



---

# FUNCIONES ASÍNCRONAS CON PHP Y AJAX

---

Anabel Martínez Perdomo



## ÍNDICE

Introducción .....	2
Tecnologías utilizadas .....	2
Estructura del proyecto.....	2
Código .....	3
Base de datos .....	3
Backend.....	5
Inicio de sesión.....	6
Registro .....	7
Cierre de sesión.....	8
Valoración anterior .....	9
Insertar nueva valoración.....	10
Frontend.....	11
JavaScript.....	11
Página principal e inicio de sesión.....	14
Página de registro.....	15
Página de productos.....	16
Demostración .....	19

## Introducción

El objetivo de este proyecto es utilizar la programación con funciones asíncronas para crear una aplicación de valoraciones. Esta aplicación consistirá en una página de inicio de sesión, una de registro y una de valoraciones de productos.

En la página de la valoración de los productos, se mostrará el nombre del usuario que ha iniciado sesión y se desplegará una lista de productos de manera dinámica desde la base de datos. Estos productos tendrán cada uno la información que aparece en la base de datos y, además, una sección en la que el usuario puede hacer click en una estrella determinada para enviar un valor a la base de datos. Esta última funcionalidad se realizará de manera asíncrona e interactiva para proporcionar una experiencia de usuario óptima.

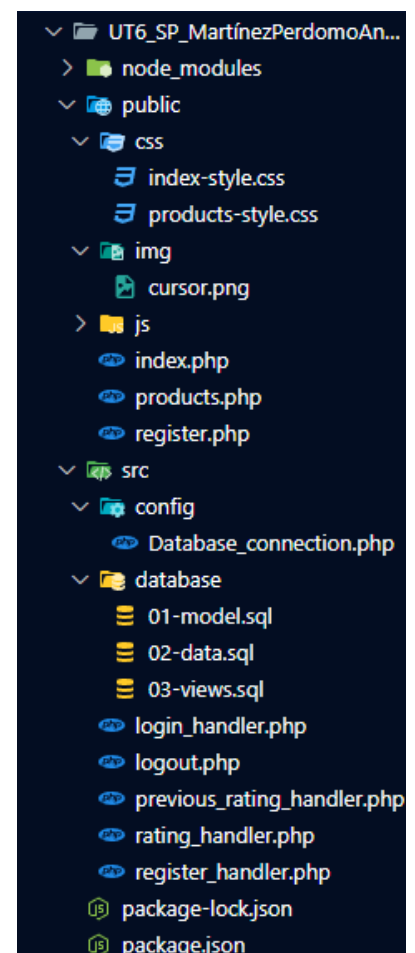
## Tecnologías utilizadas

Para este proyecto se ha utilizado **PHP** para la programación en servidor, **JavaScript** para la programación en el lado del cliente y, para la comunicación asíncrona con AJAX entre estas dos tecnologías, **Axios**, una librería de JavaScript que facilita esta función.

## Estructura del proyecto

El proyecto se divide en los siguientes directorios:

- **Node\_modules**
  - Incluye los archivos necesarios para que *node* pueda gestionar los paquetes de las librerías y módulos que necesita el proyecto.
- **Public**
  - Contiene los archivos necesarios para el *frontend*, archivos de hojas de estilo, de JavaScript y *html*.
- **Src**
  - Contiene los archivos necesarios para el funcionamiento del *backend* de la aplicación, como la clase de conexión a la base de datos, los archivos necesarios para crear esta base de datos y, por último, los *handlers* para el registro, el inicio de sesión, el cierre de sesión, la gestión de las valoraciones y de las valoraciones anteriores.



## Código

### Base de datos

Esta es la base de datos que se ha utilizado para la aplicación. Tiene tres tablas, la de usuarios, la de productos y la de los votos, que reunirá las valoraciones que los usuarios le hayan dado a los productos.

```
1  -- Eliminar La base de datos si ya existe
2  DROP DATABASE IF EXISTS video_games_rating;
3
4  -- Crear La base de datos
5  CREATE DATABASE video_games_rating;
6
7  -- Usar La base de datos
8  USE video_games_rating;
9
10 -- Crear La tabla 'usuarios'
11 CREATE TABLE
12     IF NOT EXISTS usuarios (
13         usuario INT AUTO_INCREMENT PRIMARY KEY,
14         name VARCHAR(50) NOT NULL,
15         email VARCHAR(100) UNIQUE NOT NULL,
16         pass VARCHAR(255) NOT NULL,
17         register_date DATETIME DEFAULT CURRENT_TIMESTAMP
18     );
19
20 -- Crear La tabla 'video_games'
21 CREATE TABLE
22     IF NOT EXISTS productos (
23         id INT AUTO_INCREMENT PRIMARY KEY,
24         name VARCHAR(100) NOT NULL,
25         description TEXT,
26         price DECIMAL(10, 2),
27         developer VARCHAR(100)
28     );
29
30 CREATE TABLE votos (
31     id INT AUTO_INCREMENT PRIMARY KEY,
32     cantidad INT DEFAULT 0,
33     idPr INT NOT NULL,
34     idUs INT NOT NULL, -- Cambiado a INT para que coincida con el tipo de 'usuario' en 'usuarios'
35     CONSTRAINT fk_votos_usu FOREIGN KEY (idUs) REFERENCES usuarios(usuario) ON DELETE CASCADE ON
36     UPDATE CASCADE,
37     CONSTRAINT fk_votos_pro FOREIGN KEY (idPr) REFERENCES productos(id) ON DELETE CASCADE ON
38     UPDATE CASCADE
39 );
```

