**Pruebas unitarias y de integración**

**Parte del ADMIN**

**Iniciar sesión Admin:**

public function iniciarSesionAdmin($correo, $contrasena)

    {

        // Consulta SQL para obtener la contraseña y el rol de administrador

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

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

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

        $stmt->bind\_result($hash\_contrasena, $es\_admin); // Obtener los resultados

        $stmt->fetch(); // Obtener el valor de las variables

        if (password\_verify($contrasena, $hash\_contrasena) && $es\_admin == 1) { // Verificar la contraseña y el rol de administrador

            return true; // Inicio de sesión exitoso para administrador

        } else {

            return false; // Credenciales incorrectas o no es administrador

        }

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

    }

**Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente**

**Subir noticia:**

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

{

    // Crea un objeto DateTime con la zona horaria de Bogotá

    $fecha = new DateTime('now', new DateTimeZone('America/Bogota'));

    // Formatea la fecha y hora para que se ajuste al formato de la base de datos

    $fecha\_publicacion = $fecha->format('Y-m-d H:i:s');

    // Verifica que todos los parámetros necesarios no estén vacíos

    if (!empty($titulo) && !empty($contenido) && !empty($url) && !empty($categoria\_id)) {

        // Consulta SQL para insertar una nueva noticia en la tabla 'articulos'

        $sql = "INSERT INTO articulos (titulo, fecha\_publicacion, contenido, url, categoria\_id) VALUES (?, ?, ?, ?, ?)";

        // Preparar la consulta SQL

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

        // Vincular los parámetros a la consulta preparada

        $stmt->bind\_param("ssssi", $titulo, $fecha\_publicacion, $contenido, $url, $categoria\_id);

        // Ejecutar la consulta y verificar si la inserción fue exitosa

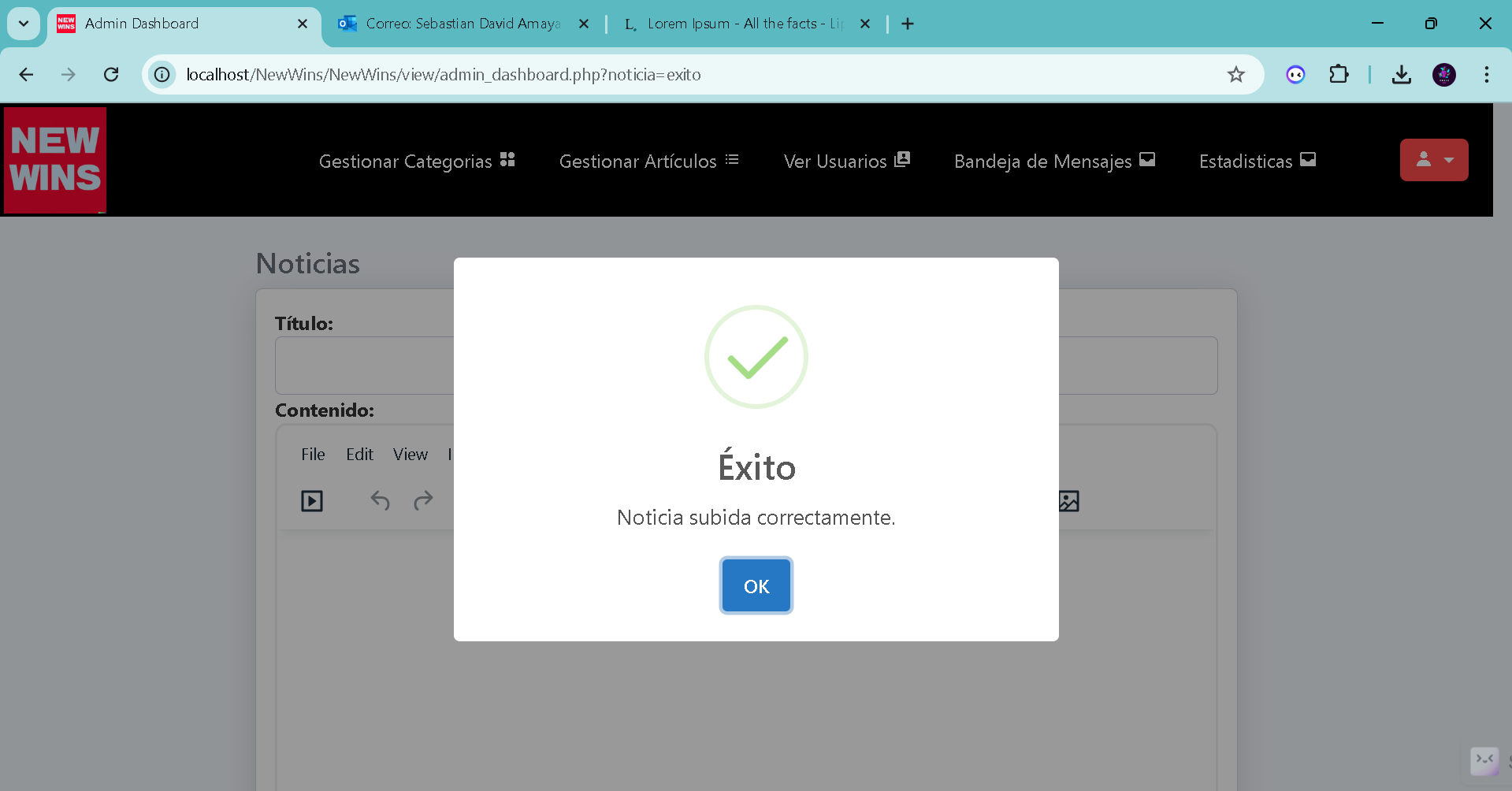
        return $stmt->execute();

