Diseño, desarrollo e implementación de un módulo de conectividad en tiempo real con fabricantes de dispositivos wearables y APIs, para aplicación de control de pacientes con problemas de tensión y/o diabetes.

Juan Camilo Londoño Jaimes

Trabajo de Grado para optar al Título de Ingeniero de Sistemas

Director

Jathinson Meneses Mendoza

Magíster en gestión, aplicación y desarrollo de software

Tutor

Francisco Javier González Silva

Líder técnico y desarrollador senior

Universidad Industrial de Santander

Facultad de Ingenierías Físicomecánicas

Escuela de Ingeniería de Sistemas e Informática

Bucaramanga

2022

Agradecimientos

A mi mamá Myriam, porque a pesar de que ya no está conmigo sigue siendo la razón que me impulsa a ser mejor persona cada día.

A mi papá Gustavo, porque sin el apoyo y el amor que me ha brindado no podría estar donde me encuentro hoy en día.

A mi hermano Andrés, por ser mi compañía y apoyo incondicional desde siempre.

A Chogo, por brindarme la compañía que necesitaba para escribir este documento.

A mis amigos. Aquellos que están conmigo desde el colegio: Laura, Juan José, Gabriela, Nicolás, Manuel, Juan David y Sergio. Aquellos que conocí durante mi etapa universitaria: Gabriela, Alejandro, Juan Sebastián, Juan Esteban, Kevin, Brayan, Angélica, Santiago, Rafael y Diego. A todos, gracias por ayudarme durante todos estos años a ser la persona que soy hoy.

A mis abuelos, tíos y primos por todas las experiencias compartidas y el amor que me han brindado a lo largo de los años.

A mi primo Juan Sebastián porque siempre ha estado ahí para subirme el ánimo cuando lo he necesitado.

A mi director de proyecto, el profesor Jathinson Meneses, por la ayuda, las sugerencias y el tiempo brindado durante la realización del proyecto.

A la empresa A&A Soluciones TIC y a Juan Carlos, por brindarme la oportunidad de realizar la práctica empresarial con ellos y las enseñanzas producto de ello.

A la Universidad Industrial de Santander por brindarme la formación personal y profesional durante los últimos cinco años.

Tabla de Contenido

Pág.

Lista de Tablas

Pág.

Lista de Figuras

Pág.

Resumen

Título: Diseño, desarrollo e implementación de un módulo de conectividad en tiempo real con fabricantes de dispositivos *wearables* y APIs, para aplicación de control de pacientes con problemas de tensión y/o diabetes.[*]

Autor: Juan Camilo Londoño Jaimes[**]

Palabras Clave: API Gateway, Arquitectura basada en microservicios, Diabetes, Hipertensión.

Descripción:

La diabetes y la hipertensión son enfermedades responsables de una gran cantidad de muertes cada año y muchas de estas son debido a la poca atención y control que los pacientes les dan a estas enfermedades. A pesar de que existen múltiples aplicaciones para llevar el control de estas enfermedades, la mayoría no cuenta con una opción sencilla de integrar dispositivos como pulseras inteligentes o glucómetros y tensiómetros digitales. Además, estas aplicaciones únicamente se encargan de recolectar datos, pero no hacen un análisis de esta información.

La empresa A&A Soluciones – TIC propone el proyecto Whatoko Health, una aplicación diseñada para llevar un control adecuado de los pacientes con diabetes y/o hipertensión. Con la aplicación se espera poder detectar cualquier anormalidad peligrosa para los pacientes y alertar inmediatamente a familiares y/o centros de atención de salud con el fin de proteger su vida y así reducir los altos números de muertes en el mundo producto de estas enfermedades.

Para el correcto funcionamiento de esta aplicación se requiere de un módulo de comunicación que permita transportar la información y mediciones de los pacientes hacia el ecosistema de Whatoko Health —sin importar qué dispositivo utilicen para tomar los datos—, e insertar la información en la base de datos del proyecto con una estructura predefinida para cada tipo de medición.

[*] Trabajo de Grado en Modalidad de Práctica Empresarial

[**] Facultad de Ingenierías Físico-Mecánicas. Escuela de Ingeniería de Sistemas e Informática. Director: Jathinson Meneses Mendoza. Magíster en gestión, aplicación y desarrollo de software.

Abstract

Title: Design, development, and implementation of a real-time connectivity module with manufacturers of wearable devices and APIs, for a control application for patients with blood pressure and/or diabetes problems.[*]

Author(s): Juan Camilo Londoño Jaimes[**]

Key Words: API Gateway, Microservices-based architecture, Diabetes, Hypertension.

Description:

Diabetes and hypertension are diseases responsible for a large number of deaths every year, many of which are due to the little attention and control given to these diseases by the patients. Despite there being multiple applications to control these diseases, most of them do not have a simple option to integrate devices such as smart bracelets or digital blood glucose and blood pressure monitors. In addition, these applications only focus on data collection without taking into consideration the analysis of this information.

The company A&A Soluciones – TIC proposes Whatoko Health, an application designed to adequately control patients with diabetes and/or hypertension. With the application it is expected to be able to detect any dangerous abnormality for patients and immediately alert relatives and/or health facilities to protect their lives and thus reduce the high number of deaths in the world due to these diseases.

For the correct operation of this application, it is required a communication module that allows the information and measurements of the patients to be transported to the Whatoko Health ecosystem and the information to be inserted in the project database with a predefined structure for each type of measurement. This process should be able to be done regardless of which device the user is using to collect the data.

---

[*] Degree Work in Internship modality

[**]Faculty of Physical-Mechanical Engineering. School of Systems Engineering and Informatics. Director: Jathinson Meneses Mendoza. Masters in software management, application, and development.

