Proyecto Fin de ciclo

Birbum

Autor: Michael Montaño Mejía
Curso: 2024/25

Índice

Introducción

El proyecto "Birbum" tiene como objetivo desarrollar una plataforma interactiva para propietarios de pájaros como mascotas. Esta plataforma permitirá a los usuarios compartir experiencias, consejos sobre el cuidado de sus aves y crear una comunidad en torno a la tenencia responsable de mascotas aviares. Además, se incluirán recursos educativos sobre las diferentes especies de pájaros, su comportamiento, alimentación y salud.

La plataforma está diseñada para ser multiplataforma, asegurando una experiencia fluida y responsiva en cualquier dispositivo gracias a las capacidades de diseño de Tailwind CSS y DaisyUI, tecnologías modernas como Laravel 12, JavaScript, AlpineJS y HTML5.

Objetivo

Crear una plataforma interactiva que combine recursos educativos, interacción social y funcionalidades avanzadas para propietarios de pájaros.

Alcance

La plataforma ofrecerá las siguientes funcionalidades principales:

1. Interacción Social:
    - Participación en foros temáticos para compartir experiencias y consejos.
    - Organización de eventos virtuales o presenciales, como charlas sobre el cuidado de aves, concursos de fotos o encuentros de propietarios.

2. Gestión de Eventos
    - Publicación y visualización de eventos relacionados con la comunidad de propietarios de pájaros.
    - Soporte para imágenes de portada y descripciones detalladas de los eventos.

3. Personalización y Notificaciones:
    - Modificación del perfil de usuario.
    - Recepción de notificaciones personalizadas sobre eventos y foros.

4. Soporte Multiplataforma:
    - Diseño responsivo y adaptado a dispositivos móviles, tablets y escritorios.

Este proyecto busca combinar educación, interacción social y tecnología moderna para crear una experiencia única y enriquecedora para los propietarios de pájaros, fomentando una comunidad activa y responsable.

Estado del arte

Los foros de internet han sido una parte fundamental de la comunicación online desde los primeros días de la web. Básicamente, son espacios virtuales donde la gente puede crear temas de conversación, responder mensajes e intercambiar información sobre cualquier tema imaginable. Lo que empezó como algo bastante simple se ha convertido en una parte importante de cómo nos comunicamos en internet.

Un poco de historia

Los foros de internet han sido una parte importante de cómo nos comunicamos online desde hace años. Surgieron como una evolución de los BBS (Bulletin Board Systems), permitiendo que personas con intereses comunes compartieran información. Durante los 2000s, plataformas como phpBB y vBulletin hicieron que los foros fueran más populares, añadiendo cosas como avatares y sistemas de reputación.

Aunque las redes sociales han reemplazado a muchos foros, los especializados, como los dedicados a pájaros, siguen siendo útiles porque ofrecen comunidades enfocadas y conocimiento profundo. Birbum toma inspiración de esta historia para crear un espacio moderno y especializado para los amantes de las aves.

Competencia

Birbum se diferencia de otras plataformas existentes al combinar elementos de interacción social y gestión de eventos en una comunidad enfocada en los propietarios de pájaros. Aunque existen foros y redes sociales como Discord (https://discord.com/) que permiten la interacción entre usuarios, Birbum se especializa en un nicho específico, ofreciendo funcionalidades adaptadas a las necesidades de los amantes de las aves.

Innovación

Birbum propone una experiencia única al integrar foros temáticos, gestión de eventos y personalización de perfiles en una sola plataforma. Además, se enfoca en fomentar una comunidad activa y responsable, ofreciendo herramientas para la organización de eventos y la participación en discusiones relevantes. La combinación de estas características asegura una cohesión efectiva y una experiencia enriquecedora para los usuarios.

Estudio de viabilidad. Método DAFO

a. Análisis DAFO

Debilidades

- Las redes sociales como Instagram y TikTok son más llamativas para los jóvenes.
- Aprender a usar los foros puede ser complicado al principio.
- Mantener la plataforma requiere tiempo y conocimientos técnicos.
- Hay que pagar por el servidor y el dominio todos los meses.
- Sin moderadores activos, la calidad del foro puede bajar mucho.
- Al principio, cuesta que los usuarios creen contenido suficiente.

Amenazas

- Muchas comunidades se están yendo a Discord o Telegram.
- Los jóvenes prefieren plataformas más visuales.
- Las leyes como el RGPD hacen que sea más difícil manejar datos.
- Los grupos de Facebook son fáciles de crear y gestionar.
- Si hay una crisis económica, menos gente invierte en hobbies.
- El spam y los trolls pueden arruinar la experiencia de los usuarios.

Fortalezas

- Es un foro especializado para criadores de pájaros, algo que no es común.
- Los usuarios que son aficionados valoran mucho el contenido técnico.
- Los foros organizan mejor la información que otras plataformas.
- Se pueden organizar eventos como concursos y exposiciones.
- El contenido no se pierde como en las redes sociales.
- Tenemos más control que si usáramos plataformas externas.
- Hay formas de ganar dinero, como publicidad o membresías premium.

Oportunidades

- Hay pocos foros modernos de pájaros en español.
- Se pueden hacer colaboraciones con tiendas del sector.
- Organizar eventos presenciales como concursos.
- Ofrecer contenido premium como cursos, guías especializadas, suscripciones premium.

b. Estudio de Mercado

Análisis de la Competencia

Competencia Directa:

- ForoAviario.com: Activo pero con diseño anticuado
- AvesExoticas.org: Menos tráfico, centrado en especies específicas
- Grupos de Facebook: Muchos pero desorganizados

Competencia Indirecta:

- Servidores Discord de criadores
- Canales de YouTube especializados
- Grupos de WhatsApp/Telegram locales

Público Objetivo

Segmento Principal (80% del mercado):	Segmento Secundario (20% del mercado):
● Criadores y dueños de aves ● Edad: Variable	● Gente curiosa ● Edad: Variable

c. Viabilidad Técnica/Económica del Proyecto

i. Recursos HW

Yo tengo un servidor de 4 vCPU, 8GB RAM, 160GB SSD

El coste mensual más o menos sería de €27.5

ii. Recursos SW

El proyecto Birbum utiliza las siguientes tecnologías y paquetes de software, organizados por categorías:

Frontend

- HTML y CSS: Lenguajes base para la estructura y el diseño de las páginas web.
- Tailwind CSS: Framework de CSS para un diseño rápido y eficiente.
- DaisyUI: Extensión de Tailwind CSS que proporciona componentes predefinidos y personalizables para acelerar el desarrollo.
- JavaScript: Lenguaje de programación que permite implementar funcionalidades complejas en las páginas web.
- AlpineJS: Framework de JavaScript para crear interfaces de usuario interactivas y reactivas.
- Vite: Herramienta de desarrollo frontend para construir y servir activos de manera rápida y eficiente.

Backend

PHP 8.3.6: Lenguaje de programación de código abierto utilizado para el desarrollo web del lado del servidor.

- Laravel: El núcleo del framework Laravel, versión 12.0.
- Laravel Tinker: Proporciona una consola interactiva para interactuar con la aplicación, útil para pruebas rápidas y depuración.
- Laravel Breeze: Un paquete para la autenticación simple y rápida, ideal para proyectos que necesitan un sistema de inicio de sesión básico.
- Pusher: Utilizado para la integración con Pusher, permitiendo notificaciones y actualizaciones en tiempo real en la aplicación.
- Servidor de correos de Google (Gmail SMTP): Utilizado para el envío de correos electrónicos, incluyendo la funcionalidad de verificación de correo electrónico en la aplicación.

Herramientas

- Git: Herramienta para el control de versiones.
- DBeaver: Herramienta para la administración de bases de datos.

- Composer 2.8.8: Gestor de dependencias para PHP que permite instalar y gestionar bibliotecas y paquetes necesarios para el desarrollo de aplicaciones en Laravel.

Infraestructura

- MariaDB 10.11.13: Sistema de gestión de bases de datos relacional de código abierto, conocido por su rendimiento y características avanzadas.
- Nginx 1.24.0: Servidor web de código abierto que sirve contenido web y gestiona solicitudes HTTP (también HTTPS), ampliamente utilizado en aplicaciones web.
- Node.js 20.19.2 y npm 10.8.2: Node.js es un entorno de ejecución para JavaScript en el lado del servidor, y npm es el gestor de paquetes que se utiliza para instalar bibliotecas y herramientas de JavaScript, como las que pueden ser necesarias para Tailwind CSS.
- Let's Encrypt: Proveedor de certificados SSL gratuitos para asegurar las conexiones HTTPS con el certbot.
- Cloudflare: Servicio de CDN y protección contra ataques DDoS.

d. Planificación temporal o agenda de trabajo

A continuación se encuentra un poco lo que sería la planificación que se planeó en un principio para el proyecto

Semana 1: Configuración inicial y estructura básica

- Instalación de Laravel, Breeze, Tailwind CSS y DaisyUI.
- Configuración de autenticación básica y layouts iniciales (app.blade.php, guest.blade.php).
- Creación del modelo, migración y controlador de Thread.
- Implementación de rutas y vistas para listar y crear hilos.

Semana 2: Navegación, categorías y usuarios

- Configuración de la navegación principal.
- Creación del modelo, migración y controlador de Category.
- Implementación de la gestión de categorías en el panel de administración.
- Configuración inicial de la gestión de usuarios (listar, editar y eliminar).