    } else {

        return false; // Devolver false si faltan datos

    }

}



**Crear categoría:**

public function crearCategoria($nombre, $descripcion, $imagen)

{

    // Verificar que el nombre y la descripción no estén vacíos

    if (!empty($nombre) && !empty($descripcion)) {

        // Consulta SQL para insertar una nueva categoría

        $stmt = $this->conn->prepare("INSERT INTO categorias (nombre, descripcion, imagen) VALUES (?, ?, ?)");

        // Vincular los parámetros a la consulta preparada

        $stmt->bind\_param("sss", $nombre, $descripcion, $imagen);

        // Ejecutar la consulta y verificar si la inserción fue exitosa

        return $stmt->execute();

    } else {

        return false; // Devolver false si faltan datos

    }

}

**Interfaz de usuario gráfica, Aplicación, PowerPoint

Descripción generada automáticamente**

**Gestionar categorías:**

public function listarCategorias()

{

    // Consulta SQL para seleccionar todas las categorías

    $sql = "SELECT \* FROM categorias";

    // Ejecutar la consulta SQL

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

    // Devolver el resultado de la consulta

    return $result;

}

**Captura de pantalla de un celular

Descripción generada automáticamente**

**Eliminar categoría:**

public function eliminarCategoria($id)

{

    // Iniciar una transacción para asegurar la integridad de los datos

    $this->conn->begin\_transaction();

    try {

        // Primero, eliminar todos los artículos relacionados con la categoría

        $sql\_articulos = "DELETE FROM articulos WHERE categoria\_id = ?";

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

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

        $stmt\_articulos->execute();

        // Luego, eliminar la categoría

        $sql\_categoria = "DELETE FROM categorias WHERE id = ?";

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

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

        $stmt\_categoria->execute();

        // Si ambas operaciones fueron exitosas, confirmar la transacción

        $this->conn->commit();

        return true;

