**Pruebas unitarias y de integración:**

**Parte del Usuario.**

**Registrarse:**public function registrarUsuario($nombre, $apellido, $nombreUsuario, $correo, $contrasena, $pais)

    {

        // Encriptar la contraseña usando password\_hash

        $contrasenaEncriptada = password\_hash($contrasena, PASSWORD\_DEFAULT);

        // Consulta SQL para insertar un nuevo usuario en la base de datos

        $sql = "INSERT INTO usuarios\_registrados (nombre\_usuario, contrasena, correo\_electronico, nombre, ubicacion, apellido) VALUES (?, ?, ?, ?, ?, ?)";

        $stmt = $this->conn->prepare($sql); // Preparar la consulta SQL

        $stmt->bind\_param("ssssss", $nombreUsuario, $contrasenaEncriptada, $correo, $nombre, $pais, $apellido); // Vincular los parámetros

        if ($stmt->execute()) { // Ejecutar la consulta

            $stmt->close(); // Cerrar la declaración

            return true; // Registro exitoso

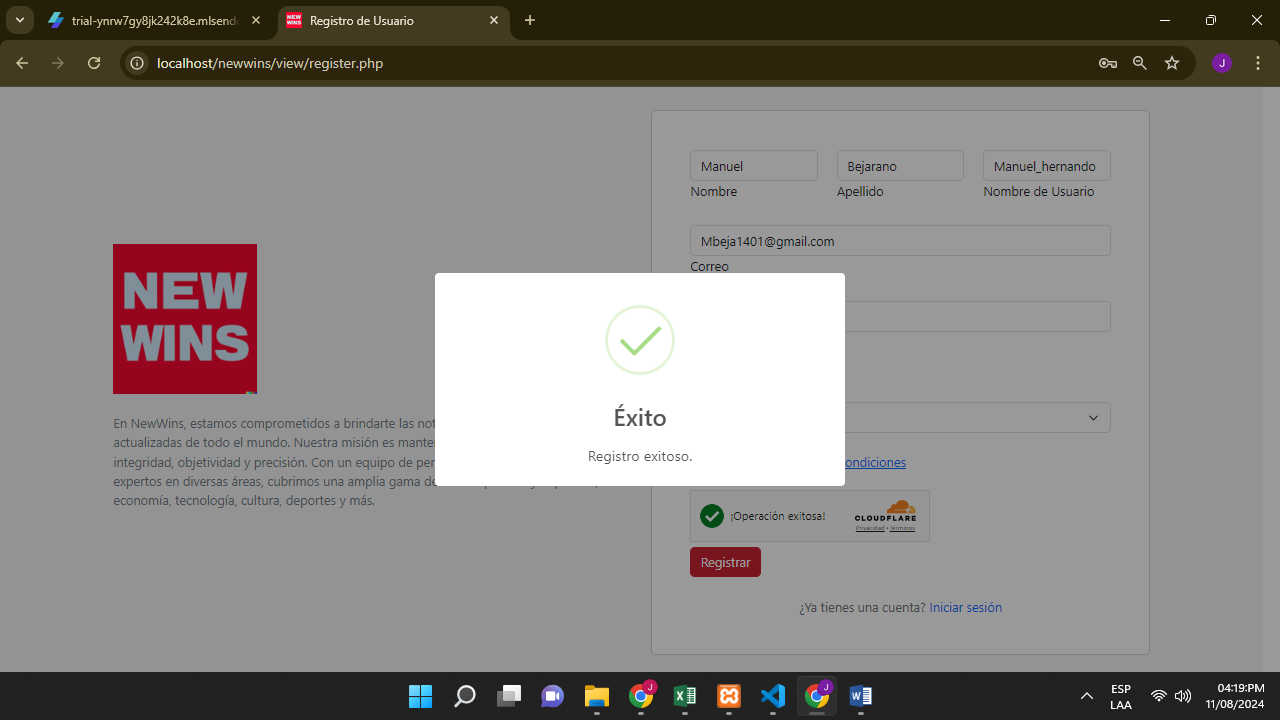
        } else {

            $stmt->close(); // Cerrar la declaración

            return false; // Error en el registro

        }

    }



**Iniciar sesión:**

public function iniciarSesion($correo, $contra)

    {

        // Consulta SQL para seleccionar al usuario por correo electrónico

        $sql = "SELECT \* FROM usuarios\_registrados WHERE correo\_electronico = ?";

        $stmt = $this->conn->prepare($sql); // Preparar la consulta SQL

        $stmt->bind\_param("s", $correo); // Vincular el parámetro de correo electrónico

        $stmt->execute(); // Ejecutar la consulta

        $result = $stmt->get\_result(); // Obtener el resultado de la consulta

        if ($result->num\_rows == 1) { // Verificar si se encontró un usuario

            $row = $result->fetch\_assoc(); // Obtener los datos del usuario

            if (password\_verify($contra, $row['contrasena'])) { // Verificar la contraseña

                // Almacenar información del usuario en la sesión

                $\_SESSION['correo'] = $row['correo\_electronico'];

                $\_SESSION['user\_id'] = $row['id'];

                $\_SESSION['user\_nombre'] = $row['nombre'];

                $\_SESSION['user\_apellido'] = $row['apellido'];

                $stmt->close(); // Cerrar la declaración

                return ['success' => true]; // Retornar éxito

            } else {

                $stmt->close(); // Cerrar la declaración

                return ['success' => false, 'message' => 'La contraseña es incorrecta.']; // Contraseña incorrecta

