Actividad 1.1.1
Desarrollo de una aplicación PHP

CONTENIDO

1.- Objetivos

- Desarrollar una aplicación mínima en PHP, compuestas por varias páginas.
- Aplicar los conocimientos adquiridos sobre PHP, incluyendo procesamiento de formularios, validación, sesiones, cookies, etc.

2.- Visión general de la actividad

En esta actividad se creará una aplicación PHP que implementará una pequeña aplicación para la gestión de la agenda de los usuarios, utilizando como almacenamiento una base de datos SQLite.

Algunas de las características de la aplicación:

- Permitirá a los usuarios conectarse (login) y desconectarse (logout) de la aplicación. También permitirá el registro de nuevos usuarios en la aplicación.
- Controlará el acceso a la aplicación, evitando que usuarios no identificados puedan acceder a ella.
- Implementará ciertas medidas de seguridad. Por ejemplo, hash de las contraseñas.
- Realizará las funciones básicas de la agenda: consulta, alta, baja y modificación de eventos en la agenda,

3.- Requisitos previos

Los requisitos previos para poder llevar a cabo la actividad son:

- Tener PHP 8.x instalado y funcionando correctamente.
- Tener instalada y activada la extensión de PHP para SQLite. Si no se tiene instalada, se puede instalar:
  - En Linux, con el gestor de paquetes adecuado
  - En Windows, la extensión está instalada y activada por defecto. Si no lo estuviera, consultar la documentación en https://www.php.net/manual/en/book.sqlite3.php

4.- Equipos de trabajo

Esta actividad se realizará preferentemente en equipos de dos personas, teniendo en cuenta las siguientes consideraciones:

- Los equipos serán, como norma general, de dos personas.
- Excepcionalmente, se permitirá realizar la práctica en solitario. Si una sola persona quedara “descolgada”, sin equipo, y quisiera unirse a un equipo de dos personas, se permitiría, pero se asignaría tareas adicionales, además de las ya indicadas en este enunciado.
- El profesor no organizará los grupos, pero se reserva el derecho de ajustar la composición de algún grupo si lo considera necesario, siempre para favorecer el equilibrio dentro de los grupos en función del nivel de los integrantes.
- El número de tareas a realizar en la actividad dependerá del tamaño del equipo de trabajo. Los equipos de dos personas tendrán que realizar más trabajo que los que realicen la actividad en solitario. El objetivo es intentar que todos los alumnos realicen una cantidad de trabajo similar.
- Se recomienda que los equipos de dos personas creen un repositorio privado en GitHub para mantener el progreso de la actividad, y para poder trabajar en equipo, aunque la actividad sea más o menos pequeña. Uno de los miembros deberá ser el propietario del repositorio y dar permisos de colaborador a su compañero.

5.- Evaluación y entrega

La evaluación se realizará por observación en el aula. Se dejará tiempo para desarrollar esta actividad durante unas cuantas sesiones, y el profesor observará el trabajo de los equipos, realizando preguntas y valorando el avance. Una vez que el equipo considere que la actividad está completada, informará al profesor, que pedirá a los alumnos que realicen una serie de pruebas, y realizará algunas preguntas sobre el código, forma de trabajo, dificultades encontradas o cualquier otro asunto relacionado con la actividad. Se valorarán cosas como, entre otras:

- Ajuste a los requisitos de la actividad. ¿Hace lo que se pedía? ¿Tiene errores?
- Estructura y claridad, uso de convenciones y estándares, elección de nombres variables, funciones o métodos, constantes, etc.
- Uso de las mejores técnicas posibles para validación o para el acceso y recorrido de arrays asociativos.
- Etc.

Para poder registrar la calificación, los alumnos entregarán el código fuente de la actividad en formato ZIP. Todos los miembros del equipo deben realizar la entrega del código.

La calificación será la misma para todos los integrantes del grupo, salvo casos evidentes en los que alguno de los integrantes no haya colaborado lo suficiente, en cuyo caso el profesor podrá calificar con menos nota a ese miembro del equipo.