Introducción

La diabetes es una de las emergencias sanitarias con más rápido crecimiento del siglo 21. Esta enfermedad fue responsable de la muerte de aproximadamente 6.7 millones de personas en edades de 20 a 79 años en 2021, es decir, 1 muerte cada 5 segundos (International Diabetes Federation, 2021). En contraste, la presión arterial elevada es responsable de aproximadamente 7.6 millones de muertes al año; más muertes que cualquier otro factor de riesgo, teniendo en cuenta que únicamente el 21% de los adultos con esta enfermedad a nivel mundial la tienen bajo control (World Health Organization, 2021).

Debido al monitoreo inadecuado que tienen algunos pacientes con diabetes y/o hipertensión, surge la necesidad de tener una manera sencilla de facilitar el control de estas enfermedades. Está comprobado que las aplicaciones pueden ser herramientas eficaces para controlar los niveles de glucosa en la sangre y la presión arterial, ya que su uso facilita el manejo de datos de salud, la comunicación entre el paciente y el proveedor de salud, entre otros (Liu, Xie, & Or, 2020). Existen aplicaciones para el control de pacientes diabéticos como MySugr, Glucose Buddy y Diabetes:M, con funcionalidades como integración con glucómetros y aplicaciones fitness, conteo de carbohidratos, recordatorios de pruebas, entre otras (Healthline, 2022). Por otro lado, existen aplicaciones cuya principal funcionalidad es llevar el control del ritmo cardiaco y presión sanguínea, tales como Smart BP, Welltory y Qardio Heart Health (MacMillan, 2020).

A pesar de que existen múltiples aplicaciones para controlar las enfermedades mencionadas, estas no cuentan con una forma sencilla de integrar múltiples pulseras inteligentes o *smart watches*, lo cual imposibilita su uso a pacientes que no cuenten con tecnologías específicas. Además, la mayoría de estas aplicaciones no realiza un análisis de los datos obtenidos, lo cual

puede servir para generar predicciones y prevenir complicaciones de salud. A partir de lo anterior se propone una aplicación que cuente con el diseño y desarrollo de una API Gateway que recolecte datos de salud de los pacientes a través de dispositivos *wearables* para tener un control constante y en tiempo real del estado de salud del paciente.

1. Planteamiento y justificación del problema

La hipertensión arterial (HTA) se presenta cuando la presión sanguínea, es decir la fuerza que ejerce la sangre contra las paredes de las arterias, se eleva por encima de cierto nivel. Esta presión se divide en dos: La presión sistólica, la cual es la presión en los vasos sanguíneos cuando el corazón se contrae o late, y la presión diastólica, la cual es la presión en los vasos sanguíneos cuando el corazón descansa entre latidos. En muchos casos se refieren a la HTA como un "asesino silencioso" debido a que una gran cantidad de gente con hipertensión no sabe que la tienen ya que esta puede no presentar síntomas. Por esta razón es importante tener una medición regular de la presión arterial, ya que una hipertensión fuera de control puede causar complicaciones como dolor de pecho, ataque al corazón, insuficiencia cardiaca y latidos irregulares que pueden conducir a la muerte. Existen múltiples factores de riesgo para la hipertensión tales como una dieta no saludable, inactividad física, consumo de tabaco o alcohol, obesidad y enfermedades coexistentes como la diabetes (World Health Organization, 2021).

La diabetes mellitus, comúnmente conocida como diabetes, es una condición crónica que ocurre cuando se elevan los niveles de glucosa en la sangre debido a la poca producción de insulina o la incapacidad de utilizar la insulina producida. La insulina es una hormona producida en el páncreas la cual permite que la glucosa que está en la sangre entre a las células del cuerpo para ser

convertida en energía o almacenada. Una deficiencia de insulina podría causar daños a múltiples órganos del cuerpo si no se controla por un período largo de tiempo. Algunos de estos daños pueden ser enfermedades cardiovasculares, daños en los nervios, daños en el riñón, amputaciones de miembros inferiores y enfermedades oculares.

Según (World Health Organization, 2021), las enfermedades cardiovasculares son la principal causa de muerte a nivel mundial con un estimado de 17.9 millones de muertes al año, las cuales representan el 32% del total de muertes anuales. La diabetes y la hipertensión arterial son factores de riesgo para el desarrollo de enfermedades cardiovasculares. La diabetes se presenta en el 40 al 60% de personas que padecen hipertensión y cuando ambas se encuentran presentes se multiplica el riesgo de padecer un infarto de miocardio, insuficiencia renal, enfermedad vascular periférica y accidentes vasculares cerebrales. Sin embargo, las consecuencias producidas por el conjunto de estas enfermedades pueden ser evitadas o suavizadas en gran parte teniendo un control constante de estas (Osakidetza, 2021). Como se mencionó previamente, el uso de aplicaciones es una manera eficaz de llevar un monitoreo de los niveles de glucosa en la sangre y la presión arterial, ya que facilita el manejo de datos de salud de los pacientes y permite diferentes funcionalidades en caso de emergencias, como la comunicación entre paciente y proveedor de salud (Liu, Xie, & Or, 2020).

A&A Soluciones – TIC es una empresa dedicada a proporcionar soluciones innovadoras basadas en tecnologías de información a empresas, clientes y consumidores en cualquier parte del mundo. Además, la empresa proporciona la oportunidad a estudiantes para que realicen sus prácticas profesionales con el fin de finalizar sus estudios y adquirir experiencia en un ambiente laboral en el proceso. En conjunto con los practicantes, la empresa desarrolla un proyecto denominado Whatoko Health: una aplicación diseñada para llevar un control detallado de los

pacientes con problemas de diabetes y/o hipertensión. Con la aplicación se espera poder detectar cualquier anormalidad peligrosa para los pacientes y alertar inmediatamente a familiares y/o centros de atención de salud con el fin de proteger su vida y así reducir los altos números de muertes en el mundo producto de estas enfermedades.

