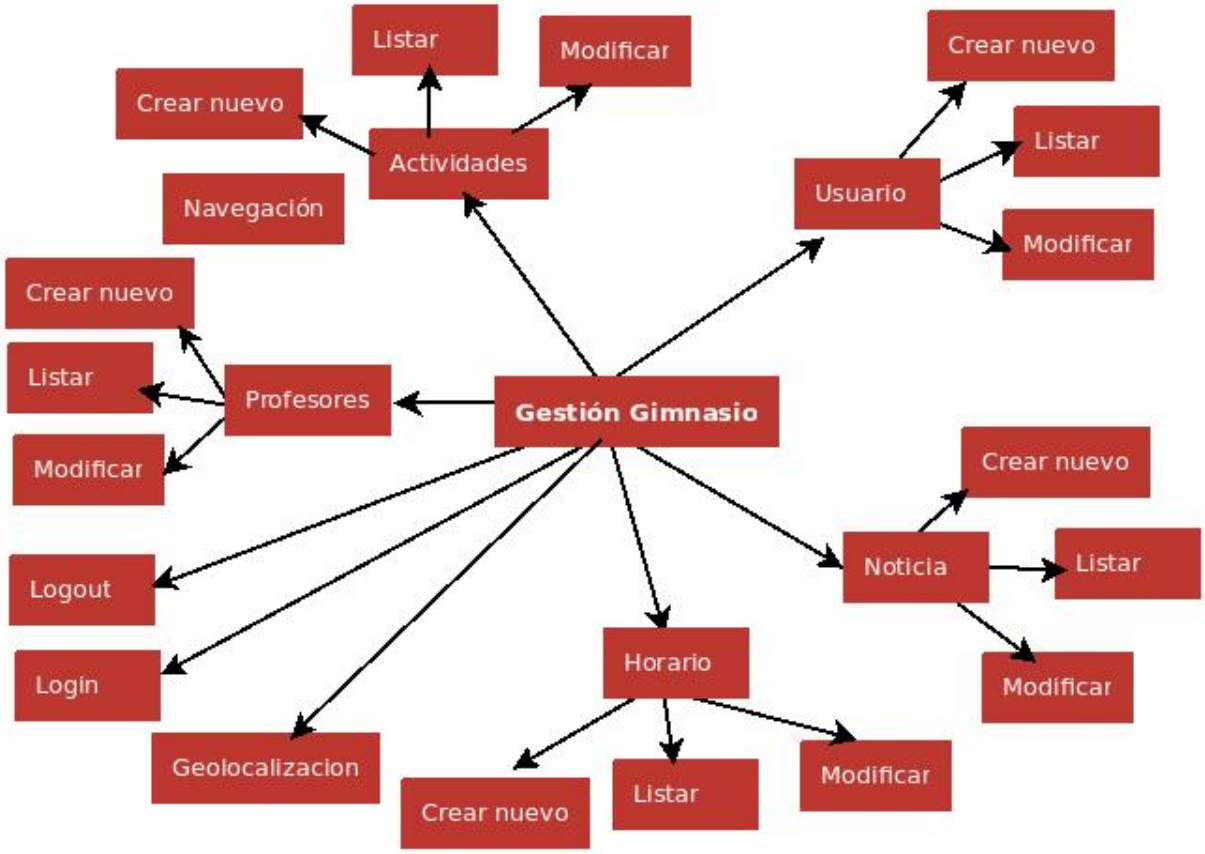
El primer paso una vez analizados los requisitos de información, deberemos identificar los contenidos que vamos a procesar. En nuestro caso se ha realizado una plantilla con las entidades que a priori van a aparecer en nuestra aplicación.

Una vez que se han identificado los requisitos de información deberemos elaborar con mayor grado de precisión una lista de contenidos de la aplicación web. Los datos que recabaremos serán la descripción, palabras clave, audiencia a la que se orienta y el formato en que se muestra. Este es un ejemplo sobre nuestra aplicación:

Título	Descripción	Palabras Clave	Audiencia	Formato
Clientes	Información de los clientes y sus deudas		Admin	Texto
Instalaciones	Muestra todas las instalaciones	Plano, instalación, sala	Todos	Mapa con texto e imágenes
Localización geográfica	Muestra la ubicación del gimnasio	Valencia, gimnasio, ubicación, localización	Todos	Mapa geográfico
Profesores	Muestra algunos datos de los profesores		Sólo clientes	Texto
Actividades	Descripción de la actividad completa	Compartir	Todos	Texto con imágenes
Horarios	Muestra los horarios de las distintas actividades		Todos	Texto
Noticias	Los responsables podrán colgar noticias de interés		Todos	Texto con imágenes
Quienes somos	Información institucional de la empresa	Empresa, gimnasio, innovación	Todos	Texto

6.2.2.2.- Requisitos funcionales

Otro aspecto importante a la hora de especificar los requisitos de nuestra aplicación, es identificar las distintas funcionalidades que los usuarios podrán realizar los usuarios en nuestro sitio web. Para ello identificaremos las distintas funcionalidades y las estructuraremos en un organigrama como el siguiente:



6.3.- Análisis

Para esta fase es imprescindible haber completado la fase de las especificaciones para poder comprender cuáles son los requisitos de nuestro sistema. Una vez recolectados todos ellos, el siguiente paso será analizar todos y cada uno de ellos para comenzar a darles forma. En este proceso comenzaremos definiendo un mapa conceptual de los apartados del sitio a partir de las unidades de información recolectadas en el proceso anterior. Posteriormente se definirán los escenarios, que son los casos de uso desgranados en los distintos usuarios de nuestra web,

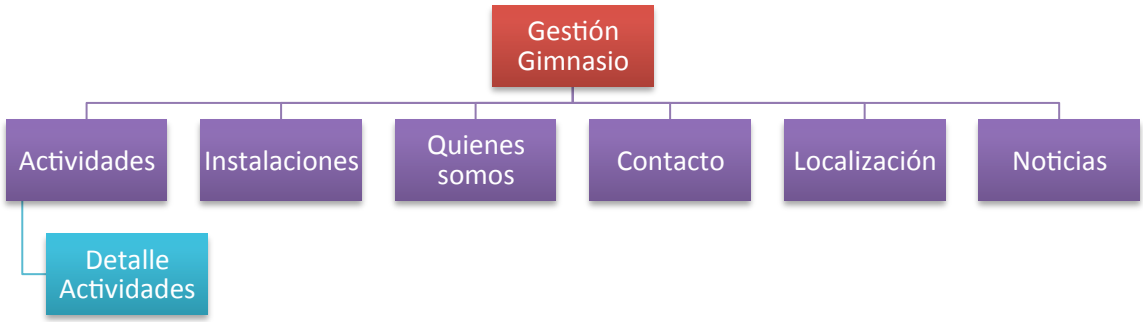
ya que no todos los usuarios pueden acceder a todas las opciones. Por último se empezará a modelar la interfaz de la web mediante los primeros bocetos.

En esta fase todavía se obviarán todos los aspectos técnicos como tecnologías a utilizar, lenguajes de programación, etc. Simplemente se analizarán los distintos requisitos obtenidos para comenzar a modelar el sistema poco a poco, identificando los elementos del sistema, y las relaciones que pueda haber entre ellos.

6.3.1.- Mapa conceptual

La idea del mapa conceptual es establecer cada una jerarquía de navegación a través del sitio web, basándose en las distintas unidades de información extraídas en apartados anteriores. Este apartado es útil para empezar a declarar toda la estructura del sitio, empezando por los escenarios de uso. Finalmente cada apartado acabará siendo un fichero independiente con su propio código.

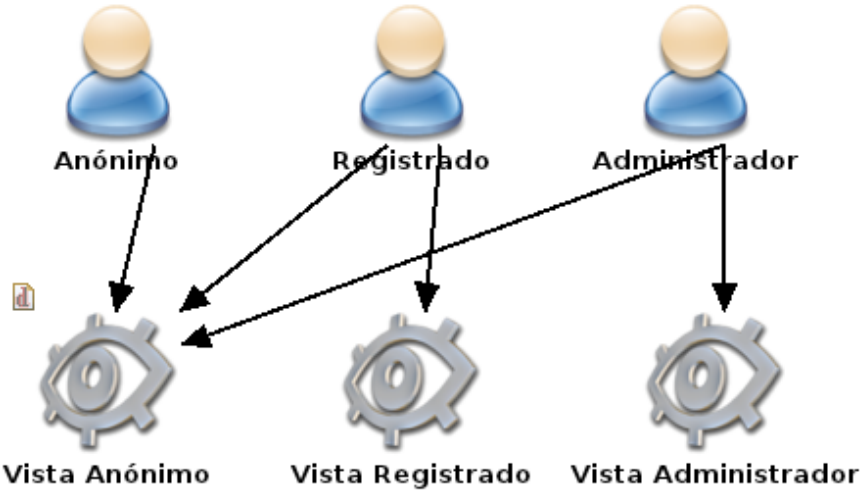
En siguientes los siguientes apartados de este proceso de análisis, iremos afinando estos apartados partiendo de la base de este mapa conceptual, hasta obtener un modelado completo del sitio.



El concepto “Gestión Gimnasio” que aparece en el nivel superior no es un concepto propiamente dicho, pero podría entenderse como la página inicial a la cual se accedería en primera instancia al entrar al sitio web, y que permitiría el acceso al resto de apartados. El resto de apartados mostrados (colores morado y azul) pueden considerarse como páginas distintas dentro de nuestro sitio, que posteriormente serán implementadas.

6.3.2.- Escenarios

Mediante la definición de los diferentes actores que jugarán un papel importante en nuestra aplicación, lograremos identificar cada uno de los roles que deberemos diferenciar antes de desarrollar la aplicación. Cada uno de ellos juega un papel diferente en nuestro sistema, por lo que a cada uno de ellos deberá asignársele distintas propiedades. Estas propiedades son tanto navegacionales, como de operaciones o visualización de contenidos.



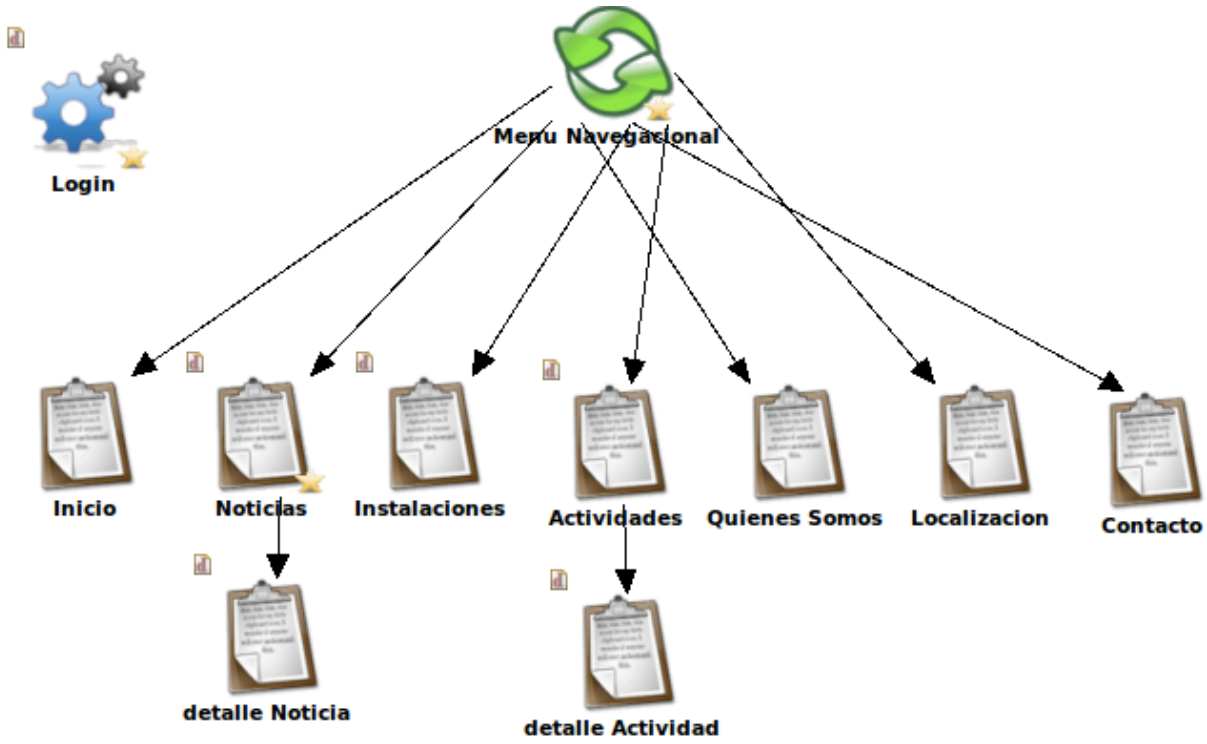
En el diagrama anterior se han declarado los tres roles que se han identificado en nuestra aplicación. El primero es el usuario anónimo que es todo aquel usuario que acceda por primera vez a nuestra web, no disponga de usuario o ha decidido no loguearse. No puede acceder a ejecutar ninguna operación salvo la de loguearse para escalar algunos privilegios. El segundo es el usuario registrado que será el rol común de nuestros clientes cuando accedan a nuestra web para efectuar una reserva de una actividad. Podrá realizar algunas acciones relativas a su propia cuenta de usuario, sus facturas o desconectarse de su sesión. Por último y no menos importante el administrador, que será aquel usuario capaz de administrar el portal web por completo, desde la introducción de noticias, hasta la gestión de usuarios. También dispone de información privilegiada sobre los usuarios.

6.3.2.1.- Vista Anónimo

La vista anónimo será aquella que permite consultar la mayor parte de la información que es pública en la red, como las actividades, instalaciones, contacto, localización. Para esta vista no será necesario ningún registro de usuario, aunque se dará la opción de insertar su usuario y contraseña para poder acceder a la vista registrado.

La parte del menú navegacional será la encargada de crear la conexión entre las distintas vistas, ya que permitirá navegar a cualquiera de ellas. Se ha procurado que posea enlaces a los apartados más relevantes del sitio web, sin sobrecargarlo demasiado, e intentando por otra parte que la importancia de todos los enlaces sea similar.

Otra parte importante de la vista Anónimo es la parte de la operación Login. Esta parte será la encargada de mostrar un formulario donde el usuario podrá indicar su nombre de usuario y contraseña para poder acceder al sitio con un nivel de privilegios diferente, y con la posibilidad de ejecutar acciones en su propio nombre (como puede ser reservar una actividad)



6.3.2.2.- Vista Registrado

A la vista registrado se accederá una vez el usuario haya realizado un inicio de sesión satisfactorio. Ésta será la vista que permita realizar la mayor parte de las acciones a un usuario común, ya que éstas van asociadas a una determinada cuenta de usuario (como efectuar reservas en las instalaciones, consultar los datos del usuario y modificarlos)

La parte de la vista Registrado posee un apartado destinado a ver la información propia de la cuenta del usuario que será la encargada de mostrar y modificar nuestros datos, y otro para que el cliente pueda ver de un vistazo las reservas que ha realizado y que están próximas a la fecha actual.

La operación de Login del usuario anónimo queda reescrita por la operación Logout, ya que un usuario registrado no tendrá que volver a hacer login hasta que no haga un logout.

6.3.2.3.- Vista Administrador

A la vista Administrador se accederá cuando se haya iniciado sesión con un usuario que dispongo de privilegios de administrador en nuestra aplicación. El método de acceso es el mismo que el usuario registrado común.

La vista Administrador contendrá un conjunto de actividades que serán de utilidad al administrador a la hora de añadir contenidos nuevos y efectuar un mantenimiento completo del sitio web. Algunas de las acciones que podrá realizar serán la inserción, modificación y eliminación de actividades, inserción de noticias, modificación de instalaciones, etc.

La operación Logout permite al usuario administrador cerrar su sesión y volver a ser usuario anónimo. Para volver a realizar operaciones de administración deberá hacer login nuevamente con su usuario.

6.3.3.- Primeros bocetos

Hablar de cómo debe ser la estructura de una página web puede resultar un concepto demasiado abstracto para explicar a otras personas, y mucho menos para discutir con un cliente acerca del formato más conveniente que deberíamos adoptar.

Esta situación puede darse en una reunión del equipo de desarrollo en la que sea conveniente tomar decisiones sobre cuáles son los requisitos visuales de una determinada página, y cuál debe ser la forma más adecuada de llevarlo a cabo. Otra posible situación es en la fase en que el cliente nos presenta la especificación de requisitos en las primeras fases, dónde deberemos mostrar al cliente un prototipo de la idea del cliente, para que pueda comprobar si se adecúa a sus requisitos (y poder hacer modificaciones sobre él)

Parece importante por lo tanto crear una representación gráfica que permita poner en común las ideas del equipo, permitiendo valorar las posibles ideas y realizar modificaciones de una forma lo más colaborativa posible. Este aspecto es particularmente importante también en la relación con el cliente, ya que la carencia de un vocabulario técnico puede impedir poner algunos puntos en común con el equipo de desarrollo.

Este tipo de representación se conoce también como wireframe, y pretende ser una guía que permita mostrar el esqueleto de una página web de una forma lo más visual posible, sin prestar excesiva atención a los detalles, pero dejando clara la disposición de los elementos que la forman, como los contenidos, los menús de navegación, las acciones que se pueden realizar, los contenidos multimedia, etc.

Estas representaciones pueden realizarse perfectamente al modo tradicional con papel y bolígrafo, pero tienen la desventaja de ser menos flexibles a modificaciones que una versión informatizada. Para el desarrollo del presente proyecto se usará un software destinado a crear este tipo de representaciones, en este caso se ha optado por una herramienta on-line llamada Cacoo.

En próximos apartados se realizará un diseño más exhaustivo de todos y cada uno de los apartados que conforman nuestro sitio web. Aquí lo único que se pretende es mostrar una vista general de cual será su forma.

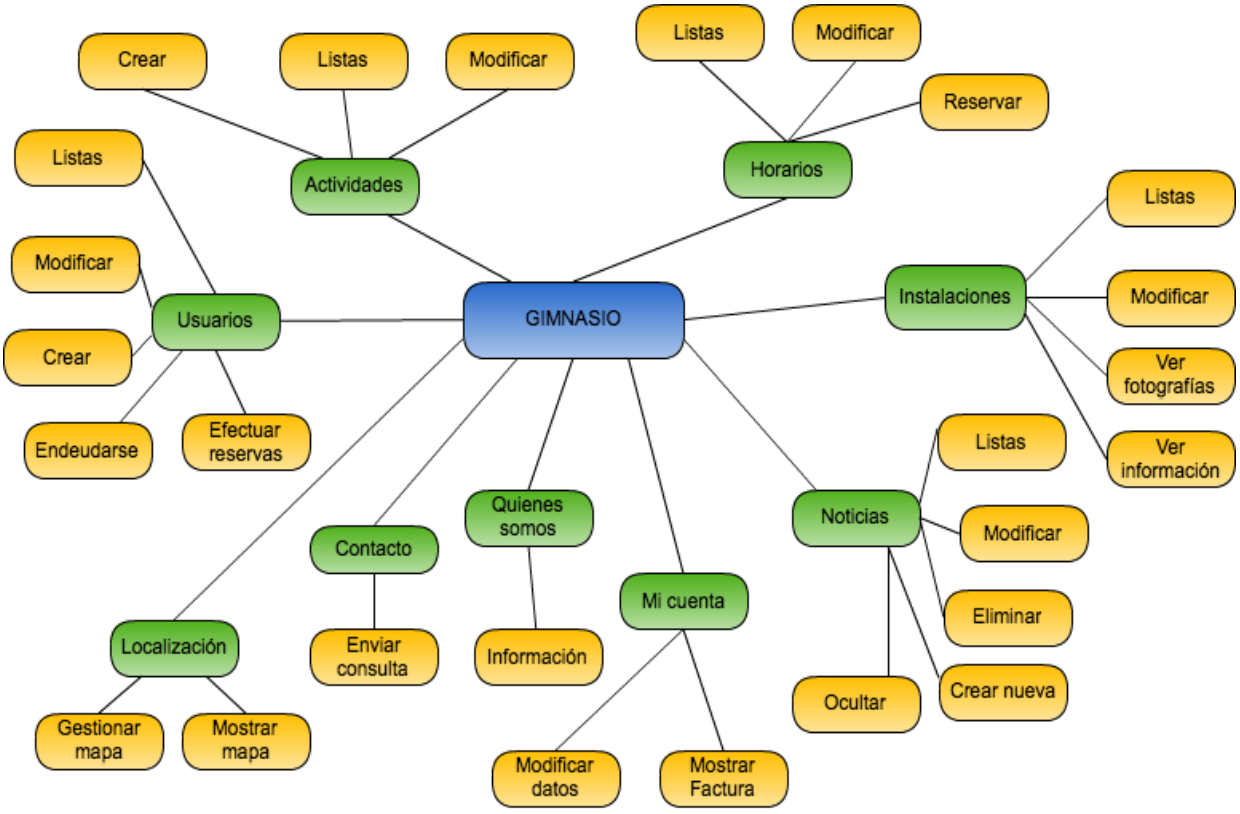
6.3.4.- Esquemas jerarquía contenidos

Es importante establecer una jerarquía de contenidos, mostrando los lugares aproximados donde estos irán localizados una vez esté desarrollada la aplicación. Este esquema será usado posteriormente en futuros apartados del proceso de desarrollo con el fin de acabar de modelar las distintas páginas.

Aquí no se pone interés al diseño o colocación de las distintos contenidos, simplemente se pretende poner de manifiesto los contenidos que aparecerán en cada apartado (página de nuestro sitio web).

6.3.5.- Esquemas jerarquía operaciones

Me modo similar al apartado anterior, también es importante analizar qué operaciones se van a poder realizar en cada una de las distintas páginas. Hay que recalcar que aquí no se hace distinción entre las operaciones a las que tiene acceso cada tipo de usuario, para ello hay que consultar los escenarios.



6.3.6.- Diagramas clases

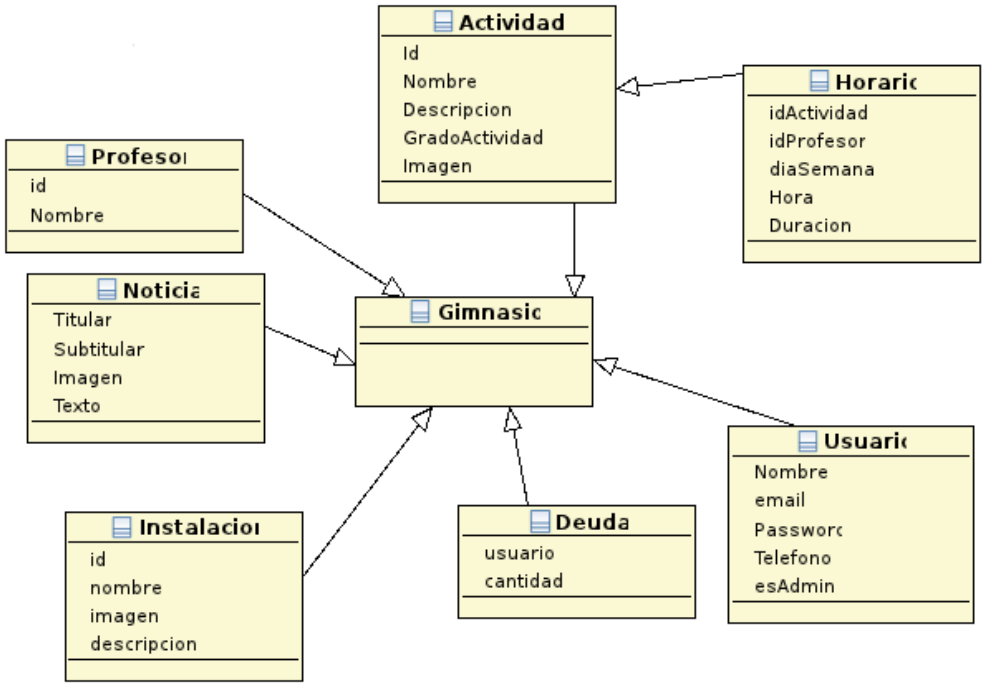
El diagrama de clases es un diagrama perteneciente al estándar UML que pretende dar respuesta a las clases utilizadas, sus atributos, y las relaciones existentes entre ellos. La finalidad de este diagrama es mostrar la estructura del sistema que se va a desarrollar. Dicho diagrama es utilizado en el proceso de análisis y diseño de los sistemas.

Cada entidad es representada como un caja, que es la representación para cada clase. La propia entidad dispone de una serie de atributos o propiedades, que serán representadas dentro de la misma caja. Dicha representación pretende indicar que dichas propiedades pertenecen a esa clase en concreto.