Para poder obtener una media de las valoraciones de todos los usuarios y la cantidad de valoraciones totales para cada producto, se ha creado esta vista de la base de datos.

```
1  USE video_games_rating;
2
3  -- VIEW THAT SHOWS AVERAGE RATING AND NUMBER OF RATINGS FOR EACH VIDEO GAME
4  CREATE VIEW
5      producto_ratings AS
6  SELECT
7      productos.id AS product_id,
8      productos.name AS product_name,
9      productos.description,
10     productos.price,
11     productos.developer,
12     COALESCE(AVG(votos.cantidad), 0) AS avg_rating,
13     COUNT(votos.id) AS ratings_count
14 FROM
15     productos
16     LEFT JOIN votos ON productos.id = votos.idPr
17 GROUP BY
18     productos.id;
```

A continuación, se ha poblado la base de datos con datos para poder trabajar con ellos. Se han añadido productos, usuarios y valoraciones para los productos por parte de estos usuarios.

```
91     (
92         'Roberto Silva',
93         'roberto.silva@example.com',
94         'roberto3637'
95     ),
96     (
97         'Mónica Rojas',
98         'monica.rojas@example.com',
99         'monica3839'
100    );
101
102    -- Insertar 20 videojuegos
103    INSERT INTO
104        productos (name, description, price, developer)
105    VALUES
106        (
107            'The Legend of Zelda: Breath of the Wild',
108            'Un juego de aventuras en un mundo abierto.',
109            59.99,
110            'Nintendo'
111        ),
112        (
113            'Cyberpunk 2077',
114            'Un RPG de mundo abierto en un futuro distópico.',
115            49.99,
116            'CD Projekt Red'
117        ),
```

## Backend

Para poder conectar nuestra aplicación con la base de datos, he creado esta clase con el método **getConnection()**, que posteriormente implementaré en el resto de páginas.

```
1  <?php
   13 references | 0 implementations
2  class Database
3  {
   1 reference
4      private static $DBhost = '127.0.0.1';
   1 reference
5      private static $DBname = 'video_games_rating';
   1 reference
6      private static $DBuser = 'root'; // Default XAMPP
   1 reference
7      private static $DBpassword = ''; // Default XAMPP
   4 references
8      private static $PDO = null;
9
   2 references
10     private function __construct() {}
11
   5 references | 0 overrides
12     public static function getConnection(): mixed
13     {
14         if (self::$PDO === null) {
15             $DBlink = 'mysql:host=' . self::$DBhost . ';dbname=' . self::$DBname;
16             try {
17                 self::$PDO = new PDO(dsn: $DBlink, username: self::$DBuser,
18                                     password: self::$DBpassword);
19                 self::$PDO->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
20             } catch (PDOException $e) {
21                 die('Connection failed: ' . $e->getMessage());
22             }
23             return self::$PDO;
24         }
25     }
```

## Inicio de sesión

Este es el código del *handler* que se encargará del inicio de sesión, `login_handler.php`.

```

1  <?php
2  session_start();
3  require_once './config/Database_connection.php';
4
5  if ($_SERVER['REQUEST_METHOD'] === 'POST') {
6      $name = $_POST['name'];
7      $password = $_POST['password'];
8
9      if (empty($name) || empty($password)) {
10         die('Please enter a name and password');
11     }
12
13     try {
14         // DB connection
15         $pdo = Database::getConnection();
16
17         // Consulta SQL para buscar al usuario por el nombre
18         $query = "SELECT usuario, name, pass FROM usuarios WHERE name = ?";
19         $stmt = $pdo->prepare($query);
20         $stmt->execute([$name]);
21
22         if ($stmt->rowCount() === 1) { //si hay usuario con ese nombre
23             $user = $stmt->fetch(PDO::FETCH_ASSOC);
24
25             // password_verify porque la pass está hashed en DB.
26             if (password_verify(password: $password, hash: $user['pass'])) {
27
28                 // Guarda variables en sesión para uso entre páginas
29                 $_SESSION['user_id'] = $user['usuario'];
30                 $_SESSION['username'] = $user['name'];
31
32                 header(header: 'Location: ../public/products.php');
33                 exit();
34             } else {
35                 die('Invalid password');
36             }
37         } else {
38             die('User not found');
39         }
40     } catch (PDOException $e) {
41         error_log(message: 'Database Error: ' . $e->getMessage());
42         die('Something went wrong. Please try again.');
```

Como podemos ver, se encarga de abrir la sesión, importar la conexión a la base de datos e interactuar con el formulario. Si el usuario no introduce tanto nombre como contraseña, se frena la ejecución.

Busca en la base de datos algún usuario con ese nombre y, si existe, verifica que la contraseña del usuario sea la que se ha introducido por formulario (utilizando el método `password_verify` para que tenga en cuenta que la contraseña ha sido encriptada). Si ese usuario existe y esa es su contraseña, **se guarda en la sesión tanto el ID del usuario como el nombre** para utilizarlo en el resto de páginas.

