HITO 2 DEL 2º TRIMESTRE DE Programación

Yordan Rosenov Matov

14/02/2025

ÍNDICE

[1. Página de Inicio: 3](#_Toc190435658)

[2. Registro de Usuarios: 3](#_Toc190435659)

[3. Inicio de Sesión: 4](#_Toc190435660)

[4. CR de Tareas: 5](#_Toc190435661)

[5. Gestión de Sesiones: 6](#_Toc190435662)

[Significado de los archivos: 7](#_Toc190435663)

Imagina que estás desarrollando un sistema de gestión de tareas. Los usuarios podrán

registrarse, iniciar sesión, agregar nuevas tareas, ver sus tareas pendientes, marcar tareas

como completadas y eliminar tareas. Además, se espera que el sitio maneje la autenticación

de usuarios, utilizando cookies y sesiones para garantizar la seguridad y la privacidad de la

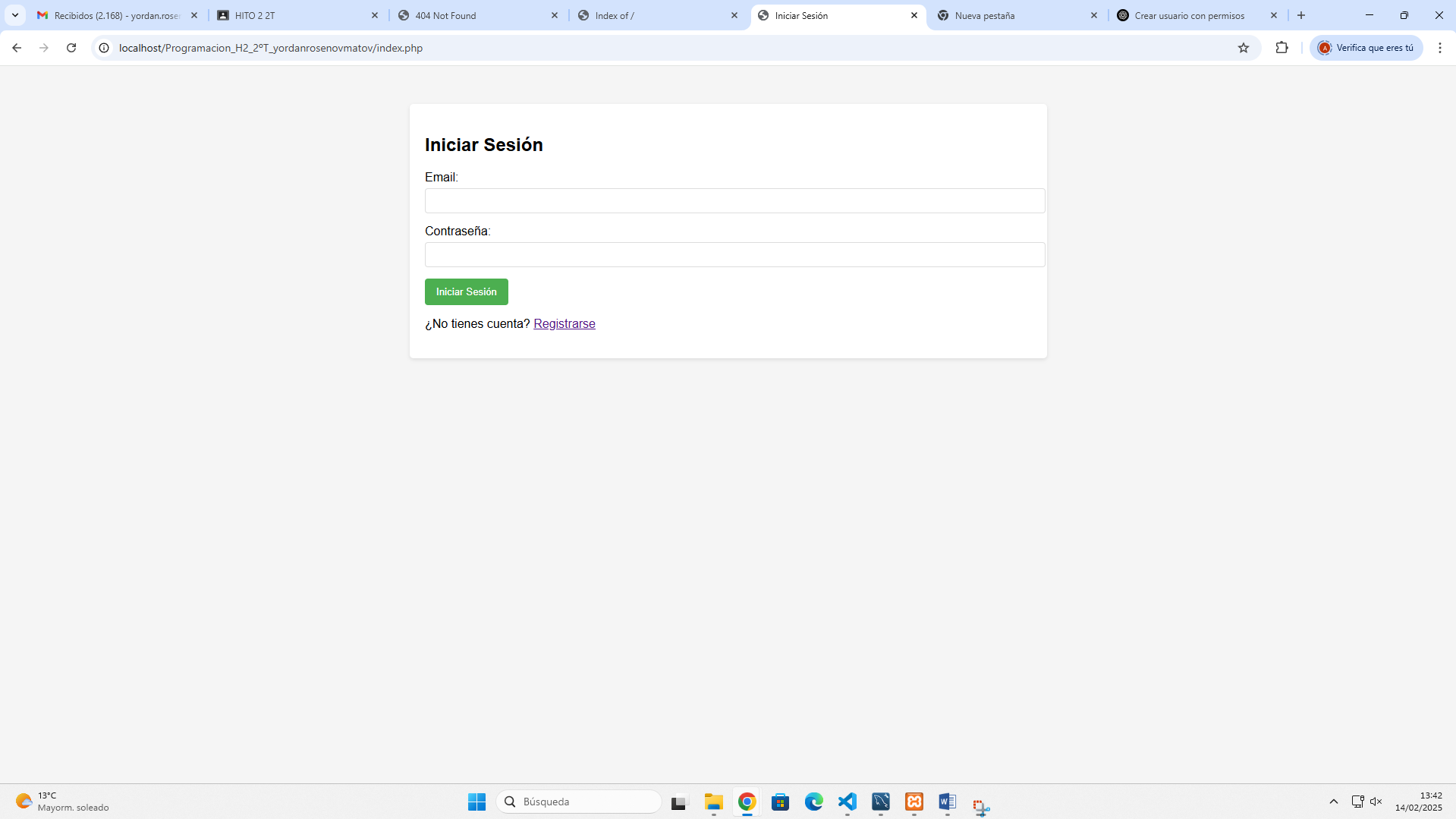
información.

Requisitos:

## 1. Página de Inicio:

- Crear una página de inicio que presente la opción de registro o inicio de sesión.

