Universidad de San Carlos de Guatemala

Centro Universitario de Occidente

División de Ciencias de la Ingeniería

Manual técnico Ola Ke Hace

Proyecto Final

Teoría de Sistemas 1

Ing. Pedro Domingo

Segundo semestre 2024

#4 - 202031953 - Hania Mirleth Mazariegos Alonzo

Índice

Marco Teórico

Análisis de la plataforma desde el pensamiento Sistémico

Componentes del Sistema

- Usuarios (Entidades clave): Publicadores, administradores, asistentes a eventos
- Posts (Publicaciones): Título, descripción, fechas, capacidad, estado.
- Eventos: Fechas de inicio/fin, ubicación, capacidad, etc.
- Asistencias: Registro de usuarios que asisten a eventos.
- Notificaciones: Sistema de alertas para los usuarios (como asistencia registrada o aprobación de posts).
- Reportes: Función para que los usuarios reporten posts.
- Búsqueda: Función de búsqueda por título, descripción o fecha.
- Moderación (Administradores): Aprobación o rechazo de posts y reportes.

Entradas del Sistema

- Acciones de los usuarios:
    - Creación de publicaciones (publicadores).
    - Registro de asistencia a eventos.
    - Reporte de publicaciones.
    - Búsqueda de eventos (por título, descripción o fechas).
    - Aprobación/rechazo de publicaciones (administradores).
- Datos de los formularios:
    - Datos de publicación (título, fecha, descripción, etc.).
    - Motivos de reporte y comentarios.
    - Rango de fechas o términos de búsqueda.

Salidas del Sistema

- Notificaciones: Para los usuarios (asistencia confirmada, posts pendientes de aprobación).
- Listado de Posts: Mostrados en el home, incluyendo detalles de eventos y estado de asistencia.
- Resultados de búsqueda: Posts filtrados según criterios de búsqueda.
- Estados de los posts: Actualización visual en la interfaz (pendiente, activo, rechazado).
- Mensajes de confirmación: Confirmación de que el usuario ha registrado asistencia o reportado un post.

Relaciones

- Usuarios y Posts: Los publicadores crean posts, los usuarios asisten a eventos y los administradores aprueban o rechazan posts.
- Notificaciones y Eventos: Las notificaciones informan sobre eventos relacionados con los posts.

- Asistencias y Posts: Cada post tiene una lista de usuarios que han registrado asistencia.
- Moderación y Posts: Los administradores influyen en el estado del post y el publicador.
- Reportes y Posts: Los usuarios pueden reportar posts, lo que genera notificaciones y posibles acciones administrativas.

Actividades Clave

- Creación de publicaciones: Los publicadores generan eventos que deben ser revisados por un administrador si están en prueba (on_test).
- Registro de asistencia: Los usuarios pueden registrar su asistencia a eventos.
- Moderación: Los administradores revisan publicaciones en estado pending y toman decisiones.
- Generación de notificaciones: El sistema crea notificaciones para varios eventos, como la aprobación de posts o el registro de asistencia.
- Reporte de publicaciones: Los usuarios reportan contenido inapropiado, lo que activa procesos de revisión.
- Búsqueda de eventos: Los usuarios realizan búsquedas por criterios específicos para encontrar eventos.

Objetivos del Sistema

- Facilitar la creación y asistencia a eventos permitiendo que los publicadores creen eventos de manera eficiente y que los usuarios se registren en ellos.
- Garantizar la moderación y calidad de las publicaciones mediante un sistema de aprobación/rechazo, asegurando que los eventos sean adecuados y aprobados.
- Aumentar la interacción y participación a través de notificaciones y mecanismos de asistencia, incentivar la participación en eventos.
- Proporcionar transparencia y seguridad mediante reportes de usuarios y moderación por administradores.

Medios para Alcanzar los Objetivos

- Formulario de creación de publicaciones: Facilitar la creación de eventos con todos los detalles necesarios (fechas, capacidad, ubicación).
- Sistema de notificaciones: Informar a los usuarios de eventos importantes relacionados con sus interacciones.
- Mecanismos de aprobación de posts: Controlar la calidad del contenido publicado.
- Base de datos eficiente: Con tablas normalizadas para usuarios, posts, asistencias, reportes y notificaciones, que permite realizar consultas rápidas y mantener la consistencia.
- Interfaz amigable con JavaScript: Uso de tecnologías como JavaScript puro para crear una experiencia interactiva y dinámica (contadores de asistencia, cuenta regresiva, etc.).

Tecnologías Utilizadas

HTML - HyperText Markup Language

HTML es el lenguaje estándar utilizado para estructurar y presentar contenido en la World Wide Web. Es el esqueleto de una página web, definiendo la estructura básica y el significado de cada elemento.

Conceptos Básicos:

● Elementos: Son las unidades fundamentales de HTML. Se representan con etiquetas, las cuales se escriben entre corchetes angulares (<) y (>). De modo que <p> define un párrafo, <h1> un encabezado de nivel 1, etc.

● Etiquetas: Cada elemento tiene una etiqueta de apertura y una de cierre. La etiqueta de apertura indica el comienzo del elemento, y la de cierre (con una barra inclinada al principio) indica su final. Por ejemplo:

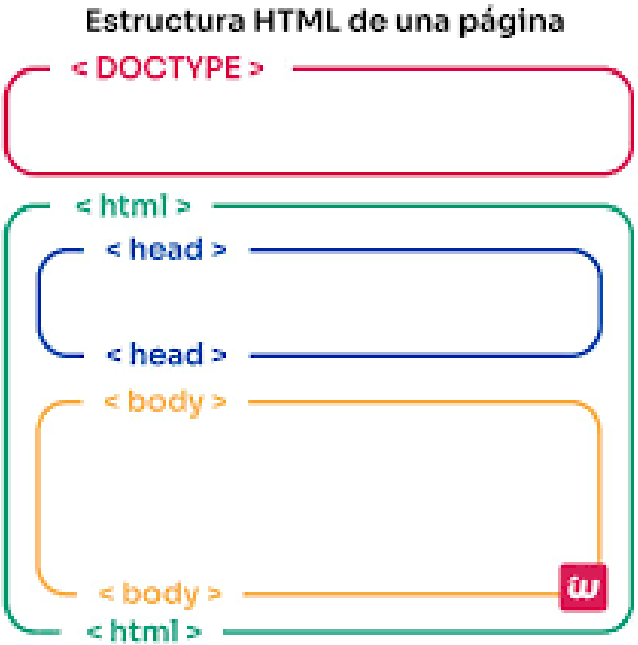
```
<p> Contenido del párrafo.</p>
```

● Atributos: Proporcionan información adicional sobre un elemento. Se añaden dentro de la etiqueta de apertura. Por ejemplo, el atributo href en un enlace <a> especifica la URL a la que enlaza:

```
<a href="https://www.example.com">Ir a Example</a>
```

Estructura de un documento HTML:

Un documento HTML básico tiene una estructura definida que incluye la declaración del tipo de documento (DOCTYPE), la etiqueta <html>, la etiqueta <head> (para metadatos) y la etiqueta <body> (para el contenido visible). De modo:



Elementos HTML Comunes

- Encabezados: <h1>, <h2>, <h3>, etc.
- Párrafos: <p>
- Enlaces: <a>
- Imágenes: <img>
- Listas: <ul> (listas sin orden), <ol> (listas ordenadas), <li> (elementos de lista)
- Tablas: <table>, <tr>, <td>
- Divs y spans: y son contenedores genéricos para agrupar elementos.

JavaScript

JavaScript surgió a mediados de los años 90 como un lenguaje de scripting diseñado para añadir interactividad a las páginas web estáticas. Creado por Brendan Eich, trabajando en Netscape, con el objetivo de hacer que la web fuera más dinámica y atractiva para los usuarios. A lo largo de los años, JavaScript ha evolucionado significativamente, convirtiéndose en uno de los lenguajes de programación más populares y versátiles del mundo.

Características Generales

- Lenguaje interpretado: El código JavaScript se ejecuta directamente por el navegador, sin necesidad de una compilación previa.
- Orientado a objetos: Aunque no es un lenguaje de objetos puros como Java, JavaScript utiliza prototipos para crear objetos y heredar propiedades.
- Débilmente tipado: No es necesario declarar el tipo de dato de las variables antes de usarlas.
- Multi-paradigma: Soporta programación orientada a objetos, funcional y procedural.
- Asíncrono: Permite ejecutar múltiples tareas de forma simultánea, lo que es fundamental para crear aplicaciones web responsivas.

Usos principales:

- Desarrollo web front-end: JavaScript es el lenguaje principal para crear interfaces de usuario interactivas, manipular el DOM (Document Object Model) y responder a eventos del usuario.
- Desarrollo web back-end**:** Node.js, un entorno de ejecución de JavaScript, permite crear aplicaciones del lado del servidor escalables y eficientes.
- Desarrollo de aplicaciones móviles: Frameworks como React Native y Ionic permiten construir aplicaciones móviles híbridas utilizando JavaScript.

- Desarrollo de juegos: JavaScript se utiliza para crear juegos web y aplicaciones de realidad virtual.
- Internet de las Cosas (IoT): JavaScript se emplea en dispositivos IoT para controlar y comunicarse con sensores y actuadores.

Ventajas:

- Facilidad de aprendizaje: Su sintaxis es relativamente sencilla y existen numerosos recursos disponibles para aprender.
- Amplia comunidad: Cuenta con una gran comunidad de desarrolladores que contribuyen a su crecimiento y ofrecen soporte.
- Versatilidad: Se puede utilizar en una amplia variedad de proyectos y plataformas.
- Rendimiento: Los motores JavaScript modernos son altamente optimizados, lo que permite crear aplicaciones rápidas y eficientes.
- Integración con otras tecnologías: Se integra fácilmente con HTML y CSS, y puede utilizarse con frameworks y bibliotecas como React, Angular y Vue.

Desventajas

- Seguridad**:** Un manejo incorrecto de JavaScript puede exponer aplicaciones a vulnerabilidades de seguridad.
- Compatibilidad entre navegadores: Diferentes navegadores pueden interpretar el código JavaScript de manera ligeramente diferente, lo que puede requerir el uso de transpiladores.
- Complejidad**:** A medida que las aplicaciones se vuelven más grandes y complejas, la gestión del código JavaScript puede volverse desafiante.

PHP Hypertext Preprocessor

PHP es un lenguaje de programación de código abierto muy popular, especialmente diseñado para el desarrollo web. Se utiliza principalmente para crear el lado del servidor (backend)
en aplicaciones web dinámicas, las cuales generan contenido personalizado en respuesta a las solicitudes de los usuarios.

Características principales:

- Fácil de aprender y usar
- Incrustado en HTML: El código PHP se puede mezclar directamente con el código HTML, facilitando la creación de páginas web dinámicas.
- Gran comunidad y soporte.
- Amplia variedad de funcionalidades
- Plataforma cruzada: Puede ejecutarse en la mayoría de los sistemas operativos y servidores web.
- Integración con otras tecnologías: Se integra fácilmente con bases de datos (MySQL, PostgreSQL, etc.), frameworks (Laravel, Symfony, etc.) y otras tecnologías web.

Normalmente se usa en el desarrollo de sitios web dinámicos con contenido personalizado, formularios de contacto, sistemas de gestión de contenido (CMS) como WordPress. Sin embargo, también se usa en desarrollo de aplicaciones web más complejas, como tiendas en línea, o incluso redes sociales pequeñas. Por último se puede usar también para el procesamiento de datos, manipulación y análisis de datos, generación de informes, etc.

Ventajas:

- Es de código abierto
- Rendimiento: Es un lenguaje rápido y eficiente.
- Flexibilidad: Se adapta a una amplia variedad de proyectos.
- Cuenta con una gran cantidad de herramientas y recursos.
- Es un lenguaje maduro y estable.

Desventajas:

- Mala calidad de manejo de errores, no hay herramientas de depuración.
- Rendimiento variable, en comparación con otros lenguajes de programación.
- Falta de estructura, su flexibilidad puede causar código desordenado y difícil de mantener
- Baja seguridad, es difícil garantizar la seguridad de los datos a menos que se tengan conocimientos profundos o se maneje a nivel de motor de base de datos.

Variables Globales en PHP

En PHP, una variable global es aquella que puede ser accedida desde cualquier parte de un script, independientemente de su ubicación dentro de funciones o bloques de código. Esto significa que su ámbito es global, es decir, abarca todo el script. Tradicionalmente, para utilizar una variable global dentro de una función, se utilizaba la palabra clave global, Sin embargo no se recomienda usarlas debido a su falta de encapsulación, el aumento de dependencias que puede hacer que el código sea menos modular y reutilizable y el riesgo de conflictos de nombres, especialmente en proyectos grandes.

Variables superglobales:

PHP proporciona un conjunto de variables predefinidas que son accesibles desde cualquier parte del script, como $_GET, $_POST, $_SESSION, etc. Estas variables son útiles para almacenar datos que se necesitan en todo el script. De modo que se pueden pasar variables como argumentos a las funciones para evitar tener que declararlas como globales y en programación orientada a objetos, se pueden encapsular los datos en objetos y acceder a ellos a través de métodos, lo que mejora la organización y la seguridad del código. Aunque no se recomienda su uso excesivo, son útiles en para configuración de la aplicación y
almacenamiento de constantes.

