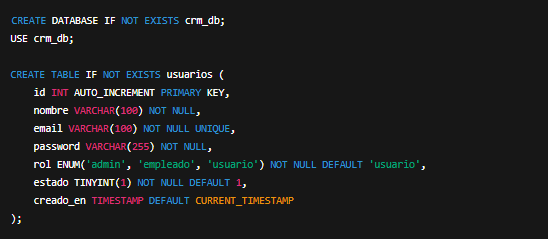
**Sistema de Gestión de Clientes (CRM- Customer Relationship Management) - Implementación del Login**

**1. Estructura de la Base de Datos (Tabla usuarios)**

Crearemos una tabla llamada usuarios con los siguientes campos:

| **Campo** | **Tipo de Dato** | **Descripción** |
| --- | --- | --- |
| id | INT (PK, AI) | Identificador único del usuario |
| nombre | VARCHAR(100) | Nombre completo del usuario |
| email | VARCHAR(100) | Correo electrónico (debe ser único) |
| password | VARCHAR(255) | Contraseña encriptada |
| rol | ENUM('admin', 'empleado', 'usuario') | Tipo de usuario |
| estado | TINYINT(1) | 1 = Activo, 0 = Inactivo |

**Nota:** Usaremos password\_hash() para almacenar las contraseñas de forma segura.



### Explicación:

* **id INT AUTO\_INCREMENT PRIMARY KEY** → Identificador único.
* **nombre VARCHAR(100) NOT NULL** → Nombre del usuario.
* **email VARCHAR(100) NOT NULL UNIQUE** → Correo único para autenticación.
* **password VARCHAR(255) NOT NULL** → Contraseña encriptada.
* **rol ENUM('admin', 'empleado', 'usuario') NOT NULL DEFAULT 'usuario'** → Control de acceso.
* **estado TINYINT(1) NOT NULL DEFAULT 1** → 1 = Activo, 0 = Inactivo.
* **creado\_en TIMESTAMP DEFAULT CURRENT\_TIMESTAMP** → Guarda la fecha y hora de creación.

📌 **Nota:** Recuerda que las contraseñas deben guardarse con password\_hash() para mayor seguridad.