Primero tenemos una página donde nos sale poder iniciar sesion o si no tenemos registrarnos para poder entrar



## 2. Registro de Usuarios:

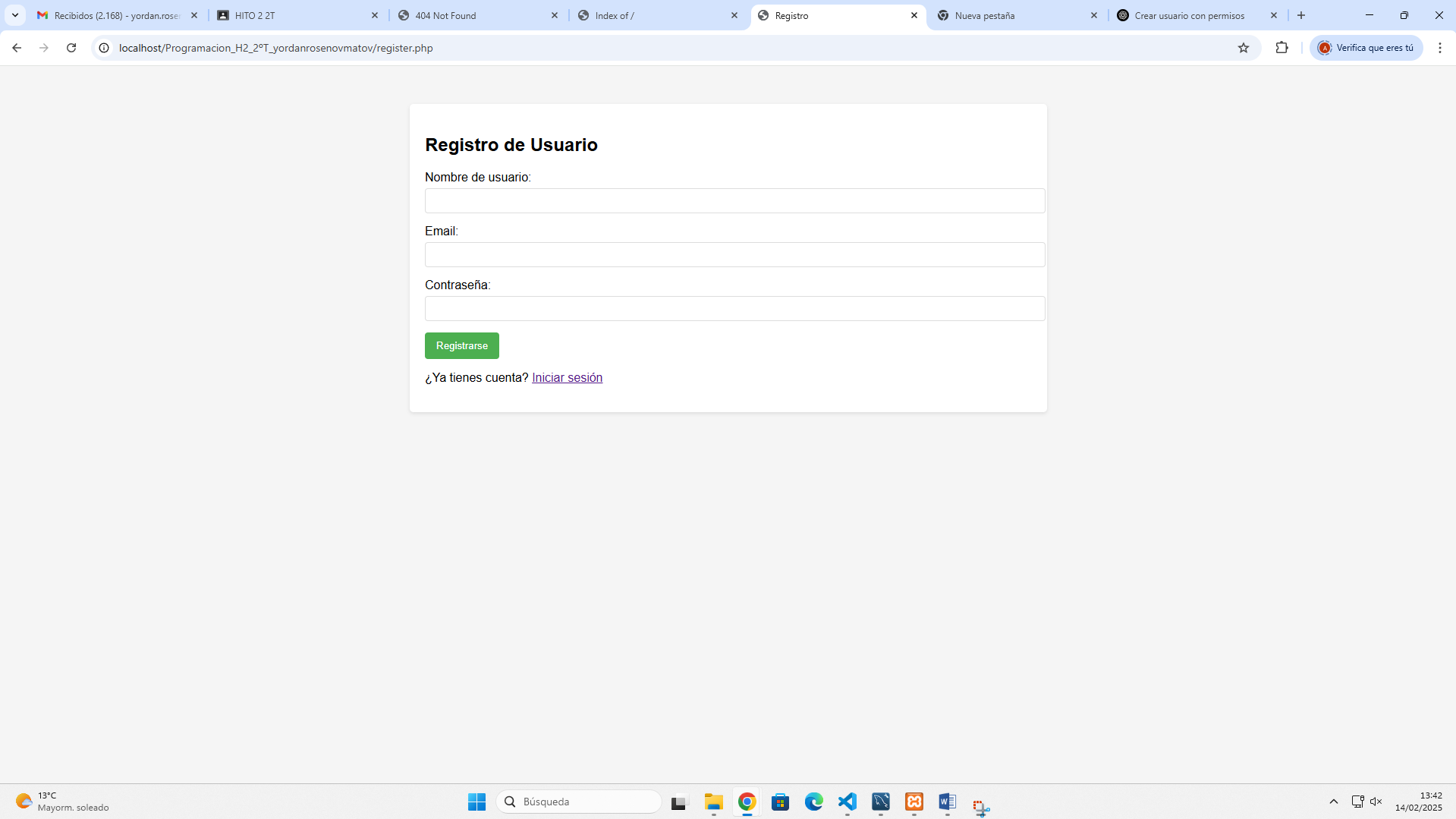
- Implementar un formulario de registro con los campos obligatorios: nombre de usuario,

correo electrónico y contraseña.

- Validar que el correo electrónico sea único en la base de datos.

- Almacenar la información del usuario en la base de datos de MySQL de manera segura

