UNIVERSIDAD DE GRANADA

Tecnologías Web

3º Grado en Ingeniería Informática

Recomendaciones

para abordar el desarrollo de un sitio web

© Prof. Javier Martínez Baena
Dpto. Ciencias de la Computación e I. A.
Universidad de Granada

DECSAI

Departamento de Ciencias de la Computación e Inteligencia Artificial

Recomendaciones para el diseño de un sitio web

1. Introducción

En este documento se hacen algunas recomendaciones sobre cómo abordar la realización de un sitio web que le serán de utilidad para las prácticas de la asignatura. En particular se establece un orden para el desarrollo de las distintas tareas a llevar a cabo. Este orden se puede alterar e incluso se pueden hacer varias tareas en paralelo. Se propone un enfoque incremental, en el que cada etapa aporta algo nuevo al sitio y, más o menos, queda terminado antes de comenzar la siguiente etapa. La metodología que se propone es informal y únicamente pretende servir de guía. En ningún caso es una metodología formal ni, posiblemente, la mejor forma de abordar un proyecto web.

2. Diseño del sitio

Lo primero que debe hacer para construir un sitio web es un diseño que incluya todas las páginas del sitio. Estos diseños se conocen como wireframes / prototipos / mockups en función del nivel de detalle o de la funcionalidad que incluyan. Se pueden hacer sobre el papel o usando alguna herramienta software, por ejemplo Pencil, que es open source y multiplataforma (http://pencil.evolus.vn/).

Durante esta etapa de diseño conseguirá una visión aproximada del sitio y, a la vez, le permitirá ir pensando sobre cómo se pueden ir implementando algunas características o detectando detalles que inicialmente no tenía previstos. En un proyecto real le permitirá también intercambiar información con el cliente ya que este, a partir de los bocetos, se hará una idea más clara de cómo podría quedar el producto que le ha encargado.

3. Maquetación

La maquetación consiste en hacer el diseño gráfico del sitio: posicionar cajas de texto, imágenes, menús, etc. Para esta tarea es fundamental tener en cuenta que todas las páginas del sitio deben presentar una interfaz homogénea. Eso se conseguirá haciendo que todas usen una misma hoja de estilo CSS. Al finalizar esta etapa debe tener una hoja de estilo CSS (pueden ser uno o varios ficheros dependiendo de la complejidad) y un documento HTML que le servirá de plantilla para las distintas páginas del sitio.

Para desarrollar la hoja de estilo es importante que intente pensar en clases CSS genéricas que le permitan aplicar determinados estilos a diferentes elementos de la página. Es decir, debe intentar crear reglas CSS lo más genéricas posibles que se puedan reutilizar en múltiples elementos. Esto evitará que tenga demasiadas reglas CSS y que la construcción y mantenimiento del sitio sea más simple.

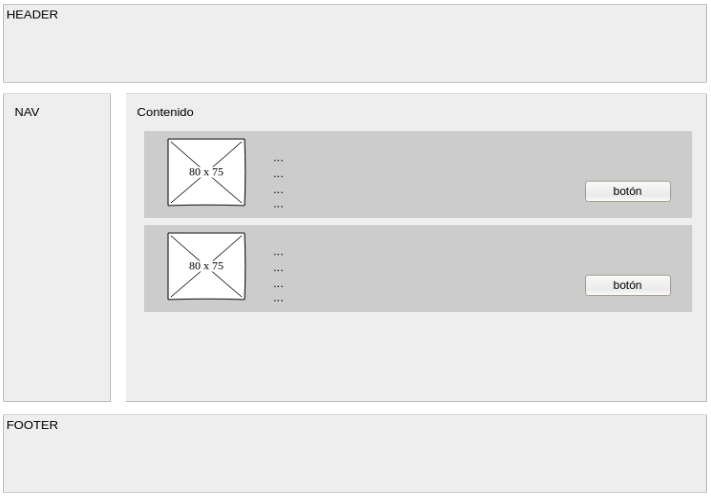
Para comenzar, puede elegir alguna de las páginas que ha obtenido en la etapa de diseño, a ser posible alguna que incluya elementos de distinto tipo representativos del sitio (listas, tablas, etc.). Su objetivo ahora es realizar la implementación en HTML+CSS de dicha página.