MariaDB

MariaDB es un sistema de gestión de bases de datos relacionales (RDBMS) de código abierto que ha ganado mucha popularidad en los últimos años. Es una excelente opción para almacenar y gestionar grandes cantidades de datos de manera organizada y eficiente.

Puntos fuertes

- Rendimiento: Conocido por su alto rendimiento y optimización.
- Compatibilidad: Es altamente compatible con MySQL, lo que facilita la migración de bases de datos existentes.
- Innovación: MariaDB se encuentra en constante desarrollo, incorporando nuevas características y mejoras de rendimiento.
- Comunidad: Cuenta con una gran comunidad de desarrolladores y usuarios, lo que garantiza un amplio soporte y recursos disponibles.
- Licencia: Al ser de código abierto, MariaDB es gratuito y puede ser utilizado sin restricciones.

Características Principales

- Utiliza el lenguaje estándar SQL para interactuar con la base de datos.
- Por trabajar con bases de datos relacionales, organiza los datos en tablas, con filas (registros) y columnas (campos).
- Su manejo de Índices acelera la búsqueda y recuperación de datos.
- Transacciones: Garantiza la integridad de los datos mediante transacciones atómicas.
- Permite replicar los datos en múltiples servidores para mejorar la disponibilidad y el rendimiento.
- Es ampliamente usada en desarrollo de aplicaciones web, análisis de datos, y aplicaciones empresariales

Ejemplos:

Crear una tabla:

```
CREATE TABLE usuarios (
id INT PRIMARY KEY,
nombre VARCHAR(50),
email VARCHAR(100)
);
```

Insertar un registro:

```
INSERT INTO usuarios (id, nombre, email)
VALUES (1, 'Juan Pérez', 'juan@example.com');
```

Consultar datos:

```
SELECT * FROM usuarios;
SELECT nombre, email FROM usuarios WHERE id = 1;
```

Triggers

Un trigger (disparador o desencadenador) es un procedimiento almacenado que se ejecuta automáticamente cuando ocurre un evento específico en una tabla. Estos eventos suelen ser operaciones de INSERT, UPDATE o DELETE.

Podemos usarlos para asegurar que las relaciones entre las tablas se mantienen consistentes, para calcular valores automáticamente o implementar reglas de negocio complejas que no se pueden expresar fácilmente con restricciones. Su estructura básica es:

```
CREATE TRIGGER nombre_trigger
BEFORE | AFTER INSERT | UPDATE | DELETE ON tabla
FOR EACH ROW
BEGIN
-- Cuerpo del trigger
END;
```

Conectar PHP a MariaDB

PHP, como lenguaje de scripting del lado del servidor, ampliamente utilizado para crear aplicaciones web dinámicas. MariaDB, es un sistema de gestión de bases de datos relacionales muy popular. Al conectar ambos, podemos crear aplicaciones web que almacenen, recuperen y manipulen datos.

Existen principalmente dos formas de conectar PHP a MariaDB:

MYSQLI:

Es una extensión nativa de PHP específicamente diseñada para interactuar con MySQL y MariaDB. Tiene una orientación procedural y está orientada a objetos, lo que brinda flexibilidad en el desarrollo. La sintaxis usada es la siguiente:

```
$conexion = new mysqli($host, $user, $pass, $dbname, $port, $socket);
```

Donde:

$host: Nombre del servidor de la base de datos.
$user: Nombre de usuario para la base de datos.
$pass: Contraseña del usuario.
$dbname: Nombre de la base de datos.
$port: Puerto (generalmente se omite).
$socket: Socket para conexiones Unix (opcional).

PDO (PHP Data Objects):

Es una abstracción de acceso a datos, proporciona una interfaz uniforme para acceder a diferentes bases de datos, no solo MySQL/MariaDB. Orientado objetos completamente.Su sintaxis es:

```
$dsn = "mysql:host=$host;dbname=$dbname";
$options = array(
PDO::ATTR_ERRMODE => PDO::ERRMODE_EXCEPTION,
PDO::ATTR_DEFAULT_FETCH_MODE => PDO::FETCH_ASSOC,
);
$conexion = new PDO($dsn, $user, $pass, $options);
```

Donde:

$dsn: Data Source Name, especifica el tipo de base de datos y los parámetros de conexión.

$options: Un array asociativo para configurar opciones adicionales, como el modo de manejo de errores y el formato de los resultados.

.htaccess

.htaccess es un archivo de configuración que se encuentra en los servidores web Apache. Actúa como un mini-configurador, permitiendo realizar ajustes específicos para un directorio en particular o para todo el sitio web, sin necesidad de modificar la configuración global del servidor.

El archivo .htaccess debe colocarse en el directorio donde deseas que las reglas se apliquen. Si quieres que las reglas se apliquen a todo el sitio, coloca el archivo en el directorio raíz de tu sitio web.

Algunas de las funcionalidades más comunes que se pueden implementar con .htaccess incluyen:

- Reescritura de URL: Permite crear URL amigables y significativas para los usuarios, ocultando la estructura real de las carpetas y archivos.
- Redirecciones: Redirige automáticamente a los usuarios de una URL a otra, ya sea de forma temporal o permanente.
- Protección de directorios: Restringe el acceso a ciertos directorios o archivos, requiriendo autenticación o bloqueando direcciones IP específicas.
- Personalización de errores: Muestra mensajes de error personalizados cuando se producen errores en el servidor.
- Configuración de encabezados: Agrega o modifica encabezados HTTP, como el encabezado Cache-Control para controlar el almacenamiento en caché.
- Gestión de tipos de MIME: Asocia extensiones de archivos con tipos MIME específicos, lo que permite que el navegador interprete correctamente los archivos.

- Bloqueo de acceso directo a archivos: Impide que los usuarios accedan directamente a archivos específicos, como imágenes o scripts.

Funcionamiento
El servidor Apache procesa los archivos .htaccess de forma secuencial, desde el directorio raíz hasta el directorio específico donde se encuentra el archivo solicitado. Cada vez que se solicita una URL, Apache busca un archivo .htaccess en el directorio actual y en todos los directorios superiores hasta encontrar uno.

XAMPP

XAMPP es un acrónimo que representa a un conjunto de software libre que facilita enormemente el desarrollo web. Sus siglas corresponden a:

- ➔ **X**: Cualquier sistema operativo (Windows, Linux, macOS)
- ➔ **A**pache: Un popular servidor web de código abierto
- ➔ **M**ariaDB/MySQL: Un sistema de gestión de bases de datos relacionales
- ➔ **P**HP: Un lenguaje de scripting ampliamente utilizado en el desarrollo web
- ➔ **P**erl: Otro lenguaje de scripting, aunque menos común en el desarrollo web actual

XAMPP proporciona un entorno de desarrollo local completo y fácil de instalar. Esto significa que se puede crear y probar las aplicaciones web directamente en la computadora, sin necesidad de un servidor remoto. Es ideal para el desarrollo de aplicaciones web utilizando PHP, HTML, CSS y JavaScript.

También es una excelente herramienta para aprender los fundamentos del desarrollo web y las tecnologías que lo componen. Facilita el proceso de realizar pruebas locales de una aplicación antes de desplegarla en un servidor en vivo.

Ventajas
- Fácil instalación.
- Todo en uno.
- Gratuito y de código abierto.
- Personalizable.
- Amplia documentación y gran comunidad de usuarios.

Al instalar XAMPP, se instala un servidor web Apache que escucha las solicitudes HTTP en un puerto específico (por defecto el 80) y las procesa. Si una solicitud llega a un archivo PHP, Apache lo envía al intérprete de PHP para que lo ejecute. Las bases de datos se almacenan en

MariaDB (o MySQL) y se pueden administrar a través de phpMyAdmin, una herramienta de administración de bases de datos web que viene incluida en XAMPP.

MVC - Modelo Vista Controlador

MVC es un patrón arquitectónico de software muy popular que se utiliza para separar las diferentes responsabilidades de una aplicación en tres partes principales:

Modelo:
Representa los datos de la aplicación y la lógica de negocio. Es decir, todo lo relacionado con la información que se va a manejar.

Vista:
Es la interfaz de usuario, lo que el usuario ve en pantalla. Muestra los datos del modelo de una manera visual y atractiva.

Controlador:
Actúa como intermediario entre el modelo y la vista. Recibe las solicitudes del usuario, actualiza el modelo y selecciona la vista adecuada para mostrar al usuario.

Beneficios

- Separación de responsabilidades: Cada componente tiene una función clara, lo que facilita el desarrollo, mantenimiento y escalabilidad de la aplicación.
- Reutilización de código: Las vistas y los controladores pueden ser reutilizados en diferentes partes de la aplicación.
- Pruebas: Es más fácil probar cada componente de forma independiente.
- Mantenibilidad: Al separar la lógica de negocio de la interfaz de usuario, es más fácil realizar cambios en la aplicación sin afectar otras partes.

Manual Técnico

Requisitos del Sistema

Para ejecutar correctamente la aplicación, se requiere al menos el siguiente software y configuraciones:

1. Servidor Web : Apache Versión 2.4.58 (Unix).
   Es fundamental que el servidor web esté configurado para interpretar archivos PHP.
2. Gestor de Base de Datos : MariaDB Versión 11.4.2.
   Se debe tener una instancia de MariaDB configurada con las credenciales de acceso especificadas en el archivo de configuración de la aplicación.
3. Lenguaje de Programación: PHP Versión 8.2.12
   Es necesario tener instalado PHP con las siguientes extensión mysqli para la conexión a la base de datos MariaDB.
4. Navegador Web:
   Cualquier navegador web moderno (Chrome, Firefox, Edge, Safari) con soporte para JavaScript y HTML5.
5. Sistema Operativo
   Windows, Linux o macOS: La aplicación debería funcionar en cualquiera de estos sistemas operativos, siempre y cuando se cumplan los requisitos anteriores.

Instalación

Configuración del Servidor:

- El documento raíz del servidor web debe apuntar al directorio raíz de la aplicación.
- Permisos de archivos: Asegurarse de que los archivos y directorios de la aplicación tengan los
- permisos de lectura y escritura adecuados para el usuario del servidor web.
- Instalación

Instalación XAMPP

1. Descargar: Visitar la página oficial de XAMPP:
   (https://www.apachefriends.org/index.html) y descargar la versión correspondiente al sistema operativo (Windows, macOS o Linux).
2. Ejecución: Ejecutar el instalador y seguir las instrucciones en pantalla. Aceptar las opciones predeterminadas.
3. Inicio de los servicios: Una vez instalado XAMPP, se deben iniciar los servicios de Apache y MySQL desde el panel de control de XAMPP.

Configuración del gestor de base de datos:

1. Acceder a phpMyAdmin: Desde un navegador acceder a http://localhost/phpmyadmin/.
2. Crear base de datos e Importar estructura: Crear una nueva base de datos y darle estructura con los scripts sql proporcionados junto con esta documentación.
3. Opcionalmente se puede ejecutar el script sample_inserts.sql para poblar ligeramente la base de datos desde un inicio.

Configuración de la aplicación

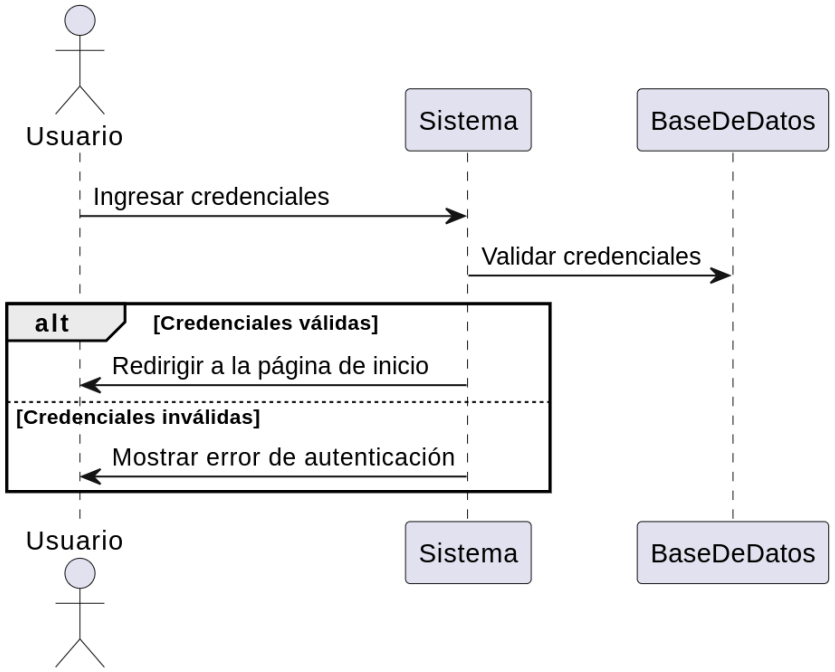
1. Ubicar:
   Se deben copiar los archivos del repositorio al directorio htdocs de XAMPP., o cambiar la configuración de XAMPP para que arranque desde la dirección de los archivos de la aplicación, ese sería el directorio raíz del servidor web local.
2. Configuración de conexión a la base de datos:
   Edita el archivo de conexión con base de datos de la aplicación, ubicado en /model/db.php, para asegurar que las credenciales de conexión a la base de datos (host, usuario, contraseña y nombre de la base de datos) sean correctas.
3. Permisos de archivos
   Verificar que los archivos de la aplicación tengan los permisos de lectura y escritura adecuados. En sistemas Linux, se pueden ajustar permisos con los comandos chmod y chown.