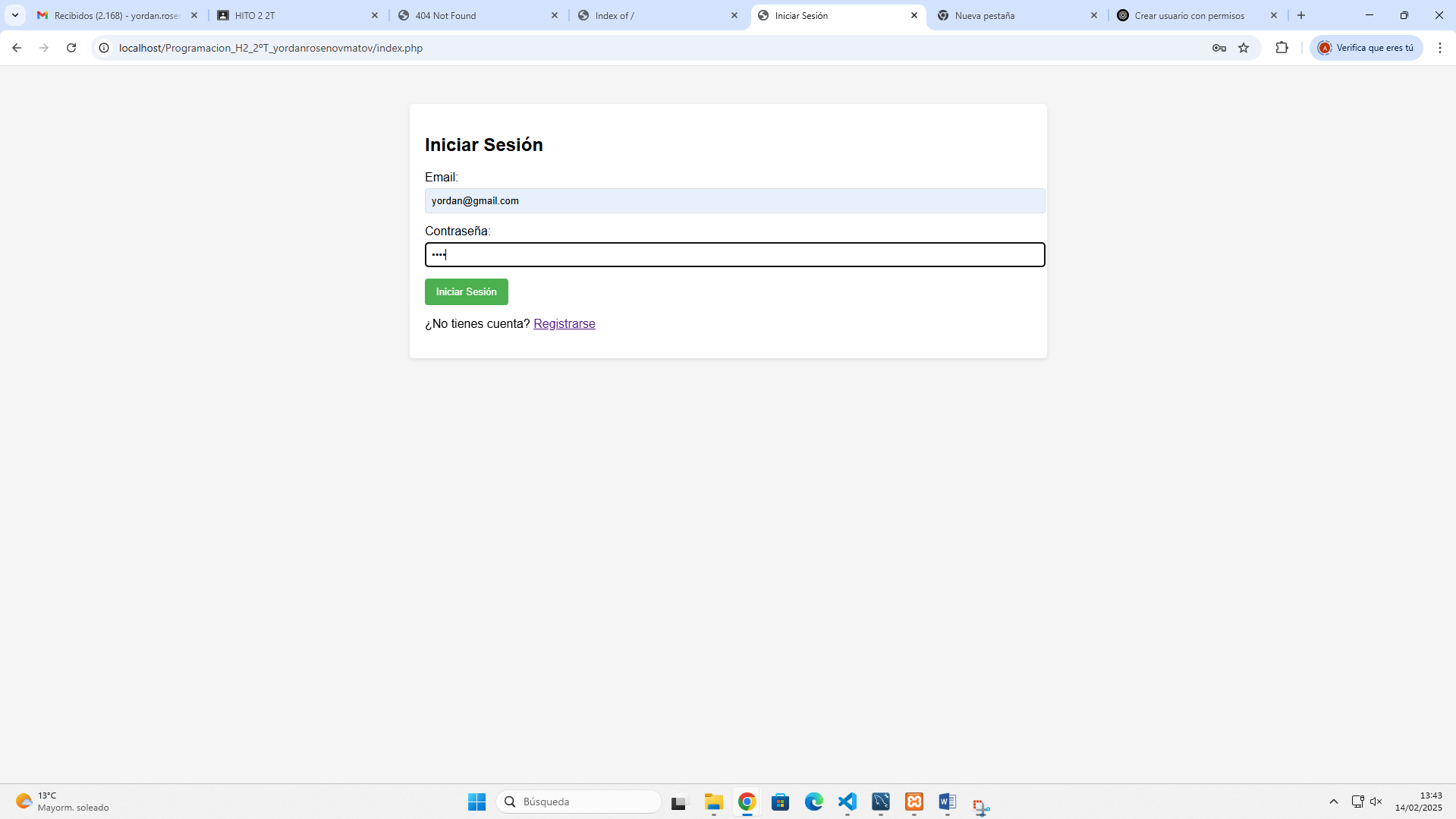
(hash de contraseña).