            }

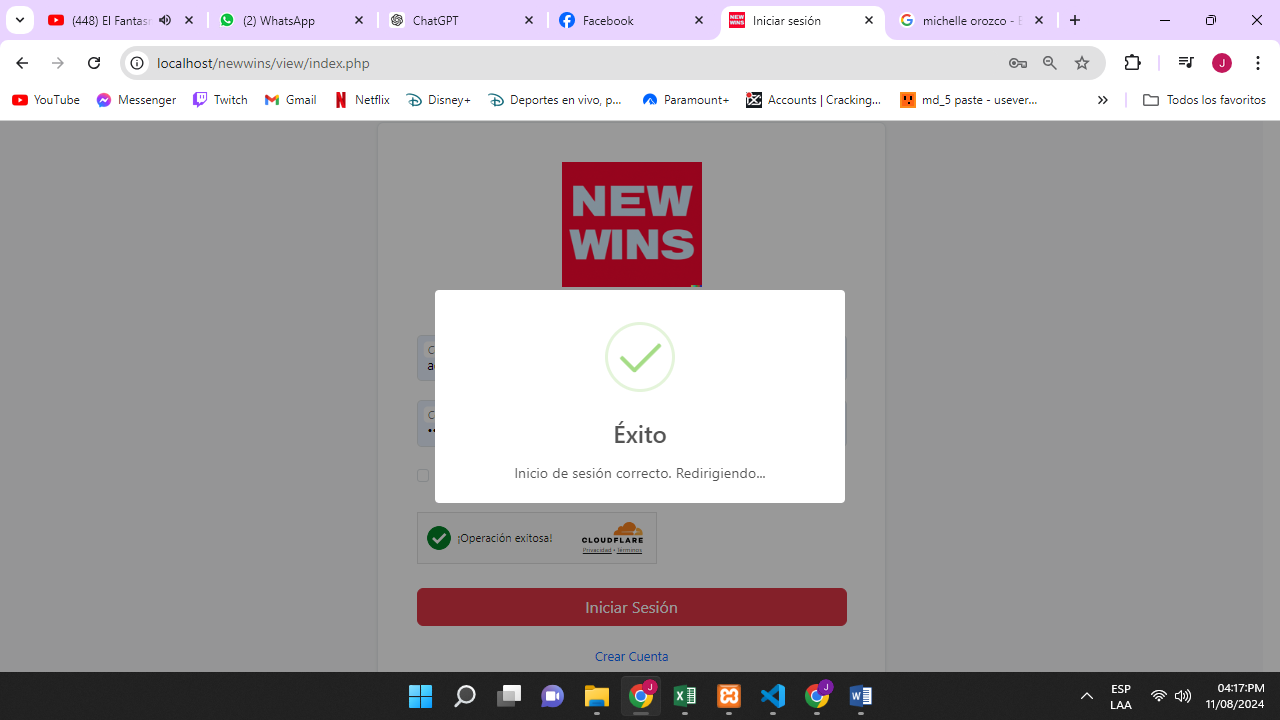
        } else {

            $stmt->close(); // Cerrar la declaración

            return ['success' => false, 'message' => 'Usuario no encontrado.']; // Usuario no encontrado

        }

    }



**Mostrar noticias:**

public function mostrarNoticias()

    {

        // Obtiene la lista de noticias llamando al método listarNoticias de GestorContenido

        $noticias = $this->gestorContenido->listarNoticias();

        // Itera sobre cada noticia en la lista

        foreach ($noticias as $noticia) {

            $id = $noticia['id']; // ID de la noticia

            $titulo = $noticia['titulo']; // Título de la noticia

            $fecha\_not = $noticia['fecha\_publicacion']; // Fecha de publicación de la noticia

            $imagen = $noticia['url']; // URL de la imagen asociada con la noticia

            // Genera el HTML para mostrar la noticia en una tarjeta de Bootstrap

            echo '<div class="col-md-4 mb-4">';

            echo '  <a href="ver\_noticia.php?id=' . $id . '" style="text-decoration:none">'; // Enlace a la página de detalle de la noticia

            echo '      <div class="card">';

            echo '          <img src="' . $imagen . '" class="card-img-top" alt="' . $titulo . '">'; // Imagen de la noticia

            echo '          <div class="card-body">';

            echo '              <h5 class="card-title">' . $titulo . '</h5>'; // Título de la noticia

            echo '              <p class="card-text"><span class="fecha-noticia" data-fecha="' . $fecha\_not . '"></span></p>'; // Fecha de la noticia (con clase para manipulación de JavaScript)

            echo '          </div>';

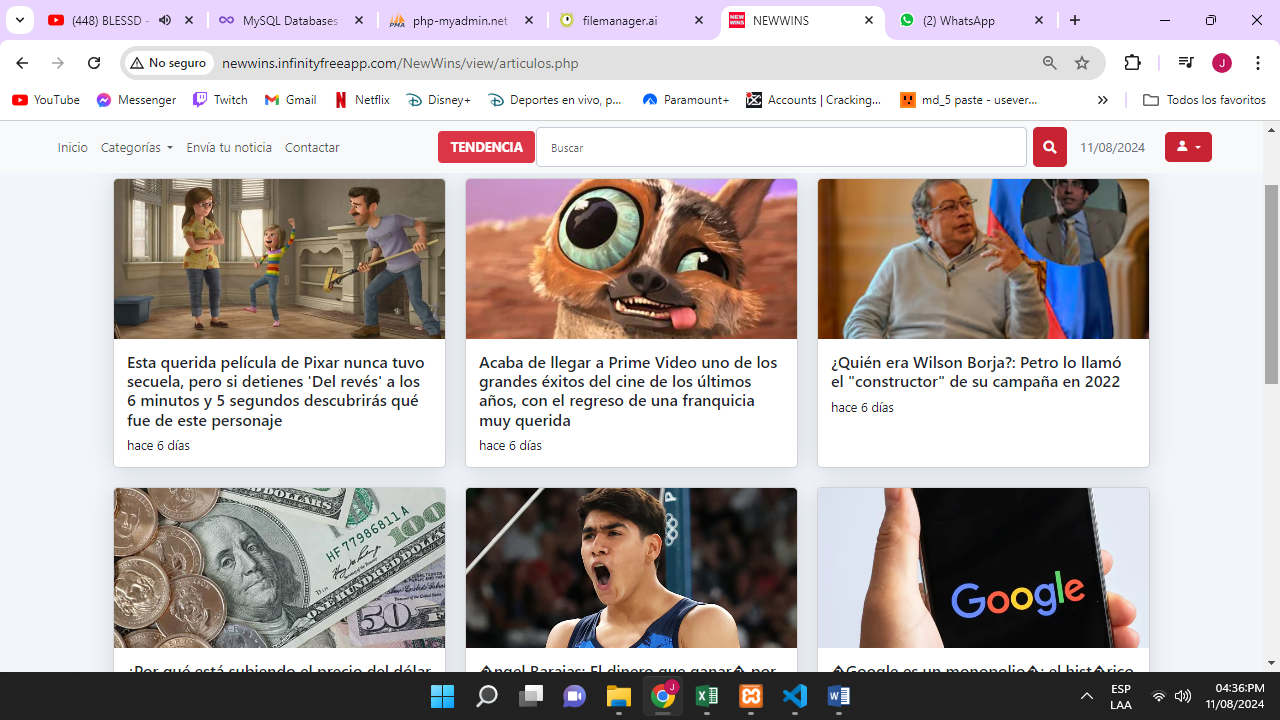
            echo '      </div>';

            echo '  </a>';

            echo '</div>';

