
	0373 - Llenguatges de Marques				
	APIs				
Act:	PHP				
Apellidos:	Vizcaíno Boixadós	Nombre:	Jan	Fecha:	23/05/2025

1. Versión A – JSON

1.1. Explicación del funcionamiento de cada archivo PHP.

Login.php

```
<?php
header("Expires: Sat, 1 Jul 2000 05:00:00 GMT");
header("Cache-Control: no-cache, must-revalidate");
header("Pragma: no-cache"); //Para borrar la cache del navegador (debug)
session_start(); //Iniciamos la sesión
if ($_SERVER["REQUEST_METHOD"] === "POST") { //Comprueba si el formulario se envia
con el metodo POST
    $email = $_POST["email"];
    $password = $_POST["password"];
    $archivo = "users.json"; //Asignamos los distintos campos que usaremos para
gestionar la sesion y el usuario.

    if (file_exists($archivo)) { //Si el archivo existe
        $usuarios = json_decode(file_get_contents($archivo), true); //Transformamos
el archivo en un array de usuarios
        foreach ($usuarios as $usuario) { //Para cada usuario
            if ($usuario["email"] === $email && $usuario["password"] === $password)
{ //Comprueba si los datos estan bien
                $_SESSION["usuario"] = $email;
                $_SESSION["rol"] = "admin";
                $_SESSION["fecha"] = date("Y-m-d H:i:s");
                $_SESSION["ip"] = $_SERVER['REMOTE_ADDR']; //asigna la información
de la sesion

                header("Location: home.php"); //Cambia la página
                exit; //Sale
            }
        }
    }
}
?>
<!--Resto del HTML-->
```

Register.php

```
<?php
if ($_SERVER["REQUEST_METHOD"] === "POST") { //Si se ejecuta el formulario con
el metodo POST
    $email = $_POST["email"];
    $password = $_POST["password"]; //Obtenemos los datos del formulario

    $archivo = "users.json"; //Nombre del archivo JSON donde se guardan los datos de
los usuarios

    if (file_exists($archivo)) { //Si el archivo existe
        $usuarios = json_decode(file_get_contents($archivo), true); //Cargamos los
datos del archivo JSON en un array
    } else {
        $usuarios = []; //Si el archivo no existe, creamos un array vacío
    }

    foreach ($usuarios as $usuario) { //Recorremos el array de usuarios
        if ($usuario["email"] === $email) { //Si ya existe un usuario con el mismo
email
            echo "<p>Este correo ya está registrado.</p>";
            exit; //Salimos
        }
    }

    $nuevoUsuario = ["email" => $email, "password" => $password]; //Creamos un nuevo
usuario con los datos del formulario
    $usuarios[] = $nuevoUsuario; //Lo agregamos al array de usuarios
    file_put_contents($archivo, json_encode($usuarios, JSON_PRETTY_PRINT));
//Guardamos los datos en el archivo JSON
    header("Location: login.php"); //Volvemos a la página de login
}
?>
<!--Resto del HTML-->
```

home.php

```
<?php
session_start(); //Empieza la sesion
if (!isset($_SESSION["usuario"])) { //Si no hay usuario
    header("Location: login.php"); //Redirige a la pagina de login
    exit; //Sale del script
}
?>
<!--Resto del HTML-->
```

logout.php

```
<?php
session_start(); //Empezamos la sesion
session_unset();
session_destroy(); //Eliminamos la sesion
header("Location: login.php"); //Redirigimos al login
exit;
```

2. Versión B – BBDD

2.1. Explicación del funcionamiento de cada archivo PHP.

conexion.php

```
<?php
$servername = "localhost";
$dbname = "apiserver";
$username = "root";
$password = ""; //Configuro los parametros par la conexion a la base de datos

$conn = mysqli_connect($servername, $username, $password, $dbname); //Conecto a la
base de datos

if (!$conn) { //Si no se conecta a la base de datos
    die("Connection failed: " . mysqli_connect_error()); //Muestra el error
}
?>
```

login.php

```
<?php
require_once 'conexion.php'; //Asigno la conexión a una variable
header("Expires: Sat, 1 Jul 2000 05:00:00 GMT");
header("Cache-Control: no-cache, must-revalidate");
header("Pragma: no-cache"); //Esto es para evitar el cache de los navegadores
session_start(); //Inicia la sesión

if ($_SERVER["REQUEST_METHOD"] === "POST") { //Si se ha enviado el formulario
    try { //Intento realizar la acción
        $email = $_POST["email"]; //Obtengo el email del formulario
        $password = $_POST["password"]; //Obtengo la contraseña del formulario

        $sql = "SELECT * FROM users WHERE email = ?"; //Consulta para obtener el
usuario
        $stmt = mysqli_prepare($conn, $sql);
        mysqli_stmt_bind_param($stmt, "s", $email); //Preparo la query
        $result = mysqli_stmt_get_result($stmt); //Ejecutamos la consulta

        if ($row = mysqli_fetch_assoc($result)) { //Si hay un usuario con ese email
            if (password_verify($password, $row['password'])) { //Verificamos la
contraseña
                $_SESSION["usuario"] = $row['email']; //Si es correcto, guardamos el
email en la sesión
                header("Location: home.php"); //Redirigimos a la página de inicio
                exit;
            } else { //Si la contraseña es incorrecta
                echo "<script>alert('Contraseña incorrecta.')</script>";
//Notificamos de que es así
            }
        } else { //Si no existe, se notifica de que es así.
            echo "<script>alert('Correo no encontrado.')</script>";
        }
    } catch (Exception $e) {
        echo "<script>alert('Error: " . $e->getMessage() . "')</script>"; //Si hay
alguna excepción, se muestra
    }
}
?>
<!--Resto del HTML-->
```

register.php

```
<?php

require_once("conexion.php"); // Llamamos a la clase de conexión a la base de datos

if ($_SERVER["REQUEST_METHOD"] === "POST") { // Verificamos si se ha enviado un formulario
    try { // Intentamos realizar la operación
        $email = $_POST["email"]; // Obtenemos el email del formulario
        $password = $_POST["password"]; // Obtenemos la contraseña del formulario

        $sql = "SELECT * FROM users WHERE email = ?"; // Preparamos la consulta SQL
        $stmt = mysqli_prepare($conn, $sql);
        mysqli_stmt_bind_param($stmt, "s", $email); //Preparamos la query
        $result = mysqli_stmt_get_result($stmt); //Lanzamos la consulta

        if ($row = mysqli_fetch_assoc($result)) { // Verificamos si hay un usuario con ese email
            if (password_verify($password, $row['password'])) { // Verificamos si la contraseña es correcta
                $_SESSION["usuario"] = $row['email']; //Guardamos el email del usuario en la sesión
                header("Location: home.php"); //Redigirimos a la página de inicio
                exit;
            } else {
                echo "<script>alert('Contraseña incorrecta.')</script>"; //Si la contraseña esta mal, notificamos
            }
        } else {
            echo "<script>alert('Correo no encontrado.')</script>"; //Si el correo no existe, notificamos
        }
    } catch (Exception $e) { //Si hay una excepción, la mostramos
        echo "<script>alert('Error: " . $e->getMessage() . "')</script>";
    }
}

?>

<!--Resto del HTML-->
```

home.php

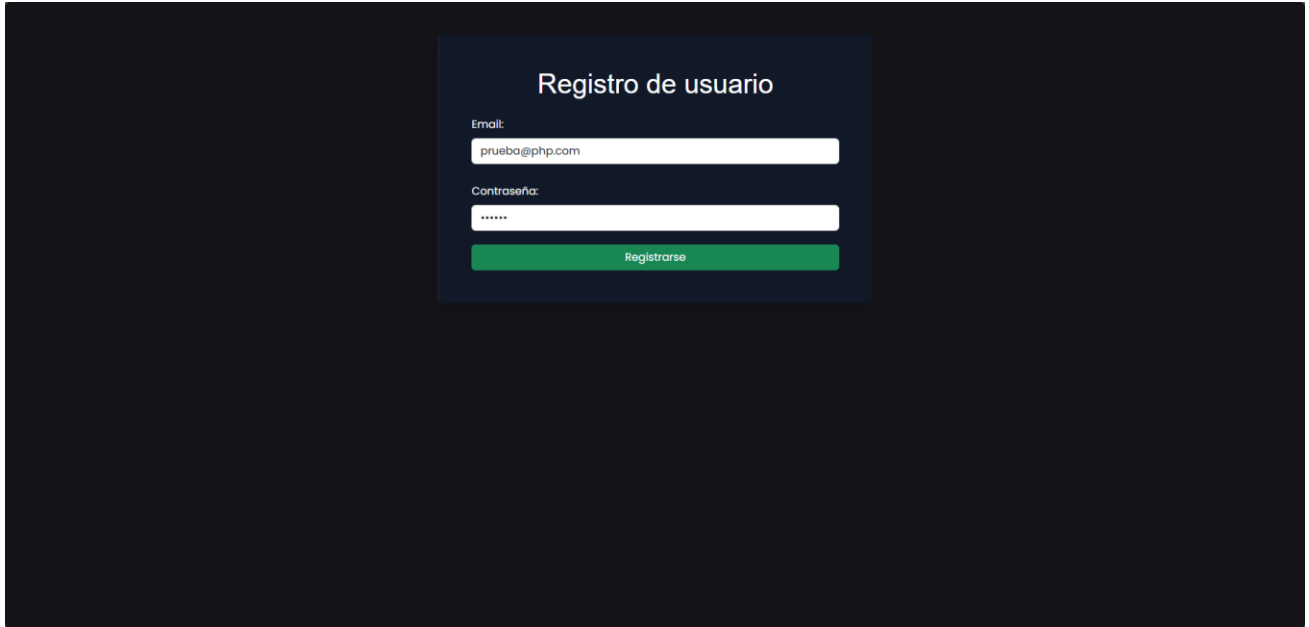
```
<?php
session_start(); //Empiezo la sesion
if (!isset($_SESSION["usuario"])) { //Si no hay usuario en la sesion
    header("Location: login.php"); //Redirijo a la pagina de login
    exit;
}
?>
<!--Resto del HTML-->
```

logout.php

```
<?php
session_start(); //Empiezo la sesión
session_unset(); //Borro todas las variables de la sesión
session_destroy(); //Destruyo la sesión
header("Location: login.php"); //Redirigimos al login
exit;
```

3. Capturas de pantalla del flujo completo (registro, login, dashboard).

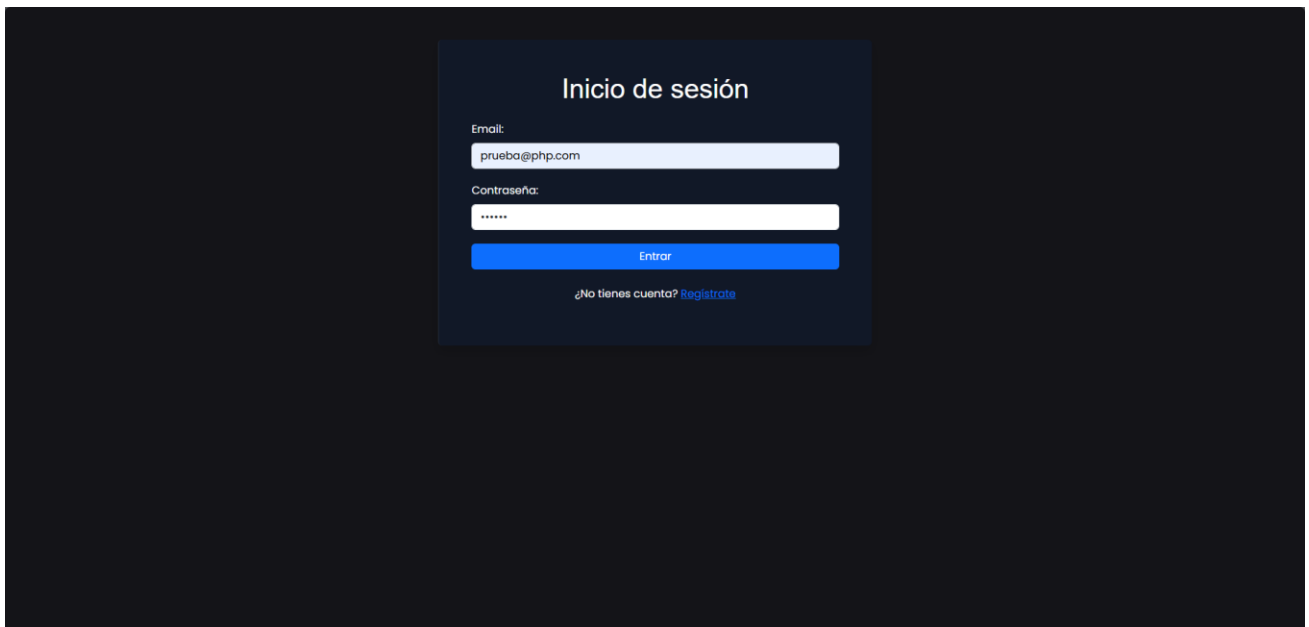
Registro



The screenshot shows a registration form titled "Registro de usuario" on a dark background. The form contains two input fields: "Email:" with the value "prueba@php.com" and "Contraseña:" with masked characters "*****". Below the fields is a green button labeled "Registrarse".

Jan Vazcano B.

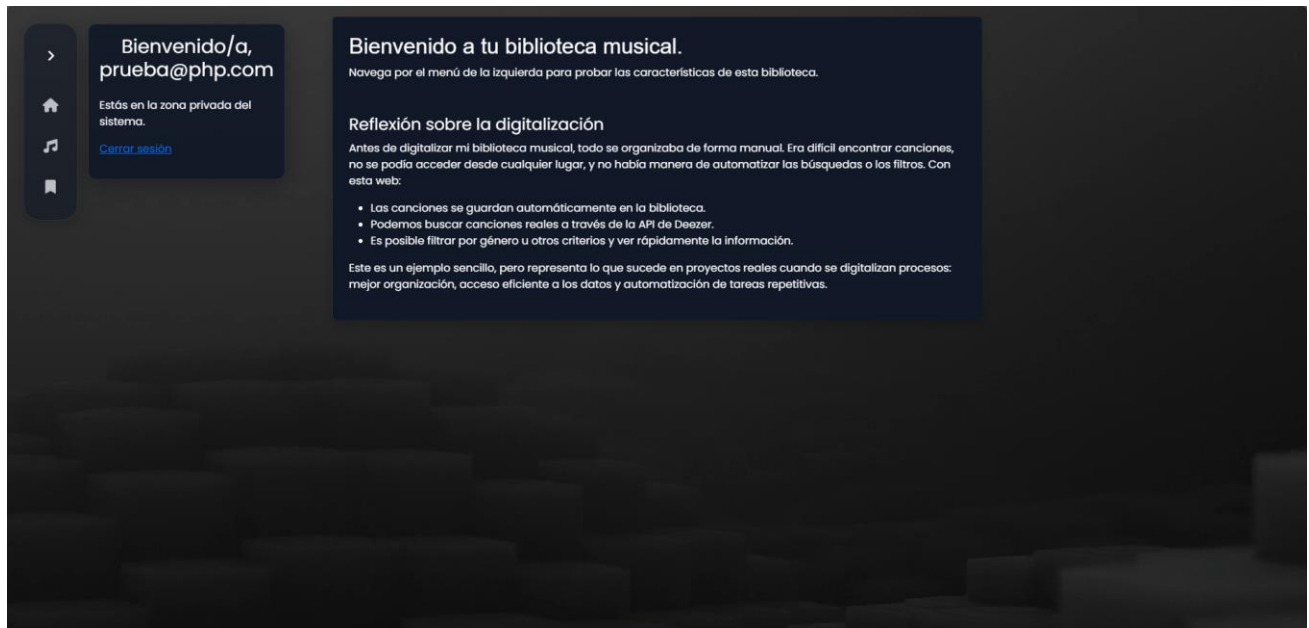
Login



The screenshot shows a login form titled "Inicio de sesión" on a dark background. The form contains two input fields: "Email:" with the value "prueba@php.com" and "Contraseña:" with masked characters "*****". Below the fields is a blue button labeled "Entrar". At the bottom of the form, there is a link that says "¿No tienes cuenta? [Regístrate](#)".

Jan Vazcano B.

Home



Bienvenido/a,
prueba@php.com

Estás en la zona privada del sistema.

[Cerrar sesión](#)

Bienvenido a tu biblioteca musical.

Navega por el menú de la izquierda para probar las características de esta biblioteca.

Reflexión sobre la digitalización

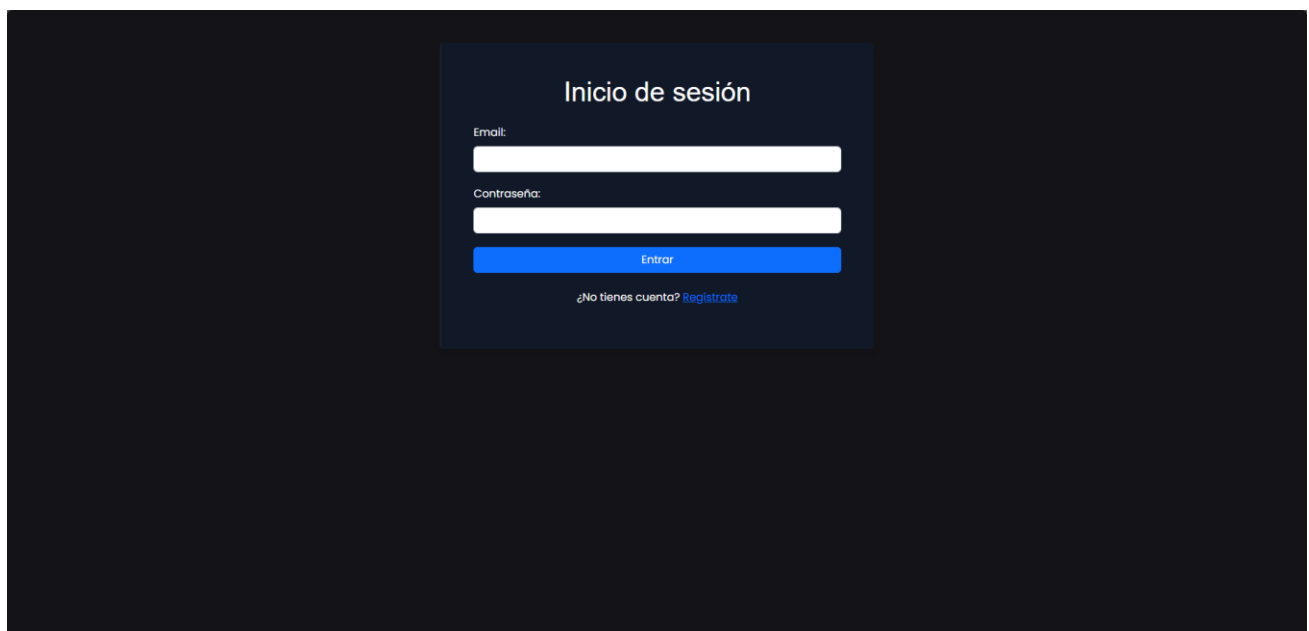
Antes de digitalizar mi biblioteca musical, todo se organizaba de forma manual. Era difícil encontrar canciones, no se podía acceder desde cualquier lugar, y no había manera de automatizar las búsquedas o los filtros. Con esta web:

- Las canciones se guardan automáticamente en la biblioteca.
- Podemos buscar canciones reales a través de la API de Deezer.
- Es posible filtrar por género u otros criterios y ver rápidamente la información.

Este es un ejemplo sencillo, pero representa lo que sucede en proyectos reales cuando se digitalizan procesos: mejor organización, acceso eficiente a los datos y automatización de tareas repetitivas.

Jan Visciano B.

Logout



Inicio de sesión

Email:

Contraseña:

[Entrar](#)

[¿No tienes cuenta? Regístrate](#)

Jan Visciano B.

4. Comentario breve sobre cómo integraste la API.

Para integrar la API no he tenido que cambiar muchas cosas. Simplemente integrar el código que tenía de la anterior práctica y adaptar algunas funciones.

Principalmente he cambiado funciones visuales de menor importancia que gestionaban el contenido de la página por el framework de JS, Alpine.

Además, he tenido que modificar la llamada de las funciones. Antes se ejecutaban cuando se cargaba el contenido (DOMContentLoaded) y ahora se cargan al cambiar de contenido con Alpine.

5. Dificultades que encontraste y cómo las resolviste.

La principal dificultad que he encontrado está relacionada con el punto anterior; integrar el JS. Me ha costado porque la llamada de las funciones y el flujo de ejecución es diferente al trabajar con PHP que al trabajar con páginas web HTML estáticas.

6. Diferencias entre la versión con JSON y con base de datos.

Característica	JSON	BBDD
Almacenamiento	Los datos se guardan en un archivo de texto plano .json	Los datos se almacenan en forma de registros en tablas en una base de datos.
Seguridad	Menor seguridad, los archivos son accesibles fácilmente.	Mayor seguridad, ya que se pueden aplicar usuarios y roles.
Rendimiento	Más lento a medida que el archivo crece.	Mejor rendimiento gracias a índices.
Consultas	Limitadas.	Rápidas si se usa SQL
Escalabilidad	No es adecuado para muchos datos.	Escalable, diseñado para manejar muchos datos.

7. Reflexión sobre la importancia de validar accesos en una aplicación real.

La validación de accesos es un elemento crítico en cualquier aplicación. Si no se gestiona adecuadamente, puede poner en riesgo los datos, la privacidad de los usuarios y la seguridad del sistema en general. Aquí algunos puntos clave:

- Protección de datos sensibles: Usuarios no autorizados podrían acceder a información privada si no se valida correctamente.
- Evita ataques como inyecciones o fuerza bruta.
- UX: Validar accesos garantiza que cada usuario acceda a lo que corresponde.

Por eso, también es importante seguir buenas prácticas, como encriptar con HASH, usar sesiones seguras, etc...