    } catch (Exception $e) {

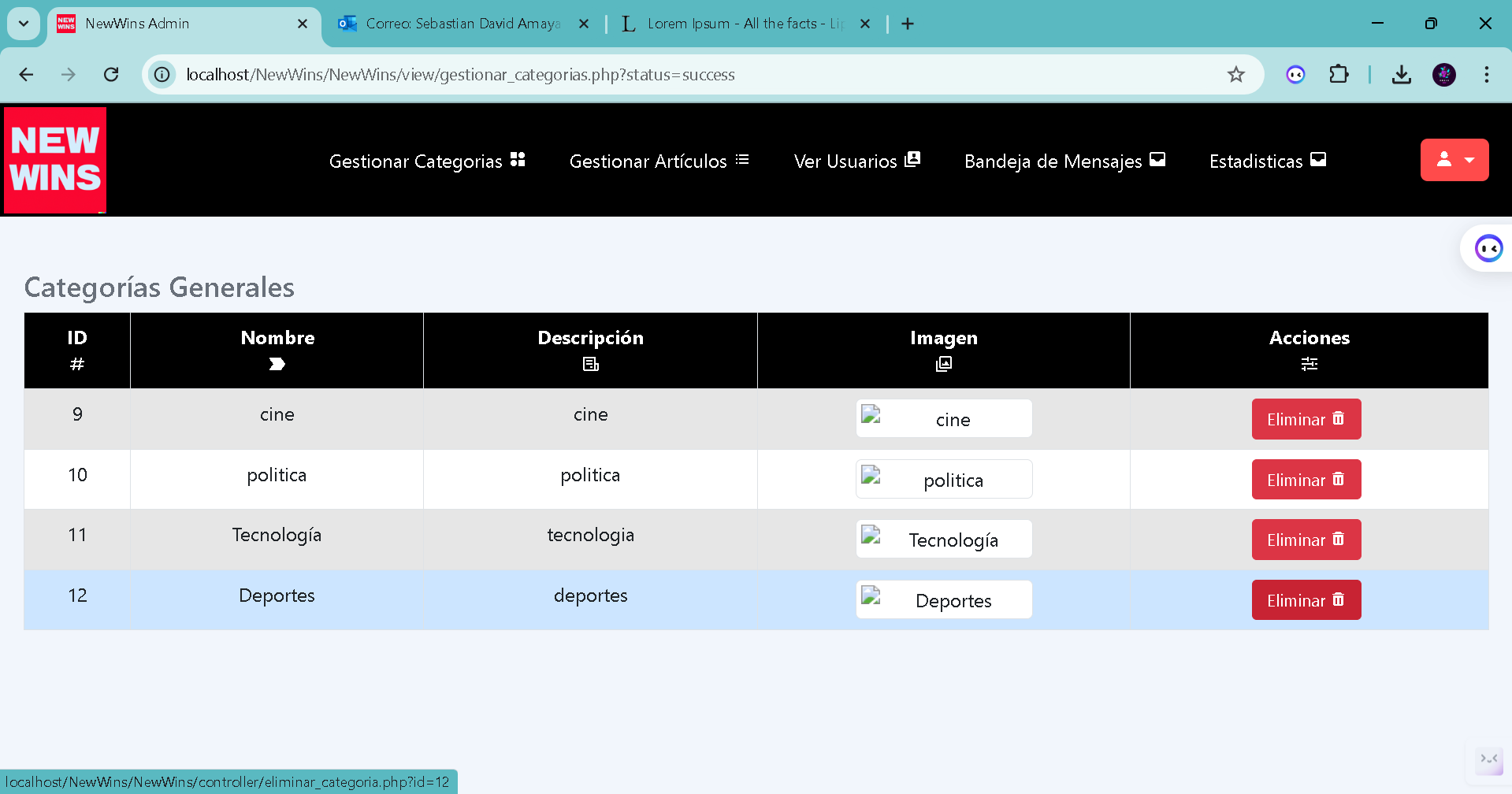
        // Si hubo un error, revertir la transacción

        $this->conn->rollback();

        return false;

    }

}

****

**Gestionar artículos:**

public function listarNoticias()

{

    // Consulta SQL para seleccionar noticias y sus categorías

    $sql = "SELECT a.id, c.nombre AS categoria, a.titulo, a.contenido, a.url, a.fecha\_publicacion

            FROM articulos a

            JOIN categorias c ON a.categoria\_id = c.id";

    // Ejecutar la consulta SQL

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

    // Devolver el resultado de la consulta

    return $result;

}

**Texto

Descripción generada automáticamente con confianza media**

**Eliminar noticia:**

public function eliminarNoticia($id)

{

    // Consulta SQL para eliminar una noticia por su ID

    $sql = "DELETE FROM articulos WHERE id = ?";