También es posible describir las operaciones que es posible realizar con ese objeto, y suele diferenciarse de las propiedades por aparecer una línea horizontal de separación entre ambas, y escribirse con el símbolo () al final. En este proyecto se ha pretendido orientar la estructura de 3 niveles, por lo que se ha evitado usar esta forma de definir métodos (a pesar de que es perfectamente válida).

Para la realización de este diagrama es conveniente la utilización de un software específico, que permita entre otras cosas definir el tipo de datos de los atributos. También permite una modificación más rápida, y es interesante para definir otro tipo de diagramas a partir del

diagrama de clases. En este caso se ha utilizado uno llamado Moskitt, gratuito y disponible para Linux, MacOS X y Windows.



6.4.- Diseño

Para el desarrollo de la fase de diseño utilizaremos los documentos que se han generado en la fase de análisis. En este caso habrá que añadirle los aspectos técnicos que permitirán llevar a cabo la implementación del sistema completo.

Esta fase tiene como objetivo principal documentar la forma de resolver los problemas que se han estudiado en la fase de análisis.

En caso de que el desarrollo vaya a ser llevado a cabo por un equipo de desarrollo, esta fase implicará la participación activa de todo el equipo.

6.4.1.- Arquitectura 3 niveles

En toda implementación software de proyectos de cierta envergadura es conveniente la utilización de una separación entre diversas capas. La separación principal viene entre la capa de diseño y la de la lógica de diseño.

La motivación de esta separación por capa viene dada por la necesidad de adelantarse a las posibles modificaciones o mejoras del código. Si en un momento dado es necesario un cambio

en el código, deberá identificarse a qué capa corresponde dicha modificación. El siguiente paso será realizar la modificación únicamente en esa capa, utilizando la interfaz de programación de la capa inferior.

De esta forma ayudaremos a mantener un código más amigable para el programado, con mayor facilidad para comprenderlo, con un mantenimiento menos costoso, y además permitiendo realizar nuevos métodos en menos tiempo.

En la gráfico inferior puede verse la separación en capas de la arquitectura escogida para este proyecto, concretamente un diseño en tres capas:

En primer lugar la capa de persistencia se encarga de los accesos a la base de datos, por lo que su implementación depende de la tecnología de base de datos utilizada. Ésta se encargará de realizar las conexiones a la base de datos y de ejecutar las consultas, inserciones, modificaciones y borrados.

En la capa de lógica se encuentran los distintos métodos que se encargan de toda la lógica de negocio, estos son aquellos que se encargan de procesar los datos para establecer acciones con ellos. Dichos datos son extraídos a través de los métodos que proporciona la capa de persistencia.

La capa de presentación se encuentra en la cumbre de toda la arquitectura, y se encarga de mostrar al usuario todos los datos una vez han sido procesados por la lógica de negocio. La utilidad de esta capa es presentación de todos los datos, y la interacción con el usuario.

6.4.2.- Capa de presentación

La capa de presentación se encarga de todos aquellos aspectos que tienen que ver con la propia visualización de la interfaz de la aplicación. Uno de los apartados serán los bocetos del diseño que cada uno de los apartados presentados en el mapa conceptual. Además será crucial

definir las vistas de cada uno de ellos especificando los contenidos que incluirá. Habrá apartados comunes a todas las páginas que se definirán de modo general. Por último se incluirán los patrones de diseño utilizados en la industria de la ingeniería web para resolver ciertos problemas de interfaz que pueden aparecer al intentar diseñar una aplicación web.

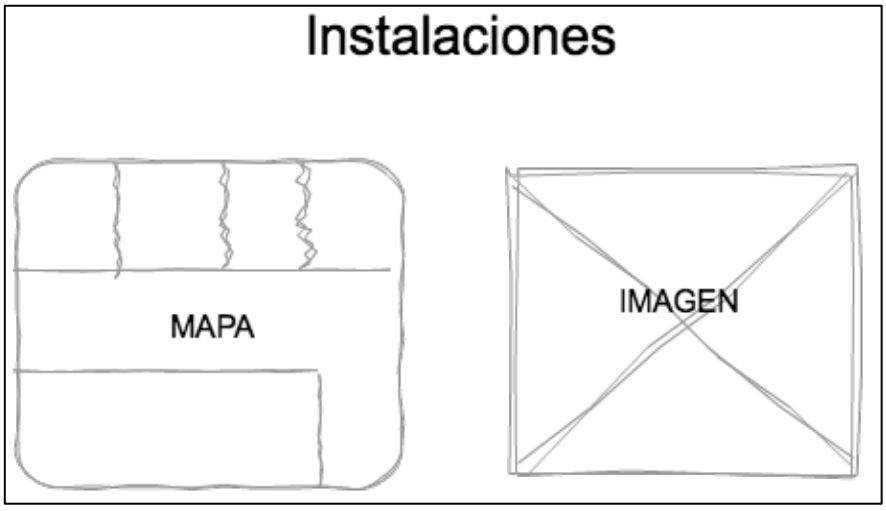
6.4.2.1.- Bocetos específicos

Una vez ya se ha realizado el primer boceto general de nuestra interfaz, y ya están declaradas las distintas vistas que va a tener nuestra aplicación, es hora de empezar a definir las diseñar los bocetos de cada uno de los distintos apartados.

Como el boceto general ya está diseñado, a partir de este punto se va a obviar el diseño ya realizado. La parte mostrada en los siguientes apartados muestra únicamente la parte que se muestra en el contenedor principal, esto es para simplificar los bocetos de los distintos apartados.

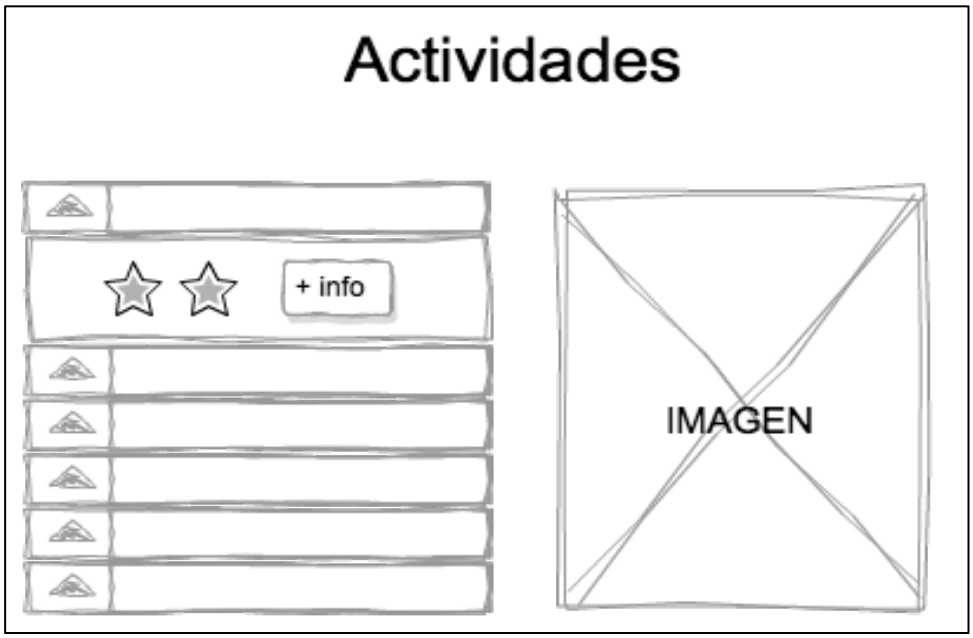
Instalaciones

El apartado "Instalaciones" mostrará dos zonas claramente diferenciadas. En la zona izquierda aparecerá el plano del gimnasio con las distintas estancias representadas en él. En la parte derecha aparecerá una imagen fija, junto con una indicación sobre el uso de la página. Sobre el plano de la parte izquierda se podrá hacer click sobre una de las distintas estancias que aparecen. Una vez realizada dicha acción, la imagen de la zona derecha cambiará para mostrar ahora una imagen de la estancia en la que se ha hecho click. A su vez también aparecerá un pequeño comentario sobre la estancia.



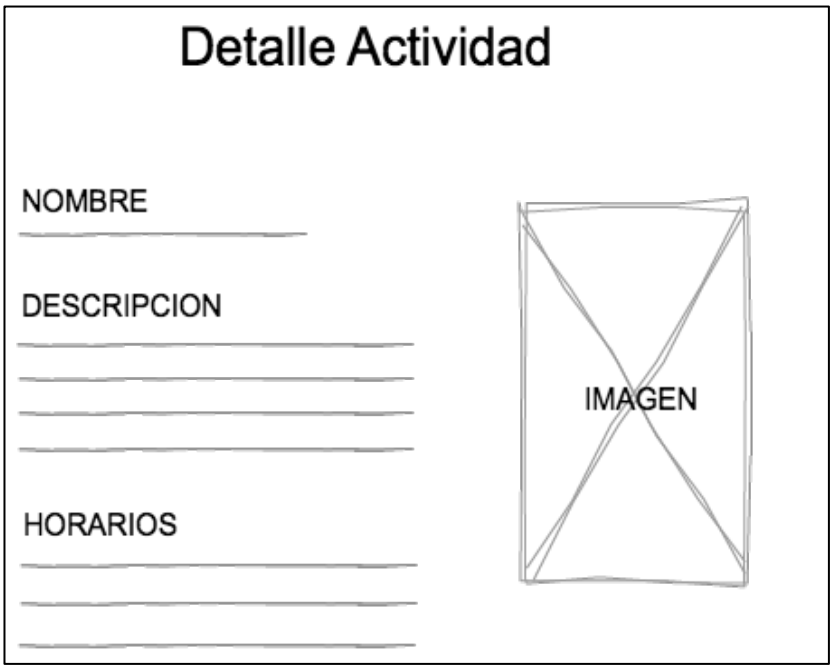
Actividades

En el apartado "Actividades" aparecerán dos zonas claramente diferenciadas, siendo la imagen de la parte derecha totalmente estática, mostrando una imagen fija. La parte izquierda muestra una estructura en forma de acordeón donde se podrán seleccionar las distintas actividades haciendo click en su respectivo nombre. Una vez que se ha seleccionado una actividad aparecerá un apartado con el grado de actividad que exige su realización y un botón donde poder conseguir más información. Este botón llevará a otra página diferente donde únicamente aparecerá información sobre esa actividad.



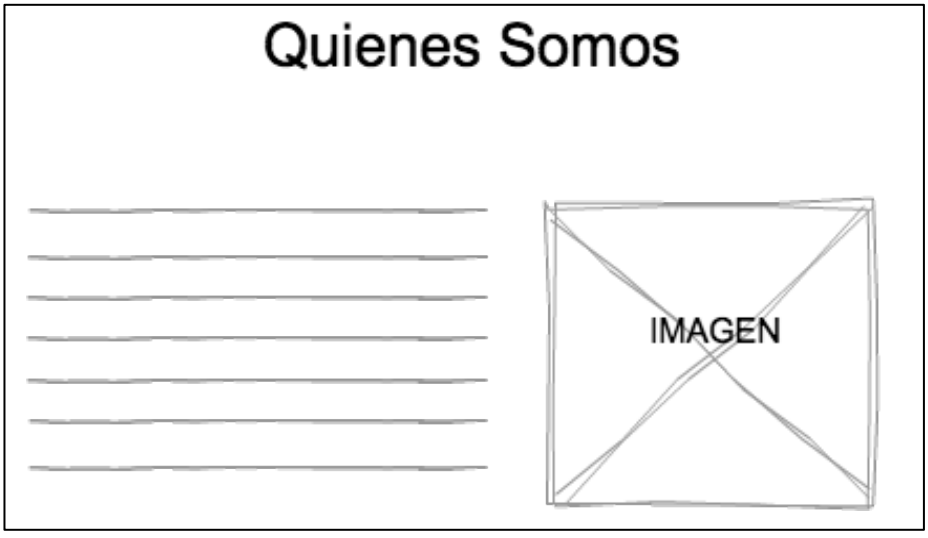
Detalle Actividad

En el apartado "Detalle actividad" aparecerá la únicamente la información relativa a la actividad que ha sido seleccionada. Aquí también aparecen dos zonas bien diferenciadas, una a la parte izquierda y otra a la derecha. En la zona derecha aparecerá una imagen representativa de la actividad seleccionada. En la parte izquierda aparecerán distintos conceptos como son el nombre de la actividad, la descripción de la misma, y los horarios en los que está disponible su reserva.



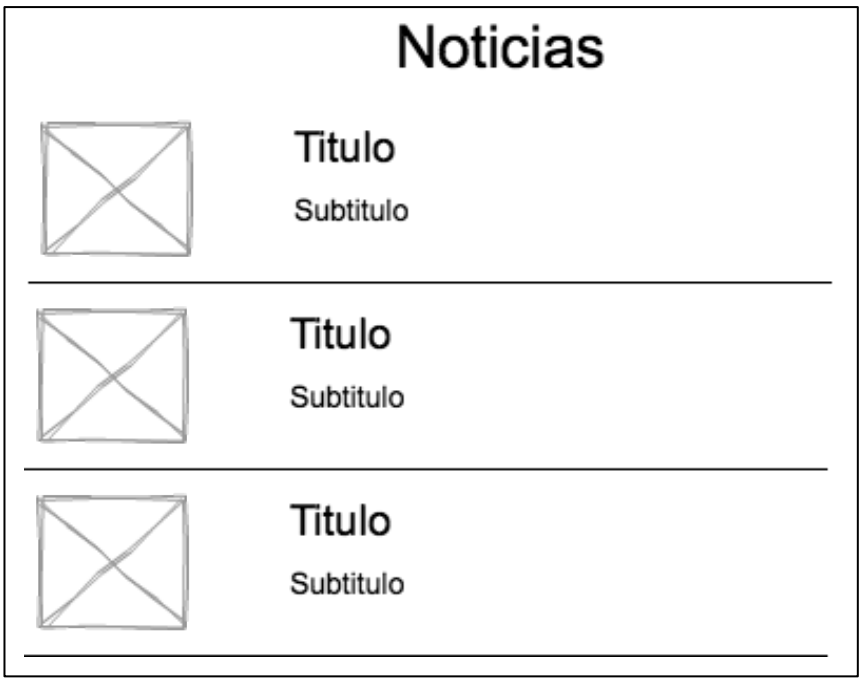
Quienes Somos

En el apartado "Quienes somos" volverán a aparecer las dos zonas comentadas en apartado anteriores. La derecha con una imagen fija cuya única motivación es estética. En la zona de la izquierda es donde aparecerá toda la información que debe aparecer en este apartado, concretamente la información institucional de la empresa o gimnasio.



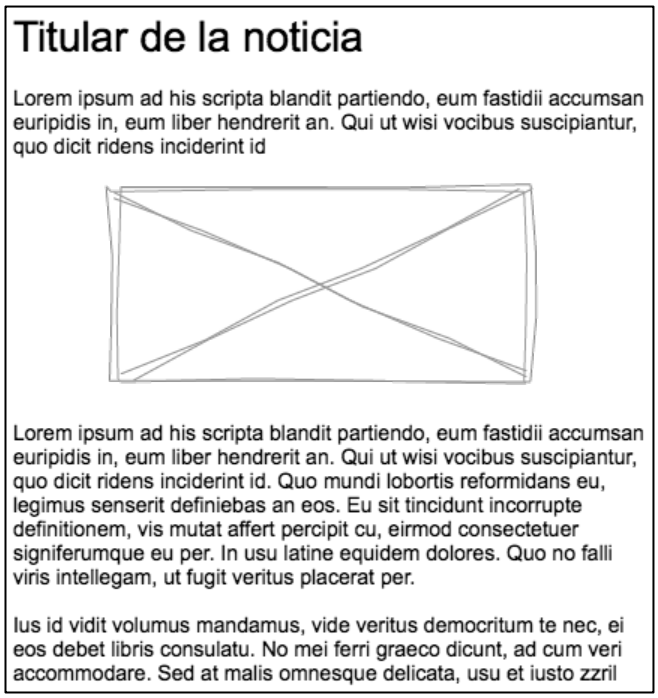
Noticias

El apartado "Noticias" muestra un listado de las últimas noticias que han aparecido en el portal. Para que sea lo más visual posible se muestra la imagen que corresponda a dicha noticia. Además se muestra el titular y el subtitular para captar el interés de los usuarios. Dichas noticias se muestran por orden de novedad, por lo que las más recientes aparecerán al principio, mientras que las últimas irán desapareciendo.



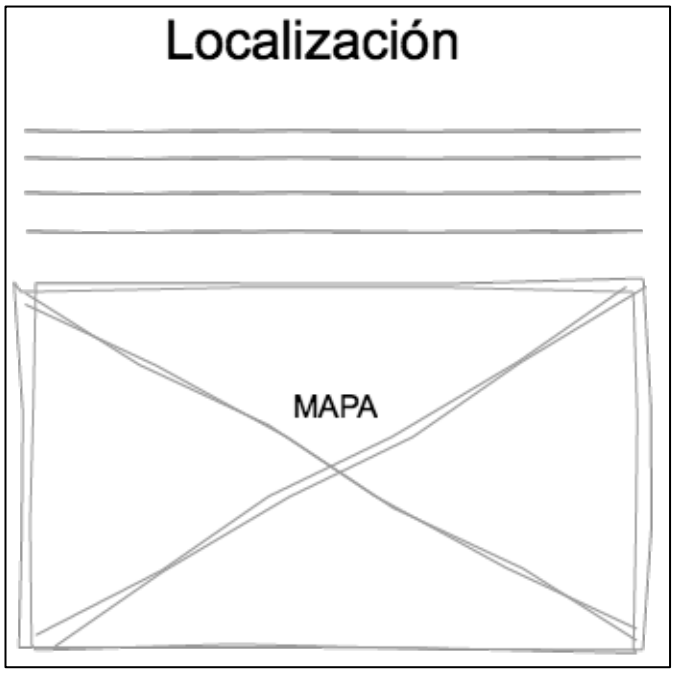
Detalle Noticia

En el apartado "Detalle Noticia" aparecerá únicamente la noticia en la que se ha hecho click en el listado de noticias. Los detalles que se mostrarán será el titular en primera instancia en un tamaño superior al resto. A continuación aparecerá el subtitulo, que no es otra cosa que un resumen de 2 o 3 líneas sobre la noticia. Después la propia noticia dispone de una imagen que será mostrada después del resumen. Por último se mostrará el texto completo de la noticia.



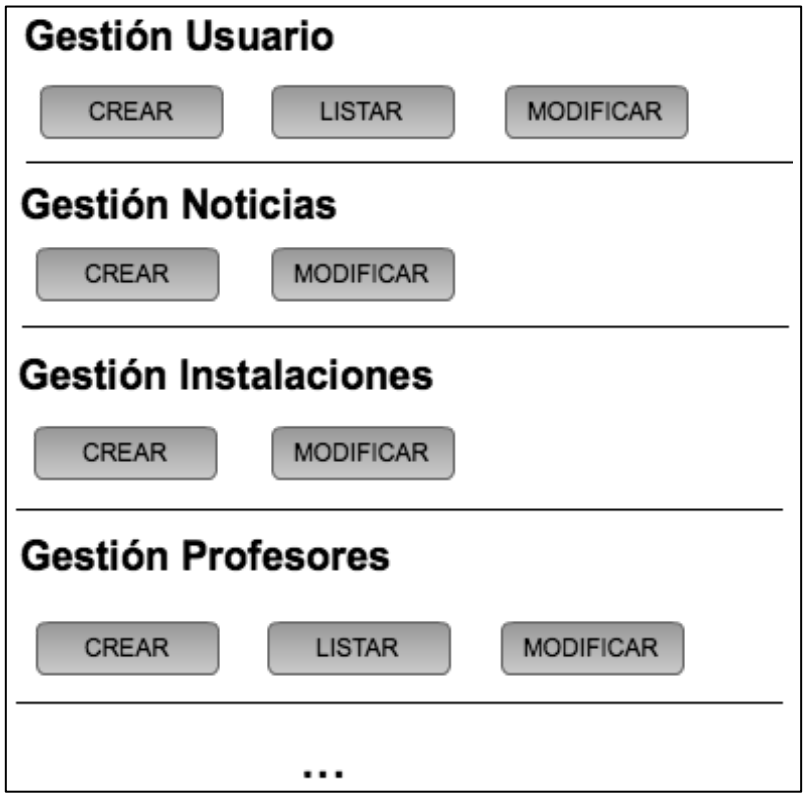
Localización

En el apartado “Localización” aparece una zona de información tipo texto donde se habla de la localización donde se encuentra el gimnasio. La siguiente zona es donde se implementa toda la funcionalidad de ofrecer una ruta hasta el gimnasio dependiendo de la posición del usuario. Esta zona mostrará un mapa ocupando la totalidad del ancho disponible para maximizar la claridad en la medida de lo posible a la hora de consultar la ruta. En caso de que esta funcionalidad no esté disponible, se ofrecerá una imagen fija con la localización del gimnasio.

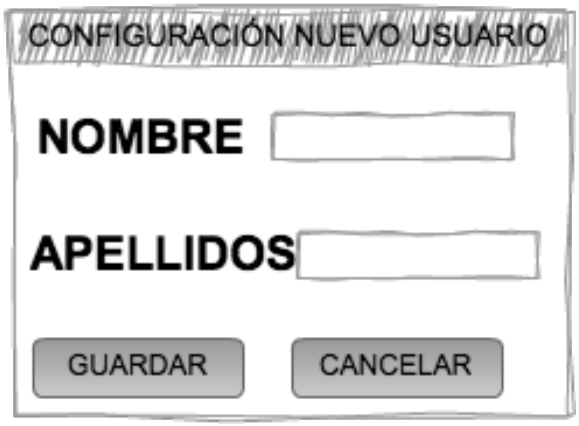


Configuración

El apartado “Configuración” será la parte donde aparecerán las distintas configuraciones que llevará a cabo el administrador del sistema. Cada unidad de información dispondrá de un apartado diferenciado (aunque en la imagen no aparecen todas), donde se podrán crear nuevos elementos, modificarlos, etc. Este apartado dispone de gran cantidad de opciones, por lo que se dispondrá de una plantilla que separe en grupos las distintas opciones disponibles.



Dentro de cada apartado de configuración, tanto los de crear un nuevo elemento, como los de modificar elementos ya existentes, aparecerá una nueva ventana donde rellenar los distintos campos que definen las propiedades del objeto. La forma de rellenar estos campos será mediante un formulario, y un botón guardar.

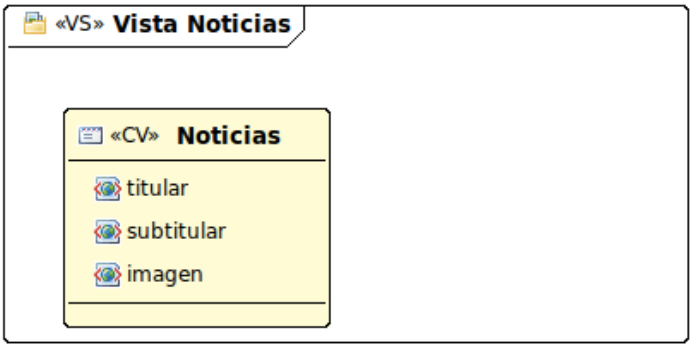


6.4.2.2.- Escenarios completos

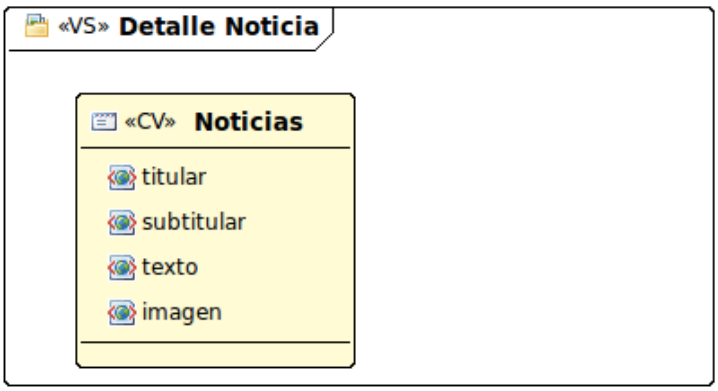
En este apartado se intentan definir los contenidos que aparecen en los distintos apartados del sitio. El resultado de nuestra aplicación vendrá definido por la unión de los bocetos y las vistas de los diferentes escenarios. Es importante haber definido el diagrama de clases correctamente antes de llegar a este paso.

Las unidades de información vienen definidas por el cuadro blanco y el rótulo en la parte superior izquierda. Las clases (que hemos definido en el diagrama de clases) aparecen en un color amarillento, y en la parte inferior de éstas, aparecen los atributos de estas clases que serán mostrados al usuario. No se tendrán en cuenta a la hora de la realización de los escenarios, los atributos de una clase cualquiera que vayan a ser utilizados únicamente para realizar operaciones internas.

