REPÚBLICA DEL ECUADOR

CARRERA EN DESARROLLO DE SOFWARE

MANUAL TÉCNICO

APLICACIÓN WEB PARA RESERVA DE ESPACIOS EN EL INSTITUTO SUPERIOR TECNOLÓGICO TENA

AUTORA: JOFRE ALEXANDER JIMENEZ LAPO.

Tena - Ecuador

2025-IIS

1. ESTRUCTURA DEL SISTEMA

El sistema utiliza arquitectura MVC simplificada con PHP nativo, MySQL y frontend con HTML/CSS/JS.

```
sistema_reservas/
├── admin/              # Panel de administración
│   ├── dashboard.php
│   ├── usuarios.php
│   ├── espacios.php
│   ├── reservas.php
│   ├── ver_reserva.php
│   └── reportes.php
├── api/                # Endpoints API REST
│   ├── notificaciones.php
│   ├── espacios.php
│   └── horarios.php
├── auth/               # Módulo de autenticación
│   ├── logout.php
│   ├── recuperar_contrasena.php
│   └── reset_contrasena.php
├── config/             # Configuración
│   └── config.php
├── css/                # Estilos CSS
├── includes/           # Archivos de inclusión
├── js/                 # Scripts JavaScript
├── usuario/            # Panel de usuario
│   ├── dashboard.php
│   ├── nueva_reserva.php
│   ├── mis_reservas.php
│   ├── ver_reserva.php
│   ├── espacios_disponibles.php
│   └── perfil.php
├── vendor/             # Dependencias PHP (Composer)
└── index.php           # Punto de entrada (Login)
```

2. INSTALACIÓN DEL SISTEMA

Paso 1: Instalación de XAMPP

1. Descargar XAMPP desde el sitio oficial: https://www.apachefriends.org/

2. Ejecutar el instalador e instalar el software en la ruta predeterminada: C:\xampp\

3. Abrir el XAMPP Control Panel.

4. Iniciar los servicios Apache y MySQL, verificando que ambos se encuentren activos.

Paso 2: Copia del Proyecto

1. Descargar el proyecto en formato comprimido (ZIP).

2. Extraer el contenido del archivo en la siguiente ruta del servidor web:

C:\xampp\htdocs\sistema_reservas\

Esta ubicación permite que el servidor Apache pueda interpretar correctamente los archivos del sistema.

Paso 3: Instalación de Dependencias

Para la instalación de las dependencias del proyecto es necesario contar previamente con Composer, gestor de dependencias para PHP.

Descarga e instalación de Composer.

1. Acceder mediante un navegador web al sitio oficial de Composer: https://getcomposer.org/

2. Descargar el instalador correspondiente al sistema operativo.

3. Ejecutar el instalador y completar el proceso de instalación utilizando la configuración predeterminada.

4. Verificar la instalación abriendo la consola de comandos (CMD) y ejecutando: composer –version.

5. Abrir la consola de comandos (CMD) y ubicarse en la carpeta del proyecto:

cd C:\xampp\htdocs\sistema_reservas

6. Ejecutar el siguiente comando para instalar las dependencias del sistema: composer install

Este proceso instalará automáticamente todas las librerías necesarias para el funcionamiento del sistema, incluyendo PHPMailer.

Paso 4: Creación de la Base de Datos

1. Abrir un navegador web y acceder a: http://localhost/phpmyadmin/

2. Seleccionar la pestaña SQL.

3. Ejecutar el siguiente comando para crear la base de datos:

CREATE DATABASE sistema_reservas_istt

CHARACTER SET utf8mb4

COLLATE utf8mb4_unicode_ci;

