Capítulo 4

Diseño del sitio web

> Oh, so they have Internet on computers now!
>
> Homer Simpson

La creación de una página web es posible con la unión de diferentes factores necesarios para su funcionamiento: aquello que se ve y es visible para el usuario (el diseño web y el contenido) y aquello que no se ve pero que es igual o más importante. Todo junto forma una estructura virtual que tiene como función hacer llegar al usuario toda la información de manera adecuada y accesible.

A la hora de empezar a trabajar en el desarrollo de la página, el primer paso consiste en elegir qué herramientas de las disponibles van a ser empleadas: HTML o XHTML, PHP o JSP, JavaScript o Flash, etc. Aquí se presentan las elecciones tomadas en nuestro caso:

- **HTML**. Hypertext Markup Language es el lenguaje con el que se define el formato de los documentos hipertexto hospedados en servidores de la World Wide Web (WWW). Básicamente se trata de un conjunto de etiquetas (tags) que sirve para precisar la forma de presentación del texto y de otros elementos del documento HTML. El último estándar es el HTML 4.01 divulgado en febrero de 2001 por el consorcio W3C. Se empleará la especificación HTML 4.01 Strict.

- **CSS**. Cascading Style Sheets u Hojas de Estilo en Cascada es una tecnología que permite crear páginas web de una manera más exacta, usando formatos unificados, inclusión de márgenes, tipos de letra, fondos, colores, etc. Las

Hojas de Estilo en Cascada se escriben dentro del código HTML de la página web o en un archivo de extensión `css` enlazado al documento HTML.

- **JavaScript**. JavaScript es un lenguaje interpretado, es decir, que no requiere compilación, utilizado principalmente en páginas web, con una sintaxis semejante a la del lenguaje Java y el lenguaje C. Permite el desarrollo de interfaces de usuario mejoradas y páginas web dinámicas. Para interactuar con una página web se provee al lenguaje JavaScript de una implementación del Document Object Model (DOM)[1]. Al igual que ocurre con las Hojas de Estilo, podremos incluir el código directamente dentro de HTML o en un archivo externo, con extensión `js` en este caso.

- **PHP**. PHP es el acrónimo de Hipertext Preprocesor. Es un lenguaje de programación interpretado, diseñado originalmente para la creación de páginas web dinámicas. Es usado principalmente en interpretación del lado del servidor (server-side scripting). Es gratuito e independiente de la plataforma, con una gran librería de funciones y mucha documentación. Su independencia estriba en que existe un módulo de PHP para casi cualquier servidor web. Ésto hace que cualquier sistema pueda ser compatible con el lenguaje y significa que permite portar el sitio desarrollado en PHP de un sistema a otro sin prácticamente ningún trabajo. PHP se escribe dentro del código HTML.

4.1. Estructura del sitio web

Lo normal, es que el sitio web parta desde una página inicial, o home, desde la que podemos acceder, de forma jerárquica a todo el contenido del sitio, a través de hiperenlaces. Como primer paso creamos el archivo `index.php` donde incluiremos el código de esta página. Este archivo estará compuesto de tres partes:

- Una declaración del tipo DOCTYPE que debe hacer referencia a una de las diferentes definiciones del tipo de documento o DTD que existen, HTML 4.01 en el caso de esta página web. Esta declaración debe ser la primera línea de nuestro documento y es necesaria para decirle al navegador qué versión de HTML es la que se usa en la página. El modo de definirla es:

