Manual técnico

1. Introducción

El manual técnico proporciona una guía detallada sobre el desarrollo, implementación y funcionamiento del sistema. En esta sección introductoria, se definen los objetivos del proyecto, se ofrece una descripción general del sistema y se detallan las tecnologías utilizadas en su desarrollo. A continuación, se presenta un resumen de cada aspecto:

1.1. Objetivos

El objetivo principal del proyecto es desarrollar una plataforma educativa interactiva que facilite la comunicación y colaboración entre estudiantes y profesores. Entre los objetivos específicos se incluyen:
Facilitar la gestión de cursos y perfiles de usuario.
Proporcionar una experiencia de aprendizaje en línea intuitiva y personalizada.
Garantizar la seguridad y confidencialidad de los datos del usuario.
Implementar funcionalidades de comunicación en tiempo real.
Mejorar la accesibilidad y usabilidad del sistema para usuarios de diferentes perfiles y dispositivos.

1.2. Descripción General del Sistema

La plataforma educativa se diseñará como un sistema web interactivo, que permita a los usuarios acceder desde cualquier dispositivo con conexión a internet. El sistema ofrecerá funcionalidades para la gestión de cursos, perfiles de usuario, interacción en tiempo real entre estudiantes y profesores, y seguridad de datos. Se utilizarán tecnologías modernas para garantizar un rendimiento óptimo y una experiencia de usuario satisfactoria.

1.3. Tecnologías Utilizadas

El desarrollo de la plataforma educativa se llevará a cabo utilizando las siguientes tecnologías:
Lenguajes de programación: HTML, CSS, JavaScript, PHP.
Frameworks y bibliotecas:Laravel.
Base de datos: MySQL.
Protocolos de comunicación: HTTP.
Seguridad: Autenticación y autorización, cifrado de datos.
Herramientas de desarrollo: Visual Studio Code, Xampp, Notion.
El uso de estas tecnologías garantiza un desarrollo eficiente, una experiencia de usuario optimizada y la seguridad de los datos del sistema.

2. Arquitectura del Sistema

En esta sección, se detalla la arquitectura del sistema, que incluye el modelo-vista-controlador (MVC), la estructura de directorios y un diagrama de componentes. A continuación, se presenta una descripción de cada aspecto:

2.1. Modelo-Vista-Controlador (MVC)

La arquitectura del sistema sigue el patrón Modelo-Vista-Controlador (MVC), que se utiliza para organizar el código en tres componentes principales:

Modelo: Representa la lógica de negocio y los datos del sistema. Incluye la interacción con la base de datos y las operaciones relacionadas con los datos.

Vista: Es responsable de la presentación de la interfaz de usuario. Incluye todo lo relacionado con la visualización de datos y la interacción del usuario.

Controlador: Actúa como intermediario entre el modelo y la vista. Gestiona las solicitudes del usuario, procesa la lógica de negocio y actualiza la vista según sea necesario.

El uso del patrón MVC permite una separación clara de las responsabilidades, lo que facilita el mantenimiento del código, la reutilización de componentes y la escalabilidad del sistema.

2.2. Estructura de Directorios

La estructura de directorios del sistema se organiza de la siguiente manera:

/app: Contiene los archivos relacionados con la lógica de la aplicación.

/models: Almacena los modelos de datos y las interacciones con la base de datos.

/views: Contiene las vistas de la interfaz de usuario.

/controllers: Incluye los controladores que gestionan las solicitudes del usuario y la lógica de negocio.

/public: Contiene los archivos estáticos accesibles públicamente, como imágenes, estilos CSS y scripts JavaScript.

/config: Contiene los archivos de configuración del sistema, como las variables de entorno y las opciones de configuración específicas del entorno.

/database: Almacena los archivos relacionados con la base de datos, como los scripts de migración y las semillas de datos.

Esta estructura de directorios proporciona una organización clara y coherente del código del sistema, lo que facilita su mantenimiento y escalabilidad.

3. Configuración del Entorno de Desarrollo

Esta sección del manual técnico proporciona instrucciones detalladas sobre la configuración del entorno de desarrollo necesario para trabajar en el proyecto. Incluye información sobre los requisitos del sistema, la instalación de herramientas como Apache (XAMPP), PHP y MySQL, y la configuración del servidor local. A continuación, se presentan los pasos a seguir:

3.1. Requisitos del Sistema

Antes de comenzar con la instalación, asegúrese de que su sistema cumpla con los siguientes requisitos mínimos:
Sistema Operativo: Compatible con Windows, Linux o macOS.
Espacio en Disco: Se recomienda al menos 1 GB de espacio libre en disco para la instalación de herramientas y archivos del proyecto.
Memoria RAM: Se recomienda al menos 2 GB de RAM para un rendimiento óptimo del entorno de desarrollo.

3.2. Instalación de Herramientas

Para configurar el entorno de desarrollo, siga estos pasos:

3.2.1. Instalación de XAMPP (Apache, MySQL, PHP)

XAMPP es un paquete de software que incluye Apache, MySQL, PHP y otras herramientas necesarias para crear un entorno de desarrollo local. Para instalar XAMPP, siga estos pasos:
Descargue la última versión de XAMPP desde el sitio web oficial: https://www.apachefriends.org/index.html
Ejecute el archivo de instalación descargado y siga las instrucciones del asistente de instalación.
Durante la instalación, seleccione los componentes necesarios, como Apache, MySQL y PHP.
Especifique la ubicación de instalación y complete el proceso de instalación.

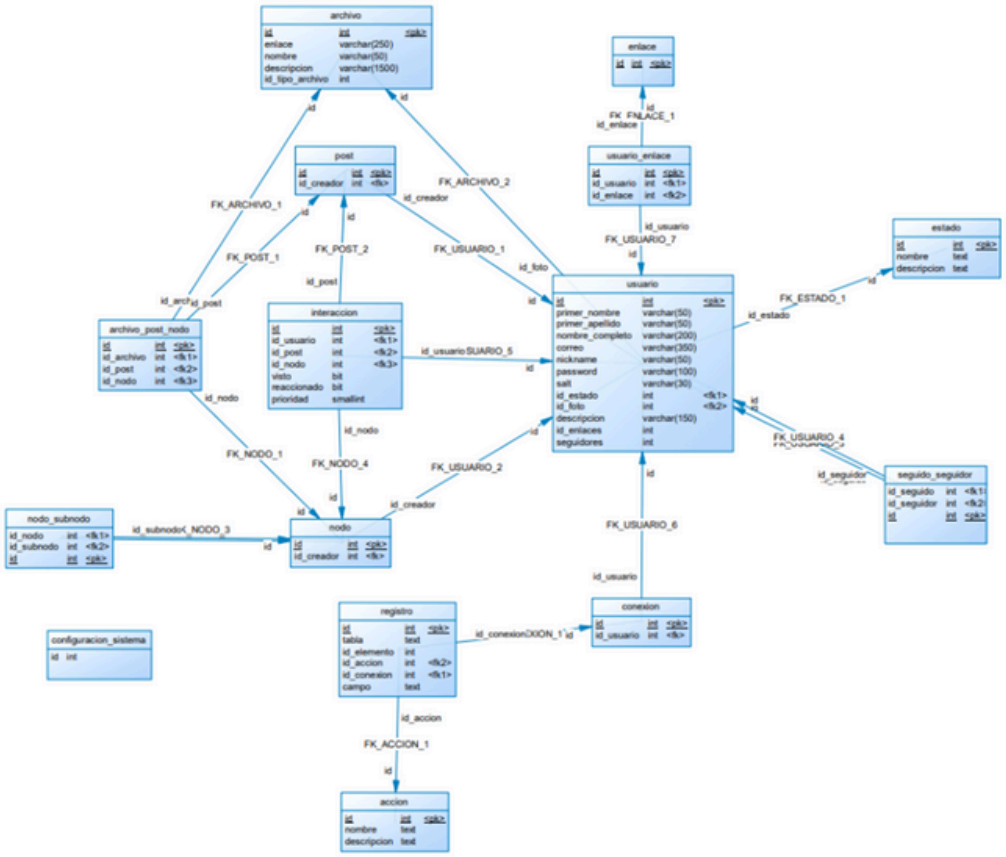
3.2.2. Configuración de XAMPP

Una vez instalado XAMPP, realice las siguientes configuraciones:
Inicie XAMPP desde el menú de inicio o la ubicación donde se instaló.
Inicie los módulos de Apache y MySQL haciendo clic en los botones correspondientes.
Verifique que Apache y MySQL estén funcionando correctamente accediendo a http://localhost en su navegador web.
Si todo está configurado correctamente, debería ver la página de inicio de XAMPP y poder acceder a la administración de MySQL.

3.3. Configuración del Servidor Local

Después de instalar XAMPP, configure el servidor local para que coincida con las necesidades del proyecto. Esto puede incluir la creación de una base de datos, la configuración de nombres de host virtuales, entre otros ajustes específicos del proyecto.
Una vez completados estos pasos, su entorno de desarrollo estará listo para trabajar en el proyecto.

4. Base de datos



Consulta en mySQL:

```
drop table if exists ACCION;
drop table if exists ARCHIVO;
drop table if exists ARCHIVO_POST_NODO;
drop table if exists CONEXION;
drop table if exists CONFIGURACION_SISTEMA;
drop table if exists ENLACE;
drop table if exists ESTADO;
drop table if exists INTERACCION;
drop table if exists NODO;
drop table if exists NODO_SUBNODO;
drop table if exists PLANTILLA_CORREO;
drop table if exists POST;
drop table if exists REGISTRO;
drop table if exists SEGUIDO_SEGUIDOR;
drop table if exists TIPO_ARCHIVO;
drop table if exists USUARIO;
```

```
/*==============================================================*/
/* Table: ACCION*/
/*==============================================================*/
create table ACCION
(
  ID              int not null,
  NOMBRE          text not null,
DESCRIPCION       text,
  primary key (ID)
);

/*==============================================================*/
/* Table: ARCHIVO*/
/*==============================================================*/
create table ARCHIVO
(
  ID              int not null,
  ENLACE          varchar(250) not null,
  NOMBRE          varchar(50) not null,
  DESCRIPCION     varchar(1500),
  ID_TIPO_ARCHIVO    int not null,
  primary key (ID)
);

/*==============================================================*/
/* Table: ARCHIVO_POST_NODO                                     */
/*==============================================================*/
create table ARCHIVO_POST_NODO
(
  ID              int not null,
ID_ARCHIVO        int not null,
  ID_POST         int,
  ID_NODO         int,
  primary key (ID)
);

/*==============================================================*/
/* Table: CONEXION*/
/*==============================================================*/
create table CONEXION
(
  ID              int not null,
```

```
D_USUARIO int not null,
TIEMPO_INICIO datetime not null,
TIEMPO_FIN datetime not null,
 IP varchar(50) not null,
NAVEGADOR varchar(50) not null,
 primary key (ID)
);

/*==============================================================*/
/* Table: CONFIGURACION_SISTEMA */
/*==============================================================*/
create table CONFIGURACION_SISTEMA
(
 ID int not null,
MAX_CARACTERES_POST int not null,
 COMISION int not null,
LONGITUD_PASSWORD int not null,
  CORREO            varchar(320) not null,
  PASSWORD          varchar(50) not null,
  SALT              varchar(50),
  primary key (ID)
);

/*==============================================================*/
/* Table: ENLACE*/
/*==============================================================*/
create table ENLACE

(
 ID int not null,
ID_USUARIO int not null,
 ENLACE varchar(1000) not null,
 primary key (ID)
);

/*==============================================================*/
/* Table: ESTADO*/
/*==============================================================*/
create table ESTADO
(
 ID int not null,
 NOMBRE text not null,
DESCRIPCION text,
 primary key (ID)
);
```

```
/*==============================================================*/
/* Table: INTERACCION                                           */
/*==============================================================*/
create table INTERACCION
(
  ID        int not null,
  ID_USUARIO     int not null,
  ID_POST       int,
  ID_NODO       int,
  VISTO        bit not null,
  REACCIONADO     bit not null,
  PRIORIDAD      smallint not null,
  ACCESO       int not null,
  primary key (ID)
);

/*==============================================================*/
/* Table: NODO                                                  */
/*==============================================================*/
create table NODO
(
  ID        int not null,
  ID_CREADOR     int not null,
  DESCRIPCION     text,
  FECHA_HORA_CREACION datetime not null,
  VISTAS       int not null,
  REACCIONES     int not null,
  PREMIUM       bit not null,
  primary key (ID)
);

/*==============================================================*/
/* Table: NODO_SUBNODO                                          */
/*==============================================================*/
create table NODO_SUBNODO
(
  ID_NODO       int,
  ID_SUBNODO     int
);

/*==============================================================*/
/* Table: PLANTILLA_CORREO                                      */
/*==============================================================*/
create table PLANTILLA_CORREO
```

```
(
  ID              int not null,
  NOMBRE          varchar(50) not null,
DESCRIPCION       varchar(250) not null,
  primary key (ID)
);

/*==============================================================*/
/* Table: POST*/
/*==============================================================*/
create table POST
(
  ID              int not null,
ID_CREADOR        int not null,
  TEXTO           text not null,
FECHA_HORA_CREACION datetime not null,
  VISTAS          int not null,
REACCIONES        int not null,
  primary key (ID)
);

/*==============================================================*/
/* Table: REGISTRO*/
/*==============================================================*/
create table REGISTRO
(
  ID              int not null,
  TABLA           text not null,
ID_ELEMENTO       int not null,
ID_ACCION         int not null,
ID_CONEXION       int not null,
  CAMPO           text not null,
  primary key (ID)
);

/*==============================================================*/
/* Table: SEGUIDO_SEGUIDOR                                      */
/*==============================================================*/
create table SEGUIDO_SEGUIDOR
(
  ID              int not null,
ID_SEGUIDO        int not null,
ID_SEGUIDOR       int not null,
  primary key (ID)
);
```

```
/*==============================================================*/
/* Table: TIPO_ARCHIVO                                          */
/*==============================================================*/
create table TIPO_ARCHIVO
(
  ID              int not null,
  NOMBRE          varchar(50) not null,
EXTENSION         varchar(10) not null,
  primary key (ID)
);

/*==============================================================*/
/* Table: USUARIO*/
/*==============================================================*/
create table USUARIO
(
  ID              int not null,
  PRIMER_NOMBRE      varchar(50) not null,
  PRIMER_APELLIDO    varchar(50) not null,
  NOMBRE_COMPLETO    varchar(200) not null,
  CORREO          varchar(350) not null,
  NICKNAME          varchar(50) not null,
  PASSWORD          varchar(100) not null,
  SALT            varchar(30),
ID_ESTADO         int not null,
  ID_FOTO          int not null,
DESCRIPCION       varchar(150) not null,
SEGUIDORES        int not null,
  primary key (ID)
);

alter table ARCHIVO add constraint FK_TIPO_ARCHIVO_1 foreign key
(ID_TIPO_ARCHIVO)

    references TIPO_ARCHIVO (ID) on delete restrict on update restrict;

alter table ARCHIVO_POST_NODO add constraint FK_FK_ARCHIVO_1 foreign key
(ID_ARCHIVO)
    references ARCHIVO (ID) on delete restrict on update restrict;

alter table ARCHIVO_POST_NODO add constraint FK_NODO_1 foreign key (ID_NODO)
    references NODO (ID) on delete restrict on update restrict;
```

```
alter table ARCHIVO_POST_NODO add constraint FK_POST_1 foreign key (ID_POST)
    references POST (ID) on delete restrict on update restrict;

alter table CONEXION add constraint FK_USUARIO_6 foreign key (ID_USUARIO)
    references USUARIO (ID) on delete restrict on update restrict;

alter table ENLACE add constraint FK_USUARIO_7 foreign key (ID_USUARIO)
    references USUARIO (ID) on delete restrict on update restrict;

alter table INTERACCION add constraint FK_NODO_4 foreign key (ID_NODO)
    references NODO (ID) on delete restrict on update restrict;

alter table INTERACCION add constraint FK_POST_2 foreign key (ID_POST)
    references POST (ID) on delete restrict on update restrict;

alter table INTERACCION add constraint FK_USUARIO_5 foreign key (ID_USUARIO)
    references USUARIO (ID) on delete restrict on update restrict;

alter table NODO add constraint FK_USUARIO_2 foreign key (ID_CREADOR)
    references USUARIO (ID) on delete restrict on update restrict;

alter table NODO_SUBNODO add constraint FK_NODO_2 foreign key (ID_SUBNODO)
    references NODO (ID) on delete restrict on update restrict;

alter table NODO_SUBNODO add constraint FK_NODO_3 foreign key (ID_NODO)
    references NODO (ID) on delete restrict on update restrict;

alter table NODO_SUBNODO add constraint FK_NODO_3 foreign key (ID_NODO)
    references NODO (ID) on delete restrict on update restrict;

alter table POST add constraint FK_USUARIO_1 foreign key (ID_CREADOR)
    references USUARIO (ID) on delete restrict on update restrict;

alter table REGISTRO add constraint FK_ACCION_1 foreign key (ID_ACCION)
    references ACCION (ID) on delete restrict on update restrict;

alter table REGISTRO add constraint FK_FK_CONEXION_1 foreign key (ID_CONEXION)
    references CONEXION (ID) on delete restrict on update restrict;

alter table SEGUIDO_SEGUIDOR add constraint FK_USUARIO_3 foreign key
(ID_SEGUIDO)
    references USUARIO (ID) on delete restrict on update restrict;
```

```
alter table SEGUIDO_SEGUIDOR add constraint FK_FK_USUARIO_4 foreign key
(ID_SEGUIDOR)
    references USUARIO (ID) on delete restrict on update restrict;

alter table USUARIO add constraint FK_ARCHIVO_2 foreign key (ID_FOTO)
    references ARCHIVO (ID) on delete restrict on update restrict;

alter table USUARIO add constraint FK_ESTADO_1 foreign key (ID_ESTADO)
    references ESTADO (ID) on delete restrict on update restrict;
```

5. Desarrollo del Frontend

El desarrollo del frontend de la plataforma se realiza utilizando tecnologías web estándar, incluyendo HTML, CSS y JavaScript. Este proceso implica el diseño de la interfaz de usuario y la integración de plantillas para garantizar una experiencia de usuario intuitiva y atractiva.

5.1 Diseño de la interfaz de Usuario

El diseño de la interfaz de usuario se centra en la creación de una experiencia visualmente atractiva y funcional para los usuarios. Se utilizan HTML y CSS para estructurar y estilizar los elementos de la interfaz, mientras que JavaScript se emplea para agregar interactividad y dinamismo.

5.2 Integración de plantillas

La integración de plantillas permite agilizar el proceso de desarrollo al utilizar diseños predefinidos y componentes reutilizables. Esto se logra mediante la incorporación de plantillas HTML y CSS preelaboradas, adaptándolas según las necesidades específicas de la plataforma

6. Desarrollo del Backend

El desarrollo del backend se centra en PHP como lenguaje de programación y la implementación de funcionalidades como registro e inicio de sesión, además de garantizar la seguridad del sistema.

6.2 Configuración del sistema

La interfaz de usuario no interactúa directamente con el desarrollo del backend, pero se ve afectada por las configuraciones realizadas por el Administrador del Sistema. Estas configuraciones incluyen ajustes generales del sistema, como límites de caracteres para publicaciones y políticas de privacidad.

6.3 Comunicación con el backend

La comunicación entre la interfaz de usuario y el backend se realiza a través de las API implementadas en PHP. Las API permiten que la interfaz de usuario envíe solicitudes al backend para realizar acciones como iniciar sesión, registrar usuarios y acceder a datos específicos del sistema.

6.4 Autenticación y autorización

Cuando un usuario interactúa con la interfaz de usuario para iniciar sesión, la información proporcionada se envía al backend a través de las API. El backend verifica la autenticidad de las credenciales ingresadas y, si son válidas, emite un token de sesión que se utiliza para autenticar al usuario en solicitudes posteriores.

7. Funcionalidades Avanzadas

La integración de APIs externas desde la interfaz de usuario permite ampliar las funcionalidades del sistema mediante la comunicación con servicios externos. Estas APIs facilitan la gestión de cursos, perfiles de usuario y la interacción entre estudiantes y profesores. Desde la interfaz, los usuarios pueden acceder a estas funcionalidades adicionales mediante interfaces intuitivas que les permiten interactuar con los datos y servicios proporcionados por las APIs externas.

7.1 Mensajera instantánea

La funcionalidad de mensajería instantánea desde la interfaz de usuario permite a los usuarios comunicarse de manera rápida y eficiente dentro de la plataforma. Los usuarios pueden enviar mensajes directos entre ellos, lo que facilita la colaboración, la discusión de temas y la resolución de dudas en tiempo real. La interfaz de mensajería proporciona una experiencia fluida y receptiva, lo que garantiza una comunicación efectiva entre los usuarios.

8. Despliegue

Para preparar el entorno de producción, se debe de contar con los siguientes elementos:

- Servidor Web Apache: Asegúrese de tener un servidor web Apache configurado y funcionando correctamente en su entorno de producción.
- PHP y MySQL: Verifique que las tecnologías PHP y MySQL estén instaladas y configuradas adecuadamente en su servidor.

9.1. Configuración del servidor web

- ·Nos aseguramos de que el servidor Apache esté configurado para admitir archivos PHP.
- ·Configuramos los ajustes de seguridad del servidor web para proteger contra accesos no autorizados y ataques.
- · Verificamos que las configuraciones de los módulos de seguridad de Apache estén actualizadas y optimizadas según las mejores prácticas de seguridad.

9.2 Subida de archivos y configuración de BBDD en el servidor

- Copie todos los archivos de su aplicación, incluidos los archivos PHP y los recursos estáticos, al directorio raíz del servidor web Apache.
- Asegúrese de que los permisos de los archivos y directorios sean los adecuados para que el servidor web pueda acceder a ellos correctamente.
- Importe la estructura de la base de datos MySQL en el servidor, utilizando el archivo SQL proporcionado.
- · Configure las credenciales de acceso a la base de datos en los archivos de configuración de su aplicación para que se conecte correctamente a la base de datos en el servidor.

10. Mantenimiento

En esta sección, nos enfocaremos en el mantenimiento del sistema desde la perspectiva de la interfaz de usuario, abordando aspectos como la gestión de versiones con Git, actualizaciones del sistema y soporte técnico.

10.1 Gestión de versiones con GitHub:

Para la gestión de versiones del sistema, se utilizará Git, un sistema de control de versiones distribuido. Desde la perspectiva del usuario, se recomienda seguir las siguientes prácticas:

- Control de Versiones de Código: Los cambios en el código se gestionan mediante commits en Git. Se recomienda a los desarrolladores realizar commits frecuentes con mensajes descriptivos que reflejen los cambios realizados.
- Control de Ramas: Se pueden utilizar ramas en Git para trabajar en nuevas características o correcciones de errores sin afectar la rama principal (master). Los usuarios pueden colaborar en diferentes ramas y fusionar cambios mediante solicitudes de extracción (pull requests).
- Integración con Plataformas de Desarrollo: Si se utiliza una plataforma de desarrollo colaborativo como GitHub o GitLab, los usuarios pueden interactuar con Git de manera más visual a través de interfaces web, lo que facilita la revisión de cambios, la gestión de problemas y la colaboración en proyectos.

10.2 Actualizaciones del sistema

Las actualizaciones del sistema se implementarán periódicamente para mejorar la funcionalidad, seguridad y rendimiento de la plataforma. Desde la perspectiva del usuario:

- Notificaciones de Actualización: Los usuarios recibirán notificaciones sobre las actualizaciones del sistema a través de mensajes en la interfaz de usuario o correos electrónicos. Estas notificaciones informarán sobre las mejoras y cambios que se han realizado.

- Proceso de Actualización: Las actualizaciones del sistema pueden requerir acciones por parte del usuario, como reiniciar la sesión o aceptar los términos y condiciones actualizados. Se proporcionarán instrucciones claras sobre cómo proceder con la actualización.
- Soporte Técnico para Actualizaciones: En caso de problemas durante el proceso de actualización, los usuarios podrán acceder al soporte técnico a través de canales como chat en vivo, correo electrónico o sistemas de tickets para recibir asistencia inmediata.

10.3 Soporte Técnico

El soporte técnico estará disponible para ayudar a los usuarios con cualquier problema o consulta relacionada con el sistema. Desde la interfaz de usuario:

- Centro de Ayuda: Se proporcionará un centro de ayuda accesible desde la interfaz de usuario, donde los usuarios podrán encontrar respuestas a preguntas frecuentes, guías de solución de problemas y tutoriales paso a paso.
- Formulario de Contacto: los usuarios podrán enviar consultas o informar sobre problemas a través de un formulario de contacto integrado en la interfaz de usuario. Este formulario enviará automáticamente la consulta al equipo de soporte para su resolución.

11. Conclusiones

La implementación de APIs para la comunicación entre distintas partes del sistema ha demostrado ser fundamental para garantizar la cohesión y el funcionamiento adecuado de la aplicación.
La configuración adecuada del sistema, junto con medidas de seguridad como el cifrado de contraseñas y la validación de entrada de usuario, son fundamentales para proteger la integridad de los datos y garantizar la privacidad de los usuarios.

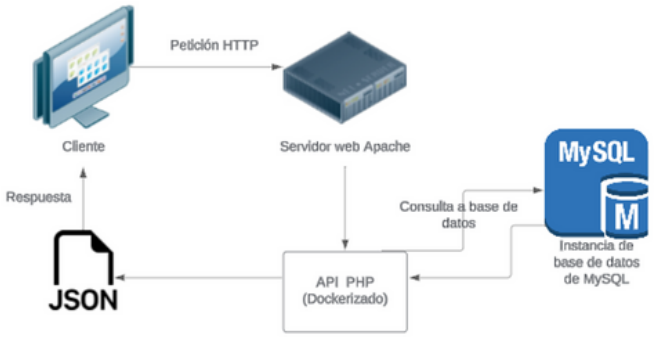
11.1 Recomendaciones para futuros desarrollos

Basándonos en nuestras experiencias, hemos recopilado las siguientes recomendaciones para futuros desarrollos de la interfaz de usuario:

- ·Optimización de la Experiencia del Usuario: Continuar mejorando la experiencia del usuario mediante la optimización de la interfaz, la respuesta en tiempo real a las acciones del usuario y la personalización de contenido.
- Automatización y Escalabilidad: Utilizar herramientas como Kubernetes para la automatización de implementaciones y la gestión eficiente de contenedores, lo que facilitará la escalabilidad del sistema y la gestión de recursos en entornos de producción.
- Colaboración Eficiente: Emplear herramientas de colaboración como Notion para facilitar la organización, planificación y colaboración entre equipos de desarrollo, lo que contribuirá a la eficiencia y al éxito del proyecto.

Anexos

1. Diagrama de arquitectura de software



2. Diagramas de casos de uso