## Registro

Este es el código de `register_handler.php`:

```

1  <?php
2  session_start(); // Inicia la sesión
3
4  // Include database connection
5  require_once '../config/Database_connection.php';
6
7  if ($_SERVER['REQUEST_METHOD'] == 'POST') {
8      // Get the form data
9      $name = $_POST['name'];
10     $email = $_POST['email'];
11     $password = $_POST['password'];
12
13     // Basic form validation
14     if (empty($name) || empty($email) || empty($password)) {
15         $_SESSION['flash_error'] = "Por favor, completa todos los campos (nombre, email y
16         contraseña).";
17         header(header: 'Location: ../public/register.php'); // Redirige de vuelta al formulario
18         exit();
19     }
20
21     // Hash the password
22     $hashedPassword = password_hash($password, PASSWORD_DEFAULT);
23
24     try {
25         // Get DB connection
26         $pdo = Database::getConnection();
27
28         // Check if the email already exists
29         $query = "SELECT usuario FROM usuarios WHERE email = :email";
30         $stmt = $pdo->prepare($query);
31         $stmt->bindParam(':email', $email);
32         $stmt->execute();

```

Abre la sesión, importa la conexión a la base de datos e interactúa con el formulario. A continuación, hace unas validaciones básicas de formulario y, si no se han completado todos los campos, muestra un mensaje de error.

Si el usuario introduce el *input* esperado, encripta la contraseña antes de enviarla a la base de datos. A continuación, se crea un bloque *try-catch* en el que se intenta conectar a la base de datos y comprobar que el email de este usuario no esté registrado con una consulta SQL.

```

31     $stmt->execute();
32
33     if ($stmt->rowCount() > 0) {
34         $_SESSION['flash_error'] = "El correo electrónico ya está registrado.
35         ";
36         header(header: 'Location: ../public/register.php'); // Redirige de
37         vuelta al formulario
38         exit();
39     } else {
40         // Insert new user into the 'usuarios' table
41         $query = "INSERT INTO usuarios (name, email, pass) VALUES (:name,
42         :email, :password)";
43         $stmt = $pdo->prepare($query);
44         $stmt->bindParam(':name', $name);
45         $stmt->bindParam(':email', $email);
46         $stmt->bindParam(':password', $hashedPassword);
47
48         if ($stmt->execute()) {
49             header(header: 'Location: ../public/index.php'); // Redirige a
50             la página de inicio de sesión
51             exit();

```



A continuación, si el correo electrónico ya está registrado, redirige al usuario de vuelta al formulario de registro con un mensaje flash en la sesión para mostrarle lo que ha ocurrido.

Si, por el contrario, el correo electrónico no existe en la base de datos, realizará una consulta SQL para insertar en la base de datos al usuario, su correo electrónico y su contraseña (encriptada anteriormente). Cuando todo esto ocurra enviará al usuario a la página de inicio de sesión.

```

46
47         header(header: 'Location: ../../public/index.php'); // Redirige a
           la página de inicio de sesión
48         exit();
49     } else {
50         // Capture detailed error if insert fails
51         $errorInfo = $stmt->errorInfo();
52         $_SESSION['flash_error'] = "Error al registrar el usuario: " .
           $errorInfo[2];
53         error_log(message: "SQL error during insert: " . $errorInfo
           [2]); // Log the error
54         header(header: 'Location: ../../public/register.php'); // Redirige
           de vuelta al formulario
55         exit();
56     }
57 }
58 } catch (PDOException $e) {
59     // Log detailed error message and display a generic message to the user
60     error_log(message: 'Database Error: ' . $e->getMessage());
61     $_SESSION['flash_error'] = 'Hubo un error en la base de datos. Inténtalo
           nuevamente más tarde.';
62     header(header: 'Location: ../../public/register.php'); // Redirige de
           vuelta al formulario
63     exit();
64 }
65 } else {
66     $_SESSION['flash_error'] = 'Método de solicitud no válido. Por favor, envía
           el formulario correctamente.';
67     header(header: 'Location: ../../public/register.php'); // Redirige de vuelta
           al formulario
68     exit();
69 }

```

Si esto falla, recogerá el fallo y lo mostrará en el mensaje flash que aparecerá una vez se recargue la página.

#### Cierre de sesión

Este es el archivo de PHP que se encarga del cierre de sesión:

```

1  <?php
2  // Iniciar sesión
3  session_start();
4
5  // Eliminar todas las variables de sesión
6  session_unset();
7
8  // Destruir la sesión
9  session_destroy();
10
11 // Redirigir al inicio o a la página de login
12 header(header: 'Location: ../public/index-login.php');
13 exit;

```

## Valoración anterior

Este es el código de **previous\_rating\_handler.php**, que se encarga de tomar, desde la base de datos, la valoración del usuario que ha iniciado sesión para cada producto.

```
1  <?php
2
3  require_once 'config/Database_connection.php';
4
5  // Inicia sesión
6  session_start();
7  $user_id = $_SESSION['user_id'];
8  // consigue el producto y el rating para el usuario que ha iniciado sesión
9  $query = "SELECT idPr, cantidad
10 FROM votos
11 WHERE idUs = :user_id";
12
13 $pdo = Database::getConnection();
14 $stmt = $pdo->prepare($query);
15 $stmt->bindParam(':user_id', $user_id);
16 $stmt->execute();
17
18 $ratings = $stmt->fetchAll(PDO::FETCH_ASSOC);
19
20 // Mapeo entre id de producto y cantidad de estrellas que el usuario ha dado
   anteriormente
21 $user_ratings_map = [];
22 foreach ($ratings as $rating) {
23     $user_ratings_map[$rating['idPr']] = $rating['cantidad'];
24 }
```

Importa la conexión a la base de datos, abre la sesión y crea la variable de sesión del ID de usuario. A continuación, a través de una consulta SQL, obtiene el producto y la valoración que le ha dado el usuario anteriormente.

Luego, crea un mapeo entre el id de producto y la valoración que el usuario ha dado anteriormente. Es decir, crea un array asociativo en el que la clave será el id de producto y el valor será la valoración que le ha dado el usuario.