En este punto no debe detenerse a pensar sobre otras tecnologías como PHP, JavaScript, bases de datos, etc. Todo eso se incorporará más adelante. Céntrese únicamente en diseñar el aspecto de una página cualquiera de su sitio. Más adelante se añadirán las citadas tecnologías.

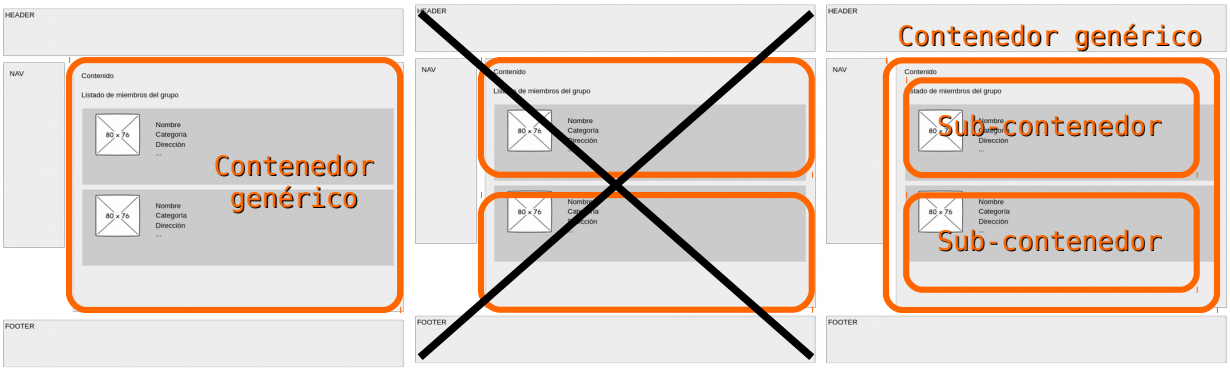
Sugerencia: si hay elementos HTML que no aparecen en la página que ha elegido para comenzar el diseño, puede añadir algunos (aunque su lugar sea otra de las páginas), de esa forma podrá incorporar las clases y reglas CSS que controlan su aspecto. Por ejemplo, si la página no contiene ninguna tabla para mostrar datos pero alguna otra página sí contendrá tablas, puede añadir ese elemento para darle formato (aunque luego se elimine de esta página en concreto). Así evitará estar editando en paralelo varios ficheros HTML para crear las reglas de estilo.

Antes de abordar la construcción del fichero CSS es importante tener un diseño de la estructura jerárquica de elementos HTML. Algunos de ellos serán comunes a todas las páginas y otros no. En su diseño HTML deberá tener en cuenta esto para crear todas las páginas del sitio de una forma

simple cambiando solo alguna parte respecto de la página plantilla que está desarrollando. Para deducir la estructura jerárquica del documento HTML debe basarse en los wireframes/mockups que ya debería tener hechos. Debe prestar especial atención a patrones que se repitan en todas las páginas. Por ejemplo, suponga que está creando una página con un aspecto similar al siguiente:



En este sitio todas las páginas mantienen el encabezado, la barra de navegación y el pie de página. Únicamente cambiará, de unas a otras, la zona de contenidos. Puede ver que dicha zona contiene múltiples elementos. Cuando cree el documento HTML, debería crear un contenedor para esa zona de contenidos y dentro de él añadir todos los elementos que necesite.



De esta forma, será fácil posicionarlo en relación con el resto de elementos de la página. Tenga en cuenta que posicionar un elemento respecto de otros es relativamente simple. Sin embargo, si en lugar de un elemento tiene varios, su posicionamiento respecto al resto de elementos se puede complicar bastante.

Aunque los estándares actuales de CSS proveen de algunas herramientas que le pueden ayudar a la organización de los elementos de la página (`flexbox`, `grid`, `table`, etc.), en este documento se usará una técnica muy básica para el diseño que solo precisa del uso de elementos `float`. Este conocimiento básico del funcionamiento de CSS le será muy útil para el diseño de páginas más complejas, incluso para aquellas que usan las citadas herramientas de más alto nivel.