        }

    }



**Filtrar por categoría:**

public function listarArticulosPorCategoria($categoria\_id)

{

    // Consulta SQL para obtener todos los artículos de una categoría específica

    $sql = "SELECT \* FROM articulos WHERE categoria\_id = ?";

    // Preparar la consulta SQL

    $stmt = $this->conn->prepare($sql);

    // Vincular el parámetro de ID de categoría a la consulta preparada

    $stmt->bind\_param("i", $categoria\_id);

    // Ejecutar la consulta

    $stmt->execute();

    $result = $stmt->get\_result();

    // Recopilar los artículos en un array

    $articulos = [];

    while ($row = $result->fetch\_assoc()) {

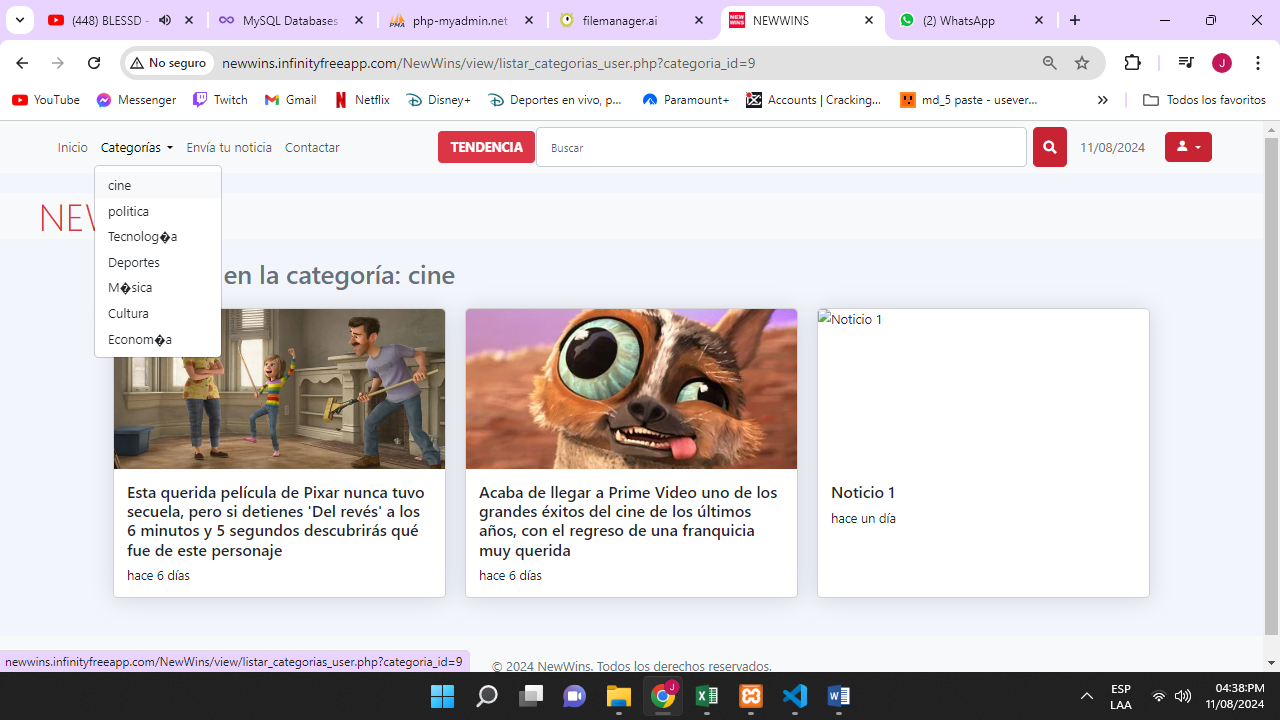
        $articulos[] = $row;

    }

    // Devolver el array de artículos

    return $articulos;

}



**Enviar Noticia a ADMIN:**

public function enviarNoticia($usuario\_id, $titulo, $contenido, $categoria\_id, $url)

{

    // Consulta SQL para insertar una noticia en la bandeja de entrada

    $sql = "INSERT INTO bandeja\_entrada (usuario\_id, titulo, contenido, categoria\_id, url, fecha\_envio) VALUES (?, ?, ?, ?, ?, NOW())";

    // Preparar la consulta SQL

    $stmt = $this->conn->prepare($sql);

    if ($stmt === false) {

        return false; // Devolver false si hubo un error preparando la consulta

    }

    // Vincular los parámetros de la noticia a la consulta preparada

    $stmt->bind\_param("issis", $usuario\_id, $titulo, $contenido, $categoria\_id, $url);