Semana 3: Diseño, responsividad y seguidores

- Refactorización de layouts con <x-app-layout> y slots.
- Implementación del diseño responsivo y tema oscuro/claro.
- Creación del modelo y migración para Follower (seguidores).
- Implementación de la funcionalidad para seguir y dejar de seguir usuarios.

Semana 4: Gestión de eventos y comentarios

- Creación del CRUD de eventos: modelo, migración, controlador y vistas.
- Integración de eventos en la navegación.
- Creación del modelo, migración y controlador de Comment.
- Implementación de comentarios en hilos y eventos.
- Mejoras visuales en formularios y tarjetas de hilos/eventos.

Semana 5: Notificaciones y tiempo real

- Configuración de Laravel Echo y Pusher para notificaciones en tiempo real.
- Implementación de suscripciones a hilos y notificaciones personalizadas.

Semana 6: Verificación y seguridad

- Activación de la verificación de correo electrónico con Gmail SMTP.
- Revisión de políticas de acceso y protección de rutas.
- Pruebas de seguridad y corrección de errores comunes.

Semana 7: Documentación y optimización

- Redacción de la documentación del proyecto.
- Limpieza de código y análisis DAFO.
- Optimización de consultas y pruebas de rendimiento.

Análisis de requisitos

i. Texto explicativo

El proyecto Birbum es una plataforma interactiva diseñada para propietarios de pájaros, que combina funcionalidades sociales, educativas y de gestión de eventos. La plataforma está optimizada para ser accesible desde múltiples dispositivos y ofrece una experiencia de usuario fluida y moderna.

Requisitos funcionales:

Multiplataforma: La plataforma debe ser accesible desde teléfonos, tablets y escritorios, con un diseño responsivo que se adapte a un ratio de aspecto 16:9.

Breadcrumbs: Presentes en todas las vistas para facilitar la navegación y proporcionar una estructura jerárquica clara del sitio.

Gestión de usuarios: Los usuarios pueden registrarse, iniciar sesión, editar su perfil y seguir a otros usuarios.

Gestión de categorías: Los administradores pueden crear, editar y eliminar categorías para organizar los hilos del foro.

Hilos y comentarios: Los usuarios pueden crear hilos, comentar en ellos y recibir notificaciones sobre nuevas interacciones.

Eventos: Los usuarios pueden crear, visualizar y participar en eventos, con soporte para imágenes de portada y descripciones detalladas.

Notificaciones en tiempo real: Implementadas con Laravel Echo y Pusher para alertar a los usuarios sobre nuevas interacciones en hilos y eventos.

Accesibilidad: Las vistas que requieren autenticación incluyen controles de acceso explícitos para proteger la información de los usuarios.

Verificación de correo electrónico: Los usuarios deben verificar su correo electrónico para acceder a ciertas funcionalidades.

Requisitos no funcionales

Seguridad:

Protección frente a inyecciones de código (SQL, XSS) mediante validación de entradas y uso de ORM (Eloquent).

Autenticación segura con hashing de contraseñas (bcrypt) y protección CSRF en formularios.

Rendimiento:

Respuesta del servidor en menos de 1 segundo para la mayoría de las solicitudes.

Uso de caché para consultas frecuentes y optimización de las rutas.

Optimización de recursos:

Imágenes comprimidas y optimizadas para reducir tiempos de carga.

Uso de Tailwind CSS para estilos ligeros y reutilizables.

Accesibilidad

Diseño responsive para dispositivos móviles y de escritorio.

Compatibilidad con navegadores modernos y soporte para modos oscuro y claro.

Vistas implementadas:

Vista 1: Inicio

Descripción: Página principal que da la bienvenida a los usuarios y muestra un resumen de los hilos más votados y recientes.

Ruta: /

Características: Botón para crear un nuevo hilo, ranking de usuarios y hilos destacados.

Accesibilidad: Visible para todos los usuarios, autenticados o no.

Vista 2: Foros

Descripción: Página que lista todos los hilos del foro.

Ruta: /threads

Características: Búsqueda de hilos, filtros por categoría, y paginación.

Accesibilidad: Visible para todos los usuarios, autenticados o no.

Vista 3: Hilo

Descripción: Vista detallada de un hilo con sus comentarios.

Ruta: /threads/{id}

Características: Mostrar contenido del hilo, comentarios, y opciones para votar o comentar.

Accesibilidad: Visible para todos los usuarios, pero comentar requiere autenticación.

Vista 4: Eventos

Descripción: Página que lista los eventos creados por los usuarios.

Ruta: /events

Características: Listado de eventos con imágenes de portada, fechas y descripciones.

Accesibilidad: Visible para todos los usuarios, autenticados o no.

Vista 5: Evento

Descripción: Vista detallada de un evento.

Ruta: /events/{id}

Características: Mostrar detalles del evento, como descripción, fecha, y asistentes.

Accesibilidad: Visible para todos los usuarios, pero participar requiere autenticación.

Vista 6: Perfil privado

Descripción: Página del perfil del usuario.

Ruta: /profile

Características: Editar información personal

Accesibilidad: Solo visible para usuarios autenticados.

Vista 7: Perfil público

Descripción: Página del perfil del usuario.

Ruta: /users/{user}

Características: Muestra información pública del usuario, como hilos creados y seguidores.

Accesibilidad: Visible para todos los usuarios, autenticados o no.

Vista 8: Seguidores

Descripción: Página que muestra la lista de seguidores de un usuario.

Ruta: /users/{user}/followers

Características: Listado de usuarios que siguen al perfil consultado, con enlaces a sus perfiles.

Accesibilidad: Visible para todos los usuarios, autenticados o no.

Vista 9: Administración

Descripción: Panel de administración para gestionar usuarios, categorías, eventos y el banner informativo.

Ruta: /admin

Características: Crear, editar y eliminar usuarios, categorías y eventos.

Accesibilidad: Solo visible para administradores autenticados.

Vista 10: Login

Descripción: Permite a los usuarios iniciar sesión en la plataforma.

Ruta: /login

Características: Enviar formulario con credenciales para inicio de sesión

Accesibilidad: Solo visible para usuarios no autenticados.

Vista 11: Registro

Descripción: Permite a los usuarios crear una nueva cuenta en la plataforma.

Ruta: /register

Características: Enviar formulario con credenciales para registro con verificación mediante email

Accesibilidad: Solo visible para usuarios no autenticados.

Vista 12: Verificación de Email

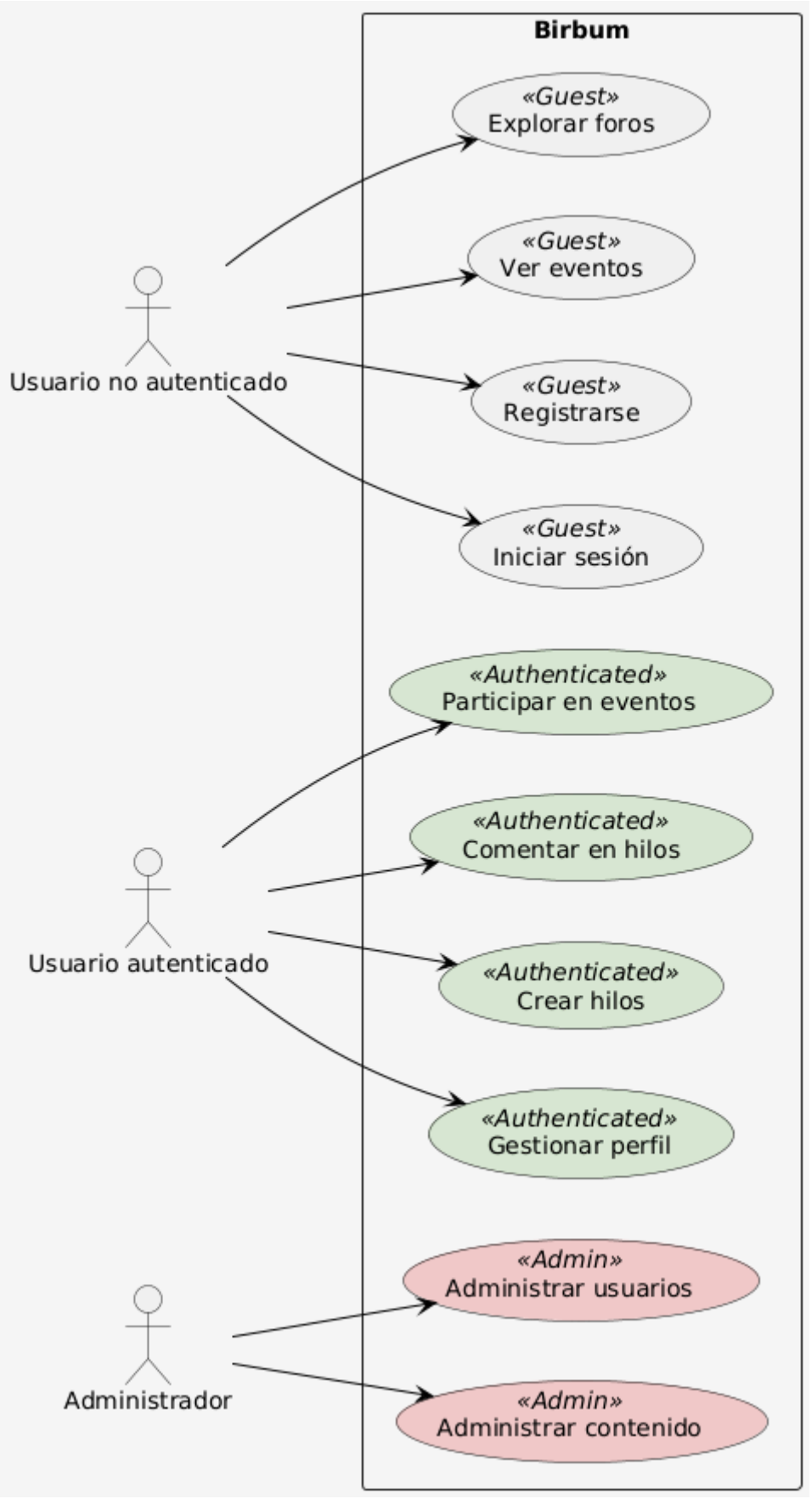
Descripción: Solicita al usuario verificar su dirección de correo electrónico tras el registro.