Para el correcto funcionamiento de la aplicación es necesario tener una forma de obtener la información médica del paciente en tiempo real. Para esto se utiliza un módulo de conexión entre la base de datos y la aplicación de Whatoko Health. Este módulo se encarga de estructurar los datos que la aplicación obtuvo previamente por medio de SDKs de los dispositivos *wearables*, APIs de aplicaciones de control de salud y consultas a los pacientes dentro de la aplicación para que, sin importar la fuente de los datos, estos tengan el mismo formato. Una vez estructurados los datos, estos se ingresan en la base de datos para que, posteriormente, puedan ser consultados por los diferentes microservicios dentro del ecosistema de Whatoko Health.

2. Objetivos

2.1 Objetivo General

Diseñar y desarrollar una API Gateway, basada en el estilo arquitectónico REST, que vincule a Whatoko Health, aplicación de control de salud, con los datos recolectados a través de pulseras inteligentes o *smartwatches* que cuenten con lectura de pulso, glucómetros digitales y aplicaciones de control de salud.

2.2 Objetivos Específicos

Establecer los parámetros de desarrollo del módulo de conectividad mediante una revisión del estado del arte.

Definir los componentes de integración por medio de la evaluación de las tecnologías de pulseras y aplicaciones presentes en el mercado.

Establecer los requerimientos para la construcción del módulo de conectividad entre el lector (dispositivo *wearable*) y el ecosistema de Whatoko.

Diseñar, mediante un lenguaje de modelado, las principales funcionalidades del prototipo.

Transferir la información (captura de mediciones de paciente) entre el lector (dispositivo *wearable*) y el ecosistema de Whatoko.

Implementar pruebas y validación para comprobar el correcto funcionamiento de la aplicación.

3. Marco de referencia

3.1 Fundamentos teóricos

A continuación, se realiza una revisión de los fundamentos teóricos para una mayor comprensión sobre el proyecto y el tema de investigación.

3.1.1 Diabetes

La diabetes es una enfermedad crónica que afecta la manera en que el cuerpo convierte la comida en energía. Cuando la comida es ingerida, gran parte de esta se descompone en azúcar y llega al torrente sanguíneo. Cuando el nivel de azúcar en la sangre aumenta, se manda una señal al páncreas para liberar insulina, hormona que se encarga de dejar pasar el azúcar de la sangre hacia las células del cuerpo para ser utilizada como energía.

Cuando una persona tiene diabetes, su cuerpo no produce suficiente insulina o no puede usar la insulina que produce de la manera correcta. Cuando esto ocurre, una gran cantidad de azúcar se queda en el torrente sanguíneo, lo cual puede producir complicaciones a largo plazo como enfermedades en el corazón, pérdida de la visión y enfermedades renales.

Actualmente no existe una cura para la diabetes, pero se puede reducir sus complicaciones por medio de la pérdida de peso, una dieta saludable y un control constante de los niveles de azúcar en la sangre (Centers for Disease Control and Prevention, 2021).

3.1.2 Hipertensión arterial

La hipertensión arterial ocurre cuando la presión sanguínea, es decir la fuerza que ejerce la sangre contra las paredes de las arterias, está por encima de cierto nivel. La presión sanguínea tiene dos lecturas diferentes: La presión sistólica representa la presión en los vasos sanguíneos cuando

el corazón se contrae o late y la presión diastólica representa la presión en los vasos sanguíneos cuando el corazón está en reposo, entre latidos.

La hipertensión puede ser diagnosticada si, en dos días diferentes, la presión sistólica es mayor o igual a 140mmHg y/o si la presión diastólica es mayor o igual a 90mmHg. Una hipertensión no controlada puede causar daños serios en el corazón y endurecimiento de las arterias, lo cual disminuiría el flujo de sangre y oxígeno al corazón. A largo plazo puede producir dolores de pecho, ataques al corazón, insuficiencia cardiaca e incluso arritmias que pueden causar la muerte (World Health Organization, 2021).

3.1.3 REST API

Una API, Interfaz de programación de aplicaciones por sus siglas en inglés, es un conjunto de normas que definen cómo las aplicaciones y dispositivos se conectan y se comunican entre ellos (IBM, 2020). El funcionamiento de una API se divide en los siguientes pasos:

1. Una aplicación cliente inicia una llamada API para solicitar información, también llamado solicitud.
2. Al recibir una solicitud válida, la API hace una llamada al programa externo o servidor web.
3. El servidor envía una respuesta a la API con la información solicitada.
4. La API transfiere la información a la aplicación cliente.

Una REST API es una API que se ajusta a los principios de diseño de REST (Representational State Transfer Architectural Style). Este tipo de APIs se comunican vía solicitudes HTTP para realizar funciones de base de datos estándar como crear, leer, actualizar y eliminar registros dentro de un recurso (IBM, 2021).