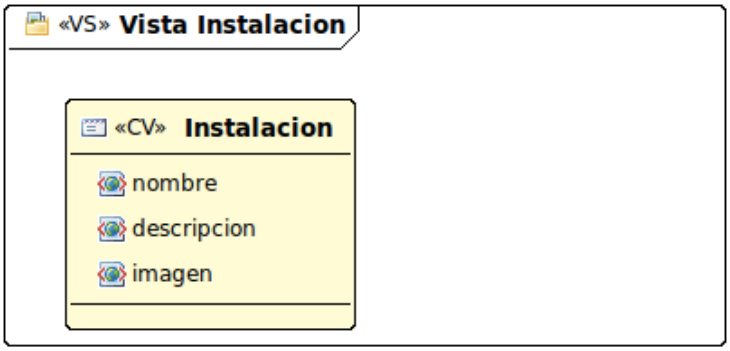
- Noticias



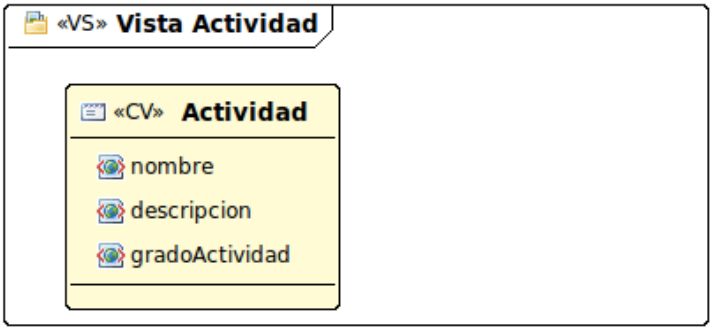
- Detalle Noticia



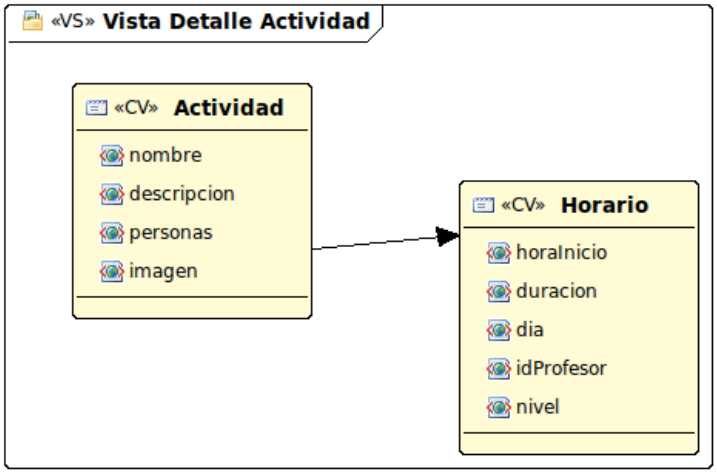
- Instalaciones



- Actividades



- Detalle Actividad



6.4.2.3.- Partes de la interfaz

En uno de los apartados anteriores se ha visto el boceto general de la aplicación, y después se han empezado a diseñar los conceptos específicos únicamente de los contenidos. Pero si prestamos atención hay una parte que ha quedado sin mencionar, y son precisamente todos aquellos apartados que no son propiamente el contenido. Entre ellos podemos mencionar la zona de login, la cabecera o el pie de página.

Zona login

Esta zona tiene varias finalidades dependiendo del momento en el que nos encontremos. En un primer término sirve de puerta de entrada al sistema para los usuarios anónimos, ya que es el lugar dónde se debe introducir el nombre de usuario y contraseña. Aquí podemos ver un ejemplo:

Una vez el usuario ya se haya logueado, se puede dar el caso de que el usuario sea administrador, o sea un cliente común. Ambos casos están diferenciados tanto en la parte gráfica, como en la gráfica. A continuación vamos a ver el usuario administrador:

Vamos a describir todos los elementos de izquierda a derecha. En primer lugar veremos un icono con un engranaje, en realidad es un enlace hacia la página de configuración. A continuación aparece un mensaje de bienvenida personalizado con el nombre del usuario que ha iniciado sesión. Por último aparece un enlace para poder desconectarse de la sesión, en cuyo caso volveremos al punto de inicio.

Ahora vamos con el caso del usuario común:

En este caso únicamente se van a comentar las diferencias con la interfaz que aparece en el mensaje del administrador. Y básicamente la diferencia sólo es una, concretamente el icono del engranaje ha cambiado. También ha cambiado el enlace al que apuntaba, que ha pasado a apuntar a un apartado en el que el usuario puede ver y modificar información propia.

Zona cabecera

Esta zona es bastante sencilla y comprende la parte superior de la web, conocida como "cabecera". Una de las motivaciones de esta zona es ofrecer al usuario una coherencia entre las distintas páginas que componen el sitio, para que sepa en todo momento que no ha abandonado nuestra web. Por otro lado, refuerza la marca de nuestro negocio incluyendo en todo momento el logotipo de nuestra empresa en la parte izquierda.

El resto de la cabecera permanecerá sin contenido a excepción de la parte derecha, donde aparecerá la zona de login mencionada en el apartado anterior.

Zona pie pagina

Esta zona está reservada para los temas legales relacionados con el propio sitio web, en nuestro caso se da información de quien es el responsable del sitio, adjuntando un enlace hacia su sitio web personal. También sirve como promoción para quien ha diseñado y programado el sitio.

Este apartado también esta relacionado con el posicionamiento para buscadores, ya que intercambiando enlaces entre el propio sitio y el sitio web del desarrollador se pueden mejorar los resultados de las búsquedas en los distintos buscadores.

Zona camino navegacional

La zona del camino navegacional depende totalmente de la página concreta del sitio en la que se encuentre el usuario. Esto es así ya que muestra el lugar donde se encuentra actualmente el usuario, precedido del camino que se ha tomado para llegar a la lugar actual. Todos los caminos que preceden a la situación actual son enlaces que llevan a sus respectivos lugares.

La motivación de esta zona es que el usuario tenga una idea clara de donde se encuentra en el momento actual, pero además que recuerde los pasos que ha tomado desde el punto de partida. Esto ayuda a volver a puntos anteriores en caso de que desee volver sobre sus pasos.

Zona menú navegacional

Este apartado va a ser el más importante en cuanto al tema de navegación de refiere, ya que va a permitir al usuario moverse por los distintos apartados del sitio web. Es importante que esta zona esté visible en todo momento, en una zona muy visible para que el usuario pueda verlo en caso de que necesite navegar hacia otro apartado la web.

Deberá tener un estilo visual que permita diferenciarlo de las zonas inmediatamente colindantes, con el objetivo de que destaque sobre el resto del contenido, y sea visible rápidamente por el usuario.

6.4.3.- Capa negocio

En esta capa se desarrollan todos aquellos métodos que sirven para dotar de funcionalidad a nuestro sitio, como pueden ser las funciones de login, logout, modificación de datos personales… La otra funcionalidad de esta capa es proporcionar los datos necesarios que requiera la capa superior, este es el ejemplo de listar actividades, listar instalaciones, etc.

Login de usuario

Entrada	Nombre de Usuario, Password
Proceso	• Comprobar que los campos no estén vacío (de lo contrario dar un error) • Comprobar que el usuario realmente existe • Comprobar que la cuenta no esté deshabilitada • Comprobar que realmente sea esa su password
Salida	Login correcto, usuario no existe, contraseña incorrecta, falta algún dato

Logout del usuario

Entrada	Ninguna
Proceso	• Finalizar la sesión en el servidor
Salida	Ok, Error

Listar actividades

Entrada	Ninguna
Proceso	• Obtener todas las actividades de la Base de Datos
Salida	Lista de actividades, Error

Modificar actividad

Entrada	objeto Actividad
Proceso	• Extraer el id del objeto Actividad • Modificar dicho registro en la base de datos

Salida	Ok, Error

Crear nueva actividad

Entrada	objeto Actividad
Proceso	• Guardar dicho registro en la base de datos
Salida	Ok, Error

Listar horarios de actividad

Entrada	Id de actividad
Proceso	• Obtener todas las horarios de la Base de Datos de esa actividad
Salida	Lista de actividades, Error

Modificar horario

Entrada	objeto Horario
Proceso	• Extraer el id del objeto Horario • Modificar dicho registro en la base de datos
Salida	Ok, Error

Crear nuevo horario

Entrada	objeto Horario
Proceso	• Guardar dicho registro en la base de datos
Salida	Ok, Error

Listar usuarios

Entrada	Ninguna
Proceso	• Obtener todos los usuarios de la Base de Datos

Salida	Lista de usuarios, Error

Modificar usuario

Entrada	objeto Usuario
Proceso	• Extraer el id del objeto Usuario • Modificar dicho registro en la base de datos
Salida	Ok, Error

Nuevo usuario

Entrada	objeto Usuario
Proceso	• Guardar dicho registro en la base de datos
Salida	Ok, Error

Modificar contraseña

Entrada	idUsuario, contraseña antigua, contraseña nueva
Proceso	• Se extrae el registro de dicho usuario en la base de datos • Se crea un objeto Usuario con dichos datos • Se modifica la contraseña de ese objeto • Se guarda dicho objeto en la Base de Datos
Salida	Ok, Error

Listar profesores

Entrada	Ninguna
Proceso	• Obtener todos los profesores de la Base de Datos
Salida	Lista de profesores, Error

Modificar profesor

Entrada	objeto Profesor
Proceso	• Extraer el id del objeto Profesor • Modificar dicho registro en la base de datos
Salida	Ok, Error

Nuevo profesor

Entrada	objeto Profesor
Proceso	• Guardar dicho registro en la base de datos
Salida	Ok, Error

Listar noticias

Entrada	Ninguna
Proceso	• Obtener todas las noticias de la Base de Datos
Salida	Lista de noticias, Error

Modificar noticia

Entrada	objeto Noticia
Proceso	• Extraer el id del objeto Noticia • Modificar dicho registro en la base de datos
Salida	Ok, Error

Nueva noticia

Entrada	objeto Noticia
Proceso	• Guardar dicho registro en la base de datos
Salida	Ok, Error

Modificar instalación

Entrada	objeto Instalacion
Proceso	• Extraer el id del objeto Instalacion • Modificar dicho registro en la base de datos
Salida	Ok, Error

Geolocalización

Entrada	Ninguna
Proceso	• Comprobar si el navegador soporta esta característica • Extraer la latitud • Extraer la longitud
Salida	Latitud y longitud donde se encuentra el usuario

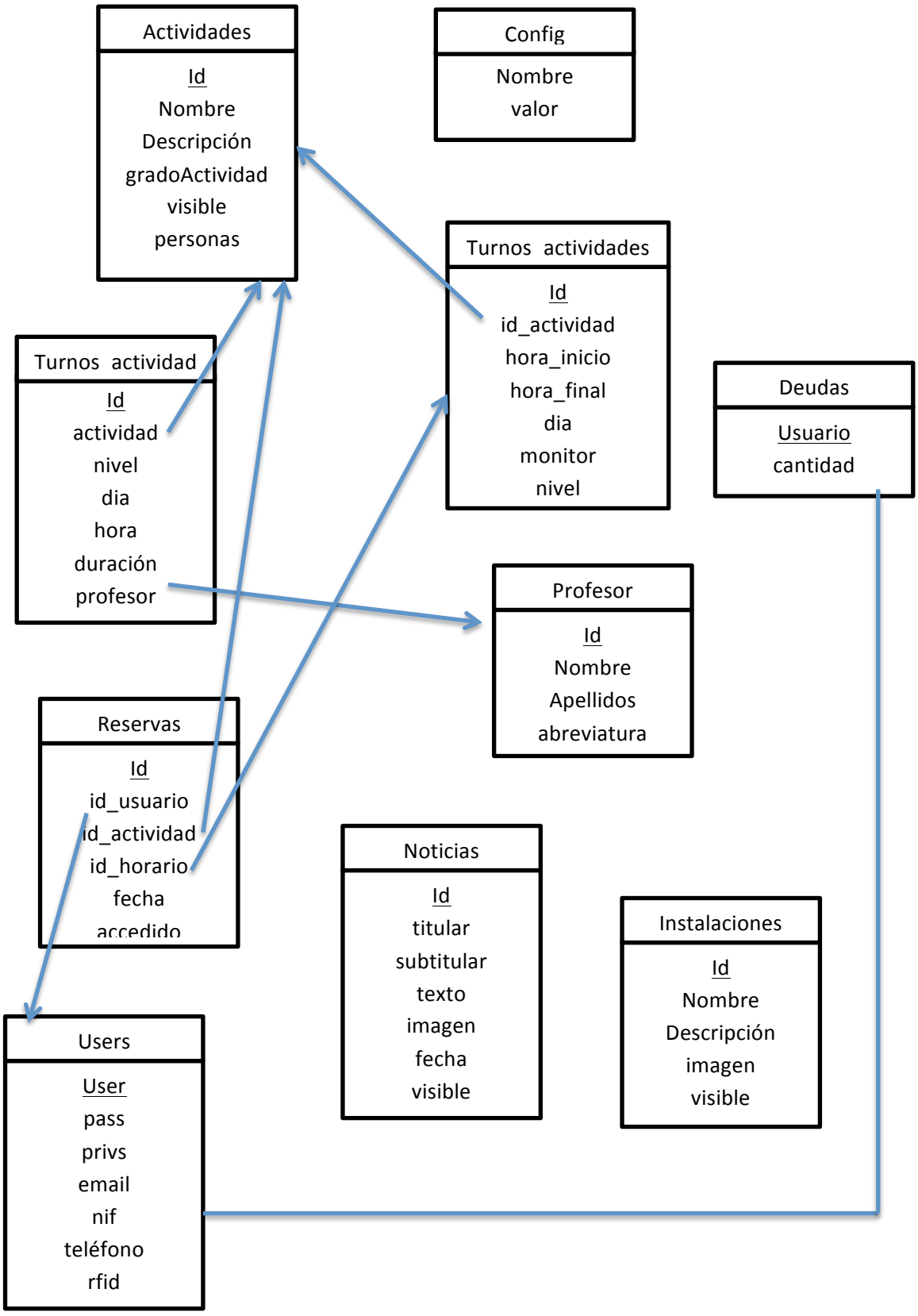
Navegación hasta el gimnasio

Entrada	Posición del usuario (latitud y longitud), y posición del gimnasio (latitud y longitud)
Proceso	• Cargar un mapa con la API de Google Maps • Cargar la posición del usuario como origen de la ruta • Cargar la posición del gimnasio como destino de la ruta • Calcular la ruta desde el origen al destino
Salida	Latitud y longitud donde se encuentra el usuario

6.4.4.- Capa persistencia

Una vez llegados a este punto, ya deberíamos tener claro qué sistema gestor de base de datos vamos a utilizar en nuestro proyecto, ya que el desarrollo de esta capa está íntimamente relacionado con él. Esto es así puesto que cada SGBD que utilicemos nos proporcionará una interfaz de programación (API) distinta a otro SGBD. Dicha API nos proporcionará diversos métodos que utilizaremos en esta capa para operaciones de consulta, inserción, modificado o borrado de las distintas tablas.

En la siguiente ilustración se puede apreciar el esquema Entidad-Relación de la BBDD:


Actividades
Id
Nombre
Descripción
gradoActividad
visible
personas
Config
Nombre
valor
Turnos actividades
Id
id_actividad
hora_inicio
hora_final
dia
monitor
nivel
Turnos actividad
Id
actividad
nivel
dia
hora
duración
profesor
Deudas
Usuario
cantidad
Profesor
Id
Nombre
Apellidos
abreviatura
Reservas
Id
id_usuario
id_actividad
id_horario
fecha
accedido
Noticias
Id
titular
subtitular
texto
imagen
fecha
visible
Instalaciones
Id
Nombre
Descripción
imagen
visible
Users
User
pass
privs
email
nif
teléfono
rfid

6.4.5.- Patrones de diseño

La propia definición de patrón nos puede dar una idea de lo que vamos a tratar en este apartado. Un patrón no es más que una serie de características que se repiten a lo largo del tiempo. En este caso concreto esas características de las que hablamos son problemas que se tienen a la hora de desarrollar una aplicación software (concretamente en ámbito de la web). La idea es presentar soluciones comunes para solventar todos estos problemas.

6.4.5.1.-Navegación

Intención

Nuestra página puede contener multitud de enlaces a otras partes de nuestro sitio web, y puede ser interesante resaltar aquellos que poseen más importancia para los usuarios.

Solución

En primer lugar se identificarán aquellos enlaces importantes que deseemos destacar como principales en nuestro sitio. Éstos deberán ofrecer una información contextual suficientemente orientativa para que los usuarios conozcan su utilidad de un único vistazo.

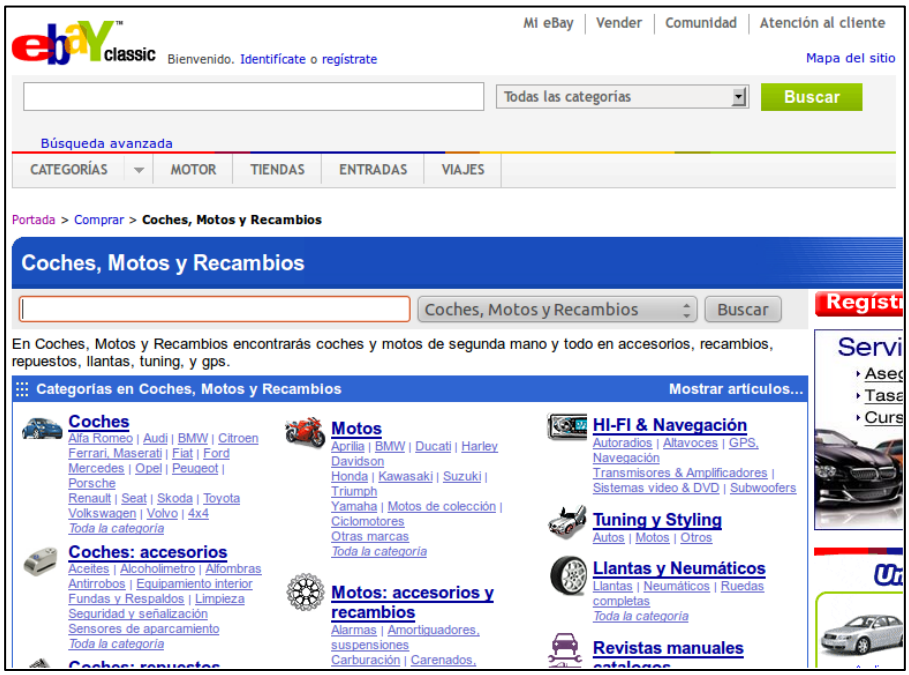
Deberemos ofrecer un patrón visual común a todos los enlaces, de forma que se dé a entender que su jerarquía es la misma, y a su vez que es mayor que la del resto de enlaces de la web.

Ejemplo

En la web de ebay.es se puede ver como existe un menú navegacional compuesto por las categorías más importantes (en este caso motor, tiendas, entradas y viajes). Suponemos que la compañía controla las estadísticas de ventas del portal para decidir qué categorías son más importantes.

Se puede ver como aparentemente todas pertenecen a una misma jerarquía, y aparecen resaltadas en un cuadro gris que hace resaltar a los enlaces sobre el resto del contenido.

Otro detalle importante es que para no sobrecargar la barra de categorías se ha optado por ofrecer un desplegable que muestra en un tamaño menor el resto de categorías.



6.4.5.2.-Pie de página

Intención

Ofrecer enlaces con la información institucional de la empresa, ayuda para los usuarios e información de contacto.

En el caso de poseer información legal importante para los usuarios, también sería conveniente añadirla, como por ejemplo copyright, marca registrada, número de expediente, información sobre protección de datos…

Solución

Esta solución es común prácticamente a todos los portales en la actualidad, por lo que los usuarios por simple inercia lo buscarán en este lugar. Por lo tanto si queremos que nuestra web sea lo más usable posible, deberemos colocarlo en el mismo lugar.

Estos enlaces carecen del mismo nivel de importancia de los del menú navegacional, por lo que su tamaño deberá ser menor. También tendremos que tener en cuenta que estos enlaces rara vez son usados por el usuario, por lo que parece lógico que su lugar no deba estar mezclado con otros contenidos de mayor relevancia.

Ejemplo

En la web de la cadena Fnac, podemos ver un ejemplo muy completo de lo que se ha comentado en los puntos anteriores.

Podemos observar un enlace para obtener ayuda, otro donde consultar las políticas de empresa, las condiciones de contratación, una página de contacto, información legal de la empresa, ofertas de trabajo, mapa de la web, y un largo etcétera.

En realidad este ejemplo me parece demasiado sobrecargado por contener gran cantidad de enlaces, que hace que se tarde mucho tiempo en encontrar el que necesitamos. Y por otra parte al ser la web de fondo blanco no se puede apreciar una diferencia visual con el resto del sitio.

6.4.5.3.-Herramientas de Login

Intención

Se pretende ofrecer al usuario una interfaz que muestre de forma clara el lugar en el que debe introducir su nombre de usuario y contraseña.

El cometido de solicitar dichos datos es que el usuario pueda acceder a información personal y a acciones que sólo puede realizar una persona registrada.

Solución

Para esta solución se ha optado por ofrecer dicha información en el mismo lugar donde se ofrece la información del carrito y del propio usuario.

Este apartado variará levemente en ambos casos, mostrando alternativamente la opción de loguear, o la de acceder a información de la cuenta.

El lugar en el que se va a colocar dicha información es la parte superior derecha de la pantalla, intentando que esté accesible al entrar a la web, y que el usuario pueda verlo cuanto antes.

Ejemplo

En la web dropbox.com, podemos observar como siguiendo con lo que es prácticamente un estándar, ubica dicha opción en la zona superior derecha de la página.

En este caso se trata de la web de inicio y se encuentra muy poco cargada de elementos visuales, pero aún así han optado por incluir un único enlace con el texto "Iniciar Sesión" con un logotipo muy claro y una flecha hacia abajo, haciendo que los campos a rellenar sean visibles únicamente cuando se haga click en este texto.

6.4.5.4.-Acordeones

Intención

Se pretende mostrar al usuario en una única página una cantidad grande de información. En vez de mostrar toda la información, se pretende organizarla visualmente de forma que pueda seleccionar cuál quiere visualizar, sin necesidad de sobrecargar la visualización.

Solución

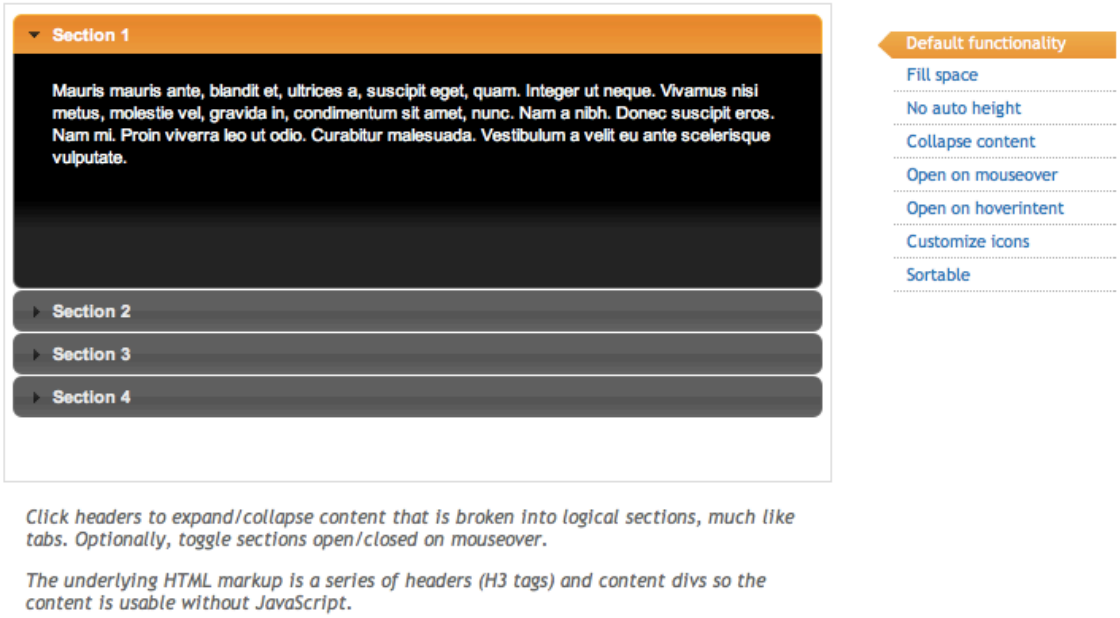
Para conseguir este efecto se puede utilizar el efecto "acordeón" que da un aspecto visual similar al del instrumento musical del que toma el nombre.

El funcionamiento es muy sencillo, si partimos de una serie de secciones o categorías, mostraremos un enlace capara cada una de ellas. Dichos enlaces forman en realidad una pila de cabeceras delimitadas visualmente de alguna forma.