    // Ejecutar la consulta y devolver true si fue exitosa, false en caso contrario

    if ($stmt->execute()) {

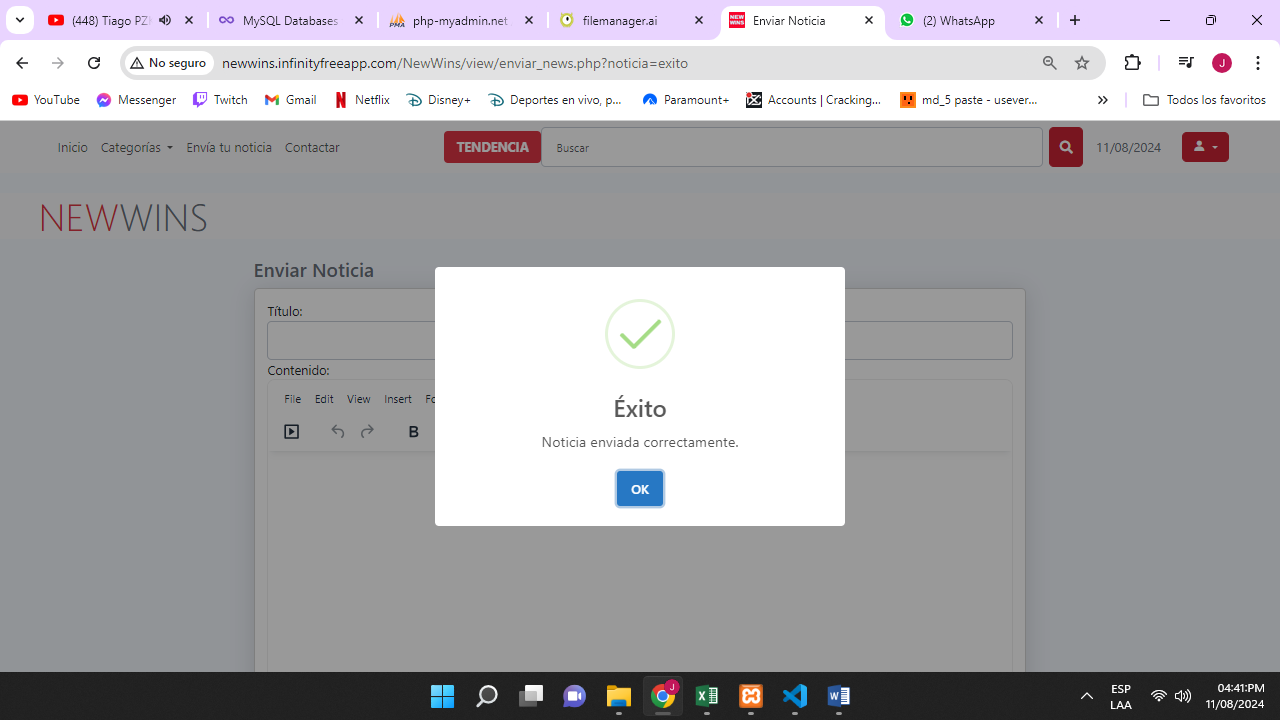
        return true;

    } else {

        return false;

    }

}



**Mostrar Tendencias:**

public function obtenerNoticiasTendencia()

{

    // Consulta SQL para obtener las noticias más populares basadas en el número de 'likes'

    $sql = "SELECT a.id, a.titulo, a.contenido, a.url, a.fecha\_publicacion

            FROM articulos a

            INNER JOIN (

                SELECT articulo\_id, COUNT(\*) AS total\_likes

                FROM valoraciones\_articulos

                WHERE valoracion = 'like'

                GROUP BY articulo\_id

            ) v ON a.id = v.articulo\_id

            ORDER BY v.total\_likes DESC

            LIMIT 10"; // Ajusta el límite según tus necesidades

    $resultado = $this->conn->query($sql);

    if ($resultado) {

        // Convertir el resultado en un array asociativo

        $noticiasArray = [];

        while ($row = $resultado->fetch\_assoc()) {

            $noticiasArray[] = $row;

        }

        return $noticiasArray;

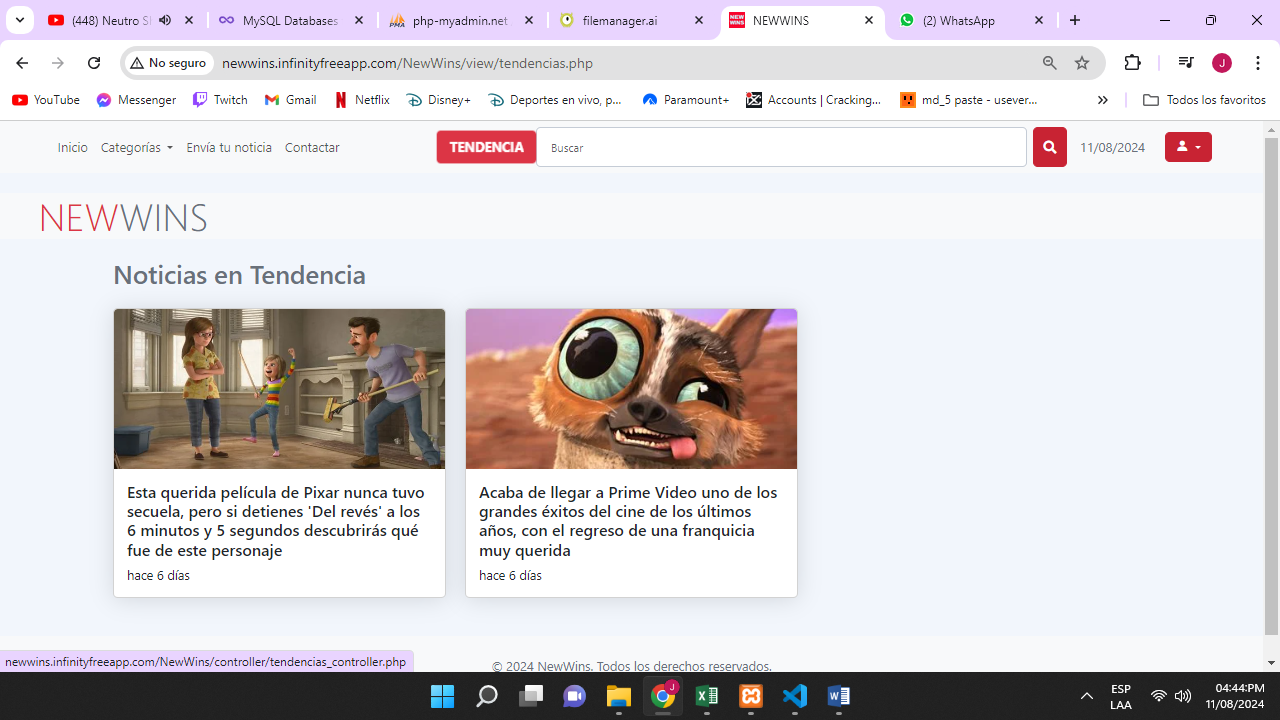
    } else {

        // Retornar un array vacío en caso de error

        return [];

