CIFP PLAIAUNDI LHII	1ª EVALUACIÓN	GRUPO	2DAW3
NOTA:		FECHA	20/11/2024
NOMBRE:			

Instrucciones 2
Requisitos mínimos 2
1. Base de Datos: Los SQL vienen dados. 2
2. Funcionalidades 2
3. Estructura de Archivos 3
Consideraciones 3
Datos para la aplicación 3
Detalles de los archivos 5
1. Archivo de Configuración de la Base de Datos: config/db.php 5
2. Controlador de Autenticación: controllers/auth.php 5
3. Controlador de Reservas: controllers/reservation.php 5
4. Modelo de Reserva: models/Reservation.php 5
5. Modelo de Recursos: models/Resource.php 6
6. Modelo de Horarios: models/TimeSlot.php 6
7. Modelo de Usuario: models/User.php 6
8. Formulario de Inicio de Sesión: views/login.php 6
9. Página de Cierre de Sesión: views/logout.php 7
10. Vista de Reservas: views/reservations.php 7
11. Página de Recursos: views/resources.php 7
Resumen de los archivos a modificar 7
Evaluación (50 puntos) 8
1. Conexión a la Base de Datos y Estructura de Carpetas (10 puntos) 8
2. Página de Login y Control de Roles (15 puntos) 8
3. Visualización y Reserva de Recursos (10 puntos) 8
5. Control de Acceso y Sesión (10 puntos) 8
4. Gestión de Reservas (Añadir, Modificar, Eliminar) (5 puntos) 9

Aplicación de Reservas de Recursos

Desarrollar una aplicación en PHP para gestionar reservas de recursos con roles de acceso diferenciados.

Instrucciones

La aplicación debe permitir a los usuarios iniciar sesión y gestionar reservas de distintos recursos en función de su rol:

1. **Usuarios normales**:
    - Pueden ver y hacer reservas.
    - Solo pueden ver sus propias reservas.
2. **Usuarios administradores**:
    - Pueden ver y hacer reservas.
    - Pueden ver todas las reservas existentes.

Requisitos mínimos

1. Base de Datos: Los SQL vienen dados.

2. Funcionalidades

A. Página de Login

- Página `login.php` que permita a los usuarios iniciar sesión.
- Diferencia la vista según el rol: si es "admin" redirígelos a una vista con permisos completos; si es "user1" o “user2”, solo a sus opciones.

B. Gestión de Reservas

- **Formulario de reserva**:
    - Muestra una lista de recursos y tramos horarios disponibles.
    - Permite seleccionar un recurso y un horario para reservarlo.
- **Visualización de reservas**:
    - **Usuarios**: pueden ver solo sus reservas.
    - **Administradores**: pueden ver todas las reservas.

C. Control de Acceso

- Implementa una lógica de autenticación para verificar que solo usuarios autenticados accedan a las páginas protegidas.

- Usa sesiones para gestionar el login de cada usuario.

3. Estructura de Archivos

Organiza tu proyecto en carpetas según el siguiente esquema:

- **/config/**
  - `db.php`: para la conexión a la base de datos.
- **/models/**
  - `User.php`: clase User con lógica de autenticación y manejo de roles.
  - `Resource.php`: clase Resource para obtener la lista de recursos.
  - `TimeSlot.php`: clase TimeSlot para manejar los horarios.
  - `Reservation.php`: clase Reservation para las operaciones de reserva.
- **/controllers/**
  - `auth.php`: controlador para el inicio y cierre de sesión.
  - `reservation.php`: controlador para manejo de reservas.
- **/views/**
  - `login.php`: formulario de inicio de sesión.
  - `logout.php`: cierre de sesión.
  - `resources.php`: página para ver y hacer reservas.
  - `reservations.php`: página de gestión de reservas.

Consideraciones

- El login debe hacerse con autenticación en la base de datos.
- Se valorará la claridad del código, el uso de comentarios y la estructura del proyecto.
- Asegúrate de probar la aplicación antes de entregarla para verificar que todas las funcionalidades básicas están operativas.