Al hacer click en una de estas cabeceras, lo que haremos será crear un nuevo espacio visual después de dicha cabecera, e inmediatamente antes de la cabecera siguiente. En dicho espacio se mostrará la información que creamos conveniente.

Ejemplo

En la web de ejemplo de la librería JqueryUI (http://jqueryui.com), que es quien implementa esta función, podemos ver una muestra de la funcionalidad que deseamos implementar en nuestro sitio. La sección seleccionada se puede apreciar en la imagen en un color naranja, y el contenido inmediatamente a continuación. El resto de secciones aparecen sin desplegar en color gris en la parte inferior.



6.4.5.5.-Inserción de datos

Intención

Cuando solicitamos datos al usuario, éste puede que no siempre introduzca los datos en el formato que nosotros deseemos, por lo que tendremos que validar los datos y hacérselo saber al usuario.

Solución

Para comprobar los datos se va a hacer una doble validación, una en el lado del cliente y otra en el lado del servidor. En este caso nos fijaremos en el lado del servidor, comprobando uno a uno todos los campos que el cliente debería introducir.

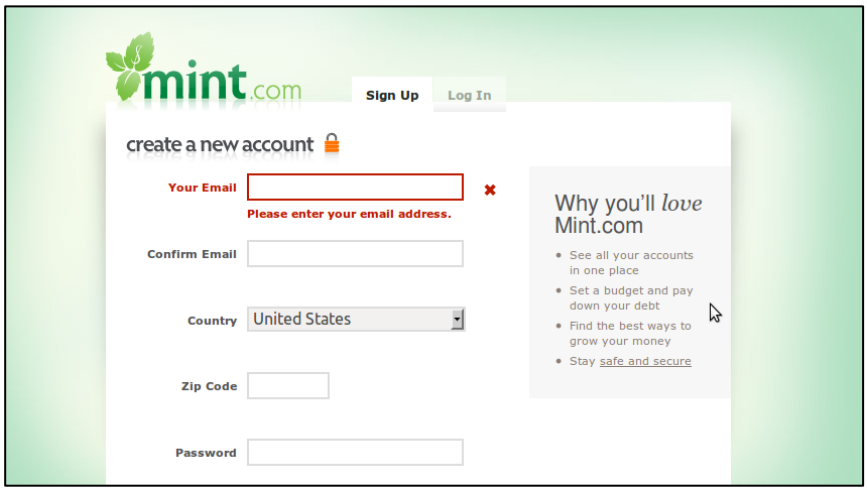
Una vez se haya comprobado, deberemos mostrarle al usuario cómo ha ido este proceso de validación para cada campo requerido. Esto es especialmente útil para casos en los que se deban insertar múltiples campos, y el usuario quiera saber en qué campo hay un error para no tener que revisarlos uno a uno.

Ejemplo

En la web de Mint, se ofrece al usuario un formulario donde deberá introducir sus datos para poder crear una nueva cuenta de usuario.

Para probar el funcionamiento, dejaremos sin escribir el campo "Your e-mail" y veremos que nos muestra un error de debemos introducir una dirección de correo electrónico.

Si por el contrario escribimos algo que no está en el formato adecuado, como puede ser una email sin el símbolo @, nos mostrará otro error diferente indicando que el email no es válido.



6.4.5.6.-Páginas de error

Intención

Es posible que en el transcurso de la navegación por el sitio, el usuario llegue a una página que no exista. Puede que sea una página antigua, o que el usuario no la recuerde bien al teclear la URL de forma manual. Se pretende que además del error, se proporcione información de utilidad sobre el sitio para ayudar al usuario a encontrar aquello que buscaba.

Solución

Ofrecer páginas de error personalizadas con un mapa que permita saber la estructura de nuestro sitio, un buscador por si el usuario sigue sin intuir dónde se encuentra el recurso que estaba solicitando. Adicionalmente se podría capturar la URL introducida para ofrecer sugerencias al usuario del recurso que buscaba.

Ejemplo

6.4.5.7.- Camino navegacional

Intención

En el transcurso de la navegación por el sitio que vamos a diseñar, debemos ofrecer al usuario una referencia del lugar donde se encuentra, y de los lugares a través de los que ha llegado. Esto ayuda a ofrecer una mejor experiencia, permitiendo situarse mejor al usuario, y referenciando los lugares por los que ha pasado.

Solución

En cada apartado puede aparecer el camino que se ha seguido antes de llegar a ese elemento. Este camino aparecerá en forma de enlace para permitir al usuario volver sobre sus pasos en cualquier momento, y diferenciando visualmente de algún modo (negrita, resaltado, coloreado…) el lugar en el que se encuentra en la actualidad. En esta última referencia no incluiría ningún enlace, ya que se trataría de la misma página.

Ejemplo

En la web española del portal eBay podemos encontrar un ejemplo perfecto de camino navegacional. Este se utiliza en el momento de ir navegando por las distintas categorías para encontrar los artículos por parte del usuario. En todo momento el usuario puede navegar por las categorías inmediatamente superiores.

6.4.5.8.- Pestañas

Intención

En nuestro sitio puede ser interesante separar distintos tipo de contenido mediante un menú navegacional, en el que se permita que los usuario puedan acceder a los recursos más importantes. Sería interesante sugerir gráficamente que existe un elemento físico que se encarga de separarlos como en la realidad podría hacerlo un archivador de carpetas.

Solución

Mediante la implementación de pestañas ayudamos al usuario a crear la idea de que son contenidos distintos y que por lo tanto es conveniente que dispongan de una cierta separación física. Además se implementará en distintos colores, de forma que la pestaña en la que nos encontremos ofrezca una distinción visual que permita distinguir nuestra ubicación actual.

Ejemplo

6.5.- Implementación

6.5.1.- Tecnologías interfaz

Para la interfaz se va a usar un documento HTML carente de todo tipo de estilos. La idea es que dicho documento defina únicamente las estructuras que permitan identificar los distintos bloques y contenidos de la página. Llegados a este punto, desarrollaremos un fichero de estilos que será un documento CSS, donde iremos asignando estilos a cada una de las estructuras identificadas en el documento HTML.

```
<html>
<body>
<div id=”#cont”>Ejemplo</div>
<div id=”#cont2”>
   <img src=”portada.jpg”>
</div>
</body>
</html>
```

Ejemplo HTML

+

```
#cont {
    margin-top: 10px;
    background-color: #FF1245
}
#cont2 img {
     width: 100px;
     height: 100px;
}
```

Ejemplo CSS

Como se puede ver en el ejemplo anterior el documento HTML declara dos contenedores “div” con identificadores “cont1” y “cont2”. En ningún momento se hace mención a colores, tamaños, márgenes, etc. En el ejemplo CSS, se puede ver como aparecen las definiciones anteriores con una serie de estilos definidos en su interior.

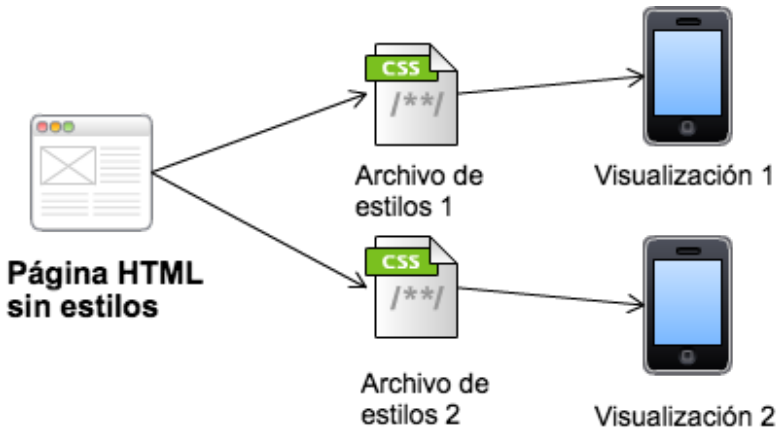
Para hacer que el archivo CSS se asocie a nuestro documento HTML tendremos que especificarlo en dicho documento mediante la siguiente línea en el apartado “<head>”:

```
<link href="ficheroCSS.css" rel="stylesheet" type="text/css">
```

Este método de definir estilos a nuestros sitios web es el recomendado por el W3C a partir de la versión HTML 4 para el mejor cumplimiento de los estándares. Las etiquetas HTML que aparecían en versiones anteriores del lenguaje para definir los distintos estilos de cada elemento, están obsoletas.

Otra de las ventajas que proporciona esta tecnología es la posibilidad de intercambiar distintas visualizaciones de una misma página, con el simple hecho de intercalar un archivos de estilos CSS u otro. Supongamos que deseamos que nuestra página sea vista en distintos dispositivos móviles, o en navegadores que no cumplen los estándares al 100%. Otra utilización sería para el cambio del estilo de la página en momentos especiales como podrían ser navidades, aniversarios, rebajas…

A continuación se puede ver un esquema del funcionamiento de esta tecnología, intercalando distintos archivos de estilos CSS sobre un mismo documento HTML.



6.5.2.- Tecnologías aplicación

Para el desarrollo de esta parte de la aplicación existe una limitación importante que tiene que ver con el lugar donde se ejecuta el código de la aplicación. Concretamente tenemos un tipo de código que se encarga de la interacción con el navegador en un primer nivel, cuyo código será ejecutado en el propio navegador. El lenguaje elegido para que se ejecute de forma nativa en el navegador (sin necesidad de añadidos) se denomina JavaScript. En un segundo término tenemos otro tipo de código que es ejecutado en el propio servidor , y que será el encargado de devolver al cliente los documentos dinámicos generados a partir de los parámetros recabados. El lenguaje utilizado es PHP.

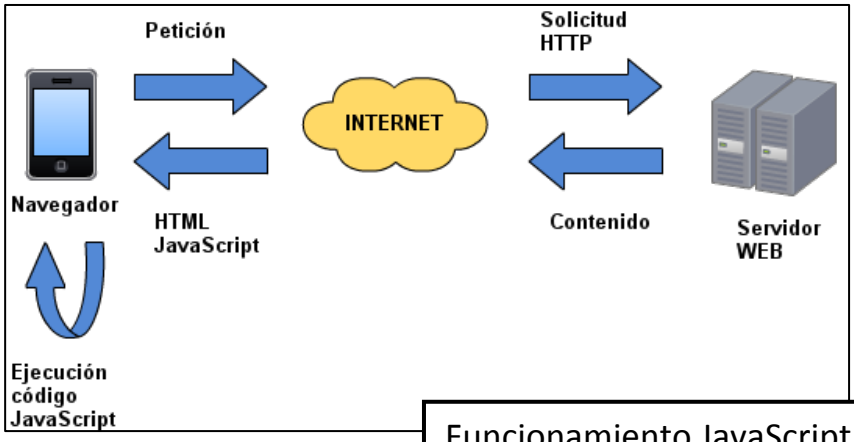
Para desarrollar mejor ambas partes, se ha desarrollado la explicación en dos apartados diferentes:

6.5.2.1- Lado del cliente

Este apartado implica siempre una dificultad adicional, ya que no podemos controlar la configuración del navegador del cliente, por lo que nuestra aplicación siempre puede tener ciertas incompatibilidades que no somos capaces de controlar.

El navegador siempre ha sido el instrumento encargado de mostrar el contenido de la navegación, bien siendo texto o contenido multimedia, y el encargado de realizar la interacción de la navegación mediante los enlaces. Por tanto las funciones que podía realizar el navegador eran un tanto limitadas.

Esto se intentó solventar intentando incluir alguna forma de que poder ejecutar código dentro del propio navegador. Estos son generalmente lenguajes de script que forman parte del contenido de la página y son capaces de interactuar con los distintos elementos que la forman.



Funcionamiento JavaScript

A lo largo de la historia de Internet, las tecnologías del lado del cliente han ido evolucionando con soluciones como JavaScript, VBScipt, ActiveX, Applets Java y elementos Flash. Los mayores problemas que han surgido siempre han sido las incompatibilidades entre los distintos navegadores. Por un lado unos estaban implementados únicamente en ciertos navegadores (VBScript y ActiveX para Internet Explorer), y otros por la necesidad de tener instalados ciertos plugins para la correcta ejecución del código (Applets Java y elementos Flash).

Actualmente podemos decir que la tecnología más compatible y que mayor auge está teniendo en los últimos tiempos es JavaScript. Hay que decir que aunque el lenguaje existe desde hace muchos años, es ahora cuando están surgiendo nuevas aplicaciones gracias a la jerarquía de objetos del navegador (DOM), a la lucha actual de todos los navegadores por la compatibilidad con los estándares del W3C, y al desarrollo de motores JavaScript capaces de ejecutar códigos más complejos en menos tiempo.

Por tanto parece obvio que para este proyecto utilizaremos el lenguaje JavaScript, y la jerarquía de objetos del navegador (DOM). Todo ello respetando los estándares fijados por el W3C, que es la encargada de redactar las especificaciones para las distintas tecnologías utilizadas en el ámbito de la Web.

Entre las utilidades de este lenguaje se encuentra la de posibilidad de interacción con cualquier elemento de las páginas. Ejemplos de ello puede ser la consulta de los datos introducidos por el cliente en los formularios, ejecución de acciones asociadas a eventos como clicks, modificación de la visualización de elementos, etc.

Otro aspecto a tener en cuenta en el uso del código en el lado del cliente es el uso de los distintos frameworks, que no son más que librerías a modo de APIs que podemos utilizar para programar nuestras aplicaciones. Usar este tipo de desarrollo tiene múltiples ventajas como la mayor facilidad de escribir código, una mayor compatibilidad, comunicación mediante AJAX, validación de formularios y manejo de eventos. Los frameworks más importantes son JQuery, Dojo Toolkit, Prototype y Mootools.

A continuación podemos ver un ejemplo de un script escrito en JavaScript implementado dentro de un documento HTML mediante las etiquetas <script></script>:

```
<script type="text/javascript">
        var mensaje = "¡Hola mundo!";
        document.write (mensaje);
</script>
```

Ejemplo JavaScript

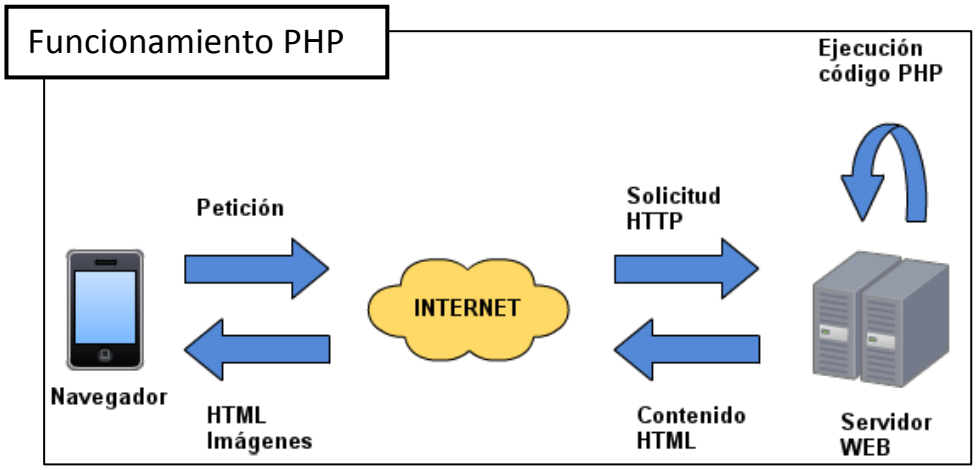
6.5.2.2- Lado del servidor

En el caso del código que se ejecute en el lado del servidor no disponemos de los problemas que se comentaban en el apartado anterior, ya que todo el código se ejecuta en el lado del servidor, por lo que seremos nosotros los encargados de configurar el entorno en el que se va a ejecutar el código.

La cantidad de tecnologías que podemos utilizar para la programación de este código es prácticamente infinita y ahora veremos por qué. Por un lado hay tecnologías específicas que están diseñadas para la ejecución de código en un servidor web, los ejemplos más conocidos de estas tecnologías son PHP, ASP.NET, JSP y Servlets de Java. Estas son las tecnologías más recomendadas a la hora de implementar el código. Debemos tener en cuenta que nuestro servidor debe ser compatible con el tipo de tecnología empleada.

Por otro lado está el uso de lenguajes genéricos mediante el uso de la tecnología CGI. Esto se trata de el navegador es capaz de redireccionar la entrada a un programa estándar, y la salida del programa al navegador para que el cliente pueda visualizarlo. Estos programas son los que pueden estar escritos en cualquier lenguaje, siempre y cuando dispongamos de el programa compilado o un intérprete que sea capaz de entenderlo. Por lo tanto podremos utilizar lenguajes tan variopintos como Perl, C o Bash.

Para nuestro proyecto vamos a utilizar PHP funcionando mediante un módulo del servidor Web Apache. En el apartado del Servidor Web están explicados todos los pasos necesarios para la creación del entorno que permita ejecutar nuestro código. Ahora vamos a pasar a ver un poco en qué consiste este lenguaje:



En primer lugar es un lenguaje de script interpretado. Esto quiere decir que el código no hay que compilarlo, si no que se interpreta en tiempo de ejecución. Además el código puede ser incrustado en el propio documento HTML, por lo que su uso será menos traumático si ya

sabemos programar documentos estáticos. Su inclusión se realiza mediante las etiquetas específicas (<?php y ?>) que delimitan el código PHP del resto del HTML.

El siguiente ejemplo muestra un documento HTML 5, en el cual podemos ver las etiquetas que delimitan el código PHP. Este código simplemente muestra el mensaje 'Esto es código PHP' en el navegador:

```
<!DOCTYPE html>
<html lang="es">
<head> <title>Gestion Gimnasio</title> </head>
<body>
        <?php
                echo 'Esto es codigo PHP';
        ?>
</body>
</html>
```

Ejemplo código PHP

6.5.3.- Tecnologías contenidos

En este apartado se tienen en cuenta todas aquellas tecnologías que sirven para ofrecer cualquier tipo de información en nuestro sitio Web. Aquí se incluye gran cantidad de información de los cuales algunos pueden ser estáticos, y otros generados de forma dinámica.

Los contenidos generados de forma estática suelen estar incluidos en el propio código HTML ya que este no va a variar en el tiempo. Si por el contrario son generados de forma dinámica, deberán ser creados mediante algún tipo de tecnología de programación (bien en el lado del cliente o bien en el lado del servidor).

Otro tipo de clasificación que puede establecer es mediante el tipo de contenido que se ofrezca: por un lado existe el contenido de tipo texto (entre el que se pueden encontrar resúmenes, descripciones, títulos, explicaciones, noticias, etc…), también tenemos contenidos en forma de imágenes (como pueden ser iconos, logotipos, ilustraciones, etc…), vídeos en formato streaming (como explicaciones de los distintos ejercicios)

Es importante que el servidor conozca el tipo contenido que vayamos a servir. Para ello ya se ha hablado de los tipos MIME, los cuales tienen que ser devueltos al cliente para que el navegador sea capaz de entenderlo. A continuación se muestran algunos de los tipos MIME más importantes:

Tipo MIME	**Extensión**
text/html	html htm

image/jpeg	jpeg jpg
image/gif	gif
application/xls	xls
application/msword	doc
application/pdf	pdf
application/rtf	rtf
image/png	png
application/x-javascript	js
application/x-shockwave-flash	swf
application/x-troff-msvideo	avi
image/x-icon	ico
audio/mpeg3	mp3
image/BMP	bmp

6.6.- Seguridad

6.6.1.- OWASP TOP 2010

"The OWASP Fundation" es una asociación sin ánimo de lucro de carácter mundial cuyo objetivo es ofrecer de manera libre y abierta una serie de herramientas, documentos, foros y otro material didáctico acerca del desarrollo y mantenimiento de aplicaciones web seguras.

Dicha fundación dispone de gran cantidad de información actualizada disponible en su sitio web (www.owasp.org) sobre los riesgos que se deben tener en cuenta sobre la seguridad de aplicaciones web. Tanto la documentación, como el software que se puede encontrar en su web son totalmente gratuitos y libres, bajo una licencia de software libre.

Con cierta frecuencia esta fundación publica una lista de los 10 mayores riesgos a los que están expuestas las aplicaciones web. Para cada uno de estos riesgos se hace un informe en el cual se explica cual es el grado de dificultad para ser llevado a cabo, el impacto que puede tener, la capacidad para ser detectado, las formas de evitarlo, ejemplos, etc.

Hay que decir que este listado llamado "OWASP Top 10 Application Security Risks" es actualmente el documento más importante en cuestión de seguridad de aplicaciones web, por lo que aquí se van a exponer los 10 riesgos comentados en la versión del documento que data de 2010.

6.6.1.1- Injection

Este tipo de ataque permite al usuario la ejecución de consultas o comandos en el servidor debido a que la aplicación ha recibido datos no validados correctamente del usuario.

La forma más común se presenta como un formulario que la aplicación muestra al usuario para solicitarle ciertos datos que éste debe rellenar. El usuario enviará en dicho formulario el ataque en forma de código modificado para que sea capaz de sortear las protecciones existentes (en caso de haberlas).

La aplicación no habrá detectado que el contenido del formulario es potencialmente peligroso para la aplicación, y la enviará al conector de la base de datos o al intérprete de órdenes correspondiente. La consulta o la orden se ejecutará en el servidor con resultados poco previsibles.

Los resultados de la consulta o de ejecución de la orden serán recibidos por la aplicación, quien a su vez los mostrará al usuario normalmente incrustados en la propia página.

Recomendaciones

Validar los datos de entrada usando listas blancas y expresiones regulares.

Seguir el principio de mínimo privilegio en las conexiones con bases de datos y otros componentes.

Evitar el uso de intérpretes siempre que sea posible. En caso de resultar necesario, utilizar APIs seguras.

6.6.1.2.- Cross Site Scripting

Esta vulnerabilidad también conocida como XSS, consiste en que un atacante envía scripts en un campo de formato texto con la finalidad de que estos sean ejecutados por el intérprete del navegador.

El campo de formato texto no tiene porque ser explícitamente un formulario o URL como en el ejemplo de las inyecciones vistas anteriormente, sino que puede estar almacenada en un foro, en el título de una noticia, en la base de datos, etc.

Este tipo de vulnerabilidades ocurren cuando la aplicación incluye en la respuesta los mismos datos que ha suministrado el usuario con anterioridad sin haber sido validados correctamente.

Aunque existen herramientas automatizadas que permiten detectar ciertos problemas XSS, debido al creciente número de tecnologías del lado del cliente, estas tareas son cada vez más complicadas de detectar de forma automatizada.

Recomendaciones

Los datos introducidos por el cliente deben ser validados, a ser posible mediante white-list.

Codificar explícitamente el formato de la salida que se devuelve al cliente.

6.6.1.3- Broken Authentication and Session Management

Este ataque permite obtener contraseñas de los clientes o identificadores de sesión, con el objetivo de suplantar la identidad de otro usuario. Los atacantes pueden ser usuarios anónimos que intentan acceder a la aplicación, usuarios con cuentas que intentan robar cuentas de los demás para obtener más privilegios, o aquellos que desean ocultar sus acciones.

Se realiza explotando de forma malintencionada las funciones de autentificación y gestión de la sesión, para obtener los datos que permitan al atacante conseguir los datos necesarios para obtener el acceso al sistema.

En este tipo de ataques suele ocurrir que ciertos datos importantes residen en el equipo del usuario, o el uso de la seguridad en la ruta por la que viajan los datos es realmente pobre. En general la gestión de la autentificación y el control de errores es realmente complejo por la cantidad de tareas que cada aplicación realiza como pueden ser la autentificación, los tiempos máximos, las cookies, la recuperación de la contraseña…

Recomendaciones

Las credenciales deben viajar por canales seguros (SSL) y almacenados de forma cifrada.

No aceptar nuevos identificadores de sesión por parte del usuario.

Implementar correctamente la función de logout en cada página.

Utilizar time-out por inactividad que posibiliten la caducidad de la sesión.

No pasar dichos datos en la URL, ya que pueden quedar registrados en el historial o proxys.

Comprobar el anterior password cuando el usuario desee cambiarlo por uno nuevo.

6.6.1.4.- Insecure Direct Object Reference

Este ataque permite que los usuarios puedan acceder a ciertos datos falseando una referencia a un objeto al cuales no debería permitírsele el acceso.

Se trata simplemente de cambiar el valor del parámetro que apuntaba a un objeto, para hacer que tras la modificación apunte a otro objeto totalmente distinto al cual no está autorizado a acceder.

Por poner un ejemplo, supongamos una dirección web como la siguiente: http://www.ejemplo.com/verusuario.php?id=203, la cual permitiría acceder a la información de el usuario cuya id sea 203. Un usuario malintencionado podría modificar dicho parámetro y sustituirlo por el id de otro usuario, para de esa manera acceder a la información de otro usuario, al cual no debería tener privilegios suficientes.

Recomendaciones

Utilizar referencias indirectas, por ejemplo verfichero.php?file=1.

Validar cualquier referencia a objetos que vengan del exterior.

