



# VULNSOCIAL

Desarrollo de Aplicaciones Web

Aitor González González

Gustavo Millán García

19/12/2024

IES EL CAÑAVERAL

Departamento de Informática

# Memoria del proyecto

## Documento técnico

### Índice

1.- Introducción .....	4
1.1 Descripción y contexto del proyecto .....	4
1.2 Motivación del proyecto .....	4
1.3 Beneficios esperados .....	4
2.- Objetivo/s generales del proyecto .....	5
3.- Objetivos específicos .....	6
4.- Contexto actual.....	7
4.1 Estado del arte .....	7
4.2 Conceptos clave.....	7
5.- Análisis de requisitos .....	8
5.1.- Diagrama de casos de uso. ....	8
5.1.1 Diagrama de casos de uso del rol Manager .....	8
5.1.2 Diagrama de casos de uso del rol Admin .....	9
5.1.3 Diagrama de casos de uso del rol usuario.....	10
5.2.- Requisitos funcionales principales: funcionalidades que debe tener la aplicación. ....	12
5.2.1 Requisitos Funcionales Core .....	12
5.2.1.1 Gestión de usuarios.....	12
5.2.1.2 Gestión de Posts.....	12
5.2.1.3 Sistema de Likes .....	13
5.2.1.4 Chat en Tiempo Real .....	13
5.2.1.5 Gestión de Comentarios y Respuestas .....	13
5.2.1.6 Panel de Control del Manager .....	13
5.2.2 Requisitos no funcionales.....	13
5.2.2.1 Rendimiento .....	14
5.2.2.2 Seguridad.....	14
5.2.2.3 Usabilidad.....	14
5.2.2.4 Escalabilidad.....	14
5.2.2.5 Compatibilidad .....	14
5.4.1 Usuario estándar (Usuario registrado).....	15
5.4.2 Administrador.....	15
5.4.3 Manager.....	15
5.4.4 Usuario no registrado .....	16

6.- Diseño de la aplicación .....	16
6.1.- Mockups o wireframes o prototipos de la interfaz gráfica de usuario. ....	16
6.2.- Arquitectura del sistema.....	26
6.3.- Diagramas de clases y de entidad-relación.....	28
6.4.- Diseño de la base de datos: esquemas y tablas.....	29
7.- Desarrollo de la aplicación .....	32
7.1.- Tecnologías y herramientas utilizadas. ....	32
7.2.- Descripción de las principales funcionalidades implementadas ilustrándolo con fragmentos de código relevantes.....	33
8.- Planificación del proyecto.....	43
8.1.- Acciones.....	43
8.1.1 Toma de requisitos .....	43
8.1.2 Análisis de los requisitos .....	43
8.1.3 Diseño.....	43
8.1.4 Codificación (Implementación).....	44
8.1.5 Pruebas .....	44
8.1.6 Documentación.....	44
8.2.- Temporalización y secuenciación .....	44
9.- Pruebas y validación .....	45
Tipos de Pruebas Realizadas: .....	45
Herramientas Utilizadas: .....	45
Plan de Pruebas: .....	45
Pruebas Concretas: .....	45
10.- Relación del proyecto con los módulos del ciclo.....	46
10.1 Sistemas Informáticos .....	46
10.2 Bases de Datos.....	46
10.3 Programación.....	46
10.4 Lenguajes de Marcas .....	46
10.5. Entornos de Desarrollo .....	47
10.6 Desarrollo Web en Entorno Cliente.....	47
10.7 Desarrollo Web en Entorno Servidor .....	47
10.8 Despliegue de Aplicaciones Web.....	47
10.8.1 Configuración del Servidor .....	47
a. Hosting Compartido.....	47
b. Servidor en la Nube (Más Personalizable) .....	47
10.8.2 Configurar el Proyecto Localmente.....	47
10.8.3 Subir Archivos al Servidor .....	48

---

Método 1: Hosting Compartido .....	48
Método 2: Servidor en la Nube.....	48
10.8.4 Configurar la Base de Datos .....	48
10.8.5 Configurar Seguridad .....	48
10.8.6 Probar la Aplicación en Producción .....	49
10.8.7 Configurar un Subdominio para Evaluación .....	49
11.- Conclusiones .....	49
12.- Proyectos futuros.....	50
12.1 Ampliaciones y mejoras .....	50
12.2 Nuevos proyectos basados en este trabajo.....	51
13.- Bibliografía/Webgrafía.....	52
Bibliografía.....	52
14.- Anexos.....	52
14.1 Vídeo demo .....	52

## 1.- Introducción

---

VulnSocial nació de la idea de que me gusta mucho investigar y la ciberseguridad, siempre estoy pensando hasta donde llega el límite la seguridad en las aplicaciones. Esta aplicación no solo permite a los usuarios postear y compartir lo que piensen, sino que también introduce un entorno seguro para que los participantes exploren y experimenten con diversas vulnerabilidades web. El objetivo principal de VulnSocial es sensibilizar a los usuarios sobre la seguridad en las aplicaciones web y concienciar que se debe invertir bastante formación en ciberseguridad, proporcionando un espacio donde puedan identificar, entender y explotar intencionadamente fallos de seguridad. Esto se logra a través de la implementación de diferentes vulnerabilidades en la plataforma, que los usuarios deben descubrir y resolver.

A diferencia de otras redes sociales que se enfocan en la publicación de imágenes o pensamientos personales, VulnSocial está diseñada específicamente para el aprendizaje práctico de la ciberseguridad. Es un CTF (Capture the Flag) donde se tiene que intentar encontrar una flag que estará oculta en una parte de la aplicación mientras se vaya resolviendo los retos. Las ideas que quiero implementar en la aplicación son vulnerabilidades reales que me he encontrado en casos reales de bug bounty y aplicaciones web.

### 1.1 Descripción y contexto del proyecto

El proyecto consiste en el desarrollo de una aplicación web llamada **VulnSocial**, que emula una red social funcional con características comunes como publicaciones, likes y comentarios. Sin embargo, esta aplicación también incluirá vulnerabilidades intencionales que los usuarios podrán explorar para comprender cómo funcionan ciertos tipos de ataques informáticos. Además, ofrecerá guías prácticas para aprender a prevenir dichas vulnerabilidades.

El entorno se centra en proporcionar un espacio educativo seguro para que desarrolladores y estudiantes de ciberseguridad puedan experimentar y adquirir conocimientos prácticos sobre la importancia de proteger aplicaciones web. Para lograr esto, la aplicación se ha construido utilizando lenguajes como PHP, JavaScript y MySQL, tecnologías ampliamente utilizadas en el desarrollo web.

### 1.2 Motivación del proyecto

La motivación principal de este proyecto radica en la creciente importancia de la ciberseguridad en el ámbito de las aplicaciones web. A pesar de los avances tecnológicos, muchas empresas y desarrolladores aún desconocen las prácticas básicas para proteger sus sistemas contra amenazas comunes. Este desconocimiento puede conducir a filtraciones de datos, pérdidas económicas y daños irreparables a la reputación de las organizaciones.

Además, el proyecto pretende llenar un vacío en el área educativa proporcionando una herramienta interactiva que permita a los usuarios no solo entender los riesgos, sino también aprender a mitigarlos. La combinación de una red social simulada con elementos vulnerables y un entorno educativo crea una experiencia única que busca fomentar una mayor conciencia sobre la importancia de la ciberseguridad.

### 1.3 Beneficios esperados

La implementación exitosa de este proyecto tiene varios beneficios tanto para los usuarios finales como para la comunidad en general:

1. **Educación en ciberseguridad:** Los usuarios podrán comprender cómo funcionan los ataques más comunes y aprender a protegerse de ellos.
2. **Plataforma interactiva:** La aplicación ofrecerá una experiencia práctica que permitirá a los usuarios interactuar directamente con vulnerabilidades reales en un entorno controlado.
3. **Concienciación:** Incrementará la conciencia sobre la importancia de implementar medidas de seguridad desde las etapas iniciales del desarrollo.
4. **Formación para desarrolladores:** Proveerá una herramienta educativa para desarrolladores interesados en aprender a construir aplicaciones más seguras.
5. **Impacto social:** Al reducir la cantidad de aplicaciones vulnerables, se contribuirá a un entorno digital más seguro para todos.

En resumen, este proyecto busca no solo solucionar un problema específico relacionado con la falta de concienciación en ciberseguridad, sino también crear un impacto positivo a través de la educación y la prevención de ataques informáticos.

## 2.- Objetivo/s generales del proyecto

El principal objetivo de este proyecto es desarrollar una aplicación web educativa que combine la funcionalidad de una red social con un entorno de aprendizaje seguro para la exploración y mitigación de vulnerabilidades web comunes. A través de este proyecto, se busca fomentar la concienciación y la educación en ciberseguridad para estudiantes, desarrolladores y cualquier persona interesada en la protección de datos y sistemas web.

### Detalles del objetivo general:

1. **Proveer un entorno interactivo y educativo:** La aplicación estará diseñada para que los usuarios puedan experimentar de manera práctica con vulnerabilidades como inyecciones SQL, Cross-Site Scripting (XSS) y otros ataques comúnmente dirigidos a aplicaciones web. El objetivo es que los usuarios no solo comprendan los riesgos, sino también las medidas necesarias para prevenirlos.
2. **Simular una red social funcional:** Desarrollar una aplicación que incluya características propias de una red social, como la publicación de contenido, comentarios, likes y perfiles de usuario. Esto permitirá a los usuarios trabajar con un entorno que refleje escenarios reales de uso en plataformas digitales.
3. **Fomentar la conciencia sobre ciberseguridad:** Crear una herramienta que permita visualizar de manera clara cómo los errores de implementación o configuración pueden ser explotados por atacantes, fomentando así una cultura de desarrollo seguro.
4. **Brindar una guía para la mitigación de vulnerabilidades:** La aplicación no solo expondrá a los usuarios a vulnerabilidades, sino que también ofrecerá soluciones y mejores prácticas para corregir y evitar estos problemas en el futuro.
5. **Contribuir al aprendizaje práctico:** Facilitar un entorno en el que estudiantes de desarrollo web, ciberseguridad y áreas relacionadas puedan poner en práctica sus conocimientos en un entorno controlado y seguro.
6. **Impacto positivo en la comunidad de desarrolladores:** Al proporcionar esta herramienta, se espera aumentar el conocimiento colectivo sobre la importancia de la seguridad en el desarrollo web, disminuyendo la cantidad de aplicaciones vulnerables en el ecosistema digital.

En definitiva, este proyecto tiene como objetivo ser una herramienta única que no solo eduque, sino que también inspire a las personas a tomar medidas proactivas para construir aplicaciones web más seguras y robustas.



### 3.- Objetivos específicos

---

Los objetivos específicos del proyecto desglosan de manera concreta y precisa las acciones necesarias para alcanzar el objetivo general planteado. Cada uno de ellos aborda un aspecto clave del desarrollo del proyecto, garantizando que todas las etapas sean consideradas de forma estructurada y ordenada. A continuación, se presentan los objetivos específicos que guían la realización de este proyecto:

1. **Identificar los requisitos funcionales y no funcionales del sistema:** Analizar y documentar las necesidades de la aplicación web, asegurándose de que cumpla tanto con las funcionalidades propias de una red social como con las necesidades educativas en ciberseguridad.
2. **Diseñar una arquitectura de software robusta y segura:** Definir una estructura técnica clara y eficiente que permita la implementación de las funcionalidades de la red social y el entorno educativo. Esto incluye la selección de las tecnologías adecuadas como PHP, MySQL y JavaScript.
3. **Implementar características propias de una red social:** Desarrollar funcionalidades como la publicación de contenido, likes, comentarios y perfiles de usuario, proporcionando una experiencia de usuario similar a las plataformas reales.
4. **Incorporar vulnerabilidades de forma intencional y controlada:** Introducir elementos como inyección SQL, Cross-Site Scripting (XSS) y otros ataques comunes para permitir que los usuarios comprendan y experimenten cómo se explotan dichas vulnerabilidades.
5. **Diseñar un módulo educativo para la mitigación de vulnerabilidades:** Crear guías interactivas y prácticas que expliquen las soluciones a las vulnerabilidades presentes en el sistema, fomentando un aprendizaje efectivo y práctico.
6. **Garantizar la integración de medidas de seguridad:** Asegurarse de que las soluciones implementadas para corregir las vulnerabilidades sean aplicadas de manera efectiva, destacando buenas prácticas en ciberseguridad.
7. **Validar el sistema a través de pruebas funcionales y de seguridad:** Realizar pruebas exhaustivas para verificar que tanto las funcionalidades como las vulnerabilidades estén implementadas correctamente y que las medidas de mitigación sean efectivas.
8. **Crear una interfaz intuitiva y amigable para los usuarios:** Diseñar una experiencia de usuario clara y accesible que permita navegar fácilmente por las diferentes secciones de la aplicación, tanto para explorar las funcionalidades de la red social como para interactuar con los módulos educativos.
9. **Proveer documentación detallada del proyecto:** Elaborar una guía completa que incluya los detalles técnicos, las decisiones de diseño, las vulnerabilidades implementadas y las soluciones propuestas, sirviendo como referencia tanto para usuarios como para desarrolladores.
10. **Promover la concienciación sobre la importancia de la ciberseguridad:** Asegurarse de que la aplicación cumpla con su objetivo principal de educar y concienciar a los usuarios sobre las amenazas comunes en aplicaciones web y las mejores prácticas para enfrentarlas.

En conjunto, estos objetivos específicos aseguran que el proyecto aborde todas las fases necesarias para su éxito, desde la concepción hasta la implementación y validación final. Cada objetivo contribuye de manera directa al cumplimiento del objetivo general, garantizando que la aplicación sea una herramienta útil y efectiva tanto para el aprendizaje como para la concienciación en ciberseguridad.

## 4.- Contexto actual

---

### 4.1 Estado del arte

En el mundo actual, la seguridad en aplicaciones web es un tema de creciente importancia debido al aumento exponencial de ataques cibernéticos y la dependencia generalizada de sistemas basados en Internet. Este proyecto se encuentra relacionado con iniciativas educativas y de concienciación en el ámbito de la ciberseguridad. A continuación, se destacan algunos trabajos y proyectos relevantes existentes:

1. **OWASP (Open Web Application Security Project):** OWASP es una organización reconocida a nivel mundial que se dedica a mejorar la seguridad del software. Entre sus recursos destacados se encuentra OWASP Juice Shop, una aplicación intencionalmente vulnerable que sirve como herramienta educativa para que desarrolladores y estudiantes experimenten con vulnerabilidades comunes de manera segura.
2. **Hack The Box y TryHackMe:** Estas plataformas son entornos prácticos donde los usuarios pueden realizar ejercicios de hacking ético en sistemas simulados. Aunque se centran más en la seguridad general y pruebas de penetración, han inspirado a muchos proyectos educativos relacionados con ciberseguridad.
3. **DVWA (Damn Vulnerable Web Application):** DVWA es otra herramienta que simula vulnerabilidades comunes en aplicaciones web, como XSS o inyecciones SQL. Su objetivo es proporcionar un entorno controlado para que los usuarios aprendan sobre las vulnerabilidades más frecuentes.
4. **PortSwigger Web Security Academy:** Esta plataforma combina teoría y práctica, proporcionando lecciones interactivas y ejercicios en un entorno simulado para aprender sobre seguridad web. Ha sido una referencia clave en el desarrollo de proyectos educativos.
5. **Aplicaciones de redes sociales:** Muchas plataformas de redes sociales, como Facebook, Twitter e Instagram, sirven de referencia en cuanto a diseño funcional y usabilidad. Estas también han sido objetivo frecuente de ataques, lo que las convierte en una fuente de casos de estudio para la aplicación de principios de seguridad.

Sin embargo, a pesar de estas herramientas, existe una falta de aplicaciones que combinen la funcionalidad de una red social con un enfoque educativo en ciberseguridad, lo que motiva el desarrollo de este proyecto.

### 4.2 Conceptos clave

Para llevar a cabo este proyecto, es necesario comprender y trabajar con varios conceptos clave relacionados con el desarrollo web, la ciberseguridad y las redes sociales. A continuación, se describen los términos fundamentales:

1. **Ciberseguridad:** La disciplina que busca proteger sistemas, redes y datos contra accesos no autorizados, ataques o daños. En este proyecto, se enfocará en vulnerabilidades comunes como XSS, inyecciones SQL y ataques CSRF.
2. **Vulnerabilidades web:** Errores de configuración o desarrollo que permiten a atacantes comprometer la seguridad de una aplicación. Este proyecto se centra en exponer y mitigar vulnerabilidades típicas en entornos de desarrollo web.
3. **Red social:** Una plataforma que permite a los usuarios interactuar entre sí mediante la publicación de contenido, comentarios, likes y otras actividades. La funcionalidad de esta red social se simulará en el proyecto para crear un entorno realista.
4. **Educación interactiva:** Un método de enseñanza que utiliza actividades prácticas para mejorar la comprensión de conceptos complejos. Este enfoque guía el diseño de la aplicación para facilitar el aprendizaje de la ciberseguridad.



5. **Entorno controlado:** Un espacio seguro donde los usuarios pueden experimentar con vulnerabilidades sin poner en riesgo datos o sistemas reales. Este concepto es fundamental para garantizar que las pruebas se realicen de manera ética y responsable.
6. **Arquitectura web:** Conjunto de componentes y estructuras que definen el funcionamiento de una aplicación web. Incluye el uso de lenguajes de programación como PHP, JavaScript y MySQL para gestionar datos y funcionalidad.
7. **Mejores prácticas de seguridad:** Lineamientos que ayudan a minimizar riesgos, como la validación de entradas, el uso de conexiones seguras (HTTPS) y el almacenamiento seguro de contraseñas mediante hashing.

Al integrar estos conceptos clave, el proyecto busca ofrecer una herramienta que no solo sea funcional y educativa, sino también relevante dentro del panorama actual de la ciberseguridad y el desarrollo de software.

## 5.- Análisis de requisitos

La parte de análisis es fundamental y describe los modelos que has construido antes de ponerte a codificar. La mayoría de vosotros tiene poca experiencia en este punto luego es normal que no os salga de manera fluida y completa. Mi recomendación es que pongáis los diagramas que hicimos en clase, pero explícalos.