Paso 5: Importación de Tablas

1. Seleccionar la base de datos sistema_reservas_istt en phpMyAdmin.

2. Acceder a la pestaña Importar.

3. Seleccionar el archivo database.sql incluido en el proyecto.

4. Presionar el botón Importar para crear las tablas correspondientes.

```
-- Base de Datos del Sistema de Reservas del ISTT

-- Crear base de datos
CREATE DATABASE IF NOT EXISTS sistema_reservas_istt
CHARACTER SET utf8mb4
COLLATE utf8mb4_unicode_ci;

USE sistema_reservas_istt;
```

```sql
-- ============================================
-- Tabla: usuarios
-- ============================================
CREATE TABLE IF NOT EXISTS usuarios (
    id INT AUTO_INCREMENT PRIMARY KEY,
    cedula VARCHAR(10) UNIQUE NOT NULL,
    nombres VARCHAR(100) NOT NULL,
    apellidos VARCHAR(100) NOT NULL,
    correo VARCHAR(100) UNIQUE NOT NULL,
    telefono VARCHAR(20),
    contrasena VARCHAR(255) NOT NULL,
    rol ENUM('administrador', 'usuario') DEFAULT 'usuario',
    estado ENUM('activo', 'inactivo') DEFAULT 'activo',
    ultima_ip VARCHAR(45),
    ultimo_acceso DATETIME,
    fecha_registro DATETIME DEFAULT CURRENT_TIMESTAMP,
    INDEX idx_cedula (cedula),
    INDEX idx_correo (correo),
    INDEX idx_rol (rol),
    INDEX idx_estado (estado)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;

-- ============================================
-- Tabla: espacios
-- ============================================
CREATE TABLE IF NOT EXISTS espacios (
    id INT AUTO_INCREMENT PRIMARY KEY,
    nombre VARCHAR(100) NOT NULL,
    tipo ENUM('laboratorio', 'biblioteca', 'aula_magna') NOT NULL,
    capacidad INT NOT NULL,
    ubicacion VARCHAR(200) NOT NULL,
    equipamiento TEXT,
    estado ENUM('disponible', 'mantenimiento', 'no_disponible') DEFAULT
'disponible',
    fecha_creacion DATETIME DEFAULT CURRENT_TIMESTAMP,
    INDEX idx_tipo (tipo),
    INDEX idx_estado (estado)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;

-- ============================================
-- Tabla: reservas
-- ============================================
CREATE TABLE IF NOT EXISTS reservas (
    id INT AUTO_INCREMENT PRIMARY KEY,
```

```sql
    usuario_id INT NOT NULL,
    espacio_id INT NOT NULL,
    fecha_reserva DATE NOT NULL,
    hora_inicio TIME NOT NULL,
    hora_fin TIME NOT NULL,
    motivo TEXT NOT NULL,
    estado ENUM('pendiente', 'aprobada', 'rechazada', 'cancelada') DEFAULT
'pendiente',
    observaciones TEXT,
    motivo_rechazo TEXT,
    -- Campos para declaración de asistencia
    asistencia_declarada TINYINT(1) DEFAULT NULL,
    asistencia_validada ENUM('pendiente', 'confirmada', 'rechazada') DEFAULT
NULL,
    fecha_declaracion_asistencia DATETIME DEFAULT NULL,
    fecha_solicitud DATETIME DEFAULT CURRENT_TIMESTAMP,
    fecha_actualizacion DATETIME DEFAULT CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP,
    FOREIGN KEY (usuario_id) REFERENCES usuarios(id) ON DELETE CASCADE,
    FOREIGN KEY (espacio_id) REFERENCES espacios(id) ON DELETE CASCADE,
    INDEX idx_usuario (usuario_id),
    INDEX idx_espacio (espacio_id),
    INDEX idx_estado (estado),
    INDEX idx_fecha_reserva (fecha_reserva),
    INDEX idx_fecha_solicitud (fecha_solicitud),
    CONSTRAINT chk_horas CHECK (hora_fin > hora_inicio)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;

-- ============================================
-- Tabla: intentos_login
-- ============================================
CREATE TABLE IF NOT EXISTS intentos_login (
