

**“Año del Bicentenario, de la consolidación de nuestra Independencia, y de la
conmemoración de las heroicas batallas de Junín y Ayacucho”**

UNIVERSIDAD NACIONAL DEL ALTIPLANO

**FACULTAD DE INGENIERÍA ESTADÍSTICA E INFORMÁTICA
ESCUELA PROFESIONAL DE INGENIERÍA ESTADÍSTICA E
INFORMÁTICA**



INFORME DE DESARROLLO E IMPLEMENTACIÓN DE LOGIN CON MIDDLEWARE EN CODEIGNITER

PRESENTADO POR:

CRISTIAN DANIEL CCOPA ACERO

CURSO:

SISTEMAS DISTRIBUIDOS / VII Semestre – A

DOCENTE:

Ing. EDSON DENIS ZANABRIA

Puno, noviembre 2024

INDICE

I.	INTRODUCCIÓN	3
II.	MARCO TEÓRICO	4
III.	OBJETIVOS	6
IV.	EQUIPOS Y MATERIALES	7
V.	PROCEDIMIENTOS.....	7
VI.	CONCLUSIONES.....	12
VII.	ANEXOS Y EVIDENCIAS.....	13

I. INTRODUCCIÓN

En el desarrollo de aplicaciones web, es fundamental contar con un sistema de autenticación y autorización que permita gestionar de manera eficiente y segura el acceso de los usuarios. La autenticación asegura que solo usuarios válidos puedan ingresar al sistema, mientras que la autorización se encarga de restringir el acceso a diferentes secciones de la aplicación según los permisos asignados. En este informe se detalla el proceso de desarrollo e implementación de un sistema de login con middleware en CodeIgniter, utilizando una base de datos diseñada previamente que incluye las tablas **usuarios**, **permisos** y **roles**.

La arquitectura de este sistema se basa en tres capas principales: la capa de usuario (autenticación), la capa de roles y permisos (autorización) y la capa de middleware, que actúa como un filtro de acceso a las distintas secciones de la aplicación. El middleware permite interceptar solicitudes y verificar si el usuario está autenticado y si posee los permisos necesarios para acceder a determinadas rutas. Esto se logra mediante el uso de hooks en CodeIgniter, los cuales permiten ejecutar código antes de que se llame al controlador de cada solicitud.

CodeIgniter es un framework de desarrollo web en PHP que destaca por su simplicidad y facilidad de uso, especialmente útil para aplicaciones pequeñas y medianas. Aunque CodeIgniter no tiene un sistema nativo de middleware como otros frameworks (por ejemplo, Laravel), su flexibilidad permite implementar funcionalidades similares mediante configuraciones adicionales. Este informe tiene como objetivo documentar el proceso de desarrollo de un sistema de login y autorización, abordando desde la estructura de la base de datos hasta la implementación del middleware y la configuración de las rutas protegidas.

II. MARCO TEÓRICO

Autenticación y Autorización

La **autenticación** es el proceso de verificar la identidad de un usuario mediante sus credenciales (generalmente un nombre de usuario y contraseña). En esta implementación, la autenticación se realiza a través de un formulario de login donde el usuario ingresa sus credenciales, las cuales se validan contra los datos almacenados en la base de datos. Si las credenciales son correctas, se crea una sesión para el usuario, que le permite acceder a las funciones autorizadas.

La **autorización**, por otro lado, es el proceso de controlar el acceso a los recursos en función de los permisos que tiene un usuario. Un sistema de autorización bien estructurado asigna roles y permisos específicos a cada usuario, lo que facilita la administración del acceso en sistemas con múltiples niveles de acceso. En esta implementación, la autorización se gestiona mediante la asignación de roles y permisos en la base de datos, donde cada usuario tiene un rol, y cada rol tiene permisos específicos para realizar ciertas acciones.

Middleware en Aplicaciones Web

El **middleware** es una capa intermedia en el flujo de ejecución de una aplicación web que permite procesar solicitudes antes de que lleguen al controlador o después de que el controlador haya respondido. En el contexto de autenticación y autorización, el middleware actúa como un filtro que verifica si el usuario tiene los permisos necesarios para acceder a una ruta específica.

