**Documento de Arquitectura de Software**

**Proyecto MindSoft**

**Versión 1.0**

**Control de versiones**

|  |  |  |  |
| --- | --- | --- | --- |
| **Fecha** | **Versión** | **Descripción** | **Autor** |
| **14/09/2024** | **1.0** | **Creación del documento.** | **Deyvi Gomez** |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

# Índice

# 

[**Índice 3**](#_ol90dilw13df)

[**Introducción 4**](#_e914bjacbc6j)

[Propósito 4](#_hlgr79e9b03l)

[Alcance 4](#_cqhqw4p9w662)

[Definicion, siglas y abreviaturas 4](#_fo9p6i8765hs)

[Vista global 4](#_8dgi8gahkkxq)

[**Marco Arquitectura 4**](#_oll4u221y0j3)

[Modelo vista controlador 4](#_5suut5tmr56v)

[Metas y restricciones arquitectónicas 4](#_wt2c6vqmsmrc)

[**Vista Física 4**](#_l47vnk81fbsm)

[**Vista Funcional o Lógica 4**](#_twrxw1k71ciu)

[**Vista de despliegue 4**](#_o5gj4fs3tlww)

[**Vista de procesos 4**](#_z91z43xbikpw)

# 

# **Introducción**

## **Propósito** El propósito de este documento es definir y describir la Arquitectura de Software para el desarrollo del proyecto “MindSoft”. Se implementará una arquitectura basada en el patrón Modelo-Vista-Controlador(MVC), la cual organizará el sistema de manera clara y eficiente. Este documento está dirigido al equipo de desarrollo para proporcionar las bases arquitectónicas y una visión global del diseño.

## **Alcance** Este documento describe la arquitectura del sistema “MindSoft”, una aplicación móvil diseñada para plataformas Android e IOS. Permitirá a los estudiantes llevar un registro de sus emociones, pensamientos y metas de forma intuitiva. Se detallan las vistas arquitectónicas del sistema, los patrones de diseño aplicados y los atributos de calidad que aseguran un funcionamiento eficiente, seguro y escalable de la aplicación.

## **Definicion, siglas y abreviaturas**

* MVC: Modelo-Vista-Controlador, patrón arquitectónico utilizado para separar la lógica de negocio, la presentación y la interacción del sistema.
* API: Interfaz que permite la comunicación entre aplicaciones.
* HTTP: Protocolo de transferencia de datos en la web.
* FastAPI: Framework web utilizado para desarrollar el backend del sistema.
* MySQL: Sistema de gestión de bases de datos relacional.
* Railway: Plataforma para alojar aplicaciones y bases de datos.
* CSS/HTML/JavaScript: Tecnologías utilizadas para el frontend de la aplicación web.
* JWT: JSON Web Token, utilizado para autenticación segura.
* Datagrip: Herramienta para administrar bases de datos.

## **Vista global** Se presentará una macro arquitectura del sistema “MindSoft” que especifica las vistas utilizadas en el proyecto, metas y restricciones arquitectónicas, así como diagramas para proporcionar una comprensión completa de la estructura y funcionalidad del sistema.

# **Marco Arquitectura**

## **Modelo vista controlador**

Es un patrón de diseño arquitectónico de software, que sirve para clasificar la información, la lógica del sistema y la interfaz que se le presenta al usuario. En este tipo de arquitectura existe un sistema central o controlador que gestiona las entradas y la salida del sistema, uno o varios modelos que se encargan de buscar los datos e información necesaria y una interfaz que muestra los resultados al usuario final.

A continuación se describen los tres componentes clave de esta arquitectura:

-Modelo (Model):

Este componente se encarga de manipular, gestionar y actualizar los datos. Si se utiliza una base de datos aquí es donde se realizan las consultas, búsquedas, filtros y actualizaciones.

-Vista (View):

Este componente se encarga de mostrarle al usuario final las pantallas, ventanas, páginas y formularios; el resultado de una solicitud.

Desde la perspectiva del programador este componente es el que se encarga del frontend; la programación de la interfaz de usuario si se trata de un aplicación de escritorio, o bien, la visualización de las páginas web (CSS, HTML, HTML5 y Javascript).

-Controlador (Controller):

Este componente se encarga de gestionar las instrucciones que se reciben, atenderlas y procesarlas. Por medio de él se comunican el modelo y la vista: solicitando los datos necesarios; manipulándolos para obtener los resultados; y entregándolos a la vista para que pueda mostrarlos.