Como sabe, la disposición normal de elementos de tipo `block` en HTML se hace apilándolos en el orden en el que van apareciendo en HTML. Cuando tenemos distribuciones multicolumna debemos usar alguna herramienta CSS que altere de alguna forma ese comportamiento "por defecto". Por tanto, partiendo de esta idea, podemos comenzar nuestra tarea identificando partes de la página que se apilan una tras otra o, dicho de otra forma, haciendo particiones horizontales de la página. En el caso anterior podríamos percibir tres partes:

1. Encabezado

2. Menú de navegación y contenidos
3. Pie de página

En el caso de la segunda parte tenemos un elemento horizontal con dos columnas.

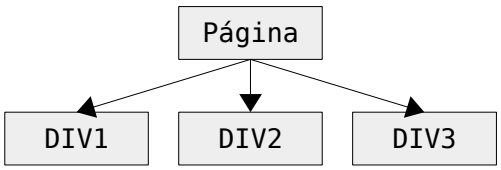
Vamos a hacer un ejemplo sobre una página real: una versión antigua de la página principal de la UGR. En ella se pueden apreciar multitud de elementos que, claramente, tienen aspecto de cajas rectangulares. La cuestión que surge es cómo organizarlos o anidarlos en el código HTML para conseguir esa disposición de una forma sencilla.

En una primera división horizontal separamos las grandes zonas de la página:

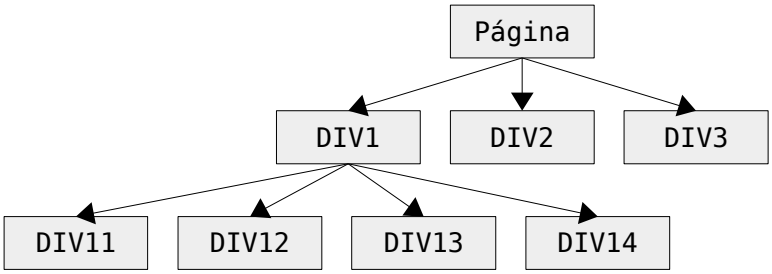
- Cabecera
- Contenidos (o cuerpo)
- Pie

Normalmente la información de cabecera y pie de página se mantienen en todas las páginas del sitio.

El cuerpo de la página es lo que habitualmente cambia de página a página. En este ejemplo el cuerpo contiene también un menú de navegación a la izquierda y una barra de anuncios a la derecha. Aunque el menú y los anuncios no son realmente el cuerpo del documento (es decir, la parte de contenidos propiamente dicha), se ha hecho un agrupamiento (`div`) de todos estos bloques porque si hubiésemos hecho tres bloques diferentes, estaríamos mezclando particionamiento horizontal con vertical, que es más difícil de maquetar. De esta forma, colocar los tres bloques (`DIV1`, `DIV2`, `DIV3`) uno a continuación de otro es trivial. Observe que, aunque en este ejemplo siempre usamos elementos `div`, estos podrían ser de cualquier otro tipo (`header`, `footer`, etc).



Siguiendo con la misma estrategia de división horizontal, procedemos a realizar la misma tarea para cada contenedor. La cabecera (`DIV1`) se puede dividir en 4 bloques horizontales. De nuevo la maquetación de estos elementos es trivial (por defecto los elementos `div` se colocan uno tras otro en vertical, que es justo lo que necesitamos).



Intentamos seguir con esta estrategia para cada nuevo bloque:

