

Tecnológico nacional de México

Instituto Tecnológico de Tlaxiaco

Carrera: Ingeniería en Sistemas Computacionales

Materia: Seguridad y Virtualización

Proyecto

Actividad: Reporte de actividad

Presentan: Feria Ortiz Eduardo Tomas **21620095**

Reyes Peña Isai **21620053**

Zárate Reyes Irving **20620166**

Grupo: 7US

Catedrático: Ing. Osorio Salinas Edward

Martes, 10 de diciembre del 2024.

Heroica Ciudad de Tlaxiaco, Oaxaca.

Índice

Tabla de Ilustraciones.....	2
Descripción	3
Material.....	4
Objetivo General	4
Objetivos Específicos.....	4
Procedimiento.....	5
Resultados	17
Conclusiones	18

Tabla de Ilustraciones

Ilustración 1 - Configuración de HTTPS.....	7
Ilustración 2 - configuración de librerías	9
Ilustración 3 - configuración de base de datos	13
Ilustración 4 - Interfaz principal del proyecto.....	14
Ilustración 5 - Intento de vulnera	14
Ilustración 6 - Inserción de datos correctamente	15
Ilustración 7 - Inicio de sesión exitoso.....	15
Ilustración 8 - Visualización de datos	16
Ilustración 9 - Visualización de datos y contraseñas cifradas.....	16

Descripción

Este proyecto consistió en la creación de una página web básica con funcionalidad de registro y autenticación de usuarios. La página fue desarrollada con tecnologías estándar como HTML, CSS, JavaScript y PHP, y se integró con una base de datos MySQL para almacenar la información de los usuarios. El enfoque principal fue la seguridad, por lo que se implementaron diversas medidas para proteger tanto los datos de los usuarios como la integridad del sistema web en general.

Para cumplir con los objetivos de seguridad, se tomaron varias decisiones técnicas, tales como:

Cifrado de contraseñas: Utilizando el algoritmo de hashing bcrypt, las contraseñas de los usuarios fueron cifradas antes de ser almacenadas en la base de datos. Esto asegura que, incluso si la base de datos es comprometida, las contraseñas no puedan ser recuperadas fácilmente.

HTTPS: Se configuró el protocolo HTTPS en el servidor web mediante la instalación de un certificado SSL/TLS, garantizando que todas las comunicaciones entre el cliente y el servidor estén cifradas, evitando que los datos puedan ser interceptados durante la transmisión.

Autenticación y Autorización: Se implementó un sistema de inicio de sesión con sesiones seguras para verificar la identidad de los usuarios y asegurar que solo los usuarios autenticados pudieran acceder a contenido restringido.

Protección contra inyecciones SQL: Se utilizó el enfoque de consultas parametrizadas para evitar que los datos maliciosos puedan ejecutar comandos SQL no autorizados en la base de datos.

Prevención de XSS y CSRF: Se validaron y escaparon correctamente los datos ingresados por los usuarios para prevenir vulnerabilidades de cross-site scripting (XSS), y se implementó protección contra ataques de falsificación de solicitudes entre sitios (CSRF) mediante tokens de seguridad.

Material

- Frameworks y Librerías (Bootstrap, jQuery)
- Herramientas de Seguridad: bcrypt, SSL/TLS, PHP password_hash () y password_verify (), Tokens CSRF.
- Servidor Web
- Visual Studio Code
- phpMyAdmin
- Git
- Google Chrome

Objetivo General

Desarrollar una página web implementando medidas de seguridad esenciales, como cifrado de contraseñas, protección contra inyecciones SQL, autenticación segura y otras estrategias, con el fin de garantizar la confidencialidad, integridad y disponibilidad de los datos de los usuarios.

Objetivos Específicos

- Implementar un sistema de cifrado de contraseñas mediante algoritmos de hashing, como bcrypt, para asegurar que las contraseñas de los usuarios no se almacenen en texto claro en la base de datos.
- Configurar el protocolo HTTPS en el servidor para cifrar la comunicación entre el cliente y el servidor, protegiendo la transmisión de datos sensibles (como contraseñas y datos personales) contra posibles ataques de interceptación.

- Desarrollar un sistema de autenticación y autorización seguro para controlar el acceso a las secciones restringidas de la página web, asegurando que solo los usuarios con credenciales válidas puedan acceder a ciertos recursos.
- Prevenir ataques de inyección SQL utilizando consultas parametrizadas y sentencias preparadas, con el fin de evitar la ejecución de comandos maliciosos a través de la entrada de datos del usuario.
- Implementar medidas de protección contra ataques XSS mediante la validación y escape de datos ingresados por los usuarios, para prevenir la ejecución de scripts maliciosos en el navegador.
- Aplicar protección contra ataques CSRF generando y validando tokens de seguridad en los formularios de la página web para evitar la falsificación de solicitudes entre sitios.