## Insertar nueva valoración

Este es el código de **rating\_handler.php**, que se encarga de gestionar las valoraciones de los usuarios:

```

1  <?php
2  session_start();
3  require_once 'config/Database_connection.php';
4
5  // Verificar que el usuario ha iniciado sesión
6  if (!isset($_SESSION['user_id'])) {
7      echo json_encode(['success' => false, 'message' => 'Usuario no autenticado.']);
8      exit;
9  }
10
11  $product_id = $_POST['product_id'];
12  $rating = $_POST['rating'];
13  $user_id = $_SESSION['user_id'];
14
15  try {
16      $pdo = Database::getConnection();
17
18      // Verificar si el usuario ya ha valorado este producto
19      $query = "SELECT id FROM votos WHERE idPr = ? AND idUs = ?";
20      $stmt = $pdo->prepare($query);
21      $stmt->execute([$product_id, $user_id]);
22
23      if ($stmt->rowCount() > 0) { // Si el usuario ya ha votado, actualiza la valoración
24
25          $query = "UPDATE votos SET cantidad = ? WHERE idPr = ? AND idUs = ?";
26          $stmt = $pdo->prepare($query);
27          $stmt->execute([$rating, $product_id, $user_id]);
28
29

```

Este *handler* abre la sesión, verifica que el usuario ha iniciado sesión y envía un mensaje con esta información. A continuación, toma las variables del formulario e intenta conectar con la base de datos para hacer una consulta SQL.

En esta consulta, se obtienen los votos para un producto por parte del usuario que haya iniciado sesión. Si esta consulta devuelve datos, se procede a una actualización de la votación, porque el **usuario solo puede tener una nota para un producto**.

```

29      } else { // Si el usuario no ha votado, inserta una nueva valoración
30
31          $query = "INSERT INTO votos (cantidad, idPr, idUs) VALUES (?, ?, ?)";
32          $stmt = $pdo->prepare($query);
33          $stmt->execute([$rating, $product_id, $user_id]);
34      }
35
36      // Obtener la nueva media de valoraciones y la cantidad de valoraciones para el producto
37      $query = "SELECT AVG(cantidad) AS avg_rating, COUNT(id) AS ratings_count FROM votos WHERE idPr = ?";
38      $stmt = $pdo->prepare($query);
39      $stmt->execute([$product_id]);
40      $result = $stmt->fetch(PDO::FETCH_ASSOC);
41
42      // Asegurarnos de que 'avg_rating' sea un número válido. Si no es un número, lo asignamos como 0.
43      $avg_rating = (float)$result['avg_rating']; // Convierte a float para evitar errores
44      $ratings_count = (int)$result['ratings_count']; // Asegura que la cantidad de valoraciones sea un número entero
45
46      // Si 'avg_rating' no es un número válido lo asigna como 0
47      if ($avg_rating === 0) {
48          $avg_rating = 0;
49      }

```

Si, por el contrario, el usuario no tiene ninguna votación para ese producto, se inserta una nueva con una consulta SQL.

A continuación, se obtiene la nueva media de las valoraciones y la cantidad de valoraciones que tiene un producto, para la actualización asíncrona del elemento en la página web. Para evitar problemas, se convierte el número de la media a un *float*, y nos aseguramos también de que la cantidad de valoraciones sea un número *int*. Si, por el contrario, el número de la media fuera un valor no válido, evitamos errores asignándole valor 0.

```
50
51 // Respuesta a frontend con nueva media y número de valoraciones
52 echo json_encode(value: [
53     'success' => true,
54     'new_avg_rating' => $avg_rating, // float: media de valoraciones
55     'new_ratings_count' => $ratings_count // Num valoraciones
56 ]);
57 } catch (PDOException $e) {
58     error_log(message: 'Database Error: ' . $e->getMessage());
59     echo json_encode(value: ['success' => false, 'message' => 'Error al procesar la solicitud
60     ']);
61 }
```

Por último, enviamos una respuesta al *frontend* con la nueva media y el número de valoraciones después de realizar las operaciones y capturamos algún posible error que haya podido darse con la conexión a la base de datos.

## Frontend

### JavaScript

Este es el código de **rating.js**, el código JavaScript que se va a encargar de que **products.php** se comunique, a través de Axios, con **rating\_handler.php**. Esto hace que luego, en la página de los productos, no tengamos que hacer ningún envío a través de ningún formulario a PHP, ya que en JavaScript tendremos eventos que hacen que al hacer click en nuestras estrellas se envíe la información.

```
1 document.addEventListener("DOMContentLoaded", function () {
2     const starContainers = document.querySelectorAll(".star-rating");
3
4     starContainers.forEach((container) => {
5         const stars = container.querySelectorAll(".star");
6         const productId = container.getAttribute("data-product-id");
7
8         // Verifica que product id sea válido
9         if (!productId) {
10             console.error(
11                 "Product ID is missing or invalid for container:",
12                 container
13             );
14             return;
15         }
16     });
17 }
```

Esta primera parte del código establece las constantes que vamos a utilizar en JavaScript y, si no existe ID de producto, lanza un error.

```

17     stars.forEach((star) => {
18         star.addEventListener("click", function () {
19             const rating = parseInt(this.getAttribute("data-value"), 10);
20
21             if (!isNaN(rating)) {
22                 updateStars(container, rating);
23                 submitRating(productId, rating);
24             } else {
25                 console.error("Invalid star value:", this.getAttribute("data-value"));
26             }
27         });
28     });
29 });
30 });

