5 de noviembre, 2025

CASOS DE USO

Caso de Uso 1: Ver e interactuar con un mensaje

Actor: Usuario sin registrar, Usuario registrado

Descripción: Permite al usuario ver un mensaje aleatorio e interactuar con él

Flujo Principal:

1. El usuario accede a la aplicación.
2. Hace clic en cualquier parte de la ventana de la aplicación.
3. Se le muestra un mensaje aleatorio de la base de datos.
4. El usuario puede responder al mensaje con una lista de emoticonos y reportar el mensaje como inapropiado.
5. Al cerrar el mensaje, el usuario no podrá volver atrás para volver a verlo.

Caso de Uso 2: Registrarse o Iniciar Sesión

Actor: Usuario sin registrar

Descripción: Permite a los usuarios registrarse en la aplicación o iniciar sesión, realizando al finalizar una doble autentificación.

Flujo Principal:

1. El usuario elige entre registrarse o iniciar sesión.
2. Si elige registrarse, ingresa los datos requeridos y el sistema crea una cuenta.
3. Si elige iniciar sesión con su cuenta de la aplicación, ingresa sus credenciales.
4. Se le pide al usuario que ingrese el código para la doble autentificación.
5. Una vez autenticado, el usuario regresa a la aplicación con acceso a su cuenta.

Caso de Uso 3: Un usuario gestiona sus propios mensajes

Actor: Usuario registrado

Descripción: Permite a un usuario ver sus mensajes, las reacciones de otros usuarios a ellos, y poder eliminarlos o editarlos.

Flujo Principal:

1. El usuario accede al panel de control de su perfil.

2. Se le muestra una lista con todos sus mensajes, las reacciones de los usuarios y botones para proceder a eliminar o editar.

3. En caso de editar, podrá editar todo el mensaje desde esa propia página, confirmar los cambios o cancelarlos.

4. En caso de eliminar, se le mostrará una notificación para que confirme que quiere eliminarlo.

Caso de Uso 4: El administrador gestiona los reportes

Actor: Usuario administrador

Descripción: El administrador gestiona manualmente si un mensaje reportado debe ser eliminado o se puede mantener.

Flujo Principal:

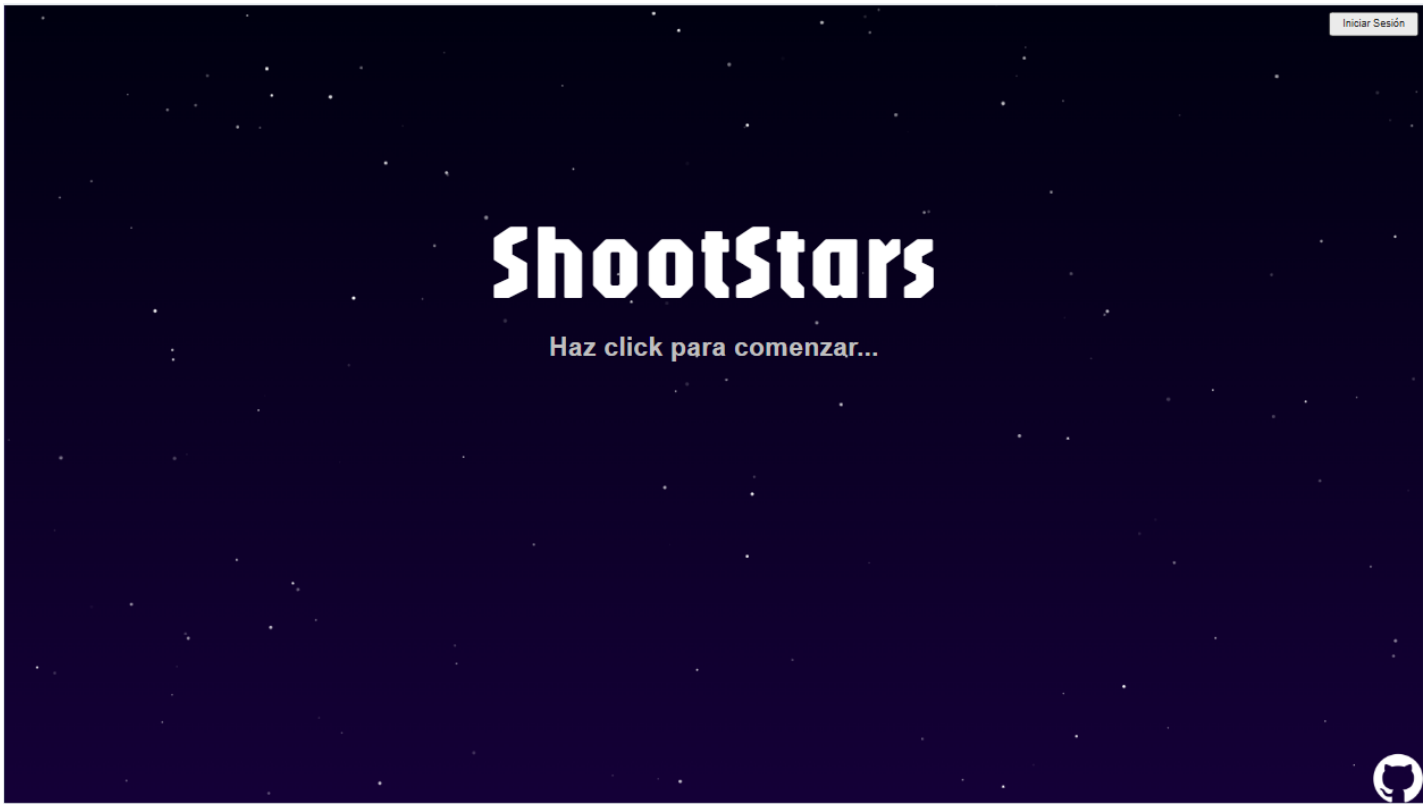
1. El administrador recibe una notificación por correo con un link al mensaje reportado o accede a todos los mensajes desde una lista en el panel de la web.

2. Se muestra el mensaje, el número de reportes y las acciones a realizar: eliminar el mensaje o dejarlo.

3. En caso de eliminar el mensaje, y si el autor acumula varios mensajes eliminados por infracciones, se bloquea su cuenta.

