Auditoría de seguridad a un entorno realista simulado

Trabajo de Fin de Grado

GRADO EN INGENIERÍA INFORMÁTICA

Anexo IV. Documentación Técnica

Julio de 2023

Autor

Marcos Panero Calles

Tutor/a

Ángel Luis Sánchez Lázaro

Tabla de contenido

Tabla de Ilustraciones

1. Introducción

El presente anexo tiene como objetivo proporcionar una explicación detallada de los diferentes ficheros y funciones del proyecto. Su propósito principal es servir como una guía y ayuda al programador durante el desarrollo y mantenimiento del sistema.

En este documento se encontrarán descripciones exhaustivas de cada fichero y función, incluyendo su propósito, parámetros de entrada y salida, y la lógica subyacente. Además, se brindará una guía de instalación de las diferentes tecnologías utilizadas en la implementación de este proyecto.

Como ya se mencionó en la memoria complementaria a este anexo el objetivo principal de esta práctica no es la implementación de la aplicación en sí, sino la investigación, se ha optado por utilizar una plantilla de estilo gratuita obtenida de internet para el diseño visual de la aplicación. La utilización de esta plantilla se hará de manera responsable y se dejará debidamente referenciada en la sección de bibliografía del proyecto.

Además, en el desarrollo de la aplicación se ha utilizado una API externa para el envío de mensajes SMS. Esta API proporciona la funcionalidad necesaria para enviar mensajes de texto a través de un proveedor de servicios externo.

2. Guía de instalación

Como ya se ha mencionado en la memoria complementaria a este anexo la configuración de tecnologías instaladas es una configuración LAMP. Se trata de un conjunto de tecnologías que se combinan para proporcionar una solución integral para el desarrollo y despliegue de sitios web dinámicos. En la estructura LAMP, cada componente tiene un rol específico:

- Linux: Es el sistema operativo utilizado como base para el entorno de desarrollo. En mi caso particular como sistema operativo se ha utilizado un Ubuntu 22.04.2 LTS-
- Apache: Es el servidor web utilizado para entregar las páginas web al navegador del usuario. Se ha instalado la versión Apache 2.4.52 Web Server
- MySQL: Es un sistema de gestión de bases de datos relacional. La versión instalada es la 8.0.3.
- PHP: Es un lenguaje de programación de servidor utilizado para crear aplicaciones web dinámicas. La versión instalada de PHP es la 8.1.2.

2.1 Instalación de Apache

Para comenzar, es recomendable actualizar el caché del administrador de paquetes antes de instalar Apache. Esto garantiza que se utilicen las versiones más recientes de los paquetes disponibles. Para actualizar el caché, se puede ejecutar el siguiente comando utilizando el comando sudo para obtener los privilegios de administrador:

```
$ sudo apt update
```

Al ejecutar este comando, se le puede solicitar que proporcione su contraseña de usuario para confirmar que tiene los privilegios correctos para administrar los paquetes del sistema utilizando el administrador de paquetes apt.

Una vez que el caché del administrador de paquetes esté actualizado, se puede proceder a la instalación de Apache. Para ello, se utiliza el siguiente comando:

```
$ sudo apt install apache2
```

Al ejecutar este comando, se le pedirá que confirme la instalación de Apache. Para confirmar, simplemente se debe presionar Y y luego presionar ENTER. Esto iniciará el proceso de instalación de Apache en el sistema.

Cuando el proceso hay finalizado puede verificar que la instalación concluyó correctamente visitando la dirección IP de su servidor en su navegador. Debería de ver algo similar a la siguiente captura de pantalla.

Ilustración 1: Página por defecto de Apache2

2.2 Instalar MySQL

A continuación, se describirán los pasos para instalar y configurar el sistema de base de datos MySQL en su servidor web.

Una vez que su servidor web está en funcionamiento, es necesario instalar el sistema de base de datos para poder almacenar y administrar los datos de su sitio. MySQL es una opción popular de sistema de gestión de bases de datos, especialmente en entornos PHP.

Para instalar MySQL, se utiliza el comando apt en el terminal:

```
$ sudo apt install mysql-server
```

Al ejecutar este comando, se le solicitará que confirme la instalación escribiendo "Y" y luego presionando ENTER.

