UNIVERSIDAD DE GRANADA

Tecnologías Web

3º Grado en Ingeniería Informática

Guión de prácticas
Lenguaje PHP

Generación dinámica de sitios web

1. Objetivo....2
2. Recetas de cocina....2
3. El patrón MVC....6
4. Entrega de la práctica....8

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

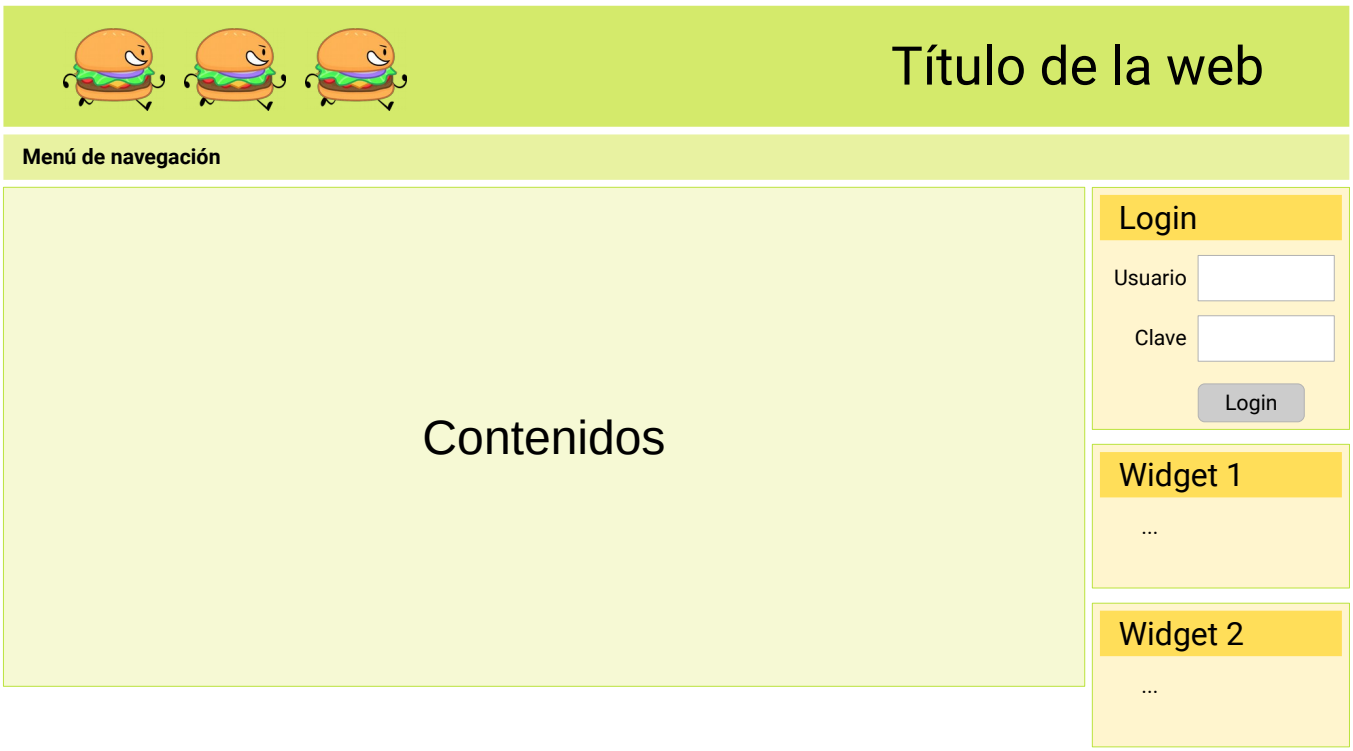
DECSAI
Departamento de Ciencias de la Computación e Inteligencia Artificial

1. Objetivo

El objetivo de esta práctica es implementar una aplicación web dinámica con PHP.