Verificar la autorización al recibir una referencia a un objeto.

6.6.1.5- Cross Site Request Forgery

Este tipo de vulnerabilidad trata de dar por válida una petición a nuestro sitio que provenga de un sitio web diferente, y que tiene por objetivo engañar a los usuarios para poder realizar diversas operaciones.

En este supuesto se aprovecha una autentificación implícita por parte de los usuarios, está bien puede ser por autentificación HTTP, cookies, autentificación SSL ente otros. Dado que estas peticiones suelen ser ocultas en etiquetas IMG o JavaScript, es difícil de ser detectado por parte de los usuarios.

La gran mayoría de estos ataques suelen realizarse debido a que las aplicaciones realizan el traspaso de informaciones mediante GET, lo cual se traduce en la URL que es fácilmente incrustable en otro tipo de páginas. Aún así no es este el único caso en el que se puede dar este ataque, por lo que se debe tener especial cuidado también en métodos POST.

Recomendaciones

Utilizar testigos aleatorios que sean enviados para validar la operación.

Verificar que la aplicación no es vulnerable a vulnerabilidades XSS.

En operaciones realmente sensibles solicitar re-autentificación del usuario.

Aceptar únicamente POST para transmitir información sensible.

6.6.1.6- Security Misconfiguration

El objeto de este ataque es explotar vulnerabilidades (normalmente en servidores web) provocadas por configuraciones de seguridad bajas o por defecto.

Por lo general el acceso que se tiene es a cuentas de usuario por defecto que se encuentran en sistemas operativos, o servidores de todo tipo, aunque también es posible que se consigan accesos a ficheros y directorios no protegidos.

El usuario no puede ver de un simple vistazo si la aplicación a la que está accediendo es segura o no, ya que para ellos sería interesante confiar sus datos personales a aplicaciones realmente seguras.

Existen herramientas de detección de vulnerabilidades y scripts automatizados que permiten realizar pruebas sobre un servidor web, comprobando una serie de variables en función de la versión del servidor que se esté ejecutando.

Recomendaciones

Usar guías de configuración sobre el servidor que utilicemos en cada caso.

Mantener actualizadas todas las versiones de todos los productos.

Analizar los cambios que transcurren en cada cambio de versión, y cómo pueden afectar.

6.6.1.7.- Failed to Restrict URL Access

Esta vulnerabilidad se trata de una falta de controles de autenticación o autorización cuando se pretende acceder a ciertos recursos que no deben ser accesibles a todos los usuarios.

Se suele dar por sentado en numerosas aplicaciones que los usuarios no conocen ciertas URLs, pero están pueden averiguarse por ingeniería inversa o consultando URL de otros usuarios.

Por ejemplo, en una URL http://www.ejemplo.com/usuario1/verDatos.xml, podemos intuir que ese fichero estará disponible para todos los usuarios, por lo que podríamos intentar solicitar la siguiente URL http://www.ejemplo.com/administrador/verDatos.xml.

Recomendaciones

Analizar los roles que ejecutará cada función en las fases iniciales de desarrollo.

Se debe controlar los permisos de cada recurso (a través del servidor o de la propia aplicación)

Llevar a cabo tests de intrusión antes de poner la aplicación en un entorno de producción.

Rechazar servir aquellos ficheros cuya extensión no consideramos segura (.sql, *.bak, *.old…)

6.6.1.8.- Unvalidated Redirects and Forwards

En este tipo de situaciones la aplicación dispone de algún tipo de mecanismo que permite ejecutar una redirección hacia otro sitio web, creando un engaño al usuario haciéndole creer que va a acceder a un dominio cuando en realidad se va a redirigir a otro dominio.

Los problemas que puedan causar dependen sobre todo de las intenciones del sitio web de destino, y pueden ir desde la solicitud de credenciales de acceso hasta instalación de malware en el equipo del cliente.

Por ejemplo, supongamos que una aplicación cuyo cometido es redirigir a otra URL se encuentra en http://www.ejemplo.com/redirect.php?url=catalogo.php, un sitio web malicioso intentará modificar los parámetros para redirigir a su propio sitio, quedando por ejemplo de la siguiente manera: http://www.ejemplo.com/redirect.php?id=2&sesion=JhsdkU9y&visible=true&...&url=www.sitiowebmalicioso.com (en este ejemplo se han introducido parámetros de relleno para crear confusión y que el usuario no detecte la URL del sitio malicioso)

Recomendaciones

Evitar en lo posible el uso de redirecciones en nuestro sitio web.

En caso de no poder evitarlo, validar la URL recibida como parámetro.

6.6.1.9.- Insecure Cryptographic Storage

Esta vulnerabilidad se puede llevar a cabo aprovechando que la aplicación web no protege con el cifrado adecuado datos especialmente sensibles como credenciales de acceso o tarjetas de crédito.

Supongamos por ejemplo un caso en el que las credenciales de acceso se encuentran perfectamente cifradas en la base de datos del sitio, pero que cada vez que un usuario realiza un acceso correcto a la aplicación éste acceso se registra en un log del servidor quedando registrado su usuario y contraseña.

Cualquier usuario que consiguiera acceder a dicho fichero tendría todas las cuentas de la aplicación que hubieran realizado algún acceso.

Recomendaciones

Verificar que realmente el cifrado realizado es correcto y suficiente.

Usar algoritmos de criptografía públicos como por ejemplo AES, RSA o SHA256.

No utilizar algoritmos de los considerados débiles como MD5 o SHA1.

No transmitir credenciales de acceso por canales inseguros.

6.6.1.10- Insufficient Transport Layer Protection

Se trata de la captura de información importante que viaja por canales inseguros, en algunos sitios se trata del modo normal de funcionamiento, y en otros un modo para poder seguir funcionando en caso de errores o averías.

Para los usuarios es muy sencillo comprobar si los datos que introduzca van a viajar por un canal seguro o no, pues los navegadores ofrecen indicadores e iconos que indican si el tipo de conexión es normal o segura.

Recomendaciones

Utilizar canales seguros en la transmisión de datos con los clientes (normalmente SSL)

Utilizar canales seguros entre componentes (por ejemplo entre Servidor Web y Base de Datos)

Seguir las leyes aplicables en materia de protección de datos y comunicaciones.

6.6.2.- Validación de la entrada de datos

Uno de los mayores problemas que tenemos que tener en cuenta a la hora de asegurar nuestra aplicación es analizar todos los datos que el usuario manda a nuestra aplicación antes de procesarlos, con el fin de detectar posibles ataques anticipándonos a los mismos.

Las expresiones regulares son una buena forma de analizar estos datos de entrada permitiéndonos encontrar caracteres ilegales en estas cadenas, y sustituirlos en su caso por otros caracteres más apropiados. Esto suele deberse a que nosotros hemos construidos los distintos métodos de los que consta nuestra aplicación de una determinada manera, usando unos determinados caracteres como delimitadores, por lo que alguien podría intentar utilizar esos caracteres para intentar engañar a nuestra aplicación.

Imaginemos por ejemplo el típico método POST por el que los distintos parámetros se pasan de la siguiente forma: "nombre=hector&apellidos=martinez&localidad=requena", alguien podría introducir comandos y enviarlos al servidor. Este proceso podría realizarse sin pasar por nuestra aplicación web, simplemente enviándolos mediante un programa externo.

Por ello es tan importante no confiar en la validación que se realiza en el lado del cliente, ya que un posible atacante podría saltarse esta limitación y enviar los datos directamente al nuestra aplicación web.

6.6.3.- Ocultar ciertos contenidos a los buscadores

Los buscadores poseen robots que rastrean automáticamente el contenido de todo nuestro sitio. Su estrategia se basa en analizar en primer lugar la página de inicio, que será la que hemos incluido en su base de datos, y empezar a rastrear siguiendo todos los enlaces que se encuentre.

Por un lado este tipo de rastreos puede parecernos excesivo en cuanto a los recursos consumidos por parte de nuestro servidor, sobre todo si nuestro sitio tiene una tasa de actualización baja. En el caso contrario se encuentran los contenidos que tienen una frecuencia de actualización constante, por lo que puede resultar conveniente no indexar el contenido, ya que cuando los usuarios intentaran acceder el contenido ya habría cambiado.

Por otro lado, quizás tengamos contenido que no deseamos que sea publicado en los buscadores, por lo que será conveniente ocultarlo para que los robots no sean capaces de indexarlo.

Cuando un robot accede a un sitio web el primer paso que realizan es buscar la existencia de un fichero llamado "robots.txt" en la raíz del sitio. Este fichero se propone para bloquear el acceso de ciertas páginas, por lo que sería buena idea incluirlos en este fichero. A continuación se muestra un ejemplo de este fichero:

```
User-agent: *
Disallow: /cgi-bin/
Disallow: /librerias/
Disallow: *.jpg
```

El único problema de este método es que aunque no se indexan los contenidos de las webs incluídas en este fichero, si se tienen en cuenta los enlaces contenidos en ellas. Por eso puede ser conveniente el uso de metaetiquetas. El siguiente puede ser un ejemplo que se debe incluir en la sección <head> de su documento HTML:

```
<meta name="robots" content="noindex">
```

6.6.4.- Scripts CGI y Documentos SSI

En los servidores destinados a servir contenido dinámico, una de las mayores inseguridades de los mismos es precisamente la generación de este tipo de contenido.

Para reducir estos riesgos vamos a intentar reducir la cantidad de tecnologías que se podrán ejecutar en nuestro servidor. En nuestro caso, la aplicación está desarrollada en PHP, por lo que evitaremos que otro tipo de contenidos dinámicos se ejecuten.

Entre aquellos más comunes que no deseamos ejecutar se encuentran los scripts CGI y los documentos SSI. El primero es simplemente una interfaz que permite ejecutar un programa escrito en cualquier lenguaje siempre y cuando dispongamos del ejecutable o el intérprete, el segundo consiste en un documento HTML común con unos comentarios que contienen comandos que se ejecutan al ser interpretados.

Ambos son ya de por sí una amplia fuente de vulnerabilidades e inseguridades de por sí, pero no tiene ningún sentido tenerlos activos ya que no serán utilizados por nuestra aplicación, por lo que los desactivaremos.

Para desactivar los scripts CGI deberemos modificar el fichero apache2.conf (en mi caso se encuentra en /etc/apache2/apache2.conf), en él deberemos asegurarnos de que no se encuentra la directiva ScriptAlias (que será del estilo de ésta):

```
ScriptAlias  /cgi-bin/  "/var/www/cgi-bin/"
```

Para desactivar los Server Side Includes (SSI) tan sólo deberemos comprobar que no está activado en ningún ámbito de Apache (directorios, servidores virtuales, etc.). Un ejemplo a evitar sería el siguiente:

```
<Directory  />
     Options  +Includes
</Directory>
```

En caso de que realmente necesitemos usar alguna de estas tecnologías deberíamos de seguir una serie de pautas de seguridad adicionales, que se pueden consultar en los recursos bibliográficos del último apartado de esta memoria.

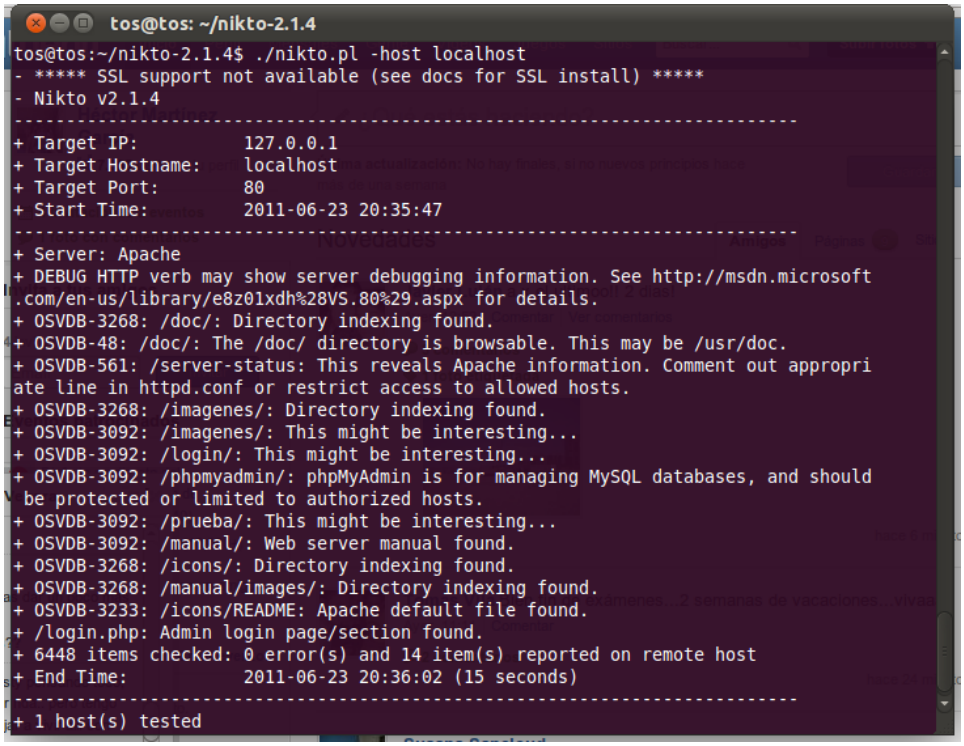
6.6.5.- Análisis de vulnerabilidades Nikto

Uno de los analizadores de vulnerabilidades más conocidos es Nikto que está basado en la famosa librería libwhisker de Perl (antíguo escáner de vulnerabilidadades). Se distribuye bajo licencia Open Source GPL, y debido al estar escrito en Perl está disponible para prácticamente todos los sistemas operativos de escritorio.

Este programa tiene una gran base de datos de distintas versiones de servidores web (unos 1000 aproximadamente), entre las que conoce las distintas vulnerabilidades de que adolece cada versión. También dispone de la técnica fingerprint de la que se habló en el apartado de Nmap para detectar sistemas operativos y distintos servidores.

Vamos a ver un ejemplo representativo ejecutando el análisis sobre el servidor que tenemos ejecutando en en propio equipi. Para ello ejecutaremos el siguiente comando:

```
$ ./nikto.pl –host localhost
```



Como se puede ver, el motor de Nikto realiza un análisis sobre la máquina que le ha sido indicada después del parámetro host. Una vez determinado el software que se ejecuta, empieza a realizarle una batería de pruebas de posibles vulnerabilidades en las que el servidor podría verse afectado.

En caso de que deseemos analizar un sitio que utilice SSL, deberemos añadir la opción –s en la ejecución del comando para forzar peticiones HTTPS, y además cambiar el puerto al 443.

6.7.-Sistema de Accesos

6.7.1.-Introducción

La motivación principal de un sistema de accesos es en primer lugar identificar las personas que acceden al gimnasio, permitiendo llevar un registro de los distintos accesos que se llevan a cabo en la jornada en la que el gimnasio tiene abiertas sus puertas.

Este registro puede ser muy útil para la gestión de nuestras instalaciones, permitiéndonos llevar un control de las personas que se encuentran en nuestras instalaciones. A continuación se van a exponer una serie de ejemplos prácticos que podemos llevar a cabo mediante la utilización de un sistema de control de accesos:

- Control de las **personas que se encuentran en nuestras instalaciones**. En el caso de que haya algún problema de seguridad que pueda causar graves daños a nuestras instalaciones podemos dar a conocer a las autoridades el número de personas que se encuentran en el interior. Estos daños pueden ser un incendio, un terremoto, una fuga de gas, o de algún producto tóxico, alerta sanitaria, o desperfectos en el edificio.

- Conocer el **grado de asistencia** a las distintas actividades. Aunque mediante las reservas podemos conocer el número de personas que han solicitado plaza en las distintas instalaciones, este número no nos da un **índice de ocupación real** de las mismas. Esto es útil para las distintas tomas de decisiones que deben llevar a cabo la dirección de las instalaciones.

- Controlar el **acceso de los socios**. Es importante asegurarse que las personas que tienen acceso a nuestras instalaciones son realmente socios de nuestro gimnasio. Esto sirve tanto para garantizar la seguridad de que ante algún problema tenemos los datos de contacto de los asistentes. Por otro lado se asegura la continuidad del negocio, asegurándonos que no hay accesos indebidos a nuestras instalaciones.

- Conocer la **utilización en las distintas horas** del día. Si cada vez que una persona se registra en el control de accesos, podemos conocer la utilización del gimnasio a las distintas horas del día. Esto es importante a la hora de extraer información para la toma de decisiones por parte de la dirección, ya que permite planificar acciones para establecer una utilización lo más uniforme posible.

- **Conocer qué personas utilizan** más las instalaciones. Esta información es importante a la hora de planificar las distintas estrategias de marketing y precios. Nos permite por un lado conocer la utilización media por persona al mes de las instalaciones.

Las ventajas que se puedan aplicar con estos datos dependen del grado de utilización de la información proporcionada por el sistema de información por parte de la dirección en la toma de decisiones. Otro factor importante son las distintas estrategias de mercadotecnia que se pretendan utilizar para la fijación de precios, y si estas son estáticas y dinámicas.

6.7.2.-Tecnologías de toma de datos

Como podía ser previsible no existe una única tecnología que nos permita realizar los propósitos que se comentaban en el punto anterior. A lo largo de los años han existido distintas tecnologías para registrar códigos que nos permitan asociarlo a personas u objetos.

A continuación se va a realizar un análisis de las distintas tecnologías aplicables para realizar esta labor:

6.7.2.1.- Manual

Esta opción más que una tecnología sería una solución de emergencia para curarnos en salud en caso de que la tecnología que hayamos elegido para el sistema de accesos tuviera un fallo.

Por ejemplo, en un supermercado disponen de códigos de barras para pasar los distintos productos en caja, pero si este sistema fallara siempre pueden teclear el código manualmente.

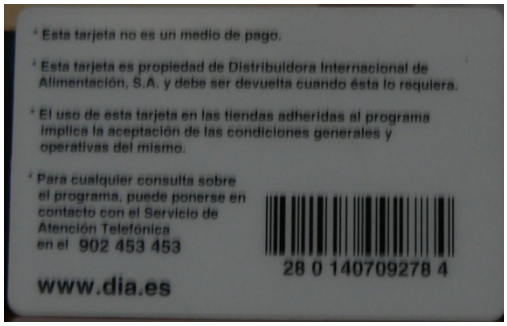
6.7.2.2.- Código de Barras

Al igual que en un supermercado utilizan el código de barras para identificar un determinado producto, nosotros podemos utilizar un código similar para identificar una persona. El funcionamiento se basa en que cada persona debe tener asignada un código, que será el que esté impreso en el código de barras. De esta forma cuando identifiquemos que ese código ha sido leído, mediante el software sabremos qué socio ha entrado o salido de nuestro gimnasio.

El uso de esta tecnología se da en gran parte en los terminales de punto de venta (TPV), en los que el vendedor pasa el código de barras por el código de barras para identificar qué producto se desea comprar y conocer atributos como su precio o características que tenga asociadas ese producto. También es ampliamente utilizado en la trazabilidad de una empresa industrial o de mensajería, donde es posible conocer en qué estado se encuentra un elemento en la cadena de producción o en qué lugar se encuentra un determinado paquete. Esto se realiza

estableciendo un control de lectura del código de barras al finalizar un determinado proceso, o al salir o llegar de un lugar.

Para la implementación de esta tecnología necesitaremos un software que permita la impresión de códigos de barras, ya que será este el encargado de transformar un número que nosotros le pasaremos en la imagen con las distintas barras que representan dicho número. Una vez disponemos de dicha imagen ya podremos imprimirla sobre cualquier superficie lisa y preferiblemente plana para que no haya problemas al ser leídas por el lector de código de barras. Evidentemente también es imprescindible disponer de un lector en cada uno de los puntos donde deseemos realizar un control de acceso.



Ejemplo de carnet del Club Dia

6.7.2.3.- Tecnología RFID

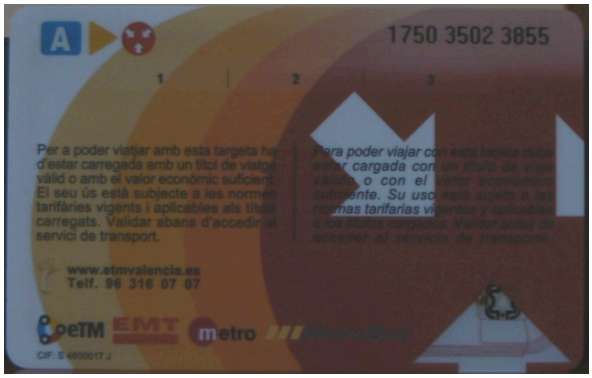
Otra posible forma para identificar a nuestros clientes podrían ser las tarjetas de radiofrecuencia. Como en el ejemplo anterior cada persona estará asignada a un código, el cual será el que esté grabado en el interior del chip RFID. Tienen la ventaja de ser sin contacto por lo que no habrá desgaste de partes móviles. Existen otras formas de utilizar esta tecnología como podría ser guardar todos los datos de los clientes en la propia tarjeta. Pero tiene un gran inconveniente, y es que es necesario un dispositivo capaz de grabar este tipo de tarjetas.

Por lo tanto nos decantaremos por el primer tipo, basado en un lector común (sin ser necesario guardar nada) y tarjetas que de fábrica ya disponen de un identificador único. Por lo tanto en vez de generar nosotros el número para escribirlo en la tarjeta, deberemos asociar el identificador existente en la tarjeta con un cliente en concreto.

Además de las posibilidades comentadas anteriormente existen algunas más, como por ejemplo las tarjetas activas y pasivas. Las primeras disponen de cierta alimentación que permite emitir el código constantemente, y el segundo es justamente al revés, ya que es el lector el que emite mediante las ondas electromagnéticas alimenta la tarjeta para leer su código. Las tarjetas pasivas tienen la ventaja de ser más económicas, son menos propensas a errores en la circuitería, y al no disponer de alimentación no es posible que esta se agote.

Además existen tarjetas RFID que funcionan en distintas frecuencias, pensadas para funcionar a mayor o menor distancia. Cuanto menor sea la frecuencia con la que funcionen, más cerca deberán encontrarse la tarjeta y el emisor. Esto es importante a la hora de tener en cuenta el rango de frecuencia electromagnética más adecuado dependiendo de la legislación de cada país.

Este tipo de tarjetas suele usarse en la industria para la trazabilidad de los distintos procesos, por ejemplo si al final de una cadena de montaje se lee un chip significa que ese lote ya está listo para pasar a la siguiente fase. Así es posible conocer el estado de cada pedido en tiempo real. Otro uso habitual es en los nuevos pasaportes



Ejemplo de tarjeta de MetroValencia con chip RFID que se puede apreciar en la esquina inferior derecha

6.7.2.3.- Tecnologías biométricas

Este tipo de tecnologías tienen la peculiaridad de permitir identificar unívocamente a una persona concreta analizando sus características anatómicas para usarlas en el proceso de identificación. No existe una única forma de implementar una tecnología de identificación biométrica, por lo que existen muchas entre las cuales se encuentran la identificación mediante huella dactilar, identificación del iris, palma de la mano, etc.

6.7.2.4.- Tecnologías banda magnética

A diferencia de otras tecnologías que podían ser utilizadas sin necesidad de ser guardados los datos que se van a leer (basándose en el identificador del chip), este tipo de tecnología sí que necesita efectuar una escritura en su primer uso para poder fijar todos los datos que estarán guardados en la propia tarjeta.

Ejemplo de tarjeta de Banco Santander con banda magnética

6.7.2.5.- Tarjetas con chip

Este tipo de tarjetas disponen un chip integrado en el cual se pueden leer y grabar las veces que queramos. La forma de operar normalmente es grabar los datos una vez en la tarjeta, normalmente en el momento de la creación de la tarjeta, o la asignación a un usuario concreto. Es posible guardar tanto identificadores como datos concretos, por ejemplo en un chip con una capacidad de 64k, podríamos guardar todos nuestros datos personales, incluso los de gran cantidad de nuestros conocidos.