En frameworks modernos como Laravel, el middleware es una funcionalidad nativa que facilita la gestión de permisos y otras tareas comunes, como la autenticación y el registro de

actividad. En CodeIgniter, aunque el middleware no está integrado de forma nativa, se puede implementar utilizando **hooks**, una característica que permite ejecutar código en puntos específicos del ciclo de vida de la aplicación, como antes de que se llame al controlador o después de que se cargue una vista. En este proyecto, el middleware implementado mediante hooks se encarga de verificar si el usuario está autenticado y si tiene permisos suficientes para acceder a cada ruta.

CodeIgniter y el Patrón MVC

CodeIgniter es un framework de desarrollo web en PHP que sigue el patrón de diseño **MVC** (Modelo-Vista-Controlador), el cual facilita la separación de lógica, presentación y manipulación de datos en una aplicación. Esta estructura hace que el código sea más modular y fácil de mantener.

- **Modelo:** La capa de Modelo se encarga de gestionar los datos de la aplicación y las operaciones con la base de datos. En este sistema de login, el modelo se utiliza para interactuar con las tablas **usuarios**, **roles** y **permisos** en la base de datos, validando las credenciales y verificando los permisos.
- **Vista:** La capa de Vista gestiona la interfaz de usuario. En este proyecto, la vista incluye el formulario de login y las páginas de error de acceso restringido.
- **Controlador:** La capa de Controlador actúa como intermediario entre el Modelo y la Vista. Recibe las solicitudes del usuario, interactúa con el Modelo para obtener o modificar datos, y luego devuelve los resultados a la Vista.

Base de Datos y Relaciones entre Tablas

En este sistema, la estructura de la base de datos es fundamental para la correcta implementación de los roles y permisos. La base de datos está compuesta por las siguientes tablas:

- **Usuarios:** Esta tabla almacena la información de cada usuario, incluyendo su nombre de usuario, contraseña (en formato hash), y el rol asignado. Cada usuario tiene un rol específico que determina sus permisos dentro de la aplicación.
- **Roles:** Define los diferentes roles que pueden existir en el sistema, como Administrador, Editor, o Usuario. Cada rol tiene permisos específicos asociados.
- **Permisos:** Contiene los diferentes permisos que se pueden asignar a los roles. Un permiso representa una acción específica que puede realizar un usuario, como crear, editar o eliminar publicaciones.
- **Usuarios_has_permisos:** Esta tabla relaciona usuarios específicos con permisos adicionales que pueden necesitar fuera de su rol estándar.
- **Roles_has_permisos:** Define la relación entre los roles y permisos, permitiendo que cada rol tenga permisos específicos asignados.

Esta estructura relacional permite definir permisos detallados para cada rol y usuario, lo cual es esencial para gestionar el acceso en aplicaciones con múltiples niveles de usuario.

III. OBJETIVOS

Objetivo General: Desarrollar e implementar un sistema de autenticación y autorización en CodeIgniter que permita gestionar el acceso de usuarios a través de roles y permisos definidos en una base de datos.

Objetivos Específicos:

Implementar un sistema de login que permita a los usuarios autenticarse en la aplicación.

Configurar un middleware que verifique la autenticación y los permisos de cada usuario.

Definir roles y permisos en la base de datos y asociarlos con los usuarios.

Proteger las rutas del sistema para que solo los usuarios autorizados puedan acceder a ellas.

IV. EQUIPOS Y MATERIALES

- **Servidor en la nube** con CodeIgniter instalado (Azure, DigitalOcean, etc.).
- **PhpMyAdmin** para la administración de la base de datos.
- **MySQL** como sistema de gestión de base de datos.
- **Editor de código** para el desarrollo de los archivos de CodeIgniter (Visual Studio Code, Sublime Text).
- **Navegador web** para pruebas y acceso a la aplicación.

V. PROCEDIMIENTOS**Paso 1: Configuración de la Base de Datos**

1. **Crear la Base de Datos:** Utiliza PhpMyAdmin o una consola de MySQL para crear la base de datos donde se almacenarán las tablas del sistema de login y autorización. Por ejemplo, puedes llamarla sistema_login.

