salesianos
ATOCHA

PROYECTO FIN DE CICLO ASIR: BINGO! SOCIAL MEDIA APP

Autor:
Robert Lita Jeler

Profesor:
Jose Manuel Prieto Gordo

Departamento:
Electrónica e Informática

Mar/2025

ÍNDICE

TABLA DE ILUSTRACIONES

I. INTRODUCCIÓN

Durante el proceso de creación de este proyecto para mi trabajo de fin de grado, me enfrenté a una realidad cada vez más evidente: la gran cantidad de redes sociales y aplicaciones que utilizamos a diario para comunicarnos, compartir contenido y mantenernos informados. Cada plataforma se especializa en una función concreta—ya sea chatear, publicar actualizaciones o seguir las últimas noticias—lo que obliga a tener múltiples aplicaciones instaladas en nuestros dispositivos. Esta fragmentación no solo resulta incómoda, sino que también complica la experiencia del usuario, al tener que alternar constantemente entre diferentes entornos digitales.

Fue en ese contexto que surgió la idea de crear “bingo!”, una aplicación que reúne en un único lugar las funciones esenciales de una red social: la creación de hilos y publicaciones, el envío de mensajes directos, la gestión de amigos y la posibilidad de comentar sobre las publicaciones. Mi objetivo principal con esta red social era simplificar la interacción en línea permitiendo a usuarios disfrutar de una experiencia integrada y sin interrupciones, donde todas las herramientas necesarias para la comunicación y el intercambio de ideas y contenido se encuentran a solo un click de distancia.

Para lograr esta integración, decidí utilizar tecnologías modernas y ampliamente reconocidas en el ámbito del desarrollo web. En el front-end, se emplearon HTML5, CSS3 y JavaScript, combinados con frameworks como Bootstrap, React, Composer, Node.js, Tailwind, que han permitido construir una interfaz intuitiva y visualmente atractiva. Gracias a un diseño responsive, “bingo!” se adapta perfectamente tanto a dispositivos de escritorio como a móviles, asegurando que la experiencia del usuario sea óptima sin importar dónde se encuentre.

Por el lado del back-end , opté por PHP junto con MySQL (MariaDB) para gestionar de manera eficiente y segura el registro de usuarios, el envío de mensajes y la publicación de contenido. Estas tecnologías, probadas y confiables, ofrecen una base robusta que permite manejar de forma segura las interacciones entre los usuarios, además de facilitar el escalado del sistema en el futuro. Se han implementado rigurosas medidas de seguridad, desde la validación y sanitización de datos en formularios hasta el uso de conexiones seguras mediante PDO, lo que garantiza la protección de la información personal y el funcionamiento correcto del sistema.

Para alojar la aplicación, configuré un servidor web utilizando una RaspBerry PI 4 con linux Ubuntu 24.04. Esta elección fue estratégica: la RaspBerry PI ofrece un entorno de producción eficiente y de bajo consumo energético, a la vez que permite la flexibilidad de conectar la red social a un dominio personalizado. Esto posibilita que “bingo!” sea accesible tanto a nivel local como global, dependiendo de la configuración y las necesidades futuras. Cada paso de la configuración se realizó meticulosamente para asegurar que el servidor pueda manejar el volumen de usuarios y datos previsto sin comprometer el rendimiento.

La seguridad y la privacidad han sido aspectos fundamentales durante todo el desarrollo del proyecto. Cada componente del sistema, desde la interfaz de usuario hasta la gestión de la base de datos, se ha diseñado teniendo en cuenta las mejores prácticas de seguridad, para prevenir vulnerabilidades y proteger la información sensible. Esta atención al detalle en materia de seguridad es vital para generar confianza en los

usuarios y garantizar que la aplicación funcione de manera confiable en cualquier situación.

Finalmente mirando hacia el futuro, se han contemplado varias mejoras y ampliaciones. Entre ellas, se destaca la posibilidad de desarrollar una versión optimizada para dispositivos móviles (APK) y la integración de un chatbot con asistencia de inteligencia artificial. Este asistente podría ayudar a describir imágenes, facilitar el uso de la aplicación y hasta verificar la veracidad de la información compartida, aportando una capa adicional de interactividad y soporte.

En resumen definitivo, esta app representa mi esfuerzo por crear una red social unificada, diseñada para mejorar la experiencia del usuario y facilitar la comunicación en un mundo digital cada vez más fragmentado. Con una base tecnológica sólida, un enfoque en la seguridad y un diseño centrado en la usabilidad, este proyecto aspira a ser una herramienta completa y accesible, capaz de integrar todas las funciones esenciales en un único y coherente entorno.

II. OBJETIVO GENERAL

El objetivo general de este proyecto es desarrollar una red social unificada que integre las funcionalidades más populares de diversas plataformas, permitiendo a los usuarios interactuar y compartir contenido desde una única plataforma.

III. OBJETIVOS ESPECÍFICOS

1. Implementar funciones clave como registro de usuarios, envío de mensajes, realizar publicaciones y subida de archivos de imagen.
2. Diseñar una interfaz intuitiva utilizando Bootstrap , CSS3 y Tailwind, que garantice una experiencia de usuario eficiente en dispositivos móviles y de escritorio.
3. Desarrollar el servidor y las bases de datos utilizando PHP para gestionar de manera eficaz las interacciones de los usuarios, almacenamiento de datos y manejo de contenido multimedia.
4. Configurar un servidor web basado en una RaspBerry PI 4 con linux Ubuntu 24.04 y que la plataforma esté disponible a través de cualquier dispositivo conectado a una misma red.
5. Implementar medidas de seguridad adecuadas para proteger los datos de los usuarios y garantizar la privacidad de sus interacciones, desde inyecciones SQL hasta accesos no autorizados en diversos puntos de la plataforma e incluso la base de datos del servidor.

IV. METODOLOGÍA

Se ha seguido un enfoque de desarrollo ágil dividido en varias etapas:

1. Investigación y planificación: se han revisado redes sociales existentes para identificar funcionalidades clave que serían integradas en la propuesta. Además de ello se ha planificado la arquitectura del sistema, tanto a nivel de front-end como de back-end.
2. Desarrollo del Front-end: se han implementado las interfaces de usuario usando HTML5, CSS3, JavaScript y Bootstrap. Se han diseñado las páginas de registro,

inicio de sesión y una página global donde se encuentran todas las funcionalidades de la red social.

3. Desarrollo del Back-end: se ha utilizado PHP para la gestión de usuarios, almacenamiento de publicaciones, mensajes y videos. Se conectó la aplicación a una base de datos MySQL para almacenar y gestionar datos.
4. Implementación del servidor web: se configuró un servidor web en Apache dentro de una Raspberry Pi 4 con sistema operativo Linux Ubuntu 24.04, asegurando que la aplicación estuviera en línea y accesible mediante un dominio personalizado.
5. Pruebas y ajustes: se han realizado pruebas de funcionalidad, usabilidad y rendimiento, y también se han corregido errores y se han realizado optimizaciones en la plataforma.

V. PLANIFICACIÓN TEMPORAL Y EVALUACIÓN DE COSTES

PLANIFICACIÓN TEMPORAL

El desarrollo se ha dividido en fases claramente definidas para garantizar un avance ordenado y facilitar la identificación y solución de posibles problemas en cada etapa. La siguiente es la planificación temporal:

Fase 1	Investigación y planificación	2 semanas
Fase 2	Diseño del Front-end (UI y experiencia de usuario)	3 semanas
Fase 3	Desarrollo del Back-end (PHP y MySQL)	3 semanas
Fase 4	Configuración del servidor web en Raspberry Pi 4	3 semanas
Fase 5	Pruebas y ajustes (Test de seguridad, usabilidad, y rendimiento)	4 semanas
Fase 6	Redacción y envío del trabajo final	A medida que se vaya avanzando el proyecto

- **Fase 1: Investigación y planificación**

Se ha realizado un análisis de las necesidades del proyecto, la identificación de las tecnologías a utilizar y la definición de los requerimientos funcionales y no funcionales.

- **Fase 2: Diseño del Front-end (UI y experiencia de usuario)**

En esta fase se desarrollaron prototipos y se diseñó la interfaz de usuario. Se trabajó en la experiencia de usuario, asegurando que la interfaz sea intuitiva y responsive, adaptándose tanto a dispositivos de escritorio como a móviles.

- **Fase 3: Desarrollo del Back-end (PHP y MySQL)**

Se implementó la lógica de negocio y se desarrollaron las funcionalidades esenciales, tales como el manejo de usuarios, publicaciones, mensajes y comentarios. Se implementaron las conexiones seguras a la base de datos mediante PDO.

- **Fase 4: Configuración del servidor web en Raspberry Pi**

Durante este período se instaló y configuró el sistema operativo (Ubuntu 24.04) en la Raspberry Pi, se configuró Apache, PHP y MariaDB, y se realizaron pruebas para asegurar que el servidor cumpliera con los requerimientos de rendimiento y seguridad.

