Gestión y Análisis de datos en Pacientes hipertensos y/o diabéticos para la mitigación de riesgos y/o complicaciones mediante modelos de inteligencia artificial.

Pablo Eduardo Serrano Rincón

Trabajo de Grado para Optar al Título de Ingeniero de Sistemas

Director

Gabriel Rodrigo Pedraza Ferreira

PhD Ciencias de la Computación

Universidad Industrial de Santander

Facultad de Ingenierías Físico-Mecánicas

Escuela de Ingeniería de Sistemas

Ingeniería de Sistemas

Bucaramanga

2022

Agradecimientos

A mi padre Pablo Eduardo por siempre confiar en mí y motivarme todos los días para nunca rendirme y seguir luchando por alcanzar mis sueños.

A mi madre Nohora que siempre ha estado pendiente de mí, brindándome su sabiduría y y proporcióname una tranquilidad que solo ella puede lograr.

A Angie García por su apoyo incondicional, sus consejos y todo el tiempo dedicado acompañando en mi día a día y ayudándome a ser la mejor versión de mí mismo.

A mi perrito paco quien trasnochó conmigo y nunca me abandonó.

A mis amigos con quienes pude compartir durante toda la etapa universitaria, me apoyaron durante todo este proceso y hoy soy una mejor persona gracias a ellos.

A la empresa A&A Soluciones – TIC y su CEO Juan Carlos Abaunza por abrirme las puertas en esta organización y ayudarme a completar mi formación profesional.

A mi director de proyecto por aceptar dirigir este proyecto y brindarme su conocimiento para conseguir completar esta etapa de la mejor manera.

A la Universidad Industrial de Santander por abrirme sus puertas y permitirme formarme personal y profesionalmente.

Tabla de Contenido

Pag

Lista de Tablas

Pág.

Lista de Figuras

Pág.

Resumen

Título: Gestión y Análisis de datos en Pacientes hipertensos y/o diabéticos para la mitigación de riesgos y/o complicaciones mediante modelos de inteligencia artificial [*]

Autor: Pablo Eduardo Serrano Rincón [**]

Palabras Clave: Machine Learning, Deep Learning, Microservicios, Bases de datos, Ciencia de Datos.

Descripción: El control de la diabetes y la hipertensión es una tarea rigurosa, la cual requiere tener la mejor atención médica en el menor tiempo posible. Una rápida atención puede ayudar a evitar la progresión de la enfermedad. Hoy en día, la tecnología nos brinda dispositivos con la capacidad de generar miles de datos, los cuales pueden usarse para construir conocimiento en la detección temprana de riesgos.

El proyecto presente propone el diseño de un motor de análisis de datos que apoye el tratamiento de estos pacientes brindando un entorno inteligente. El motor tendrá la funcionalidad de gestionar los datos de dispositivos Wearables en una base de datos, los cuales serán procesados y analizados por modelos de inteligencia artificial para identificar y predecir el desarrollo de factores de riesgo que puedan incidir en la aparición de complicaciones. A partir de estos datos, se diseñarán informes que serán utilizados por los servicios REST Whatoko para visualización en un dashboard, de tal manera que puedan ser interpretados tanto por el personal Médico como el paciente.

---

[*] Trabajo de Grado

[**] Facultad de Ingenierías Físico-mecánicas. Escuela de Ingeniería de Sistemas e Informática. Ingeniería de Sistemas. Director: Gabriel Rodrigo Pedraza Ferreira. Doctorado en ciencias de la computación. Tutor: Francisco Javier González Silva. Ingeniería de Sistemas.

Abstract

Title: Management and Analysis of data in hypertensive and/or diabetic patients for the mitigation of risks and/or complications though artificial intelligence models [*]

Author(s): Pablo Eduardo Serrano Rincón [**]

Key Words: Machine Learning, Deep Learning, Data bases, Microservices, Data Science

Description: The control of diabetes and hypertension is a rigorous task, which requires having the best medical care in the shortest possible time. Prompt care can help prevent disease progression. Nowadays, technology gives us devices with the capacity to generate thousands of data, which can be used to build knowledge in the early detection of risks.

The present project proposes the design of a data analysis engine that supports the treatment of these patients by providing an intelligent environment. The engine will have the functionality to manage data from Wearable devices in a database, which will be processed and analyzed by artificial intelligence models to identify and predict the development of risk factors that may affect the appearance of complications. Based on these data, reports will be designed that will be used by the Whatoko REST services for display in a dashboard, so that they can be interpreted by both the medical staff and the patient.

---

[*] Trabajo de Grado

[**] Facultad de Ingenierías Físico-mecánicas. Escuela de Ingeniería de Sistemas e Informática. Ingeniería de Sistemas. Director: Gabriel Rodrigo Pedraza Ferreira. Doctorado en ciencias de la computación. Tutor: Francisco Javier González Silva. Ingeniería de Sistemas.

Introducción

El control de la hipertensión y la diabetes representa un desafío para los médicos y especialistas a la hora de tomar decisiones preventivas. Según el Ministerio de Salud (Ministerio de Salud, 2000) la diabetes es causante de complicaciones como accidentes cerebrovasculares, enfermedades cardiovasculares, pérdida de la vista, insuficiencia renal y amputación de miembros. La hipertensión es causante de infarto de miocardio e insuficiencia cardíaca.

La adherencia al tratamiento de estas enfermedades involucra cambios abruptos en el estilo de vida del paciente, puesto que se debe seguir un estricto tratamiento farmacológico acompañado de un constante monitoreo de hábitos y signos vitales para prevenir cualquier descompensación como consecuencia de valores anormales en sus mediciones (Martínez y otros, 2015). No obstante, la falta de adherencia al tratamiento ha llegado a ser un problema que se presenta con frecuencia. Estudios han demostrado que se esto se debe a factores como la edad, la falta de acompañamiento médico, el desconocimiento de factores de riesgo, la falta de disciplina en la automedicación, entre otros factores (Martínez y otros, 2015). Como consecuencia de esto, en muchos casos estos controles pasan a ser una tarea improvisada para el personal médico, permitiendo la progresión de la enfermedad y la aparición de graves complicaciones. Ante esta problemática, surge la necesidad de implementar una estrategia que aporte un efectivo acompañamiento al tratamiento, de tal manera que, ante cualquier anomalía, sea posible tomar acciones rápidas.

Actualmente, se usa una gran cantidad de dispositivos en diferentes campos de la medicina, los cuales cuentan con sensores para generar datos en tiempo real. La inteligencia artificial y la minería de datos han demostrado ser herramientas esenciales en este ámbito. El poder que ofrecen

estas herramientas deriva de la capacidad de metodologías para extraer patrones y diseñar modelos a partir de estos. Gracias a esto, se han implementado soluciones tecnológicas con la capacidad de acompañar el tratamiento de enfermedades (Ioannis Kavakiotis, 2017). Aplicaciones como Cardiogram de Apple, presentan una red neuronal llamada Deep Heart, la cual tiene como fin detectar anomalías en los picos de frecuencia cardíaca con un 82% de certeza. Estudios han demostrado que este modelo puede detectar hipertensión arterial o apnea del sueño con el constante monitoreo y toma de datos que realiza el dispositivo SmartWatch (ilumivu, 2022). Por otro lado, la empresa de dispositivos Wearables Accu-check presenta MySugr Pro, una aplicación destinada al control de la diabetes llevando seguimiento de las mediciones de glucemia, actividad diaria y dieta del usuario (Accu-check, 2019).

Como propuesta, dada la capacidad tecnológica que tienen los dispositivos wearables, con la ayuda de modelos de inteligencia artificial, se diseñará un motor de análisis de datos que integre diferentes modelos para apoyar la prevención de complicaciones identificando los patrones que puedan incidir en la aparición de complicaciones. Con esto, se pretende mejorar la adherencia al tratamiento proporcionando un monitoreo más completo, rápido y preciso, de tal manera que tanto el personal médico como el paciente, puedan estar informados. A futuro, se espera tener un producto que no solamente se enfoque en tratar de informar los signos vitales del paciente, sino que también pueda predecir y prevenir los riesgos que implica el comportamiento de estos, recibiendo la atención médica de manera oportuna.

1. Planteamiento y Justificación del Problema

La Hipertensión Arterial (HTA) y la Diabetes Mellitus pertenecen al grupo de enfermedades no transmisibles que más causan muertes prematuras en el mundo. La OMS (Organización mundial de la salud) en 2019 situó a la diabetes como la novena causa de muerte en el mundo con 1,5 millones de muertes directas (OMS, 2021). Mientras que la hipertensión es responsable de 1.6 millones de muertes al año en la región de América (OMS, 2021). Se estima que hay más de 500 millones de personas con diabetes y 1400 millones con hipertensión en el mundo. Así mismo, la prevalencia de estas enfermedades ha tenido un rápido crecimiento en países de recursos medianos y bajos, lo cual ha causado preocupación debido a las proyecciones del aumento de casos en las siguientes décadas (OPS, s.f.).

El panorama de Latinoamérica con respecto al tratamiento de estas enfermedades no es alentador, dado que es más alta la probabilidad de sufrir complicaciones comparado a los países desarrollados (L. Veliz-Rojasa, 2014). En Colombia, aproximadamente 4 de cada 10 adultos sufren de hipertensión arterial, pero el 60% de estos no lo sabe aún (Ministerio de Salud Y Protección Social, 2020). La Asociación Colombiana de Diabetes ha estimado que el 7% de la población colombiana mayor de 30 años tiene Diabetes tipo 2 y alrededor de un 30 a 40% de los afectados desconocen su enfermedad (Ministerio de Salud Y Protección Social, 2020). Los riesgos de la hipertensión se ven incrementados si coexiste con otras enfermedades, especialmente la diabetes, donde la probabilidad de padecer hipertensión es 2 a 4 veces más alta que una persona no diabética.

La severidad de las complicaciones y los medios que requieren para su control requieren un gasto económico muy alto, dado que se necesita una gran cantidad de fármacos y un buen

acompañamiento médico para reducir la progresión de complicaciones (Ministerio de Salud, 2000). Entre estas están accidentes cerebrovasculares, nefropatías, enfermedades cardiovasculares, insuficiencia renal, ceguera, amputación de miembros, entre otros (OPS, s.f.). Además, en muchos casos la atención médica suele estar saturada de pacientes, y a menudo es difícil para las personas adherirse a las rutinas de autocontrol (Hartz y otros, 2016). Por ejemplo, muchos pacientes no controlan su nivel de glucosa en la sangre o no se inyectan insulina con frecuencia, otros no evalúan correctamente los factores de riesgo que puedan emerger de cada medición periódica o la actividad física no es una prioridad en su día a día, lo cual aumenta el sedentarismo y se convierte en un factor de riesgo en la Diabetes e hipertensión (Mattos Martinez & Ochoa Fierro, 2015).

Como solución, las tecnologías de la información pueden ayudar a los pacientes a mejorar la adherencia en sus tratamientos con el uso de la telemedicina. Un entorno inteligente que permite conectar al médico y el paciente mediante dispositivos inteligentes, donde los datos de medición como la glucemia o frecuencia cardíaca se puedan almacenar, procesar, analizar y presentar como información importante en el diagnóstico y pronóstico de la enfermedad (EL-Sappagh y otros, 2019).

A&A soluciones – tic, empresa con el objetivo de proporcionar innovaciones tecnológicas y de valor a consumidores, empresas y clientes en cualquier parte del mundo, brinda la oportunidad a practicantes de complementar su formación profesional en un ambiente laboral que pueda proporcionarle la experiencia necesaria en cualquier área de desarrollo de software. Para lograr el objetivo, A&A soluciones propone Whatoko Health, el cual es un producto que interviene en el campo de la telemedicina con el objetivo de acompañar el tratamiento y prevenir complicaciones en pacientes con hipertensión y/o diabetes.

El proyecto presente contribuirá a Whatoko Health con el diseño de un motor de análisis de datos que permita gestionar los datos recolectados por dispositivos de medición en una base de datos. Estos datos serán procesados y analizados por modelos de inteligencia artificial, los cuales tendrán la tarea de aprender, identificar y predecir el desarrollo de factores de riesgo que puedan comprometer la salud del paciente. Posteriormente, se construirán informes a partir de los resultados que serán transmitidos por medio de servicios REST a los servicios Web de Whatoko Health, de tal manera que la información pueda ser consumida e interpretada tanto por el personal médico como por el paciente.

2. Objetivo

2.1 Objetivo General

Desarrollar un motor de análisis de datos que permita detectar y alertar en tiempo real anomalías que representen un riesgo en pacientes hipertensos y/o diabéticos mediante el uso de modelos de inteligencia artificial.

2.2 Objetivos Específicos

Diseñar el esquema de base de datos que permita almacenar y gestionar la información de usuarios y mediciones.

Diseñar modelos predictivos que permitan detectar y predecir anomalías relacionadas a hipertensión y/o diabetes.

Diseñar un motor de análisis de datos que integre modelos de inteligencia artificial y minería de datos para procesar, analizar y presentar informes de la condición actual del paciente

Diseñar REST API en arquitectura de microservicios para transmitir los reportes de resultados a los Servicios de Whatoko Health.

Diseñar REST API para desplegar modelos de inteligencia artificial en Dashboard.

3. Maco de referencia

Con la finalidad de entender el contenido que se desarrollará en el proyecto, a continuación, se describirán los conceptos teóricos y estado del arte.

3.1 Marco Teórico

3.1.1 Hipertensión Arterial

La hipertensión arterial es una enfermedad que aumenta su número de afectados según la edad de los individuos, en promedio el 10% de las personas entre 20 a 40 años se ven afectados por esta enfermedad, mientras que el 50% de las personas entre 50 a 60 son afectadas. Esta enfermedad ocasiona que la sangre ejerza una presión contra las paredes de las arterias, que son grandes vasos sanguinos por los cuales circula la sangre. No se han encontrado que existan umbrales estrictos para definir la línea entre riesgo y seguridad. Sin embargo, según el consenso internacional, la presión arterial sistólica persistente por encima de 139 mmHg o la presión arterial diastólica por encima de 89 mmHg se asocia con el patrón de la hipertensión. El padecimiento de esta enfermedad puede aumentar el riesgo de enfermedades cardiovasculares, accidentes cerebrales, insuficiencia renal y más. (OMS, 2021).

3.1.2 Diabetes Mellitus

La diabetes mellitus pertenece al grupo de enfermedades crónicas que se presenta cuando el páncreas no genera suficiente insulina o cuando el organismo por problemas genéticos no utiliza de manera adecuada esta hormona. Para una persona diabética, los niveles de glucemia generalmente llegan a niveles más altos de lo normal. El efecto común de este evento es la diabetes no controlada o Hiperglucemia. Mantener los niveles de glucemia en estado de Hiperglucemia puede provocar daños en los órganos y aparición de distintos factores de riesgo (OMS, 2021).

3.1.3 Metodologías Ágiles

Un modelo de desarrollo ágil se define como un proceso incremental, cooperativo, sencillo y adaptativo. Estas se explican mediante una serie de consejos y principios junto a técnicas que

hacen que la entrega de un proyecto tenga una menor dificultad y evitando de esta manera los caminos burocráticos de las metodologías tradicionales, generando poca documentación y no haciendo uso de métodos formales. Estas metodologías ponen de relevancia que la capacidad de respuesta a un cambio es más importante que el seguimiento estricto de un plan (Maida & Pacienzia, 2015).

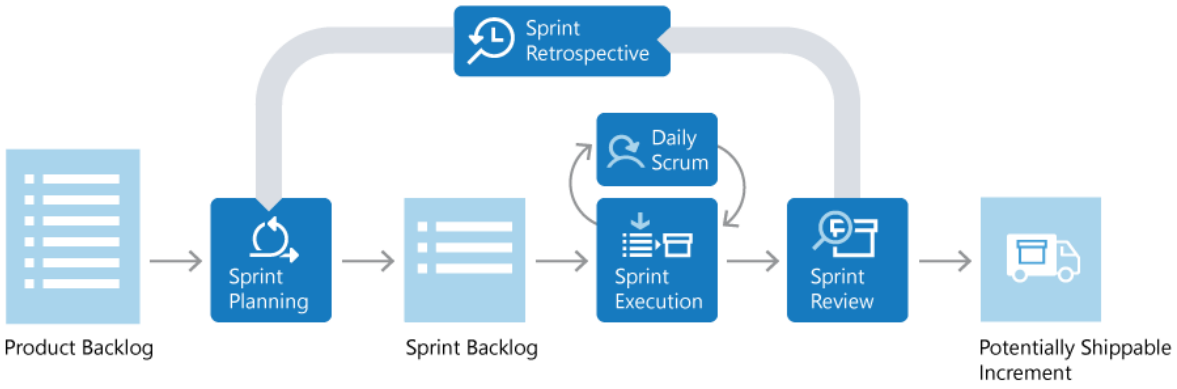
3.1.4 Scrum

Scrum es un marco de trabajo utilizado para el desarrollo ágil que permite dar soporte a la innovación basándose en equipos auto gestionados (Hema y otros, 2020).

Su filosofía se basa en dos conceptos: El empirismo y el pensamiento Lean, donde el empirismo establece que el conocimiento surge de la experiencia. Por otro lado, el pensamiento Lean reduce el desperdicio y se centra en lo importante (Schwaber & Sutherland, 2020). Scrum emplea un enfoque iterativo e incremental, de manera que se pueda controlar el riesgo conformando grupos de personas que en conjunto representan las diversas habilidades y experiencias para hacer el trabajo (Schwaber & Sutherland, 2020).

Scrum cuenta con unos principios fundamentales llamados pilares, los cuales sustentan la base del éxito de scrum. Estos pilares se basan en la transparencia, inspección y adaptación. Por esta razón, los aspectos claves del proceso deben ser visibles para todos los implicados en el resultad, de manera que se deba inspeccionar frecuentemente los artefactos y el progreso para detectar desviaciones indeseadas. El proceso debe ajustarse cuando se detectan desviaciones fuera de los límites aceptables. En base a estos principios el equipo scrum debe incorporar y practicar sus valores: compromiso, coraje, foco, apertura y respeto (Microsoft, 2022)

Figura 1
Microsoft. Ciclo de vida del marco de trabajo SCRUM.



Nota: Ciclo de Vida del marco SCRUM. Tomado de mijacobs. (s. f.). *¿Qué es Scrum? - Azure DevOps*. Recuperado 10 de noviembre de 2022, de https://learn.microsoft.com/es-es/devops/plan/what-is-scrum

3.1.4.1 Scrum Team. La unidad básica de Scrum es un pequeño grupo de personas compuesto por un Scrum Master, un propietario del producto y los desarrolladores. Es una unidad cohesiva de profesionales enfocados en una meta a la vez, la meta del producto (Schwaber & Sutherland, 2020).

El equipo scrum es el responsable de todas aquellas actividades relacionadas con el producto, incluida la colaboración, validación, el mantenimiento, operaciones, experimentación y el desarrollo de las partes interesadas (Schwaber & Sutherland, 2020). Scrum define tres responsabilidades específicas que se debe tener en todo equipo Scrum, las cuales son: Developers, Product Owner y el Scrum Master (Schwaber & Sutherland, 2020).

3.1.4.2 Scrum Máster. El Scrum Máster es el responsable de establecer cómo se deben seguir los conceptos definidos en la guía Scrum (Schwaber & Sutherland, 2020). Como objetivo

principal, es orientar a todos a comprender la teoría y la práctica, tanto dentro del Scrum Team como de la organización involucrada (Schwaber & Sutherland, 2020). Para ello guía los miembros para tener un comportamiento autogestionado y multifuncional.

3.1.4.3 Sprint. Los Sprints son el corazón del Scrum y es donde los requerimientos se convierten en valor. Es un período breve de tiempo fijo en que el equipo Scrum trabaja para completar una cantidad de trabajo establecida (Schwaber & Sutherland, 2020). Cada Sprint puede considerarse un proyecto corto que en el tiempo irá incrementando el valor del producto final (Schwaber & Sutherland, 2020).

3.1.4.4 Sprint Planning. El Sprint Planning es uno de los cinco eventos de Scrum y es el primero que se hace al comenzar un Sprint. Es una reunión del Scrum Team donde se va a planificar qué es lo que se va a hacer durante el Sprint y cómo se va a hacer. El Product Owner propone cómo el producto podría incrementar su valor mediante el producto backlog, del cual saldrá la lista de Sprints. Luego, todo el equipo define en colaboración el sprint Goal. La planificación de Sprint tiene un límite de tiempo máximo de ocho horas para un Sprint de un mes (Schwaber & Sutherland, 2020). Para cada elemento del Product Backlog seleccionado, el desarrollador programa la creación del trabajo requerido para satisfacer la finalización del incremento definido. Esto generalmente se hace desglosado los elementos del Product Backlog en elementos de trabajo más pequeños de un día o menos.