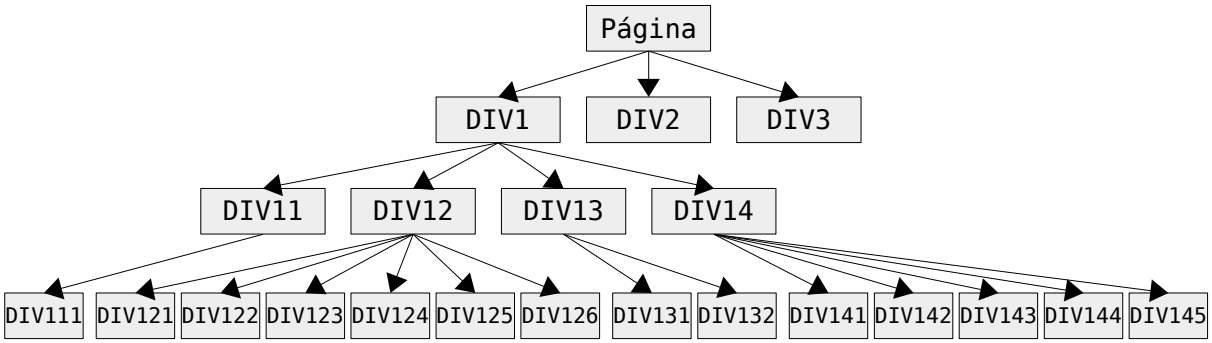
`DIV11`: en este caso no parece haber más divisiones horizontales por lo que analizamos si existen bloques de contenidos diferentes, es decir, agrupamientos de elementos que se posicionan de forma conjunta respecto a `DIV11`. Podemos ver que a la derecha hay un texto ("buscar"), una caja de entrada (`input`) y una imagen en forma de lupa. Esos tres elementos se pueden agrupar en un bloque para posicionarlos de forma conjunta. El otro elemento de la caja `DIV11` es una imagen de fondo.

`DIV12`: contiene una serie de items que pueden variar en contenido (texto o imágenes). Como todos parecen distribuirse de forma homogénea, se podrían incluir como elementos de una lista o como nuevos bloques (`div`) para posicionarlos todos de manera homogénea en horizontal (se puede usar la propiedad `float`).

`DIV13`: aquí podemos ver de nuevo una distribución similar a la del bloque `DIV11`, es decir, una imagen de fondo y un bloque de elementos a la derecha. Otra posibilidad es considerar dos bloques diferentes: una imagen a la izquierda y un bloque de elementos a la derecha. Ambos son esquemas simples de implementar.

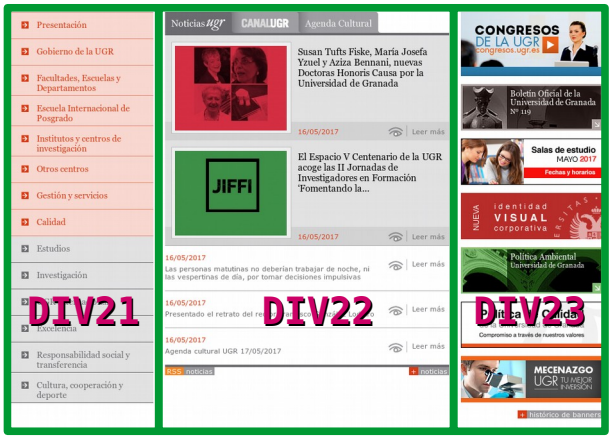
`DIV14`: el esquema de este bloque es similar al del bloque `DIV12`, se trata de una secuencia de elementos distribuidos horizontalmente de forma homogénea por lo que podríamos implementarlos como una lista o como bloques div.

Llegados a este punto la jerarquía del documento HTML es la siguiente:

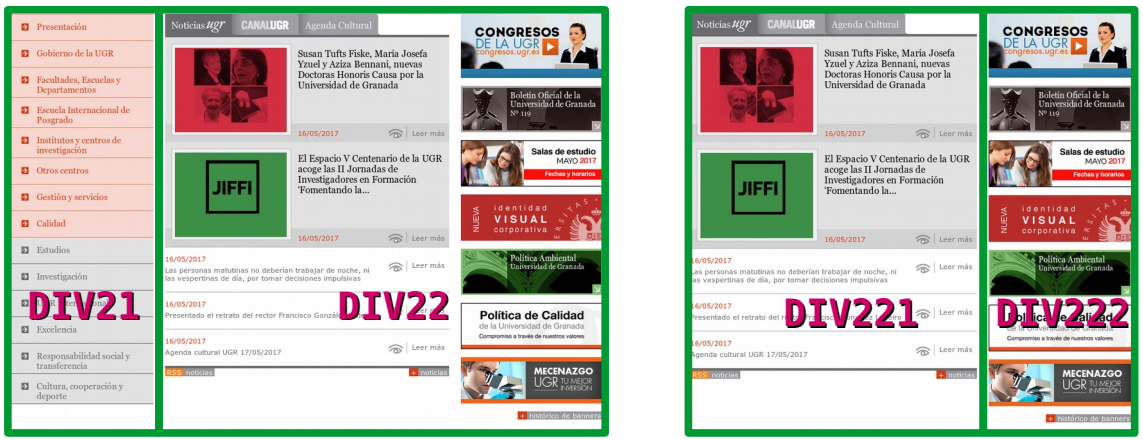


Para seguir con la maquetación de la zona de contenidos (`DIV2`) aplicaremos un esquema idéntico. En este caso tenemos tres columnas: menú de navegación a la izquierda, contenidos en el centro y anuncios laterales a la derecha.

Una posibilidad es disponer tres bloques div al mismo nivel en la jerarquía. No sería difícil colocarlos con anchos fijos o proporcionales o bien colocando los de los laterales con `float` y jugando con los márgenes para situar el central.



Otra posibilidad es seguir con el sistema de división binaria como vemos en la figura inferior.



Ahora en cada nueva caja aplicamos la misma idea: primero veremos si es factible una división horizontal y, si no, intentamos hacer la división vertical. En la siguiente figura se ilustra una posible partición de la zona de contenidos (no completa):

Se han representado con colores los elementos de los distintos niveles en la jerarquía (anidamientos): verde para el primer nivel, azul para el segundo, rosa para el tercero y amarillo para el cuarto. La imagen solo pretende ilustrar la idea ya que el particionamiento no se ha hecho completamente.

Es importante que los agrupamientos de items que se hacen en cada nivel de la descomposición se hagan de acuerdo a la semántica del contenido. Por ejemplo, la descomposición de primer nivel en la que salen tres bloques (cabecera, cuerpo y pie) también podría haberse hecho poniendo al mismo nivel en la jerarquía los bloques `DIV11`, `DIV12`, `DIV13`, `DIV14`, `DIV2`, `DIV3`. Sin embargo, esta descomposición está poniendo a un mismo nivel bloques que conceptualmente no lo son: `DIV11` es una parte de un encabezado mientras que `DIV3` es un pie de página completo. Técnicamente es

viable y se puede conseguir el mismo resultado, pero esto podría complicar futuros cambios en el diseño de la página.

Al finalizar esta etapa debe tener:

- Un fichero de estilo CSS que incluye reglas para todos (o casi todos) los elementos que puedan aparecer en tu sitio web.
- Un fichero HTML con una de las páginas de tu sitio más o menos finalizada (a falta de añadir información dinámica real).

4. Creación de un sitio web estático

Ahora que tenemos una página con un aspecto más o menos acabado, pasamos a crear el resto de páginas. Para ello tenemos varias alternativas:

- Usar funciones PHP para crear las distintas partes de la página. Cada función hará `echo` del código HTML necesario. Por ejemplo, una función `HTMLcabecera()` hará `echo` de los tags `DOCTYPE`, `head`, ...
- Usar plantillas para las distintas partes de la página. Se creará pseudo-código HTML para cada parte de la página y se guardará en un fichero. Al crear la página, habrá que ir haciendo `echo` del pseudo-código de esas plantillas haciendo las sustituciones correspondientes. Por ejemplo, una plantilla podría tener el código de la cabecera, en la que el título de la página (`title`) podría ser una variable a sustituir. El código PHP debería coger la plantilla, sustituir esa variable por el valor adecuado y hacer `echo` del resultado (HTML).
- Usar HTML para las distintas partes de la página. Aquellas partes de la página que sean idénticas para todas las páginas se pueden guardar en un fichero .html y hacer `include` de ese fichero para enviar ese código.
- ...

En este ejemplo usaremos funciones PHP para generar cada parte de la página. Se puede "trocear" el código de la página que se ha obtenido en el paso anterior (diseño) e ir modularizándolo en forma de funciones. El resultado será la misma página pero esta vez generada haciendo llamadas a esas funciones PHP.