```

Lo siguiente que vemos en el código es que, para cada estrella, se crea un evento que provoca que, cuando el usuario hace click, se ejecuta una función que toma el valor de la estrella, dado desde el HTML de manera dinámica a través del bucle de generación, y lo convierte en un número entero decimal. Si este valor es válido, actualiza las estrellas y envía el valor a través de las funciones que veremos a continuación.

```

32 // Actualiza las estrellas del producto
33 function updateStars(container, rating) {
34     const stars = container.querySelectorAll(".star");
35     stars.forEach((star, index) => {
36         if (index < rating) {
37             star.classList.add("filled");
38         } else {
39             star.classList.remove("filled");
40         }
41     });
42 }
43
44 // Envía la información de la valoración al servidor
45 function submitRating(productId, rating) {
46     // Añadido para depuración por consola :)
47
48     console.log("Sending rating for Product ID:", productId, "Rating:", rating);
49
50     // Cambiar a formato x-www-form-urlencoded en lugar de JSON para enviar datos
51     // "product_id=123&rating=4". Esto permite acceder desde $_POST en php sin tener que
52     // decodificar json.
53     const params = new URLSearchParams();
54     params.append("product_id", productId);
55     params.append("rating", rating);
56 }

```

La función **updateStars()** hace que las estrellas se llenen al pulsar una estrella superior. Esto causa que, si se pulsa la cuarta estrella, se rellenen todas, la cuarta incluida.

La función **submitRating()** envía un mensaje por consola, para facilitar la depuración en caso de que haya que hacer pruebas y, a continuación, cambia el formato de los datos a **x-www-form-urlencoded**. El motivo de esto es para evitar problemas con la interacción con PHP. PHP normalmente acepta datos a través de la variable global **\$\_POST**, por lo que vamos a utilizar este formato, que envía los datos en un formato que, por ejemplo, para el producto "123" con

valoración "4" sería `"product_id=123&rating=4"`. Esto se parece a la forma que hay de comunicar la información a PHP entre páginas de manera automática, sin pasar por formularios.

```
55 // Usa axios para enviar a PHP
56
57 axios
58   .post("../src/rating_handler.php", params)
59   .then(function (response) {
60     if (response.data.success) {
61       updateProductRow(productId, response.data);
62     } else {
63       // Mostrar el error en consola para más detalles
64       console.error("Error from server:", response.data.message);
65       alert("Error: " + response.data.message);
66     }
67   })
68   .catch(function (error) {
69     // Depuración en caso de error en la solicitud
70     console.error("Error while submitting rating:", error);
71     alert("Error while submitting rating. Please try again later.");
72   });
73 }
```

La siguiente parte de nuestro JavaScript hace una llamada a **Axios** para enviar a `rating_handler.php` los parámetros que hemos definido anteriormente en formato `x-www-form-urlencoded`. La respuesta será una llamada a la función `updateProductRow`, que veremos a continuación:

```
75 // Actualiza la fila del producto para mostrar la nueva info
76 function updateProductRow(productId, data) {
77   const row = document.getElementById("producto-" + productId);
78   if (row) {
79     const avgRating = data.new_avg_rating;
80
81     if (avgRating !== null && avgRating !== undefined && !isNaN(avgRating)) {
82       row.querySelector(".avg-rating").textContent = avgRating.toFixed(1);
83     } else {
84       row.querySelector(".avg-rating").textContent = "N/A"; // mostrar N/A si no hay
85       // calificación válida
86     }
87
88     row.querySelector(".ratings-count").textContent = data.new_ratings_count;
89   }
90 }
```

En esta función se define la fila del producto que se va a actualizar. Se define que la media de valoraciones va a ser la media que se reciba de la respuesta a través de **Axios**. Si este valor no está vacío, ni indefinido, y es un valor numérico válido, se define ese campo de la tabla de nuevo. Si, por el contrario, es un número no válido, para evitar errores, lo actualizaremos con el contenido "N/A".

Por último, actualizamos la cantidad de valoraciones con la que nos dará la respuesta de **Axios**.

Página principal e inicio de sesión

Este es el código de la página principal, en la que se debe iniciar sesión:

```
UT6_SP_MartínezPerdomoAnabel > public > index.php > html > body
1  <!DOCTYPE html>
2  <html lang="en">
3
4  <head>
5      <meta charset="UTF-8">
6      <meta name="viewport" content="width=device-width, initial-scale=1.0">
7      <title>Login - Video Game Rating</title>
8      <link href="../css/index-style.css" rel="stylesheet">
9  </head>
10
11 <body>
12     <div class="container">
13         <h1>Te damos la bienvenida a Video Game Rating App!</h1>
14         <p>Por favor, inicia sesión para empezar a valorar tus juegos favoritos.</p>
15
16         <h2>Iniciar sesión</h2>
17         <form action="../src/login_handler.php" method="POST">
18             <label for="name">Nombre de usuario:</label>
19             <input type="text" id="name" name="name" required>
20
21             <label for="password">Contraseña:</label>
22             <input type="password" id="password" name="password" required>
23
24             <button type="submit">Iniciar sesión</button>
25         </form>
26         <p>¿No tienes cuenta? <a href="../register.php">Regístrate</a></p>
27     </div>
28
29 </body>
```

Tiene un formulario que envía la información al **login\_handler.php** que hemos tratado anteriormente.

