Como objetivo de esta práctica es que, al finalizar, adquiramos la capacidad y conocimiento para poder iniciar sesión, acceder a un dashboard protegido y restringir una página solo para rol admin con las herramientas de PHP y MySQL.

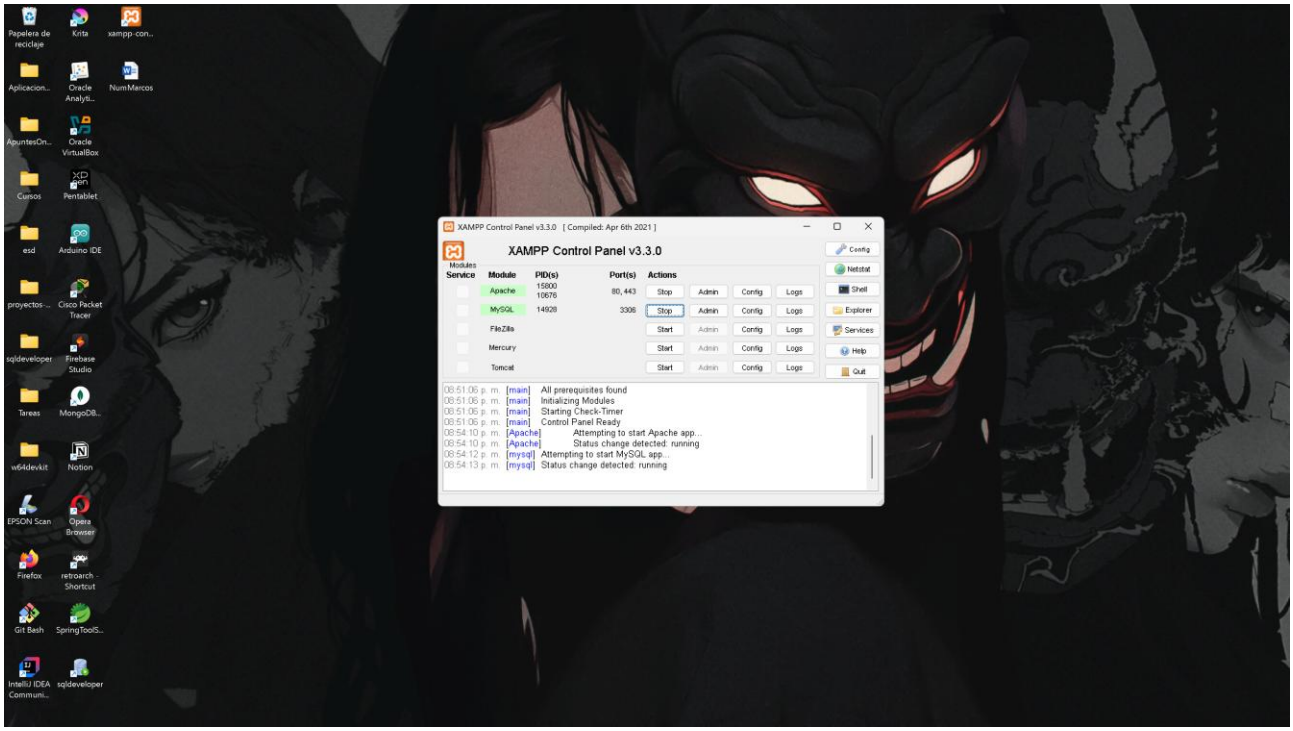

Ilustración 1. iniciar apache y MySQL mediante xamp

En este punto solo iniciamos apache y mysql con en xamp de no tenerlo hay que hacer la instalación correspondiente.