Datos para la aplicación

SQL creación de la base de datos:

```
-- Creación de la base de datos
CREATE DATABASE reservas_db;
USE reservas_db;

-- Tabla de Usuarios
CREATE TABLE Users (
    id INT AUTO_INCREMENT PRIMARY KEY,
    username VARCHAR(50) NOT NULL UNIQUE,
    password VARCHAR(255) NOT NULL,
    role ENUM('user', 'admin') NOT NULL DEFAULT 'user'
```

```
);

-- Tabla de Recursos
CREATE TABLE Resources (
    id INT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(100) NOT NULL,
    type VARCHAR(50) NOT NULL,
    location VARCHAR(100) NOT NULL
);

-- Tabla de Horarios
CREATE TABLE TimeSlots (
    id INT AUTO_INCREMENT PRIMARY KEY,
    start_time TIME NOT NULL,
    end_time TIME NOT NULL,
    UNIQUE KEY (start_time, end_time)
);

-- Tabla de Reservas
CREATE TABLE Reservations (
    id INT AUTO_INCREMENT PRIMARY KEY,
    user_id INT NOT NULL,
    resource_id INT NOT NULL,
    timeslot_id INT NOT NULL,
    date DATE NOT NULL,
    FOREIGN KEY (user_id) REFERENCES Users(id) ON DELETE CASCADE,
    FOREIGN KEY (resource_id) REFERENCES Resources(id) ON DELETE CASCADE,
    FOREIGN KEY (timeslot_id) REFERENCES TimeSlots(id) ON DELETE CASCADE,
    UNIQUE KEY (resource_id, timeslot_id, date)
);
```

Inserción de datos

seed_data.php ejecutarlo una vez y comprobar que se han introducido los datos correctamente.

Las contraseñas en la tabla Users :

- Usuario admin con contraseña admin123.
- Usuario user1 con contraseña user123.
- Usuario user2 con contraseña user456.

Detalles de los archivos

1. Archivo de Configuración de la Base de Datos: `config/db.php`

Este archivo configura la conexión a la base de datos utilizando PDO.

---

2. Controlador de Autenticación: `controllers/auth.php`

Este archivo gestiona el inicio y cierre de sesión de los usuarios.

- **Función `login`**: Inicia sesión verificando el nombre de usuario y contraseña y guardando los datos en la sesión. Si las credenciales son correctas, redirige a la página de recursos.
- **Función `logout`**: Cierra la sesión y redirige al formulario de inicio de sesión.

3. Controlador de Reservas: `controllers/reservation.php`

Este archivo se encarga de procesar las solicitudes de reserva.

- **Propósito**: Recibir datos del formulario, validar la información, y luego guardar la reserva en la base de datos.
- **Función Principal**: Inserta una nueva reserva en la base de datos. Redirige de vuelta a `resources.php` con un mensaje de éxito o error dependiendo del resultado de la operación.

---

4. Modelo de Reserva: `models/Reservation.php`

Este archivo define la clase `Reservation` para gestionar las reservas.

- **Propósito**: Crear reservas y obtener la lista de reservas, ya sea de un usuario específico o de todos los usuarios.
- **Método `addReservation`**: Inserta una nueva reserva en la base de datos.
- **Método `getUserReservations` y `getAllReservations`**: Obtiene las reservas del usuario actual o todas las reservas (para los administradores).

5. Modelo de Recursos: `models/Resource.php`

Este archivo define la clase `Resource` y permite obtener la lista de recursos disponibles para ser reservados.

- **Propósito**: Proporcionar una forma de acceder a los datos de los recursos almacenados en la base de datos.
- **Método `getAll`**: Recupera todos los recursos disponibles en la tabla `Resources`

6. Modelo de Horarios: `models/TimeSlot.php`

Define la clase `TimeSlot`, que gestiona los intervalos de tiempo en los que se pueden realizar reservas.

- **Propósito**: Proporcionar acceso a los intervalos de tiempo disponibles.
- **Método `getAll`**: Recupera todos los horarios desde la tabla `TimeSlots`.

7. Modelo de Usuario: `models/User.php`

Este archivo define la clase `User`, que representa a los usuarios de la aplicación.