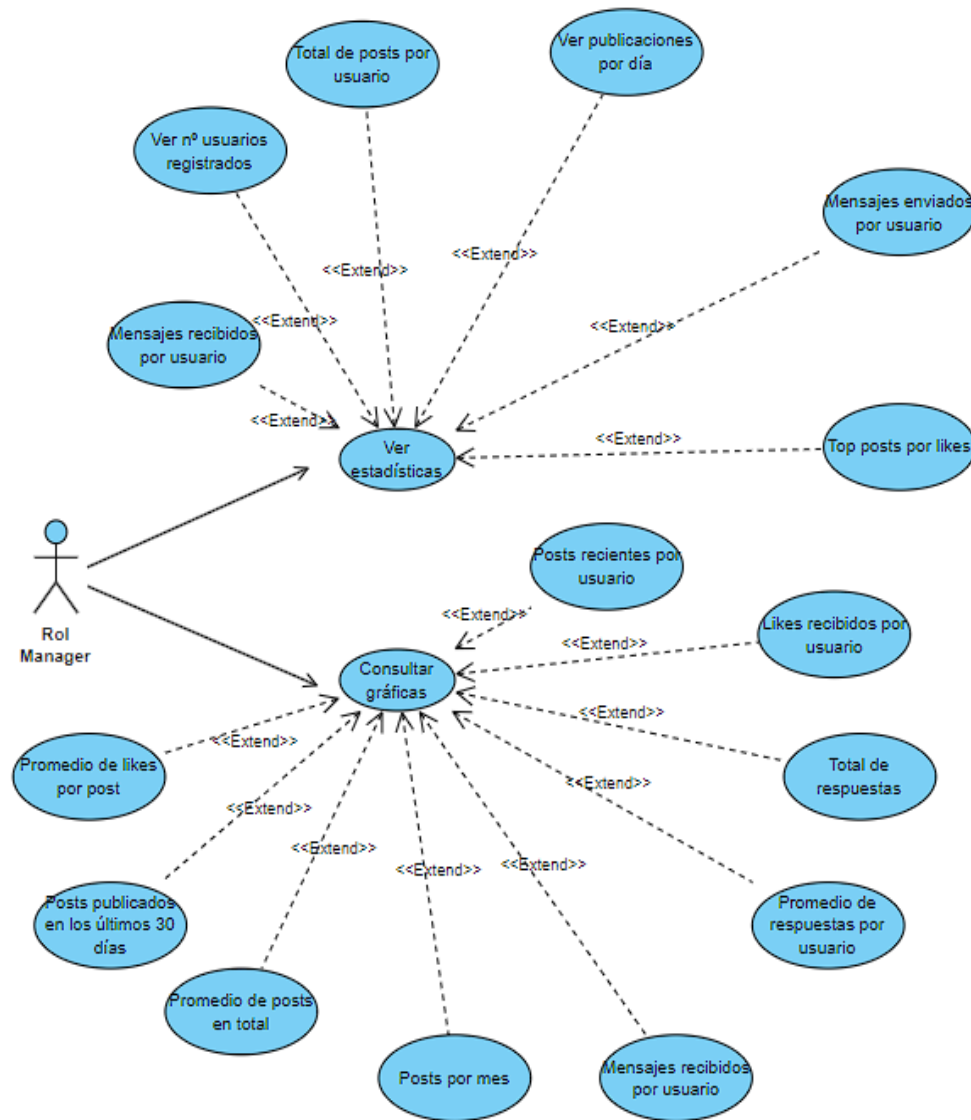
### 5.1.- Diagrama de casos de uso.

#### 5.1.1 Diagrama de casos de uso del rol Manager

Dentro de la administración del sitio web, se definen 3 roles: el manager, el administrador y el usuario sin privilegios. En esta sección se detallarán las funciones que puede llevar a cabo el manager, las cuales son las siguientes:

**Ver indicadores:** Dentro del panel de administración, el manager puede observar en tiempo real los indicadores clave, que incluyen ver el número de usuarios registrados, post publicados por usuario, número de respuestas en las publicaciones, número de publicaciones por día, número de post por mes, promedio de posts por usuario y el número de posts en un rango de tiempo.

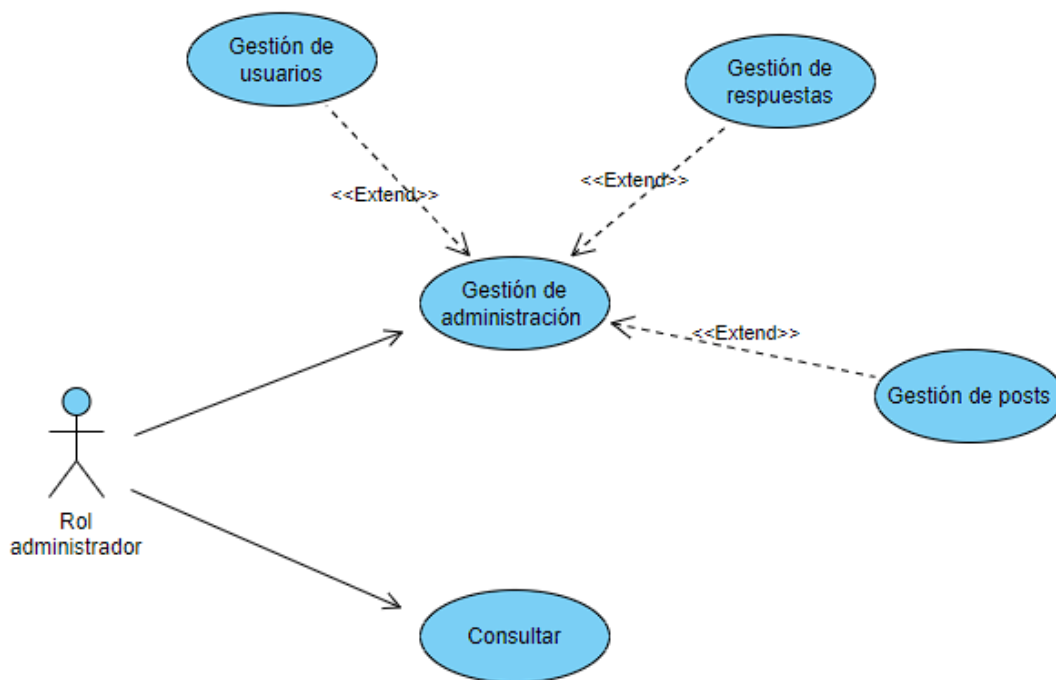
**Ver gráficas:** En la misma pantalla el manager tiene acceso a las gráficas que reflejan los posts publicados por usuario para poder visualizar el usuario que tiene más interacción en la aplicación.



### 5.1.2 Diagrama de casos de uso del rol Admin

Dentro de las funciones del administrador destacan varias que están implementadas como la gestión administrativa de la aplicación. En este caso destacan la gestión de posts y de los usuarios que albergan dentro de la aplicación.

- **Gestión de usuarios:** Una vez estamos logueados como administrador podemos ver un panel de administración de usuarios donde podemos ver todos los posts realizados. Eliminar el usuario que queramos independientemente del post que haya publicado ya que también se eliminará el post publicado por ese usuario ya que la base de datos está hecha en cascada. Otra función implementada es poder editar el usuario como su password, nombre de usuario o email.
- **Gestión de posts:** En esta gestión de posts podemos realizar lo mismo que con la gestión de los usuarios.
- **Gestión de respuestas:** En esta gestión de respuestas podemos realizar lo mismo que hemos mencionado anteriormente.



### 5.1.3 Diagrama de casos de uso del rol usuario

El objetivo de una aplicación como una red social con vulnerabilidades web es satisfacer la necesidad de poder postear en el momento que quieras del día un post y que los usuarios puedan visualizar lo que quieras expresar en ese momento.

El objetivo de una aplicación como una red social con vulnerabilidades web es satisfacer la necesidad de poder postear en el momento que quieras del día un post y que los usuarios puedan visualizar lo que quieras expresar en ese momento. 10

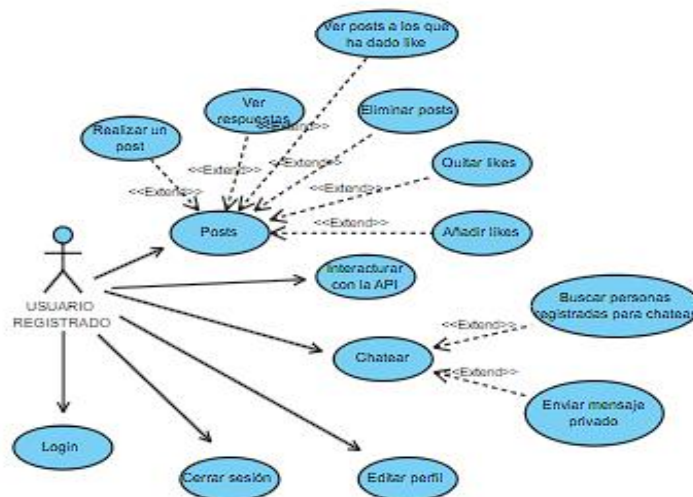
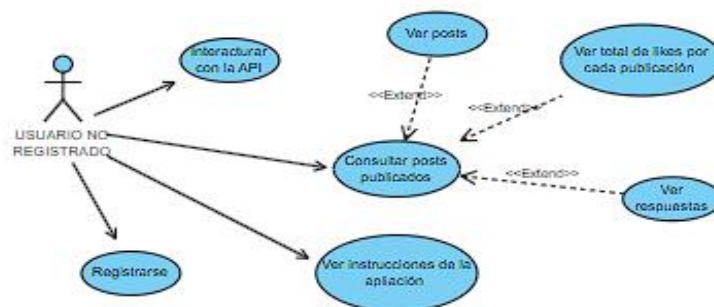
#### USUARIO NO REGISTRADO

- **Consultar posts publicados:** Un usuario no registrado puede consultar los posts publicados de otros usuarios.
- **Ver instrucciones de la aplicación:** Donde se detalla el fin de la aplicación y las vulnerabilidades que tiene la aplicación web para poder resolver y tomar conciencia de lo importante que es la ciberseguridad.
- **Ver respuestas:** Una vez un usuario realiza un post, habrán usuarios que puedan responder a ese mismo post y se pueda visualizar que usuarios han realizado una respuesta al post.
- **Registrarse:** Puede registrarse para poder acceder a la aplicación y poder usarla.
- **Interactuar con la API:** Podrá interactuar de forma totalmente proactiva con la API de virustotal.
- **Ver posts:** Podrá ver los posts publicados en la aplicación.
- **Ver total de likes de cada post:** Podrá ver el total de likes que tiene cada post.

#### USUARIO REGISTRADO

- **Creación de posts:** El usuario podrá realizar un post (publicación) adjuntando imagen o solo texto para que los demás usuarios puedan visualizarla.
- **Ver respuestas:** Una vez un usuario realiza un post, habrán usuarios que puedan responder a ese mismo post y se pueda visualizar que usuarios han realizado una respuesta al post.
- **Chatear:** Una vez logueado el usuario podrá visualizar un apartado donde tendrá un chat adjunto y podrá chatear por mensajes privados con los demás usuarios.

- **Editar perfil:** Cuando estas logueado, te aparecerá en el usuario un apartado para poder editar tus datos como correo, nombre de usuario o password.
- **Cerrar sesión:** Al estar logueado es esencial poder cerrar sesión para que otro usuario pueda loguearse con sus credenciales.
- **Interactuar con la API:** Podrá interactuar de forma totalmente proactiva con la API de virustotal.
- **Login:** Podrá loguearse de nuevo con sus credenciales.
- **Enviar mensaje privado:** Podrá interactuar con los usuarios registrados en la aplicación por mensaje privado.
- **Dar like:** Se podrá dar like a los posts de los usuarios.
- **Eliminar post:** En la sección de mi perfil se podrá eliminar un post que ha creado el usuario.



## 5.2.- Requisitos funcionales principales: funcionalidades que debe tener la aplicación.

En esta sección se detallan los requisitos funcionales principales del sistema VulnSocial, los cuales se dividen en dos categorías: Core y Operativos. A continuación se presentan detalladamente estos requisitos funcionales, resaltando su importancia y su impacto en el funcionamiento global de la aplicación VulnSocial.

### 5.2.1 Requisitos Funcionales Core

Los requisitos Core abarcan las funcionalidades fundamentales que son esenciales para el funcionamiento del sistema, que contribuyen directamente a la experiencia del usuario y al cumplimiento de los objetivos del negocio. Los requisitos Core del sistema son los siguientes:

#### 5.2.1.1 Gestión de usuarios

1. **Registro de usuarios:**
  - Permitir a los usuarios registrarse proporcionando un nombre de usuario único, correo electrónico, contraseña y una foto de perfil (avatar).
  - Validaciones para garantizar que los datos ingresados sean correctos y únicos.
2. **Inicio de sesión:**
  - Autenticación de usuarios registrados mediante correo electrónico y contraseña.
  - Gestión de sesiones seguras para los usuarios logueados.
3. **Edición de perfil:**
  - Permitir que los usuarios actualicen su información personal (nombre, correo electrónico, contraseña y avatar).
  - Validar los datos al momento de realizar los cambios.
4. **Roles de usuarios:**
  - Implementación de roles básicos: **Usuario** y **Manager**.
  - Permitir que el manager acceda a dashboards y estadísticas adicionales.

#### 5.2.1.2 Gestión de Posts

1. **Creación de posts:**
  - Los usuarios pueden crear posts con contenido de texto y multimedia (imágenes o videos).
  - Soporte para hashtags en los posts.
2. **Visualización de posts:**
  - Mostrar los posts en un feed principal con contenido ordenado cronológicamente.
  - Posibilidad de filtrar o buscar por hashtags y contenido.
3. **Edición y eliminación de posts:**
  - Permitir a los usuarios editar o eliminar sus propios posts.
  - Validar que solo el creador del post pueda realizar estas acciones.

### 5.2.1.3 Sistema de Likes

1. **Dar y quitar likes:**
  - Los usuarios pueden dar y quitar likes a los posts.
  - Solo se permite un like por usuario por post.
2. **Visualización de likes:**
  - Mostrar el número total de likes de cada post junto con un botón de interacción visual (corazón rojo/gris).
3. **Gestión de likes:**
  - Permitir al usuario ver los posts a los que les ha dado like, con opción de quitarlos desde su perfil.

### 5.2.1.4 Chat en Tiempo Real

1. **Mensajería privada:**
  - Permitir el envío y recepción de mensajes entre usuarios en tiempo real.
2. **Listado de conversaciones:**
  - Mostrar un listado de conversaciones con otros usuarios.
  - Posibilidad de acceder al historial de mensajes.
3. **Notificaciones de mensajes:**
  - Mostrar notificaciones de mensajes nuevos mientras el usuario está logueado.

### 5.2.1.5 Gestión de Comentarios y Respuestas

1. **Comentarios en posts:**
  - Los usuarios pueden dejar comentarios en los posts de otros usuarios.
  - Mostrar los comentarios organizados debajo del post.
2. **Respuestas a comentarios:**
  - Permitir a los usuarios responder a comentarios de manera anidada.

### 5.2.1.6 Panel de Control del Manager

1. **Dashboards de estadísticas:**
  - Visualizar estadísticas de uso, como:
    - Posts por usuario.
    - Likes por usuario y por fecha.
    - Mensajes enviados y recibidos.
    - Usuarios más activos.
    - Posts con más interacciones (likes y comentarios).
2. **Gestión de usuarios:**
  - Posibilidad de filtrar y visualizar información de los usuarios registrados.
  - Analizar la actividad reciente en la plataforma.

## 5.2.2 Requisitos no funcionales

Por otro lado, los requisitos Operativos son aquellas funcionalidades esenciales para el funcionamiento del sistema que, aunque no cumplen directamente los objetivos fundamentales del negocio, contribuyen significativamente a la experiencia del usuario y complementan las funcionalidades Core. A continuación, se muestran los requisitos funcionales operativos de la aplicación.



### 5.2.2.1 Rendimiento

- El sistema debe garantizar un tiempo de respuesta medio inferior a **1 segundo** en el acceso y carga de páginas principales como el feed de posts, el perfil del usuario y la funcionalidad de likes.
- La base de datos debe estar optimizada para manejar consultas de gran volumen, como la carga de posts, likes y mensajes, asegurando un rendimiento estable con hasta **500 usuarios concurrentes**.
- La aplicación debe ser capaz de soportar al menos **10,000 usuarios registrados** y **1,000 posts diarios**, sin comprometer el rendimiento general del sistema.

### 5.2.2.2 Seguridad

- **Autenticidad:** El sistema debe requerir autenticación obligatoria para acceder a funcionalidades críticas, como crear posts, enviar mensajes y dar likes.
- **Integridad:** Los datos almacenados deben ser protegidos contra modificaciones no autorizadas mediante validación de entradas tanto en el cliente como en el servidor.

### 5.2.2.3 Usabilidad

- La interfaz de usuario debe ser intuitiva, con un diseño **responsive** que se adapte correctamente a dispositivos móviles, tablets y pantallas de escritorio.
- El tiempo de aprendizaje del sistema no debe exceder los **15 minutos** para usuarios nuevos gracias a una navegación simple y explicaciones claras en los formularios y botones.
- La aplicación debe proporcionar mensajes de error claros y detallados cuando se produzcan fallos, con soluciones sugeridas para el usuario.
- Los usuarios deben poder realizar acciones frecuentes (crear posts, dar likes, enviar mensajes) con no más de **3 clics**.

### 5.2.2.4 Escalabilidad

- El sistema debe ser escalable para soportar un aumento significativo en el número de usuarios y datos. Esto incluye la capacidad de migrar a bases de datos distribuidas o utilizar servicios en la nube en caso de crecimiento.
- Debe ser posible implementar microservicios para manejar tareas específicas, como el procesamiento de mensajes o la gestión de likes.

### 5.2.2.5 Compatibilidad

- El sistema debe ser compatible con los navegadores modernos más utilizados: **Google Chrome, Mozilla Firefox, Microsoft Edge y Safari**.
- La aplicación debe garantizar una integración adecuada con servicios externos, como APIs de terceros (por ejemplo, para análisis de URLs).

## 5.4.- Descripción de los usuarios y sus necesidades.

En la aplicación, los usuarios se clasifican en varios roles con permisos y funcionalidades específicas. Cada rol tiene necesidades particulares que el sistema busca cubrir. A continuación, se detallan los roles existentes y las necesidades asociadas:

### 5.4.1 Usuario estándar (Usuario registrado)

Este es el rol principal y más común en la aplicación. Corresponde a cualquier usuario que se registre en el sistema y utilice las funcionalidades básicas.

- **Registro y autenticación:** Poder registrarse en el sistema de forma segura y autenticarse para acceder a sus funcionalidades.
- **Gestión de perfil:** Editar su información personal, como nombre de usuario, correo electrónico y foto de perfil.
- **Creación de contenido:** Publicar posts en la red social.
- **Interacción con otros usuarios:**
  - Dar y quitar likes a los posts de otros usuarios.
  - Comentar y responder a publicaciones.
  - Participar en chats privados con otros usuarios.
- **Acceso personalizado:** Visualizar solo las publicaciones relacionadas con su actividad.

### 5.4.2 Administrador

El administrador tiene permisos avanzados para gestionar el sistema. Su principal responsabilidad es mantener la aplicación en funcionamiento, gestionar usuarios y garantizar la seguridad.

- **Gestión de usuarios:** Ver y gestionar la lista completa de usuarios registrados.
- **Gestión de contenido:** Modificar o eliminar publicaciones.
- **Gestión del sistema:** Monitorear la actividad global, incluidos los logs de actividad de usuarios y las estadísticas del sistema.
- **Seguridad:** Garantizar la protección del sistema contra vulnerabilidades y accesos no autorizados.
- **Estadísticas:** Acceder a dashboards avanzados con datos como:
  - Total de usuarios activos.
  - Posts creados por día.
  - Número de likes y mensajes enviados.

### 5.4.3 Manager

El manager es un rol intermedio que tiene acceso a estadísticas y análisis del sistema. Este usuario puede visualizar métricas importantes para la toma de decisiones, pero no tiene acceso directo a la gestión de usuarios o contenido.

- **Acceso a dashboards:** Visualizar datos clave, como:
  - Usuarios más activos.
  - Tendencias de likes y mensajes enviados.
  - Promedios de respuestas y publicaciones por usuario.
- **Seguimiento de actividad:** Identificar patrones de uso y rendimiento de la aplicación para proponer mejoras.

### 5.4.4 Usuario no registrado

#### Descripción:

Un visitante es cualquier persona que accede a la aplicación sin estar registrada o autenticada. Su acceso está restringido a funcionalidades públicas.

#### Necesidades:

- **Exploración básica:** Navegar por las publicaciones públicas para conocer el contenido general de la aplicación.
- **Motivación para registrarse:** Visualizar un mensaje claro que incentive a registrarse para acceder a todas las funcionalidades.