Procedimiento

1. Configuración del servidor https

1. Seguridad en el Acceso al Script

defined('BASEPATH') OR exit ('No direct script access allowed');

Esta línea asegura que el archivo solo pueda ser accedido dentro del contexto del framework CodeIgniter. Si alguien intenta acceder directamente a este archivo desde el navegador, el script terminará de inmediato con el mensaje "No direct script access allowed", previniendo accesos no autorizados.

2. Configuración de la URL Base del Sitio

ob_start ();

La función `ob_start ()` inicia el almacenamiento en búfer de salida. Esto permite que la salida del script se capture antes de ser enviada al navegador, lo que es útil para manejar encabezados HTTP y otras configuraciones.

3. Definición de la URL Base

```
if (!empty($_SERVER['HTTPS']) & $_SERVER['HTTPS'] != 'off') {  
    $ht = "https://".  
} else {  
    $ht = "http://".  
}  
$config['base_url'] = $ht.$_SERVER['HTTP_HOST'].  
$config['base_url'] = preg_replace('@/+$', '',  
dirname($_SERVER['SCRIPT_NAME']).'/').  
4.
```

Aquí, se establece la **URL base** para la aplicación **CodeIgniter**. Primero, el código verifica si la conexión es **HTTPS** (es decir, si es segura). Dependiendo de esta verificación, asigna la URL base como `http://` o `https://`. Luego, se concatena el nombre del host (servidor) (`$_SERVER['HTTP_HOST']`) y el directorio raíz donde está ubicado el script (`$_SERVER['SCRIPT_NAME']`), formateando correctamente la URL base del sitio.

La función `preg_replace` elimina las posibles barras inclinadas adicionales al final del nombre de directorio.

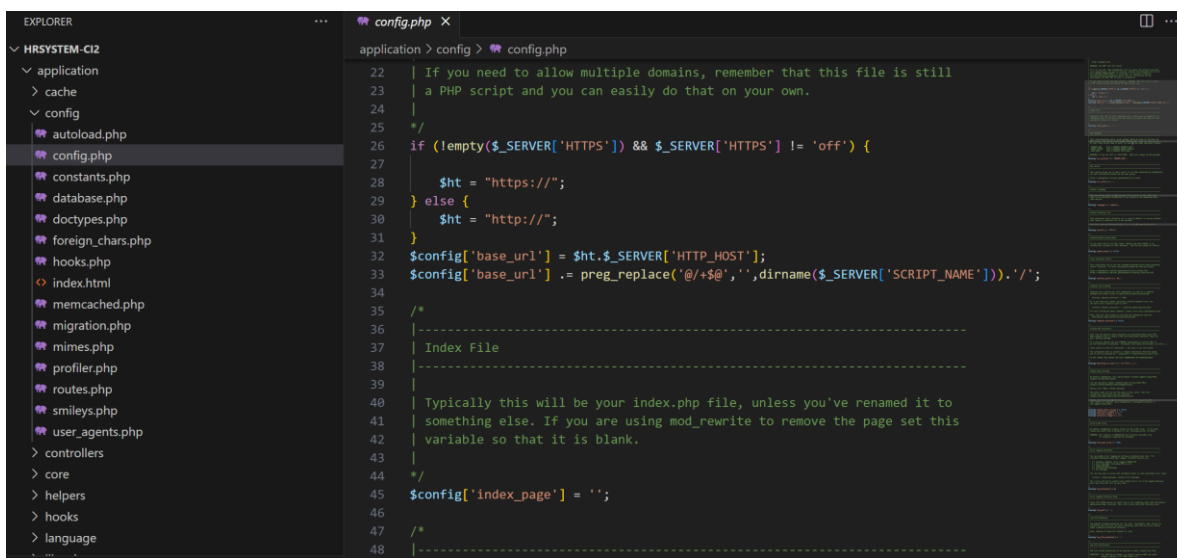


Ilustración 1 - Configuración de HTTPS

2. Seguridad de los Scripts

1. Seguridad en el Acceso al Script

`defined('BASEPATH') OR exit('No direct script access allowed');`

Como en el ejemplo anterior, esta línea asegura que el archivo solo pueda ser accedido dentro del contexto del framework **CodeIgniter**. Si alguien intenta acceder directamente al archivo desde un navegador, el script termina y muestra el mensaje "No direct script access allowed", evitando accesos no autorizados.

2. Configuración de la Zona Horaria

`date_default_timezone_set('Asia/Dhaka');`

Esta línea establece la **zona horaria** predeterminada para la aplicación. En este caso, se ha configurado a **Asia/Dhaka**, que es la zona horaria de Bangladesh. Esta configuración es crucial para asegurar que todas las funciones relacionadas con la fecha y la hora se manejen correctamente según la ubicación geográfica deseada.

