UNIVERSIDAD DE GRANADA



# Tecnologías Web

## 3º Grado en Ingeniería Informática

# Proyecto Web

## Lista de la compra



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

Imagen de portada obtenida de Freepik

DECSAI
Departamento de Ciencias de la Computación e Inteligencia Artificial

# 1. Objetivos del proyecto

Con este proyecto se pretende que el alumno aplique de forma práctica gran parte de los conocimientos adquiridos durante el desarrollo de la asignatura. Consistirá en el desarrollo de una aplicación web sencilla pero suficientemente completa de manera que el resultado final sea totalmente funcional y comparable a cualquier aplicación web real (sin hacer demasiado énfasis en aspectos estéticos). Dicha aplicación, en líneas generales, dispondrá de:

- Una interfaz totalmente funcional que cubra las necesidades básicas expuestas en este guión.
- Una base de datos que almacene información relacionada con la aplicación y que permita que el contenido del sitio sea dinámico.
- Una gestión básica de usuarios. El sitio web mostrará diferentes vistas dependiendo el tipo de usuario que lo visite en cada momento.

Algunos requisitos técnicos básicos que deberá incorporar:

- Gestión correcta de sesiones.
- Operaciones CRUD (*Create*, *Read*, *Update*, *Delete*) para mantener actualizada la base de datos, es decir, formularios de creación, lectura, actualización y borrado de registros que permitan la gestión completa del sistema desde la aplicación web.
  - Uso de formularios "*sticky*".
  - Peticiones de confirmación de operaciones CRUD al usuario.
  - Validación de datos en el servidor ~~y, de forma secundaria, en el cliente~~.
- Uso de *cookies*.
- La implementación deberá cumplir con unos mínimos de calidad.

Para el éxito en el desarrollo de este proyecto va a necesitar, principalmente:

- HTML+CSS para el desarrollo del *frontend*.
- PHP para el desarrollo del *backend*.

~~Se planteará también el uso de JavaScript para el *frontend*, aunque su uso será menos relevante. El motivo no es otro que la temporización del temario de la asignatura. JavaScript se estudia al final del mismo y, por tanto, no queda demasiado tiempo para incorporarlo a este proyecto.~~

No se considera necesario el uso de JavaScript.

En internet podrá encontrar muchos sitios web con aplicaciones similares a la planteada. Puede usarlos como referencia para el diseño e incluso para la implementación.

Todas las capturas de pantalla, menús, etc de este guión son orientativos y el proyecto no tiene porqué parecerse en absoluto a ellos. Únicamente se pretende ilustrar la funcionalidad de la aplicación.

Se recomienda leer este documento por completo antes de iniciar la práctica para tener una visión global de lo que se pide y, así, se pueda abordar el diseño con suficiente previsión. El documento contiene una descripción del sistema como podría hacerla un potencial cliente e incorpora algunos detalles técnicos o sugerencias de implementación. Al final se indican algunas reglas para la elaboración, entrega y evaluación del proyecto.

# 2. La aplicación web

La aplicación web que debe implementar tiene como finalidad la gestión de una **lista de la compra compartida** por múltiples personas. Son dos las principales utilidades de la misma:

- Disponer de múltiples listas de la compra por parte de un usuario. Por ejemplo, puede disponer de una lista para el supermercado, otra para ropa, etc.
- Compartir esas listas con otras personas de forma que todas puedan contribuir a crearlas o

usarlas.

Esta es una aplicación típica en unidades familiares en donde las tareas domésticas se hacen de forma indistinta por varios de sus miembros. Cada vez que alguien detecta que hay una necesidad la apunta en la lista de la compra y cada vez que alguien va a comprar puede consultar la lista con los apuntes de todos los usuarios.

## 3. Información que debe gestionar el sistema

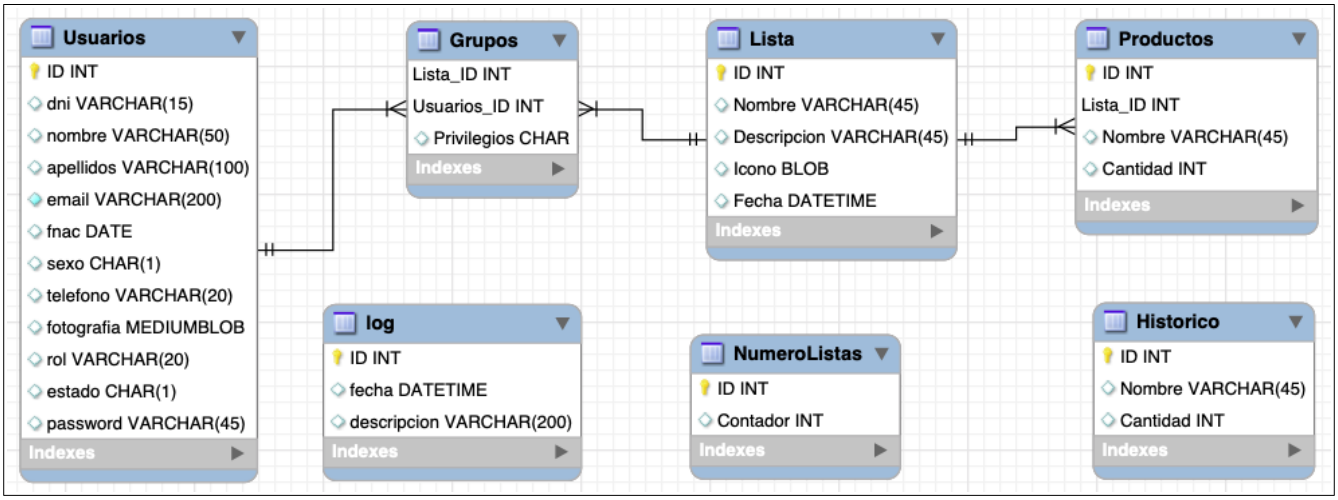
En esta sección se indican los elementos principales de información que debe manejar el sistema, incluyendo algunos apuntes sobre el tipo de dato de cada uno de ellos. Este sistema debe almacenar información sobre:

- Lista de la compra. El sistema permitirá tener múltiples listas de la compra de forma que varios usuarios pueden interactuar con ellas añadiendo productos o quitando productos:
  - Propietario [Clave externa].
  - Nombre de la lista [Cadena corta].
  - Descripción [Cadena larga].
  - Fecha de creación [Fecha].
  - Lista de usuarios con acceso a la lista. Habrá usuarios con permiso de edición y otros con permiso de lectura.
  - Productos anotados en la lista:
    - Nombre del producto [Cadena corta].
    - Cantidad del producto [Entero].
- Histórico de compras. Cada vez que se compre un producto de una lista se actualizará un contador de compras de dicho producto (y se eliminará de la lista):
  - Nombre del producto [Cadena corta].
  - Número de veces que se ha comprado [Entero].
- Histórico de listas. La aplicación llevará un contador con el número total de listas creadas desde que se instala la aplicación.
- Usuarios. Listado con todos los usuarios del sistema. Los datos que hemos de almacenar son estos:
  - DNI [Cadena corta].
  - Nombre [Cadena corta].
  - Apellidos [Cadena corta].
  - Teléfono [Cadena corta].
  - Email [Cadena corta].
  - Fecha de nacimiento [Fecha].
  - Sexo [Carácter o booleano].
  - Fotografía [Binario].
  - Clave [Cadena corta]. Los usuarios podrán acceder al sistema por lo que necesitarán disponer de una clave.
  - Estado [Carácter]. Indica el estado del usuario en el sistema (ver más adelante).
  - Rol [Carácter o cadena corta]. Tipo de usuario del sistema (administrador, normal).

De forma adicional, la aplicación dispondrá de un cuaderno de bitácora (*log*) en el que irá registrando los eventos más relevantes que se vayan produciendo durante la ejecución. Para ello, cada vez que ocurra algo de interés, anotará:

- Fecha [Fecha]. Cuándo ha ocurrido.
- Descripción [Cadena larga]. Qué ha ocurrido.

Por tanto, una de las primeras cosas que deberá plantearse es el diseño de la base de datos necesaria para gestionar la información. A modo de sugerencia y en concordancia con la información previa, se sugiere el siguiente diseño:




Este diseño puede sufrir variaciones pero se recomienda que se consulte con el profesor para asegurarse de que cumple con las necesidades del sistema.

## 4. Descripción de las funciones de cada tipo de usuario

Los usuarios de la página pueden ser de distinto tipo:

1. Visitantes. Son los usuarios que navegan por el sitio sin ningún tipo de autenticación, es decir, cualquier persona que acceda a nuestro sitio sin estar registrado en el mismo.
2. Registrado. Son usuarios que están registrados en nuestro sitio y que pueden participar en las listas de la compra.
3. Administradores. Son usuarios registrados que, además de hacer lo que hacen el resto de usuarios, también pueden administrar a otros usuarios y realizar tareas de mantenimiento del sitio.

La aplicación web permitirá que un usuario se identifique pasando de ser "visitante" a alguno de los otros tipos de usuarios en función de su perfil y dependiendo del tipo de usuario la aplicación mostrará unos contenidos u otros. En particular, cambiará el menú de navegación, que solo mostrará aquellas acciones que estén definidas para el tipo de usuario en cuestión.

### 4.1. Visitantes de la página

Los visitantes de la página son usuarios que no se han identificado en el sistema (cualquiera que visite la página web). La aplicación permitirá que un visitante pueda:

- Ver estadísticas de uso del sistema:
  - Número de listas creadas en total.
  - Número de listas activas (listas que no han sido borradas).
  - Número de listas con elementos pendientes de compra (que mantienen algún elemento anotado).
  - Histórico de productos comprados. El usuario podrá decidir el criterio de ordenación de este listado:
    - Orden alfabético.

        - De más a menos veces comprado.
        - De menos a más veces comprado.
- Solicitar su registro en el sistema. Con esta opción, un visitante podrá rellenar su ficha completa y hacer una solicitud para que lo den de alta.
- Loguearse. Deberá poder identificarse si está dado de alta en el sistema. Un usuario que se haya identificado en el sistema (que sería de tipo registrado o administrador) deberá poder desloguearse para volver a ser un visitante (o usuario anónimo).

En este punto toma sentido el campo "estado" de la información que se almacena del usuario. Este campo puede codificar información sobre si el usuario puede o no acceder al sistema o si es un usuario válido. Cuando un visitante rellena el formulario de solicitud, su información puede almacenarse en la misma tabla que el resto de usuarios pero poniendo algún valor especial en este campo. De esta forma se guarda su información pero no se le permite acceder al sistema mientras no haya sido validado por el administrador.

## 4.2. Registrados

Los registrados son usuarios que se han identificado en el sistema con algunas atribuciones adicionales. Podrán ver la misma información que los visitantes de la página y, además, un usuario con este perfil podrá realizar otras acciones.

Debe tener en cuenta que un usuario debe poder utilizar múltiples listas de la compra. Además, los permisos sobre cada lista pueden variar. Cuando un usuario crea una lista nueva este se convierte en su propietario (una lista solo puede tener un propietario). El propietario puede añadir o eliminar miembros a la lista de forma que puedan colaborar entre todos. Al añadir un nuevo miembro, el propietario decide si este podrá añadir, modificar y eliminar productos o, por contra, solo podrá visualizar los productos sin posibilidad de modificarlos. En el primer caso diremos que se trata de un usuario editor (con privilegios de edición de la lista) y en el segundo de un usuario lector (solo puede ver la lista). Ni los editores ni los lectores pueden modificar el icono de la lista, su nombre o su descripción.

La lista de acciones que puede realizar un usuario registrado es la siguiente:

- Modificar algunos de sus datos personales. Concretamente estos:
    - Email.
    - Teléfono.
    - Fotografía.
    - Clave.
- Ver un listado con las listas de la compra en las que participa:
    - Si tiene permisos de edición (propietario y editor) en una lista podrá:
        - Añadir nuevos productos.
        - Borrar productos.
        - Modificar la cantidad de un producto.
        - Marcar productos como comprados.
        - Puede ver la lista de miembros de la lista pero no puede modificarla.
    - Si solo tiene permisos de lectura: solo puede ver la lista de productos y de miembros de la lista pero no puede modificar nada.
    - Podrá filtrar el listado de listas en base a estos criterios:
        - Ver las listas de las que es propietario.
        - Ver las listas en las que no es propietario y tiene permisos solo de lectura.
        - Ver las listas en las que no es propietario y tiene permisos de edición.

        - Ver las listas en las que participa (propietario, editor o lector) que contienen una subcadena en su nombre o descripción.
    - La ordenación del listado se podrá hacer en base a:
        - Orden alfabético del nombre de la lista (ascendente).
        - Fecha de creación (descendente).
- Administrar sus listas de la compra:
    - Crear una lista nueva.
    - Editar información de la lista (nombre, descripción e icono).
    - Borrar la lista.
    - Añadir usuarios a la lista con permiso de:
        - Lectura.
        - Edición.

### 4.2.1. Ver las listas

Un usuario registrado únicamente puede ver las listas en las que participa y necesitará disponer de un formulario para obtener listados de dichas listas. Como podrían ser muchas, el usuario podrá filtrar los resultados permitiendo ver las listas que cumplen uno o más de los siguientes criterios:

- (A) Listas de las que es propietario.
- (B) Listas con permiso de edición.
- (C) Listas con permiso de lectura.
- (D) Listas que contienen una subcadena en su nombre o descripción.

Observe que podrá marcar o rellenar cero o más de estos items y el sistema buscará las listas que cumplen con todos los criterios utilizados: [ (A) or (B) or (C) ] and (D).

Además, el listado que se obtenga podrá ordenarse en base a distintos criterios. El formulario podría presentar un aspecto similar al siguiente:


Observe que en cada lista se muestran botones o enlaces para realizar las acciones que están permitidas sobre ella dependiendo de si se es propietario, lector o editor:

- Acceder a la lista en modo edición (si se es propietario o editor).
- Borrar la lista (si se es propietario).

- Ver la lista (si se es lector).

La página dispondrá al final una barra de avance y retroceso de página similar a la de la figura.

Puede pensar un diseño alternativo siempre y cuando mantenga esta funcionalidad y sea fácilmente utilizable.

### 4.2.2. Lista de la compra

A partir del listado, es posible acceder a listas de la compra individuales. En la siguiente figura tiene tres vistas distintas de una lista de la compra:

- Izquierda. Es la vista del propietario.
- Centro. Es la vista de un usuario editor.
- Derecha. Es la vista de un usuario lector.

Observe que puede reutilizar el código PHP/HTML del formulario para, con pequeñas variaciones, conseguir los tres formularios. Por ejemplo, la diferencia entre el formulario central y el de la derecha consiste en que se han omitido los campos de entrada para nuevos productos, se han omitido los botones que hay al lado de cada producto y se ha deshabilitado la entrada para el resto de campos.


Además del icono, el título de la lista y la descripción, puede ver un bloque a la derecha en el que se ven los usuarios con acceso a ella. Puede mostrar la identidad de ellos de forma abreviada (nombre y apellidos, tipo de acceso -lector/editor-, etc.). Además, en el caso de que el usuario que está visualizando la lista sea su propietario, este podrá eliminar a los usuarios de la lista o añadir otros nuevos para lo que se habilitarán enlaces o botones.

La lista en sí consiste en:

- Una primera línea que permitirá añadir nuevos productos en caso de que se tenga dicho privilegio. Tras añadir un nuevo producto se mostrará el listado actualizado. Alternativamente puede tener un enlace o botón para ir a un nuevo formulario que permita insertar nuevos productos.
- Una nueva línea por cada producto de la lista. Contendrá un formulario con el producto y la cantidad (editables ambos) y botones para modificarlo, marcarlo como comprado o borrarlo de manera que:
  - Si se pulsa el botón de comprado, el producto se eliminará de la lista y se anotará en el histórico de pedidos.
  - Si se pulsa el botón de borrado el producto se borrará de la lista (pero no se almacena en el histórico).
  - Si se pulsa el botón de modificar el producto se actualizará la información de acuerdo al contenido de las cajas de entrada.

  Tras realizar la acción se volverá a presentar el listado actualizado.

Puede pensar un diseño alternativo siempre y cuando mantenga esta funcionalidad y sea fácilmente usable.

### Añadir un miembro a la lista

Para añadir nuevos miembros a la lista deberá diseñar un formulario al efecto. Tenga en cuenta que deben ser usuarios registrados del sistema por lo que puede optar por:

- Mostrar una lista con todos los usuarios del sistema para elegir uno. Esto tiene el inconveniente de que el listado puede ser muy largo y no hay privacidad.
- Rellenar una caja de texto con el email del usuario. Al añadirlo el sistema comprobará si está dado de alta o no y actuará en consecuencia: si está dado de alta lo añadirá como miembro y si no mostrará de nuevo el formulario indicando que no se puede añadir. Esta opción es menos cómoda que la anterior para el usuario.

Además del usuario a añadir deberá indicar si este tendrá privilegios de lectura o de edición.

El formulario podría integrarse de alguna forma en la misma caja de la derecha de la lista de la compra si se estima conveniente (en sustitución del botón “Añadir miembro”).

## 4.3. Administradores

Los usuarios administradores son los que pueden gestionar la información básica del sistema y podrán realizar las siguientes tareas:

- Gestión de usuarios. Podrán obtener listados de usuarios, añadir nuevos usuarios, editar todos sus datos y borrarlos.
- Obtener un listado de todas las listas de la compra del sistema. Pueden hacer con ellas lo mismo que sus propietarios (editarlas, borrarlas, modificarlas).
- Visualizar el *log* del sistema.
- Operaciones de copia de seguridad y restauración de la base de datos.

Los usuarios administradores podrán realizar también todas las tareas de los visitantes (ver estadísticas) y los usuarios registrados (manipular sus propias listas de la compra).

# 5. Administración de usuarios

Para la administración de usuarios del sistema, deberá diseñar un formulario de captura que será muy versátil ya que tendrá que reutilizarlo en distintas partes de la aplicación con distinta funcionalidad en cada caso. Dicho formulario tendrá campos de entrada para todos los datos de un usuario pero podrá configurarse para:

- Presentarse con los campos vacíos. Este se usará para que los administradores puedan dar de alta nuevos usuarios.
- Presentarse con datos de un usuario. Este se usará cuando un administrador tenga que modificar los datos de un usuario del sistema.
- Presentarse con datos de un usuario que no sean editables. Este se usará cuando el objetivo sea únicamente el de mostrar datos sin posibilidad de que sean modificados. Por ejemplo, antes de hacer la inserción de un nuevo usuario, los datos se mostrarán para que el administrador los revise antes de hacer la operación en la base de datos. Observe que en este caso, en realidad no tienen porqué presentarse en forma de formulario (salvo por el botón de validación). Sin embargo, tiene la ventaja de poder reutilizar código e ilustrar el uso de atributos de HTML como `disabled` o `readonly`.
- Presentarse con datos de un usuario de manera que solo sean modificables algunos de ellos. Esto se usará cuando sea un usuario registrado quien vaya a modificar sus propios datos. Recuerde que solo podrá modificar el email, el teléfono, la fotografía y su clave. En este caso debería omitir los campos de rol y estado, que se utilizan para gestiones internas del sistema y no son de interés para el usuario.
- Presentarse con los campos vacíos pero omitiendo algunos de ellos. El objetivo es presentar el formulario de registro para usuarios visitantes. Deberá omitir los campos de rol y estado, que serán determinados automáticamente por el sistema (rol: “no-

administrador" y estado: "inactivo").

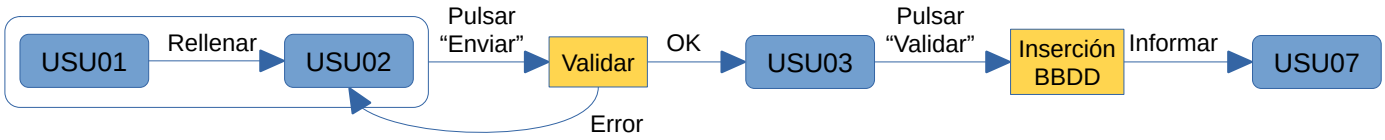
- ... y algún otro que pueda ser útil.

## 5.1. Dar de alta nuevo usuario

Este procedimiento solo puede realizarlo un administrador y le permite dar de alta nuevos usuarios en el sistema. La secuencia de formularios que se muestran es la siguiente:


- USU01: Se presenta un formulario en blanco.
- USU02 (se trata del mismo formulario que USU01): El usuario administrador rellena el formulario y pulsa "Enviar datos".
- USU03: El sistema presenta la información al usuario para que confirme la acción.
- USU07: Si el administrador confirma el formulario anterior, se procede a la inserción de datos en la BBDD y a mostrar información sobre el resultado de la operación.

El diagrama de estados asociado sería similar al siguiente:




- La caja "Validar" consiste en la validación, mediante código PHP, de los datos recibidos desde el formulario y se ejecutaría tras pulsar el botón de envío del formulario inicial.
- La caja "Inserción BBDD" consiste en realizar la operación de inserción de datos en la BBDD.

Observe que este esquema es típico en la mayoría de los procesos de formulario que tendrá que hacer en la aplicación web:

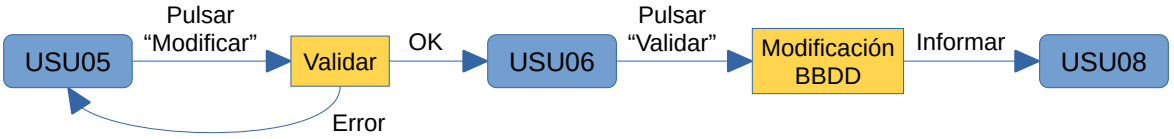
1. Presentar formulario: modificar o rellenar y pulsar un botón para enviarlo.
    1. El formulario puede estar vacío si se va a añadir información nueva.
    2. El formulario puede contener algunos datos si se pretende modificar información ya existente en el sistema o corregir algún dato mal introducido.
2. El servidor recibe los datos del formulario y los valida.
    1. Si la validación falla, devuelve el formulario conservando los datos para que sean reparados y vueltos a enviar.
    2. Si la validación es correcta, devuelve una página mostrando los datos que se van a insertar y espera confirmación por parte del usuario. Para esto no sería estrictamente necesario usar un formulario pero se puede aprovechar el diseño que ya tenemos y podemos reutilizar el código para mostrar la información (deshabilitando la entrada en los controles HTML).
3. Si el usuario confirma el envío en el paso 2.2 se procede a realizar la inserción en la BBDD y se devuelve una página informando del resultado de la operación.

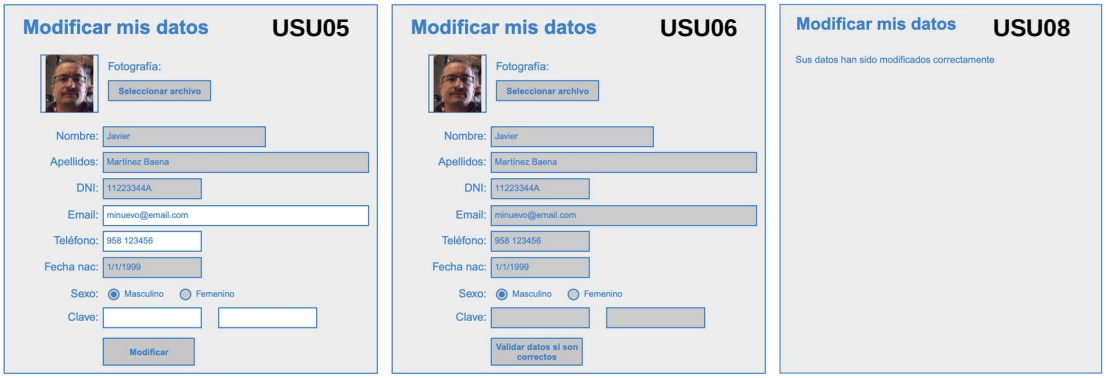
## 5.2. Edición de un usuario

Dependiendo del tipo de usuario que esté identificado en el sistema podrá modificar unos u otros datos.

Los usuarios de tipo administrador podrán editar todos los datos de cualquier usuario del sistema y, para ello, deberá seguir el esquema de la sección 5.1. En esta ocasión, el formulario inicial se mostrará con datos del usuario que se desea modificar en lugar de con los campos en blanco.

Los usuarios de tipo “registrado” podrán editar solo algunos de sus propios datos tal y como se indicó en la sección 4.2. Por tanto, en ese caso, el diagrama de estados y la secuencia de formularios o pasos sería la siguiente:




## 5.3. Borrado de usuarios

Los usuarios solo pueden ser borrados por los administradores. Para ello, una vez determinado el usuario a borrar, se presentará la información del mismo con un formulario similar al USU03 para pedir confirmación previa al borrado efectivo de la BBDD.

## 5.4. Autoregistro de usuarios

Un usuario visitante del sistema podrá solicitar que le den de alta en el mismo. Para ello deberá rellenar un formulario como el USU04 y seguir un procedimiento similar:


Una vez recibidos los datos, el servidor los almacenará en la BBDD y almacenará en el campo “estado” de la tabla de usuario un valor que indique que el usuario aun no puede acceder a la aplicación y que está a la espera de que un administrador verifique que los datos son correctos y le dé permiso.

Para que el administrador pueda activar a un usuario que está en espera tras el autoregistro,

puede reutilizar de nuevo el formulario para que tenga un aspecto similar al siguiente:


Puede optar por hacer que los campos sean editables o no. Observe que dispone de tres botones:

1. Activar e informar. El administrador, una vez haya comprobado que los datos facilitados son coherentes, procede a modificar el estado del usuario para permitirle acceder al sistema y le enviaría un email informándole de esto.
2. Informar de error. Si los datos que se han recibido contienen errores, al pulsar este botón se muestra un formulario con una caja de texto para que el administrador escriba una descripción del error y se la envíe por email al usuario.
3. Borrar usuario. Permite el borrado de esta solicitud sin más.

En esta práctica no hay que enviar realmente el email. Basta con que muestre un mensaje indicando que se ha enviado. Pero, si optase por implementar esta funcionalidad, tenga en cuenta que debe almacenar en void.ugr.es sus credenciales de acceso al servidor de correo para que la aplicación web pueda conectarse a él y enviar correos. Tratándose de un dato sensible, debería crear una cuenta de correo expresamente para la realización de esta práctica y no usar, en ningún caso, las credenciales de su cuenta de correo personal o institucional.

## 5.5. Listados de usuarios

En varios procesos del sitio web deberá disponer de una página que muestre listados de usuarios para buscar alguno concreto:

- Para modificar sus datos.
- Para activarlo.
- Para borrarlo.
- ...