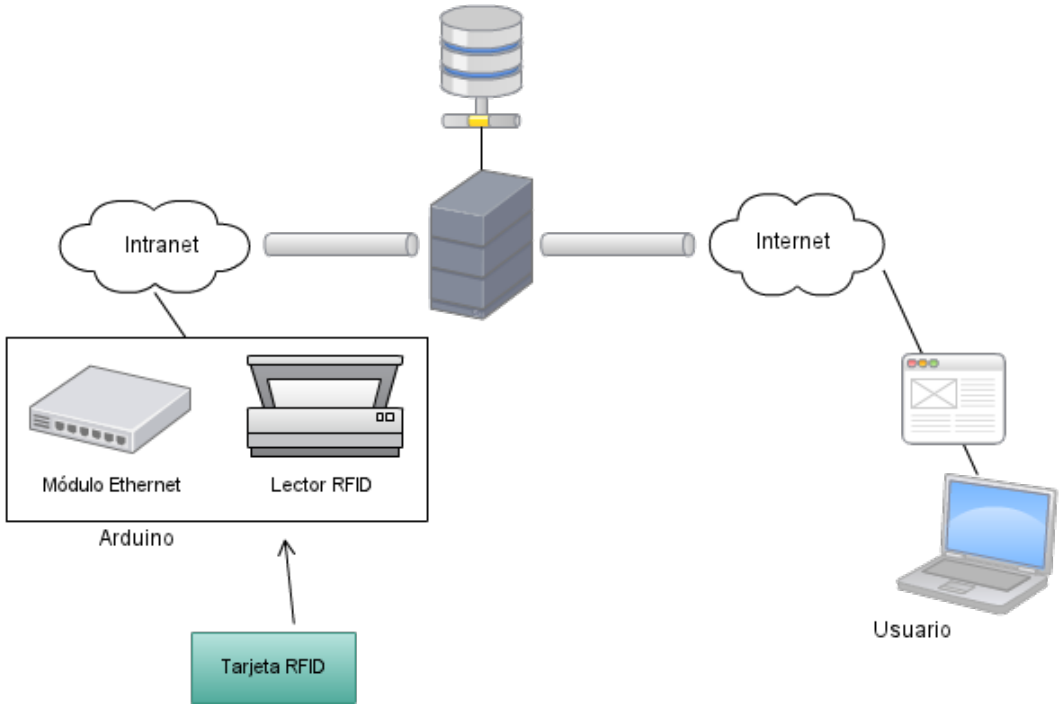
Tienden a utilizarse en campañas de fidelización de clientes como en la imagen, o para identificación de los trabajadores a la hora de entrar y salir de una fábrica. Últimamente están sustituyendo progresivamente a las tarjetas bancarias de banda magnéticas, ya que su variante con chip permite una mayor seguridad e integridad de los datos. También es ampliamente utilizada por las operadoras de telefonía móvil para almacenar datos como los contactos o los mensajes de texto.

Ejemplo de tarjeta de Springfield con chip

6.7.3.-Posibles configuraciones

Durante el estudio del apartado del sistema de accesos se propusieron varias posibles configuraciones. Una de ellas se realizaba mediante una plataforma llamada Arduino que integrará tanto el lector de RFID, como el microcontrolador que enviará los datos al servidor web. La otra posibilidad planteada es mediante un lector de RFID con conexión USB conectado a un ordenador que ejecutará un software específico.

6.7.3.1- Todo en uno implementado con Arduino

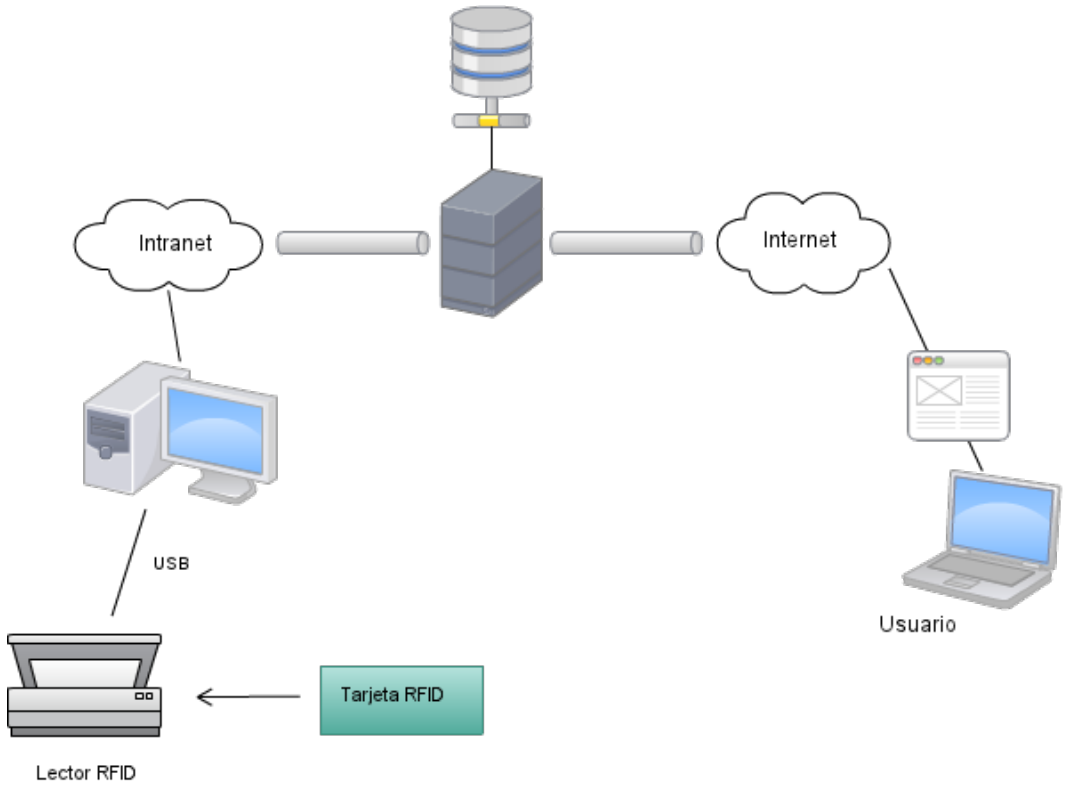


El principal elemento diferenciador de esta configuración está en la parte encargada de la lectura de tarjetas y la comunicación con el servidor a través de la intranet. En este caso se ha optado por implantar un sistema llamado Arduino, que es una plataforma de hardware libre compuesta por un microcontrolador Atmega (normalmente Atmega168). Esta plataforma puede usarse y modificarse libremente sin tener que adquirir ningún tipo de licencia, por lo que nos permite adaptar el hardware a las necesidades específicas de nuestro proyecto permitiendo ahorrar costes entre otras ventajas.

Es posible añadirle una serie de módulos a la placa Arduino, ya que por sí sola únicamente es capaz de leer y escribir señales analógicas y digitales a través de sus pins. Las funcionalidades concretas que queremos extender son en primer lugar el módulo capaz de leer las tarjetas RFID, y por otro el módulo que es capaz de conectarse vía Ethernet al servidor para transferirle los datos.

La plataforma Arduino utiliza su propio lenguaje de programación de alto nivel, de forma que el desarrollo es mucho más rápido y sencillo. La sintaxis es muy similar a C o Java (aunque en realidad el lenguaje está basado en otro llamado Processing). Para empezar a desarrollar en esta plataforma podemos descargar el IDE que está disponible para Linux, Mac y Windows. Puede descargarse forma libre en esta web: http://arduino.cc/es/Main/Software

6.7.3.2-PC independiente con lector USB



En este caso el dispositivo Lector RFID no existe ningún elemento que disponga de autonomía propia que sea capaz de leer las tarjetas, y a su vez enviar los datos al servidor de la aplicación. Para lograrlo se van a unir dos dispositivos diferentes, por un lado tendremos un lector de tarjetas con conexión USB, y por otro lado tenemos un ordenador al cual tenemos conectado el lector de tarjetas. El sistema de conexión es muy sencillo ya que se alimenta y envía/recibe datos por el puerto USB.

El problema de este tipo de configuración es la dependencia de un ordenador físico, con su correspondiente consumo eléctrico, mantenimiento, espacio físico que ocupa, etc. Por ello este equipo puede ser del personal de seguridad que haya en la entrada de nuestras instalaciones. Como ventaja podemos mencionar que en caso de fallo del ordenador, éste puede ser reemplazada por cualquier otro. En caso de que el fallo provenga del dispositivo de lectura de las tarjetas, dicho código siempre podrá ser tecleado por un operador que introduzca todos los datos de forma manual.

6.7.4.-Elementos

Para la implementación concreta que vamos a desarrollar en este caso, se va a tener en cuenta la opción del lector USB conectado a un PC. Con anterioridad se ha explicado el funcionamiento del esquema muy por encima, por lo que ahora se pretende dar un mayor grado de detalle de la opción elegida.

La tarjeta que se va a utilizar no dispone de impresión alguna salvo una numeración en la parte delantera, cuyas 10 primeras cifras (en este caso 0001045038) coinciden con el código de identificación del chip RFID. Dicho chip es invisible al ojo humano, ya que no se advierte su presencia desde el exterior al permanecer totalmente oculto en el interior.

Este tipo de tarjetas son fácilmente conseguibles en múltiples formatos, desde la tarjeta con formato brazalete para llevarla en la muñeca, hasta otras con formato de pinza para que los corredores las lleven en sus zapatillas. También es posible ser impresa con diferentes motivos como el logo del gimnasio, marca de la empresa, instrucciones de uso, o condiciones varias.

Esta es una de las tarjetas que se va a utilizar como ejemplo en este proyecto:

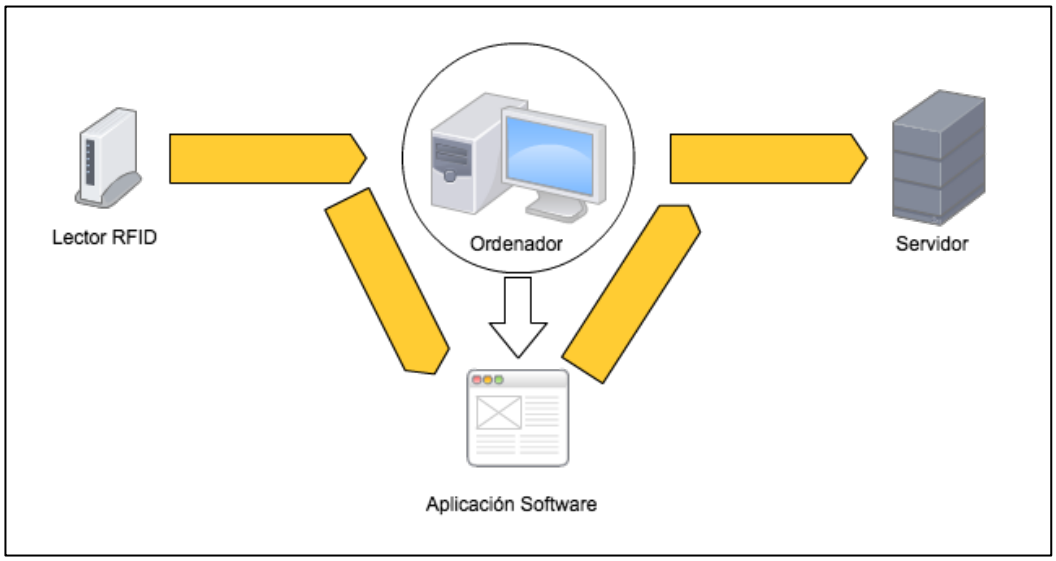
Como complemento a la tarjeta RFID debe haber un dispositivo que sea capaz de extraer el contenido que se almacena en su interior. En este caso se trata de un dispositivo USB que se conecta directamente a un puerto libre del ordenador. No es necesaria la instalación de ningún dispositivo, ya que es reconocido como un teclado común (comprobado en Linux, Mac y Windows).

El funcionamiento es realmente sencillo, ya que una vez conectado el dispositivo, únicamente será necesario acercar una de las tarjetas a una distancia de 8 cm o menos. Una vez detecte dicha tarjeta, enviará su código por el puerto USB. Si nosotros deseamos leerla, necesitaremos un programa que lea de la entrada estándar como puede ser Bloc de notas, Gedit, Kate o TextEdit.

6.7.5.-Funcionamiento

En este apartado se va a hacer hincapié en el funcionamiento del sistema de control de accesos. En apartados anteriores ya se han mencionado los dispositivos que van a intervenir en este proceso, pero falta uno muy esencial: el software. Nada de esto tendría ningún sentido si no hubiera un software capaz de controlar y gestionar todo el entramado.

Este software se ha diseñado para ser muy sencillo, pero puede refinarse hasta ser realmente complejo (ya que incluso podría darse el caso de instalar un dispositivo de acceso en la puerta de cada ubicación. Básicamente el cometido es leer la tarjeta para enviarla mediante un paquete http al servidor web que será quien lo procese.



En esta imagen se puede apreciar el Flujo de Datos que se mantiene en el proceso de lectura de la tarjeta RFID. En primer lugar cuando la tarjeta se acerca al lector RFID, esta genera un impulso al ordenador donde se envía el código leído mediante el puerto USB. En este ordenador hay ejecutándose una aplicación software esperando la llegada de tarjetas leídas. Cada vez que recibe una de estas tarjetas, envía un paquete HTTP al servidor web, que es el que se encarga de su proceso en última instancia.

Será el servidor web quien busque en la base de datos de reservas, aquella a la que se está accediendo, y marque esta de alguna forma para que quede constancia de que dicho usuario ha accedido a la aplicación.

En la imagen que se muestra a continuación se puede ver el proceso completo que conlleva la realización de un acceso, desde la lectura de una tarjeta de RFID a la inserción del registro de acceso en la base de datos.

7.-Evaluación

El paso final después de la implementación de toda la aplicación web, es la valoración del resultado obtenido. En primer lugar tendremos que observar la aplicación una vez desarrollada para poder entrar a hacer una valoración sobre el resultado. Una vez vistos los resultados, deberían identificarse las posibles mejoras para sucesivas iteraciones en el proceso de desarrollo.

Lamentablemente para ofrecer una valoración de su uso, este software debería implantarse en un gimnasio, preferiblemente con una prueba piloto. Esto es así porque debería ser el cliente quien valorara si la aplicación cumple con sus expectativas y resuelves sus necesidades. Por ello en este apartado se ha obviado este apartado a espera de que algún cliente se interese en la compra de la aplicación.

7.1.- Resultados finales

Como bien hemos comentado, en este primer apartado vamos a explorar el resultado final de los distintos apartados de los que consta nuestro sitio web. Hay que destacar que esta aplicación todavía tiene mucho por mejorar, por lo que en versiones sucesivas podrá cambiar sustancialmente tanto su visualización, como sus opciones y herramientas.

Pantalla de inicio

En primer lugar la pantalla de inicio, se ha incluido en la parte izquierda un contenedor donde las distintas noticias van apareciendo de forma animada. Aparece tanto la imagen que se le puede agregar, como el titular de la misma. Aparecerán hasta un máximo de tres noticias, las cuales ofrecerán navegabilidad hacia la noticia completa haciendo click en alguno de los elementos visibles.

En la parte derecha se ha optado por ofrecer una serie de accesos muy visuales a las distintas secciones de nuestro sitio web. Aunque pueda parecer repetitivo, dado que es la misma función que desarrolla el menú navegacional, se ha incluido para ofrecer un contenido más amigable al usuario en sus primeras visitas.

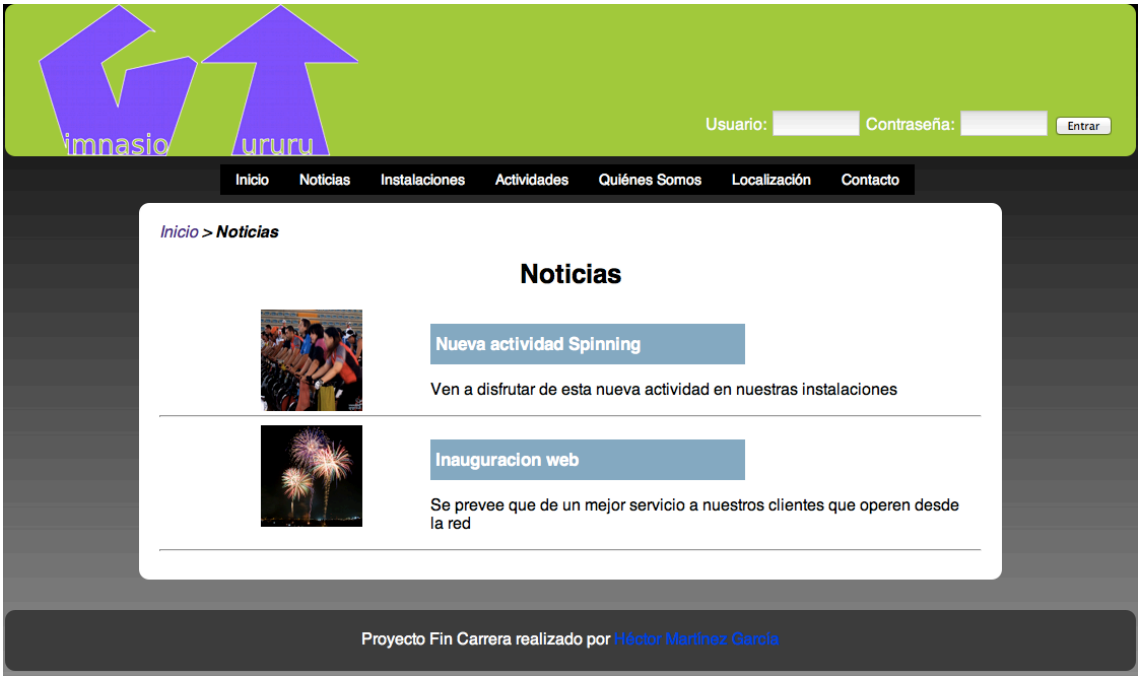
Sección noticias

En la sección noticias aparecerá un listado de las últimas noticias publicadas por el administrador, hasta un máximo de 15. A este apartado se podrá acceder únicamente desde el menú navegacional de la parte superior, o desde su homólogo en forma de imágenes de la página principal. Esto es así ya que si hacemos click en una noticia concreta, únicamente nos aparecerá una sección donde se visualizará esa noticia en detalle.

Dichas noticias aparecen en orden de publicación, por lo que las últimas en ser publicadas aparecen en la parte superior. Conforme el administrador va añadiendo nuevas noticias estas van apareciendo cada vez más abajo.

Los detalles de cada noticia que aparecen en esta sección son: una miniatura en la parte izquierda de la imagen que contiene la noticia, el titular destacado sobre todo negro y letras blancas en la parte superior, y por debajo de ésta el subtitular de la noticia.

Se ofrece navegabilidad a la sección que muestra la noticia completa tanto en la imagen en miniatura, como en el titular de la noticia.

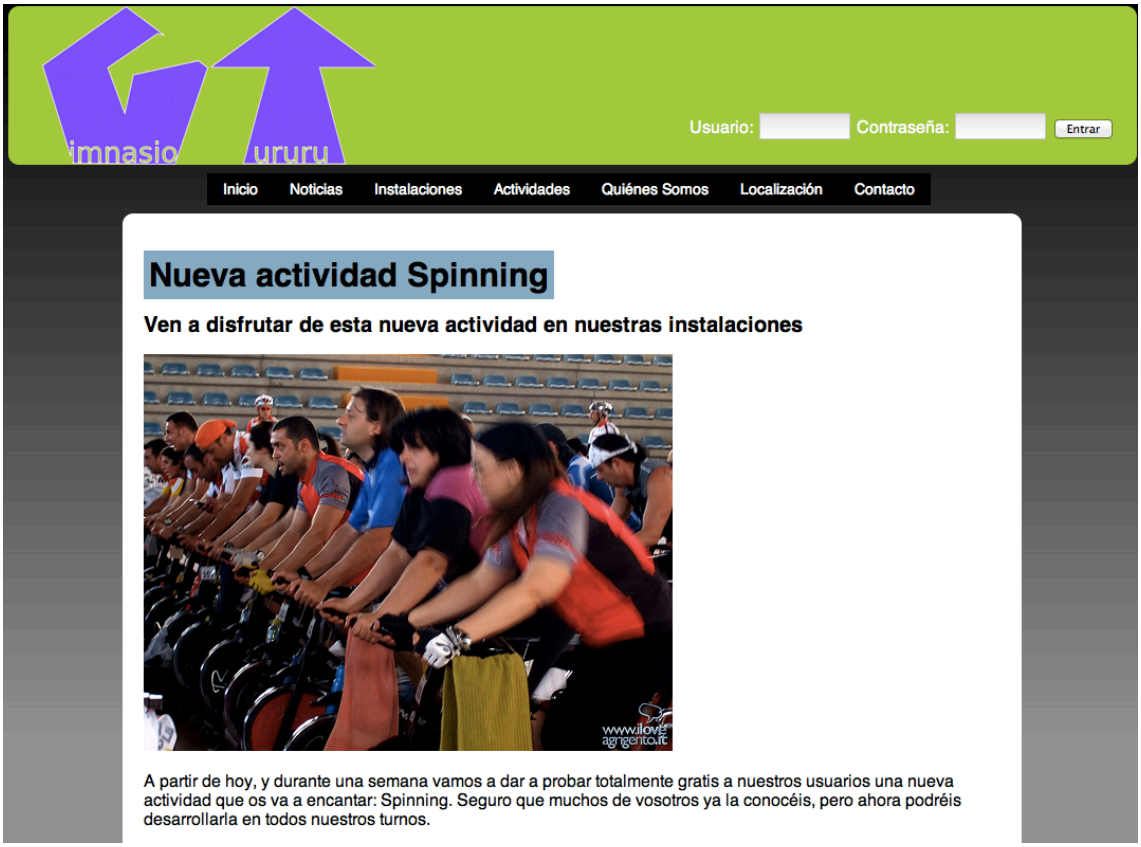


Sección detalles noticia

En la sección "detalles noticia" se muestra el contenido completo de la noticia que haya seleccionado el usuario. Esta selección bien podrá haberse realizado en el carrusel de noticias que aparece en la página de inicio, o bien en haciendo click en alguna noticia de las que aparecen en la sección noticias.

En primer término aparece el titular de una forma destacada sobre un fondo azulado y con un tamaño de letra prominente. Este especial tratamiento del texto pretende que llame la atención desde el primer instante en el que el usuario entra en contacto con la noticia, para de esta forma el titular sea leído con anterioridad al resto. Un poco más abajo se puede ver en letra más discreta (aunque más grande y resaltada de lo normal) el subtitular de la noticia. Este apartado puede ocupar entre 1 y 3 líneas de visualización.

Posteriormente aparece una imagen relacionada con la noticia, y que se pretende que sea uno de los elementos principales de la sección (junto con el titular). Esta imagen deberá ser lo más llamativa posible, procurando que disponga de colores vivos, y que tenga la mayor relación posible con el título de la noticia. Hay que tener en cuenta que dicha imagen aparecerá también en el carrusel de noticias de la página de inicio.

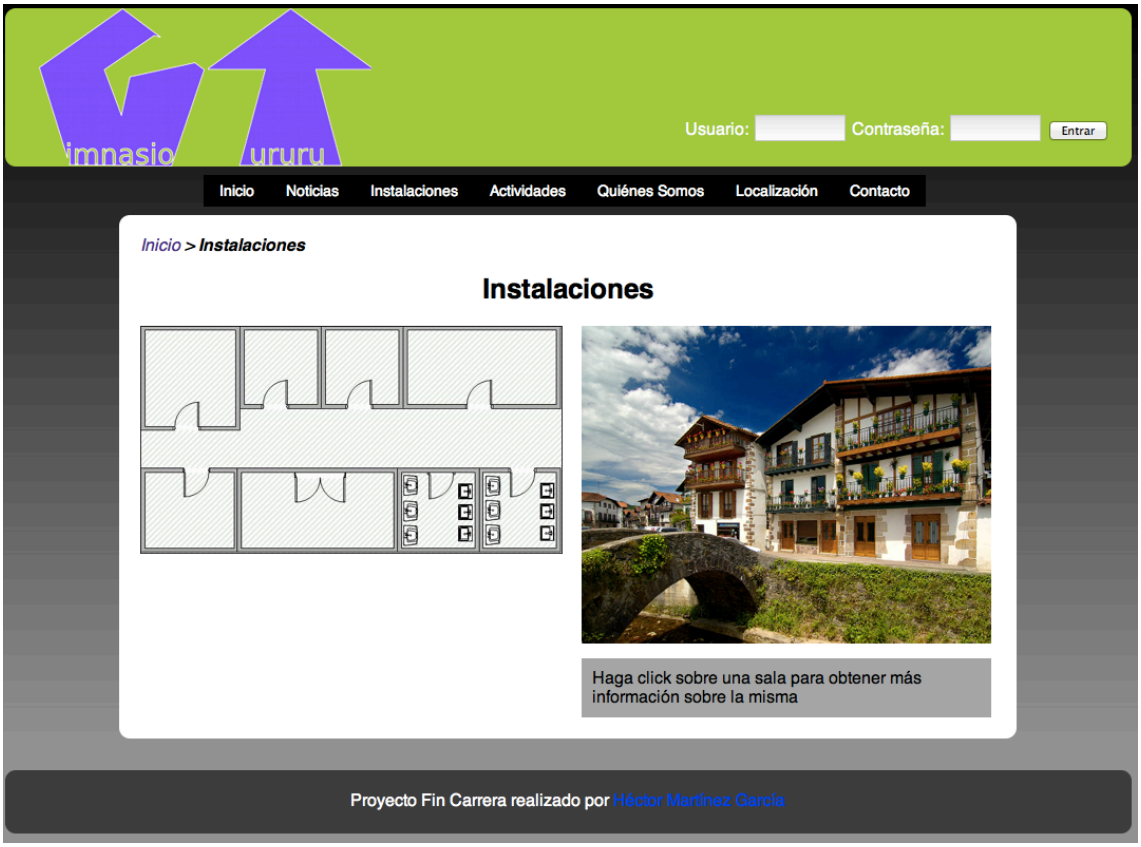


Sección Instalaciones

En la sección instalaciones se puede apreciar en la parte izquierda un plano de las instalaciones del gimnasio, y en la derecha una imagen puramente decorativa. Sobre cada una de las estancias reflejadas en el plano podrá hacerse click con el ratón para poder mostrar más información.