La arquitectura MVC es ideal para nuestra aplicación móvil porque nos permitirá organizar el código de manera clara, separando la lógica de negocio, la interfaz de usuario y el control de las interacciones. Esto facilita el mantenimiento y escalabilidad de la aplicación, permitiendo agregar funciones como estadísticas o seguimiento de objetivos sin afectar otras áreas. Además, la reutilización del código y la facilidad para realizar pruebas unitarias aseguran una aplicación eficiente y estable, fundamental para el propósito de nuestra app.

## 

## **Metas y restricciones arquitectónicas**

|  |  |  |  |
| --- | --- | --- | --- |
| **Atributos de calidad “Observables”** | | | |
| **Atributo** | **Descripción** | **Patrón de Arquitectura** | **Aplicación** |
| Rendimiento | Respuesta rápida al ingresar datos y visualizar estadísticas. | **MVC** | La app debe responder instantáneamente al registro diario de emociones y metas. |
| Usabilidad | Interfaz intuitiva para estudiantes, fácil de navegar. | **MVC** | La UI debe ser sencilla, con entradas rápidas para el diario y opciones de visualización clara**.** |
| Escalabilidad | Capacidad para añadir nuevas funcionalidades en el futuro. | **MVC** | Permite agregar fácilmente nuevas características como seguimiento avanzado o recordatorios. |
| Disponibilidad | Acceso constante a los servicios de la aplicación. | **MVC** | La app debe permitir el registro de datos incluso sin conexión y sincronizarlos después |
| Seguridad | Protección de los datos emocionales de los estudiantes | **MVC** | Los registros del diario y datos de usuario deben estar encriptados para garantizar privacidad. |

|  |  |  |  |
| --- | --- | --- | --- |
| **Atributos de calidad “No Observables”** | | | |
| **Atributo** | **Descripción** | **Patrón de Arquitectura** | **Aplicación** |
| Mantenibilidad | Facilidad para realizar actualizaciones y correcciones. | **MVC** | El código debe permitir actualizaciones sencillas para nuevas características sin afectar el resto. |
| Modularidad | Separación clara entre lógica, presentación e interacción. | **MVC** | Separar las funcionalidades del diario, estadísticas y metas para una fácil modificación. |
| Reusabilidad | Capacidad de reutilizar partes de la aplicación en otros módulos. | **MVC** | Permitir agregar fácilmente nuevas características como seguimiento avanzado o recordatorios. |
| Testabilidad | Acceso constante a los servicios de la aplicación. | **MVC** | Cada componente debe ser testeable individualmente, asegurando la calidad del sistema completo |

# 

# **Vista Física**

Diagrama informal de la vista física de MindSoft

El Diagrama informal de la vista física contempla el uso del aplicativo por parte del usuario, quien ingresa al aplicativo móvil desde diversos dispositivos, como dispositivos móviles. El usuario accede al aplicativo móvil de MindSoft, la cual está alojada en un servidor de aplicaciones.

Los atributos de calidad presentes son

**Portabilidad**: La aplicación funcionará en múltiples plataformas, específicamente Android e iOS.

**Usabilidad**: Se menciona que la aplicación debe ser fácil de aprender y entender, lo que indica que está diseñada para proporcionar una buena experiencia de usuario.

**Seguridad**: Se hace referencia a la protección de los datos sensibles, lo que implica que la seguridad es una prioridad para garantizar la confidencialidad de la información de los usuarios.

**Mantenibilidad**: Aunque no se menciona explícitamente, dado que la aplicación está pensada para ser utilizada por estudiantes, se podría inferir que el código debe ser mantenible para poder agregar mejoras, corregir errores o adaptarse a futuras necesidades.

**Eficiencia**: La aplicación debería tener un rendimiento óptimo en dispositivos móviles, evitando ralentizaciones o mal funcionamiento, aunque este atributo no se menciona directamente en los requisitos.

**Fiabilidad**: La aplicación debe ser confiable en cuanto a la recolección, almacenamiento y visualización de datos, asegurando que no se pierda información relevante para los usuarios.