Acceso a la aplicación

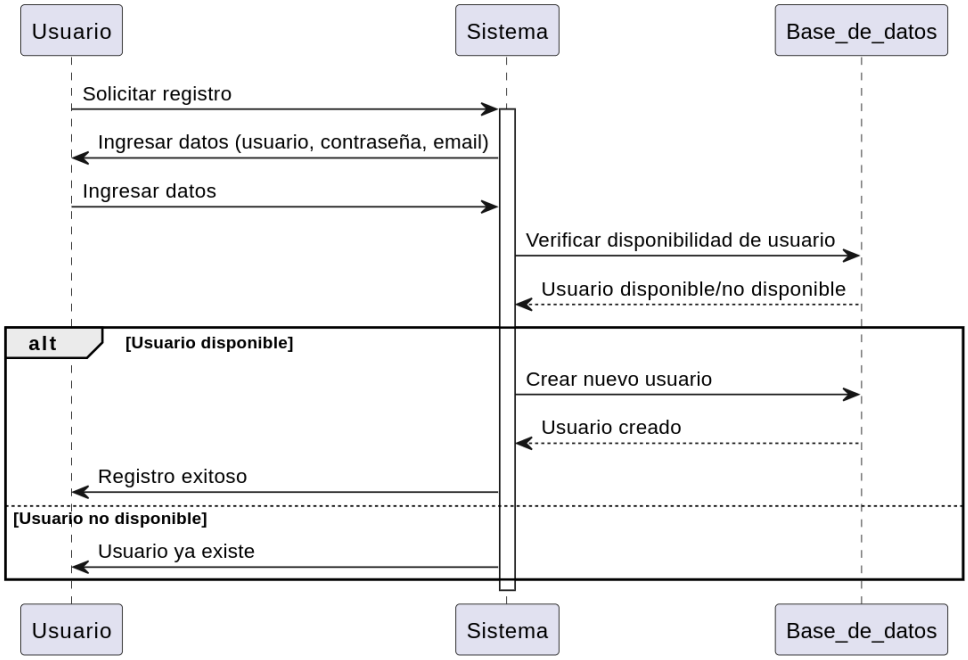
1. Abrir navegador:
   Abrir un navegador web y escribir la siguiente dirección en la barra de direcciones:
   http://localhost/o_k_h
2. En caso de haber modificado o ubicado de forma diferente la carpeta raíz de la aplicación, reemplazar o_k_h con el nombre de la carpeta donde están los archivos de la aplicación dentro del directorio htdocs de XAMPP.
3. Verificar funcionamiento:
   Debería verse la página principal de la aplicación. Si todo está configurado correctamente, debería ser posible interactuar con las diferentes funcionalidades de la aplicación.