Por tanto, debería disponer de un módulo que genere dicho listado de forma que se muestre en él alguna información resumida de los usuarios y botones para realizar determinadas acciones dependiendo de los privilegios de quien hace el listado. En la siguiente figura se ve un ejemplo:

Se trata de un ejemplo, por lo que en su aplicación podría optar por mostrar otros datos y poner otros botones.

La página mostrará al final una barra de navegación para ir avanzando por el listado. Tenga en cuenta que un listado único con miles de usuarios no es algo operativo.

## 6. Gestión de la base de datos

En relación con este tema, la aplicación deberá incluir para los usuarios administradores las siguientes opciones:

- Borrar la BBDD completa. Tenga en cuenta que si lo borra todo podría quedarse sin ningún usuario administrador que permita continuar trabajando con la aplicación lo que la dejaría en un estado inservible. Para evitarlo, puede dejar en la BBDD al menos el usuario administrador que está ejecutando la operación o bien crear alguno por defecto.
- Crear una copia de seguridad de la BBDD. Consistirá en devolver un fichero con la secuencia de cláusulas SQL que permitirían restaurar la BBDD completa en caso de necesidad.
- Restaurar la BBDD a partir de un fichero de cláusulas SQL subido desde un formulario. Al realizar esta operación deberá borrar previamente la información que haya almacenada para evitar duplicidad (con cuidado de no dejarla inservible en caso de borrar todos los administradores).
- Restaurar la BBDD a partir de los datos de ejemplo indicados en la sección 11.1.

Las operaciones de borrado y restauración masivas, pueden plantear problemas con las restricciones de claves externas. Es frecuente deshabilitar la comprobación de claves externas durante estos procesos. También puede haber detalles importantes relacionados con la configuración del SGBD ya que cabe la posibilidad de que tras borrar y crear una tabla, si esta información no queda asentada en la BBDD, las operaciones de inserción o consulta posteriores fallen. Para ello deberá gestionar adecuadamente transacciones que garanticen que el proceso es correcto.

## 7. Puesta en funcionamiento de la aplicación

Para instalar la aplicación en un nuevo servidor web deberá, además de copiar los ficheros necesarios (.php, .html, ...) y realizar posibles ajustes en la configuración del servidor web, pensar cómo se inicializa la base de datos. Hay varias alternativas:

1. La más básica sería acceder al SGBD a través de alguna aplicación habilitada para ello (phpMyAdmin, etc.) y crear las tablas de forma manual así como, al menos, un primer usuario administrador. Esto no siempre es posible y requiere conocimientos específicos sobre administración de BBDD.
2. La propia aplicación web que está desarrollando podría hacer la creación de tablas de forma automática. Al acceder por primera vez a la aplicación esta realizará alguna acción sobre la BBDD (comprobar login o alguna otra consulta). En ese momento debería detectar que las tablas no existen y podría proceder a ejecutar un script de creación de tablas (y el usuario administrador por defecto).
3. Otra técnica consiste en que, al copiar los ficheros de la aplicación en el servidor, se incluya algún fichero con un nombre especial (puede estar vacío). La primera vez que un usuario acceda a la aplicación detectaría que ese fichero existe y procedería a la creación de las tablas para, una vez creadas, borrar el fichero.

La opción 2 tiene como ventaja sobre la 3 que no depende de la existencia o no se ese fichero inicial y permite que la BBDD se regenere automáticamente cada vez que se borran las tablas de la BBDD.

En este proyecto debe implementar la opción 2. En la sección 6 ya habrá implementado módulos que realizan las tareas necesarias (en buena medida) para llevar a cabo esto.

# 8. Log de la aplicación

Se debe mantener un registro (*log*) con los eventos principales que se van produciendo en la aplicación:

- Cada vez que se identifica un usuario.
- Cada vez que cierra la sesión un usuario identificado.
- Intentos de identificación erróneos.
- Cada vez que un usuario hace alguna acción que modifique la BBDD:
  - Inserción de nuevos usuarios.
  - Edición de usuarios.
  - Borrado de usuarios.
  - Creación de una lista.
  - Borrado de una lista.
  - Añadir un producto.
  - etc.
- ...

Este registro podría mantenerse en un simple fichero pero en este proyecto deberá almacenarlo en una tabla de la BBDD. La información que debe almacenarse es:

- Hora y fecha del evento.
- Breve descripción (una cadena de texto).

Cualquier usuario administrador podrá ver este registro. En el listado aparecerán primero las entradas más recientes.

# 9. Cuestiones genéricas sobre diseño e implementación

## 9.1. Diseño del sitio

El diseño de sitios web es un tema complejo que no se aborda en la asignatura. En esta práctica, tanto el diseño gráfico como el maquetado del sitio son libres. Se pueden entender las capturas de pantalla mostradas solo como sugerencias y se han incluido con el único propósito de ilustrar la funcionalidad requerida. La valoración tendrá en cuenta, sobre todo, la parte técnica y funcional del sitio (uso de las tecnologías estudiadas, usabilidad, etc).

La parte estética o artística pasa a un segundo plano salvo que sea tan mala que impida un uso razonable de la web o demuestre un bajo conocimiento de las tecnologías empleadas, en cuyo caso será valorada de forma negativa.

Deberá considerar si es más adecuado usar un diseño de ancho fijo o fluido. En cualquier caso, la página deberá visualizarse en pantallas de ancho mayor o igual a 800px y en diferentes navegadores.

La práctica deberá funcionar bien sobre cualquier navegador así que se recomienda que el alumno la pruebe en varios de ellos para asegurarse de que no hay problemas de compatibilidad. La corrección se hará con Firefox y Chrome (versiones actuales).

El uso de un diseño adaptable se valorará positivamente. No es necesario que sea muy sofisticado pero, al menos, debe permitir una visualización correcta en dispositivos con ancho inferior a 800px.

## 9.2. Formularios del sitio

Todos los formularios del sitio deben cumplir las siguientes normas:

- Deben ser de tipo *sticky* cuando tenga sentido. Por ejemplo en todas las operaciones de edición.
- La validación de datos del formulario debe hacerse en el servidor (PHP) en cualquier caso.

Para asegurar que esta funciona correctamente deberá incluir el atributo novalidate en todos los formularios de la aplicación.