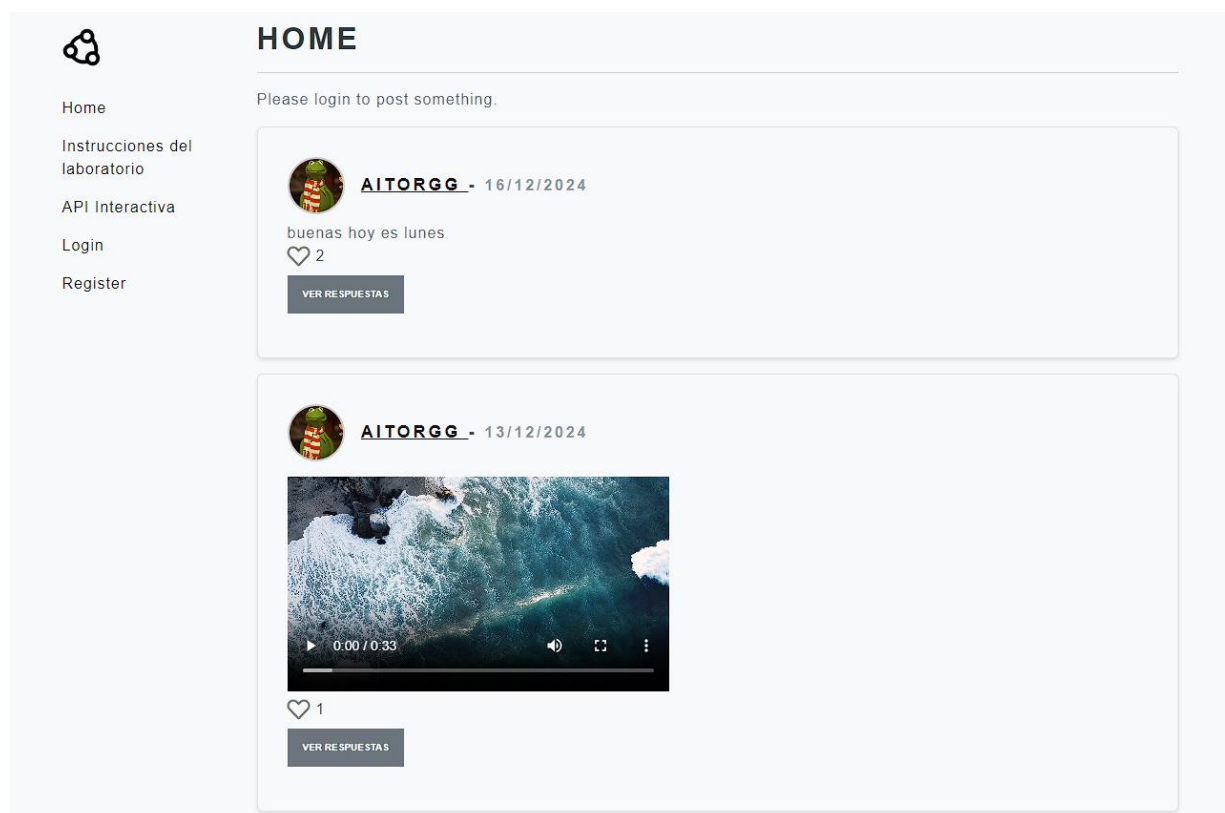
## 6.- Diseño de la aplicación

Este apartado tendrá las decisiones que se han tomado para la realización del proyecto:

### 6.1.- Mockups o wireframes o prototipos de la interfaz gráfica de usuario.

#### INTERFAZ DE USUARIO NO REGISTRADO

En esta interfaz que podemos ver a continuación vemos la parte principal de la aplicación donde se visualiza la parte principal de nuestra aplicación donde se muestran los posts de los usuarios, junto a los likes que tiene cada post junto a sus respectivas respuestas.



En las siguientes imágenes se puede visualizar la API interactiva que está creada con una API Key de virustotal donde se podrán consultar webs fraudulentas para ver si estamos comprometidos si nos metemos en el enlace.



- Home
- Login
- Register

## VERIFICACIÓN DE URL EN VIRUSTOTAL

Verificar

En este caso hemos creado un enlace malicioso con un acortador. En este caso hay muchos antivirus que detecta inofensivo, pero más abajo saltan como malicioso.



- Home
- Login
- Register

## VERIFICACIÓN DE URL EN VIRUSTOTAL

Verificar

Fecha del análisis: 9/1/2025, 17:04:46

**RESULTADOS POR MOTOR DE DETECCIÓN:**

Artists Against 419: Inofensivo Versión del motor: N/A Última modificación: N/A Nombre del motor: Artists Against 419
Acronis: Inofensivo Versión del motor: N/A Última modificación: N/A Nombre del motor: Acronis
Abusix: Inofensivo Versión del motor: N/A Última modificación: N/A Nombre del motor: Abusix
ADMINUSLabs: Inofensivo Versión del motor: N/A Última modificación: N/A Nombre del motor: ADMINUSLabs
Lionic: Inofensivo Versión del motor: N/A Última modificación: N/A Nombre del motor: Lionic

Axur: No detectado Versión del motor: N/A Última modificación: N/A Nombre del motor: Axur
benkow.cc: Inofensivo Versión del motor: N/A Última modificación: N/A Nombre del motor: benkow.cc
Bfore.Ai PreCrime: Malicioso Versión del motor: N/A Última modificación: N/A Nombre del motor: Bfore.Ai PreCrime
BitDefender: Inofensivo Versión del motor: N/A Última modificación: N/A Nombre del motor: BitDefender
Bkav: No detectado Versión del motor: N/A Última modificación: N/A Nombre del motor: Bkav

En este apartado es el de instrucciones del proyecto, donde se visualizan las instrucciones de este proyecto donde se recapitulan las vulnerabilidades web que podemos explotar y ver de qué manera funcionan para comprenderlas y tomar conciencia. Junto a ello se encuentra el top 10 de OWASP de vulnerabilidades web, para que podamos cómo funciona cada una de ellas detalladamente.



Home  
Login  
Register

## INSTRUCCIONES DEL PROYECTO

Este proyecto tiene como objetivo mostrar vulnerabilidades web comunes para educar sobre ciberseguridad. Explora cada vulnerabilidad para comprender sus riesgos y cómo mitigarlas.

### INSTRUCCIONES DEL PROYECTO

A continuación, se detallan los pasos y características de la aplicación que debes completar para el proyecto. Sigue estas instrucciones cuidadosamente para cumplir con los objetivos establecidos.

- Implementar la autenticación con vulnerabilidades controladas.
- Diseñar el CRUD de usuarios y publicaciones.
- Integrar dashboards estadísticos utilizando Chart.js.
- Probar y documentar las vulnerabilidades web.

VER INSTRUCCIONES

#### 1. INYECCIÓN SQL

Explora cómo los ataques de inyección SQL afectan a las aplicaciones web.

VER DETALLES

#### 2. CSRF

Descubre cómo los ataques CSRF pueden engañar a los usuarios para realizar acciones no deseadas sin su consentimiento.

VER DETALLES

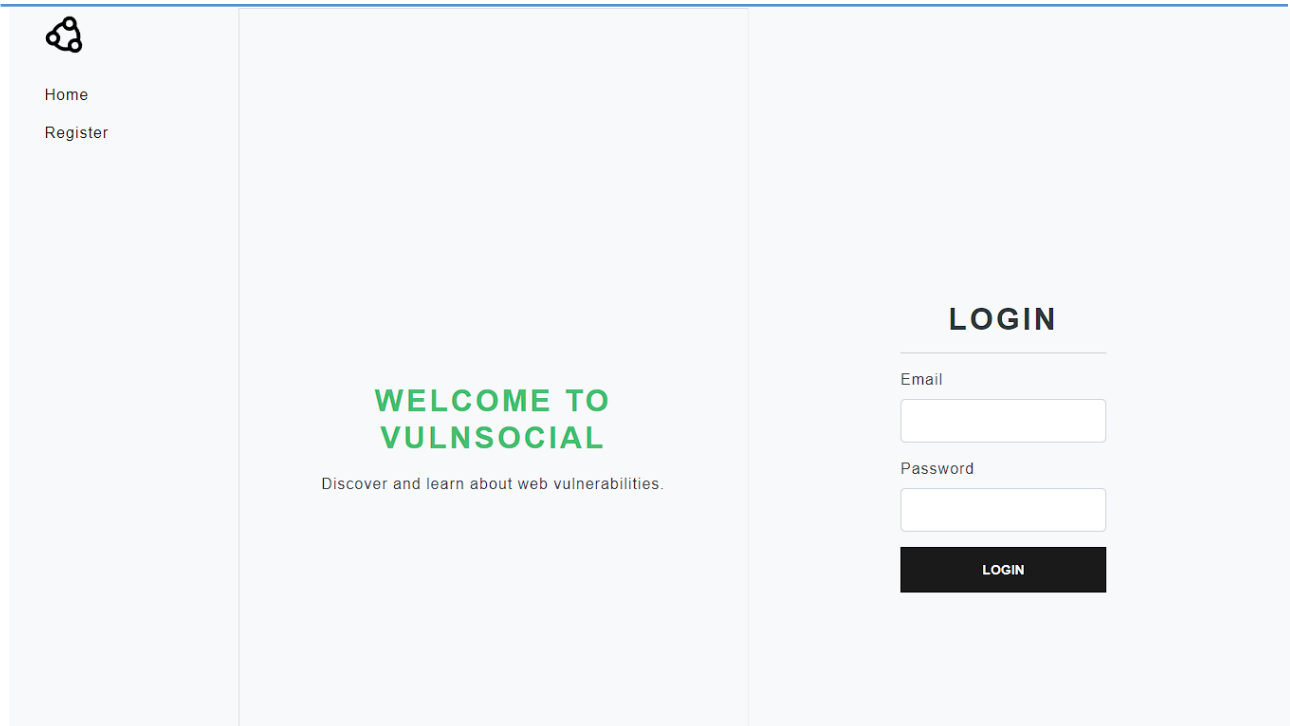
#### 3. SSRF

Analiza cómo los ataques SSRF permiten a los atacantes interactuar con recursos internos de la red a través de solicitudes maliciosas.

VER DETALLES

En este apartado nos encontramos con el Login y el Registro para poder loguearnos o registrarnos con una nueva cuenta.

18



The image shows the login page of the Vulnsocial application. It features a three-column layout. The left column contains a logo and navigation links for 'Home' and 'Register'. The middle column displays a 'WELCOME TO VULNSOCIAL' message with the tagline 'Discover and learn about web vulnerabilities.' The right column contains a 'LOGIN' form with fields for 'Email' and 'Password', and a 'LOGIN' button.

Home  
Register

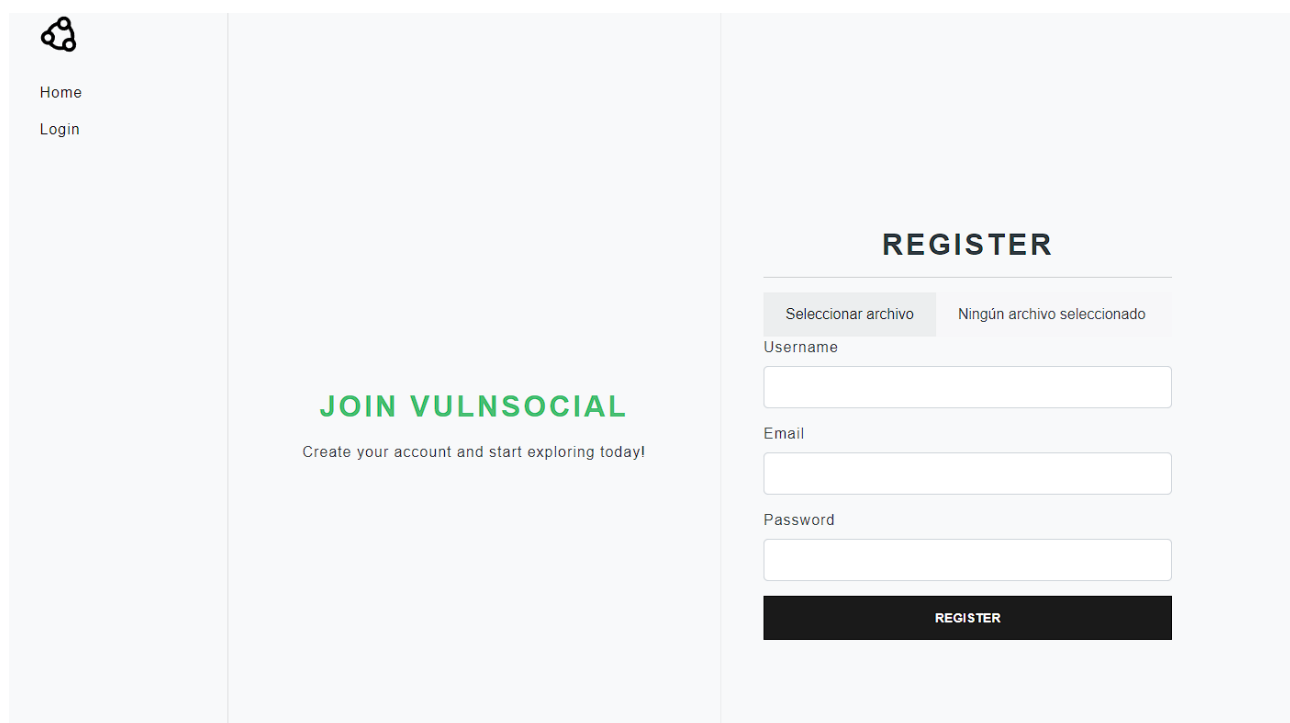
**WELCOME TO VULNSOCIAL**  
Discover and learn about web vulnerabilities.

**LOGIN**

Email

Password

**LOGIN**



The image shows the register page of the Vulnsocial application. It features a three-column layout. The left column contains a logo and navigation links for 'Home' and 'Login'. The middle column displays a 'JOIN VULNSOCIAL' message with the tagline 'Create your account and start exploring today!'. The right column contains a 'REGISTER' form with a file selection dropdown, and fields for 'Username', 'Email', and 'Password', and a 'REGISTER' button.

Home  
Login

**JOIN VULNSOCIAL**  
Create your account and start exploring today!

**REGISTER**

Seleccionar archivo Ningún archivo seleccionado

Username

Email

Password

**REGISTER**

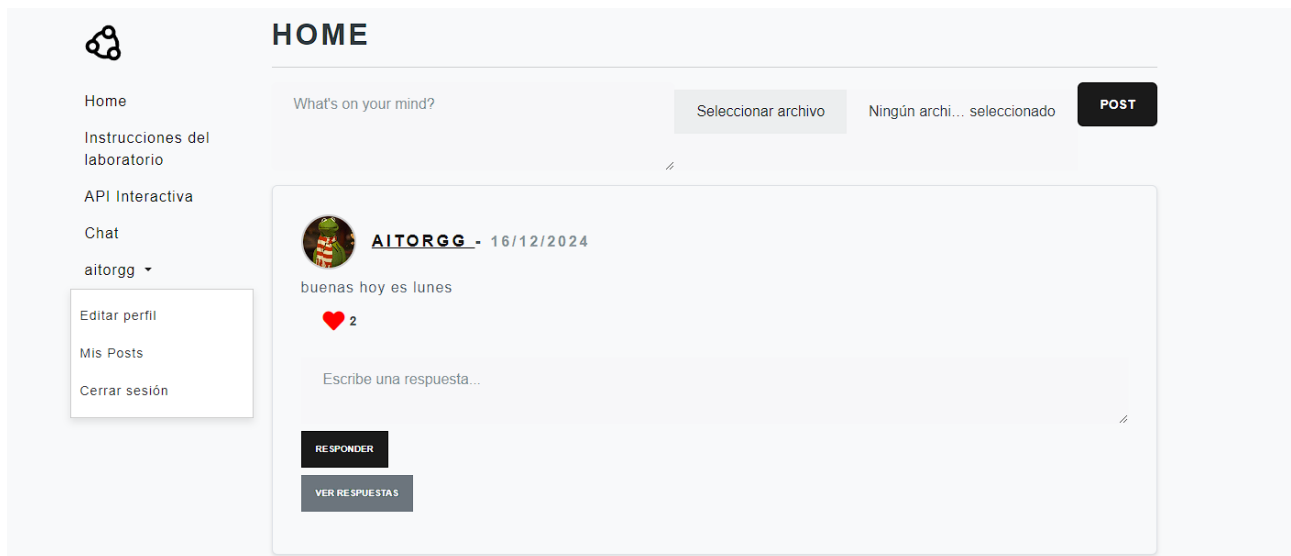
## INTERFAZ DE USUARIO REGISTRADO

En esta interfaz a diferencia del anterior es con un usuario previamente registrado y logueado. Donde podemos observar a diferencia de las imágenes anteriores, son que se nos despliegan diferentes opciones como:

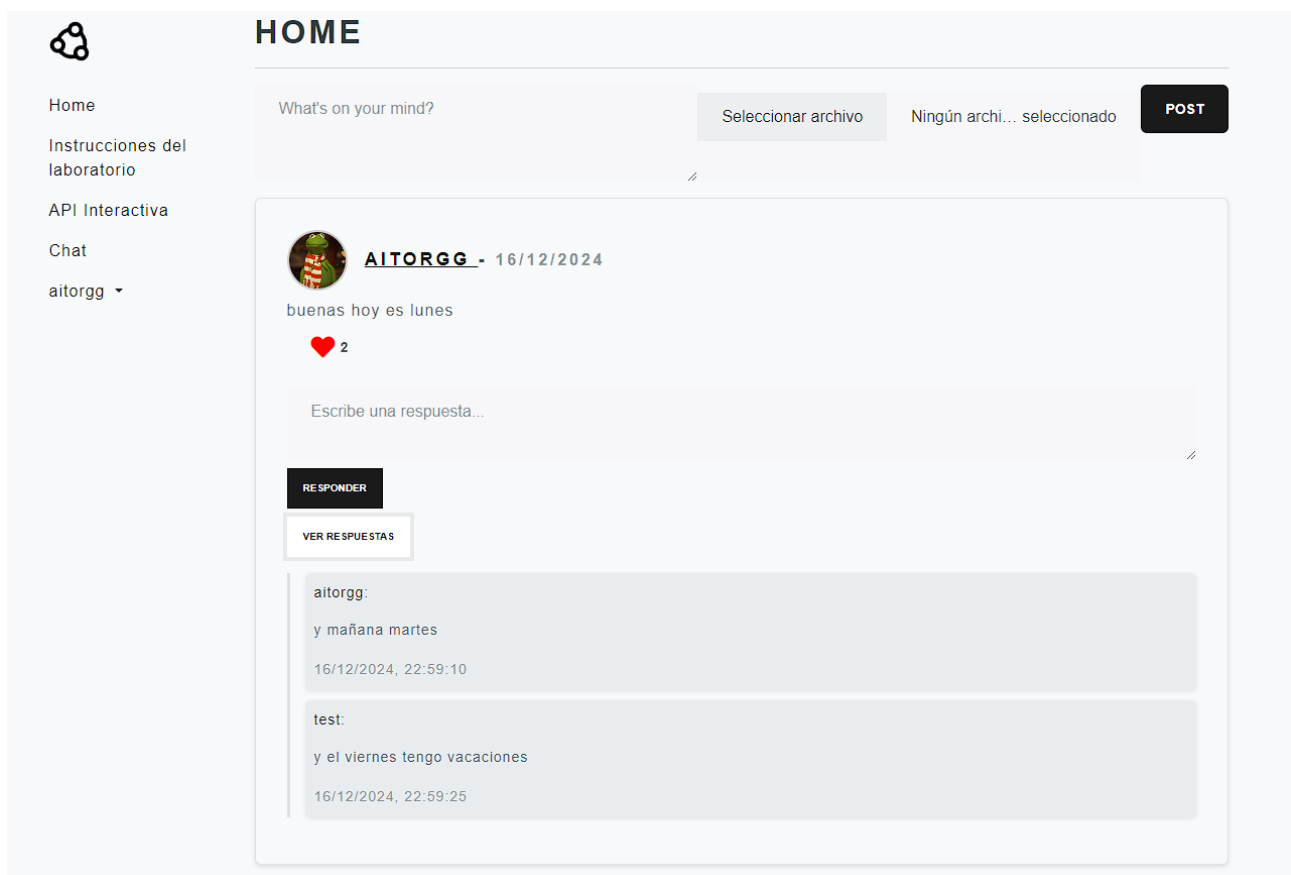
- Chat
- Editar Perfil
- Mis Posts



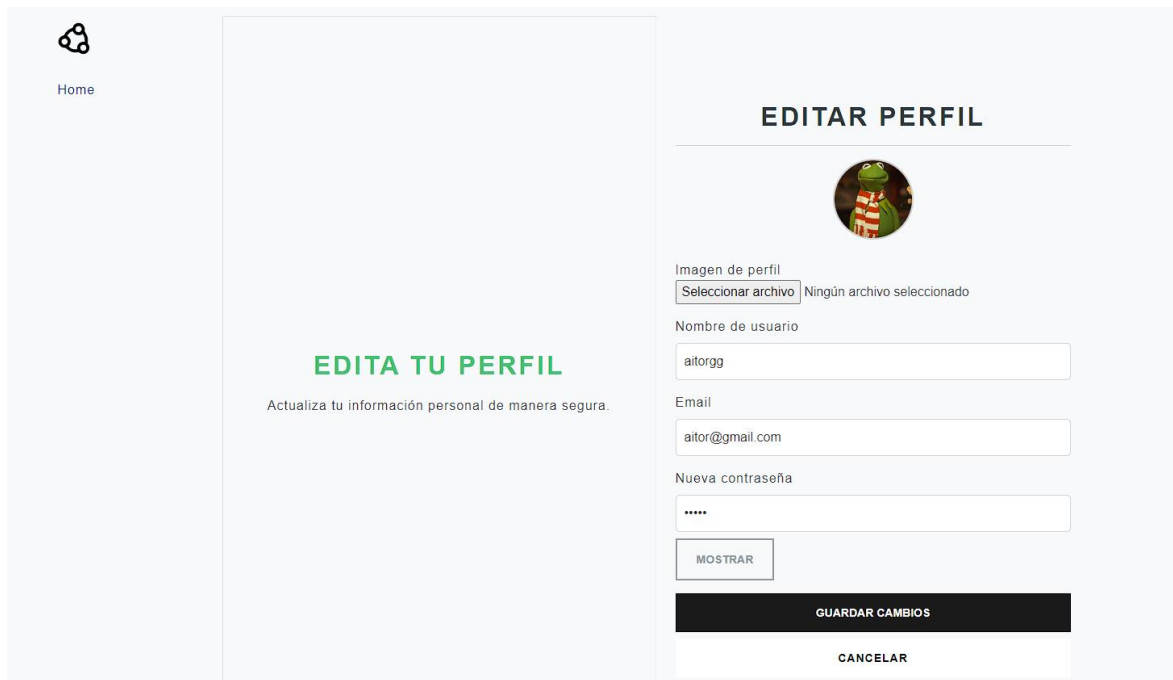
- Cerrar Sesión
- Responder
- Postear



Como estamos viendo si le da a ver respuestas se podrán visualizar las respuestas de los usuarios en ese post en específico. También podremos dar like a la publicación o quitar el like. En esta pantalla podremos también postear y subir imágenes.