Diagramas de Secuencia

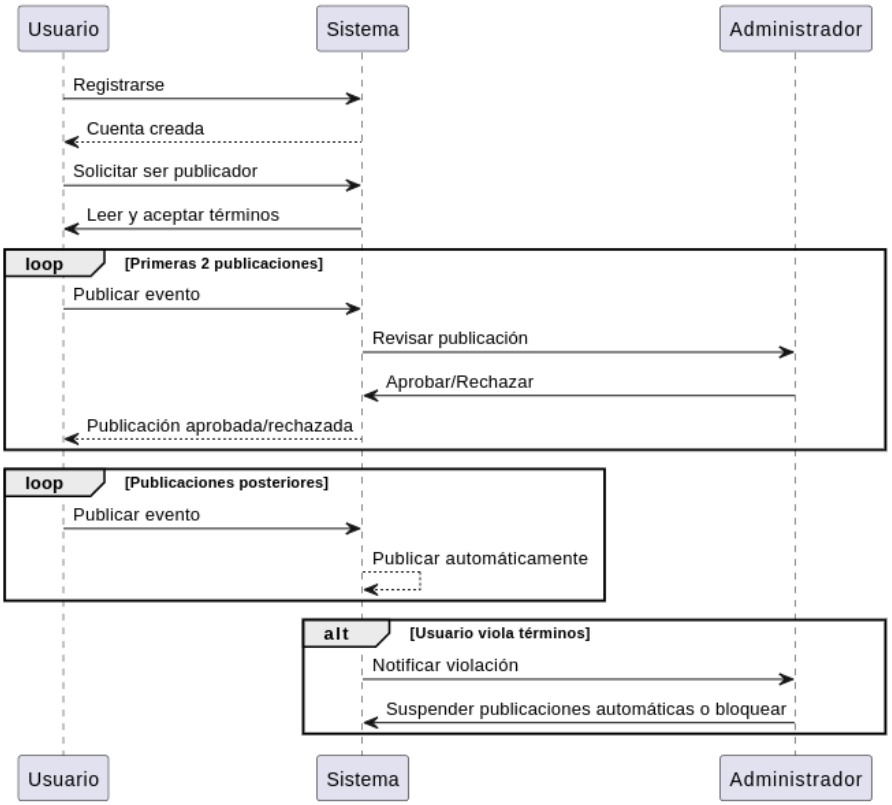
Iniciar sesión



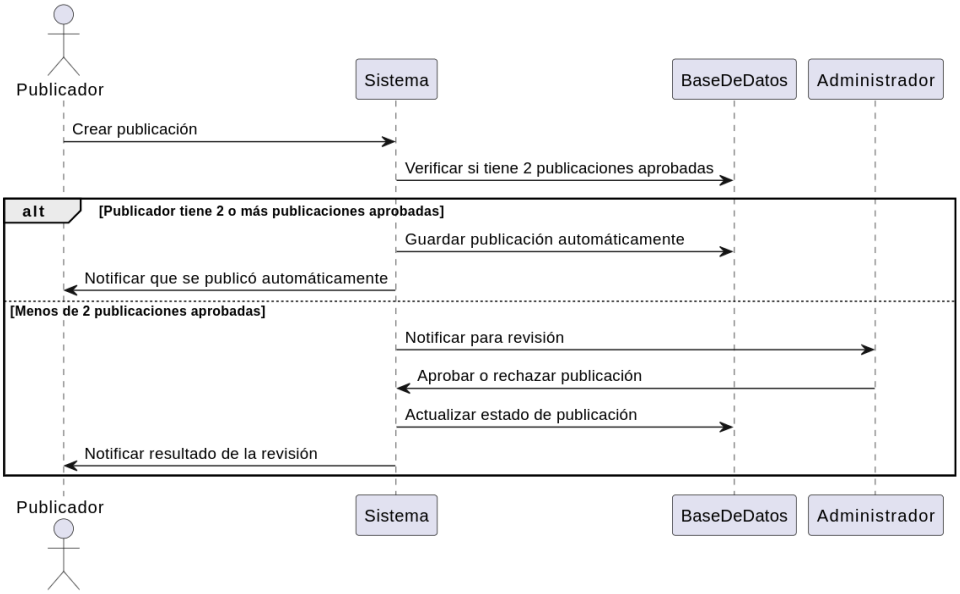
Registrarse / Crear cuenta



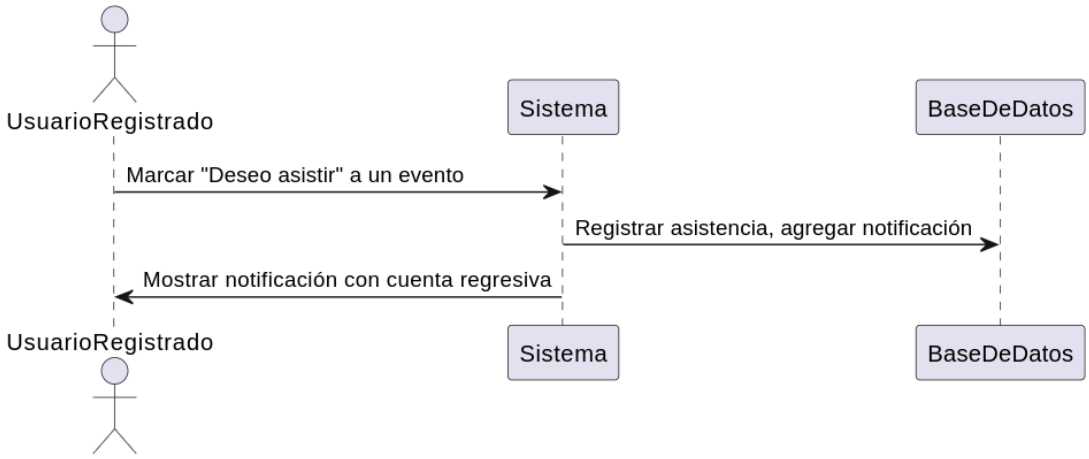
Convertirse en Publicador



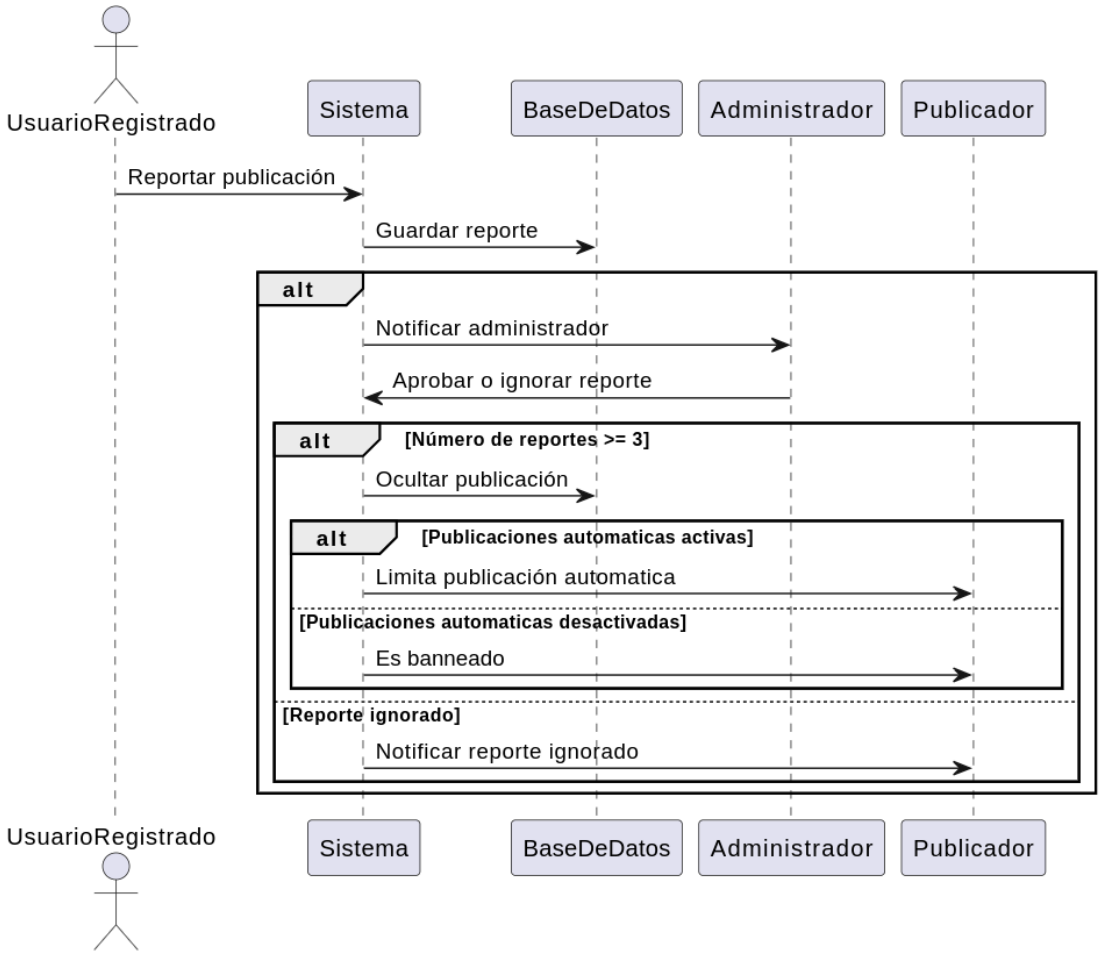
Crear nueva publicación



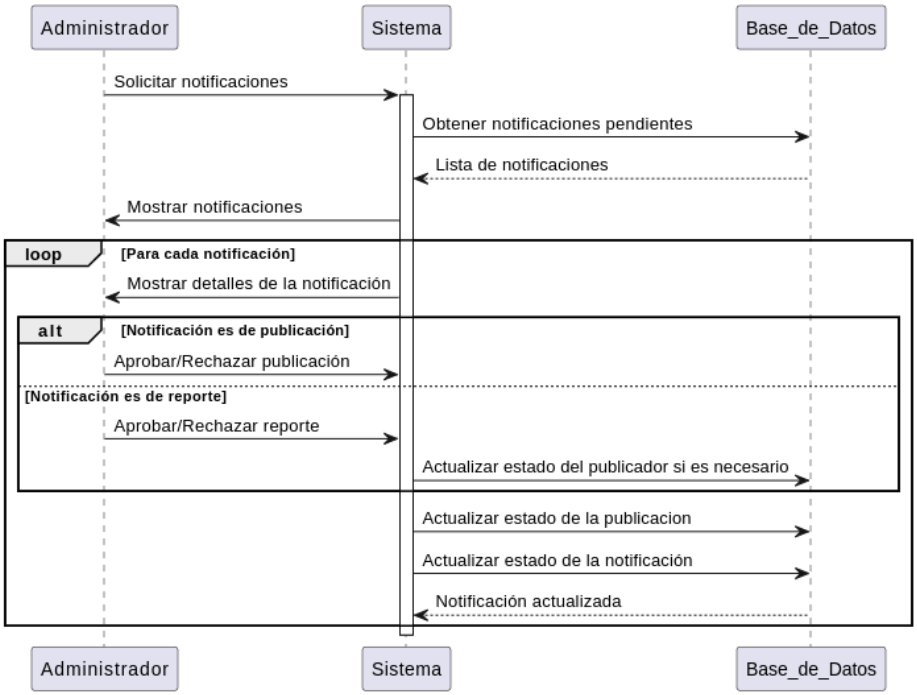
Registrar asistencia



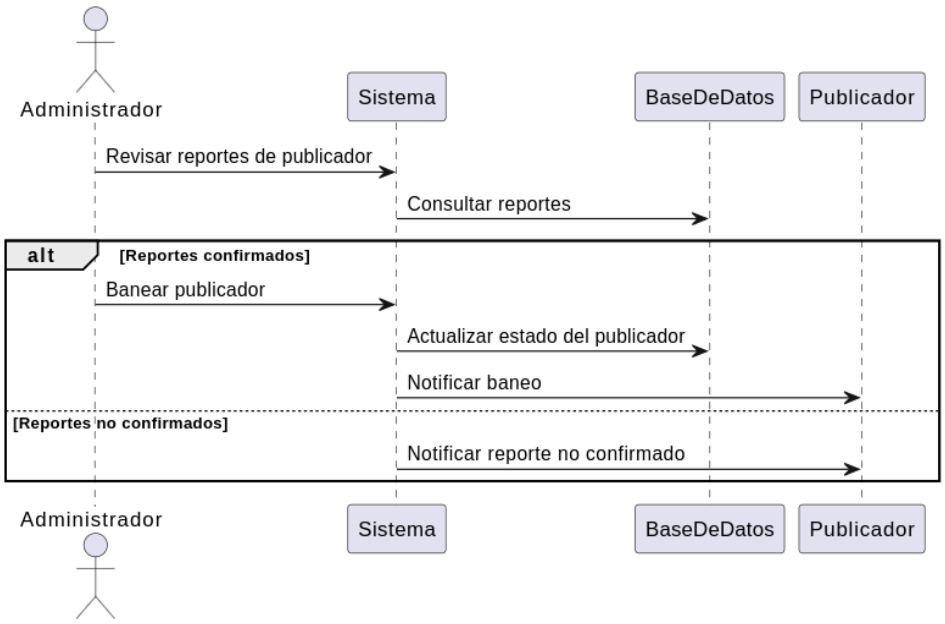
Reportar publicación



Revisar notificaciones (Administrador)