Después de la instalación, se recomienda ejecutar un script de seguridad preinstalado que viene con MySQL. Este script eliminará algunas configuraciones predeterminadas inseguras y bloqueará el acceso no autorizado a su sistema de base de datos.

Para ejecutar el script interactivo, se utiliza el siguiente comando:

```
$ sudo mysql_secure_installation
```

Independientemente de si habilitó el complemento de validación de contraseña, se le pedirá que seleccione y confirme una contraseña para el usuario "root" de MySQL.

Si habilitó la validación de contraseña, se le mostrará la fortaleza estimada de la contraseña que ingresó y se le preguntará si desea continuar con esa contraseña. Si está satisfecho con la contraseña actual, puede ingresar "Y" para confirmar.

A continuación, se le presentarán algunas otras preguntas durante el proceso de seguridad, y se recomienda responder "Y" y presionar ENTER en cada indicación. Esto eliminará algunos usuarios anónimos y la base de datos de prueba, deshabilitará los inicios de sesión remotos para el usuario "root" y aplicará las nuevas reglas de seguridad de inmediato.

Una vez finalizado el proceso, puede probar si puede iniciar sesión en la consola de MySQL escribiendo el siguiente comando:

```
$ sudo mysql
```

Esto iniciará una sesión en el servidor MySQL como el usuario "root" de la base de datos administrativa.

2.3 Instalación de PHP

Una vez que ha instalado Apache para servir contenido web y MySQL para almacenar y administrar datos en su servidor, el siguiente paso es configurar PHP, que es el componente encargado de procesar el código y mostrar contenido dinámico a los usuarios finales. Para permitir la comunicación entre PHP y bases de datos MySQL, necesitará instalar el módulo php-mysql. Además, deberá habilitar el módulo libapache2-mod-php en Apache para que pueda manejar archivos PHP. Los paquetes principales de PHP se instalarán automáticamente como dependencias.

Para llevar a cabo la instalación de estos paquetes, ejecute el siguiente comando:

```
$ sudo apt install php libapache2-mod-php php-mysql
```

Una vez finalizada la instalación, puede verificar la versión de PHP ejecutando el siguiente comando:

```
$ php -v
```

3. Estructura del sistema

Dentro de la carpeta UsalCoins podremos encontrar todos los directorios y ficheros necesarios para la implementación de la aplicación web. El contenido de esta carpeta es una copia idéntica al contenido del directorio raíz del servidor Apache implementado para este proyecto. El cual esta estructurado de la siguiente forma:

/css: Directorio que contiene los ficheros de estilos utilizados por la aplicación.
/img: En este directorio se encuentran todas las imágenes que podemos encontrar navegando por las diferentes páginas de la web.
/js: Contiene un script en JavaScript el cual permite implementar las diferentes animaciones de la aplicación.
/tareas: Este será el directorio donde se almacenarán las distintas tareas subidas por los usuarios. Cada usuario tendrá una carpeta personal con su nombre.
/vendor: En este directorio están contenidos los ficheros de configuración necesarios para el uso de la API de envío de mensajes SMS.

Junto a estos directorios encontraremos varios scripts en PHP los cuales implementan tanto las vistas como las funciones lógicas que componen el funcionamiento de la aplicación. El contenido y la utilidad de cada uno de estos se explicará detalladamente en el siguiente apartado.

4. Documentación

En este apartado se explicarán, adjuntado capturas, todas las funciones lógicas que componen la aplicación. No se entrará en detalle sobre las vistas HTML ni las funciones utilizadas para crear las animaciones o dar estilo a la web debido a que se consideran redundantes ya que han sido implementadas por otra persona.

