Tarea online

Título de la tarea: Trabajar con bases de datos en PHP.

Unidad: 02

Ciclo formativo y módulo: DAW, Desarrollo Web en Entorno Servidor.

Curso académico: 2020/21

¿Qué contenidos o resultados de aprendizaje trabajaremos?

- Crear una aplicación web (en PHP) que use una base de dato (MySQL) como el origen de los datos manejados por la aplicación. E esa aplicación se deberán realizar los procesos de:
  - Conectar con la base de datos.
  - Insertar información en la base de datos.
  - Rescatar información de la base de datos.
  - Modificar información de la base de datos.
  - Eliminar información de la base de datos.
- Validar y convertir los datos recibidos vía GET y POST para evita posibles problemas de seguridad.

1.1.- Ejercicio 1: preparación del entorno

En este primer ejercicio lo que tienes que hacer es simplemente prepara una estructura inicial del proyecto y la base de datos.

Para empezar, crea una base de datos en MySQL llamada **`INCMOTIV`**, y dentr de ella, crea la tabla siguiente:

```
CREATE TABLE usuarios (
    id int(11) NOT NULL AUTO_INCREMENT PRIMARY KEY,
    nombre varchar(30) NOT NULL,
    apellidos varchar(50) NOT NULL,
    email varchar(50) UNIQUE NOT NULL,
    dni varchar(10) UNIQUE NOT NULL,
    password varchar(128) NOT NULL,
    irrd varchar(32) UNIQUE NOT NULL,
    creacion timestamp NOT NULL DEFAULT current_timestamp(),
    validacion datetime,
    superuser boolean NOT NULL DEFAULT false
);
```

Después crea el proyecto **`DWES02`** con la siguiente estructura inicial:

- **`./conf`** : carpeta con los archivos de configuración.
  - **`./conf/db.conf.php`**: Archivo que contendrá la informació necesaria para conectar con la base de datos usando **`PDO`**. L información a almacenará será DSN, usuario y contraseña, y s almacenará en las constantes: **`DB_DSN`**, **`DB_USER`**, **`DB_PASSWD`**.
- **`./forms`** : carpeta que contendrá los formularios (fragmentos de códig HTML junto al código PHP necesario). Los formularios de esta carpet serán incluidos desde los ejercicios.
- **`./libs`** : carpeta que contendrá archivos con librerías.
  - **`./libs/conn.php`** : archivo que contendrá una función par conectar a la base de datos usando PDO y las constante definidas en **`db.conf.php`**. Puedes incluir aquí también otra funciones auxiliares relativas a la conexión.
  - **`./libs/users.dao.php`** : archivo que contendrá las funciones qu hacen las operaciones con la base de datos.
  - **`./libs/users.data.php`** : archivo que contendrá las funciones par validar y sanear los datos recibidos externamente (vía POST).

Los archivos **`users.dao.php`** y **`users.data.php`** los debes crear vacíos d momento (en los siguientes ejercicios irás creando funciones en dicho

archivos). De igual forma, los formularios de la carpeta `./forms` se irá creando en los siguientes ejercicios.

> **Es importante que sepas que los archivos `conn.php`, `users.dao.php`, `users.data.php` y `db.conf.php` deben incluirse siempre usando `include_once`, pero no deben incluirse entre sí, deberán incluirse desde los ejercicios. Por ejemplo: `conn.php` no debe hacer un `include_once` de `db.conf.php`.**

Después de crear la estructura anterior, en el archivo **`conn.php`**:

- Crea una función llamada **`connect`** (sin parámetros) que retorne un conexión **`PDO`** para la base de datos configurada con las constante **`DB_DSN`**, **`DB_USER`** y **`DB_PASSWD`**. Esta función, además deberá indicar siguiente atributo a la hora de crear la conexión **`PDO`** con la base d datos:
  - **`array(PDO::ATTR_ERRMODE => PDO::ERRMODE_EXCEPTION)`**
- En caso de que la conexión no pueda realizarse, el método deber retornar **`false`**.

El parámetro **`array(PDO::ATTR_ERRMODE => PDO::ERRMODE_EXCEPTION)`** hará qu cuando se produzca un error al usar **`PDO`** para realizar una consulta o establecer la conexión con la base de datos, se genere una excepción (qu podrás capturar usando **`try-catch`**):

```
try {
          ... uso de PDO ...
} catch (PDOException $ex) {
          ... tratamiento del error ...
}
```

> **Importante: dentro de la función `connect` deberás hacer uso de un `try-catch` para asegurarte de que no hay error al establecer la conexión.**

> **Importante: en el resto de los ejercicios deberás usar la función `connect` para conectar a la base de datos.**

Debes conocer

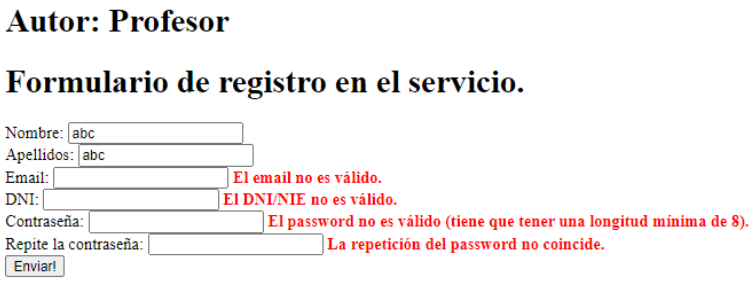
Si has instalado XAMPP, es muy sencillo crear y administrar una base de datos MySQL o MariaDB, dado que puedes usar phpMyAdmin para ello (y aunque no estés usando XAMPP puedes instalar dicha aplicación web).