    id INT AUTO_INCREMENT PRIMARY KEY,
    usuario_id INT,
    ip VARCHAR(45) NOT NULL,
    correo_intentado VARCHAR(100),
    user_agent VARCHAR(255),
    exitoso TINYINT(1) DEFAULT 0,
    fecha_intento DATETIME DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (usuario_id) REFERENCES usuarios(id) ON DELETE CASCADE,
    INDEX idx_usuario (usuario_id),
    INDEX idx_fecha (fecha_intento),
    INDEX idx_exitoso (exitoso),
    INDEX idx_correo (correo_intentado)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;
```

```sql
-- ============================================
-- Tabla: tokens_recuperacion
-- ============================================
CREATE TABLE IF NOT EXISTS tokens_recuperacion (
    id INT AUTO_INCREMENT PRIMARY KEY,
    usuario_id INT NOT NULL,
    token VARCHAR(64) UNIQUE NOT NULL,
    fecha_expiracion DATETIME NOT NULL,
    usado TINYINT(1) DEFAULT 0,
    fecha_creacion DATETIME DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (usuario_id) REFERENCES usuarios(id) ON DELETE CASCADE,
    INDEX idx_token (token),
    INDEX idx_usuario (usuario_id),
    INDEX idx_expiracion (fecha_expiracion)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;

-- ============================================
-- Tabla: notificaciones
-- ============================================
CREATE TABLE IF NOT EXISTS notificaciones (
    id INT AUTO_INCREMENT PRIMARY KEY,
    usuario_id INT NOT NULL,
    reserva_id INT,
    tipo ENUM('info', 'exito', 'alerta', 'error') DEFAULT 'info',
    titulo VARCHAR(100) NOT NULL,
    mensaje TEXT NOT NULL,
    leida TINYINT(1) DEFAULT 0,
    fecha_creacion DATETIME DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (usuario_id) REFERENCES usuarios(id) ON DELETE CASCADE,
    FOREIGN KEY (reserva_id) REFERENCES reservas(id) ON DELETE SET NULL,
    INDEX idx_usuario_leida (usuario_id, leida),
    INDEX idx_fecha (fecha_creacion),
    INDEX idx_reserva (reserva_id)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;

-- ============================================
-- Datos de Ejemplo (Opcional)
-- ============================================

-- Insertar usuario administrador de ejemplo
-- ⚠ IMPORTANTE: Cambiar la contraseña después del primer inicio de sesión
-- Contraseña por defecto: "123456"
INSERT INTO usuarios (
    cedula,
```

```
    nombres,
    apellidos,
    correo,
    telefono,
    contrasena,
    rol,
    estado,
    fecha_registro
) VALUES (
    '2200334494',
    'Admin',
    'Sistema',
    'lapoalexander28@gmail.com',
    '0987654321',
    '$2y$10$auxgFxy07yEM0Tdl8h8A3OM9e3rKafjKon9/RhaucT8QtRsaOsiM', -- Hash de
"123456"
    'administrador',
    'activo',
    NOW()
) ON DUPLICATE KEY UPDATE id=id;

-- Insertar espacios de ejemplo
INSERT INTO espacios (nombre, tipo, capacidad, ubicacion, equipamiento, estado)
VALUES
('Laboratorio de Computación 1', 'laboratorio', 30, 'Edificio A - Piso 2', '30
computadoras, proyector, pizarra digital', 'disponible'),
('Laboratorio de Computación 2', 'laboratorio', 25, 'Edificio A - Piso 2', '25
computadoras, proyector', 'disponible'),
('Biblioteca Central', 'biblioteca', 50, 'Edificio B - Piso 1', 'Mesas de
estudio, estanterías, área de lectura', 'disponible'),
('Aula Magna', 'aula_magna', 200, 'Edificio Principal - Piso 1', 'Sistema de
sonido, proyector, escenario', 'disponible'),
('Laboratorio de Redes', 'laboratorio', 20, 'Edificio A - Piso 3', 'Equipos de
red, servidores, herramientas', 'disponible')
ON DUPLICATE KEY UPDATE id=id;
```