Elementos físicos:

* Usuarios: Estudiantes que utilizan la aplicación para registrar su estado de ánimo, pensamientos de gratitud, calificaciones diarias y gestionar sus objetivos personales. Acceden desde sus dispositivos móviles.
* Dispositivos electrónicos: Principalmente dispositivos móviles como smartphones y tablets, ya que la aplicación está diseñada para plataformas Android e iOS.
* Internet: La conexión que permite a los dispositivos móviles comunicarse con el backend, registrar y sincronizar los datos entre los usuarios y el servidor.
* Servidor: El servidor donde se aloja la lógica del backend, utilizando FastAPI como framework para manejar las solicitudes y respuestas de los usuarios.
* Base de datos: Almacenará información sensible como los registros de estados de ánimo, calificaciones, pensamientos de gratitud y objetivos personales. Utiliza MySQL, alojado en Railway, y administrado con Datagrip.

Diagrama informal físico - MindSoft

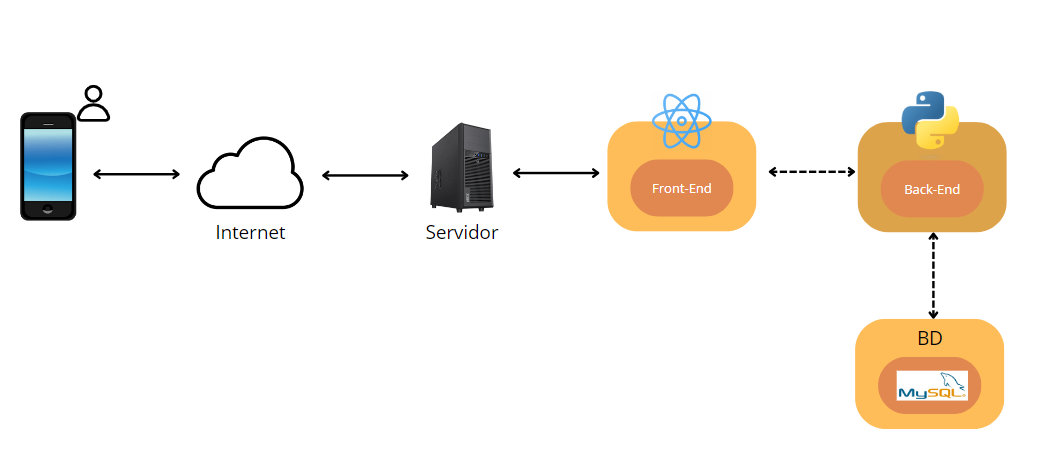
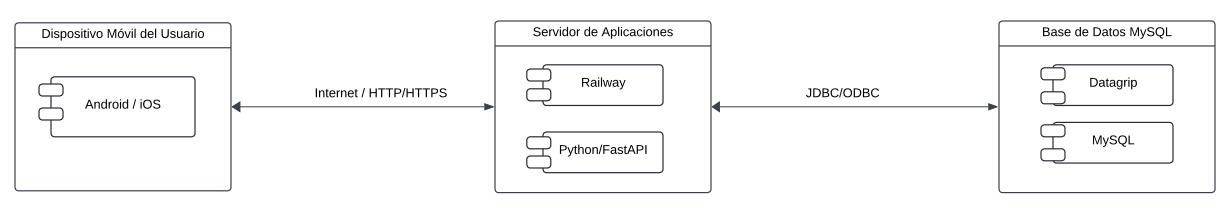


Diagrama de Despliegue con Vista Física



# **Vista Funcional o Lógica**

Diagrama de Clases de MindSoft

El siguiente diagrama es una representación visual de las clases del sistema y las relaciones que existen entre ellas. Estas clases son útiles pues son las necesarias para que el sistema funcione correctamente con las funcionalidades establecidas.