Es muy sencillo acceder, simplemente arranca el servidor Apache y el servidor MySQL desde el panel de control de XAMPP y accede a la página:

http://localhost/phpmyadmin

1.2.- Ejercicio 2: registro de usuarios

En este ejercicio tienes que hacer un script PHP llamado



`ejercicio2.php` que permita a un empleado registrarse en la aplicación. E proceso de registro tiene los siguientes requisitos:

- Cuando el script **`ejercicio2.php`** no reciba datos, o estos sea incorrectos, debe mostrar un formulario para solicitar al empleado qu desee registrarse los siguientes datos:
  - Nombre del empleado.
  - Apellidos del empleado.
  - Email del empleado.
  - DNI o NIE del empleado.
  - Contraseña deseada.
  - Repetición de la contraseña.
- El formulario anterior enviará los datos a este mismo scri (**`ejercicio2.php`**) a través del método **`POST`** para que sean procesados. E fragmento de código correspondiente al formulario se almacenará e un archivo separado en la carpeta **`forms`**.
- Cuando los datos sean incorrectos se volverá a mostrar el formulari con los datos correctos rellenos y los datos incorrectos vacíos, except para la contraseña y su repetición que siempre aparecerán vacíos.

Antes de continuar con el proceso de registro, los datos deben **sanearse validarse**. Este proceso es sumamente importante, dado que debemc evitar a toda costa cualquier ataque de inyección SQL. Esta parte, deb realizarse siguiendo las siguientes premisas:

- **Por motivos de seguridad**, los datos recibidos siempre se **saneará y validarán** de la siguiente forma:
  - Todos los datos, excepto la contraseña y su repetición, s limpiarán de espacios anteriores y posteriores.
  - En todos los datos excepto el email, la contraseña y s repetición, se eliminarán posibles etiquetas HTML y se escapará caracteres especiales (como comillas simples por **`"&#39;"`** comillas dobles por **`"&quot;"`**). Esto puede realizarse fácilment con la función **`filter_input`**.
  - El email se convertirá en minúsculas.

    - El DNI o NIE se convertirá en minúsculas.
- Los datos recibidos se considerarán incorrectos cuando, **después d ser saneados**, tienen alguno de los siguientes problemas:
    - El nombre tiene menos de dos caracteres o más de 30 .
    - Los apellidos tiene menos de dos caracteres o más de 50.
    - El email no tiene formato de email (esto se puede comprobar co una expresión regular o bien con la función `filter_var`). E necesario ser estrictos aquí, dado que por seguridad, un email n debe contener caracteres no esperados como "comillas simples o "comillas dobles".
    - El email tiene más de 50 caracteres.
    - El DNI o NIE en minúsculas no tiene el formato esperado (esto s puede comprobar con una expresión regular, aunque también s puede usar la función `filter_var`). Para validar el DNI o NI puedes usar la siguiente expresión regular:
        - `/^(\d{8}[a-zA-Z]{1}|[XxYyZz]{1}\d{7}[a-zA-Z]{1})$/`
    - La contraseña tiene menos de 8 caracteres.
    - La repetición de la contraseña no coincide con la contraseña.
- Cuando los datos recibidos sean incorrectos, se mostrarán al usuari mensajes de error acordes a los errores encontrados.

Una vez que los datos han sido saneados y validados, se procederá realizar el registro (insertar los datos en la base de datos):

- Se guardarán en la base de datos los siguientes datos:
    - en el campo "`nombre`" de la base de datos, se almacenará nombre saneado proporcionado por el empleado,
    - en el campo "`apellidos`" se almacenarán los apellidos saneado proporcionado por el empleado,
    - en el campo "`DNI`" se almacenará el DNI o NIE saneado proporcionado por el empleado,
    - en el campo "`email`" se almacenará el email saneado proporcionado por el empleado,
    - en el campo "`IRRD`" se almacenará un número aleatorio de cifras,
    - y en el campo "`password`" de la base de datos se almacenará u resumen SHA2 del email proporcionado concatenado a l contraseña proporcionada en el formulario. Esto quiere decir qu no se almacenará la contraseña directamente. Para esto puede utilizar la función SHA2 de MySQL para hacer el cálculo:
        - Por ejemplo, si el email es `"email@correo.es"` y la contraseñ es `"holaHOLA"`, en el campo `password` de la base de datos s almacenará `SHA2('email@correo.esholaHOLA',0)`, donde `SHA` en este caso es una función de MySQL.
- Al guardar en la base de datos puede ocurrir que ya exista un usuari con dicho `email` o dicho `dni`, en ese caso, debe informarse al usuario d que no se puede crear el usuario dado que ya existe un usuario co ese email o DNI/NIE.

A la hora de realizar el ejercicio:

- Se valorará especialmente que se estructure bien el ejercicio e funciones separadas reutilizables (reutilizar adecuadamente reduce tiempo de desarrollo drásticamente).
- Al ejecutar una consulta o sentencia SQL con PDO debes usar `try` `catch` para capturar posibles excepciones.
- Las funciones para almacenar datos deben crearse en el archiv `users.dao.php`.

- Las funciones para validar y sanear datos deben crearse en el archiv `users.data.php`.
- Los fragmentos correspondientes a formularios deben almacenarse e la carpeta `./forms`.
- No es obligatorio, pero se valorará positivamente el uso d transacciones a la hora de ejecutar las consultas.

Para saber más

En muchas aplicaciones web es terriblemente común almacenar la contraseña en la base de datos directamente, y esto es un grave error de diseño, dado que un fallo de seguridad puede revelar miles de contraseñas. Por ello, es común no almacenar la contraseña, sino un resumen o HASH de la misma en combinación con otros datos. El algoritmo HASH más utilizado hoy día para esto es SHA-256. Antes se utilizaba el algoritmo MD5, pero se encontraron vulnerabilidades en dicho algoritmo y hoy día no se recomienda. Para saber más sobre las funciones HASH, haz clic en el siguiente enlace:

¿Qué es un hash?

1.3.- Ejercicio 3: obtener el código de validación

En este

Autor: Profesor

Formulario para obtener código de validación.