2. Recetas de cocina

En este ejercicio deberá crear una infraestructura básica para desarrollar una aplicación web que permita mantener un recetario de cocina. Dicha aplicación consistirá en un conjunto de páginas web con un marco común como el de la figura:



En donde se pueden apreciar estos elementos:

- Encabezado: título de la web, logotipo, ...
- Menú de navegación.
- Barra lateral con elementos variables como login, widgets con información relevante, anuncios, etc.
- Pie de página.

Además, hay una zona central para mostrar los contenidos específicos según la navegación del usuario. Dicha zona deberá poder mostrar la siguiente información:

- Página de inicio. Será una página inicial de entrada al sitio con, por ejemplo, algún mensaje de bienvenida.
- Página de contacto. Incluirá información sobre el autor del sitio y un formulario para enviar sugerencias o preguntas.
- Página con listado de recetas.

Puesto que todas las páginas del sitio tienen una estructura común, deberá usar PHP para evitar el copia/pega en los diferentes ficheros HTML que forman el sitio web. Deberá implementar una aplicación correctamente modularizada para poder generar todas las páginas del sitio de forma dinámica.

Aunque no se considera obligatorio en este ejercicio, puede ser interesante que la organización de

los ficheros y del código siga el patrón MVC (Modelo-Vista-Controlador). Esto no quiere decir que haya que usar un paradigma orientado a objetos necesariamente, también se puede usar el paradigma procedural. En cualquier caso, puede comenzar el ejercicio sin tener en cuenta este patrón estructural y, más adelante, refactorizar el código para adaptarlo a él.

Características adicionales:

- La aplicación debe mantener una base de datos de recetas sencilla.
- La página mostrará contenidos diferentes dependiendo de si el visitante está identificado en el sistema o no.

A continuación se especifícan con más detalle los requisitos de las diferentes páginas y elementos.

2.1. Menú de navegación

Será un menú de navegación horizontal. Debe implementar en PHP funcionalidad que permita que este menú se genere de forma dinámica para poder mostrar opciones distintas dependiendo de si el usuario que está visitando el sitio está identificado o no.

Si se trata de un usuario sin identificar, las opciones serán estas:

- Ir a la página de inicio.
- Ver un listado de recetas.
- Ir a la página de contacto.

Si se trata de un usuario identificado las opciones serán estas:

- Ir a la página de inicio.
- Ver un listado de recetas.
- Añadir receta nueva.
- Ir a la página de contacto.

2.2. Identificación de usuarios

La aplicación debe permitir que un visitante se identifique desde la caja de login. No hay que implementar ningún módulo de gestión de usuarios ni tener una tabla de usuarios en la base de datos. Únicamente se comprobará si el usuario "admin" con clave "clave" se identifica como tal o no y esto se puede hacer directamente en el código PHP.

Cuando el usuario admin se ha identificado, el comportamiento de la aplicación cambiará, permitiéndole realizar acciones extra e incluso modificando la apariencia de algunas páginas. Por ejemplo, un usuario admin podrá editar y borrar recetas.

Una vez identificado un usuario ya no tiene sentido que aparezca la caja de login. En su lugar aparecerá otra caja indicando que el usuario está logeado y un botón o enlace para desautenticarse.

2.3. Base de datos

Deberá construir una base de datos para alojar el listado de recetas. Solo necesitará una tabla con los siguientes datos:

- Título
- Autor
- Categoría
- Descripción
- Ingredientes
- Preparación

- Fotografía

La creación de la tabla se hará de forma manual desde alguna aplicación de administración a tal efecto (phpMyAdmin, MyWebSQL, ...).

2.4. Barra lateral

En la barra lateral se mostrará un cuadro para que el usuario pueda identificarse. Además, se podrán colocar algunos widgets para mostrar información de interés. Deberá incluir uno para mostrar el número total de recetas almacenadas.