Como se ha dictado antes, tenemos la edición de perfil donde podremos tocar nuestros datos y tener también una previsualización de nuestra imagen para poder cambiarla y poder mostrar la password según queramos.



Home

## EDITAR PERFIL

Imagen de perfil  
Seleccionar archivo Ningún archivo seleccionado

Nombre de usuario  
aitorgg

Email  
aitor@gmail.com

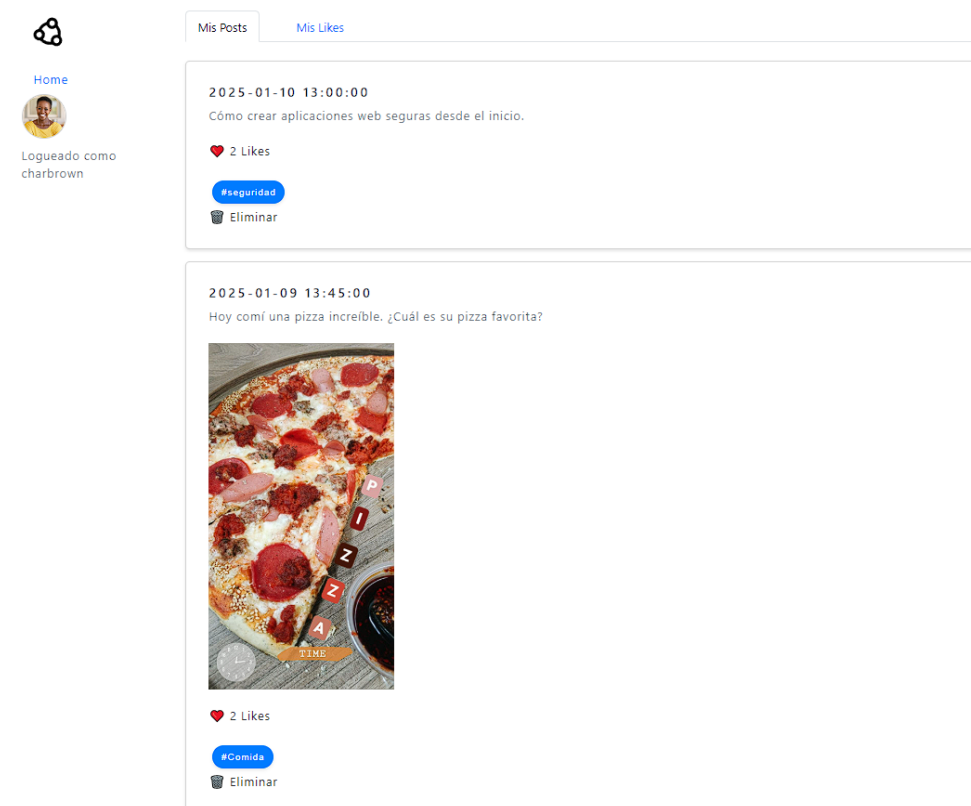
Nueva contraseña  
\*\*\*\*\*

MOSTRAR

GUARDAR CAMBIOS

CANCELAR

Este es el apartado Mis Posts, donde será nuestro perfil principal y podremos ver los posts que hemos publicado a la par que poder eliminarlos y podemos ver los likes que hemos dado en los posts de los usuarios y también eliminarlos.



Home

Logueado como charbrown

## Mis Posts

2025-01-10 13:00:00

Cómo crear aplicaciones web seguras desde el inicio.

2 Likes

#seguridad

Eliminar

2025-01-09 13:45:00

Hoy comí una pizza increíble. ¿Cuál es su pizza favorita?

2 Likes

#Comida

Eliminar



Home

Logueado como  
charbrown

Mis Posts

Mis Likes



LIAM\_GALLAGHER

9/1/2025, 16:30:00

Este fin de semana se realiza un evento de tecnología en mi ciudad, ¡no me lo quiero perder!



3

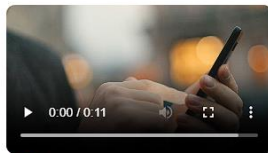
#EventoTech



EMMA\_WATSON

9/1/2025, 11:00:00

¡Nuevo video tutorial sobre ciberseguridad! Échenle un vistazo.



1

#Ciberseguridad

En este apartado podemos ver el Chat, donde se visualizan los usuarios que están registrados y podremos socializar con ellos por privado. Mantener una conversación al igual que se podrá visualizar si está online o desconectado.



Home

Logueado como  
charbrown

charbrown

Activo ahora

Seleccione un usuario para iniciar  
el chat

liam\_gallagher

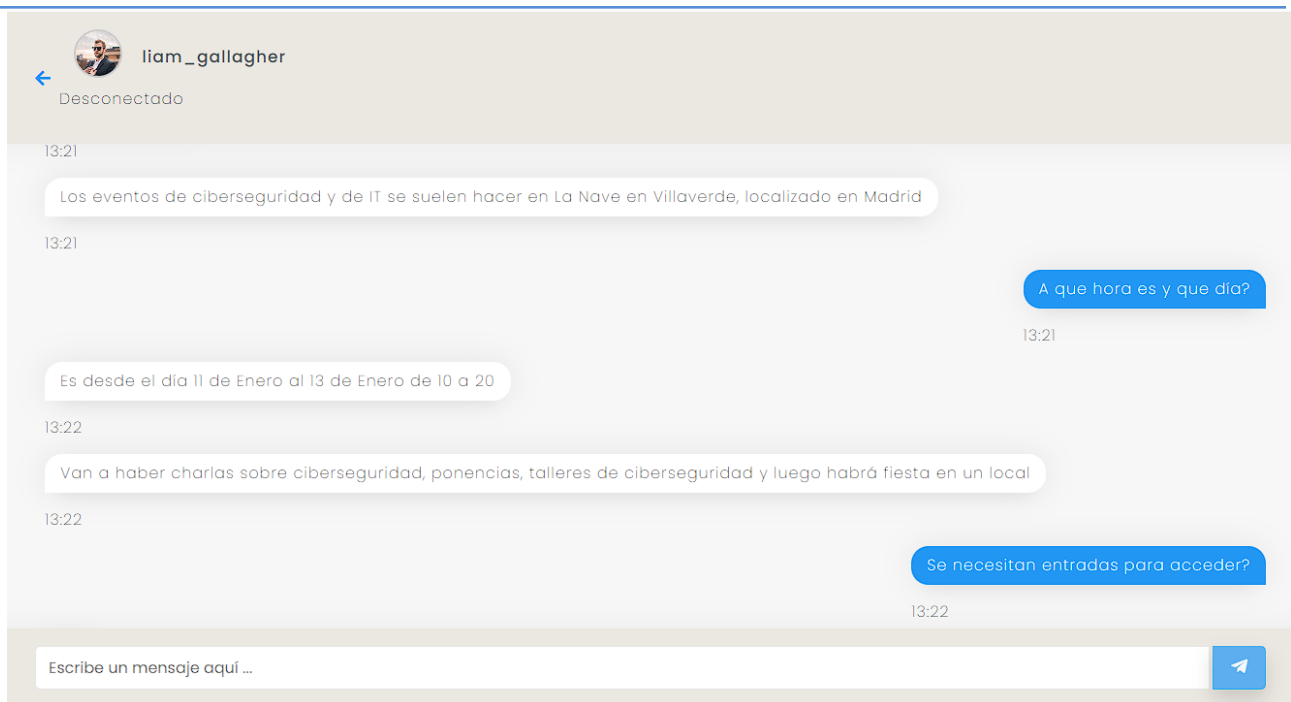
[Sí! Estuvo genial hubo charl...](#)

aitorgg

[No hay mensajes disponibles](#)

emma\_watson

[No hay mensajes disponibles](#)

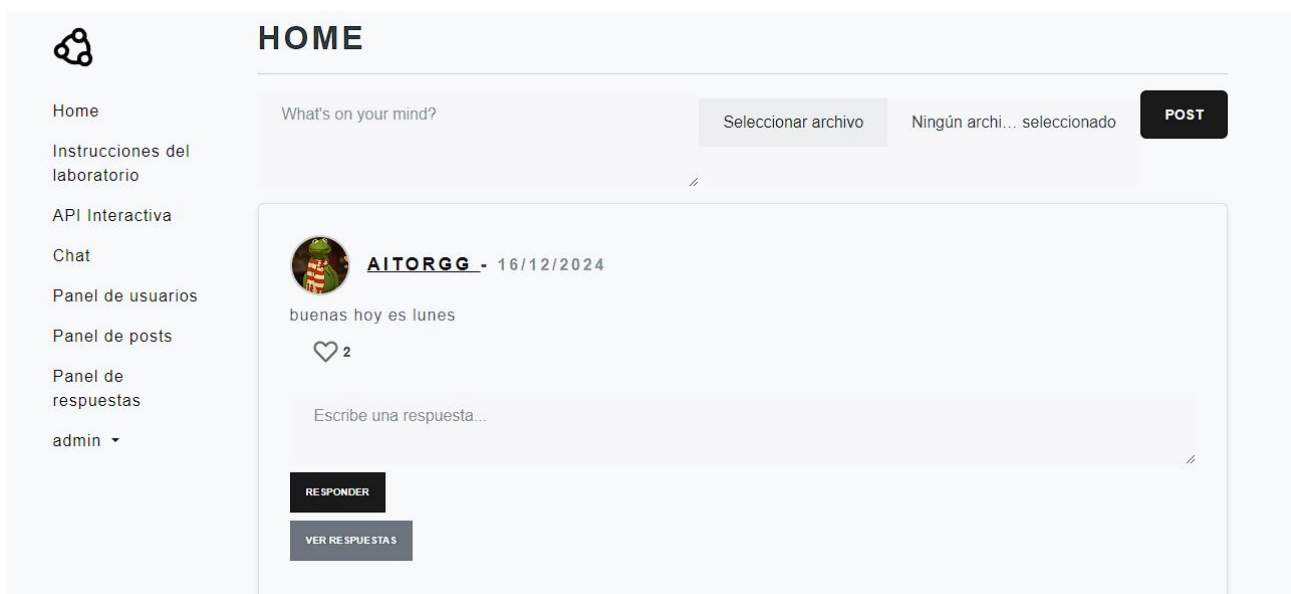


## INTERFAZ DE USUARIO ADMINISTRADOR

En este apartado tendremos las nuevas opciones como administrador como son:

- Panel de usuarios
- Panel de posts
- Panel de respuestas

En estas diferentes opciones podremos eliminar y editar los usuarios, posts y respuestas.





Home

Logueado como admin

## PANEL DE USUARIOS

ID	USERNAME	EMAIL	PASSWORD	ADMIN	ELIMINAR	EDITAR
1	admin	admin@gmail.com	admin	Sí	ELIMINAR	EDITAR
2	argus	argus@socially	argus	No	ELIMINAR	EDITAR
3	zeus	zeus@socially	zeus	No	ELIMINAR	EDITAR
4	pluto	pluto@socially	pluto	No	ELIMINAR	EDITAR
5	ares	ares@socially	ares	No	ELIMINAR	EDITAR
7	aitorgg	aitor@gmail.com	aitor	No	ELIMINAR	EDITAR
8	test	test@gmail.com	test	No	ELIMINAR	EDITAR

7

aitorgg

aitor@gmail.com

aitor

No

ELIMINAR

EDITAR

8

test

test@gmail.com

test

No

ELIMINAR

EDITAR

15

manager

ELIMINAR

EDITAR

18

Fernando

ELIMINAR

EDITAR

23

Gustavo

ELIMINAR

EDITAR

30

pruebafotos

pruebafotos@gmail.com

12345678

No

ELIMINAR

EDITAR

33

login

loginhack10@gmail.com

login12345

No

**¿Estás seguro?**

¡No podrás recuperar este post después de eliminarlo!


Sí, eliminar

Cancelar

EDITOR USUARIO	
Username	<input type="text" value="pruebafotos"/>
ID de Usuario	<input type="text" value="30"/>
Email	<input type="text" value="pruebafotos@gmail.com"/>
Password	<input type="text" value="12345678"/>
Admin	<input type="text" value="No"/>
<input type="button" value="GUARDAR"/> <input type="button" value="CERRAR"/>	

## INTERFAZ DE USUARIO MANAGER

En esta interfaz como administrador manager tenemos a diferencia del resto un panel de manager donde se podrán visualizar diferentes estadísticas y gráficas a tiempo real de la aplicación.

  
Home

Instrucciones del laboratorio

API Interactiva

Chat


Panel de Manager

manager ▾


## HOME

What's on your mind?


Seleccionar archivo Ningún archi... seleccionado

**AITORGG** · 16/12/2024


buenas hoy es lunes

 2

Escribe una respuesta...

**AITORGG** · 15/12/2024

esto es un xss

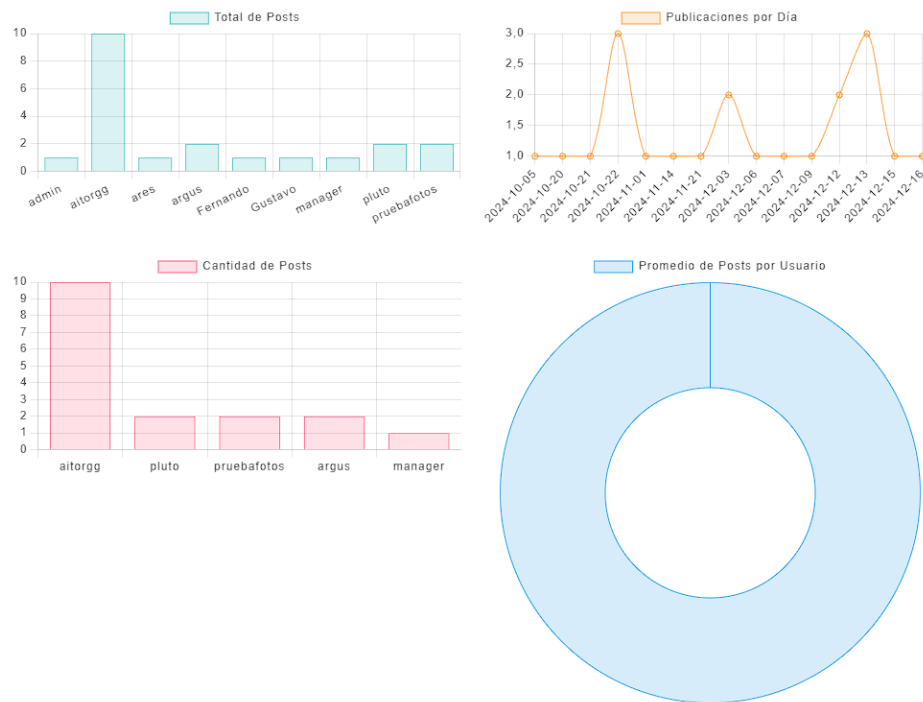
 0

Escribe una respuesta...



[Home](#)Logueado como  
manager

## DASHBOARD MANAGER



## 6.2.- Arquitectura del sistema.

La arquitectura del sistema implementado sigue el modelo **Cliente-Servidor**, utilizando un enfoque **MVC (Modelo-Vista-Controlador)** para estructurar el código de forma clara y escalable. A continuación, se detallan los componentes principales:

### Cliente

El cliente es la interfaz gráfica que interactúa directamente con el usuario. Está desarrollado con **HTML**, **CSS** y **JavaScript** y proporciona una experiencia dinámica gracias al uso de tecnologías como **AJAX** para realizar solicitudes asíncronas al servidor. Este lado incluye las siguientes características:

- Formulario de inicio de sesión y registro.
- Panel de administración con **CRUDs** dinámicos para usuarios y publicaciones.
- Secciones de interacción con funcionalidades como "Me gusta", respuestas a posts y un chat en tiempo real.
- Integración con la API de VirusTotal para la verificación de archivos o URLs.

### Servidor

El servidor está desarrollado en **PHP** y maneja las peticiones enviadas por el cliente. A través del servidor se procesan los datos y se interactúa con la base de datos **MySQL**. Las principales responsabilidades del servidor son:

- Gestión de roles, incluyendo permisos de administrador, usuarios regulares y un rol de manager para visualizar estadísticas.

- Procesamiento de datos para las publicaciones, respuestas y subida de archivos.
- Implementación de vulnerabilidades controladas para cumplir el objetivo de concienciar sobre ciberseguridad.

## Base de Datos

El sistema utiliza **MySQL** como sistema de gestión de bases de datos, organizado en varias tablas que representan los elementos principales de la aplicación. Entre ellas:

- Tabla de usuarios: Incluye información sobre nombres, roles, contraseñas encriptadas y más.
- Tabla de publicaciones: Almacena el contenido, imágenes asociadas y metadatos como la fecha de publicación.
- Tabla de respuestas: Relaciona los comentarios realizados con las publicaciones específicas.
- Tabla de registros de actividad: Almacena logs para auditar las operaciones importantes realizadas en el sistema.
- Tabla de vulnerabilidades: Almacena las vulnerabilidades del owasp TOP 10 para que se rellenen automáticamente cuando un usuario puede poner un hashtag.
- Tabla de likes: Guarda los ID de los likes que dan cada usuario en cada post.