2. **Crear las Tablas:** Crea las tablas **usuarios**, **roles**, **permisos**, **usuarios_has_permisos**, y **roles_has_permisos** en la base de datos, siguiendo la estructura de relaciones del diseño de base de datos.

Ejemplo de comandos SQL para la creación de tablas:

CREATE TABLE Roles (

ID_rols INT **PRIMARY KEY** AUTO_INCREMENT,

nombre_rol VARCHAR(40) **NOT NULL**

);

CREATE TABLE permisos (

idpermisos INT **PRIMARY KEY** AUTO_INCREMENT,

nombre_permiso VARCHAR(40) **NOT NULL**

);

CREATE TABLE Usuarios (

idUsuarios INT **PRIMARY KEY** AUTO_INCREMENT,

usuario VARCHAR(50) **NOT NULL**,

password VARCHAR(200) **NOT NULL**,

nombres VARCHAR(100),

apellidos VARCHAR(100),

email VARCHAR(50),

tipo_usuario TINYINT(1),

Roles_ID_rols INT,

FOREIGN KEY (Roles_ID_rols) **REFERENCES** Roles(ID_rols)

);

CREATE TABLE Usuarios_has_permisos (

Usuarios_idUsuarios INT,

permisos_idpermisos INT,

PRIMARY KEY (Usuarios_idUsuarios, permisos_idpermisos),

FOREIGN KEY (Usuarios_idUsuarios) **REFERENCES** Usuarios(idUsuarios),

FOREIGN KEY (permisos_idpermisos) **REFERENCES** permisos(idpermisos)


```
);
```

```
CREATE TABLE Roles_has_permisos (
    Roles_ID_rol INT,
    permisos_idpermisos INT,
    PRIMARY KEY (Roles_ID_rol, permisos_idpermisos),
    FOREIGN KEY (Roles_ID_rol) REFERENCES Roles(ID_rol),
    FOREIGN KEY (permisos_idpermisos) REFERENCES permisos(idpermisos)
);
```

Poblar la Base de Datos: Ingresa datos de prueba en las tablas de **roles**, **permisos**, y **usuarios** para verificar el sistema de login. Asegúrate de hash las contraseñas al almacenar usuarios.

(Ver Figura A.1 en Anexos: Estructura de la base de datos en PhpMyAdmin)

Paso 2: Configuración del Middleware para Autenticación y Autorización

Habilitar Hooks en CodeIgniter: En el archivo `application/config/config.php`, habilita los hooks para poder usar middleware en CodeIgniter.

```
$config['enable_hooks'] = TRUE;
```

Definir un Hook en CodeIgniter: Configura el archivo de hooks `application/config/hooks.php` para crear un hook que cargue un middleware de autenticación y autorización antes de cada controlador.

```
$hook['pre_controller'][] = [
    'class' => 'AuthMiddleware',
    'function' => 'checkAuth',
    'filename' => 'AuthMiddleware.php',
```

```
'filepath' => 'hooks',
'params' => []
];
```

Crear el Archivo del Middleware AuthMiddleware.php: En la carpeta application/hooks/, crea el archivo AuthMiddleware.php con la función checkAuth, que verificará si el usuario está autenticado y si tiene permisos para acceder a la ruta solicitada.

```
<?php
class AuthMiddleware {
    public function checkAuth() {
        $CI =& get_instance();
        $CI->load->library('session');
        $CI->load->helper('url');

        // Verificar si el usuario está autenticado
        if (!$CI->session->userdata('logged_in')) {
            redirect('auth/login');
        } else {
            // Verificar permisos del usuario
            $CI->load->model('User_model');
            $user_id = $CI->session->userdata('user_id');
            $controller = $CI->uri->segment(1); // Obtiene el nombre del controlador como permiso

            if (!$CI->User_model->tiene_permiso($user_id, $controller)) {
                show_error('No tienes permiso para acceder a esta sección.', 403);
            }
        }
    }
}
```

1. *(Ver Figura A.2 en Anexos: Configuración del hook en CodeIgniter)*

Paso 3: Desarrollo del Modelo User_model

Crear el Modelo User_model: En application/models/, crea User_model.php para gestionar la autenticación de usuarios y la verificación de permisos. Define las funciones autenticar y tiene_permiso.