2.5. Página de contacto

En esta página se mostrará información de contacto del autor y un formulario que permita enviar alguna pregunta o sugerencia. Deberá leer los siguientes datos:

- Nombre.
- Correo electrónico.
- Teléfono.
- Comentario.

Deberá implementar un script PHP que reciba la información y que:

- Valide los datos. Si alguno no es correcto deberá mostrar de nuevo el formulario procurando mantener el contenido de los campos que sí son correctos e indicando los errores cometidos. De esta forma el usuario tendrá la oportunidad de reparar el error y enviar de nuevo los datos (formulario sticky). Las restricciones de los datos son estas:
  - El nombre no puede estar vacío.
  - El email no puede estar vacío y debe ser un email válido.
  - El teléfono sí puede estar vacío pero en caso de contener datos deben corresponderse con un teléfono válido. Se deben poder indicar tanto prefijo nacional como internacional opcionalmente.
  - El comentario no puede estar vacío.
- Procese los datos. En caso de que los datos sean correctos simplemente se mostrará un mensaje informativo indicando que el comentario ha sido enviado y los datos que se han recibido. En realidad no se envía ni registra nada en el sistema, se trata solo de probar que la recepción es correcta.

2.6. Página con listado de recetas

Esta página tiene dos partes:

- Formulario de ordenación y búsqueda. Permitirá filtrar y ordenar las recetes del listado. Incluirá un campo de búsqueda de forma que solo se visualizarán las recetas que incluyan el texto de ese campo en el título. Incluirá un radiobutton para preguntar si se desea que el orden alfabético sea ascendente o descendente.
- A continuación habrá un listado con las recetas que cumplan el criterio del formulario anterior.

Cada receta debe mostrar tres botones:

- Uno para visualizar la receta.
- Uno para editar la receta. Este solo se mostrará si el usuario está identificado.
- Uno para borrar la receta. Este solo se mostrará si el usuario está identificado.

2.7. Operaciones CRUD sobre una receta

CRUD es el acrónimo de Create-Read-Update-Delete. Esencialmente se asocia a las operaciones tradicionales de mantenimiento de registros en las tablas de la base de datos (puede estar referido al mantenimiento conjunto de varias tablas si existe relación entre ellas).

2.7.1. Mostrar una receta

La receta se mostrará con este formato

y los botones dependerán de si el usuario está o no identificado.

2.7.2. Crear y modificar una receta

Para crear una receta deberá crearse un formulario que solicite los datos de la misma. Al enviar el formulario, el servidor comprobará que los datos son adecuados (la validación consistirá en comprobar que no hay campos vacíos). Si no lo son volverá a solicitarlos recuperando los que sí son correctos e indicando los errores detectados (formulario sticky).

En caso de que los datos sean correctos deberá solicitar la confirmación antes de hacer la inserción en la base de datos. Para ello debe mostrar un nuevo formulario que muestre la información recibida y que tenga un único botón para confirmar la operación. Puesto que esencialmente la información mostrada coincide con la solicitada en el formulario previo, podría reutilizarlo pero con los campos de entrada deshabilitados (de esa forma se ve el contenido pero no son editables).

Finalmente, tras la confirmación, se procederá a la inserción del registro en la base de datos y se mostrará un mensaje informando sobre el éxito (o no) de la operación. En esta información se

debe incluir, al menos, el título de la receta creada.

La operación de edición o modificación de una receta podrá reutilizar gran parte del código usado para crearla. La secuencia es similar salvo por dos detalles:

- En lugar del formulario vacío que se utilizó para empezar el proceso de creación, esta vez se mostrará el mismo formulario pero con los datos recuperados de la receta que se va a modificar.
- Al hacer la modificación en la base de datos, necesitamos hacer un UPDATE en lugar de un INSERT. Por ello, necesitamos saber la clave primaria de la receta que estamos editando y que deberíamos haber recuperado al comienzo del proceso.