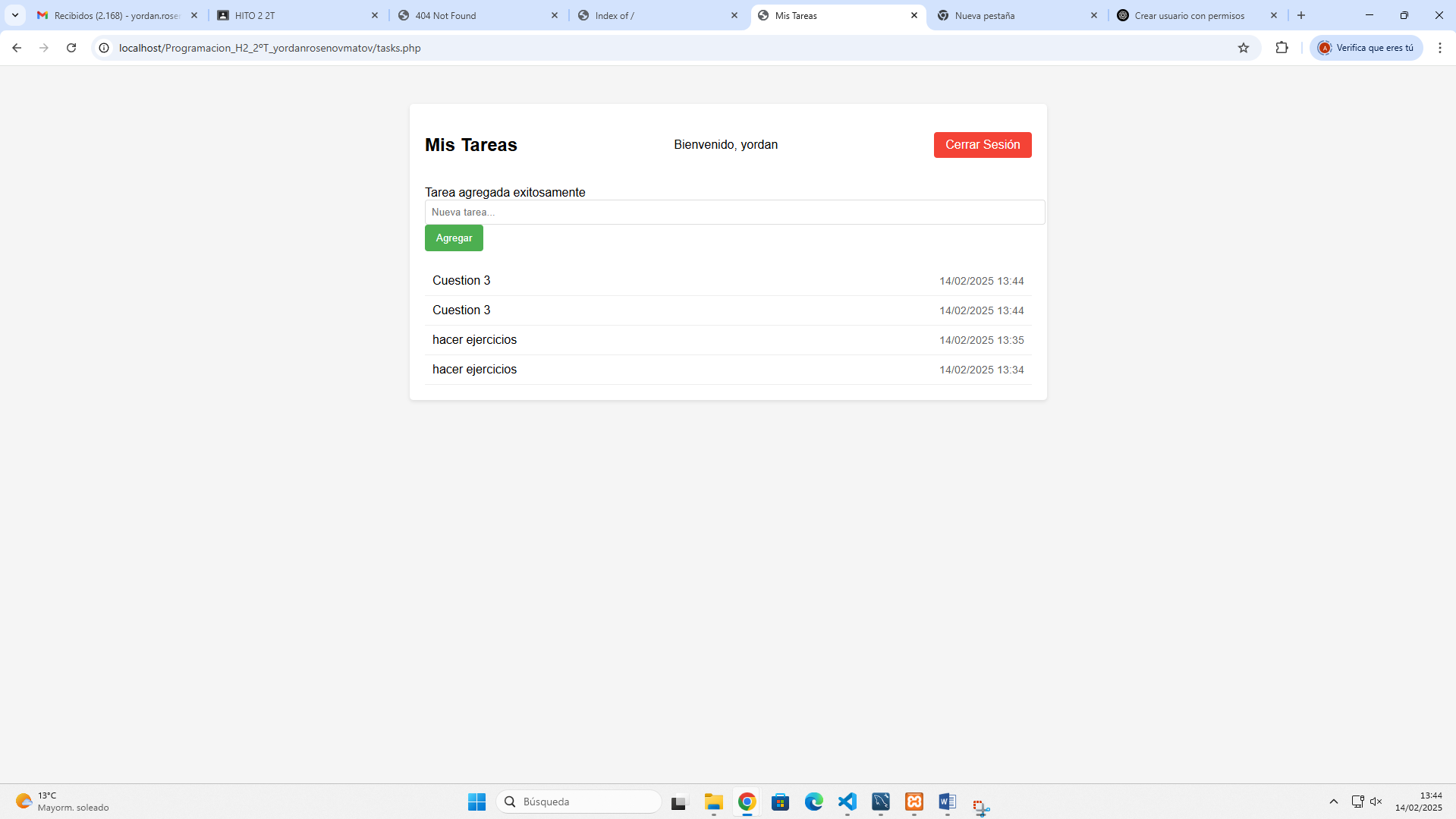


## 3. Inicio de Sesión:

- Crear un formulario de inicio de sesión que valide las credenciales ingresadas.

- Utilizar sesiones para mantener al usuario autenticado durante su sesión.