    }

}



**Buscar noticias:**

public function buscarNoticias($termino)

{

    // Dividir el término de búsqueda en palabras individuales

    $terminos = explode(' ', $termino);

    // Iniciar la consulta SQL con la base de la consulta

    $sql = "SELECT \* FROM articulos WHERE ";

    // Inicializar arrays para los parámetros y tipos

    $params = [];

    $types = '';

    // Construir la consulta SQL dinámica para buscar en el título y el contenido

    foreach ($terminos as $index => $palabra) {

        if ($index > 0) {

            $sql .= " OR ";

        }

        $sql .= "(titulo LIKE ? OR contenido LIKE ?)";

        // Agregar los parámetros con comodines '%' para búsqueda parcial

        $params[] = "%" . $palabra . "%";

        $params[] = "%" . $palabra . "%";

        // Agregar el tipo de parámetro 'ss' para cada par de parámetros de tipo string

        $types .= 'ss';

    }

    // Preparar la consulta SQL

    $stmt = $this->conn->prepare($sql);

    // Vincular los parámetros de búsqueda a la consulta preparada

    $stmt->bind\_param($types, ...$params);

    // Ejecutar la consulta

    $stmt->execute();

    $result = $stmt->get\_result();

    // Recopilar los resultados en un array

    $noticias = [];

    while ($row = $result->fetch\_assoc()) {

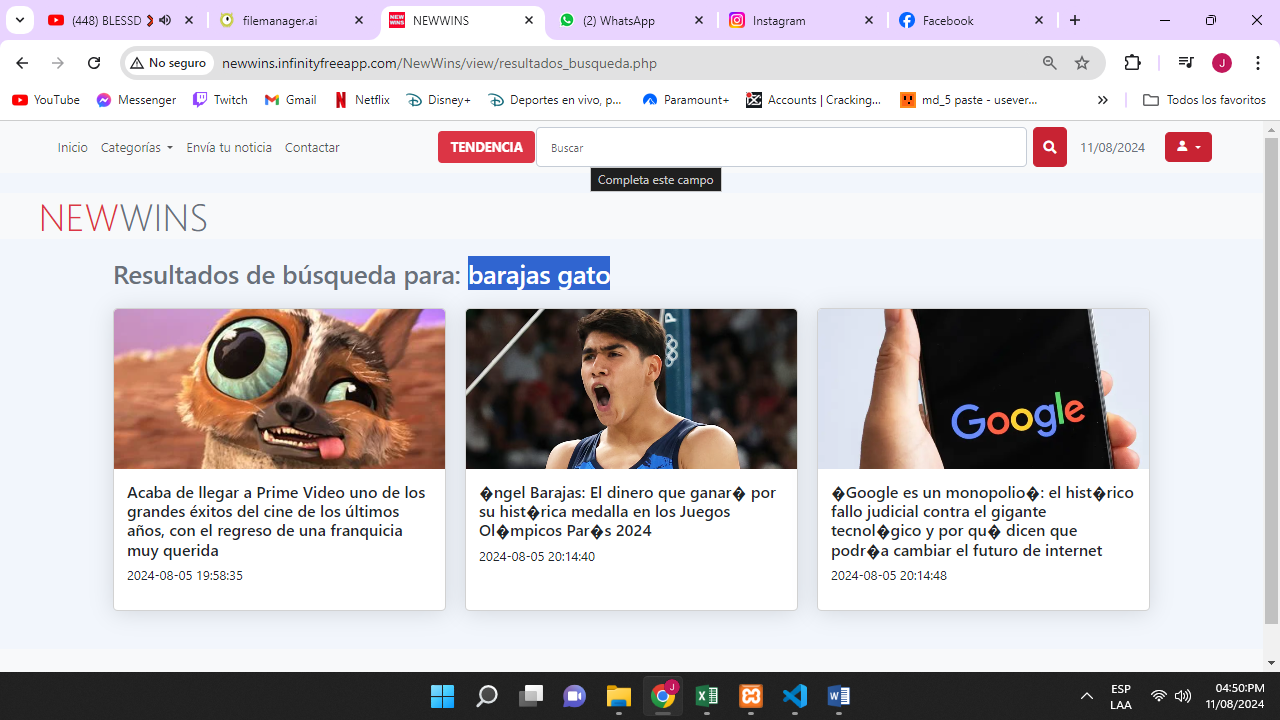
        $noticias[] = $row;

    }

    // Devolver el array de noticias encontradas

    return $noticias;

}



**Dar valoraciones:**

public function agregarValoracion($articulo\_id, $usuario\_id, $valoracion)

{

    // Verificar si ya existe una valoración del artículo por el usuario

    $valoracionExistente = $this->obtenerValoracionExistente($articulo\_id, $usuario\_id);

    if ($valoracionExistente !== null) {

        // Actualizar la valoración existente si ya existe

        return $this->actualizarValoracion($articulo\_id, $usuario\_id, $valoracion);

    } else {

        // Insertar una nueva valoración si no existe

        $sql = "INSERT INTO valoraciones\_articulos (articulo\_id, usuario\_id, valoracion) VALUES (?, ?, ?)";

        $stmt = $this->conn->prepare($sql);

        $stmt->bind\_param('iis', $articulo\_id, $usuario\_id, $valoracion);