3.1.4 SDK

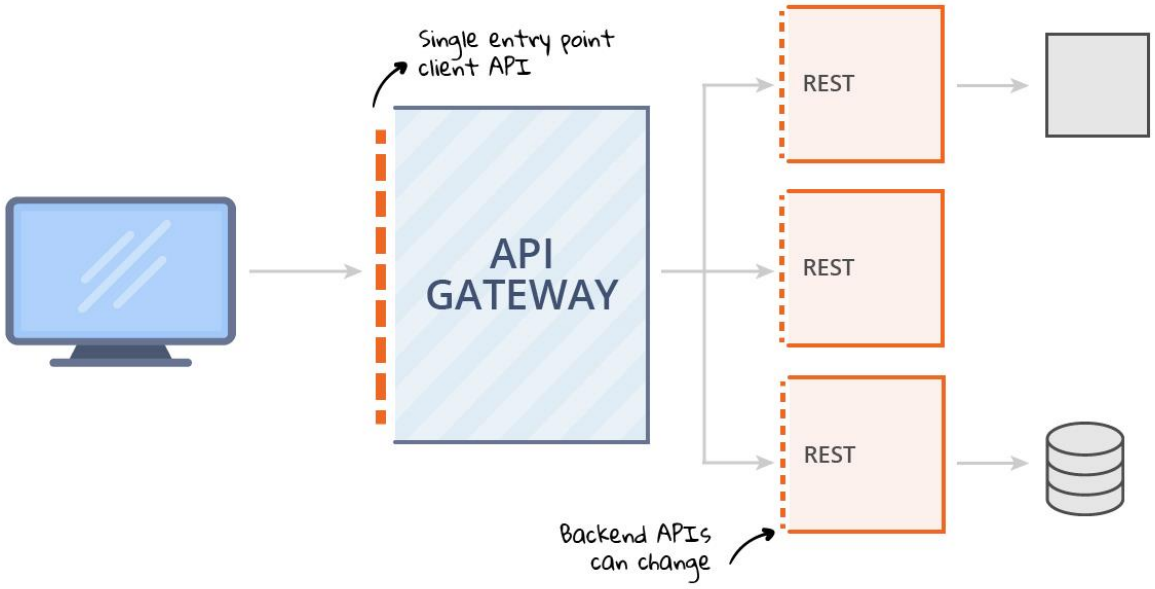
Un SDK, Kit de desarrollo de software por sus siglas en inglés, es un conjunto de herramientas proporcionadas por fabricantes de una plataforma de *hardware*, un sistema operativo o un lenguaje de programación. Los SDKs ayudan a los desarrolladores de software a crear aplicaciones para ese sistema, lenguaje o plataforma específica. Usualmente, estos cuentan con un compilador, un depurador y una API, pero también pueden incluir herramientas como documentación, librerías, editores, entre otras (Red Hat, 2020).

3.1.5 API Gateway

Una API Gateway es una herramienta para administrar APIs, la cual se encuentra ubicada entre un cliente y una colección de servicios *backend*. En el caso de aplicaciones basadas en microservicios, un API Gateway funciona como un único punto de entrada al sistema. Esta herramienta es responsable del enrutamiento de solicitudes, composición y traducción de protocolos y puede optimizar el sistema. Algunas de las solicitudes que llegan a la API Gateway son enrutadas al servicio *backend* adecuado, en otros casos se invocan múltiples servicios *backend* y se agregan los resultados. En caso de que haya algún fallo en alguno de los servicios, la API Gateway puede enmascarar estos fallos retornando datos predeterminados o guardados en caché (Nginx, s.f.).

Figura 1

Arquitectura de una API Gateway



Nota. Recuperado de: https://www.connecting-software.com/es/blog/what-is-an-api-gateway-how-it-can-actually-deliver-practical-results/

3.1.6 Dispositivos wearables

Los dispositivos *wearables* hacen referencia al grupo de tecnología y dispositivos que son diseñados para ser utilizados a lo largo del día. Estos dispositivos se pueden conectar con otros dispositivos como computadores y celulares. Algunos ejemplos de estos son:

- *Smartbands*: Son dispositivos que se encargan de recolectar información sobre la actividad diaria de una persona. Estos pueden guardar información sobre la cantidad de pasos recorridos, frecuencia cardiaca promedio, tiempo de sueño, entre otros. Estos dispositivos se pueden sincronizar con aplicaciones que analicen los datos y permita al usuario ver tendencias y patrones en su actividad diaria.

- *Smartwatches*: Estos dispositivos pueden ser sincronizados con un teléfono inteligente para recibir notificaciones como llamadas, mensajes y correos. Además, la mayoría de

estos dispositivos funcionan también como una *smartband*, recolectando datos sobre la salud del usuario a lo largo del día (GCFGlobal, s.f.).

3.1.7 Internet of Things

El "Internet de las cosas" hace referencia a la red de objetos físicos que cuentan con sensores, software y otras tecnologías con el fin de conectar e intercambiar datos con otros dispositivos y sistemas a través de Internet. Hoy en día se pueden interconectar, con mínima intervención humana, todo tipo de dispositivos de uso diario como lo son aparatos de cocina, termostatos, monitores para bebés, carros, entre otros.

Unos de los dispositivos más importantes son los dispositivos *wearables*, ya que estos permiten que las personas entiendan mejor su salud y puedan llevar un control detallado de la misma (Oracle).

3.1.8 Telemedicina

La telemedicina es el uso de la tecnología con el fin de que un paciente tenga citas o interacciones con su médico u otra persona de su equipo de atención médica. La utilidad de la telemedicina surge porque es posible que una persona no necesite salir de su casa para ser examinado o puede que a esta se le dificulte llegar hasta el centro de atención de salud. Algunas de las tecnologías que se utilizan para hacer uso de la telemedicina son:

- Videollamadas en computadores o teléfonos inteligentes para que haya una comunicación visual en tiempo real entre el paciente y el profesional de la salud.
- Monitoreo remoto de niveles sanguíneos o presión arterial por medio de dispositivos que se pueden conectar a Internet.

- Portales de pacientes en línea en donde se puede tener una conversación con el personal de salud, así como también ver resultados de exámenes, recetas, citas, entre otros (American Cancer Society, 2020).

3.1.9 Control de Versiones

El control de versiones es la práctica de llevar un seguimiento y administrar los cambios a un código de software. Esta práctica es especialmente útil cuando se implementa en equipos de trabajo en donde múltiples personas tienen que hacer cambios en un mismo proyecto de software. Algunos de los beneficios de utilizar un controlador de versiones son:

- Se tiene un historial de cambios a largo plazo de cada archivo existente dentro de un proyecto. Esto permite la posibilidad de volver a una versión anterior del proyecto en caso de que se necesite.
- Se tiene la posibilidad de crear ramas las cuales permiten desarrollar funcionalidades en una copia del proyecto para no interrumpir el trabajo de las demás personas. Cuando la funcionalidad esté completa, se tiene la opción se unir la rama en la que se estaba trabajando con la rama principal del proyecto. Las ramas permiten tener múltiples flujos de trabajo independientes entre sí.
- Se tiene trazabilidad de todo el proyecto, es decir, se sabe cuándo se hicieron cambios a cada archivo junto con un mensaje que explica los cambios. Esta característica facilita la estimación de futuros desarrollos de software (Atlassian, s.f.).