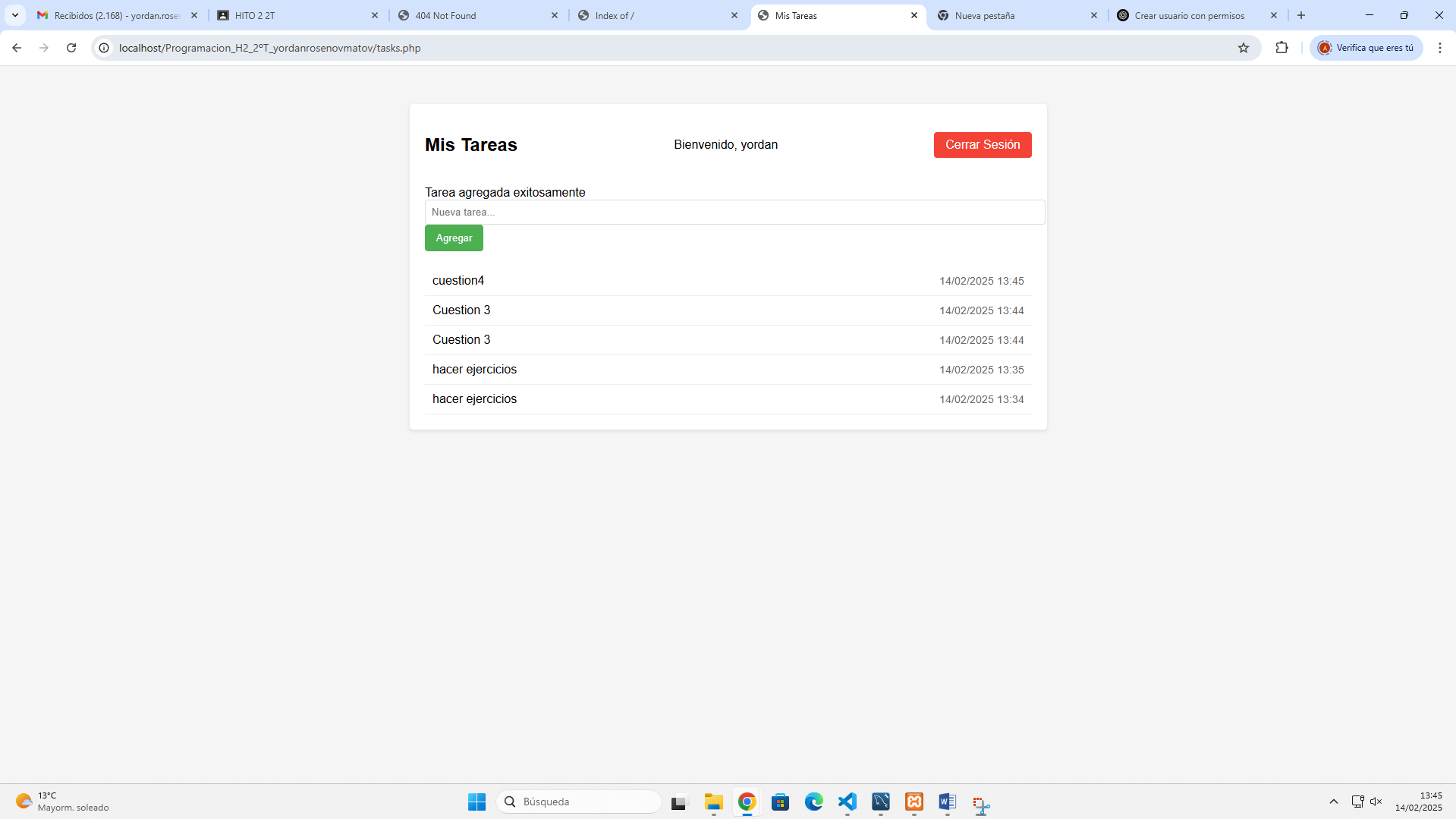


## 4. CR de Tareas:

sólo será visible para los usuarios validados

- Desarrollar una página donde los usuarios puedan agregar nuevas tareas.

- Mostrar el listado de tareas asociados al usuario identificado en el Sistema

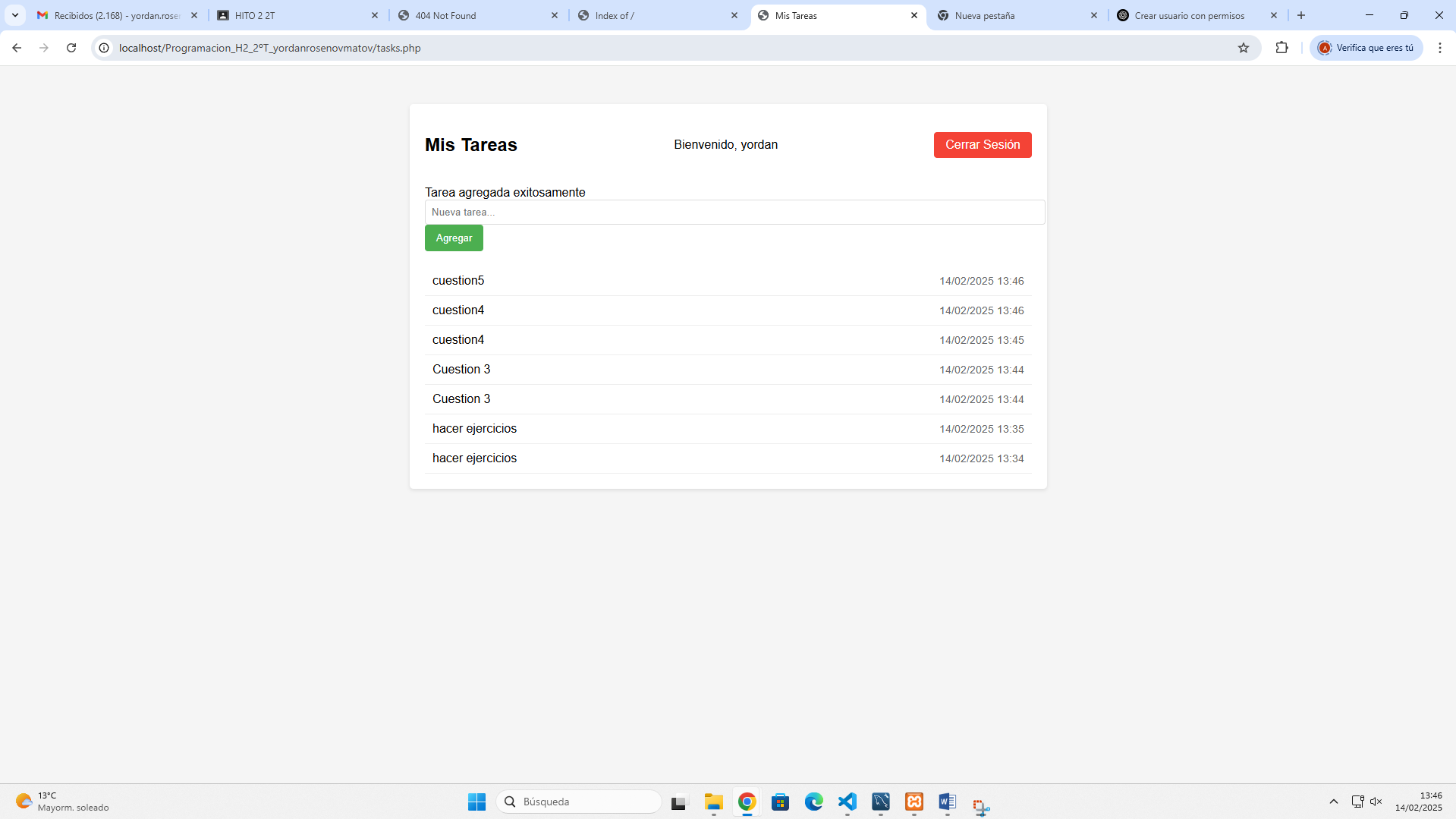


## 5. Gestión de Sesiones:

- Crear un sistema que maneje el cierre de sesión de forma segura.