Mockups

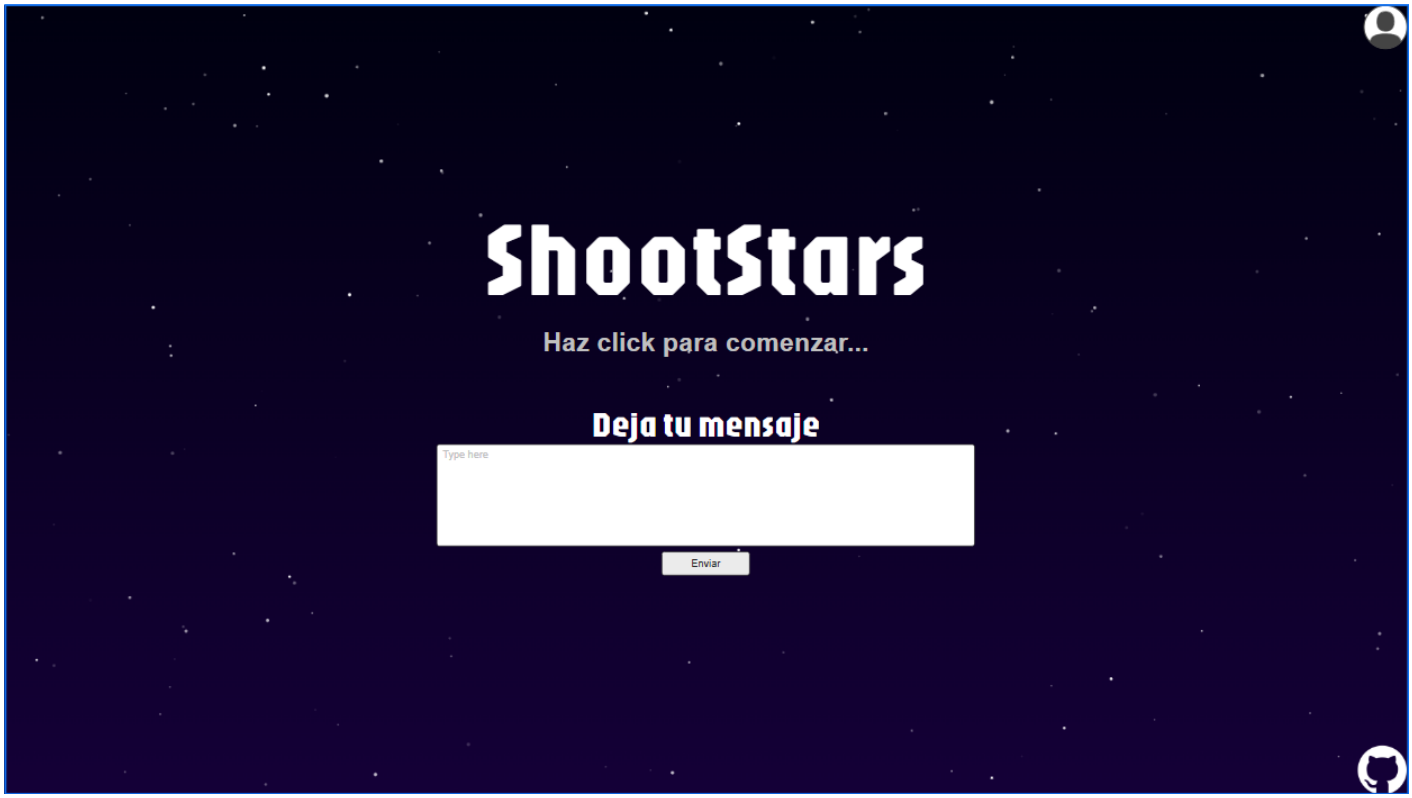
Página principal



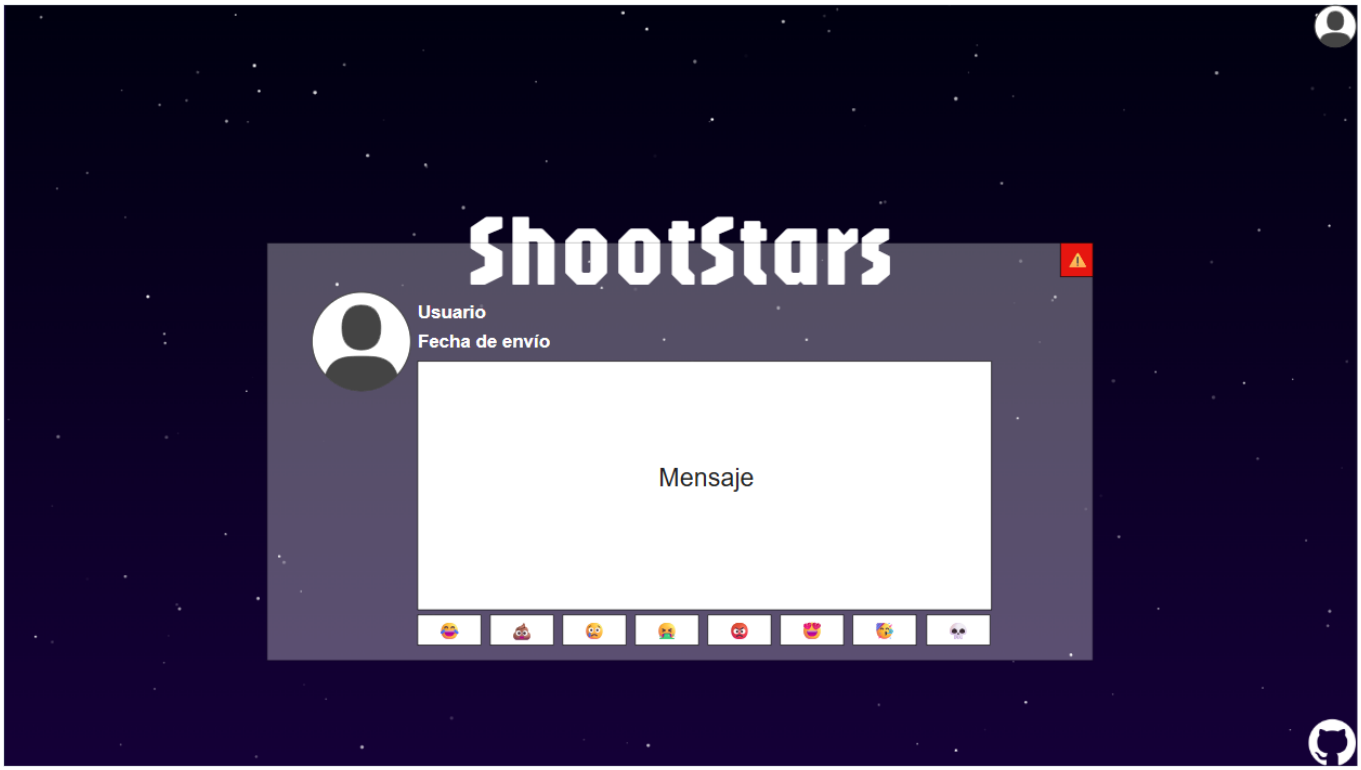
Iniciar Sesión

Registrarse

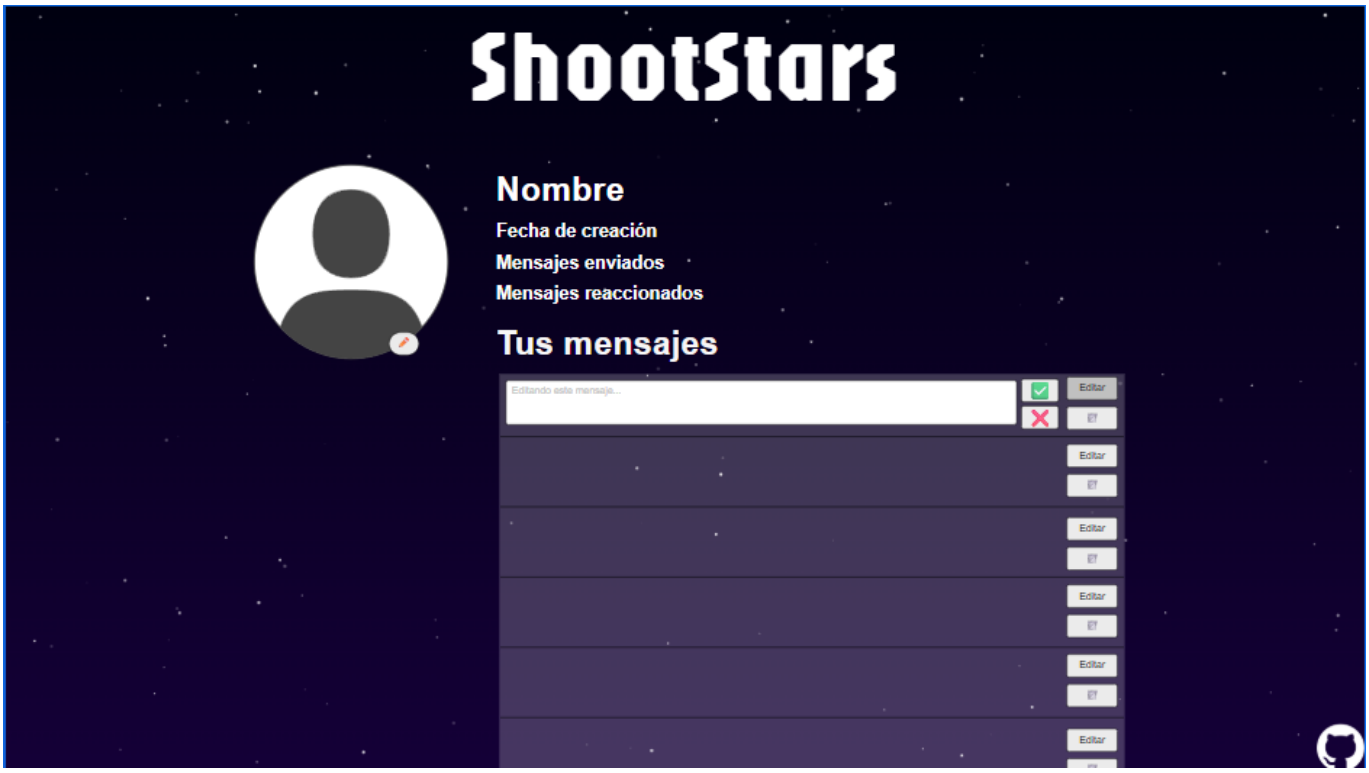
Una vez registrado



Cuando el usuario haga click en cualquier parte

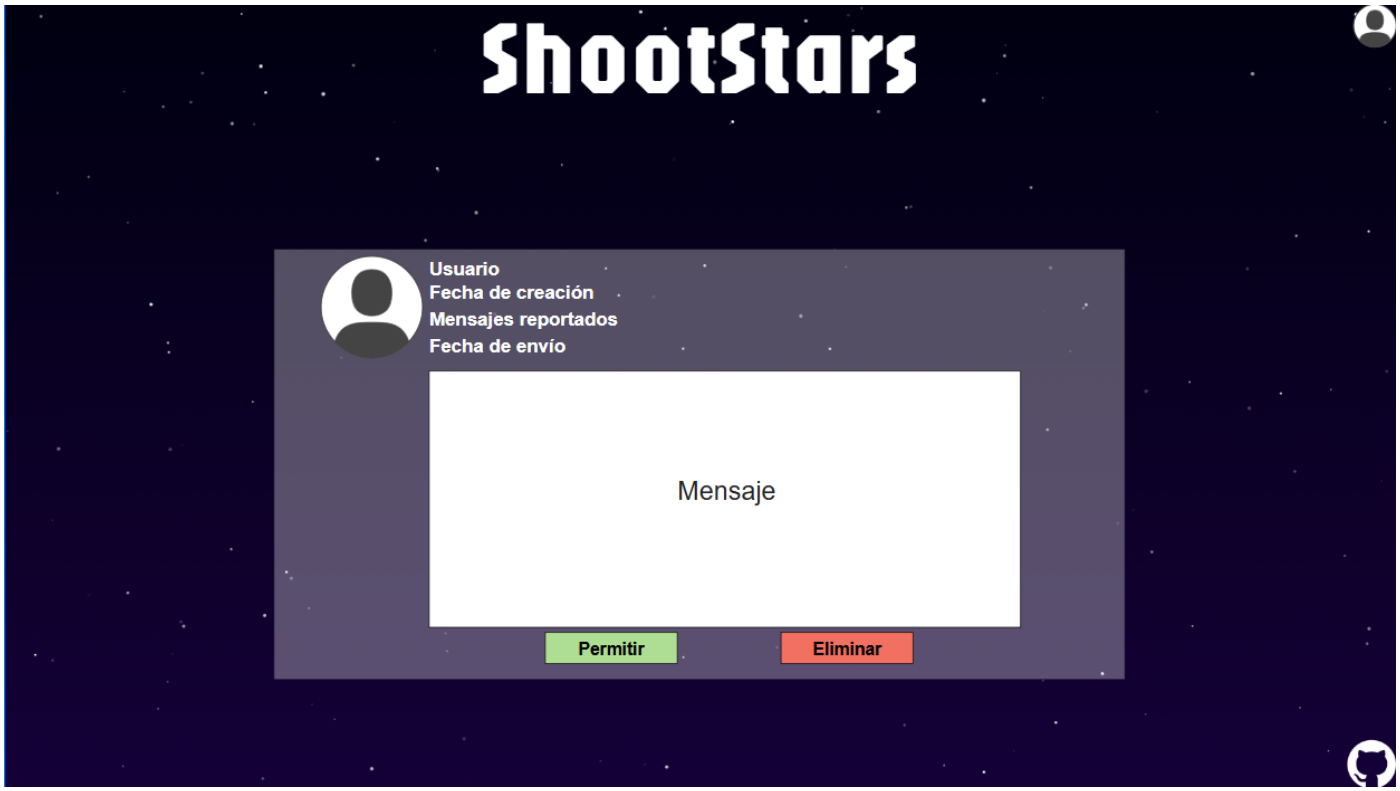


Página del usuario



Página de gestión del administrador

Gestión de mensaje



Diseño de la base de datos

USUARIOS ──< MENSAJES ──< REACCIONES

│ │

│ └──< REPORTES

│

└──< BLOQUEOS

Tabla: usuarios

Campos:

- id_usuario (INT, PK, AI): Identificador único del usuario.
- nombre_usuario (VARCHAR(50)): Nombre visible del usuario.