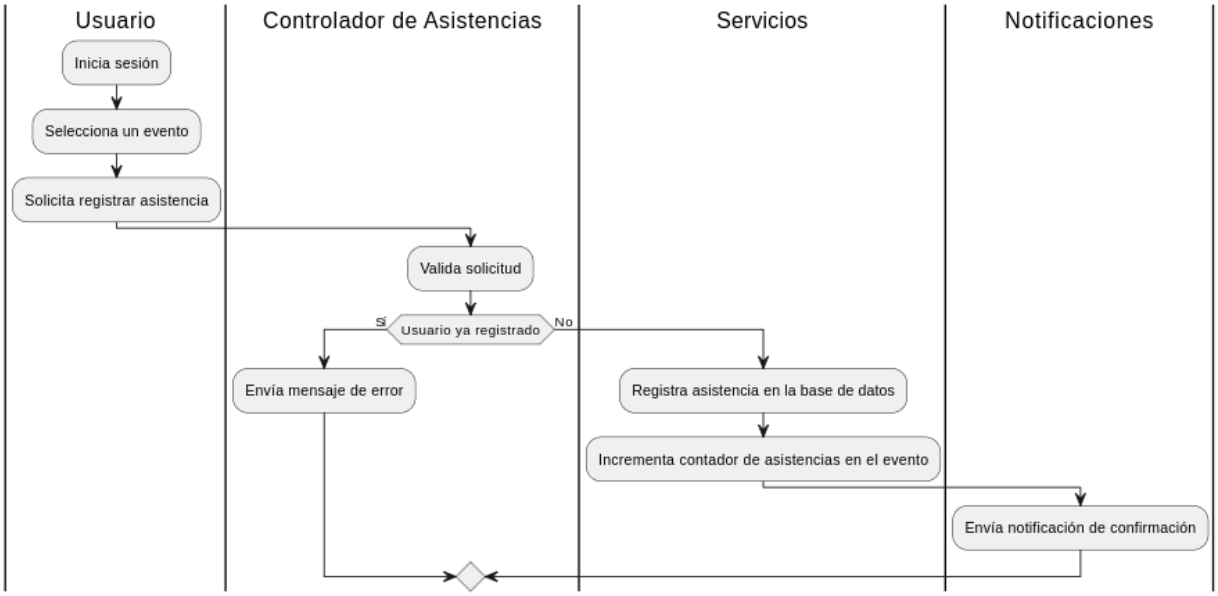


Aprobar un reporte de publicación

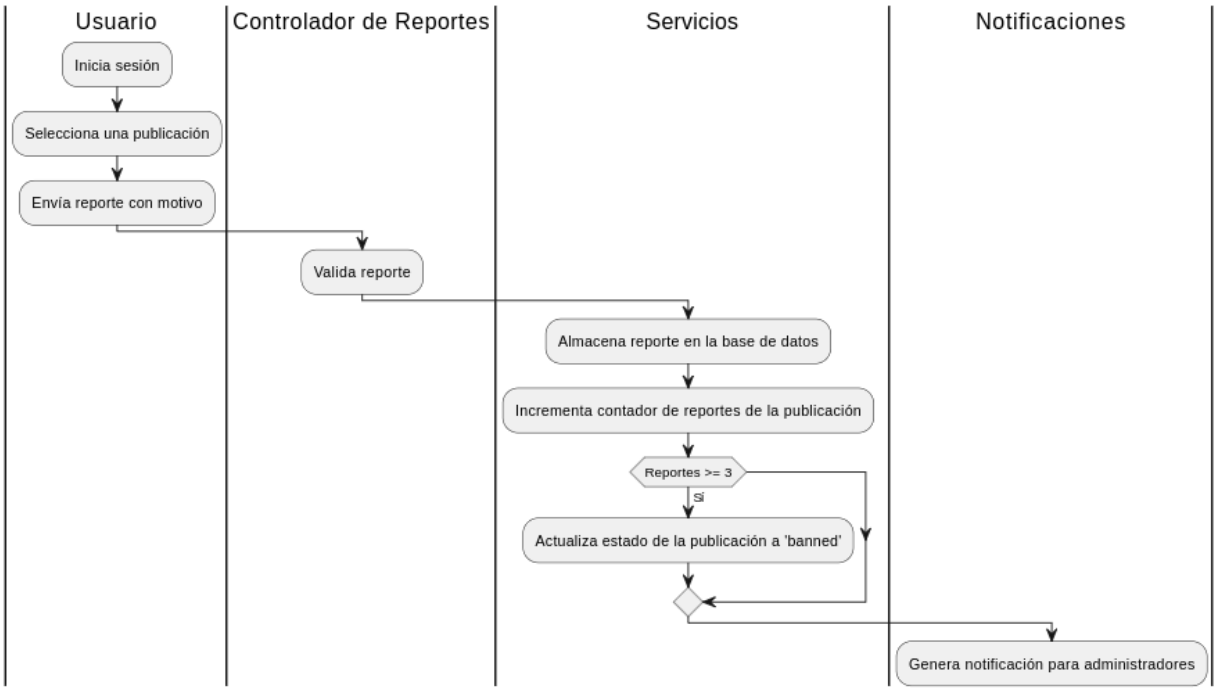


Diagramas de actividades

Registrar asistencia



Reportar publicación



Aprobar publicación

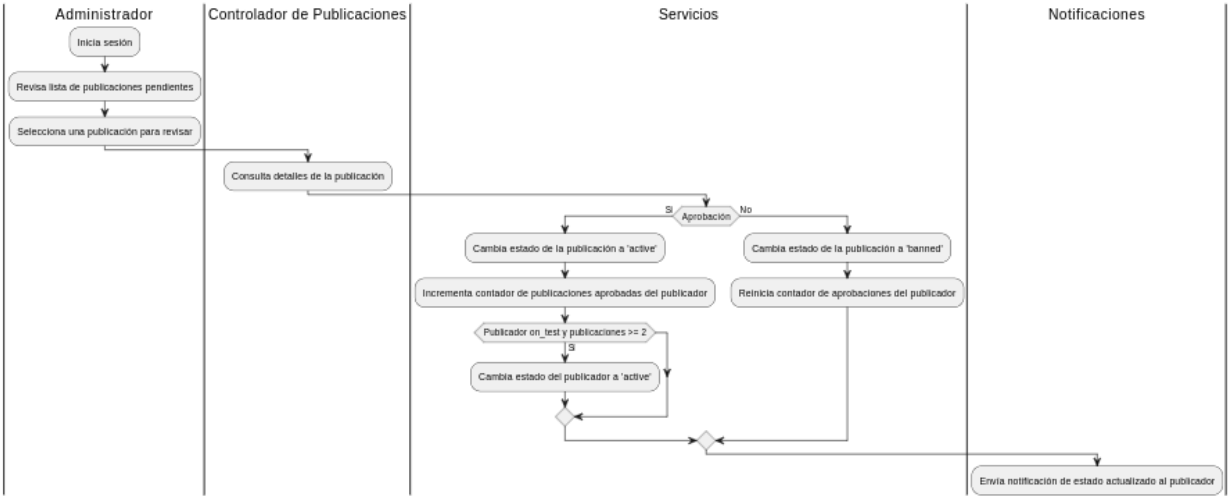


Diagrama de casos de uso

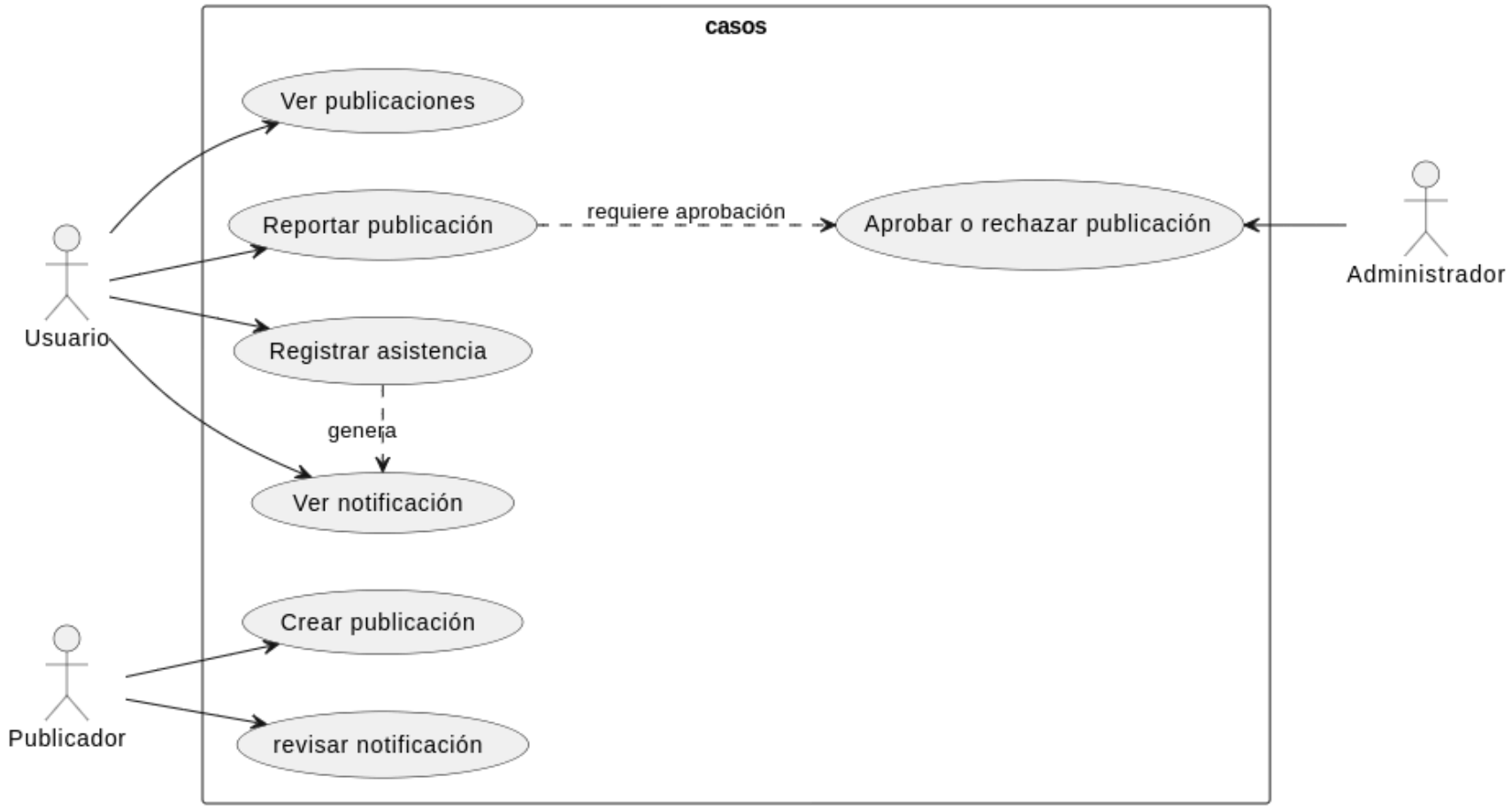


Diagrama de despliegue

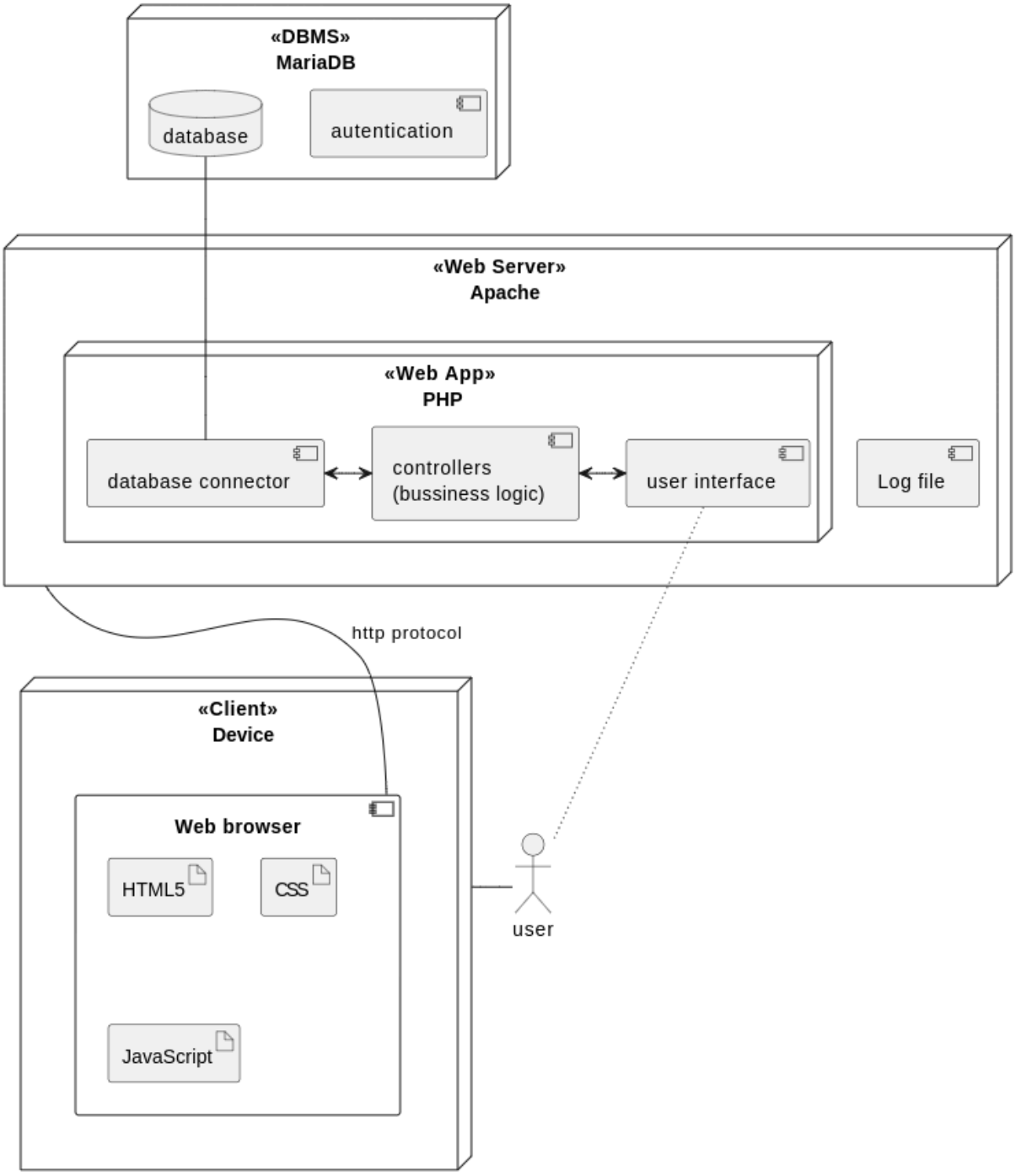


Diagrama de paquetes

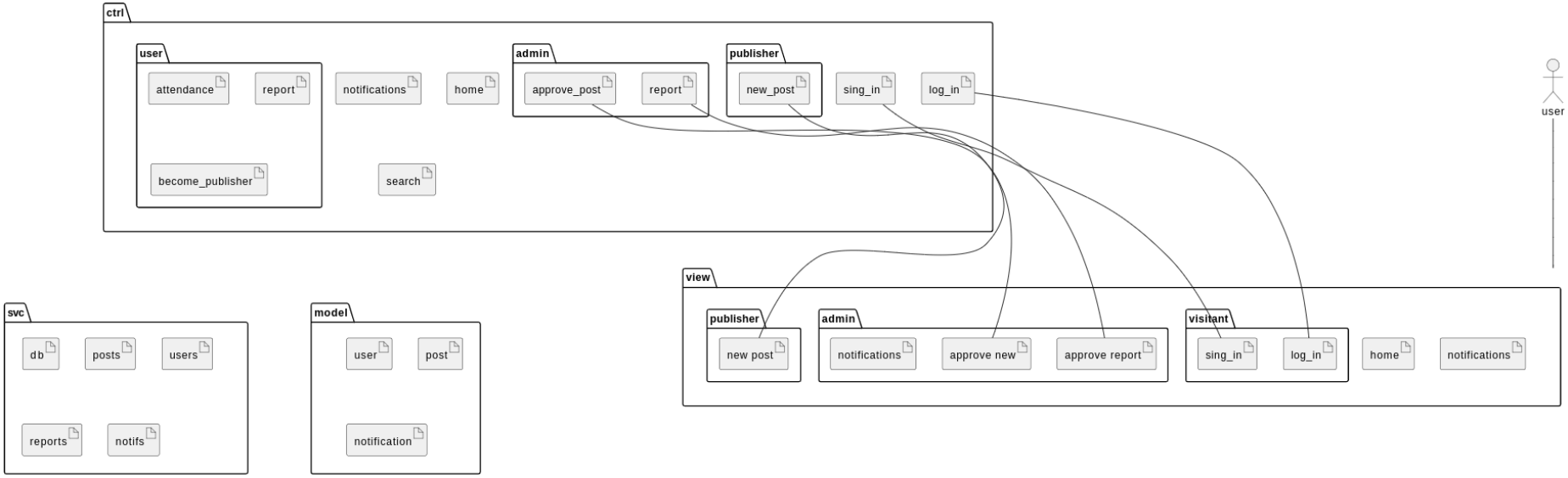


Diagrama de clases

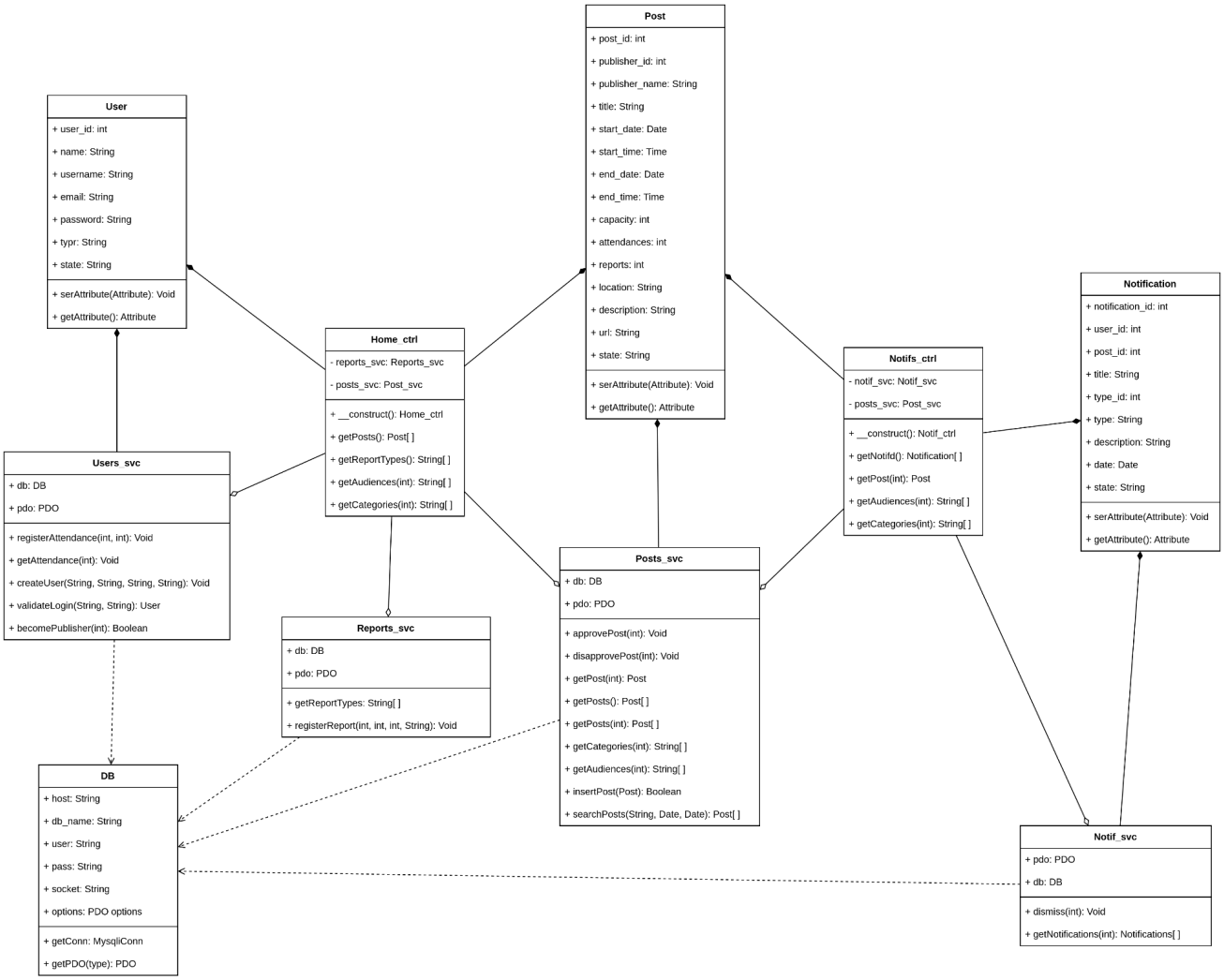
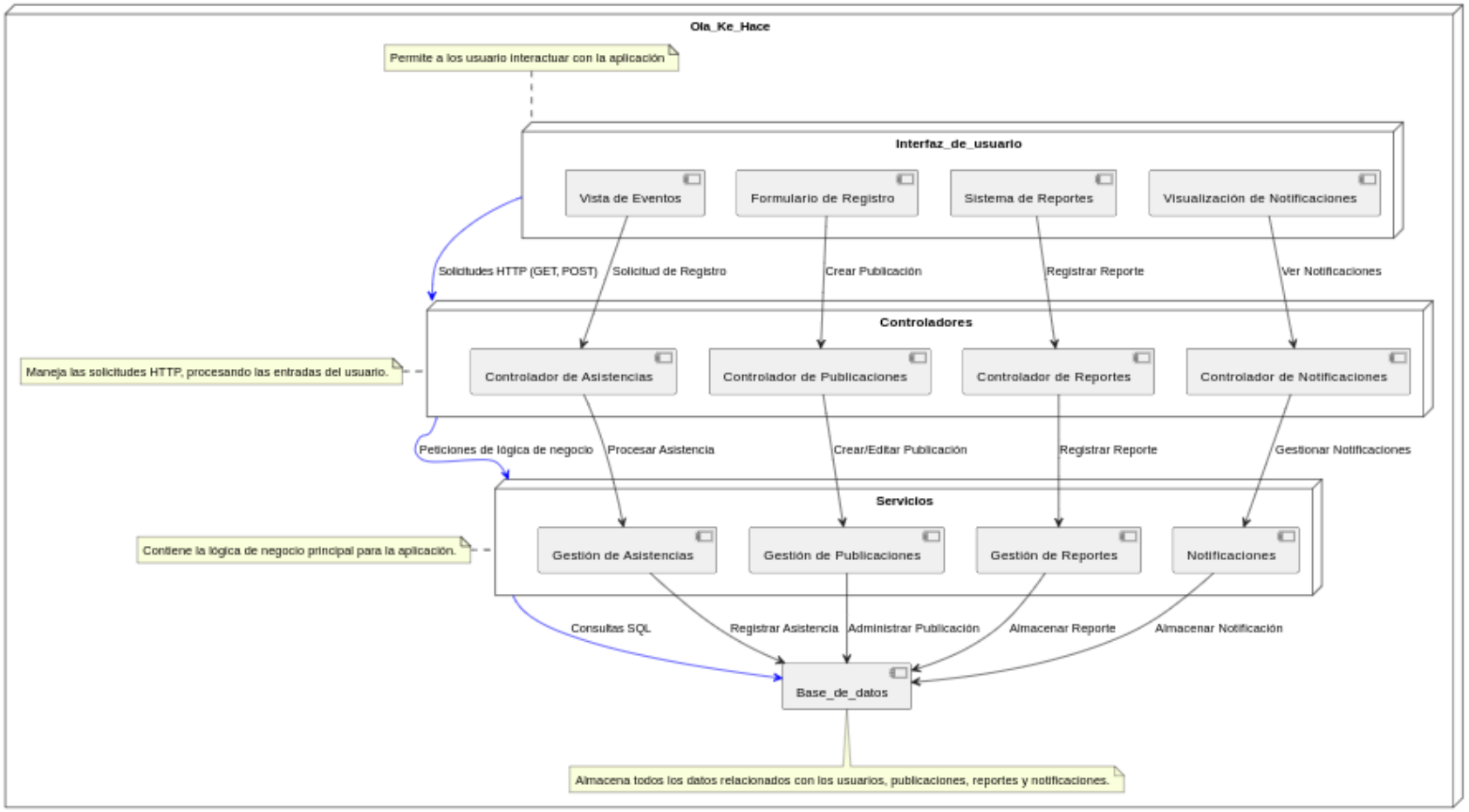