Ruta: /verify-email

Características: Envio de correo de verificación mediante gmail

Accesibilidad: Visible solo para usuarios autenticados que no hayan verificado su email.

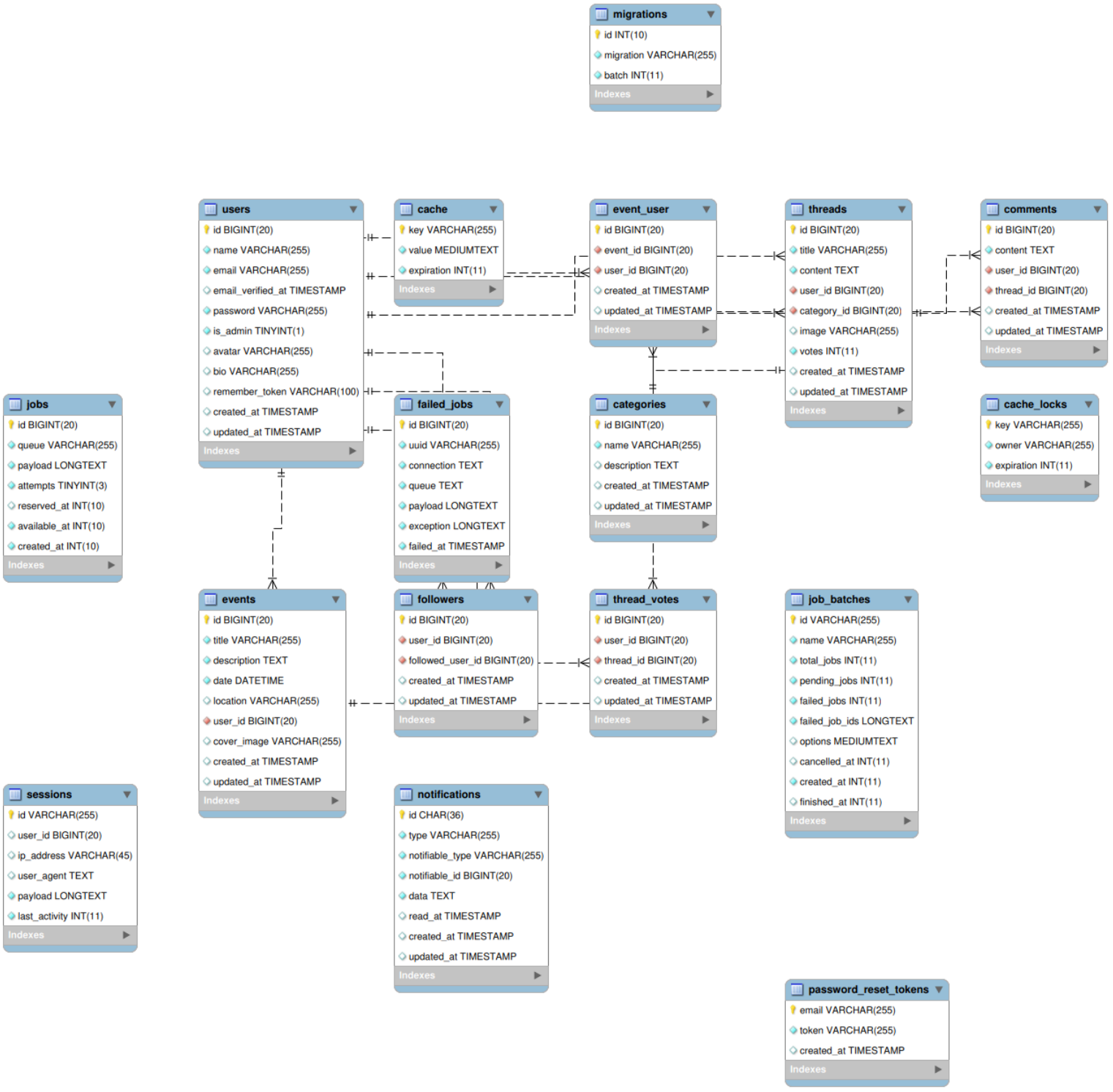
ii. Diagrama de casos de uso relevante



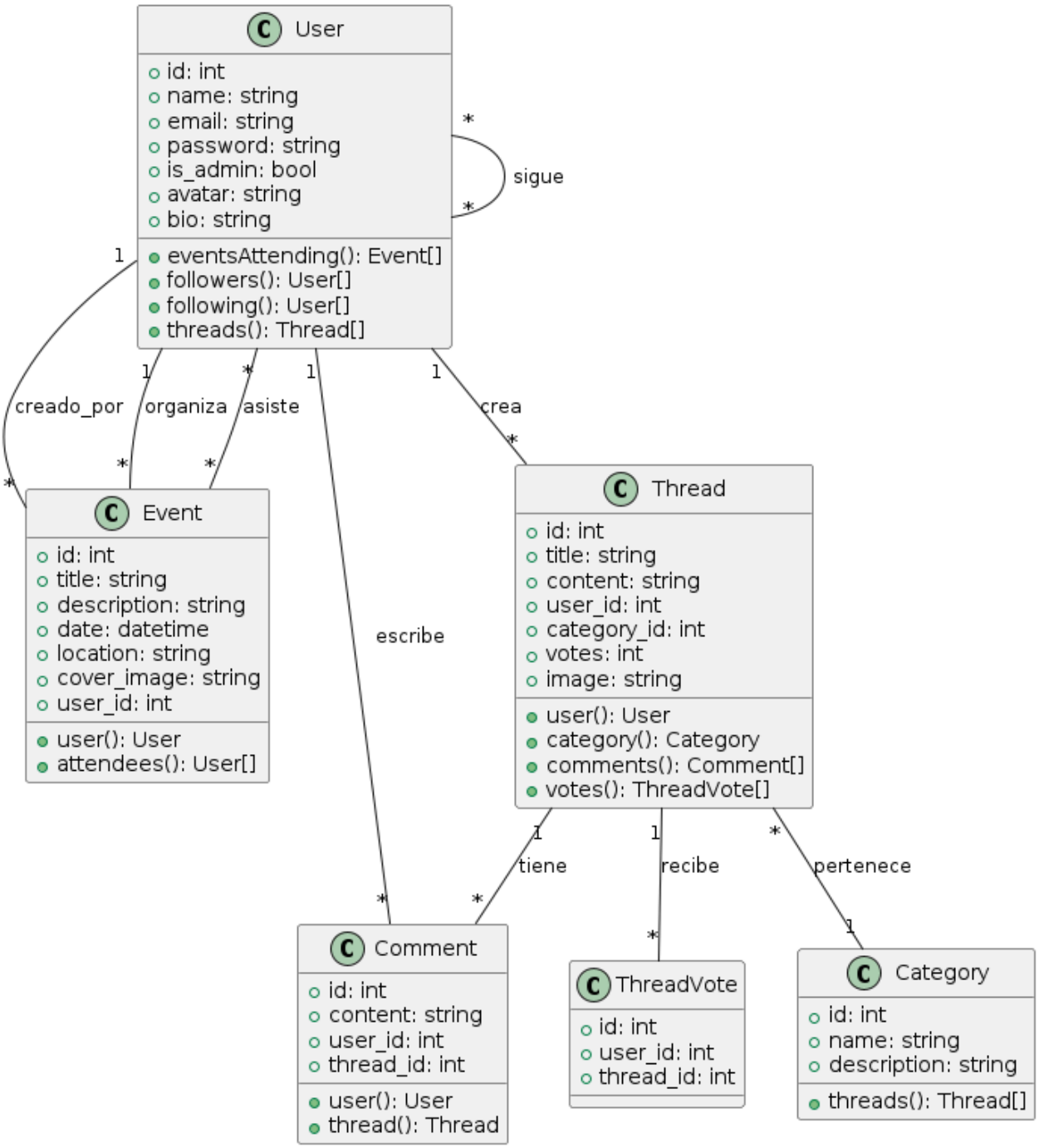
Diseño

a. Diseño Conceptual Entidad Relación

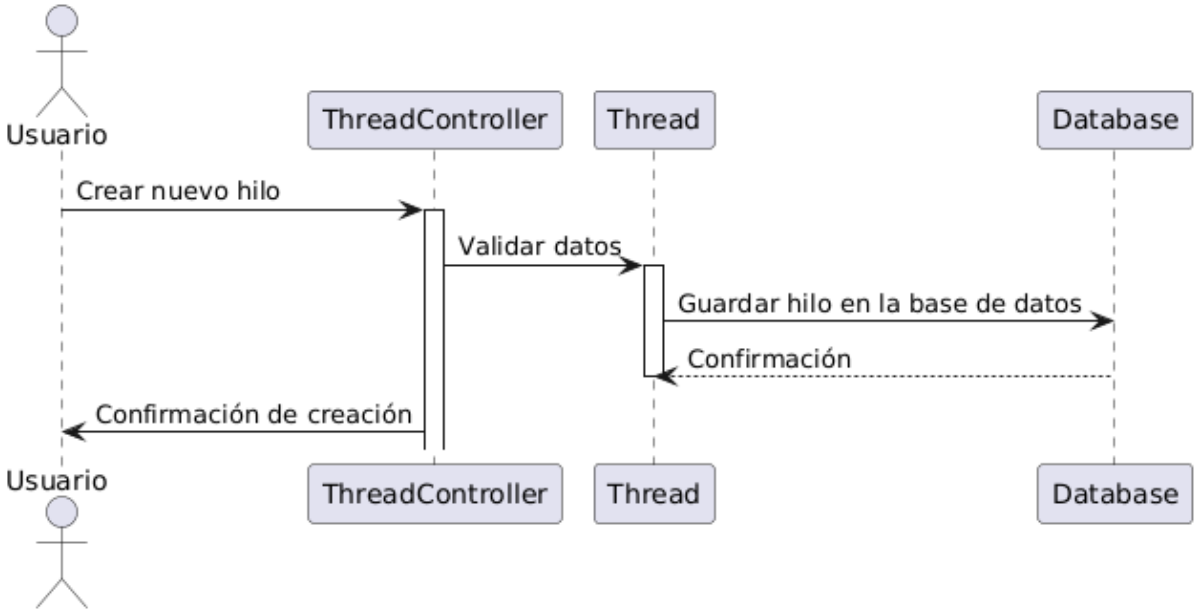
b. Diseño Lógico Relacional



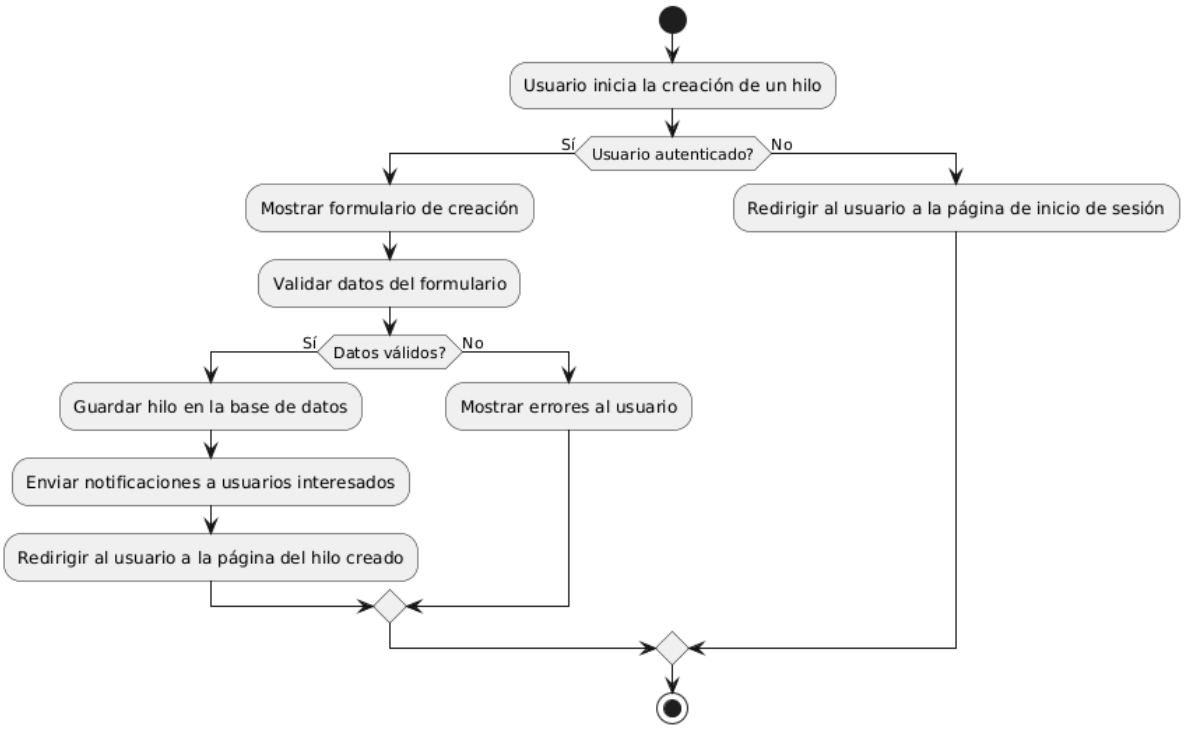
c. Diagrama de clases



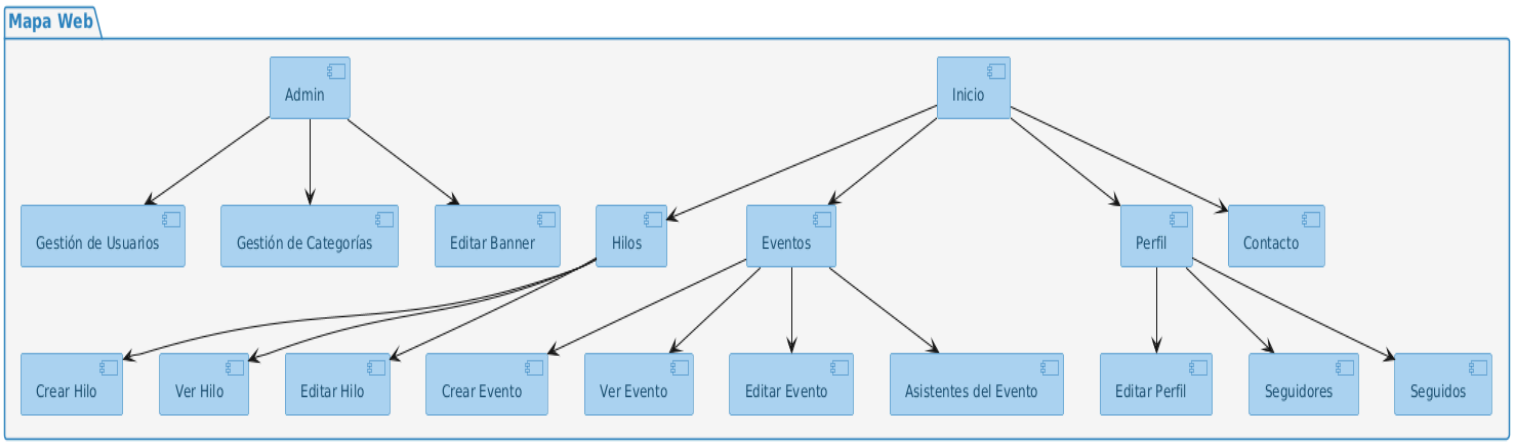
d. Diagrama de secuencias - Creación del hilo (lo principal)



e. Diagrama de actividad - Creación del hilo (lo principal)



f. Mapa Web. Gráfico que muestra los enlaces entre páginas



Codificación

a. Tecnologías elegidas y su justificación

Front End

HTML y CSS: Es el lenguaje base para estructurar el contenido de las páginas web, mientras que CSS se utiliza para darles estilo y diseño. En el proyecto Birbum, estas tecnologías son esenciales para construir una interfaz de usuario clara y accesible, asegurando que los usuarios puedan interactuar fácilmente con las funcionalidades de la plataforma.

Tailwind CSS: Este framework de CSS permite un diseño rápido y eficiente gracias a su enfoque basado en utilidades. En Birbum, Tailwind CSS facilita la creación de interfaces modernas y responsivas, reduciendo el tiempo de desarrollo y asegurando consistencia en el diseño.

DaisyUI: Este framework de CSS permite un diseño rápido y eficiente gracias a su enfoque basado en utilidades. En Birbum, Tailwind CSS facilita la creación de interfaces modernas y responsivas, reduciendo el tiempo de desarrollo y asegurando consistencia en el diseño.

JavaScript: Este lenguaje de programación es fundamental para añadir interactividad a las páginas web. En Birbum, se utiliza para implementar funcionalidades dinámicas, como la actualización de contenido sin recargar la página y la validación de formularios en el cliente.