**2. Proceso del Inicio de Sesión (Lógica Completa)**

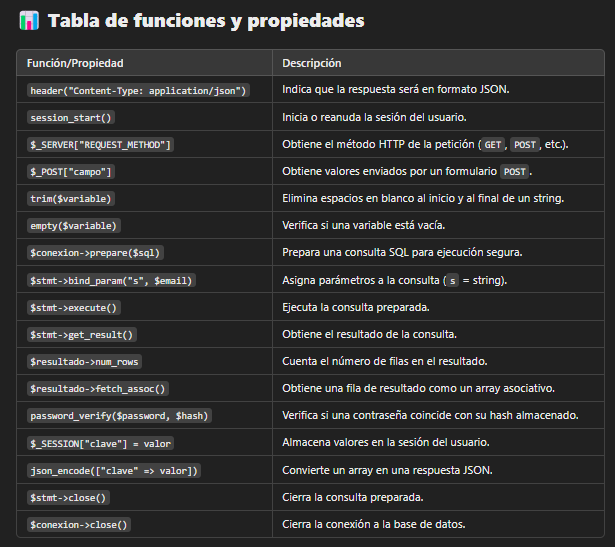
**Paso 1: Formulario de Login**

* Se solicitará **email** y **contraseña**.
* <h2>Iniciar Sesión</h2>
* <form id="loginForm">
* <label for="email">Correo electrónico:</label>
* <input type="email" id="email" name="email" required>
* <label for="password">Contraseña:</label>
* <input type="password" id="password" name="password" required>
* <button type="submit">Ingresar</button>
* </form>
* <p id="mensaje"></p>
* Validación en **JavaScript** antes de enviarlo al backend.
* document.addEventListener("DOMContentLoaded", function () {
* const loginForm = document.getElementById("loginForm");
* const mensaje = document.getElementById("mensaje");
* loginForm.addEventListener("submit", async function (event) {
* event.preventDefault();
* const email = document.getElementById("email").value.trim(); // trim() elimina espacios en blanco al inicio y al final.
* const password = document.getElementById("password").value.trim();
* // Validación de los campos vaciós
* if (!email || !password) {
* mensaje.textContent = "❌ Todos los campos son obligatorios";
* mensaje.style.color = "red";
* return;
* }
* // Se agregan email y password a formData.
* const formData = new FormData();
* formData.append("email", email);
* formData.append("password", password);
* console.log("Enviando datos a login.php..."); // 🔍 Verificar si fetch() se ejecuta
* try {
* const response = await fetch("login.php", {
* method: "POST",
* body: formData //  Envía formData al servidor.
* });
* console.log("Respuesta recibida:", response); // 🔍 Verificar respuesta
* const data = await response.json();
* if (data.success) {
* mensaje.textContent = "✅ Inicio de sesión exitoso. Redirigiendo...";
* mensaje.style.color = "green";
* setTimeout(() => {
* window.location.href = data.redirect;
* }, 1500); // Usa setTimeout() para redirigir después de 1.5 segundos.
* } else {
* mensaje.textContent = "❌ " + data.message;
* mensaje.style.color = "red";
* }
* } catch (error) {
* console.error("Error en fetch:", error); // 🔍 Verificar errores en fetch
* mensaje.textContent = "❌ Error en la conexión con el servidor";
* mensaje.style.color = "red";
* }
* });
* });



**Paso 2: Envío de Datos al Servidor (PHP + MySQL)**

* Verificamos si el usuario existe en la base de datos.
* Si el usuario **no existe**, mostramos un mensaje de error.
* Si el usuario **existe**, validamos la contraseña con password\_verify().
* Si la contraseña es incorrecta, mostramos un mensaje de error.
* Si la contraseña es correcta, iniciamos sesión y guardamos los datos del usuario en $\_SESSION.
* <?php
* header("Content-Type: application/json"); // 🔹 Esto soluciona errores de JSON
* // Asegura que la respuesta del servidor sea interpretada como JSON por el navegador
* session\_start(); // Inicia o reanuda una sesión para poder almacenar datos del usuario
* include 'conexion.php';
* // 🔹 Verificación del método HTTP
* if ($\_SERVER["REQUEST\_METHOD"] !== "POST") {
* echo json\_encode(["success" => false, "message" => "Método no permitido."]);
* exit;
* }
* // 🔹 Obtener desde $\_POST y limpiar datos
* $email = trim($\_POST["email"]); // trim() elimina espacios en blanco al inicio y al final.
* $password = trim($\_POST["password"]);
* // 🔹 Validación de campos vacíos
* if (empty($email) || empty($password)) {
* echo json\_encode(["success" => false, "message" => "Todos los campos son obligatorios."]);
* exit;
* }
* // Verificar si el usuario existe
* $sql = "SELECT id, nombre, email, password, rol, estado FROM usuarios WHERE email = ?";
* $stmt = $conexion->prepare($sql); //  vincula $email a la consulta como tipo string (s).
* $stmt->bind\_param("s", $email);
* $stmt->execute();
* $resultado = $stmt->get\_result();
* // 🔹 Verifica si el usuario existe
* if ($resultado->num\_rows === 1) {
* $usuario = $resultado->fetch\_assoc();
* // 🔹 Verifica si la cuenta está activa
* if ($usuario["estado"] == 0) {
* echo json\_encode(["success" => false, "message" => "Tu cuenta está inactiva."]);
* exit;
* }
* // 🔹 Verificación de contraseña
* if (password\_verify($password, $usuario["password"])) { // password\_verify() compara la contraseña ingresada con la almacenada en la base de datos (que está encriptada con password\_hash()).
* // 🔹 Almacenar datos en sesión
* $\_SESSION["usuario\_id"] = $usuario["id"];
* $\_SESSION["usuario\_nombre"] = $usuario["nombre"];
* $\_SESSION["usuario\_rol"] = $usuario["rol"];
* // Guarda en $\_SESSION el id, nombre y rol del usuario para usarlos en otras páginas.
* // 🔹 Redirigir según el rol
* $redirect = ($usuario["rol"] == "admin") ? "admin\_dashboard.php" : "dashboard.php"; // Si el usuario es admin, se redirige a admin\_dashboard.php,  Si no, se redirige a dashboard.php.
* echo json\_encode(["success" => true, "redirect" => $redirect]);
* } else {
* echo json\_encode(["success" => false, "message" => "Contraseña incorrecta."]);
* }
* } else {
* echo json\_encode(["success" => false, "message" => "El usuario no existe."]);
* }
* $stmt->close();
* $conexion->close();