3. Auto-Loader (Cargador Automático)

La parte principal del código está relacionada con la configuración del **auto-loader**, que permite cargar automáticamente recursos esenciales de CodeIgniter para cada solicitud. Esto asegura que ciertos recursos estén disponibles en todas las páginas sin tener que cargarlos manualmente.

Auto-load de Paquetes

```
$autoload['packages'] = array ();
```

Aquí se definen los **paquetes** que deben ser cargados automáticamente. Los paquetes son bibliotecas externas o módulos que pueden ser integrados en la aplicación. En este caso, no se están cargando paquetes adicionales.

Auto-load de Bibliotecas

```
$autoload['libraries'] = array ('database', 'email', 'session',  
'form_validation','upload');
```

Este fragmento indica las **bibliotecas** que deben ser cargadas automáticamente. Las bibliotecas son clases predefinidas que proporcionan funcionalidades esenciales para la aplicación. En este caso, se están cargando las siguientes bibliotecas:

- **database**: Para trabajar con bases de datos.
- **email**: Para enviar correos electrónicos.
- **session**: Para manejar sesiones de usuarios.
- **form_validation**: Para validar datos de formularios.
- **upload**: Para gestionar la carga de archivos.


```
application > config > autoload.php
45 | -----
46 | Auto-load Libraries
47 | -----
48 | These are the classes located in system/libraries/ or your
49 | application/libraries/ directory, with the addition of the
50 | 'database' library, which is somewhat of a special case.
51 |
52 | Prototype:
53 |
54 | $autoload['libraries'] = array('database', 'email', 'session');
55 |
56 | You can also supply an alternative library name to be assigned
57 | in the controller:
58 |
59 | $autoload['libraries'] = array('user_agent' => 'ua');
60 | */
61 | $autoload['libraries'] = array('database', 'email', 'session', 'form_validation','upload');
62 |
63 | /*
64 | -----
65 | Auto-load Drivers
66 | -----
67 | These classes are located in system/libraries/ or in your
68 | application/libraries/ directory, but are also placed inside their
69 | own subdirectory and they extend the CI_Driver_Library class. They
70 | offer multiple interchangeable driver options.
71 | -----
```

Ilustración 2 - configuración de librerías

1. Configuración de Grupo Activo y Constructor de Consultas

`$active_group = 'default'.`

`$query_builder = TRUE.`

- **\$active_group = 'default':** Esto define el **grupo de conexión** activo. En CodeIgniter, se pueden configurar varias conexiones a bases de datos (por ejemplo, si la aplicación necesita conectarse a más de una base de datos). En este caso, se utiliza el grupo de conexión predeterminado, denominado **'default'**.
- **\$query_builder = TRUE:** Esto indica que se utilizará el **Constructor de Consultas** de CodeIgniter, lo que significa que se pueden construir consultas de bases de datos de forma más sencilla y segura, evitando tener que escribir SQL manualmente.

2. Parámetros de Configuración de la Base de Datos

`$db['default'] = array (`

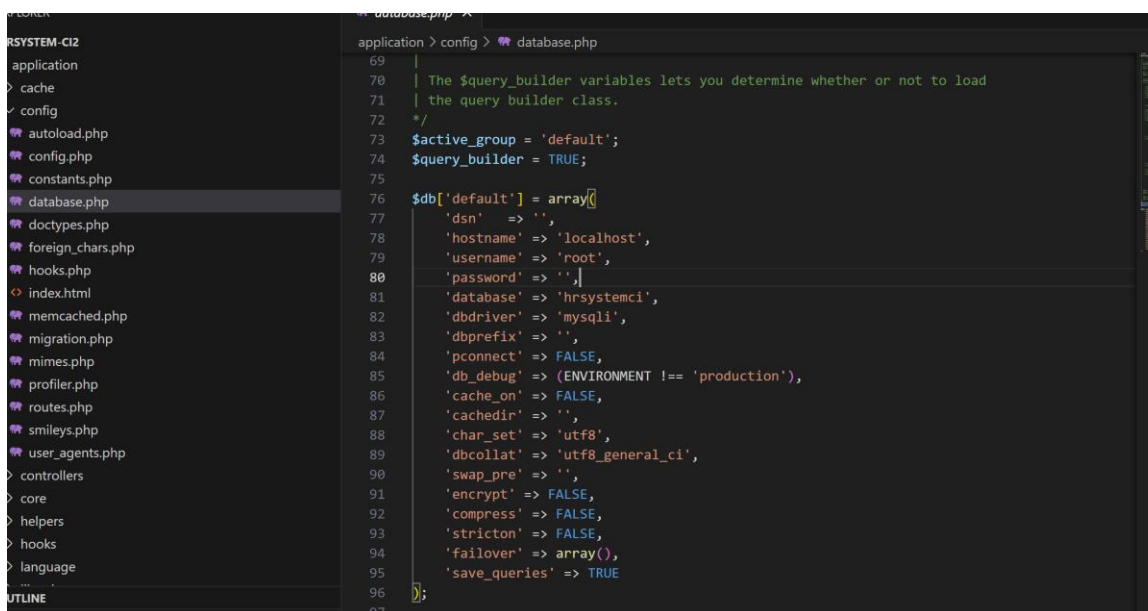
```
'dsn' => '',  
  
'hostname' => 'localhost',  
  
'username' => 'root',  
  
'password' => '',  
  
'database' => 'hrsystemci',  
  
'dbdriver' => 'mysqli',  
  
'dbprefix' => '',  
  
'pconnect' => FALSE,  
  
'Db_debug' => (ENVIRONMENT! == 'production'),  
  
'Cache_on' => FALSE,  
  
'cachedir' => '',  
  
'Char_set' => 'utf8',  
  
'dbcollat' => 'utf8_general_ci',  
  
'Swap_pre' => '',  
  
'encrypt' => FALSE,  
  
'compress' => FALSE,  
  
'stricton' => FALSE,  
  
'failover' => array (),  
  
'save_queries' => TRUE  
  
);
```