Paso 6: Configuración de la Conexión a la Base de Datos

Editar el archivo config.php ubicado en la raíz del proyecto y configurar los parámetros de conexión:

Ilustración 4

Conexión con la base de datos

```php
// Configuración de la base de datos
define('DB_HOST', 'localhost');
define('DB_NAME', 'sistema_reservas_istt'); // Cambiar según tu configuración
define('DB_USER', 'root');
define('DB_PASS', '');

```

Nota. Configuración utilizada para establecer la conexión entre el sistema y la base de datos MySQL.

Paso 7: Configuración del Correo Electrónico (Gmail)

1. Acceder a la configuración de seguridad de la cuenta de Gmail:

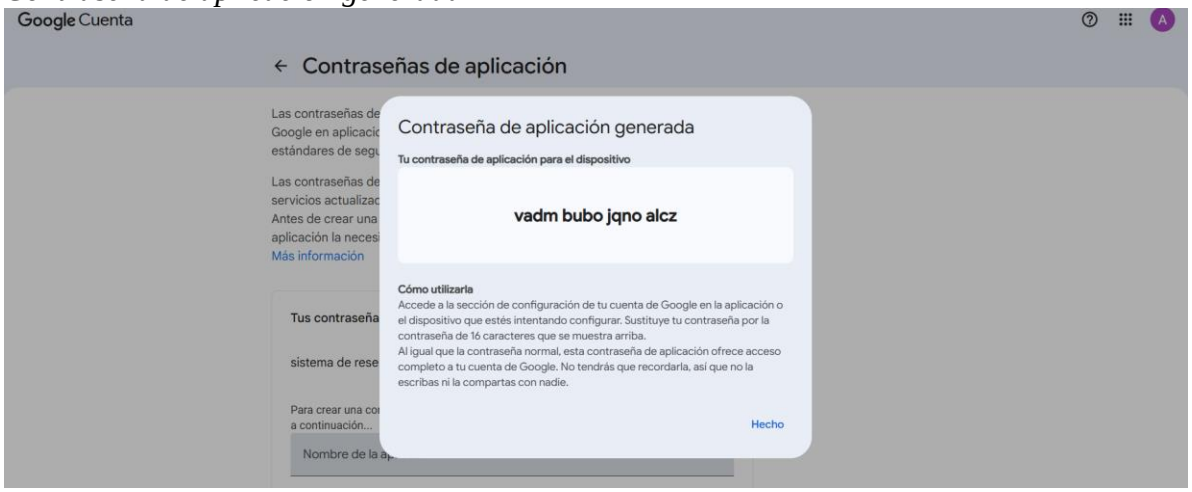
https://myaccount.google.com/security

2. Activar la verificación en dos pasos.

3. Generar una contraseña de aplicación.

Ilustración 5

Contraseña de aplicación generada



4. Configurar los siguientes parámetros en el archivo config.php:

Ilustración 6

Configuración del servicio SMTP con Gmail

```
// Configuración de correo electrónico (Gmail)
define('SMTP_HOST', 'smtp.gmail.com');
define('SMTP_PORT', 587);
define('SMTP_USER', 'lapoalexander28@gmail.com'); // Cambiar por tu correo Gmail
define('SMTP_PASS', 'dsvytjxnhsufuxob'); // Cambiar por tu contraseña de aplicación de Gmail
define('SMTP_FROM_EMAIL', 'lapoalexander28@gmail.com'); // Cambiar por tu correo Gmail
define('SMTP_FROM_NAME', 'Sistema de Reservas IST Tena');
```

Nota. Configuración utilizada para establecer la conexión del sistema con el servidor SMTP de Gmail.

Paso 8: Verificación de la Instalación

1. Acceder al sistema mediante un navegador web:

http://localhost/sistema_reservas/

2. Iniciar sesión con las credenciales del administrador.

Credenciales por defecto: usuario: 2200334494, contraseña: 123456.

7. CODIFICACION DEL SISTEMA

7.1 Configuración del sistema (Config.php)

```
<?php
// Configuración de seguridad de sesiones
ini_set('session.cookie_httponly', 1);      // Previene acceso JavaScript a cookies
ini_set('session.cookie_samesite', 'Strict'); // Protección CSRF
ini_set('session.use_strict_mode', 1);      // Rechaza IDs de sesión desconocidos

// Conexión a base de datos con PDO
function conectarDB() {
  try {
    $conexion = new PDO(
      "mysql:host=localhost;dbname=sistema_reservas_istt;charset=utf8mb4",
      "root", "",
      [
        PDO::ATTR_ERRMODE => PDO::ERRMODE_EXCEPTION,
        PDO::ATTR_DEFAULT_FETCH_MODE => PDO::FETCH_ASSOC,
        PDO::ATTR_EMULATE_PREPARES => false  // Consultas preparadas nativas
      ]
    );
    return $conexion;
  } catch(PDOException $e) {
    die("Error de conexión: " . $e->getMessage());
  }
}

// Sanitización de datos de entrada para prevenir XSS
function limpiarDatos($data) {
  $data = trim($data);            // Elimina espacios en blanco
  $data = stripslashes($data);    // Elimina barras invertidas
  $data = htmlspecialchars($data);  // Convierte caracteres especiales a entidades HTML
  return $data;
}

// Verificación de autenticación
function estaLogueado() {
  return isset($_SESSION['usuario_id']) && !empty($_SESSION['usuario_id']);
}

// Verificación de rol administrador
function esAdministrador() {
```

```
    return isset($_SESSION['rol']) && $_SESSION['rol'] === 'administrador';
}

// Redirección HTTP
function redirigir($url) {
    header("Location: " . $url);
    exit;
}

// Envío de correo con PHPMailer y Gmail SMTP
function enviarCorreo($destinatario, $nombre_destinatario, $asunto, $mensaje_html) {
    require_once __DIR__ . '/vendor/autoload.php';

    $mail = new PHPMailer\PHPMailer\PHPMailer(true);
    $mail->isSMTP();
    $mail->Host = 'smtp.gmail.com';
    $mail->SMTPAuth = true;
    $mail->Username = SMTP_USER;
    $mail->Password = SMTP_PASS;
    $mail->SMTPSecure = PHPMailer\PHPMailer\PHPMailer::ENCRYPTION_STARTTLS;
    $mail->Port = 587;
    $mail->CharSet = 'UTF-8';
    $mail->Timeout = 5;

    $mail->setFrom(SMTP_FROM_EMAIL, SMTP_FROM_NAME);
    $mail->addAddress($destinatario, $nombre_destinatario);
    $mail->isHTML(true);
    $mail->Subject = $asunto;
    $mail->Body = $mensaje_html;

    $mail->send();
    return ['success' => true];
}
```