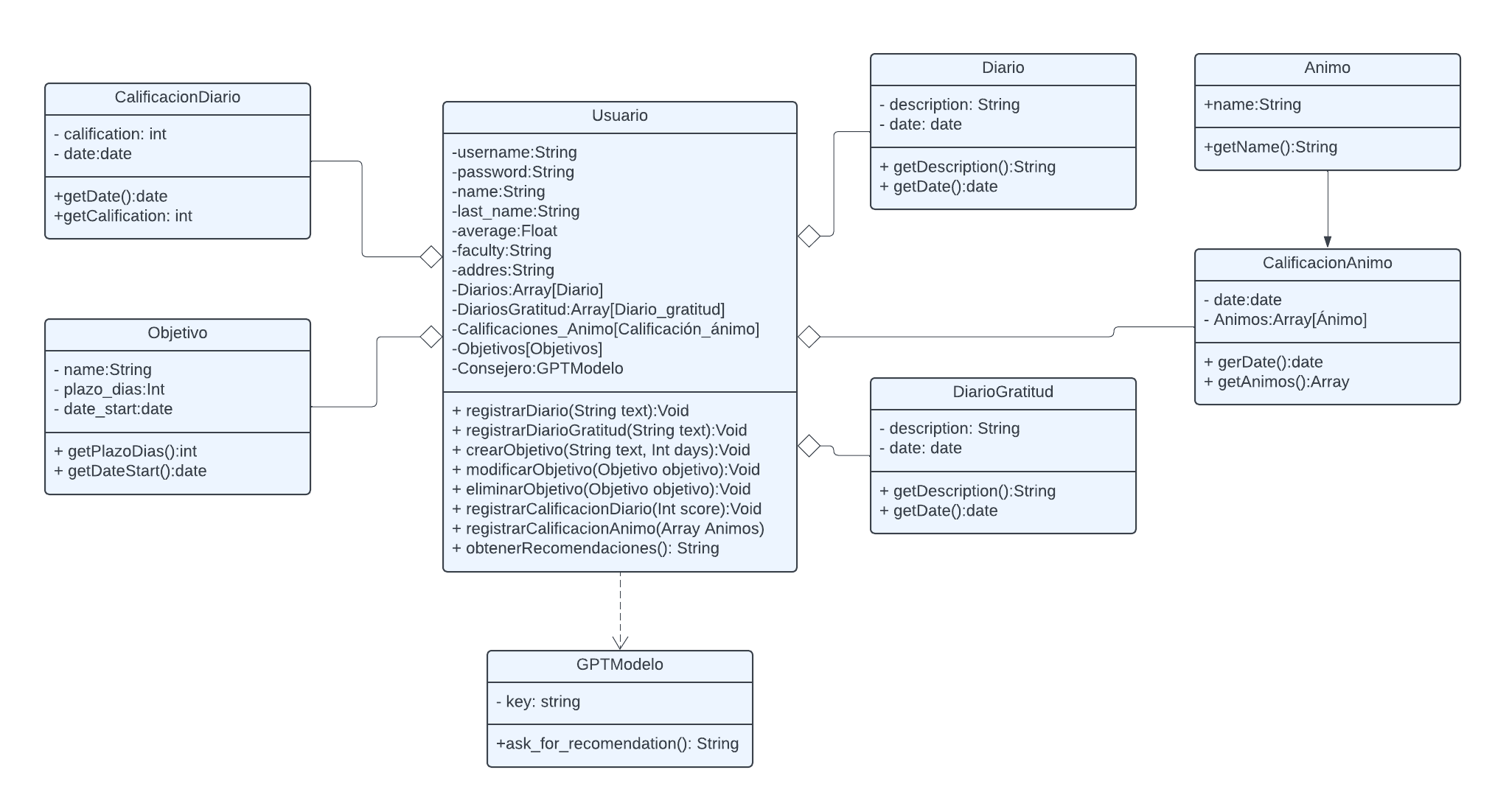
Los atributos de calidad presentes son:

**Mantenibilidad**: Las clases están bien definidas con una única responsabilidad facilitando la modificación del sistema para corregir errores.

**Seguridad**: Las visibilidades de los atributos y métodos están definidos adecuadamente para proteger al sistema contra accesos no autorizados y vulnerabilidades.

**Rendimiento**: Evasión de demasiados atributos y métodos que pueden volverse difíciles de gestionar para obtener mejores tiempos de respuesta, un mejor uso de memoria y capacidad de procesamiento.

**Comprensibilidad**: Uso de nombres claros y descriptivos para las clases, atributos y métodos. No sobrecargar de detalles innecesarios para que el diagrama sea fácil de entender.



# **Vista de Despliegue**

Diagrama de Despliegue de MindSoft

Este diagrama describe las configuraciones de redes físicas sobre las cuales el software será desarrollado. Se indican los nodos físicos que interactúan con la aplicación móvil, incluyendo los nodos del servidor web, servidor de base de datos y los componentes asociados.

## Componentes

* Nodo de Dispositivo Móvil

Descripción: Dispositivo desde el cual el usuario accede a la aplicación móvil MindSoft. Estos dispositivos pueden ser smartphones o tablets con sistemas operativos Android o iOS.