(Ver Figura A.3 en Anexos: Código del modelo User_model)

Paso 4: Creación del Controlador de Autenticación (Auth)

Crear el Controlador Auth.php: En application/controllers/, crea el controlador Auth.php que maneja el proceso de login y logout de los usuarios. Define las funciones login, autenticar, y logout.

(Ver Figura A.4 en Anexos: Código del controlador Auth)

Paso 5: Creación de la Vista de Login

Crear la Vista login.php: En application/views/, crea login.php, que contiene el formulario de autenticación para los usuarios. Agrega un formulario donde se ingresen el nombre de usuario y la contraseña.

Ejemplo de código:

```
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="UTF-8">
  <title>Login</title>
</head>
```

```

<body>
  <h2>Iniciar Sesión</h2>
  <?php if ($this->session->flashdata('error')): ?>
    <p style="color: red;"><?php echo $this->session->flashdata('error'); ?></p>
  <?php endif; ?>
  <form action="<?php echo site_url('auth/autenticar'); ?>" method="post">
    <label>Usuario:</label>
    <input type="text" name="usuario" required><br>
    <label>Contraseña:</label>
    <input type="password" name="password" required><br>
    <button type="submit">Iniciar Sesión</button>
  </form>
</body>
</html>

```

(Ver Figura A.5 en Anexos: Formulario de login)

Paso 6: Pruebas y Verificación de Autorización

1. **Prueba de Acceso al Sistema:** Utiliza usuarios con diferentes roles y permisos para verificar el correcto funcionamiento del login y middleware.
2. **Validación de Autenticación:** Asegúrate de que los usuarios no autenticados sean redirigidos al formulario de login.
3. **Validación de Autorización:** Comprueba que los usuarios sin permisos para ciertas rutas reciban un mensaje de error 403 cuando intenten acceder a ellas.
4. *(Ver Figura A.6 en Anexos: Pruebas de acceso y autorización)*

VI. CONCLUSIONES

La implementación de un sistema de login con middleware en CodeIgniter permite mejorar la seguridad y gestión de usuarios en la aplicación, asegurando que cada usuario acceda únicamente a las secciones autorizadas según sus roles y permisos.

CodeIgniter, aunque no cuenta con soporte nativo para middleware, ofrece flexibilidad a través de hooks, permitiendo implementar funcionalidades avanzadas de autorización.

La estructuración de la base de datos con las tablas **usuarios**, **roles** y **permisos** facilita la administración de permisos, especialmente en sistemas donde se requiere un control detallado de acceso.

El uso de este sistema de autenticación y autorización mejora la mantenibilidad del código, al separar la lógica de acceso del controlador y centralizarla en el middleware y el modelo de usuarios.