7.2 Inicio de sesión (Index.php)

```
<?php

require_once 'config.php';


if ($_SERVER['REQUEST_METHOD'] === 'POST') {
```

```
    $cedula = limpiarDatos($_POST['cedula']);
    $contrasena = $_POST['contrasena'];
    $ip_actual = obtenerIP();

    $db = conectarDB();
    $stmt = $db->prepare("SELECT * FROM usuarios WHERE cedula = ? AND estado = 'activo'");
    $stmt->execute([$cedula]);
    $usuario = $stmt->fetch();

    // Verificación de contraseña con algoritmo bcrypt
    if ($usuario && password_verify($contrasena, $usuario['contrasena'])) {

        // Registrar intento de login exitoso para auditoría
        $stmt = $db->prepare("
            INSERT INTO intentos_login (usuario_id, ip, correo_intentado, user_agent, exitoso,
fecha_intento)
            VALUES (?, ?, ?, ?, 1, NOW())
        ");
        $stmt->execute([$usuario['id'], $ip_actual, $cedula, $_SERVER['HTTP_USER_AGENT']]);

        // Verificar actividad sospechosa (cambio de IP)
        $actividad = verificarActividadSospechosa($usuario['id'], $ip_actual);
        if ($actividad['sospechosa']) {
            enviarAlertaActividadSospechosa($usuario['id'], $actividad['tipo'], $ip_actual);
        }
```

```
        // Actualizar última IP del usuario
        $stmt = $db->prepare("UPDATE usuarios SET ultima_ip = ?, ultimo_acceso = NOW() WHERE id = ?");
        $stmt->execute([$ip_actual, $usuario['id']]);

        // Regenerar ID de sesión para prevenir Session Fixation
        session_regenerate_id(true);

        // Crear variables de sesión
        $_SESSION['usuario_id'] = $usuario['id'];
        $_SESSION['cedula'] = $usuario['cedula'];
        $_SESSION['nombres'] = $usuario['nombres'];
        $_SESSION['rol'] = $usuario['rol'];

        // Redirección según rol
        if ($usuario['rol'] === 'administrador') {
            redirigir('admin/dashboard.php');
        } else {
            redirigir('usuario/dashboard.php');
        }
    } else {
        // Registrar intento fallido
        $stmt = $db->prepare("
            INSERT INTO intentos_login (usuario_id, ip, correo_intentado, user_agent, exitoso, fecha_intento)
            VALUES (?, ?, ?, ?, 0, NOW())
        ");
```

```
        $stmt->execute([$usuario['id'] ?? null, $ip_actual, $cedula, $_SERVER['HTTP_USER_AGENT']]);
        $error = 'Cédula o contraseña incorrecta';
    }
}
```