**Paso 3: Redirección Según el Rol**

* **Administrador:** Redirigimos al panel de administración.
* **Empleado:** Redirigimos a su dashboard con acceso limitado.
* **Usuario estándar:** Accede solo a información permitida.
* // 🔹 Redirigir según el rol
* $redirect = ($usuario["rol"] == "admin") ? "admin\_dashboard.php" : "dashboard.php"; // Si el usuario es admin, se redirige a admin\_dashboard.php,  Si no, se redirige a dashboard.php.
* echo json\_encode(["success" => true, "redirect" => $redirect]);

**Paso 4: Protección de Rutas**

* Si un usuario no está autenticado, lo redirigimos al login.
* Si intenta acceder a un área sin permisos, lo redirigimos según su rol.

✅ **Insertar un usuario administrador (inser\_admin.php)**

<?php

include 'conexion.php';

// 3️⃣ Variables de usuario administrador

$nombre = "Admin";

$email = "samyr.perezpabon@gmail.com";

$password = "admin123";

$rol = "admin";

$estado = 1;

// 4️⃣ Encriptar la contraseña

$password\_hash = password\_hash($password, PASSWORD\_DEFAULT); // Se encripta la contraseña con password\_hash()

// PASSWORD\_DEFAULT: Usa el algoritmo de encriptación más reciente.

// 5️⃣ Consulta SQL con `?` como marcadores de posición

$sql = "INSERT INTO usuarios (nombre, email, password, rol, estado) VALUES (?, ?, ?, ?, ?)";

// 6️⃣ Preparar la sentencia

$stmt = $conexion->prepare($sql);

if (!$stmt) {

    die("❌ Error en la preparación de la consulta: " . $conexion->error);

}

// 7️⃣ Vincular parámetros y ejecutar la consulta

$stmt->bind\_param("ssssi", $nombre, $email, $password\_hash, $rol, $estado);

$stmt->execute();

echo "✅ Usuario administrador creado correctamente.";

// 8️⃣ Cerrar conexión

$stmt->close();

$conexion->close();

**3. Flujo de Lógica en Código (Resumido)**

1. **Usuario ingresa email y contraseña en el formulario.**
2. **JavaScript valida los campos antes de enviarlos al servidor.**
3. **PHP recibe los datos y busca el email en la base de datos.**
4. **Si el usuario existe, compara la contraseña con password\_verify().**
5. **Si es correcta, guarda los datos en $\_SESSION y redirige según su rol.**
6. **Si es incorrecta, muestra un error.**
7. **Si el usuario intenta acceder sin sesión, lo redirige al login.**

**4. Seguridad Adicional**

✅ **Hash de Contraseña:** Usaremos password\_hash() y password\_verify() para proteger las contraseñas. ✅ **Protección de Rutas:** Evitaremos acceso a áreas restringidas con sesiones y roles. ✅ **Medidas Anti-Ataques:** Implementaremos protección contra **SQL Injection** y **XSS**.

Con estos pasos, aseguramos un sistema de login seguro y funcional para nuestro CRM.