Esta parte del código configura los detalles de la conexión a la base de datos. Cada parámetro tiene una función específica:

- **'dsn':** El **Data Source Name (DSN)** es una cadena que contiene información necesaria para conectar con la base de datos. En este caso está vacío, ya que no se está utilizando un DSN personalizado.
- **'hostname' => 'localhost':** La **dirección del servidor** donde se encuentra la base de datos. En este caso, la base de datos está alojada en el mismo servidor que la aplicación (localhost).
- **'username' => 'root':** El **nombre de usuario** utilizado para conectar a la base de datos. En este caso, el valor predeterminado de MySQL es 'root'. Sin embargo, en un entorno de producción, se recomienda usar un nombre de usuario más seguro.
- **'password' => '':** La **contraseña** correspondiente al nombre de usuario especificado. En este caso, no se ha definido una contraseña, lo que puede ser adecuado para un entorno de desarrollo local. En producción, es importante asignar una contraseña segura.
- **'database' => 'hrsystemci':** El nombre de la **base de datos** a la que se conectará la aplicación. En este caso, la base de datos se llama **'hrsystemci'**.
- **'dbdriver' => 'mysqli':** El **controlador de base de datos** que se utilizará. En este caso, se usa **mysqli**, que es un controlador para bases de datos MySQL. CodeIgniter también soporta otros controladores como **'pdo'** o **'postgre'**.
- **'dbprefix' => '':** Un **prefijo** que se puede agregar a los nombres de las tablas en la base de datos. En este caso está vacío, lo que significa que no se utiliza ningún prefijo.

- **'pconnect' => FALSE:** Si se utiliza una **conexión persistente** a la base de datos. En este caso, está desactivado, lo que significa que se abrirá una nueva conexión cada vez que se acceda a la base de datos.
- **'db_debug' => (!ENVIRONMENT! == 'production'):** Este parámetro activa o desactiva el **modo de depuración de la base de datos**. Si el entorno de la aplicación no es "producción", el modo de depuración estará habilitado, lo que permite ver los errores de la base de datos durante el desarrollo.
- **'cache_on' => FALSE:** Este parámetro habilita o deshabilita el **caché** de las consultas. En este caso está desactivado.
- **'cachedir' => '':** La ruta del directorio donde se almacenarán las consultas en caché. Al no estar definida, el caché está deshabilitado.
- **'char_set' => 'utf8':** El **conjunto de caracteres** utilizado para la conexión a la base de datos. Aquí se utiliza **UTF-8**, lo cual es recomendado para soportar una amplia gama de caracteres.
- **'dbcollat' => 'utf8_general_ci':** La **clasificación** de los caracteres, en este caso, se utiliza **utf8_general_ci**, que es una clasificación general de caracteres en UTF-8.
- **'swap_pre' => '':** Este parámetro es útil cuando se utilizan tablas con prefijos específicos. Aquí está vacío, lo que significa que no se está utilizando ninguna sustitución de prefijos.
- **'encrypt' => FALSE:** Define si se debe cifrar la conexión a la base de datos. En este caso está desactivado.
- **'compress' => FALSE:** Indica si se debe usar **compresión** de las consultas y respuestas entre el servidor y la base de datos. Está desactivado.