Deberás controlar y mostrar mensajes al usuario para todas los posibles errores o

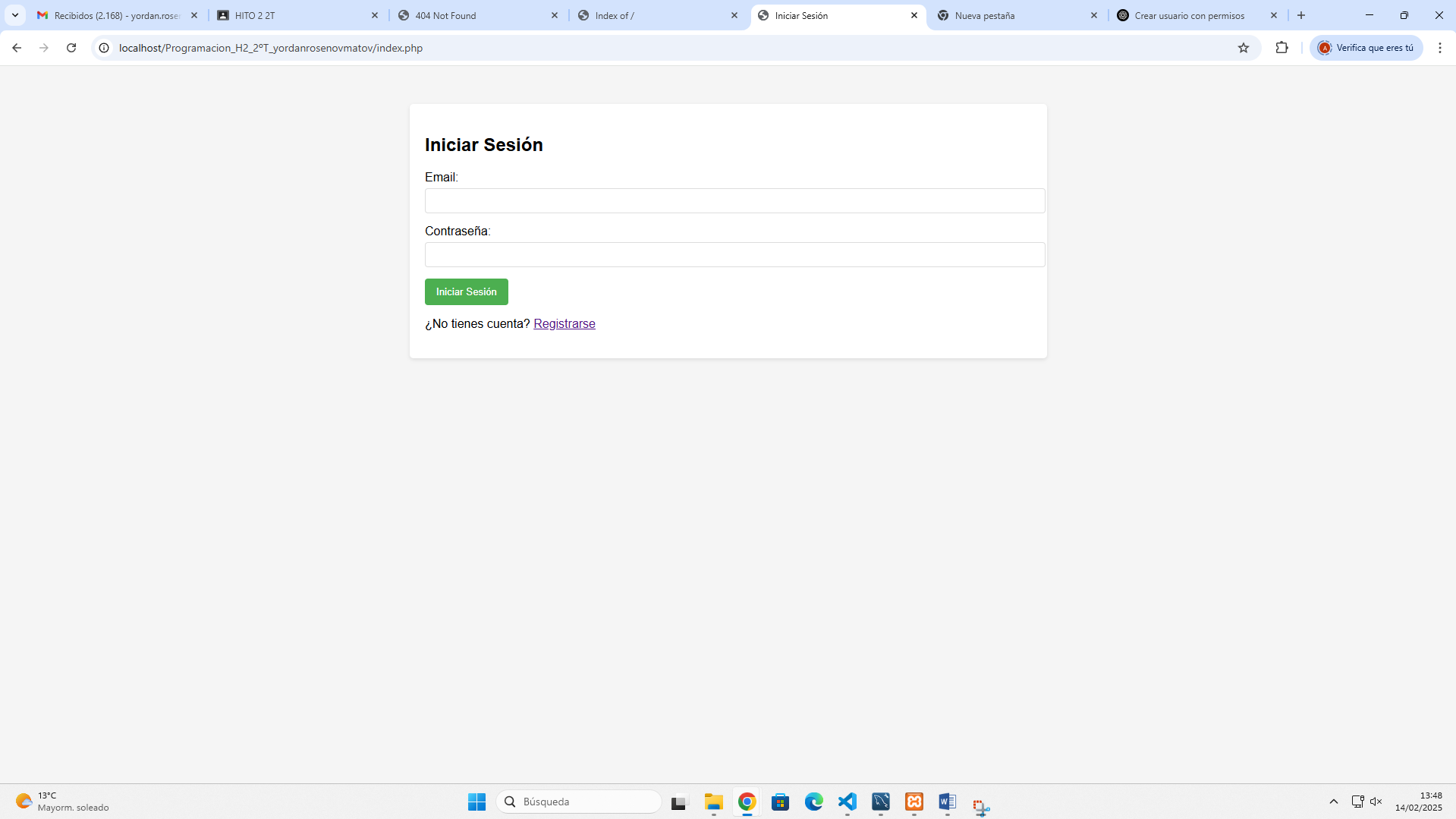
excepciones.



Pinchamos “Cerrar Sesión”



Y al pinchar en cerrar session, volvemos al inicio:



## Significado de los archivos:

-Config.php

* El archivo de la configuración principal.
* Define la conexión a la base de datos.
* Tambien contiene la function connectDB() que nos establece la conexión con MySQL.
* Inicia sesión con PHP.
* Tambien contiene una función checkAuth() para verificar si el usuario está autenticado.
* Tiene la función cleanInput() que limpia los datos de entrada.

<?php

define('DB\_HOST', 'localhost');

define('DB\_USER', 'admin\_task');

define('DB\_PASS', '1234');

define('DB\_NAME', 'task\_manager');

function connectDB() {

    try {

        $db = new PDO(

            "mysql:host=" . DB\_HOST . ";dbname=" . DB\_NAME,

            DB\_USER,

            DB\_PASS,

            [PDO::ATTR\_ERRMODE => PDO::ERRMODE\_EXCEPTION]

        );

        return $db;

    } catch(PDOException $e) {

        die("Error de conexión: " . $e->getMessage());

    }

}

session\_start();

function checkAuth() {

    if (!isset($\_SESSION['user\_id'])) {

        header('Location: index.php');

        exit();

    }

}

function cleanInput($data) {

    return htmlspecialchars(strip\_tags(trim($data)));

}

?>

-Register.php

* Maneja el registro de nuevos usuarios
* Hay un formulario html con campos para nombre de usuario,gmail, y contraseña
* Valida que;

-Los campos no esten vacíos.