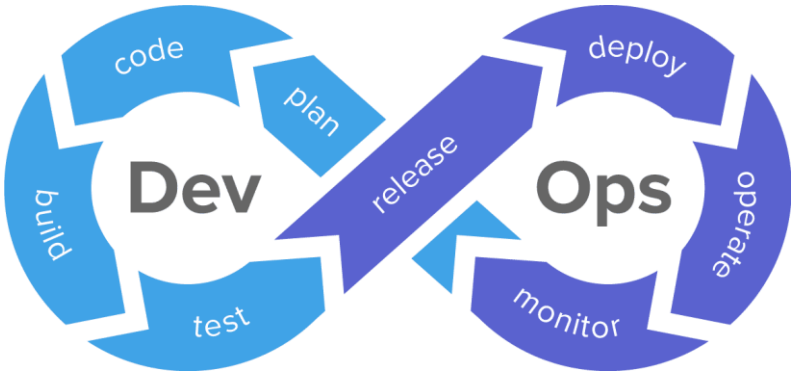
3.1.4.5 Daily Scrum. El Daily Scrum es un evento de la práctica SCRUM que se encarga de inspeccionar el progreso de cada incremento del Sprint (Schwaber & Sutherland, 2020). Esto es utilizado por los Developers para estar siempre informados del progreso realizado y lo qué se

requiere en la siguiente iteración. Esto permite crear un enfoque y organización para la autogestión (Schwaber & Sutherland, 2020).

3.1.4.6 Product Backlog. El Product Backlog es la lista ordenada de todos los elementos que definen el objetivo del producto. Dicha lista se estructura como una pila donde los elementos que se encuentran en la parte superior son los que mayor valor aportan al negocio. Cada elemento se le conoce como PBI(Product Backlog Item) (Schwaber & Sutherland, 2020).

3.1.4.7 DevOps. La palabra "DevOps" es una combinación de las palabras "desarrollo" y "operación" (AWS, 2022). En algunos modelos DevOps, el control de calidad y la seguridad realizados por los equipos están más integrados, es decir, las áreas de desarrollo y las operaciones intervienen durante una parte significativa del ciclo de vida completo de la aplicación (Verona y otros, 2016). Cuando la seguridad es una prioridad principal para todos los miembros de DevOps, a veces se lo denomina operaciones de desarrollo de seguridad. Los equipos utilizan diferentes prácticas para automatizar procesos que antes eran manuales y lentos a través de un conjunto de herramientas y técnicas que ayudan a operar y mejorar el desarrollo de aplicaciones de manera rápida y confiable. Estas herramientas ayudan a los ingenieros a realizar tareas de forma independiente que normalmente requieren la participación de otros equipos, lo que mejora aún más el rendimiento del equipo (Verona y otros, 2016).

Figura 2
Gráfico de DevOps



Nota: Ciclo de DevOps. Tomado de *DEVOPS – Grupo Linux S.A.* (s. f.). Recuperado 10 de noviembre de 2022, de https://www.grupolinux.net/main/devops-colombia/

DevOps es un conjunto de prácticas, herramientas y creencias filosóficas utilizadas para que las empresas y/o grupos de desarrollo puedan mejorar la habilidad de entregar servicios o aplicaciones a un ritmo elevado (AWS, 2022). Estas prácticas, herramientas y creencias se basan en el desarrollo ágil de software, la integración continua (CI – Continuous Integration) y entrega continua (CD – Coninuous Delivery), la infraestructura como código (IaC – Ingrasturucture as Code), el control de versiones, la gestión de la configuración, monitorización continua y el desarrollo basado en microservicios (AWS, 2022). La metodología de desarrollo DevOps se da a partir de la creación de un equipo integral que se encarga de todo el ciclo de vida del software, desde el desarrollo y las pruebas hasta el despliegue y las operaciones (AWS, 2022).

3.1.4.8 Deep Learning. Deep Learning (aprendizaje profundo) es un conjunto de algoritmos de aprendizaje automático (Machine Learning) que se basan en el funcionamiento biológico del cerebro humano, aunque lejos de igualar su capacidad, permite que los modelos computacionales que se componen de múltiples capas de procesamiento aprendan

representaciones de datos con múltiples niveles de abstracción (LeCun y otros, 2015). Estos métodos han mejorado drásticamente el estado del arte en el reconocimiento de voz, el reconocimiento de objetos visuales, la detección de objetos y muchos otros dominios, como el descubrimiento de fármacos y la genómica. El aprendizaje profundo descubre estructuras complejas en grandes conjuntos de datos mediante el uso de algoritmos de retro propagación para decirle a la máquina cómo cambiar sus parámetros internos, que se utilizan para calcular la representación de cada capa en función de la representación de la capa anterior. Las redes convolucionales profundas han generado avances en el procesamiento de imágenes, video, voz y audio, mientras que las redes recurrentes han arrojado luz sobre datos secuenciales como texto y voz (LeCun y otros, 2015).

3.1.5 Machine Learning

Machine Learning (Aprendizaje automático) es una rama de la inteligencia artificial que aplica algoritmos sistemáticamente para encontrar patrones ocultos en las relaciones de los datos (Campesato, 2020). En términos de alto nivel, el aprendizaje automático puede resolver tareas que son imposibles o demasiado engorrosas con los lenguajes de programación “tradicionales”. Un filtro de Spam para correos electrónicos es un ejemplo temprano de aprendizaje automático (Khanna & Awad, 2015).

3.1.6 Rest API

Una API de REST, o API de RESTful, es una interfaz de programación de aplicaciones que basa en los límites de la arquitectura REST (RetHat, 2020).

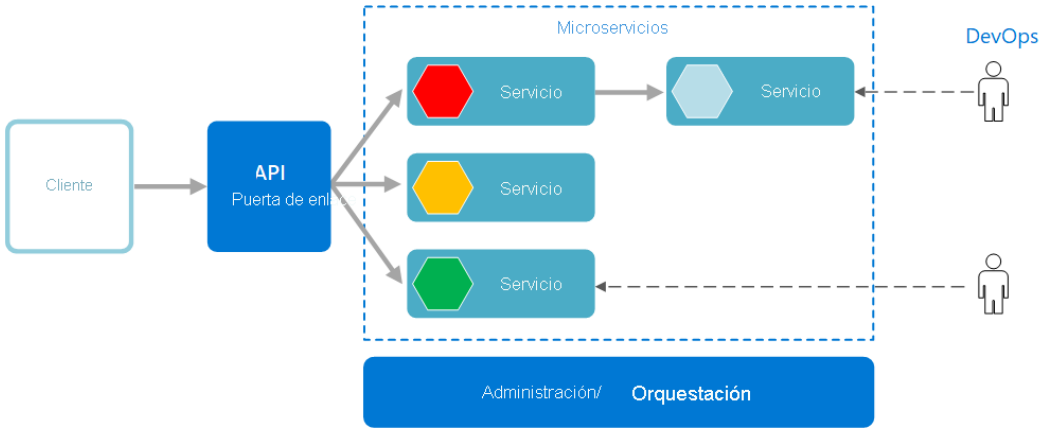
La API es una analogía al contrato entre un proveedor de información(servidor) y el usuario(cliente), de manera que el proveedor entrega el contenido o información mediante la

llamada(petición) que realiza el usuario hasta recibir una respuesta. En resumen, las API permiten interactuar con una computadora o sistemas para así recuperar datos o realizar varias funciones de manera que el sistema pueda entender y satisfacer sus necesidades (RetHat, 2020).

Las características de un API REST son el uso del modelo REST (Transferencia de estado representacional), usa el protocolo HTTP que resulta conocido entre los desarrolladores web, es independiente del lenguaje, debido a que se puede consumir el servicio a través de cualquier lenguaje diferente al del API. Dicho protocolo implementa las operaciones de CRUD en las aplicaciones web (RetHat, 2020).

3.1.7 Arquitectura de Microservicios

Figura 3
Microsoft. (2022). Estilo de arquitectura de microservicios



Nota: Arquitectura de Microservicios. Tomado de *DEVOPS – Grupo Linux S.A.* (s. f.). Recuperado 10 de noviembre de 2022, de https://www.grupolinux.net/main/devops-colombia/

Las arquitecturas basadas en microservicios surgen para dar respuesta a los problemas de escalado de aplicaciones, en particular cuando se trata de aplicaciones distribuidas (Roldán Martinez y otros, 2018). Esta arquitectura consiste en crear una sola aplicación como un conjunto de servicios pequeños e independientes con su propia persistencia de la información (Microsoft, 2022). Cada servicio se ejecuta en su propio entorno y se comunica con otros servicios mediante una interfaz definida en las aplicaciones REST(API), el cual es el protocolo HTTP. Una de las características más notables de esta arquitectura, es que cada microservicio puede diferir en tecnología respecto a los demás, lo que significa que facilita la evolución y escalabilidad de cada componente sin dañar el funcionamiento general (Roldán Martinez y otros, 2018).

3.1.8 Docker

Docker es una plataforma con la funcionalidad de desarrollar y desplegar aplicaciones en unidades básicas estandarizadas llamadas contenedores (AWS, 2022). Estos contenedores contienen librerías, herramientas de sistema, código y tiempo de ejecución. La forma en que funciona Docker, es similar a cómo una máquina virtual, de manera que virtualiza el hardware del servidor con la diferencia de que comparte el kernell de Linux.

3.1.9 Kubernets

Kubernets es una plataforma portable y extensible de código abierto que permite administrar cargas de trabajo y servicios. Esta se caracteriza como una plataforma que puede contener contenedores, microservicios o portable de nube (Kubernetes, 2022).

Kubernets ofrece un entorno de administración centrado en contenedores. Para ello, orquesta la infraestructura de cómputo, redes y almacenamiento para que las cargas de trabajo no tengan que ser realizadas por los usuarios.

3.1.10 CI/CD Integración y Distribución Continua

La CI/CD es un método para distribuir aplicaciones a los clientes con mayor frecuencia, gracias al uso de la automatización en el ciclo de vida de las aplicaciones. Los principales términos que conforman esta tarea son la integración, la distribución y la implementación continua (RedHat, 2018). Esto consiste en una solución para los problemas que puede generar la integración del código nuevo para los equipos de desarrollo y de operaciones.

En términos de alto nivel, la distribución continua se refiere a cómo los desarrolladores automatizan los cambios que se generan desde el repositorio de producción hasta exponerlos a los stakeholders. La Integración continua se enfoca en optimizar la combinación de cambios nuevos por distintos desarrolladores en un proyecto compartido (RedHat, 2018).

3.1.11 Control de Versiones

El control de versiones (VCS por sus siglas en inglés) es un sistema que permite registrar los cambios realizados en un archivo o conjunto de archivos transcurrido en el tiempo, de manera que se pueda contar una trazabilidad entre las distintas versiones y poder moverse entre las distintas versiones (Git, 2022). Usar un sistema de control de versiones es una buena práctica al momento de realizar un proyecto. Dicho sistema permite a los usuarios regresar a versiones anteriores, comparar cambios a lo largo del tiempo, ver quién modificó por última vez algo que pueda estar causando problemas, ver quién introdujo el problema, cuándo, y mucho más. Con git, además es posible recuperar archivos en el peor de los casos si se arruinan o se pierden archivos (Git, 2022). Esta necesidad surge debido a la dificultad de gestionar las copias de un software, puesto que para lanzar un programa y hacer correcciones, en muchos casos estas correcciones llegaron a corromper

el funcionamiento general del software, por lo cual, hacía muy tediosa la tarea de gestionar las versiones al momento de desplegarlo.

3.1.12 Base de Datos

Las bases de datos se conforman como un conjunto de información almacenados en un sistema llamado usualmente gestor de base de datos (DMBS). Un sistema de base de datos se conoce como el conjunto conformado por el gestor de base de datos, los datos y las aplicaciones complementadas (Microsoft, 2022).. En la actualidad, los tipos más utilizado de base de datos son los siguientes

3.1.12.1 Base de Datos Relacionales. Las bases de datos relacionales nacen debido a la repetición y el almacenamiento de datos vacíos que se podía llegar a tener en un almacenamiento continúo conformado a partir de columnas en una plantilla ordenada (Oracle, 2022). En este sentido las bases de datos relacionales permiten llevar un mejor almacenamiento de los datos con menos repeticiones y valores vacíos. Las relaciones entre las tablas de una base de datos relacional permiten poder agrupar de diferentes maneras la información según se requiera. Hoy en día existen distintos softwares que permiten crear y gestionar bases de datos relacionales entre ellos se puede destacar a MySQL, Oracle, Cassandra, SQLite, entre otros. El lenguaje estándar para la creación de las bases de datos relacionales se conoce como SQL (Structured Query Lenguage) el cual se ha mantenido como un estándar entre los distintos softwares que lo utiliza. Estos sistemas de gestión de bases de datos permiten ciertas propiedades conocidas como ACID que suelen garantizar que las transacciones se procesen de manera fiable en una base de datos. Estas propiedades son la atomicidad, la cual nos dice que un cambio debe de completarse en su totalidad o no se debe hacer en absoluto. Consistencia que siempre se debe conducir a un estado válido de la base de datos.

Aislamiento donde los procesos no deben de afectar cambios que se estén realizando al mismo tiempo y por último la durabilidad que debe conservar los cambios, aunque se produzcan fallos en la base de datos o el sistema completo (Piperlab, 2021).

3.1.12.2 Base de Datos No Relacionales. Las bases de datos no relacionales o también conocidas como NoSQL (No solo SQL), difiere del modelo clásico de los sistemas de gestión de bases de datos relacionales debido en principio no usa SQL como lenguaje principal de consultas, además no garantiza completamente las propiedades ACID. Los sistemas de bases de datos no SQL se crecieron por grandes compañías como Google, Amazon o Facebook las cuales necesitaban manejar grandes volúmenes de datos y se percataron que la coherencia de los datos procesados no era más importante que el rendimiento al procesarlos, por este motivo las bases de datos NoSQL se usan para el manejo de grandes volúmenes de datos que permiten realizar en menor tiempo las consultas (Microsoft, 2022).

3.1.13 Modelo Vista Controlador (MVC)

La modelo vista controlador (MVC) es un patrón de arquitectura muy implementado en el desarrollo de software, el cual divide la lógica de la aplicación en tres capas, separando la representación de los procesos en estas tres capas (Universidad de Alicante). Los beneficios que tiene este patrón de arquitectura, es el aislamiento de la lógica de negocio de la interfaz y la conexión a la base de datos. En este patrón de arquitectura las partes cumplen las siguientes funciones

- Modelo: representa la información de la aplicación y las reglas para manipular esos datos. Los modelos se utilizan principalmente para administrar las persistencias a las tablas de la

base de datos. Donde la mayor parte de la lógica comercial de la aplicación se genera en el modelo.

- Vista: representa la interfaz de usuario de la aplicación, dicho de otra manera, se encarga de gestionar las platillas de la aplicación, y de proporcionar los datos al navegador u herramientas para realizar solicitudes a la aplicación.
- Controlador: se encarga de controlar los servicios de los modelos a las vistas, en otras palabras, se encarga de procesar las peticiones, interrogar a los modelos en busca de datos y pasar esos datos a las vistas.

3.2 Estado del arte

En el proyecto presente se quiere indagar, conocer y clasificar las distintas formas tecnológicas en que un paciente puede ser atendido y/o monitoreado. Entender el contexto actual de la telemedicina, los autocuidados y las problemáticas de salud pública que aquejan a nuestra sociedad y, que, en muchos casos, no son atendidas, diagnosticadas o monitorizadas a tiempo.

En este texto se tratará el tema de aplicaciones móviles para el cuidado médico (telemedicina, Smart health, health care, entre otros).

Con esto en mente, se quiere investigar sobre cómo el análisis de datos y la inteligencia artificial aportan el conocimiento de patrones en los datos que suministran los dispositivos wearables. Patrones que pueden indicar lo que está pasando o lo que es probable que pueda pasar, y con ello, usarlo como una herramienta para el autocuidado y la toma de decisiones. De esta manera, este proyecto pretende ser primero una indagación sobre algunos conceptos propios de las nuevas tecnologías y a su vez conocer elementos relacionados con las aplicaciones web o móviles conectadas a dispositivos de monitoreo médico (monitores, dispositivos Wearables, entre otros).

3.2.1 Motor de Predicción de Niveles de Glucemia

Según el artículo (Aliberti, 2019) se realizó la implementación de sistemas de monitoreo continuo de glucemia. Esto, mediante el uso de dispositivos de monitoreo y el consumo de los datos, se proyecta el comportamiento de la concentración glucémica con aprendizaje profundo. Esto con el fin de tener una prevención temprana de estados hiperglucémicos o hipoglucémicos peligrosos, lo cual aporta a la toma de decisiones en el tratamiento de diabetes.

A partir de este artículo se puede resaltar la importancia de la inteligencia artificial aplicada a la medicina, y de cómo al integrarse con un sistema de monitoreo puede anticiparse a eventos catastróficos como los niveles de hiperglucemia. Además de ello, es una referencia clara a los objetivos que se quieren alcanzar en este proyecto como lo es tratar de predecir anomalías con base a grandes cantidades de datos.

3.2.2 Cardiogram

Cardiogram es un asistente personal de atención médica, el cual mediante el uso de dispositivos wearables como Apple Watch, entre otros, realiza un monitoreo continuo de medidas de sueño o actividad física de los usuarios (Cardiogram, 2022). Con estos datos y el uso de su motor de inteligencia llamado Deep Heart ha logrado encontrar correlación y patrones para detectar Apnea del sueño y/o hipertensión con un 82% de precisión. La importancia de esta red neuronal se basa en la capacidad de aprovechar los datos no etiquetados como frecuencia cardíaca en reposo o frecuencia cardíaca variada (Ballinger y otros, 2018).

El estudio y análisis de este modelo resalta la importancia que pueden tener los sistemas de monitoreo acompañados de una inteligencia artificial complementados en una aplicación móvil que puede informar de anomalías.

3.2.3 Zeep Life

Zeep Life es una aplicación que monitorea el sueño, la actividad diaria y los entrenamientos del usuario mediante la vinculación de dispositivos wearables de distintos modelos Smart Watch como Amazfit, Xiaomi, Garmin, entre otros (Zeep, s.f.). La aplicación cuenta con un motor de inteligencia artificial que indican el estado de niveles de sueño, hábitos y actividad física con el fin de informar al usuario e invitar a mejorar estos niveles para su autocuidado.

Uno de los algoritmos que más destaca en esta aplicación es el PAI Personal Activity Intelligence, el cual es un indicador de actividad física basada en los últimos 7 días con mediciones de actividad y frecuencia cardíaca. El objetivo es tener una puntuación de 100 PAI o más. Estudios han demostrado que mantener este puntaje ayuda a prevenir el riesgo de muerte por enfermedades cardiovasculares, diabetes e incluso aumentar la esperanza de vida.

Este aplicativo nos da un ejemplo de cómo un algoritmo puede ayudar a prevenir los factores de riesgo de hipertensión o diabetes como lo es el sedentarismo (Amazfit, s.f.).

4. Desarrollo Del Proyecto

4.1 Entorno de Trabajo

A&ATic es una empresa de personal remoto que ha estado en el mercado por varios años trabajando y realizando sus actividades de manera remota, por lo que implementan ciertas herramientas que ayudan y facilitan la comunicación entre todos los integrantes de la empresa. El conjunto de herramientas y prácticas utilizadas se engloban en lo que se puede definir como el entorno de trabajo o entorno laboral. A continuación se explicará este entorno paso por paso.

4.1.2 Herramientas

La función de estas herramientas dentro de la organización es poder mejorar y organizar los procesos que se desarrollan al interior, con el propósito dar cumplimiento parámetros de evaluación de los procesos.

4.1.2.1 Jira. Es una herramienta desarrollada por Atlassian que permite gestionar y desarrollar software a equipos de desarrolladores siguiendo una metodología scrum de manera ágil y colaborativa, ayuda a reducir los tiempos, planificar, supervisar y medir la flexibilidad del equipo. Esto se logra a través del uso de las herramientas que incorpora, como un tablero Kanban, generación de sprints a los cuales se le pueden crear historias de usuarios con tareas, asignar responsables, definir tiempos, controlar los errores y conectar a repositorios remotos como gitlab, bitbucket o github.

4.1.2.2 Confluence. Es una de las variadas herramientas con las cuales cuenta Atlassian la cual permite trabajar de manera colaborativa, permitiendo dar a conocer los avances que se realizan en el proyecto a los demás integrantes del equipo sobre distintos aspectos del desarrollo, tales la documentación requerida sobre la arquitectura, manejo de los estándares de calidad, información sobre proyecto, para gestionar mejor la documentación y conocimientos que se han registrado durante todo el proceso de desarrollo.

4.1.2.3 Skype. Es un software de comunicación que permite realizar llamadas y comunicarse a través de chats, ya sea con personas individuales o en grupo de personas más grandes, por lo que en las organizaciones es una herramienta que ayuda a los equipos poder gestionar mejor los avances gracias a la facilidad que se tiene para comunicarse con los demás integrantes y poder realizar videollamadas que ayudan a mejorar el desarrollo de las actividades

dentro de la organización, esta herramienta se usa para dar cumplimiento a los eventos propuestos en una metodología scrum.

4.2 Análisis y Planeación

En esta fase se realizó el análisis de los objetivos establecidos en el proyecto para definir los requerimientos, arquitectura del proyecto y tecnologías que se implementaron en la fase de desarrollo.

4.2.1 Requerimientos

La definición de requerimientos es el punto de partida para definir qué acciones debe realizar el producto. Para el contexto de Whatoko health y el presente proyecto, se implementó un sistema de gestión y análisis de datos para usuarios con un perfil de Diabetes y/o Hipertensión. A partir de esto, se definió a qué datos se les iba a realizar la persistencia, cómo se organizaron, qué tecnologías se utilizaron y qué funcionalidades se implementaron. Todo esto se respondió a partir del levantamiento de requerimientos, donde cada uno describe la funcionalidad del sistema. A continuación, se ilustrarán los requerimientos definidos en el proyecto.