Para el caso de este proyecto se utilizará Git, el cual es el sistema de control de versiones más utilizado y cuenta con todas las características previamente mencionadas.

3.1.10 DevOps

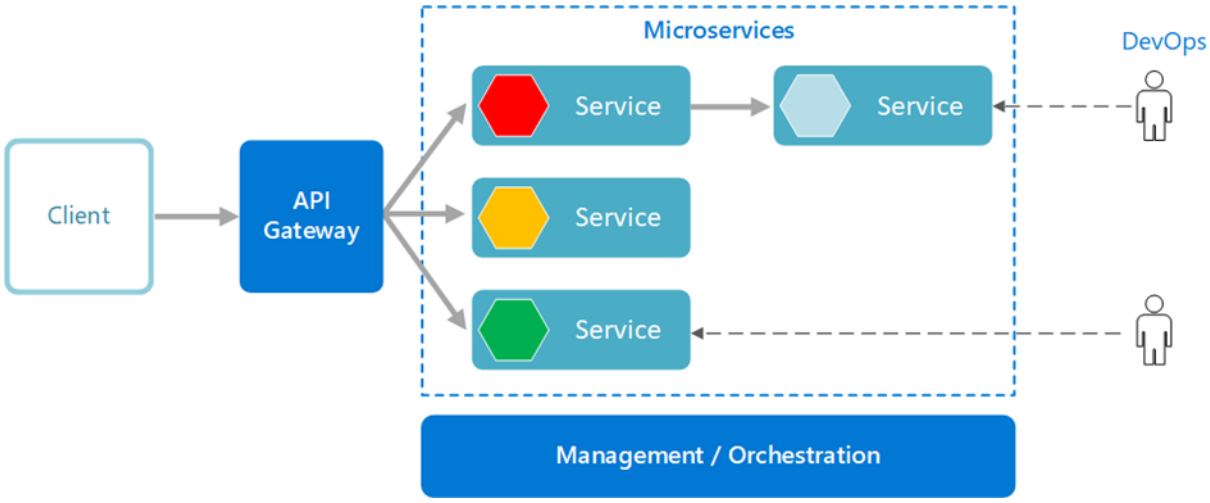
DevOps es un conjunto de prácticas, herramientas y creencias utilizadas para permitir que las empresas mejoren en su habilidad de entregar aplicaciones y servicios a alta velocidad. En este modelo se tiene un solo equipo que se encarga de todo el ciclo de vida de las aplicaciones, desde el desarrollo y las pruebas hasta el despliegue y las operaciones, desarrollando en el equipo una gama de habilidades que no se limitan a una única función. Los principales beneficios de utilizar DevOps son la velocidad en que se desarrollan las cosas, la entrega rápida y eficaz, la fiabilidad que provee a los productos desarrollados, el fortalecimiento del trabajo en equipo y la seguridad que el modelo provee (Amazon Web Services, s.f.).

3.1.11 Arquitectura Basada en Microservicios

Este tipo de arquitectura consiste utilizar servicios pequeños y autónomos. Estos servicios se caracterizan por ser desarrollados en una base de código separada y ser desplegados independientemente, lo cual permite a los desarrolladores actualizar los servicios sin tener que reconstruir y redesplegar toda la aplicación. Además, estos servicios tienen la tarea de persistir datos, ya se propios o de una fuente externa ya que se pueden comunicar entre sí por medio de APIs. El hecho de que los microservicios sean independientes también permite que se utilicen diferentes tecnologías para los diferentes servicios (Microsoft technical documentation, 2022).

Figura 2

Arquitectura basada en microservicios



Nota. Recuperado de: https://docs.microsoft.com/en-us/azure/architecture/guide/architecture-styles/microservices

3.1.12 Whatoko Health

Whatoko Health es un ecosistema diseñado por la empresa A&A Soluciones – TIC con el fin de llevar un control detallado de los pacientes con problemas de diabetes y/o tensión. Con las aplicaciones que componen este ecosistema se espera poder detectar cualquier anormalidad peligrosa para los pacientes y alertar inmediatamente a familiares y/o centros de atención de salud con el fin de proteger su vida y así reducir los altos números de muertes en el mundo producto de estas enfermedades.

3.1.13 Metodologías Ágiles

Las metodologías ágiles surgen a partir de la insuficiencia que tienen las metodologías tradicionales para resolver algunos problemas. Estas se basan en dos aspectos fundamentales: retraso de decisiones y planificación adaptativa. Un modelo de desarrollo ágil de caracteriza por ser un proceso incremental (hay entregas frecuentes repartidas en ciclos rápidos), cooperativo (hay comunicación constante entre clientes y desarrolladores), sencillo (es fácil de implementar en un

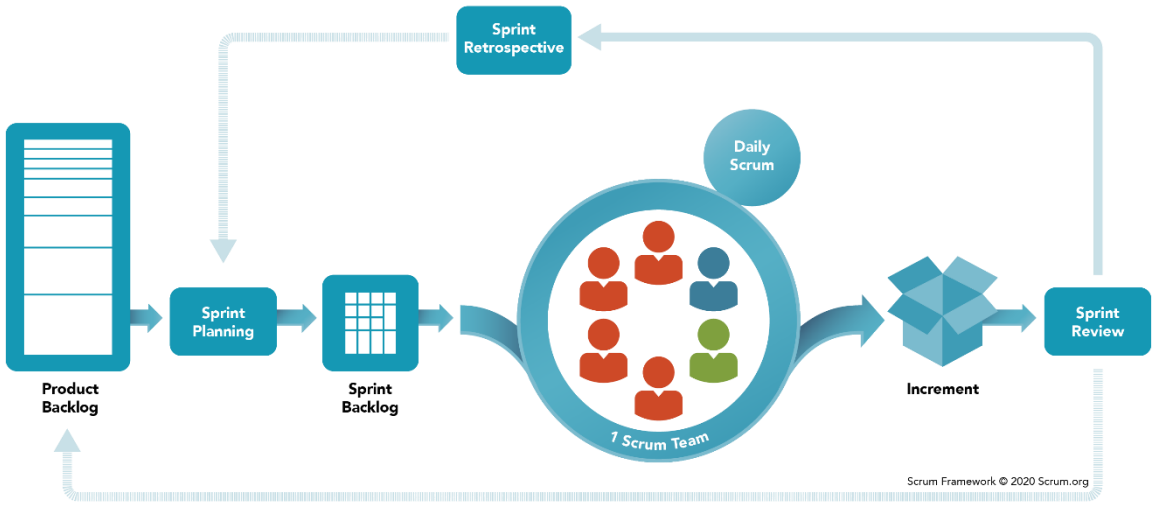
equipo de trabajo) y adaptativo (es posible realizar cambios en último momento) (Pacienzia & Maida, 2015).