Una vez esté hecha esa página con PHP (equivalente a la original), hay que decidir cómo se van a generar las demás:

- La página creada (por ejemplo `index.php`) será la única del sitio accesible por URL (luego se pueden añadir otras para tareas más específicas). Es decir, el uruario del sitio siempre usará la URL http://servidor.com/path/index.php para ver cualquier página de nuestra aplicación. De esta forma, al ejecutar la página, se debe añadir en la URL un query string indicando qué función de la aplicación se desea visitar. Las URL del sitio web serían similares a estas:
  http://void.ugr.es/pepito1617/index.php?p=inicio para ir a la página de inicio
  http://void.ugr.es/pepito1617/index.php?p=proyectos para ver un listado de proyectos
  En `index.php` habrá una estructura condicional que mostrará un contenido u otro en función de la variable `$_GET['p']`
- Otra opción es replicar la estructura de esa primera página en otros ficheros (uno por cada nueva página) cambiando el código PHP que muestra la zona de contenidos principales. Esto es una variante de la técnica copiar/pegar por lo que no se recomienda.

Se recomienda la primera opción.

De cara a la implementación en PHP es recomendable también que se mantengan ordenados los ficheros. Además del código de `index.php` o de las páginas principales, se necesitarán ficheros auxiliares PHP en los que se almacenan todas las funciones que se usan para generar el sitio:

- funciones que generan código HTML (cabeceras, pie de página, etc)
- funciones para hacer tareas auxiliares
- funciones para acceder a la BBDD (más adelante)
- ...

Se pueden agrupar las funciones en distintos ficheros según sea su cometido. Por ejemplo, podría

haber un fichero en el que estén almacenadas todas aquellas funciones que sirven para enviar código HTML más o menos estático relativo a la maquetación del sitio, otro fichero para las funciones que muestran formularios, otro fichero para las funciones que acceden a la BBDD, etc.

Al finalizar este punto debería estar operativa una versión del sitio con todas las páginas. Algunas de ellas no mostrarán nada puesto que todavía no hay BBDD y, por tanto, no hay información que ofrecer.

5. Diseño de la BBDD

En este punto debe plantearse hacer el diseño de la BBDD. Necesitará un modelo E-R y crear la BBDD usando, por ejemplo, phpMyAdmin. Se pueden rellenar algunos datos de prueba en algunas tablas.

Realmente, el diseño de la BBDD se puede hacer antes y/o en paralelo con el desarrollo de las páginas HTML+CSS o PHP. Pero hasta que no llegue a este punto no lo necesitará realmente para seguir avanzando en la construcción de la aplicación.

6. Prueba de accesos a la BBDD desde PHP

Una vez creada la BBDD elegimos alguna de las páginas del sitio que muestran información de la BBDD para probar la conexión con la BBDD. Para ello comenzamos creando el fichero PHP que contiene las credenciales de conexión a la BBDD.

Para los accesos a la BBDD puede optar por:

- Usar la interfaz mysqli estándar. Asegúrese de sanear adecuadamente los datos enviados a la BBDD (`mysqli_real_scape_string`).
- Usar sentencias preparadas
- ... usar alguna otra alternativa de acceso a BBDD (PDO, ...)

Si agrupamos en un fichero PHP todas aquellas funciones que hacen accesos a la BBDD (conexión, consultas, inserciones, etc.) será más fácil, por ejemplo, cambiar la API para acceso a BBDD o resolver algún problema que se haya detectado y que afecte a varios módulos de acceso a BBDD. Es buena idea separar las tareas de acceso a la BBDD de las tareas que controlan la lógica de la aplicación.

7. Módulo de autenticación de usuarios

Una vez probada la conexión con BBDD, podemos empezar añadiendo el código o módulo de autenticación de usuarios. Crearemos algún usuario de forma manual en la BBDD que nos permita hacer pruebas. Para esta tarea haremos uso de sesiones PHP siguiendo un esquema similar al que se ha explicado en clase.