4.1 admin.php

Este script será el encargado de cargar la vista del área privada para los usuarios administradores. La primera función lógica que encontramos en este es la que aparece en la siguiente captura, la cual inicia las cookies de sesión, controla que el usuario que intenta acceder a la vista tiene permisos de acceso, y muestra el mensaje de confirmación tras realizar un nuevo alta de usuarios.

```php
<?php
  session_start();

  // Verificar si el usuario ha iniciado sesión
  if (!isset($_SESSION['loggedin']) || $_SESSION['loggedin'] !== true) {
    $_SESSION['error_message'] = '¡Vaya! Parece que no tienes permisos.<br>Este incidente será reportado.';
    $_SESSION['reporte'] = "admin.php";
    include'reportar.php';
    header("Location: login.php");
    exit;
  }
  if (!isset($_SESSION['rol']) || $_SESSION['rol'] !== '0') {
    $_SESSION['error_message'] = '¡Vaya! Parece que no tienes permisos.<br>Este incidente será reportado.';
    $_SESSION['reporte'] = "admin.php";
    include'reportar.php';
    header("Location: login.php");
    exit;
  }
  $alta_mensaje = isset($_SESSION['alta_mensaje']) ? $_SESSION['alta_mensaje'] : '';
  if (isset($_SESSION['alta_mensaje'])) {
          unset($_SESSION['alta_mensaje']);
  }
  if ($alta_mensaje !== '') {
    echo "<script>alert('$alta_mensaje');</script>";;
  }
?>
```

Ilustración 2: Captura de admin.php I

Si seguimos leyendo el código encontraremos un pequeño script en JavaScript el cual es el encargado de implementar la lógica del selector de rol situado en la esquina superior derecha de la página.

```
    <script>
    $(document).ready(function() {
        $('#selectorRol').on('change', function() {
            var selectedOption = $(this).val();

            $.ajax({
                type: 'POST',
                url: 'admin.php',
                data: { option: selectedOption },
                success: function(response) {
                    // Hacer algo con la respuesta del servidor si es necesario

                    if (selectedOption === 'admin') {
                        window.location.href = 'admin.php';
                    } else if (selectedOption === 'profe') {
                        window.location.href = 'profesor.php';
                    } else if (selectedOption === 'alumno') {
                        window.location.href = 'alumno.php';
                    }
                }
            });
        });
    });
</script>
```

Ilustración 3: Captura de admin.php II

Por último, al igual que en el resto de las áreas privadas de la plataforma en el encabezado de la sección encontramos un pequeño script que permite mostrar el nombre del usuario y los UsalCoins que tiene acumulados.

```
<?php
    $nombre = $_SESSION['nombre'];
    $puntos = $_SESSION['puntos'];
    echo "<h1 class='display-2 mb-4'>¡Hola, $nombre!</h1>";
    echo "<h2 class='display-6 mb-10' data-aos='fade-down'>Tienes $puntos UsalCoins</h2>";
?>
```

Ilustración 4: Captura de admin.php III

4.2 alumno.php

Al igual que en caso anterior se trata del script encargado de mostrar la página web privada de los usuarios con rol de alumno. Como novedad encontramos la siguiente función que simplemente permite cargar la tabla con todas las tareas asignadas al alumno en cuestión. Para realizar esto utiliza los datos devueltos por la función PHP contenida en el script "cargarTareas.php" el cual se explicará a continuación.

```
<script>
        // Obtener la referencia a la tabla en HTML
        var tabla = document.getElementById('tablaTareas');
        var fechaActual = new Date();

        // Crear una instancia de XMLHttpRequest
        var xhttp = new XMLHttpRequest();

        // Configurar el método y la URL para obtener los datos de la tabla SQL
        xhttp.open('GET', 'cargarTareas.php', true);

        // Definir la función de callback cuando la solicitud AJAX se complete
        xhttp.onload = function() {
          if (this.status === 200) {
            // Convertir la respuesta JSON en un objeto JavaScript
            var datos = JSON.parse(this.responseText);

            // Generar el contenido de la tabla dinámicamente
            var contenidoTabla = '';
            for (var i = 0; i < datos.length; i++) {
              contenidoTabla += '<tr>';
              contenidoTabla += '<td>' + datos[i].asignatura + '</td>';
              contenidoTabla += '<td>' + datos[i].nombre_tarea + '</td>';
              contenidoTabla += '<td>' + datos[i].descripcion + '</td>';
              contenidoTabla += '<td>' + datos[i].fecha_limite + '</td>';
              var fechaLimiteObjeto = new Date(datos[i].fecha_limite);

              if (fechaActual < fechaLimiteObjeto || datos[i].estado != 0) {
                      if(datos[i].estado == 0){
                        contenidoTabla += '<td>' + '<p style="color: black; text-align: center;">No disponible</p>' + '</td>';
                        contenidoTabla += '<td>' + '<p style="color: orange"><b>Pendiente</b></p>' + '</td>';
                        if(datos[i].nota == 0)
                        contenidoTabla += '<td>' + '-' + '</td>';
                      else
                        contenidoTabla += '<td>' + datos[i].nota  + '</td>';
                      }else if (datos[i].estado == 1){
                        contenidoTabla += '<td>';
                        contenidoTabla += '<button class="btn btn-primary btn-shadow btn-lg ver-entrega" data-entrega="' + datos[i].entrega + '">Ver entrega</button>';
                        contenidoTabla += '</td>';
                        contenidoTabla += '<td>' + '<p style="color: green"><b>Entregada</b></p>' + '</td>';
                        if(datos[i].nota == 0)
                        contenidoTabla += '<td>' + '-' + '</td>';
                      else
                        contenidoTabla += '<td>' + datos[i].nota  + '</td>';
```