- **Fase 5: Pruebas y ajustes (Test de seguridad, usabilidad y rendimiento)**

Se llevaron a cabo pruebas funcionales, de seguridad y de usabilidad para detectar y corregir posibles errores, optimizando la experiencia del usuario y la estabilidad del sistema.

- **Fase 6: Redacción y envío del trabajo final**

La documentación y redacción final se ha ido actualizando conforme se desarrollaban las distintas fases del proyecto, permitiendo reflejar de forma fiel y actualizada todo el proceso de desarrollo.

EVALUACIÓN DE COSTES

El desarrollo del proyecto ha requerido invertir en hardware específico para asegurar la operatividad y escalabilidad del sistema. Los principales costes son:

- **Raspberry Pi 4:** 120,84 €

La elección de la Raspberry Pi 4 se debe a su bajo consumo energético, su capacidad de procesamiento adecuada para el proyecto y la flexibilidad para configurarla como servidor web.

- **Tarjeta de memoria de 64GB (marca PNY):** 9,99 €

Esta tarjeta proporciona un espacio suficiente para almacenar datos, respaldar una base de datos con aproximadamente 20,000 usuarios, 60,000 publicaciones/mensajes/comentarios y 5,000 imágenes, asegurando que la infraestructura pueda crecer de forma controlada.

VI. FUNDAMENTOS TEÓRICOS

En esta sección se exponen las bases conceptuales y técnicas que sustentan el desarrollo de Bingo! Social Media App. Se abordan los principios fundamentales del diseño de aplicaciones web, la arquitectura cliente-servidor y las buenas prácticas de seguridad y usabilidad. Cada uno de estos pilares ha sido cuidadosamente seleccionado e implementado para garantizar una experiencia de usuario óptima, segura y escalable.

ARQUITECTURA CLIENTE-SERVIDOR

La arquitectura cliente-servidor es un modelo fundamental en el desarrollo de aplicaciones web modernas. Este modelo separa las responsabilidades entre el cliente (front-end) y el servidor (back-end), lo que permite distribuir las tareas de procesamiento y almacenamiento de manera eficiente. En Bingo! Social Media App, esta arquitectura se ha implementado de la siguiente manera:

- **Cliente front-end**: la interfaz de usuario ha sido diseñada para ser intuitiva y accesible. Utilizando frameworks como Bootstrap y Tailwind CSS, se ha garantizado un diseño responsivo que se adapta a cualquier dispositivo, ya sea un móvil, una tableta o un ordenador de escritorio. Este enfoque mobile-first asegura que todos los usuarios, independientemente de su dispositivo, tengan una experiencia coherente y agradable.
- **Servidor back-end**: el servidor maneja todas las solicitudes del cliente, procesando datos y realizando operaciones de almacenamiento. En este caso, se ha utilizado PHP con conexiones seguras a bases de datos MySQL/MariaDB mediante PDO. Este enfoque garantiza que el procesamiento de datos sea rápido y seguro, lo cual es esencial para mantener una experiencia de usuario ágil y permitir la escalabilidad del sistema según la demanda.

La separación entre cliente y servidor no solo mejora el rendimiento y la seguridad, sino que también facilita la escalabilidad del sistema. A medida que la base de usuarios crece, se pueden añadir más recursos al servidor sin afectar la experiencia del usuario en el cliente.

DISEÑO RESPONSIVE Y USABILIDAD

El diseño responsivo es crucial en el desarrollo de aplicaciones web modernas. Un enfoque mobile-first asegura que la aplicación esté optimizada para dispositivos móviles desde el inicio, en lugar de adaptarse después. En Bingo! Social Media App, se han utilizado frameworks como Bootstrap y Tailwind CSS para lograr una interfaz adaptable y amigable.

- **Bootstrap**: este popular framework CSS proporciona una serie de componentes y utilidades predefinidas que facilitan la creación de interfaces responsivas. Su sistema de grid permite organizar el contenido de manera flexible, adaptándose a diferentes tamaños de pantalla.
- **Tailwind CSS**: este framework de CSS utilitario ofrece una gran personalización y control sobre el diseño, permitiendo aplicar estilos directamente en los elementos HTML. La combinación de Bootstrap y Tailwind CSS nos ha permitido crear una interfaz de usuario coherente y adaptable, mejorando significativamente la usabilidad de la aplicación.

El enfoque en la usabilidad asegura que los usuarios puedan navegar y utilizar la aplicación de manera intuitiva. Se han realizado pruebas de usabilidad para identificar posibles mejoras y garantizar que la experiencia del usuario sea lo más fluida posible.

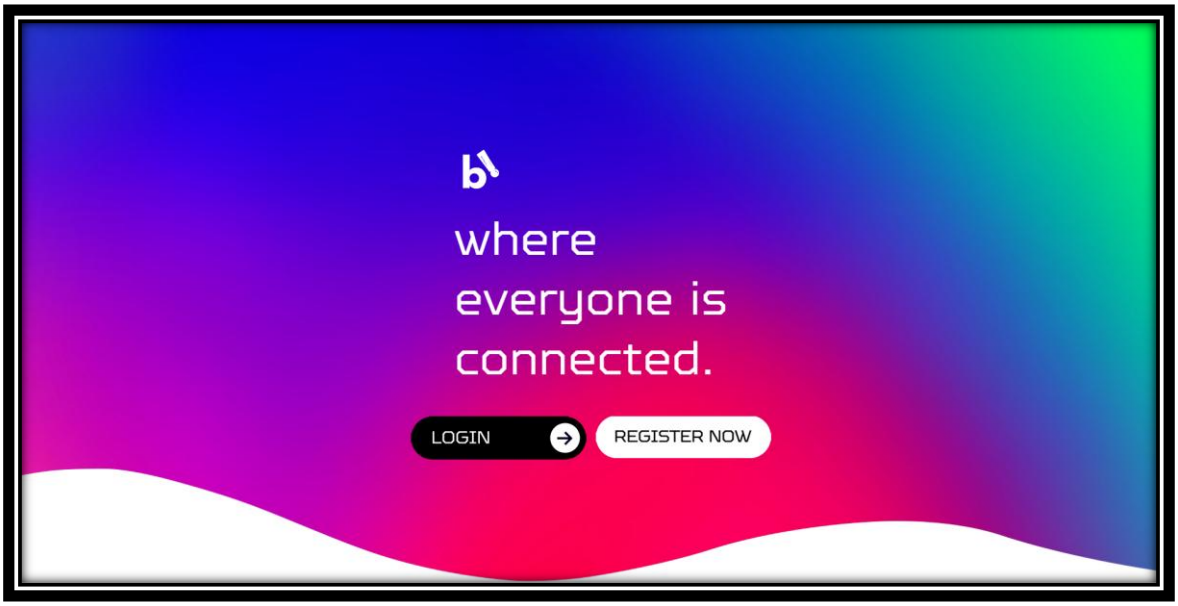

Ilustración 1: Página de inicio de la aplicación

SEGURIDAD EN APLICACIONES WEB

La seguridad es un aspecto crítico en el desarrollo de cualquier aplicación web. En el proyecto se han implementado diversas técnicas para proteger la información del usuario y prevenir ataques malintencionados:

- **Validación y Sanitización de Datos**: todos los datos ingresados por los usuarios son validados y sanitizados para evitar la inyección de código malicioso. Esto incluye la utilización de funciones como *htmlspecialchars()* en PHP para prevenir ataques de Cross-Site Scripting (XSS).
- **Conexiones Seguras**: se ha garantizado el uso de conexiones seguras (HTTPS) para proteger la transmisión de datos entre el cliente y el servidor. Además, las conexiones a la base de datos se manejan de manera segura utilizando PDO (PHP Data Objects), lo cual previene la inyección SQL.
- **Gestión de Sesiones**: la gestión adecuada de sesiones es fundamental para la autenticación y la protección de las cuentas de usuario. Se han implementado mecanismos de inicio y cierre de sesión seguros, asegurando que solo usuarios autenticados puedan acceder a sus datos personales.

Estas medidas de seguridad no solo protegen la información del usuario, sino que también aumentan la confianza en la aplicación, lo que es esencial para su éxito a largo plazo.

PRINCIPIOS DE DESARROLLO ÁGIL

La metodología ágil ha sido clave en el desarrollo. Este enfoque nos ha permitido iterar rápidamente sobre el diseño y la implementación, integrando feedback continuo de los usuarios y facilitando ajustes en función de las pruebas realizadas.

- **Iteración Rápida**: la metodología ágil se basa en ciclos de desarrollo cortos, llamados sprints, que permiten iterar rápidamente y realizar mejoras continuas.

Este enfoque ha sido esencial para responder de manera efectiva a las necesidades cambiantes de los usuarios y del proyecto.

- **Feedback Continuo**: integrar el feedback de los usuarios a lo largo del desarrollo ha permitido identificar y solucionar problemas de manera temprana. Las pruebas regulares y la comunicación constante con los usuarios han sido fundamentales para garantizar que el producto final cumpla con sus expectativas y necesidades.
- **Flexibilidad**: la metodología ágil ha permitido ser flexibles y adaptarnos a los cambios en los requisitos del proyecto. Esto ha sido especialmente importante en un entorno dinámico como el desarrollo de una aplicación de redes sociales, donde las tendencias y las necesidades de los usuarios pueden cambiar rápidamente.

VII. ANÁLISIS DE REQUISITOS

El análisis de requisitos se ha dividido en dos grandes categorías: funcionales y no funcionales.

REQUISITOS FUNCIONALES

- **Registro y autenticación de usuarios:**

Permitir el registro seguro, verificación de datos y autenticación mediante sesiones.

- **Gestión de publicaciones y hilos:**

Creación, edición y eliminación de publicaciones en formato de hilos, permitiendo respuestas y debates estructurados.

- **Mensajes directos:**

Implementación de un sistema de mensajería privada en tiempo real utilizando AJAX y la Fetch API.

- **Gestión de amigos:**

Funcionalidades para enviar, aceptar y gestionar solicitudes de amistad.

- **Comentarios y reacciones:**

Los usuarios pueden comentar publicaciones y reaccionar ante ellas, fomentando la interacción.

REQUISITOS NO FUNCIONALES

- **Seguridad:**

Protección contra inyecciones SQL, XSS y otros ataques, mediante el uso de PDO y sanitización de entradas.

- **Rendimiento y escalabilidad:**

La arquitectura debe soportar el crecimiento previsto (20,000 usuarios, 60,000 interacciones y 5,000 imágenes) sin pérdida de rendimiento.

- **Usabilidad:**

La interfaz debe ser intuitiva, accesible y responsive, facilitando la experiencia tanto para usuarios novatos como expertos.

- **Mantenibilidad:**

El código debe estar modularizado y documentado, permitiendo futuras ampliaciones y una fácil gestión de errores.

VIII. PROTOTIPOS DE INTERFACES

Antes de iniciar el desarrollo, se realizaron diversos prototipos de interfaces:

- **Prototipos en papel:**

Se esbozaron ideas y se definieron los flujos de interacción en papel, permitiendo visualizar la estructura de la aplicación y el recorrido del usuario.

- **Herramientas Digitales:**

Se utilizó la herramienta de Figma para diseñar versiones digitales de la interfaz, facilitando la experimentación y la obtención de feedback temprano. Estos prototipos ayudaron a definir el diseño de la interfaz y a identificar mejoras en la experiencia de usuario.

IX. DISEÑO

DISEÑO ARQUITECTÓNICO

El sistema se basa en la arquitectura cliente-servidor:

- **Front-end:**

Desarrollado en HTML5, CSS3 y JavaScript, se utiliza Bootstrap y Tailwind CSS para crear una interfaz dinámica y responsiva. Este enfoque asegura que la aplicación se adapte a distintos dispositivos y resoluciones.

- **Back-end:**

Utilizando PHP, se implementa la lógica de negocio y se gestionan las interacciones con la base de datos MySQL/MariaDB. El uso de PDO y consultas preparadas garantiza la seguridad de las operaciones.

- **Comunicación Asíncrona:**

El uso de AJAX y la Fetch API permite la actualización en tiempo real de la interfaz, sin necesidad de recargar la página, lo que mejora notablemente la experiencia del usuario.

DISEÑO DE DATOS

El modelo de datos se ha diseñado para gestionar eficazmente las interacciones y relaciones entre usuarios. A continuación se muestra el esquema entidad-relación (E/R) utilizado para crear la base de datos:

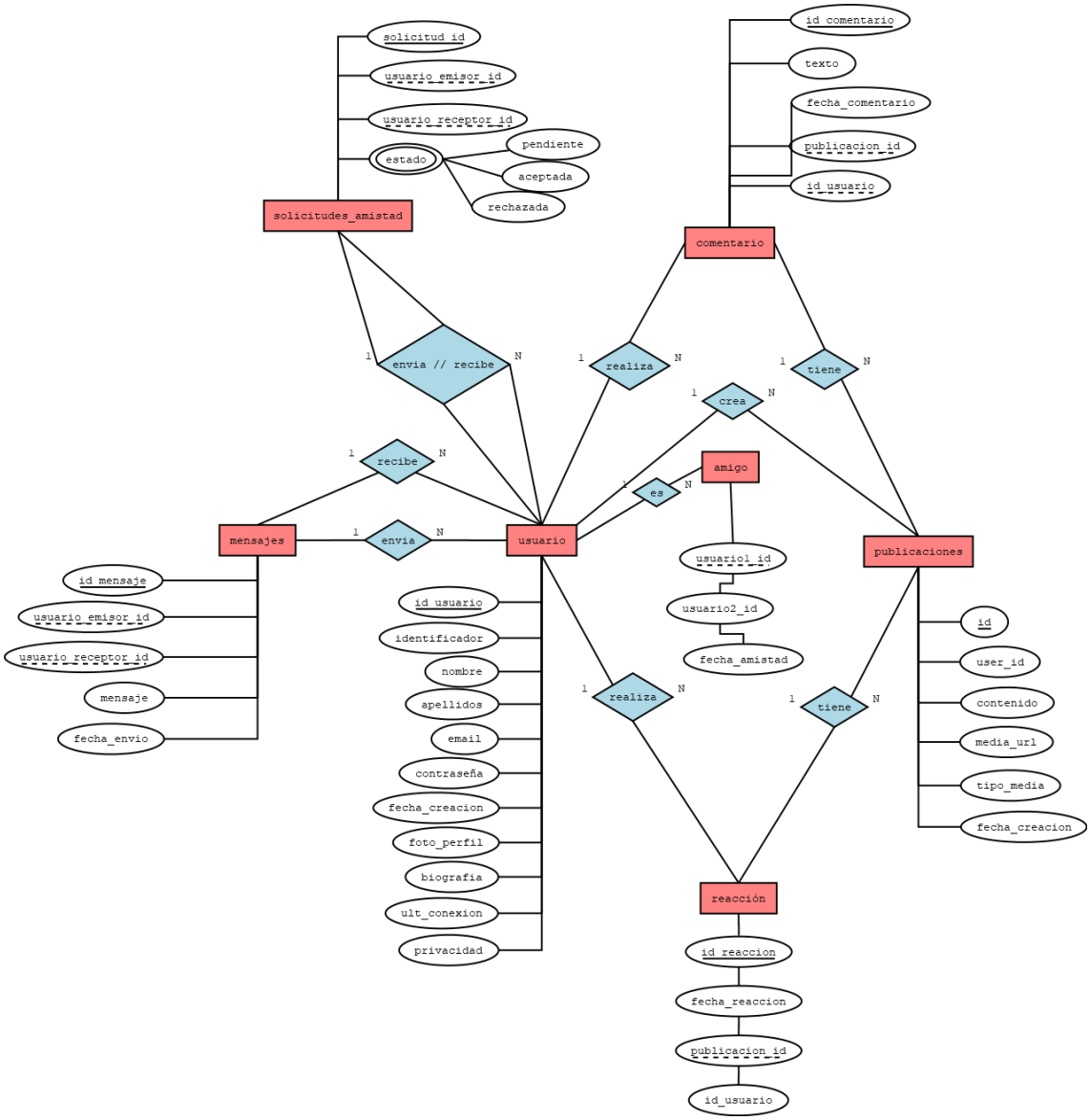

Ilustración 2: Diseño del esquema Entidad/Relación de la BBDD

- **Usuarios y publicaciones:** un usuario puede tener múltiples publicaciones.
- **Usuarios y amigos:** se establece una relación bidireccional que permite gestionar las solicitudes y confirmaciones de amistad.
- **Publicaciones, comentarios y reacciones:** cada publicación puede tener varios comentarios y reacciones, facilitando la interacción.
- **Mensajes directos:** se registra la comunicación entre dos usuarios, permitiendo el seguimiento de conversaciones.

Tras este modelo entidad-relación, se presentará un esquema relacional elaborado con la herramienta Dia, donde se representan gráficamente las entidades, atributos y relaciones definidas en nuestra base de datos.

USUARIO (id_usuario, identificador, nombre, apellidos, email, contraseña, fecha_creacion, foto_perfil, biografia, ult_conexion, privacidad)

AMIGOS (**usuario1_id**, **usuario2_id**, fecha_amistad)

PUBLICACIONES (id, **user_id**, contenido, media_url, tipo_media, fecha_creacion)

COMENTARIO (id_comentario, texto, fecha_comentario, **publicacion_id**, **id_usuario**)

MENSAJES (id_mensaje, **usuario_emisor_id**, **usuario_receptor_id**, mensaje, fecha_envio)

REACCION (id_reaccion, fecha_reaccion, **publicacion_id**, **id_usuario**)

SOLICITUDES_AMISTAD (solicitud_id, **usuario_emisor_id**, **usuario_receptor_id**, estado)

Por último, se muestra el esquema DDL (Data Definition Language) de la base de datos. Este esquema proporciona una visión detallada de la estructura y definición de las tablas que componen nuestra base de datos bingo_db, permitiendo comprender cómo están organizados y relacionados los datos dentro del proyecto.

```
CREATE DATABASE IF NOT EXISTS bingo_db;
USE bingo_db;

CREATE TABLE usuario (
  id_usuario INT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
  identificador VARCHAR(20) CHARACTER SET utf8mb4 COLLATE
utf8mb4_general_ci NOT NULL,
  nombre VARCHAR(30) CHARACTER SET utf8mb4 COLLATE utf8mb4_general_ci,
  apellidos VARCHAR(40) CHARACTER SET utf8mb4 COLLATE utf8mb4_general_ci,
  email VARCHAR(50) CHARACTER SET utf8mb4 COLLATE utf8mb4_general_ci,
  contraseña VARCHAR(255) CHARACTER SET utf8mb4 COLLATE
utf8mb4_general_ci,
  fecha_creacion DATE,
  foto_perfil VARCHAR(255) CHARACTER SET utf8mb4 COLLATE
utf8mb4_general_ci,
  biografia VARCHAR(200) CHARACTER SET utf8mb4 COLLATE
utf8mb4_general_ci,
  ult_conexion DATETIME DEFAULT NULL,
  privacidad ENUM('privado','publico') CHARACTER SET utf8mb4 COLLATE
utf8mb4_general_ci NOT NULL DEFAULT 'publico'
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;
```

```
CREATE TABLE amigos (
  usuario1_id INT UNSIGNED NOT NULL,
  usuario2_id INT UNSIGNED NOT NULL,
  fecha_amistad DATETIME DEFAULT CURRENT_TIMESTAMP,
  PRIMARY KEY (usuario1_id, usuario2_id),
  CONSTRAINT fk_amigos_1 FOREIGN KEY (usuario1_id) REFERENCES usuario
(id_usuario) ON DELETE CASCADE,
  CONSTRAINT fk_amigos_2 FOREIGN KEY (usuario2_id) REFERENCES usuario
(id_usuario) ON DELETE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;

CREATE TABLE publicaciones (
  id INT AUTO_INCREMENT PRIMARY KEY,
  user_id INT UNSIGNED NOT NULL,
  contenido TEXT CHARACTER SET utf8mb4 COLLATE utf8mb4_general_ci NOT
NULL,
  media_url VARCHAR(255) CHARACTER SET utf8mb4 COLLATE utf8mb4_general_ci
DEFAULT NULL,
  tipo_media ENUM('image','video') CHARACTER SET utf8mb4 COLLATE
utf8mb4_general_ci DEFAULT NULL,
  fecha_creacion DATETIME NOT NULL,
  CONSTRAINT fk_publicaciones FOREIGN KEY (user_id) REFERENCES usuario
(id_usuario) ON DELETE CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;

CREATE TABLE comentario (
  id_comentario INT AUTO_INCREMENT PRIMARY KEY,
  texto VARCHAR(100) CHARACTER SET utf8mb4 COLLATE utf8mb4_general_ci
DEFAULT NULL,
  fecha_comentario DATE DEFAULT NULL,
  publicacion_id INT NOT NULL,
  id_usuario INT UNSIGNED NOT NULL,
  CONSTRAINT fk_comentario_publicacion FOREIGN KEY (publicacion_id)
REFERENCES publicaciones (id) ON DELETE CASCADE,
  CONSTRAINT fk_comentario_usuario FOREIGN KEY (id_usuario) REFERENCES
usuario (id_usuario)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;

CREATE TABLE mensajes (
  id_mensaje INT AUTO_INCREMENT PRIMARY KEY,
  usuario_emisor_id INT UNSIGNED NOT NULL,
  usuario_receptor_id INT UNSIGNED NOT NULL,
  mensaje TEXT CHARACTER SET utf8mb4 COLLATE utf8mb4_general_ci NOT NULL,
  fecha_envio DATETIME DEFAULT CURRENT_TIMESTAMP,
  CONSTRAINT fk_mensajes_1 FOREIGN KEY (usuario_emisor_id) REFERENCES
usuario (id_usuario) ON DELETE CASCADE,
```

```
  CONSTRAINT fk_mensajes_2 FOREIGN KEY (usuario_receptor_id) REFERENCES
usuario (id_usuario) ON DELETE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;

CREATE TABLE reaccion (
  id_reaccion INT AUTO_INCREMENT PRIMARY KEY,
  fecha_reaccion DATE DEFAULT NULL,
  publicacion_id INT NOT NULL,
  id_usuario INT UNSIGNED NOT NULL,
  CONSTRAINT fk_reaccion_publicacion FOREIGN KEY (publicacion_id)
REFERENCES publicaciones (id) ON DELETE CASCADE,
  CONSTRAINT fk_reaccion_usuario FOREIGN KEY (id_usuario) REFERENCES
usuario (id_usuario)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;

CREATE TABLE solicitudes_amistad (
  solicitud_id INT AUTO_INCREMENT PRIMARY KEY,
  usuario_emisor_id INT UNSIGNED NOT NULL,
  usuario_receptor_id INT UNSIGNED NOT NULL,
  estado ENUM('pendiente','aceptada','rechazada') CHARACTER SET utf8mb4
COLLATE utf8mb4_general_ci DEFAULT 'pendiente',
  CONSTRAINT fk_solicitudes_amistad_1 FOREIGN KEY (usuario_emisor_id)
REFERENCES usuario (id_usuario),
  CONSTRAINT fk_solicitudes_amistad_2 FOREIGN KEY (usuario_receptor_id)
REFERENCES usuario (id_usuario)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;
```

X. CODIFICACIÓN

La codificación del proyecto se ha basado en:

- **Buenas prácticas de programación:**

Uso de código modular, comentarios claros y separación de responsabilidades entre el front-end y el back-end.

- **Seguridad:**

Validación de datos, uso de consultas preparadas con PDO y gestión adecuada de sesiones.

- **Eficiencia:**

Optimización de consultas a la base de datos y utilización de técnicas asíncronas (AJAX y Fetch API) para minimizar la latencia.

El código fuente se ha estructurado en módulos que facilitan el mantenimiento y la incorporación de nuevas funcionalidades sin afectar la estabilidad general del sistema.

ENTORNO DE PROGRAMACIÓN

El entorno de desarrollo ha sido configurado para maximizar la productividad y asegurar la calidad del código:

- **Sistema operativo:** Ubuntu 24.04 en una RaspBerry PI 4.
- **Editor de código:** Visual Studio Code, configurado para trabajar remotamente mediante SSH.
- **Control de versiones:** Git, que ha permitido gestionar de forma ordenada los cambios a lo largo del desarrollo.
- **Plataforma de colaboración:** GitHub, aprovechando herramientas como GitHub Copilot para asistencia en la codificación.

LENGUAJES Y HERRAMIENTAS

- **Frontend:**
  - HTML5, CSS3 y JavaScript.
  - Frameworks de diseño como Bootstrap 5.3.0-alpha1 y Tailwind CSS.
  - Librerías de iconografía: Font Awesome 6.1.1 y Bootstrap Icons 1.11.3.
- **Backend:**
  - PHP (Vanilla) para la lógica.
  - MySQL/MariaDB, gestionado mediante PDO.
- **Comunicación Asíncrona:**
  - AJAX y Fetch API para actualizar contenidos sin recargar la página.
- **Hardware:**
  - RaspBerry PI 4 configurada con Ubuntu 24.04, que proporciona un servidor eficiente y escalable.

ASPECTOS RELEVANTES DE LA IMPLEMENTACIÓN

La implementación del proyecto ha sido llevada a cabo utilizando un conjunto de frameworks y tecnologías que han permitido desarrollar una aplicación robusta, segura y escalable. A continuación se describen en detalle los aspectos más importantes:

a) Frameworks y tecnologías principales:
   a. Frontend:
      i. Bootstrap 5.3.0-alpha1, que proporciona una amplia gama de componentes predefinidos y un sistema de grid que facilita la creación de layouts responsivos y modernos.
      ii. Tailwind CSS, que ofrece clases utilitarias que permiten un control detallado del diseño, espaciado y flexbox, complementando a Bootstrap para obtener una interfaz personalizable.

iii. Font Awesome 6.1.1 y Bootstrap Icons 1.11.3, que permiten la integración de iconografía de alta calidad, mejorando la estética y usabilidad de la aplicación.

b. Backend:
   i. PHP vanilla, donde se ha empleado para implementar la lógica de negocio y el procesamiento del lado del servidor, garantizando un desarrollo ágil y flexible.
   ii. MariaDB con PDO para conexiones, que es el sistema de gestión de bases de datos elegido por su robustez, seguridad y facilidad para establecer conexiones seguras mediante PDO, lo que permite la prevención de inyecciones SQL.

c. JavaScript:
   i. Vanilla JavaScript (ES6+), que se utiliza para implementar funcionalidades interactivas y dinámicas en el lado del cliente.
   ii. AJAX y Fetch API, que permiten la comunicación asíncrona entre el cliente y el servidor, lo que facilita la actualización en tiempo real de los contenidos sin recargar la página.

b) Ejemplos de uso y funcionalidad.
   a. Bootstrap y Tailwind CSS

Estos frameworks se combinan para crear interfaces responsivas y estéticamente atractivas.

```
  <!-- Bootstrap CSS -->
  <link
    href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-
alpha1/dist/css/bootstrap.min.css"
    rel="stylesheet"
  >

  <!-- Tailwind CSS CDN -->
  <script src="https://cdn.tailwindcss.com"></script>

  <link rel="stylesheet" href="styles/style.css">
</head>
<body>
  <!-- HERO SECTION -->
  <div class="hero-section d-flex flex-column items-center justify-
content-center text-center vh-100 h-screen relative">
    <video autoplay muted loop id="hero-video" class="video-bg absolute
top-0 left-0 w-full h-full object-cover opacity-100">
      <source src="src/img/bg-animated.mp4" type="video/mp4">
    </video>

    <div class="titulo-logo z-10">
      <div class="logo mb-4">
        <img src="src/img/logo.png" alt="bingo! logo" class="logo-img
max-w-[80px]">
```

```
      </div>
      <div class="principal mb-5">
        <h1 class="titulo-inicio text-white text-6xl md:text-[80px]
leading-tight">
          where <br> everyone is <br> connected.
        </h1>
      </div>
    </div>

    <div class="botones d-flex justify-content-center z-10">
```

- Utilización combinada para crear layouts responsivos.
- Clases utilitarias para espaciado y flexbox.
- Sistema de grid para organización de contenido.

b. PHP con PDO

Se utiliza para gestionar conexiones seguras a la base de datos y prevenir vulnerabilidades.

```
<?php
    $host = 'localhost';
    $dbname = 'bingo_db';
    $username = 'bingo-adm';
    $password = 'C0ntr4$3ñ4';

    try {
        $dsn = "mysql:host=$host;dbname=$dbname;charset=utf8mb4"; // Se
agrega charset=utf8mb4
        $options = [
            PDO::ATTR_ERRMODE            => PDO::ERRMODE_EXCEPTION,
            PDO::ATTR_DEFAULT_FETCH_MODE => PDO::FETCH_ASSOC,
        ];
        $pdo = new PDO($dsn, $username, $password, $options);
    } catch (PDOException $e) {
        die("Error en la conexión: " . $e->getMessage());
    }
?>
```

- Manejo seguro de conexiones a base de datos.
- Preparación de consultas para prevenir SQL injection.
- Gestión de errores mediante try-catch.

c. AJAX y Fetch API

Permiten implementar funcionalidades de actualización en tiempo real.

```
document.addEventListener('DOMContentLoaded', function () {
    // Búsqueda de usuarios
    searchInput.addEventListener('input', function () {
        const searchQuery = this.value.trim();

        if (searchQuery.length > 0) {
            fetch(`../php/search.php?query=${encodeURIComponent(searchQue
ry)}`)
                .then(response => {
                    if (!response.ok) {
                        throw new Error('Error en la solicitud de
búsqueda');
                    }
                    return response.json();
                })
                .then(data => {
                    searchResultsBox.innerHTML = '';

                    if (data.length > 0) {
                        data.forEach(user => {
                            if (user.id_usuario ===
parseInt(currentUserId)) return;

                            const userItem =
document.createElement('li');
                            userItem.classList.add('search-result-item');

                            userItem.innerHTML = `
                                <img class="result-profile-pic"
src="${user.foto_perfil ? user.foto_perfil : '../src/img/default-
avatar.png'}" alt="Foto de ${user.nombre}">
                                <span>${user.nombre} <span style="color:
grey;">@${user.identificador}</span></span>
                            `;

                            userItem.addEventListener('click', () => {
                                showUserProfile(user.id_usuario);
                                searchResultsBox.style.display = 'none';
                                searchInput.value = '';
                            });

                            searchResultsBox.appendChild(userItem);
                        });
```

- Búsqueda de usuarios en tiempo real.
- Actualizaciones de interfaz sin recargar la página.
- Manejo de solicitudes asíncronas.

d. Sistema de subida de archivos multimedia

Implementa la subida y validación de imágenes de forma segura.

```php
if (isset($_FILES['foto_perfil']) && $_FILES['foto_perfil']['error'] == UPLOAD_ERR_OK) {
        $fileTmpPath = $_FILES['foto_perfil']['tmp_name'];
        $fileName = $_FILES['foto_perfil']['name'];
        $fileNameCmps = explode(".", $fileName);
        $fileExtension = strtolower(end($fileNameCmps));

        // Extensiones permitidas
        $allowedfileExtensions = ['jpg', 'jpeg', 'png'];

        if (in_array($fileExtension, $allowedfileExtensions)) {
            // Definir la ruta de destino donde guardar la imagen en el servidor local
            $uploadFileDir = '../uploads/';
            $newFileName = md5(time() . $fileName) . '.' . $fileExtension; // Crear un nombre único para evitar conflictos
            $dest_path = $uploadFileDir . $newFileName;

            // Crear el directorio si no existe
            if (!is_dir($uploadFileDir)) {
                mkdir($uploadFileDir, 0777, true);
            }

            // Mover el archivo subido a la carpeta de destino
            if (move_uploaded_file($fileTmpPath, $dest_path)) {
                $fotoPerfil = '../uploads/' . $newFileName; // Guardar la ruta relativa para la base de datos
            } else {
                echo "Hubo un error al mover el archivo de imagen.";
            }
        } else {
            echo "Solo se permiten archivos con extensión JPG, JPEG, o PNG.";
        }
    }
```

- Gestión segura de subida de archivos.
- Validación de tipos MIME.
- Procesamiento de imágenes con Cropper.js

e. Cropper

Permite al usuario recortar una imagen antes de subirla.

```
    <script>
        let cropper;
        let croppedImageBlob = null; // Variable global para almacenar el
blob de la imagen recortada

        function showCropper(event) {
            const file = event.target.files[0];
            if (file) {
                const reader = new FileReader();
                reader.onload = function (e) {
                    const imageToCrop =
document.getElementById('imageToCrop');
                    imageToCrop.src = e.target.result;
                    document.getElementById('cropperModal').style.display
= 'block';

                    if (cropper) {
                        cropper.destroy();
                    }
                    cropper = new Cropper(imageToCrop, {
                        aspectRatio: 1,
                        viewMode: 1,
                    });
                };
                reader.readAsDataURL(file);
            }
        }

        function closeCropperModal() {
            document.getElementById('cropperModal').style.display =
'none';
            if (cropper) {
                cropper.destroy();
            }
        }

        document.getElementById('cropImageButton').addEventListener('clic
k', function () {
            if (cropper) {
                const canvas = cropper.getCroppedCanvas({
                    width: 200,
                    height: 200,
                });
```

```
                const profilePicPreview =
document.getElementById('profilePicPreview');
                profilePicPreview.src = canvas.toDataURL();

                canvas.toBlob((blob) => {
                    croppedImageBlob = blob; // Almacenar el blob en la
variable global
                    closeCropperModal();
                });
            }
        });

        document.querySelector('form').addEventListener('submit',
function (event) {
            event.preventDefault(); // Evitar el envío normal del
formulario

            const formData = new FormData(this); // 'this' hace
referencia al formulario
            if (croppedImageBlob) {
                formData.set('foto_perfil', croppedImageBlob,
'croppedImage.png');
            }

            // Enviar el formulario mediante AJAX
            fetch('completar_perfil.php?user_id=<?=
htmlspecialchars($userId) ?>', {
                method: 'POST',
                body: formData
            })
            .then(response => {
                if (response.ok) {
                    window.location.href = "../main/index.php";
                } else {
                    console.error('Error al actualizar el perfil');
                }
            })
            .catch(error => {
                console.error('Error de red:', error);
            });
        });
    </script>
```

f. Google OAuth

Integración del inicio de sesión con una cuenta de Google usando librerías de Composer y Node.js y modificando el código PHP para distinguir entre un registro y un inicio de sesión con esta puesta en marcha de servicio de login de la marca.

```php
<?php

// Configuración del Cliente de Google

$client = new Google_Client();
$client->setClientId('567096029490-
5sbh7rc3i309di4dn46geg702a1gstpi.apps.googleusercontent.com');
$client->setClientSecret('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');
$client->setRedirectUri('http://app.bingo.es/auth.php');
$client->addScope(['email', 'profile']);

// Manejo del Código de Autenticación de Google


if (isset($_GET['code'])) {
    error_log("Google code received: " . $_GET['code']);

    $token = $client->fetchAccessTokenWithAuthCode($_GET['code']);
    if (!isset($token['error'])) {
        $client->setAccessToken($token);
        $oauth = new Google_Service_Oauth2($client);
        $googleUser = $oauth->userinfo->get();

        $email = $googleUser->email;
        $name = $googleUser->name;
        $identificador = strtolower(str_replace(' ', '', $name));

        error_log("Google user info: email=" . $email . ", name=" . $name
. ", generated id=" . $identificador);

    } else {
        error_log("Google token error: " . json_encode($token));
        $login_errors[] = "Error al obtener el token de Google.";
    }
}

// Generación de la URL de Autenticación de Google

$googleLoginUrl = $client->createAuthUrl();
?>

<!-- // Botón de Inicio de Sesión con Google en el Formulario HTML -->
```

```
<div class="social-icons">
    <a href="<?= htmlspecialchars($googleLoginUrl) ?>" class="icon">
        <img src="src/icons/google.webp" alt="Login con Google">
    </a>
</div>
```

XI. CONFIGURACIÓN DE LA RASPBERRY PI 4 CON UBUNTU 24.04

PREPARACIÓN DEL SISTEMA OPERATIVO

El primer paso es la descarga y flasheo de la imagen de Ubuntu 24.04 para RaspBerry PI: se obtiene la imagen oficial de Ubuntu 24.04 desde el sitio web de Ubuntu para RaspBerry PI y utilizando la herramienta balenaEtcher se graba la imagen en la tarjeta de memoria de 64GB.

CONFIGURACIÓN INICIAL DE LA RASPBERRY PI 4

Una vez flasheada la tarjeta, se procede a realizar los siguientes pasos:

- Primer arranque: conectar la Raspberry Pi a un monitor, teclado y ratón, y encender el dispositivo.
- Configuración básica: seleccionar el idioma, la zona horaria, configurar la red y crear un usuario administrador.
- Actualización del sistema: ejecutar comandos para actualizar el sistema:

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

CONFIGURACIÓN DEL ENTORNO DE DESARROLLO

Para preparar el entorno de desarrollo se instalan los paquetes y dependencias necesarios:

```
sudo apt install apache2 php libapache2-mod-php php-mysql mariadb-server
-y
```

- Configuración de seguridad en MariaDB: ejecutar sudo mysql_secure_installation para asegurar la base de datos.
    - Usuario bbdd: **bingo-adm**
    - Contraseña segura: **C0ntr4$3ñ4**
- Configuración de red y acceso remoto: se configura el firewall (UFW) y se habilita el acceso remoto mediante SSH:

```
sudo ufw allow OpenSSH
sudo ufw allow 'Apache Full'
sudo ufw enable
```

PANEL DE CONTROL DEL SERVIDOR

A continuación se muestra cómo se ha integrado un panel de control para el servidor de la RaspBerry, diseñado para que, en caso de conectar una pantalla al servidor, se abra automáticamente una ventana del navegador que muestre información relevante sobre la red y el estado de los servicios. Este script se ejecuta al iniciar sesión y se encarga de arrancar servicios esenciales (MySQL, Apache2 y SSH), activar un entorno virtual de Python y lanzar una aplicación Flask que renderiza la página de control.

El siguiente script se ejecuta automáticamente al iniciar sesión en el sistema y realiza los siguientes pasos:

1. Inicia los servicios de MySQL, Apache2 y SSH.
2. Activa el entorno virtual de Python donde se encuentra la aplicación del panel de control.
3. Ejecuta la aplicación Flask, la cual abre una ventana en el navegador apuntando a 127.0.0.1:5000.

Este es el contenido de *script_inicio.sh* que se encuentra en la raíz del usuario:

```
systemctl start mysql
systemctl start apache2
systemctl start ssh

source ~/Desktop/panel_control/venv/bin/activate
python ~/Desktop/panel_control/panel_control.py

./Desktop/panel_control/templates/panel.html
```

La estructura de directorios del panel de control es la siguiente:

```
bingo-adm@bingoadm-server:~/Desktop$ ls -l panel_control/
total 12
-rw-r--r-- 1 root      root      1473 Mar 12 22:34 panel_control.py
drwxrwxr-x 2 bingo-adm bingo-adm 4096 Mar 12 22:32 templates
drwxr-xr-x 5 bingo-adm bingo-adm 4096 Nov 12 09:02 venv
```

En ella, se puede observar:

- **panel_control.py:** script principal de la aplicación Flask.
- **templates/**: directorio que contiene la plantilla HTML (panel.html).
- **venv/**: entorno virtual de Python con las dependencias necesarias.

Código de la aplicación Flask de *panel_control.py*:

```
from flask import Flask, render_template
import psutil
import subprocess
import socket
import netifaces

app = Flask(__name__)

def verificar_servicio(nombre):
    try:
        resultado = subprocess.run(["systemctl", "is-active", nombre],
stdout=subprocess.PIPE)
        estado = resultado.stdout.decode().strip()
        return estado == "active"
    except Exception as e:
        print(f"Error al verificar el servicio {nombre}: {e}")
        return False

def obtener_info_red():
    interfaz = netifaces.gateways()['default'][netifaces.AF_INET][1]  #
Obtener la interfaz principal
    info = netifaces.ifaddresses(interfaz)

    ipv4 = info[netifaces.AF_INET][0]['addr']
    mascara = info[netifaces.AF_INET][0]['netmask']

    dns = []
    with open("/etc/resolv.conf", "r") as file:
        for line in file:
            if line.startswith("nameserver"):
                dns.append(line.split()[1])

    return ipv4, mascara, dns

@app.route('/')
def panel():
    estado_apache = "Activo" if verificar_servicio("apache2") else
"Inactivo"
    estado_mysql = "Activo" if verificar_servicio("mysql") else
"Inactivo"
    estado_sftp = "Activo" if verificar_servicio("ssh") else "Inactivo"

    ipv4, mascara, dns = obtener_info_red()
```

```
    return render_template("panel.html", estado_apache=estado_apache,
estado_mysql=estado_mysql, estado_sftp=estado_sftp, ipv4=ipv4,
mascara=mascara, dns=dns)

if __name__ == '__main__':
    app.run(host='0.0.0.0', port=5000)
```

Funcionamiento:

- Verificación de Servicios: la función verificar_servicio utiliza systemctl para comprobar si un servicio (como Apache2, MySQL o SSH) se encuentra activo.
- Obtención de Información de Red: la función obtener_info_red utiliza el módulo netifaces para extraer la dirección IPv4, la máscara de red y los servidores DNS configurados en el archivo /etc/resolv.conf.
- Ruta Principal: la ruta / renderiza la plantilla panel.html, pasando información sobre el estado de los servicios y los datos de red.

XII. DESARROLLO REMOTO CON VISUAL STUDIO CODE

Visual Studio Code se ha configurado para trabajar en la Raspberry Pi a través de SSH, permitiendo editar el código directamente en el servidor remoto sin necesidad de copiar archivos manualmente.

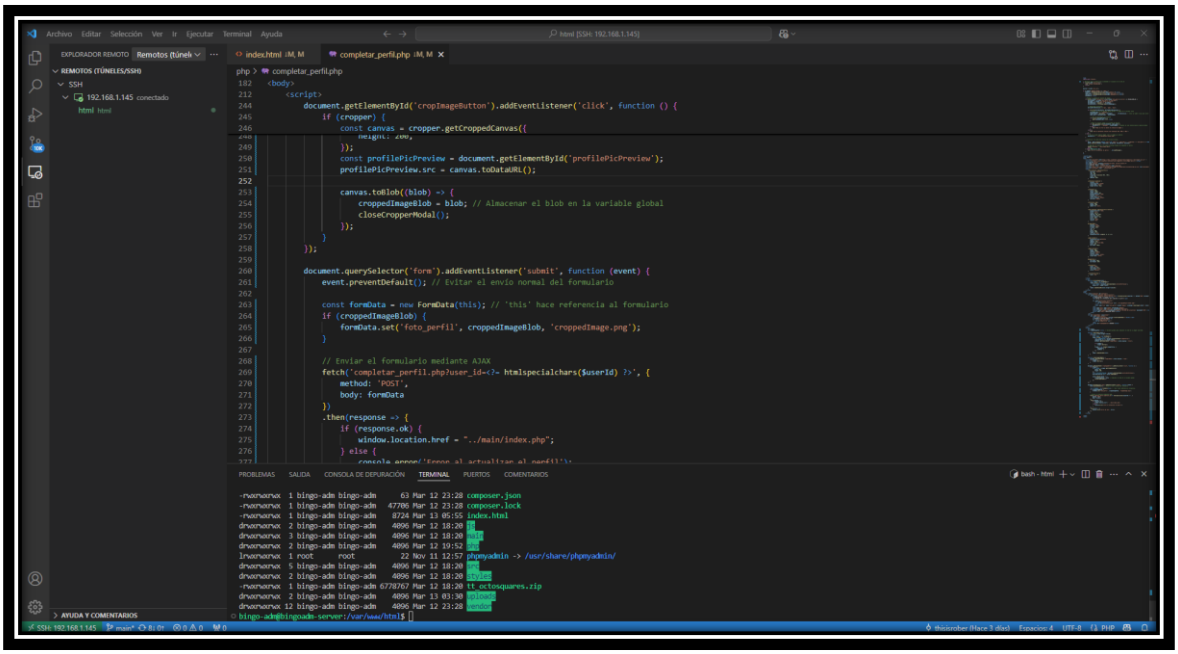

Ilustración 3: Vista del editor Visual Studio Code conectado a RaspBerry PI 4 vía SSH

USO DE GITHUB COPILOT, BUENAS PRÁCTICAS EN EL USO DE LA IA E IMPORTANCIA DE LA INTERVENCIÓN HUMANA

La integración de GitHub Copilot en el desarrollo de este proyecto se ha realizado de manera consciente y estratégica, aprovechando al máximo sus capacidades de

inteligencia artificial para complementar el trabajo manual y experto de programación. Esta herramienta no ha venido a reemplazar la intervención humana, sino a actuar como un asistente que aporta sugerencias valiosas en tareas complejas y cotidianas, lo que ha permitido identificar patrones comunes y proponer soluciones eficientes que han contribuido significativamente al éxito del proyecto.

GitHub Copilot ha sido utilizado para optimizar funciones críticas y acelerar el proceso de codificación, pero siempre bajo una revisión y validación cuidadosa por mi parte (es decir, una intervención humana). Este enfoque ha garantizado que cada sugerencia se ajuste a las necesidades específicas del proyecto, manteniendo la robustez y calidad del código. ¿Quiere decir esto que el proyecto ha sido realizado completamente por la inteligencia artificial? **Evidentemente no**.

Gracias a esta herramienta, se ha logrado implementar una programación exhaustiva y compleja, alcanzando niveles de eficiencia y sofisticación que rara vez se observan en otros proyectos de fin de grado. Con esto quiero decir que a pesar de buenas prácticas de programación y la limitación de las capacidades como humano que tengo, hoy en día la inteligencia artificial se ha convertido en una parte fundamental que es prácticamente imprescindible en cualquier cosa, ya sea programar o estudiar. Y para este proyecto, evidentemente ha sido muy útil para realizar ya sea apartados de documentación, código, ayuda y revisión de errores, y mucho más.

En conjunto con otras herramientas de inteligencia artificial, como ChatGPT, GitHub Copilot ha permitido superar retos técnicos significativos, facilitando la resolución de problemas y la integración de nuevas funcionalidades de manera ágil. Este uso consciente de la IA ha sido decisivo para llevar a cabo un desarrollo innovador, demostrando que, cuando se aplica de forma crítica y responsable, la inteligencia artificial puede ser un aliado indispensable en la creación de soluciones tecnológicas de alta calidad.

XIII. PRUEBAS DE EJECUCIÓN

PRUEBAS FUNCIONALES

Se han realizado pruebas exhaustivas para comprobar que cada funcionalidad (registro, publicaciones, mensajes, gestión de amigos y comentarios) funciona según lo previsto. Se han simulado múltiples escenarios de uso para detectar posibles fallos.

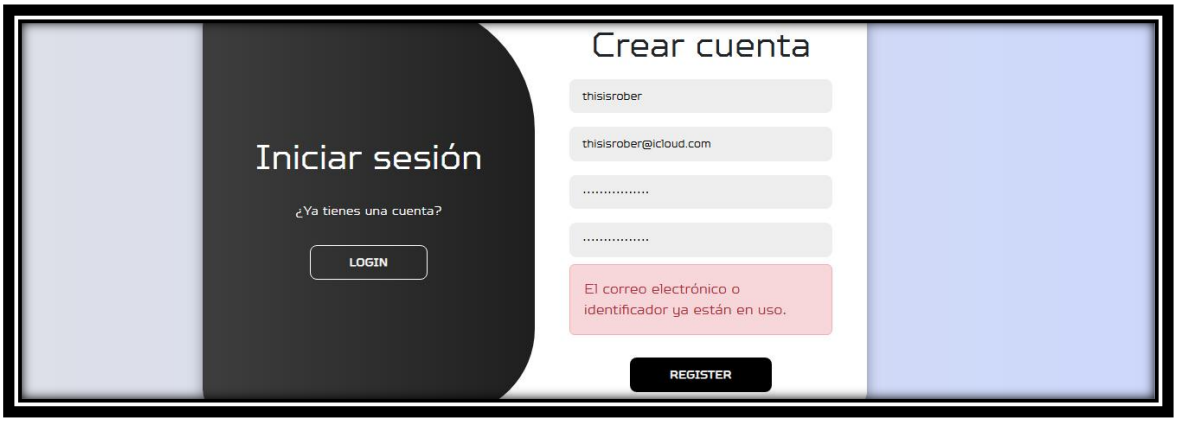

Ilustración 4: Prueba funcional de registro no exitoso

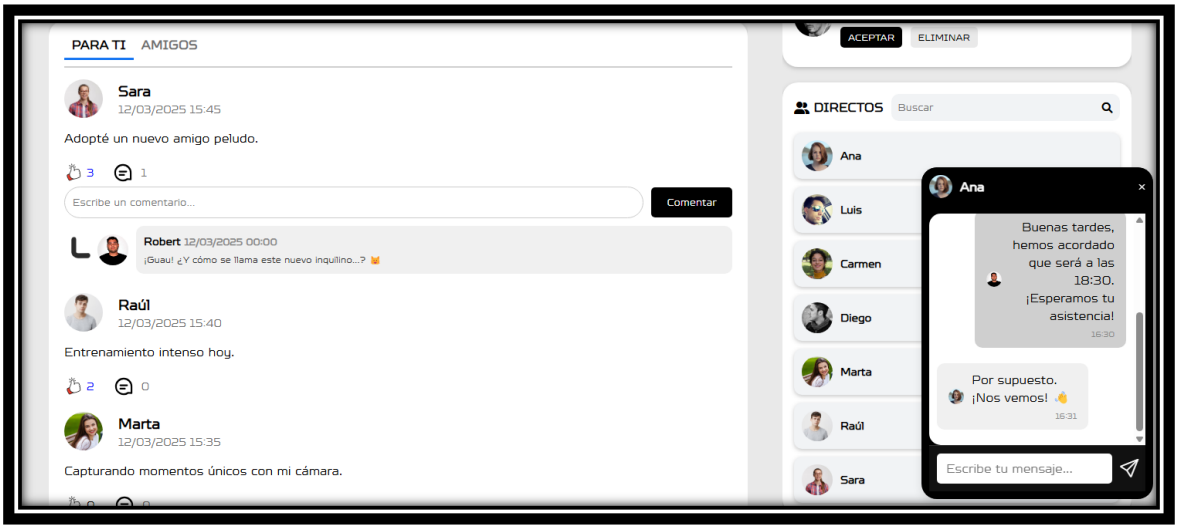

Ilustración 5: Prueba funcional de chat, comentario y reacción

PRUEBAS DE USABILIDAD

La experiencia de usuario se ha evaluado mediante sesiones de testeo con usuarios reales. Se han obtenido comentarios sobre la navegación, la intuitividad de la interfaz y la eficiencia de la interacción.

Ilustración 6: Prueba de usabilidad del buscador de usuarios

Ilustración 7: Prueba de usabilidad de solicitud de amistad

PRUEBAS DE ACCESIBILIDAD

Utilizando la herramienta TAW, se han verificado aspectos de accesibilidad para asegurar que la aplicación cumple con las normativas y es usable por personas con diferentes capacidades. Esto incluye:

- Contraste adecuado de colores.
- Navegación a través del teclado.
- Compatibilidad con lectores de pantalla.

Ilustración 8: Vista previa de la página de inicio de Bootstrap y Tailwind

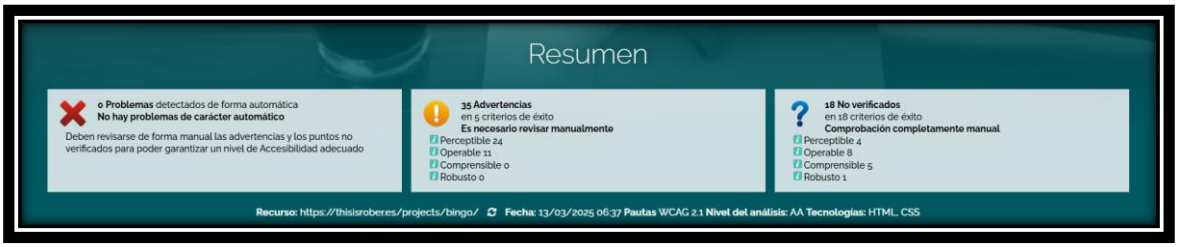

Ilustración 9: Prueba de accesibilidad mediante TAW

XIV. POSIBLES MEJORAS AL PROYECTO

Tras la finalización del proyecto, se han considerado las siguientes mejoras que se podrían realizar para hacer de esta red más robusta y versátil, al igual que más intuitiva y poder adaptarla a las nuevas tecnologías como puede ser la Inteligencia Artificial.

1. Implementación de funcionalidades esperadas anteriores, tales como las historias como en Instagram, vídeos cortos como TikTok, publicaciones con imágenes y videos como en X o en Facebook, etc.
2. Mejora en la accesibilidad: desarrollar una versión de la aplicación que funcione tanto en navegadores web como en dispositivos móviles (es decir, actualmente la página web está realizada para poder ser utilizada en equipos con altas

resoluciones, pero con la tecnología adecuada se podría realizar una versión APK para poder ser accesible desde una aplicación en sí, y no desde un navegador a través de un dispositivo móvil).

3. Chatbot con asistente de IA: integrar un chatbot que proponga descripciones a imágenes, ayude a usar la aplicación y entienda publicaciones, incluso comprobando si son verídicas.
4. Dominio real: se podría integrar el servidor a un dominio real de un coste adecuado para que cualquier usuario pueda acceder desde cualquier parte del mundo. Desde diversas opciones, se han contemplado las siguientes:
    - Compra de un dominio y después implementarlo con un servicio DNS en la Raspberry PI 4.
    - Implementar toda la página web en servidores ya existentes, como Hostalia, Hostinger, etc.

XV. CONCLUSIONES

CONCLUSIONES SOBRE EL TRABAJO REALIZADO

El desarrollo de este proyecto ha sido un proceso enriquecedor que ha permitido integrar diversas tecnologías de vanguardia para crear una plataforma social unificada. Este proyecto ha logrado simplificar significativamente la experiencia del usuario, centralizando las funciones esenciales en una única interfaz, lo que elimina la necesidad de alternar entre múltiples aplicaciones. La integración cuidadosa de herramientas de front-end y back-end, junto con un enfoque en la seguridad y usabilidad, ha permitido transformar una idea compleja en un sistema funcional y atractivo, demostrando el valor de una planificación y ejecución meticulosa.

CONCLUSIONES SOBRE EL SISTEMA DESARROLLADO

El sistema desarrollado evidencia un equilibrio óptimo entre funcionalidad, seguridad y usabilidad. La elección de la RaspBerry PI 4 junto con una configuración de servidor optimizada, garantiza un rendimiento estable y escalable, incluso frente a un volumen considerable de usuarios y datos. La arquitectura modular y el uso de tecnologías probadas (como PHP, MySQL/MariaDB, AJAX y frameworks de diseño responsivo) aseguran que la aplicación no solo cumpla con los requisitos actuales, sino que también esté preparada para futuras expansiones y mejoras. La solución técnica implementada se muestra robusta y adaptable, sentando las bases para un crecimiento sostenido.

CONCLUSIONES PERSONALES

A nivel personal, este proyecto ha representado un desafío estimulante y una oportunidad única para profundizar en mis conocimientos de desarrollo web, seguridad informática y administración de servidores. La experiencia adquirida durante estos meses al integrar múltiples Frameworks y herramientas, y al trabajar en un entorno de programación real y escalable, ha sido invaluable. He aprendido a valorar la importancia del diseño modular y la aplicación rigurosa de buenas prácticas en cada fase del desarrollo. Este proceso me ha permitido crecer profesionalmente y me ha preparado para afrontar retos tecnológicos aún mayores en el futuro.

En conjunto, estas conclusiones reflejan un trabajo sólido y una visión de futuro que sienta las bases para el crecimiento y la evolución continua de la aplicación.

XVI. BIBLIOGRAFÍA COMENTADA

La elaboración del proyecto ha sido posible gracias a la integración de múltiples recursos académicos, técnicos y prácticos que han servido como base para la implementación de las diversas funcionalidades y la optimización de las tecnologías utilizadas. Gran parte del contenido básico y los fundamentos teóricos se han extraído de la documentación y materiales impartidos en el Aula Virtual del colegio, complementados con fuentes especializadas y documentación oficial en línea. A continuación se presenta una bibliografía extensa y comentada que refleja el soporte teórico y práctico detrás del proyecto.

APUNTES Y MATERIALES ACADÉMICOS

Los apuntes y tutoriales disponibles en el Aula Virtual han sido el pilar fundamental del conocimiento aplicado en este proyecto, desde la programación en HTML5, CSS3 y JavaScript hasta conceptos avanzados en PHP, MySQL/MariaDB y la implementación de prácticas de seguridad en aplicaciones web.

DIRECCIONES WEB

- **Bootstrap Team.** *Bootstrap Documentation*. Bootstrap. Recuperado de https://getbootstrap.com/
  La documentación oficial de Bootstrap ha facilitado el uso de componentes y sistemas de grid para lograr una interfaz responsiva y moderna.
- **Tailwind Labs.** *Tailwind CSS Documentation*. Tailwind CSS. Recuperado de https://tailwindcss.com/
  Comentario: Tailwind CSS complementa a Bootstrap ofreciendo clases utilitarias para un control detallado del diseño, permitiendo personalizaciones precisas en la interfaz.
- **The PHP Group.** *PHP Manual*. PHP: Hypertext Preprocessor. Recuperado de https://www.php.net/manual/es/
- **Canonical Ltd.** *Ubuntu for Raspberry Pi*. Ubuntu. Última actualización consultada en 2025. Recuperado de https://ubuntu.com/download/raspberry-pi
- **TAW.** *Recursos para Pruebas de Accesibilidad Web*. TAW. Recuperado de http://www.tawdis.net/
- **MariaDB Corporation.** *MariaDB Documentation*. MariaDB. Recuperado de https://mariadb.com/kb/en/
  La documentación de MariaDB ha sido clave para diseñar y optimizar la base de datos, incluyendo la configuración para el soporte de emojis y otras funciones avanzadas.
- **Cropper.js Team.** *Cropper.js Documentation*. Cropper.js. Recuperado de https://fengyuanchen.github.io/cropperjs/
  Esta guía ha permitido integrar Cropper.js en el proyecto para ofrecer a los usuarios la posibilidad de recortar imágenes antes de subirlas, mejorando la gestión de archivos multimedia.
- **Google.** *Google OAuth 2.0 Documentation*. Google Developers. Recuperado de https://developers.google.com/identity/protocols/oauth2
  La documentación de Google OAuth ha servido de referencia para integrar

opciones de autenticación seguras y eficientes, facilitando el inicio de sesión a través de cuentas de Google.

- **Mozilla Developer Network.** (2023). *AJAX Guide*. MDN Web Docs. Recuperado de https://developer.mozilla.org/en-US/docs/Web/Guide/AJAX
  Los recursos sobre AJAX en MDN han sido fundamentales para implementar comunicaciones asíncronas, mejorando la interacción en tiempo real sin recargar la página.
- **Mozilla Developer Network.** *Fetch API Documentation*. MDN Web Docs. Recuperado de https://developer.mozilla.org/en-US/docs/Web/API/Fetch_API
  La documentación sobre la Fetch API complementa el uso de AJAX, facilitando solicitudes asíncronas para actualizar dinámicamente los contenidos.
- **Visual Studio Code.** *Visual Studio Code Documentation*. Microsoft. Recuperado de https://code.visualstudio.com/docs
  La documentación oficial de Visual Studio Code ha sido crucial para configurar el entorno de desarrollo remoto, permitiendo la integración con GitHub Copilot y la mejora continua del código.
- **GitHub.** *GitHub Copilot Documentation*. Recuperado de https://docs.github.com/en/copilot
  La guía de GitHub Copilot ha ayudado a integrar esta herramienta de inteligencia artificial en el flujo de trabajo, facilitando sugerencias de código y optimizando funciones de manera complementaria a la intervención humana.
- **GitHub.** *GitHub Documentation*. GitHub. Recuperado de https://docs.github.com/
  La documentación oficial de GitHub proporciona las directrices necesarias para el uso adecuado del control de versiones, la colaboración en equipo y la integración de herramientas de IA.
- **OpenAI.** *ChatGPT Prompt Engineering Guide*. OpenAI. Recuperado de https://help.openai.com/en/articles/6783457-chatgpt-prompt-engineering
  Esta guía ha sido utilizada para mejorar la formulación de prompts en inteligencia artificial, optimizando la interacción y los resultados obtenidos de herramientas como ChatGPT.