Debe asegurarse de:

- Comprobar que según el tipo de usuario (no identificado, identificado, administrador, ...) los menús generados en la página son diferentes (cada tipo de usuario puede hacer distintas tareas en el sitio web).
- Que cada usuario solo pueda ver aquellas páginas para las que tiene permiso. No basta con mostrar un menú de navegación dependiendo del tipo de usuario y que oculte algunas páginas a las que no debería acceder puesto que el usuario siempre puede escribir directamente la URL en la barra de direcciones del navegador.
- Algunos de los ficheros PHP que tenga en su aplicación no serán páginas como tales sino que contendrán funciones de apoyo o segmentos de código que le ayuden a crear las páginas del sitio. Piense, de nuevo, que el usuario podría escribir la URL de acceso a esos ficheros y que son un peligro potencial si no se protege el acceso. Puede proteger esos ficheros colocándolos en carpetas protegidas por el servidor o bien protegerlos con código PHP que solo permita su ejecución si es incluyéndolos en otras páginas (y no ejecutándolos mediante su URL).

En relación con la autenticación de usuarios, tenga en cuenta que no es buena idea almacenar las claves en plano en la BBDD. Lo que se hace habitualmente es almacenarlas cifradas de manera que si un atacante accediese a la BBDD no se pueda hacer con ellas. Además, se cifran con métodos no reversibles, es decir, que permitan cifrar un texto pero no descifrarlo. En caso de que un atacante consiga una clave cifrada sería imposible obtener la clave en plano. Más información:

- https://www.sitepoint.com/password-hashing-in-php/
- https://alias.io/2010/01/store-passwords-safely-with-php-and-mysql/

8. Crear código PHP para accesos a la BBDD

Una vez probada la BBDD ya se pueden realizar el resto de páginas que hacen consultas, inserciones o borrados en la BBDD.

Además de sanear los datos enviados a la BBDD para hacer consultas, inserciones, etc con `mysqli_real_scape_string` (si no se hacen consultas preparadas) también debe tener especial cuidado con sanear los datos que provienen de la BBDD o de formularios y que se incluirán en el código HTML (`htmlentities`, `htmlspecialchars`)

9. Añadir funcionalidad JavaScript

En este punto la aplicación debería ser completamente funcional. Ahora podemos abordar una mejora en la usabilidad del sitio añadiendo validación de datos de formularios en el lado del cliente con JavaScript.

Hasta ahora, cuando se accede a un formulario y se pulsa el botón de enviar, los datos son recibidos por un script PHP y este es el encargado de validarlos. En caso de que haya datos incorrectos el script muestra de nuevo el formulario incluyendo algún tipo de información que alerta al usuario de los posibles errores. Además, el formulario recupera la información que envió previamente el usuario para evitar que tenga que escribirla de nuevo (sticky forms). Este proceso sigue siendo necesario, es decir, la validación de datos en el lado del servidor es obligatoria ya que el script que los recibe nunca tendrá garantías de que hayan sido validados en el cliente.

El problema que presenta este modelo es que cada error que comete el usuario al introducir un dato requiere de una conexión al servidor, el envío de datos, la validación y una respuesta del servidor ... tarea que consume tiempo y ancho de banda. Lo que se pretende hacer con la validación en el lado del cliente es evitar ese consumo innecesario de tiempo y ancho de banda ya que con JavaScript podemos comprobar que los datos son válidos antes de enviar el formulario.

Además, al hacer la validación con JavaScript se puede mejorar la información que se presenta al usuario sobre los errores cometidos, siendo más fácil o cómodo insertar mensajes de error cercanos al lugar donde se ha cometido el fallo.

10. Inclusión de código AJAX

Si en el sitio hay páginas que pueden hacer uso de AJAX este es un buen momento para abordar su implementación. Con esta tecnología podemos mejorar la usabilidad del sitio. AJAX permite que una página web cargada en el cliente, mediante JavaScript, establezca una conexión con el servidor para enviar o solicitar datos. Esta conexión no implica cargar una nueva página por lo que puede acelerar la aplicación y disminuir el consumo de ancho de banda.