## Flujo de Trabajo

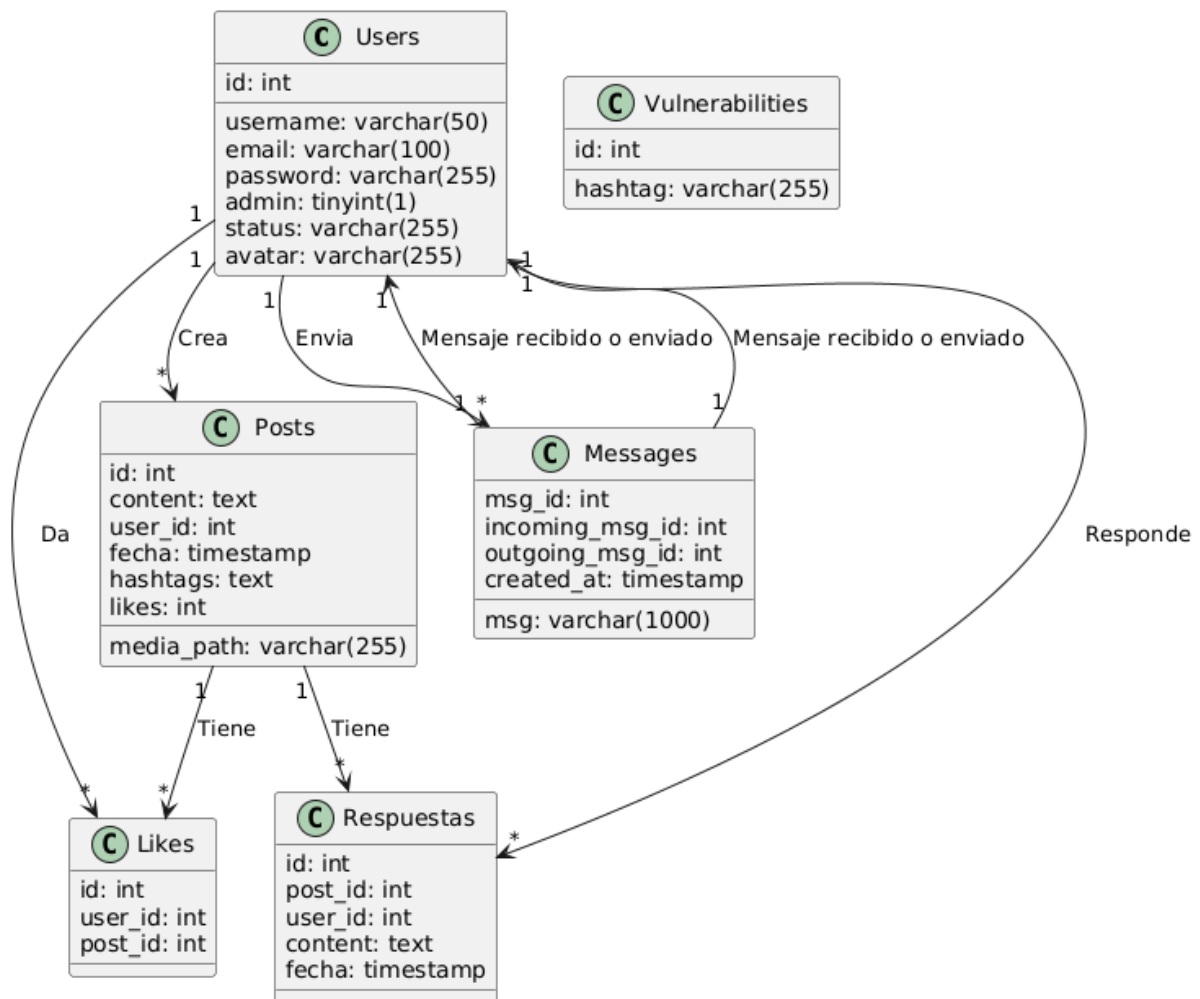
1. El usuario realiza una solicitud desde la interfaz cliente, como publicar un post o verificar una url en VirusTotal.
2. El servidor recibe la solicitud, la procesa y, si es necesario, interactúa con la base de datos.
3. El servidor responde al cliente con los datos solicitados, que son renderizados dinámicamente en la interfaz utilizando JavaScript.

## Beneficios de la Arquitectura

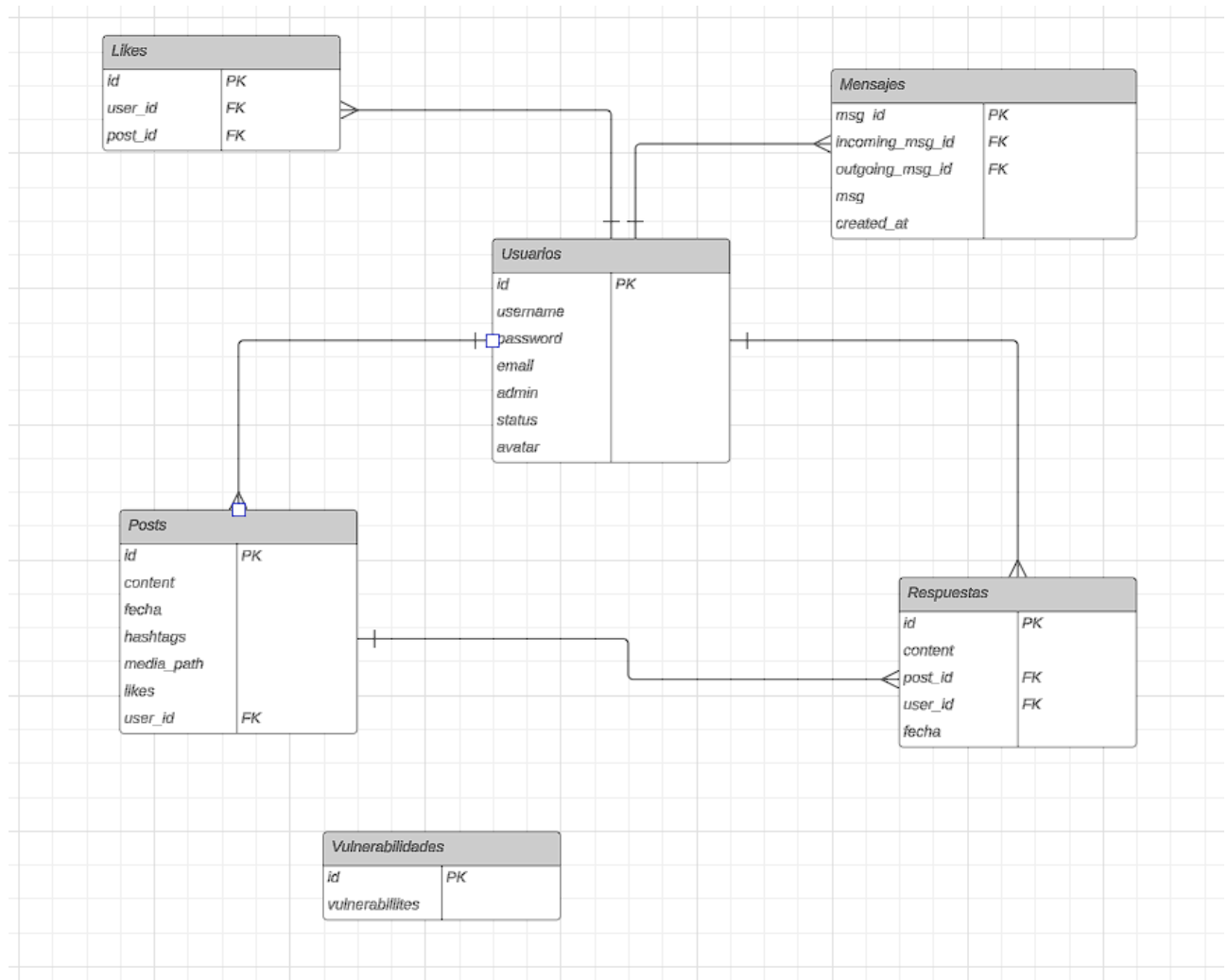
- **Modularidad:** La separación en capas facilita el mantenimiento y la expansión del sistema.
- **Escalabilidad:** Nuevas funciones, como estadísticas avanzadas o más integraciones de APIs, pueden añadirse sin afectar las partes existentes.
- **Seguridad:** La división cliente-servidor permite implementar validaciones en el backend y evitar la exposición de datos sensibles.

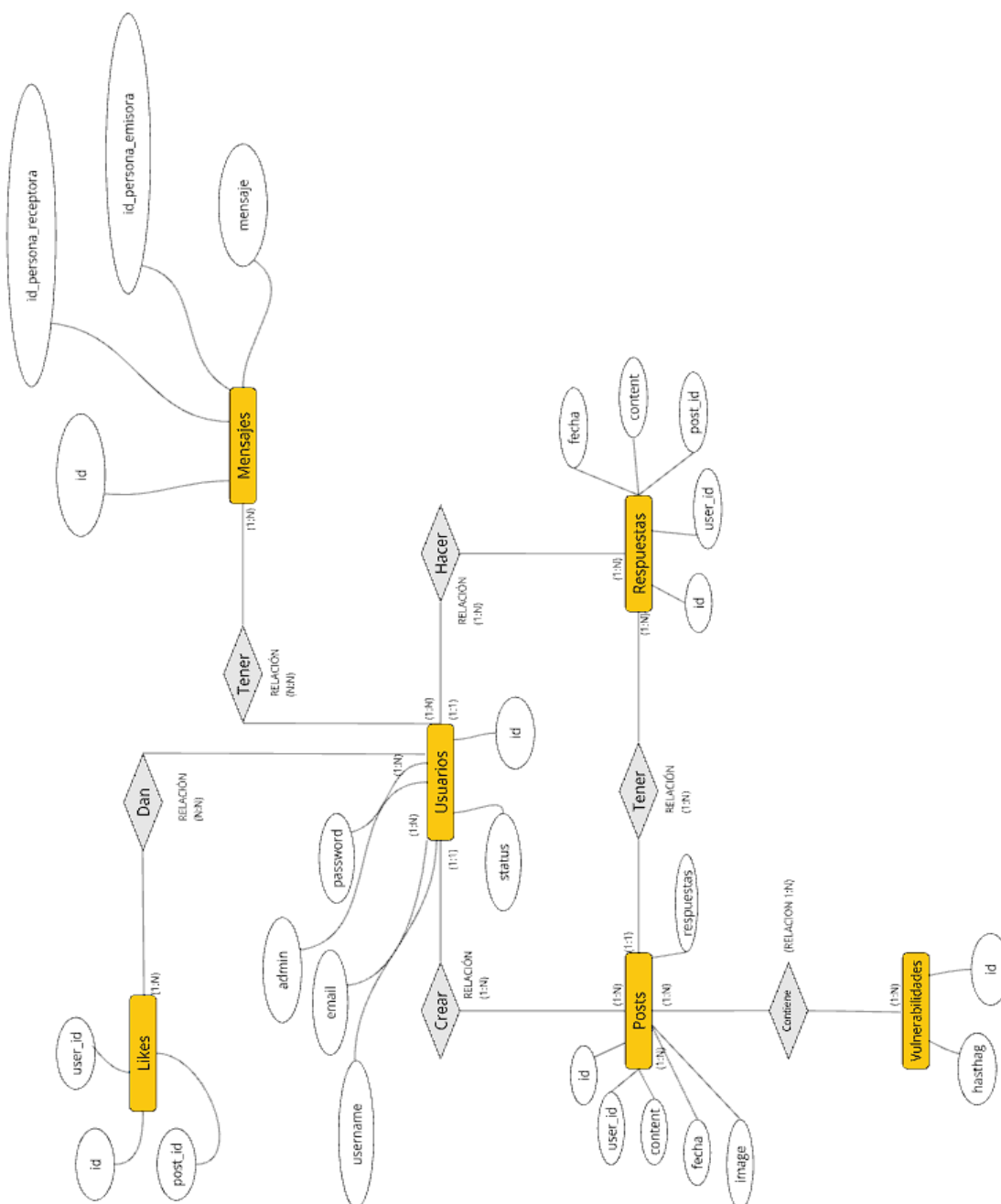
Esta arquitectura asegura una experiencia de usuario fluida mientras cumple con los objetivos educativos del proyecto, combinando funcionalidad práctica con elementos que refuercen la concienciación en ciberseguridad.

### 6.3.- Diagramas de clases y de entidad-relación.



## 6.4.- Diseño de la base de datos: esquemas y tablas.









## 7.- Desarrollo de la aplicación

---

### 7.1.- Tecnologías y herramientas utilizadas (lenguajes de programación, frameworks, bases de datos, etc.).

#### 7.1.1 Parte cliente

La parte cliente de la aplicación VulnSocial se ha desarrollado utilizando JavaScript puro junto con Bootstrap y jQuery. El uso de JavaScript puro permite una mayor flexibilidad y control sobre la lógica de la aplicación, facilitando la implementación de interacciones dinámicas. Además, la inclusión de jQuery simplifica la manipulación del DOM y la gestión de eventos, lo que mejora la experiencia del usuario.

Dado que VulnSocial es un CTF enfocado en la explotación de vulnerabilidades web, en algunos casos no se implementarán controles de errores en la validación de entradas. Esto permitirá a los participantes experimentar con técnicas de explotación, como inyecciones SQL y XSS, en un entorno seguro y controlado.

#### 7.1.2 Parte servidora

La parte servidora de VulnSocial se desarrolla en PHP, centrándose en la sanitización de datos y la implementación de medidas de seguridad. En esta parte, se realizarán validaciones en ciertos casos, pero habrá intencionadamente situaciones donde se omitirán controles de errores para permitir que los usuarios encuentren y exploten vulnerabilidades web.

##### **Autenticación**

Para el sistema de autenticación de usuarios, se utilizará un enfoque básico donde los usuarios se autenticarán mediante credenciales que se gestionan en el servidor. Se implementará una lógica simple que permitirá a los usuarios iniciar sesión y acceder a las funcionalidades de la aplicación. Sin embargo, en algunos casos, no se proporcionarán mensajes de error claros para permitir la exploración de vulnerabilidades en el proceso de autenticación.

##### **Sanear y validar datos**

Para evitar la entrada a nuestro servidor y a la base de datos de código malicioso o de datos incoherentes, hemos saneado y validados los datos necesarios que llegan desde el cliente para la inserción de usuarios. Para ello he usado **expresiones regulares y funciones**, que nos permite según nuestro criterio poder sanear y validar los datos antes siquiera de que podamos llegar a obtenerlos o procesarlos para la posterior entrada a la base de datos. Realizamos la comprobación de que un email esté escrito correctamente, que haya un mínimo de caracteres o quitamos los espacios de cada dato.

##### **Seguridad**

En el apartado de seguridad, aunque se aplicarán ciertas medidas, también se dejarán espacios deliberados para que los usuarios experimenten con la explotación de fallos de seguridad. Por ejemplo, se usarán prácticas de sanitización para proteger la base de datos contra inyecciones, pero habrá puntos en los que se permitirán entradas maliciosas para fomentar el aprendizaje.

La entrada de datos se validará y saneará en algunos casos, utilizando funciones PHP para eliminar caracteres peligrosos. Esto proporcionará a los usuarios la oportunidad de observar cómo estas vulnerabilidades pueden ser aprovechadas en aplicaciones web reales.

#### 7.1.3 Base de datos

MySQL ha sido la base de datos seleccionada para llevar a cabo esta aplicación. Se trata de un sistema de gestión de bases de datos relacional que ofrece una sólida escalabilidad y flexibilidad. MySQL almacena datos en tablas organizadas en filas y columnas, permitiendo relaciones entre diferentes conjuntos de datos y garantizando la integridad de la información.

### PHPMyAdmin

Para la gestión de la base de datos, se ha utilizado PHPMyAdmin, una herramienta de software libre escrita en PHP que proporciona una interfaz web fácil de usar para administrar bases de datos MySQL. PHPMyAdmin permite realizar operaciones como crear, modificar y eliminar bases de datos, tablas y registros de manera intuitiva y eficiente.

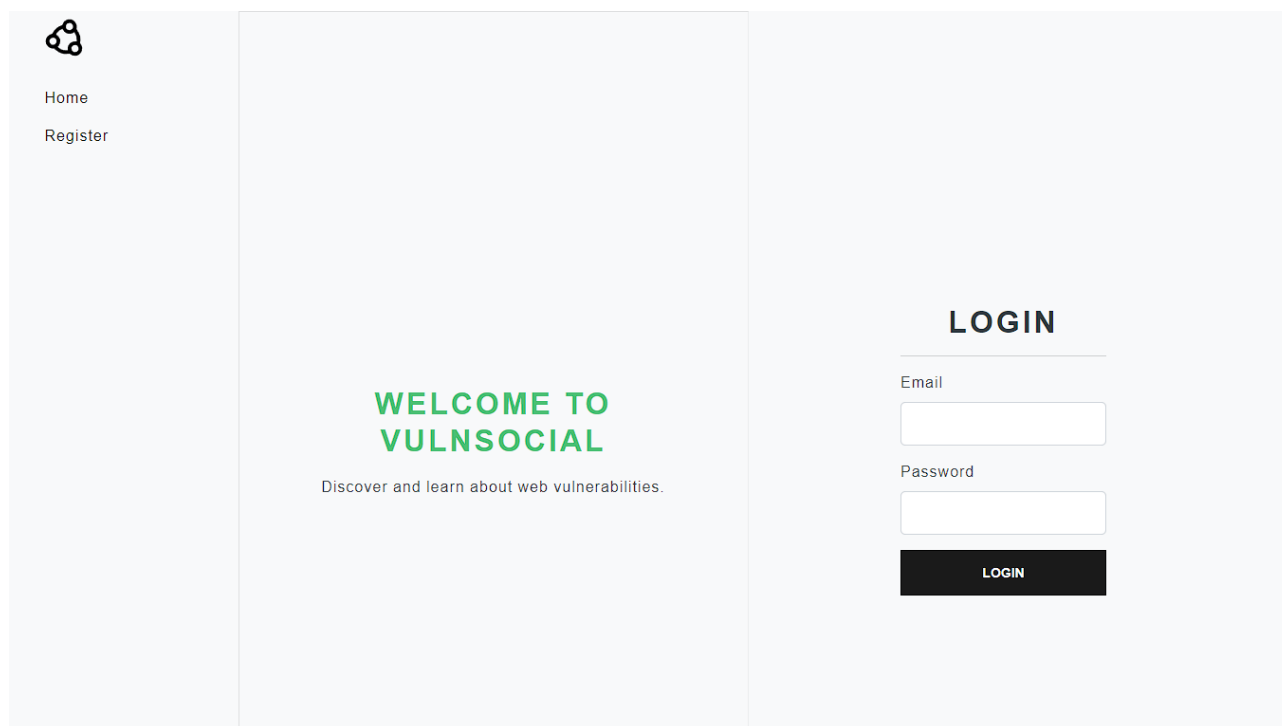
A través de PHPMyAdmin, se puede ejecutar consultas SQL directamente, gestionar la estructura de las tablas y realizar tareas de mantenimiento. Además, esta herramienta ofrece la posibilidad de crear usuarios con permisos específicos, lo que permite restringir o permitir el acceso a los datos de la aplicación según las necesidades de seguridad.

La elección de MySQL y PHPMyAdmin para VulnSocial asegura que la aplicación cuente con una base de datos robusta y accesible, facilitando tanto el almacenamiento de datos como la administración de la información de manera efectiva.

## 7.2.- Descripción de las principales funcionalidades implementadas ilustrándolo con fragmentos de código relevantes.

Como VulnSocial trata de una red social con diferentes vulnerabilidades se van a mostrar las vulnerabilidades en el código y como tratar de mitigarla con un código seguro.