4.2.1.1 Levantamiento de Requerimientos

Tabla 1

Gestión de perfil lípido

Prioridad	**Alta**
Identificación	RF11
Nombre	Gestión de perfil biomédico

Descripción	La base de datos debe permitir gestionar datos relacionados al biotipo del usuario

Tabla 2

Gestión de hábitos

Prioridad	**Alta**
Identificación	RF12
Nombre	Gestión de hábitos
Descripción	La base de datos debe permitir almacenar y gestionar información de todos los tipos de hábitos del usuario

Tabla 3

Gestión de perfil lípido

Prioridad	**Media**
Identificación	RF13
Nombre	Gestión de perfil lípidico
Descripción	La base de datos debe permitir almacenar y gestionar datos de colesterol del usuario

Tabla 4

Gestión de mediciones

Prioridad	**Alta**
Identificación	RF14
Nombre	Gestión de mediciones
Descripción	La base de datos debe permitir almacenar y gestionar todos los datos que suministrarán los dispositivos de monitoreo que se conectarán al aplicativo

Tabla 5

Gestión de Reportes de Dashboard

Prioridad	**Alta**
Identificación	RF15
Nombre	Gestión de Reportes de Dashboard
Descripción	Se debe permitir procesar los datos recolectados de todos los usuarios para entregar informes del comportamiento poblacional e individual

Tabla 6

Generador de Predicciones

Prioridad	**Alta**
Identificación	RF16
Nombre	Generador de Predicciones

Descripción	El motor de inteligencia artificial debe permitir automatizar el proceso de identificación de anomalías con la información almacenada en base de datos.

Tabla 7

Generador de Alertas

Prioridad	**Alta**
Identificación	**RF17**
Nombre	Generar Alertas
Descripción	El motor de inteligencia artificial debe generar alertas cuando se detecten anomalías.

4.2.2 Modelado de Base de Datos

Como se mencionó anteriormente, la idea general es almacenar toda la información proveniente de dispositivos wereables e información básica del usuario. Para ello se recolectó información de datos personales, datos sociodemográficos, perfil biométrico, hábitos, perfil lípido, diagnóstico de enfermedades y mediciones de los dispositivos. Teniendo en cuenta esto, en las siguientes tablas se describen los datos a gestionar.

Tabla 8

Tabla relacionada con los perfiles de usuario.

Tipo de datos	Descripción

Datos Personales	Datos del usuario con relación a identificación, credenciales de autenticación
Datos Descriptivos	Datos que describen la etnia, género, edad o biología del usuario.
Hábitos	Cualquier dato que pueda brindar información de los hábitos físicos, alimenticios, intelectuales, entre otros.
Perfil del Biotipo	Información relacionada al físico del usuario, como altura, peso, índice de masa corporal, entre otros.
Diagnósticos	Historial de enfermedades entre las que incluye si tiene diabetes o hipertensión.
Deportes	Deportes que realiza el usuario en su rutina diaria o que realizará en el momento que sincronice el dispositivo wearable con la aplicación

Tabla 9

Datos relacionados a las mediciones del usuario.

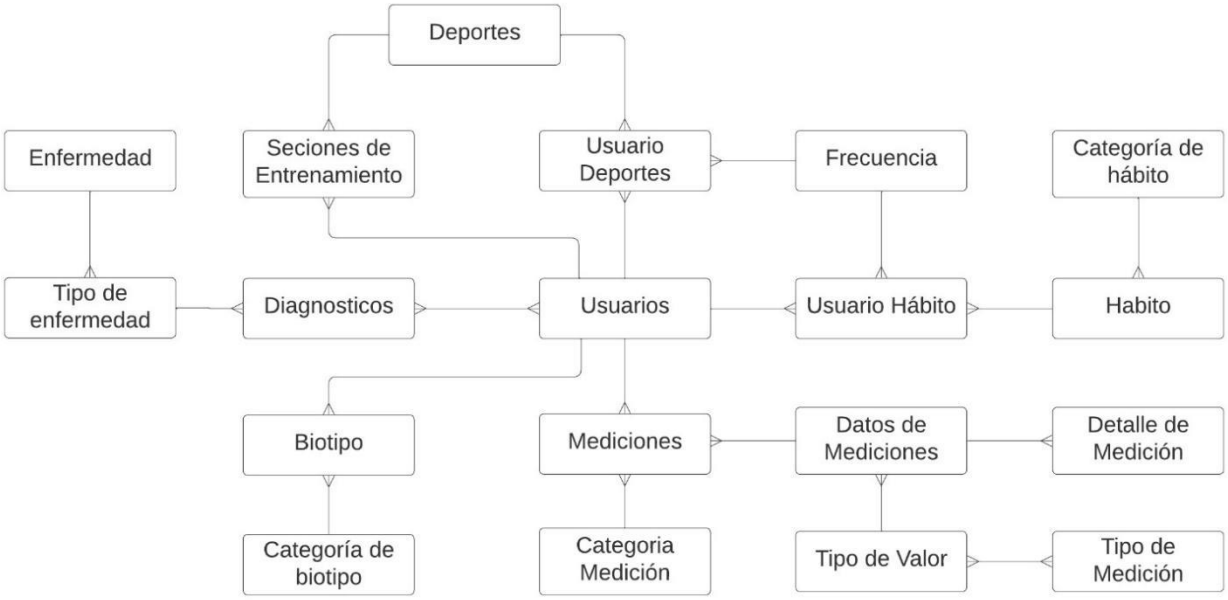
Tipo de medición	**Valores**	**Descripción**
Ritmo Cardíaco	Ritmo cardíaco continuo	Ritmo cardíaco que se toma cada minuto
	Ritmo cardíaco en reposo	Ritmo cardíaco que se toma cuando no se detecta movimiento
Actividad	Calorías	Calorías quemadas en unidades Beat

	Pasos	Número de pasos que realiza el usuario mediante el sensor podómetro
	Distancia	Distancia calculada a partir del movimiento del usuario
Glucemia	Glucemia en ayunas	Nivel de glucosa en la sangre tomada antes de comer
	Glucemia después de comer	Nivel de glucosa en la sangre tomada 2 horas mínimo después de comer.
	Glucemia Tarde	Nivel de glucosa en la sangre después de almuerzo
	Glucemia Noche	Nivel de glucosa en la sangre antes de dormir
	Glucemia general	Nivel de glucosa en la sangre tomada en cualquier momento.
Tensión Arterial	Presión Arterial Sistólica	Esta medición representa el evento cuando los ventrículos del corazón se contraen. Es la cifra más alta.
	Presión Arterial Diastólica	Esta medición representa el evento cuando los ventrículos del corazón se relajan. Dicha medición es la cifra más baja.
Colesterol	Colesterol Total	Nivel de colesterol a nivel general calculado a partir de las lipoproteínas como HDL, LDL y triglicéridos.
	Colesterol Bueno (HDL)	Colesterol de las lipoproteínas de alta densidad.

	Colesterol Malo (LDL)	Al colesterol de las lipoproteínas de baja densidad.
	Triglicéridos	Tipo de grasa fabricada en el hígado. Importante para medir los niveles de sobrepeso de una persona.
Saturación de oxígeno	Saturación	Porcentaje de saturación de oxígeno en sangre.
Actividad Física	Frecuencia cardiaca	Oscilación de latidos del corazón en una unidad de tiempo, generalmente minutos.
	MET	Unidad de medida del índice metabólico, igual a la cantidad de calor emitido por una persona en posición sedente por metro cuadrado de piel.

Teniendo en cuenta la información de las tablas, se diseñó el modelo entidad relación para la capa de gestión datos y mediciones del usuario.

Figura 4
Modelo Entidad Relación



4.2.2.1 Gestión de Mediciones. El proceso de gestión de mediciones se realiza en 6 tablas las cuales se explicarán a continuación.

4.2.2.1.1 Mediciones. Tiene la finalidad de almacenar la relación usuario y medición. Cada medición se almacena en una fecha o intervalo de tiempo, además de guardar la relación de categoría según el tipo de medición.

4.2.2.1.2 Datos de medición. Tiene la finalidad de almacenar los distintos valores que puede tener un tipo de medición, los cuales pueden ser de uno o más valores. Cada valor se etiquetará según el tipo de valor.

4.2.2.1.3 Tipos de valor. Tiene la finalidad de etiquetar los valores que se almacenan en la tabla datos de medición, de tal manera que pueda identificar si es un valor de pulso, glucemia, entre otros.

4.2.2.1.4 Tipos de Medición. Esta tabla tiene la finalidad de almacenar los diversos tipos de medición. Por ejemplo, ritmo cardíaco, glucemia, tensión arterial, entre otros.

4.2.2.1.5 Categorías de Medición. Esta tabla almacena las categorías que clasifican las mediciones según el tipo de medición. Cada lista de categorías estará relacionada a un tipo de medición.

4.2.2.1.6 Detalle de Medición. Esta tabla almacena información adicional de la medición. Por ejemplo, si la medición de glucosa es en ayunas o post comida. Si la medición de tensión arterial se tomó de pie o acostado.

4.2.2.2 Gestión de Perfil del Biotipo. En este apartado se gestionan datos de forma histórica, de tal manera que se pueda tener un seguimiento de los cambios físicos que tendrá el usuario en el tiempo.

4.2.2.2.1 Biotipos. Tiene la finalidad de almacenar en el tiempo los valores de peso, estatura, circunferencia abdominal e índice de masa corporal del usuario.

4.2.2.2.2 Categoría del biotipo. Tiene la finalidad de clasificar la contextura del paciente, de tal manera que se tenga un informe de si se encuentra en condiciones normales o en sobrepeso.

4.2.3 Propuesta de Modelos de Inteligencia Artificial.

A partir del conjunto de datos definido anteriormente se realizó la propuesta de modelos predictivos para Whatoko Health.

4.2.3.1 Modelo Predictivo para el Diagnostico del Riesgo de Diabetes Mellitus 2. La implementación de este modelo se basó en un artículo de investigación de metodologías para predecir el comportamiento metabólico de las mediciones de glucosa (Cabrera Rode, 2017). El

objetivo del modelo es predecir cuál es la probabilidad de diagnóstico de diabetes mellitus tipo 2 en los próximos 2.5 años. Para ello, se tuvieron en cuenta las siguientes variables y criterios.

Tabla 10

Puntuación de riesgos para predecir diabetes tipo 2 a los 2,5 años.

Campo	Criterio	Puntaje
Género	FEMENINO	0
	MASCULINO	1
Triglicéridos(mmol/L)	0 – 1.7	0
	> 1.7	3
Circunferencia abdominal(cm)	0 – 98	0
	99 – 105	1
	>105	3
Glucosa en Ayunas	0 – 6.1	0
	>6.1	2
Altura	0-168	3
	>168	0
Enfermedad Cardiovascular	NO	0
	SI	4
Hipertensión Arterial	NO	0
	SI	2

Nota. Puntuación de riesgos para predecir diabetes tipo 2 a los 2,5 años. Cabrera Rode, E., Rodríguez Camerón, V., Rodríguez, J., Cubas Dueñas, I., Álvarez Álvarez, A., Arnold Domínguez, Y., & Díaz Díaz, O. (2017). Evaluación de tres metodologías para la predicción del riesgo de alteraciones del metabolismo de la glucosa en sujetos con sobrepeso y obesidad. *Revista Cubana de Endocrinología*, *28*(2), 0-0.

A partir de estos valores, se suman los puntos, donde el resultado es el equivalente a una probabilidad de riesgo. Esta probabilidad de riesgo se categorizó en 3 intervalos para describir la zona de riesgo en la que se encuentra el usuario: RIESGO BAJO, RIESGO MEDIO Y RIESGO ALTO.

Tabla 11

Predicción Individual del riesgo de padecer diabetes a los 2,5 años.

Puntos	**Riesgo de Diagnóstico**
0	12.51
1	14.46
2	16.73
3	19.23
4	22.1
5	25.32
6	28.92
7	32.91

8	37.29
9	42.09
10	47.15
11	52.56
12	58.18
13	63.92
14	69.63
15	75.18
16	80.39
17	85.11
18	89.21

Nota. Probabilidades de Riesgo tomado de Cabrera Rode, E., Rodríguez Camerón, V., Rodríguez, J., Cubas Dueñas, I., Álvarez Álvarez, A., Arnold Domínguez, Y., & Díaz Díaz, O. (2017). Evaluación de tres metodologías para la predicción del riesgo de alteraciones del metabolismo de la glucosa en sujetos con sobrepeso y obesidad. *Revista Cubana de Endocrinología*, *28*(2), 0-0.

Con el resultado de la tabla 2, se obtiene la variable objetivo, la cual representa la probabilidad de riesgo de padecer Diabetes tipo 2. Con todo lo mencionado anteriormente, se realizó la función hipótesis de una regresión lineal múltiple.

$$f_w\,(x) = w_0 + w_1\,x_1 + w_2\,x_2 + w_3\,x_3 + w_4\,x_4 + \cdots + w_n\,x_n$$

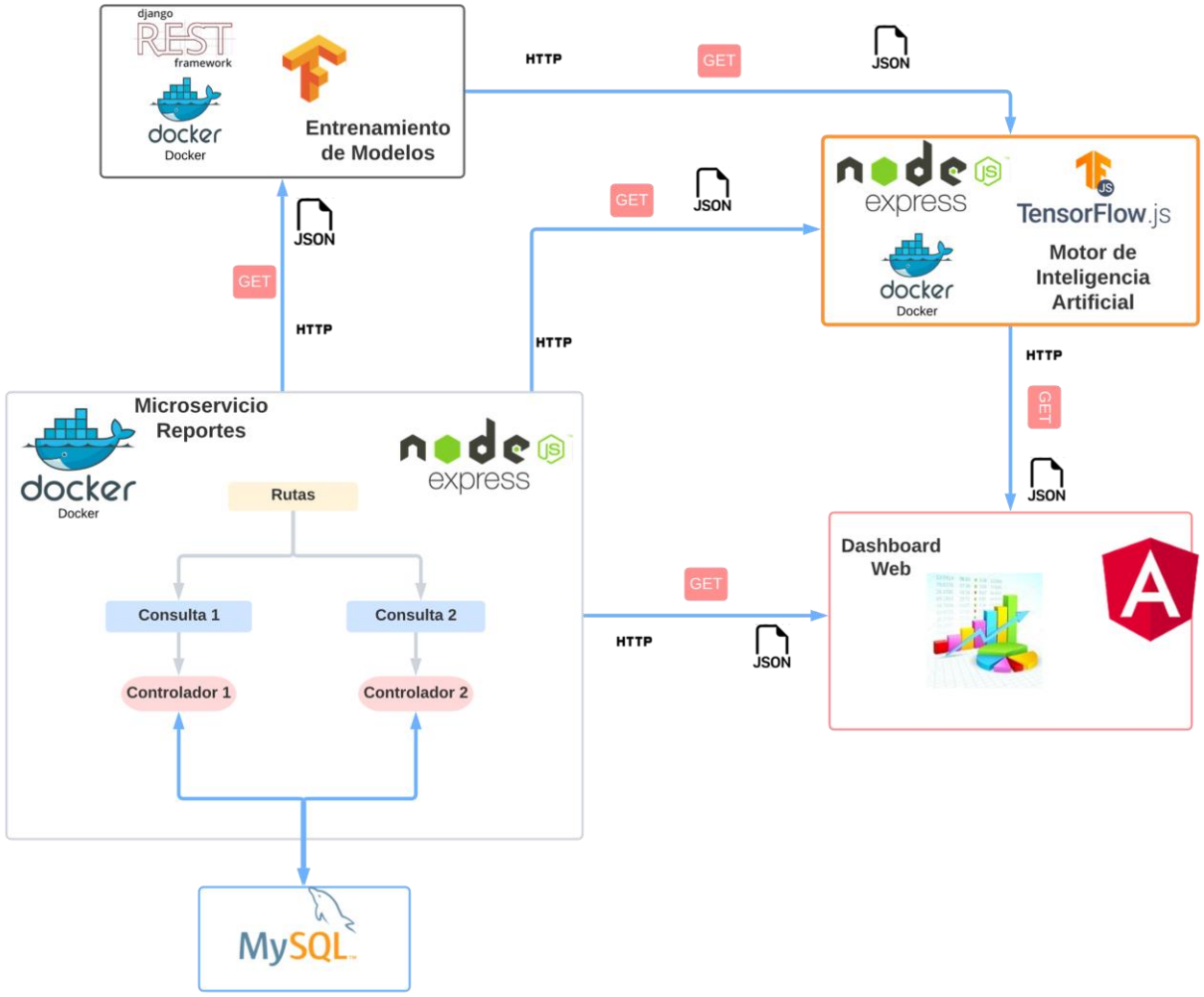
Con ello se define el conjunto de datos de entrenamiento con las condiciones de cada variable. Este modelo se implementó en la Fase de desarrollo, la cual se explicará más adelante.

4.2.4 Definición de Arquitectura del proyecto

Una vez definido el esquema de base de datos, se realizó el análisis del flujo que describe cómo los datos se convierten en información para alimentar las tareas que realizar el motor de inteligencia artificial. En esta sección se explicarán los distintos servicios que conformaron este proyecto.

4.2.4.1 Modulo de Reportes. El módulo de reportes se encarga de realizar consultas y entregar información con base en el análisis exploratorio de los mismos. Este módulo funciona mediante la implementación de una API REST, el cual contiene una colección de rutas (endpoints) que son el resultado de consultar a la base de datos y encapsular la información en estructuras JSON. Cada consulta tendrá un contexto distinto, entre los cuales será entregar reportes para alimentar el dashboard web y la información que requiere el motor de inteligencia artificial.

Figura 5
Flujo de información de los servicios.

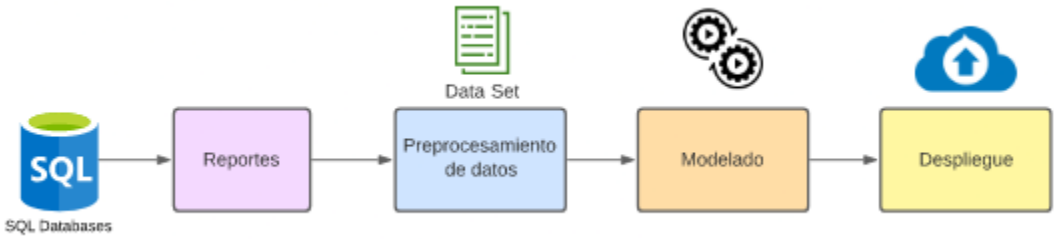


En el diagrama se puede observar cómo el microservicio de reportes trae la información de la base de datos y la entrega a los demás servicios. Este proceso lo realiza el controlador, el cual contiene la lógica que intercepta la petición del cliente y manipula la información consultada según el contexto. El resultado de esto es una estructura JSON, la cual contiene entidades que encapsulan los datos provenientes de la base de datos. Cada controlador realiza una consulta distinta. Por ejemplo, traer la información de las mediciones de glucosa, obtener el número de usuarios

diagnosticados con diabetes mellitus tipo 2 agrupados por género, entre otras. Finalmente, el enrutador se encargará de identificar y exponer el proceso que realiza el controlador, de manera que sea el insumo que cada servicio utiliza para obtener la información que el cliente necesite.

4.2.4.2 Modulo de Entrenamiento de Modelos de Inteligencia Artificial. El módulo de entrenamientos es la capa esencial que se encarga de generar modelos para despliegue en el motor.

Figura 6
Flujo de información hasta llegar al resultado.

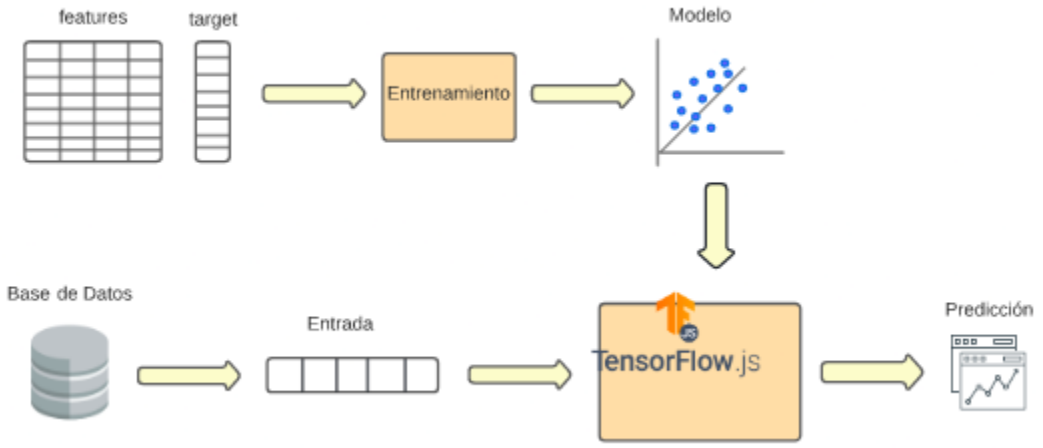


En el diagrama se ilustra el flujo que realiza el módulo de entrenamientos. En primer lugar, debe solicitar los datos que vienen de reportes. Estos datos vendrán en un formato crudo, donde luego pasarán al proceso de mapeo, estructuración, limpieza y preprocesamiento de datos. En esta etapa se definirá el conjunto de datos de entrenamiento que servirá de insumo para la etapa de modelado. Para la etapa de modelado se hará el proceso de entrenamiento según la tarea que se defina: clasificación o regresión en el contexto de aprendizaje supervisado. Finalmente, se hace el análisis de resultados de entrenamiento con el objetivo de seleccionar el modelo para despliegue.