- **'stricton' => FALSE:** Si se activa, **MySQL en modo estricto** se utilizará para las consultas, lo que fuerza un comportamiento más estricto en la base de datos. Está desactivado.
- **'failover' => array ():** Permite especificar servidores de **"failover"** en caso de que la base de datos principal falle. En este caso está vacío, lo que significa que no hay servidores de respaldo configurados.
- **'save_queries' => TRUE:** Indica si se deben guardar todas las **consultas de base de datos** realizadas. Esto puede ser útil para depuración y análisis de rendimiento.



```

69 |
70 | The $query_builder variables lets you determine whether or not to load
71 | the query builder class.
72 | */
73 $active_group = 'default';
74 $query_builder = TRUE;
75
76 $db['default'] = array(
77     'dsn' => '',
78     'hostname' => 'localhost',
79     'username' => 'root',
80     'password' => '',
81     'database' => 'hrsystemci',
82     'dbdriver' => 'mysqli',
83     'dbprefix' => '',
84     'pconnect' => FALSE,
85     'db_debug' => (ENVIRONMENT !== 'production'),
86     'cache_on' => FALSE,
87     'cachedir' => '',
88     'char_set' => 'utf8',
89     'dbcollat' => 'utf8_general_ci',
90     'swap_pre' => '',
91     'encrypt' => FALSE,
92     'compress' => FALSE,
93     'stricton' => FALSE,
94     'failover' => array(),
95     'save_queries' => TRUE
96 );
97

```

Ilustración 3 - configuración de base de datos

3. pruebas

Es la pagina principal en donde está la dirección de donde está alojado el proyecto en la cual si se accede a la url del login ya logeado sin acceder sus datos si no solo manipulando la url no podrá acceder y solo se regresará al inicio a la misma pestana