- email (VARCHAR(100)): Correo electrónico (único).
- password_hash (VARCHAR(255)): Contraseña cifrada.
- rol (ENUM('usuario','admin')): Tipo de cuenta.
- creado_en (DATETIME): Fecha de creación.

Relaciones:

Un usuario puede tener muchos mensajes, reacciones y reportes.

Tabla: mensajes

Campos:

- id_mensaje (INT, PK, AI): Identificador único del mensaje.
- id_usuario (INT, FK): Autor del mensaje.
- contenido (TEXT): Texto del mensaje.
- creado_en (DATETIME): Fecha de publicación.
- editado_en (DATETIME NULL): Fecha de última edición.
- eliminado (BOOLEAN): Indica si fue eliminado.
- Reportes (INT): número de reportes que lleva el mensaje

Relaciones:

Un usuario puede tener muchos mensajes.

Un mensaje puede tener muchas reacciones y reportes.

Tabla: reacciones

Campos:

- id_reaccion (INT, PK, AI): Identificador único de la reacción.
- id_mensaje (INT, FK): Mensaje al que pertenece.
- id_usuario (INT, FK): Usuario que realizó la reacción.
- tipo (ENUM('me_gusta','risa','triste','sorpresa','enfado')): Tipo de reacción.
- creada_en (DATETIME): Fecha de la reacción.

Relaciones:

Un mensaje puede tener muchas reacciones. Cada usuario puede reaccionar una sola vez por mensaje (restricción UNIQUE(id_mensaje, id_usuario)).

Tabla: reportes

Campos:

- id_reporte (INT, PK, AI): Identificador único del reporte.
- id_mensaje (INT, FK): Mensaje reportado.
- id_usuario (INT, FK): Usuario que reporta.
- motivo (VARCHAR(255)): Razón del reporte.
- fecha_reporte (DATETIME): Fecha en que se realizó el reporte.
- revisado (BOOLEAN): Indica si el reporte fue gestionado por el administrador.

Relaciones:

Un mensaje puede tener múltiples reportes.

Cuando el número de reportes supera un umbral, se notifica al administrador.

Tabla: bloqueos

Campos:

- id_bloqueo (INT, PK, AI): Identificador único del bloqueo.
- id_usuario (INT, FK): Usuario al que se le aplicó el bloqueo.
- fecha_bloqueo (DATETIME): Fecha en la que se realizó el bloqueo.