En primer lugar se va a mostrar la vulnerabilidad **SQL Injection** que tenemos en el login. Aquí tenemos el login que se compone de un formulario donde se tiene que introducir el email y password. En este caso para poder realizar la inyección sql, hay que ver si el desarrollador de la página la consulta PHP no la ha realizado como sentencia preparada. En este caso, lo que significa es que los parámetros van en “claro” en la sentencia sql, lo que permitiría una inyección.



The screenshot displays the VulnSocial application interface. On the left, a sidebar contains a logo and navigation links for 'Home' and 'Register'. The main content area is split into two sections. The left section features a green 'WELCOME TO VULNSOCIAL' heading and the subtitle 'Discover and learn about web vulnerabilities.' The right section is titled 'LOGIN' and contains a form with 'Email' and 'Password' input fields, followed by a black 'LOGIN' button.

En este caso no deberían de realizarse así las consultas sql porque son código inseguro. En este caso voy a enseñar de qué manera estaría bien realizado para prevenir ataques de este estilo.

```
<?php
session_start();
include_once "../config/config.php";

if ($_SERVER['REQUEST_METHOD'] === 'POST') {
    $email = $_POST['email'];
    $password = $_POST['password'];

    // Consulta vulnerable a SQL Injection
    $sql = "SELECT * FROM users WHERE email = '$email' AND password = '$password'";
    $stmt = $pdo->query($sql);
    $user = $stmt->fetch(PDO::FETCH_ASSOC);

    if ($user) {
        // Si el usuario existe, crear la sesión
        $_SESSION['username'] = $user['username'];
        $_SESSION['email'] = $user['email'];
        $_SESSION['password'] = $user['password'];
        $_SESSION['id'] = $user['id'];
        $_SESSION['avatar'] = $user['avatar'];
        $_SESSION['admin'] = $user['admin'];

        $status = "Activo ahora";

        // Actualizar el estado del usuario en la base de datos
        $updateSql = "UPDATE users SET status = :status WHERE id = :id";
        $updateStmt = $pdo->prepare($updateSql);
        $updateStmt->execute([
            ':status' => $status,
            ':id' => $user['id']
        ]);

        $_SESSION['status'] = $status;

        echo json_encode([
            "success" => true,
            "idUserario" => $user['id'],
            "isAdmin" => $user['admin'],
            "email" => $user['email'],
            "avatar" => $user['avatar'],
            "status" => $status
        ]);
    } else {
        // Si falla, devolver un error
        echo json_encode(["success" => false, "message" => "Invalid email or password"]);
    }
}
?>
```

Como podemos ver en este código otros detalles que en la vista previa del código inseguro son:

- Falta de sanitización de datos, en este caso que lo que se introduzca sea un email válido.
- Verificar que la password sea válida.

Y lo más importante se puede ver cómo se realizaría una “prepared statement” para poder evitar el ataque SQL Injection.

```
$email = filter_input(INPUT_POST, 'email', FILTER_VALIDATE_EMAIL);
$password = $_POST['password'];

if (!$email || empty($password)) {
    echo json_encode(["success" => false, "message" => "Invalid input"]);
    exit;
}

try {
    // Consulta segura usando sentencias preparadas
    $sql = "SELECT * FROM users WHERE email = :email";
    $stmt = $pdo->prepare($sql);
    $stmt->execute([':email' => $email]);
    $user = $stmt->fetch(PDO::FETCH_ASSOC);

    // Verificar si el usuario existe y la contraseña es válida
    if ($user && password_verify($password, $user['password'])) {
        // Crear sesión segura
        $_SESSION['username'] = htmlspecialchars($user['username']);
        $_SESSION['email'] = $user['email'];
        $_SESSION['id'] = $user['id'];
        $_SESSION['avatar'] = $user['avatar'];
        $_SESSION['admin'] = $user['admin'];
    }
}
```

En segundo lugar, vamos a ver otra vulnerabilidad la cuál es el **XSS (Cross-Site Scripting)**. En este caso esta vulnerabilidad se aprovecha del saneamiento de los datos y la forma en la que se validan los datos tanto en la parte del servidor como en la del cliente. Un consejo es que siempre se validen en ambas partes y no solo en una porque ahí es donde los ciberdelincuentes pueden aprovechar para poder atacar esos fallos de seguridad.

Como se puede ver en el siguiente fragmento de código, se debería de sanitizar la entrada de la variable del contenido del texto del post porque es vulnerable a xss. También en la subida de imágenes no se valida lo que se sube, entonces también se puede subir algo que no es una imagen. En este caso, vamos a subir un fichero que sea php donde se pueda ejecutar contenido y acceder al servidor en este caso si estuviera alojado la web podríamos acceder al contenido del servidor con una reverse Shell que hubiéramos creado previamente. Para la demostración se ha subido un archivo php propio para poder realizar un open redirect (redireccionamiento a una web fraudulenta que podamos haber creado nosotros mismos).

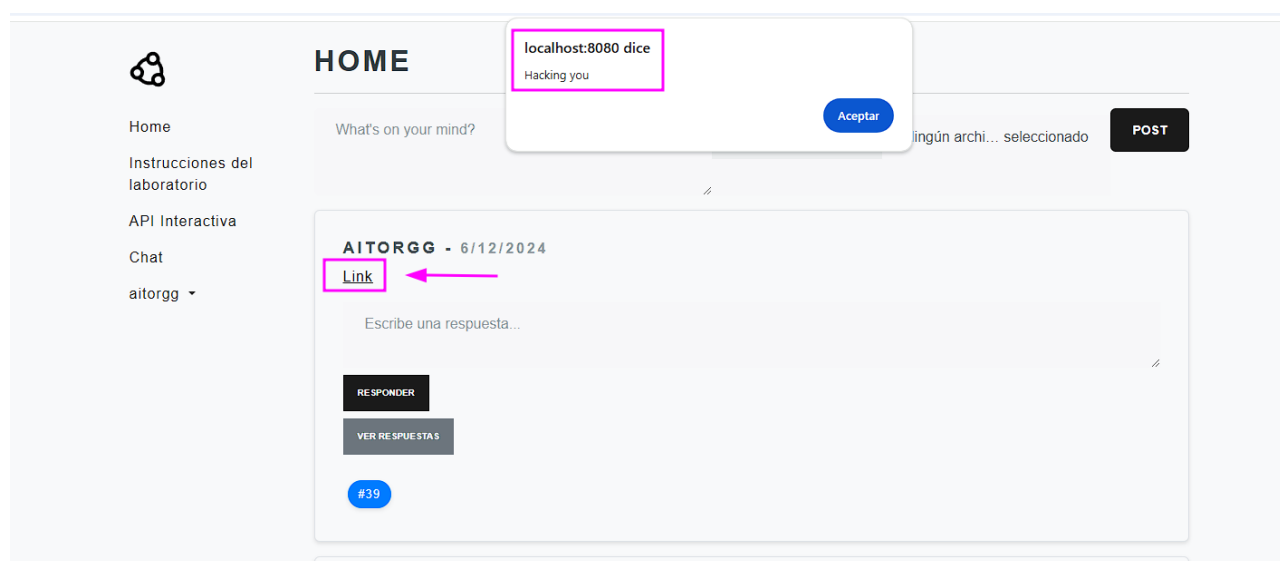
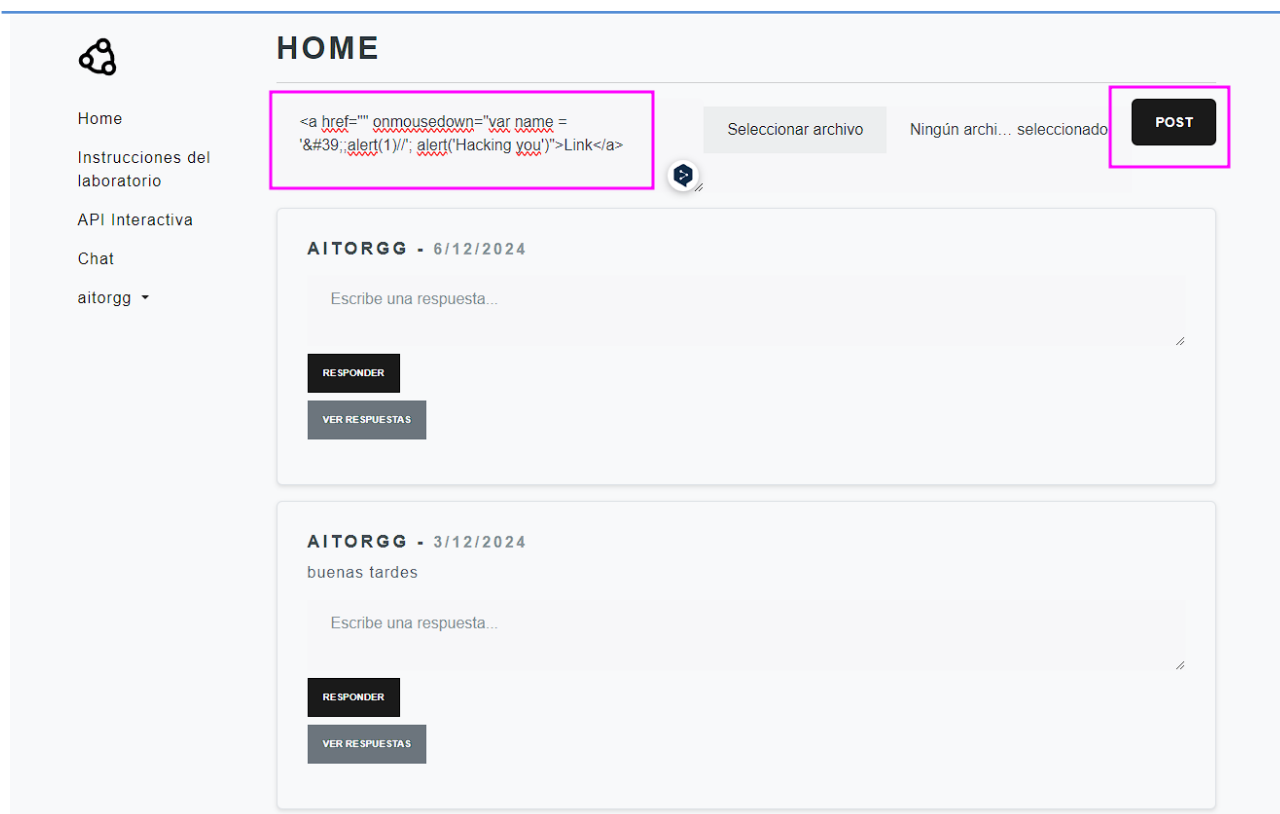
```
1 <?php
2 session_start();
3 include_once '../config/config.php';
4
5 if ($_SERVER['REQUEST_METHOD'] === 'POST') {
6     if (isset($_SESSION['id'])) {
7         $userId = $_SESSION['id'];
8         $postContent = !empty($_POST['post']) ? $_POST['post'] : null;
9
10        // Extraer hashtags del contenido del post
11        preg_match_all('/#(\w+)/', $postContent, $matches);
12        $hashtags = array_unique($matches[0]); // Extrae hashtags y elimina duplicados
13
14        // Eliminar los hashtags del contenido para que no se muestren como texto
15        $postContent = preg_replace('/#(\w+)/', '', $postContent);
16
17        // Manejo de archivos multimedia (imágenes, videos, otros)
18        $mediaPath = null;
19        if (isset($_FILES['media']) && $_FILES['media']['error'] === 0) {
20            $extension = pathinfo($_FILES['media']['name'], PATHINFO_EXTENSION);
21            $mediaName = uniqid("uploaded_") . "." . $extension;
22            $targetPath = "../uploads/" . $mediaName;
23
24            // Mover el archivo al directorio de destino
25            if (move_uploaded_file($_FILES['media']['tmp_name'], $targetPath)) {
26                $mediaPath = $mediaName;
27            } else {
28                echo json_encode(["success" => false, "message" => "Error al subir el archivo"]);
29                exit;
30            }
31        }
32
33        if ($postContent || $mediaPath) {
34            try {
35                $hashtagsJson = json_encode($hashtags);
36
37                // Inserción en la base de datos
38                $stmt = $pdo->prepare("INSERT INTO posts (user_id, content, media_path, hashtags, fecha) VALUES (:user_id, :content, :media_path, :hashtags, NOW())");
39                $stmt->execute([
40                    'user_id' => $userId,
41                    'content' => $postContent,
42                    'media_path' => $mediaPath,
43                    'hashtags' => $hashtagsJson
44                ]);
45
46                echo json_encode(["success" => true, "message" => "Post creado con éxito"]);
47            } catch (PDOException $e) {
48                echo json_encode(["success" => false, "message" => "Error de base de datos: " . $e->getMessage()]);
49            }
50        } else {
51            echo json_encode(["success" => false, "message" => "Error: se requiere texto o archivo multimedia para el post"]);
52        }
53    } else {
54        echo json_encode(["success" => false, "message" => "Error: usuario no logueado"]);
55    }
56 } else {
57     echo json_encode(["success" => false, "message" => "Método no permitido"]);
58 }
59 ?>
```

The screenshot shows the AITORGG application interface. On the left is a sidebar with navigation links: Home, Instrucciones del laboratorio, API Interactiva, Chat, and a dropdown for aitorgg. The main header area includes a 'HOME' title, a search bar containing the payload `<script>alert('1')</script>`, a 'Seleccionar archivo' button, a status indicator 'Ningún archi... seleccionado', and a 'POST' button. Below the header, two chat messages from 'AITORGG - 3/12/2024' are visible. The first message says 'buenas tardes' and the second says 'vamos a probar a hacer otra vulnerabilidad en el login'. Each message has a text input field with the placeholder 'Escribe una respuesta...', a 'RESPONDER' button, and a 'VER RESPUESTAS' button. In the second chat message, a blue button labeled '#sqlinjection' is also present.

Como podemos ver se procesa el post correctamente con las etiquetas javascript, lo cuál es un indicio de que es vulnerable a xss.

This screenshot shows the same AITORGG application interface. The search bar now contains the text 'What's on your mind?'. The 'POST' button is still visible. The chat messages from 'AITORGG - 6/12/2024' are now empty, showing only the 'Escribe una respuesta...' placeholder in the input field. The 'RESPONDER' and 'VER RESPUESTAS' buttons remain below the input field.

Entonces para poder procesar un xss, se va a realizar un payload para poder ejecutar un alert en la propia aplicación, de esta siguiente manera.



Para evitar estas vulnerabilidades mencionadas anteriormente vamos a mejorar este código:

### 1. Sanitización del contenido del post:

- Se utiliza **htmlspecialchars()** para codificar caracteres especiales en el texto del post, evitando que se ejecute código HTML o JavaScript malicioso.

### 2. Validación del archivo multimedia:

- Solo se permiten extensiones específicas definidas en **\$allowedExtensions**. Esto evita que se suban archivos peligrosos como scripts maliciosos.

### 3. Salidas seguras en la interfaz:

- Asegurarse de usar **htmlspecialchars()** o un equivalente cuando se muestra el contenido del post en la interfaz.

```
if (isset($_SESSION['id'])) {
    $userId = $_SESSION['id'];

    // Validar y sanitizar la entrada del contenido del post
    $postContent = !empty($_POST['post']) ? htmlspecialchars($_POST['post'], ENT_QUOTES, 'UTF-8') : null;

    // Extraer hashtags del contenido del post de manera segura
    preg_match_all('/#(\w+)/', $postContent, $matches);
    $hashtags = array_unique($matches[0]);

    $postContent = preg_replace('/#(\w+)/', '', $postContent);

    // Manejo seguro de archivos multimedia (imágenes, videos, otros)
    $mediaPath = null;
    if (isset($_FILES['media']) && $_FILES['media']['error'] === 0) {
        $allowedExtensions = ['jpg', 'jpeg', 'png', 'gif', 'mp4']; // Extensiones permitidas
        $extension = strtolower(pathinfo($_FILES['media']['name'], PATHINFO_EXTENSION));

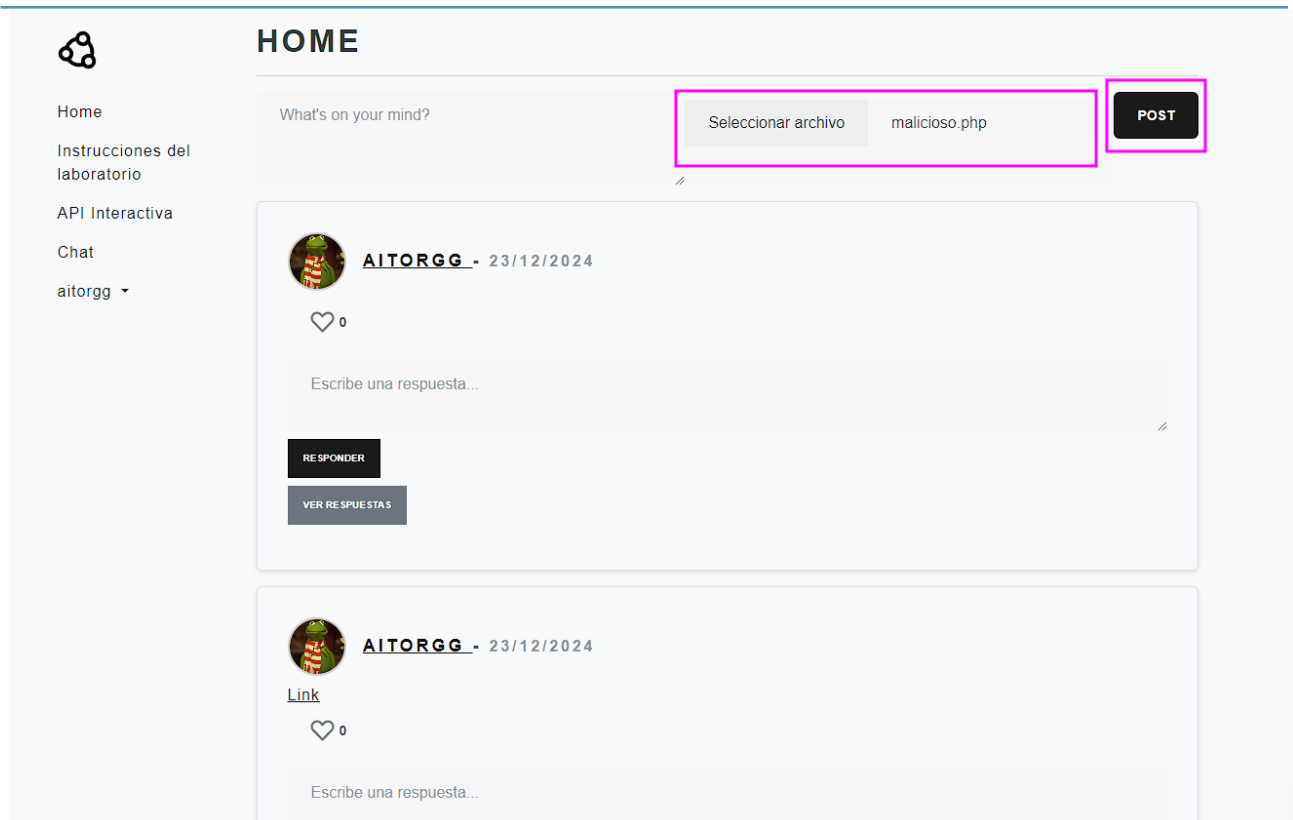
        if (!in_array($extension, $allowedExtensions)) {
            echo json_encode(['success' => false, 'message' => "Tipo de archivo no permitido"]);
            exit;
        }

        $mediaName = uniqid("uploaded_") . "." . $extension;
        $targetPath = "../../uploads/" . $mediaName;

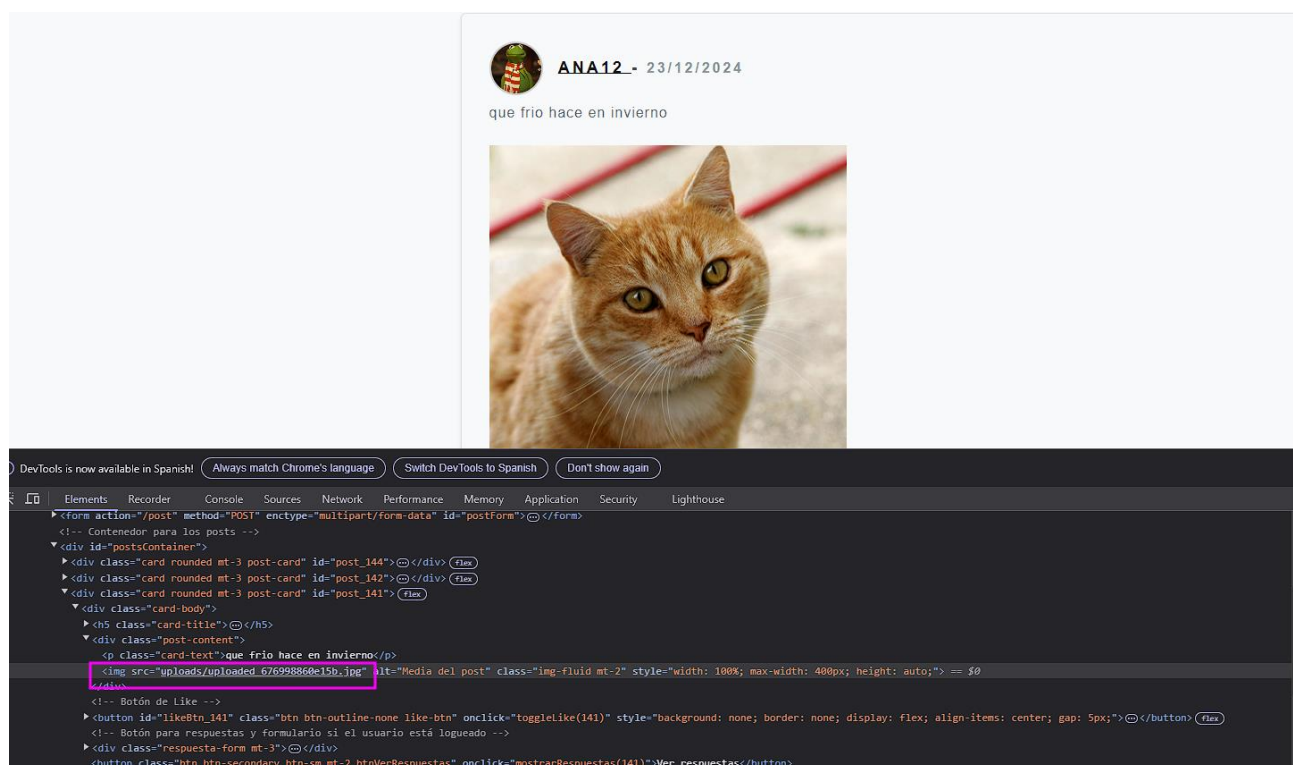
        // Mover el archivo al directorio de destino de manera segura
        if (move_uploaded_file($_FILES['media']['tmp_name'], $targetPath)) {
            $mediaPath = $mediaName;
        } else {
            echo json_encode(['success' => false, 'message' => "Error al subir el archivo"]);
            exit;
        }
    }
}
```