-Que el gmail sea válido y único en la base de datos.

-La contraseña cumpla con los requisites.

-transforma la contraseña antes de guardarla.

-Muestra mensajes de éxito o error según corresponda.

<?php

require\_once 'config.php';

$errors = [];

$success = false;

if ($\_SERVER['REQUEST\_METHOD'] == 'POST') {

    $username = cleanInput($\_POST['username']);

    $email = cleanInput($\_POST['email']);

    $password = $\_POST['password'];

    if (empty($username)) $errors[] = "El nombre de usuario es obligatorio";

    if (empty($email)) $errors[] = "El email es obligatorio";

    if (empty($password)) $errors[] = "La contraseña es obligatoria";

    if (!filter\_var($email, FILTER\_VALIDATE\_EMAIL)) $errors[] = "Email inválido";

    if (empty($errors)) {

        $db = connectDB();

        $stmt = $db->prepare("SELECT id FROM users WHERE email = ?");

        $stmt->execute([$email]);

        if ($stmt->rowCount() > 0) {

            $errors[] = "Este email ya está registrado";

        } else {

            $hashedPassword = password\_hash($password, PASSWORD\_DEFAULT);

            $stmt = $db->prepare("INSERT INTO users (username, email, password) VALUES (?, ?, ?)");

            try {

                $stmt->execute([$username, $email, $hashedPassword]);

                $success = true;

            } catch(PDOException $e) {

                $errors[] = "Error al registrar usuario: " . $e->getMessage();

            }

        }

    }

}

?>

<!DOCTYPE html>

<html>

<head>

    <title>Registro</title>

    <link rel="stylesheet" href="style.css">

</head>

<body>

    <div class="container">

        <h2>Registro de Usuario</h2>

        <?php if ($success): ?>

            <div class="success">

                ¡Registro exitoso! <a href="index.php">Iniciar sesión</a>

            </div>

        <?php endif; ?>

        <?php if (!empty($errors)): ?>

            <div class="errors">

                <?php foreach($errors as $error): ?>

                    <p><?php echo $error; ?></p>

                <?php endforeach; ?>

            </div>

        <?php endif; ?>

        <form method="POST" action="">

            <div class="form-group">

                <label>Nombre de usuario:</label>

                <input type="text" name="username" required>

            </div>

            <div class="form-group">

                <label>Email:</label>

                <input type="email" name="email" required>

            </div>

            <div class="form-group">

                <label>Contraseña:</label>

                <input type="password" name="password" required>

            </div>

            <button type="submit">Registrarse</button>

        </form>

        <p>¿Ya tienes cuenta? <a href="index.php">Iniciar sesión</a></p>

    </div>

</body>

</html>

Index.phpç

* Gestiona la auntenticación de usuarios
* Presenta un formulario con campos para email y contraseña
* Verifica las credenciales contra la base de datos
* Si son correctas

-Crea una session para el usuario

-Almacena el ID y usa el nombre de usuario en variables de sesion

-Redirige a tasks.php

-Si son incorrectas, muestra un mensaje de error

<?php

require\_once 'config.php';

$errors = [];

if ($\_SERVER['REQUEST\_METHOD'] == 'POST') {

    $email = cleanInput($\_POST['email']);

    $password = $\_POST['password'];

    if (empty($email) || empty($password)) {

        $errors[] = "Todos los campos son obligatorios";

    } else {

        $db = connectDB();

        $stmt = $db->prepare("SELECT id, password, username FROM users WHERE email = ?");

        $stmt->execute([$email]);

        if ($user = $stmt->fetch()) {

            if (password\_verify($password, $user['password'])) {

                $\_SESSION['user\_id'] = $user['id'];

                $\_SESSION['username'] = $user['username'];

                header('Location: tasks.php');

                exit();

            } else {

                $errors[] = "Credenciales incorrectas";

            }

        } else {

            $errors[] = "Credenciales incorrectas";

        }

    }

}

?>

<!DOCTYPE html>

<html>

<head>

    <title>Iniciar Sesión</title>

    <link rel="stylesheet" href="style.css">

</head>

<body>

    <div class="container">

        <h2>Iniciar Sesión</h2>

        <?php if (!empty($errors)): ?>

            <div class="errors">

                <?php foreach($errors as $error): ?>

                    <p><?php echo $error; ?></p>

                <?php endforeach; ?>

            </div>

        <?php endif; ?>

        <form method="POST" action="">

            <div class="form-group">

                <label>Email:</label>

                <input type="email" name="email" required>

            </div>

            <div class="form-group">

                <label>Contraseña:</label>

                <input type="password" name="password" required>

            </div>

            <button type="submit">Iniciar Sesión</button>

        </form>

        <p>¿No tienes cuenta? <a href="register.php">Registrarse</a></p>

    </div>

</body>

</html>

Tasks.php

* Es la página principal después del index..
* Verifica que el usuario esté autenticado.
* Pérmite:

-Ver el listado de tareas del usuario actual.