Relaciones:

Un usuario solo puede tener un bloqueo si tiene varias infracciones.

Relaciones principales entre tablas:

usuarios (1 - N) mensajes

mensajes (1 - N) reacciones

mensajes (1 - N) reportes

usuarios (1 - N) reacciones

usuarios (1 - N) reportes

usuarios (1 - 1) bloqueos

Introducción

En el contexto actual de la comunicación digital, las redes sociales y las plataformas de mensajería instantánea han adquirido un papel fundamental en la vida cotidiana de los usuarios. La inmediatez, la constante interacción y la posibilidad de compartir contenidos de manera masiva han transformado profundamente la forma en la que las personas se expresan y se relacionan. No obstante, esta hiperconectividad también ha generado un entorno donde la información se vuelve excesiva, permanente y, en muchas ocasiones, carente de espontaneidad o autenticidad.

Con el objetivo de ofrecer una alternativa más efímera, personal y controlada, este proyecto propone el desarrollo de una aplicación web de mensajes efímeros con sistema de reacciones y moderación de contenido, orientada a fomentar una comunicación más libre, sencilla y emocional entre los usuarios. La propuesta combina aspectos técnicos propios del desarrollo web con una reflexión sobre el uso responsable de la tecnología y la gestión ética del contenido digital.

El funcionamiento principal de la aplicación se basa en la visualización aleatoria de mensajes almacenados en una base de datos. Al acceder a la plataforma, el usuario podrá pulsar en cualquier parte de la pantalla para recibir un mensaje seleccionado de manera aleatoria, al cual podrá reaccionar mediante un “me gusta” o diferentes emoticonos. La característica efímera del sistema se manifiesta en que, una vez cerrado, el mensaje no podrá volver a visualizarse, lo que contribuye a reforzar la idea de unicidad y transitoriedad en la comunicación. Esta decisión de diseño se inspira en la metáfora de las “estrellas fugaces”, representando cada mensaje como un instante breve e irrepetible dentro de un universo digital en constante movimiento.

La aplicación permitirá además que los usuarios registrados creen y gestionen sus propios mensajes desde un panel de control personal, donde podrán consultar las reacciones recibidas, editar el contenido o eliminarlo en cualquier momento.

Para garantizar un entorno seguro y adecuado, se implementará un sistema de moderación y reporte de contenido. Cualquier usuario podrá señalar un mensaje como inapropiado, y una vez se alcance un número determinado de reportes, el sistema notificará automáticamente al administrador mediante correo electrónico. El administrador, tras revisar el contenido reportado, podrá decidir si procede su eliminación o si debe mantenerse visible. En caso de reincidencia por parte de un mismo usuario, el sistema contemplará la suspensión de su cuenta.

Desde el punto de vista técnico, el proyecto se desarrollará mediante un conjunto de tecnologías web consolidadas y de libre uso. En la capa de presentación, se emplearán HTML y CSS para el diseño visual y la estructura de la interfaz de usuario, junto con JavaScript para la gestión de la interacción dinámica y la comunicación asíncrona con el servidor. En la capa de lógica de negocio, se utilizará PHP como lenguaje principal del lado del servidor, encargado de procesar las peticiones, gestionar las sesiones de usuario y

comunicar con la base de datos. Por último, la persistencia de los datos se llevará a cabo mediante SQL, empleando un sistema de gestión de bases de datos MySQL.

El desarrollo se realizará en el entorno de Visual Studio Code, aprovechando su compatibilidad con múltiples lenguajes y extensiones, y se integrará Git y GitHub como herramientas de control de versiones y respaldo del código. Asimismo, se hará uso de PHPMailer para el envío de correos electrónicos relacionados con el sistema de doble autenticación y las notificaciones administrativas. La gestión de la base de datos se realizará mediante MySQL Workbench, facilitando la creación, diseño y mantenimiento de las estructuras de datos.