Ilustración 5: Captura de alumno.php

4.3 cargarTareas.php

Este fichero realiza una consulta a la base de datos cargando en un array todas las tareas que tiene asignadas el alumno que accede a su página web personal. Devuelve este array al fichero “alumno.php” en formato JSON.

```php
<?php
        session_start();
        $nombre = $_SESSION['nombre'];

        require('config.php');

        // Crear la conexión
        $conn = new mysqli($dbHost, $dbUser, $dbPassword, $dbName);

        // Verificar la conexión
        if ($conn->connect_error) {
            die("Error en la conexión a la base de datos: " . $conn->connect_error);
        }

        // Preparar la consulta SQL para obtener los datos de la tabla
        $sql = "SELECT tt.*, tu.*
                FROM tabla_tareas tt
                JOIN tareas_usuarios tu ON tt.id = tu.tarea_id
                JOIN usuarios u ON u.Id = tu.usuario_id
                WHERE u.User = '$nombre'";

        // Ejecutar la consulta
        $resultado = $conn->query($sql);

        // Verificar si se obtuvieron resultados
        if ($resultado->num_rows > 0) {
            // Crear un array para almacenar los datos
            $datos = array();

            // Recorrer los resultados y guardar los datos en el array
            while ($fila = $resultado->fetch_assoc()) {
                $datos[] = $fila;
            }

            // Devolver los datos en formato JSON
            echo json_encode($datos);
        } else {
            // No se encontraron resultados
            echo "No se encontraron datos en la tabla.";
        }
        $conn->close();
?>
```

Ilustración 6: Captura de cargarTareas.php

4.4 cargarTareasProf.php

Este fichero realiza una función idéntica al anterior, simplemente cambia la consulta a realizar ya que su objetivo es devolver un array al profesor con únicamente las tareas que tengan el estado de entregado.

```php
<?php
        session_start();

        $nombre = $_SESSION['nombre'];

        require('config.php');

        // Crear la conexión
        $conn = new mysqli($dbHost, $dbUser, $dbPassword, $dbName);

        // Verificar la conexión
        if ($conn->connect_error) {
            die("Error en la conexión a la base de datos: " . $conn->connect_error);
        }

        // Preparar la consulta SQL para obtener los datos de la tabla
        $sql = "SELECT t.*, tu.*, u.User AS nombre_usuario
                FROM tabla_tareas AS t
                INNER JOIN tareas_usuarios AS tu ON t.id = tu.tarea_id
                INNER JOIN usuarios AS u ON tu.usuario_id = u.Id
                ORDER BY u.User;";

        // Ejecutar la consulta
        $resultado = $conn->query($sql);

        // Verificar si se obtuvieron resultados
        if ($resultado->num_rows > 0) {
            // Crear un array para almacenar los datos
            $datosAlumnos = array();

            // Recorrer los resultados y guardar los datos en el array
            while ($fila = $resultado->fetch_assoc()) {
                $datosAlumnos[] = $fila;
            }

            // Devolver los datos en formato JSON
            echo json_encode($datosAlumnos);
        } else {
            // No se encontraron resultados
            echo "No se encontraron datos en la tabla.";
        }

        // Cerrar la conexión
        $conn->close();
?>
```

Ilustración 7: Captura de cargarTareasProf.php

4.5 cerrarSesion.php

Este sencillo fichero es ejecutado cada vez que un usuario cierra su sesión en la plataforma. Para realizar esta acción correctamente destruye la sesión activa y redirige al usuario a la página de inicio.

```php
<?php
        session_start();

        session_unset();

        session_destroy();

        header("Location: index.html");
        exit;
?>
```