AlpineJS: Este framework ligero de JavaScript permite crear interfaces reactivas e interactivas de manera sencilla. En Birbum, AlpineJS se utiliza para manejar interacciones como menús desplegables y modales, mejorando la experiencia del usuario sin necesidad de bibliotecas más pesadas.

Blade: Blade es el motor de plantillas de Laravel que permite crear vistas dinámicas de manera sencilla. En Birbum, se utiliza para estructurar las páginas web y combinar HTML con datos del backend, facilitando la reutilización de componentes y la organización del código.

Vite: Vite es una herramienta de desarrollo frontend que permite una construcción rápida y eficiente de los activos de la aplicación, como CSS y JavaScript. En Birbum, se utiliza para optimizar el tiempo de desarrollo y mejorar el rendimiento en producción, gracias a su servidor de desarrollo rápido y su capacidad para dividir el código en módulos.

Backend

PHP 8.3.6: PHP es un lenguaje de programación de código abierto ampliamente utilizado para el desarrollo web del lado del servidor. En Birbum, PHP permite manejar la lógica de negocio, interactuar con la base de datos y procesar solicitudes del cliente de manera eficiente.

Laravel: Laravel es un framework de PHP que simplifica el desarrollo web gracias a su estructura organizada y herramientas integradas. En Birbum, Laravel facilita la implementación de funcionalidades como autenticación, manejo de rutas y acceso a la base de datos.

Laravel Tinker: Esta consola interactiva es útil para realizar pruebas rápidas y depuración. En Birbum, Tinker permite a los desarrolladores interactuar directamente con los modelos y datos de la aplicación durante el desarrollo.

Laravel Breeze: Este paquete proporciona una solución simple para la autenticación. En Birbum, Breeze se utiliza para implementar un sistema de inicio de sesión básico, ahorrando tiempo y esfuerzo en el desarrollo de esta funcionalidad.

Pusher: Este paquete permite integrar Pusher para notificaciones y actualizaciones en tiempo real. En Birbum, se utiliza para mejorar la interacción en tiempo real, como notificaciones de nuevos comentarios o eventos.

Servidor de correos de Google (Gmail SMTP): Gmail SMTP se utiliza para enviar correos electrónicos, como los de verificación de cuenta y recuperación de contraseña. En Birbum, esta funcionalidad es crucial para garantizar la seguridad y comunicación con los usuarios.

Herramientas

Git: Git es una herramienta esencial para el control de versiones. En Birbum, se utiliza para gestionar el código fuente, permitiendo a los desarrolladores colaborar de manera eficiente y mantener un historial de cambios.

DBeaver: Esta herramienta facilita la administración de bases de datos. En Birbum, DBeaver se utiliza para gestionar y consultar la base de datos durante el desarrollo y la depuración.

Composer 2.8.8: Composer es un gestor de dependencias para PHP. En Birbum, se utiliza para instalar y gestionar bibliotecas y paquetes necesarios, como los de Laravel y otros componentes del backend.

Infraestructura

MariaDB 10.11.13: Este sistema de gestión de bases de datos relacional es conocido por su rendimiento y características avanzadas. En Birbum, MariaDB almacena y gestiona los datos de usuarios, foros y eventos de manera eficiente.

Nginx 1.24.0: Nginx es un servidor web de alto rendimiento. En Birbum, se utiliza para servir contenido web y manejar solicitudes HTTP, asegurando una experiencia rápida y confiable para los usuarios.

Node.js 20.19.2 y npm 10.8.2: Node.js es un entorno de ejecución para JavaScript en el lado del servidor, y npm es su gestor de paquetes. En Birbum, se utilizan para instalar y gestionar herramientas de frontend como Tailwind CSS y DaisyUI.

Let's Encrypt: Let's Encrypt proporciona certificados SSL gratuitos. En Birbum, se utiliza para asegurar las conexiones HTTPS, protegiendo la información de los usuarios y cumpliendo con los estándares de seguridad.

Cloudflare: Cloudflare actúa como un CDN y ofrece protección contra ataques DDoS. En Birbum, se utiliza para mejorar el rendimiento y la seguridad de la aplicación, asegurando una experiencia fluida para los usuarios.

b. Entorno servidor

i. Descripción general

El backend de Birbum está construido con Laravel siguiendo la arquitectura MVC (Modelo-Vista-Controlador):

Modelos: Gestionan la lógica de negocio y las relaciones entre entidades como User, Thread, Comment y Event. Estos modelos interactúan directamente con la base de datos mediante Eloquent ORM.

Controladores: Procesan las peticiones HTTP, validan los datos recibidos y coordinan la interacción entre los modelos y las vistas.

Vistas: Utilizan templates Blade para renderizar el HTML final que se envía al cliente, asegurando una experiencia de usuario consistente.

Rutas: Definidas en web.php, están organizadas por funcionalidad y protegidas por middleware para garantizar la seguridad y el acceso adecuado.

Middleware: Los middlewares en Laravel se utilizan para filtrar las solicitudes HTTP antes de que lleguen a los controladores. En Birbum, se emplean para garantizar que solo los usuarios autorizados puedan acceder a ciertas rutas, como las de administración o las relacionadas con el perfil privado. Ejemplo:

Este middleware asegura que solo los usuarios autenticados puedan acceder a la ruta /profile, protegiendo la información sensible del perfil.

```
Route::middleware(['auth'])->group(function () {
    Route::get('/profile', [ProfileController::class, 'show']);
});
```

Políticas: Las políticas de Laravel definen la autorización para realizar acciones específicas en los modelos. En Birbum, se utilizan para controlar el acceso a recursos como comentarios, hilos y eventos, asegurando que solo los usuarios con permisos adecuados puedan modificarlos o eliminarlos. Ejemplo:

Este código verifica si el usuario actual tiene permiso para actualizar un comentario específico, utilizando la política asociada al modelo `Comment`.

```
$this->authorize('update', $comment);
```

Eventos de Pusher: Los eventos de Pusher son mensajes que se envían desde el servidor a los clientes conectados a un canal en tiempo real. En Birbum, se utilizan para notificar a los usuarios sobre acciones importantes, como la creación de nuevos comentarios o la eliminación de contenido. Esto mejora la experiencia del usuario al mantener la interfaz actualizada sin necesidad de recargar la página. La integración con Pusher se realiza a través del driver configurado en config/broadcasting.php, y Laravel facilita la emisión de estos eventos mediante su sistema de broadcasting. Ejemplo:

Este código emite un evento llamado NewCommentEvent para notificar a otros usuarios conectados sobre un nuevo comentario. Se utiliza el sistema de broadcasting de Laravel junto con Pusher para enviar el evento en tiempo real.

```
broadcast(new NewCommentEvent($comment))->toOthers();
```

ii. Seguridad. Evitar inyección en bases de datos

Eloquent ORM: Todas las consultas a la base de datos utilizan el ORM de Laravel, que implementa consultas preparadas automáticamente para prevenir inyecciones SQL.

Validación de entrada: Se utiliza el sistema de validación de Laravel en todos los controladores antes de procesar los datos, asegurando que sean seguros y válidos.

Tokens CSRF: Laravel incluye protección automática contra ataques de falsificación de solicitudes entre sitios mediante tokens CSRF.

Mass Assignment Protection: Los modelos utilizan la propiedad `$fillable` para controlar qué campos pueden ser asignados masivamente, evitando vulnerabilidades.

iii. Evitar o capturar errores y warnings

Try-catch en JavaScript: Implementados en las peticiones fetch para manejar errores de red y respuestas inesperadas del servidor.

Validación en controladores: Los datos se validan exhaustivamente antes de ser procesados, reduciendo la posibilidad de errores.

Manejo de errores en formularios: Se muestran mensajes de error específicos al usuario para mejorar la experiencia y facilitar la corrección de datos.

Logs de Laravel: Configurados para registrar errores del sistema, lo que facilita el debugging y el mantenimiento del proyecto.

c. Entorno Cliente

Descripción general

El entorno cliente de Birbum está diseñado para ofrecer una experiencia de usuario fluida, accesible y moderna. Este entorno combina tecnologías avanzadas para garantizar que la interfaz sea atractiva, funcional y fácil de usar en diferentes dispositivos y navegadores.

Tailwind CSS: Este framework utiliza un sistema de clases utilitarias que permite un diseño rápido y consistente. Su capacidad para generar diseños responsivos sin necesidad de escribir CSS personalizado ha sido fundamental para garantizar que la interfaz se adapte a diferentes dispositivos. Además, su soporte para temas oscuros y claros mejora la accesibilidad y la experiencia del usuario.

DaisyUI: Como extensión de Tailwind CSS, DaisyUI ha proporcionado componentes predefinidos como botones, formularios y modales, lo que ha acelerado significativamente el desarrollo de la interfaz. La facilidad para personalizar estos componentes ha permitido adaptarlos al diseño único de Birbum, manteniendo un estilo profesional y uniforme.

AlpineJS: Este framework ligero es ideal para añadir interactividad a la interfaz sin sobrecargar el frontend. Funcionalidades como menús desplegables, modales y cambios dinámicos en la interfaz se han implementado de manera eficiente, mejorando la experiencia del usuario sin necesidad de bibliotecas más pesadas.

Fetch API: La Fetch API ha sido clave para manejar la comunicación entre el cliente y el servidor. Su capacidad para realizar solicitudes HTTP de manera eficiente ha permitido implementar acciones como el envío de comentarios y la eliminación de contenido en tiempo real, mejorando la fluidez y la interacción en la plataforma.

Asegurar la funcionalidad en los navegadores más usados

Tailwind CSS: Framework que genera CSS optimizado y compatible con navegadores modernos.

Testing cross-browser: Verificación manual en navegadores como Chrome, Firefox, Safari y Edge para asegurar que todas las funcionalidades operen correctamente.

Responsive design: Diseño adaptativo que funciona en desktop, tablet y móvil, asegurando una experiencia consistente en diferentes dispositivos.

d. Documentación interna de código

i. Descripción de cada fichero. Nombre y función

Modelos (app/Models/)

Category.php

Función: Modelo Eloquent que representa las categorías del sistema. Gestiona la estructura jerárquica de categorías y sus relaciones con hilos.

Comment.php

Función: Modelo que representa los comentarios en hilos. Maneja relaciones con usuarios, hilos y notificaciones.

Event.php

Función: Modelo que representa eventos creados por usuarios. Gestiona asistentes, relaciones con usuarios y detalles del evento.

Thread.php

Función: Modelo que representa hilos en los foros. Maneja relaciones con usuarios, categorías, comentarios y votos.

User.php

Función: Modelo de usuario extendido de Authenticatable. Gestiona datos de perfil, relaciones con hilos, comentarios y eventos.

Controladores (app/Http/Controllers/)

EventController.php

Función: Controlador para la gestión de eventos. Maneja la creación, edición, visualización y eliminación de eventos.

ProfileController.php

Función: Controlador para la gestión de perfiles de usuario. Permite la edición de información personal y la visualización de seguidores.

CommentController.php

Función: Controlador para la gestión de comentarios en hilos. Maneja la creación, edición y eliminación de comentarios.

AdminCategoryController.php

Función: Gestiona las categorías desde el panel de administración, permitiendo su creación, edición y eliminación.

AdminController.php

Función: Controlador principal para funcionalidades administrativas generales.

AdminUserController.php

Función: Gestiona usuarios desde el panel de administración, incluyendo roles y permisos.

ContactController.php

Función: Maneja el formulario de contacto y el envío de mensajes al equipo de soporte.

EventAttendeesController.php

Función: Gestiona la lista de asistentes a eventos, permitiendo su visualización y administración.

FollowerController.php

Función: Controla las relaciones de seguidores entre usuarios, permitiendo seguir y dejar de seguir.

NotificationController.php

Función: Gestiona las notificaciones de usuario, incluyendo su visualización y marcado como leídas.

ProfileFollowersController.php

Función: Muestra y administra los seguidores de un perfil de usuario.

ThreadVoteController.php

Función: Gestiona los votos en hilos, permitiendo a los usuarios votar positiva o negativamente.

Middleware (app/Http/Middleware/)

auth.php

Función: Middleware que asegura que el usuario esté autenticado antes de acceder a ciertas rutas. Este middleware es parte de Laravel y no requiere una clase explícita en app/Http/Middleware.

verified

Función: Middleware que asegura que el usuario haya verificado su correo electrónico antes de acceder a ciertas rutas.

IsAdmin.php

Función: Middleware que asegura que el usuario tenga permisos de administrador antes de acceder a ciertas rutas.

Sistema de Rutas (`routes/web.php`)

Función: Define todas las rutas web de la aplicación con sus respectivos middlewares y controladores. Incluye rutas para autenticación, foros, eventos y administración.

JavaScript (resources/js/)

app.js

Función: Archivo principal de JavaScript que inicializa Alpine.js y otros scripts necesarios para la aplicación.

echo-comments.js

Función: Configura Pusher para la transmisión en tiempo real de comentarios en hilos.

CSS (resources/css/)

app.css

Función: Archivo CSS principal que incluye Tailwind CSS y estilos personalizados. Define comportamientos como scroll suave y estilos responsivos.

Vistas Principales (`resources/views/`)

welcome.blade.php

Función: Vista principal de bienvenida. Proporciona un punto de entrada para usuarios no autenticados.

dashboard.blade.php

Función: Vista del panel principal para usuarios autenticados. Muestra estadísticas y accesos rápidos.

profile/show.blade.php

Función: Vista del perfil público de un usuario. Muestra información básica y seguidores.

Subcarpetas de vistas:

admin/

categories/: Subcarpeta para vistas relacionadas con categorías.

users/: Subcarpeta para vistas relacionadas con la gestión de usuarios.

dashboard.blade.php: Vista del panel de administración.

edit-banner.blade.php: Vista para editar el banner.

auth/

confirm-password.blade.php: Vista para confirmar la contraseña.

forgot-password.blade.php: Vista para recuperar la contraseña.

login.blade.php: Vista de inicio de sesión.
register.blade.php: Vista de registro.
reset-password.blade.php: Vista para restablecer la contraseña.
verify-email.blade.php: Vista para verificar el correo electrónico.

components/
Archivos reutilizables como:
application-logo.blade.php
auth-session-status.blade.php
breadcrumbs.blade.php
footer.blade.php
header.blade.php
modal.blade.php
notification-menu.blade.php
thread-preview.blade.php
user-avatar.blade.php

errors/
404.blade.php: Vista para manejar errores 404.

events/
attendees.blade.php: Vista para mostrar asistentes a eventos.
create.blade.php: Vista para crear un evento.
edit.blade.php: Vista para editar un evento.
index.blade.php: Vista para listar eventos.
show.blade.php: Vista para mostrar detalles de un evento.

layouts/
app.blade.php: Plantilla base para la aplicación.
guest.blade.php: Plantilla para vistas de invitados.

profile/
edit.blade.php: Vista para editar el perfil.
followers.blade.php: Vista para mostrar seguidores.
partials/: Subcarpeta con fragmentos reutilizables.
show.blade.php: Vista para mostrar el perfil.

profile/partials/
delete-user-form.blade.php: Formulario para eliminar la cuenta del usuario.
update-password-form.blade.php: Formulario para actualizar la contraseña del usuario.
update-profile-information-form.blade.php: Formulario para actualizar la información del perfil del usuario.

threads/

create.blade.php: Vista para crear un hilo.
edit.blade.php: Vista para editar un hilo.
index.blade.php: Vista para listar hilos.
show.blade.php: Vista para mostrar detalles de un hilo.

ii. Descripción de cada función(destacados)

Modelos (app/Models/)

User
- eventsAttending: Devuelve los eventos a los que el usuario está asistiendo.
- getAvatarUrlAttribute: Obtiene la URL del avatar del usuario.
- followers: Devuelve los seguidores del usuario.
- following: Devuelve los usuarios que el usuario está siguiendo.
- threads: Devuelve los hilos creados por el usuario.

Thread
- category: Devuelve la categoría asociada al hilo.
- user: Devuelve el usuario que creó el hilo.
- comments: Devuelve los comentarios asociados al hilo.
- votes: Devuelve los votos asociados al hilo.
- votedBy: Verifica si un usuario específico ha votado por el hilo.

Event
- user: Devuelve el usuario que creó el evento.
- attendees: Devuelve los usuarios que asisten al evento.

Notification
- user: Devuelve el usuario al que pertenece la notificación.
- markAsRead: Marca la notificación como leída.
- delete: Elimina la notificación.

Vistas (resources/views/)

threads/create.blade.php
- updateTitleCounter: Actualiza el contador de caracteres del título.
- updateContentCounter: Actualiza el contador de caracteres del contenido.

components/header.blade.php
- markNotificationRead: Marca una notificación como leída.

- markNotificationReadAndRedirect: Marca una notificación como leída y redirige a una URL específica.

profile/partials/update-profile-information-form.blade.php
- updateBioCounterProfile: Actualiza el contador de caracteres de la biografía del perfil.

notifications/index.blade.php
- renderNotification: Renderiza una notificación en el menú de notificaciones con estilos adaptados al modo claro/oscuro.

Controladores (app/Http/Controllers/)

AuthenticatedSessionController
- create: Devuelve la vista para iniciar sesión.
- store: Procesa la solicitud de inicio de sesión y redirige al usuario.
- destroy: Cierra la sesión del usuario y redirige a la página principal.

AdminUserController
- index: Lista todos los usuarios registrados.
- edit: Muestra el formulario para editar un usuario específico.
- update: Actualiza la información de un usuario en la base de datos.
- destroy: Elimina un usuario del sistema.

ThreadController
- index: Muestra una lista de todos los hilos disponibles.
- create: Devuelve la vista para crear un nuevo hilo.
- store: Guarda un nuevo hilo en la base de datos.
- show: Muestra los detalles de un hilo específico.
- edit: Devuelve la vista para editar un hilo existente.
- update: Actualiza un hilo en la base de datos.
- destroy: Elimina un hilo de la base de datos.

EventController
- join: Permite a un usuario unirse a un evento.
- leave: Permite a un usuario salir de un evento.
- index: Lista todos los eventos disponibles.
- create: Devuelve la vista para crear un nuevo evento.
- store: Guarda un nuevo evento en la base de datos.
- show: Muestra los detalles de un evento específico.
- edit: Devuelve la vista para editar un evento existente.
- update: Actualiza un evento en la base de datos.
- destroy: Elimina un evento de la base de datos.

CommentController
- store: Guarda un nuevo comentario asociado a un hilo.
- destroy: Elimina un comentario específico.

NotificationController
- index: Muestra todas las notificaciones del usuario.
- markAsRead: Marca una notificación específica como leída.
- markAllAsRead: Marca todas las notificaciones como leídas.
- delete: Elimina una notificación específica.
- deleteAll: Elimina todas las notificaciones del usuario.

Notifications (app/Http/Notifications/)

ThreadCommented.php
- broadcastOn: Define el canal de transmisión para la notificación.
- toDatabase: Convierte la notificación en un formato adecuado para la base de datos.

Policies (app/Http/Policies/)

CommentPolicy.php
- delete: Determina si el usuario puede borrar un comentario. Permite esta acción si el usuario es el autor del comentario o un administrador.

ThreadPolicy
- delete: Determina si el usuario puede eliminar un hilo. Permite esta acción si el usuario es el autor del hilo o un administrador.

EventPolicy
- update: Determina si el usuario puede actualizar un evento. Permite esta acción si el usuario es el creador del evento o un administrador.
- delete: Determina si el usuario puede eliminar un evento. Permite esta acción si el usuario es el creador del evento o un administrador.

JavaScript (resources/js/)

echo-comments.js
- removeCommentFromUI: Elimina un comentario de la interfaz de usuario.
- renderNewComment: Renderiza un nuevo comentario en la interfaz de usuario.

notifications.js
- loadNotifications: Carga las notificaciones del usuario de forma asíncrona y actualiza la lista en el menú de notificaciones.
- markNotificationRead: Marca una notificación específica como leída mediante una solicitud al servidor.

e. Documentación externa

Laravel Framework

- Eloquent ORM: https://laravel.com/docs/eloquent
- Routing: https://laravel.com/docs/routing
- Blade Templates: https://laravel.com/docs/blade
- Middleware: https://laravel.com/docs/middleware
- Broadcasting: https://laravel.com/docs/broadcasting
- File Storage: https://laravel.com/docs//filesystem
- Notifications: https://laravel.com/docs/notifications
- Events: https://laravel.com/docs/events
- Migrations: https://laravel.com/docs/migrations

Front end

- CSS: https://developer.mozilla.org/es/docs/Web/CSS
- Tailwind CSS: https://tailwindcss.com/docs/
- DaisyUI: https://daisyui.com/docs/
- Javascript: https://developer.mozilla.org/es/docs/Web/JavaScript
- FetchAPI: https://developer.mozilla.org/en-US/docs/Web/API/Fetch_API
-AlpineJS: https://alpinejs.dev/docs
- Laravel Echo: https://laravel.com/docs/broadcasting#client-side-installation
- Pusher: https://pusher.com/docs

Backend

- Laravel 12.0 Documentation: https://laravel.com/docs/12.x

Herramientas

- Vite: https://vite.dev/guide/assets.html
- Laravel Vite Plguin: https://github.com/laravel/vite-plugin

Infraestructura

- NodeJS: https://nodejs.org/docs/latest/api/
- Nginx: https://nginx.org/en/docs/
- Let’s Encrypt: https://letsencrypt.org/docs/
- Certbot: https://certbot.eff.org/
- Monolog: https://seldaek.github.io/monolog/
- Cloudflare: https://developers.cloudflare.com/rules/

i. Manual del usuario

Accesible desde la web

La página es accesible desde https://birbum.xyz/.

Funcionalidades principales

Usuario normal

Birbum ofrece a los usuarios una experiencia única para interactuar con la comunidad y gestionar su presencia en la plataforma. Los usuarios pueden registrarse fácilmente, personalizar su perfil con información y avatares, y participar activamente en foros creando hilos, comentando y votando publicaciones. Los hilos son espacios de discusión donde los usuarios pueden compartir ideas, plantear preguntas o debatir sobre temas de interés. Cada hilo puede recibir comentarios y votos, lo que permite destacar los contenidos más relevantes y fomentar la interacción entre los miembros de la comunidad. Además, los usuarios tienen la posibilidad de seguir hilos específicos para recibir notificaciones sobre nuevas actividades relacionadas.

En cuanto a los eventos, Birbum permite a los usuarios explorar una variedad de actividades organizadas por la comunidad o los administradores. Los eventos son una excelente oportunidad para conectar con otros usuarios, participar en actividades exclusivas y mantenerse al día con las novedades de la plataforma. Aunque solo los administradores pueden crear eventos, los usuarios pueden unirse a ellos, interactuar con otros participantes y recibir actualizaciones en tiempo real sobre cualquier cambio o novedad relacionada con los eventos.

El sistema de notificaciones en tiempo real asegura que los usuarios estén siempre informados sobre las actividades más relevantes, como interacciones en sus publicaciones, actualizaciones en eventos o nuevos seguidores. Esto garantiza una experiencia dinámica y conectada dentro de la plataforma.

Administrador

Los administradores disfrutan de herramientas avanzadas para gestionar la plataforma de manera eficiente. Pueden supervisar y moderar el contenido, asegurando que la comunidad se mantenga segura y organizada. Además, tienen la capacidad de gestionar usuarios, crear categorías y añadir eventos exclusivos que fomenten la participación activa de la comunidad. Los administradores también pueden destacar hilos importantes o moderar discusiones para mantener un ambiente respetuoso y constructivo.

Diseño y rendimiento

Birbum está diseñado para ofrecer una experiencia fluida y atractiva en cualquier dispositivo, garantizando que los usuarios puedan disfrutar de todas las funcionalidades sin interrupciones. La plataforma prioriza la seguridad y la eficiencia, asegurando que los datos de los usuarios estén protegidos y que las interacciones sean rápidas y confiables.

Despliegue

a. Guía de instalación

El despliegue de Birbum es un proceso sencillo que puedes realizar siguiendo estos pasos básicos. A continuación, te explicamos cómo hacerlo de manera clara y simple:

1. Actualiza el sistema

Antes de comenzar, asegúrate de que tu sistema esté actualizado:

```
sudo apt update && sudo apt upgrade -y
```

2. Instala MariaDB

Configura la base de datos con MariaDB:

```
sudo apt install mariadb-server -y

sudo systemctl enable mariadb

sudo mysql_secure_installation
```

3. Crea la base de datos y usuario

```
sudo mysql -u root -e "CREATE DATABASE birbum CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci;"

sudo mysql -u root -e "CREATE USER 'birbum'@'localhost' IDENTIFIED BY 'tu_password;"

sudo mysql -u root -e "GRANT ALL PRIVILEGES ON birbum.* TO 'birbum'@'localhost';"

sudo mysql -u root -e "FLUSH PRIVILEGES;"
```

4. Instala PHP y extensiones necesarias

Birbum requiere PHP y algunas extensiones específicas:

```
sudo add-apt-repository ppa:ondrej/php -y

sudo apt install php8.3 php8.3-fpm php8.3-mysql php8.3-xml php8.3-mbstring php8.3-curl -y
```

sudo systemctl enable nginx

5. Instala Composer

Composer es el gestor de dependencias de PHP:

curl -sS https://getcomposer.org/installer | php

sudo mv composer.phar /usr/local/bin/composer

6. Instala Node.js y npm

Node.js y npm son necesarios para compilar los assets:

curl -fsSL https://deb.nodesource.com/setup_20.x | sudo -E bash -

sudo apt install nodejs -y

7. Clona tu proyecto y configura permisos

Descarga el proyecto desde tu repositorio y ajusta los permisos:

cd /var/www

sudo git clone TU_REPO_GIT_URL birbum

sudo chown -R $USER:www-data birbum

cd birbum

8. Instala dependencias del proyecto

Instala las dependencias necesarias:

composer install

npm install

9. Configura el archivo de entorno

Copia y edita el archivo `.env` para ajustar la configuración:

cp .env.example .env

nano .env

CONFIGURAREMOS SEGÚN CREDENCIALES

10. Genera la clave de la app y ejecuta migraciones

Configura la aplicación y prepara la base de datos:

```
php artisan key:generate
```

```
php artisan migrate --seed
```

11. Configura Nginx

Asegúrate de que Nginx esté configurado correctamente para servir tu aplicación Laravel.
Edita el archivo de configuración de Nginx:

```
sudo nano /etc/nginx/sites-available/default
```

Incluye lo siguiente en la configuración:

```
location ~ \.php$ {
    include snippets/fastcgi-php.conf;
    fastcgi_pass unix:/var/run/php/php8.3-fpm.sock;
    fastcgi_param SCRIPT_FILENAME $document_root$fastcgi_script_name;
    include fastcgi_params;
}
```

Crear un enlace simbólico para habilitar la configuración del dominio:

Si has creado un nuevo archivo de configuración para tu dominio en `/etc/nginx/sites-available/`, enlázalo a `sites-enabled`:

```
sudo ln -s /etc/nginx/sites-available/your-domain.com /etc/nginx/sites-enabled/
```

Asegúrate de reemplazar `your-domain.com` con el nombre de tu archivo de configuración real.

Probar la configuración de Nginx:

Antes de reiniciar Nginx, verifica que la configuración sea válida:

```
sudo nginx -t
```

Reiniciar Nginx:

Si no hay errores, reinicia Nginx para aplicar los cambios:

```
sudo systemctl restart nginx
```

12. Compila los assets

Compila los archivos CSS y JavaScript:

```
npm run build
```

13. Configura permisos finales

Asegúrate de que los permisos sean correctos:

```
sudo chown -R www-data:www-data birbum/
```

14. Establecimiento del SSL

Instalar Certbot y el plugin para Nginx:

```
sudo apt update
sudo apt install certbot python3-certbot-nginx
```

Obtener un certificado SSL:
Ejecuta el siguiente comando para obtener y configurar automáticamente un certificado SSL para tu dominio:

```
sudo certbot --nginx -d your-domain.com -d www.your-domain.com
```

- Sustituye `your-domain.com` y `www.your-domain.com` por tu dominio real.
- Certbot configurará automáticamente tu archivo de configuración de Nginx para usar el certificado.

Verificar la renovación automática:

Certbot instala automáticamente un cron job para renovar los certificados. Puedes verificarlo ejecutando:

```
sudo systemctl list-timers | grep certbot
```

Reiniciar Nginx:

Después de configurar el certificado, reinicia Nginx para aplicar los cambios:

```
sudo systemctl restart nginx
```

Configura un bloque de servidor NGINX para redirigir tráfico HTTP a HTTPS y manejar SSL:

```
  server {
      listen 80;
      server_name your-domain.com www.your-domain.com;

      # Redirigir todo el tráfico HTTP a HTTPS
      return 301 https://$host$request_uri;
  }

  server {
      listen 443 ssl;
      server_name your-domain.com www.your-domain.com;

      root /var/www/birbum/public;
      index index.php index.html;

      ssl_certificate /etc/letsencrypt/live/your-domain.com/fullchain.pem;
      ssl_certificate_key /etc/letsencrypt/live/your-domain.com/privkey.pem;
      ssl_trusted_certificate /etc/letsencrypt/live/your-domain.com/chain.pem;

      ssl_protocols TLSv1.2 TLSv1.3;
      ssl_ciphers HIGH:!aNULL:!MD5;

      location / {
          try_files $uri $uri/ /index.php?$query_string;
      }

      location ~ \.php$ {
          include snippets/fastcgi-php.conf;
          fastcgi_pass unix:/var/run/php/php8.1-fpm.sock; # Ajusta la versión de PHP según
corresponda
          fastcgi_param SCRIPT_FILENAME $document_root$fastcgi_script_name;
          include fastcgi_params;
```

```
    }

    location ~ /\.ht {
        deny all;
    }
}
```

Asegúrate de reemplazar `your-domain.com` y `www.your-domain.com` con tu dominio real.

15. Configurar el Firewall

Activar el firewall:
Asegúrate de que el firewall esté activado para proteger el servidor:
sudo ufw enable

Permitir solo tráfico HTTP y HTTPS:
Configura el firewall para permitir únicamente el tráfico en los puertos 80 (HTTP) y 443 (HTTPS):
sudo ufw allow 'Nginx Full'

Bloquear todo el tráfico no autorizado:
Configura el firewall para denegar cualquier otro tráfico:
sudo ufw default deny incoming
sudo ufw default allow outgoing

Verificar el estado del firewall:
Asegúrate de que las reglas se hayan aplicado correctamente:
sudo ufw status

Ahora como apuntaremos Cloudflare a nuestro servidor, debemos de permitir sólo conexiones de IPs públicas

Obtener las IPs públicas de Cloudflare:
Cloudflare publica sus rangos de IP en su documentación oficial. Puedes obtenerlos desde:
https://www.cloudflare.com/ips/

Configurar reglas en el firewall:
Permite únicamente las IPs públicas de Cloudflare en los puertos 80 (HTTP) y 443 (HTTPS):
```
for ip in $(curl -s https://www.cloudflare.com/ips-v4); do
    sudo ufw allow from $ip to any port 80
```

```
    sudo ufw allow from $ip to any port 443
done
```

Bloquear todo el tráfico no autorizado:
Asegúrate de denegar cualquier otra conexión que no provenga de las IPs de Cloudflare:

```
sudo ufw default deny incoming
sudo ufw default allow outgoing
```

Verificar las reglas del firewall:
Confirma que las reglas se hayan aplicado correctamente:

```
sudo ufw status
```

Reinicia Nginx para aplicar los cambios:

```
sudo systemctl restart nginx
```

¡Listo! Ahora tu proyecto Birbum debería estar desplegado y funcionando correctamente.

b. Ficheros de configuración

.env

```
APP_NAME=Laravel
APP_ENV=local
APP_KEY=base64:tDEp2U52WiR803ReTRCrIUukN/r2ojGzA1vEnQD+vXs=
APP_DEBUG=true
APP_URL=https://birbum.com

# --- Pusher Broadcasting ---
BROADCAST_DRIVER=pusher
PUSHER_APP_ID=2005553
PUSHER_APP_KEY=417e0688f9faf71c393b
PUSHER_APP_SECRET=7bb7a36a5e6796b1bd82
PUSHER_APP_CLUSTER=eu


APP_LOCALE=en
APP_FALLBACK_LOCALE=en
APP_FAKER_LOCALE=en_US

APP_MAINTENANCE_DRIVER=file
```

```
# APP_MAINTENANCE_STORE=database

PHP_CLI_SERVER_WORKERS=4

BCRYPT_ROUNDS=12

LOG_CHANNEL=stack
LOG_STACK=single
LOG_DEPRECATIONS_CHANNEL=null
LOG_LEVEL=debug

DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=birbum
DB_USERNAME=root
DB_PASSWORD=C9n54as3na404.

SESSION_DRIVER=database
SESSION_LIFETIME=120
SESSION_ENCRYPT=false
SESSION_PATH=/
SESSION_DOMAIN=null

## BROADCAST_CONNECTION=log
FILESYSTEM_DISK=local
QUEUE_CONNECTION=database

CACHE_STORE=database
# CACHE_PREFIX=

MEMCACHED_HOST=127.0.0.1

REDIS_CLIENT=phpredis
REDIS_HOST=127.0.0.1
REDIS_PASSWORD=null
REDIS_PORT=6379

MAIL_MAILER=smtp
MAIL_HOST=smtp.gmail.com
MAIL_PORT=587
```

```
MAIL_USERNAME=corvinumdev@gmail.com
MAIL_PASSWORD="moud laof qydp tdvi"
MAIL_ENCRYPTION=tls
MAIL_FROM_ADDRESS=corvinumdev@gmail.com
MAIL_FROM_NAME="Birbum"

AWS_ACCESS_KEY_ID=
AWS_SECRET_ACCESS_KEY=
AWS_DEFAULT_REGION=us-east-1
AWS_BUCKET=
AWS_USE_PATH_STYLE_ENDPOINT=false

VITE_APP_NAME="${APP_NAME}"
VITE_PUSHER_APP_KEY=417e0688f9faf71c393b
VITE_PUSHER_APP_CLUSTER=eu
```

nginx conf

```
server {


    server_name birbum.xyz www.birbum.xyz;
    root /var/www/birbum/public;


    index index.php index.html index.htm;


    location / {
        try_files $uri $uri/ /index.php?$query_string;
    }


    location ~ \.php$ {
        include snippets/fastcgi-php.conf;
        fastcgi_pass unix:/var/run/php/php8.3-fpm.sock;
        fastcgi_param SCRIPT_FILENAME $document_root$fastcgi_script_name;
        include fastcgi_params;
    }

    location ~ /\.ht {
        deny all;
    }

    listen 443 ssl; # managed by Certbot
    ssl_certificate /etc/letsencrypt/live/birbum.duckdns.org/fullchain.pem; # managed by
Certbot
    ssl_certificate_key /etc/letsencrypt/live/birbum.duckdns.org/privkey.pem; # managed by
Certbot
    include /etc/letsencrypt/options-ssl-nginx.conf; # managed by Certbot
    ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem; # managed by Certbot


}
server {
    if ($host = birbum.duckdns.org) {
        return 301 https://$host$request_uri;
    } # managed by Certbot
```

```
    listen 80;


    server_name birbum.duckdns.org www.birbum.duckdns.org;
    return 404; # managed by Certbot


}
```

c. Descripción del servidor hosting utilizado(propio)

Actualmente se encuentra hosteado desde un equipo portátil en mi propia casa, conjunto a eso mantengo los puertos de mi router abiertos para lograr las conexiones a mí página.

Además he alquilado un dominio en GoDaddy por unos 75 céntimos, siendo este birbum.xyz.

Para protegerme de las malas conexiones se está utilizando Cloudflare con las medidas de reglas por country y otras features que provee Cloudflare para detectar bots.

Existe una forma más seguro que la que utilizo yo apuntando directamente a mí IP que es Cloudflare tunnel, pero si restringimos el acceso en el propio router a conexiones de IPs públicas de Cloudflare, no se encuentra realmente ningún tipo de problema. Esto especialmente va para contrarrestar el syn flood o más conocido como denegación de servicios, ya que podrían seguir llegando hasta nuestra máquina sabiendo la IP del router, ya que solo estamos apuntando a nuestra IP con Cloudflare, por lo que con este restringimiento, evitamos que puedas saltarse el paso del firewall.

Herramientas de apoyo

- Control de versiones con Github

Se ha utilizado para el acceso remoto y respaldos del código fuente en GitHub, ya que así se garantiza que el proyecto esté respaldado y accesible desde cualquier lugar, lo que es crucial por si ocurre cualquier emergencia o para realizar setups u desarrollos en otros dispositivos, a más de tener el código de una manera más accesible desde múltiples dispositivos.

Conclusiones

a. Conclusiones sobre el trabajo realizado

Realmente crear un proyecto desde cero con todo lo que conlleva el prediseño, las ideas, luego el desarrollo del front, back, infraestructura, y las miles de cosas que no pueden salir bien a la primera, y que hay que realizar muchos intentos, me doy cuenta que no hay siempre que abarcar demasiado, ya que tenía en un primer momento la idea de meter más cosas aún, ya que pensaba que tenía mucha falta de contenido, pero es que un proyecto se puede complicar e ir añadiendo cosas infinitamente.

b. Conclusiones personales

Trabajar en Birbum ha sido una experiencia enriquecedora, permitiendo aplicar y profundizar en el uso de tecnologías modernas y buenas prácticas de desarrollo. Además, la interacción con un proyecto más o menos real ha reforzado la importancia de la documentación clara y la planificación estructurada para el éxito del desarrollo.

He aprendido sobre todo a que hay que gestionar mejor los tiempos de desarrollo y el que dedicaremos a según que apartados de un proyecto, ya que a veces no vale tanto la pena, es más importante primero sacar las metas, y luego los adornos.

c. Posibles ampliaciones y mejoras

Aunque el proyecto cumple con los objetivos iniciales, existen oportunidades para ampliarlo y mejorarlo. Entre ellas, se podría implementar un sistema de gamificación para aumentar la participación de los usuarios, como medallas, suscripciones premium, integrar inteligencia artificial para personalizar las recomendaciones de contenido de alguna manera y optimizar aún más el rendimiento del frontend.

Bibliografía

Estado del arte
https://www.youtube.com/watch?v=wDwlLBKbWzE
https://core.ac.uk/download/pdf/51065259.pdf

Analytics de foros
https://www.semrush.com/website/

Cosas sobre montar foros
https://www.hoswedaje.com/web/como-empezar-a-montar-un-foro-o-comunidad/
https://www.ionos.co.uk/digitalguide/websites/website-creation/creating-a-forum-how-it-works/

Una guía de cloudflare
https://technoogies.com/es/la-gu%C3%ADa-definitiva-para-las-reglas-de-p%C3%A1gina-y-gratis-de-cloudflare/

Editor de los múltiples diagramas
https://www.planttext.com/

Para los certificados SSL
https://letsencrypt.org/

Algunos iconos de la web en svg
https://heroicons.com/