Diagrama de componentes



Base de Datos

Diagrama Entidad Relación (Peter Chen):

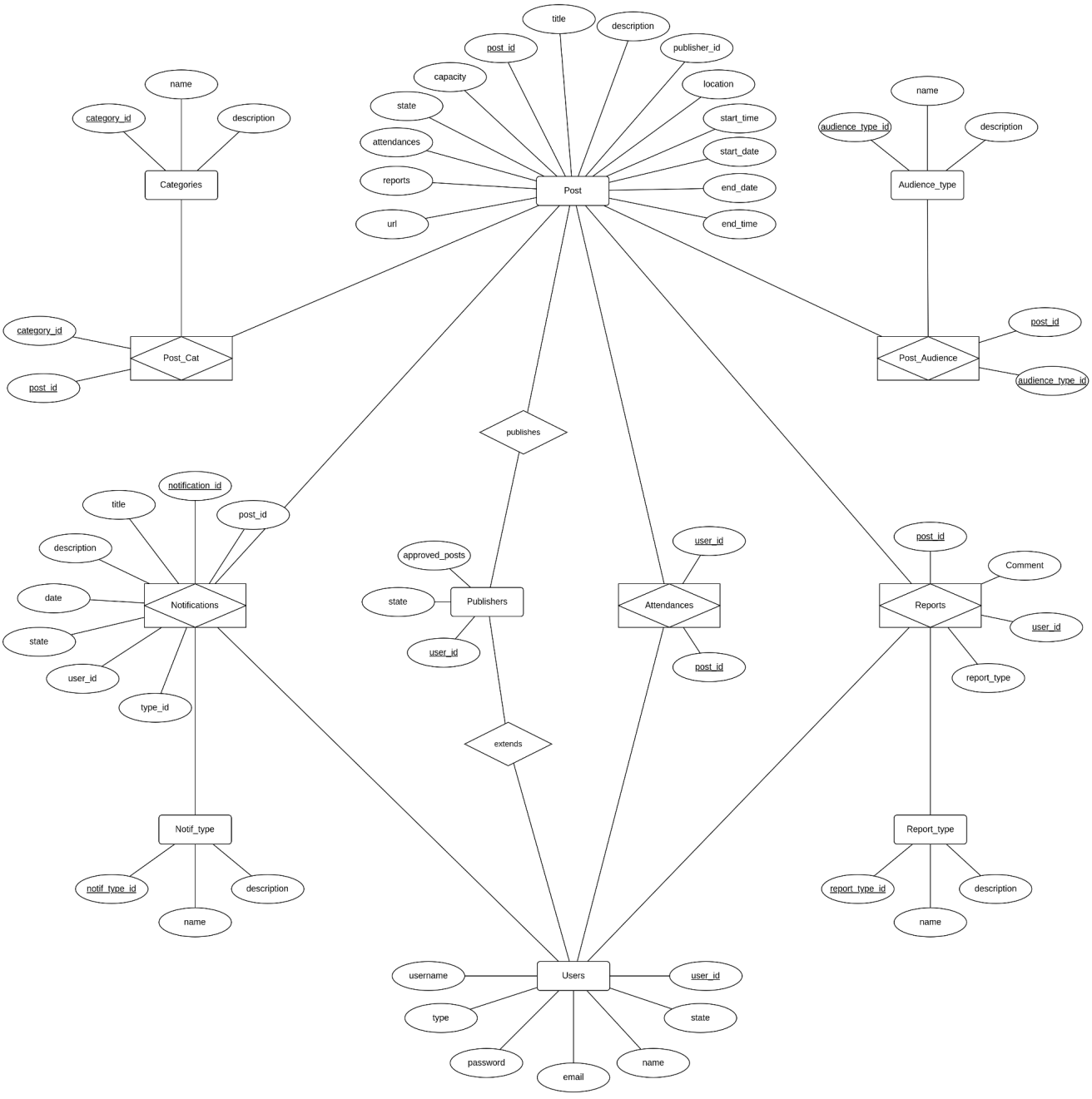
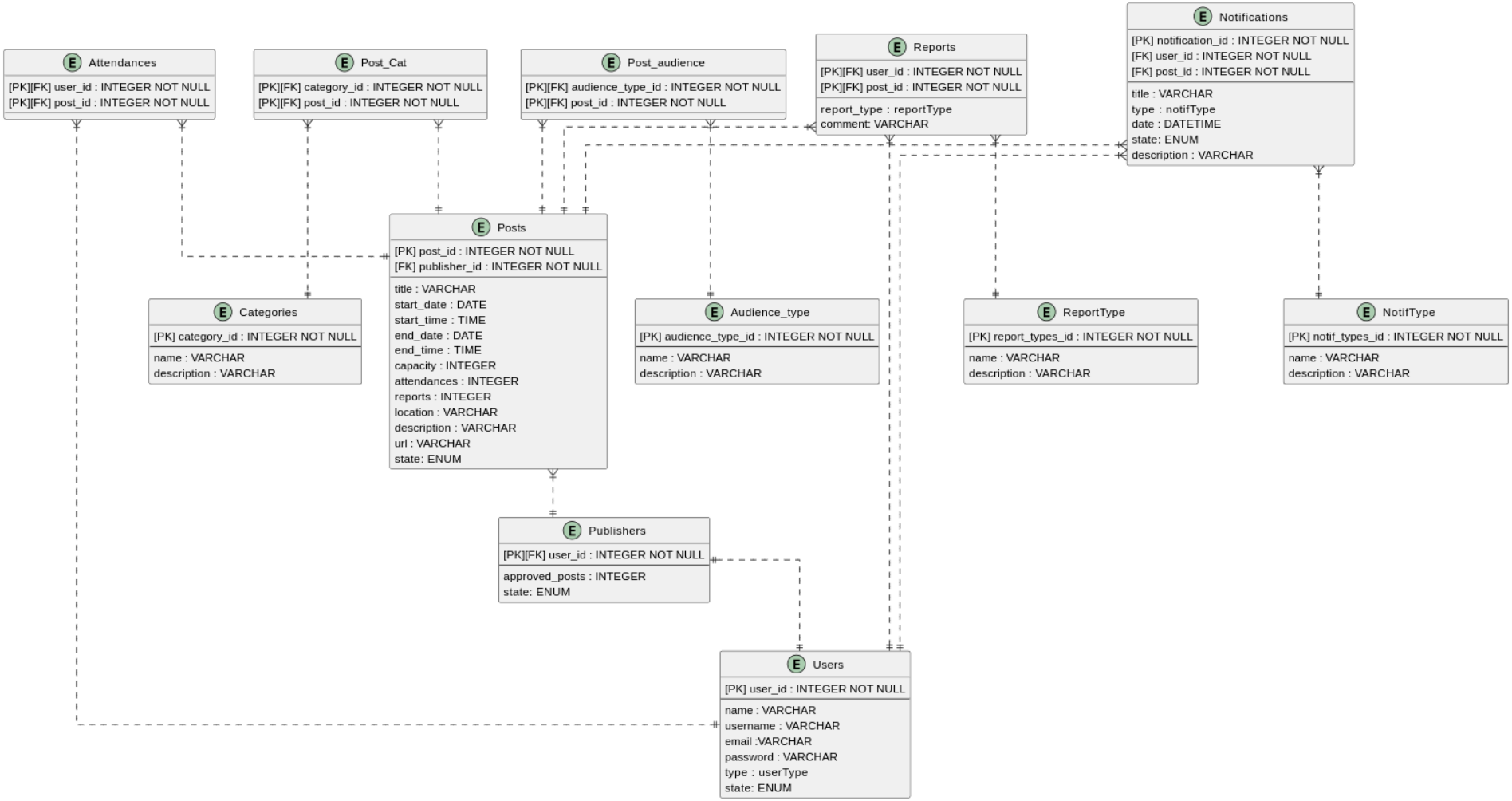


Diagrama de tablas:



Diccionario de la base de datos:

1. Tabla ***users***

Campo	**Info**	**Tipo**	**Descripción**
user_id	PRIMARY KEY	INT	Identificador único para cada usuario.
name		VARCHAR	Nombre completo del usuario
username	UNIQUE	VARCHAR	Nombre de usuario único
email		VARCHAR	Correo electrónico del usuario
password		VARCHAR	contraseña encriptada del usuario
type		ENUM	Tipo de usuario: ‘admin’, ‘publisher’, ‘user’
state		ENUM	Estado del usuario: ‘active’, ‘inactive’

2. Tabla ***publishers***

Campo	**Info**	**Tipo**	**Descripción**
user_id	PRIMARY KEY FOREIGN KEY	INT	Identificador único para cada usuario. referencia a tabla *users*
approved_posts	Default = 0	INT	Contador de posts aprobados por un admin consecutivos
state		ENUM	Estado del publicador: ‘active’, ‘on_test’, ‘banned’

3. Tabla ***posts***

Campo	**Info**	**Tipo**	**Descripción**
post_id	PRIMARY KEY	INT	Identificador único de la publicación
publisher_id	FOREIGN KEY	INT	Identificador del publicador (referencia a *publishers*).
title		VARCHAR	Título del evento
start_date		DATE	Fecha de inicio del evento
start_time		TIME	Hora de inicio del evento
end_date		DATE	Fecha del fin del evento
end_time		TIME	Hora del fin del evento
capacity		INT	Capacidad de asistentes permitidos
attendances	Default = 0	INT	Contador de usuarios que han registrado su asistencia al evento
reports	Default = 0	INT	Contador de reportes que se han registrado para la publicación
location		VARCHAR	Ubicación del evento
description		VARCHAR	Descripción del evento
url		VARCHAR	Url relacionada al evento
state		ENUM	Estado del post: ‘active’, ‘pending’, ‘banned’

4. Tabla ***categories***

Campo	**Info**	**Tipo**	**Descripción**
category_id	PRIMARY KEY	INT	Identificador único de una categoría
name		VARCHAR	Nombre de la categoría
description		VARCHAR	Descripción de la categoría

5. Tabla ***post_cat***

Campo	**Info**	**Tipo**	**Descripción**
category_id	PRIMARY KEY FOREIGN KEY	INT	Identificador de la categoría (referencia a *categories*).
post_id	PRIMARY KEY FOREIGN KEY	INT	Identificador del post (referencia a *posts*).

6. Tabla ***audience_types***

Campo	**Info**	**Tipo**	**Descripción**
audience_type_id	PRIMARY KEY	INT	Identificador único del tipo de público
name		VARCHAR	Nombre del tipo de audiencia
description		VARCHAR	Descripción del tipo de audiencia

7. Tabla ***post_audience***

Campo	**Info**	**Tipo**	**Descripción**
audience_type_id	PRIMARY KEY FOREIGN KEY	INT	Identificador del tipo de audiencia (referencia a *audience_types*).
post_id	PRIMARY KEY FOREIGN KEY	INT	Identificador del post (referencia a *posts*).