4.2.4.3 Motor de Inteligencia Artificial. El motor de inteligencia artificial es la capa encargada de generar informes de anomalías o riesgos para el contexto de los usuarios de Whatoko Health. Los informes son generados a partir de tareas programadas, las cuales se ejecutan de forma

periódica en un tiempo específico. A continuación, se explicará el paso a paso de la funcionalidad definida para este servicio.

Figura 7
Funcionalidad del motor de inteligencia Artificial



En el diagrama se observa cómo un modelo que ya ha sido entrenado es usado en producción para alimentarse de nuevas entradas, las cuales son traídas de una fuente de datos. El resultado final es la predicción, la cual es el evento que describe cuál será el comportamiento que tendrán los datos. Con base en lo mencionado anteriormente, es necesario definir los módulos que se distribuirán las tareas hasta la predicción. Para ello, se estructuró de la siguiente manera:

4.2.4.3.1 Automatización con Tareas Programadas. Este módulo contiene la lógica para gestionar las distintas tareas de análisis y predicción que realiza el motor en un momento determinado.

4.2.4.3.2 Petición de Datos. En este módulo se define la lógica encargada de realizar las peticiones a la fuente de datos.

4.2.4.3.3 Mapeo y Preprocesamiento de datos. Luego de realizar la petición de datos, es necesario utilizar un paradigma de programación que permita manipular las estructuras JSON y convertirla en un conjunto de datos más simple, de tal manera que a partir de esto se puedan generar las entradas para un determinado modelo.

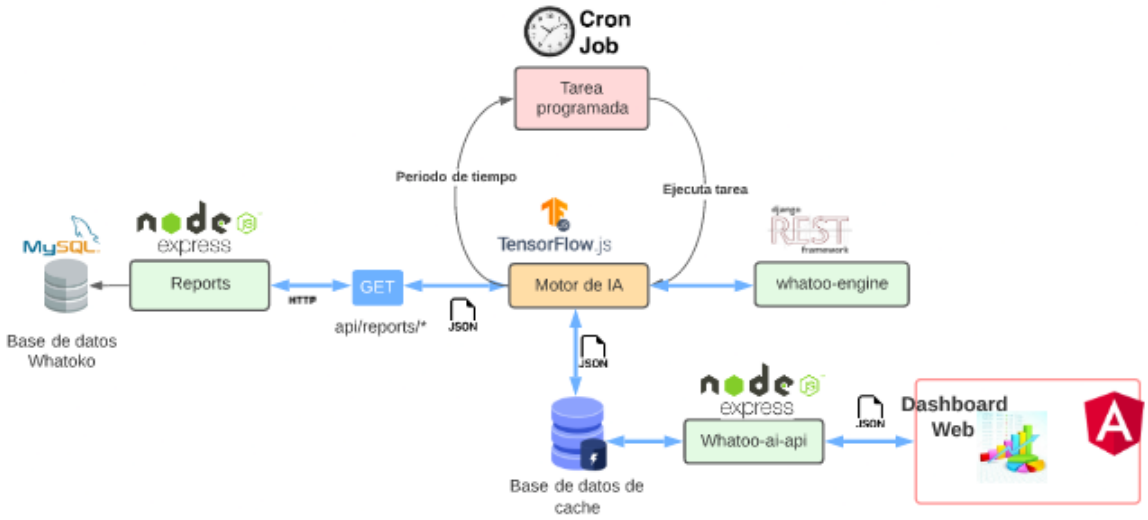
4.2.4.3.4 Despliegue de Modelos. Una vez obtenido el conjunto de datos para realizar predicciones, es claro que se necesita saber cuál es el objetivo de los datos. Para ello, este módulo con los modelos entrenados, los cuales son generados a partir del microservicio de entrenamiento y que se cargarán en el momento que se requiera realizar la tarea de predicción.

4.2.4.3.5 Generación de Predicciones. Este módulo se encarga de realizar el mapeo de las salidas(predicciones) del modelo entrenado.

4.2.4.3.6 Almacenamiento en Base de Datos de Cache. Este módulo se encarga de almacenar los resultados del análisis y predicción de datos como objetos clave-valor.

Teniendo en cuenta lo anterior, se realizó el siguiente diagrama que ilustra el funcionamiento del proyecto.

Figura 8
Flujo del funcionamiento del Motor de Inteligencia Artificial.



En el diagrama se ilustra el funcionamiento del motor de Inteligencia Artificial. Inicialmente se parte de una fuente de datos, que para este caso es el servicio de reportería, ya que este contiene la colección de consultas de mediciones e información del usuario. Posteriormente, desencadenará las estructuras JSON en las que vienen encapsulados los datos y se definirá un conjunto de entradas con las cuales un determinado modelo generará las predicciones. Estas predicciones se van a persistir en una base de datos de cache, de tal manera que la aplicación REST cargue dichos informes y los exponga mediante endpoints. Estos endpoints entregarán estos resultados en formato JSON para alimentar el Dashboard Web u otros servicios que tengan la función de generar alertas a los usuarios de Whatoko Health. Es necesario tener en cuenta que al ser un servicio que consume muchos recursos en el proceso, se debe implementar la lógica para tener un comportamiento asíncrono y se ejecute como una tarea programada. Es por ello por lo

que se implementó un Cronjob, el cual es un proceso que se ejecuta en segundo plano y tiene la función de ejecutar tareas de forma periódica en un tiempo específico.

4.2.5 Definición de Tecnologías

Para el desarrollo del proyecto, es necesario tener claro cuál es el objetivo. Con base en ese objetivo se definen las tecnologías que mejor se adaptan al proyecto, de tal manera que haya un equilibrio entre funcionalidad y rendimiento. A continuación, se explicarán las tecnologías que se utilizaron en el proyecto.

4.2.5.1 Base de Datos. La Base de datos es fundamental para este proyecto, dado que es la capa que se encarga de persistir los datos de forma organizada y estructurada. Para ello se usó MySQL, el cual es un sistema de gestión de bases de datos relacionales de código abierto. Entre las ventajas que ofrece están el ser gratuita, fácil de configurar, buen rendimiento para grandes volúmenes de datos, fácil integración con los distintos ambientes de desarrollo, compatibilidad con distintas plataformas como Linux, macOS, Windows, entre otros. Además de ello, se utilizó gitlab como herramienta para el versionamiento de los esquemas de base de datos.

4.2.5.2 Backend. El desarrollo Backend es la parte fundamental para el funcionamiento del proyecto. Teniendo como contexto la implementación de un motor de inteligencia artificial, a continuación, se especificarán las tecnologías que se usaron para las distintas capas del proyecto.

4.2.5.2.1 Node JS. Es el entorno de ejecución basado en Javascript para la capa asíncrona de alto rendimiento basado en el motor v8 de Google y es una de las plataformas más utilizadas en el desarrollo de microservicios REST. Ofrece un amplio catálogo de paquetes para el desarrollo Backend mediante su gestor de paquetes NPM (Node Packet Manager). Node ofrece alto

rendimiento en tiempo real. Esta tecnología se usará para el desarrollo de los microservicios mediante la librería Express, la cual es un ecosistema para el diseño de aplicaciones REST.

4.2.5.2.2 Tensorflow JS. Es la plataforma de código abierto para aprendizaje profundo más popular. Ofrece un ecosistema de herramientas y bibliotecas para la implementación y despliegue de modelos en el navegador. La plataforma adaptada para Node JS permite entrenar y desplegar modelos en la nube, ofreciendo un buen rendimiento acompañado de la programación asíncrona. Esta tecnología se usó para el despliegue de la inteligencia artificial de Whatoko usando express como se explicará más adelante.

4.2.5.2.3 Django. Es un ecosistema de aplicaciones Web basado en Python. Ofrece un desarrollo rápido y limpio, el cual cuentan con un potente ORM (Object Relational Mapping) para enlazar el esquema de base de datos relacional con objetos. Además de ello, ofrece la infraestructura de terceros para la implementación de servicios REST, Django Restframework. Esta tecnología se usó para diseñar el microservicio que se encarga de hacer el análisis exploratorio de la información cruda de la base de datos para posteriormente entrenar modelos de inteligencia artificial usando Tensorflow y keras.

4.2.5.2.4 Tensorflow. Tensorflow es una biblioteca de código abierto enfocada al desarrollo de modelos de aprendizaje profundo. Su ecosistema permite desarrollar la abstracción de redes neuronales para las distintas tareas de Inteligencia Artificial. Esta tecnología se complementó con el ecosistema de Django Restframework para entrenar modelos Deep Learning usando keras, los cuales mediante traductores a Tensorflow JS, se encargó de entregar modelos para despliegue.

4.2.5.3 Control de Versiones. El control de versiones de un proyecto es la capa esencial para tener una trazabilidad en cada valor agregado al producto. En el mercado existen distintas herramientas que comparten las funciones de gestionar los cambios en el código fuente a lo largo de tiempo. Sobre todo, si se quiere automatizar el ciclo de vida del desarrollo de software con Devops. Para ello se usó Gitlab, ya que cuenta con bastantes herramientas, de las cuales destaca la implementación de DevOps.

4.2.5.4 Despliegue. El proyecto se realizó a través de contenedores de Docker los cuales se encargan de encapsular toda la lógica y dependencias del servicio desarrollado. Esto se hizo a partir de Docker compose, el cual es un orquestador de contenedores que permite ejecutar distintas imágenes para dar solución a un servicio. Esto se logra gracias a que Docker compose permite gestionar los contenedores en una red. Para el proyecto se crearon los módulos de conexión a la base de datos el cual da servicio a través de un API Gateway desarrollado con nodejs que permite gestionar los recursos, los cuales son consumidos por el módulo de entrenamiento de modelos y el motor de inteligencia artificial. La facilidad que tiene docker para poder simplificar el proceso de despliegue hace que los proyectos sean fáciles de gestionar y escalar con diferentes servicios, permitiendo enfocarse en la lógica del negocio.

4.2.5.5 Documentación. La documentación es importante para el proyecto, ya que en ella se realiza el intercambio de conocimientos al equipo de desarrollo. Esto consiste en describir brevemente cómo funcionan los procesos que se han implementado en cada capa de software, de tal manera que cualquier miembro que se sume al proyecto en un momento determinado pueda capacitarse de una forma más sencilla. Para ello se utilizó Compodoc, la cual es una herramienta para la documentación de código. Esta herramienta consiste en generar un documento HTML a

partir de las líneas documentadas en el código. En este documento se específica la descripción de cada función, variable o clase definida en el código.

4.3 Desarrollo del Proyecto

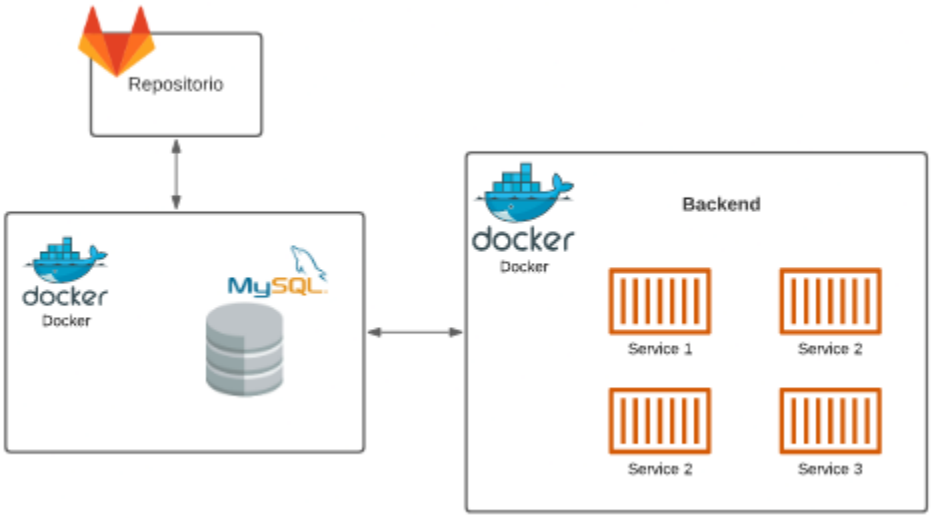
En la práctica se desarrolló el proyecto siguiendo SCRUM como metodología ágile. Para ello, con el Product Owner se planteó el Product Backlog, el cual es la lista de tareas ordenadas por prioridades que se obtiene a partir de los requerimientos del producto. A continuación, se explicará el desarrollo, prueba y despliegue de cada incremento.

4.3.1 Base de Datos

La base de datos fue el punto de partida para la etapa de desarrollo, ya que a partir de ella se establecieron las funcionalidades de cada Microservicio, además de ello, gestionar las mediciones que entregan los dispositivos Wereables. Este proceso se llevó a cabo utilizando LucidChart como herramienta para diseñar el diagrama del primer modelo entidad relación. La definición de este diagrama fue un proceso iterativo siguiendo la metodología SCRUM, donde en cada iteración se hacía revisión por parte del líder técnico, el cual aportó realimentado el modelo hasta tener una versión final para desarrollo. En la etapa de implementación se utilizó Workbench como herramienta para generar el primer esquema en MySQL. A partir de este esquema se realizó el proceso de despliegue, el cual consistió en hacer la configuración para los diferentes ambientes del proyecto. A continuación, se explicará la configuración en cada ambiente.

4.3.1.1 Ambiente Local. En el ambiente local, se usó un contenedor para automatizar la inicialización de la base de datos en cada Máquina anfitrión. Esto se realizó a través de Docker-compose.

Figura 9
Configuración de Base de Datos en Ambiente Local



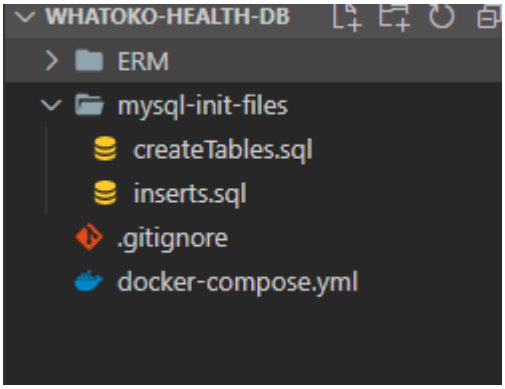
En el diagrama se puede visualizar la orquestación de la base de datos en el ambiente local. Como se puede observar, el control de versiones del esquema se hizo a través de un repositorio en Gitlab. En el momento del despliegue, docker-compose lanza el contenedor de base de datos creando la estructura de las tablas e inicializando la inserción de datos de prueba. Posteriormente, crea la red principal de Whatoko Health para exponerla a los microservicios de Backend. Teniendo esto en cuenta, se explicarán los directorios que conforman este ambiente.

4.3.1.1.1 ERM. En este directorio se almacenan las diferentes versiones del modelo entidad relación generado a partir de Workbench. Con cada esquema, se genera el script para inicializar la base de datos.

4.3.1.1.2 Mysql-init-files. En este directorio se almacenan los archivos ejecutables para inicializar la creación de la base de datos e insertar el conjunto de datos de prueba. Este proceso está automatizado a realizarse cada vez que se orquesten los contenedores.

4.3.1.1.3 Docker-compose. Es archivo .yml se encarga de orquestar el contenedor de MySQL inicializando la creación del esquema de base de datos, la inserción de datos de prueba y la exposición a los microservicios en la red de Whatoko. En la *Figura 10* se puede visualizar la estructura de carpetas.

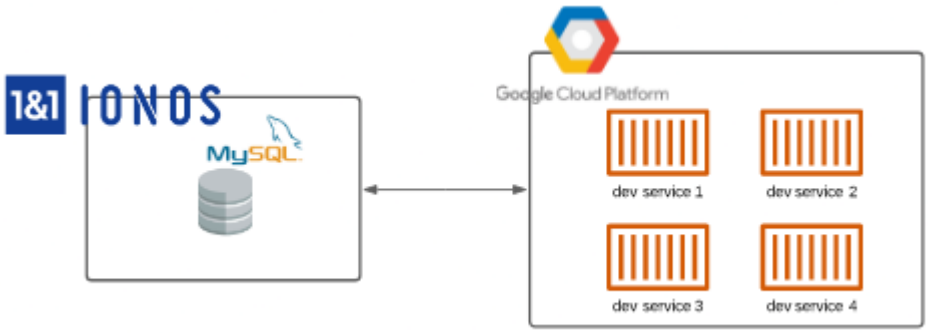
Figura 10
Directorios del Repositorio de Base de Datos



4.3.1.2 Ambiente de Desarrollo.

Para el ambiente de desarrollo, el esquema de base de datos se alojó en los servicios de gestión de base de datos en IONOS. Para ello se instanció un host que se encarga de inicializar la base de datos. Para realizar la conexión remota se utilizaron las credenciales de conexión de dicha instancia. En esta se proporciona el Host, el nombre de la base de datos, el usuario administrador y la contraseña.

Figura 11
Configuración de Base de Datos en Ambiente de Desarrollo



4.3.2 Modulo de Reportes

Como se mencionó en la etapa de planeación, el módulo de reportería tiene el objetivo de tomar los datos almacenados en la base de datos para posteriormente entregar información preprocesada al dashboard y el motor de inteligencia artificial.

4.3.2.1 Definición de Consultas. Antes de comenzar el desarrollo, es importante definir el qué hará y cómo lo hará. Para ello se agrupó la lista de consultas que entrega la información de la siguiente manera:

- Entregar informes al dashboard Web
- Entregar datos y mediciones al motor de inteligencia artificial
- Entregar datos y mediciones individuales de cada Usuario registrado en Whatoko Health para consumo en aplicación móvil.
- Entregar datos y mediciones al módulo de entrenamiento de modelos.

4.3.2.2 Desarrollo del Microservicio. El desarrollo de este microservicio se realizó utilizando Typescript como lenguaje de programación. Este desarrollo se compone de 3 capas.

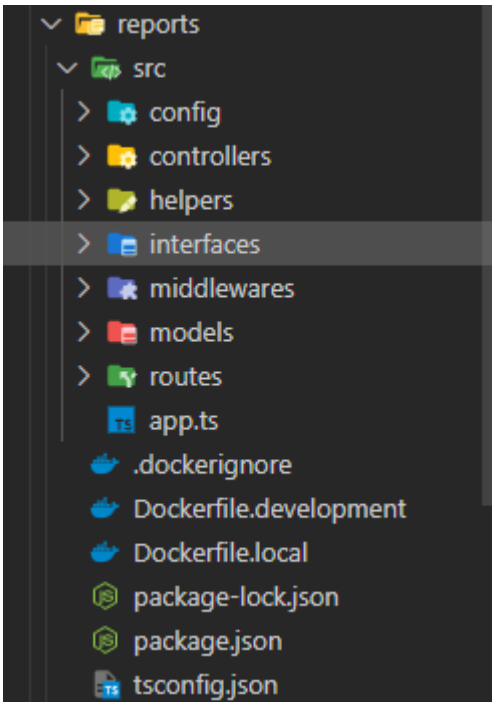
4.3.2.2.1 Servidor API REST. La capa de servidor se realizó con la librería Express, la cual es una biblioteca de Node Js que proporciona las herramientas para diseñar la abstracción de una API REST. En este proceso se definió la ruta principal del microservicio, la lista de controladores, la lista de rutas, el puerto y host en el que el servidor escuchará las peticiones.

4.3.2.2.2 Modelos ORM. Los modelos ORM son la capa encargada de realizar el mapeo de las entidades(tablas) a clases. Para ello se utilizó la librería Sequelize, la cual es una heramienta que proporciona las funciones para diseñar las clases homólogas a cada tabla de la base de datos.

4.3.2.2.3 Pruebas. Esta capa es la encargada de realizar pruebas en cada ruta implementada. Para ello se utilizó POSTMAN, la cual es una herramienta que tiene la función de probar el flujo de información de aplicaciones REST.

4.3.2.3 Estructura de Carpetas. La estructura del proyecto se basó en el patrón de diseño Modelo Vista Controlador (Universidad de Alicante). A continuación, se explicará la funcionalidad de cada directorio.

Figura 12
Estructura de carpetas de reportería.



4.3.2.3.1 Archivos Dockerfile. Estos archivos de extensión. yml contienen el lenguaje de construcción del contenedor de Docker. Ambos archivos tienen la tarea de instalar las dependencias necesarias para desplegar en cualquier máquina anfitrión. Cada Dockerfile se diferencia según el ambiente en el que se ejecuta. Para el ambiente local se despliega a partir de la orquestación de contenedores con Docker-compose, en el cual se le asigna la red, el host y el puerto expuesto para comunicarse con los demás servicios. El ambiente de desarrollo se despliega en Google Cloud con GKE, cargando la imagen en Gitlab-Registry y orquestando el servicio usando kubernetes.

4.3.2.3.2 Config. La carpeta config contiene la configuración de la base de datos a partir de la biblioteca de manejo de base de datos con Objetos ORM (Object Relational Mapping)

Sequelize. En ella se realiza la asignación de variables de entorno que contienen las cadenas de conexión a las bases de datos. La conexión al ambiente local o desarrollo se valida a partir del entorno de ejecución. Posteriormente se instancia un objeto de Sequelize utilizando el patrón de diseño Singleton, donde automáticamente se iniciará una sola vez en el momento que se despliega el contenedor.