    // Preparar la consulta SQL

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

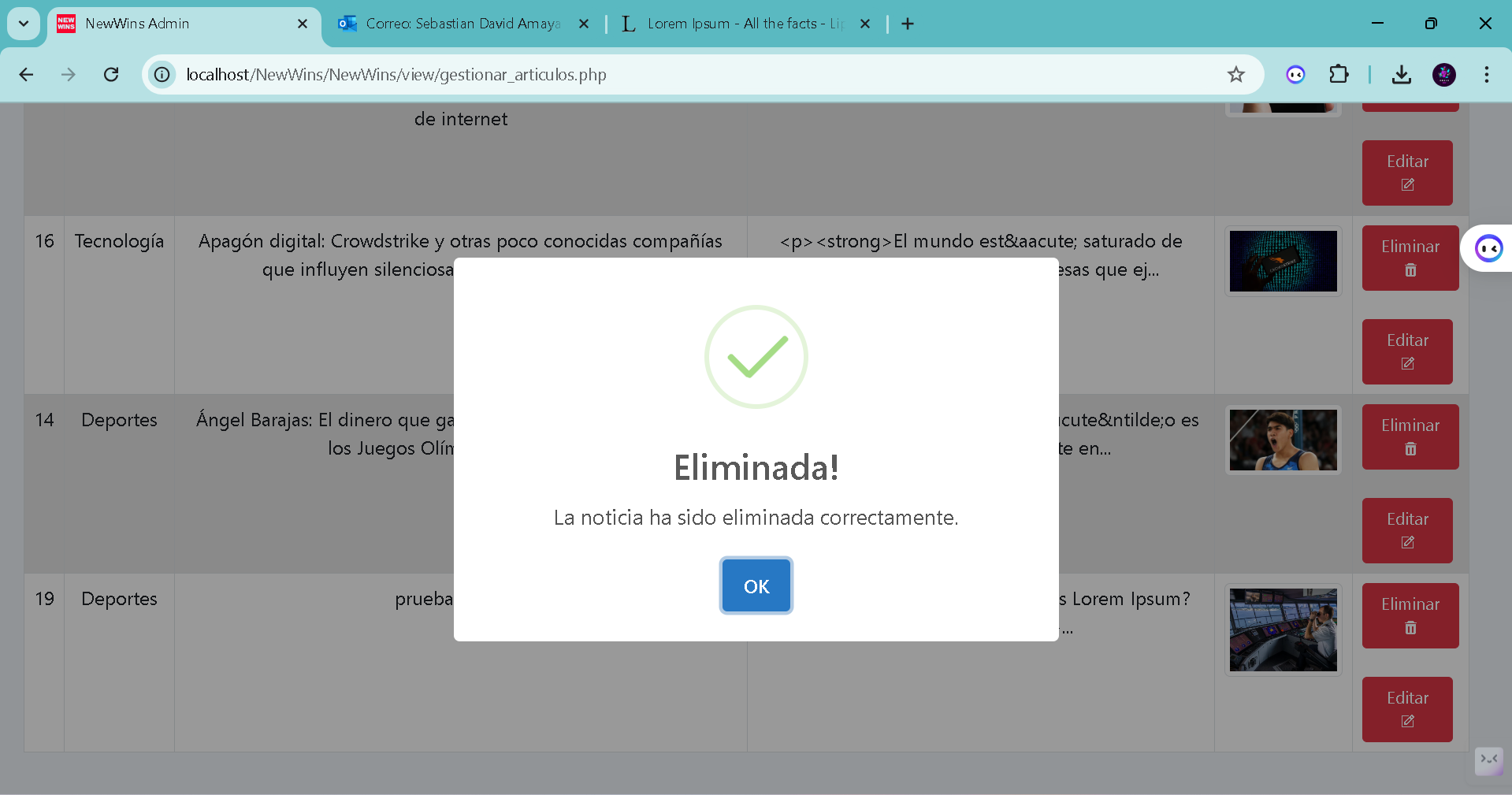
    // Vincular el parámetro de ID a la consulta preparada

    $stmt->bind\_param("i", $id);

    // Ejecutar la consulta y verificar si la eliminación fue exitosa

    return $stmt->execute();

}

****

**Editar noticia:**

public function editarNoticia($id, $titulo, $contenido, $url, $categoria\_id)

{

    // Consulta SQL para actualizar una noticia existente en la base de datos

    $sql = "UPDATE articulos SET titulo = ?, contenido = ?, url = ?, categoria\_id = ? WHERE id = ?";

    // Preparar la consulta SQL

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

    // Vincular los parámetros a la consulta preparada

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

    // Ejecutar la consulta y verificar si la actualización fue exitosa

    return $stmt->execute();

}

**Interfaz de usuario gráfica, Aplicación, Sitio web

Descripción generada automáticamente**

**Listar usuarios:**

    public function listarUsuarios()

{

    // Consulta SQL para obtener el ID, nombre de usuario y correo electrónico de todos los usuarios

    $sql = "SELECT id, nombre\_usuario, correo\_electronico, es\_admin FROM usuarios\_registrados";

    // Ejecutar la consulta SQL

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

    // Crear un array para almacenar los usuarios

    $usuarios = array();

    // Verificar si la consulta devuelve resultados

    if ($result->num\_rows > 0) {

        // Recorrer los resultados de la consulta

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

            // Añadir cada usuario al array

            $usuarios[] = $row;

        }

    }

    // Retornar el array de usuarios

    return $usuarios;

}

**Tabla

Descripción generada automáticamente**

**Quitar Admin y dar Admin:**

public function darAdmin($idUsuario) {

    // Consulta SQL que llama al procedimiento almacenado para asignar el rol de administrador

    $sql = "SELECT darAdmin(?) AS result";

    // 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();

    // Obtener el resultado de la consulta ejecutada y almacenarlo en un array asociativo

    $result = $stmt->get\_result()->fetch\_assoc();

    // Retornar verdadero si el resultado es mayor a 0, indicando éxito; falso en caso contrario

    return $result['result'] > 0;

}

public function quitarAdmin($idUsuario) {

    // Consulta SQL que llama al procedimiento almacenado para revocar el rol de administrador

    $sql = "SELECT quitarAdmin(?) AS result";

    // 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();

    // Obtener el resultado de la consulta ejecutada y almacenarlo en un array asociativo

    $result = $stmt->get\_result()->fetch\_assoc();

    // Retornar verdadero si el resultado es mayor a 0, indicando éxito; falso en caso contrario

    return $result['result'] > 0;

}

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

Descripción generada automáticamente**

**Listar bandeja de entrada:**

public function obtenerArticulos()

    {

        // Consulta SQL para obtener los artículos y los nombres de usuario asociados

        $sql = "SELECT b.id, u.nombre\_usuario, b.fecha\_envio, b.url, b.titulo, b.contenido

                FROM bandeja\_entrada b

                JOIN usuarios\_registrados u ON b.usuario\_id = u.id";

        // Ejecutar la consulta SQL

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

        // Verificar si hubo un error en la consulta SQL

        if ($result === false) {

            die("Error en la consulta SQL: " . $this->conn->error);

        }

        // Crear un array para almacenar los artículos

        $articulos = [];

        // Recorrer los resultados de la consulta y agregar cada fila al array de artículos

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

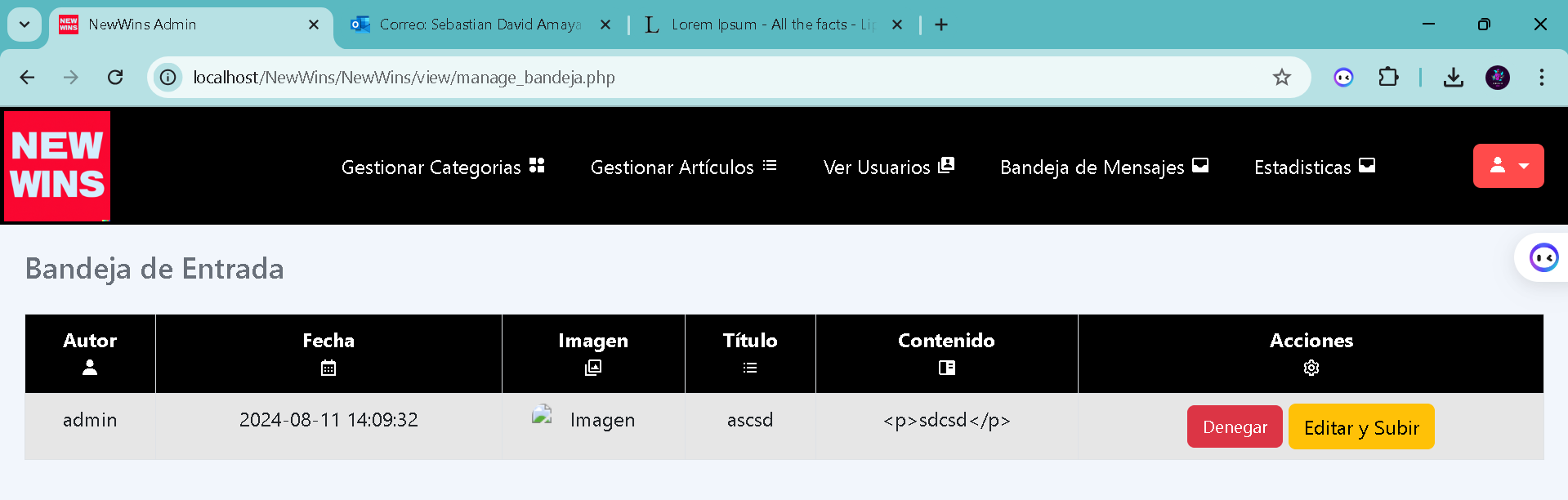
            $articulos[] = $row;