En cuanto al entorno de despliegue, la aplicación será alojada en un servidor propio configurado con el sistema operativo Debian, el cual actuará tanto como servidor web mediante Apache, como servidor de base de datos. Esta elección responde a la intención de disponer de un entorno de ejecución completamente controlado, garantizando la estabilidad, la personalización del sistema y la independencia respecto a servicios de alojamiento externos.

El proyecto se desarrollará en un período estimado de diez semanas, siguiendo una metodología ágil que abarcará las fases de planificación, diseño, desarrollo, pruebas y despliegue final. Durante todo el proceso se elaborará la documentación correspondiente, con el objetivo de asegurar la trazabilidad, la reproducibilidad y la correcta evaluación del proyecto en todas sus etapas.

Metodología de trabajo

Para el desarrollo de este proyecto se utilizará una metodología ágil e incremental, adaptada a las características del propio desarrollo y al entorno de trabajo individual. Esta metodología permitirá una evolución progresiva del sistema, realizando entregas funcionales parciales que puedan ser probadas y mejoradas en cada iteración.

El enfoque se basa en dividir el trabajo en fases cortas y cíclicas, en las que se desarrollan y prueban partes concretas de la aplicación antes de avanzar a la siguiente. Cada ciclo incluirá tareas de diseño, programación, pruebas y documentación.

Asimismo, se mantendrá una documentación continua durante todas las fases, registrando los avances técnicos, las decisiones de diseño y las incidencias encontradas.\

Presupuesto

Concepto	**Coste aproximado**
Servidor propio	150 €
Software y licencias	0 €
Dominio	0 €
Valoración del trabajo (10€/h)	1.700 €
Total aproximado	**≈1.850 €**

Planificación de tiempos

Fase 1. Planificación y análisis (Semana 1)

Duración estimada: 1 semana
Objetivos:

- Definir los requisitos funcionales y no funcionales del sistema.
- Identificar los actores principales y casos de uso.
- Elaborar el diseño preliminar de la base de datos y la arquitectura de la aplicación.
- Establecer el entorno de trabajo (repositorio Git, servidor Debian, herramientas de desarrollo).

Fase 2. Diseño de la interfaz y la base de datos (Semanas 2–3)

Duración estimada: 2 semanas
Objetivos:

- Diseñar los bocetos de las principales pantallas de la aplicación.
- Crear el esquema físico de la base de datos y las relaciones entre tablas.
- Definir el estilo visual y paleta de colores inspirada en la temática espacial.
- Configurar las tablas iniciales en MySQL y probar su conexión con PHP.

Fase 3. Desarrollo del backend (Semanas 4–6)

Duración estimada: 3 semanas
Objetivos:

- Implementar el sistema de registro e inicio de sesión.
- Integrar la doble autenticación (2FA) con envío de código mediante PHPMailer.
- Desarrollar las operaciones CRUD de los mensajes (crear, editar, eliminar, listar).
- Implementar las reacciones y el sistema de reportes.
- Crear los endpoints necesarios para la comunicación con el frontend (AJAX).

Fase 4. Desarrollo del frontend (Semanas 6–8)

Duración estimada: 2 semanas
Objetivos:

- Crear la interfaz visual y los componentes principales en HTML, CSS y JavaScript.
- Implementar el sistema de mensajes aleatorios con transiciones y efectos visuales.
- Integrar la gestión de reacciones y reportes mediante llamadas asíncronas.
- Adaptar el diseño a diferentes dispositivos (diseño responsive).

Fase 5. Pruebas, optimización y documentación (Semanas 9–10)

Duración estimada: 2 semanas
Objetivos:

- Realizar pruebas unitarias y de integración.
- Comprobar el rendimiento del servidor y optimizar las consultas SQL.
- Verificar la seguridad del sistema (validación de formularios, cifrado, gestión de sesiones).
- Elaborar la documentación técnica y de usuario final.
- Terminar la memoria final y la presentación del proyecto.