Ilustración 8: Captura de cerrarSesion.php

4.6 compra.php

Se trata del fichero encargado de la gestión de las compras realizadas por los usuarios. Entre sus funciones están la de comprobar que el artículo se encuentra disponible, que el usuario dispone de suficientes UsalCoins y la registrar las compras realizadas en la base de datos.

```php
$productoID = $_GET['id'];


$sql = "SELECT * FROM tabla_tienda WHERE id=$productoID";
$result = $conn->query($sql);
$row = $result->fetch_assoc();

$precio = $row['precio_producto'];
$disponible = $row['disponible'];

$sql2 = "SELECT * FROM usuarios WHERE User='$nombre'";
$result = $conn->query($sql2);
$row2 = $result->fetch_assoc();

$usalcoins =  $row2['Points'];
$usuarioID = $row2['Id'];
$fechaActual = date('d-m-Y');

if($usalcoins > $precio && $disponible == 1){
        $usalcoins = $usalcoins - $precio;
        $sql3 = "UPDATE usuarios SET Points = $usalcoins WHERE User = '$nombre'";
        $result = $conn->query($sql3);
        $_SESSION['puntos'] = $usalcoins;

        $sql4 = "INSERT INTO compras_tienda (usuario_id, producto_id) VALUES ('$usuarioID', '$productoID')";
        $result = $conn->query($sql4);

        $_SESSION['compra'] = "¡Compra realizada correctamente! Acuda al establecimiento indicado a reclamar su producto.";
        header("Location: tienda.php");
        exit;
}else if ($disponible == 0){
        $_SESSION['compra'] = "Este producto no se encuentra disponible en estos momentos, disculpa las molestias.";
        header("Location: tienda.php");
        exit;
}else{
        $_SESSION['compra'] = "¡Vaya! Parece que no tienes suficientes UsalCoins.";
        header("Location: tienda.php");
        exit;
}
```

Ilustración 9: Captura de compra.php

4.7 config.php

Este es uno de los ficheros más importantes de todos, contienes las credenciales de acceso a la base de datos y los tokens necesarios para el envío de SMS a través de la API externa. Por seguridad no adjunto captura.

4.8 corregirTarea.php

Contiene la lógica ejecutada cuando un profesor evalúa alguna de las tareas enviadas por los alumnos. Será el encargado de actualizar el estado de la tarea, almacenar la nota asignada a la tarea y realizar la suma de UsalCoins correspondiente tanto al profesor que corrigió la tarea como al alumno que la entregó.

```php
if ($_SERVER['REQUEST_METHOD'] === 'POST') {
        $alumno = $_POST['formulario1'];
        $tarea = $_POST['formulario2'];
        $nota = $_POST['formulario3'];
        $puntos = $nota * 5;

                $sql = "UPDATE tareas_usuarios
                        SET estado = 2
                        WHERE tarea_id = (
                          SELECT id
                          FROM tabla_tareas
                          WHERE nombre_tarea = '$tarea'
                        )
                        AND usuario_id = (
                          SELECT id
                          FROM usuarios
                          WHERE User = '$alumno'
                        )";


                $conn->query($sql);

                $sql2 = "UPDATE tareas_usuarios
                        SET nota = $nota
                        WHERE tarea_id = (
                          SELECT id
                          FROM tabla_tareas
                          WHERE nombre_tarea = '$tarea'
                        )
                        AND usuario_id = (
                          SELECT id
                          FROM usuarios
                          WHERE User = '$alumno'
                        )";


                $conn->query($sql2);

                $sql3 = "UPDATE usuarios SET Points = Points + 5 WHERE User = '$nombre';";
                $conn->query($sql3);

                $sql4 = "UPDATE usuarios SET Points = Points + $puntos WHERE User = '$alumno';";
                $conn->query($sql4);

                $sql5 = "SELECT * FROM usuarios WHERE User = '$nombre';";
                $result = $conn->query($sql5);
                $row = $result->fetch_assoc();
```

Ilustración 10: Captura de corregirTarea.php

4.9 crearTarea.php

Como su propio nombre indica este fichero es el encargado de añadir la nueva tarea publicada por un profesor a la base de datos además de asignársela a todos los alumnos correspondientes.

```php
if (isset($_POST['submit'])) {
    // Obtener los datos del formulario
    $nombreTarea = $_POST['nombreTarea'];
    $asignatura = $_POST['formulario'];
    $descripcion = $_POST['message'];
    $fechaLimite = $_POST['fecha'];

    // Preparar la consulta SQL
    $sql1 = "INSERT INTO tabla_tareas (nombre_tarea, asignatura, descripcion, fecha_limite) VALUES ('$nombreTarea', '$asignatura', '$descripcion', '$fechaLimite')";
    $conn->query($sql1);

        $sql2 = "INSERT INTO tareas_usuarios (tarea_id, usuario_id)
                SELECT LAST_INSERT_ID(), ID
                FROM usuarios
                WHERE rol IN (0, 2)";
```

Ilustración 11: Captura de crearTarea.php

4.10 enviarCodigo.php

Cuando un administrador da de alta un nuevo usuario este script enviará el SMS correspondiente al número introducido mediante el uso de una API llamada Twillio y además almacenará en la base de datos el código generado para el envío del SMS.

```php
<?php
    session_start();

    require('config.php');


    $conn = new mysqli($dbHost, $dbUser, $dbPassword, $dbName);

    if ($conn->connect_error) {
        die("Error en la conexión a la base de datos: " . $conn->connect_error);
    }

    if (isset($_POST['submit'])) {
        $nombre = $_POST['nombre'];
        $numero = $_POST['numero'];
        $rol = $_POST['selectorRol'];

        $numeroCompleto = '+34' . '' . $numero;
        $token = rand(11111, 99999);

        $sql1 = "INSERT INTO verificacion_usuarios (telefono_usuario, rol_usuario, codigo_verificacion) VALUES ('$numero', '$rol', '$token')";
        $conn->query($sql1);

        $conn->close();


        $body = '¡Hola '. $nombre . '! Has sido invitado/a a para registrarte en USAL COINS visite https://usalcoins.ddns.net/registro.php e introduce tu número de teléfono junto con el siguiente código: ' . $token . '.';

        $command = "curl 'https://api.twilio.com/2010-04-01/Accounts/$user/Messages.json' -X POST " .
            "--data-urlencode 'To=$numeroCompleto' " .
            "--data-urlencode 'From=$fron' " .
            "--data-urlencode 'Body=$body' " .
            "-u $user:$pass";

        $output = shell_exec($command);

        $_SESSION['alta_mensaje'] = 'Mensaje enviado con éxito.' . $numeroCompleto;
        header("Location: admin.php");
        exit;
    }
```

Ilustración 12: Captura de enviarCodigo.php

4.11 login.php

Este script será el encargado de mostrar la vista del formulario de inicio de sesión para poder acceder a la plataforma. Como funciones lógicas podemos destacar la encargada de mostrar el mensaje de error debajo del formulario en el caso de que las credenciales introducidas sean incorrectas.

```php
<?php
    session_start();
    $error_message = isset($_SESSION['error_message']) ? $_SESSION['error_message'] : '';

    if (isset($_SESSION['error_message'])) {
      unset($_SESSION['error_message']);
}
?>
<?php
  if ($error_message !== '') {
    echo '<p style="color: red;">' . $error_message . '</p>';
  }
?>
```

Ilustración 13: Captura de login.php

4.12 procesar_login.php

Como su propio nombre indica contiene la lógica ejecutada para comprobar que las credenciales capturadas en el formulario de inicio de sesión son correctas. En caso afirmativo creará las cookies de sesión del usuario y lo reconducirá a su página personal privada. En caso contrario enviará al script anterior el mensaje de que las credenciales no son válidas.

```php
$sql = "SELECT * FROM usuarios WHERE User = '$username'";
$result = $link->query($sql);


mysqli_close($link);

// Verificar si se encontraron resultados
if ($result->num_rows > 0) {
    // Obtener la fila de resultados
    $row = $result->fetch_assoc();
    $hash = $row['Password'];
} else {
    $_SESSION['error_message'] = 'Nombre de usuario o contraseña incorrectos.';
    header("Location: login.php");
    exit;
}

if (!password_verify($password, $hash)) {
    $_SESSION['error_message'] = 'Nombre de usuario o contraseña incorrectos.';
    header("Location: login.php");
    exit;
}else{
    session_start();
    $_SESSION['loggedin'] = true;
    $_SESSION['nombre'] = $row['User'];
    $_SESSION['puntos'] = $row['Points'];
    $_SESSION['rol'] = $row['Rol'];
    if($row['Rol'] == 0){
        header("Location: admin.php");
        exit();
    }else if ($row['Rol'] == 1){
        header("Location: profesor.php");
    }else{
        header("Location: alumno.php");
    }
}
```