Dicha información constará en primer lugar de una fotografía de dicha estancia, procurando que sea lo más impactante posible para el usuario para llamar la atención de los usuarios que nunca han visitado nuestro gimnasio. Todas las estancias deberán estar lo más limpias posibles a la hora de la realización de la fotografías, y siempre intentarán tomarse las zonas que dispongan de más material y mejores perspectivas.

En la parte inferior de dicha fotografía aparece un cuadro de texto en el que se muestra una breve descripción sobre la estancia en la que se ha hecho click. Esta descripción ocupa un par de líneas a lo sumo.

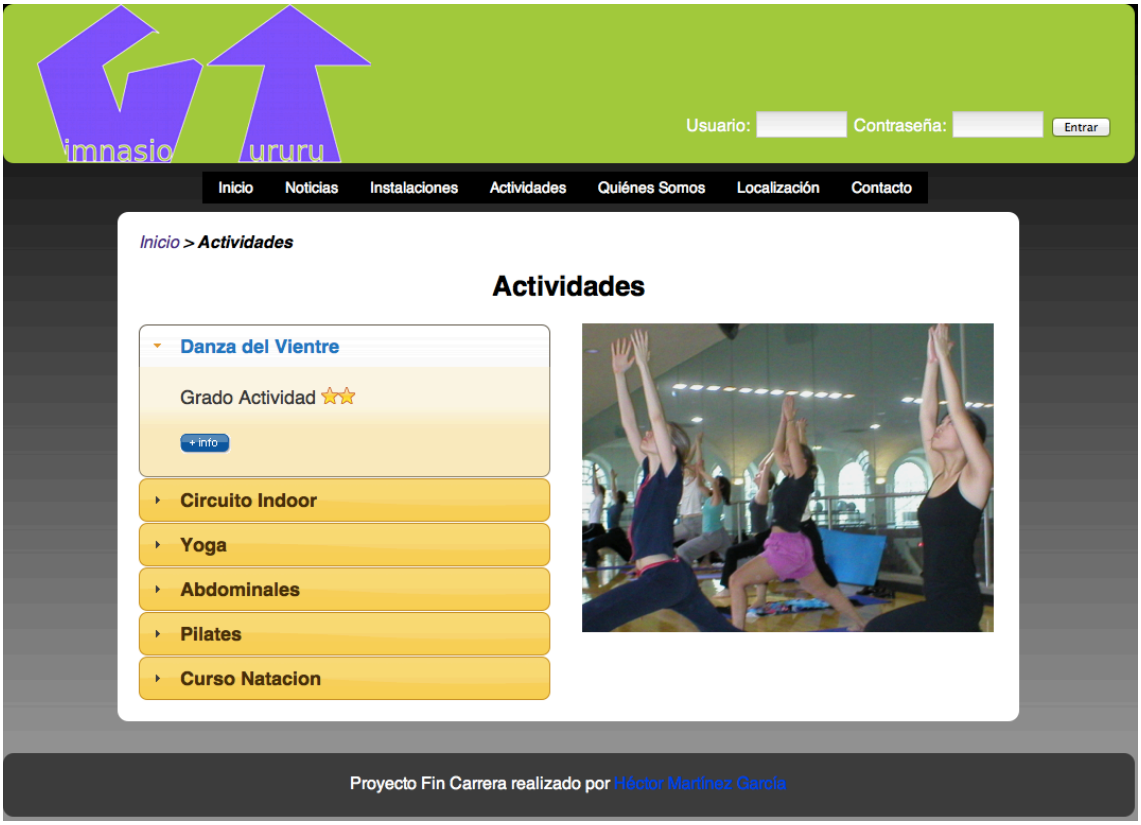


Sección Actividades

En la sección actividades se puede ver en la parte izquierda un listado de todas las actividades dirigidas de las que dispone el gimnasio, y en la parte derecha una imagen puramente decorativa. Al mismo tiempo al que vamos a haciendo click en las cabeceras de las distintas actividades, irá apareciendo un espacio visual dedicada a dicha actividad.

En este espacio extra se puede ver un indicador que nos muestra el grado de actividad que presenta la actividad seleccionada. De esta forma se podrá acceder a este dato sin necesidad de entrar a los detalles de la información.

A su vez, en este espacio extra aparece un botón rotulado como “+ info” que permite ir a otra sección que nos muestra una página con toda la información sobre la actividad.

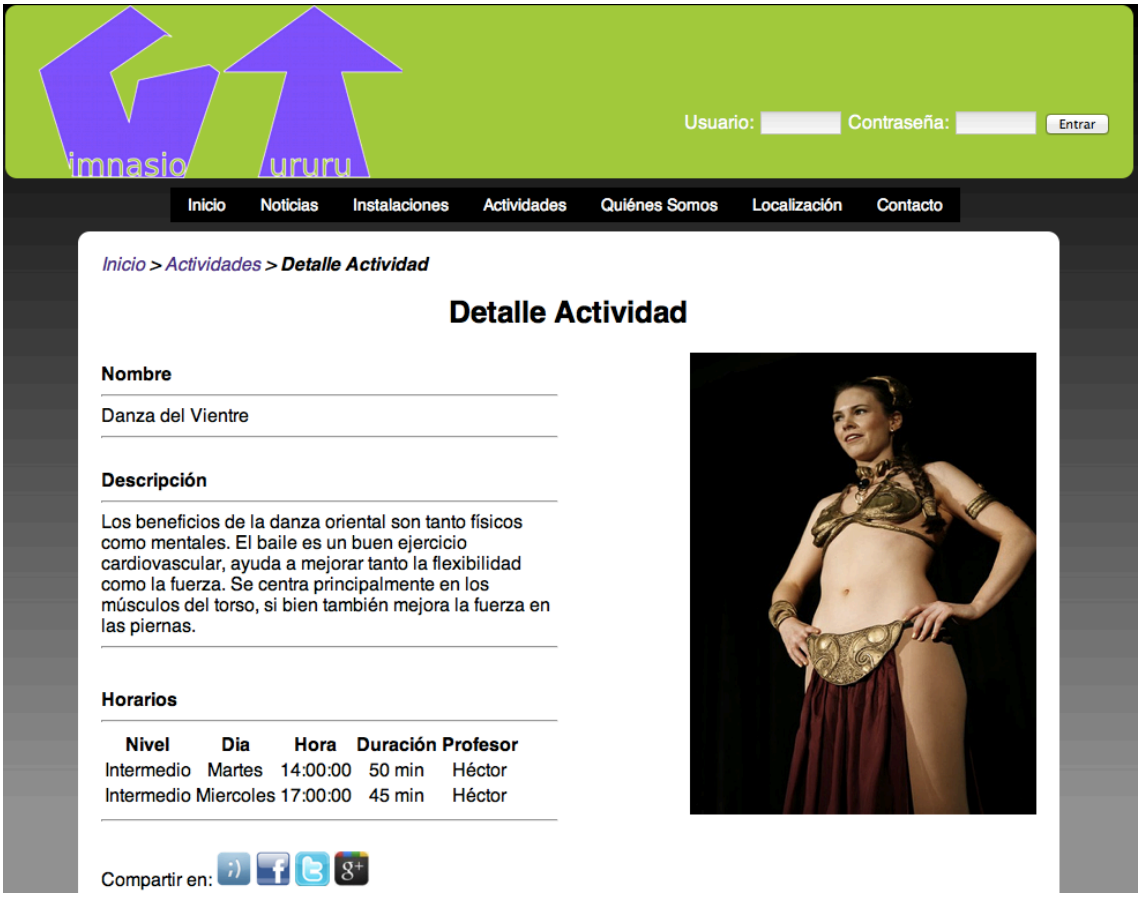


Sección Detalle Actividad

En la sección detalle actividad se puede ver todos los detalles de la actividad que se ha seleccionado en la lista de actividades de la sección anterior. La parte más destacable a primera vista en esta sección es la fotografía que aparece en la parte derecha, y que pretende mostrar un ejemplo de desarrollo de esa actividad.

En la parte izquierda es donde aparecen todos los datos de la actividad seleccionada. En primer lugar aparece su nombre, cuyo tamaño se aconseja sea menor a 5 palabras. En segundo lugar podemos ver una breve descripción, que aunque no tiene tamaño máximo ni mínimo, debería tener una longitud aproximada de unas 40 o 50 palabras. En tercer lugar podemos ver los horarios en los cuales se imparte dicha actividad. En este apartado se mostrará el nivel de dicha actividad, el día de la semana y la hora a la que se imparte, la duración de la misma y el profesor de la misma. En caso de acceder mediante un usuario registrado, este apartado nos permitirá reservar una plaza.

Por último lugar en la parte baja se plantea un sistema para que los clientes puedan compartir nuestra web si la consideran de interés para otros usuarios. Esto ayudará a promocionar nuestro sitio a través de los propios clientes.



Sección Quienes Somos

La sección quienes somos es tremendamente sencilla, y únicamente pretende mostrar información institucional sobre la empresa. Esta información podrá modificarse a gusto de la empresa que desee implantar esta aplicación.

En el caso concreto de este proyecto se ha incluido información ficticia sobre el gimnasio. En concreto el año de fundación del mismo, los valores en los que se basa la empresa, etc.

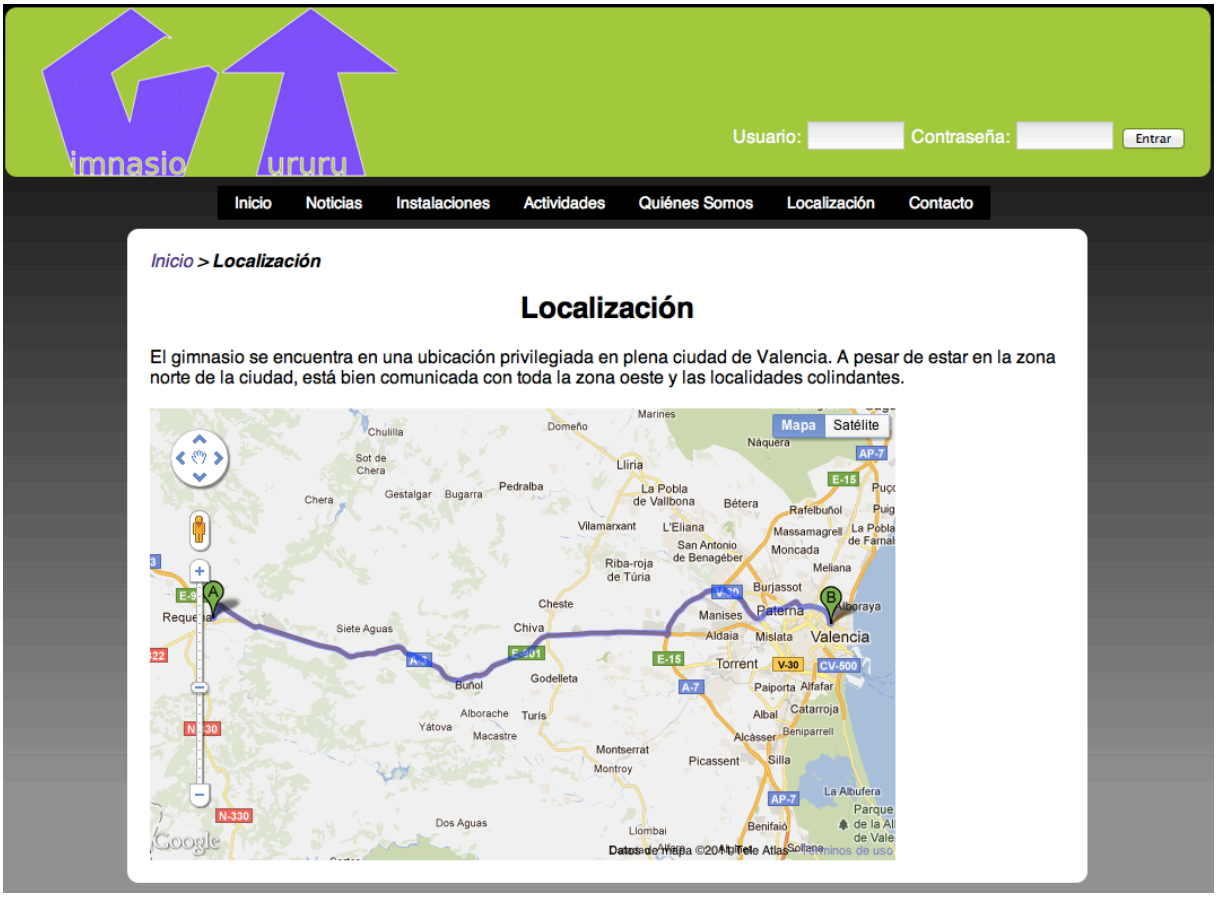
También se muestra una fotografía que pretende dar una imagen de empresa con trabajadores unidos y bien comunicados. Esto aporta una imagen de profesionalidad al gimnasio.

Sección Localización

En la sección localización se ofrece una de las funcionalidades más novedosas, ya que se basa en una de las funcionalidades de la futura implementación de HTML 5. Esta funcionalidad es la de geolocalización, y a pesar de no ser todavía un estándar, todos los navegadores actuales lo están ofreciendo mediante una API común a todos ellos.

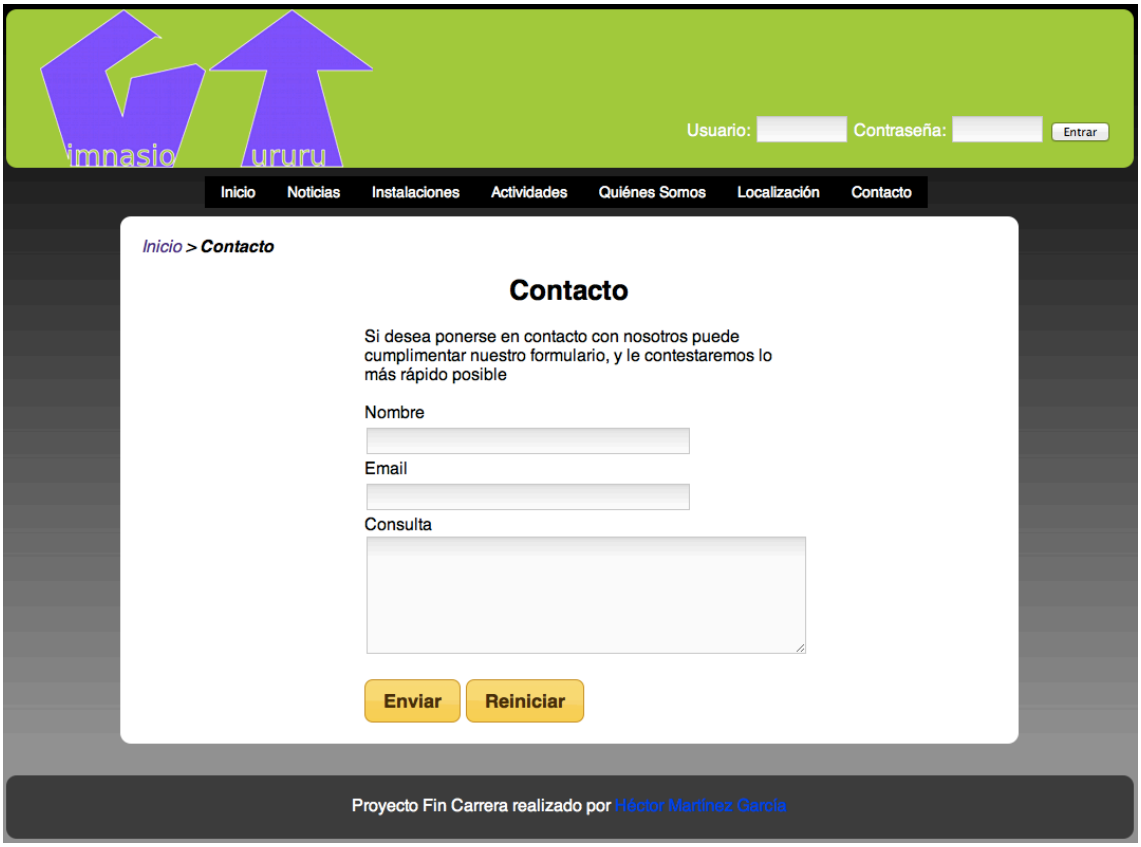
Mediante estos datos de geolocalización, podemos generar una ruta hacia nuestro gimnasio. La forma de lograr esto es mediante la API de Google Maps, que permite generar una ruta a partir de un punto de partida (el escogido mediante geolocalización) y un punto de llegada (el propio gimnasio). Toda esta información aparece reflejada en el mapa de la zona para que el usuario sea capaz de identificar más rápidamente el trazado del itinerario.

En el supuesto fictício del desarrollo de este proyecto fin de carrera se ha supuesto que las coordenadas de latitud y longitud del gimnasio son las que pertenecen a la ETSINF. En una ampliación del mapa se podría apreciar con más claridad que en la imagen que se muestra a continuación.



Sección Contacto

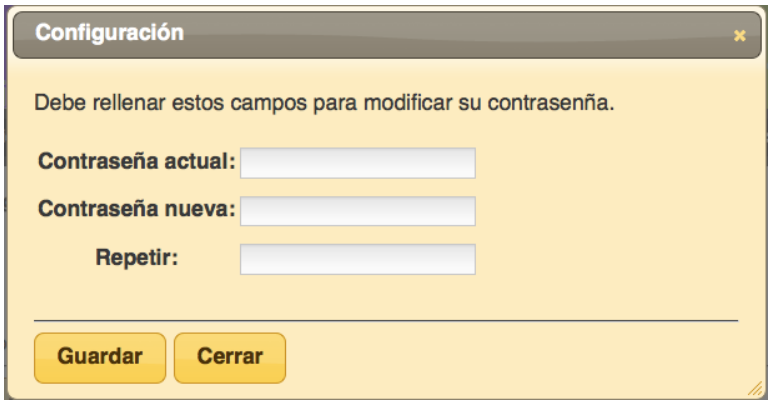
En la sección contacto no se pretende dar ninguna información, aunque al principio del proyecto se pensó que podría mostrar datos de contacto como el correo electrónico, la dirección postal, el número de teléfono, etc. En vez de esto se ha optado implementar una funcionalidad de contacto vía telemática integrada en la propia aplicación, de esta forma se evita tener que derivar el contacto a otro medio distinto a Internet. Dicha vía telemática consiste en un formulario que el cliente debe completar con su consulta, y con un e-mail para poder recibir una contestación por parte de la empresa.



Sección Configuración

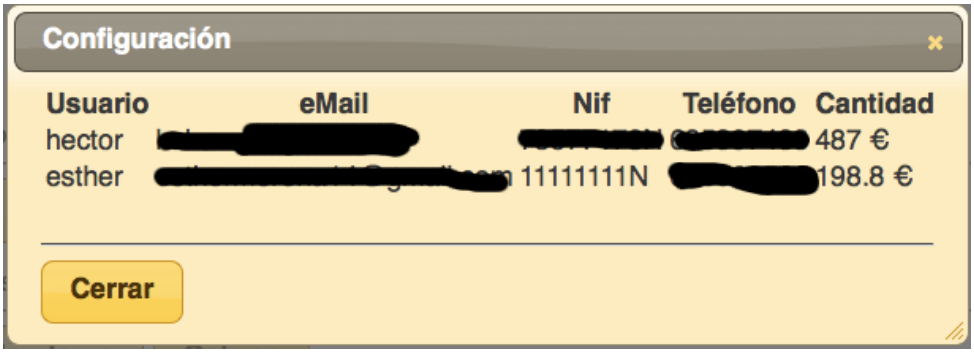
Esta sección está disponible únicamente para los administradores del sistema, por lo que un usuario común tiene acceso a la misma. El acceso se realiza mediante el icono con forma de engranaje que aparece en la parte superior, junto al nombre de usuario y el enlace de logout.