7.3 Panel de usuario (dashboard.php)

```
<?php
require_once '../config.php';

// Verificación de autenticación
if (!estaLogueado()) {
    redirigir('../index.php');
}

$db = conectarDB();

// Consulta de estadísticas del usuario con prepared statements
$stmt = $db->prepare("SELECT COUNT(*) as total FROM reservas WHERE usuario_id = ? AND estado = 'aprobada'");
$stmt->execute([$_SESSION['usuario_id']]);
$reservas_aprobadas = $stmt->fetch()['total'];

$stmt = $db->prepare("SELECT COUNT(*) as total FROM reservas WHERE usuario_id = ? AND estado = 'pendiente'");
$stmt->execute([$_SESSION['usuario_id']]);
$reservas_pendientes = $stmt->fetch()['total'];
```

```
// Últimas 10 reservas del usuario con JOIN
$stmt = $db->prepare("
  SELECT r.*, e.nombre as espacio_nombre, e.tipo
  FROM reservas r
  JOIN espacios e ON r.espacio_id = e.id
  WHERE r.usuario_id = ?
  ORDER BY r.fecha_reserva DESC
  LIMIT 10
");
$stmt->execute([$_SESSION['usuario_id']]);
$mis_reservas = $stmt->fetchAll();
```

7.4 Nueva reserva (usuario/nueva_reserva.php)

```
<?php
require_once '../config.php';

if (!estaLogueado()) {
  redirigir('../index.php');
}

if ($_SERVER['REQUEST_METHOD'] === 'POST') {
  $espacio_id = limpiarDatos($_POST['espacio_id']);
  $fecha_reserva = limpiarDatos($_POST['fecha_reserva']);
  $hora_inicio = limpiarDatos($_POST['hora_inicio']);
  $hora_fin = limpiarDatos($_POST['hora_fin']);
  $motivo = limpiarDatos($_POST['motivo']);

  // Validación de campos obligatorios
  if (empty($espacio_id) || empty($fecha_reserva) || empty($hora_inicio) ||
empty($hora_fin) || empty($motivo)) {
    $mensaje = 'Todos los campos son obligatorios';
  }
  // Validación de orden de horas
  elseif ($hora_fin <= $hora_inicio) {
    $mensaje = 'La hora de fin debe ser mayor a la hora de inicio';
```

```
}
// Validación de fecha no pasada
elseif (strtotime($fecha_reserva) < strtotime(date('Y-m-d'))) {
   $mensaje = 'No se pueden hacer reservas en fechas pasadas';
}
// Validación de hora no pasada (si es hoy)
elseif ($fecha_reserva === date('Y-m-d') && $hora_inicio < date('H:i')) {
   $mensaje = 'No se pueden hacer reservas en horarios que ya han pasado';
}
else {
   // Validación de horarios institucionales permitidos
   $hora_inicio_time = strtotime($hora_inicio);
   $hora_fin_time = strtotime($hora_fin);

   $manana_inicio = strtotime('08:00');
   $manana_fin = strtotime('12:00');
   $tarde_inicio = strtotime('14:00');
   $tarde_fin = strtotime('22:00');

   $horario_valido = false;

   // Verificar horario matutino (08:00 - 12:00)
   if ($hora_inicio_time >= $manana_inicio && $hora_fin_time <= $manana_fin) {
      $horario_valido = true;
   }
   // Verificar horario vespertino/nocturno (14:00 - 22:00)
   elseif ($hora_inicio_time >= $tarde_inicio && $hora_fin_time <= $tarde_fin) {
      $horario_valido = true;
   }

   if (!$horario_valido) {
      $mensaje = 'Horarios permitidos: 08:00-12:00 y 14:00-22:00';
   } else {
      // Verificación de conflictos con reservas APROBADAS
      $db = conectarDB();
      $stmt = $db->prepare("
         SELECT COUNT(*) as total
         FROM reservas
         WHERE espacio_id = ?
         AND fecha_reserva = ?
         AND estado = 'aprobada'
         AND (
            (hora_inicio <= ? AND hora_fin > ?) OR   -- Solapa al inicio
```

```
                (hora_inicio < ? AND hora_fin >= ?) OR   -- Solapa al final
                (hora_inicio >= ? AND hora_fin <= ?)     -- Contenida
            )
        ");
        $stmt->execute([
            $espacio_id, $fecha_reserva,
            $hora_inicio, $hora_inicio,
            $hora_fin, $hora_fin,
            $hora_inicio, $hora_fin
        ]);

        if ($stmt->fetch()['total'] > 0) {
            $mensaje = 'El espacio ya está reservado en ese horario';
        } else {
            // Crear reserva con estado pendiente
            $stmt = $db->prepare("
                INSERT INTO reservas (usuario_id, espacio_id, fecha_reserva, hora_inicio,
hora_fin, motivo, estado)
                VALUES (?, ?, ?, ?, ?, ?, 'pendiente')
            ");
            $stmt->execute([$_SESSION['usuario_id'], $espacio_id, $fecha_reserva, $hora_inicio,
$hora_fin, $motivo]);

            $reserva_id = $db->lastInsertId();

            // Notificación al administrador
            crearNotificacionNuevaReservaAdmin($reserva_id);
            enviarCorreoNuevaReservaAdmin($reserva_id);
        }
    }
  }
}
```