Página de registro

Este es el código de **register.php**:

```

1  <?php
2  session_start(); // Inicia la sesión
3
4  // Verifica si hay mensajes flash de error o éxito
5  $flash_error = isset($_SESSION['flash_error']) ? $_SESSION['flash_error'] : null;
6
7  // Elimina los mensajes flash después de mostrarlos
8  unset($_SESSION['flash_error']);
9  ?>
10
11 <!DOCTYPE html>
12 <html lang="en">
13
14 <head>
15     <meta charset="UTF-8">
16     <meta name="viewport" content="width=device-width, initial-scale=1.0">
17     <title>Register - Video Game Rating</title>
18     <link href="/css/index-style.css" rel="stylesheet">
19 </head>
20
21 <body>
22     <div class="container">
23         <h1>Welcome to Video Game Rating App!</h1>
24         <p>We are excited to have you here. Please register to start rating your favorite
25         games!</p>
26
27         <!-- Mostrar mensajes de error -->
28         <?php if ($flash_error): ?>
29             <div class="flash-message error">
30                 <?php echo htmlspecialchars(string: $flash_error); ?>
31             </div>
32         <?php endif; ?>
33
34         <h2>Register</h2>
35         <form action="/src/register_handler.php" method="POST">
36             <label for="name">Username:</label>
37             <input type="text" id="name" name="name" required>
38
39             <label for="email">Email:</label>
40             <input type="email" id="email" name="email" required>
41
42             <label for="password">Password:</label>
43             <input type="password" id="password" name="password" required>
44
45             <button type="submit">Register</button>
46         </form>
47         <p>¿Ya tienes cuenta? <a href="index.php">Inicia sesión</a></p>
48     </div>
49 </body>
50 </html>

```

Como podemos ver, contiene una condición que implica que, si hay definida una variable para mensajes de error, los mostrará. Esta variable viene dada por la sesión.

A continuación, vemos un formulario que envía la información a **register\_handler.php**, que se encargará de gestionarla.



## Página de productos

Este es el código de **products.php**, que se divide en un bloque PHP y un bloque HTML:

```

1  <?php
2  require_once '../src/previous_rating_handler.php';
3
4  // Verifica si la sesión está iniciada
5  if (!isset($_SESSION['user_id'])) {
6      header(header: 'Location: index.php');
7      exit;
8  }
9
10 // DB
11 require_once '../src/config/Database_connection.php';
12
13 // DB conexión
14 try {
15     $pdo = Database::getConnection();
16
17     // Obtiene * de la vista
18     $query = "SELECT * FROM producto_ratings";
19     $stmt = $pdo->prepare(query: $query);
20     $stmt->execute();
21
22     // Obtener todos los productos
23     $productos = $stmt->fetchAll(PDO::FETCH_ASSOC);
24
25     if (!$productos) {
26         throw new Exception(message: 'No se encontraron productos en la base de datos.');
```

En el bloque PHP, podemos ver que se implementa **previous\_rating\_handler.php** para gestionar que cuando se inicie la página se carguen las valoraciones teniendo en cuenta las valoraciones anteriores del usuario que ha iniciado sesión.

A continuación, se verifica si la sesión está iniciada. Como la página de productos está protegida sesión, no se podrá acceder a esta sin haber pasado por el formulario de inicio de sesión.

Lo siguiente que hace este bloque PHP es intentar la conexión a la base de datos para obtener toda la información de la vista que creamos para esto y también para obtener todos los productos disponibles desde la base de datos.

```

36 <!DOCTYPE html>
37 <html lang="es">
38
39 <head>
40     <meta charset="UTF-8">
41     <meta name="viewport" content="width=device-width, initial-scale=1.0">
42     <title>Video Game Rating App - Productos</title>
43     <link rel="stylesheet" href="../css/products-style.css">
44     <script src="https://cdn.jsdelivr.net/npm/axios/dist/axios.min.js"></script>
45 </head>
46
```

En el bloque HTML, podemos ver que se introduce el CDN de **Axios** para preparar la interacción asíncrona.

```
47 <body>
48   <header>
49     <h1>Video Game Rating App</h1>
50     <!-- Muestra el usuario Loggeado -->
51     <p>Valoraciones de <?php echo $_SESSION['username']; ?></p>
52     <a href="../src/logout.php">Cerrar sesión</a>
53   </header>
54   <div class="container">
55     <h1>Videojuegos</h1>
56     <div class="table-container">
57       <table>
58         <thead>
59           <tr>
60             <th>Nombre</th>
61             <th>Descripción</th>
62             <th>Precio</th>
63             <th>Desarrollador</th>
64             <th>Media</th>
65             <th>Valoraciones</th>
66             <th>Tu Valoración</th>
67           </tr>
68         </thead>
```

Lo siguiente que vemos, en el cuerpo de la página, es el header, que incluye el título de la página, un mensaje personalizado con el nombre de usuario que haya iniciado sesión y un botón para cerrar sesión.

A continuación, se crea un contenedor en el que se verá la tabla y sus encabezados de tabla.

```

69         <tbody>
70             <?php if ($productos): ?>
71                 <?php foreach ($productos as $producto): ?>
72                     <tr id="producto-<?php echo $producto['product_id']; ?>">
73
74                         <td data-label="Nombre"><?php echo htmlspecialchars
75                             (string: $producto['product_name']); ?></td>
76
77                         <td data-label="Descripción"><?php echo htmlspecialchars
78                             (string: $producto['description']); ?></td>
79
80                         <td data-label="Precio"><?php echo number_format
81                             (num: $producto['price'], decimals: 2); ?> €</td>
82
83                         <td data-label="Desarrollador"><?php echo htmlspecialchars
84                             (string: $producto['developer']); ?></td>
85
86                         <td data-label="Media" class="avg-rating"><?php echo
87                             number_format(num: $producto['avg_rating'], decimals: 1); ?></
88                             td>
89
90                         <td data-label="Valoraciones" class="ratings-count"><?php
91                             echo $producto['ratings_count']; ?></td>
92
93                         <td data-label="Tu Valoración">
94                             <div class="star-rating" data-product-id="<?php echo
95                                 $producto['product_id']; ?>">
96                                 <!-- Crea estrellas con loop, da a cada estrella el
97                                 valor de i -->
98                                 <?php for ($i = 1; $i <= 5; $i++): ?>
99                                     <span class="star <?php echo (isset
100                                         ($user_ratings_map[$producto['product_id']]) &&
101                                         $user_ratings_map[$producto['product_id']] >=
102                                         $i) ? 'filled' : ''; ?>"
103                                         data-value="<?php echo $i; ?>">★</span>
104                                 <?php endfor; ?>
105                             </div>
106                         </td>
107                     </tr>
108                 <?php endforeach; ?>
109             <?php else:
110                 <tr>
111                     <td colspan="7">No hay productos disponibles.</td>
112                 </tr>
113             <?php endif; ?>

```

Se genera, de manera dinámica, para cada producto que haya en la base de datos, un conjunto de los siguientes datos:

- Nombre del producto
- Descripción del producto
- Precio del producto
- Desarrollador
- Media de las valoraciones
- Cantidad de valoraciones

Y, además, un campo que contiene estrellas que actúan como botones para la valoración interactiva. Al crearse por un bucle en el que reciben el valor de cuántos bucles se hayan repetido, es completamente escalable, si quisiéramos aumentar la cantidad de estrellas solamente deberíamos cambiar el límite del bucle. Además, si la página obtiene que ya existe una valoración anterior por parte de este usuario (a través de **previous\_rating\_handler.php**), se

le dará la clase “*filled*”, lo que hará que se muestre la estrella coloreada para que la experiencia de usuario sea eficiente y divertida.

```

107     <footer>
108         <p>0 Anabel Martínez Perdomo <br><a href="https://github.com/Rachni"><svg
            class="github-link" xmlns="http://www.w3.org/2000/svg" viewBox="0 0 24 24"
            fill="none" stroke="currentColor" stroke-linecap="round" stroke-linejoin="round"
            width="24" height="24" stroke-width="2">
109             <path d="M9 19c-4.3 1.4 -4.3 -2.5 -6 -3m12 5v-3.5c0 -1 .1 -1.4 -.5 -2c2.8
            -.3 5.5 -1.4 5.5 -6a4.6 4.6 0 0 0 -1.3 -3.2a4.2 4.2 0 0 0 -.1 -3.2s-1.1 -.
            3 -3.5 1.3a12.3 12.3 0 0 0 -6.2 0c-2.4 -1.6 -3.5 -1.3 -3.5 -1.3a4.2 4.2 0
            0 0 -.1 3.2a4.6 4.6 0 0 0 -1.3 3.2c0 4.6 2.7 5.7 5.5 6c-.6 .6 -.6 1.2 -.5
            2v3.5"></path>
110             </svg>
111         </svg></a> </p>
112     </footer>
113 </body>
114
115 </html>

```

El último bloque que nos queda por revisar del HTML simplemente es un *footer* que contiene información sobre quién ha desarrollado esta página web y un enlace.

## Demostración

Una vez entramos en la página, nos encontraremos con este formulario de inicio de sesión:

**Te damos la bienvenida a Video Game Rating App!**

Por favor, inicia sesión para empezar a valorar tus juegos favoritos.