        }

        // Devolver el array de artículos

        return $articulos;

    }

****

**Editar y subir noticia desde la bandeja:**

public function subirNoticiaBandeja($id, $titulo, $contenido, $url, $categoria\_id)

{

    // Obtener los detalles del artículo desde la bandeja de entrada

    $sql\_select = "SELECT usuario\_id, titulo, contenido, url, categoria\_id FROM bandeja\_entrada WHERE id = ?";

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

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

    $stmt\_select->execute();

    $resultado = $stmt\_select->get\_result()->fetch\_assoc();

    $stmt\_select->close();

    if (!$resultado) {

        // Si no se encuentra el artículo en la bandeja de entrada, devolver false

        return false;

    }

    // Insertar el artículo en la tabla de artículos

    $sql\_insert = "INSERT INTO articulos (titulo, fecha\_publicacion, contenido, url, categoria\_id) VALUES (?, NOW(), ?, ?, ?)";

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

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

    $exito = $stmt\_insert->execute();

    $stmt\_insert->close();

    if ($exito) {

        // Si la inserción es exitosa, eliminar el artículo de la bandeja de entrada

        $sql\_delete = "DELETE FROM bandeja\_entrada WHERE id = ?";

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

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

        $stmt\_delete->execute();

        $stmt\_delete->close();

    }

    return $exito;

}

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

Descripción generada automáticamente**

**Eliminar de bandeja:**

public function eliminarDeBandeja($id)

{

    // Consulta SQL para eliminar un artículo de la bandeja de entrada

    $sql = "DELETE FROM bandeja\_entrada WHERE id = ?";

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

    // Vincular el parámetro: id como entero

    $stmt->bind\_param("i", $id);

    // Ejecutar la consulta y devolver el resultado

    return $stmt->execute();

}

****

**Obtener estadísticas por rango de fecha:**

public function obtenerEstadisticasPorRangoFechas($fechaInicio, $fechaFin) {

    // Asegúrate de escapar las fechas para evitar inyecciones SQL

    $fechaInicio = $this->conn->real\_escape\_string($fechaInicio);

    $fechaFin = $this->conn->real\_escape\_string($fechaFin);

    // Consulta SQL para obtener estadísticas basadas en el rango de fechas proporcionado

    $query = "

        SELECT

            (SELECT COUNT(\*) FROM articulos WHERE DATE(fecha\_publicacion) BETWEEN ? AND ?) AS total\_articulos,

            (SELECT COUNT(\*) FROM valoraciones\_articulos WHERE DATE(fecha\_hora) BETWEEN ? AND ?) AS total\_valoraciones,

            (SELECT COUNT(\*) FROM valoraciones\_articulos WHERE valoracion = 'like' AND DATE(fecha\_hora) BETWEEN ? AND ?) AS total\_likes,

            (SELECT COUNT(\*) FROM valoraciones\_articulos WHERE valoracion = 'dislike' AND DATE(fecha\_hora) BETWEEN ? AND ?) AS total\_dislikes,

            (SELECT COUNT(\*) FROM usuarios\_registrados WHERE DATE(fecha\_registro) BETWEEN ? AND ?) AS total\_usuarios,

            (SELECT COUNT(\*) FROM bandeja\_entrada WHERE DATE(fecha\_envio) BETWEEN ? AND ?) AS total\_articulos\_bandeja

    ";

    // Prepara la declaración

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

    if (!$stmt) {

        throw new Exception('Error en la preparación de la consulta: ' . $this->conn->error);

    }

    // Vincula los parámetros

    $stmt->bind\_param('ssssssssssss', $fechaInicio, $fechaFin, $fechaInicio, $fechaFin, $fechaInicio, $fechaFin, $fechaInicio, $fechaFin, $fechaInicio, $fechaFin, $fechaInicio, $fechaFin);

    // Ejecuta la consulta

    $stmt->execute();

    // Obtén el resultado

    $result = $stmt->get\_result();

    $estadisticas = $result->fetch\_assoc();

    // Cierra la declaración

    $stmt->close();

    return $estadisticas;

}

**Tabla

Descripción generada automáticamente**

**Actualizar perfil usuario:**

public function actualizarPerfil($idUsuario, $username, $nombre, $apellido, $ubicacion, $correo)