8. Tabla ***attendances***

Campo	**Info**	**Tipo**	**Descripción**
user_id	PRIMARY KEY FOREIGN KEY	INT	Identificador de la asistencia (referencia a *users*).
post_id	PRIMARY KEY FOREIGN KEY	INT	Identificador del post (referencia a *posts*).

9. Tabla ***report_types***

Campo	**Info**	**Tipo**	**Descripción**
report_type_id	PRIMARY KEY	INT	Identificador único del tipo de reporte
name		VARCHAR	Nombre del tipo de reporte
description		VARCHAR	Descripción del tipo de reporte

10. Tabla ***notif_types***

Campo	**Info**	**Tipo**	**Descripción**
notif_type_id	PRIMARY KEY	INT	Identificador único del tipo de notificación
name		VARCHAR	Nombre del tipo de notificación

description		VARCHAR	Descripción del tipo de notificación

11. Tabla ***reports***

Campo	**Info**	**Tipo**	**Descripción**
user_id	PRIMARY KEY FOREIGN KEY	INT	Identificador de la asistencia (referencia a *users*).
post_id	PRIMARY KEY FOREIGN KEY	INT	Identificador del post (referencia a *posts*).
report_type	FOREIGN KEY	INT	Tipo de reporte (referencia a *report_types*).
comment		VARCHAR	Comentario del usuario sobre el reporte

12. Tabla ***notifications***

Campo	**Info**	**Tipo**	**Descripción**
notification_id	PRIMARY KEY	INT	Identificador de la notifición
user_id	FOREIGN KEY	INT	Identificador del usuario ql que va dirigida (referencia a *users*).
post_id	FOREIGN KEY	INT	Identificador del post relacionado (referencia a *posts*).
report_type	FOREIGN KEY	INT	Tipo de notificación (referencia a *report_types*).
title		VARCHAR	Título de la notificación
description		VARCHAR	Descripción de la notificación
date		DATE	Fecha en que se genera la notificación
state		ENUM	Estado de la notificación ‘active’, ‘dismissed’

Funciones SQL

1. Vista ***getPosts***
   Propósito:
   La vista *getPosts* permite obtener un listado de los posts activos más recientes, junto con el nombre del publicador. Esto es útil para mostrar a los usuarios los posts actuales en la aplicación.
   Lógica:
   a. Selecciona todas las columnas de la tabla posts y el nombre del publicador desde la tabla *users*.
   b. Filtra los posts cuyo estado sea 'active'.
   c. Ordena los resultados por el *post_id* en orden descendente (desde los posts más recientes).
   d. Limita el resultado a los últimos 15 posts.
   Devuelve:
   Un conjunto de filas que incluye todos los datos del post (*post_id, publisher_id, title, start_date*, etc.) y el nombre del publicador (*publisher_name*), mostrando solo las publicaciones activas y recientes.

2. Trigger ***trigger_increment_attendances***
   Propósito:
   Este trigger se utiliza para aumentar el contador de asistencias (*attendances*) en la tabla posts cuando se inserta un nuevo registro de asistencia en la tabla *attendances*. Además, crea una notificación para el usuario, confirmando su registro de asistencia al evento.
   Lógica:
   a. Después de cada inserción en la tabla *attendances*, aumenta el valor del contador *attendances* en la tabla posts para el *post_id* correspondiente.
   b. Inserta una nueva notificación en la tabla *notifications*, asociando el *post_id* y el *user_id* del evento. El tipo de notificación es de asistencia registrada (*type_id* = 4), con un título y descripción específicos.
   Devuelve:
   No devuelve un valor directo, pero actualiza la tabla posts incrementando el contador de asistencias, y genera una notificación para el usuario en la tabla notifications

3. Trigger ***trigger_decrement_attendances***
   Propósito:
   Reduce el contador de asistencias (*attendances*) en la tabla posts cuando se elimina un registro de asistencia de la tabla *attendances*. Esto mantiene el contador en posts actualizado.
   Lógica:

a. Después de eliminar un registro en la tabla attendances, el trigger reduce en uno el valor de attendances en la tabla posts para el *post_id* afectado.

Devuelve:

No devuelve un valor, pero actualiza la tabla *posts* decrementando el contador de asistencias del evento respectivo cuando un usuario retira su asistencia.

4. Trigger ***after_publisher_insert***

   Propósito:

   Actualiza automáticamente el tipo de usuario a 'publisher' en la tabla *users* cuando se inserta un nuevo registro en la tabla *publishers*. Esto para asegurar que cada publicador tenga el tipo de usuario adecuado.

   Lógica:

   a. Tras la inserción en la tabla *publishers*, el trigger actualiza el campo type del usuario correspondiente en la tabla users, asignándole el valor 'publisher' basado en el *user_id*.

   Devuelve:

   No tiene una salida visible, pero asegura que el campo type de *users* esté alineado con el rol del usuario en la aplicación, asignando automáticamente el rol de 'publisher'.

5. Trigger ***after_report_insert***

   Propósito:

   Este trigger gestiona los reportes de publicaciones. Cuando un usuario reporta un post, el trigger incrementa el contador de reportes para dicho post. Si un post recibe 3 o más reportes, su estado se cambia automáticamente a 'banned'. Además, se genera una notificación para los administradores alertando del reporte.

   Lógica:

   a. Después de cada inserción en la tabla reports, se incrementa el contador de reportes (*reports*) en la tabla posts para el *post_id* correspondiente.
   b. Si el contador de reportes alcanza 3 o más, el estado del post se actualiza a 'banned'.
   c. Finalmente, se inserta una notificación en la tabla *notifications* para todos los administradores (type = 'admin'), alertándolos sobre el reporte del post.

   Devuelve:

   Actualiza la tabla *posts* incrementando el contador de reportes y, cuando corresponde, cambia el estado del post a 'banned'. Además, genera una notificación en la tabla *notifications* para los administradores.

6. Trigger ***before_post_insert***

   Propósito:

Este trigger verifica el estado del publicador antes de insertar un nuevo post. Dependiendo del estado del publicador ('active', 'on_test' o 'banned'), se determina el estado inicial del post o se bloquea la inserción si el publicador está en estado 'banned'.

Lógica:

a. Se obtiene el estado del publicador (*publisher_state*) mediante su *user_id*.
b. Si el publicador está 'banned', el trigger lanza un error y evita la inserción del post.
c. Si el estado del publicador es 'on_test', el estado del nuevo post se establece como 'pending'.
d. Si el estado del publicador es 'active', el post se marca como 'active'.

Devuelve:

No devuelve un valor directo, pero controla el estado del post antes de insertarlo y evita su inserción si el publicador está bloqueado.

7. Trigger ***after_post_insert***

Propósito:

Este trigger genera una notificación para los administradores cuando un nuevo post requiere aprobación. Se activa después de insertar un post en estado 'pending', que ocurre cuando el publicador está en período de prueba ('on_test').

Lógica:

a. Si el estado del nuevo post es 'pending', el trigger inserta una notificación en la tabla *notifications* para todos los administradores, informándoles que hay un nuevo post pendiente de revisión.

Devuelve:

No tiene una salida visible directa, pero crea una notificación en la tabla *notifications* para que los administradores revisen el post pendiente.

8. Trigger ***after_post_update***

Propósito:

Este trigger gestiona los cambios en el estado de los posts cuando un administrador los aprueba o rechaza. Al aprobar un post pendiente, aumenta el contador *approved_posts* para el publicador. Si el contador alcanza 2 o más, el estado del publicador cambia a 'active'. En caso de rechazo ('banned'), el contador de posts aprobados se reinicia a cero.

Lógica:

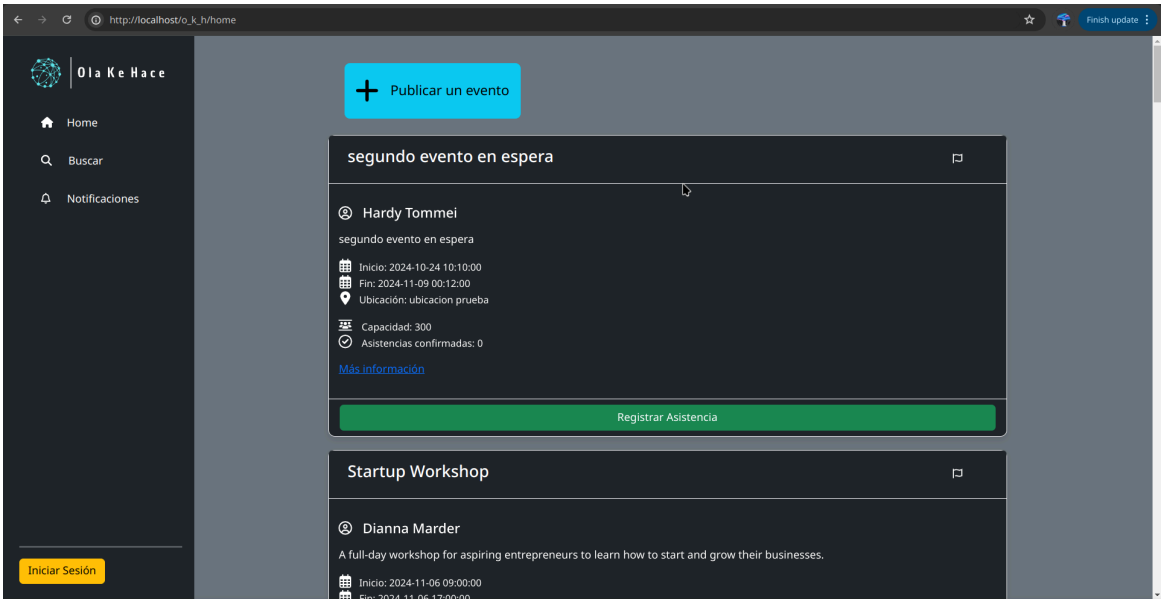
a. Si el estado cambia de 'pending' a 'active', se incrementa el contador *approved_posts* del publicador en la tabla publishers.
b. Si el contador de *approved_posts* alcanza 2 o más, el estado del publicador se actualiza a 'active'.
c. Si el estado del post cambia de 'pending' a 'banned', el contador de *approved_posts* del publicador se restablece a cero.

Devuelve:

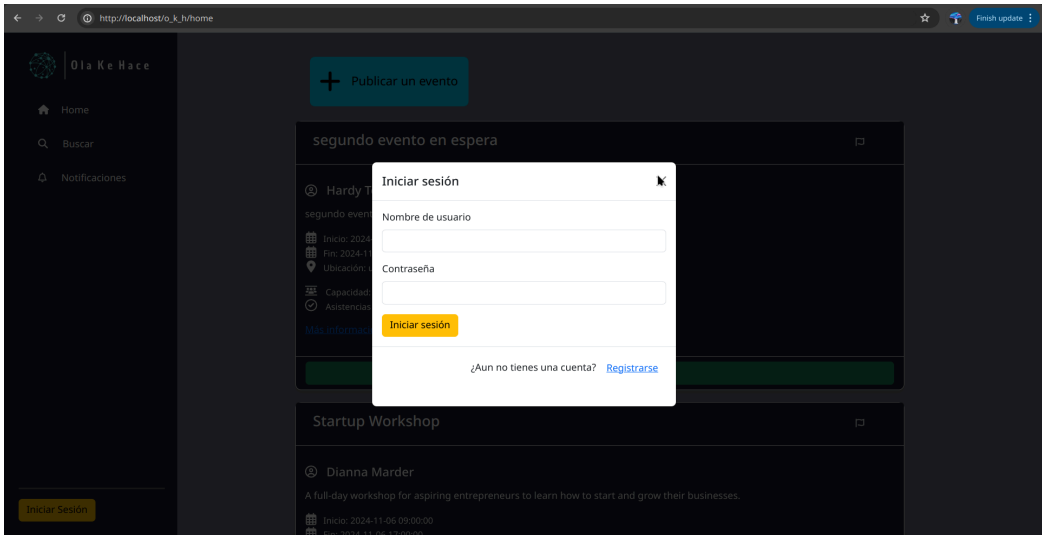
No genera una salida directa, pero mantiene los estados y contadores actualizados en las tablas *posts* y *publishers* en función de la actividad de los administradores.