[1] El Document Object Model (DOM) es una interfaz de programación de aplicaciones (API) para documentos HTML y XML. Define la estructura lógica de los documentos y el modo en que se accede y manipula un documento.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/
    strict.dtd">
```

- El encabezado, encerrado por la etiqueta `head`, que contiene algunas de estas definiciones:
  - El título del documento: Describe brevemente sobre qué trata el documento.

    ```
    <title>Redes de Ordenadores</title>
    ```

  - Declaraciones de estilos: Grupo de definiciones de clases (hojas de estilos) usadas por otros tags para establecer características visuales.

    ```
    <link href="/css/redes.css" rel="stylesheet" type="text/css" />
    ```

  - Scripts: Conjunto de funciones declaradas dentro de la etiqueta `script` para proveer funcionalidad al documento.

    ```
    <script type="text/javascript" src="/js/func_varias.js"></script>
    ```

  - Declaraciones meta: Declaradas mediante la etiqueta `meta`, define atributos y valores personalizados para el documento.

    ```
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
    ```

- El cuerpo del documento, dentro de la etiqueta `body`, donde se encontrará todo el texto, imágenes, colores, gráficos, etc. a mostrar por el navegador web. La forma en la que es mostrado todo este contenido viene especificado por las hojas de estilo definidas en la cabecera.

Una vez elegido el DTD a emplear y definida la cabecera, será necesario plantearse el cómo estructurar el contenido de la página, el cómo mostrarlo. Como norma general todas las páginas mantendrán la misma estructura. Por ejemplo, tendrán los mismos logo o título, menú de navegación, pie de página, etc. Sólo iremos cambiando el contenido. Nos serviremos de un boceto (ver figura 4.1) como guía visual a la hora de escribir el código que defina la estructura (HTML) y el estilo (CSS) básico de estas páginas.

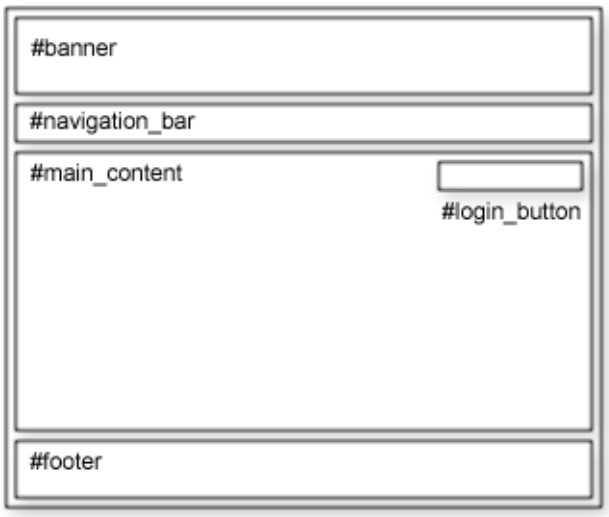

Figura 4.1: Maquetación de la página web

En lugar de crear una página distinta para cada enlace interno dentro de nuestra web, se ha optado por utilizar una única página (`index.php`) que funcionara como plantilla. Dentro de esta página única incluimos el siguiente código:

```
<!-- Contenido -->
<div id="main_content">
<noscript>
<div class="alert">La página que estás viendo requiere para su funcionamiento el uso de
     JavaScript. Si lo ha deshabilitado de manera intencionada, por favor vuelva a activarlo
     .</div>
</noscript>
<div id="msg"></div>
<?php
try {
  ob_start();
  if (isset($_GET['aplicacion'])) {
    include('include/aplicacion.php');
  } elseif (isset($_GET['turnos'])) {
    include('include/turnos.php');
  } elseif (isset($_GET['administrar'])) {
    include('include/administrar.php');
  } else {
    include('include/inicio.php');
  }
}
catch (Exception $excepcion) {
  ob_clean();
  if (Usuario::esAdmin()) {
    echo '<div class="error">'.(string) $excepcion.'</div>';
```

```
  } else {
    echo '<div class="error">Se he producido un error durante la carga de la página, pó
        ngase en contacto con el administrador.</div>';
  }
}
ob_end_flush();
?></div>
```

De esta manera y gracias al uso de variables URL y la función de PHP `include()` nos será posible cargar contenido del resto de las páginas dentro de la sección `#main_content`. Los bloques try y catch así como las funciones `ob_start()`, `ob_clean()` y `ob_end_flush()` nos permiten controlar el resultado a mostrar por pantalla en caso de producirse un error.

La única excepción a esta estructura será la página correspondiente a la aplicación, la cual contendrá la interfaz de usuario para la realización de las prácticas. Ésto es debido a que dados los requerimientos de esta página es necesario emplear un diseño diferente al resto.

Podemos destacar también las siguientes líneas de código dentro del archivo `index.php`:

```
<li<?php if (isset($_GET['inicio'])) echo ' class="active"'; ?>><a href="/">Inicio</a></li>
```

Nos permitirá mostrar de manera visual en que sección de la página nos encotramos desde el menú de navegacion

```
<li<?php if (isset($_GET['aplicacion'])) echo ' class="active"'; ?>><a href="" onclick="
    ventanas.abrir('/aplicacion/', 'Aplicaci\'on', 'height=715,width=960,resizable=no,
    menubar=no,location=no,scrollbars=no,toolbar=no,status=no'); return false;">Aplicaci\'
    on</a></li>
```

Abrirá la página correspondiente a la aplicación en una nueva ventana con los párametros indicados.