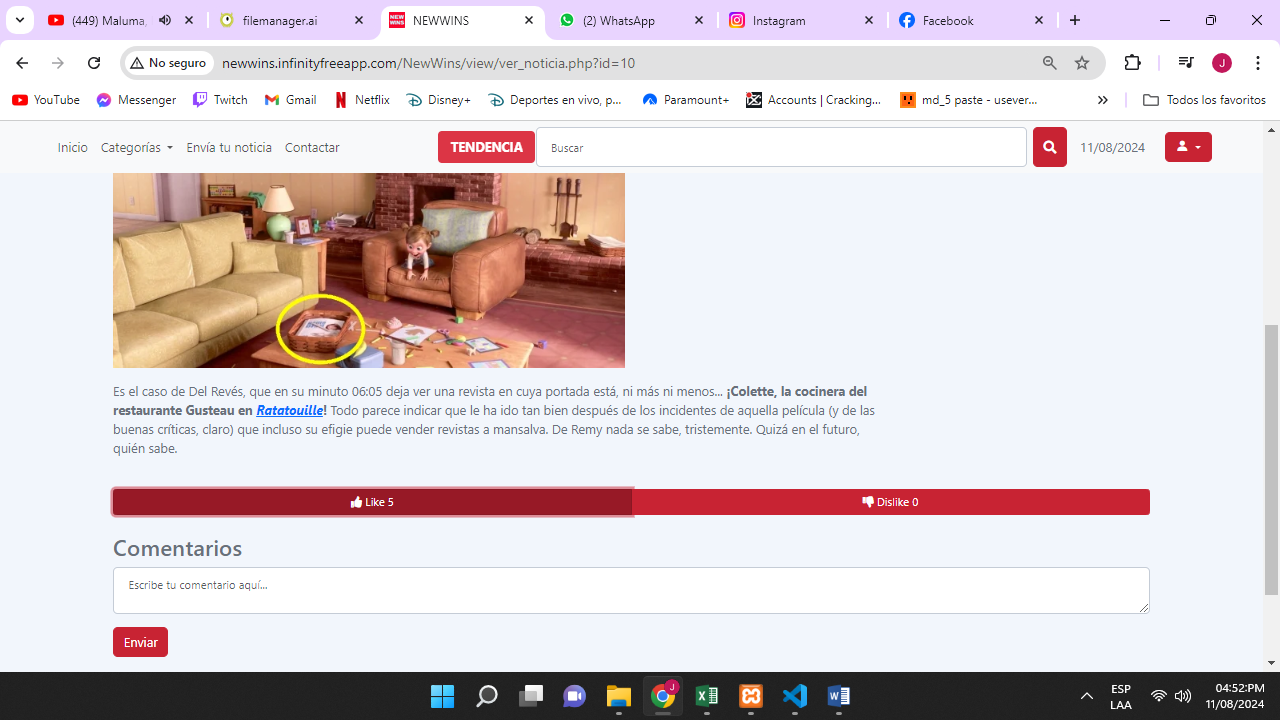
        $stmt->execute();

        // Devolver true si se insertó la valoración

        return $stmt->affected\_rows > 0;

    }

}



**Hacer comentario:**

public function agregarComentario($usuario, $texto, $puntuacion, $articulo\_id)

{

    // Consulta SQL para insertar un nuevo comentario

    $sql = "INSERT INTO comentarios (usuario, texto, puntuacion, articulo\_id) VALUES (?, ?, ?, ?)";

    // Preparar la consulta SQL

    $stmt = $this->conn->prepare($sql);

    // Vincular los parámetros del comentario a la consulta preparada

    $stmt->bind\_param("ssii", $usuario, $texto, $puntuacion, $articulo\_id);

    // Ejecutar la consulta y manejar posibles errores

    if (!$stmt->execute()) {

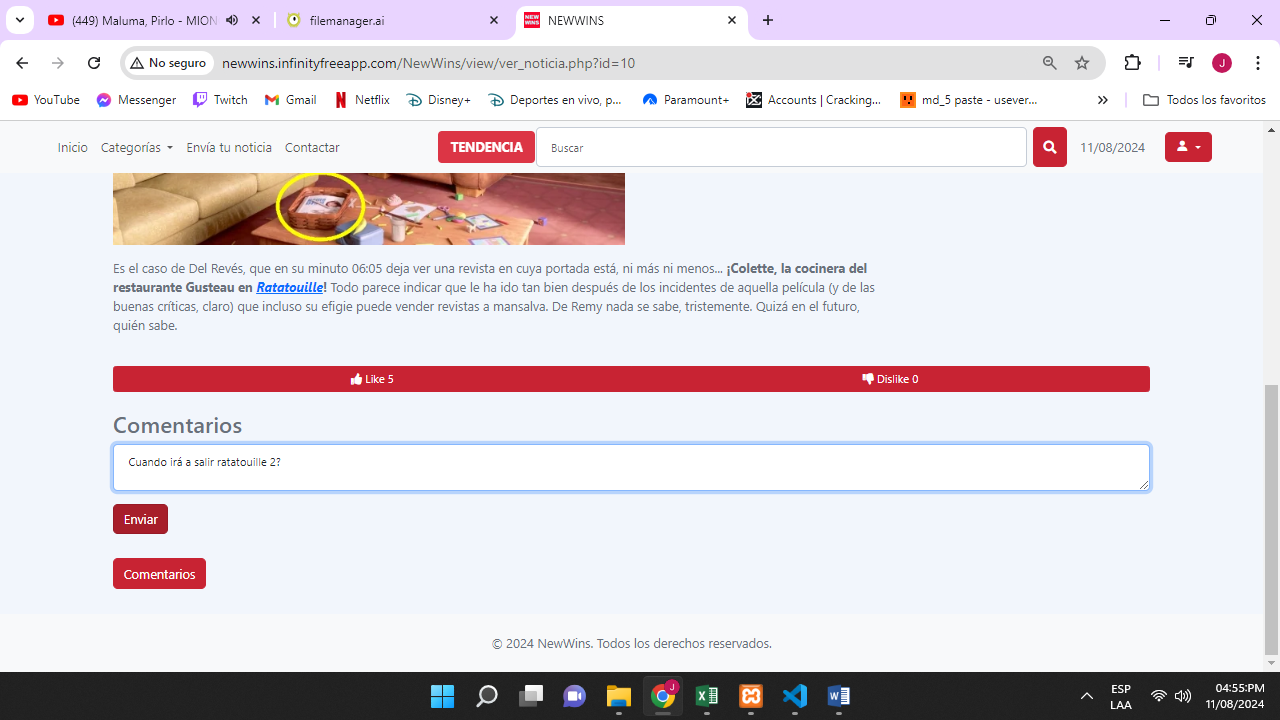
        throw new Exception("Error al agregar comentario: " . $stmt->error);

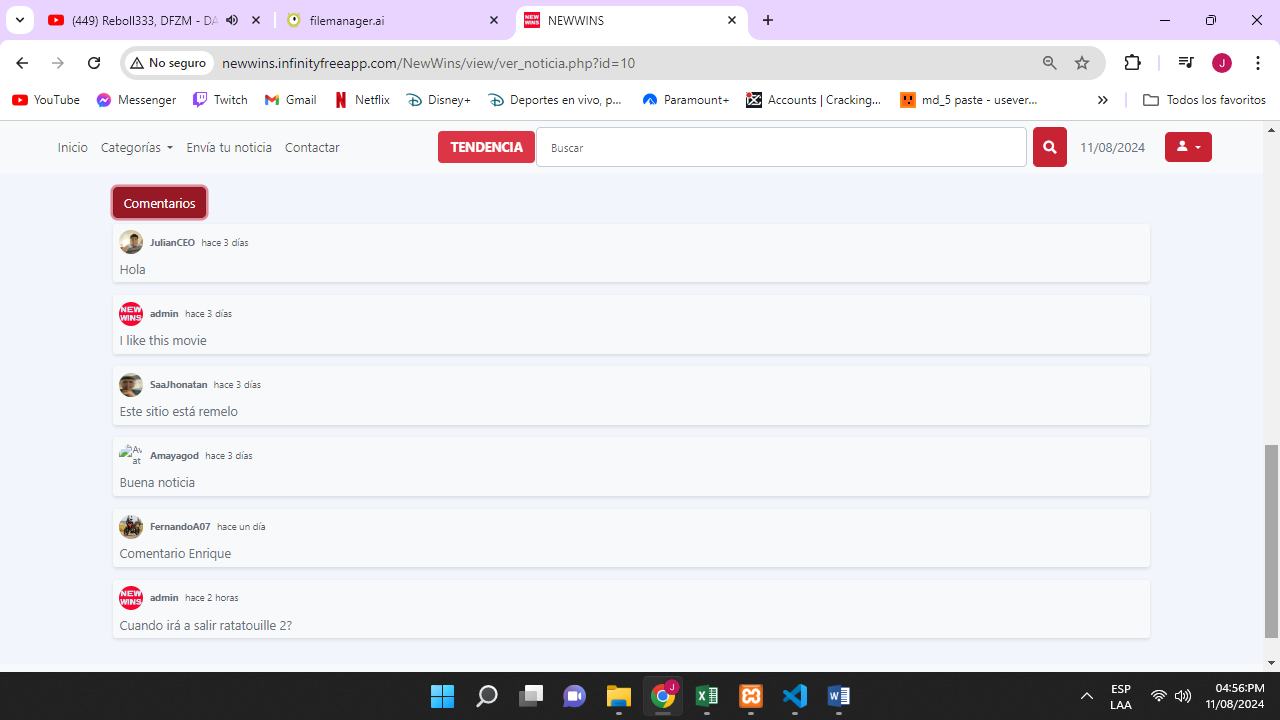
    }

    // Cerrar la declaración

    $stmt->close();

}





            $articulos[] = $row;

**Interfaz de usuario gráfica, Texto, Aplicación, Chat o mensaje de texto

Descripción generada automáticamente**

**Actualizar perfil usuario:**

public static function updateUser($currentEmail, $username, $firstName, $lastName, $location, $email)

{

    // Crear conexión a la base de datos

    $conn = ConexionBD::obtenerConexion();

    // Verificar la conexión a la base de datos

    if ($conn->connect\_error) {

        throw new Exception("Error de conexión: " . $conn->connect\_error);

    }

    // Preparar la consulta SQL para actualizar el usuario

    $stmt = $conn->prepare("UPDATE usuarios\_registrados SET nombre\_usuario = ?, nombre = ?, apellido = ?, ubicacion = ?, correo\_electronico = ? WHERE correo\_electronico = ?");

    // Enlazar los parámetros a la consulta preparada

    $stmt->bind\_param("ssssss", $username, $firstName, $lastName, $location, $email, $currentEmail);

    // Ejecutar la consulta

    $stmt->execute();

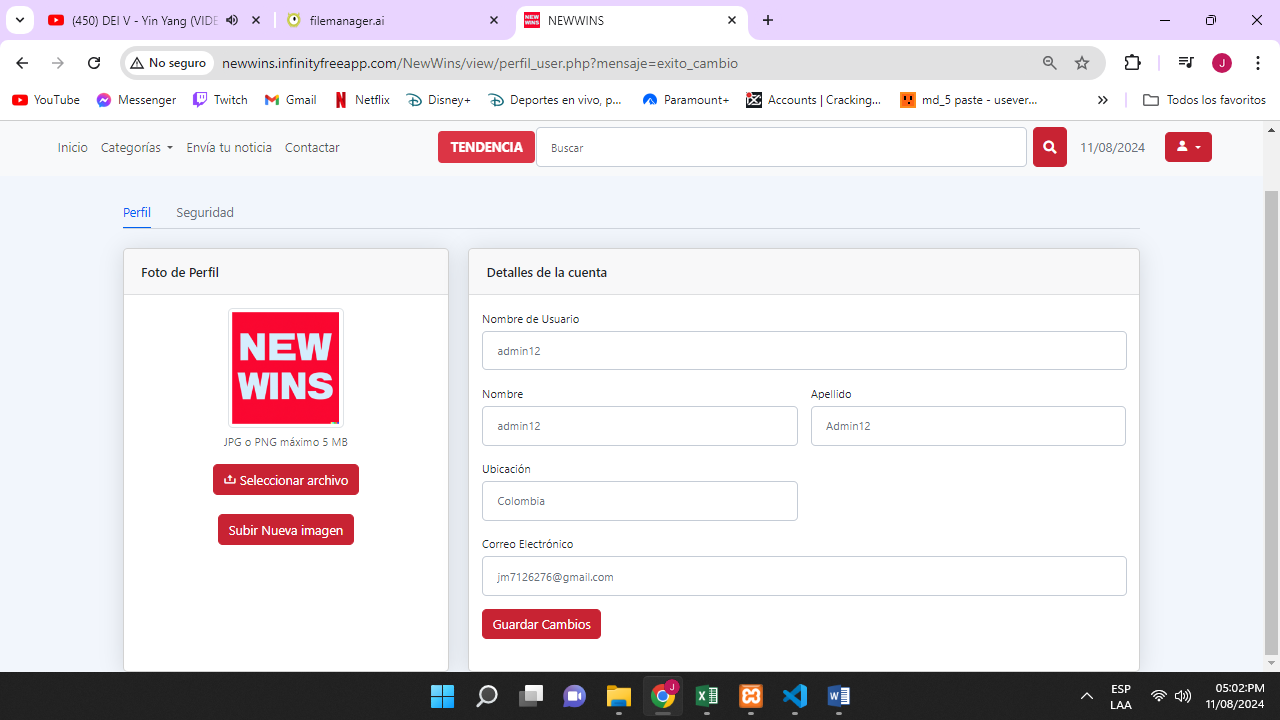
    // Cerrar la declaración preparada

    $stmt->close();