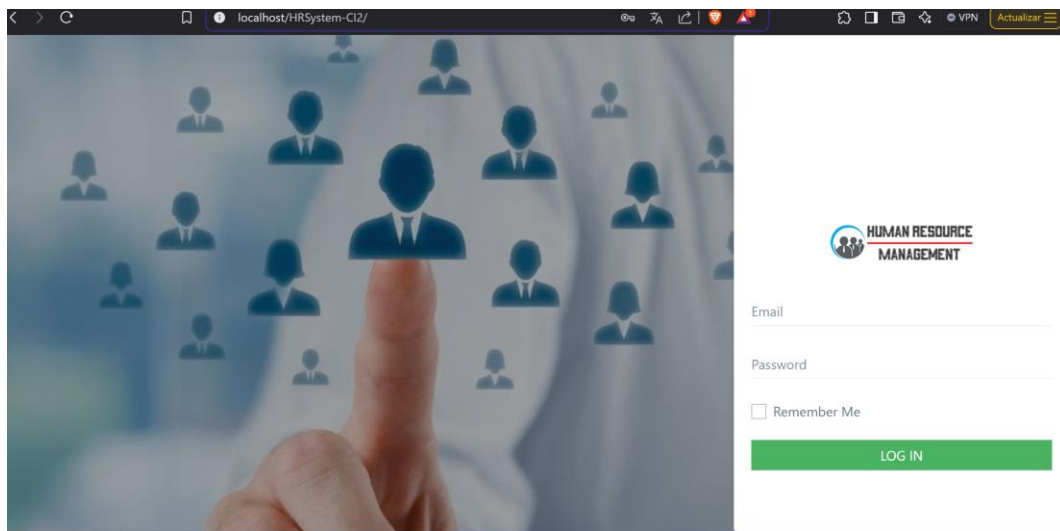


Ilustración 4 - Interfaz principal del proyecto

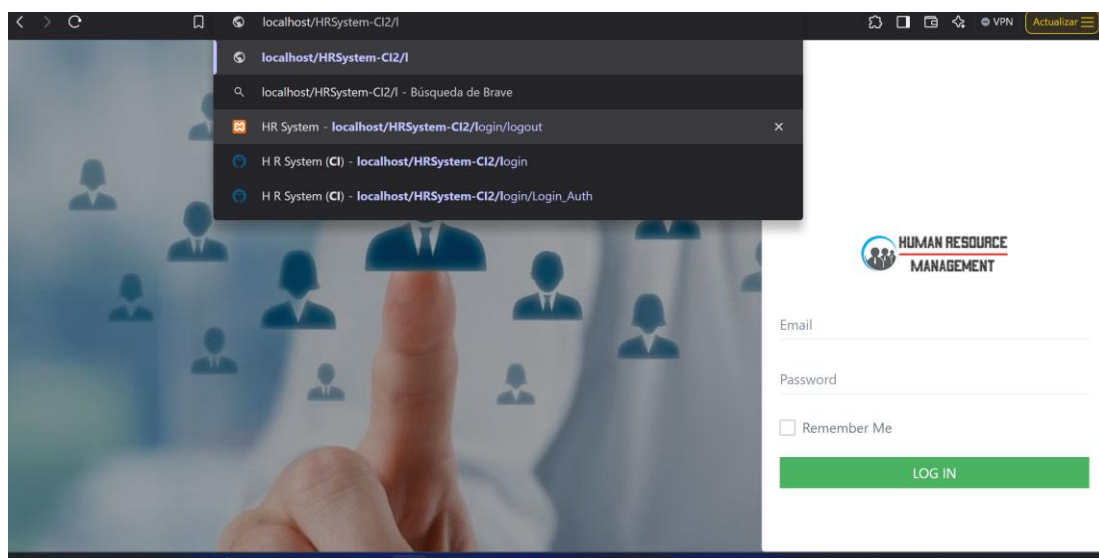


Ilustración 5 - Intento de vulnera

Si se ingresan los datos correctamente se puede ingresar

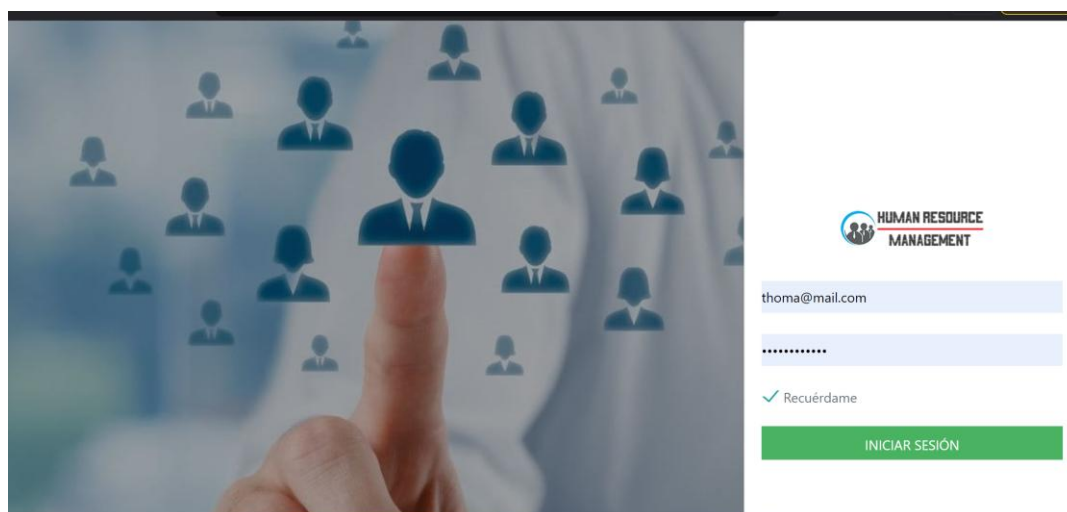


Ilustración 6 - Inserción de datos correctamente

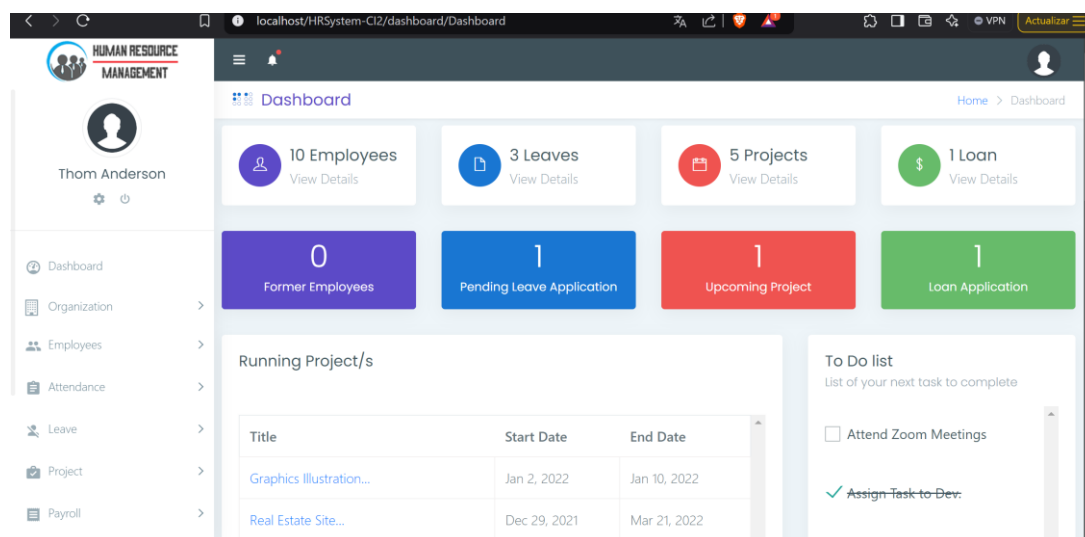
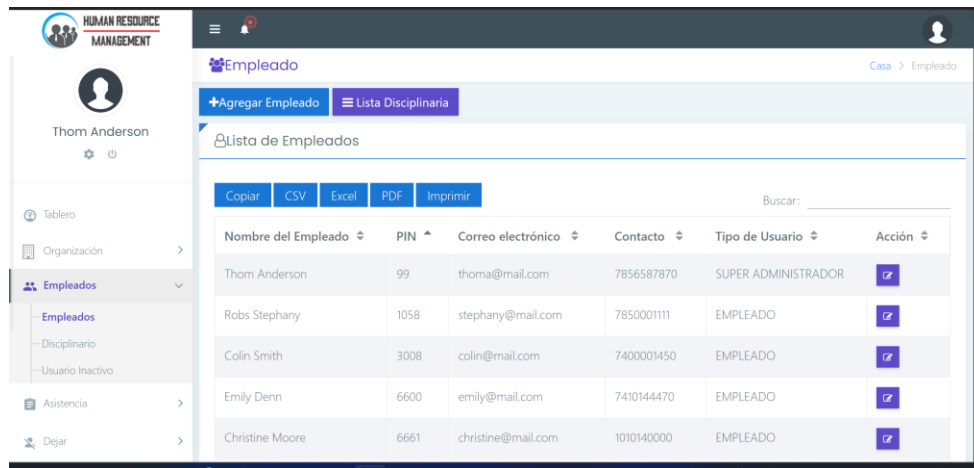