4.3.2.3.3 Controllers. La carpeta controllers contiene la lógica de consulta a la base de datos y el mapeo esta para entregarla al cliente. Contiene 10 archivos de Typescript, donde cada contiene una Clase con sus métodos de consulta y respuesta según el tipo de información. Por ejemplo: mediciones de actividad, mediciones de colesterol, mediciones de glucosa, mediciones de ritmo cardíaco, mediciones de sueño, mediciones de tensión, datos detallados de los usuarios paciente, entrenamientos y diagnósticos. Estos métodos utilizan el protocolo GET, los cuales mediante las clases ORM definidas por Sequelize, realizan las consultas SQL abstrayendo el lenguaje DML (Lenguaje de manipulación de datos) y retornando objetos JSON estructurados por cada entidad.

Figura 13
Lista de controladores.

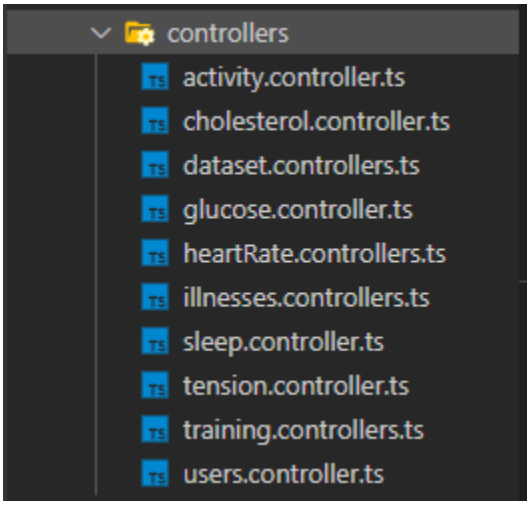


Figura 14
Ejemplo de consulta con Sequelize

```
public async getGlucoseByUser(request: Request, response: Response): Promise<Response> {
    try {
        /**Id del usuario  */
        const { id } = request.params;
        /**Parámetros opcionales */
        const { start_date = '2001-01-01', end_date = '2040-01-01', page = 0, limit = 1000,
        glucose_start = 0, glucose_end = 500 } = request.query;

        /**Consulta */
        const data = await Measurement.findAll({
            where: {
                userId: id,
                'date': { [Op.between]: [start_date.toString(), end_date.toString()] }
            },
            attributes: [ 'date'],
            include: [
                {
                    model: DataMeassurement,
                    required: true,
                    attributes: ['value'],
                    include: [{ model: ValueType, attributes: ['name'] }],
                    where: {
                        value: {
                            [Op.between]: [glucose_start, glucose_end]
                        }
                    }
                },{
                    model: MeasurementCategories,
                    required: true,
                    attributes: ['name'],
                    where: { measurementTypeId: 5 }
                },{
                    model: User,
                    required: true,
                    attributes: []
                },{
                    model: MeasurementDetail,
                    attributes: ['id'],
                    include: [{ model: Option, attributes: ['name'] }]
```

4.3.2.3.4 Models. La carpeta models contiene todos los objetos de sequelize que mapean las entidades del esquema de base de datos. Estos cuentan con la ventaja de tener una manipulación de datos más rápida y sencilla, además de poder definir atributos con base en los campos de la tabla origen. Por ejemplo, obtener la edad a partir del campo de fecha de nacimiento mediante un

método Get. Además de eso, se tiene el modelo servidor, el cual es la abstracción de una instancia de Express, en el cual se realiza la configuración de rutas, la configuración de middlewares, la definición de la ruta principal de la API, la inicialización de la base de datos y el puerto en el que el servidor escuchará las peticiones. Este servidor se instancia usando el patrón de diseño Singleton, el cual se ejecuta en el archivo principal.

4.3.2.3.5 Middlewares. La carpeta middlewares contiene las funciones middleware que se encargan de validar la comunicación intermedia entre el cliente y el servidor. Esto con la finalidad de validar los datos o parámetros de consulta que se envían a los endpoints. Para ello se apoya en las funciones de validación de base de datos que se encuentra en helpers.

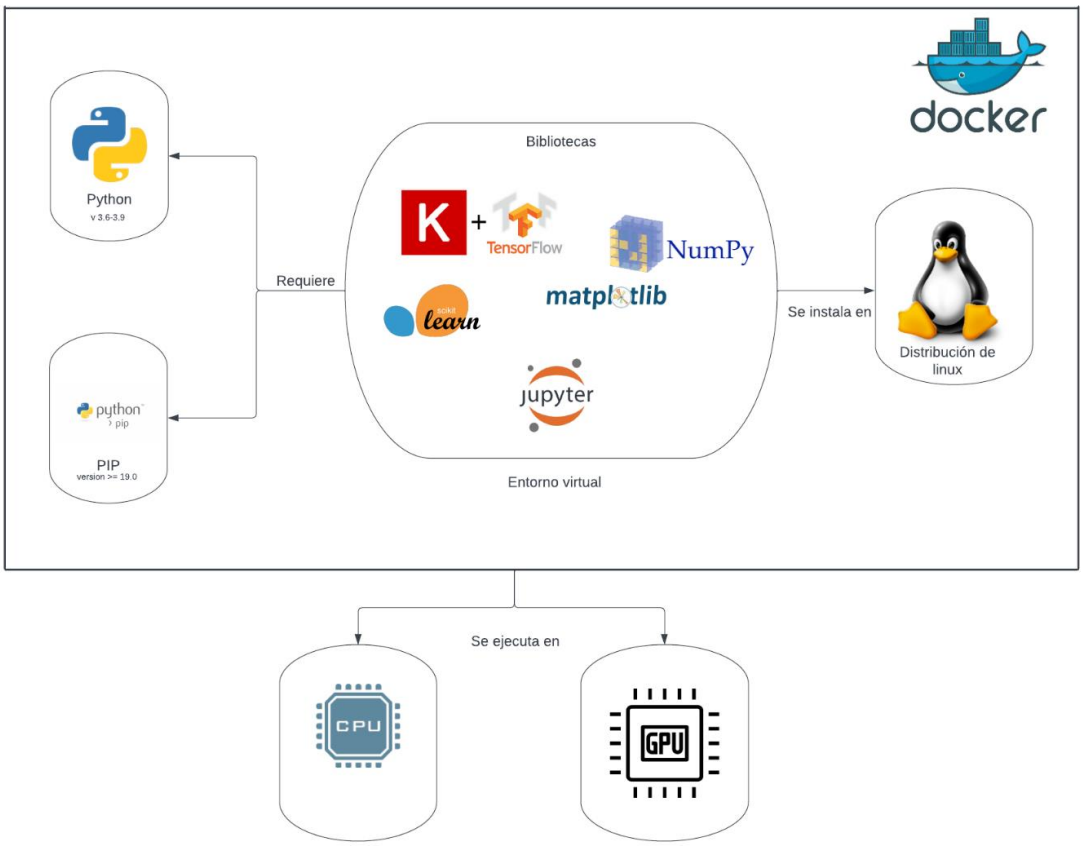
4.3.2.3.6 Routes. La carpeta routes contiene el archivo de enrutamiento de Express, el cual instancia un objeto router que mediante los métodos HTTP, enruta los controladores con el servidor. Cada ruta contiene una serie de middlewares que validan los parámetros de consulta. Por ejemplo, que la fecha de consulta sea válida. El usuario tenga una sesión activa validando el JWT(Json Web Token) generado a partir del inicio de sesión exitoso. Que el usuario consultado exista.

4.3.3 Servicio de Entrenamiento de Modelos

El microservicio de entrenamientos es la capa esencial para la generación de modelos de keras con Tensorflow. Para ello, se diseñó una aplicación REST usando Python con Django Restframework. Las funciones que realiza este servicio son definir conjuntos de datos de entrenamiento a partir de la información almacenada en base de datos. Estos datos son utilizados para entrenar los modelos propuestos en la fase de planeación utilizando aprendizaje profundo, los cuales finalmente son entregados a despliegue utilizando el protocolo HTTP.

4.3.3.1 Configuración de entorno de Tensorflow. En este entorno se implementaron todas las funciones para el proceso de modelado, entrenamiento y análisis de Resultados. Para ello, se utilizó la librería Tensorflow, con la cual se definieron las estructuras de redes neuronales con Keras. La interacción se realiza a través de notebooks utilizando Jupyter Notebook. Este entorno se configuró para que funcionara solamente en ambiente local. Para el despliegue se utilizó Docker.

Figura 15
Configuración de entorno local de Tensorflow



Como se puede observar en el diagrama, este entorno se ejecuta en una distribución de Linux compartiendo el kernell y los recursos de la máquina anfitrión. Las bibliotecas requieren de

una versión de Python con su gestor de paquetes PIP para poder ejecutarse en el contenedor. Teniendo en cuenta esto, se utilizó una imagen oficial de Tensorflow que contiene configurado el despliegue de la interacción con Jupyter Notebook.

Figura 16
Ejecución del contenedor de Tensorflow.

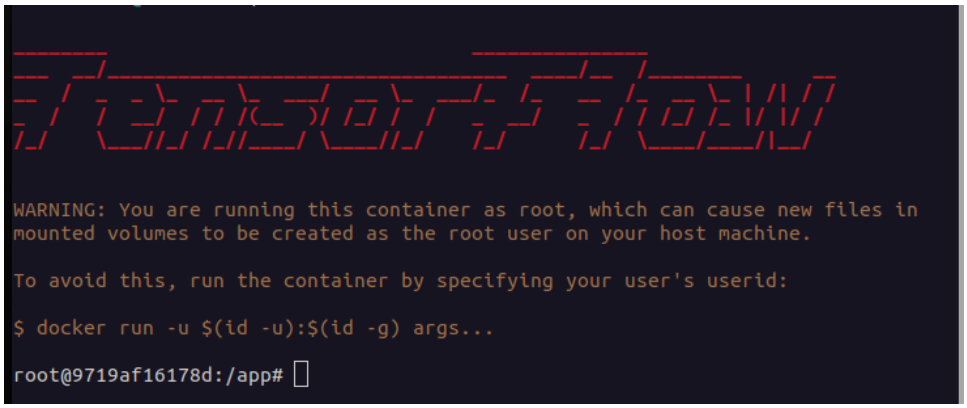
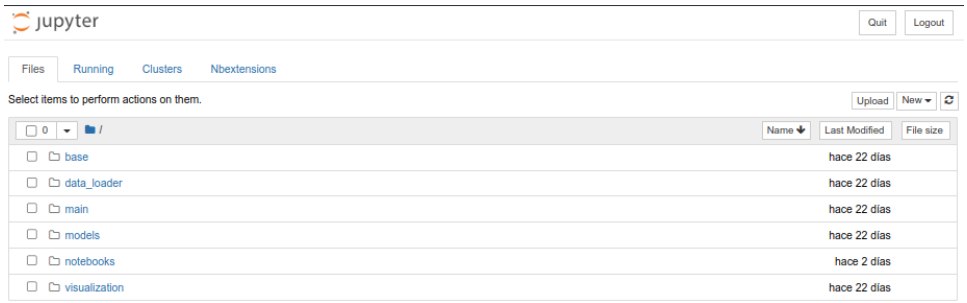


Figura 17
Entorno de Jupyter Notebook.



4.3.3.2 Configuración de API REST. Este servicio es el encargado de definir la conexión a la fuente de datos y procesarlos hasta entregarlos al entorno de tensorflow. Además de ello, contiene los controladores que se encargan de cargar los modelos entrenados y entregarlos al Motor

de inteligencia artificial cuando este lo solicite. A continuación, se explicará el desarrollo de este servidor.

Figura 18
Estructura del Microservicio de Inteligencia Artificial

En el siguiente diagrama se observa la comunicación que tendrá el servidor. Inicialmente realizará peticiones de datos los cuales son traídos como información cruda en estructuras JSON. Dichas estructuras son mapeadas utilizando objetos instanciados de clases en Python. Posteriormente entregará la información preprocesada en conjuntos de datos que llegan al entorno de Tensorflow, donde finalmente se realizan las tareas de análisis de datos, etiquetación de conjuntos de datos y etapa de entrenamiento. A continuación, se explicará con detalle las capas que componen este servicio para alimentar el entorno de tensorflow y entregar modelos a despliegue.

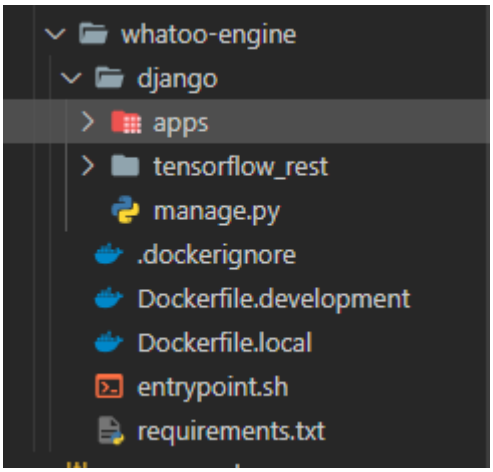
4.3.3.2.1 Preprocesamiento de Datos. Esta fase es importante para el manejo de los datos, ya que es necesario definir la metodología para manipular las estructuras JSON que almacenan las diferentes entidades con sus respectivos datos. Para dicha tarea, se usó el paradigma de programación orientada a objetos como capa encargada de realizar el mapeo de las respuestas JSON a instancias de clases, de tal manera que a partir de métodos get, realice el preprocesamiento de estos datos y retorne un valor cuantitativo o cualitativo.

4.3.3.2.2 Entrega de Modelos a Despliegue. En esta capa se definieron los controladores que realizan la tarea de cargar el modelo, transformarlo a formato JSON para lectura en Tensoflow JS y entregarlo mediante un endpoint.

4.3.3.2.3 Reentrenamiento de Modelos. Esta capa contiene los controladores para realizar el rentrenamiento de Modelos con nuevos datos.

4.3.3.3 Estructura de directorios del Servidor. En esta sección se explicará la estructura de los directorios del proyecto. Para ello se utilizó la estructura generada por Django la cual se basa en el patrón de diseño modelo-vista-controlador, aunque con ciertas modificaciones para integrar las funciones del entorno de Tensorflow.

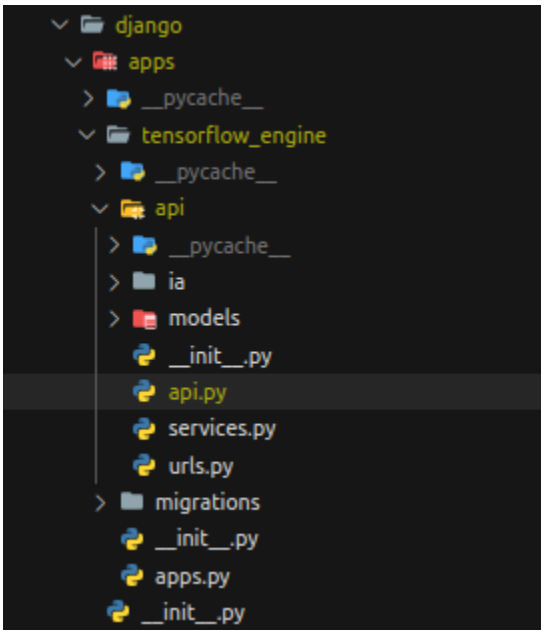
Figura 19
Estructura de carpetas del servicio de entrenamientos.



4.3.3.3.1 Django. Contiene toda la lógica del funcionamiento del servicio de Django. En ella se encuentra la carpeta apps, la cual cuenta con las aplicaciones encargadas de realizar funcionalidades de preprocesar datos, definir conjuntos de datos, modelar estructuras de aprendizaje profundo.

4.3.3.3.2 Apps. En esta carpeta se almacena la lógica desarrollada con Django, que contiene la lista de módulos del microservicio, implementados en la carpeta tensorflow_engine, la cual tiene la colección endpoints y controladores que realizan las tareas de preprocesamiento y entrenamiento de datos. Todo este funcionamiento se almacena en la carpeta api. Los módulos con los que cuenta este servicio son, el módulo de mapeo de datos y de inteligencia artificial, cuyo uso es para el entrenamiento y despliegue de modelos en tensorflow.

Figura 20
Estructura del motor de inteligencia artificial

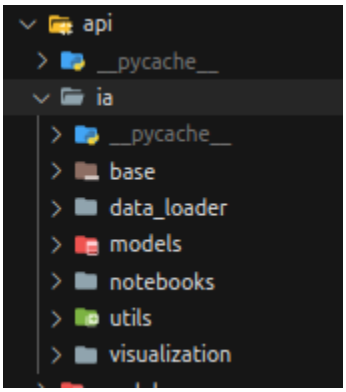


4.3.3.3.3 Archivos Dockerfile. Los dockerfiles contiene la lógica para desplegar el contenedor. Cada dockerfile se diferencia según el ambiente de ejecución.

4.3.3.4 Estructura de directorios del Entorno de Tensorflow. En esta sección se explicará la estructura de los directorios del proyecto. Para ello se utilizó la estructura generada

por Django la cual se basa en el patrón de diseño modelo-vista-controlador, aunque con ciertas modificaciones para integrar las funciones del entorno de tensorflow.

Figura 21
Estructura de carpetas del módulo de inteligencia artificial.



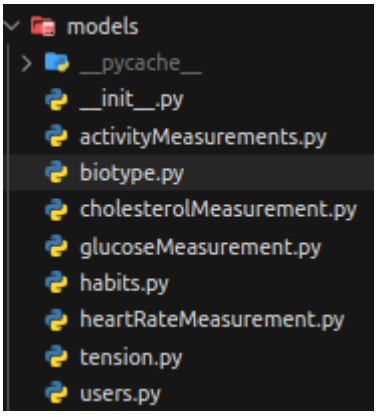
4.3.3.4.1 Base. El directorio base contiene la lógica que define la estructura de los modelos, la configuración del entrenamiento y la generación de estos en un formato de despliegue. Por ejemplo, la función base para una red neuronal por defecto define las capas de entrada y salida. El uso de esta función se encarga de parametrizar el número de capas, el número de neuronas por capa oculta, la función de costo y la función de la capa de salida, de tal manera que sea reutilizable en diferentes entrenamientos. Para la configuración del entorno de entrenamiento de una red neuronal se necesita definir cuáles serán los hiperparámetros que usarán para entrenar y ajustarse a los datos. Este entorno contiene las funciones que definen qué métrica se utilizará para medir el modelo, cuál será la función de costo que encontrará los mejores pesos de entrenamiento, cómo se manejará el costo computacional en cada iteración, qué tipo de algoritmo de optimización se usará para ajustar los pesos en el back propagation, y cuáles serán las características de entrada y la columna objetivo. Para la generación de modelos en ambiente de despliegue, se tienen las

funciones para generar la arquitectura del modelo y los pesos que se usarán para lanzar a producción.

4.3.3.4.2 Data_loader. La carpeta data_loader contiene la clase encargada de preparar los datos de entrenamiento. Como insumo recibe el resultado del mapeado de datos a partir de las peticiones al microservicio de reportes. Por ejemplo, para definir los datos de entrenamiento que se usarán para predecir el riesgo de diabetes, esta clase contiene el método con el cual selecciona las variables predictoras a partir de los atributos de cada objeto. Luego de esto, calcula la columna objetivo que será la finalidad del modelo para entrenar el patrón buscado. Finalmente, retorna el conjunto de datos preprocesados para entrenar.

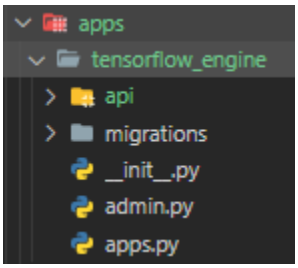
4.3.3.4.3 Models. La carpeta models almacenará los archivos generados a partir del entrenamiento. Estos archivos son utilizados para desplegar el modelo en producción, los cuales serán cargados y entregados mediante endpoints.

Figura 22
Clases encargadas de mapear los datos.



4.3.3.4.4 Visualization. El directorio visualization contiene las funciones para visualizar los resultados de entrenamiento, de tal manera que se pueda realizar un análisis de la función de costo y la varianza de los resultados en cada iteración reciclando las gráficas de análisis.

Figura 23
Directorios bases del desarrollo del motor de inteligencia artificial.



4.3.4 Motor de Inteligencia Artificial

Como se mencionó en la fase de planeación, el motor de inteligencia artificial es la capa encargada de realizar tareas de predicción para detectar anomalías o riesgos en los usuarios de Whatoko Health. Dicho servicio entrega informes de resultados al Dashboard Web, los cuales se generan de forma iterativa en un periodo de tiempo. En esta sección se explicará el desarrollo, prueba y despliegue del motor con el que se cumplió el quinto Objetivo específico.

4.3.4.1 Desarrollo del Motor. El desarrollo de este servicio se llevó a cabo utilizando Typescript como lenguaje de programación principal. Este desarrollo se compone de la siguiente manera:

4.3.4.1.1 Despliegue de Inteligencia Artificial. En esta capa se realizó el despliegue de modelos de aprendizaje profundo entrenados en Keras. Para ello se utilizó la librería Tensorflow JS, con la cual se realiza el procedimiento de cargar los modelos en formato JSON y definir las funciones para generar predicciones.

4.3.4.1.2 Api Rest. En esta capa se realizó el despliegue del servidor REST el cual tiene la tarea de cargar los informes de resultados de predicción y entregarlos a los demás servicios. Para ello se utilizó la librería Express con la cual se definió la arquitectura del modelo servidor, definiendo los controladores, rutas de petición y la ruta principal.