Manual de usuario

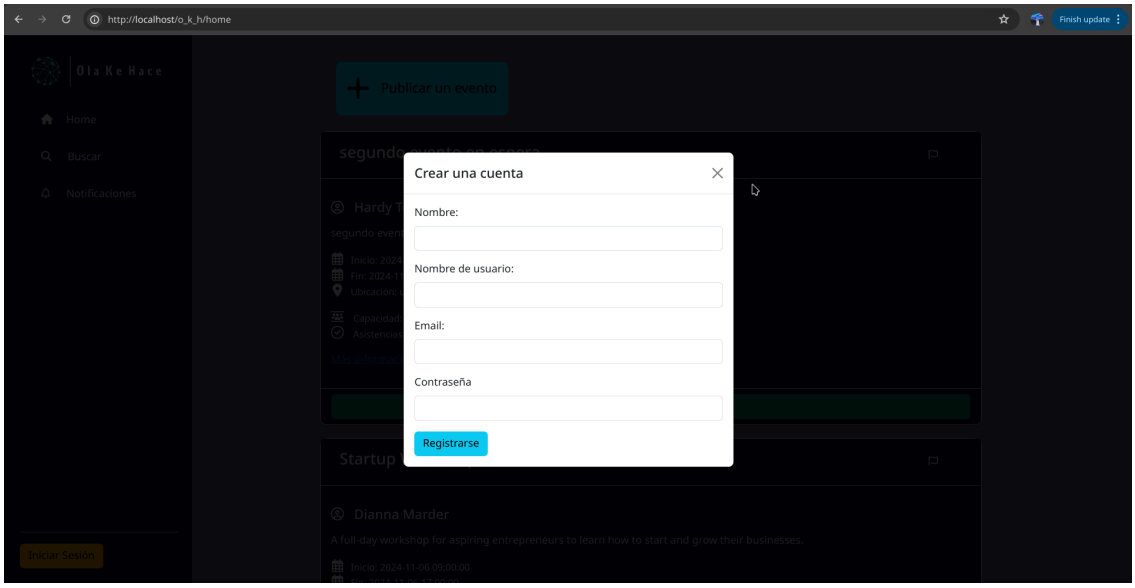
Al ejecutar la aplicación se encontrará la siguiente pantalla:



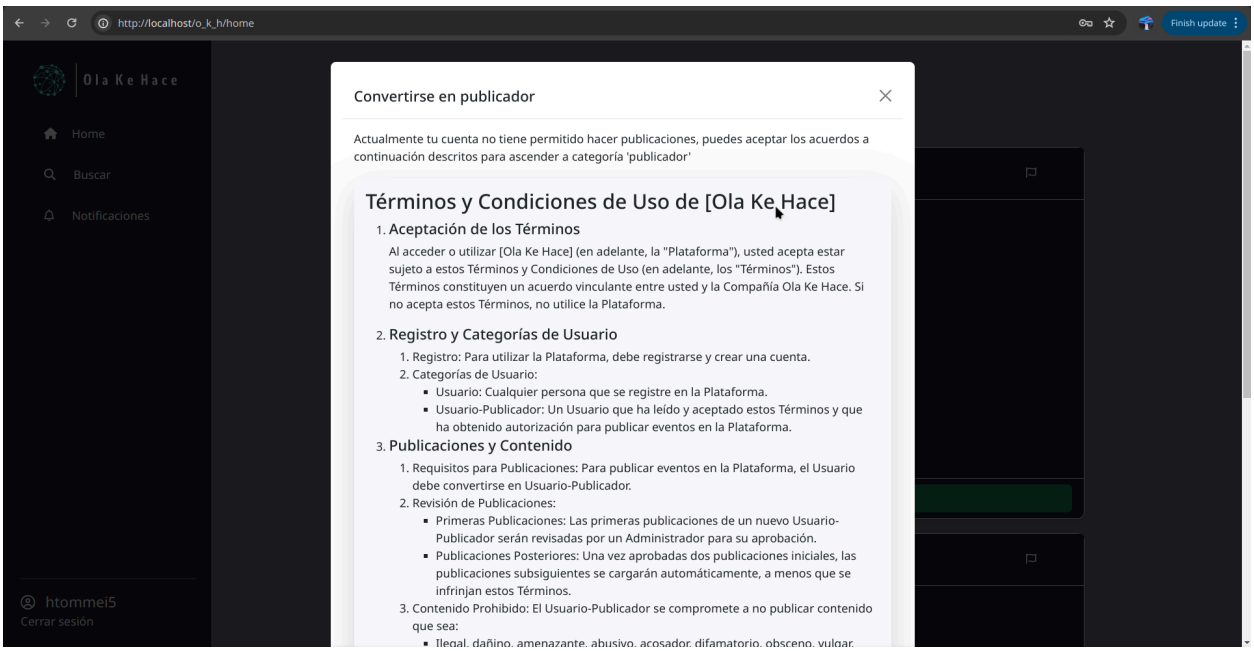
En el lado izquierdo se encuentra el menú que facilita la navegación por la aplicación. En la parte central se listan las últimas publicaciones de eventos visibles en la plataforma. El visitante puede explorar el sitio y revisar las publicaciones pero deberá iniciar sesión para realizar cualquier otra acción. Si ya tiene cuenta puede iniciar sesión ingresando sus credenciales.



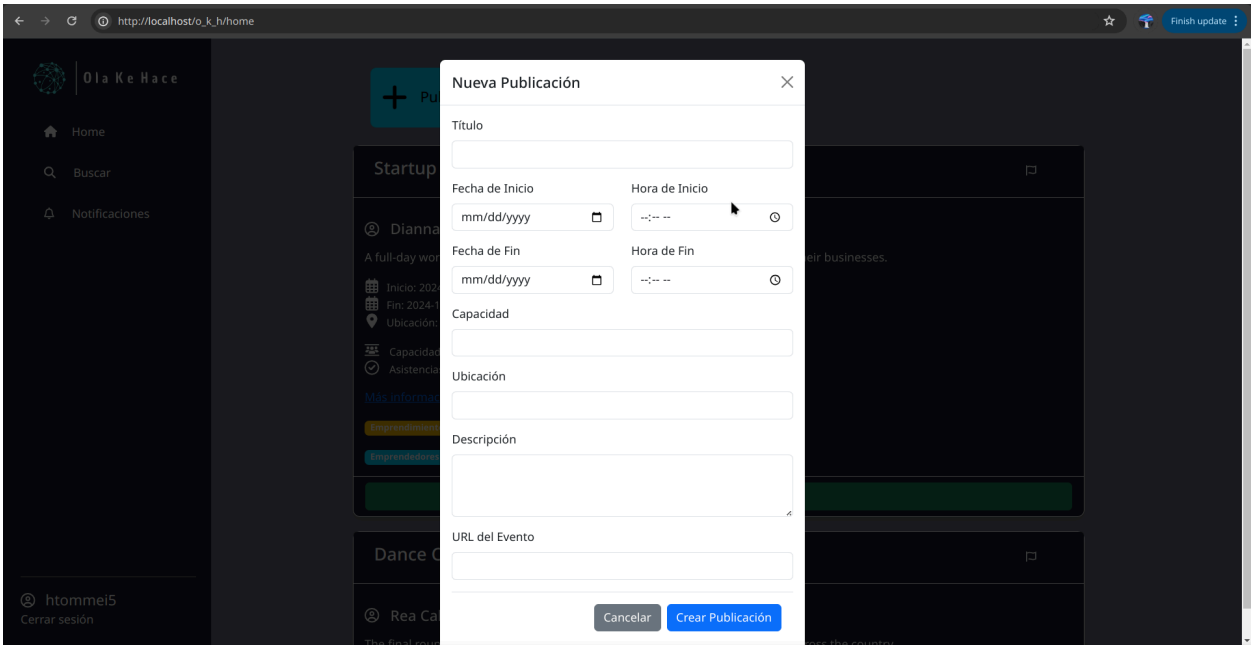
En caso de no tener cuenta puede hacer click sobre la palabra ‘Registrarse’ para crear su cuenta:



Deberá llenar los campos solicitados y si todo está en orden se registrará su usuario y se iniciará sesión.
Para hacer una publicación, el usuario debe ser de tipo publicador, en caso de no serlo deberá primero aceptar los términos y condiciones para registrarse como publicador:

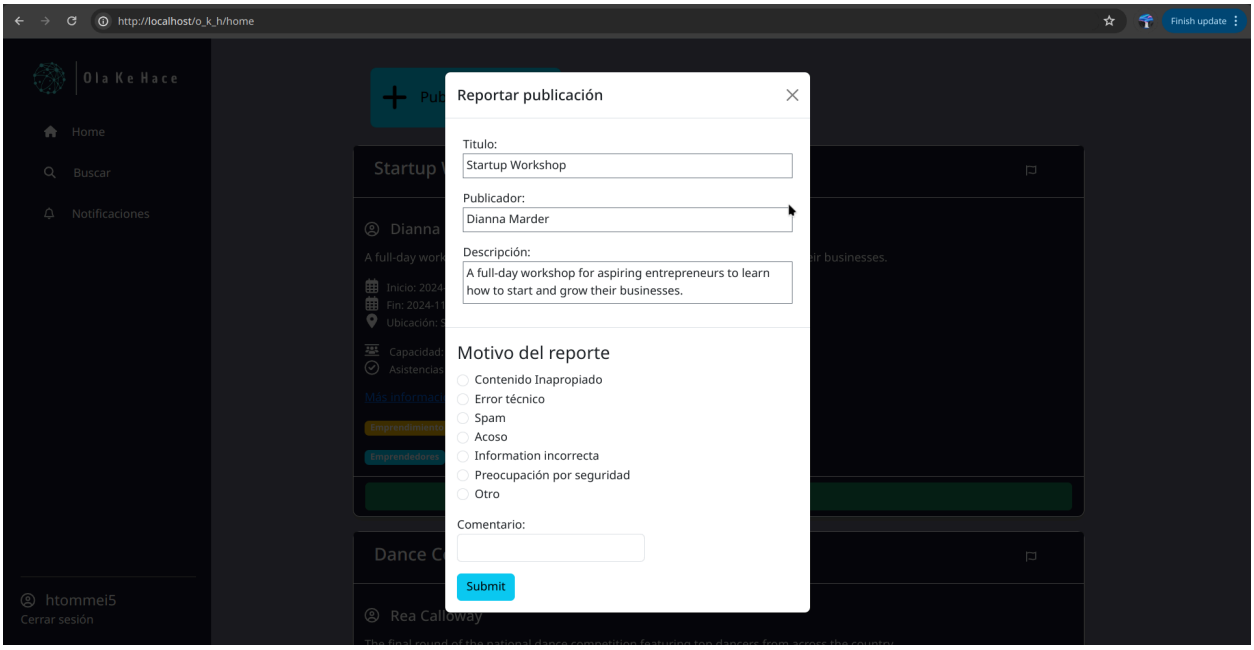


Después de ese proceso podrá hacer publicaciones llenando el siguiente formulario:

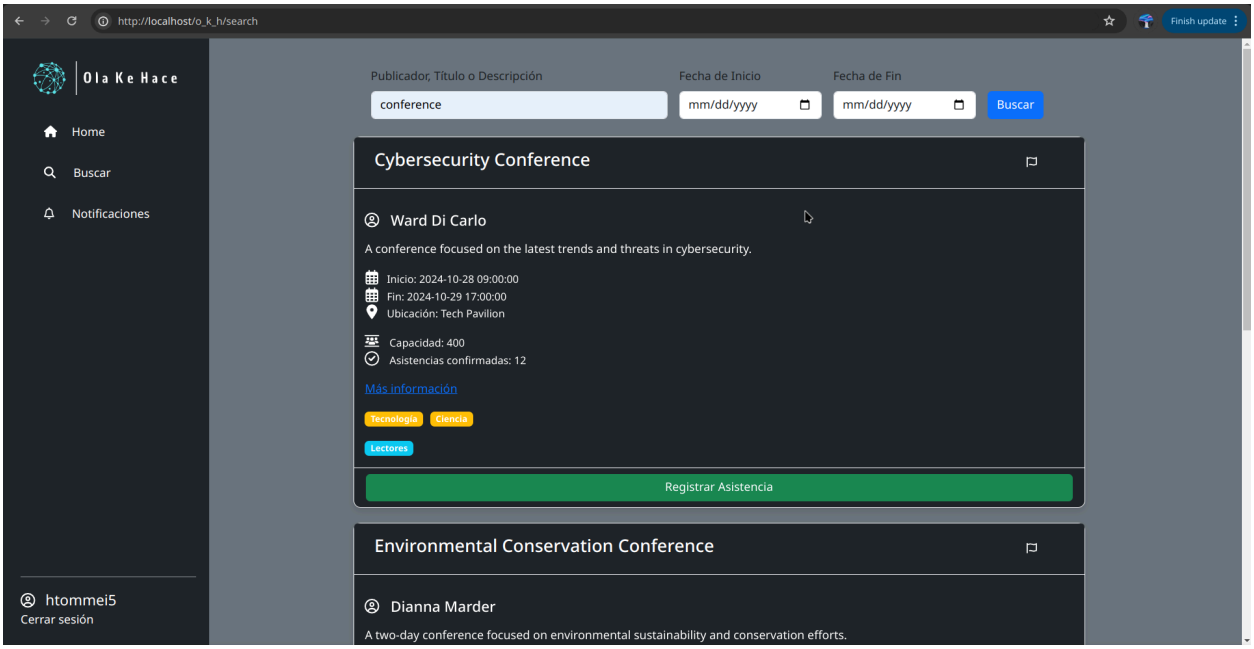


Luego aparecerá la publicación en el feed principal en caso de ser publicador activo o luego de haber sido aprobado por un administrador.

En caso de necesitar reportar una publicación se da click sobre la bandera en la esquina superior derecha de la publicación y se llena el formulario para registrar el reporte



Para realizar una búsqueda, se da click sobre ‘buscar’ en el menú lateral y se llenan los parámetros con los que se desea realizar la búsqueda, los resultados se mostrarán debajo:



Para ver las notificaciones de los eventos a los que se ha registrado asistencia se debe dirigir al apartado de notificaciones desde el menú: