



ugr | Universidad
de **Granada**

TRABAJO FIN DE GRADO
INGENIERÍA EN INFORMÁTICA

Resilience

Autor

Juan Carlos López Morales

Directores

María Jesús Rodríguez Sánchez



FACULTAD DE EDUCACIÓN, ECONOMÍA, Y TECNOLOGÍA DE
CEUTA

—
Ceuta, 7 de noviembre de 2023

Yo, **Juan Carlos López Morales**, alumno de la titulación Ingeniería Informática de la **Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación de la Universidad de Granada**, con DNI 01648131C, autorizo la ubicación de la siguiente copia de mi Trabajo Fin de Grado en la biblioteca del centro para que pueda ser consultada por las personas que lo deseen.

Agradecimientos

Agradecer a todos los que me han apoyado de cualquier forma en este camino, pero sobre todo tengo que mencionar, en el ámbito universitario, a Antonio Marfil y María Jesús Rodríguez por apoyarme y confiar siempre en mi, a Reduan Mustafa y Jorge Martín por ser los mejores compañeros que me puedo llevar de esta etapa de mi vida, y en el ámbito personal a Laura Fernández por el apoyo incondicional, y mis padres, Paqui Morales y Juan Carlos López, y mi hermano Alejandro López, que da igual lo que pase, son y serán mis pilares en la vida.

Índice general

1. Introducción	13
1.1. Motivación	13
1.1.1. Motivación General	13
1.1.2. Motivaciones Técnicas	13
1.1.2.1. Elección de Microservicios	13
1.1.2.2. Elección de MongoDB y MySQL	13
1.1.2.3. Elección de React	14
1.2. Objetivos	14
1.2.1. Objetivo Principal	14
1.2.2. Objetivos Específicos	14
1.3. Diagrama de Gantt	15
1.4. Aplicaciones relacionadas con la salud mental	15
1.4.1. Ejemplos actuales en diferentes sectores	15
1.4.2. Aplicación de inspiración	17
1.4.2.1. Clue:[9]	17
1.4.2.2. Resilience:	18
1.4.3. Potencial de crecimiento y futuras tendencias.	19
2. Fundamentos teóricos	21
2.1. Arquitectura de microservicios	21
2.1.1. Historia de los Microservicios	21
2.1.2. Arquitectura monolítica	22
2.1.3. Arquitecturas monolíticas y microservicios	23
2.1.4. Ventajas y desventajas de los Microservicios	24
2.1.4.1. Ventajas	24
2.1.4.2. Desventajas	24
2.1.5. Características generales de los microservicios	25
2.2. Inteligencia artificial	26
2.2.1. Historia y Evolución	26
2.2.1.1. Primeros Años (1950s-1970s):	26
2.2.1.2. Años Intermedios (1980s-2000s):	26
2.2.1.3. Era Moderna (2010s-Presente):	27
2.2.2. Principales Subcampos de la IA	27
2.2.3. Herramientas y Frameworks	27
2.2.4. Conclusión	28

3. Intersección entre Microservicios e Inteligencia Artificial	29
3.1. Escalabilidad	29
3.2. Rapidez en la Iteración	29
3.3. Políglota	30
3.4. Reutilización	30
3.5. Desacoplamiento	30
3.6. Mantenimiento	30
4. Diseño y desarrollo del sistema	31
4.1. Definición del Problema	31
4.2. Análisis	31
4.2.1. Requisitos funcionales	31
4.2.2. Requisitos no funcionales	32
4.2.3. Restricciones semánticas	33
4.2.4. Casos de uso	34
4.2.4.1. Registro de usuarios	34
4.2.4.2. Gestión de perfil	34
4.2.4.3. Registro Diario de Rutina y Estado Emocional.	35
4.2.4.4. Procesamiento de Datos con IA	35
4.2.4.5. Ejercicios Diarios de Respiración	36
4.2.4.6. Historial y graficos	36
4.2.4.7. Privacidad y seguridad	36
4.2.5. Tarjetas casos de uso	36
4.3. Diseño	44
4.3.1. Diseño de la Base de Datos	44
4.3.2. Diagrama entidad-relación	46
4.3.3. Elección y diseño de los microservicios de la aplicación .	46
5. Implementación	49
5.1. Marco tecnológico	49
5.1.1. Lenguajes de Programación y Frameworks	49
5.1.1.1. NodeJS	49
5.1.1.2. Herramientas y tecnologías basadas en Python	49
5.2. Herramientas de Desarrollo	50
5.3. Características de la implementación de la arquitectura	51
5.4. Implementación de los microservicios	53
5.4.1. Contenerización y Orquestación	54
5.4.2. Frontend	54
5.5. Evaluación	54
6. Modelo entrenado para la IA	57
6.1. Fuente del código	57
6.2. Detalles del modelo entrenado	57
6.2.1. Variables para interpretar resultados	57
6.2.2. Aplicación del modelo	58
6.2.3. Aplicación en Resilience	59
6.3. Resultados	60
6.3.1. Evaluación del Modelo	60
6.3.2. Interpretación de los Datos	60
6.3.3. Realización del análisis	60

6.3.4.	Descripción de Datos	60
6.3.4.1.	Visualización de Datos	60
6.3.4.2.	Búsqueda de Patrones	60
6.3.4.3.	Modelado Predictivo	61
6.3.4.4.	Cambio Temporal de Emociones	61
6.3.4.5.	Análisis de Texto y Sentimientos	61
6.3.4.6.	Implementación del Código	61
6.3.5.	Descubrimientos Clave	61
7.	Desafíos y consideraciones éticas	63
7.1.	Problemas de seguridad y privacidad.	63
7.2.	Consideraciones sobre sesgos y justicia en la IA.	64
7.3.	Implicaciones éticas y sociales.	65
8.	Conclusión	67
8.1.	Reflexión sobre los objetivos alcanzados.	67
8.2.	Proyecciones futuras y recomendaciones.	67
A.	Anexo I	73
A.1.	Manual de instalación	73
A.1.1.	Back-end	73
A.1.2.	Front-end	73
B.	Anexo II	75
B.1.	Manual de uso	75
B.1.1.	Inicio de sesión y registro	75
B.1.2.	Home	76
B.1.2.1.	Ejemplo de feedback	77
B.1.3.	Calendario	78
B.1.4.	Add	79
B.1.5.	Ánalysis	79
B.1.6.	Respiración y desconexión	80

Índice de figuras

1.1. Diagrama de Gantt	15
1.2. Imagen de Daylio. Fuente: Daylio	16
1.3. Imagen de MoodPath.Fuente: MoodPath	16
1.4. Imagen de PossibleSelf.Fuente: PossibleSelf	17
1.5. Imagen de Clue. Fuente: Clue	18
1.6. Imagen de Resilience. Fuente: elaboración propia	19
1.7. Imagen de Resilience. Fuente: elaboración propia	19
2.1. Imagen del esquema de microservicios. Fuente: Google imágenes	24
4.1. Caso de uso Inicio de sesión	34
4.2. Caso de uso Gestión de perfil	34
4.3. Caso de uso Registro Diario de Rutina y Estado Emocional . .	35
4.4. Caso de uso Procesamiento de Datos con IA	35
4.5. Caso de uso Ejercicios Diarios de Respiración	36
4.6. Caso de uso Historial y Gráficos	36
4.7. Caso de uso Privacidad y Seguridad	36
4.8. Diagrama Entidad-Relación. Fuente: propia	46
4.9. Esquema de los microservicios del sistema. Fuente: elaboración propia	47
6.1. Cabecera de la función	59
6.2. Gráfico comparativo de PHQ y GAD score por género	62
B.1. Imagen de Resilience Inicio de sesión. Fuente: propia	75
B.2. Imagen de Resilience Registro de Usuario. Fuente: propia . . .	76
B.3. Imagen de Resilience Menú de inicio. Fuente: propia	76
B.4. Imagen de Resilience Menú de inicio. Fuente: propia	77
B.5. Imagen de Resilience Usuario sin recomendar tratamiento. Fuen- te: propia	77
B.6. Imagen de Resilience Usuario que se le recomienda tratamiento. Fuente: propia	78
B.7. Imagen de Resilience Calendario. Fuente: propia	78
B.8. Imagen de Resilience Preguntas de salud mental. Fuente: propia	79
B.9. Imagen de Resilience Inserción de estado del usuario. Fuente: propia	79
B.10.Imagen de Resilience Tests del usuario. Fuente: propia	80
B.11.Imagen de Resilience Test Gad Score. Fuente: propia	80
B.12.Imagen de Resilience Videos Recomendados por la Aplicación. Fuente: propia	81

Índice de cuadros

4.1. Caso de uso Inicio de sesión	37
4.2. Caso de uso Gestión de perfil	37
4.3. Caso de uso Registro diario de rutina y estado emocional	37
4.4. Caso de uso Procesamiento datos con IA	37
4.5. Caso de uso Ejercicios Diario de respiración	38
4.6. Caso de uso Historial y gráficos	38
4.7. Caso de uso Privacidad y seguridad	38
4.8. Caso de uso Ver Perfil	38
4.9. Caso de uso Editar información Personal	39
4.10. Caso de uso Modificar contraseña	39
4.11. Caso de uso Configurar privacidad	39
4.12. Caso de uso Eliminar Cuenta	39
4.13. Caso de uso Registrar Estado Anímico	40
4.14. Caso de uso Registrar Horas de Sueño	40
4.15. Caso de uso Registrar Nivel de estrés	40
4.16. Caso de uso Registrar Receptividad	40
4.17. Caso de uso Registrar Motivación	41
4.18. Caso de uso Registrar Ganas de Vivir	41
4.19. Caso de uso Registrar Ganas de Comer	41
4.20. Caso de uso Registrar Apatía	41
4.21. Caso de uso Registrar Actividad Deportiva	42
4.22. Caso de uso Registrar Estado Emocional	42
4.23. Caso de uso Ingresar Parámetros Salud Mental	42
4.24. Caso de uso Procesar datos con IA	42
4.25. Caso de uso Acceder pestaña videos	43
4.26. Caso de uso Escuchar Charlas relajantes	43
4.27. Caso de uso Ver videos de Yoga	43
4.28. Caso de uso Ver videos de Meditación	43
4.29. Caso de uso Ver Gráfico de Estado Mental	44
4.30. Caso de uso Revisar Historial de Accesos	44

Capítulo 1

Introducción

1.1. Motivación

1.1.1. Motivación General

El ámbito de la salud mental ha sido tradicionalmente abordado desde una perspectiva clínica y personalizada. Sin embargo, con el avance tecnológico y la digitalización, surgen oportunidades para desarrollar herramientas innovadoras que puedan proporcionar un apoyo adicional tanto a profesionales de la salud mental como a los propios usuarios. Esta aplicación nace de la necesidad de brindar una plataforma integral que permita la gestión de usuarios y sus datos relacionados con la salud mental, y al mismo tiempo, proporcione una evaluación en tiempo real del nivel de estrés de los usuarios mediante algoritmos de inteligencia artificial. La finalidad es doble: por un lado, otorgar a los profesionales una herramienta que facilite la identificación temprana de niveles elevados de estrés y, por otro, empoderar a los usuarios al proporcionarles información objetiva y actualizada sobre su propio estado emocional, sabiendo si podría necesitar ayuda. De esta manera, la aplicación se posiciona en la intersección de la psicología y la tecnología, abriendo puertas para intervenciones más proactivas y basadas en datos en el campo de la salud mental.

1.1.2. Motivaciones Técnicas

1.1.2.1. Elección de Microservicios

Optamos por una arquitectura de microservicios debido a su escalabilidad y flexibilidad. Esta arquitectura permite descomponer la aplicación en componentes más pequeños y autónomos, facilitando la adaptación y mejora continua. Además, el uso de microservicios facilita la gestión de diferentes aspectos de la aplicación de manera independiente, como la gestión de usuarios y los datos de salud, lo que proporciona una robustez y resiliencia mayores en caso de fallos o necesidades de actualización.

1.1.2.2. Elección de MongoDB y MySQL

La elección de MongoDB está motivada por su naturaleza de base de datos NoSQL, lo que la hace adecuada para manejar grandes volúmenes de datos estructurados y no estructurados de manera eficiente. En el contexto de salud

mental, los datos pueden variar ampliamente y MongoDB permite una adaptabilidad y escalabilidad adecuadas. Por otro lado, MySQL se eligió por ser una solución de base de datos relacional consolidada, excelente para gestionar datos estructurados con relaciones bien definidas, como la información de usuario.

1.1.2.3. Elección de React

React fue seleccionado para el Frontend por ser una biblioteca de JavaScript eficiente, multiplataforma y flexible para construir interfaces de usuario. Ofrece un modelo de componentes reutilizables, lo que facilita el desarrollo y mantenimiento. Además, su naturaleza virtual DOM permite actualizaciones rápidas y eficientes, proporcionando una experiencia fluida al usuario. Aunque existen otras soluciones en el mercado, como Vue o Angular, React se destaca por su gran comunidad, amplia documentación y adaptabilidad a diferentes contextos de desarrollo.

1.2. Objetivos

1.2.1. Objetivo Principal

Desarrollar una aplicación integrada en el ámbito de la salud mental que, a través de la gestión de datos y algoritmos de inteligencia artificial, evalúe, aconseje y proporcione retroalimentación en tiempo real sobre el nivel de estrés de los usuarios o posible tratamiento, contribuyendo así a una detección temprana y una mejor intervención en problemas relacionados con el bienestar emocional.

1.2.2. Objetivos Específicos

1. **Gestión Eficiente de Usuarios:** Implementar un sistema robusto y seguro de gestión de usuarios que permita el registro, autenticación y actualización de datos personales y de salud mental de forma intuitiva y amigable para el usuario.
2. **Recopilación y Almacenamiento de Datos:** Diseñar una estructura de base de datos que sea capaz de almacenar y gestionar tanto datos estructurados (mediante MySQL) como no estructurados (mediante MongoDB), garantizando la integridad y privacidad de la información.
3. **Uso de Algoritmos de IA:** Usar un algoritmo entrenado de inteligencia artificial que, con base en los datos proporcionados por el usuario, evalúen el nivel de estrés y generen recomendaciones o alertas según sea necesario.
4. **Interfaz Amigable:** Utilizar React para desarrollar una interfaz de usuario moderna, intuitiva y responsive que permita una interacción fluida y una navegación sencilla por la plataforma.
5. **Integración de Microservicios:** Implementar una arquitectura de microservicios que permita la modularidad y escalabilidad del proyecto, facilitando su mantenimiento y posibles expansiones en el futuro.

6. **Promoción del Autocuidado:** A través de la aplicación, fomentar que los usuarios tomen conciencia de su salud mental, proporcionando herramientas e información que les permitan adoptar prácticas de autocuidado y, en caso necesario, buscar ayuda profesional.
7. **Feedback Continuo:** Establecer mecanismos de retroalimentación para que los usuarios puedan compartir sus experiencias, sugerencias o problemas con la aplicación, permitiendo así mejoras constantes en la plataforma.
8. **Adaptabilidad y Expansión:** Construir la aplicación de manera que pueda adaptarse a nuevas necesidades o requerimientos en el ámbito de la salud mental, y que pueda ser expandida con nuevas funcionalidades o módulos en el futuro.

1.3. Diagrama de Gantt

Un diagrama o gráfico de Gantt es una herramienta esencial en la gestión de proyectos, ya que proporciona una proyección temporal detallada de cada sección del proyecto. Su utilidad radica en ofrecer una visión panorámica del consumo de recursos en cada etapa, facilitando así una organización eficiente [22].

En lo que respecta a mi proyecto, se ha puesto un énfasis considerable en el diseño y desarrollo del backend, frontend y el entrenamiento de la Inteligencia Artificial. En la práctica, esta priorización ha sido acertada, ya que estas áreas han requerido la mayor cantidad de tiempo y esfuerzo. El aprendizaje de nuevos lenguajes de programación, el montaje de microservicios y la conexión y dockerización de los mismos han sido desafíos notables en el desarrollo del backend. Por otra parte, el desarrollo del frontend ha sido una tarea tan amigable como engorrosa en según qué momento. A pesar de esto, he quedado satisfecho con el resultado y considero que cumple con los objetivos que nos propusimos.

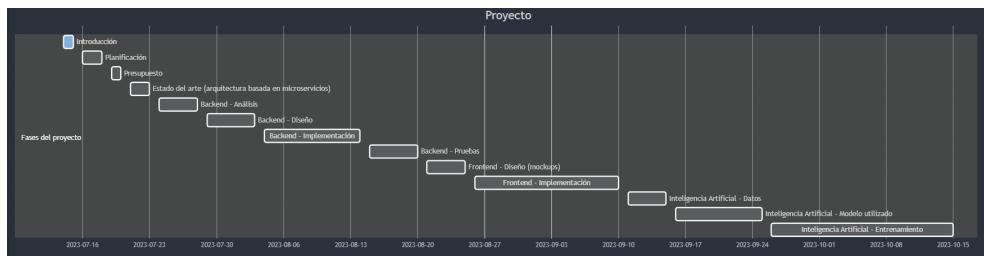


Figura 1.1: Diagrama de Gantt

1.4. Aplicaciones relacionadas con la salud mental

1.4.1. Ejemplos actuales en diferentes sectores

- **Daylio[10]:** Esta es una aplicación de registro de humor que permite a los usuarios llevar un diario de sus estados de ánimo diarios y actividades. Al igual que tu propuesta, Daylio se centra en registrar el estado de

árbito del usuario y, basándose en ello, proporciona insights y patrones relacionados con sus actividades y estados emocionales.