{

    // Consulta SQL para actualizar los datos del perfil del usuario

    $sql = "UPDATE usuarios\_registrados SET nombre\_usuario = ?, nombre = ?, apellido = ?, ubicacion = ?, correo\_electronico = ? WHERE id = ?";

    // Preparar la consulta SQL

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

    // Verificar si la preparación de la consulta fue exitosa

    if ($stmt === false) {

        die('Error en la preparación de la consulta: ' . $this->conn->error);

    }

    // Enlazar los parámetros a la consulta preparada

    $stmt->bind\_param("sssssi", $username, $nombre, $apellido, $ubicacion, $correo, $idUsuario);

    // Ejecutar la consulta

    if ($stmt->execute()) {

        // Retornar verdadero si la actualización fue exitosa

        return true;

    } else {

        // Retornar falso si hubo un error en la actualización

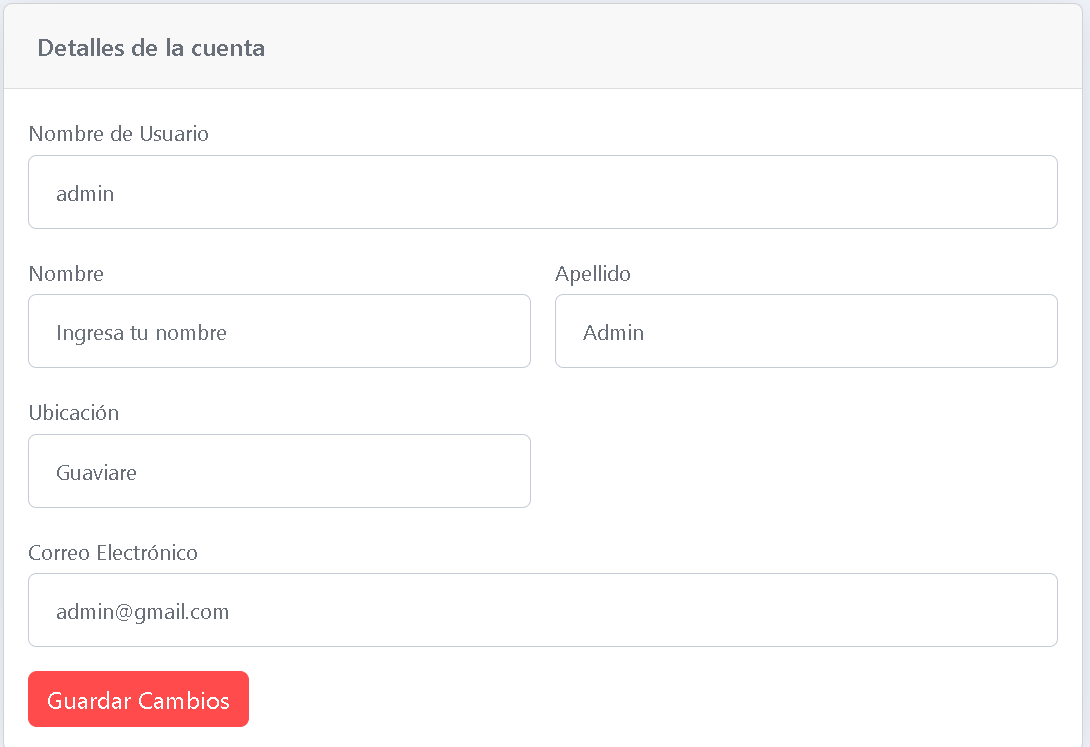
        return false;

    }

    // Cerrar la declaración preparada

    $stmt->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);