Debe cuidar de que solo un usuario identificado pueda editar o insertar datos. Aunque puede hacerlo de varias formas, suponga que para crear una receta utiliza una URL como https://miservidor.com/recetas/crear.php. Si un usuario no identificado accede a esa URL debería obtener un mensaje informativo indicándole que no tiene permiso para hacer esa operación. Observe que esa URL puede obtenerse a partir de algún enlace ofrecido por la propia aplicación web (por ejemplo un enlace del menú de navegación) o puede ser tecleado por un usuario (aunque no se muestre en ninguna parte de la aplicación).

2.8. Generación de páginas dinámicas

Debe implementar código PHP que genere todas las páginas web del sitio de forma dinámica. Esto permitirá, entre otras cosas, evitar repetir código HTML estático en las distintas páginas del sitio. Deberá decidir la organización de ficheros del sitio web y el esquema de las URL para navegar por el sitio.

Puede hacerlo de varias formas:

- Una posibilidad es crear una página PHP para cada página del sitio, como por ejemplo:
  - `https://void.ugr.es/~tweb/index.php` (página de bienvenida)
  - `https://void.ugr.es/~tweb/listado.php` (listado de recetas)
  - `https://void.ugr.es/~tweb/contacto.php` (página de contacto)
- Otra posibilidad es crear una única página web, index.php, y usar el query string para determinar la acción a realizar. Por ejemplo:
  - `https://void.ugr.es/~tweb/index.php` (página de bienvenida)
  - `https://void.ugr.es/~tweb/index.php?acc=listado` (listado de recetas)
  - `https://void.ugr.es/~tweb/index.php?acc=contacto` (página de contacto)

En cualquier caso, el código deberá estar correctamente modularizado y organizado.

3. El patrón MVC

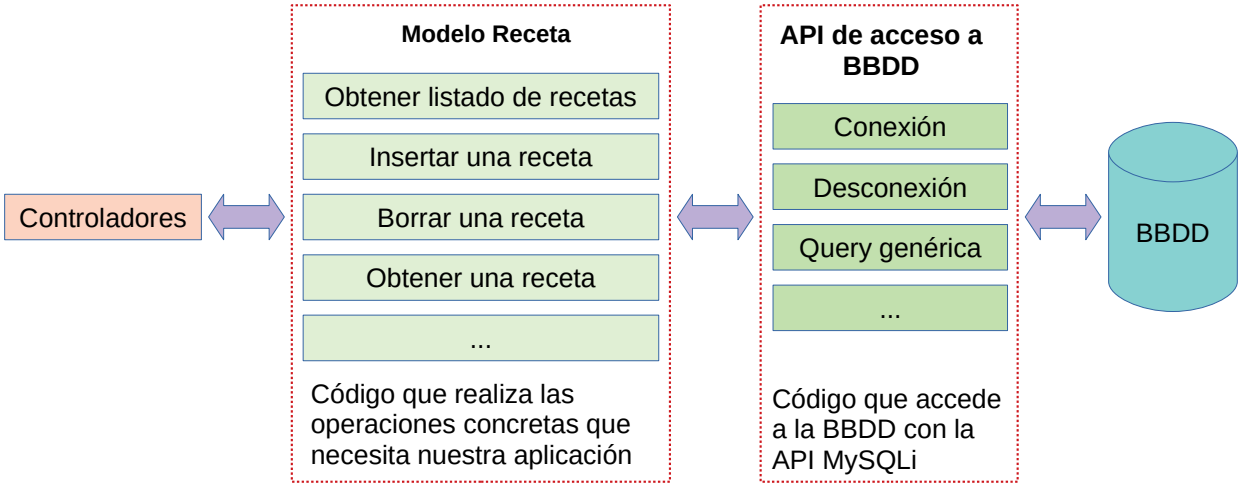
Aunque no opte por aplicar un patrón MVC para la organización de su código, al menos debería procurar mantener una estructura de directorios y ficheros que mantenga una organizacion razonable del código. En caso de que opte por usar el patrón MVC, en esta sección se dan algunas indicaciones.

Deberá tener tres directorios:

- `View/` En este directorio tendrá ficheros de tipo HTML, plantillas y ficheros PHP que tengan código destinado a crear páginas o partes de estas, es decir, que generan código HTML.
- `Model/` En este directorio se almacenan ficheros PHP con código para acceder a la BBDD e independizar el resto de la aplicación de la representación concreta y del sistema gestor de BBDD que se esté usando.
- `Controller/` En este directorio están los ficheros que, usando las vistas y los modelos, responden a las peticiones de los usuarios del sitio.

Por ejemplo, se necesitará código para generar el encabezado de cada página, el footer, el menú de navegación, etc. Esto se puede hacer usando únicamente código PHP o con una mezcla de PHP y plantillas HTML. Todo ese código estará en el directorio `View/`. Puede pensar que aquí tendrá funciones PHP que generan diferentes trozos de las distintas páginas que componen el sitio.

El código que tenga acceso directo a la BBDD se almacenará en `Model/`. Por ejemplo, funciones para la conexión y desconexión a la BBDD, funciones que hacen consultas, actualizaciones, borrados, etc. Se recomienda tener dos niveles de abstracción:



- API de acceso a BBDD. Estas funciones (o clases) son las encargadas de ejecutar las funciones de la API concreta de acceso al SGBD que estemos usando. En nuestro caso, al trabajar con MySQL, aquí se harán todas las llamadas a funciones de la API MySQLi de PHP como, por ejemplo, `mysqli_connect`, `mysqli_query`, `mysqli_fetch_all`, etc (o sus equivalentes PDO). Es aquí también donde se incluirán las cláusulas SQL de consulta, modificación, borrado etc. (`SELECT`, `UPDATE`, `DELETE`, etc). Esto se hace así para independizar la aplicación del SGBD concreto. Sería muy sencillo cambiar a PostgreSQL, SQLite, MongoDB u otro SGBD solo modificando estas pocas funciones.
- Modelo Receta. Es importante también independizar la aplicación de la estructura concreta de nuestra BBDD, es decir, de las tablas concretas que utilicemos para almacenar la información. Por eso se construyen los modelos, que no son más que una API (montada sobre la API de acceso a BBDD) que ofrece al resto de la aplicación la vista que necesita de la BBDD. Por ejemplo, en nuestro caso, dispondremos de una función que reciba como argumento el ID de una receta y que devolverá todos los datos que haya almacenados sobre ella. Habrá otra función que reciba como argumento los datos de una receta (provenientes de algún formulario web) y la insertará en la BBDD. Estas funciones, en nuestro caso, construirán las cláusulas SQL y las ejecutarán a travñes de la API de acceso a BBDD. Contendrán la lógica de acceso a la BBDD. Por ejemplo, la función que inserta una receta tendrá que comprobar si existía previamente (coincidencia en el título), en cuyo caso habrá que decidir si se hace una actualización o no se hace nada (por ejemplo).

Finalmente, los controladores serán aquellos módulos que responden a peticiones concretas de los usuarios. Por ejemplo, si un usuario quiere obtener el listado de recetas, el controlador deberá invocar las acciones del modelo receta que obtienen los datos de la BBDD y, a continuación, invocar las funciones de vista que generan el código HTML que muestra esa información. Finalizará devolviendo ese código HTML como resultado de su ejecución. La disposición de estos controladores se puede plantear de múltiples formas:

- Tener un único fichero `index.php` con argumentos GET (query string) para indicar la acción a realizar. En este caso, en el directorio raiz de la aplicación tendremos el fichero index.php y en la carpeta `Controller/` todos los ficheros con código PHP que sean usados desde `index.php`.
  Por ejemplo, podemos disponer que cuando el usuario acceda a la URL `https://localhost/~tweb/recetario/index.php?accion=list` el sitio web muestre el listado de recetas. Para ello, podemos disponer de uno o varios ficheros en `Controller/` que ejecuten

código de modelos y vistas para generar el listado solicitado. Desde `index.php` invocaremos el código de esos ficheros de la forma oportuna. En este caso, `index.php` cumple la función de "enrutador": analiza la URL y carga el código PHP (controlador) demandado.

- Tener un fichero PHP diferente para cada página que se puede generar en el sitio web. Siguiendo con el ejemplo anterior, podríamos disponer la siguiente URL para obtener el listado de recetas: `https://localhost/~tweb/recetario/controller/listado.php`. Sería conveniente seguir teniendo un `index.php` en el raíz de nuestra aplicación que, por ejemplo, muestre una página de bienvenida o alguna página por defecto (podría redirigir a algún controlador).
- Tener un único fichero `index.php` que hace el enrutado para ejecutar el código de los controladores pero, en lugar de usar un query string para determinar la acción a ejecutar, se puede configurar Apache para redirigir cualquier URL a ese fichero `index.php`. De esta forma, la acción concreta a ejecutar podría escribirse como parte de la URL. En el caso anterior, podríamos disponer que la URL para obtener el catálogo de libros sea `https://localhost/~tweb/recetario/recetas/listado`. Cuando Apache reciba esa petición, hará una redirección a `https://localhost/~tweb/recetario/index.php` pero se podrá analizar la URL original y determinar que el usuario desea ejecutar una acción etiquetada como "`recetas/listado`".

Tenga en cuenta que el sitio web contiene multitud de ficheros (HTML, PHP, ...) y que desde cualquier cliente se podría escribir la URL que accede a cualquiera de ellos pero, sin embargo, muchos de ellos no tiene sentido que sean accedidos de esa forma. Por ejemplo, suponga que tenemos un fichero `Model/conexion.php` que contiene las funciones que ejecutan las acciones de conexión y dexconexión de la BBDD. El cliente podría solicitar la URL `https://localhost/~tweb/recetario/Model/conexion.php` y sería válido. Lo normal en ese caso es que no ocurra nada ya que, en principio, ese fichero solo contendrá definiciones de funciones pero no llamadas a ninguna de ellas ni código ejecutable PHP. La consecuencia es que el usuario recibe como resultado una página en blanco. En esta situación, el usuario debería recibir algún mensaje de error indicando que la URL no es válida: ese fichero no está destinado a ser usado directamente desde el cliente sino que contiene funciones que permiten modularizar nuestra aplicación y serán ejecutadas desde otros ficheros PHP. Por tanto, debe impedir de alguna forma que esto ocurra: solo debe permitir que sean válidas aquellas URL que generan una página válida. Para ello puede usar un fichero `.htaccess` que proteja frente a accesos indebidos a carpetas o incluir código PHP en los ficheros de manera que si son accedidos desde una URL hagan una redirección a la página principal, etc.

4. Entrega de la práctica

La entrega consistirá en subir el ejercicio al servidor web de la asignatura (`void.ugr.es`) en los plazos establecidos por el profesor durante el curso. Se evaluará únicamente el material subido al servidor.

Para esta práctica en concreto:

- Dentro de su carpeta `public_html` debe incluir una carpeta llamada "`practicaPHP`".
- Dentro de esa carpeta debe incluir todos los ficheros de la práctica.
- Debe existir un fichero llamado `index.php` con la página de inicio del sitio web.

De esta forma, para acceder al sitio web de la práctica que ha creado se usará la URL:

```
https://void.ugr.es/~USUARIO/practicaPHP
```

Observaciones:

- No cumplir con alguno de los requisitos de entrega invalidará la entrega completa.
- No se admitirá ni evaluará ninguna entrega una vez finalizado el plazo.
- No se admitirán entregas por ningún otro medio.

Recomendación: no deje para el último momento la subida de la práctica a `void.ugr.es`.