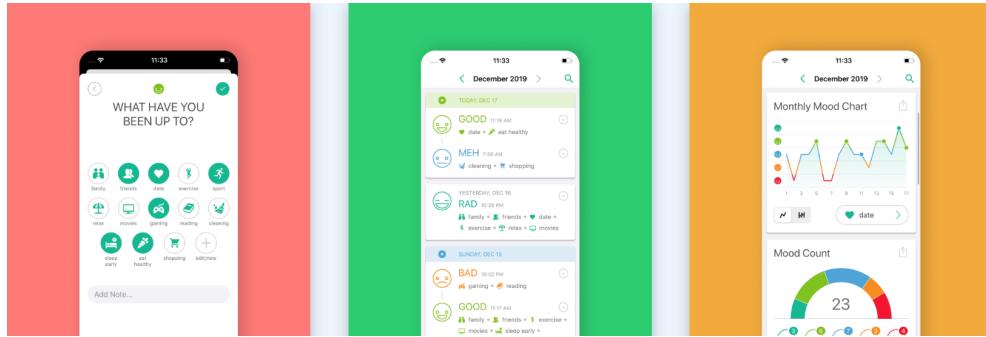


Figura 1.2: Imagen de Daylio. Fuente: Daylio

- **Moodpath[26]:** Es una aplicación diseñada para rastrear y mejorar la salud mental. Los usuarios responden preguntas diarias para evaluar su bienestar y, a partir de estas respuestas, la aplicación ofrece recursos y recomendaciones. Además de registrar el estado de ánimo, Moodpath también ofrece contenido educativo y recomendaciones basadas en la salud mental del usuario, similar a tu idea de ofrecer videos de respiración y yoga.

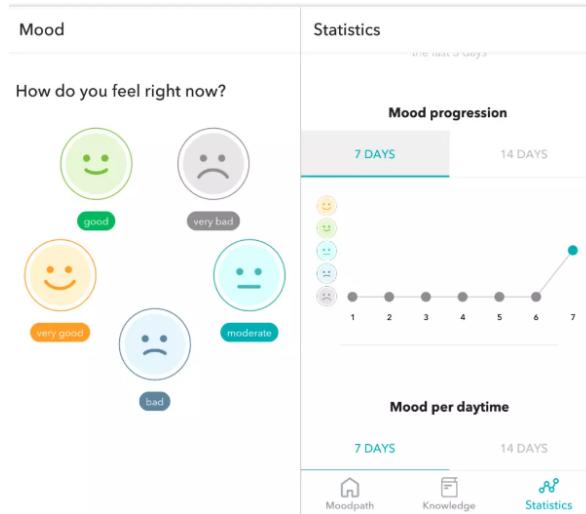


Figura 1.3: Imagen de MoodPath. Fuente: MoodPath

- **My Possible Self[28]:** La similitud con esta aplicación permite a los usuarios rastrear su estado de ánimo y ofrece módulos terapéuticos basados en la terapia cognitivo-conductual (TCC) para ayudar a los usuarios a abordar problemas específicos. Combina seguimiento de humor con intervenciones y recursos específicos para mejorar el bienestar.

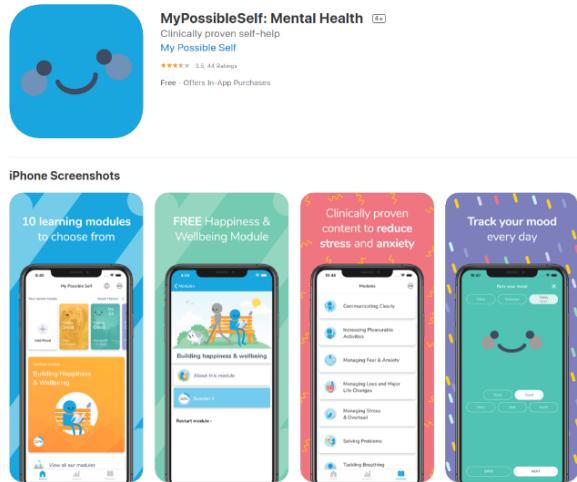


Figura 1.4: Imagen de PossibleSelf. Fuente: PossibleSelf

1.4.2. Aplicación de inspiración

1.4.2.1. Clue:[9]

1. **Enfoque Específico:** Clue es una aplicación centrada en el seguimiento del ciclo menstrual. Su diseño y funcionalidades están orientados principalmente a ayudar a las personas a entender su ciclo, predecir fechas de menstruación y ovulación, y registrar síntomas relacionados.
2. **Registro Detallado en su Nicho:** Si bien Clue se centra en un área específica de la salud, ofrece un registro detallado dentro de ese nicho, permitiendo a las usuarias hacer un seguimiento de síntomas, estados de ánimo, flujo, entre otros, relacionados con el ciclo menstrual.
3. **Educación y Conciencia:** Clue no solo se basa en el registro de datos, sino que también se esfuerza por educar a sus usuarios sobre salud reproductiva, ofreciendo insights basados en los datos ingresados y proporcionando información contextual sobre el ciclo menstrual.

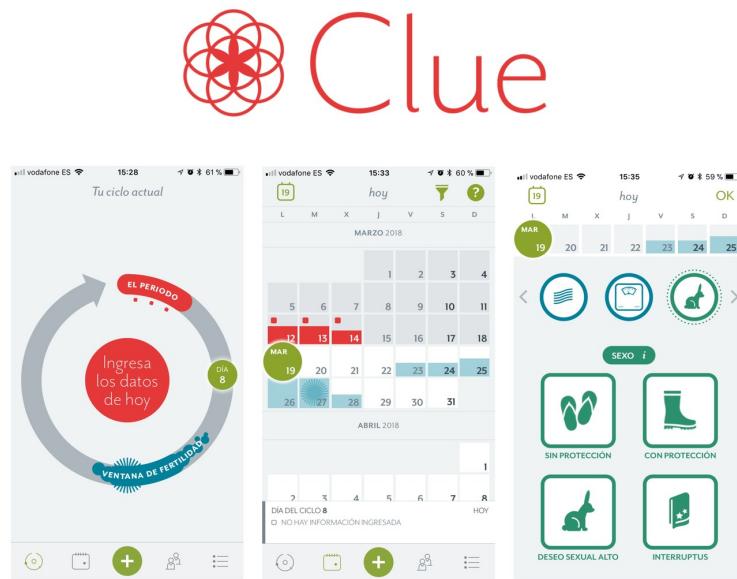


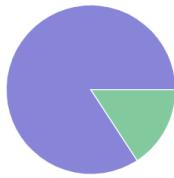
Figura 1.5: Imagen de Clue. Fuente: Clue

1.4.2.2. Resilience:

1. **Enfoque Holístico:** Esta aplicación tiene un enfoque amplio y holístico al permitir a los usuarios registrar datos relacionados con su estado de ánimo, valores de salud, sueño, entre otros. Esto proporciona una visión completa de la salud física y mental del individuo.
2. **Recomendaciones Personalizadas:** Además del registro de datos, la aplicación propuesta ofrece recomendaciones adaptadas a los patrones de salud y bienestar del usuario, como videos de respiración y yoga, lo que potencia una experiencia más interactiva y personalizada.
3. **Diversidad de Métricas:** A diferencia de muchas aplicaciones especializadas, la propuesta integra una variedad de métricas que se extienden más allá de un área específica, abordando tanto el bienestar físico como el mental.

jHola, Juan Carlitos !

Aquí tienes un sencillo diagrama de cómo te tiene sentido en los últimos 9 días



Vaya, parece que estos días has estado de capa caída ¡No te preocupes!
Recuerda que hay muchas herramientas haciendo clic en la libreta que te pueden ayudar
Sin embargo, si ves que esta situación se alarga,
siempre puedes buscar ayuda.



Figura 1.6: Imagen de Resilience. Fuente: elaboración propia

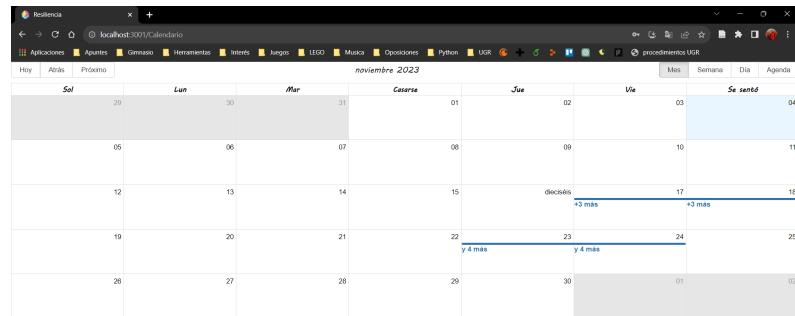


Figura 1.7: Imagen de Resilience. Fuente: elaboración propia

1.4.3. Potencial de crecimiento y futuras tendencias.

El mercado de aplicaciones de salud y bienestar ha experimentado un crecimiento exponencial en los últimos años, impulsado por una creciente conciencia sobre la importancia del cuidado personal, el bienestar mental y la salud preventiva.[32] Esta aplicación, que combina el registro del estado de ánimo con valores de salud, sueño y recomendaciones personalizadas, se posiciona en un nicho que tiene el potencial de abordar múltiples necesidades de los usuarios en un solo lugar. Esta integración de características es particularmente relevante en el contexto actual, donde los individuos buscan soluciones holísticas para mejorar su calidad de vida, y donde el bienestar mental y físico se considera intrínsecamente entrelazado.

Mirando hacia el futuro, se espera que la personalización, impulsada por la inteligencia artificial y el aprendizaje automático, domine el panorama de las aplicaciones de salud y bienestar. Las tendencias apuntan hacia un futuro en el que las aplicaciones serán capaces de ofrecer recomendaciones y consejos aún

más adaptados a las necesidades y circunstancias específicas de cada usuario. En este escenario, tu aplicación tiene el potencial de evolucionar al integrar análisis predictivos que puedan anticipar desafíos de salud o bienestar y ofrecer soluciones proactivas.

Además, la creciente integración de dispositivos portátiles y la Internet de las Cosas (IoT) en el ámbito de la salud^[2] presenta una oportunidad para que tu aplicación se sincronice y recoja datos de una variedad de fuentes, como relojes inteligentes, monitores de sueño o incluso ropas inteligentes. Esta tendencia hacia la conectividad aumentará la precisión y relevancia de las recomendaciones, mejorando la experiencia del usuario y, en última instancia, sus resultados de salud.

Finalmente, a medida que más personas en todo el mundo adopten prácticas de mindfulness, meditación y técnicas de relajación para enfrentar el estrés y los desafíos de la vida moderna, el potencial de crecimiento de tu aplicación se amplía. Al ofrecer una plataforma que no solo registra y rastrea, sino que también educa y empodera a los usuarios con herramientas tangibles como videos de respiración y yoga, tu aplicación está bien posicionada para liderar y adaptarse a las futuras tendencias en el ámbito del bienestar digital.

Capítulo 2

Fundamentos teóricos

2.1. Arquitectura de microservicios

Los microservicios representan un enfoque arquitectónico en el que una aplicación se compone de pequeños servicios autónomos que se ejecutan en su propio proceso y se comunican entre sí a través de mecanismos ligeros, generalmente APIs HTTP[13]. Cada servicio es altamente desacoplado y representa una pequeña funcionalidad específica de la aplicación global[16]

2.1.1. Historia de los Microservicios

La evolución y los fundamentos de los microservicios representan un esfuerzo constante por mejorar la interacción entre diversas plataformas, simplificar procesos y crear sistemas más intuitivos. Generalmente, se entiende que los microservicios son una estrategia de desarrollo de software que estructura una aplicación como una colección de servicios que interactúan entre sí de manera flexible. Estos servicios, pequeños y autónomos, colaboran para el funcionamiento de una aplicación de software, utilizando servicios minuciosos y protocolos eficientes.[40]

Fue en 2005 cuando el Dr. Peter Rodgers introdujo el concepto de "Micro-Web-Services." en un evento sobre computación en la nube, desafiando las ideas tradicionales y abogando por componentes de software que soportaran los microservicios en la web [12]. Durante su charla, propuso un modelo de microservicios que más tarde se convertiría en una realidad. Al detallar el funcionamiento de complejos conjuntos de servicios detrás de sencillas interfaces URI, planteó la posibilidad de exponer cualquier servicio, independientemente de su tamaño. Además, describió cómo una plataforma de microservicios web bien estructurada podría incorporar los principios fundamentales de los servicios web y REST, así como técnicas de programación y flujos de trabajo al estilo de Unix, para alcanzar una flexibilidad sin precedentes y una simplicidad superior en arquitecturas basadas en servicios.

El concepto de microservicios no es nuevo, pero ha ganado popularidad en la última década. Antes de los microservicios, la tendencia predominante era la arquitectura monolítica, donde todas las funcionalidades de una aplicación residían en una única unidad indivisible. Sin embargo, con el crecimiento

de las aplicaciones y la necesidad de escalabilidad, mantener y evolucionar aplicaciones monolíticas se volvía complejo.

En la década de 2010, empresas como Netflix, Amazon y Spotify comenzaron a adoptar una arquitectura de microservicios para abordar los desafíos de escalabilidad y acelerar el ciclo de desarrollo. Esta adopción por parte de grandes empresas tecnológicas impulsó la popularidad del enfoque de microservicios. [7]

2.1.2. Arquitectura monolítica

El término “arquitectura monolítica” se refiere a un enfoque de desarrollo de software en el que todos los componentes de una aplicación están interconectados y funcionan como un solo sistema unificado. Este enfoque ha sido la norma en la industria del software durante mucho tiempo. [29]

En una arquitectura de este tipo, la aplicación se desarrolla y se despliega como una sola unidad, sin importar cuán modular sea el código fuente. Esto significa que cualquier cambio, por pequeño que sea, requiere que toda la aplicación sea reconstruida y desplegada. [1]

Para escalar una aplicación monolítica, normalmente se ejecutan múltiples copias idénticas de la aplicación detrás de un balanceador de carga. Los componentes o módulos suelen estar altamente acoplados, lo que significa que un cambio en un área puede tener efectos en otras áreas de la aplicación.[1]

Una ventaja de este enfoque es que, dado que todo opera en un solo entorno, se garantiza la consistencia en las operaciones, especialmente las transaccionales. Además, en las primeras etapas de desarrollo, es más fácil y rápido construir y probar aplicaciones monolíticas. Esto es particularmente cierto para proyectos más pequeños con requisitos menos complejos. [11]

Sin embargo, a medida que la aplicación crece, puede ser más difícil hacer cambios, escalar y mantener el código, especialmente si no se siguen buenas prácticas de programación. En la mayoría de los casos, las aplicaciones monolíticas están escritas en un solo lenguaje de programación.[1]

Un inconveniente importante es que si una parte específica de la aplicación falla, puede afectar a toda la aplicación debido a su interrelación. Aunque el código puede estar modularizado en diferentes archivos o módulos, generalmente se despliega como un solo artefacto.[29]

Las arquitecturas monolíticas pueden ser adecuadas para aplicaciones más simples o para proyectos con requisitos bien definidos que tienen menos probabilidad de cambiar. Sin embargo, las aplicaciones más grandes y complejas a menudo enfrentan desafíos en términos de escalabilidad, mantenibilidad y adaptabilidad con este enfoque. Por esta razón, algunas organizaciones están migrando hacia arquitecturas basadas en microservicios, que ofrecen mayor

flexibilidad pero también presentan sus propios desafíos.[11]

2.1.3. Arquitecturas monolíticas y microservicios

En la construcción de sistemas de software, las arquitecturas tradicionales, a menudo denominadas monolíticas, y las basadas en servicios independientes, conocidas comúnmente como microservicios, representan dos enfoques distintos. Ambos tienen ventajas y desafíos inherentes, lo que se refleja en la siguiente comparativa [27][37]:

- **Desarrollo y Despliegue:** En un sistema monolítico, cualquier modificación requiere desplegar toda la aplicación nuevamente, lo que a veces puede ser un proceso más prolongado. En contraste, las arquitecturas basadas en servicios independientes permiten que cada servicio sea desarrollado y desplegado de forma individual, ofreciendo más agilidad y rapidez en la incorporación de cambios.
- **Escalabilidad:** Las aplicaciones tradicionales se escalan duplicando la aplicación en su totalidad en varios servidores o instancias. Por otro lado, los sistemas basados en microservicios permiten escalar solo aquellos servicios que enfrentan una mayor demanda, optimizando recursos.
- **Acoplamiento:** Los sistemas tradicionales tienden a tener un alto grado de interdependencia entre sus módulos. Los servicios independientes promueven un bajo acoplamiento, operando de forma autónoma y comunicándose mediante interfaces bien definidas.
- **Flexibilidad de Tecnologías:** En un enfoque monolítico, generalmente, se está restringido a una sola tecnología o stack. Las arquitecturas modernas de servicios permiten que cada uno pueda ser desarrollado en la tecnología más adecuada para su propósito.
- **Resiliencia:** En un sistema monolítico, un error en una parte puede comprometer toda la aplicación. Con los servicios independientes, un fallo en uno de ellos generalmente no afecta a los demás, garantizando una mayor disponibilidad.
- **Mantenimiento:** Con el tiempo, las aplicaciones monolíticas pueden volverse más difíciles de mantener debido a su creciente complejidad. Las arquitecturas de microservicios, al estar modularizadas, facilitan el mantenimiento y la iteración rápida.
- **Coherencia Operativa:** Mientras que las operaciones en un monolito suelen ser consistentes debido a un entorno unificado, los sistemas basados en servicios requieren estrategias adicionales para garantizar la consistencia en las transacciones que abarcan múltiples servicios.
- **Latencia:** En un monolítico, la comunicación entre módulos suele ser más directa y rápida. Los servicios independientes, al comunicarse a través de la red, pueden introducir latencia adicional, aunque con las herramientas adecuadas, este impacto puede minimizarse.

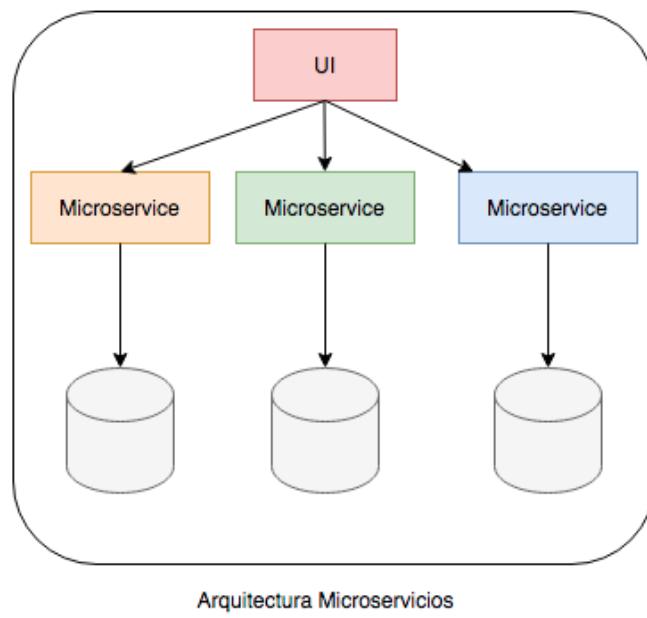


Figura 2.1: Imagen del esquema de microservicios. Fuente: Google imágenes

2.1.4. Ventajas y desventajas de los Microservicios

La arquitectura de microservicios ha emergido como una solución poderosa frente a las aplicaciones monolíticas tradicionales, ofreciendo mayor flexibilidad y escalabilidad. Aunque es ampliamente adoptada por su modularidad y eficiencia, es importante considerar tanto sus beneficios como sus desafíos al aplicarla en entornos de desarrollo de software. [38]

2.1.4.1. Ventajas

- **Escalabilidad:** Los servicios individuales pueden escalar de manera independiente según las demandas.
- **Desarrollo Paralelo:** Los equipos pueden trabajar en diferentes servicios al mismo tiempo, acelerando el desarrollo.
- **Despliegue Independiente:** Un cambio en un servicio no requiere el despliegue de toda la aplicación.
- **Tolerancia a Fallos:** Si un servicio falla, no significa que toda la aplicación se vea comprometida.
- **Flexibilidad Tecnológica:** Diferentes servicios pueden utilizar diferentes tecnologías, lenguajes o bases de datos.

2.1.4.2. Desventajas

- **Complejidad Añadida:** La gestión de múltiples servicios puede ser más compleja que una aplicación monolítica.

- **Latencia:** La comunicación entre servicios puede introducir latencia.
- **Desafíos de Seguridad:** Se deben establecer protocolos de seguridad entre servicios.
- **Gestión de Datos:** La coherencia y la gestión de datos entre servicios pueden ser desafiantes.

2.1.5. Características generales de los microservicios

Tal y como se ha mencionado anteriormente, los microservicios tienen unas características que le diferencian del software monolítico [21] y hacen este tipo de software más complejo, pero también nos aportan numerosos beneficios ya mencionados anteriormente basados en la modularidad de este tipo de software que hacen que esta arquitectura sea una elección poderosa para el desarrollo de aplicaciones modernas y escalables [16].

Descomposición en Componentes Independientes: La arquitectura de microservicios descompone una aplicación en una serie de componentes autónomos e independientes. Cada uno de estos componentes se enfoca en realizar una tarea específica y concreta. Esta descomposición en componentes más pequeños permite que el desarrollo y el mantenimiento sean más manejables y escalables.

Comunicación a través de API: Los microservicios se comunican entre sí a través de APIs (Interfaz de Programación de Aplicaciones). La comunicación generalmente se realiza utilizando protocolos como HTTP/HTTPS. Esta comunicación basada en API permite que los microservicios interactúen de manera eficiente y se comuniquen de manera segura.

Despliegue Independiente: Cada microservicio se puede desarrollar, probar y desplegar de forma independiente de los demás. Esto significa que los cambios o actualizaciones en un microservicio no afectan a los demás. La independencia en el despliegue facilita la entrega continua y acelera el ciclo de desarrollo.

Escalabilidad Individual: Los microservicios ofrecen la capacidad de escalar cada componente por separado según la demanda. Esto mejora la eficiencia de recursos, ya que solo los microservicios que necesitan escalar lo hacen, en lugar de tener que escalar toda la aplicación.

Tecnología y Lenguaje Diversificados: Los microservicios pueden estar escritos en diferentes lenguajes de programación y utilizar tecnologías distintas según las necesidades específicas de cada servicio. Esta flexibilidad permite elegir la tecnología adecuada para cada componente y optimizar su rendimiento.

Gestión de Datos Propia: Cada microservicio puede tener su propia base de datos o sistema de almacenamiento de datos. Esto facilita la gestión

de datos relacionados con la funcionalidad específica de cada servicio y evita problemas de interferencia entre servicios.

Tolerancia a Fallos: Los microservicios están diseñados para ser tolerantes a fallos. Si uno de ellos experimenta un fallo o un problema, no debería afectar a los demás. Esto aumenta la robustez y la resiliencia del sistema en su conjunto.

Fácil Mantenimiento y Evolución: La modularidad de los microservicios facilita el mantenimiento y la actualización de la aplicación en el tiempo. Se pueden agregar nuevos servicios o realizar cambios en los existentes sin afectar a otros componentes, lo que simplifica la gestión del software a largo plazo.

Agilidad en el Desarrollo: Los equipos de desarrollo pueden trabajar en diferentes microservicios de forma paralela. Esto acelera el desarrollo y permite la entrega más rápida de nuevas funcionalidades. Además, cada equipo puede estar especializado en un área particular de la aplicación.

Complejidad en la Gestión: A pesar de las ventajas, la gestión de múltiples microservicios puede introducir complejidad en términos de coordinación, seguridad y monitoreo. Es importante implementar prácticas de gestión adecuadas y herramientas de monitoreo para garantizar un funcionamiento eficiente y seguro del sistema.

2.2. Inteligencia artificial

2.2.1. Historia y Evolución

La inteligencia artificial es una disciplina dentro de la informática que busca emular la inteligencia humana mediante algoritmos y sistemas computacionales. Su origen y concepto inicial se puede trazar hasta pensadores clásicos que soñaron con la idea de máquinas que podrían imitar al ser humano. Sin embargo, el término "inteligencia artificial" fue acuñado en 1956 por John McCarthy para la conferencia de Dartmouth, marcando el nacimiento oficial de la IA como un campo académico[8].

2.2.1.1. Primeros Años (1950s-1970s):

Investigaciones Pioneras: Estudio de redes neuronales, máquinas de Turing y la idea de máquinas universales. Desarrollos Iniciales: Creación de ELIZA y SHRDLU, sistemas capaces de procesar el lenguaje natural para interactuar con humanos.

Retos: Limitaciones tecnológicas y falta de capacidad de procesamiento.

2.2.1.2. Años Intermedios (1980s-2000s):

Auge de los Sistemas Expertos: Máquinas diseñadas para imitar la habilidad de expertos humanos en campos específicos. Desarrollo de Algoritmos:

Algoritmos genéticos, redes neuronales y máquinas de soporte vectorial comienzan a ser más populares. Financiamiento y Soporte: Aumento en la inversión en investigación de IA por parte de empresas y gobiernos. [20] [7]

2.2.1.3. Era Moderna (2010s-Presente):

Aprendizaje Profundo: Con el resurgimiento de las redes neuronales, especialmente las redes neuronales convolucionales (CNN) y las redes neuronales recurrentes (RNN), la IA ha alcanzado capacidades sin precedentes en áreas como visión por computadora y procesamiento de lenguaje natural. Aplicaciones Reales: Autopilotos de coches, asistentes virtuales, sistemas de recomendación y diagnóstico médico, por mencionar algunos.

2.2.2. Principales Subcampos de la IA

- **Aprendizaje Automático (Machine Learning):** Rama de la IA que permite a las máquinas aprender de los datos. Usa estadísticas para encontrar patrones en datos masivos.
- **Aprendizaje Profundo (Deep Learning):** Subconjunto del aprendizaje automático que usa redes neuronales con muchos niveles (o capas) para analizar diferentes tipos de datos.
- **Procesamiento de Lenguaje Natural (NLP):** Se centra en la interacción entre las computadoras y el lenguaje humano, permitiendo que las máquinas entiendan y respondan a entradas de lenguaje humano.
- **Robótica:** Diseño y creación de robots que pueden realizar tareas sin intervención humana.
- **Visión por Computadora:** Permite a las máquinas interpretar y decidir a partir de información visual, similar a la vista humana.

2.2.3. Herramientas y Frameworks

La evolución de la IA ha sido posible gracias a la aparición de herramientas y frameworks especializados que facilitan el diseño, entrenamiento y despliegue de modelos de IA.

- **TensorFlow:** Desarrollado por Google Brain, es uno de los frameworks más populares para el aprendizaje profundo. Es conocido por su flexibilidad y capacidad para trabajar con grandes conjuntos de datos. [39]
- **PyTorch:** Desarrollado por Facebook, es otra herramienta popular para el aprendizaje profundo, conocida por su dinámica de cómputo y facilidad para la investigación académica.[33]
- **Keras:** Interfaz de alto nivel para redes neuronales, que puede funcionar sobre TensorFlow o Theano, permitiendo una prototipación rápida. [24]
- **Scikit-learn:** Biblioteca de Python centrada en herramientas de aprendizaje automático para minería y análisis de datos. [36]
- **NLTK y SpaCy:** Bibliotecas enfocadas en el procesamiento de lenguaje natural, ofrecen herramientas para el análisis de texto. [23]

2.2.4. Conclusión

La inteligencia artificial ha experimentado un rápido crecimiento y evolución desde sus inicios, impulsada por los avances tecnológicos y la creciente disponibilidad de datos. Los desarrollos actuales en IA prometen revolucionar múltiples sectores y seguirán siendo un área de investigación y desarrollo clave en el futuro previsible.

Capítulo 3

Intersección entre Microservicios e Inteligencia Artificial

La convergencia de la Inteligencia Artificial (IA) y los microservicios representan una de las uniones más prometedoras en el mundo de la tecnología de la información. Esta intersección ofrece a las organizaciones la capacidad de aprovechar la escalabilidad y flexibilidad de los microservicios junto con el poder de análisis y aprendizaje de los modelos de IA. Veamos cómo se intersectan y las ventajas que ofrecen[15][29]:

3.1. Escalabilidad

En cuanto a los microservicios, se refiere a la capacidad de un sistema para manejar un aumento en la carga de trabajo o demanda, añadiendo recursos al sistema.

La escalabilidad se manifiesta en la habilidad de una aplicación de manejar un incremento en las solicitudes añadiendo más instancias de un servicio específico. En lugar de escalar toda la aplicación monolítica, solo se escalan los servicios que están experimentando una mayor demanda. Costo-efectividad, ya que solo se incrementan los recursos donde se necesita.

Los modelos de IA, dependiendo de su complejidad, pueden consumir muchos recursos durante el entrenamiento y la inferencia. Si un modelo de IA necesita más recursos, el servicio que lo contiene puede escalar para satisfacer esa demanda sin afectar otros servicios. Permite a las organizaciones desplegar modelos de IA más complejos y de mayor calidad sin comprometer el rendimiento del sistema global.

3.2. Rapidez en la Iteración

Para los microservicios, la rapidez con la que se pueden hacer cambios en una aplicación y poner esos cambios en producción.

Al ser independientes, los microservicios permiten a los equipos desarrollar,

probar y desplegar servicios individuales sin tener que reconfigurar o redeployar toda la aplicación.

Los algoritmos y modelos de IA requieren iteraciones constantes para mejorar su precisión y rendimiento. La arquitectura de microservicios facilita estas iteraciones rápidas, permitiendo actualizar un modelo o algoritmo y desplegarlo en producción sin afectar otros componentes.[7]

3.3. Políglota

La capacidad de utilizar múltiples lenguajes de programación dentro de una única aplicación. Los microservicios permiten que cada servicio se desarrolle en el lenguaje que sea más adecuado para esa tarea específica, maximizando la eficiencia y el rendimiento.[7]

Diferentes bibliotecas y herramientas de IA tienen preferencias de lenguaje. Al utilizar una arquitectura de microservicios, se puede elegir el mejor lenguaje y herramienta para cada modelo o algoritmo específico.[7]

3.4. Reutilización

Una vez que un modelo de IA se encapsula dentro de un microservicio, ese servicio se convierte en un recurso reutilizable. Puede ser invocado por diferentes partes de la aplicación o incluso por diferentes aplicaciones, promoviendo la coherencia y reduciendo el esfuerzo duplicado.[7]

3.5. Desacoplamiento

Los microservicios permiten separar diferentes funcionalidades de una aplicación en servicios independientes que interactúan entre sí a través de interfaces bien definidas.[27]

Se puede mantener la lógica de IA separada de la lógica de negocio, permitiendo a los equipos especializados trabajar en lo que mejor saben hacer sin interferencias.[27]

3.6. Mantenimiento

Facilita la tarea de monitorear, mantener y solucionar problemas en una aplicación. Si surge un problema con un servicio específico, puede ser identificado, aislado y corregido sin tener que interferir con otros servicios. Esto es especialmente crucial en IA, donde un modelo puede necesitar ser recalibrado o reentrenado.[27]

Capítulo 4

Diseño y desarrollo del sistema

4.1. Definición del Problema

En el contexto actual de la tecnología, donde la rapidez, eficiencia y escalabilidad son esenciales para satisfacer las demandas cambiantes de los usuarios, el diseño y desarrollo de sistemas que integren tanto microservicios como Inteligencia Artificial (IA) se ha convertido en un reto crítico. El Trabajo Fin de Grado (TFG) busca abordar esta intersección compleja y multifacética. No obstante, el desafío radica no solo en la mera implementación técnica de estas soluciones, sino también en garantizar que el sistema resultante sea robusto, mantenable y capaz de evolucionar con las futuras demandas y avances en ambos campos. Al incorporar IA en una arquitectura basada en microservicios, se enfrentan problemáticas asociadas a la gestión adecuada de datos, la escalabilidad de los modelos de aprendizaje automático y la garantía de un tiempo de respuesta eficiente, sin sacrificar la precisión o la calidad de los resultados. Además, se suma la necesidad de diseñar microservicios que puedan comunicarse entre sí de manera fluida, proporcionando una solución integrada que sea cohesiva y coherente en su conjunto.

4.2. Análisis

4.2.1. Requisitos funcionales

- **Registro de Usuarios:** Los usuarios pueden registrarse en la aplicación proporcionando su correo electrónico y contraseña. Se deben validar las direcciones de correo electrónico para evitar registros duplicados.
- **Inicio de Sesión:** Los usuarios registrados pueden iniciar sesión en la aplicación utilizando su correo electrónico y contraseña.
- **Gestión de datos del usuario:** Los usuarios pueden editar su perfil, incluyendo la información personal, como nombre y foto de perfil.
- **Registro Diario de Rutina y Estado Emocional:** Los usuarios pueden ingresar datos diarios sobre su rutina, como horas de sueño, actividad

física, alimentación, etc. También pueden registrar su estado emocional diario utilizando una escala o selección de emociones.

- **Almacenamiento de Datos:** Los datos de rutina y estado emocional se almacenan de forma segura y se asocian con cada usuario.
- **Procesamiento de Datos con IA:** La IA analiza los datos de rutina y estado emocional para identificar patrones y tendencias a lo largo del tiempo. La IA proporciona estimaciones de si necesitará tratamiento en función de la rutina y el estado emocional del usuario.
- **Recomendaciones Personalizadas:** La aplicación ofrece recomendaciones personalizadas basadas en los datos recopilados y el análisis de la IA. Se proporcionan ejercicios y actividades para controlar la ansiedad, adaptados a las necesidades individuales de cada usuario.
- **Historial y Gráficos:** Los usuarios pueden consultar su historial de rutina y estado emocional en forma de gráficos y estadísticas. La aplicación muestra visualmente la evolución de los datos a lo largo del tiempo.
- **Privacidad y Seguridad:** Se deben implementar medidas de seguridad robustas para proteger los datos de los usuarios. Los datos personales y médicos de los usuarios deben mantenerse confidenciales y cumplir con las regulaciones de privacidad.
- **Soporte Técnico:** Los usuarios deben tener acceso a un soporte técnico para resolver problemas técnicos y responder a preguntas relacionadas con la aplicación.
- **Actualizaciones Continuas:** La aplicación debe recibir actualizaciones periódicas para mejorar la funcionalidad y abordar problemas de seguridad o rendimiento.
- **Compatibilidad con Múltiplataforma:** La aplicación debe ser compatible con dispositivos móviles (iOS y Android) y versiones de navegador web.

4.2.2. Requisitos no funcionales

- **Seguridad de Datos:** La aplicación debe cumplir con las normativas de seguridad de datos, asegurando que los datos personales y de salud mental de los usuarios estén protegidos contra accesos no autorizados.
- **Privacidad de Datos:** Debe garantizarse la privacidad de los datos del usuario, asegurando que la información sensible se maneje de manera confidencial y se obtenga el consentimiento informado de los usuarios para su procesamiento.
- **Escalabilidad:** La aplicación debe ser capaz de escalar para acomodar un crecimiento en el número de usuarios sin afectar significativamente el rendimiento.
- **Disponibilidad:** La aplicación debe estar disponible para su uso la mayor parte del tiempo. Se debe establecer un nivel de disponibilidad objetivo (por ejemplo, 99.9 %) y garantizar que se cumpla.

- **Rendimiento:** La aplicación debe ser eficiente en términos de velocidad de respuesta y rendimiento, especialmente al realizar análisis de datos con IA.
- **Interfaz de Usuario Amigable:** La interfaz de usuario debe ser intuitiva y fácil de usar, garantizando una experiencia agradable para los usuarios.
- **Compatibilidad de Plataformas:** La aplicación debe ser compatible con una variedad de dispositivos y navegadores web, incluyendo dispositivos móviles (iOS y Android) y navegadores populares.
- **Mantenibilidad:** La aplicación debe estar diseñada de manera que sea fácil de mantener y actualizar para corregir errores, agregar nuevas características o mejorar la seguridad.
- **Tiempo de Respuesta de la IA:** La IA utilizada en la aplicación debe tener un tiempo de respuesta razonable para proporcionar estimaciones de futuros valores de manera eficiente.
- **Documentación:** Debe proporcionarse documentación completa que incluya guías de usuario, manuales técnicos y documentación de seguridad y privacidad.
- **Cumplimiento Regulatorio:** La aplicación debe cumplir con las regulaciones y estándares de privacidad de datos aplicables en la jurisdicción en la que se utilice.
- **Copias de Seguridad y Recuperación de Datos:** Debe implementarse un sistema de copias de seguridad regular de los datos del usuario y una capacidad de recuperación en caso de fallos.
- **Tiempo de Respuesta a Notificaciones:** Las notificaciones y recordatorios de la aplicación deben entregarse con un tiempo de respuesta rápido para asegurar la participación del usuario.
- **Localización y Traducción:** Si la aplicación se dirige a una audiencia global, debe ser localizable y traducible a diferentes idiomas y regiones.
- **Testeo y Control de Calidad:** La aplicación debe someterse a pruebas exhaustivas para identificar y solucionar errores antes de su lanzamiento.
- **Actualizaciones de Software:** Se deben implementar procesos para facilitar actualizaciones de software sin interrupciones significativas en el servicio.

4.2.3. Restricciones semánticas

El desarrollo y operación de la aplicación de psicología debe respetar las siguientes restricciones semánticas, derivadas de los requisitos funcionales y no funcionales:

1. **Unidad del Usuario:** Cada dirección de correo electrónico proporcionada para el registro de usuarios debe ser única. Esto asegura que cada usuario tenga un identificador único y evita duplicidades.

2. **Integridad de Datos:** Todos los datos ingresados por el usuario, ya sean datos de rutina, estado emocional o información personal, deben ser validados y almacenados de manera íntegra y consistente.
3. **Personalización Basada en Datos:** Las recomendaciones y análisis proporcionados por la IA deben ser generados exclusivamente a partir de los datos recopilados del usuario correspondiente, asegurando personalización y precisión.
4. **Integridad de la Sesión:** La aplicación debe manejar correctamente la creación, duración y terminación de sesiones de usuario para proteger su privacidad y seguridad.

Al seguir estas restricciones semánticas, se garantiza que la aplicación no solo cumple con las expectativas técnicas y funcionales, sino también con las éticas y legales, brindando un servicio seguro, confiable y valioso para sus usuarios.

4.2.4. Casos de uso

4.2.4.1. Registro de usuarios



Figura 4.1: Caso de uso Inicio de sesión

4.2.4.2. Gestión de perfil

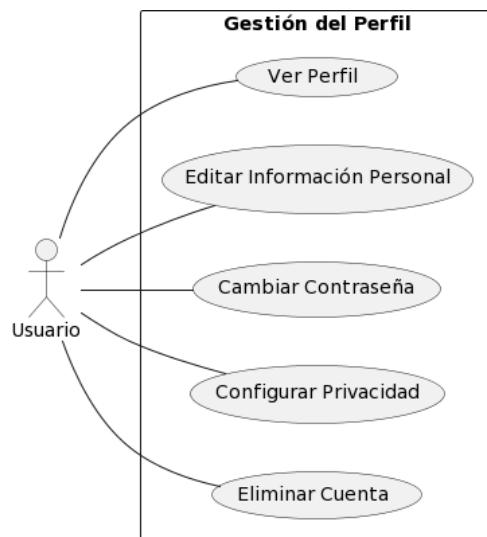


Figura 4.2: Caso de uso Gestión de perfil

4.2.4.3. Registro Diario de Rutina y Estado Emocional.

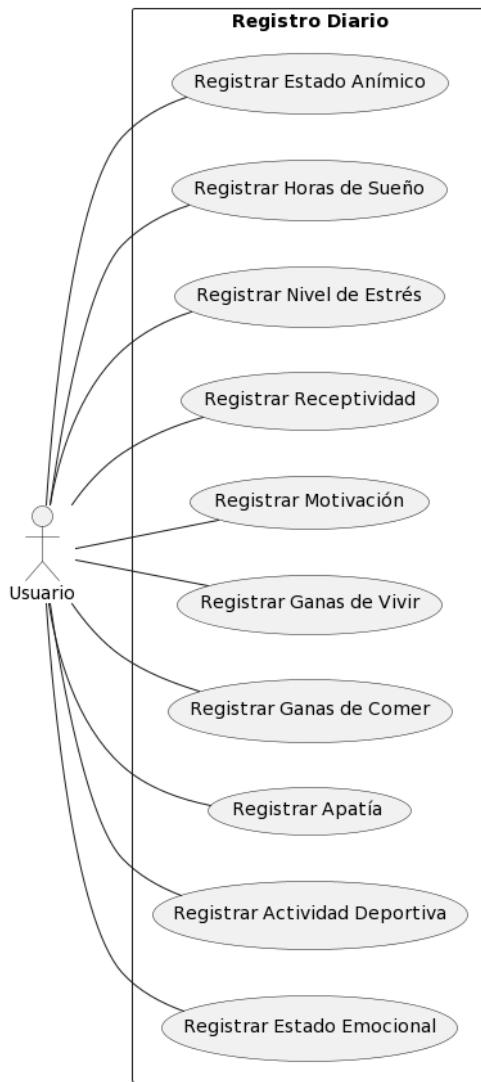


Figura 4.3: Caso de uso Registro Diario de Rutina y Estado Emocional

4.2.4.4. Procesamiento de Datos con IA



Figura 4.4: Caso de uso Procesamiento de Datos con IA

4.2.4.5. Ejercicios Diarios de Respiración

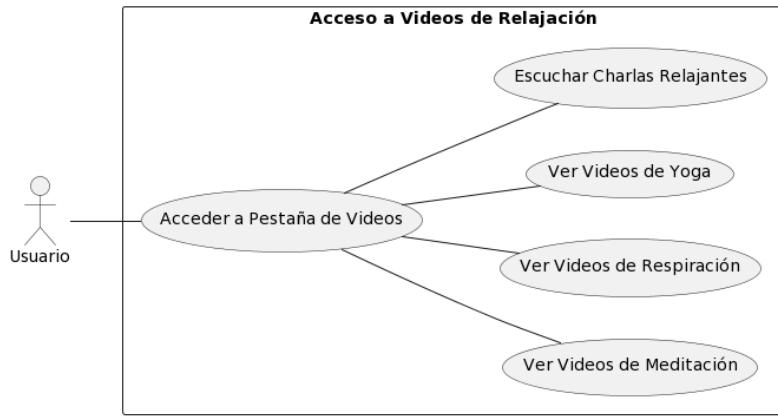


Figura 4.5: Caso de uso Ejercicios Diarios de Respiración

4.2.4.6. Historial y graficos



Figura 4.6: Caso de uso Historial y Gráficos

4.2.4.7. Privacidad y seguridad

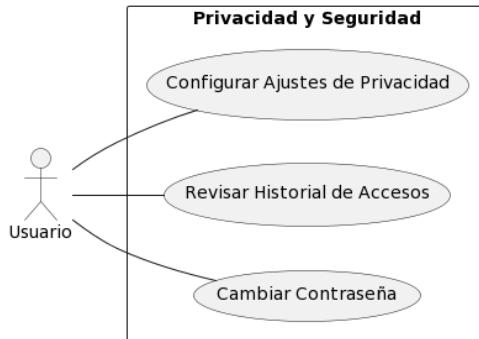


Figura 4.7: Caso de uso Privacidad y Seguridad

4.2.5. Tarjetas casos de uso

Inicio de sesión

Casos de uso	Inicio de sesión	01
Actores	Usuario, sistema	
Precondición	correo y nickname exclusivos	
Postcondición	Que se validen correctamente las credenciales	
Propósito		
	Registrar e iniciar sesión del usuario para poder acceder al sistema	

Cuadro 4.1: Caso de uso Inicio de sesión

Gestión de perfil

Casos de uso	Gestión de perfil	02
Actores	Usuario	
Precondición	Iniciar sesión	
Postcondición	Parámetros correctos	
Propósito		
	Editar datos del usuario	

Cuadro 4.2: Caso de uso Gestión de perfil

Registro diario de rutina y estado emocional

Casos de uso	Registro diario de rutina y estado emocional	03
Actores	Usuario	
Precondición	Iniciar sesión	
Postcondición	Valoración del estado mental del usuario	
Propósito		
	Registro diario de la salud mental del usuario	

Cuadro 4.3: Caso de uso Registro diario de rutina y estado emocional

Procesamiento de datos con IA

Casos de uso	Procesamiento de datos con IA}	04
Actores	Usuario, sistema	
Precondición	Iniciar sesión, ingresar valores de salud mental del usuario	
Postcondición	Valoración del estado mental del usuario	
Propósito		
	Valorar el estado mental del usuario y proporcionar un valor	

Cuadro 4.4: Caso de uso Procesamiento datos con IA

Ejercicios diarios de respiración

Casos de uso	Ejercicios diarios de respiración	05
Actores	Usuario	
Precondición	Iniciar sesión	
Postcondición		
Propósito		
Mejorar la salud mental del usuario con ejercicios de respiración y paz mental		

Cuadro 4.5: Caso de uso Ejercicios Diario de respiración

Historial y gráficos

Casos de uso	Historial y gráficos	06
Actores	Usuario, sistema	
Precondición	Iniciar sesión, introducir parámetros salud mental	
Postcondición	Diagrama con los resultados	
Propósito		
Ver de forma clara la situación actual en cuanto a salud mental se refiere del usuario		

Cuadro 4.6: Caso de uso Historial y gráficos

Privacidad y seguridad

Casos de uso	Privacidad y seguridad	07
Actores	Usuario	
Precondición	Iniciar sesión	
Postcondición	Gestión de datos del usuario	
Propósito		
Asegurar la buena gestión de los datos del usuario		

Cuadro 4.7: Caso de uso Privacidad y seguridad

Ver perfil

Casos de uso	Ver perfil	08
Actores	Usuario	
Precondición	Iniciar sesión	
Postcondición		
Propósito		
Visualización de los datos del usuario		

Cuadro 4.8: Caso de uso Ver Perfil

Editar Información Personal

Casos de uso	Editar información Personal	09
Actores	Usuario	
Precondición	Iniciar sesión	
Postcondición		
Propósito		
Modificar los datos del usuario		

Cuadro 4.9: Caso de uso Editar información Personal

Modificar contraseña

Casos de uso	Cambiar contraseña	10
Actores	Usuario	
Precondición	Iniciar sesión	
Postcondición		
Propósito		
Modificar la contraseña del usuario		

Cuadro 4.10: Caso de uso Modificar contraseña

Configurar Privacidad

Casos de uso	Configurar privacidad	11
Actores	Usuario	
Precondición	Iniciar sesión	
Postcondición		
Propósito		
Modificar el control de datos del usuario de forma explícita		

Cuadro 4.11: Caso de uso Configurar privacidad

Eliminar Cuenta

Casos de uso	Eliminar cuenta	12
Actores	Usuario	
Precondición	Iniciar sesión	
Postcondición	Eliminación de la cuenta	
Propósito		
Se eliminará la cuenta del usuario con sus datos		

Cuadro 4.12: Caso de uso Eliminar Cuenta

Registro Estado Anímico

Casos de uso	Registrar Estado Anímico	13
Actores	Usuario	
Precondición	Iniciar sesión	
Postcondición	Registrar estado del usuario	
Propósito		
	Se registrarán el estado anímico del usuario para su posterior procesamiento de datos	

Cuadro 4.13: Caso de uso Registrar Estado Anímico

Registrar Horas de Sueño

Casos de uso	Registrar Horas de Sueño	14
Actores	Usuario	
Precondición	Iniciar sesión	
Postcondición	Registrar estado del usuario	
Propósito		
	Se registrarán las horas de sueño del usuario para su posterior procesamiento de datos	

Cuadro 4.14: Caso de uso Registrar Horas de Sueño

Registrar Nivel de Estrés

Casos de uso	Registrar Nivel de Estrés	15
Actores	Usuario	
Precondición	Iniciar sesión	
Postcondición	Registrar estado del usuario	
Propósito		
	Se registrará el nivel de estrés del usuario para su posterior procesamiento de datos	

Cuadro 4.15: Caso de uso Registrar Nivel de estrés

Registrar Receptividad

Casos de uso	Registrar Receptividad	16
Actores	Usuario	
Precondición	Iniciar sesión	
Postcondición	Registrar estado del usuario	
Propósito		
	Se registrará la receptividad del usuario para su posterior procesamiento de datos	

Cuadro 4.16: Caso de uso Registrar Receptividad

Registrar Motivación

Casos de uso	Registrar Motivación	17
Actores	Usuario	
Precondición	Iniciar sesión	
Postcondición	Registrar estado del usuario	
Propósito		
	Se registrará la motivación del usuario para su posterior procesamiento de datos	

Cuadro 4.17: Caso de uso Registrar Motivación

Registrar Ganas de Vivir

Casos de uso	Registrar Ganas de Vivir	18
Actores	Usuario	
Precondición	Iniciar sesión	
Postcondición	Registrar estado del usuario	
Propósito		
	Se registrarán las ganas de vivir del usuario para su posterior procesamiento de datos	

Cuadro 4.18: Caso de uso Registrar Ganas de Vivir

Registrar Ganas de Comer

Casos de uso	Registrar Ganas de Comer	19
Actores	Usuario	
Precondición	Iniciar sesión	
Postcondición	Registrar estado del usuario	
Propósito		
	Se registrará el hambre del usuario para su posterior procesamiento de datos	

Cuadro 4.19: Caso de uso Registrar Ganas de Comer

Registrar Apatía

Casos de uso	Registrar Ganas de Apatía	20
Actores	Usuario	
Precondición	Iniciar sesión	
Postcondición	Registrar estado del usuario	
Propósito		
	Se registrará la apatía del usuario para su posterior procesamiento de datos	

Cuadro 4.20: Caso de uso Registrar Apatía

Registrar Actividad Deportiva

Casos de uso	Registrar Actividad Deportiva	21
Actores	Usuario	
Precondición	Iniciar sesión	
Postcondición	Registrar estado del usuario	
Propósito		
	Se registrará cuánto deporte ha realizado el usuario para su posterior procesamiento de datos	

Cuadro 4.21: Caso de uso Registrar Actividad Deportiva

Registrar Estado Emocional

Casos de uso	Registrar Estado Emocional	22
Actores	Usuario	
Precondición	Iniciar sesión	
Postcondición	Registrar estado del usuario	
Propósito		
	Se registrará el estado emocional del usuario para su posterior procesamiento de datos	

Cuadro 4.22: Caso de uso Registrar Estado Emocional

Ingresar Parámetros Salud Mental

Casos de uso	Ingresar Parámetros Salud Mental	23
Actores	Usuario	
Precondición	Iniciar sesión	
Postcondición	Almacenar los datos de salud del usuario	
Propósito		
	Se almacenarán los datos de salud del usuario en la base de datos correspondiente	

Cuadro 4.23: Caso de uso Ingresar Parámetros Salud Mental

Procesar datos con IA

Casos de uso	Procesar datos con IA	24
Actores	Usuario	
Precondición	Iniciar sesión, introducir parámetros salud mental.	
Postcondición	Obtención de resultados de salud mental	
Propósito		
	Se procederán a obtener resultados de la salud mental del usuario según el procesamiento de los parámetros previamente introducidos	

Cuadro 4.24: Caso de uso Procesar datos con IA

Acceder pestaña videos

Casos de uso	Acceder pestaña videos	25
Actores	Usuario	
Precondición	Iniciar sesión.	
Postcondición	Acceso a videos de mindfulness	
Propósito		
	Se desplegarán los videos recomendados por la aplicación para su posterior visualización	

Cuadro 4.25: Caso de uso Acceder pestaña videos

Escuchar Charlas relajantes

Casos de uso	Escuchar Charlas relajantes	26
Actores	Usuario	
Precondición	Iniciar sesión.	
Postcondición	Relajación del usuario	
Propósito		
	Se desplegarán los videos recomendados por la aplicación en los que se escucharán charlas para la mejora de la relajación del usuario	

Cuadro 4.26: Caso de uso Escuchar Charlas relajantes

Ver videos de Yoga

Casos de uso	Ver videos de Yoga	27
Actores	Usuario	
Precondición	Iniciar sesión.	
Postcondición	Relajación del usuario	
Propósito		
	Se desplegarán los videos recomendados por la aplicación en los que se verán clases de yoga para la mejora de la relajación del usuario	

Cuadro 4.27: Caso de uso Ver videos de Yoga

Ver videos de Meditación

Casos de uso	Ver videos de Meditación	28
Actores	Usuario	
Precondición	Iniciar sesión.	
Postcondición	Relajación del usuario	
Propósito		
	Se desplegarán los videos recomendados por la aplicación en los que se verán clases de relajación para la mejora del estado anímico del usuario	

Cuadro 4.28: Caso de uso Ver videos de Meditación

Ver Gráfico de Estado Mental

Casos de uso	Ver Gráfico de Estado Mental	29
Actores	Usuario	
Precondición	Iniciar sesión, introducir parámetros salud mental	
Postcondición	Relajación del usuario	
Propósito		
	Se visualizará un gráfico estimando de forma gráfica el estado de salud mental del usuario	

Cuadro 4.29: Caso de uso Ver Gráfico de Estado Mental

Revisar Historial de Accesos

Casos de uso	Revisar historial de Accesos	30
Actores	Usuario	
Precondición	Iniciar sesión, introducir parámetros salud mental	
Postcondición	Edición y visualización de datos del usuario	
Propósito		
	Se visualizarán los datos del usuario insertados para su posterior edición	
	u observación	

Cuadro 4.30: Caso de uso Revisar Historial de Accesos

4.3. Diseño

El diseño de esta aplicación surge de una necesidad tangible de ofrecer un enfoque holístico y personalizado hacia el bienestar mental y físico. En un mundo donde el estrés y la ansiedad son cada vez más prevalentes, una herramienta que pueda monitorear y ofrecer recomendaciones adaptadas a las necesidades individuales no es un lujo, sino una necesidad. La decisión de implementar la aplicación como un sistema híbrido de microservicios fue impulsada por la flexibilidad que esto proporciona, permitiendo escalabilidad y mantenimiento eficiente, así como la facilidad de integración de diferentes tecnologías y servicios de inteligencia artificial. Se optó por un diseño intuitivo y centrado en el usuario, con una interfaz amigable que incentiva la interacción constante y facilita la recopilación de datos de manera no intrusiva. La justificación del diseño está anclada en la premisa de que, al ser más conscientes de nuestros estados de salud y emocionales, podemos tomar decisiones más informadas para mejorar nuestra calidad de vida. Por lo tanto, el diseño de la aplicación no solo responde a un imperativo funcional, sino que también se alinea con una visión ética, buscando fomentar un mayor bienestar en la comunidad.

4.3.1. Diseño de la Base de Datos

Para Resilience hemos diseñado tres bases de datos (uno para cada micro-servicio):

- **Microservicio userData:** El cual contendrá todos los datos del usuario en una tabla 'user' con los atributos:

- **name:** Con el nombre del usuario
- **surname:** Con el apellido del usuario
- **nickname:** Con el mote del usuario y a la vez su ID
- **email:** Con el email del usuario
- **password:** Con la contraseña del usuario

- **Microservicio healthData:** Esta tendrá una colección llamada 'healthData' la cual tendrá los siguiente atributos:

- **id:** Identificador único del usuario.
- **school_year:** Año escolar actual.
- **age:** Edad del usuario.
- **gender:** Género del usuario.
- **bmi:** Índice de masa corporal, una medida de la salud corporal.
- **who_bmi:** Clasificación de la OMS para el BMI.
- **phq_score:** Puntuación en el cuestionario de salud del paciente, utilizado para detectar depresión.
- **depression_severity:** Severidad de la depresión.
- **depressiveness:** Nivel de tristeza o abatimiento.
- **suicidal:** Indica si hay pensamientos suicidas.
- **depression_diagnosis:** Indica si hay un diagnóstico de depresión.
- **depression_treatment:** Indica si el usuario está recibiendo tratamiento para la depresión.
- **gad_score:** Puntuación en la escala de ansiedad generalizada.
- **anxiety_severity:** Severidad de la ansiedad.
- **anxiousness:** Nivel de ansiedad.
- **anxiety_diagnosis:** Indica si hay un diagnóstico de ansiedad .
- **anxiety_treatment:** Indica si el usuario está recibiendo tratamiento para la ansiedad .
- **epworth_score:** Puntuación en la escala de somnolencia de Epworth , utilizada para evaluar la somnolencia diurna.
- **sleepiness:** Nivel de somnolencia .
- **timestamp:** Marca de tiempo de cuando se registraron estos datos

- **Microservicio IA:** Este recibirá los datos por parte de healthData y los procesará internamente para almacenarlos en Processing.IA, la cual estará formada por dos atributos:

- **id:** Identificador unico del usuario
- **need_treatment:** Variable booleana que determina si el usuario necesita o no tratamiento.

4.3.2. Diagrama entidad-relación

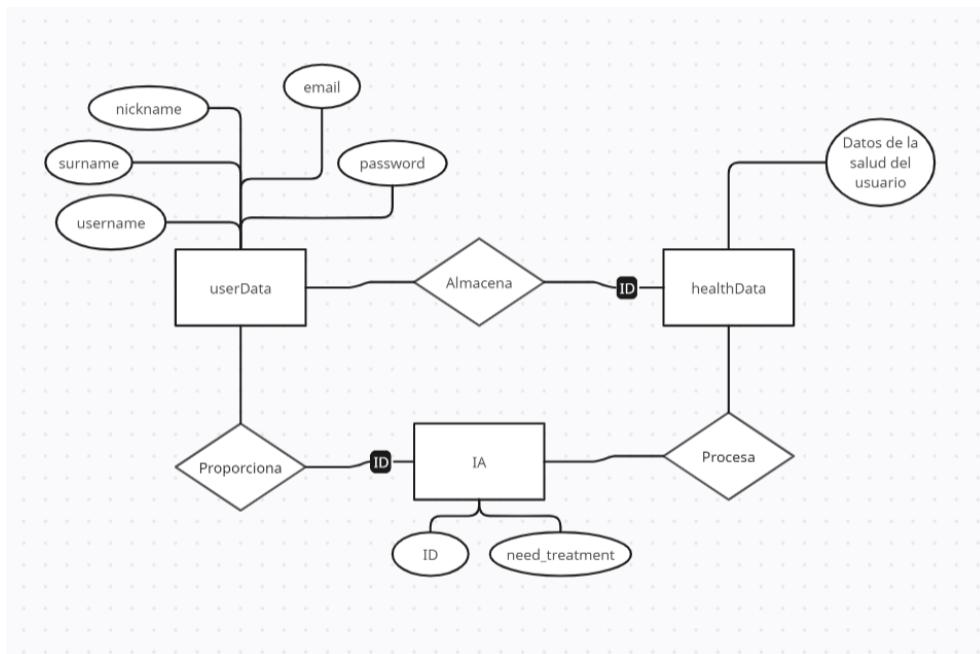


Figura 4.8: Diagrama Entidad-Relación. Fuente: propia

4.3.3. Elección y diseño de los microservicios de la aplicación

En el desarrollo de aplicaciones modernas, la arquitectura de microservicios juega un papel crucial al dividir una aplicación en conjuntos de servicios más pequeños, cada uno ejecutándose en su propio proceso y comunicándose con mecanismos ligeros, a menudo una API HTTP. Este enfoque no solo permite una mayor modularidad sino que también facilita el escalado y la mantención de las aplicaciones. A continuación, se detalla la función de cada microservicio en la aplicación propuesta:

Microservicio de Autenticación

El microservicio de autenticación se construye utilizando NodeJS, un entorno de ejecución para JavaScript del lado del servidor, que ofrece un rendimiento óptimo y una gestión eficiente de la asincronía a través de su modelo de eventos no bloqueante. La elección de MySQL como sistema de gestión de bases de datos se debe a su robustez y eficiencia en la manipulación de grandes volúmenes de datos.

Funcionalidades clave:

- **Inicio de Sesión:** Este servicio gestiona las credenciales de los usuarios y facilita la autenticación segura.
- **Gestión de Usuarios:** Se encarga de la creación de nuevos usuarios, actualización de datos de perfil y gestión de contraseñas.
- **Seguridad de Datos:** Implementa medidas de seguridad para proteger la información sensible de los usuarios.

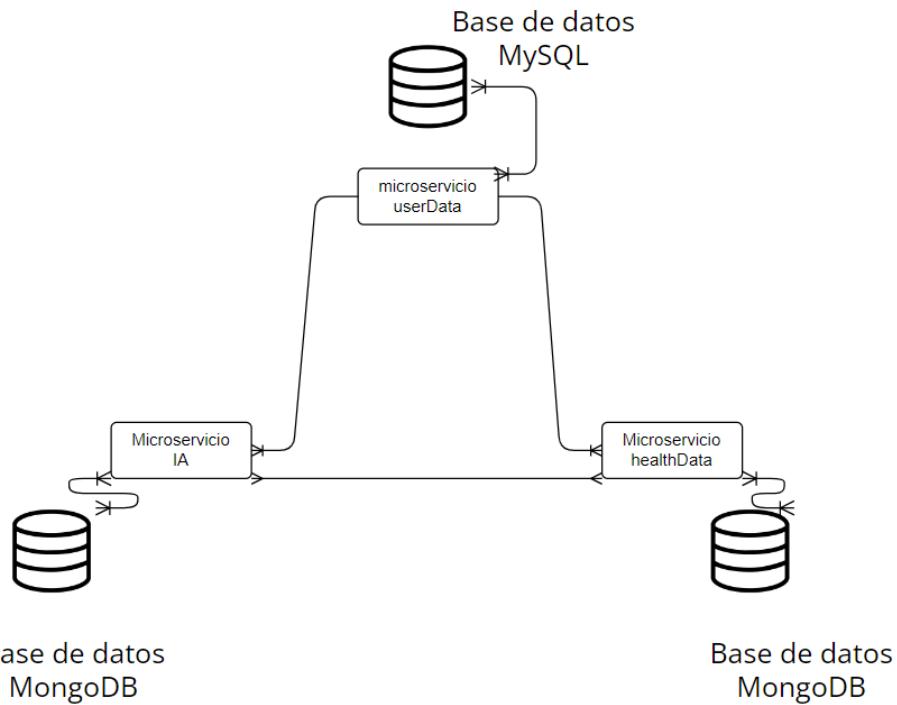


Figura 4.9: Esquema de los microservicios del sistema. Fuente: elaboración propia

Microservicio de Datos de Salud

FastAPI es un framework moderno y de alto rendimiento para la construcción de APIs con Python 3.7+, basado en estándares como OpenAPI y JSON Schema. Se escoge por su facilidad de uso, velocidad y compatibilidad con herramientas de aprendizaje automático. MongoDB, una base de datos NoSQL, es ideal para almacenar datos de salud de los usuarios debido a su escalabilidad y flexibilidad con estructuras de datos no estructuradas.

Funcionalidades clave:

- **Recolección de Datos:** Recopila datos de salud del usuario como el estado de ánimo, horas de sueño y estadísticas vitales.
- **API de Consultas:** Ofrece endpoints para que los usuarios o la IA puedan recuperar o enviar información de salud.
- **Integración con IA:** Facilita la interacción con el servicio de inteligencia artificial para el análisis de datos de salud.

Microservicio de Inteligencia Artificial

Flask es un microframework para Python que proporciona herramientas simples para desarrollar aplicaciones web rápidamente. Es ligero, fácil de desplegar y escalar, lo que lo hace adecuado para el microservicio de IA que procesa datos de salud. Este microservicio utiliza la misma base de datos MongoDB para mantener la consistencia y facilitar el acceso a los datos necesarios para el procesamiento.

Funcionalidades clave:

- **Procesamiento de Datos:** Analiza los datos de salud utilizando algoritmos de inteligencia artificial para identificar patrones o condiciones.
- **Recomendación de Tratamiento:** Basado en el análisis, determina si se deben ofrecer consejos de autocuidado o si se recomienda buscar ayuda profesional.
- **Interfaz de IA:** Proporciona una API para que otros servicios interactúen con las funcionalidades de IA.

Cada microservicio está diseñado para operar de manera independiente, lo que permite actualizaciones y mantenimiento sin interrumpir el funcionamiento de otros servicios. Además, esta modularidad facilita la posibilidad de escalar o modificar un servicio específico según sea necesario, sin afectar a toda la infraestructura de la aplicación.

Capítulo 5

Implementación

5.1. Marco tecnológico

5.1.1. Lenguajes de Programación y Frameworks

5.1.1.1. NodeJS¹

Implementar un microservicio con Node.js conlleva varios desafíos técnicos que pueden variar según la complejidad del proyecto y los requisitos específicos del sistema. Aquí están algunos de los problemas más desafiantes que los desarrolladores suelen enfrentar:

- **Gestión de la Asincronía:** Node.js se basa en un modelo de I/O no bloqueante y asincrónico, lo cual es excelente para la escalabilidad, pero puede resultar complicado al manejar el flujo de ejecución, especialmente en situaciones que requieren operaciones secuenciales o dependencias entre llamadas de funciones.
- **Manejo de Errores:** La naturaleza asincrónica de Node.js también introduce complejidad en el manejo de errores. Los errores deben manejarse correctamente para evitar la caída del microservicio. El manejo de promesas y el uso de `async/await` deben hacerse con cuidado para atrapar y propagar errores de manera efectiva.
- **Seguridad:** Asegurar la comunicación entre microservicios es crítico, especialmente si se exponen puntos finales a internet. La implementación de autenticación y autorización, como tokens JWT, OAuth, y la atención a las vulnerabilidades conocidas son desafíos constantes.
- **Pruebas:** Testear microservicios en Node.js implica garantizar que funcionen de manera aislada y en conjunto con otros servicios. Esto puede incluir pruebas unitarias, pruebas de integración, pruebas de carga y pruebas de contratos.

5.1.1.2. Herramientas y tecnologías basadas en Python

FastAPI²

¹<https://nodejs.org/en>

²<https://fastapi.tiangolo.com>

- **Rendimiento y Concurrency:** La gestión de la concurrencia para mantener un rendimiento óptimo en FastAPI, que se basa en Starlette y uvicorn, puede ser complicada, especialmente al profundizar en asincronía en Python.
- **Validación de Datos:** Configurar validaciones complejas con Pydantic en FastAPI puede ser tedioso, lo cual puede llevar a una lógica de validación intrincada.
- **Documentación:** Mantener la documentación actualizada alineada con los cambios en la API es un esfuerzo continuo en FastAPI, a pesar de su generación automática.
- **Errores de Dependencia:** Las incompatibilidades o actualizaciones de las dependencias en FastAPI pueden romper la aplicación si no se manejan cuidadosamente.
- **Escalabilidad:** La escalabilidad debe considerarse desde el principio, utilizando prácticas como Docker y Kubernetes en FastAPI para facilitar esto en producción.

Flask³

- **Estructura de la Aplicación:** Flask es un microframework que ofrece mucha libertad pero también requiere una estructuración cuidadosa del proyecto y selección de componentes adicionales por parte del desarrollador.
- **Gestión de Estado:** La gestión de estado en aplicaciones Flask es un desafío, especialmente cuando se requiere persistencia a través de solicitudes o para usuarios.
- **Asincronía:** Introducir asincronía en Flask puede no ser tan directo y a menudo requiere extensiones adicionales o el uso de herramientas como Celery para tareas en background.
- **Concurrencia:** Para la producción, Flask requiere un servidor WSGI como Gunicorn para manejar la concurrencia, lo cual implica configuración adicional y optimización.
- **Seguridad:** Implementar medidas de seguridad es crucial en Flask para proteger contra ataques web comunes como CSRF, XSS y inyecciones SQL.
- **Monitoreo y Logging:** Establecer un sistema de monitoreo y registro efectivo es vital en Flask y puede ser complejo en aplicaciones distribuidas.

5.2. Herramientas de Desarrollo

- **PyCharm⁴**

³<https://flask.palletsprojects.com/en/3.0.x/>

⁴<https://www.jetbrains.com/pycharm/>

- **Descripción:** IDE avanzado desarrollado por JetBrains específicamente para la programación en Python. Ofrece análisis de código, un depurador gráfico, un probador de unidad integrado, integración con sistemas de control de versiones, y soporta el desarrollo web con Django.
- **Uso en el proyecto:** Desarrollo de microservicios en Python, con herramientas integradas para mejorar la productividad y la calidad del código.

- **Visual Studio Code**⁵

- **Descripción:** Editor de código fuente desarrollado por Microsoft, ligero pero potente, que soporta múltiples lenguajes de programación y tiene una amplia gama de extensiones disponibles.
- **Uso en el proyecto:** Desarrollo de la interfaz de usuario en React, utilizando su extenso ecosistema de extensiones para facilitar el desarrollo de aplicaciones web modernas.

- **Git**⁶

- **Descripción:** Sistema de control de versiones distribuido para rastrear cambios en el código fuente durante el desarrollo de software.
- **Uso en el proyecto:** Manejo del versionado del código, colaboración entre desarrolladores y seguimiento de tareas.

- **GitHub**⁷

- **Descripción:** Plataforma de alojamiento de código que utiliza Git, facilita la colaboración y el control de versiones.
- **Uso en el proyecto:** Almacenamiento del repositorio del código, seguimiento de incidencias y documentación del proyecto.

5.3. Características de la implementación de la arquitectura

Contenedores Dockerizados: La arquitectura se basa en el uso de contenedores Docker para encapsular y distribuir las aplicaciones. Cada uno de los tres APIs (Node.js con MySQL, Python con FastAPI y MongoDB, Python con Flask y MongoDB) se encuentra dentro de un contenedor Docker independiente. Esto facilita la gestión, el despliegue y la portabilidad de las aplicaciones, ya que cada contenedor contiene todas las dependencias necesarias y se ejecuta de manera aislada del sistema host. [41][19][6]

Diversidad Tecnológica: La arquitectura destaca por la diversidad tecnológica utilizada en cada uno de los APIs. Esto permite aprovechar las fortalezas de diferentes tecnologías según las necesidades específicas de cada servicio.

⁵<https://code.visualstudio.com>

⁶<https://git-scm.com>

⁷<https://github.com>

La elección de Node.js, Python con FastAPI y Python con Flask muestra una adaptación precisa a los requisitos de cada API.[4][15][7]

Bases de Datos Variadas: En esta arquitectura, se utilizan dos tipos diferentes de bases de datos: MySQL y MongoDB. La elección de MySQL es adecuada para datos estructurados, como en el caso de la API Node.js, mientras que MongoDB se utiliza para manejar datos no estructurados y semiestructurados, como en las APIs desarrolladas con Python. Esta variedad de bases de datos demuestra una estrategia de elección de tecnología según las necesidades de almacenamiento de datos.[15][7][29]

FastAPI para Alta Velocidad: La elección de FastAPI en una de las APIs desarrolladas en Python es una característica diferenciadora notable. FastAPI es conocido por su alta velocidad de ejecución y su capacidad para crear APIs rápidas y eficientes. Esta elección puede tener un impacto significativo en el rendimiento de la aplicación y la capacidad de respuesta de la API correspondiente.[4][15][41]

Dockerización para Reproducibilidad: La decisión de utilizar contenedores Docker en todos los APIs proporciona un alto nivel de reproducibilidad. Cada API se ejecuta en su propio entorno aislado y contiene todas las dependencias necesarias. Esto asegura que las aplicaciones se comporten de la misma manera en diferentes entornos y simplifica la configuración y el despliegue. [4][15][7]

Facilita la Escalabilidad: La arquitectura basada en contenedores Docker permite que cada API sea escalable de manera independiente según las necesidades. Si una de las APIs experimenta un aumento en la carga de trabajo, puedes escalar ese contenedor específico sin afectar a los demás, lo que mejora la eficiencia de recursos y el rendimiento.[4][15][7]

Flexibilidad en el Desarrollo: La variedad de tecnologías y bases de datos utilizadas en tu arquitectura proporciona flexibilidad en el desarrollo. Cada equipo puede elegir las herramientas más adecuadas para su tarea específica, lo que optimiza el desarrollo y la productividad.[4][15][37][41]

Dificultades en la Gestión de Diversidad: A pesar de las ventajas, la gestión de una arquitectura diversificada puede introducir desafíos en términos de coordinación, monitoreo y mantenimiento. Es fundamental implementar prácticas sólidas de gestión y monitoreo para garantizar un funcionamiento eficiente y seguro del sistema en su conjunto.[4][15][7]

Esta arquitectura diversificada y basada en contenedores Docker ofrece un enfoque versátil y potente para el desarrollo de aplicaciones modernas y escalables.[4][37][41]

5.4. Implementación de los microservicios

Para la implementación de los microservicios se han tomado las siguientes decisiones, teniendo en cuenta que se pretendía mostrar como podemos tener un ecosistema de microservicios usando diferentes lenguajes de programación y tecnologías para su implementación de cada uno, y como en su conjunto pueden crear un software gracias a la orquestación de dichos microservicios.