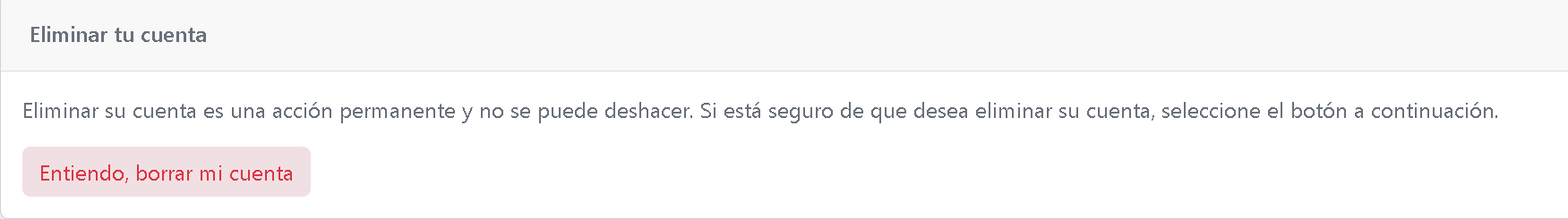
    // Ejecutar la consulta

    $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

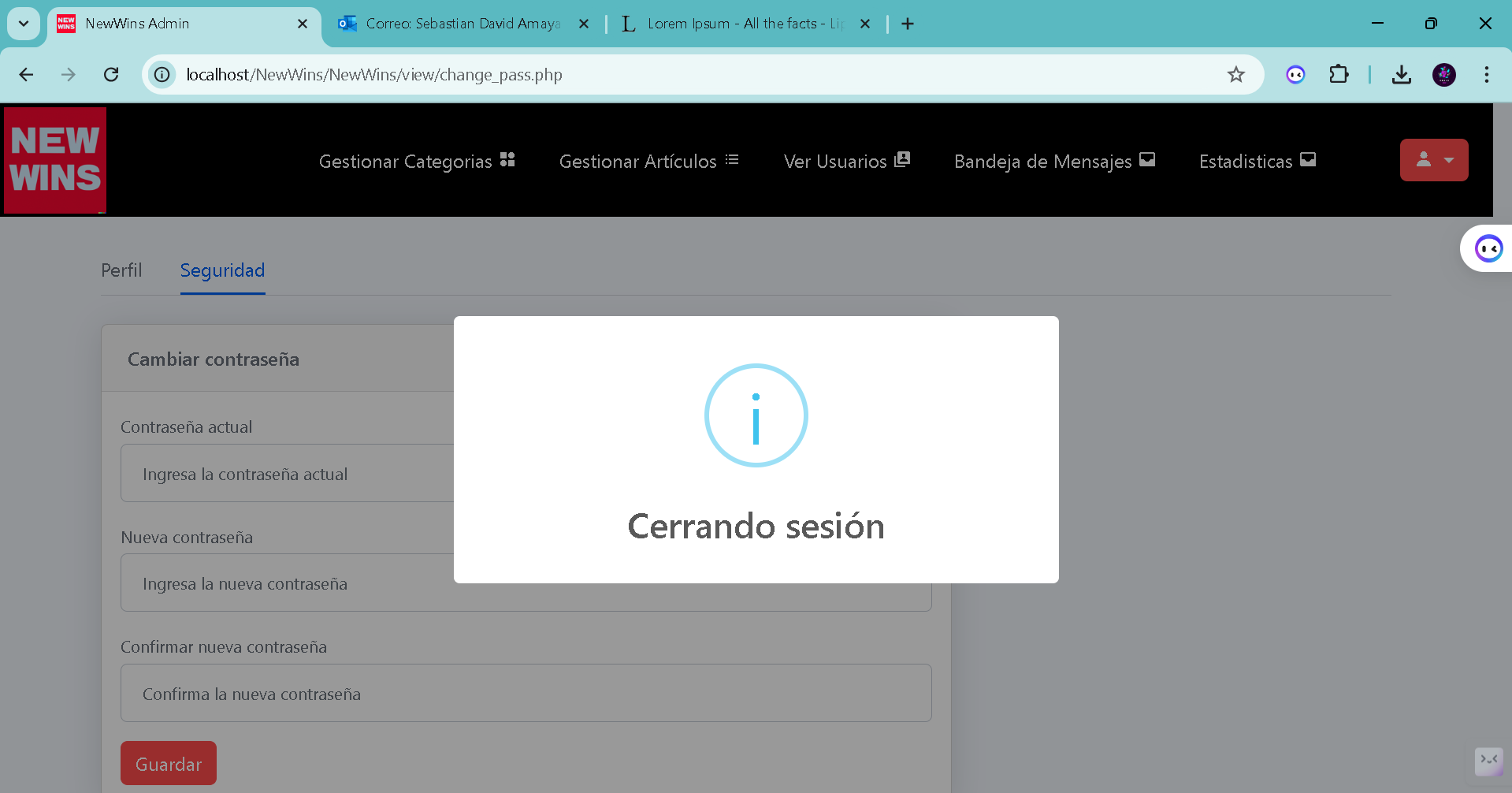
session\_destroy(); // Elimina la sesión actual del servidor

// 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

?>

****