MANUAL DE PROCESOS: SISTEMA DE AUTENTICACIÓN OTP Y AUDITORÍA

1. Introducción

Este documento describe el flujo lógico y técnico para el sistema de acceso seguro mediante contraseña temporal (OTP) y registro de actividad (Auditoría) desarrollado en PHP.

---

2. Diagrama de Flujo del Proceso

---

3. Descripción de Procesos por Módulo

Módulo A: Registro de Usuario

1. **Captura de datos:** El sistema recibe `email` y `password`.
2. **Seguridad:** La contraseña se procesa mediante la función `password_hash($pass, PASSWORD_BCRYPT)`.
3. **Persistencia:** Se inserta el registro en la tabla `users`.
4. **Auditoría:** Se inserta un registro en `audit_logs` con la acción `USER_REGISTRATION`.

Módulo B: Inicio de Sesión (Fase 1 - Credenciales)

1. **Validación:** El sistema busca el `email` en la base de datos.
2. **Verificación:** Se compara el hash con `password_verify()`.
3. **Generación de OTP:**
    - Si es correcto, se genera un código aleatorio de 6 dígitos: `$otp = str_pad(random_int(0, 999999), 6, '0', STR_PAD_LEFT);`.
    - Se calcula la expiración: `date('Y-m-d H:i:s', strtotime('+5 minutes'))`.
    - Se guarda el OTP en la tabla `otp_codes`.
4. **Envío:** Se envía el código al correo del usuario (usando `mail()` o PHPMailer).
5. **Auditoría:** Se registra `LOGIN_ATTEMPT_SUCCESS` o `LOGIN_ATTEMPT_FAILED`.

Módulo C: Verificación de OTP (Fase 2)

1. **Recepción:** El usuario ingresa el código recibido.
2. **Validación en DB:**
    - Se busca el código más reciente para ese `user_id` que no haya sido usado (`used = 0`).
    - Se verifica que `expires_at` sea mayor a la hora actual.
3. **Consumo:** Si es válido, se marca el código como `used = 1`.

4. **Sesión:** Se inicia la sesión de PHP `session_start()` y se redirige al Dashboard.
5. **Auditoría:** Se registra `OTP_VERIFICATION_SUCCESS`.

---

4. Estructura de la Base de Datos (MySQL)

SQL

```
CREATE TABLE users (
    id INT AUTO_INCREMENT PRIMARY KEY,
    email VARCHAR(255) UNIQUE NOT NULL,
    password_hash VARCHAR(255) NOT NULL,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
) ENGINE=InnoDB;

CREATE TABLE otp_codes (
    id INT AUTO_INCREMENT PRIMARY KEY,
    user_id INT NOT NULL,
    code VARCHAR(6) NOT NULL,
    expires_at DATETIME NOT NULL,
    used TINYINT(1) DEFAULT 0,
    FOREIGN KEY (user_id) REFERENCES users(id)
) ENGINE=InnoDB;

CREATE TABLE audit_logs (
    id INT AUTO_INCREMENT PRIMARY KEY,
    user_id INT,
    action VARCHAR(50),
    ip_address VARCHAR(45),
    details TEXT,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
) ENGINE=InnoDB;
```

---

5. Consideraciones de Seguridad

- **Prevención de Inyección SQL:** Es obligatorio el uso de **PDO** o **MySQLi** con sentencias preparadas (*Prepared Statements*).
- **Protección Brute Force:** El sistema debe bloquear la cuenta o la IP tras 5 intentos fallidos registrados en la tabla de auditoría.
- **Sesiones Seguras:** Configurar `session.cookie_httponly = 1` en el archivo `php.ini`.

1. Descarga Manual de PHPMailer

1. Ve al repositorio oficial en GitHub: PHPMailer GitHub.
2. Haz clic en el botón verde **"Code"** y luego en **"Download ZIP"**.
3. Descomprime la carpeta y copia las carpetas `src` a tu proyecto. Renombra la carpeta a `phpmailer` para que sea más fácil de ubicar.

Debes asegurarte de tener estos archivos en tu carpeta:

- `phpmailer/src/Exception.php`
- `phpmailer/src/PHPMailer.php`
- `phpmailer/src/SMTP.php`

---

2. El Procesador PHP Actualizado (enviar_otp.php)

Sustituiremos el `autoload` de Composer por los `require` manuales.

```
<?php
// Carga manual de las librerías
require 'phpmailer/src/Exception.php';
require 'phpmailer/src/PHPMailer.php';
require 'phpmailer/src/SMTP.php';

use PHPMailer\PHPMailer\PHPMailer;
use PHPMailer\PHPMailer\Exception;

session_start();
header('Content-Type: application/json');

if ($_SERVER['REQUEST_METHOD'] == 'POST') {
    $email = $_POST['email'];
    $usuario = $_POST['usuario'];

    // Generamos OTP de 6 dígitos
    $otp = rand(100000, 999999);

    // Guardamos en sesión para validar después
    $_SESSION['otp_sistema'] = $otp;
    $_SESSION['email_usuario'] = $email;

    $mail = new PHPMailer(true);

    try {
        // Configuración del Servidor Gmail
        $mail->isSMTP();
        $mail->Host       = 'smtp.gmail.com';
        $mail->SMTPAuth   = true;
        $mail->Username   = 'tu_correo@gmail.com';
```

```
    $mail->Password   = 'abcd efgh ijkl mnop'; // TUS 16 CARACTERES DE GOOGLE
    $mail->SMTPSecure = PHPMailer::ENCRYPTION_STARTTLS;
    $mail->Port       = 587;
    $mail->CharSet    = 'UTF-8';

    // Destinatarios
    $mail->setFrom('tu_correo@gmail.com', 'Sistema de Registro');
    $mail->addAddress($email);

    // Contenido del Correo
    $mail->isHTML(true);
    $mail->Subject = "Código de Verificación: $otp";
    $mail->Body    = "
      <div style='font-family: sans-serif; border: 1px solid #ddd; padding: 20px; border-radius:
10px;'>
        <h2>Hola, $usuario</h2>
        <p>Tu código de registro es:</p>
        <h1 style='color: #6366f1; letter-spacing: 5px;'>$otp</h1>
        <p>Este código expirará pronto.</p>
      </div>";

    $mail->send();
    echo json_encode(['status' => 'success']);
  } catch (Exception $e) {
    echo json_encode(['status' => 'error', 'message' => "Error: {$mail->ErrorInfo}"]);
  }
}
```

3. Script para Validar el OTP (validar_otp.php)

Para que el sistema funcione, necesitas un segundo archivo PHP que verifique si lo que el usuario escribió en los cuadritos es correcto.

```
<?php
session_start();
header('Content-Type: application/json');

if ($_SERVER['REQUEST_METHOD'] == 'POST') {
  // Recibimos los 6 números y los unimos
  $otp_usuario = implode('', $_POST['otp_inputs']);

  if (isset($_SESSION['otp_sistema']) && $otp_usuario == $_SESSION['otp_sistema']) {
    // ÉXITO
    unset($_SESSION['otp_sistema']); // Limpiamos el código usado
    echo json_encode(['status' => 'success', 'message' => '¡Registro completado con éxito!']);
  } else {
    // ERROR
    echo json_encode(['status' => 'error', 'message' => 'El código es incorrecto.']);
  }
}
```