Protocolo de Comunicación: HTTP/HTTPS

Interacción: Envía solicitudes a la API del servidor de aplicaciones para interactuar con la funcionalidad de la aplicación móvil.

* Nodo del Servidor de Aplicaciones

Descripción: Servidor desplegado en la plataforma Railway que aloja la lógica del backend de la aplicación. Utiliza Python con el framework FastAPI para gestionar la lógica del negocio y exponer servicios a través de APIs.

Funcionalidad de la Aplicación: Maneja la lógica de negocio, procesa solicitudes de la aplicación móvil y gestiona la comunicación con la base de datos. Proporciona una capa de servicios y APIs que permiten la interacción con la aplicación móvil.

Interacción: Recibe solicitudes de la aplicación móvil, procesa la lógica de negocio y envía consultas a la base de datos.

* Nodo del Servidor de Base de Datos

Descripción: Sistema de gestión de base de datos MySQL, que puede estar integrado en el entorno de Railway o en un servicio de base de datos en la nube separado.

Funcionalidad de la Aplicación: Almacena datos persistentes como la información del usuario, registros de sesiones, y datos relacionados con las actividades de salud mental.

Interacción: Recibe y responde a las consultas de la lógica de negocio gestionada por el servidor de aplicaciones.

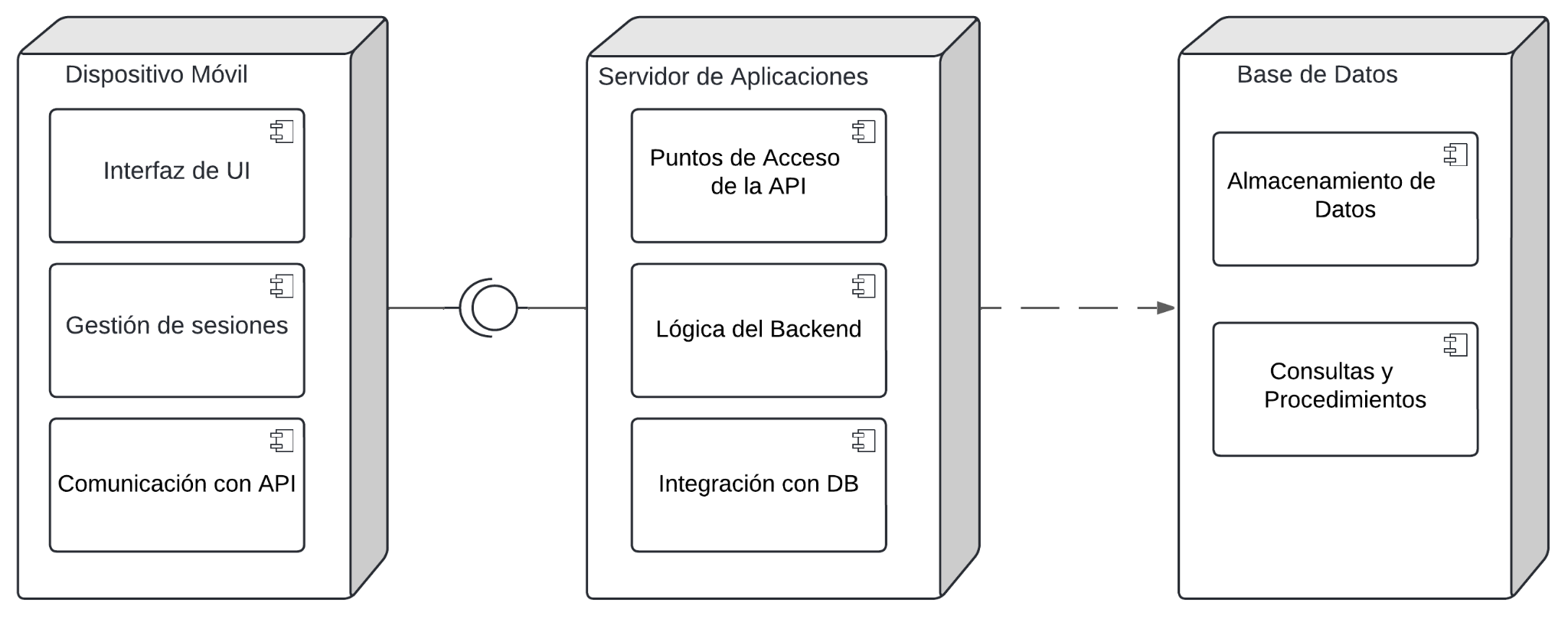
* Lógica de Backend

Descripción: Integra las interfaces de APIs y gestiona la lógica del negocio creada por microservicios.