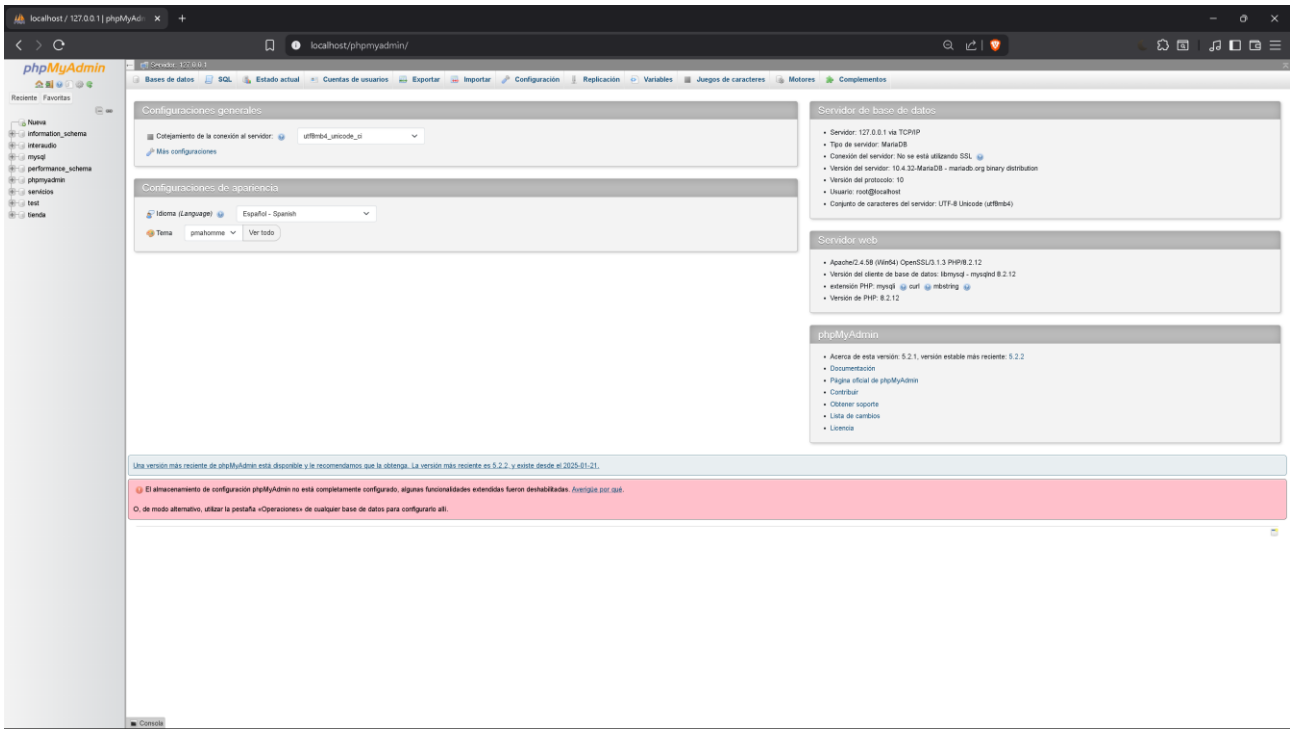

Ilustración 2. abrir phpmyadmin

Abrimos en el navegador de nuestra preferencia phpmyadmin o dando un clic en el botón admin de MySQL en xamp.

```sql
CREATE DATABASE IF NOT EXISTS i4app_db CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci;
USE i4app_db;

CREATE TABLE IF NOT EXISTS roles (
    id TYNYINT UNSIGNED PRIMARY KEY AUTO_INCREMENT,
    name VARCHAR(50) UNIQUE NOT NULL
);

CREATE TABLE IF NOT EXISTS users (
    id INT UNSIGNED PRIMARY KEY AUTO_INCREMENT,
    name VARCHAR(100) NOT NULL,
    email VARCHAR(120) UNIQUE NOT NULL,
    password_hash VARCHAR(255) NOT NULL,
    role_id TINYINT UNSIGNED NOT NULL,
    status ENUM ('active', 'blocked') DEFAULT 'active',
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (role_id) REFERENCES roles(id)
);

INSERT IGNORE INTO roles (name) VALUES ('admin'), ('user');

INSERT INTO users (name, email, password_hash, role_id, status) VALUES
('Admin vaca', 'vaca@i4app.local', 'admin', 1, 'active'),

INSERT INTO users (name, email, password_hash, role_id, status) VALUES
('User prueba', 'prueba@i4app.local', 'prueba' ,2, 'active');
```

Ilustración 3. codigo sql base de datos

Ejecutamos el código SQL para crear la base de datos y la tabla de usuarios con sus roles.