Email: El email no es válido.
Contraseña:
Enviar!

tercer ejercicio tienes que hacer un script llamado **`ejercicio3.php`** en el qu se permita a los empleados obtener el código de validación (campo **`IRRD`** d la base de datos). Esta sección tiene los siguientes requisitos:

- Cuando el script no reciba datos o estos sean incorrectos, se mostrar un formulario que solicite:
  - Email del empleado,
  - Contraseña del empleado.
- El formulario anterior enviará los datos a este mismo scri (**`ejercicio3.php`**) a través del método **`POST`** para que sean procesados. E fragmento de código correspondiente al formulario se almacenara e un archivo separado en la carpeta **`forms`**.
- Cuando los datos sean incorrectos volverá a aparecer siempre formulario vacío.

Antes de mostrar el código de verificación, por seguridad, deberá hacers obligatoriamente la validación y saneado de los datos recibido vía POST:

- **Por motivos de seguridad**, los datos recibidos siempr se **sanearán** de la siguiente forma:
  - El email se limpiará de espacios anteriores y posteriores.
  - El email se convertirá en minúsculas.
- Los **datos recibidos se considerarán no válidos cuando**, después de ser saneados, tienen alguno de los siguientes problemas:
  - El email no tiene formato de email (esto se puede comprobar co una expresión regular o bien con la función **`filter_var`**). E necesario ser estrictos aquí, dado que por seguridad, un email n debe contener caracteres no esperados como "comillas simples o "comillas dobles".
  - La contraseña indicada como "contraseña actual" no correspond con la almacenada en la base de datos. Para realizar est comprobación es necesario:
    - Con los datos recibidos vía POST es necesario calcular resumen SHA2 del email concatenado en la contraseñ facilitada por el usuario, dado que la contraseña no s almacena directamente en la base de datos.

        - El resumen SHA2 calculado se cotejará con el almacenad en la base de datos para dicho usuario.
    - Cuando el email sea incorrecto o la contraseña no coincida, s mostrará al usuario un mensaje advirtiendo del problema.

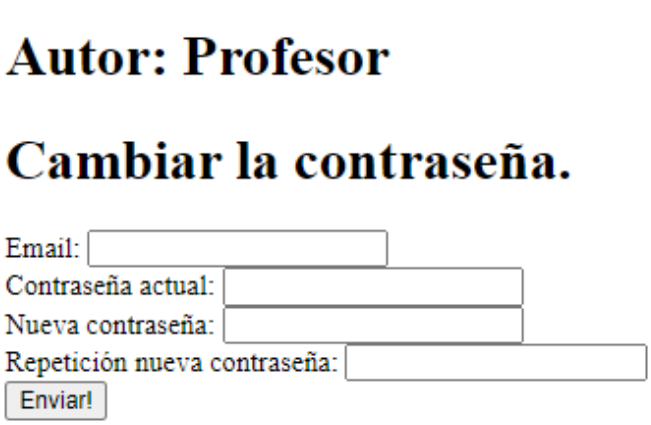
Una vez que los datos han sido saneados y validados, se procederá mostrar al usuario el código de validación almacenado en la base de datc para este usuario (`IRRD`).

A la hora de realizar el ejercicio ten en cuenta que:

- Se valorará especialmente que se estructure bien el ejercicio e funciones separadas reutilizables (reutilizar adecuadamente reduce tiempo de desarrollo drásticamente).
- Al ejecutar una consulta o sentencia SQL con PDO debes usar `try` `catch` para capturar posibles excepciones.
- Las funciones para almacenar datos deben crearse en el archiv `users.dao.php`.
- Las funciones para validar y sanear datos deben crearse en el archiv `users.data.php`.
- Los fragmentos correspondientes a formularios deben almacenarse e la carpeta `./forms`.

1.4.- Ejercicio 4: cambiar la contraseña

En este ejercicio tienes que hacer un script PHP



llamado **`ejercicio4.php`** que permita a un empleado cambiar su contraseñ en la aplicación web. El proceso de cambio de contraseña tiene lo siguientes requisitos:

- Cuando el script **`ejercicio4.php`** no reciba datos, o estos sea incorrectos, debe mostrar un formulario que solicite los siguiente datos:
  - Email del empleado.
  - Contraseña actual.
  - Contraseña nueva.
  - Repetición de la contraseña nueva.
- El formulario anterior enviará los datos a este mismo scri (**`ejercicio4.php`**) a través del método **`POST`** para que sean procesados. E fragmento de código correspondiente al formulario se almacenara e un archivo separado en la carpeta **`forms`**.
- Cuando los datos sean incorrectos se volverá a mostrar el formulari con los campos vacíos.

Antes de continuar con el proceso de registro, los datos deben **sanearse validarse**. Esta parte, debe realizarse siguiendo las siguiente premisas:

- **Por motivos de seguridad**, los datos recibidos siempr se **sanearán** de la siguiente forma:
  - El email se limpiará de espacios anteriores y posteriores.
  - El email se convertirá en minúsculas.
- Los **datos recibidos se considerarán no válidos cuando**, después de ser saneados, tienen alguno de los siguientes problemas:
  - El email no tiene formato de email (esto se puede comprobar co una expresión regular o bien con la función **`filter_var`**). E necesario ser estrictos aquí, dado que por seguridad, un email n debe contener caracteres no esperados como "comillas simples o "comillas dobles".

    - La contraseña indicada como "contraseña actual" no corresponde con la almacenada en la base de datos. Para realizar esta comprobación es necesario:
        - Con los datos recibidos vía POST es necesario calcular el resumen SHA2 del email concatenado en la contraseña facilitada por el usuario, dado que la contraseña no se almacena directamente en la base de datos.
        - El resumen SHA2 calculado se cotejará con el almacenado en la base de datos para dicho usuario.
    - La contraseña nueva y su repetición no coinciden.
- Cuando los datos recibidos no sean válidos se mostrará al usuario un mensaje de error acorde al error encontrado.