**Iniciar sesión**

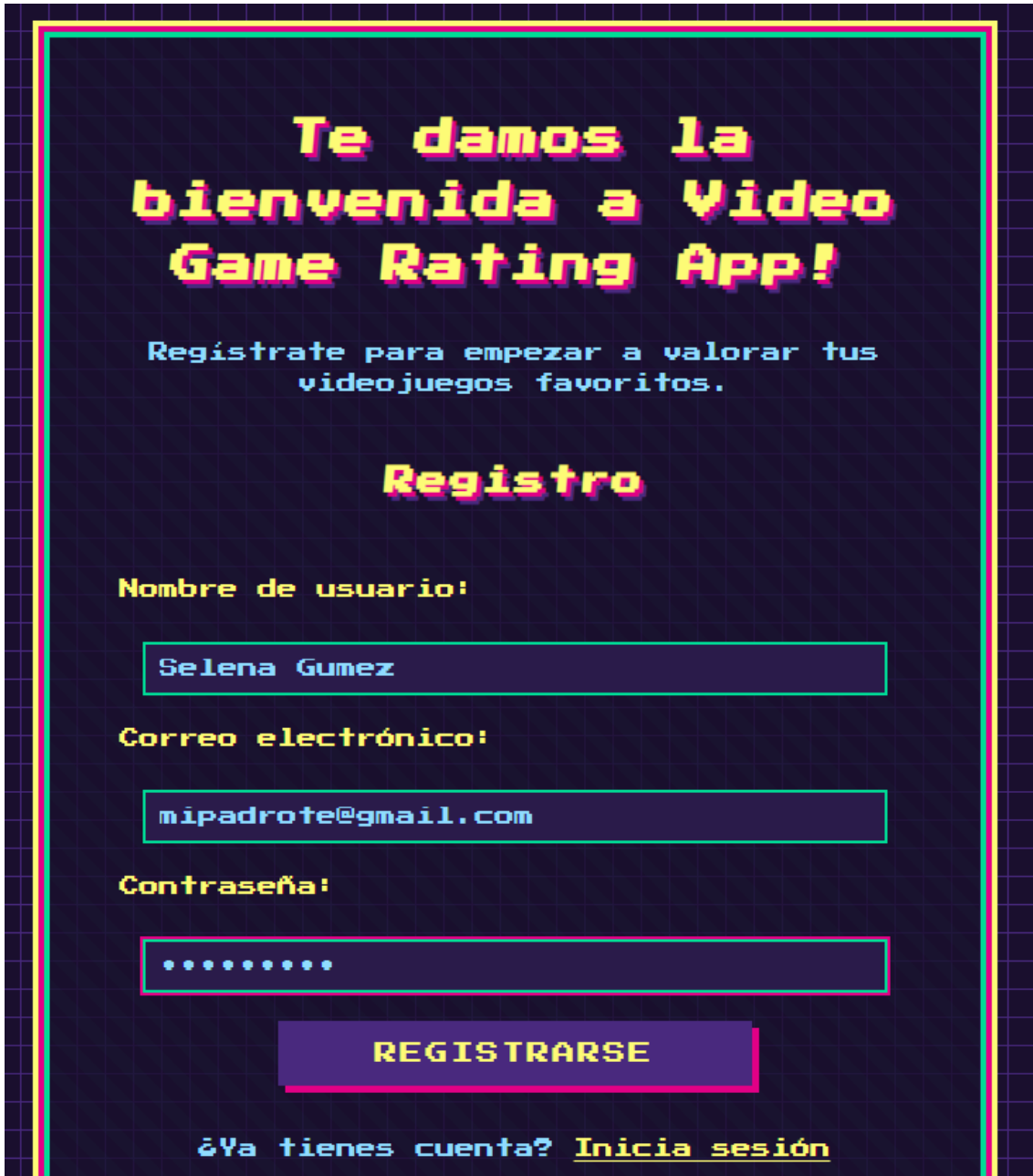
Nombre de usuario:

Contraseña:

**INICIAR SESIÓN**

¿No tienes cuenta? [Regístrate](#)

Si no tenemos cuenta, podemos pulsar el botón **Regístrate** para acceder a esta página con formulario de registro:

A registration form for a 'Video Game Rating App' with a retro, pixelated aesthetic. The background is dark blue with a light blue grid. The text is in a yellow, pixelated font. The form includes fields for username, email, and password, followed by a 'REGISTRARSE' button and a link to 'Inicia sesión' if the user already has an account.

**Te damos la bienvenida a Video Game Rating App!**

Regístrate para empezar a valorar tus videojuegos favoritos.

**Registro**

Nombre de usuario:

Correo electrónico:

Contraseña:

**REGISTRARSE**

¿Ya tienes cuenta? [Inicia sesión](#)

Una vez pulsamos el botón de **REGISTRARSE**, se crea este usuario en la base de datos:

26	Selena Gumez	mipadrote@gmail.com	\$2y\$10\$EuISMSR0
----	--------------	---------------------	--------------------

Ahora, podemos acceder a la página con estas credenciales.

Al acceder a **products.php**, a través del inicio de sesión, veremos esto:

Video Game Rating App						
Valoraciones de Selena Gumez						Cerrar sesión
Videojuegos						
NOMBRE	DESCRIPCIÓN	PRECIO	DESARROLLADOR	MEDIA	VALORACIONES	TU VALORACIÓN
The Legend of Zelda: Breath of the Wild	Un juego de aventuras en un mundo abierto.	59.99 €	Nintendo	3.6	7	★★★★★
Cyberpunk 2077	Un RPG de mundo abierto en un futuro distópico.	49.99 €	CD Projekt Red	4.9	7	★★★★★
Elden Ring	Un juego de acción y rol en un mundo de fantasía.	69.99 €	FromSoftware	3.9	7	★★★★★

Como podemos apreciar, se genera un *header* dinámico con el nombre del usuario que acaba de iniciar sesión. Además, se genera la lista de manera dinámica y se generan también las cinco estrellas en color gris que, cuando el usuario pasa el cursor por encima, se vuelven amarillas, para indicar interactividad.

Si decidiésemos valorar *Cyberpunk 2077* con cinco estrellas, *Elden Ring* con cinco, *The legend of Zelda: Breath of the Wild* con una, pero *Super Mario Odyssey* lo dejásemos sin valorar, podríamos ver esto:

Video Game Rating App						
Valoraciones de Selena Gumez						Cerrar sesión
NOMBRE	DESCRIPCIÓN	PRECIO	DESARROLLADOR	MEDIA	VALORACIONES	TU VALORACIÓN
The Legend of Zelda: Breath of the Wild	Un juego de aventuras en un mundo abierto.	59.99 €	Nintendo	3.3	8	★ ★ ★ ★ ★
Cyberpunk 2077	Un RPG de mundo abierto en un futuro distópico.	49.99 €	CD Projekt Red	4.9	8	★★★★★
Elden Ring	Un juego de acción y rol en un mundo de fantasía.	69.99 €	FromSoftware	4.0	8	★★★★★
Super Mario Odyssey	Una aventura épica de Mario en diferentes reinos.	59.99 €	Nintendo	3.1	7	★★★★★

Además, como hemos implementado la función de que se rellenen las estrellas si existe una valoración por parte de este usuario en la base de datos, podríamos cerrar sesión y cuando volviésemos a iniciarla veríamos la página de la misma manera.