Por último lugar vamos a realizar la vulnerabilidad Local File Inclusion, en lo que consiste es en poder subir un archivo que no sea el que debería poderse subir como es una imagen o un vídeo. En este caso al no validar los datos de la subida de ficheros, se va a subir un fichero .php donde se pueda ejecutar código y a su vez escalar esa vulnerabilidad a un Open Redirect. Esta vulnerabilidad consiste en poder redirigir a un usuario de una web, a una web fraudulenta que esté fuera de la red de la propia página como puede ser una web de phishing.

Para ver si podemos escalar a un Open Redirect, vamos a intentar subir un fichero que no sea en formato foto, se ha creado un script donde se redirigirá al usuario una vez se pueda ejecutar el script. Para ello, se subirá la imagen en el servidor, donde se tendrá que acceder a la ruta para acceder.

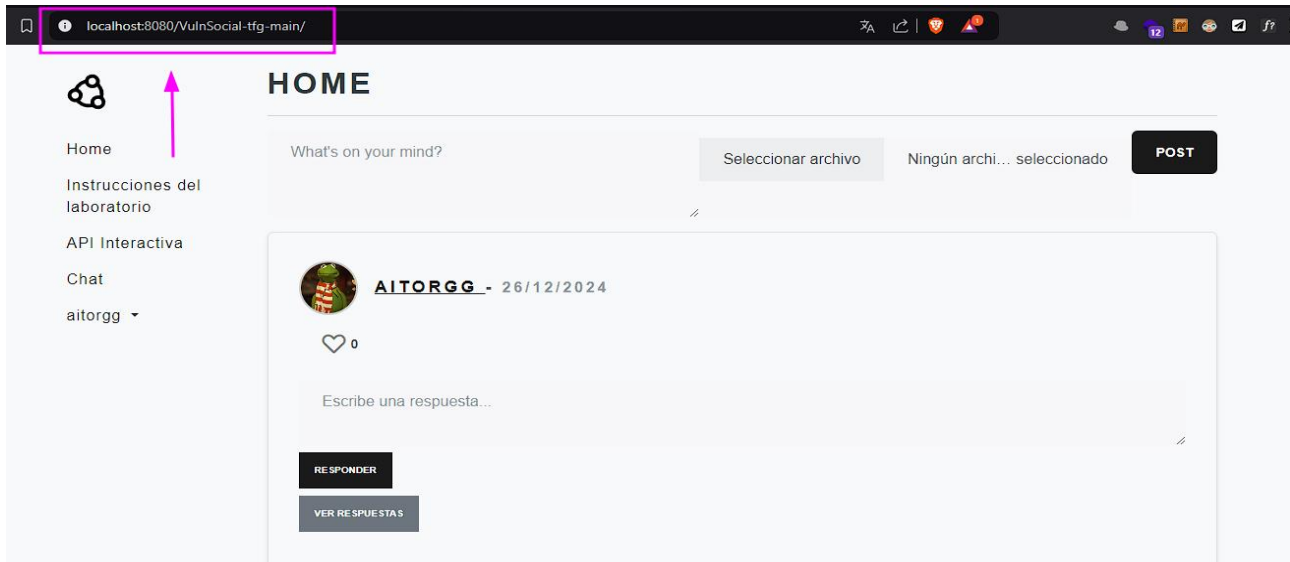


Vamos a un post donde se haya subido una imagen para poder visualizar la ruta donde se alojan las imágenes. En este caso se va a intentar acceder a esa ruta, cambiando los rutas de la url, de la página.



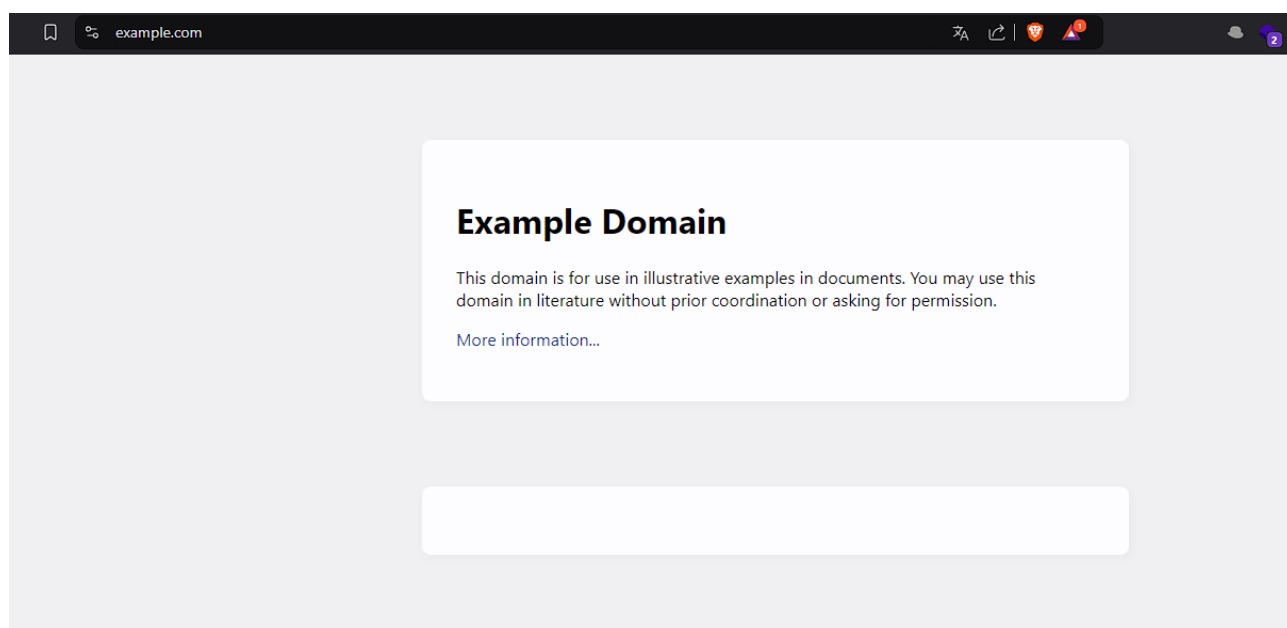
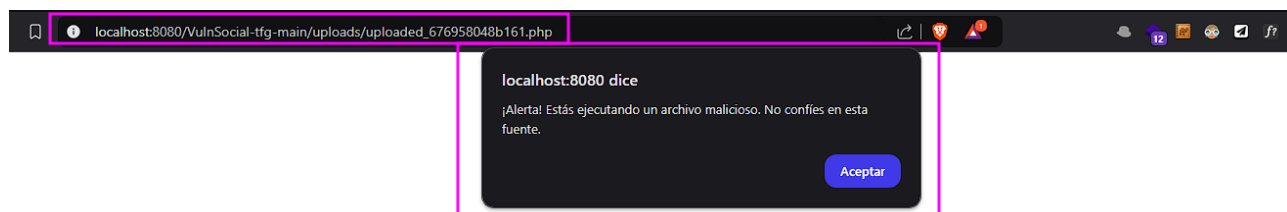


En este caso, tenemos la ruta del inicio, donde se muestran las posts principalmente, como hemos visto anteriormente la ruta donde se alojan las imágenes está en la ruta **uploads/**.



Como se puede ver, se está en un entorno local. Si fuera en un entorno de producción. Se tendría que restringir para los usuarios esta misma ruta para que no se pueda acceder a datos sensibles de otros usuarios, o a recursos internos de la propia aplicación. Se va a ejecutar el archivo malicioso que se ha subido en PHP y ver qué sucede. Se ejecuta un alert donde nos está avisando que es un archivo malicioso, tras aceptar el alert, nos redirecciona a la página que hemos creado y tras esperar unos segundos nos redirecciona a una página que se ha puesto de ejemplo en este caso, pero que podría haber sido una página de phishing.





Para poder ver mejor cómo funciona este script, se va a adjuntar el código que se ha planteado:

```
uploaded_676d38214e9bb.php X
VulnSocial-tfg-main > uploads > uploaded_676d38214e9bb.php
1  <?php
2  echo "<!DOCTYPE html>"
3  <html lang='es'>
4  <head>
5      <meta charset='UTF-8'>
6      <meta name='viewport' content='width=device-width, initial-scale=1.0'>
7      <title>Fichero Malicioso</title>
8      <style>
9          body {
10              background-color: #333;
11              color: white;
12              font-family: Arial, sans-serif;
13              text-align: center;
14              padding-top: 50px;
15          }
16          .warning {
17              font-size: 50px;
18              font-weight: bold;
19              color: red;
20          }
21          .message {
22              font-size: 30px;
23              margin-top: 20px;
24          }
25      </style>
26 </head>
27 <body>
28     <div class='warning'>¡CUIDADO! Este archivo podría ser peligroso</div>
29     <div class='message'>
30         Esto es un fichero malicioso. <br>El contenido ha sido ejecutado en tu navegador.
31     </div>
32     <script>
33         // Mostrar un mensaje emergente en el navegador
34         alert('¡Alerta! Estás ejecutando un archivo malicioso. No confíes en esta fuente.');
```

Para evitar estas vulnerabilidades mencionadas anteriormente vamos a mejorar este código:

1. Validación y sanitización de las rutas en Local File Inclusion (LFI):
  - Se usa **realpath()** para obtener la ruta absoluta del archivo y se verifica que esté dentro del directorio permitido. Esto previene que un atacante manipule la ruta para incluir archivos fuera de los límites permitidos.
  - Solo se permiten nombres de archivos predefinidos que cumplen con un patrón específico.
2. Validación de URLs para evitar Open Redirect:
  - Se valida que las rutas de redirección sean internas o correspondan a un conjunto de URLs predefinidas.
  - En lugar aceptar cualquier entrada del usuario, se utiliza un sistema de mapeo seguro para las rutas válidas.

## 8.- Planificación del proyecto

---

### 8.1.- Acciones

A continuación, se detallan los bloques de acciones principales y las tareas específicas dentro de cada uno de ellos:

#### 8.1.1 Toma de requisitos

La primera etapa del desarrollo de VulnSocial consistió en identificar las necesidades del sistema y definir sus funcionalidades principales.

- **Reunión inicial:** Se realizó una sesión de brainstorming para identificar las características clave, como el registro de usuarios, la creación y gestión de posts, los likes, el chat en tiempo real y la administración de estadísticas en dashboards.
- **Investigación de proyectos similares:** Se revisaron herramientas como OWASP Juice Shop y PortSwigger Web Security Academy para identificar áreas de mejora y funcionalidad educativa en ciberseguridad.
- **Elaboración del documento de requisitos:** Se clasificaron los requisitos en funcionales y no funcionales. Se definió que el sistema debía ser educativo y ofrecer vulnerabilidades simuladas, como inyección SQL, XSS y Open Redirect.

#### 8.1.2 Análisis de los requisitos

En esta fase, se representaron los requisitos de forma estructurada para asegurar que todas las funcionalidades estuvieran correctamente definidas.

- **Diagrama de casos de uso:** Se diseñó un diagrama UML para identificar las interacciones entre los roles de la aplicación (usuario, administrador y manager) y sus respectivas funcionalidades.
- **Especificaciones funcionales:** Se detallaron flujos como el registro y login de usuarios, la publicación de posts, la interacción mediante likes, y la administración de estadísticas.
- **Especificaciones no funcionales:** Se definieron métricas de rendimiento, como tiempos de respuesta inferiores a un segundo, y medidas de seguridad, como la autenticación de usuarios y la validación de entradas en el servidor.

#### 8.1.3 Diseño

La etapa de diseño fue clave para convertir los requisitos en un modelo funcional.

- **Diseño de la base de datos:** Se crearon tablas para manejar usuarios, posts, likes, mensajes del chat y respuestas, aplicando principios de normalización para optimizar consultas y evitar redundancias.
- **Diseño de interfaces:** Se desarrollaron prototipos para las páginas principales, como la vista de perfil, la sección de posts, los dashboards de administración y el chat en tiempo real. Las interfaces fueron diseñadas para ser intuitivas y responsivas.
- **Flujos de procesos:** Se documentaron procesos clave, como el flujo para dar y quitar likes y el envío de mensajes en el chat.
- **Arquitectura del sistema:** Se definió una arquitectura cliente-servidor utilizando PHP en el backend y JavaScript para el frontend.

### 8.1.4 Codificación (Implementación)

La implementación de VulnSocial fue una etapa intensiva que transformó el diseño en un sistema funcional.

- **Backend:** Se desarrollaron los endpoints en PHP para manejar operaciones como autenticación, creación de posts, gestión de likes y envío de mensajes. Además, se integraron las funcionalidades de dashboards con consultas SQL optimizadas.
- **Frontend:** Se implementaron interfaces dinámicas utilizando JavaScript y tecnologías como Fetch API. La interacción en tiempo real, como el chat y la actualización de likes, se integró con WebSockets.
- **Dashboards:** Se desarrollaron estadísticas en tiempo real para el administrador y el manager, utilizando Chart.js para graficar datos como likes por usuario, mensajes enviados y recibidos, y posts recientes.

### 8.1.5 Pruebas

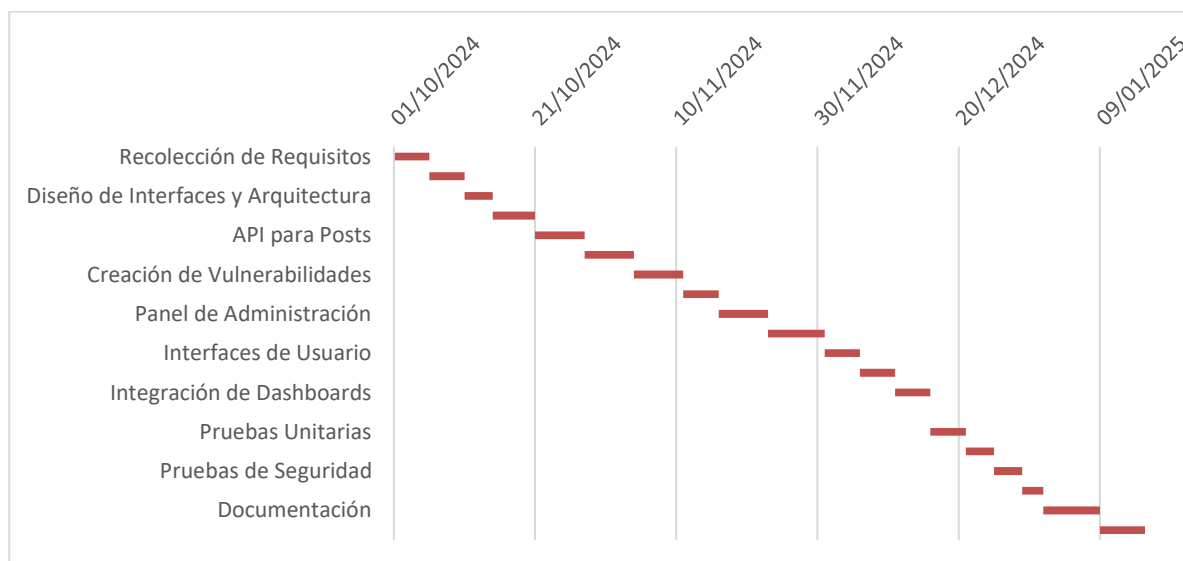
Las pruebas fueron esenciales para garantizar el correcto funcionamiento del sistema y la seguridad del mismo.

- **Pruebas unitarias:** Se validaron módulos individuales, como la funcionalidad de likes, el registro de usuarios y el envío de mensajes. Estas pruebas se realizaron utilizando PHPUnit para PHP.
- **Pruebas de integración:** Se verificó la interacción entre el cliente y el servidor, comprobando que las solicitudes AJAX y las respuestas del servidor fueran correctas.
- **Pruebas de seguridad:** Se simularon ataques como inyecciones SQL y XSS en un entorno controlado para garantizar que el sistema fuera resistente.
- **Pruebas de rendimiento:** Se evaluó el tiempo de respuesta del sistema con herramientas como Apache JMeter, asegurando que la aplicación pudiera manejar múltiples usuarios concurrentes.
- **Pruebas funcionales:** Se probaron flujos completos, como el registro, creación de posts y la interacción mediante likes, para garantizar una experiencia de usuario fluida.

### 8.1.6 Documentación

La última fase consistió en documentar la manera de como realizar las vulnerabilidades en la aplicación y cómo solucionarlas para que los desarrolladores realicen un código seguro.