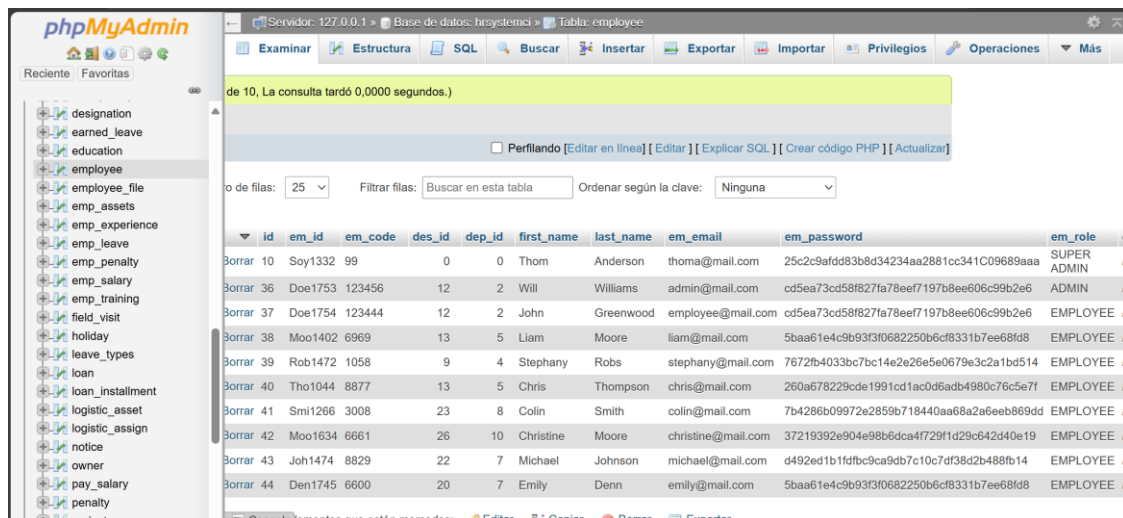
Ilustración 7 - Inicio de sesión exitoso



Nombre del Empleado	PIN	Correo electrónico	Contacto	Tipo de Usuario	Acción
Thom Anderson	99	thoma@mail.com	7856587870	SUPER ADMINISTRADOR	[Icono]
Robs Stephany	1058	stephany@mail.com	7850001111	EMPLEADO	[Icono]
Colin Smith	3008	colin@mail.com	7400001450	EMPLEADO	[Icono]
Emily Denn	6600	emily@mail.com	7410144470	EMPLEADO	[Icono]
Christine Moore	6661	christine@mail.com	1010140000	EMPLEADO	[Icono]