Una vez que los datos han sido saneados y validados, se procederá a realizar las funciones esperadas:

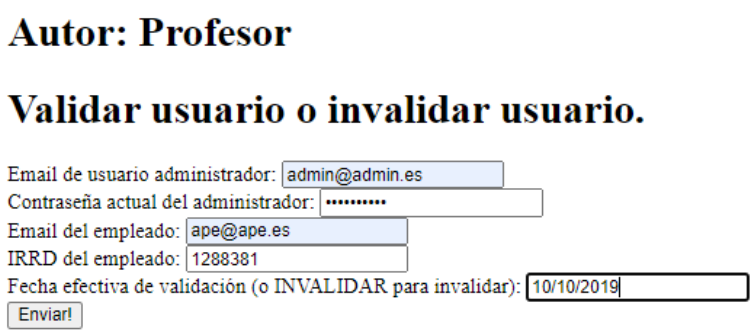
- Se modificará la contraseña en la base de datos, almacenando en el campo **`password`** de la tabla correspondiente el resumen SHA2 de la concatenación del email del usuario y la nueva contraseña.
- Se generará un nuevo IRRD para el usuario.
- Después de realizar el cambio, se mostrará al usuario el resultado de la operación.

A la hora de realizar el ejercicio:

- Se valorará especialmente que se estructure bien el ejercicio en funciones separadas reutilizables (reutilizar adecuadamente reduce el tiempo de desarrollo drásticamente).
- Al ejecutar una consulta o sentencia SQL con PDO debes usar **`try catch`** para capturar posibles excepciones.
- Las funciones para almacenar datos deben crearse en el archivo **`users.dao.php`**.
- Las funciones para validar y sanear datos deben crearse en el archivo **`users.data.php`**.
- Los fragmentos correspondientes a formularios deben almacenarse en la carpeta **`./forms`**.
- No es obligatorio, pero se valorará positivamente el uso de transacciones a la hora de ejecutar las consultas.

1.5.- Ejercicio 5: validar e invalidar un usuario

En este ejercicio tienes que hacer un script PHP



llamado **`ejercicio5.php`** que permita a un administrador cambiar validar a u usuario. El proceso de validación tiene los siguientes requisitos:

- Cuando el script **`ejercicio5.php`** no reciba datos, o estos sea incorrectos, debe mostrar un formulario que solicite los siguiente datos:
  - Email del supervisor.
  - Contraseña del supervisor.
  - Email del empleado a validar.
  - IRRD del empleado a validar.
  - Fecha efectiva de validación (puede ser una fecha con format **`dd/mm/aaaa`** o bien el texto **`INVALIDAR`** en mayúsculas.
- El formulario anterior enviará los datos a este mismo scri (**`ejercicio5.php`**) a través del método **`POST`** para que sean procesados. E fragmento de código correspondiente al formulario se almacenara e un archivo separado en la carpeta **`forms`**.
- Cuando los datos sean incorrectos se volverá a mostrar el formulari con todos los datos vacíos.

Antes de continuar con el proceso de registro, los datos deben **sanearse validarse**. Esta parte, debe realizarse siguiendo las siguiente premisas:

- **Por motivos de seguridad**, los datos recibidos siempr se **sanearán** de la siguiente forma:
  - Ambas direcciones de email se limpiará de espacios anteriores posteriores.
  - Ambas direcciones de email se convertirá en minúsculas.
  - El IRRD se limpiará de espacios anteriores y posteriores.
  - La fecha efectiva de validación se limpiará de espacios anteriore y posteriores.
- Los **datos recibidos se considerarán no válidos cuando**, después de ser saneados, tienen alguno de los siguientes problemas:
  - **Alguno de los email no tiene formato de email** (esto se pued comprobar con una expresión regular o bien con

función `filter_var`). Es necesario ser estrictos aquí, dado que po seguridad, un email no debe contener caracteres no esperado como "comillas simples" o "comillas dobles".
    - **El IRRD del email del empleado a validar no está compuest por 8 números** (para realizar esto puedes usar una expresió regular o bien con la función `filter_var`).
    - **Cuando la fecha es distinta al texto "`INVALIDAR`"** en mayúscula hay que comprobar que la fecha tiene un formato válido. En es caso hay que comprobar que **la fecha efectiva de validación n tiene el formato "`dd/mm/aaaa`", o es una fecha imposible.** Par realizar esta comprobación puedes usar la funció `date_parse_from_format` usando el formato "`j/m/Y`". Fíjate que dich método retorna un array con las partes de la fecha separadas, que en dicho array hay dos elementos destinados a indicar si l fecha es errónea (`warnings` y `errors`), aparecerán sendos errore cuando la fecha no tenga el formato esperado o cuando la fech no sea una fecha posible (31 de febrero, por ejemplo).
    - **La contraseña indicada para el supervisor no correspond con la almacenada en la base de datos**. Para realizar est comprobación es necesario:
        - Con los datos recibidos vía POST es necesario calcular resumen SHA2 del email concatenado en la contraseñ facilitada por el usuario, dado que la contraseña no s almacena directamente en la base de datos.
        - El resumen SHA2 calculado se cotejará con el almacenad en la base de datos para el supervisor.
    - **El email indicado como email de supervisor no tien permisos de administración**. Para comprobar si el email d supervisor corresponde al de un superusuario es necesari verificar en la base de datos que dicho registro tiene el camp `superuser` a `1` o `true` (MySQL almacena los booleanos como u entero).
    - **El email indicado es el de un supervisor**. No puede invalidars ni validarse un usuario supervisor.
    - **El email del empleado no existe en la base de datos, o bien el IRRD indicado no coincide con el IRRD almacenado en l base de datos para el email de empleado facilitado.**
- Cuando los datos recibidos no sean válidos se mostrará al usuario u mensaje de error acorde al error encontrado.

Una vez que los datos han sido saneados y validados, se procederá realizar las funciones esperadas:

- Si la fecha de validación coincide con el texto "`INVALIDAR`", el camp `validacion` de la base de datos se pondrá a `NULL`, de esa forma s realizará la invalidación del registro.
- Si la fecha de validación tiene el formato de fecha válido, el camp `validacion` de la base de datos se establecerá a la fecha indicada en formulario.
- Después de realizar el cambio, se mostrará al usuario el resultado d la operación (si la invalidación o la validación se llevo a cab correctamente).