3.1.14 Scrum

Scrum es un proceso iterativo e incremental utilizado para desarrollar un producto o administrar un trabajo. Este proceso se centra en cómo los miembros de un equipo tienen que trabajar para ser funcionales en un ambiente de constante cambio. Al final de cada iteración se espera producir un conjunto potencial de funcionalidad dependiendo del producto a desarrollar. A continuación, se tiene un diagrama del flujo en un proyecto manejado con Scrum y el proceso que se llevará a cabo durante el desarrollo del proyecto (Awad, 2005).

Figura 3

Flujo del framework Scrum



Nota. Recuperado de: https://www.scrum.org/resources/what-is-scrum

1. Primero se desarrolla un product backlog, el cual es una lista de todas las características y cambios que se deben realizar en el sistema. Esta lista se va modificando con el tiempo y el encargado de mantenerla es el product owner.

2. A partir de esta lista se plantean los sprints. Es un proceso de aproximadamente un mes en donde los participantes se adaptan a variables ambientales cambiantes como requerimientos, tiempo, recursos, tecnología, entre otras. De este proceso debe resultar un incremento de software potencialmente entregable.

3. En cada sprint se realiza una reunión de planeación del sprint, en donde deben estar presentes los clientes, usuarios, gerentes, product owner y todo el equipo de Scrum. En estas reuniones se establecen el conjunto de objetivos y funcionalidades y se habla sobre cómo el producto será implementado durante el sprint.

4. A partir de la reunión se formula un sprint backlog. Este es una lista de todas las características que se le asignan a un sprint específico. Cuando todas se cumplen, se entrega una nueva iteración del producto o sistema.

5. A lo largo del sprint se hace reuniones diarias llamadas daily scrum. Estas reuniones son cortas y sirven para llevar un seguimiento del progreso que lleva el equipo y hablar sobre cualquier obstáculo que se haya encontrado (Awad, 2005).

6. Al finalizar un sprint, se realiza un sprint review. En este, el equipo y los involucrados revisan qué se logró durante el sprint y qué cambió en el producto o sistema. Basándose en esa información, se decide qué se hará después, en algunos casos se puede modificar el producto backlog según las necesidades.

7. Por último, se realiza un sprint retrospective. El equipo se reúne para inspeccionar cómo estuvo el último sprint en cuanto a individuos, interacciones, procesos y herramientas. En

esta reunión se suelen discutir puntos como qué se hizo bien durante el sprint, qué se puede mejorar y en qué se comprometen los participantes a mejorar para el siguiente sprint (Scrum, s.f.).

3.2 Antecedentes del tema

Hoy en día existen diversas herramientas que le permiten a las personas con enfermedades llevar un control adecuado de estas. Los dispositivos *wearables* son una de estas herramientas, ya que estos tienen la función de estar constantemente monitoreando datos como el pulso cardíaco, sin embargo, los fabricantes de estos dispositivos no cuentan con herramientas adicionales que permitan tener un control más eficaz de las enfermedades, ya que en la mayoría de los casos cuentan únicamente con funciones de lectura de datos. A continuación, se presenta un proyecto y una aplicación que se han centrado en mejorar la atención a pacientes con diabetes y/o hipertensión, además de algunas características que no poseen estos y se esperan implementar en el proyecto propuesto en este documento.

3.2.1 Sistemas para el auto monitoreo de la glucemia y tensión arterial

En 2017 las estudiantes Catherine Pulido y Margarita Valencia de la Pontificia Universidad Javeriana desarrollaron un sistema para asistir a pacientes con diabetes tipo 2 en el auto monitoreo de sus niveles de glucosa y tensión arterial. Este sistema captura los datos a partir de sensores, glucómetro y tensiómetro para posteriormente visualizarlos a través de un servidor y una aplicación móvil previamente desarrollada (Pulido Rozo & Valencia Collazos, 2017).

Este sistema logra cumplir objetivos similares al proyecto que se pretende desarrollar como lo es el monitoreo de datos de salud en pacientes con diabetes y/o hipertensión, sin embargo, para su funcionamiento se hace necesario de una plataforma hardware específica que hay que desarrollar para cada paciente, lo cual imposibilita el acceso a la herramienta a muchas personas. La principal ventaja del proyecto propuesto es que hace uso de dispositivos *wearables*, los cuales

pueden ser adquiridos por cualquier persona y permiten tener un monitoreo constante del paciente sin importar su ubicación. La posibilidad de utilizar estos dispositivos existe gracias a la API Gateway que se pretende desarrollar, la cual permite que haya una comunicación en tiempo real entre los distintos fabricantes de los dispositivos y el ecosistema de Whatoko Health.

3.2.2 Blood Pressure & Glucose Pal

Blood Pressure & Glucose Pal es una aplicación cuyo propósito es asistir en el seguimiento de la presión arterial, frecuencia cardíaca, azúcar en la sangre, paso y registro de medicamentos tomados en pacientes con enfermedades como la diabetes y la hipertensión. Esta aplicación cuenta con múltiples funcionalidades como recordatorios para tomar medicamentos, generación de tablas con la tendencia de los datos de salud recopilados, reportes diarios para revisar el estado de salud, entre otras. Sin embargo, esta aplicación solo cuenta con integración del dispositivo *wearables* de Apple, el Apple Watch, lo cual se convierte en un inconveniente para las personas que no tienen la posibilidad de adquirir este dispositivo (LINKLINKS LTD, 2021).