6.- Creación de equipos y preparación del entorno de desarrollo

6.1.- Creación de grupos

En clase, se formarán los grupos, y el profesor anotará los componentes de cada grupo, y les asignará un número, del 01 en adelante. Hasta que no se hayan formado los grupos no se podrá avanzar con la actividad.

Una vez creados los equipos, en el aula virtual se publicará un listado de estos y sus componentes.

6.2.- Preparación de entorno

Para facilitar el trabajo de los alumnos, se han creado una serie de ficheros con código PHP listo para funcionar, y que los alumnos deberán utilizar.

Los pasos para inicializar el entorno de desarrollo son:

- Descargar del aula virtual el fichero “base-code.zip”
- Descomprimir el fichero en una carpeta, y renombrar la carpeta a equipo-xx, siendo xx el número de equipo.
- Revisar el contenido de la carpeta, que debería ser:
    - Directorio “data-access”
        - Fichero “CalendarDataAccess.php”
    - Directorio “entities”
        - Fichero “Event.php”
        - Fichero “User.php”
    - Directorio “Utils”
        - Fichero “SecUtils.php”
    - Fichero “tests.php”

Una vez revisado el contenido, se podría comenzar el desarrollo.

6.3.- Descripción de los ficheros descargados

User.php – Clase para representar los usuarios. Es una clase sólo para manejo de datos, con constructor, getters y setters, pero sin funcionalidad.

Event.php – Clase para representar los eventos de la agenda. Es una clase sólo para manejo de datos, con constructor, getters y setters, pero sin funcionalidad.

CalendarDataAccess.php – Clase que permite la conexión a la base de datos y que ofrece los siguientes métodos:

- __construct($dbFile) – Constructor que inicializa la conexión a la base de datos SQLite. Crea las tablas necesarias y genera datos iniciales si las tablas están vacías. Recibe la ubicación del fichero de base de datos.
- getUserById(int $userId): ?User – Recupera un usuario de la base de datos utilizando su ID. Devuelve un objeto User si se encuentra el usuario, o null si no.
- getUserByEmail(string $email): ?User – Obtiene un usuario a partir de su correo electrónico. Devuelve un objeto User si se encuentra el usuario, o null si no.
- getAllUsers(): array – Recupera todos los usuarios de la base de datos y devuelve un array de objetos User.
- createUser(User $user): bool –Crea un nuevo usuario en la base de datos. Devuelve true si la operación tiene éxito, o false si ocurre un error.
- updateUser(User $user): bool –Actualiza la información de un usuario existente en la base de datos. Devuelve true si la operación tiene éxito, o false si ocurre un error.
- deleteUserById(int $userId): bool – Elimina un usuario de la base de datos utilizando su ID. Devuelve true si la operación tiene éxito, o false si ocurre un error.
- getEventById(int $eventId): ?Event – Recupera un evento de la base de datos utilizando su ID. Devuelve un objeto Event si se encuentra el evento, o null si no.
- getEventsByUserId(int $userId): array – Obtiene todos los eventos asociados a un usuario específico. Devuelve un array de objetos Event.
- createEvent(Event $event): bool – Crea un nuevo evento en la base de datos. Devuelve true si la operación tiene éxito, o false si ocurre un error.
- updateEvent(Event $event): bool – Actualiza un evento existente en la base de datos. Devuelve true si la operación tiene éxito, o false si ocurre un error.
- deleteEvent(int $eventId): bool – Elimina un evento de la base de datos utilizando su ID. Devuelve true si la operación tiene éxito, o false si ocurre un error.

SecUtils.php – Clase con métodos estáticos para encriptar y desencriptar datos. El atributo estático "$ENCRIPTION_KEY_SECRET" se usa para generar una clave privada para la encriptación. Si se cambia este valor entre ejecuciones, los datos previamente encriptados no podrán desencriptarse. Se recomienda fijar un valor en ese atributo (si se quiere cambiar) y no modificarlo más adelante. Estos métodos podrían usarse para guardar cookies en cliente sin exponer su contenido.

Tests.php – Página con pruebas de los métodos de la capa de acceso a datos. Ojo: esta página de pruebas elimina la base de datos que se hubiera creado. Está pensada para verificar que todo funciona, y como guía de uso de algunos métodos, pero si se ejecuta hay que ser conscientes de que eliminará la base de datos, y que la capa de acceso a datos la volverá a crear con unos datos iniciales.