7.5 Panel del administrador (admin/dashboard.php)

```
<?php
require_once '../config.php';

// Verificación de autenticación Y rol administrador
if (!estaLogueado() || !esAdministrador()) {
  redirigir('../index.php');
}
```

```
$db = conectarDB();

// Estadísticas globales del sistema
$stmt = $db->query("SELECT COUNT(*) as total FROM usuarios WHERE estado = 'activo'");
$total_usuarios = $stmt->fetch()['total'];

$stmt = $db->query("SELECT COUNT(*) as total FROM espacios WHERE estado =
'disponible'");
$total_espacios = $stmt->fetch()['total'];

$stmt = $db->query("SELECT COUNT(*) as total FROM reservas WHERE estado =
'pendiente'");
$reservas_pendientes = $stmt->fetch()['total'];

// Reservas del día actual con CURDATE()
$stmt = $db->query("SELECT COUNT(*) as total FROM reservas WHERE fecha_reserva =
CURDATE()");
$reservas_hoy = $stmt->fetch()['total'];

// Últimas reservas con JOIN múltiple
$stmt = $db->query("
  SELECT r.*, u.nombres, u.apellidos, e.nombre as espacio_nombre
  FROM reservas r
  JOIN usuarios u ON r.usuario_id = u.id
  JOIN espacios e ON r.espacio_id = e.id
  ORDER BY r.fecha_solicitud DESC
  LIMIT 10
");
$ultimas_reservas = $stmt->fetchAll();
```