VII. ANEXOS Y EVIDENCIAS

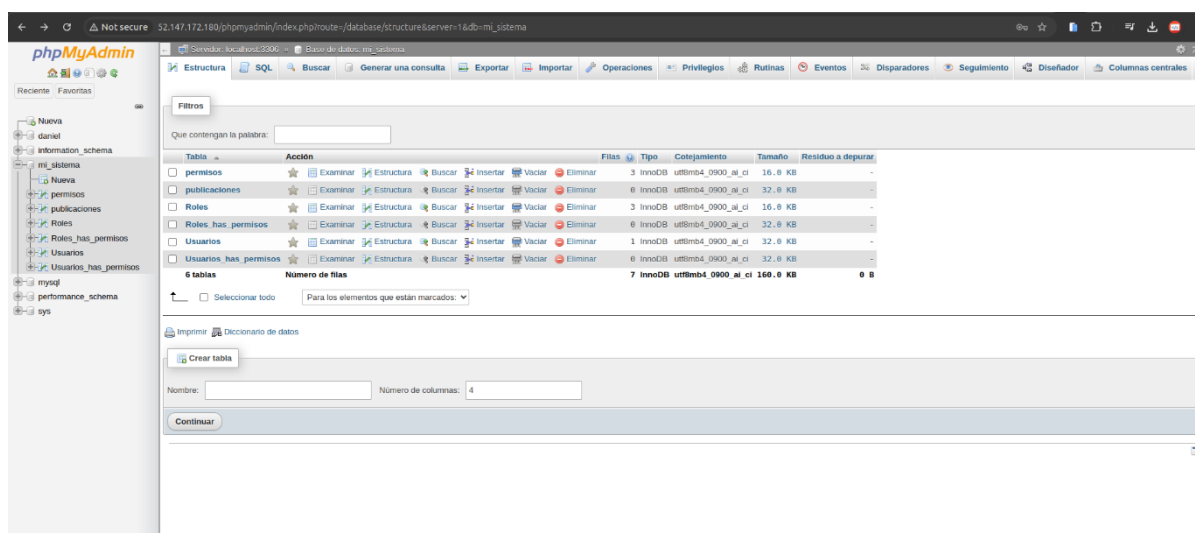
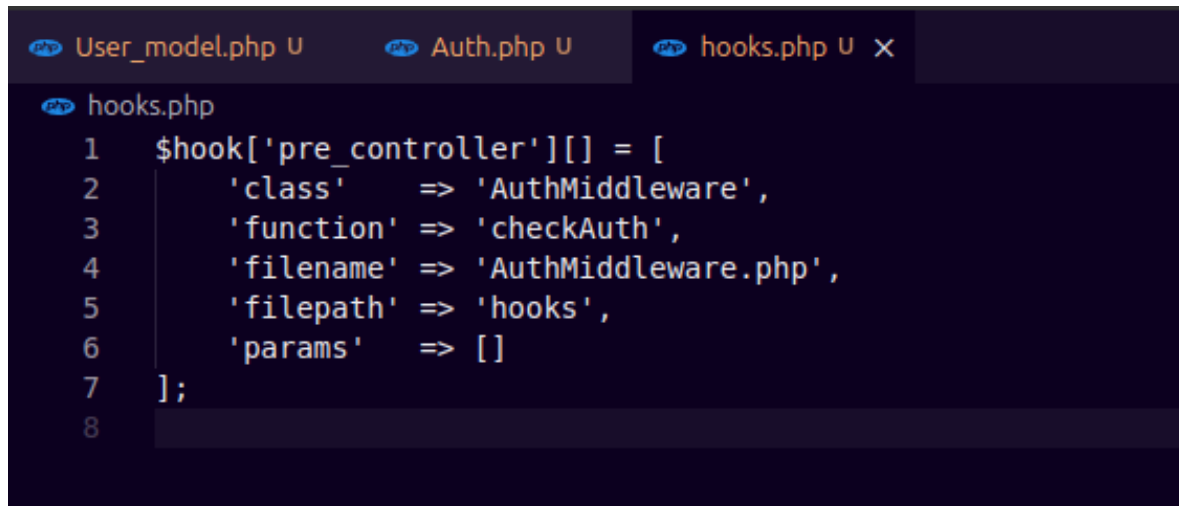