## 8.2.- Temporalización y secuenciación



## 9.- Pruebas y validación

Para garantizar que **VulnSocial** cumple con los requisitos y objetivos definidos, se llevaron a cabo diferentes tipos de pruebas que permitieron verificar tanto el funcionamiento correcto de la aplicación como la efectividad de las vulnerabilidades implementadas. A continuación, se describen las pruebas realizadas, las herramientas utilizadas y el plan de pruebas implementado:

### Tipos de Pruebas Realizadas:

1. **Pruebas Funcionales:** Estas pruebas validaron que todas las funcionalidades de la aplicación (inicio de sesión, publicación de posts, gestión de usuarios, etc.) funcionaran correctamente según los requisitos definidos. Se probaron escenarios como el registro de usuarios, el inicio de sesión con SQL Injection y la respuesta a posts.
2. **Pruebas de Sistemas:** Se verificó que la integración de todos los módulos y la base de datos funcionaran de manera correcta. Se evaluó el comportamiento de la aplicación en diferentes entornos (local y servidor).
3. **Pruebas Unitarias:** Se realizaron pruebas unitarias para comprobar que las funciones individuales del back-end y front-end, como la validación de formularios y las consultas a la base de datos, se ejecutaran correctamente.
4. **Pruebas de Integración:** Se probó la interacción entre el front-end y el back-end, verificando que las solicitudes AJAX de publicación de posts y la visualización de datos desde la base de datos se realizaran de manera eficiente y sin errores.

### Herramientas Utilizadas:

- **PHP Unit:** Para las pruebas unitarias del código PHP, asegurando que las funciones de autenticación, validación y CRUD de usuarios y posts funcionaran correctamente.
- **Selenium:** Para realizar pruebas de sistemas y usabilidad, automatizando interacciones en la interfaz de usuario como el inicio de sesión y la creación de publicaciones.
- **JMeter:** Para realizar pruebas de rendimiento, simulando múltiples usuarios interactuando con la aplicación al mismo tiempo, y evaluando la carga en el servidor y las consultas a la base de datos.
- **Postman:** Para verificar la correcta implementación de las API y asegurar que las respuestas de los endpoints fueran las esperadas.

### Plan de Pruebas:

1. **Objetivo:** Verificar que la aplicación cumple con los requisitos funcionales y de seguridad establecidos, y garantizar su rendimiento bajo condiciones de carga.
2. **Alcance:** Se probaron los siguientes módulos de la aplicación: autenticación, gestión de posts, respuestas a posts, administración de usuarios, panel de manager, y manejo de vulnerabilidades.
3. **Metodología:**
  - Se establecieron escenarios de prueba para cada funcionalidad.
  - Se realizaron pruebas manuales y automatizadas para verificar que el sistema respondiera como se esperaba bajo condiciones normales y extremas (carga alta).
  - Se documentaron los errores encontrados y se corrigieron.

### Pruebas Concretas:

1. **Prueba de SQL Injection:** Se intentó inyectar código malicioso en los campos de inicio de sesión y registro, para asegurar que las vulnerabilidades fueran correctas y que no se produjeran errores de seguridad (intencionales para fines educativos).
2. **Prueba de XSS:** Se introdujeron scripts maliciosos en los campos de comentarios y publicaciones para comprobar que la vulnerabilidad fuera efectiva.

3. **Prueba de Rendimiento:** Usando **JMeter**, se simuló una carga de 50 usuarios simultáneos realizando diversas acciones (inicio de sesión, publicación de posts) para evaluar la estabilidad y el rendimiento del sistema.
4. **Prueba de Usabilidad:** Mediante **Selenium**, se automatizaron flujos de usuario como el registro, inicio de sesión, creación y edición de posts para verificar que no hubiera fallos en la interfaz y la experiencia fuera la esperada.

Este proceso de pruebas garantizó que **VulnSocial** cumpliera con los requisitos funcionales, tuviera una experiencia de usuario óptima, y fuera robusta frente a las vulnerabilidades implementadas.

## 10.- Relación del proyecto con los módulos del ciclo

Este proyecto tiene una relación directa y significativa con los módulos impartidos en el ciclo formativo de **Desarrollo de Aplicaciones Web (DAW)**. Cada uno de los aspectos abordados en el proyecto se conecta con los conocimientos y habilidades adquiridos a lo largo de las diferentes asignaturas del ciclo. A continuación, se detalla la relación del proyecto con cada módulo:

### 10.1 Sistemas Informáticos

- Este módulo sentó las bases para la comprensión y configuración del entorno de desarrollo del proyecto, incluyendo servidores web locales, configuraciones de bases de datos y gestión de redes.
- La implementación de entornos controlados para la simulación de vulnerabilidades aprovecha los conceptos aprendidos en este módulo, como la configuración de servidores y seguridad básica del sistema operativo.

### 10.2 Bases de Datos

- La funcionalidad de la aplicación depende en gran medida del diseño y la gestión de la base de datos.
- La creación de tablas para usuarios, posts, likes y respuestas, así como el diseño de consultas SQL avanzadas para la funcionalidad de likes, filtros y reportes, están directamente relacionados con este módulo.
- Además, se aplicaron principios de normalización y optimización de consultas aprendidos durante el curso.

### 10.3 Programación

- La lógica del proyecto, incluyendo la interacción entre el frontend y el backend, se desarrolló gracias a las habilidades de programación adquiridas en este módulo.
- La implementación de funciones como autenticación de usuarios, manejo de sesiones, lógica de likes y eliminación de posts reflejan la aplicación de estructuras de control y algoritmos básicos.
- También se trabajó con programación orientada a objetos (POO) para organizar el código de forma eficiente.

### 10.4 Lenguajes de Marcas

- La estructura HTML y CSS del proyecto, junto con la integración de JavaScript en el cliente, se fundamentaron en los conocimientos adquiridos en este módulo.
- También se aplicaron técnicas de validación y manipulación de datos en el lado cliente para garantizar una experiencia de usuario coherente.

## 10.5. Entornos de Desarrollo

- El uso de herramientas como control de versiones con Git y la integración de editores de código (Visual Studio Code) fue clave en la organización y gestión del desarrollo del proyecto.
- Además, se emplearon técnicas de depuración para resolver problemas y optimizar el rendimiento del sistema.

## 10.6 Desarrollo Web en Entorno Cliente

- La interacción dinámica de la aplicación, como la actualización de likes en tiempo real y el cambio de tabs en el perfil del usuario, se implementó con JavaScript y tecnologías como Fetch API.
- También se trabajaron conceptos de manipulación del DOM y eventos, fundamentales para la interactividad del sistema.

## 10.7 Desarrollo Web en Entorno Servidor

- El backend del proyecto se construyó en PHP, integrando conceptos de este módulo, como la creación de APIs, manejo de sesiones y validación en el servidor.
- La comunicación entre el cliente y el servidor, junto con la seguridad en las peticiones (CSRF y validación de datos), reflejan directamente el aprendizaje en este módulo.

## 10.8 Despliegue de Aplicaciones Web

- La preparación del proyecto para ser desplegado, incluyendo la configuración de bases de datos remotas y la gestión de un servidor web, están vinculadas a este módulo.
- Se trabajaron aspectos como la organización de los archivos del proyecto, la seguridad en el despliegue y el uso de entornos de producción y desarrollo.

En este caso esta aplicación no está desplegada pero aquí se detallan los pasos de cómo se realizaría:

### 10.8.1 Configuración del Servidor

#### a. Hosting Compartido

- Contratando un hosting que soporte **PHP** y bases de datos **MySQL** (como Hostinger, SiteGround, o Ionos).
- Se accedería al **Panel de Control** (como cPanel o Plesk) para subir los archivos y configurar la base de datos.

#### b. Servidor en la Nube (Más Personalizable)

- En este caso se usaría un servicio como **AWS**, **DigitalOcean**, **Linode** o **Vercel**.
- Configurando un servidor Linux con Apache o Nginx, PHP y MySQL.

### 10.8.2 Configurar el Proyecto Localmente

1. **Revisa las rutas:** Cambia las rutas absolutas o relativas para que sean compatibles con un servidor en producción.
  - Por ejemplo, usa `$_SERVER['DOCUMENT_ROOT']` para rutas dinámicas en PHP.
2. **Configura las variables de conexión de la base de datos:** Usa un archivo centralizado como `config.php` para que los datos puedan ser cambiados fácilmente



---

### 10.8.3 Subir Archivos al Servidor

#### Método 1: Hosting Compartido

1. Accede al **panel de control** (cPanel, Plesk).
2. Sube tus archivos al directorio `public_html` o la carpeta principal usando el administrador de archivos o un cliente FTP como **FileZilla**.
3. Importa tu base de datos:
  - Usa **phpMyAdmin** para importar el archivo `.sql`.

#### Método 2: Servidor en la Nube

1. Conéctate al servidor usando SSH:

```
ssh usuario@ip_servidor
```

2. Instalando un stack LAMP o LEMP:

```
sudo apt update  
sudo apt install apache2 php mysql-server
```

3. Configura el directorio del sitio:
  - Colocando los archivos en `/var/www/html` y ajustando los permisos:

```
sudo chown -R www-data:www-data /var/www/html  
sudo chmod -R 755 /var/www/html
```

4. Configuración de la base de datos:
  - Con phpMyAdmin para crear/importar la base de datos.

### 10.8.4 Configurar la Base de Datos

1. **Crea la base de datos en el servidor.**
  - Usa phpMyAdmin:

```
CREATE DATABASE vulnerable_app;
```

### 10.8.5 Configurar Seguridad

1. **HTTPS:**
  - Obtener un certificado SSL gratuito con **Let's Encrypt** o usa el que proporcione hostinger.
  - Configura HTTPS para todas las rutas.
2. **Ocultar errores de PHP:**
  - En `php.ini`, desactivar la visualización de errores:

```
display_errors = Off  
log_errors = On  
error_log = /var/log/php_errors.log
```

### 3. Crea un usuario seguro para MySQL:

```
CREATE USER 'usuario'@'localhost' IDENTIFIED BY 'contraseña';  
GRANT ALL PRIVILEGES ON mi_base_de_datos.* TO 'usuario'@'localhost';  
FLUSH PRIVILEGES;
```

### 4. Valida datos de entrada:

- Usa filtros y validaciones para prevenir SQL Injection, XSS y CSRF.

## 10.8.6 Probar la Aplicación en Producción

1. Acceder a la aplicación usando el dominio o IP pública.
2. Realiza pruebas exhaustivas de cada funcionalidad:
  - Login, registro, posteo de posts, subida de imágenes.

## 10.8.7 Configurar un Subdominio para Evaluación

En este caso presentaría el proyecto con un subdominio como vulnSocial.com

## 11.- Conclusiones

La realización de este proyecto ha sido una experiencia enriquecedora y desafiante, que me ha permitido explorar en profundidad el desarrollo de aplicaciones web con un enfoque educativo y de concienciación en ciberseguridad. A lo largo del proceso, he logrado cumplir con la mayoría de los objetivos planteados, aunque también he encontrado áreas de mejora y aspectos que han quedado pendientes debido a limitaciones de tiempo y recursos.

### Estado del proyecto

El proyecto ha alcanzado un estado funcional y cumple con los objetivos principales definidos al inicio. Se han desarrollado las siguientes funcionalidades clave:

- Creación de una red social básica con publicación de contenido, likes y comentarios.
- Simulación de vulnerabilidades web comunes como inyección SQL y XSS, ofreciendo un entorno controlado para aprendizaje.
- Implementación de un sistema de roles (administrador, manager y usuario) con vistas y funcionalidades personalizadas.
- Desarrollo de funcionalidades avanzadas como el sistema de likes dinámico y un panel de estadísticas para el rol de manager.

Sin embargo, algunas características adicionales, como la integración completa de inteligencia artificial para la detección de amenazas o la gamificación avanzada, no pudieron ser implementadas en esta fase del proyecto debido a la complejidad técnica y el tiempo limitado. Estas áreas representan oportunidades para futuras ampliaciones.

### Reflexión sobre el proceso

#### 1. Definición de requisitos

Durante el desarrollo, entendí la importancia de definir de forma clara y detallada los requisitos funcionales antes de comenzar con la implementación. Al principio, algunos aspectos del proyecto no estaban completamente definidos, lo que generó ajustes y correcciones posteriores. Este aprendizaje

me ha demostrado que invertir tiempo en una planificación sólida reduce significativamente los problemas durante el desarrollo.

## 2. Modularidad y escalabilidad:

Tener una visión general del proyecto desde el inicio me permitió estructurarlo de forma modular, dividiendo las funcionalidades en componentes independientes. Esta estrategia no solo facilitó la implementación y pruebas, sino que también asegura que el proyecto sea escalable y pueda ser ampliado en el futuro sin grandes complicaciones.

## 3. Gestión del tiempo

La temporalización del proyecto fue uno de los mayores desafíos. Aunque las tareas principales fueron completadas, algunas funcionalidades quedaron fuera del alcance por falta de tiempo como el aprendizaje del fine tuning para poder implementar un modelado de datos para la implementación de una IA experta en ciberseguridad. Esto me ha enseñado la importancia de priorizar y ajustar las expectativas en función de los recursos disponibles.

## 4. Aprendizaje continuo

El desarrollo de este proyecto no solo me permitió afianzar conocimientos técnicos, sino que también me expuso a nuevas tecnologías y conceptos, como la simulación de vulnerabilidades y el uso de mejores prácticas en ciberseguridad. Además, trabajar con roles de usuario y permisos me hizo comprender mejor la arquitectura y lógica de aplicaciones complejas.

## Resultados y proyección

El proyecto no solo cumple con su propósito educativo y funcional, sino que también sienta las bases para desarrollos futuros. Ha servido como una herramienta para demostrar cómo se pueden integrar principios de ciberseguridad en el desarrollo web, al tiempo que proporciona un entorno interactivo para la concienciación y el aprendizaje.

En retrospectiva, este proyecto ha superado mis expectativas en muchos aspectos, pero también me ha dejado con una motivación renovada para seguir aprendiendo y mejorando. Me siento satisfecho con los resultados alcanzados y seguro de que este trabajo puede evolucionar hacia algo aún más impactante y útil en el futuro.

## 12.- Proyectos futuros

El proyecto desarrollado establece una base sólida para abordar la ciberseguridad desde una perspectiva educativa y práctica. Sin embargo, existen múltiples formas de ampliar, mejorar y diversificar las funcionalidades existentes. A continuación, se presentan propuestas de ampliaciones, mejoras y nuevos proyectos basados en este trabajo:

### 12.1 Ampliaciones y mejoras

1. **Ampliación de vulnerabilidades simuladas:** Incorporar más vulnerabilidades comunes en aplicaciones web, como ataques de deserialización insegura, vulnerabilidades en GraphQL y técnicas de explotación en APIs RESTful. Estas nuevas funcionalidades permitirían a los usuarios profundizar aún más en el aprendizaje de la ciberseguridad.
2. **Mejora de la experiencia del usuario**
  - Optimizar la interfaz de usuario para facilitar la navegación y la interacción.
  - Implementar un diseño adaptable y accesible para usuarios con diferentes capacidades.

### 3. Integración de inteligencia artificial

Utilizar IA para proporcionar funcionalidades avanzadas de detección y aprendizaje en ciberseguridad, como:

- **Sistemas de detección de anomalías:** Implementar modelos de aprendizaje automático que analicen los datos en tiempo real para identificar comportamientos sospechosos en las publicaciones o en el uso de la plataforma.
  - **Asistente virtual:** Incorporar un chatbot con IA que ofrezca explicaciones detalladas sobre cada vulnerabilidad simulada y sugiera soluciones de mitigación.
  - **Generación automática de escenarios:** Utilizar IA para crear automáticamente nuevos retos y vulnerabilidades personalizadas basadas en el historial de aprendizaje del usuario.
4. **Ampliación de la funcionalidad de red social**
- Añadir comentarios con menciones y notificaciones en tiempo real.
  - Implementar un sistema avanzado de recomendaciones basadas en el contenido y las interacciones del usuario.
5. **Gamificación**
- Añadir un sistema de puntuación y logros para motivar a los usuarios a completar retos relacionados con ciberseguridad.
  - Introducir niveles de dificultad adaptables según el progreso del usuario.
6. **Soporte multilingüe**
- Traducir la plataforma a múltiples idiomas para ampliar su alcance a usuarios internacionales.

## 12.2 Nuevos proyectos basados en este trabajo

1. **Plataforma integral de ciberseguridad para empresas**  
Partiendo de este proyecto, se podría desarrollar una plataforma completa que ofrezca simulaciones de ataques personalizados para empleados de empresas, combinando concienciación y formación en ciberseguridad.
2. **Aplicación móvil**  
Desarrollar una versión móvil de la plataforma que permita a los usuarios aprender y practicar sobre ciberseguridad en cualquier lugar y momento.
3. **Integración con plataformas educativas**
  - Convertir el proyecto en un complemento de plataformas de e-learning, como Moodle o Canvas, permitiendo a los profesores integrar simulaciones de ciberseguridad en sus cursos.
  - Implementar una API que facilite la comunicación con sistemas educativos externos.
4. **Análisis en tiempo real con IA avanzada**  
Crear una herramienta que permita a los administradores de sistemas analizar sus propias aplicaciones en tiempo real para identificar vulnerabilidades. Este proyecto podría combinar aprendizaje automático con técnicas de hacking ético para ofrecer recomendaciones personalizadas.
5. **Simulador de ataques de redes sociales**  
Basándose en las funcionalidades actuales, desarrollar un simulador que permita explorar vulnerabilidades específicas de redes sociales, como phishing o ataques de ingeniería social, en un entorno controlado.
6. **Certificación en ciberseguridad**  
Crear un programa de certificación basado en la plataforma, donde los usuarios puedan realizar un curso completo de ciberseguridad y obtener credenciales oficiales al superar todos los módulos y retos.
7. **Monetización del proyecto**
  - Implementar un modelo de suscripción para acceso a contenido avanzado.
  - Introducir publicidad no intrusiva o colaboraciones con empresas de ciberseguridad para financiar el desarrollo y mantenimiento.

En resumen, este proyecto no solo proporciona un marco educativo y funcional en ciberseguridad, sino que también abre un abanico de posibilidades para futuros desarrollos que impacten tanto en el ámbito educativo como profesional. La integración de IA y nuevas tecnologías representa un eje clave para estas expansiones, posicionando el proyecto como una herramienta innovadora y relevante en el panorama actual de la ciberseguridad.

## 13.- Bibliografía/Webgrafía

---

### Bibliografía

*Diagrams Net.* (s.f.). Obtenido de <https://app.diagrams.net/>: <https://app.diagrams.net/>

*FreePublicApis* <https://www.freepublicapis.com/>. (s.f.). Obtenido de <https://www.freepublicapis.com/>

*GitHub.* (s.f.). Obtenido de <https://github.com/>: <https://github.com/>

<https://app.diagrams.net/>. (s.f.). Obtenido de <https://app.diagrams.net/>

*JavaScript* . (s.f.). Obtenido de <https://developer.mozilla.org/es/docs/Web/JavaScript>

*Jquery.* (s.f.). Obtenido de <https://jquery.com/>: <https://jquery.com/>

*Materializecss.* (2014). Obtenido de <https://materializecss.com/>.

*NPM* . (s.f.). Obtenido de <https://www.npmjs.com/>: <https://www.npmjs.com/>

*PHP.* (s.f.). Obtenido de <https://www.php.net/>: <https://www.php.net/>

*Resend.* (s.f.). Obtenido de <https://resend.com/emails>: <https://resend.com/emails>

*VirusTotal API.* (s.f.). Obtenido de <https://docs.virustotal.com/reference/overview>

*Visual Paradigm.* (s.f.). Obtenido de <https://online.visual-paradigm.com/>: <https://online.visual-paradigm.com/>

*Xampp.* (s.f.). Obtenido de <https://www.apachefriends.org/>: <https://www.apachefriends.org/>

## 14.- Anexos

---

### 14.1 Vídeo demo

Se adjuntará el vídeo demo en la propia entrega del TFG junto al correspondiente proyecto y esta memoria.

<https://youtu.be/TDSKBX7RFUg>