7.6 Gestión de reservas

```
<?php
require_once '../config.php';

if (!estaLogueado() || !esAdministrador()) {
  redirigir('../index.php');
}

$db = conectarDB();
```

```
if (isset($_GET['accion']) && isset($_GET['id'])) {
  $reserva_id = $_GET['id'];
  $accion = $_GET['accion'];

  if ($accion === 'aprobar') {
    // Obtener datos de la reserva a aprobar
    $stmt = $db->prepare("SELECT espacio_id, fecha_reserva, hora_inicio, hora_fin FROM
reservas WHERE id = ?");
    $stmt->execute([$reserva_id]);
    $reserva_actual = $stmt->fetch();

    // Verificar conflictos con otras reservas APROBADAS
    $stmt = $db->prepare("
      SELECT COUNT(*) as total
      FROM reservas
      WHERE espacio_id = ? AND fecha_reserva = ? AND estado = 'aprobada' AND id != ?
      AND (
        (hora_inicio <= ? AND hora_fin > ?) OR
        (hora_inicio < ? AND hora_fin >= ?) OR
        (hora_inicio >= ? AND hora_fin <= ?)
      )
    ");
    $stmt->execute([
      $reserva_actual['espacio_id'], $reserva_actual['fecha_reserva'], $reserva_id,
      $reserva_actual['hora_inicio'], $reserva_actual['hora_inicio'],
      $reserva_actual['hora_fin'], $reserva_actual['hora_fin'],
      $reserva_actual['hora_inicio'], $reserva_actual['hora_fin']
    ]);

    if ($stmt->fetch()['total'] > 0) {
      $mensaje = 'Conflicto: existe otra reserva aprobada para el mismo horario';
    } else {
      $stmt = $db->prepare("UPDATE reservas SET estado = 'aprobada' WHERE id = ?");
      $stmt->execute([$reserva_id]);

      crearNotificacionCambioEstado($reserva_id, 'aprobada');
      enviarNotificacionReserva($reserva_id, 'aprobar');
    }
  }
  elseif ($accion === 'cancelar') {
    $stmt = $db->prepare("SELECT espacio_id, fecha_reserva, hora_inicio FROM reservas
WHERE id = ?");
    $stmt->execute([$reserva_id]);
```

```
      $reserva = $stmt->fetch();

      // Validación temporal con DateTime
      $fecha_hora_reserva = new DateTime($reserva['fecha_reserva'] . ' ' .
$reserva['hora_inicio']);
      $ahora = new DateTime();
      $diferencia = $ahora->diff($fecha_hora_reserva);
      $horas_restantes = ($diferencia->days * 24) + $diferencia->h;

      // Validar que no haya pasado
      if ($fecha_hora_reserva < $ahora) {
        $mensaje = 'No se puede cancelar: la reserva ya pasó';
      }
      // Validar mínimo 24 horas de anticipación
      elseif ($horas_restantes < 24) {
        $mensaje = 'No se puede cancelar: faltan menos de 24 horas';
      }
      else {
        $db->beginTransaction();
        $stmt = $db->prepare("UPDATE reservas SET estado = 'cancelada' WHERE id = ?");
        $stmt->execute([$reserva_id]);
        $stmt = $db->prepare("UPDATE espacios SET estado = 'disponible' WHERE id = ?");
        $stmt->execute([$reserva['espacio_id']]);
        $db->commit();

        crearNotificacionCambioEstado($reserva_id, 'cancelada');
        enviarNotificacionReserva($reserva_id, 'cancelar');
      }
    }
}

// Filtros por estado y fecha
$filtro_estado = $_GET['estado'] ?? '';
$filtro_fecha = $_GET['fecha'] ?? '';

$sql = "SELECT r.*, u.nombres, u.apellidos, e.nombre as espacio_nombre
      FROM reservas r
      JOIN usuarios u ON r.usuario_id = u.id
      JOIN espacios e ON r.espacio_id = e.id WHERE 1=1";
$params = [];

if ($filtro_estado) {
   $sql .= " AND r.estado = ?";
```

```
    $params[] = $filtro_estado;
}
if ($filtro_fecha) {
    $sql .= " AND r.fecha_reserva = ?";
    $params[] = $filtro_fecha;
}
$sql .= " ORDER BY r.fecha_solicitud DESC";

$stmt = $db->prepare($sql);
$stmt->execute($params);
$reservas = $stmt->fetchAll();
```

7.7 Recuperar contraseña (recuperar_contrasena.php)

```
<?php
require_once 'config.php';

if ($_SERVER['REQUEST_METHOD'] === 'POST') {
    $cedula = limpiarDatos($_POST['cedula']);

    $db = conectarDB();
    $stmt = $db->prepare("SELECT id, nombres, apellidos, correo FROM usuarios WHERE
cedula = ? AND estado = 'activo'");
    $stmt->execute([$cedula]);
    $usuario = $stmt->fetch();

    if ($usuario && !empty($usuario['correo'])) {
        // Generar token de recuperación
        $token = generarTokenRecuperacion($usuario['id']);
        enviarCorreoRecuperacion($usuario['id'], $token);
        $mensaje = 'Se ha enviado un enlace de recuperación a su correo';
    } else {
        // Mensaje genérico por seguridad (evita enumeración de usuarios)
        $mensaje = 'Si existe una cuenta con esa cédula, recibirá un enlace de recuperación';
    }
}

// Función de generación de token (en config.php)
function generarTokenRecuperacion($usuario_id) {
    $db = conectarDB();

    // Genera 32 bytes aleatorios criptográficamente seguros (CSPRNG)
```

```
    // bin2hex() convierte a 64 caracteres hexadecimales
    $token = bin2hex(random_bytes(32));

    // Token válido por 1 hora
    $expiracion = date('Y-m-d H:i:s', strtotime('+1 hour'));

    $stmt = $db->prepare("UPDATE usuarios SET token_recuperacion = ?, token_expiracion =
? WHERE id = ?");
    $stmt->execute([$token, $expiracion, $usuario_id]);

    return $token;
}

// Función de validación de token
function validarTokenRecuperacion($token) {
    $db = conectarDB();

    // Verifica token válido y no expirado
    $stmt = $db->prepare("
        SELECT id, nombres, correo
        FROM usuarios
        WHERE token_recuperacion = ?
        AND token_expiracion > NOW()
        AND estado = 'activo'
    ");
    $stmt->execute([$token]);
    return $stmt->fetch();
}
```