-Agregar nuevas tareas.

-Ver la fecha de creación de cada tarea

* Muestra el nombre del usuario y opción de cerrar sesión
* Las tareas se ordenan por fecha de creación (Más recientes primero)

<?php

require\_once 'config.php';

checkAuth();

$db = connectDB();

$user\_id = $\_SESSION['user\_id'];

$message = '';

if ($\_SERVER['REQUEST\_METHOD'] == 'POST' && isset($\_POST['add\_task'])) {

    $task\_title = cleanInput($\_POST['task\_title']);

    if (!empty($task\_title)) {

        $stmt = $db->prepare("INSERT INTO task (user\_id, title, created\_at) VALUES (?, ?, NOW())");

        try {

            $stmt->execute([$user\_id, $task\_title]);

            $message = "Tarea agregada exitosamente";

        } catch(PDOException $e) {

            $message = "Error al agregar la tarea: " . $e->getMessage();

        }

    }

}

$stmt = $db->prepare("SELECT \* FROM task WHERE user\_id = ? ORDER BY created\_at DESC");

$stmt->execute([$user\_id]);

$tasks = $stmt->fetchAll();

?>

<!DOCTYPE html>

<html>

<head>

    <title>Mis Tareas</title>

    <link rel="stylesheet" href="style.css">

</head>

<body>

    <div class="container">

        <div class="header">

            <h2>Mis Tareas</h2>

            <p>Bienvenido, <?php echo htmlspecialchars($\_SESSION['username']); ?></p>

            <a href="logout.php" class="logout-btn">Cerrar Sesión</a>

        </div>

        <?php if (!empty($message)): ?>

            <div class="message">

                <?php echo $message; ?>

            </div>

        <?php endif; ?>

        <form method="POST" action="" class="task-form">

            <div class="form-group">

                <input type="text" name="task\_title" placeholder="Nueva tarea..." required>

                <button type="submit" name="add\_task">Agregar</button>

            </div>

        </form>

        <div class="tasks-list">

            <?php if (empty($tasks)): ?>

                <p>No hay tareas pendientes.</p>

            <?php else: ?>

                <?php foreach($tasks as $task): ?>

                    <div class="task-item">

                        <span><?php echo htmlspecialchars($task['title']); ?></span>

                        <span class="task-date">

                            <?php echo date('d/m/Y H:i', strtotime($task['created\_at'])); ?>

                        </span>

                    </div>

                <?php endforeach; ?>

            <?php endif; ?>

        </div>

    </div>

</body>

</html>

Logout.php

* Maneja el cierre de sesión
* Destruye la sesión actual
* Redirige al usuario a la página de login

<?php

require\_once 'config.php';

session\_destroy();

header('Location: index.php');

exit();

?>

Style.css

* Contiene todos los estilos del sistema
* Define el diseño responsivo
* Incluye estilos para:

-Formularios

-Mensajes de error y éxito

-Lista de tareas

-Botones y enlaces

-Layouts y contenedores

body {

    font-family: Arial, sans-serif;

    margin: 0;

    padding: 0;

    background-color: #f5f5f5;

}

.container {

    max-width: 800px;

    margin: 50px auto;

    padding: 20px;

    background-color: white;

    border-radius: 5px;

    box-shadow: 0 2px 5px rgba(0,0,0,0.1);

}

.header {

    display: flex;

    justify-content: space-between;

    align-items: center;

    margin-bottom: 20px;

}

.form-group {

    margin-bottom: 15px;

}

.form-group label {

    display: block;

    margin-bottom: 5px;

}

.form-group input {

    width: 100%;

    padding: 8px;

    border: 1px solid #ddd;

    border-radius: 4px;

}

button {

    background-color: #4CAF50;

    color: white;

    padding: 10px 15px;

    border: none;

    border-radius: 4px;

    cursor: pointer;

}

button:hover {

    background-color: #45a049;

}

.errors {

    background-color: #ffebee;

    color: #c62828;

    padding: 10px;

    margin-bottom: 15px;

    border-radius: 4px;

}

.success {

    background-color: #e8f5e9;

    color: #2e7d32;

    padding: 10px;

    margin-bottom: 15px;

    border-radius: 4px;

}

.task-form {

    margin-bottom: 20px;

}

.tasks-list {

    margin-top: 20px;

}

.task-item {

    padding: 10px;

    border-bottom: 1px solid #eee;

    display: flex;

    justify-content: space-between;

    align-items: center;

}

.task-date {

    color: #666;

    font-size: 0.9em;

}

.logout-btn {

    background-color: #f44336;

    color: white;

    padding: 8px 15px;

    text-decoration: none;

    border-radius: 4px;

}

.logout-btn:hover {

    background-color: #d32f2f;

}