Interacción: Maneja la lógica de negocio y la comunicación con la capa de acceso a datos para realizar operaciones en la base de datos.

## Atributos

* Modificabilidad: La separación de responsabilidades entre la aplicación móvil, el servidor de aplicaciones y la base de datos facilita la modificación y mantenimiento del sistema. La modularización del software permite realizar cambios en una capa sin afectar a las demás, promoviendo una mayor adaptabilidad y escalabilidad del sistema.
* Escalabilidad: La arquitectura permite escalar el servidor de aplicaciones y la base de datos de manera independiente, adaptándose a un aumento en el número de usuarios o en el volumen de datos sin impactar la funcionalidad general de la aplicación.
* Seguridad: La comunicación entre los dispositivos móviles y el servidor de aplicaciones se realiza a través de HTTP/HTTPS, asegurando que los datos transmitidos estén cifrados y protegidos contra posibles interceptaciones.



-

# **Vista de Procesos**

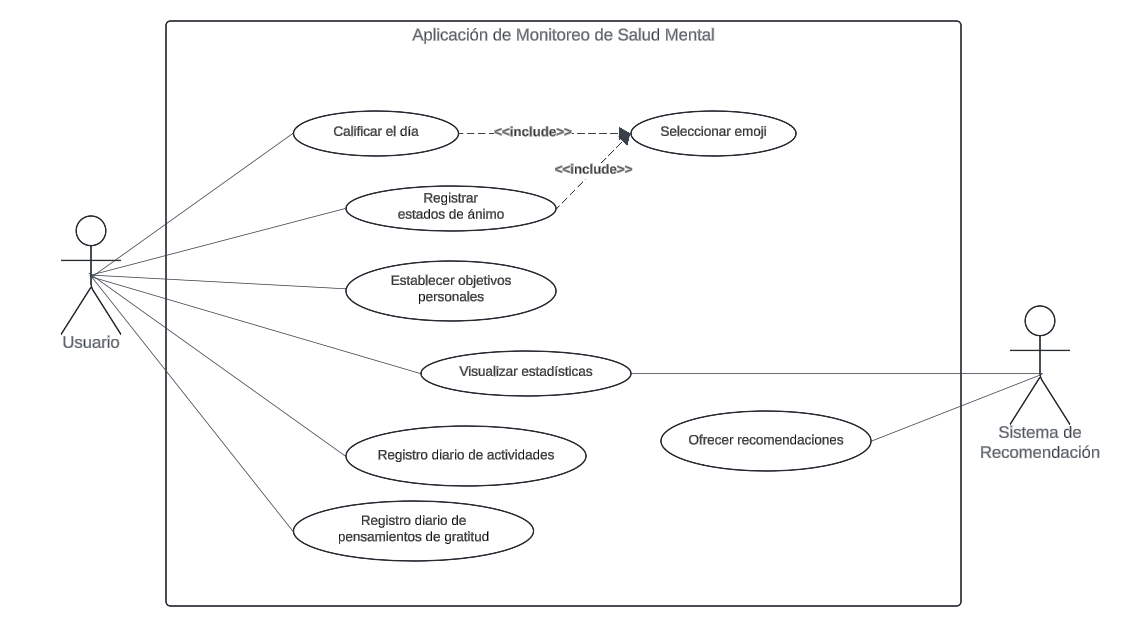
Diagrama de casos de uso

El siguiente diagrama de casos de uso representa los requisitos funcionales de la aplicación MindSoft. También representa las interacciones entre el usuario y el sistema de recomendaciones, asimismo, cómo el sistema responde a estas acciones.

Los atributos de calidad presentes son:

**Escalabilidad**: La inclusión del sistema de recomendaciones sugiere que la aplicación puede evolucionar y ofrecer nuevas funcionalidades según los datos ingresados por el usuario.

**Usabilidad**: El sistema permite al usuario realizar tareas relacionadas con su bienestar como: registrar su estado de ánimo, calificación diaria y establecer objetivos personales.

****