Microservicio userData: Node.js con MySQL

▪ Node.js

- **Descripción:** Entorno de ejecución de JavaScript en el servidor, que permite construir aplicaciones de red escalables.[17]
- **Uso en el proyecto:** Se utilizó para crear microservicios responsables de la lógica de negocio y la gestión de la base de datos.
- **Ventajas:** Asíncrona nativa y manejo eficiente de I/O no bloqueante, ideal para servicios que requieren alta concurrencia.

▪ MySQL

- **Descripción:** Sistema de gestión de base de datos relacional (RDBMS) basado en SQL. [14]
- **Uso en el proyecto:** Almacenamiento de datos estructurados para aspectos del sistema que se beneficiaban de un esquema relacional, como el registro de usuarios.
- **Ventajas:** Integridad referencial, consultas optimizadas y un modelo de datos bien definido.

Microservicio HealthData: FastAPI con Python y MongoDB

▪ FastAPI

- **Descripción:** Framework de Python moderno y rápido para construir APIs con Python 3.7+, basado en estándares como OpenAPI y JSON Schema.[34]
- **Uso en el proyecto:** Implementación de endpoints de API para servicios que requieren alta velocidad de ejecución y facilidad de implementación.
- **Ventajas:** Velocidad de ejecución cercana a Node.js, validación automática de datos y generación de documentación interactiva.

▪ MongoDB

- **Descripción:** Base de datos NoSQL orientada a documentos, que ofrece una gran escalabilidad y flexibilidad.[5]
- **Uso en el proyecto:** Almacenamiento de datos no estructurados y semi-estructurados, como las entradas del diario de estados de ánimo y métricas de salud.
- **Ventajas:** Esquemas de datos flexibles que se adaptan a la naturaleza heterogénea de los datos en aplicaciones de IA y ML.

Microservicio IA: Flask con MongoDB

▪ Flask

- **Descripción:** Microframework de Python para desarrollar aplicaciones web.[30]
- **Uso en el proyecto:** Creación de microservicios simples y ligeros para manejar funciones específicas dentro de la aplicación.
- **Ventajas:** Simpleza y extensibilidad, con la capacidad de escalar según sea necesario.

5.4.1. Contenerización y Orquestación

▪ Docker

- **Descripción:** Plataforma de contenedorización que permite empaquetar aplicaciones en contenedores, asegurando la consistencia entre entornos de desarrollo y producción.[4]
- **Uso en el proyecto:** Cada microservicio fue dockerizado para asegurar un despliegue fácil y replicable, independientemente del entorno.
- **Ventajas:** Simplificación de la configuración, escalabilidad de los servicios y aislamiento de dependencias.

5.4.2. Frontend

▪ React

- **Descripción:** Biblioteca de JavaScript para construir interfaces de usuario de forma declarativa y eficiente. [35]
- **Uso en el proyecto:** Desarrollo del frontend de la aplicación, incluyendo la interactividad y la gestión del estado de la UI.
- **Ventajas:** Componentización de la UI, estado reactivo y un ecosistema rico en herramientas y componentes.

5.5. Evaluación

En la fase de desarrollo de este proyecto, hemos identificado aspectos clave en la funcionalidad de la aplicación que destacan su capacidad para ofrecer un seguimiento personalizado de la salud mental y física de los usuarios. A través del uso diligente de la aplicación, hemos podido observar que es posible identificar patrones que sugieren cuándo un usuario podría beneficiarse de intervenciones profesionales o cambios en su rutina diaria. En particular, la aplicación ha demostrado ser una herramienta valiosa en la detección temprana de indicadores que pueden señalar la necesidad de tratamiento o consejería adicional. Estos hallazgos son prometedores, ya que abren la puerta a futuras investigaciones sobre cómo las tecnologías digitales pueden servir como un complemento a los métodos tradicionales de diagnóstico y apoyo en el bienestar mental. Sin embargo, es importante reconocer que la aplicación no reemplaza la evaluación profesional de la salud; más bien, se presenta como un apoyo

preventivo y una primera línea de reconocimiento de posibles problemas. Estos resultados resaltan el potencial de la aplicación para fomentar una mayor autoconciencia en los usuarios respecto a su salud y bienestar general.

Capítulo 6

Modelo entrenado para la IA

Una parte fundamental de Resilience se basa en el procesamiento de datos para la correcta función de la IA. Para ello, me he basado en un kaggle verificado de internet ya que yo no poseo las habilidades como para determinar si una persona se considera estar en buen estado o no.

6.1. Fuente del código

El modelo entrenado se ha obtenido de Kaggle¹. Se trata de un notebook en python el cual detecta la depresión y la ansiedad del usuario compartido por João e Thayla y Vane Woll[26]. Este proyecto surge de la gravedad que consiste la ansiedad hoy en día y de los factores depresivos que este conlleva, ya que un buen estilo de vida a tiempo siempre será mejor que cualquier tratamiento.

6.2. Detalles del modelo entrenado

6.2.1. Variables para interpretar resultados

Si tenemos que destacar dos variables del modelo sobre todas en sin duda phq score y gad score.

▪ GAD Score

- Se basa en el cuestionario GAD-7[31], un autoinforme de siete ítems.
- Evalúa la presencia de síntomas de ansiedad generalizada en las últimas dos semanas.
- Las preguntas están relacionadas con la preocupación excesiva, el nerviosismo, la inquietud y el miedo.
- Los ítems se puntuán en una escala de 0 (nunca) a 3 (casi todos los días), con un puntaje total de 0-21.
- Puntajes de 5, 10 y 15 son los umbrales para ansiedad leve, moderada y severa, respectivamente.
- Ampliamente validado y utilizado en entornos clínicos y de investigación.

¹<https://www.kaggle.com/code/geovaniwoll/machine-learningproject>

■ PHQ Score

- Utiliza el cuestionario PHQ-9[25], que consta de nueve ítems basados en los criterios de diagnóstico del DSM-IV para la depresión.
- Mide la frecuencia de los síntomas de depresión experimentados en las últimas dos semanas.
- Incluye preguntas sobre el estado de ánimo, pérdida de interés, cambios en el sueño y el apetito, baja autoestima y dificultad en la concentración.
- Cada ítem se califica de 0 (nada en absoluto) a 3 (casi todos los días), con un puntaje total que varía de 0 a 27.
- Puntajes de 5, 10, 15 y 20 representan depresión leve, moderada, moderadamente severa y severa, respectivamente.
- Es una herramienta de cribado rápida y eficiente para la depresión y puede ayudar en la monitorización de la gravedad de los síntomas y en la respuesta al tratamiento.

6.2.2. Aplicación del modelo

- **Importación de Librerías:** Se importan las librerías necesarias para la manipulación, análisis y visualización de datos.
 - Pandas para la manipulación y análisis de datos.
 - NumPy para el manejo de arrays y matrices.
 - Matplotlib y Seaborn para la visualización de datos.
 - Scikit-learn para modelado estadístico y machine learning.
- **Carga del Conjunto de Datos:** Los datos se cargan desde un archivo CSV.
 - Utilización de Pandas para leer el archivo CSV y cargarlo en un DataFrame.
- **Exploración Inicial de Datos:** Visualización de las primeras entradas y estadísticas descriptivas.
 - Utilización de métodos de Pandas para obtener una vista previa de los datos.
 - Descripción estadística de las características principales del conjunto de datos.
- **Limpieza y Preparación de Datos:** Procesamiento de datos para eliminar valores faltantes y transformar variables categóricas.
 - Eliminación o imputación de valores faltantes.
 - Transformación de variables categóricas mediante codificación.
- **Análisis de Correlación:** Cálculo de la matriz de correlación para entender las relaciones entre variables.

- Generación de una matriz de correlación para identificar posibles correlaciones.
- **Visualización de Datos:** Creación de gráficos para la exploración visual de los datos.
 - Utilización de gráficos de barras, histogramas y diagramas de dispersión para visualizar la distribución y relaciones de los datos.
- **Creación de Nuevas Variables:** Combinación de variables existentes para formar nuevas métricas.
 - Ingeniería de características para mejorar el conjunto de datos y posiblemente la eficacia del modelo.
- **Análisis por Grupos y Visualización:** Análisis agrupado y visualización de la distribución de puntuaciones.
 - Estudio de cómo las diferentes categorías o grupos dentro de los datos afectan las variables de interés.
- **Construcción del Modelo de Regresión Logística:** División del conjunto de datos, entrenamiento del modelo y evaluación.
 - División del conjunto de datos en conjuntos de entrenamiento y prueba.
 - Entrenamiento de un modelo de regresión logística.
 - Evaluación del modelo utilizando métricas apropiadas.
- **Balanceo de Datos Mediante Oversampling:** Aplicación de técnicas de remuestreo para balancear el conjunto de datos.
 - Implementación de técnicas de oversampling para abordar el desequilibrio de clases.

6.2.3. Aplicación en Resilience

A la hora de implementar este notebook en nuestra IA hemos necesitado de tres parámetros principales: phq score, gad score y género

```
def predict_treatment_status(phq_score, gad_score, gender):
```

Figura 6.1: Cabecera de la función

Estos datos se procesaran con el modelo entrenado y nos devolverá un true o false en caso de que el usuario pudiera o no necesitar tratamiento psicológico.

Para tener una aplicación más encapsulada cara al usuario, la aplicación en vez de decir si el usuario necesita tratamiento o no, se le recomendará actividades de relajación de la propia aplicación, o simplemente se le felicitará por su estado mental.

6.3. Resultados

6.3.1. Evaluación del Modelo

Presentación de las métricas de precisión y el informe de clasificación del modelo.

- Análisis de las métricas de rendimiento del modelo para evaluar su precisión y recall.
- Discusión sobre la matriz de confusión y otros informes de clasificación.

6.3.2. Interpretación de los Datos

- **Variables Demográficas:** Incluyen edad, género, nivel de educación, estado civil, empleo, entre otros.
- **Puntuaciones de Escalas Psicométricas:** Datos obtenidos a través de cuestionarios estandarizados para medir la depresión y la ansiedad.
- **Datos Históricos y Clínicos:** Información sobre historial médico, uso de medicamentos, y tratamiento psicoterapéutico.

6.3.3. Realización del análisis

El análisis del modelo se realiza utilizando los siguientes métodos:

1. Análisis Descriptivo.
2. Visualización de Datos.
3. Correlaciones.
4. Modelado Estadístico.
5. Pruebas de Hipótesis.

6.3.4. Descripción de Datos

- Resumen de los datos básicos, incluyendo medias, medianas y rangos, para obtener una primera imagen clara de la información recogida.

6.3.4.1. Visualización de Datos

- Utilización de herramientas gráficas como histogramas, gráficos de barras y diagramas de dispersión para mostrar visualmente la distribución y las relaciones entre las variables.

6.3.4.2. Búsqueda de Patrones

- Empleo de métodos estadísticos para identificar relaciones consistentes entre variables, utilizando medidas como el coeficiente de correlación.

6.3.4.3. Modelado Predictivo

- Aplicación de modelos estadísticos como la regresión lineal, que nos permite entender y predecir el comportamiento de una variable basándonos en otra (por ejemplo, cómo el nivel de ansiedad puede predecir la búsqueda de ayuda).

6.3.4.4. Cambio Temporal de Emociones

- Análisis de series temporales para examinar cómo cambian las emociones a lo largo del tiempo y detectar posibles patrones o ciclos.

6.3.4.5. Análisis de Texto y Sentimientos

- Aplicación de técnicas de procesamiento de lenguaje natural para extraer y cuantificar los temas y emociones expresados en textos escritos, como las respuestas abiertas de una encuesta.

6.3.4.6. Implementación del Código

- Detalle del código escrito en lenguaje Python para realizar el análisis, incluyendo la limpieza de datos, transformaciones y la aplicación de modelos estadísticos y de machine learning.

6.3.5. Descubrimientos Clave

- Identificación de una mayor propensión de las mujeres a buscar ayuda comparado con los hombres, basado en el análisis de datos.
- Establecimiento de una relación directa entre la severidad de la tristeza o ansiedad y la iniciativa de buscar ayuda, utilizando la regresión lineal para cuantificar esta relación.

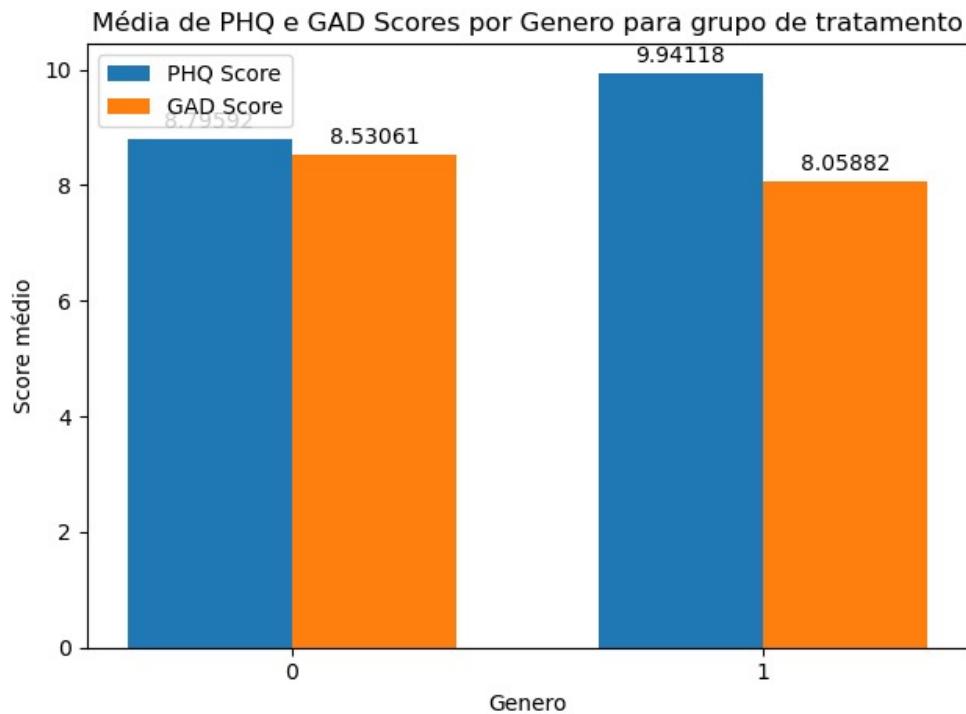


Figura 6.2: Gráfico comparativo de PHQ y GAD score por género

Capítulo 7

Desafíos y consideraciones éticas

7.1. Problemas de seguridad y privacidad.

Las aplicaciones que recopilan y almacenan datos sensibles sobre la salud y el bienestar de los usuarios enfrentan múltiples desafíos en términos de seguridad y privacidad. A continuación, se enumeran los problemas potenciales que pueden surgir dada la naturaleza de los datos[3]:

1. **Revelación de Datos Sensibles:** Dado que la aplicación registra datos como el estado de ánimo, patrones de sueño y otros valores de salud, existe el riesgo de que esta información caiga en manos equivocadas, lo que podría llevar a la discriminación, el estigma o incluso el chantaje.
2. **Vulnerabilidades en la Seguridad de Datos:** Sin medidas de seguridad adecuadas, los hackers pueden explotar vulnerabilidades en la aplicación o en el backend, llevando a filtraciones de datos.
3. **Uso no Autorizado:** Si los mecanismos de autenticación no son lo suficientemente robustos, personas no autorizadas podrían acceder a las cuentas de otros usuarios y ver o modificar sus datos.
4. **Compartir Datos con Terceros:** A veces, las aplicaciones pueden compartir datos con terceros, ya sea para análisis, publicidad o por otras razones. Si esto no se comunica claramente al usuario, o se hace sin su consentimiento, es una violación grave de la privacidad.
5. **Falta de Encriptación:** Los datos, tanto en tránsito como en reposo, deben estar encriptados. Si no se utiliza encriptación, la información podría ser interceptada o accedida por actores malintencionados.
6. **Almacenamiento Inseguro:** Si los datos se almacenan localmente en el dispositivo del usuario sin las medidas de seguridad adecuadas, podría ser vulnerable a ataques si el dispositivo se pierde o es robado.
7. **Cumplimiento Regulatorio:** Dependiendo de la jurisdicción, puede haber leyes y regulaciones específicas relacionadas con la recopilación y almacenamiento de datos de salud. No cumplir con estos reglamentos podría resultar en sanciones o litigios.

8. **Recolección Excesiva de Datos:** Recopilar más datos de los necesarios puede exponer a los usuarios a riesgos innecesarios. Es esencial adherirse al principio de minimización de datos.
9. **Consentimiento del Usuario:** No obtener un consentimiento claro y explícito del usuario para recopilar y procesar sus datos es un problema ético y, en muchos lugares, legal.
10. **Actualizaciones de Software:** Las vulnerabilidades de seguridad pueden surgir con el tiempo, y si la aplicación no recibe actualizaciones regulares, podría quedar expuesta a nuevos riesgos.
11. **Respaldo de Datos:** Si no se implementan procedimientos de respaldo adecuados, los datos podrían perderse en caso de fallos del sistema.
12. **Desaparición de Datos tras eliminación:** Incluso después de que los usuarios eliminen su cuenta o cierta información, los datos pueden persistir en copias de respaldo o logs. Asegurarse de que los datos se borren de manera efectiva es crucial.

Dada la naturaleza delicada de la información manejada por la aplicación, es imperativo adoptar un enfoque proactivo para abordar estos desafíos de seguridad y privacidad, garantizando así la confianza y seguridad de los usuarios.

7.2. Consideraciones sobre sesgos y justicia en la IA.

La Inteligencia Artificial (IA) tiene el potencial de transformar numerosas industrias, ofreciendo soluciones optimizadas y automatizadas a problemas complejos. Sin embargo, su implementación trae consigo desafíos éticos y prácticos, especialmente en relación con el sesgo y la justicia. A continuación, se describen las principales consideraciones: [15][29] [18]

1. **Datos de Entrenamiento Sesgados:** Si los datos utilizados para entrenar modelos de IA reflejan prejuicios o desigualdades existentes en la sociedad, estos modelos pueden perpetuar o incluso exacerbar estos sesgos.[41]
2. **Transparencia y Explicabilidad:** Los sistemas de IA, especialmente los basados en aprendizaje profundo, pueden actuar como cajas negras, dificultando la identificación y corrección de decisiones injustas o prejuiciadas.
3. **Automatización de Decisiones Críticas:** Cuando se confía en la IA para tomar decisiones que afectan a la vida de las personas, como la selección laboral, diagnósticos médicos o decisiones judiciales, los errores pueden tener graves consecuencias y perpetuar desigualdades.[41]
4. **Representación Diversa:** Los equipos que desarrollan sistemas de IA deben ser diversos e inclusivos para asegurar que las soluciones desarrolladas consideren diferentes perspectivas y minimicen los sesgos.