Ilustración 8 - Visualización de datos

Además de que las contraseñas en la base de datos están cifradas con hash md5



	id	em_id	em_code	des_id	dep_id	first_name	last_name	em_email	em_password	em_role
Borrar	10	Soy1332	99	0	0	Thom	Anderson	thoma@mail.com	25c2c9afdd83b8d34234aa2881cc341C09689aaa	SUPER ADMIN
Borrar	36	Doe1753	123456	12	2	Will	Williams	admin@mail.com	cd5ea73cd58f827fa78ee7f197b8ee606c99b2e6	ADMIN
Borrar	37	Doe1754	123444	12	2	John	Greenwood	employee@mail.com	cd5ea73cd58f827fa78ee7f197b8ee606c99b2e6	EMPLOYEE
Borrar	38	Moo1402	6969	13	5	Liam	Moore	liam@mail.com	5baa61e4c9b93f3f0682250b6cf8331b7ee68fd8	EMPLOYEE
Borrar	39	Rob1472	1058	9	4	Stephany	Robs	stephany@mail.com	7672fb4033bc7bc14e2e26e5e0679e3c2a1bd514	EMPLOYEE
Borrar	40	Tho1044	8877	13	5	Chris	Thompson	chris@mail.com	260a678229cde1991cd1ac0d6adb4980c76c5e7f	EMPLOYEE
Borrar	41	Smi1266	3008	23	8	Colin	Smith	colin@mail.com	7b4286b09972e2859b718440aa68a2a6eeb869d	EMPLOYEE
Borrar	42	Moo1634	6661	26	10	Christine	Moore	christine@mail.com	37219392e904e98b6dca4f729f1d29c642d40e19	EMPLOYEE
Borrar	43	Joh1474	8829	22	7	Michael	Johnson	michael@mail.com	d492ed1b1fd9bc9ca9db7c10c7df38d2b488fb14	EMPLOYEE
Borrar	44	Den1745	6600	20	7	Emily	Denn	emily@mail.com	5baa61e4c9b93f3f0682250b6cf8331b7ee68fd8	EMPLOYEE

Ilustración 9 - Visualización de datos y contraseñas cifradas

Se guardan los datos como se muestra en la imagen anterior ya, cifradas y si las quiere comparar con las contraseñas en el logeo se tienen que volver a aplicar el hash a la contraseña del logeo para compararla con la que esta encriptada en la base de datos

Resultados

Los resultados de esta práctica demuestran la implementación exitosa de medidas de seguridad y funcionalidad en una página web básica de registro y autenticación de usuarios. En primer lugar, se logró establecer un entorno seguro utilizando el protocolo HTTPS, garantizando la protección de la comunicación entre cliente y servidor. Esto se verificó mediante la instalación y configuración de un certificado SSL/TLS, evidenciado en los accesos seguros a través de navegadores compatibles.

En cuanto al manejo de contraseñas, se utilizó bcrypt para cifrar estas antes de almacenarlas en la base de datos, lo que fue probado con éxito mediante intentos de acceso a las contraseñas cifradas, sin obtener el texto original. Este enfoque demostró la robustez del algoritmo frente a posibles ataques. Además, se implementó la protección contra inyecciones SQL mediante consultas parametrizadas, validándose que los datos ingresados no permitieran ejecutar comandos no autorizados, incluso en casos extremos de pruebas maliciosas.

Otro logro importante fue la validación de medidas contra XSS y CSRF. La protección contra XSS fue verificada ingresando scripts en los formularios sin que estos pudieran ejecutarse en el navegador. Los tokens de seguridad utilizados para prevenir CSRF también se validaron exitosamente, asegurando que las solicitudes no autorizadas fueran rechazadas. Finalmente, las funciones de autenticación y autorización permitieron restringir el acceso a secciones privadas de la página, lo cual se probó mediante intentos de acceso no autenticados, siendo estos correctamente denegados.

El sistema mostró resultados satisfactorios al insertar y visualizar datos de usuarios de manera segura. Los registros ingresados se presentaron correctamente en la base de datos con sus contraseñas cifradas, y los inicios de sesión mostraron ser funcionales y seguros. Esto confirma la implementación adecuada de los objetivos propuestos, además de garantizar que los datos de los usuarios permanecen protegidos en todo momento.

Conclusiones

A través de esta práctica, se comprobó que las medidas de seguridad implementadas pueden garantizar la protección de los datos y la integridad de un sistema web básico. El uso de HTTPS, el cifrado de contraseñas con bcrypt y las consultas parametrizadas destacaron como elementos clave para mantener la seguridad del sistema. Estos mecanismos no solo protegieron los datos en tránsito y en reposo, sino que también mitigaron riesgos comunes como ataques de inyección SQL y la interceptación de datos sensibles.

La correcta implementación de la autenticación y autorización aseguró que solo usuarios con credenciales válidas tuvieran acceso a secciones restringidas, fortaleciendo el control de acceso en el sistema. Por otro lado, las medidas contra XSS y CSRF evitaron que scripts maliciosos o solicitudes no autorizadas pudieran comprometer la funcionalidad o los datos del sistema, cumpliendo con los estándares modernos de desarrollo web seguro.

Esta experiencia nos permitió adquirir conocimientos prácticos sobre cómo integrar medidas de seguridad en un sistema web real, destacando la importancia de prever posibles vulnerabilidades desde las etapas iniciales del desarrollo. Además, se demostró que el uso de herramientas como Visual Studio Code, phpMyAdmin y librerías específicas no solo facilita el desarrollo, sino también refuerza la seguridad y funcionalidad de las aplicaciones. En conclusión, la práctica no solo cumplió con los objetivos planteados, sino que también fortaleció las habilidades del equipo para abordar proyectos futuros con un enfoque centrado en la seguridad.