Ilustración 14: Captura de procesar_login.php

4.13 procesar_registro.php

Este fichero es muy similar al "procesar_login.php" como novedad, tras comprobar que el código de registro es correcto lo elimina de la base de datos, limitándolo a ser de un solo uso. Además, por supuesto, creará un nuevo usuario en la plataforma con los datos capturados del formulario de registro.

```php
$sql = "SELECT * FROM verificacion_usuarios WHERE telefono_usuario = '$numero' AND codigo_verificacion = '$token'";
$result = $link->query($sql);

if ($result->num_rows > 0) {
    $row = $result->fetch_assoc();
    $_SESSION['rol'] = $row['rol_usuario'];
    $rol = $_SESSION['rol'];
} else {
    $_SESSION['error_message'] = 'Número de teléfono o código de invitación erroneos.';
    header("Location: registro.php");
    exit;
}

$hash = password_hash($password, PASSWORD_DEFAULT);

$sql1 = "INSERT INTO usuarios (User, Password, Nombre, Apellidos, Telefono, Rol, Points, Creation_Date) VALUES ('$username', '$hash', '$name', '$surname', '$numero', '$rol', '0', NOW())";
$link->query($sql1);


$sql2 = "DELETE FROM verificacion_usuarios WHERE telefono_usuario = '$numero' AND codigo_verificacion = '$token';";
$link->query($sql2);

$_SESSION['loggedin'] = true;
$_SESSION['nombre'] = $username;
$_SESSION['puntos'] = '0';

$link->close();

if($rol == 0){
        header("Location: admin.php");
}else if ($rol == 1){
        header("Location: profesor.php");
}else{
        header("Location: alumno.php");
}
```

Ilustración 15: Captura de procesar_registro.php

4.14 profesor.php

Este fichero se encarga de mostrar la vista correspondiente al área personal de un profesor. En términos de funciones lógicas no difiere en nada con el contenido del fichero "alumno.php" por este motivo no se adjunta ninguna captura.

4.15 registro.php

Este script será el encargado de mostrar el formulario de registro a los nuevos usuarios de la plataforma. Como función a destacar frente al fichero "login.php" encontramos la función de la siguiente captura. La cual permite que, al introducir la contraseña a crear, cada vez que un usuario pulse una tecla, comprobar si la contraseña cumple con los criterios preestablecidos de seguridad.

```
<script>

var delayTimer;

function verificarSeguridad() {
  clearTimeout(delayTimer);
  delayTimer = setTimeout(function() {
    var password = document.getElementById("password").value;

    var minLength = 8;
    var isValid = true;
    var errorMsg = "";

    // Verificar la longitud minima
    if (password.length < minLength) {
      isValid = false;
      errorMsg = "La contraseña debe tener al menos " + minLength + " caracteres.";
    }

    // Verificar contraseñas comunes
    var commonPasswords = ["12345678", "password", "qwertyui"];
    if (commonPasswords.includes(password)) {
      isValid = false;
      errorMsg = "La contraseña es demasiado común. Por favor, elige una contraseña más segura.";
    }

    var btnRegistrarse = document.getElementById("btn-registrarse");

    // Mostrar el resultado
    var feedbackElement = document.getElementById("password-feedback");
    if (isValid) {
      feedbackElement.textContent = "La contraseña es segura y cumple con los requisitos.";
      feedbackElement.style.color = "green";
      btnRegistrarse.disabled = false;
    } else {
      feedbackElement.textContent = errorMsg;
      feedbackElement.style.color = "red";
      btnRegistrarse.disabled = true;
    }
  }, 200); // Retraso de 500 ms (0.5 segundos)
}

</script>
```

Ilustración 16: Captura de registro.php

4.16 tienda.php

Por último, tenemos el script tienda.php el cual muestra la vista de la web de la tienda. En cuanto a funciones lógicas no contiene ninguna novedad respecto a los anteriores ficheros encargados de mostrar las vistas de las otras secciones de la plataforma.