A la hora de realizar el ejercicio:

- Se valorará especialmente que se estructure bien el ejercicio e funciones separadas reutilizables (reutilizar adecuadamente reduce tiempo de desarrollo drásticamente).

- Al ejecutar una consulta o sentencia SQL con PDO debes usar **`try catch`** para capturar posibles excepciones.
- Las funciones para almacenar datos deben crearse en archivo **`users.dao.php`**.
- Las funciones para validar y sanear datos deben crearse en archivo **`users.data.php`**.
- Los fragmentos correspondientes a formularios deben almacenarse e la carpeta **`./forms`**.
- No es obligatorio, pero se valorará positivamente el uso d transacciones a la hora de ejecutar las consultas.

Debes conocer

Ten en cuenta que MySQL almacena las fechas con formato **`"aaaa-mm-dd"`**, por lo que si el usuario facilita la fecha en formato "dd/mm/aaaa" tendrás que transformarla. Para ello puedes aprovecharte de que el método `date_parse_from_format` retorna un array con la fecha "partida", de manera que puedes recomponerla a tu manera:

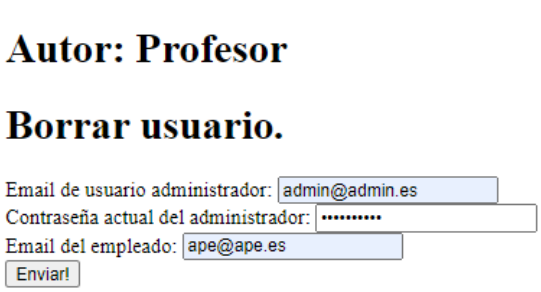
```
Array
(
    [year] => 2009
    [month] => 1
    [day] => 6
    [hour] => 13
    [minute] => 0
    [second] => 0
    [fraction] =>
    [warning_count] => 0
    [warnings] => Array
        (
        )

    [error_count] => 0
    [errors] => Array
        (
        )

    [is_localtime] => 1
    [zone_type] => 1
    [zone] => 3600
    [is_dst] =>
)
```

Por otro lado, a la hora de establecer un campo a **`NULL`** en la base de datos, debes poner el valor **`NULL`** directamente en la consulta, por ejemplo:

```
"UPDATE coche SET  propietario = NULL WHERE ..."
```

1.6.- Ejercicio 6: borrar un usuario

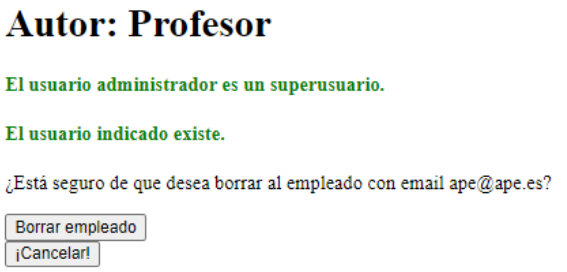
En este ejercicio tienes que hacer dos scripts PHP llamados `ejercicio6.php` y `ejercicio6confirm.php` que permita a un administrador borrar a un usuario de la base de datos. El proceso de borrado tiene los siguientes requisitos:



- Cuando el script `ejercicio6.php` no reciba datos, o estos sean incorrectos, deb mostrar un formulario que solicite los siguientes datos:
  - Email del supervisor.
  - Contraseña del supervisor.
  - Email del empleado a borrar.
- El formulario anterior enviará los datos al script `ejercicio6.php` a través del método `POST` para que sean procesados. El fragmento de código correspondiente al formulario se almacenara en un archivo separad en la carpeta `forms`.
- En el script `ejercicio6.php` se realizará el saneado y validación de lo datos, **pero no se realizará ningún borrado en la base de datos**. S los datos son válidos, el script `ejercicio6.php` mostrará:
  - Un texto que pregunte al administrador si desea continuar con l operación, mostrando el email del empleado a borrar.
  - Un botón con el texto `"Borrar empleado"`. El botón `"Borra empleado"` irá en un formulario en el cuál todos los datos s reenviarán vía POST al script `ejercicio6confirm.php` a través d campos `hidden`. Será en `ejercicio6confirm.php` donde se hag efectiva la operación de borrado.
  - Un botón con el texto `"Cancelar borrado"`. Al hacer clic en est botón se volverá a cargar al script `ejercicio6.php`, donde s mostrará el formulario inicial vacío.
- En el script `ejercicio6confirm.php` se volverá a realizar el saneado validado de los datos recibidos de la confirmación, y si estos so correctos, se procederá a borrar al empleado de la base de datos.

Como en ejercicios anteriores, **por seguridad, los datos deben sanearse validarse** (tanto en `ejercicio6.php` como en `ejercicio6confirm.php`):

- **Por motivos de seguridad**, los datos recibidos siempr se **sanearán** de la siguiente forma:
  - Las direcciones de email se limpiará de espacios anteriores posteriores, y se convertirán a minúsculas.
- Los **datos recibidos se considerarán no válidos cuando**, después de ser saneados, tienen alguno de los siguientes problemas:



- **Alguno de los email no tiene formato de email** (esto se puede comprobar con una expresión regular o bien con la función `filter_var`). Es necesario ser estrictos aquí, dado que por seguridad, un email no debe contener caracteres no esperados como "comillas simples" "comillas dobles".
- **La contraseña indicada para el supervisor no correspond con la almacenada en la base de datos**. Para realizar est comprobación es necesario:
  - Con los datos recibidos vía POST es necesario calcular resumen SHA2 del email concatenado en la contraseñ facilitada por el usuario, dado que la contraseña no s almacena directamente en la base de datos.
  - El resumen SHA2 calculado se cotejará con el almacenad en la base de datos para el supervisor.
- **El email indicado como email de supervisor no tien permisos de administración**. Para comprobar si el email d supervisor corresponde al de un superusuario es necesari verificar en la base de datos que dicho registro tiene el camp `superuser` a `1` o `true` (MySQL almacena los booleanos como u entero).
- **El email de empleado no corresponde a ningún emplead almacenado en la base de datos.** Si el empleado no est registrado en la base de datos, no puede borrarse.
- **El email del empleado a borrar coincide con el email de u administrador.** Debes tener en cuenta que un administrador n puede borrar a otro administrador ni a si mismo.