5. **Pruebas y Validaciones Continuas:** Es crucial realizar pruebas periódicas de los sistemas de IA en condiciones del mundo real para detectar y corregir sesgos y otras injusticias. [41]
6. **Responsabilidad y Rendición de Cuentas:** Debe existir un mecanismo claro para responsabilizar a las organizaciones y desarrolladores cuando los sistemas de IA causen daño o tomen decisiones injustas. [41]
7. **Normativas y Regulaciones:** Las políticas y regulaciones pueden guiar el desarrollo ético y justo de la IA, garantizando que se consideren y aborden los sesgos.
8. **Educación y Sensibilización:** Es esencial educar a los desarrolladores, usuarios y partes interesadas sobre los riesgos asociados con el sesgo en la IA y cómo mitigarlos.
9. **Participación Comunitaria:** Involucrar a las comunidades afectadas en el diseño y revisión de sistemas de IA puede ayudar a identificar y rectificar sesgos y desigualdades.
10. **Estándares Éticos:** Adoptar estándares éticos claros en el desarrollo y uso de la IA es fundamental para garantizar su justicia y equidad.

Estas consideraciones son fundamentales para garantizar que la IA no solo sea innovadora y eficiente, sino también ética y justa. A medida que la tecnología avanza, es esencial que la comunidad global colabore para abordar estos desafíos y garantizar que la IA beneficie a todos, sin discriminación ni perjuicio.

7.3. Implicaciones éticas y sociales.

La Inteligencia Artificial se ha integrado rápidamente en diversos aspectos de nuestra vida cotidiana, ofreciendo avances sin precedentes en comodidad, eficiencia y posibilidades. Sin embargo, como cualquier herramienta poderosa, trae consigo retos significativos que requieren una reflexión y acción cuidadosa. Las implicaciones éticas y sociales de la IA no son meras eventualidades a considerar en el futuro, sino problemas actuales y urgentes que afectan nuestra privacidad, autonomía, economía y, en última instancia, nuestra humanidad. Es crucial que la comunidad global, que abarca desde desarrolladores hasta legisladores y ciudadanos, aborde estas preocupaciones de manera proactiva. Debemos esforzarnos por implementar y regular la IA de tal manera que sus beneficios sean maximizados mientras se minimizan sus riesgos. Asimismo, es imperativo que la educación y la conciencia en torno a estas cuestiones se promuevan ampliamente, garantizando que el progreso tecnológico marche al unísono con nuestros valores éticos y principios morales. En última instancia, la IA, con todo su potencial, debe ser una extensión de nuestra humanidad y no una amenaza para ella. A continuación, se describen las principales implicaciones éticas y sociales relacionadas con la IA: [15][7][29] [41]

1. **Privacidad y Vigilancia:** La IA posibilita una vigilancia masiva, pudiendo comprometer la privacidad individual. Los sistemas de reconocimiento facial y análisis de comportamiento pueden ser utilizados de

maneras invasivas y, a menudo, sin el consentimiento informado de las personas.

2. **Autonomía y Dependencia:** Con la automatización de decisiones y procesos, existe el riesgo de que las personas pierdan autonomía, llegando a depender excesivamente de sistemas de IA para tomar decisiones cotidianas.
3. **Desigualdad Económica:** La IA puede exacerbar las desigualdades económicas si las habilidades y trabajos tradicionales se ven desplazados sin estrategias de reinserción y reeducación adecuadas.
4. **Manipulación y Autenticidad:** La tecnología de IA puede generar contenidos falsos o manipulados, como los "deepfakes", lo que puede tener implicaciones en áreas como la política, el periodismo o las relaciones personales.
5. **Responsabilidad y Culpabilidad:** Determinar la responsabilidad en caso de fallos o decisiones erróneas de un sistema de IA es complejo, especialmente en sistemas autónomos como vehículos autónomos.
6. **Transparencia y Derecho a la Explicación:** La naturaleza a menudo opaca de los sistemas de IA, en particular de las redes neuronales profundas, plantea preguntas sobre la transparencia y el derecho de los individuos a comprender las decisiones que afectan sus vidas.
7. **Humanidad y Relaciones Sociales:** La interacción creciente con asistentes virtuales y robots puede afectar la naturaleza de las relaciones humanas y cómo nos relacionamos con la tecnología y entre nosotros.
8. **Moralidad y Valores:** Imbuir a los sistemas de IA con moralidad y valores es un desafío, y existe el riesgo de que estos sistemas reflejen o perpetúen valores y prejuicios existentes en lugar de ideales éticos.
9. **Control y Autonomía de la Máquina:** A medida que los sistemas de IA adquieren mayor autonomía, surgen preocupaciones sobre el control humano sobre estas máquinas y su potencial para actuar de maneras no previstas o no deseadas.

La IA, al ser una herramienta poderosa, lleva consigo una gran responsabilidad. Es imperativo que los desarrolladores, legisladores, y la sociedad en general, aborden proactivamente estas preocupaciones para guiar el desarrollo y aplicación de la IA de manera que beneficie a la humanidad y preserve valores éticos y sociales.

Capítulo 8

Conclusión

8.1. Reflexión sobre los objetivos alcanzados.

La culminación de mi Trabajo de Fin de Grado representa no solo el final de un viaje académico, sino también el comienzo de una nueva etapa en mi desarrollo profesional y personal. A lo largo de este proyecto, me propuse metas ambiciosas, que por momentos parecían estar más allá de mi alcance. Sin embargo, a medida que avanzaba, me di cuenta de la capacidad intrínseca que poseo para superar obstáculos y adaptarme a los retos que se presentaban.

Es cierto que, al final del proceso, tuve que enfrentar el desafío del tiempo y apurar los últimos detalles. Aunque esto pudo haber añadido cierto estrés al proceso, también me brindó una invaluable lección sobre la importancia de la gestión del tiempo, la adaptabilidad y la determinación para cumplir con los objetivos propuestos.

Cada etapa, cada obstáculo superado y cada logro alcanzado en el desarrollo de la aplicación me ha permitido reafirmar mi pasión y motivación por el mundo de la tecnología y la inteligencia artificial. Además, este TFG se ha convertido en una representación tangible de mi crecimiento y evolución, mostrándome de lo que soy realmente capaz cuando me lo propongo.

Al mirar atrás y reflexionar sobre todo el camino recorrido, siento un profundo sentido de satisfacción y gratitud. Este proyecto no solo me ha permitido demostrar mis habilidades y conocimientos adquiridos, sino que también ha encendido una chispa en mi interior, impulsándome a seguir aprendiendo y explorando nuevas fronteras en el campo de la tecnología. Estoy convencido de que este TFG es solo el comienzo de muchas más aventuras y logros en mi carrera profesional, y estoy ansioso por descubrir lo que el futuro tiene reservado para mí.

8.2. Proyecciones futuras y recomendaciones.

A medida que este proyecto avanza, es evidente el potencial inmenso que presenta para el futuro. Visualizo una era donde la aplicación se transforme al incorporar algoritmos más sofisticados de aprendizaje automático que adapten las recomendaciones en tiempo real, basándose en las respuestas y comportamientos del usuario. La fusión con la tecnología wearable, como re-

lojes inteligentes, podría ofrecer una perspectiva aún más profunda sobre el bienestar del usuario, proporcionando datos precisos sobre patrones de sueño y otros indicadores vitales. Mientras la biblioteca de contenido, como videos de respiración y yoga, continúa expandiéndose, es imperativo seguir probando y ajustando las funcionalidades según las necesidades del usuario, manteniendo siempre la ética y privacidad al frente. La educación sobre cómo la aplicación utiliza la inteligencia artificial y el potencial de añadir características adicionales, como el seguimiento de la dieta o recordatorios de meditación, son pasos naturales hacia adelante. Esta evolución, combinada con un enfoque constante en el feedback del usuario y la integridad de los datos, asegurará que el proyecto no solo crezca, sino que prospere y marque una diferencia real en la vida de sus usuarios.

A medida que reflexiono sobre el desarrollo de este proyecto, es claro que hay horizontes amplios por explorar en trabajos futuros. Desde una perspectiva tecnológica, sería valioso integrar más algoritmos avanzados de Inteligencia Artificial, permitiendo así predicciones y recomendaciones más personalizadas. La incorporación de técnicas de procesamiento de lenguaje natural, por ejemplo, podría permitir a la aplicación entender y procesar mejor los registros escritos de los usuarios sobre su estado de ánimo y bienestar. Además, las futuras iteraciones podrían beneficiarse de una revisión más profunda en áreas que, por ser la primera vez que las abordaba, no pude analizar con el detenimiento que hubiera deseado. Mi experiencia en este proyecto ha sido inmensamente enriquecedora, dotándome de habilidades y perspicacia que me permitirán abordar desafíos similares con mayor eficacia en el futuro. La próxima vez que enfrente un desafío de esta magnitud, tendré la confianza de saber que he superado obstáculos similares anteriormente, permitiéndome abordar el proyecto con mayor soltura y decisión. A futuros desarrolladores, les recomendaría no subestimar la importancia de las pruebas y el feedback continuo de los usuarios, ya que son vitales para asegurar que la aplicación no solo funcione técnicamente, sino que realmente responda a las necesidades y expectativas del usuario final.

Bibliografía

- [1] ¿Qué es la arquitectura orientada a los servicios? es. URL: <https://www.redhat.com/es/topics/cloud-native-apps/what-is-service-oriented-architecture> (visitado 04-11-2023).
- [2] Sunday Adeola Ajagbe et al. “Internet of medical things (IoMT): applications, challenges, and prospects in a data-driven technology”. En: *Intelligent Healthcare: Infrastructure, Algorithms and Management* (2022), págs. 299-319.
- [3] ARANOVA. *Seguridad y protección de datos de las Apps de Salud o mHealth*. es-ES. URL: <https://aranova.es/blog/seguridad-y-proteccion-de-datos-de-las-apps-de-salud-o-mhealth> (visitado 04-11-2023).
- [4] baeldung. *Dockerizing a Spring Boot Application — Baeldung*. en-US. Ago. de 2016. URL: <https://www.baeldung.com/dockerizing-spring-boot-application> (visitado 27-07-2023).
- [5] K. Banker. *MongoDB in Action (3rd Edition)*. Manning Publications, 2021. ISBN: 978-1617291609.
- [6] Priya Bansal y Abdelkader Ouda. “Study on Integration of FastAPI and Machine Learning for Continuous Authentication of Behavioral Biometrics”. En: *2022 International Symposium on Networks, Computers and Communications (ISNCC)*. Jul. de 2022, págs. 1-6. DOI: 10.1109/ISNCC55209.2022.9851790. URL: <https://ieeexplore.ieee.org/abstract/document/9851790> (visitado 04-11-2023).
- [7] Antonio Bucchiarone et al. *Microservices: Science and Engineering*. en. Google-Books-ID: WqjDDwAAQBAJ. Springer Nature, dic. de 2019. ISBN: 978-3-030-31646-4.
- [8] Bruce G Buchanan. “A (very) brief history of artificial intelligence”. En: *Ai Magazine* 26.4 (2005), págs. 53-53.
- [9] Clue: Calendario de ovulación y periodo para iOS, Android y watchOS. es. URL: <https://helloclue.com/es> (visitado 07-11-2023).
- [10] Daylio - Journal, Diary and Mood Tracker. es. URL: <https://daylio.net/> (visitado 05-11-2023).
- [11] Omar Al-Debagy y Peter Martinek. “A comparative review of microservices and monolithic architectures”. En: *2018 IEEE 18th International Symposium on Computational Intelligence and Informatics (CINTI)*. IEEE. 2018, págs. 000149-000154.

- [12] Vishva Desai, Yash Koladia y Suvarna Pansambal. “Microservices: architecture and technologies”. En: *Int. J. Res. Appl. Sci. Eng. Technol* 8.10 (2020), págs. 679-686.
- [13] Nicola Dragoni et al. “Microservices: yesterday, today, and tomorrow”. En: *Present and ulterior software engineering* (2017), págs. 195-216.
- [14] P. DuBois. *MySQL (6th Edition)*. O'Reilly Media, 2022. ISBN: 978-1492080491.
- [15] *Ética de la inteligencia artificial — Articles*. URL: <https://www.unesco.org/es/artificial-intelligence/recommendation-ethics> (visitado 04-11-2023).
- [16] Keith D. Foote. *A Brief History of Microservices*. es. Abr. de 2021. URL: <https://www.dataversity.net/a-brief-history-of-microservices/> (visitado 03-11-2023).
- [17] Node.js Foundation. *Node.js Reference Documentation*. 2019. URL: <https://nodejs.org/docs/latest/api/>.
- [18] GooApps®. *Apps de salud, Privacidad y Protección de datos*. es. Mayo de 2022. URL: <https://gooapps.es/2022/05/24/apps-medicas-privacidad-y-proteccion-de-datos/> (visitado 04-11-2023).
- [19] Miguel Grinberg. *Flask Web Development: Developing Web Applications with Python*. en. Google-Books-ID: cVIPDwAAQBAJ. .O'Reilly Media, Inc., mar. de 2018. ISBN: 978-1-4919-9169-5.
- [20] Jeff Heaton. “Ian Goodfellow, Yoshua Bengio, and Aaron Courville: Deep learning”. en. En: *Genetic Programming and Evolvable Machines* 19.1 (jun. de 2018), págs. 305-307. ISSN: 1573-7632. DOI: 10.1007/s10710-017-9314-z. URL: <https://doi.org/10.1007/s10710-017-9314-z> (visitado 04-11-2023).
- [21] Robert Heinrich et al. “Performance engineering for microservices: research challenges and directions”. En: *Proceedings of the 8th ACM/SPEC on international conference on performance engineering companion*. 2017, págs. 223-226.
- [22] María Alejandra Hinojosa. “Diagrama de gantt”. En: *Producción, procesos y operaciones* 48 (2003).
- [23] <http://59e3fd97d0784951aaf980d5dbb23a79.pages.ubembed.com/a63a8306-6775-4b20-ad89-947344a832cc/>. URL: <https://59e3fd97d0784951aaf980d5dbb23a79.pages.ubembed.com/a63a8306-6775-4b20-ad89-947344a832cc/a.html?closedAt=0> (visitado 06-11-2023).
- [24] *Keras: Deep Learning for humans*. URL: <https://keras.io/> (visitado 06-11-2023).
- [25] Kurt Kroenke, Robert L Spitzer y Janet BW Williams. “The PHQ-9: validity of a brief depression severity measure”. En: *Journal of general internal medicine* 16.9 (2001), págs. 606-613.
- [26] Carlos Martínez. *Combate la depresión y la ansiedad con Moodpath*. es-ES. Section: Análisis y reviews. Dic. de 2018. URL: <https://blog.uptodown.com/moodpath-ansiedad-depresion-android/> (visitado 05-11-2023).

- [27] *Monolítico frente a microservicios: diferencia entre arquitecturas de desarrollo de software - AWS.* es-ES. URL: <https://aws.amazon.com/es/compare/the-difference-between-monolithic-and-microservices-architecture/> (visitado 04-11-2023).
- [28] *My Possible Self: The Mental Health App.* es. URL: <https://www.mypossibleself.com/> (visitado 05-11-2023).
- [29] Sam Newman. *Building Microservices.* en. Google-Books-ID: ZvM5EAAAQBAJ. .O'Reilly Media, Inc.", jul. de 2021. ISBN: 978-1-4920-3397-4.
- [30] Pallets Projects. *Flask Documentation.* 2022. URL: <https://flask.palletsprojects.com/en/2.0.x/>.
- [31] Faye Plummer et al. "Screening for anxiety disorders with the GAD-7 and GAD-2: a systematic review and diagnostic metaanalysis". En: *General hospital psychiatry* 39 (2016), págs. 24-31.
- [32] Europa Press. *McKinsey.- El mercado del bienestar goza de buena salud y está en auge, según McKinsey.* (SCHEME=ISO639) es. Publisher: Europa Press. Sep. de 2021. URL: <https://www.europapress.es/consulting-news/noticia-mckinsey-mercado-bienestar-goza-buena-salud-auge-mckinsey-20210903110800.html> (visitado 07-11-2023).
- [33] *PyTorch.* es. URL: <https://www.pytorch.org> (visitado 04-11-2023).
- [34] Sebastián Ramírez. *FastAPI Documentation.* 2020. URL: <https://fastapi.tiangolo.com/>.
- [35] *React.* en. URL: <https://react.dev/> (visitado 29-10-2023).
- [36] *scikit-learn: machine learning in Python — scikit-learn 1.3.2 documentation.* URL: <https://scikit-learn.org/stable/> (visitado 06-11-2023).
- [37] *Spring Initializr.* URL: <https://start.spring.io/> (visitado 27-07-2023).
- [38] Davide Taibi, Valentina Lenarduzzi y Claus Pahl. "Architectural patterns for microservices: a systematic mapping study". En: *CLOSER 2018: Proceedings of the 8th International Conference on Cloud Computing and Services Science; Funchal, Madeira, Portugal, 19-21 March 2018.* SciTePress. 2018.
- [39] *TensorFlow.* URL: <https://www.tensorflow.org/?hl=es-419> (visitado 04-11-2023).
- [40] *The History of Microservices: From Thought Experiment to Industry Standard.* es. URL: <https://www.linkedin.com/pulse/history-microservices-from-thought-experiment-industry-muaath-bin-ali> (visitado 06-11-2023).
- [41] *Why Docker — Docker.* en-US. Nov. de 2021. URL: <https://www.docker.com/why-docker/> (visitado 29-10-2023).

Apéndice A

Anexo I

A.1. Manual de instalación

Para este proyecto tendremos que tener en cuenta la tecnología del back-end y el front-end. Para que todo funcione necesitaremos principalmente dos programas: Docker y React.

A.1.1. Back-end

Para desplegar el back-end no necesitaremos descargar ninguna dependencia adicional más que docker, ya que el propio Docker-compose se encargará de establecer la configuración del mismo para su despliegue.