Modificar contraseña

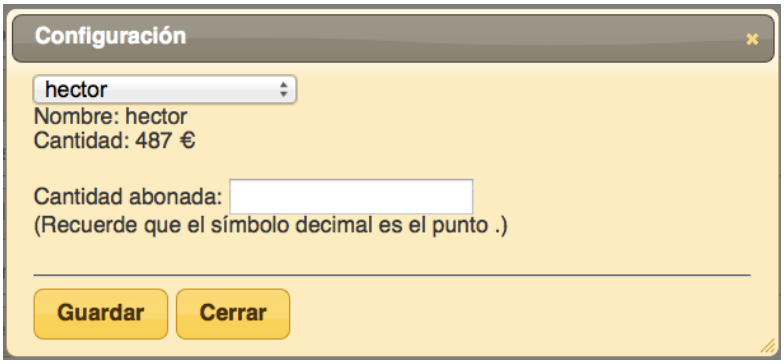


Modificar Datos Personales

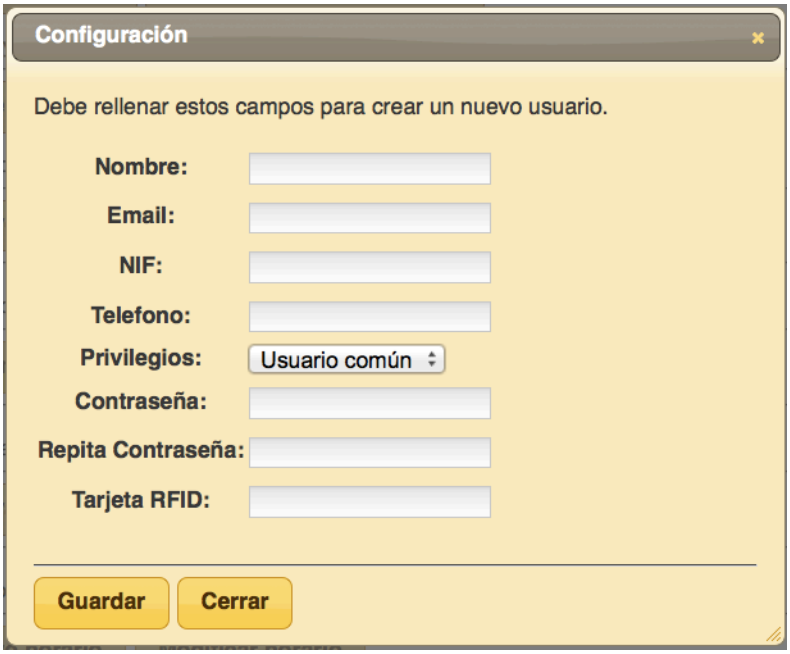
Listar Deudas



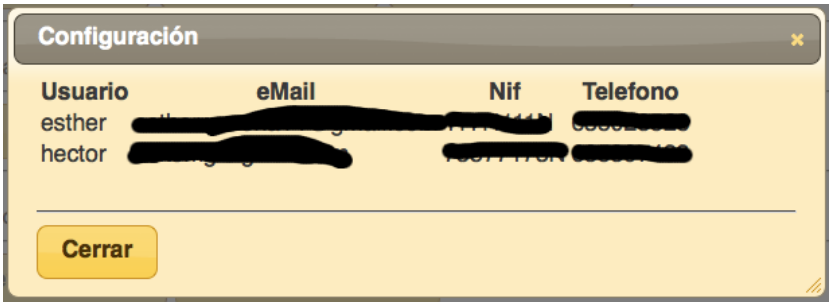
Gestionar cobros



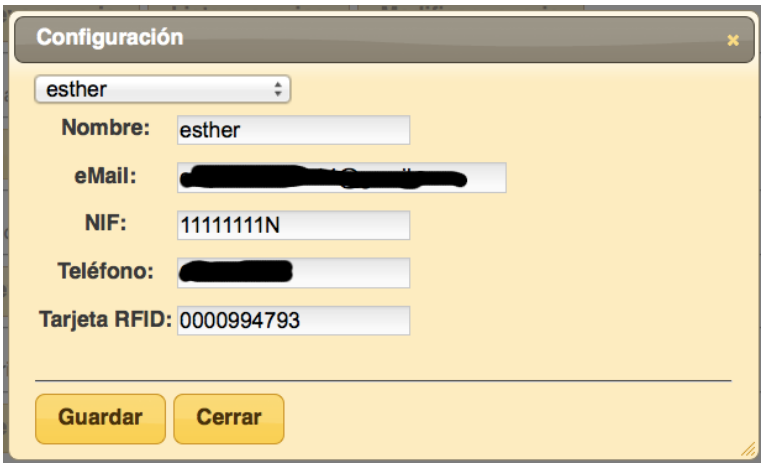
Nuevo usuario



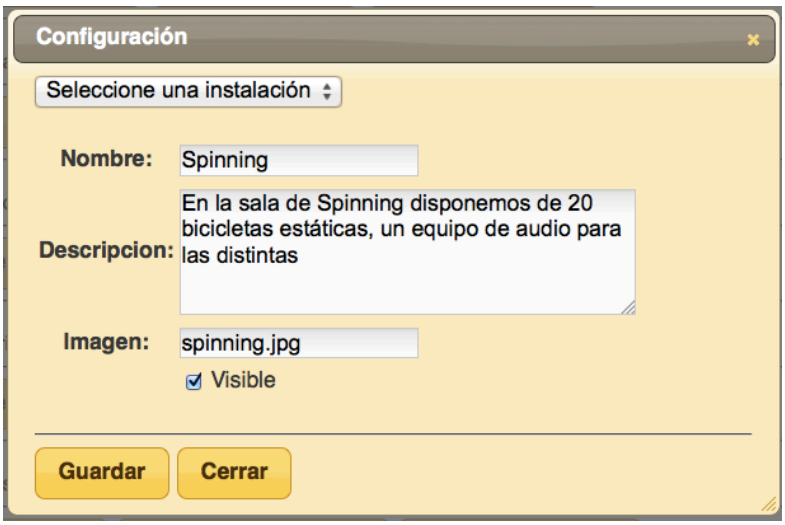
Listar usuarios



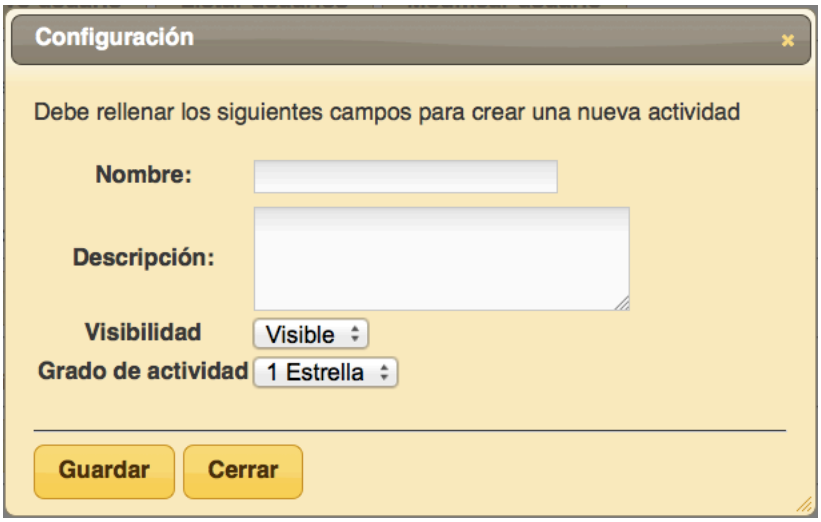
Modificar usuarios



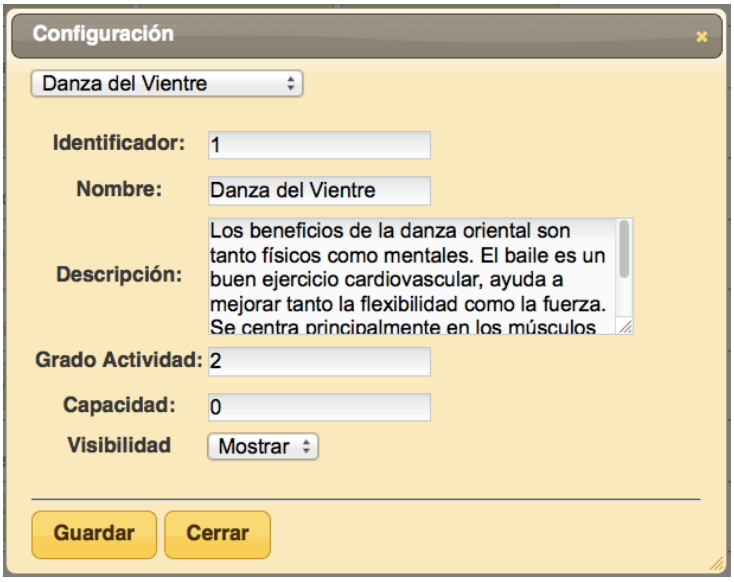
Modificar instalaciones



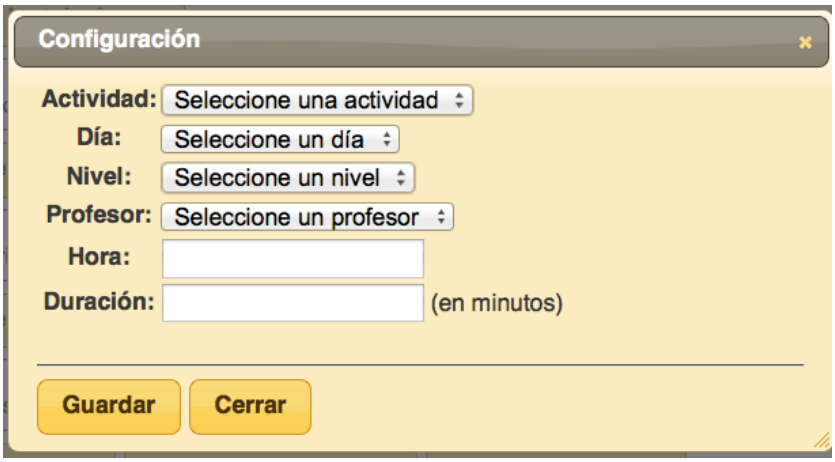
Crear nueva actividad



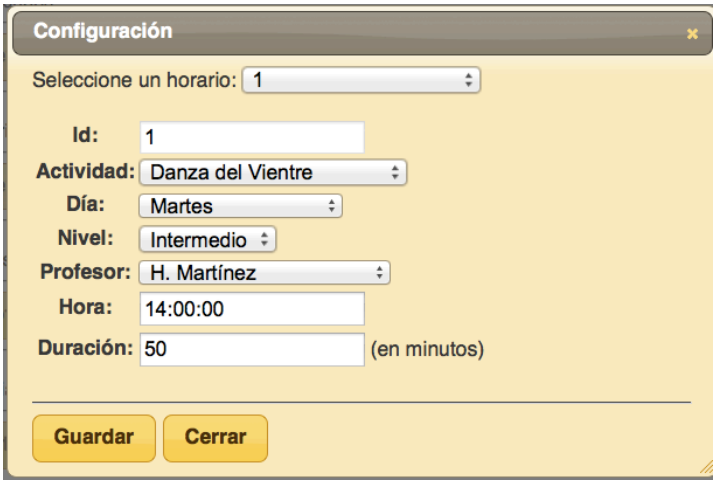
Modificar actividad



Crear nuevo horario

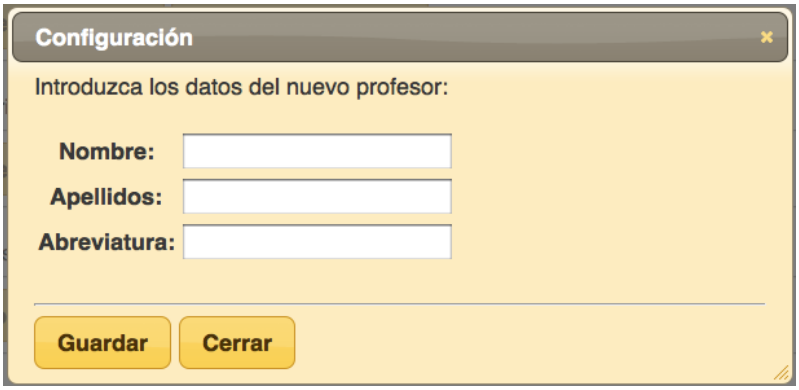


Modificar horario

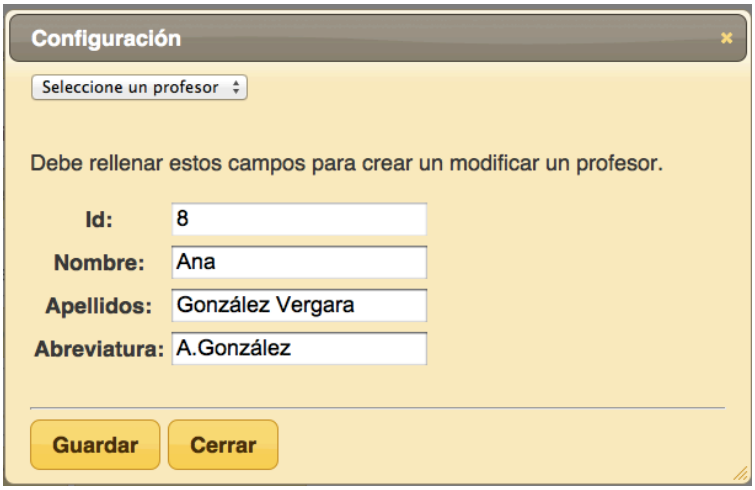


Listar profesores

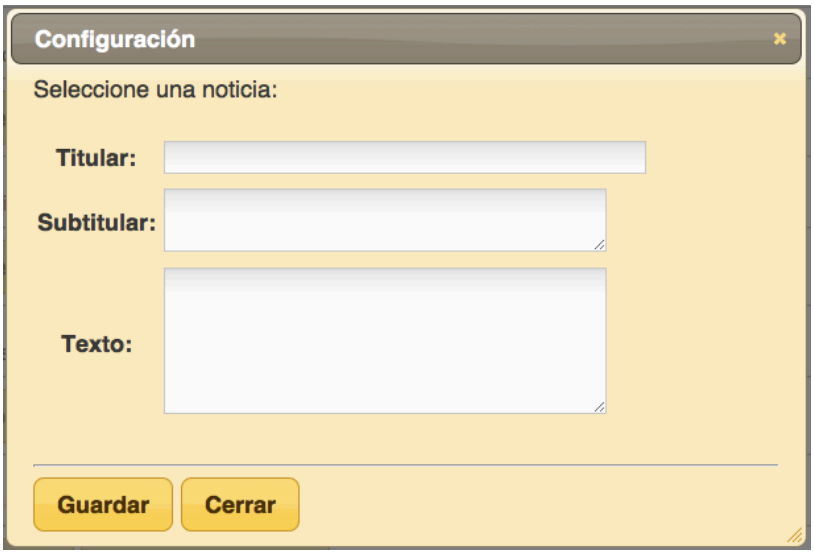
Crear nuevo profesor



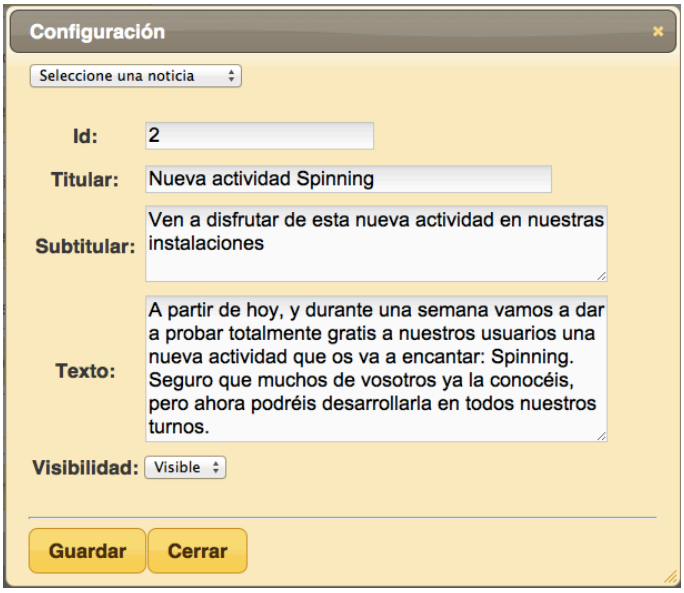
Modificar profesor



Nueva noticia



Modificar noticia



7.2.- Posibles mejoras

En esta sección se pretende hacer una reflexión sobre las posibles deficiencias encontradas tanto en el servidor web, la aplicación web y el sistema de accesos. Es importante dar un punto de vista imparcial, intentando identificar todas aquellos posibles defectos que hayan ido apareciendo, pero también aquellos que está previsto que aparezcan en un fututo.

También hay que destacar que no solo los defectos se pueden mejorar, si no todo aquello desarrollado que no presente ningún tipo de problema, puede someterse a un proceso de mejora continua que permita a una aplicación mantener unos estándares de calidad altos a lo largo del tiempo.

Todas estas posibles mejoras detectadas deberán ser documentadas convenientemente para poder ser evaluadas en futuras iteraciones del proceso de desarrollo. Si la evaluación de éstas es positiva, se procederá a su desarrollo en futuras versiones de la aplicación.

7.2.1.- Servidor Web

En esta primera versión se ha optado por empezar a generar la configuración desde la instalación inicial del servidor, lo cual es un problema para usuarios noveles que deseen implantar este sistema en su empresa. Sería conveniente realizar un autoinstalable que permita al usuario instalar el servidor con un mínimo de opciones, permitiendo que la configuración quede fijada desde un primer inicio. Esto ayudará también a la resolución de posibles problemas, ya que conseguiremos una unificación de todos los sistemas implantados.

Aquí no se ha tenido en cuenta las posibles actualizaciones del servidor, salvo el comentario de que es conveniente disponer de una versión totalmente actualizada. Pues bien, este es un tema importante a tener en cuenta, y en el cual el papel del administrador del sistema debería tener poco peso. La idea es disponer de un script que ejecute consultas para detectar si existe una nueva versión del servidor para poder descargarla. Existiría un problema para el que habría que encontrar una solución para el tema de la pérdida de servicio temporal que se produciría a la hora de la actualización.

7.2.2.- Aplicación Web

En cuanto a la aplicación, existen temas que son de competencia directa de un diseñador gráfico. Un ejemplo de esto es el logotipo y demás elementos gráficos, que podrían mejorar notablemente la estética del sitio web haciéndola más atractiva a clientes y usuarios. Las fotografías también son otro tema importante, ya que las referidas a las instalaciones deberán ser específicas para cada gimnasio. Deberán ser realizadas también por un fotógrafo profesional, y probablemente retocadas por un diseñador gráfico que adecúe los parámetros relativos a colores, luces, etc.

Otro tema a considerar serían las políticas de cobro y tarificación de los distintos gimnasios, ya que es muy probable que existan diferentes criterios en este sentido. Sería interesante hacer un estudio de cuántas formas tienen los distintos gimnasios de tarificar y cobrar a sus clientes. La intención es implementar todas estas posibles formas en la aplicación, ya que de esta forma en futuras versiones se pueda ofrecer un sistema más completo. Así en la propia configuración de la aplicación se pueda decantar por la opción más conveniente para su negocio.

También puede ser conveniente implementar un sistema de ofertas para ciertos clientes, de forma que clientes con ciertos privilegios como pueden ser clientes con grandes ventas, o clientes descontentos. De esta forma también se podrían implementar ofertas a las distintas actividades ofertadas.

7.2.3.- Sistema de accesos

En esta primera versión de este software su configuración es nula, y de hecho problemática, ya que si deseamos modificar algún parámetro, deberíamos acceder al código fuente para cambiarlo y volverlo a compilar. Para futuras versiones será necesario incidir en esta posibilidad que permita una implantación más cómoda.

La seguridad, aunque se ha tenido en cuenta, es todavía muy deficiente. En futuras versiones debería establecerse una conexión segura entre la aplicación software y el servidor web, de forma que no pueda ser interceptada dicha comunicación. También sería conveniente eliminar por completo la información confidencial que se muestra por pantalla (número de la tarjeta

RFID) y sustituirla por información más útil si así se cree conveniente. Sería conveniente también que existiera un sistema de acceso mediante contraseña para la utilización de este software.

Aunque la parte visual no ha sido tenida en cuenta en ningún momento, la aplicación podría ser más atractiva y amigable en cuanto a su interfaz y su uso. Se pretendió que la aplicación fuera portable, por lo que se desarrolló en Java, ya que de esta forma se podrá ejecutar en la mayoría de los sistemas operativos modernos (siempre y cuando dispongan de la máquina virtual Java instalada).

Sin duda la mejora más interesante para llevarse a cabo sería la introducción de este elemento en la propia aplicación web. Se estuvieron haciendo pruebas, y aunque funcionaba, el proceso no era todo lo automático que se deseaba, ya que antes de leer la tarjeta requería la intervención humana (por el cual motivo fue descartada esta posibilidad).

8.-Agradecimientos y Licencias

En primer lugar agradecer la paciencia que han tenido todos conmigo a la hora de realizar este proyecto final de carrera dada la cantidad ingente de horas que se le ha dedicado al mismo. Gracias en especial a mi novia, mis amigos y toda mi familia por lo mucho que ha tenido que aguantar. Este proyecto tampoco habría sido posible sin la información de todos los que han aportado ese granito de arena explicándome su día a día con su gimnasio, su forma de trabajar, y las ideas de mejora.

Las capturas de pantalla de ejemplo, los derechos pertenecen a sus respectivos propietarios.

Todas las fotografías incrustadas tanto en la memoria como en la propia aplicación web poseen su respectiva licencia Creative Commons. Si lo desea puede contactar con dichos autores y ver su portfolio en las siguientes direcciones.

http://www.flickr.com/photos/orangeacid

http://www.flickr.com/photos/bayasaa

http://www.flickr.com/photos/matsuyuk

http://www.flickr.com/photos/viriyincy/

http://www.flickr.com/photos/eivind1983

http://www.flickr.com/photos/flyingsinger

http://www.flickr.com/photos/iloveagrigento

http://www.flickr.com/photos/midiman

http://www.flickr.com/photos/drb62

http://www.flickr.com/photos/hotelcasavelas/

http://www.flickr.com/photos/rinkjustice/

http://www.flickr.com/photos/starwarsblog

El resto de material gráfico, como pueden ser diagramas, esquemas, etc., es de cosecha propia y ha sido realizada por diversos programas como Cacoo, Dia, Gimp, Inkscape y Word.

La música incluida en el vídeo de presentación de la aplicación pertenece a Digital Memories, puede obtener más información en su web oficial o aquí: www.jamendo.com/es/album/91153

La producción del vídeo la he realizado yo con un software que se incluye con los ordenadores de Apple llamado iMovie. Dicho software pertenece a un paquete llamado iLife ’11.

9.-Bibliografía

9.1.-Recursos on-line

Estadísticas de Uso de Servidores Web
http://news.netcraft.com/archives/2011/01/12/january-2011-web-server-survey-4.html

Licencia Apache
http://es.wikipedia.org/wiki/Licencia_Apache

DISEÑO Y USO DE BASES DE DATOS RELACIONALES
http://www.php-hispano.net/articulos/modelo-uso-bases-datos-relacionales.html?pag=1

Cómo hacer que todos los navegadores rendericen HTML5 (incluso IE6)
http://www.elwebmaster.com/articulos/como-hacer-que-todos-los-navegadores-rendericen-html5-incluso-ie6

Página web oficial de OWASP
http://www.owasp.org

Web oficial del proyecto "httpd" Apache
http://httpd.apache.org/

Documentación sobre la instalación de Apache
http://httpd.apache.org/docs/2.2/install.html

Documentación general sobre la versión 2.2 de Apache
http://httpd.apache.org/docs/2.2/

OWASP TOP 10 2010
https://www.owasp.org/index.php/Top_10_2010-Main

Traducción del TOP10 de OWASP
http://owasptop10.googlecode.com/files/OWASP%20Top%2010%20-%202010%20Spanish.pdf

Como eliminar la información sobre versiones de los headers HTTP
http://systemadmin.es/2009/01/como-eliminar-la-informacion-sobre-versiones-de-los-headers-http

Descripción de las directivas del núcleo de php.ini
http://es.php.net/manual/es/ini.core.php

Falsificando el fingerprint de Linux con Iptables
http://omniumpotentior.wordpress.com/2010/12/06/falsificando-el-fingerprint-de-linux-con-iptables/

Mostrar lista de cabeceras de servidores web
http://www.codigomaestro.com/apache/mostrar-lista-de-cabeceras-de-servidores-web/

Manejando errores 404 con Apache
http://www.rolandovera.com/2009/02/05/manejando-errores-404-con-apache/

Tipografía: Wikipedia
http://es.wikipedia.org/wiki/Tipografía

Tipografías serif y sans-serif
http://www.desarrolloweb.com/articulos/1652.php

Fuentes seguras para la web
http://www.elwebmaster.com/articulos/fuentes-seguras-para-la-web-como-y-cuales-son

Core fonts for the Web
http://es.wikipedia.org/wiki/Core_fonts_for_the_Web

PHP: Wikipedia
http://es.wikipedia.org/wiki/PHP

Geolocalización en HTML 5
http://jafrancov.com/2010/07/geolocalizacion-con-html5/

Comparación de frameworks javascript
http://www.maestrosdelweb.com/editorial/comparacion-frameworks-javascript/

Tipos MIME
http://www.htmlquick.com/es/reference/mime-types.html

Software Apache Benchmark
http://www.softhoy.com/apache-benchmark.html

Adquisición de Red Hat Enterprise Linux
http://www.es.redhat.com/products/rhel/purchasing_guide.php

Etimología de 10 distribuciones Linux
http://alt1040.com/2011/05/etimologia-distribuciones-linux

Apache License
http://es.wikipedia.org/wiki/Apache_License

Licencias Apache
http://www.apache.org/licenses/

Servidor Apache
http://es.wikipedia.org/wiki/Servidor_HTTP_Apache

Logrotate, para gestionar y rotar los logs de nuestro S.O.
http://www.linux-party.com/modules.php?name=News&file=article&sid=5778

Bloquear o eliminar páginas mediante un archivo robots.txt
http://www.google.com/support/webmasters/bin/answer.py?hl=es&answer=156449

Uso de las metaetiquetas para bloquear el acceso a su sitio
http://www.google.com/support/webmasters/bin/answer.py?answer=93710

Iniciativa para la accesibilidad web del W3C (Inglés)
www.w3c.org/WAI

Pautas de accesibilidad web
http://www.codexexempla.org/traducciones/pautas-accesibilidad-contenido-web-2.0.htm

Intel Xeon
http://en.wikipedia.org/wiki/Xeon

SDRAM DDR 3
http://es.wikipedia.org/wiki/DDR3

Metaetiquetas
http://www.desarrolloweb.com/wiki/metaetiquetas.html

Script
http://es.wikipedia.org/wiki/Script

Log
http://es.wikipedia.org/wiki/Log_(registro)

Autentificación
http://msdn.microsoft.com/es-es/library/syf5yeat.aspx

Depuración
http://es.wikipedia.org/wiki/Depuración_de_programas

Caché (Inglés)
http://www.mnot.net/cache_docs/

Monitorización de Servidores de Internet
http://es.wikipedia.org/wiki/Monitorización_de_Servidores_de_Internet

9.2.-Recursos bibliográficos

Designing Interfaces: Patterns for Effective Interaction Design 2n Edition
Jenifer Tidwell (O'Reilly 2010)

Ingeniería de la web y patrones de diseño
Mª Paloma Díaz, Susana Montero, Ignacio Aedo (Pearson 2005)

Seguridad y Comercio en el Web
Simson Garfinkel, Gene Spafford (O'Reilly 1999)

Claves Hackers de Sitios Web
Mike Shema (McGraw-Hill 2004)

Hackers Edición 2009
Mª Teresa Jimeno, Carlos Míguez, Abel Mariano Matas, Justo Pérez (Anaya 2009)

No me hagas pensar: Una aproximación a la usabilidad web
Steve Krug (Prentice Hall 2006)

Ingeniería de la web y patrones de diseño
Paloma Díaz, Susana Montero, Ignacio Aedo (Pearson Prentice Hall 2005)

Rediseño y desarrollo de sitios web
Kelly Goto, Emily Cotler (Anaya 2005)

La biblia de Apache 2
Mohammed J. Kabir (Anaya 2003)

Manual de estilo web
Patrick J. Lynch, Sarah Horton (Gustavo Gili 2004)

Una guía para la realización y supervisión de PFC en el ámbito de la web
Félix Buendía García (UPV 2008)

Introducción a las organizaciones y sus sistemas de información
Andrés Boza, Llanos Cuenca, José Onofre, Juan Vicente Oltra, José María Torralba (UPV 2007)

Operaciones con bases de datos ofimáticas y corporativas
Fco. Javier Martín (Ra-Ma 2004)

Optimización de sitios web
Andew B. King (Anaya Multimedia 2003)

Curso de CSS
Christopher Schmitt (Anaya Multimedia 2010)

Usabilidad: diseño de sitios Web
Jakob Nielsen (Prentice Hall 2000)