- Cuando los datos recibidos no sean válidos se mostrará al usuario u mensaje de error acorde al error encontrado.

Una vez que los datos han sido saneados y validados, se procederá realizar las funciones esperadas:

- En el script `ejercicio6.php` se mostrarán las opciones de confirma borrado o cancelarlo conforme a lo especificado anteriormente.
- En el script `ejercicio6confirm.php` se realizará el borrado del emplead en cuestión, mostrando un mensaje acorde al resultado de dich operación.

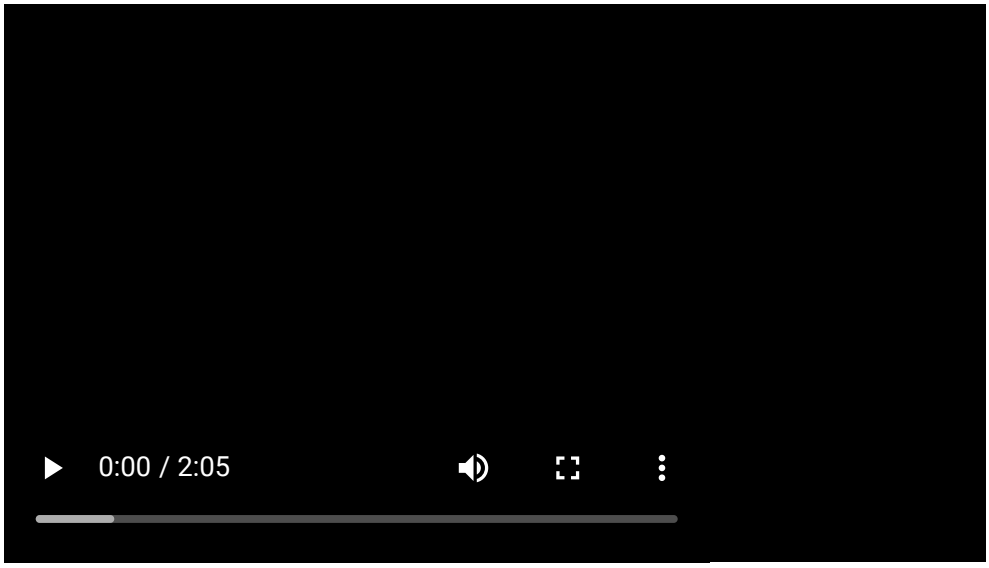
A la hora de realizar el ejercicio:

- Se valorará especialmente que se estructure bien el ejercicio e funciones separadas reutilizables (reutilizar adecuadamente reduce tiempo de desarrollo drásticamente).
- Al ejecutar una consulta o sentencia SQL con PDO debes usar `try` `catch` para capturar posibles excepciones.
- Las funciones para almacenar datos deben crearse en archivo `users.dao.php`.
- Las funciones para validar y sanear datos deben crearse en archivo `users.data.php`.
- Los fragmentos correspondientes a formularios deben almacenarse e la carpeta `./forms`.

- No es obligatorio, pero se valorará positivamente el uso d transacciones a la hora de ejecutar las consultas.

En el siguiente vídeo se ilustra el funcionamiento del ejercicio 6:

Vídeo que ilustra el funcionamiento del ejercicio 6

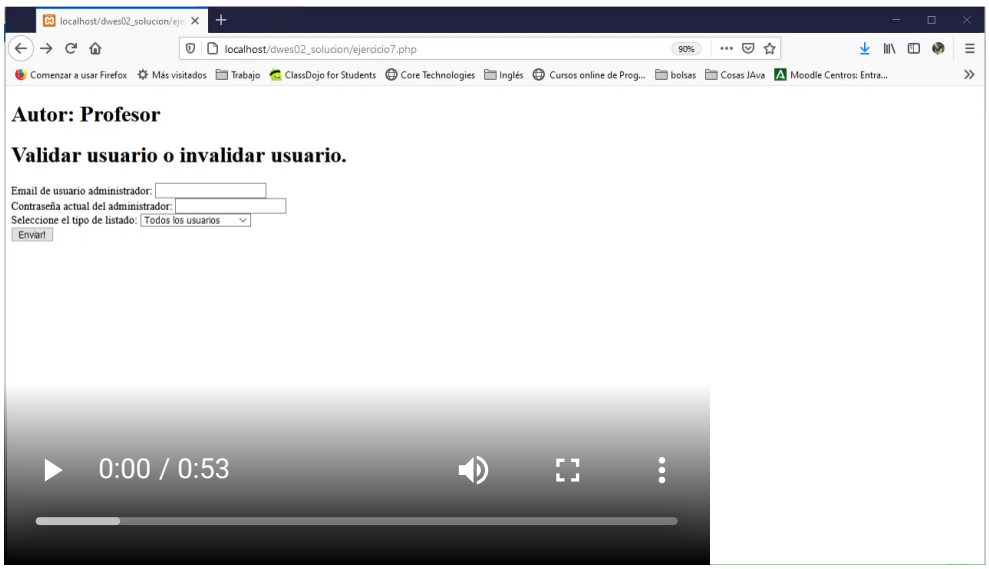


Salvador Romero (Elaboración propia). *Vídeo que ilustra el funcionamiento del ejercicio 6* (CC BY-SA)

1.7.- Ejercicio 7: listar usuarios

En este ejercicio tienes que hacer un script PHP llamado **`ejercicio7.php`** qu permita a un administrador obtener la lista de usuarios registrados en sistema. A continuación tienes un vídeo que ilustra el funcionamiento de est ejercicio:

Vídeo que ilustra el funcionamiento del ejercicio 7



Salvador Romero Villegas (elaboración propia). *Vídeo que ilustra el funcionamiento del ejercicio 7* (CC BY-SA)

- Cuando el script **`ejercicio7.php`** no reciba datos, o estos sea incorrectos, debe mostrar un formulario que solicite los siguiente datos:
  - Email del supervisor.
  - Contraseña del supervisor.
  - Un desplegable con las siguientes 3 opciones:
    - Mostrar todos los usuarios.
    - Mostrar usuarios no validados.
    - Mostrar usuarios validados.
- El formulario anterior enviará los datos al mism script (**`ejercicio7.php`**) a través del método **`POST`** para que sea procesados. El fragmento de código correspondiente al formulario s almacenara en un archivo separado en la carpeta **`forms`**.
- En el script **`ejercicio7.php`** se realizará el saneado y validación de lo datos. Si los datos son válidos, el script mostrará, dependiendo de l opción elegida:
  - Un listado con todos los usuarios.
  - Un listado con los usuarios no validados.
  - Un listado con los usuarios validados.