- Cada dato deberá ser validado de acuerdo a su tipo. Por ejemplo, la validación de un email será distinta que la de un apellido.
- ~~Deberá incluir validación en el cliente (`JavaScript`). Desde un punto de vista técnico, esta es secundaria y se hace solo para mejorar la usabilidad del sitio.~~
- En la medida de lo posible, los datos de un proceso deben recogerse en un único formulario. No se admitirán soluciones que obliguen a un usuario a pasar por varios formularios de forma innecesaria (para realizar un proceso que podría haberse hecho con un único formulario).
  - Ejemplo 1: Para crear un usuario se piden todos los datos en un único formulario en lugar de pedir el nombre, en un segundo formulario los apellidos, en un tercer formulario el DNI, etc.
  - Ejemplo 2: Para editar la información personal de un usuario debe hacerse en un único formulario que incluya todos los datos modificables. No se admitirá una solución que obligue al usuario a pasar por varios formularios para hacer alguna modificación.
- No deberá mostrar o solicitar información que sea interna del sistema (por ejemplo códigos, claves primarias de la BBDD, *hash* de passwords, etc).
- La modificación o inserción de datos, como norma general, serán procesos que pidan confirmación, es decir, que necesitarán de:
  - Una primera página que muestra el formulario para editar o añadir información nueva.
  - Una segunda página que pedirá confirmación de inserción o edición. En este paso la información no debe poder modificarse, simplemente se muestra para que el usuario sepa lo que va a modificar o insertar.
  - Una tercera página que informe del éxito o no de la operación
- El borrado de datos, como norma general, también pedirá confirmación por lo que se necesita:
  - Un primer formulario que presenta la información a borrar y pide confirmación. En este caso, el formulario podría tener como único `input` el botón de `submit`, el resto de datos podrían ser `inputs` no editables o incluso tags `HTML` que no sean controles del formulario.
  - Una segunda página que informa del resultado de la operación.

En los procesos de formulario que necesitan varios pasos debe tener en cuenta que:

- La información que se rellena en el primer formulario se envía normalmente con el método `POST`.
- Esa información llega al segundo formulario (o página) para mostrarla pero sin posibilidad de edición. En este caso lo habitual será poner los controles de tipo `input` con el atributo `disabled` o `readonly` activado (se pueden ver pero no modificar). Debe tener en cuenta que al enviar el formulario es posible que estos controles no sean enviados por lo que deberá estudiar la forma de que lleguen, si fuese necesario, al tercer formulario que es el que realiza la operación de inserción o modificación en la BBDD. En estos casos es frecuente duplicar la información en inputs de tipo `hidden` para pasarlos de un formulario a otro mediante `POST` (o `GET`).
- En particular, no es posible rellenar el atributo value de un input de tipo `file` desde PHP por lo que no es posible reenviar ficheros de un formulario a otro. Una solución es que tras enviar el fichero desde el primer formulario, se utilicen variables de sesión para recuperar la información en los siguientes formularios. Esta solución es válida también para cualquier tipo de `input` y minimiza el trasiego de datos entre formularios: una vez enviados los datos desde el primer formulario estos se almacenan en el servidor en variables de sesión y en los siguientes formularios sólo se necesita pasar de uno a otro una pequeña porción de información.

## 9.3. Diseño y organización del código

La organización interna del sitio es libre. Se puede optar por distintos enfoques y/o paradigmas de programación:

- Paradigma procedural u orientado a objetos.
- Tener un único fichero (index.php) para gestionar el sitio completo.
  - Usando parámetros GET para ver los distintos contenidos.
  - Usando algún tipo de enrutador para analizar la URL en lugar de usar parámetros GET.
- Un fichero PHP diferente para cada página del sitio.
- Un diseño de tipo MVC.
- ...

En cualquier caso **debe evitar la técnica del "copia/pega"**, que será evaluada de forma negativa (e incluso podría implicar obtener una nota insuficiente para superar el proyecto). Dependiendo del diseño o enfoque utilizado, se admite un copia/pega básico con el esqueleto de una página PHP siempre y cuando los elementos que incluya sean mínimos y correctamente modularizados. Por ejemplo para incluir código HTML o PHP o para hacer algunas llamadas a funciones PHP que maqueten la página o comprueben la autenticación de usuarios. Si tiene dudas consulte con el profesor.

En cuanto a la organización de los ficheros también se deja libertad pero procure que su organización en carpetas sea razonable. Procure también proteger aquellos ficheros que contengan información sensible (datos de acceso a MySQL, log, copias de seguridad, etc) en lugares no accesibles desde internet.

## 9.4. Uso de cookies

Deberá hacer uso de *cookies* para facilitar las búsquedas. Cuando un usuario accede al formulario de búsqueda de listas y ejecuta la consulta, se almacenarán en forma de *cookies* los valores usados en los campos del formulario.

La próxima vez que el usuario acceda al formulario para realizar una nueva búsqueda, la aplicación comprobará si existen esas *cookies* y las usará para mostrar el formulario relleno con los valores que contengan, facilitando así que se repita la última búsqueda realizada. Tenga en cuenta que solo se almacena la última búsqueda.

## 9.5. Uso de JavaScript

~~Debe~~ Puede utilizar JavaScript para realizar algunas mejoras del sistema que, aunque no afectan a la funcionalidad del mismo, sí mejorarían la experiencia de usuario. En cualquier caso no es un requisito de esta práctica sino que es algo voluntario y opcional. Algunas sugerencias:

- Se pueden ocultar o mostrar formularios o determinados ítems para facilitar la legibilidad de la página. Por ejemplo:
  - Al mostrar un listado de ítems, es posible que no nos interese toda la información de cada ítem. Podríamos ver inicialmente únicamente alguna información básica del ítem y, al pulsar sobre él, ampliar la información.
  - ...

  Observe que la funcionalidad de la aplicación no varía en caso de que, por algún motivo, el navegador no pueda ejecutar el código JavaScript.
- Se puede usar JavaScript para hacer la validación de datos en el cliente. Esto no nos exime de hacer la validación en el servidor, que es obligatoria en cualquier caso.

  Si el navegador no puede ejecutar el código JavaScript, simplemente afectaría a la usabilidad pero no sería un problema para la aplicación.
- ...

Se recomienda dejar esta tecnología para el final del proyecto. Una vez finalizado se pueden añadir estas pequeñas mejoras con poco esfuerzo y, de paso, evitarán que hagamos nuestra aplicación dependiente de JavaScript.

## 9.6. Uso de AJAX

El uso de tecnología AJAX no se considera obligatorio en este proyecto aunque su uso en alguna parte se valorará de forma positiva.

Algunos lugares en los que sería apropiado su uso:

- Paginación de listados de resultados. Al pulsar sobre los botones del paginador la página se actualizaría de forma dinámica sustituyendo los datos.
- Sustitución del paginador. En lugar de él, puede detectar cuando el usuario alcanza el final de la página de un listado y hacer automáticamente una petición AJAX para añadir nuevos datos.
- Ampliación de información de ítems. Por ejemplo, al hacer un listado de ítems podría mostrarse únicamente un resumen de la información y un botón para ampliar información. Al pulsar el botón se contactaría con el servidor para obtener el resto de datos y se mostrarían en la misma página.

## 9.7. El patrón MVC

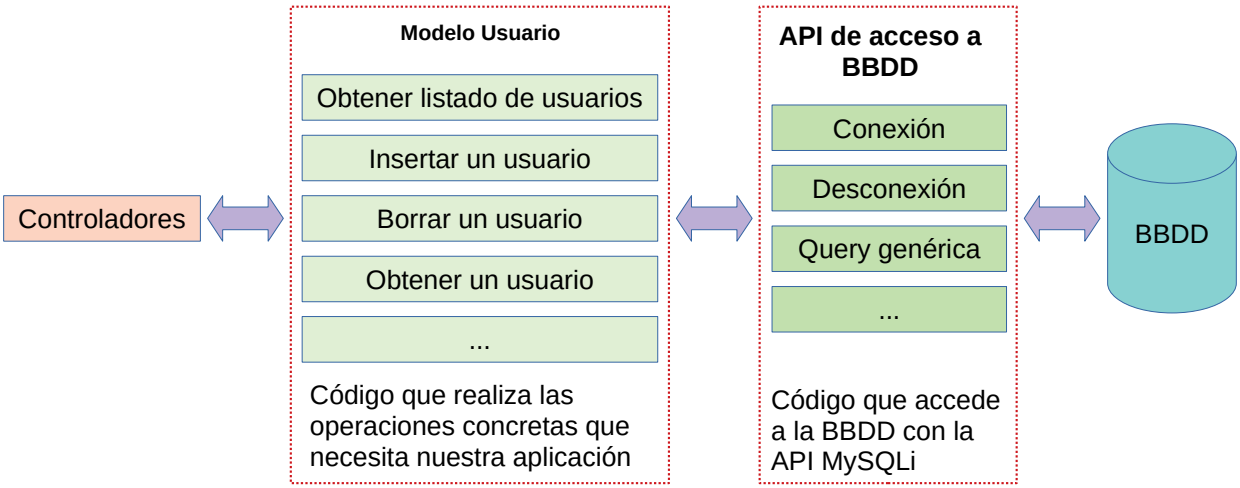
Aunque no opte por aplicar un patrón MVC para la organización de su código, al menos debería procurar mantener una estructura de directorios y ficheros que mantenga una organizacion razonable del código. En caso de que opte por usar el patrón MVC, en esta sección se dan algunas indicaciones.

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 Usuario. 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 un usuario y que devolverá todos los datos que haya almacenados sobre él. Habrá otra función que reciba como argumento los datos de un usuario (provenientes de algún formulario web) y lo insertará en la BBDD. Estas funciones, en nuestro caso, construirán las cláusulas SQL y las ejecutarán a través de la API de acceso a BBDD. Contendrán la lógica de acceso a la BBDD. Por ejemplo, la función que inserta un usuario tendrá que comprobar si existía previamente (coincidencia en el DNI), 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 usuarios, el controlador deberá invocar las acciones del modelo usuario 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/index.php?accion=list` el sitio web muestre el listado de usuarios. Para ello, podemos tener 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 usuarios: `https://localhost/~tweb/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 listado de usuarios sea `https://localhost/~tweb/usuarios/listado`. Cuando Apache reciba esa petición, hará una redirección a `https://localhost/~tweb/index.php` pero se podrá analizar la URL original y determinar que el usuario desea ejecutar una acción etiquetada como "`usuarios/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/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.
El esquema previo muestra el caso del modelo para la gestión de usuarios pero lo normal es tener múltiples modelos según la información con la que trabajan.

## 10. Aspectos principales de la evaluación

Se evaluará la práctica en todas sus vertientes aunque, como ya se ha dicho, el diseño visual pasará a un segundo plano siempre y cuando sea funcional y usable. Se enumeran a continuación algunos detalles a tener en cuenta de cara a la evaluación de la práctica:

- Se deberá prestar especial atención al cumplimiento de todas las normas explicadas en este documento. El no cumplimiento de las mismas podrá implicar una calificación de "cero".
- La aplicación debe funcionar correctamente en el servidor void.ugr.es. En caso de que falle la práctica, podrá tener una calificación de "cero".
- La base de datos debe contener datos de prueba que permitan comprobar el correcto funcionamiento de todas las páginas del sitio. Así mismo, debe suministrarse el login y clave de todos los usuarios registrados en el sistema así como el rol de cada uno de ellos. Ver sección 11.1. El no cumplimiento de este punto puede implicar una calificación de "cero".
- Se deberán utilizar elementos de HTML y CSS estándares y con suficiente implantación en los navegadores actuales. La práctica se podrá probar indistintamente (al menos) en Firefox y Chrome.

**Diseño de la web:**

- La interfaz de la aplicación debe ser homogénea en todas las páginas. Se valorará muy negativamente el cambio de diseño en diferentes páginas del sitio.
- La interfaz debe ser funcional: tamaño proporcionado del texto y de otros elementos, colores "razonables", elementos bien posicionados, ...
- Se valorará que el estilo sea fluido y adaptable frente a estilos de ancho fijo. Aunque si la página tiene un elevado componente gráfico/artístico sí se valorará igualmente el maquetado fijo.
- Se valorará el uso de diseño adaptable para móviles o tabletas.
- Los formularios serán de tipo "sticky" y deben facilitar la inserción y edición de datos.
- Tras procesar datos provenientes de un formulario se debe informar al usuario sobre el éxito o no de la operación realizada.
- Aunque como ya se ha dicho la valoración principal de la página se hará en base a criterios técnicos, también se valorarán positivamente los diseños que tengan un aspecto más

profesional.

**Base de datos:**

Aunque no se pide mucho detalle en el diseño de la base de datos, sí se requiere que en la documentación del proyecto se incluya, al menos, y de forma breve:

- Un modelo E-R.
- El modelo relacional que se obtiene a partir del modelo E-R.
- Breve descripción de las tablas obtenidas.

El modelo relacional explicado en la documentación debe corresponderse con el implementado en la aplicación. El no cumplimiento de este punto puede implicar una calificación de "cero".

En la entrega de la práctica debe incluir un fichero de restauración de la BBDD que contenga todas las cláusulas que permiten borrar (DROP) y crear (CREATE) las tablas de la aplicación así como las inserciones (INSERT) de los datos de prueba. Debe indicar, en la documentación, el nombre del fichero (incluyendo la ruta si no está en el directorio raíz).

Tenga en cuenta que un mal funcionamiento del borrado y restauración de la BBDD puede conllevar a dejar inutilizable la aplicación y esto podría implicar obtener una calificación de “cero”. Por tanto, asegúrese que que incluye esta funcionalidad solo si funciona correctamente.

**Implementación:**

- Se comprobarán algunas cuestiones de seguridad sencillas como, por ejemplo:
  - Saneado de datos en formularios.
  - Consultas seguras a bases de datos.
  - Validación de formularios adecuada en el servidor.
  - Acceso de usuarios no identificados a las páginas destinadas a usuarios identificados.
- Se valorará por igual el paradigma de programación usado con PHP: programación procedural u orientada a objetos.
- Se valorará la calidad del código desarrollado:
  - Muchas de las páginas del sitio hacen tareas comunes o muy similares: no debe usar la técnica del "copia/pega" sino modularizar adecuadamente su código. Por ejemplo: se deberían reutilizar los formularios para añadir nuevos registros y para editarlos o confirmar su actualización; bastaría con crear un formulario tipo y activar o desactivar campos o botones según la tarea.
  - Calidad técnica.
  - Estilo.
  - Documentación interna / comentarios.
  - Organización del código en ficheros.

En la asignatura no se instruye sobre el uso de ningún tipo de framework. Previa consulta al profesor, se permite el uso de algún framework de CSS siempre y cuando quede acreditado que el alumno tiene los conocimientos impartidos en la asignatura además de los conocimientos adecuados sobre el framework en cuestión.

**En ningún caso se permite usar frameworks de PHP ni de JavaScript.**

## 11. Entrega de la práctica

Al comienzo del curso al alumno se le ha facilitado un usuario y clave para acceder al servidor `void.ugr.es`. En dicho servidor hay una carpeta `public_html` en la que el alumno puede subir páginas web, siendo estas accesibles a través de cualquier navegador. Dichas páginas se encuentran protegidas también por el usuario y clave del alumno. El profesor puede acceder en todo momento a toda la información subida al servidor.

Dentro de la carpeta `public_html`, deberá crear una subcarpeta llamada **`proyecto`** en la que subirá la práctica desarrollada de manera que sea funcional. Debe existir en dicha carpeta un fichero `index.html` o `index.php` tal que, al cargarlo, se visualice la aplicación desarrollada.

Por ejemplo, si el alumno tuviese como nombre de usuario "joseperez1920" la práctica debería visualizarse al cargar la siguiente URL:

```
https://void.ugr.es/~joseperez1920/proyecto
```

Cualquier documentación adicional que necesite esta práctica deberá estar contenida en un

fichero llamado `documentacion.pdf` alojado también en la carpeta del proyecto.

- No cumplir con alguno de los requisitos de entrega invalidará la entrega completa.
- No se admitirán entregas pasado el plazo establecido para ello bajo.
- No se admitirán entregas por ningún otro medio.

## 11.1. Datos de ejemplo

Se considera obligatorio que la aplicación web entregada incluya información que tenga sentido e ilustre correctamente su funcionamiento. Lo más apropiado será usar algún ejemplo real para evitar invenciones fuera de lugar. En caso de que se opte por usar datos ficticios no se permitirá usar textos sin sentido o muy genéricos (es decir, no se deben usar nombres o textos como "aheitc4mw2xsko" o "Texto de prueba"). Así mismo, se habrán incluido datos de prueba para todas las partes de la aplicación (usuarios de varios tipos, textos, fotografías, etc.).

Debe incluir, al menos, los siguientes datos de ejemplo en la aplicación:

| **Usuario/email** | **Clave** | **Privilegios** | **Listas** |
|---|---|---|---|
| admin@void.ugr.es | 1234 | Administrador | --- |
| juan@void.ugr.es | 1234 | Administrador | Lista1: 3 artículos<br>+ana editor, +maria lector |
| maria@void.ugr.es | 1234 | --- | Lista2: 7 artículos<br>+juan lector |
| | | | Lista3: 5 artículos<br>+juan editor, +ana lector |
| | | | Lista4: 2 artículos |
| | | | Lista5: 3 artículos<br>+juan lector |
| | | | Lista6: 4 artículos |
| ana@void.ugr.es | 1234 | --- | --- |
| pepe@void.ugr.es | 1234 | --- | Lista7: 3 artículos<br>(sin miembros adicionales) |

Deberá rellenar el resto de datos con valores ficticios: datos de los usuarios, descripción de las listas, fotografías, etc.

También deberá disponer de un histórico de, al menos, 10 productos comprados.

Para poder probar la función de avance de página en los listados de la sección 4.2.1, los registros se mostrarán de 3 en 3.

## 11.2. Documentación de la aplicación

Debe incluir en la entrega un único fichero en formato pdf que incluya, al menos, los siguientes elementos:

- Identificación de el/los estudiante/s que ha/n realizado la práctica.
- En caso de incluir usuarios adicionales a los indicados en la sección 11.1 indique el nombre de usuario y clave de los mismos.
- Nombre del fichero de restauración de la BBDD con datos de prueba.
- Diseño previo de la aplicación (mockups, wireframe, etc)
- Diseño de la BBDD (modelo E-R y modelo relacional).
- Explicaciones técnicas que considere relevantes para la evaluación de la práctica o que quiera poner en valor por algún motivo.
- Listado de los ítems opcionales que haya incluido en su proyecto.

Además, el código desarrollado (HTML, CSS, PHP, JavaScript) debe estar convenientemente comentado.

**En el pie de página del sitio web incluirá obligatoriamente un enlace al fichero pdf con la documentación del proyecto.**

## 11.3. Prácticas en equipo

Esta práctica se puede desarrollar de forma individual o en pareja. La autoría debe quedar reflejada la documentación entregada así como en los ficheros fuente del proyecto. En caso de que se vaya a realizar en parejas se le debe comunicar al profesor durante las dos semanas siguientes a la propuesta de esta práctica (a la publicación de este documento). **No comunicar este dato en tiempo y forma supondrá que la práctica deberá entregarse de forma individual.**

En caso de que se haga en pareja, podrá solicitar al profesor un usuario/clave para alojarla que sea compartido por los integrantes del equipo de forma que se mantenga la privacidad en el resto de prácticas.

**En el pie de página del sitio web incluirá obligatoriamente el nombre de el/los autor/es del proyecto.**

## 11.4. Publicidad de la práctica

Se recomienda no publicar la práctica (ni esta ni otras) en repositorios públicos para velar por su privacidad **hasta la finalización del curso académico**. Según la normativa de evaluación y de calificación de los estudiantes de la Universidad de Granada:

*Articulo 13. Desarrollo de las pruebas de evaluación*

> *7. Los estudiantes están obligados a actuar en las pruebas de evaluación de acuerdo con los principios de mérito individual y autenticidad del ejercicio. Cualquier actuación contraria en este sentido, aunque sea detectada en el proceso de evaluación de la prueba, que quede acreditada por parte del profesorado, dará lugar a la calificación numérica de cero, la cual no tendrá carácter de sanción, con independencia de las responsabilidades disciplinarias a que haya lugar.*

*Artículo 15. Originalidad de los trabajos y pruebas.*

> *2. El plagio, entendido como la presentación de un trabajo u obra hecho por otra persona como propio o la copia de textos sin citar su procedencia y dándolos como de elaboración propia, conllevará automáticamente la calificación numérica de cero en la asignatura en la que se hubiera detectado, independientemente del resto de las calificaciones que el estudiante hubiera obtenido. Esta consecuencia debe entenderse sin perjuicio de las responsabilidades disciplinarias en las que pudieran incurrir los estudiantes que plagien.*