4. Desarrollo

4.1 Entorno de trabajo

Debido a la pandemia generada por la enfermedad COVID-19 muchas empresas tuvieron que recurrir a trabajar remotamente, sin embargo, desde sus inicios, A&A Soluciones TIC ha venido implementando esta modalidad de trabajo. Algunas de las ventajas de implementar esta forma de trabajar remotamente son:

- Reducción o eliminación del tiempo empleado para llegar a un lugar físico de trabajo.

- Reducción de la necesidad de tener un espacio físico para trabajar, lo cual se traduce en ahorro para la empresa.
- Permite expandir la cantidad de candidatos al momento de contratar a alguien nuevo, ya que esta persona no debe estar en el mismo lugar ni es necesario invertir en gastos de traslado.

A pesar de que implementar este tipo de trabajo tiene ventajas para la empresa, también viene con retos. Al no estar físicamente en el mismo lugar de trabajo, puede haber complicaciones al momento de comunicarse entre miembros de la empresa. Afortunadamente, esta dificultad es aliviada con el uso de diversas herramientas que permiten que la comunicación y el desempeño de los miembros mejore.

4.1.1 Herramientas

En la empresa A&A Soluciones se utilizan diversas herramientas de comunicación y organización de la información para garantizar la productividad de los procesos. A continuación, se especifican algunas de las herramientas y su utilidad.

4.1.1.1 Skype

Skype es una herramienta de comunicación que permite realizar chats, llamadas y videollamadas de manera fácil y gratuita. En la empresa se utiliza Skype para toda la comunicación diaria con los diferentes equipos de trabajo. Para el desarrollo de la práctica se creó un grupo en Skype con las personas que conforman el equipo y en este grupo se realiza toda la comunicación de los avances del proyecto, así como también las reuniones semanales, las cuales quedan grabadas y archivadas en el mismo grupo.

4.1.1.2 Jira

Jira es una herramienta desarrollada por Atlassian utilizada para la gestión de proyectos, seguimiento de errores e incidencias. En la práctica se utilizó Jira para hacer seguimiento del proyecto de Whatoko Health, registrando el backlog y los sprints para así asignar tareas a cada uno de los miembros del equipo. Al momento de desarrollar una tarea, el software permite actualizar el estado, llevar el tiempo utilizado en el desarrollo de esta, dejar comentarios, adjuntar archivos o enlaces que tengan relevancia e incluso enlazar la tarea a una rama de un repositorio. Además, a cada tarea se le puede asignar un responsable, un informador, etiquetas que la describen, una estimación de tiempo y un nivel de prioridad para tener un mayor nivel de organización durante el desarrollo del proyecto.

4.1.1.3 Confluence

Confluence es otra herramienta desarrollada por Atlassian la cual se utiliza para compartir, guardar y trabajar con información de manera colaborativa. Durante la práctica se utilizó Confluence para documentar todo lo relacionado al proyecto. Se tuvo un espacio designado para Whatoko Health en donde se documentó sobre Arquitectura e infraestructura, estándares de calidad, aplicaciones móviles, información sobre dispositivos, base de datos, *frontend* y *backend*. En este espacio también se realizaron pruebas de concepto para verificar el potencial y la utilidad de algunas herramientas que se pensaban integrar al proyecto. Además, también se tenía una sección de base de conocimiento, la cual contenía enlaces de interés con información sobre temas y herramientas relacionadas con el proyecto. Toda esta información puede ser accedida y editada por cualquier miembro del equipo para tener en todo momento la información más completa posible.

4.1.1.4 Google Drive

Drive es una herramienta de Google utilizada para alojar archivos e información en la nube. Durante la práctica se utilizó esta herramienta para alojar toda la información relevante para el desarrollo del proyecto. Se creó una carpeta en donde se guardaron gráficas, tablas, información sobre fabricantes de dispositivos, logos de la aplicación, grabaciones de las reuniones semanales, documentación de las enfermedades, entre otras cosas. El uso de esta herramienta permitió tener un lugar común para guardar toda la información y archivos relacionados al proyecto y que todos los miembros del proyecto tuvieran acceso a ellos.

4.1.2 DevOps

Como se mencionó anteriormente, utilizando prácticas de DevOps se tiene un solo equipo que se encarga de todo el ciclo de vida de las aplicaciones, desde el desarrollo y las pruebas hasta el despliegue y las operaciones. Para el caso de la práctica se implementaron estas prácticas, en donde cada miembro del equipo tuvo como tarea realizar actividades de todo tipo, como por ejemplo la planificación de la estructura del API Gateway, el desarrollo del microservicio de fabricantes, entre otras. El uso de estas prácticas permitió la obtención de conocimientos en todas las etapas del desarrollo de software, además, al seguirlas adecuadamente permite que el producto final se desarrolle con mayor agilidad y sea más confiable.

4.1.3 Metodología Scrum en la práctica

Durante la práctica se implementaron los pasos que componen el framework de Scrum, con ligeras modificaciones. A continuación, se describe cómo se implementaron cada uno de los pasos al desarrollo de la práctica.

1. Para el desarrollo del *product backlog* inicialmente se dio la oportunidad a los practicantes de intentar hacer una lista de todas las características que debería

tener el producto, con su respectiva descripción, prioridad y complejidad. Al no tener muy clara la idea del proyecto a desarrollar, hubo múltiples características faltantes, por lo que este proceso fue iterado por uno de los profesionales de la empresa para así tener el *product backlog* lo más completo posible.