7.- Aplicación a desarrollar

Se debe desarrollar una aplicación con las siguientes páginas:

- index.php – Será la página de inicio de la aplicación, y contendrá el formulario de login.
- register.php – Página para registrarse en la aplicación.
- events.php – Página con el listado de eventos del usuario conectado.
- new-event.php – Página para crear un nuevo evento para el usuario conectado.
- edit-event.php – Página para modificar un evento existente, del usuario conectado.
- delete-event.php – Página para eliminar un evento del usuario conectado.
- change-password.php – Página para cambiar la contraseña del usuario conectado.
- logout.php – Página para desconectar de la aplicación.

Los equipos de una persona tienen que hacer sólo las páginas index, register, events, new-event, delete-event y logout.

Los de dos personas tienen que hacer todas las páginas. Se añaden edit-event, y change-password.

7.1.- Consideraciones generales

Como se puede observar, es una aplicación en la que los usuarios deben iniciar sesión antes de poder trabajar con sus eventos de calendario. Algunas observaciones sobre la persistencia de sesión, la seguridad, y el comportamiento de las páginas

- Es obligatorio usar las sesiones de PHP para mantener la sesión del usuario.
- Cuando se produzcan eventos críticos en la sesión del usuario, debe regenerarse el id de sesión. Se consideran eventos críticos:
  - Login y logout
  - Cambio de contraseña
- Es obligatorio haber iniciado sesión para acceder a TODAS las páginas salvo:
  - Página index.php
  - Página register.php

  Si se accede a cualquiera de estas dos páginas con sesión iniciada, se redirigirá automáticamente a "events.php"
- Si se accede a cualquier página (salvo index y register) sin sesión iniciada, se redirigirá automáticamente a index.php.
- Las contraseñas se guardarán como un hash de la contraseña, no se guardarán "en abierto". Ver https://www.php.net/manual/en/function.password-hash.php para más información de cómo calcular el hash de una contraseña en PHP. Guardar un hash es la forma recomendada para la validación de contraseñas.
- Todas las páginas en las que haya sesión iniciada deben tener en algún punto (se recomienda la cabecera):
  - El nombre y apellidos del usuario conectado
  - Un enlace para desconectar
  - Un enlace para cambiar la contraseña
  - Un enlace para editar el perfil del usuario
- Todos los formularios deben validar los campos en el cliente y en el servidor, y el envío debe hacerse por POST.
- Si al enviar un formulario se producen errores, al volverse a cargar, los datos deberán permanecer tal y como los había escrito el usuario, salvo las contraseñas, que deben estar vacías.
- Se anima a separar el código en múltiples ficheros PHP, sobre todo si hay posibilidad de reutilización.

- Crear estructuras de carpetas si se considera adecuado. Al realizar include o require usar __DIR__ para que no falle la búsqueda de scripts a incluir cuando se despliegue en otros entornos.
- Es obligatorio el uso de Bootstrap 5 para dar un aspecto más profesional a la página. Se puede asumir que la aplicación se usará sólo desde un pc de sobremesa / portátil. No es necesario tener en cuenta el formato para otros dispositivos (móviles, tablets, etc.).

7.2.- index.php – Formulario de login

Esta página será el punto de entrada a la aplicación. Requisitos:

- A esta página sólo se puede acceder sin sesión iniciada. Si se accede con sesión iniciada se redirigirá a la página "events.php"
- Tendrá un formulario con dos campos:
  - Correo electrónico – Obligatorio – Valida formato de correo electrónico.
  - Contraseña – Obligatorio
- El formulario se enviará a la misma página, y:
  - Si falla la validación (incluyendo usuario y/o contraseña incorrectos) mostrará un mensaje de error por encima del formulario, y volverá a mostrar el formulario.
  - Si el usuario existe y la contraseña es correcta, se redirigirá events.php.
- Debe contener un enlace para acceder a "register.php" con un texto "Regístrate" o similar.

Como las contraseñas en la BD están guardadas como un hash, se deberá realizar una verificación de la contraseña, comparándola con el hash almacenado.