- **Propósito**: Manejar la autenticación del usuario y determinar si un usuario es administrador o usuario normal.
- **Método `login`**: Comprueba si las credenciales de inicio de sesión son correctas.
- **Método `isAdmin`**: Devuelve `true` si el usuario es administrador, lo que permite personalizar el acceso en las vistas.

---

8. Formulario de Inicio de Sesión: `views/login.php`

Este archivo HTML proporciona el formulario de inicio de sesión.

- **Propósito**: Permitir a los usuarios ingresar sus credenciales (nombre de usuario y contraseña) para autenticarse.
- **Acción del Formulario**: El formulario envía los datos al controlador de autenticación (`auth.php`) para validar las credenciales.

9. Página de Cierre de Sesión: `views/logout.php`

Este archivo cierra la sesión del usuario y lo redirige a la página de inicio de sesión.

- **Propósito**: Terminar la sesión de usuario y eliminar sus datos de sesión.
- **Función**: Limpia las variables de sesión y destruye la sesión antes de redirigir al formulario de inicio de sesión.

10. Vista de Reservas: `views/reservations.php`

Muestra la lista de reservas del usuario actual o de todos los usuarios (si es administrador).

- **Propósito**: Mostrar las reservas en una tabla con información detallada (nombres de usuario, recurso, horario y fecha).
- **Uso de `getUserReservations` y `getAllReservations`**: Dependiendo del rol, muestra solo las reservas propias o todas las reservas.

11. Página de Recursos: `views/resources.php`

Esta página muestra los recursos y permite realizar una reserva seleccionando un recurso, un horario y una fecha.

- **Propósito**: Proporcionar una interfaz para que los usuarios realicen reservas.
- **Formulario de Reserva**: Al seleccionar un recurso, horario y fecha, el formulario envía los datos para procesar la reserva.

Resumen de los archivos a modificar

- db.php
- auth.php
- resources.php
- Resource.php
- Reservation.php
- reservations.php:

- Añadir en todas las páginas necesarias un control de usuario logueado si no vuelve a login

---

Evaluación (50 puntos)

1. Conexión a la Base de Datos y Estructura de Carpetas (10 puntos)

- **Conexión a la Base de Datos**:
  - **5 puntos**: se conecta correctamente a la base de datos utilizando PDO.
- **Estructura de Carpetas**:
  - **5 puntos**: Cada archivo se encuentra en la carpeta correcta.

2. Página de Login y Control de Roles (15 puntos)

- **Página de Login**:
  - **5 puntos**: La autenticación utiliza `password_verify` para validar la contraseña frente al hash almacenado en la base de datos.
- **Control de Roles (10 puntos)**:
  - **5 puntos**: El sistema distingue entre roles de usuario (`user`) y administrador (`admin`).
  - **5 puntos**: Las vistas y permisos están correctamente controlados, por ejemplo, los usuarios no pueden acceder a las vistas o funciones de administrador.

3. Visualización y Reserva de Recursos (10 puntos)

- **Visualización de Recursos**:
  - **5 puntos**: La lista de recursos y horarios se obtiene dinámicamente de la base de datos.
- **Formulario de Reserva**:
  - **5 puntos**: El formulario envía los datos al php correspondiente, y el usuario recibe mensajes de éxito o error al realizar la reserva.

5. Control de Acceso y Sesión (10 puntos)

- **Control de Acceso (10 puntos)**:
  - **5 puntos**: Los usuarios no autenticados son redirigidos al login si intentan acceder a áreas protegidas.

- **5 puntos**: La sesión incluye correctamente los datos del usuario, como el ID y el rol, y se comprueba en cada página protegida.

4. Gestión de Reservas (5 puntos)

- **Añadir Reservas (5 puntos)**:
  - **5 puntos**: La reserva se inserta correctamente en la base de datos.

BONUS:

- **Modificar y Eliminar Reservas (10 puntos)**:
  - **5 puntos**: Los usuarios pueden modificar o eliminar solo sus reservas.
  - **5 puntos**: Los administradores tienen acceso a gestionar todas las reservas