Figura A.1: Estructura de la base de datos en PhpMyAdmin

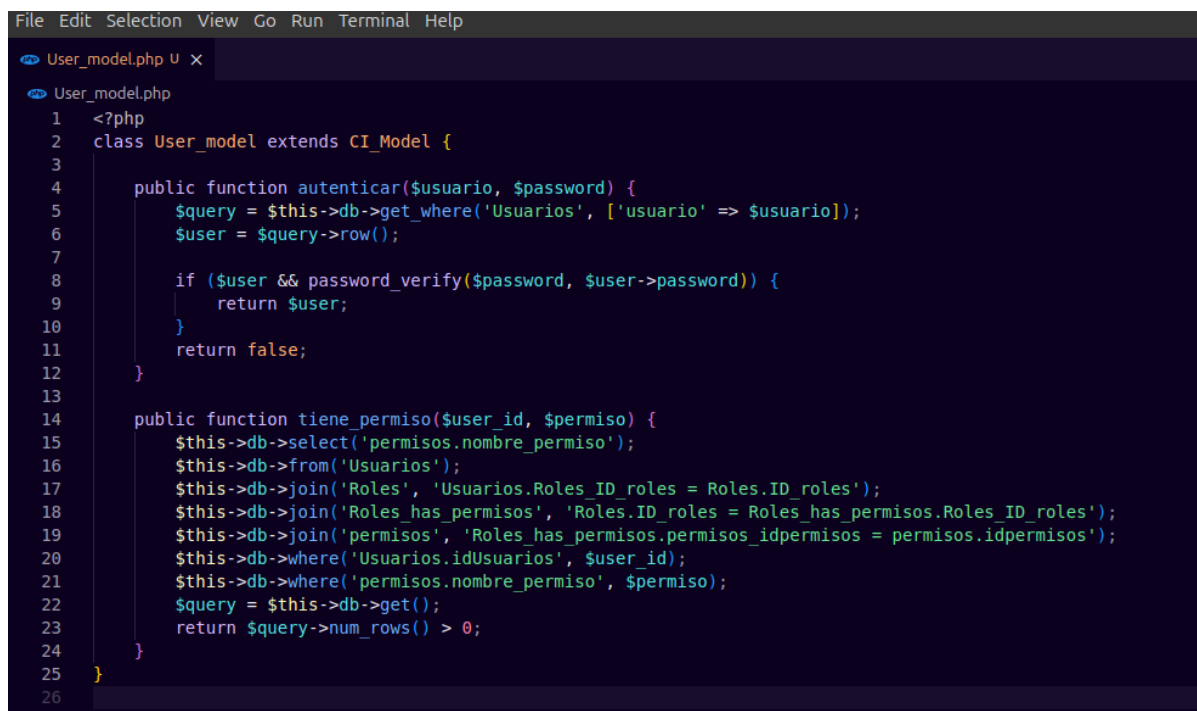
Captura de la base de datos con las tablas **usuarios**, **roles**, **permisos**, **usuarios_has_permisos** y **roles_has_permisos** según el diseño.



```
1 $hook['pre_controller'][] = [
2     'class'    => 'AuthMiddleware',
3     'function' => 'checkAuth',
4     'filename' => 'AuthMiddleware.php',
5     'filepath' => 'hooks',
6     'params'   => []
7 ];
8
```

Figura A.2: Configuración del hook en CodeIgniter

Configuración del hook en el archivo `application/config/hooks.php`, que carga el middleware `AuthMiddleware` para gestionar la autenticación y autorización.



```

File Edit Selection View Go Run Terminal Help
User_model.php U X
User_model.php
1 <?php
2 class User_model extends CI_Model {
3
4     public function autenticar($usuario, $password) {
5         $query = $this->db->get_where('Usuarios', ['usuario' => $usuario]);
6         $user = $query->row();
7
8         if ($user && password_verify($password, $user->password)) {
9             return $user;
10        }
11        return false;
12    }
13
14    public function tiene_permiso($user_id, $permiso) {
15        $this->db->select('permisos.nombre_permiso');
16        $this->db->from('Usuarios');
17        $this->db->join('Roles', 'Usuarios.Roles_ID_rols = Roles.ID_rols');
18        $this->db->join('Roles has permisos', 'Roles.ID_rols = Roles has permisos.Roles_ID_rols');
19        $this->db->join('permisos', 'Roles_has_permisos.permisos_idpermisos = permisos.idpermisos');
20        $this->db->where('Usuarios.idUsuarios', $user_id);
21        $this->db->where('permisos.nombre_permiso', $permiso);
22        $query = $this->db->get();
23        return $query->num_rows() > 0;
24    }
25 }
26

```

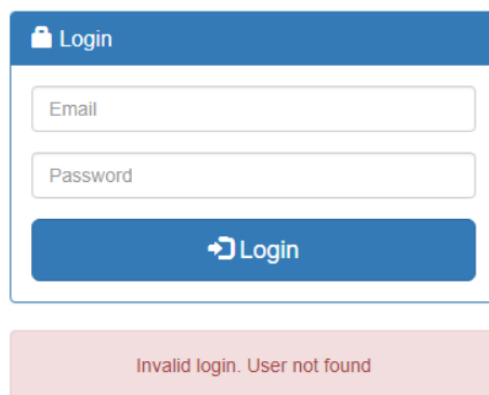
Figura A.3: Código del modelo User_model