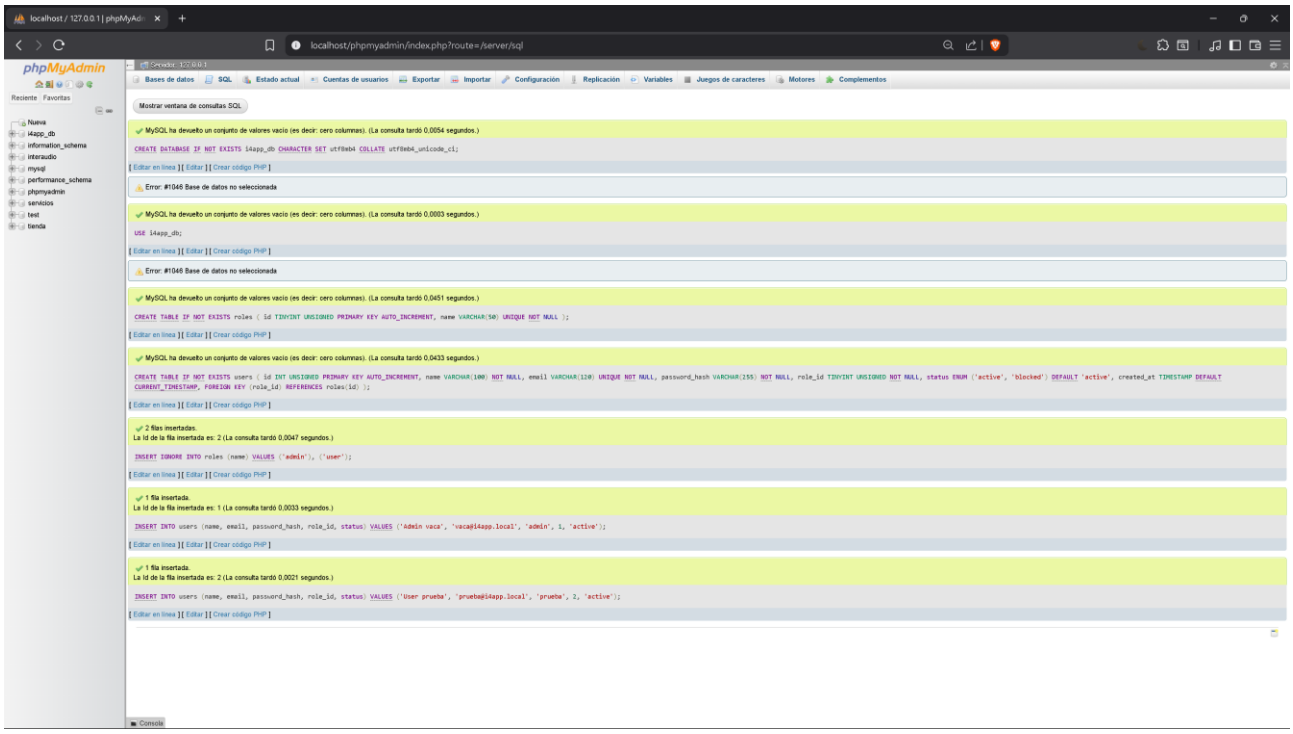

Ilustración 4. creacion de la base de datos

Verificamos en phpMyAdmin que la base de datos y la tabla se hayan creado correctamente.

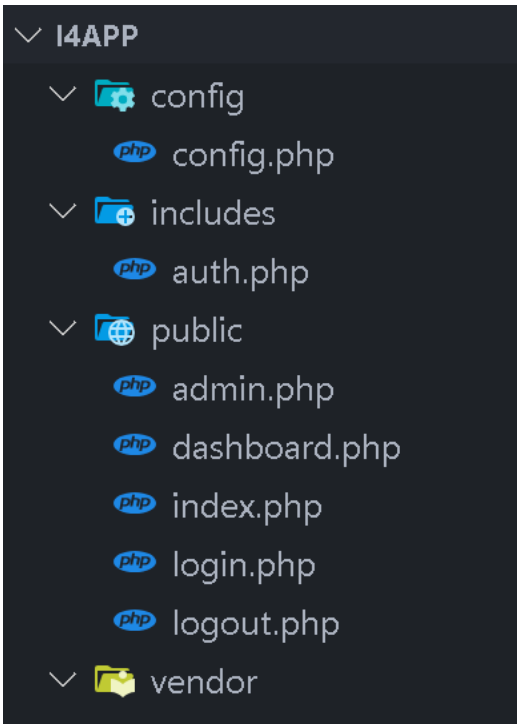

Ilustración 5. estructura del proyecto

Organizamos la estructura del proyecto con los archivos PHP necesarios para el login y dashboard.

```php
<?php

declare(strict_types=1);

$DB_HOST = '127.0.0.1';
$DB_NAME = 'i4app_db';
$DB_USER = 'root';
$DB_PASS = '';

$options = [
    PDO::ATTR_ERRMODE => PDO::ERRMODE_EXCEPTION,
    PDO::ATTR_DEFAULT_FETCH_MODE => PDO::FETCH_ASSOC,
];

try {
    $pdo = new PDO("mysql:host=$DB_HOST;dbname=$DB_NAME;charset=utf8mb4", $DB_USER,
$DB_PASS, $options);
} catch (PDOException $e) {
    die("error de conexion: " . $e->getMessage());
}

session_start();
```

Ilustración 6. código config.php

Configuramos la conexión a la base de datos usando PDO para mayor seguridad.

```php
<?php

declare(strict_types=1);

function isLoggedIn(): bool
{
    return isset($_SESSION['user_id']);
}

function require_login(): void
{
    if (!isLoggedIn()) {
        header('location: index.php?msg=login_required');
        exit();
    }
}

function require_role(string $roleName): void
{
    if (!isset($_SESSION['role_name']) || $_SESSION['role_name'] !== $roleName) {
        header('location: dashboard.php?msg=forbidden');
        exit;
    }
}

function regenerate_session(): void
{
    if (session_status() === PHP_SESSION_ACTIVE) {
        session_regenerate_id(true);
    }
}
```

Ilustración 7. codigo auth.php

Creamos el archivo auth.php para verificar si el usuario ya inició sesión antes de acceder a las páginas.

```php
<?php require_once __DIR__ . '/../config/config.php'; ?>
<!DOCTYPE html>
<html lang="en">

<head>
    <meta>charset="UTF-8">
    <title>Login I4App</title>
    <link rel="stylesheet" href=
"https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/css/bootstrap.min.css">
</head>

<body class="container py-5">
    <h1 class="mb-4">Inicio de sesion</h1>
    <?php if (isset($_GET['msg'])): ?>
        <div class="alert alert-warning"><?php echo htmlspecialchars($_GET['msg']); ?>
</div>
    <?php endif; ?>
    <form action="login.php" method="post" class="card p-4">
        <div class="mb_3">
            <label class="form-label">Email:</label>
            <input type="email" name="email" class="form-control" required autofocus>
        </div>
    </form>
    <div class="mb-3">
        <label class="form-label">Contraseña:</label>
        <input type="password" name="password" class="form-control" required>
    </div>
    <button class="btn btn-primary">Entrar</button>
</body>

</html>
```

Ilustración 8. codigo index.php

En index.php redirigimos al usuario a login si no está autenticado, usando la lógica de auth.php.

```php
<?php
require_once __DIR__ . '/../config/config.php';
require_once __DIR__ . '/../includes/auth.php';

$email = $_POST['email'] ?? "";
$pass = $_POST['password'] ?? "";
if (!$email || !$pass) {
    header('Location: index.php?msg=Datos incompletos');
    exit;
}

$stmt = $pdo->prepare("
    SELECT u.id, u.name, u.email, u.password_hash, r.name AS role_name FROM users u
    JOIN roles r ON r.id = u.role_id
    WHERE u.email = :email AND u.status = 'active'
    LIMIT 1
");
$stmt->execute([':email' => $email]);
$user = $stmt->fetch();

if (!$user || !password_verify($pass, $user['password_hash'])) {
    header('Location: index.php?msg=Credenciales inválidas');
    exit;
}

regenerate_session();
$_SESSION["user_id"] = (int)$user['id'];
$_SESSION["user_name"] = $user['name'];
$_SESSION["role_name"] = $user['role_name'];

header('Location: dashboard.php');
```

Ilustración 9. codigo login.php

Implementamos el formulario de login que valida usuario y contraseña usando password_verify().

```php
<?php
require_once __DIR__ . '/../config/config.php';
require_once __DIR__ . '/../includes/auth.php';
require_login();
?>
<!DOCTYPE html>
<html lang="es">

<head>
    <meta charset="UTF-8">
    <title>Dashboard</title>
    <link rel="stylesheet" href=
"https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/css/bootstrap.min.css">
</head>

<body class="container py-5">
    <h1>Bienvenido, <?php echo htmlspecialchars($_SESSION['user_name']); ?></h1>
    <p>Tu rol es: <strong><?php echo htmlspecialchars($_SESSION['role_name']); ?></
strong></p>
    <a class="btn btn-secondary" href="admin.php">Zona Administrador</a>
    <a class="btn btn-outline-danger" href="logout.php">Cerrar sesión</a>
</body>

</html>
```

Ilustración 10. codigo dashboard.php

Creamos dashboard.php que muestra contenido diferente dependiendo del rol del usuario.

```php
<?php
require_once __DIR__ . '/../config/config.php';
require_once __DIR__ . '/../includes/auth.php';
require_login();
require_role('admin');
?>
<!DOCTYPE html>
<html lang="es">

<head>
    <meta charset="UTF-8">
    <title>Admin - Solo Administradores</title>
    <link rel="stylesheet" href=
"https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/css/bootstrap.min.css">
</head>

<body class="container py-5">
    <h1>Panel de Administración</h1>
    <p>Solo usuarios con rol <strong>admin</strong> pueden ver esto.</p>
    <a class="btn btn-primary" href="dashboard.php">Volver</a>
</body>

</html>
```

Ilustración 11. codigo admin.php

En admin.php restringimos el acceso solo a usuarios con rol de administrador.

```php
<?php
require_once __DIR__  . '/../config/config.php';
$_SESSION = [];
if (ini_get("session.use_cookies")) {
    $params = session_get_cookie_params();
    setcookie(session_name(), '', time() - 42000,
    $params["path"], $params["domain"],
    $params["secure"], $params["httponly"]);
};
session_destroy();
header('Location: index.php?msg=Session_cenrada');
```

Ilustración 12. codigo logout.php

Implementamos logout.php para destruir la sesión y redirigir al usuario al login.

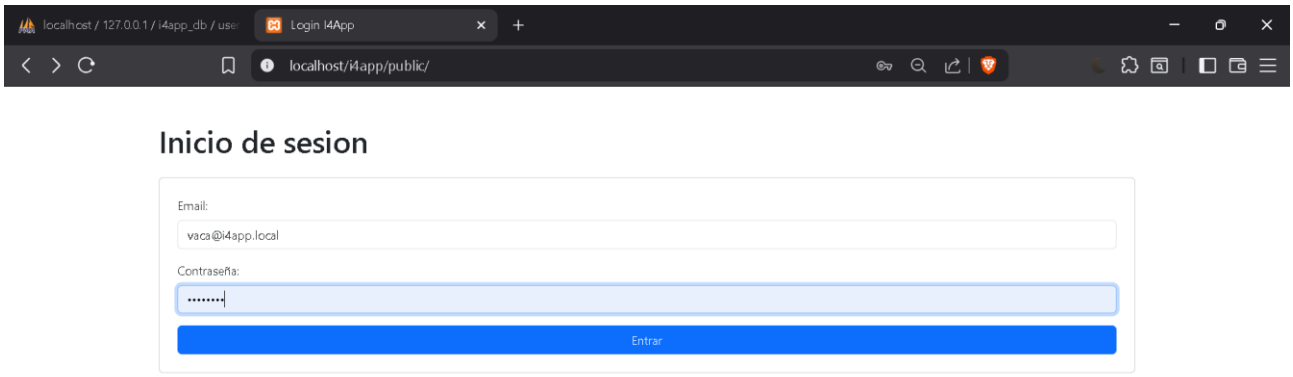

Ilustración 13.inicio de sesion admin

Probamos el inicio de sesión con un usuario administrador para verificar el acceso a todas las funciones.

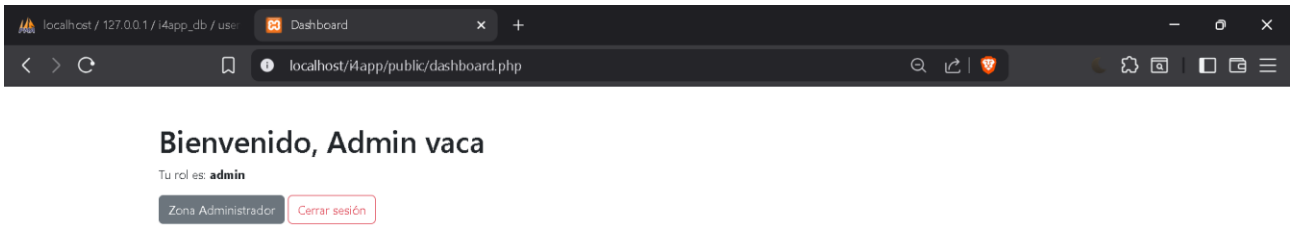

Ilustración 14. inicio de sesion admin

Comprobamos que el administrador puede acceder a la página admin.php exclusiva para su rol.

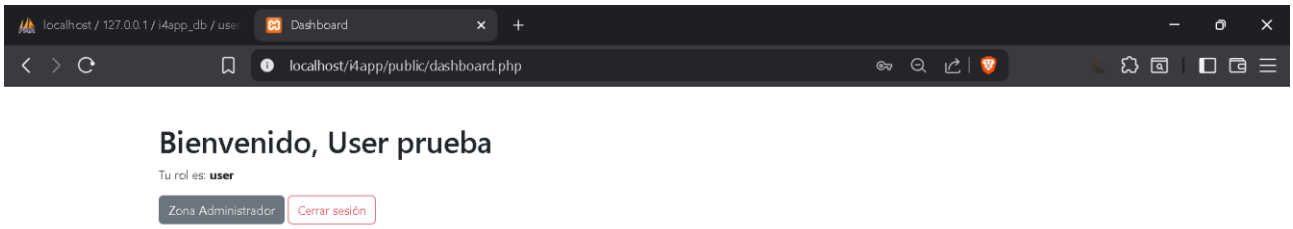

Ilustración 15. inicio de sesion usuario

Verificamos el inicio de sesión de un usuario normal y su acceso restringido al área de administrador.

1. ¿Por qué usar password_hash()/password_verify() en lugar de contraseñas planas?

Para no almacenar las contraseñas en texto plano en la base de datos. Así se protege la información de los usuarios y se aumenta la seguridad del sistema.

2. ¿Qué riesgo se evita con PDO + sentencias preparadas?

Se evita el riesgo de inyección SQL, que permite a un atacante ejecutar código malicioso en la base de datos. Las sentencias preparadas separan los datos del comando SQL.

3. ¿Dónde se aplica autenticación y dónde autorización en este flujo?

La autenticación se aplica en el login, verificando usuario y contraseña. La autorización se ve en el dashboard y admin.php, donde se controla el acceso según el rol del usuario.

4. ¿Qué mejorarías para pasar este flujo a producción (HTTPS, cookies seguras, etc.)?

Se debe implementar HTTPS, usar cookies seguras, añectar validaciones más robustas y considerar la rotación de sesiones para aumentar la seguridad en un entorno real.

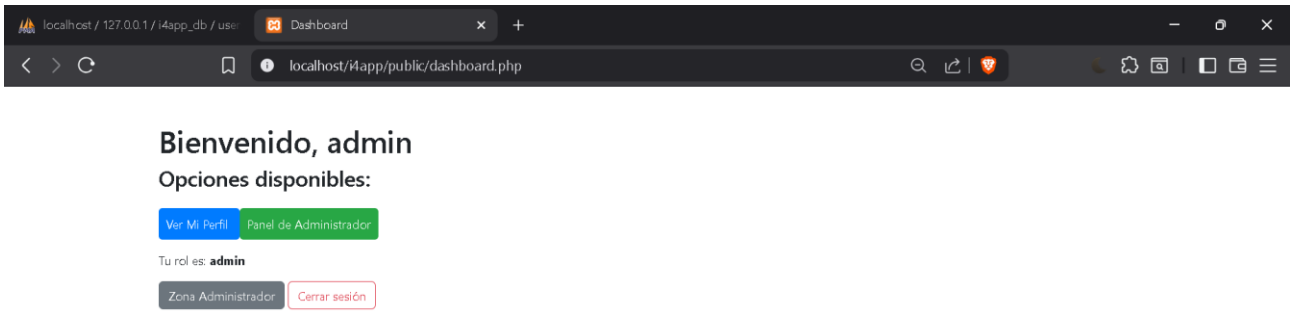

Ilustración 16. dashboard.php modificado

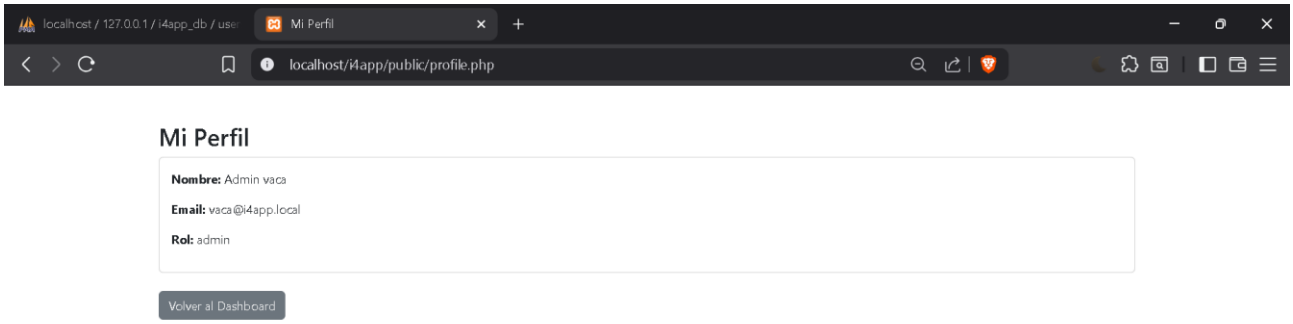

Ilustración 17. vista de profile.php

DECLARATORIA DE USO DE INTELIGENCIA ARTIFICIAL

Para la realización de este ejercicio se utilizó asistencia de IA con el propósito de agilizar el desarrollo y asegurar las mejores prácticas de seguridad en la implementación.