Ver https://www.php.net/manual/en/function.password-verify.php para más información de la verificación de contraseñas.

7.3.- register.php – Formulario de registro

Esta página permitirá el registro en la aplicación. Requisitos:

- A esta página sólo se puede acceder sin sesión iniciada. Si se accede con sesión iniciada se redirigirá a la página "events.php"
- Tendrá un formulario con los campos:
  - Correo electrónico – Obligatorio – Valida formato de correo electrónico.
  - Nombre – Obligatorio
  - Apellidos – Obligatorio
  - Fecha de nacimiento – Obligatorio
  - Contraseña – Obligatorio – Mínimo 8 caracteres. Debe contener al menos un número y una letra (no puede ser solo números ni solo letras).
  - Repetir contraseña – Obligatorio – Mínimo 8 caracteres y debe ser igual que la otra contraseña introducida.
- El formulario se enviará a la misma página, y:
  - Si falla la validación (faltan campos, contraseñas no coinciden, longitud insuficiente, no tiene un número o letra, etc.) mostrará un mensaje de error por encima del formulario, y volverá a mostrar el formulario.
  - Si existe ya un usuario con el correo electrónico introducido, mostrará un mensaje de error por encima del formulario, y volverá a mostrar el formulario.
  - Si todos los campos son correctos y no se está usando ya el correo electrónico, creará un usuario con los datos recibidos. La contraseña debe guardarse como un hash en la BD.
  - Una vez creado el usuario, mostrará un mensaje "Usuario creado" y un enlace para volver a la página de login, donde se podrá probar el nuevo usuario.

7.4.- events.php – Listado de eventos del usuario

Esta será la página "principal" de la aplicación. Requisitos:

- Mostrará, en una tabla, TODOS los eventos *del usuario conectado*, uno por cada fila.
- La última celda de cada fila tendrá dos enlaces:
  - Editar – Enlazará con la página edit-event.php, para editar el evento. El enlace pasará el id del evento a la página, usando un parámetro en la query string.
  - Eliminar – Enlazará con la página delete-event.phph, para eliminar el evento. El enlace pasará el id del evento a la página, usando un parámetro en la query string.
- Antes y después de la tabla de eventos, aparecerá un enlace "nuevo evento", que enlazará con la página "new-event.php"
- Utilizar font-awesome o similar para que estos enlaces (nuevo, editar, eliminar) sean iconos, y no texto. Además, debes añadir texto "oculto" para que las personas ciegas puedan entender qué son estos iconos.

Tutorial para usar iconos FA: https://medium.com/@diego.coder/agrega-iconos-de-font-awesome-en-tu-sitio-web-49660efc50ee

Cómo usar clases CSS de Bootstrap para lectores de pantalla, y algunas otras consideraciones de accesibilidad: https://getbootstrap.com/docs/5.0/getting-started/accessibility/

7.5.- new-event.php – Creación de nuevo evento

Permitirá crear un nuevo evento asociado al usuario conectado. Requisitos:

- Contendrá un formulario con todos los campos asociados al evento:
  - Título – Obligatorio – Texto
  - Descripción – Opcional – Texto largo (textarea)
  - Fecha y hora de inicio – Obligatorio – Usar un control o controles que permitan seleccionar la fecha y hora, no un control de tipo texto.
  - Fecha y hora de fin – Obligatorio – Usar un control o controles que permitan seleccionar la fecha y hora, no un control de tipo texto. Esta fecha+hora debe ser posterior a la fecha y hora de inicio.
- El formulario se enviará a la misma página y:
  - Si falla la validación mostrará un mensaje de error por encima del formulario, y volverá a mostrar el formulario.
  - Si todos los datos son correctos, se creará un nuevo evento en la base de datos, asociado al usuario conectado, y se redirigirá al listado de eventos.

7.6.- edit-event.php – Modificación de evento

Permitirá modificar un evento asociado al usuario conectado. Requisitos:

- La página recibirá el id del evento a modificar a través de un parámetro en la query string. Por ejemplo: *edit-event.php?id=3*.
- Por seguridad, se debe comprobar:
  - Que existe un evento con el id recibido.
  - Si existe, que está asociado al usuario conectado y no a otro.