    // Cerrar la conexión a la base de datos

    $conn->close();

}



**Subir foto de perfil:**

public function subirFotoPerfil($userEmail, $file)

{

    // Verificar si hay algún error en el archivo subido

    if ($file['error'] !== UPLOAD\_ERR\_OK) {

        return 'Error al subir el archivo.';

    }

    // Verificar el tamaño del archivo (máximo 5MB)

    if ($file['size'] > 5242880) {

        return 'El archivo es demasiado grande. El tamaño máximo permitido es de 5MB.';

    }

    // Verificar el tipo de archivo

    $allowedTypes = ['image/jpeg', 'image/png', 'image/gif'];

    if (!in\_array($file['type'], $allowedTypes)) {

        return 'Tipo de archivo no permitido. Solo se permiten archivos JPG, PNG y GIF.';

    }

    // Mover el archivo subido a la carpeta de destino

    $uploadDir = '../uploads/perfiles/';

    $fileName = basename($file['name']);

    $uploadFilePath = $uploadDir . $fileName;

    if (!move\_uploaded\_file($file['tmp\_name'], $uploadFilePath)) {

        return 'Error al mover el archivo subido.';

    }

    // Actualizar la base de datos con la nueva ruta de la foto de perfil

    $sql = "UPDATE usuarios\_registrados SET foto\_perfil = ? WHERE correo\_electronico = ?";

    $stmt = $this->conn->prepare($sql);

    $stmt->bind\_param('ss', $uploadFilePath, $userEmail);

    if ($stmt->execute()) {

        return true; // Subida exitosa

    } else {

        return 'Error al actualizar la base de datos.';

    }

}

**Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente**

**Cambiar contraseña:**

public function cambiarContrasena($correo, $currentPassword, $newPassword)

{

    // Obtener la contraseña actual del usuario desde la base de datos

    $conn = ConexionBD::obtenerConexion();

    $stmt = $this->conn->prepare("SELECT contrasena FROM usuarios\_registrados WHERE correo\_electronico = ?");

    $stmt->bind\_param("s", $correo);

    $stmt->execute();

    $stmt->bind\_result($hash\_contrasena);

    $stmt->fetch();

    $stmt->close();

    // Verificar si la contraseña actual proporcionada coincide con la almacenada en la base de datos

    if (password\_verify($currentPassword, $hash\_contrasena)) {

        // Encriptar la nueva contraseña

        $newHashPassword = password\_hash($newPassword, PASSWORD\_DEFAULT);

        // Actualizar la contraseña en la base de datos con la nueva encriptada

        $stmt = $this->conn->prepare("UPDATE usuarios\_registrados SET contrasena = ? WHERE correo\_electronico = ?");

        $stmt->bind\_param("ss", $newHashPassword, $correo);

        $stmt->execute();

        $stmt->close();

        return true; // Contraseña cambiada exitosamente

    } else {

        return false; // La contraseña actual es incorrecta

    }

}

**Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente**

**Eliminar cuenta:**

public function eliminarUsuario($idUsuario) {

    // Consulta SQL que llama al procedimiento almacenado para eliminar un usuario

    $sql = "CALL EliminarUsuario(?)";

    // Preparar la consulta SQL para ejecutar con parámetros

    $stmt = $this->conn->prepare($sql);

    // Enlazar el parámetro $idUsuario a la consulta preparada

    $stmt->bind\_param("i", $idUsuario);

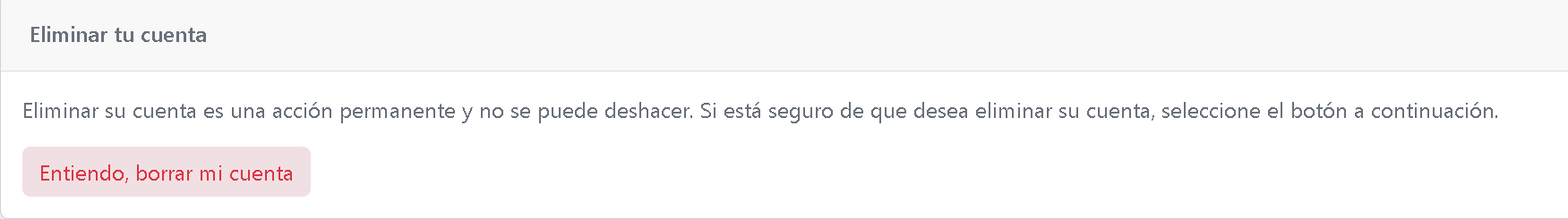
    // Ejecutar la consulta

    $stmt->execute();

    // Retornar verdadero si al menos una fila fue afectada, indicando que la eliminación fue exitosa

    return $stmt->affected\_rows > 0;

}

****

**Cerrar sesión:**

<?php

// Iniciar la sesión si no está iniciada

if (!isset($\_SESSION)) {

    session\_start(); // Inicia una nueva sesión o reanuda la existente

}

// Destruir todas las variables de sesión

$\_SESSION = array(); // Limpia todas las variables de sesión

// Destruir la sesión actual

session\_destroy(); // Elimina la sesión actual del servidor

// Redirigir al usuario a la página de inicio de sesión o a donde desees

header("Location: ../view/index.php"); // Redirige al usuario a la página de inicio

exit(); // Termina la ejecución del script para evitar cualquier código adicional

?>