Captura del modelo User_model.php, que contiene las funciones autenticar y tiene_permiso para verificar la autenticación y los permisos en la base de datos.

```
User_model.php U Auth.php X
Auth.php
2  class Auth extends CI_Controller {
10  public function login() {
12  }
13
14  public function autenticar() {
15      $usuario = $this->input->post('usuario');
16      $password = $this->input->post('password');
17
18      $user = $this->User_model->autenticar($usuario, $password);
19
20      if ($user) {
21          $this->session->set_userdata([
22              'user_id' => $user->idUsuarios,
23              'logged_in' => true,
24              'rol_id' => $user->Roles_ID_roles
25          ]);
26          redirect('dashboard');
27      } else {
28          $this->session->set_flashdata('error', 'Usuario o contraseña incorrectos');
29          redirect('auth/login');
30      }
31  }
32
33  public function logout() {
34      $this->session->sess_destroy();
35      redirect('auth/login');
36  }
37  }
```

Figura A.4: Código del controlador Auth

Captura del controlador Auth.php, que incluye la lógica de login, logout y autenticación de usuarios.



The image shows a web form titled "Login" with a blue header. Below the header are two input fields: "Email" and "Password". Below these fields is a blue button with a white arrow icon and the text "Login". Below the button is a red error message box that says "Invalid login. User not found".

Figura A.5: Formulario de login

Captura de la vista login.php, donde el usuario ingresa su nombre de usuario y contraseña para autenticarse.

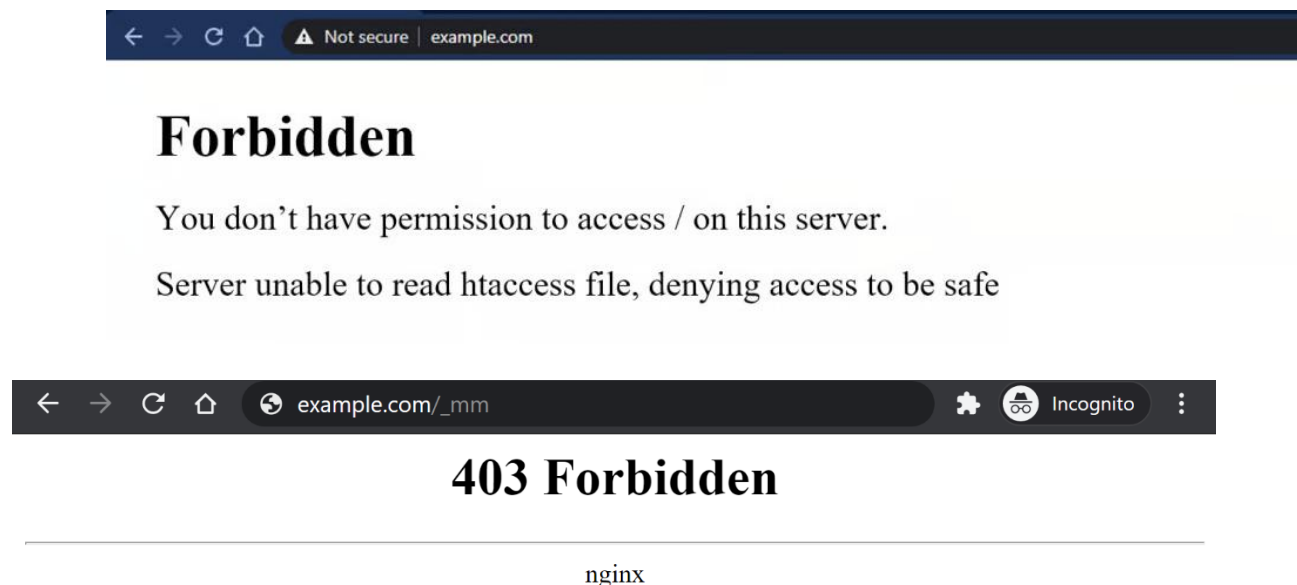


Figura A.6: Pruebas de acceso y autorización

Captura de las pruebas de acceso a diferentes rutas con usuarios con distintos roles y permisos, incluyendo el mensaje de error 403 para usuarios sin autorización.