```
<div id="login_button"><ul>
  <?php if (Usuario::logged()) {?>
  <li><a href="" onclick="usuario.logout(); return false;">Logout</a></li>
  <?php } else {?>
  <li><a href="" onclick="usuario.login(); return false;">Login</a></li>
  <?php }?>
</ul></div>
```

Mostrará uno u otro botón en función de si el usuario se encuentra actualmente identificado en la página.

Una vez finalizada la creación de esta página principal podremos centrarnos en escribir los archivos con el contenido de las diferentes secciones que incluiremos dentro de nuestro sitio web:

- **Inicio.** Su contenido, definido en el archivo `inicio.php`, incluirá información general para el usuario sobre el funcionamiento de todo el sistema de prácticas. En caso de estar definidas las fechas para el acceso a la reserva de turnos o a la aplicación, estas se presentarán también dentro de esta sección.
- **Aplicación.** No es una sección propiamente dicha dentro de nuestro sitio web. Al hacer click en el enlace correspondiente en el menú de navegación se abrirá una nueva ventana externa donde se cargara la interfaz gráfica de usuario de la aplicación web. El funcionamiento y estructura tanto de la interfaz como de la lógica de la aplicación se detallan en apartados posteriores (ver apartado 5).
- **Turnos.** Es en esta sección donde se presentaran los turnos definidos por el administrador y los usuarios podrán realizar la reserva de los mismo. El código de esta sección lo encontraremos en el archivo `turnos.php`.
- **Administración.** Esta sección será solo accesible por aquellos usuarios con derechos de administración. En la misma se presentan un conjunto de formularios a través de los cuales el administrador podrá añadir, editor o borrar: cuentas de usuario, turnos, reservas y fechas de acceso. El administrador también podrá acceder al archivo de log a través de esta sección. Todo el código asociado a esta sección se encuentra en definido en los siguientes archivos: `administrar.php`, `admin_acceso.php`, `admin_reservas.php`, `admin_turnos.php` y `admin_usuarios.php`.

Todos estos archivos no serán accedidos directamente desde un navegador web sino, que como ya se ha comentado, serán incluidos dentro de la página principal. Por esta razón los agruparemos todos dentro del directorio `include/` y se prohibirá el acceso mediante el archivo de configuración correspondiente del servidor web.

En cuanto al resto de archivo que crearemos para nuestro sitio web, no existe una regla exacta de cómo organizarlos, ya que depende de muchos factores: número de páginas, cómo queremos navegar entre ellas, cómo se organiza el contenido, etc. En nuestro caso se han situado en el directorio raíz aquellas páginas a las que el usuario puede acceder de manera directa guardando el resto de archivos en subcarpetas según su tipo o funcionalidad.

- `index.php` : Dentro de esta página vamos a definir las secciones principales de la página: cabecera, menú de navegación, contenido principal, pie de página, etc.

- `aplicacion.php`: página a través de la cual se realizan las prácticas. Haciendo uso de una interfaz simple, permitirá al usuario alterar de manera controlada las configuraciones de los distintos elementos de la subred de prácticas reservada.

- `include/` : Directorio donde se encuentran los archivos con el contenido de las diferentes secciones de nuestro sitio a excepción de la correspondiente a la aplicación.

- `images/` : Carpeta con todos los elementos gráficos empleados en la web (`gif`, `jpeg`, `png`, etc.).

- `css/` : Donde se almacenan las hojas de estilo (CSS).

- `js/` : Usada para todos los archivos en JavaScript.

- `admin/` : Mientras que el resto de archivos se centran en la generación del código HTML, así como del aspecto gráfico de la interfaz del usuario, los scripts PHP contenidos en esta carpeta conforman el núcleo de la aplicación.

4.2. Comunicación cliente-servidor: AJAX

AJAX son las siglas de Asynchronous JavaScript And XML. No es un lenguaje de programación sino un conjunto de tecnologías (HTML-JavaScript-CSS-XML) que nos permiten hacer páginas de Internet más interactivas o RIAs. La característica fundamental de AJAX es permitir actualizar parte de una página con información que se encuentra en el servidor sin tener que refrescar completamente la página. De modo similar podemos enviar información al servidor. El uso de AJAX nos permitirá aumentar la interactividad, velocidad y usabilidad de la web.

Cuando el usuario ejecute alguna acción que requiera una respuesta del servidor se realiza una llamada asíncrona al servidor web utilizando el objeto `XMLHhttpRequest` de JavaScript para transferir los datos necesarios y recibir la respuesta a la petición. La respuesta es procesada en el cliente y se actúa sobre la página según proceda.

Figura 4.2: Comparación entre el modelo tradicional y el modelo de AJAX

La flexibilidad del objeto `XMLHttpRequest` permite el envío de los parámetros por otros medios alternativos a la tradicional 'query string'. Es posible realizar una petición al servidor enviando los parámetros en formatos como XML o JSON. En nuestro caso haremos uso de JSON al tratarse de un formato mucho más compacto y ligero que XML, además de ser mucho más fácil de procesar en el navegador del usuario.

Para lograr una mayor consistencia en la comunicación se ha optado por emplear siempre el mismo formato en la respuesta enviada por el servidor:

```
json_encode(array('stdout' => '',
                'stderr' => array('error' => '', 'info' => array())))
```

Como es posible observar, la respuesta estará compuesta por `stdout`, donde se devolverá la información oportuna en ausencia de errores, y por `stderr`, donde se especificarán en caso de producirse un error el tipo y la información asociada al mismo. Los tipos de errores que se contemplan son tres: 'validacion', en caso de error en la información enviada por el cliente, 'apliacion', cuando se produce un error no crítico durante el manejo de la aplicación web, y 'excepcion', cuando la petición genera un error grave en el servidor.

Es necesario que toda la información recibida en el servidor sea validada por

motivos de seguridad. Para tal fin creamos la clase `Validacion` (definida en el archivo `func_comunes.php`) encargada de validar todos y cada uno de los valores enviados por el cliente.

4.3. Scripts para la interacción con el usuario

Dejando a un lado la aplicación, cuando un usuario accede al sitio web debe de ser capaz de realizar una serie de acciones tales como identificarse o reservar un turno. Este apartado está dedicado a explicar el funcionamiento de los script, tanto del lado del cliente (JavaScript) como del lado del servidor (PHP), encargados de esta tarea.

Del lado del cliente nos encontramos con la clase `usuario` definida en el archivo `func_varias.js`. En ella se contemplan las cuatro acciones que puede llevar a cabo el usuario: iniciar la sesión, cerrar la sesión, reservar un turno y cancelar una reserva previa. Cuando el usuario quiera realizar cualquiera de estas acciones se enviará una petición al servidor y se procesará la respuesta recibida. En función de esta respuesta, se hará uso de la clase `mensaje`, también definida en este archivo, para mostrar mensajes de información, alerta o error al usuario.

En el servidor se encuentra el archivo `usuario.php` el cual evalúa la petición y genera la respuesta pertinente. Para ello hace uso de la clase `Usuario` definida en el archivo `func_comunes.php`. En esta clase se definen todos los métodos relacionados con las acciones que podrá llevar a cabo un usuario o el acceso del mismo al sistema, lo cual comprende:

- Inicio y fin de la sesión del usuario (`login()` y `logout()`) así como la comprobación del estado actual de la sesión (logged()).
- Comprobación de los permisos del usuario (`esAdmin()`).
- Reserva de un turno y cancelación del mismo (`reservar()` y `cancelarReserva()`).
- Observar si un usuario tiene acceso a la reserva de turnos o a la aplicación web (`accesoReserva()` y `accesoApliacion()`).
- Actualizar el estado de una red en función de si un usuario se encuentre o no haciendo uso de la misma (`accedeRed()` y `abandonaRed()`).

4.4. Scripts para la administración del sistema

A través de nuestro sitio web hemos dotado al administrador la capacidad de llevar a cabo un conjunto de tareas tales como la administración de las cuentas de usuario, la creación y edición de turnos y reservas, el establecimiento de las diferentes fechas de acceso o el acceso al log de la aplicación. En este apartado se explicará de manera general el código creado para tal fin.

Cuando un usuario con derechos de administración inicie una sesión, podrá acceder a un conjunto de páginas definidas dentro de la sección correspondiente en el menú de navegación. Estas páginas contiene los formularios necesarios para llevar a cabo las tareas antes mencionadas.

Además de la sección de administración, también se carga en el cliente el archivo `administrar.js`. Este archivo contiene la definición de una única clase que contiene los métodos para mostrar los formularios por pantalla, enviar la petición al servidor y procesar la respuesta mostrando por ejemplo los errores contenidos en el formulario.

Para el procesamiento de las peticiones del administrador se han creado una serie de clases definidas en el archivo `administrar.php`. Para simplificar el tratamiento de las peticiones todas se procesan haciendo uso de una interfaz común. Para este fin se ha creado una clase abstracta denominada `Administrar` de la cual heredarán las clases encargadas del tratamiento de cada uno de los distintos formularios: `AdministrarAcceso`, `AdministrarTurnos`, `AdministrarUsuarios` y `AdministrarReserva`. Todas estas clases deberán sobrescribir los métodos `add()`, `edit()`, `del()` y `delall()`.

Cuando el administrador envía el formulario, el código situado en el archivo (`proc_form.php`) entra en acción. Determina primero que formulario es el que ha recibido y obtiene una instancia de la clase correspondiente para su procesado. Una vez instanciada la clase, solo resta realizar la llamada al método correspondiente a la acción requerida por el administrador y devolver la respuesta al cliente.

Antes de realizar cualquier modificación en la base de datos se llevan a cabo todas las comprobaciones necesarias para evitar por ejemplo el solapamiento de turnos o la eliminación de la cuenta del administrador principal. Además, en el caso de eliminar una cuenta de usuario, se eliminarán todos los archivos almacenados en el servidor asociados a dicha cuenta.