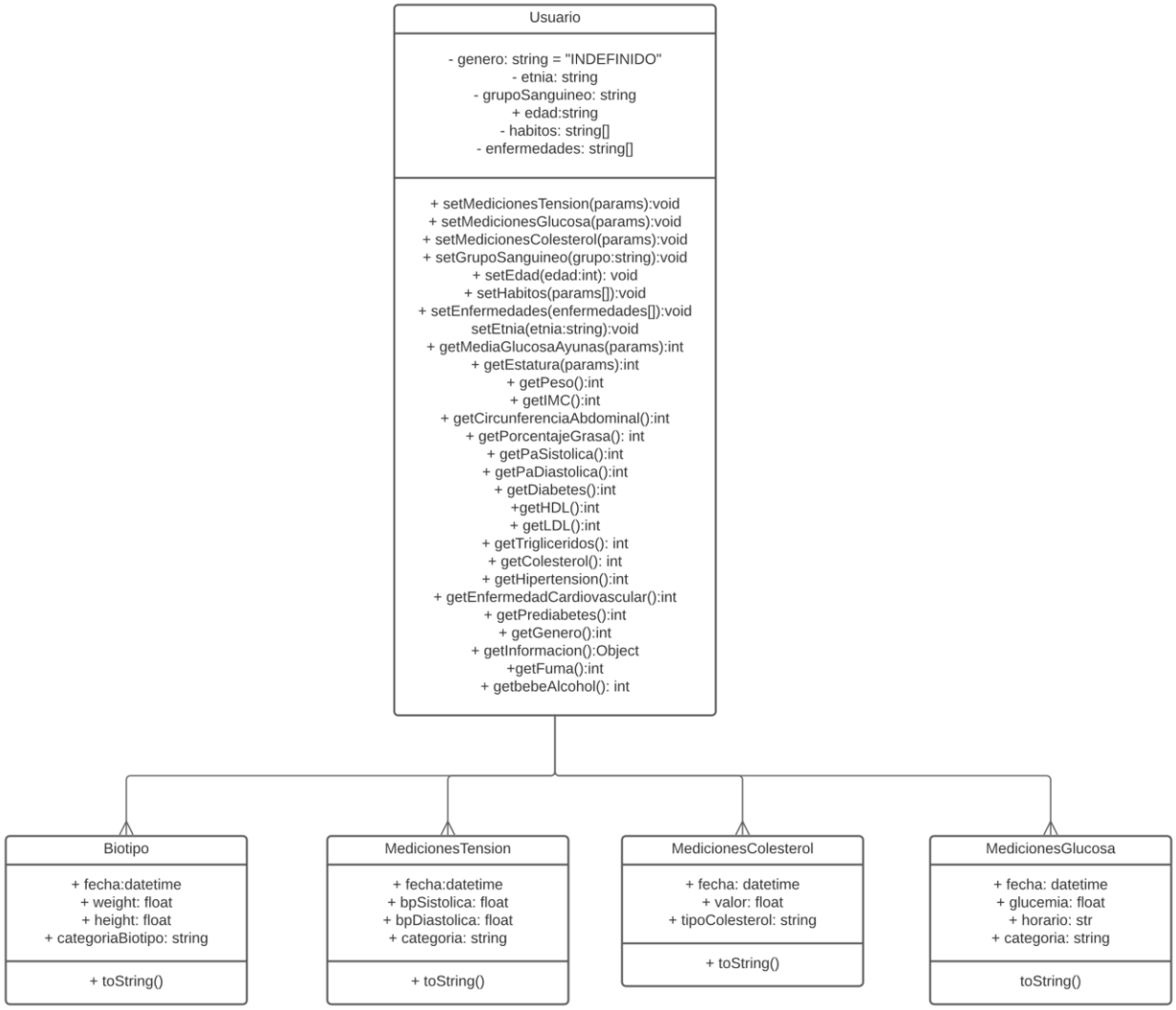
4.3.4.1.3 Tareas Programadas. Esta capa es la encargada de gestionar las tareas que realizará el motor de forma periódica. Para ello se utilizó un Cronjob, el cual se implementó con Node-cron. Esta librería permite configurar un servicio que se ejecutará en segundo plano para ejecutar las tareas del modelo en el tiempo que se le defina.

4.3.4.2 Preprocesamiento de Datos. Para que el motor de inteligencia artificial pueda realizar las tareas de análisis y predicción, es necesario tener datos estructurados y limpios. En primer lugar, cuando se realiza la petición a la fuente de datos, estos se reciben estructurados en objetos JSON. La desventaja de esto es que manipular los datos contenidos puede llevar tiempo, por lo cual se requiere de un método rápido y sencillo para desestructurar el contenido y definir un conjunto de datos fácil de manipular. Para ello se utilizó el paradigma de programación orientada a objetos como capa encargada de realizar dicha desestructuración. La ventaja que ofrece es que permite definir atributos que representan información de un conjunto de datos crudos, lo cual facilita la minería de datos y la definición de diversos conjuntos de entradas a partir de un conjunto de instancias. La capa de preprocesamiento de datos se compone de 3 partes. El objetivo de esta capa es poder traer datos de los usuarios con sus mediciones y definir varios conjuntos de datos para predicción. Cada conjunto de datos alimentará una tarea específica de un modelo entrenado. A continuación, se explicará cómo se compone esta capa.

4.3.4.2.1 Petición de Datos. En esta fase se definió un módulo para realizar peticiones a la fuente de datos. Para ello se utilizó una clase encargada de abstraer la conexión a la fuente de datos realizando varias peticiones HTTP. Cada petición es una consulta distinta, la cual se unificará con las demás en una sola instancia que finalmente se almacena en un arreglo.

4.3.4.2.2 Mapeo de Datos. En esta fase se definió la lógica que permite reunificar las distintas consultas en una instancia de una clase. Para ello se utilizaron varias clases que contienen los atributos análogos a los datos encapsulados en las estructuras JSON. Teniendo como base los modelos propuestos en la fase de planeación se diseñaron las clases encargadas de mapear los datos necesarios. En el siguiente diagrama UML se puede observar cómo se definieron estas clases.

Figura 24
Modelo desestructurador de objetos json.

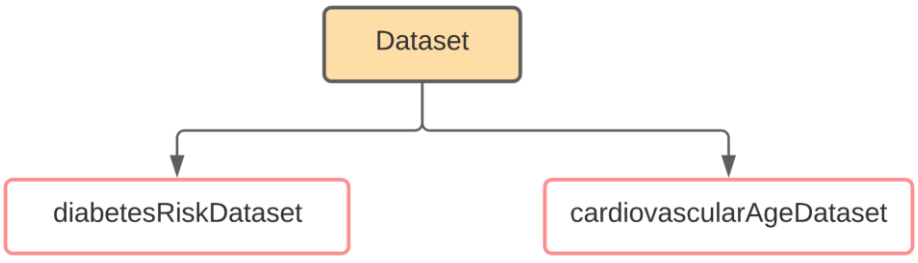


En el diagrama UML, se observan las clases encargadas de realizar la desestructuración de los objetos JSON. Los datos de cada usuario con sus mediciones estarán reunificados en la clase Usuario. Las mediciones se insertarán mediante los métodos Set que contiene la clase. El preprocesamiento se realiza a partir de los métodos Get. Por ejemplo, para la lista de mediciones de glucosa, estás se almacenan con su fecha, valor de glucemia, categoría y horario de la muestra.

Si se requiere obtener la media de glucosa de los últimos 7 días en el horario de Ayunas, se calculará y retornará mediante un método Get. A partir de estos atributos se definen diferentes conjuntos de datos con diferentes variables.

4.3.4.2.3 Definición de Conjunto de Entradas. Definida la estructura de los datos en instancias de Clase, el paso siguiente es definir los conjuntos de entradas. Cada conjunto de entradas contiene las variables predictoras para alimentar un modelo entrenado. Por ejemplo, para realizar la predicción de riesgo de diabetes tipo 2, se requieren las variables de Género, nivel de triglicéridos, media de glucosa en ayunas, si tiene hipertensión, etc. Esta definición de variables se realiza a partir de una Clase, la cual en sus métodos contiene la lógica para definir las entradas para un modelo específico. En la siguiente imagen se observa que la clase Dataset contiene 2 métodos. Estos 2 métodos retornan el conjunto de datos para predecir el riesgo de diabetes, así como también la predicción de la edad cardiovascular.

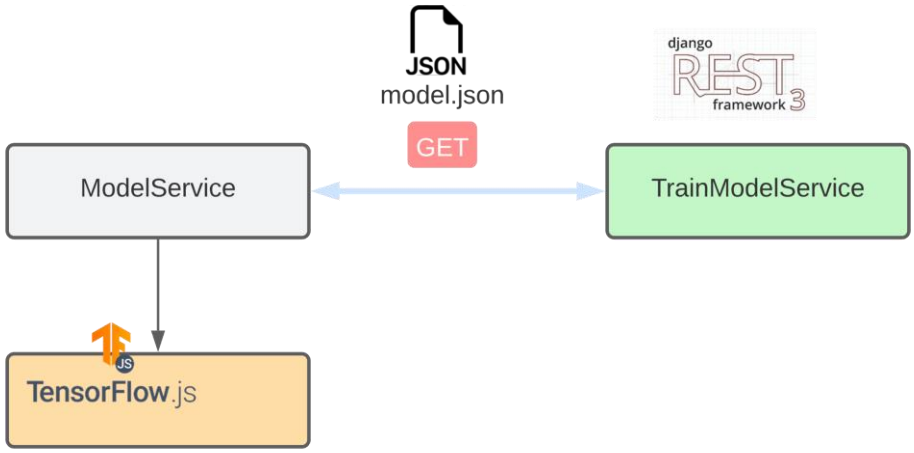
Figura 25
Estructurado de datos.



4.3.4.3 Despliegue de Modelos de Inteligencia Artificial. En esta capa se gestionaron los diferentes modelos de inteligencia artificial que conforman el motor de Whatoko Health. Estos modelos entrenados se generaron a partir del microservicio de entrenamiento con Keras., el

resultado del entrenamiento generó dos archivos para desplegar en Tensorflow JS. El consumo de estos modelos se realizó conectando via HTTP al microservicio de entrenamiento, el cual contiene los endpoints para entregar estos modelos en un cuerpo JSON. En la imagen se puede observar el proceso de despliegue de modelos. La clase ModelService realiza la petición via HTTP al microservicio de entrenamiento de modelos. Esta carga el modelo entrenado y lo traduce a formato de javascript el cual es entregado en formato JSON. Finalmente, este modelo es almacenado en local para que el motor de inteligencia artificial lo ejecute según la necesidad.

Figura 26
Resultado de predicción de datos desde la base de datos.



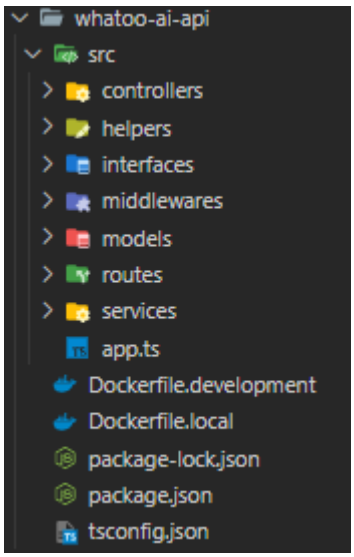
4.3.4.4 Ejecución por Tareas Programadas. Las tareas programadas son ejecuciones que realiza el entorno cada cierto tiempo, esto beneficia en no saturar los recursos de la máquina, dado que solo se ejecutará una sola vez. El comportamiento se definió mediante una clase que se encarga de gestionar todas las tareas del motor de inteligencia artificial. Esto se ejecuta en segundo plano por medio de un cronjob, en el cual se definió la fecha de ejecución para todos los domingos a las 12 de la noche. Una vez llega el momento de ejecutarse, el motor realiza las siguientes tareas.

- Petición a la fuente de datos
- Mapeo y preprocesamiento de datos
- Definición de conjunto de entradas
- Despliegue de modelos a partir de las entradas.
- Generación de predicciones
- Almacenamiento de las predicciones mapeadas en base de datos de cache.

Una vez realizado el procedimiento anterior, los informes quedan listos con su fecha de generación para que el dashboard Web proceda a consumirlos.

4.3.4.5 Estructura de Carpetas. La estructura de directorios se realizó utilizando el patrón de diseño Modelo Vista Controlador. A continuación, se explicará la estructura de carpetas del proyecto.

Figura 27
Estructura de carpetas del microservicio de los modelos IA.



4.3.3.5.1 Controllers. El directorio controllers contiene la lógica para desplegar los modelos de inteligencia artificial y entregar predicciones. En este directorio se tienen dos archivos TypeScript, los cuales cada uno es una clase que contiene métodos de tipo controlador. La clase ModelController contiene los controladores que se encargan de hacer peticiones al microservicio de Inteligencia Artificial (Whatoo-engine). Posteriormente, se entregan los modelos generados para almacenarlos en un directorio que servirá de ruta para desplegar y realizar predicciones cuando el motor se ejecute. La clase PredictionsController contiene la lógica para cargar los objetos clave-valor que son el resultado del análisis del motor, dichos objetos son entregados a través de estructuras JSON.

4.3.3.5.2 Helpers. El directorio helpers contiene lógica para definir el conjunto de datos, normalizar el conjunto de datos y mapear el resultado de las predicciones. Esto se realiza mediante el uso de clases. Además de ello, también contiene el directorio que almacena los modelos que se cargan en despliegue.

4.3.3.5.3 Database. En este directorio se realizó la configuración de la base de datos de cache, donde además se persisten los resultados en objetos JSON.

4.3.3.5.4 Middlewares. El directorio middlewares contiene las funciones que se encargan de validar la sesión activa del usuario con JWT. Esto permite que sólo los usuarios con una sesión activa pueden acceder a estas rutas.

4.3.3.5.5 Models. El directorio models contiene las clases e interfaces que se encargan de mapear las respuestas JSON del microservicio de reportes.

4.3.3.5.6 Routes. El directorio routes contiene la colección de rutas que exponen los resultados de predicción a los servicios externos. El funcionamiento se basa en utilizar el patrón

de diseño singleton, donde se instancia un objeto router que contiene los métodos de cualquier protocolo de la petición HTTP. En cada ruta se establece el nombre, la lista de middlewares de validación y el controlador que ejecuta la consulta. Cada ruta está protegida con el uso de JWT.

4.3.3.5.7 Services. Este directorio contiene las clases encargadas de abstraer la conexión con los servicios de entrenamiento de modelos y reportería. Además, también cuenta con la clase encargada de configurar la aplicación REST para entregar informes al dashboard.

4.4 Fase de Pruebas

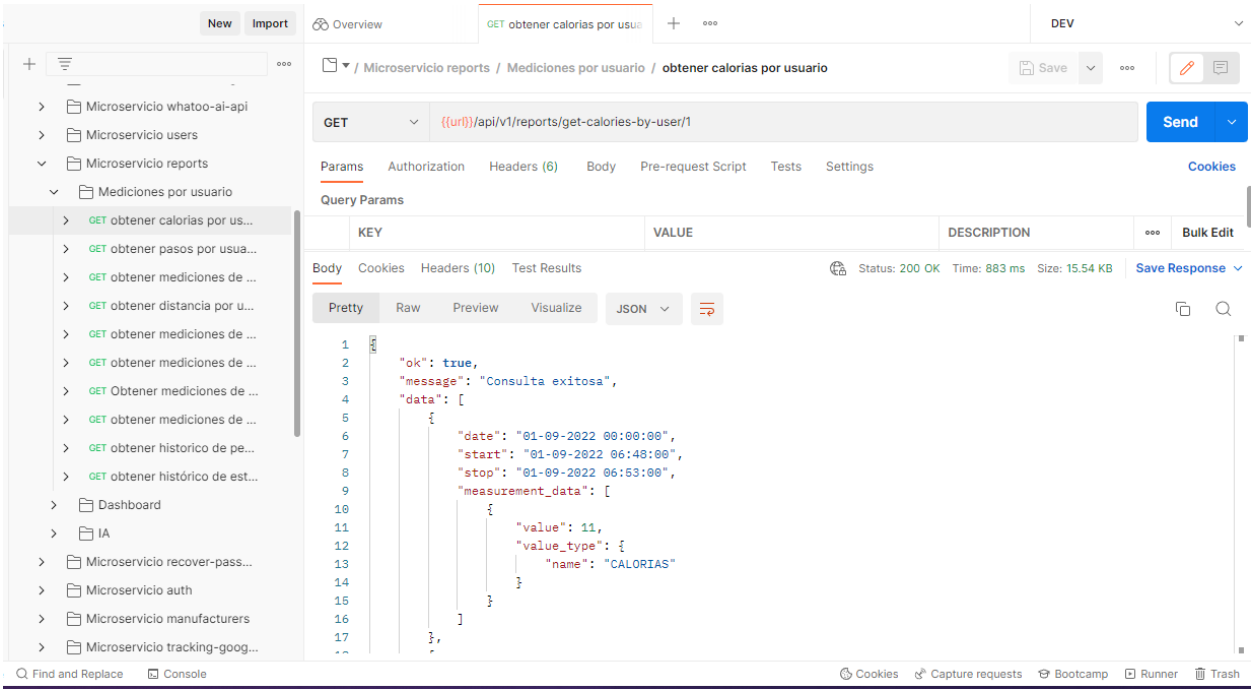
En esta sección, se realizaron las pruebas correspondientes al entorno de análisis de datos. Para ello se utilizó POSTMAN como herramienta de prueba en cada microservicio.

4.4.1 Microservicio de Reportería

El desarrollo final de este microservicio fue la colección de rutas que entregan datos de usuarios, mediciones por usuario e informes al dashboard. Con el uso de POSTMAN se verificó el funcionamiento en el ambiente de desarrollo realizando la petición en cada endpoint. Cada uno contiene parámetros de consulta para filtrar por usuario, intervalo de fecha, paginado y límite de consultas.

Figura 28

Petición exitosa a la ruta de obtener calorías por Usuario.



Para este ejemplo se consultaron las mediciones de calorías para un usuario específico. Como se puede observar en la imagen, el servicio respondió de forma esperada con el cuerpo JSON. Este contiene la fecha de la medición, el intervalo en el que se quemaron las calorías y el respectivo valor que se almacena en la cadena JSON de measurement_data.

Figura 29

Petición de usuarios agrupados por rango etareo y género.

En la imagen se observa el cuerpo del JSON de la consulta que retorna los usuarios registrados en Whatoko Health agrupados por rango de edad y género.

Figura 30

Petición exitosa a la ruta de mediciones de ritmo cardíaco.

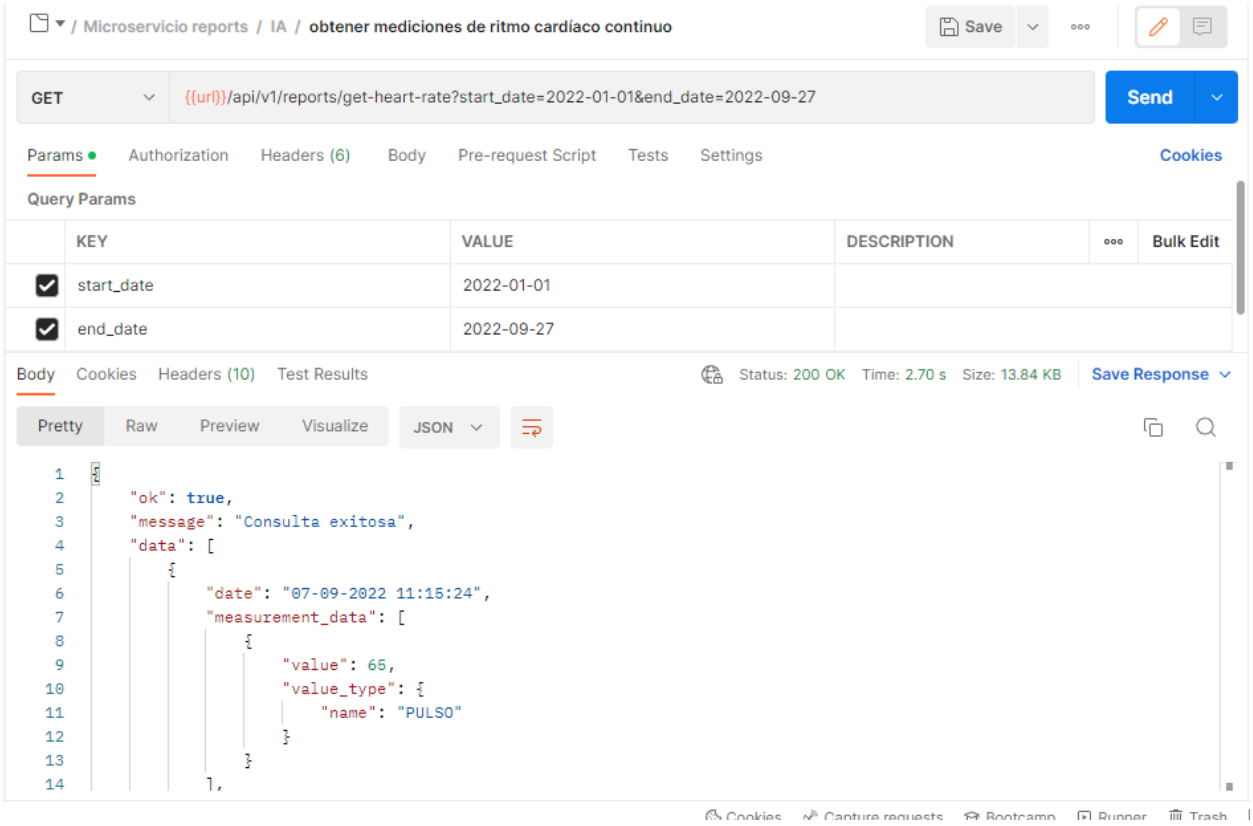
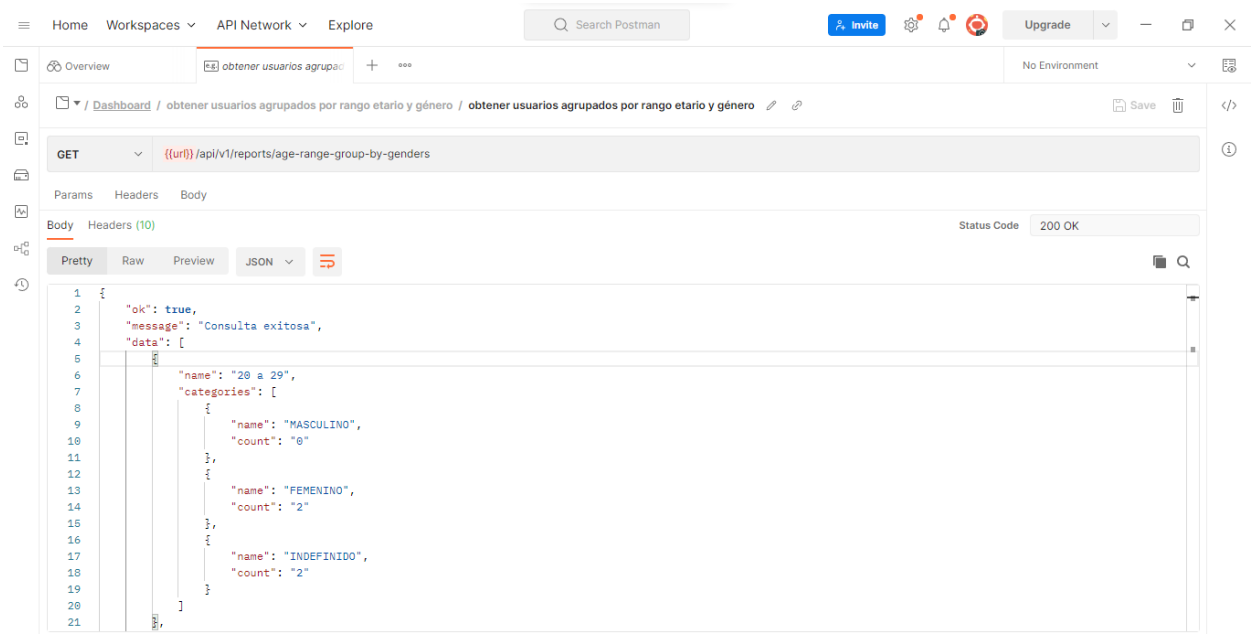


Figura 31
Obtener mediciones de ritmo cardíaco continuo por usuario


```
{
    "ok": true,
    "message": "Consulta exitosa",
    "data": [
        {
            "date": "12-09-2022 00:00:00",
            "measurement_data": [
                {
                    "value": 61,
                    "value_type": {
                        "name": "PULSO"
                    }
                }
            ],
            "measurement_category": {
                "name": "RELAJADO"
            }
        },
```


Este endpoint representa las mediciones de ritmo cardiaco continuo, el cual es la medición que se toma durante todo el día. Cada pulso está clasificado según la categoría con respecto a la frecuencia cardíaca máxima del usuario.

Figura 32
Interfaz de Postman.



Cada endpoint contiene parámetros de consulta con el objetivo de reducir el estrés de procesamiento en cada petición. Por ejemplo, consultar mediciones por un intervalo de tiempo definido mediante los parámetros start_date y end_date. En la figura 9 se puede observar el resultado de una consulta de pasos.

Figura 33
Respuesta de Consulta de Pasos.

```
{
    "ok": true,
    "message": "Consulta exitosa",
    "data": [
        {
            "date": "05-01-2022 00:00:00",
            "start": "05-01-2022 09:59:00",
            "stop": "05-01-2022 10:05:00",
            "measurement_data": [
                {
                    "value": 162,
                    "value_type": {
                        "name": "PASOS"
                    }
                }
            ]
        },
        {
            "date": "05-01-2022 00:00:00",
            "start": "05-01-2022 10:06:00",
            "stop": "05-01-2022 10:13:00",
```


Este endpoint consulta los datos de pasos para un usuario específico. En ello se especifica la fecha de la medición, el intervalo de tiempo en el que se registraron los pasos y el número de pasos tomados en ese intervalo.

Figura 34

Respuesta de usuarios agrupados por género y rango etareo.


```
{
    "ok": true,
    "message": "Consulta exitosa",
    "data": [
        {
            "name": "20 a 29",
            "categories": [
                {
                    "name": "MASCULINO",
                    "count": "0"
                },
                {
                    "name": "FEMENINO",
                    "count": "2"
                },
                {
                    "name": "INDEFINIDO",
                    "count": "2"
                }
            ]
        },
```

Figura 35
Obtener Calorías por Usuario


```
{
    "ok": true,
    "message": "Consulta exitosa",
    "data": [
        {
            "date": "27-08-2022 00:00:00",
            "start": "27-08-2022 09:59:00",
            "stop": "27-08-2022 10:05:00",
            "measurement_data": [
                {
                    "value": 14,
                    "value_type": {
                        "name": "CALORIAS"
                    }
                }
            ]
        },
```

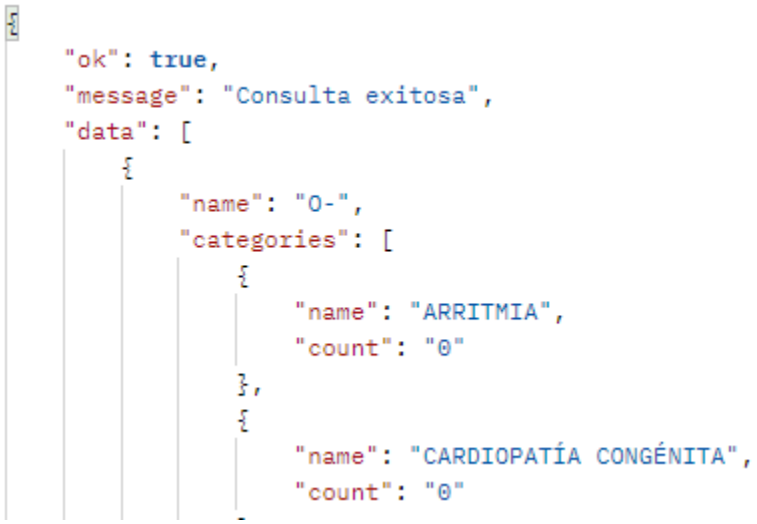

Este endpoint entrega las mediciones de calorías quemadas para un usuario específico.

Figura 36
Obtener mediciones de Glucosa


```
{
    "ok": true,
    "message": "Consulta exitosa",
    "data": [
        {
            "date": "08-08-2022 07:34:00",
            "userId": 1,
            "measurement_data": [
                {
                    "value": 78
                }
            ],
            "measurement_category": {
                "name": "NORMAL"
            },
            "measurement_details": [
                {
                    "measurements_id": 147080,
                    "option": {
                        "name": "GENERAL"
                    }
```


Este endpoint entrega las mediciones de glucosa para un usuario específico. Estas se entregan por la fecha en la que se tomó la medición, con la clasificación de la medición y el horario de comida en la que se tomó la medición.

Figura 37
Obtener Enfermedades por Grupo Sanguíneo



```
{
    "ok": true,
    "message": "Consulta exitosa",
    "data": [
        {
            "name": "O-",
            "categories": [
                {
                    "name": "ARRITMIA",
                    "count": "0"
                },
                {
                    "name": "CARDIOPATÍA CONGÉNITA",
                    "count": "0"
```


Este endpoint entrega los usuarios agrupados por enfermedades y grupo sanguíneo.

Figura 38
Obtener mediciones de Tensión Arterial


```
"data": [
    {
        "date": "28-08-2022 04:03:00",
        "id": 314860,
        "userId": 27,
        "measurement_data": [
            {
                "value": 131,
                "value_type": {
                    "name": "PRESION ARTERIAL SISTOLICA"
                }
            },
            {
                "value": 86,
                "value_type": {
                    "name": "PRESION ARTERIAL DIASTOLICA"
                }
            }
        ],
        "measurement_category": {
            "name": "HIPERTENSIÓN 1"
```


Este endpoint entrega los registros de las mediciones de presión arterial sistólica y diastólica de los usuarios registrados.

Figura 39
Obtener Mediciones de Colesterol

```
{
    "ok": true,
    "messsage": "Consulta exitosa",
    "data": [
        {
            "date": "31-07-2022 00:00:00",
            "id": 1,
            "userId": 1,
            "measurement_data": [
                {
                    "value": 59.1,
                    "value_type": {
                        "name": "HDL"
                    }
                }
            ]
        },
```


Este endpoint entrega las mediciones de colesterol para un usuario específico. Por ejemplo, colesterol total, triglicéridos, colesterol bueno y colesterol malo.

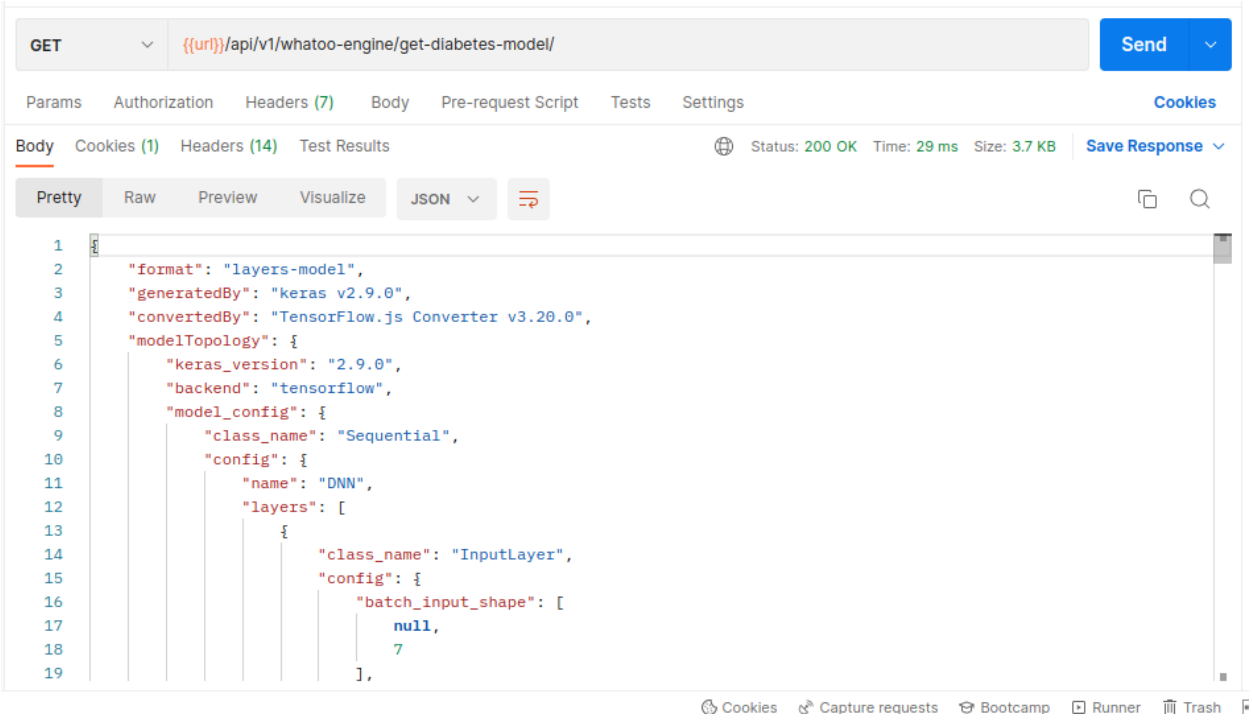
4.4.2 Servicio de Entrenamientos

Para probar el funcionamiento del servicio de entrenamientos, se utilizó POSTMAN para verificar la ejecución de entrenamientos y entrega de modelos.

4.4.2.1 Generación de Modelos Entrenados en Keras.

Esta prueba se realizó en POSTMAN haciendo la petición a la ruta que entrega el modelo entrenado.

Figura 40
Modelo de Capas de Red neuronal en JSON.



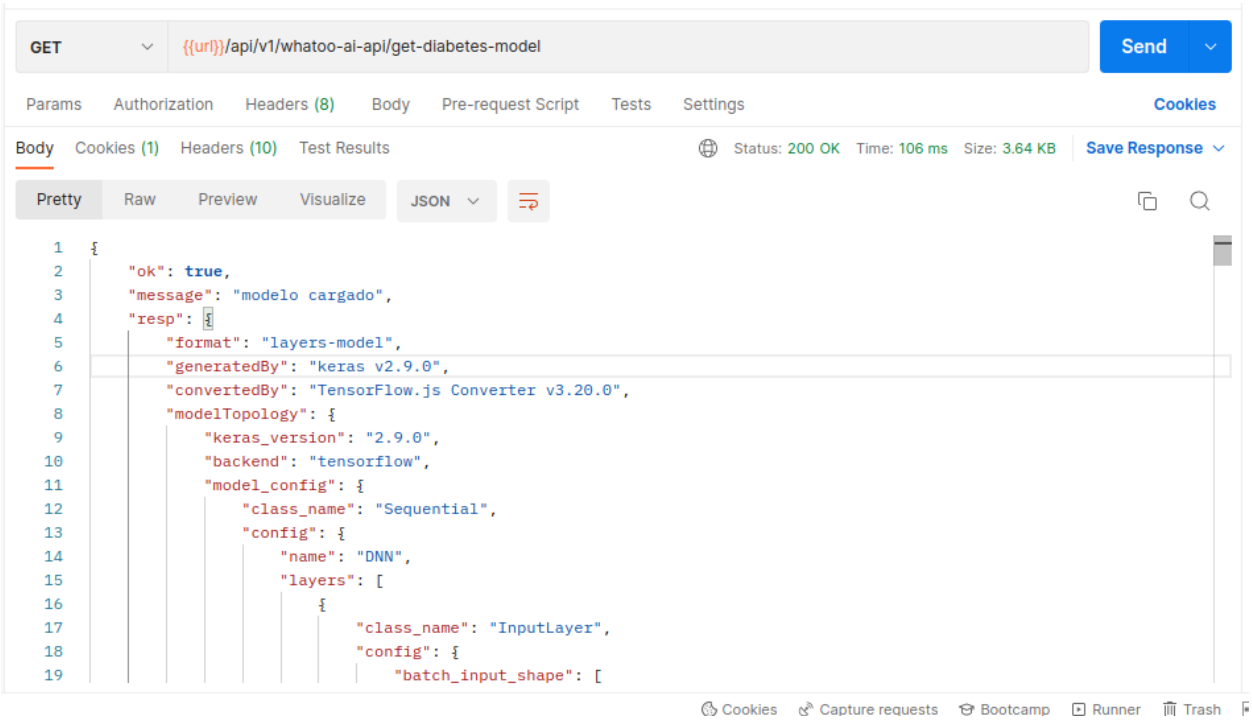
4.4.3 Motor de Inteligencia Artificial.

Para probar el motor de inteligencia artificial, se realizó el procedimiento de prueba en la colección de rutas que accionan las tareas del motor. Para ello se realizó la prueba del modelo que predice el riesgo de diagnóstico de diabetes en 2.5 años. Esta prueba se realizó utilizando la herramienta POSTMAN, en la cual se realizará el siguiente paso a paso.

- Cargar el modelo de diabetes y desplegarlo en desarrollo
- Ejecutar la tarea de realizar la predicción obteniendo los datos de reportería
- Recibir la respuesta de la predicción de las categorías en las que se clasificaron los usuarios según el riesgo y la lista de usuarios que entraron en la categoría de alto riesgo.

4.4.3.1 Cargue del Modelo. En esta sección se realizó la prueba para desplegar modelos entrenados en keras, los cuales son entregados por el microservicio de entrenamientos. Para ello, se usó POSTMAN como herramienta de testing en la ruta que consume un determinado modelo.

Figura 41
Clasificación de niveles de glucemia de los usuarios en Alto riesgo.



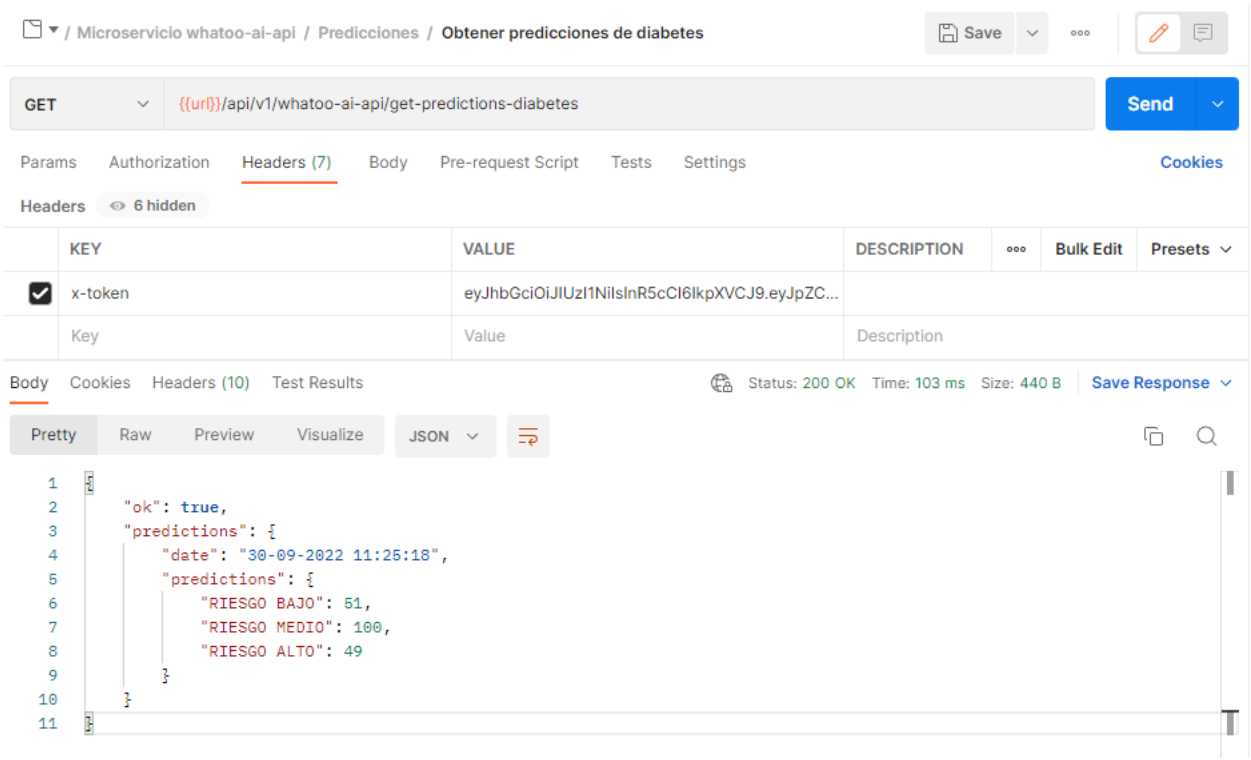
4.4.3.2 Ejecución de las tareas del Motor de Inteligencia Artificial. En esta prueba se validó el funcionamiento del motor de inteligencia artificial en desarrollo. Para ello se realizaron las siguientes pruebas.

- Ejecución de Tareas Programadas con periodicidad.
- Ejecución de las tareas Programadas por petición a la Api

4.4.3.3 Entrega de informes del motor de Inteligencia Artificial. En esta prueba se validó el funcionamiento de la entrega de resultados del motor de inteligencia artificial, los cuales son consumidos por el dashboard. Para ello se realizaron las siguientes pruebas.

4.4.3.3.1 Prueba de Entrega de Informes de la API. En esta prueba se validó el funcionamiento del motor de inteligencia Artificial para la entrega de informes. Para ello se probó la colección de rutas en POSTMAN, de tal manera que el resultado esperado sea el cuerpo JSON que consume el Dashboard Web.