Como en ejercicios anteriores, **por seguridad, los datos deben sanearse validarse**:

- **Por motivos de seguridad**, los datos recibidos siempr se **sanearán** de la siguiente forma:
  - La dirección de email se limpiará de espacios anteriores posteriores, y se convertirán a minúsculas.
- Los **datos recibidos se considerarán no válidos cuando**, después de ser saneados, tienen alguno de los siguientes problemas:
  - **El email no tiene formato de email** (esto se puede comproba con una expresión regular o bien con la función `filter_var`). E necesario ser estrictos aquí, dado que por seguridad, un email n debe contener caracteres no esperados como "comillas simples o "comillas dobles".
  - **El tipo de listado no es uno de los disponibles.** Se verificar que se ha pasado una de las tres opciones disponibles: mostra todos, mostrar no validados o mostrar validados.
    - Nota: la opción recibida vía POST no deberá pasars directamente a la consulta SQL, puede ser un problema d seguridad.
  - **La contraseña indicada para el supervisor no correspond con la almacenada en la base de datos**. Para realizar est comprobación es necesario:
    - Con los datos recibidos vía POST es necesario calcular resumen SHA2 del email concatenado en la contraseñ facilitada por el usuario, dado que la contraseña no s almacena directamente en la base de datos.
    - El resumen SHA2 calculado se cotejará con el almacenad en la base de datos para el supervisor.
  - **El email indicado como email de supervisor no tien permisos de administración**. Para comprobar si el email d supervisor corresponde al de un superusuario es necesari verificar en la base de datos que dicho registro tiene el camp `superuser` a `1` o `true` (MySQL almacena los booleanos como u entero).
- Cuando los datos recibidos no sean válidos se mostrará al usuario u mensaje de error acorde al error encontrado.

Una vez que los datos han sido saneados y validados, se procederá realizar las funciones esperadas:

- Si los datos son válidos, el script mostrará, dependiendo del tipo d lista a mostrar, una tabla HTML con:
  - Todos los usuarios.
  - Los usuarios no validados.
  - Los usuarios validados.
- Independientemente del tipo de lista a mostrar para cada usuario s incluirá la siguiente información:
  - Nombre del empleado o empleada.
  - Apellidos del empleado o empleada.
  - Email.
  - DNI o NIE.
  - Fecha de creación del registro (en formato DD/MM/AAAA).
  - Fecha de validación (en formato DD/MM/AAAA).
- Además, para cada registro aparecerán 3 botones (excepto para lo usuarios administradores que no aparecerá ninguno):
  - Si se trata de un usuario no validado habrá:
    - Un botón para validar al usuario.
    - Un botón para borrar al usuario.

    - Si se trata de un usuario validado:
        - Un botón para invalidar al usuario.
        - Un botón para borrar al usuario.
- Además, al principio del formulario aparecerán 3 botones:
    - Uno para mostrar un listado con todos los usuarios.
    - Otro para mostrar un listado con los usuarios no validados.
    - Y un último botón para mostrar los usuarios validados.
    - Estos tres botones podrán ser también realizados con u desplegable y un botón.

Cada uno de los botones anteriores enviará vía POST, y usando campos tip `hidden`, los datos correspondientes al script correspondiente:

- Para validar o invalidar a un usuario se enviarán los datc correspondientes al script `ejercicio5.php`.
- Para borrar a un usuario se enviarán los datos correspondientes a script `ejercicio6.php`.
- Para cambiar el tipo de listado se enviarán los datos correspondiente al script `ejercicio7.php` (este mismo script).

A la hora de realizar el ejercicio:

- Se valorará especialmente que se estructure bien el ejercicio e funciones separadas reutilizables (reutilizar adecuadamente reduce e tiempo de desarrollo drásticamente).
- Al ejecutar una consulta o sentencia SQL con PDO debes usar `try` `catch` para capturar posibles excepciones.
- Las funciones para almacenar datos deben crearse en e archivo `users.dao.php`.
- Las funciones para validar y sanear datos deben crearse en e archivo `users.data.php`.
- Los fragmentos correspondientes a formularios deben almacenarse e la carpeta `./forms`.

Debes conocer

Como se comentó antes, en la base de datos MySQL está almacenada la fecha en formato `aaaa-mm-dd`, sin embargo, en cualquier aplicación web, debe mostrarse la fecha en formato correspondiente a la localización del usuario. En España se usa el formato `dd/mm/aaaa`, y por ello, es necesario convertir la fecha almacenada en MySQL al formato usado en España.

PHP tiene una clase llamada Datetime que facilita estas transformaciones, pero, no la usaremos para este ejercicio. En su lugar puedes convertirla "manualmente" usando la función date_parse_from_format o bien, usando la función `DATE_FORMAT` de MySQL. Aquí tienes un ejemplo del uso de esta última:

Uso de DATE_FORMAT de MySQL (en inglés).