Con el comando docker compose up -d se levantará tanto el contenedor como su base de datos correspondiente.

A.1.2. Front-end

Para desplegar el front-end no necesitaremos descargar tampoco ninguna dependencia puesto que, al igual que en el back-end, se configurará automáticamente.

Para desplegar el front-end tendremos que ponernos en la carpeta raíz del programa y con el comando npm start arrancará React.

Apéndice B

Anexo II

B.1. Manual de uso

En este apartado procederemos a mostrar la aplicación explicando la interfaz y la funcionalidad de la misma.

B.1.1. Inicio de sesión y registro

Aquí podemos observar un inicio de sesión el cual garantiza la obtención de datos del usuario y la seguridad de los datos del mismo.

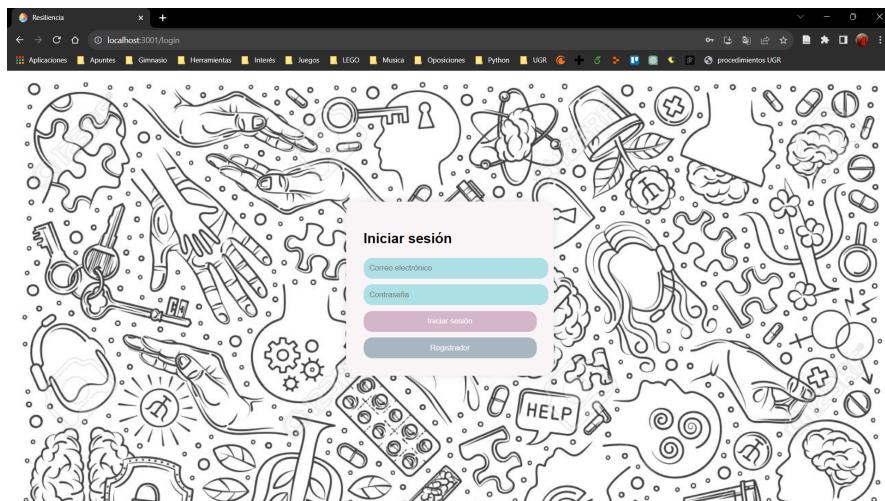


Figura B.1: Imagen de Resilience Inicio de sesión. Fuente: propia

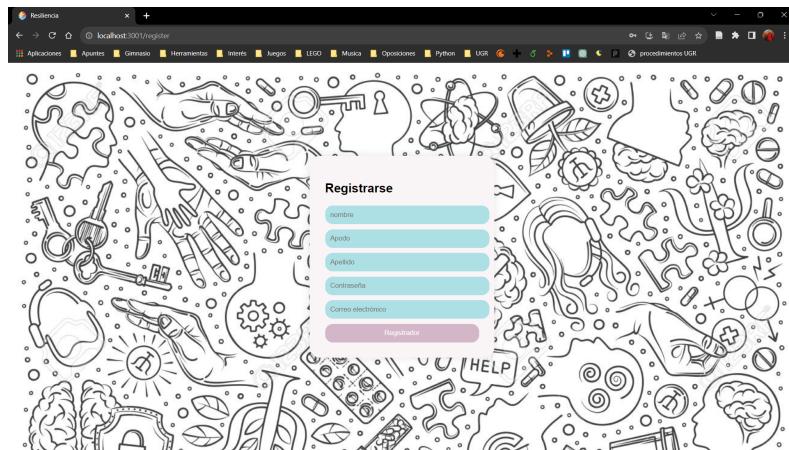


Figura B.2: Imagen de Resilience Registro de Usuario. Fuente: propia

B.1.2. Home

En esta imagen podemos ver la pantalla principal de la aplicación, la cual nos muestra un diagrama de nuestro estado mental actual y un breve feedback de nuestra situación mental actual. Como podemos observar tenemos un ícono con una interrogación en la cual nos explicará para que podamos interpretar fácilmente nuestro diagrama.

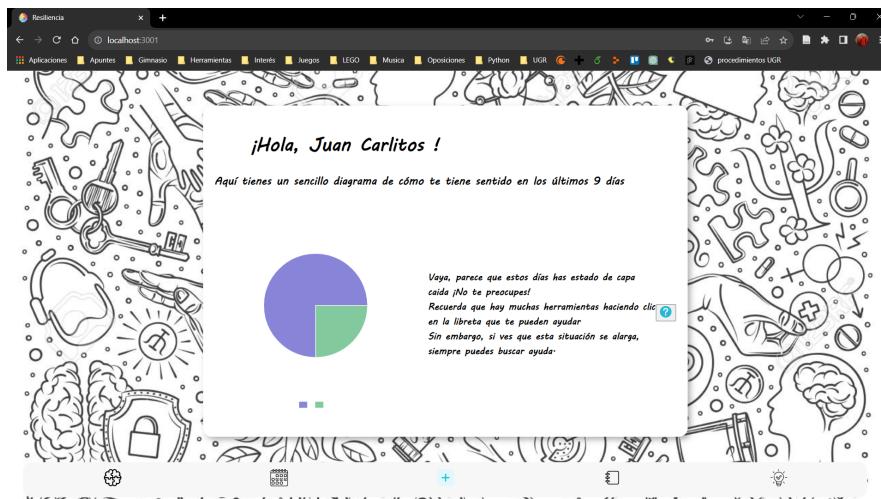


Figura B.3: Imagen de Resilience Menú de inicio. Fuente: propia

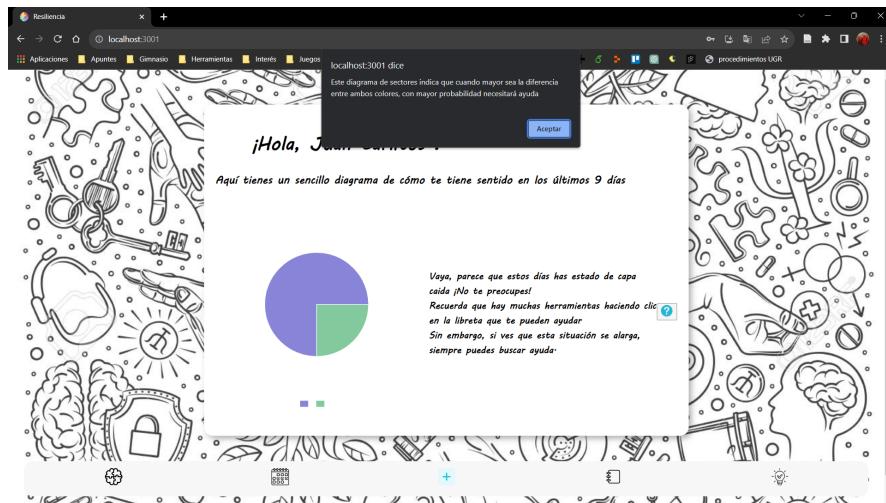


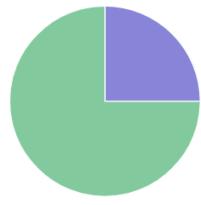
Figura B.4: Imagen de Resilience Menú de inicio. Fuente: propia

B.1.2.1. Ejemplo de feedback

En este caso, tenemos que el usuario está bien psicológicamente y nos devuelve un mensaje de ánimo.

jHola, Juan Carlitos !

Aquí tienes un sencillo diagrama de cómo te tiene sentido en los últimos 9 días



*jGuau! Estos días has estado genial Enhorabuena, sigue así
Igualmente, recuerda que si lo necesitas siempre tendrás recursos haciendo clic en la libreta.*

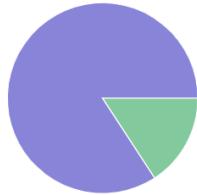


Figura B.5: Imagen de Resilience Usuario sin recomendar tratamiento. Fuente: propia

Sin embargo, en el caso que el usuario lleve una mala racha tendrá una respuesta por parte de la aplicación como esta, en la cual le recomienda la realización de ejercicios de relajación y que sea consciente de su situación.

¡Hola, Juan Carlitos !

Aquí tienes un sencillo diagrama de cómo te tiene sentido en los últimos 9 días



Vaya, parece que estos días has estado de capa caída ¡No te preocupes!

Recuerda que hay muchas herramientas haciendo clic en la libreta que te pueden ayudar

Sin embargo, si ves que esta situación se alarga, siempre puedes buscar ayuda.



Figura B.6: Imagen de Resilience Usuario que se le recomienda tratamiento.
Fuente: propia

B.1.3. Calendario

Para el apartado del calendario, dispondremos del usuario del que clicaremos en el día que queramos almacenar nuestro estado emocional, por si algún día se nos pasó o queramos revisarlo.

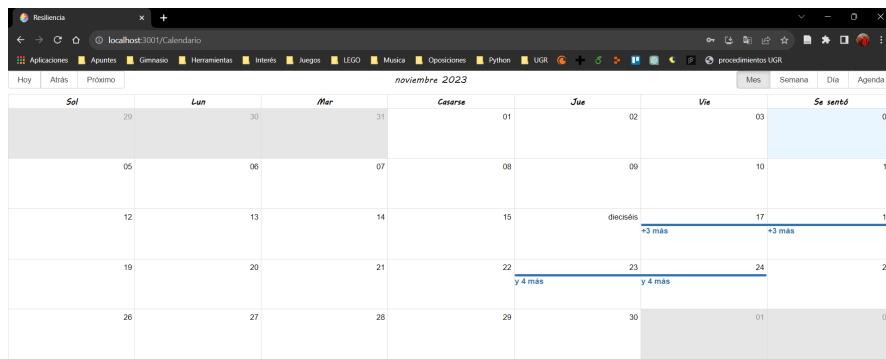


Figura B.7: Imagen de Resilience Calendario. Fuente: propia

Aquí podemos ver la interfaz en la que registraremos nuestro estado de ánimo, veremos el registro de ese día con opción a eliminar alguna respuesta y opción a moverse entre ellas. Para registrar tu estado emocional tendremos que clicar en el ícono con el que más identificado nos sintamos.

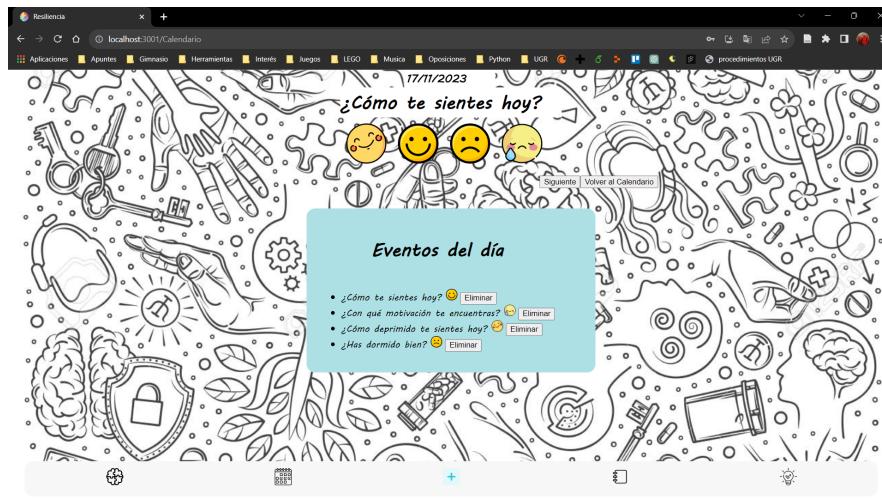


Figura B.8: Imagen de Resilience Preguntas de salud mental. Fuente: propia

B.1.4. Add

Una pestaña minimalista donde guardaremos cómo nos habremos sentido ese día para nosotros mismos.

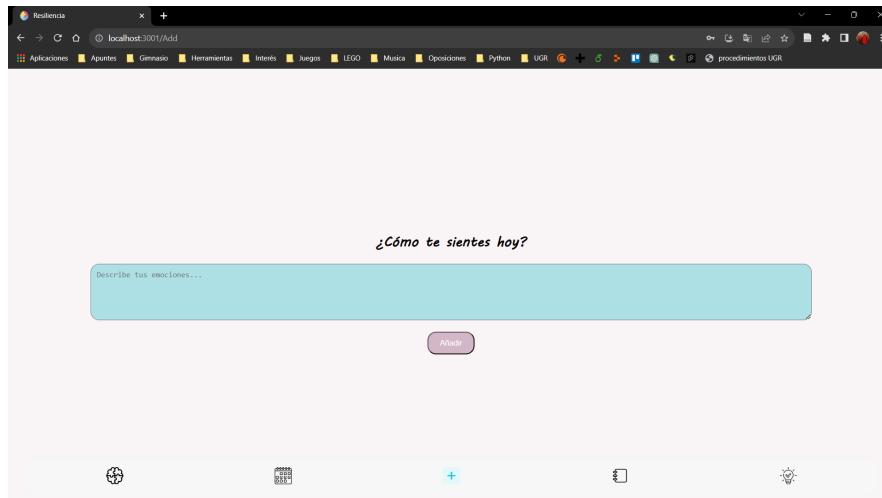


Figura B.9: Imagen de Resilience Inserción de estado del usuario. Fuente: propia

B.1.5. Análisis

En este apartado nos encontramos con la pestaña más técnica donde el usuario podrá realizar test psicológicos para saber su estado de ánimo. Estos test psicológicos son formatos ya preestablecidos en psicología compuestos por phq score 9, el cual es un test de 9 preguntas, y gad score, un test de 7 preguntas.

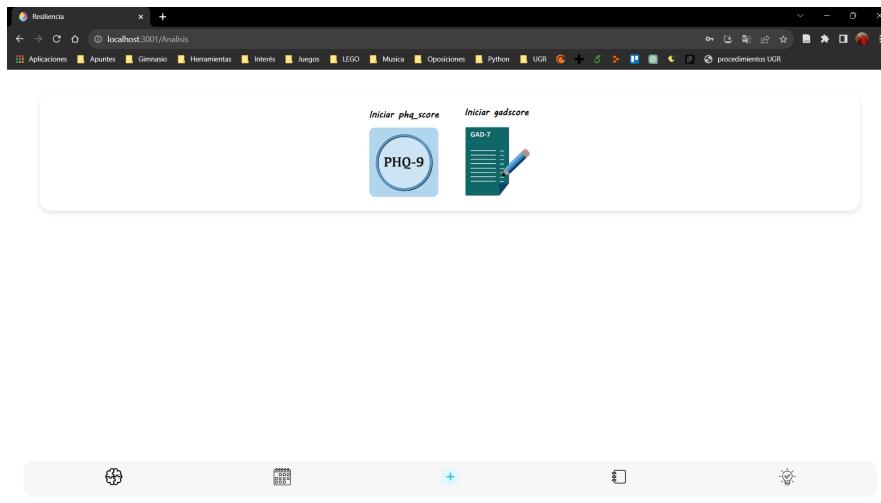


Figura B.10: Imagen de Resilience Tests del usuario. Fuente: propia

Aquí podemos ver el contenido de dicho test.

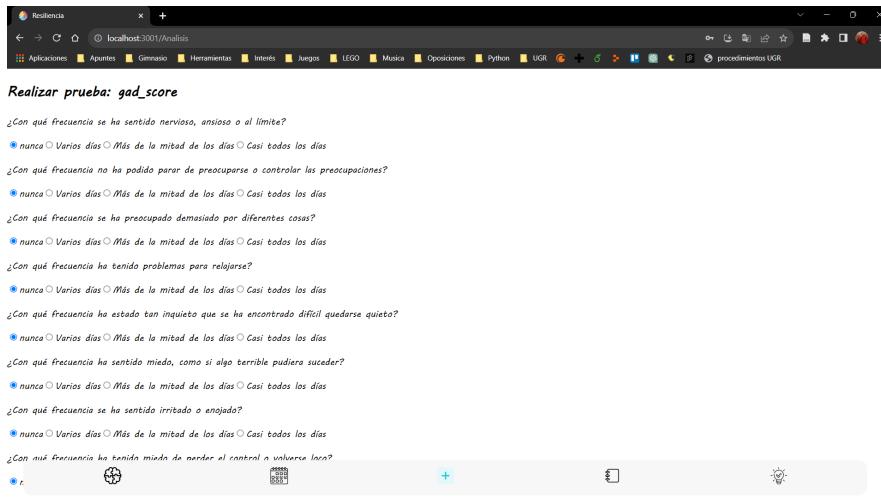


Figura B.11: Imagen de Resilience Test Gad Score. Fuente: propia

B.1.6. Respiración y desconexión

Y para finalizar tenemos aquí la zona de respiración y desconexión donde el usuario podrá reproducir cualquier vídeo para realizar ejercicios de relajación, yoga o cualquiera que necesite.

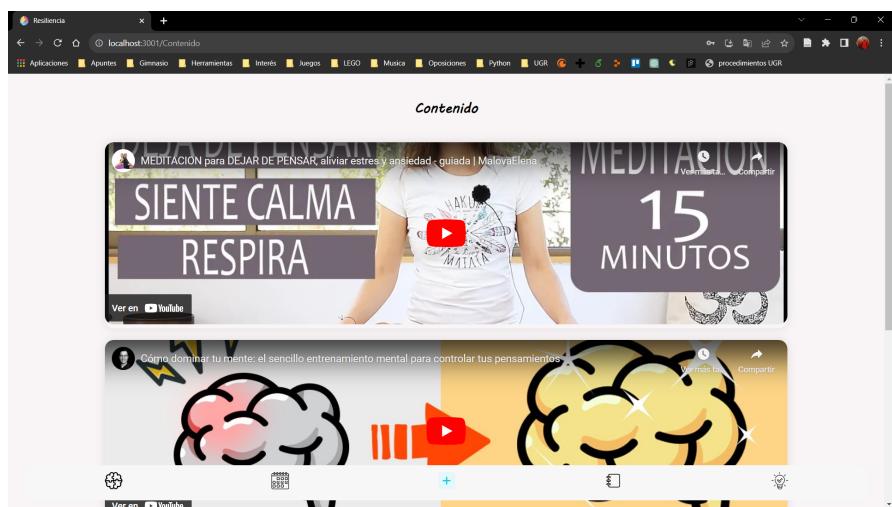


Figura B.12: Imagen de Resilience Videos Recomendados por la Aplicación.
Fuente: propia