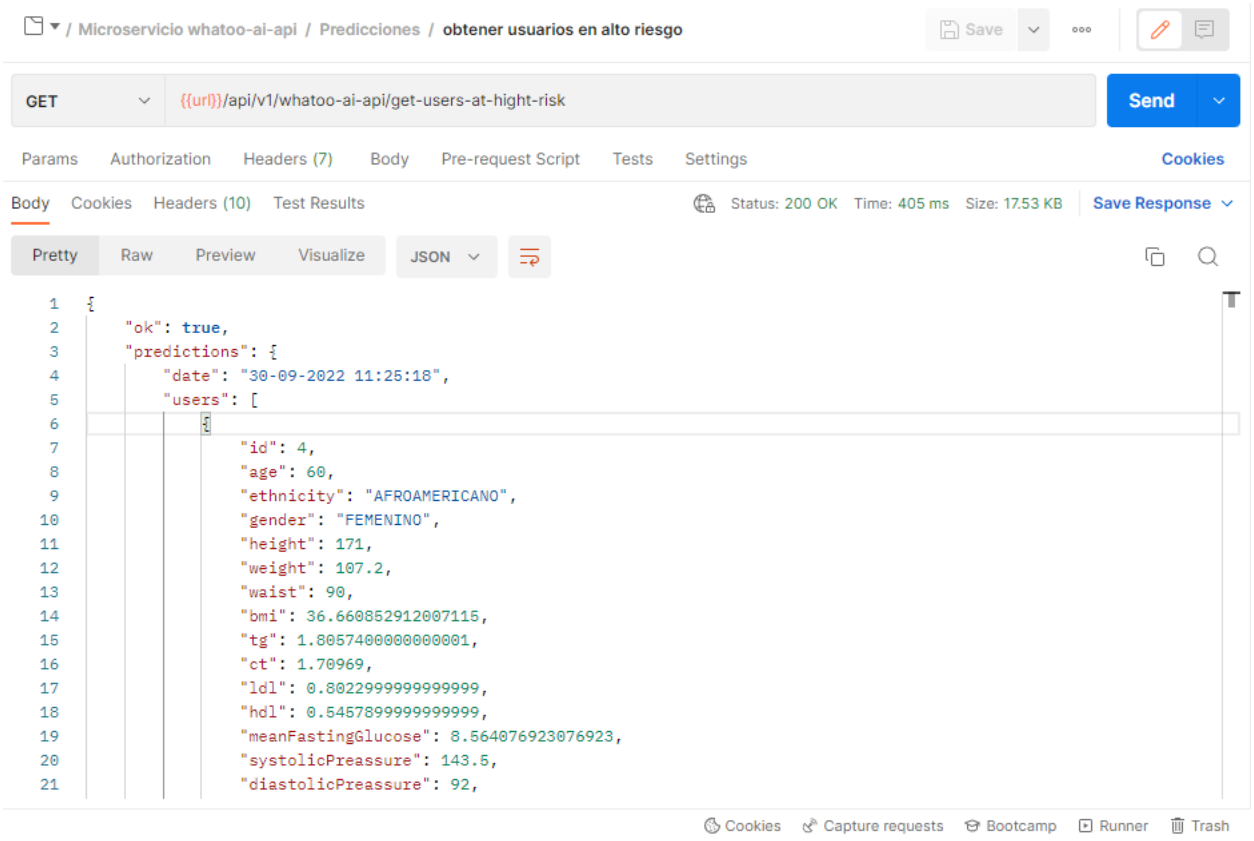
Figura 42
Obtener clasificación de predicciones según zona de riesgo.



En esta imagen se ilustra la respuesta en el ambiente de desarrollo. Se puede ver que la respuesta fue exitosa y se retorna el cuerpo JSON que es el resultado de la predicción generada

por el modelo. El cuerpo representa el proceso de análisis y predicción de los 200 usuarios de prueba, en los cuales agrupó por categoría de Riesgo de diagnóstico de diabetes tipo 2.

Figura 43
Generación de alertas a los usuarios en categoría de alto riesgo.



En esta respuesta se observan los usuarios que entraron en categoría de alto riesgo. El cuerpo JSON contiene la lista de estos usuarios con su ID y el resultado de su información preprocesada. Esta información será consumida por el microservicio de notificaciones, el cual se encargará de recibir esta respuesta y hacer difusión de alertas a los usuarios identificados.

Figura 44

Reporte de usuarios en alto riesgo agrupados por género y etnia.

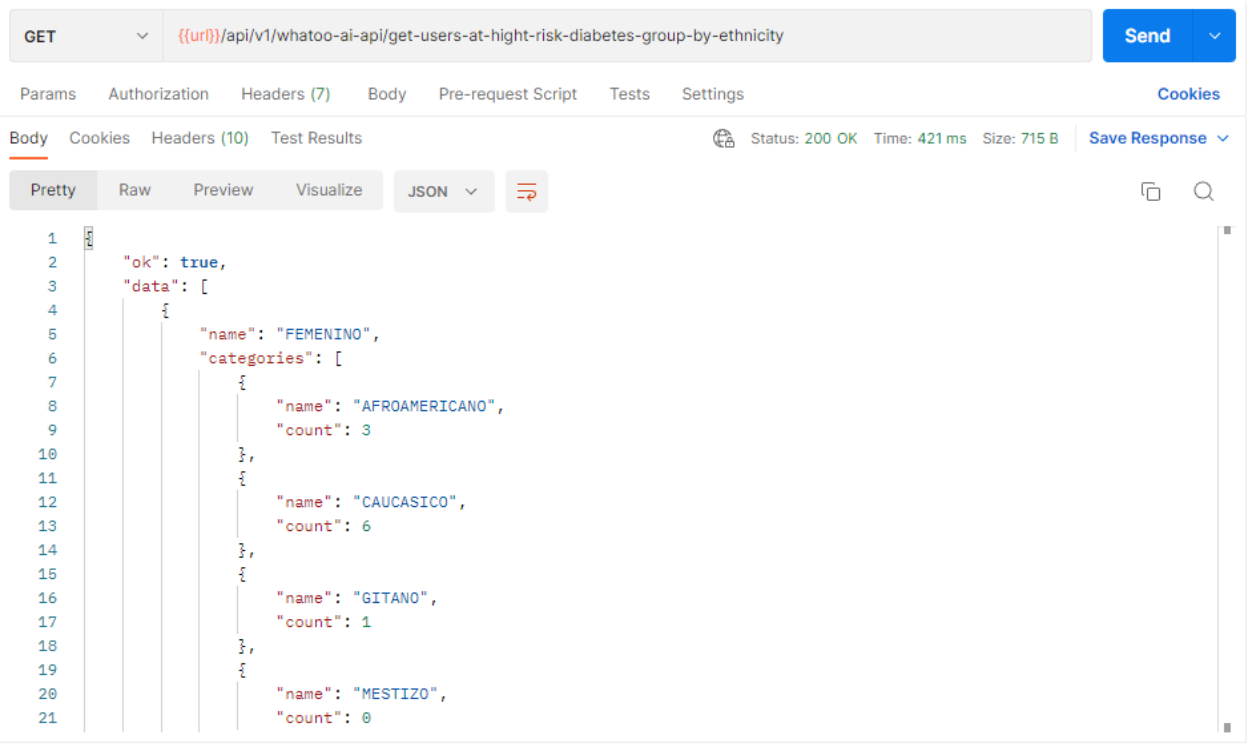
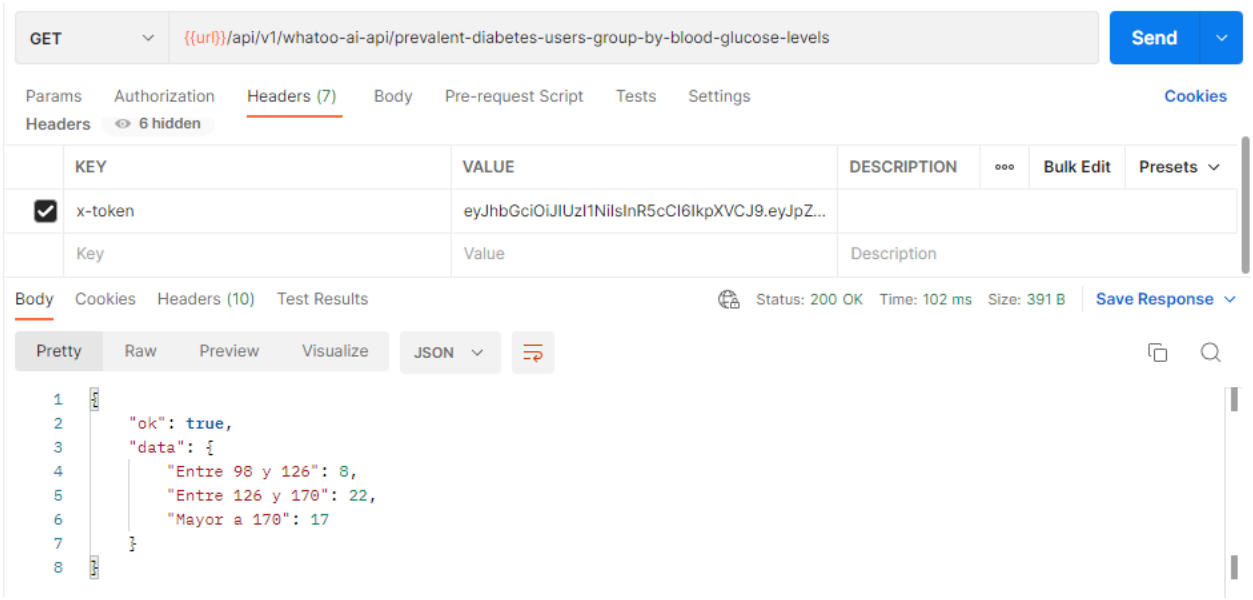


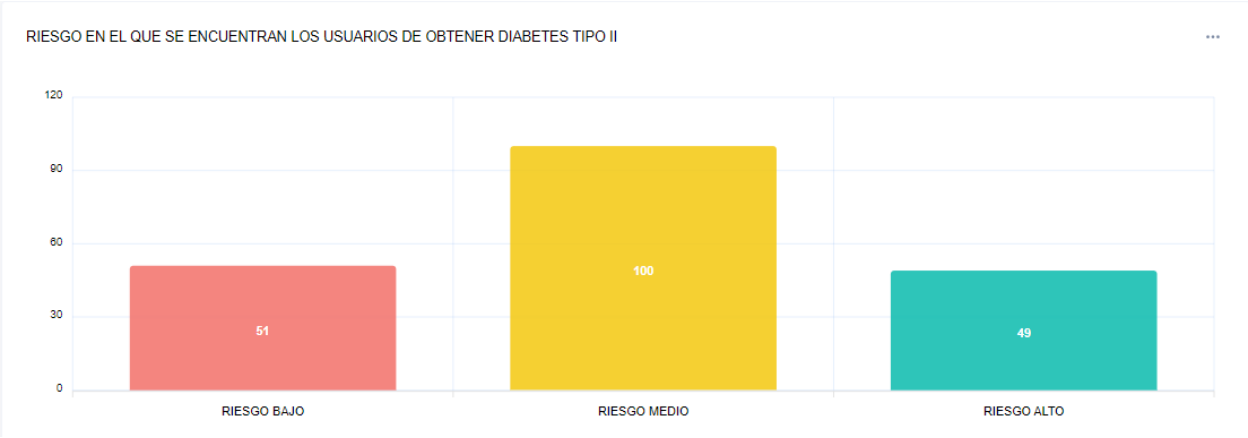
Figura 45

Clasificación de niveles de glucemia de los usuarios en Alto riesgo.



4.4.3.3.2 Prueba de Visualización en Dashboard. En esta prueba se validó el funcionamiento del Motor en el dashboard Web, el cual fue desarrollado en otro proyecto. Para ello se tomaron capturas de la gráfica que consume el endpoint de predicciones.

Figura 46
Gráfico de barras de distribución de zonas de riesgo.



Como se puede observar en el gráfico de barras, se consume el endpoint que genera la predicción de los usuarios con riesgo de diagnóstico de diabetes tipo 2 agrupándolos por zona de riesgo.

5. Conclusiones

En este documento se presentó el trabajo realizado durante la práctica empresarial en A&A Soluciones -TIC. En ella se realizó la implementación del módulo gestión de los datos de usuarios y el desarrollo de un motor de inteligencia artificial integrado al Dashboard de Whatoko Health.

Durante la práctica se presentaron bastantes dificultades en el desarrollo del proyecto. Entre ellas, el poco conocimiento de las tecnologías en los ambientes de desarrollo, la falta de entendimiento de la problemática del proyecto y las funcionalidades que se requerían. Sin embargo, con el acompañamiento recibido por parte del líder Técnico y el producto Owner, se tuvo la correcta orientación para el cumplimiento de los objetivos específicos. En este proyecto se

destaca el desarrollo del módulo de reportería para gestionar las mediciones y entregar informes al dashboard web. La implementación del servicio de entrenamiento de modelos en la cual se definen los modelos propuestos para despliegue. El motor de inteligencia artificial que se encarga de realizar las distintas rutinas de predicción a partir de los conjuntos de datos.

La justificación se basó en cómo la inteligencia artificial acompañada de la correcta gestión y organización de los datos, se pueden encontrar patrones para detectar factores de riesgo relacionados a la diabetes e Hipertensión. Aunque el proyecto aún no cumple con el objetivo de poder ser eficiente en la tarea de alertar a los usuarios, la arquitectura y el flujo utilizado para llegar al resultado esperado cumple con la funcionalidad básica de lo que un motor de inteligencia artificial debe realizar en un aplicativo enfocado al autocuidado.

A futuro se espera implementar más tareas de predicción fomentando la participación en grupos de investigación con acompañamiento médico, de tal manera que se logre el objetivo de salvar vidas. Es claro que a medida que se ingresen nuevos dispositivos y tipos de medición, es probable realizar la refactorización de los componentes actuales.

La experiencia vivida en la práctica empresarial permitió desarrollar habilidades laborales y profesionales en un ambiente real de desarrollo de software. Se pudo complementar el conocimiento adquirido en el programa de ingeniería de sistemas y el conocimiento exigido por la empresa.

Finalmente, se concluye el cumplimiento satisfactorio en el desarrollo profesional logrado a través de la empresa. Esto permite tener un panorama claro de lo que se vive en el mundo real.

Referencias Bibliográficas

Accu-check. (16 de 05 de 2019). *MySugr*. https://www.accu-chek.com.co/mysugr

Aliberti, A. (2019). A Multi-Patient Data-Driven Approach to Blood Glucose Prediction. *IEEE Access, 7*. https://doi.org/10.1109/ACCESS.2019.2919184

Amazfit. (s.f.). *¿Cómo uso el PAI?* ¿Cómo uso el PAI?: https://support.amazfit.com/es/faq/8143

AWS. (2022). *aws*. aws: https://aws.amazon.com/es/devops/what-is-devops/

AWS. (21 de 02 de 2022). *Contenedores de Docker | ¿Qué es Docker?* AWS: https://aws.amazon.com/es/docker/

Ballinger, B., Hisieh, J., Singh, A., Sohoni, N., Wang, J., Tison, G. H., Marcus, G. M., Sanchez, J. M., Maguire, C., Olgin, J. E., & Pletcher, M. J. (2018). DeepHeart: Semi-Supervised Sequence Learning for Cardiovascular Risk Prediction. https://doi.org/https://doi.org/10.48550/arXiv.1802.02511

Campesato, O. (2020). *Machine Learning and Deep Learning.* Mercury Learning And Information. https://doi.org/https://ebookcentral.proquest.com/lib/bibliouis-ebooks/detail.action?docID=6032875.

Cardiogram. (19 de 02 de 2022). *Cardiogram – What's your heart telling you?* Cardiogram – What's your heart telling you?: https://www.cardiogr.am/

EL-Sappagh, S., El-Masri, S., Kim, K., Ali, A., & S Kwak, K. (2019). Mobile Health Technologies for Diabetes Mellitus: Current State and Future Challenges. *IEEE Access, Vol 7*. https://doi.org/ 10.1109/ACCESS.2018.2881001

Git. (15 de 02 de 2022). *Git--Acerca del control de Versiones.* Git--Acerca del control de Versiones: https://git-scm.com/book/es/v2/Inicio---Sobre-el-Control-de-Versiones-Acerca-del-Control-de-Versiones

Hartz, J., Yingling, L., & Powell-Wiley, T. M. (2016). Use of Mobile Health Technology in the Prevention and Management of Diabetes Mellitus. (130). https://doi.org/https://doi.org/10.1007/s11886-016-0796-8

Hema, S. T., Naresh Kumar, S., Padmaja, S., Rama Krishna, C., CB, & Mahender, K. (2020). Scrum: An Effective Software Development Agile Tool. *IOP Conference Series. Materials Science and Engineering*(981). https://doi.org/https://doi.org/10.1088/1757-899X/981/2/022060

ilumivu. (2022). *Learn your heart's language*. Cardiogram Heart Rate Monitor.

Ioannis Kavakiotis, O. T. (2017). Machine Learning and Data Mining Methods in Diabetes Research. *Computational and Structural Biotechnology Journal,, 15*, 104-116. https://doi.org/https://doi.org/10.1016/j.csbj.2016.12.005

Khanna, & Awad, M. (2015). *fficient Learning Machines : Theories, Concepts, and Applications For Engineers and System Designers.* Apress. https://doi.org/https://doi.org/10.1007/978-1-4302-5990-9

Kubernetes. (21 de 02 de 2022). *¿Qué es Kubernetes?* ¿Qué es Kubernetes?: https://kubernetes.io/es/docs/concepts/overview/what-is-kubernetes/

L. Veliz-Rojasa, S. M.-P. (2014). Adherencia terapéutica y control de los factores de. *Enfermería Universitaria*, 9. https://doi.org/https://doi.org/10.1016/j.reu.2015.05.003

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. Nature. *Nature*(521). https://doi.org/http://dx.doi.org/10.1038/nature14539

Maida, E. G., & Pacienzia, J. (2015). Tesis de Licenciatura en Sistemas y Computación, metodologías de desarollo de Software. *Universidad Católica Argentina*, 116.

Martínez, M., Ochora Fierro, G. C., & Del pilar, A. (2015). Adherencia al tratamiento en pacientes con diabetes mellitus tipo 2 en el hospital de bosa, Bogotá entre Agosto y octubre de 2015.

Microsoft. (25 de 07 de 2022). *Base de datos NoSQL: ¿qué es NoSQL?* Base de datos NoSQL: ¿qué es NoSQL?: https://azure.microsoft.com/es-es/resources/cloud-computing-dictionary/what-is-nosql-database

Microsoft. (2022). *Conceptos básicos sobre bases de datos.* Conceptos básicos sobre bases de datos: https://support.microsoft.com/es-es/office/conceptos-b%C3%A1sicos-sobre-bases-de-datos-a849ac16-07c7-4a31-9948-3c8c94a7c204#__toc257378454

Microsoft. (14 de 02 de 2022). *Estilo de arquitectura de microservicios—Azure Application Architecture Guide.* Estilo de arquitectura de microservicios—Azure Application Architecture Guide: https://docs.microsoft.com/es-es/azure/architecture/guide/architecture-styles/microservices

Microsoft. (22 de 06 de 2022). *Microsoft*. Microsoft: https://learn.microsoft.com/es-es/devops/plan/what-is-scrum

Ministerio de Salud. (s.f.). Manual de Prevención y Control de CTB: https://www.minsalud.gov.co/sites/rid/Lists/BibliotecaDigital/RIDE/VS/PP/32Atencion%20de%20la%20Diabetes%20tipo%202.PDF

Ministerio de Salud Y Protección Social. (14 de 06 de 2020). “Conoce tus números” para prevenir la hipertensión arterial: https://www.minsalud.gov.co/Paginas/Conoce-tus-numeros-para-prevenir-la-hipertension-arterial.aspx

OMS. (10 de 11 de 2021). *Diabetes*. Who: https://www.who.int/es/news-room/fact-sheets/detail/diabetes

OMS. (25 de 08 de 2021). *Hipertensión*. Who: https://www.who.int/es/news-room/fact-sheets/detail/hypertension

OPS. (s.f.). *Diabetes - OPS/OMS*. Organización Panamericana de la Salud: https://www.paho.org/es/temas/diabetes

Oracle. (2022). *¿Qué es una base de datos relacional (sistema de gestión de bases de datos relacionales)?* ¿Qué es una base de datos relacional (sistema de gestión de bases de datos relacionales)?: https://www.oracle.com/co/database/what-is-a-relational-database/

Piperlab. (2021). *Conjunto de Propiedades ACID.* Conjunto de Propiedades ACID: https://piperlab.es/glosario-de-big-data/acid/#:~:text=ACID%20es%20el%20conjunto%20de,no%20modificar%20nada%20en%20absoluto

RedHat. (31 de 01 de 2018). *¿Qué son la integración y la distribución continuas (CI/CD)?* ¿Qué son la integración y la distribución continuas (CI/CD)?: https://www.redhat.com/es/topics/devops/what-is-ci-cd

RetHat. (8 de 05 de 2020). *Red hat*. RedHat: https://www.redhat.com/en/topics/api/what-is-a-rest-api

Roldán Martinez, D., Valderas Aranda, P., & Torres Bosch, V. (2018). *Microservicios Un enfoque integrado.* Ra-Ma.

S. El-Sappagh, F. A.-M. (2019). Mobile Health Technologies for Diabetes Mellitus: Current State and Future Challenge. *IEEE Access, 7*(21917-21947). https://doi.org/10.1109/ACCESS.2018.2881001

Schwaber, K., & Sutherland, J. (2020). *The Scrum Guide.* The Scrum Guide: https://scrumguides.org/scrum-guide.html

Universidad de Alicante. (s.f.). *Modelo vista controlador (MVC).* Modelo vista controlador (MVC): https://si.ua.es/es/documentacion/asp-net-mvc-3/1-dia/modelo-vista-controlador-mvc.html

Verona, J., Duffy, M., & Swartout, P. (2016). *Learning DevOps: Continuously Deliver Better Software.* Packt Publishing. https://doi.org/9781787126619

Zeep. (s.f.). *About-us*. About-us: https://www.zepp.com/