2.- Información de interés

Recursos necesarios y recomendaciones

- Como ya sabes, para escribir aplicaciones en PHP necesitarás u entorno XAMPP, LAMPP o similar. No te vamos a pedir que instales u entorno u otro, o que trabajes con un sistema operativo concreto, per si que todos los **archivos vayan codificados en UTF-8** y que **la rutas a los archivos sean siempre relativas**.
- Como plataforma de desarrollo (para escribir código), te aconsejamo **NetBeans 12**. Es la misma herramienta que se usa en otros módulos no tendrás que cambiar de herramienta.
- No abordes la tarea hasta que hayas repasado todos los contenidos, hecho al menos uno de los intentos para el examen en línea, de form que hayas tenido ocasión de consultar y resolver en los foros cualquie duda que te haya podido surgir.
- No olvides revisar los **ejercicios resueltos** que se incluyen en l unidad, tanto dentro de los apartados como en los anexos específicos
- Todos los ejercicios son susceptibles de mejora, y seguro que te gust hacer los ejercicios perfectos, pero para que no te falte tiempo, primer resuelve el ejercicio tal y como se pide, y luego, dedica tiempo cualquier mejora que quieras hacer.
- **Quien desee tener el entorno lo más parecido posible a lo que s va a encontrar en los ordenadores de las sedes dónde s realizarán los exámenes a ordenador en junio, debe descargar lo instaladores de la página que se indica en el apartado "software de la programación didáctica** http://iesaguadulce.es/softwaredistancia/
  - La versión pre-instalada tiene PHP 5.4, aunque es posible que s actualice la versión, no es nada seguro por ahora y se avisar conforme se acerque el examen.
    - Aunque la versión 5 de PHP sigue siendo la más usad globalmente, cada vez la 7 está siendo más popular sobr todo por su eficiencia, y dentro de lo que cabe es bastant compatible hacia atrás.
  - En el examen será posible trabajar con versiones portables d XAMPP desde un pendrive.

En esta actividad se recomienda el uso de funciones `filter_input` `filter_var` para validar y sanear los datos:

- Usa `filter_input` para validar o sanear los datos recibidos vía GET POST. Esta función tiene 3 parámetros:
  - Origen del dato (`INPUT_GET`, `INPUT_POST`,...) para especificar si s trata de parámetros recibidos vía GET o POST.
  - Cadena con el nombre del parámetro.
  - Validación o saneado a realizar en dicha variable.

    - Por ejemplo:
        - `$alias= filter_input(INPUT_POST, 'alias' FILTER_SANITIZE_STRING);`
        - En este ejemplo se sanea el dato borrando etiquetas HTM y otros elementos no deseados.
- Usa **`filter_var`** para validar o sanear una variable. Esta función tiene parámetros:
    - Nombre de la variable.
    - Validación o saneado a realizar sobre dicha variable.
    - Por ejemplo:
        - `$userIp = filter_var($userIp, FILTER_VALIDATE_IP);`
        - En este ejemplo se comprueba que la cadena contiene un dirección IP válida.

Cuando se ejecutan estos métodos pueden ocurrir tres cosas:

- Que no pase el filtro de validación (si se trata de un filtro de validación En ese caso, el método retornará **`false`**.
- En el caso de **`filter_input`**, puede ocurrir que no exista el parámetr indicado en **`$_POST`**, **`$_GET`**, etc., en dicho caso retornará **`null`**.
- Que el saneado o validación se lleve correctamente, en tal caso, s retornará el contenidos saneado.

Recuerda que en PHP la comprobación de valor y tipo se hace con "`===`" Échale un vistazo a la página de PHP siguiente para ver todos los filtrado de validación y saneado que hay: tipos de filtros.

Aquí tienes un pequeño ejemplo de uso:

```
$newip = filter_input(INPUT_POST, 'newip', FILTER_VALIDATE_IP);

if ($newip===null) {

//El parámetro no se envío por POST

} else if ($newip===false) {

//El parámetro no se valido, por lo que no es una ip correcta

} else {

 //Correcto! Aquí tendrías que realizar otras validaciones extra que necesite

 }
```

Por último, aquí tienes algunos enlaces que pueden resultarte útiles:

- http://www.mclibre.org/consultar/php/lecciones/php-db-pdo.html
- http://www.mclibre.org/consultar/php/lecciones/php-db-inyeccion-sql.html
- Ejemplos de PHP, con bases de datos:
    - Bases de datos 1
    - Bases de datos 2 - Comentarios
    - Bases de datos 3

Indicaciones de entrega

Una vez realizada la tarea, el envío se realizará a través de la plataforma. Comprime la carpeta del proyecto en un fichero .zip y nómbralo siguiendo las siguientes pautas:

Apellido1_Apellido2_Nombre_DWES_Tarea02

3.- Evaluación de la tarea

Criterios de evaluación implicados

- Se establecen conexión con un SGBD correctamente.
- Se utilizan de bases de datos relacionales para persistir la información usada en la aplicación web.
- Se hacen correctamente operaciones de recuperación y modificación de información.
- Se recuperan datos de la base de datos y se utilizan dicho datos en los resultados mostrados al usuario web.
- Se utilizan formularios para modificar o eliminar la información de la base de datos.

Peggy_Marco (Pixabay License

- Se ejecutan sentencias SQL de selección, inserción, modificación eliminación de datos, y se gestionan posibles errores existentes.
- Se hace uso de transacciones cuando la situación lo requiera.
- Se toman medidas para evitar posibles problemas de seguridad en l aplicación web.

¿Cómo valoramos y puntuamos tu tarea?

En esta puedes observar la puntuación máxima asignada a cada ejercici de la tarea:

Rúbrica de la tarea	
Se ha implementado el ejercicio 1 apropiadamente y sin errores.	**hasta 0,5 puntos**
Se ha implementado el ejercicio 2 apropiadamente y funciona correctamente.	**hasta 2,0 puntos**
Se ha implementado el ejercicio 3 apropiadamente y funciona correctamente.	**hasta 1,0 puntos**

Se ha implementado el ejercicio 4 apropiadamente y funciona correctamente.	**hasta 1,0 puntos**
Se ha implementado el ejercicio 5 apropiadamente y funciona correctamente.	**hasta 1,75 puntos**
Se ha implementado el ejercicio 6 apropiadamente y funciona correctamente.	**hasta 1,75 puntos**
Se ha implementado el ejercicio 7 apropiadamente y funciona correctamente.	**hasta 2,0 puntos**