  Si se produce cualquiera de estos errores, se mostrará un mensaje "No se puede acceder al evento porque no existe o porque no tiene permisos para verlo", y un enlace "Volver al listado de eventos", que permitirá al usuario volver a este listado.

- Si el evento existe y pertenece al usuario conectado, se mostrará un formulario con todos los campos asociados al evento:
  - Título – Obligatorio – Texto
  - Descripción – Opcional – Texto largo (textarea)
  - Fecha y hora de inicio – Obligatorio – Usar un control o controles que permitan seleccionar la fecha y hora, no un control de tipo texto.
  - Fecha y hora de fin – Obligatorio – Usar un control o controles que permitan seleccionar la fecha y hora, no un control de tipo texto. Esta fecha+hora debe ser posterior a la fecha y hora de inicio.

  Todos los campos estarán rellenos con los datos del evento.
- El formulario se enviará a la misma página y:
  - Si falla la validación mostrará un mensaje de error por encima del formulario, y volverá a mostrar el formulario.
  - Si todos los datos son correctos, se modificará el evento en la base de datos, permaneciendo asociado al usuario conectado, y se redirigirá al listado de eventos.

7.7.- delete-event.php – Eliminar evento

Permitirá eliminar un evento asociado al usuario conectado. Requisitos:

- La página recibirá el id del evento a eliminar a través de un parámetro en la query string. Por ejemplo: *delete-event.php?id=3*.
- Por seguridad, se debe comprobar:
  - Que existe un evento con el id recibido.
  - Si existe, que está asociado al usuario conectado y no a otro.

  Si se produce cualquiera de estos errores, se mostrará un mensaje "No se puede acceder al evento porque no existe o porque no tiene permisos para verlo", y un enlace "Volver al listado de eventos", que permitirá al usuario volver a este listado.
- Si el evento existe y pertenece al usuario conectado, se mostrará un formulario con la pregunta "¿Seguro que desea eliminar el evento <título>?" Y dos botones:
  - Sí, eliminar el evento
  - No, volver al listado de eventos.
- El formulario se enviará a la misma página y, en función del botón pulsado:
  - Eliminará el evento y redirigirá al listado de eventos
  - Redirigirá al listado de eventos, sin eliminarlo.

Recordatorio: para saber qué botón se pulsó se puede añadir name y value a los botones, y se pasarán al script PHP, que podrá acceder a ellos a través de $_POST.

7.8.- change-password.php – Cambiar contraseña

Permitirá al usuario cambiar su contraseña. Requisitos:

- Se mostrará un formulario con los campos:
  - Contraseña actual – Obligatorio
  - Nueva contraseña – Obligatorio – Mínimo 8 caracteres. Debe contener al menos un número y una letra (no puede ser solo números ni solo letras).
  - Repetir nueva contraseña – Obligatorio – Mínimo 8 caracteres y debe ser igual que la otra nueva contraseña introducida.
- El formulario se enviará a la misma página, y:

- Si falla la validación (contraseña original incorrecta, contraseñas no coinciden, longitud insuficiente, etc.) mostrará un mensaje de error por encima del formulario, y volverá a mostrar el formulario.
- Si todos los campos son correctos, modificará la contraseña del usuario (recordar que hay que guardarla como un hash en la BD.
- Mostrará un mensaje "Contraseña modificada", y mostrará un enlace para volver al listado de eventos.

7.9.- logout.php – Desconectar

Permitirá al usuario desconectar de la aplicación, cerrar su sesión. Requisitos:

- Se mostrará un formulario con la pregunta "¿Seguro que desea desconectar?" Y dos botones:
  - Sí, desconectar
  - No, volver al listado de eventos.
- El formulario se enviará a la misma página y, en función del botón pulsado:
  - Cerrará la sesión y redirigirá al login
  - Redirigirá al listado de eventos, sin cerrar la sesión.

De nuevo, recordatorio: para saber qué botón se pulsó se puede añadir name y value a los botones, y se pasarán al script PHP, que podrá acceder a ellos a través de $_POST.