Ejemplos de situaciones en donde se puede aplicar esta técnica:

- Al mostrar listados de datos. En clase se ha visto como hacer un listado paginado de una lista de registros obtenidos de la BBDD añadiendo una barra de navegación inferior (Primero, Anterior, Siguiente, Último) en la que cada etiqueta es un enlace con una URL a un script PHP al que se le pasa un query string con los datos del rango de registros que se desea visualizar. Cada vez que pulsamos uno de esos enlaces se carga una nueva página (incluyendo encabezados, menús, pie de página, etc). En realidad, lo único que deseamos modificar es la tabla de datos, no el resto de la página. Esto se puede conseguir haciendo una petición AJAX al pulsar uno de los botones. La petición es asíncrona: se hace la consulta al servidor (al script PHP), este la procesa (recupera los registros que deseamos de

la BBDD) y devuelve una respuesta (los registros recuperados). Cuando la página recibe la respuesta del script PHP, lo que hace es modificar el DOM borrando los nodos que ya no deseamos mostrar (los datos de la tabla actuales) e insertando nuevos nodos (los datos que acabamos de recibir). De esta forma, al pulsar los botones (o enlaces) del menú de paginación no cargamos la página completa; solo transferimos desde el servidor la información nueva que deseamos mostrar. Ni siquiera transferimos las etiquetas HTML para la nueva información ya que esto se genera con JavaScript.

- Al editar (o borrar) algún registro. Si tenemos un listado de datos con un botón de editar al lado de cada item, lo que hemos hecho hasta ahora es hacer una petición a un script PHP que muestra un formulario para editar los datos. Podríamos sustituir esa acción por una petición AJAX de forma que al pulsar el botón de enviar, en lugar de ejecutar el "action" del formulario, lo que hacemos es modificar el DOM con JavaScript para que se muestre el formulario en la misma página que estamos viendo y cuando hayamos hecho la edición del registro, en lugar de enviar el formulario, podemos hacer un envío de datos con AJAX. Este envío sería similar al que se hace con un envío de formulario normal, la diferencia es que tras hacer el envío seguiremos en la misma página esperando una respuesta del servidor en lugar de saltar a otra página (lo que implicaría mayor consumo de ancho de banda y tiempo de espera de carga de la página). Al recibir la respuesta del servidor (que podría ser un simple código indicando si la modificación se ha podido hacer o no) seguiremos estando en la misma página evitando así la carga de otra. En función del código recibido podríamos mostrar algún mensaje al usuario o no hacer nada.
- Formularios de búsqueda. De forma similar al ejemplo anterior de edición de un registro podemos usar AJAX para pedir registros de acuerdo a unos criterios de búsqueda y mostrarlos cuando lleguen en la misma página del formulario de búsqueda. O incluso ir actualizando la lista de resultados conforme se va rellenando el formulario.

Para implementar la técnica AJAX debe incluir el código JavaScript correspondiente y programar los correspondientes scripts PHP que procesen las peticiones. De cara a la organización del código, y en caso de que sean varios los scripts, se puede crear una carpeta específica para ellos.

Observe que estos scripts también deben ejecutar `session_start()` ya que normalmente requerirán que el usuario esté identificado. La estructura de estos scripts es diferente a la del resto de páginas del sitio ya que no están pensados para que el usuario los ejecute desde la URL del navegador. No generan código HTML sino que generan datos para atender la solicitud de JavaScript. Es frecuente que esos datos tengan formato JSON. JavaScript les puede pasar información a través del query string o a través de las cabeceras HTTP de forma que los scripts podrán hacer uso de las variables `$_POST` y `$_GET`.

Como se indicó al comienzo de este documento, esta guía le permitirá avanzar en el desarrollo de las prácticas siguiendo la temporización que marca el programa de la asignatura. En un proyecto real podría alterar el orden de determinadas tareas dado que supuestamente ya conoce todas las tecnologías implicadas en el desarrollo. Por ejemplo, la inclusión de código JavaScript de validación se puede hacer antes de la implementación del back-end o la previsión de páginas que incluyen tecnología AJAX se puede planificar antes.

Por otra parte, el esquema de diseño de página usando float tampoco es algo formal ni, posiblemente, sea la herramienta que utilice para el diseño global de una página profesional dado que existen alternativas mucho más potentes con un alto grado de implantación en los navegadores actuales (flex, grid, etc).