4.17 reportar.php

Este script será invocado automáticamente por el sistema cada vez que se produzca un intento de acceso no autorizado a cualquiera de las vistas descritas hasta ahora. Será el encargado de obtener la dirección IPv4 del usuario, rastrear su localización y almacenar este evento en la base de datos. Todo este proceso se realiza con el objetivo de estudiar posibles aumentos de accesos no autorizados al sistema y poder banear a los usuarios malintencionados de la plataforma.

```php
$ip = $_SERVER['REMOTE_ADDR'];

$apiUrl = "http://ip-api.com/json/{$ip}";

$reemplazos = array(
    'á' => 'a',
    'é' => 'e',
    'í' => 'i',
    'ó' => 'o',
    'ú' => 'u',
    'Á' => 'A',
    'É' => 'E',
    'Í' => 'I',
    'Ó' => 'O',
    'Ú' => 'U',
    '\'' => ''
);

$conn = new mysqli($dbHost, $dbUser, $dbPassword, $dbName);
$response = file_get_contents($apiUrl);

if ($response !== false) {
    $data = json_decode($response);

    if ($data->status === 'success') {
        $pais = $textoSinAcentos = strtr($data->country, $reemplazos);
        $region = $textoSinAcentos = strtr($data->regionName, $reemplazos);
        $ciudad = $textoSinAcentos = strtr($data->city, $reemplazos);
        $fechaHoraActual = date('Y-m-d H:i:s');
        $accion = $_SESSION['reporte'];

        if ($conn->connect_error) {
                die("Error en la conexión a la base de datos: " . $conn->connect_error);
        }
        $sql = "INSERT INTO usuarios_reportados (ip_usuario, pais, region, ciudad, fecha, accion) VALUES ('$ip', '$pais', '$region', '$ciudad', '$fechaHoraActual', '$accion')";
        $conn->query($sql);

    }
}
```

Ilustración 17: Captura de reportar.php

4.18 subirTarea.php

Script ejecutado cada vez que un alumno entrega una tarea publicada por un profesor. Entre sus funciones encontramos la de comprobar la extensión del archivo, subir el fichero entregado a una localización específica del servidor y cambiar el estado de la entrega de pendiente por entregada.

```php
// Comprobar si se subió un archivo correctamente
if (isset($_FILES['archivo']) && $_FILES['archivo']['error'] === UPLOAD_ERR_OK) {
        $directorioDestino = 'tareas/' . $nombre . '/';

        if (!file_exists($directorioDestino)) {
                mkdir($directorioDestino, 0777, true);
        }

        $fileExtension = pathinfo($_FILES['archivo']['name'], PATHINFO_EXTENSION);
        if ($fileExtension != 'pdf' && $fileExtension != 'zip' && $fileExtension != 'jpg') {
                $_SESSION['tarea_mensaje'] = "¡Vaya! El archivo tiene una extensión no permitida.";
                header("Location: alumno.php");
                exit;

        }

        $nombreArchivo = $_FILES['archivo']['name'];
        $rutaArchivo = $_FILES['archivo']['tmp_name'];
        $rutaDestino = $directorioDestino . $nombreArchivo;
        $tarea = $_POST['formulario'];

        // Mover el archivo a la ubicación deseada
        move_uploaded_file($rutaArchivo, $rutaDestino);


        $sql = "UPDATE tareas_usuarios
                SET estado = 1
                WHERE tarea_id = (
                  SELECT id
                  FROM tabla_tareas
                  WHERE nombre_tarea = '$tarea'
                )
                AND usuario_id = (
                  SELECT id
                  FROM usuarios
                  WHERE User = '$nombre'
                )";
```

Ilustración 18: Captura de subirTarea.php

5. Bibliografía

Digital Ocena, Guía para la instalación de Apache, MySQL y PHP en Ubuntu 22.04 – https://www.digitalocean.com/community/tutorials/how-to-install-linux-apache-mysql-php-lamp-stack-on-ubuntu-22-04

Free CSS, Plantilla CSS utilizada para el diseño de la página web - https://www.free-css.com/free-css-templates/page275/roxy

Twillio, API para el envío de mensajes SMS - https://www.twilio.com/en-us