2. El *product owner* fue el encargado del planteamiento de los *sprints* de forma independiente y se encargó de subir cada sprint a la plataforma de Jira.
3. Durante la práctica no se realizaron reuniones específicas para los *sprints*. La diferencia es que se realizaban reuniones semanales en donde todos los miembros del equipo comentaban lo que se logró y lo que quedó faltando durante la semana, así como también dudas que se tuvieran sobre algún sprint o tarea. En algunos casos, el *product owner* podría considerar agregar nuevas tareas dependiendo de cómo se vaya desarrollando el sprint.
4. Al igual que con el planteamiento de los *sprints*, el *product owner* se encargó de los *sprint backlogs*. Con esto, posteriormente, se agregaron las diferentes tareas pertenecientes a los diferentes *sprints* en la plataforma de Jira.
5. En vez de realizar las reuniones diarias, en la empresa, cada día al empezar y finalizar jornada, se escribe un mensaje por el grupo de Skype. Al iniciar se comenta con el equipo qué se hará durante la jornada y al acabar se menciona qué se logró, qué cosas quedaron faltando y qué dificultades se tuvo durante la jornada. De este modo, todos los miembros del equipo saben en qué están trabajando los demás y qué dificultades están teniendo.
6. Al igual que con las reuniones para los *sprints*, no se tienen reuniones específicas para los *sprint reviews* o para los *sprint retrospectives*. Se utilizaron las reuniones

semanales para determinar en qué momento el *sprint* queda terminado y hacer una recopilación de lo que se hizo durante el *sprint*, así como evaluar el rendimiento durante el *sprint* para ver en qué se puede mejorar. A partir de esto, se decide si es necesario añadir más tareas al *sprint* o seguir con el siguiente.

4.2 Fase de planeación

Con esta fase se dio cumplimiento de los primeros tres objetivos específicos (Ver sección 2.2). En esta fase se definieron los parámetros y requerimientos del módulo de conectividad, los dispositivos que se planean utilizar en la fase de integración, las tecnologías que se planean utilizar y la arquitectura que tiene el API Gateway.

4.2.1 Examinación de fabricantes

La primera tarea de la fase de planeación fue hacer una investigación de los fabricantes de dispositivos *wearables*, glucómetros y tensiómetros más vendidos en el mercado. Estos fabricantes cuentan con dispositivos, los cuales disponen de SDKs para poder conectarlos a la aplicación y acceder a las lecturas de los usuarios, pero algunos de estos también cuentan con aplicaciones de control de salud, de las cuales se pueden obtener las mediciones del usuario por medio de APIs públicas de los fabricantes.

El principal objetivo de esta investigación fue identificar los datos que se pueden obtener por medio de cada API y SDK de los fabricantes para así tener una base homologada con las mediciones brindadas por la mayoría de los fabricantes. A partir de la investigación se obtuvo una lista de dispositivos y aplicaciones de diversos fabricantes con múltiples datos y mediciones brindados por cada uno de ellos.

4.2.1.1 Aplicaciones de control de salud

A continuación, se muestra la lista de aplicaciones que se investigaron con el respectivo nombre de su fabricante:

- IHealth (Apple)
- Samsung Health (Samsung)
- FitBit (FitBit)
- Garmin Connect (Garmin)
- Zepp Life (Xiaomi)
- Huawei Health (Huawei)
- Polar Flow (Polar)
- Google Fit (Google)
- MySugr (Accu-chek)
- Omron Connect (Omron)

Para cada uno de estos fabricantes se hizo una lista de los diferentes tipos de información que se registran dentro de cada aplicación y que se pueden obtener por medio de una API. Con esta información, se creó la siguiente tabla, en donde la X representa que la información se encuentra dentro de la aplicación del fabricante.

Tabla 1

Información presente en las aplicaciones de salud de los fabricantes

	Apple	Samsung	FitBit	Garmin	Xiaomi	Huawei	Polar	Google	mySugr	Omron Connect
Género	X	X	X	X	X	X	X	X	X	X
Tipo de sangre	X									
Fecha de nacimiento	X	X	X	X	X	X	X	X	X	X
Altura	X	X	X	X	X	X	X	X		X

Peso	X	X	X	X	X	X	X	X		X
BMI	X	X	X	X	X	X	X	X		
Porcentaje de grasa	X	X	X	X				X		
Circunferencia de la cintura	X	X								
Ritmo cardiaco	X	X	X	X	X	X	X	X		X
Ritmo cardiaco en reposo	X	X	X	X	X	X	X	X		
Promedio de ritmo cardiaco caminando	X	X	X	X	X	X	X	X		
Saturación de oxígeno	X	X		X	X	X		X		
Temperatura del cuerpo	X	X			X	X		X		
Presión sistólica	X	X			X	X		X		X
Presión diastólica	X	X			X	X		X		X
Frecuencia respiratoria	X			X				X		
Cantidad de alcohol en la sangre	X									
Nivel de glucosa en la sangre	X	X			X	X		X	X	
Cantidad de pasos diarios	X	X	X	X	X	X	X	X		
Distancia recorrida caminando/corriendo	X	X	X	X	X	X	X	X		
Uso de silla de ruedas	X									
Calorías quemadas	X	X	X	X	X	X	X	X		
Tiempo de inicio de sueño	X	X	X	X	X	X	X	X		
Tiempo de fin de sueño	X	X	X	X	X	X	X	X		
Tiempo de sueño ligero		X	X	X	X	X	X	X		
Tiempo de sueño profundo		X	X	X	X	X	X	X		
Tiempo de sueño REM		X	X	X	X	X	X	X		
Tipo de diabetes									X	
Terapia de diabetes (Bolo, bomba, sin)									X	
PAI					X					

4.2.1.2 Dispositivos

Al igual que con las aplicaciones de control de salud, se hizo un seguimiento de los dispositivos más comunes en el mercado y se realizó una revisión de las mediciones que cada uno de estos dispositivos registraba. A continuación, se muestra la lista de los dispositivos investigados junto con el nombre del fabricante de este.

- Apple Watch (Apple)
- Samsung Galaxy Watch / Fit (Samsung)
- FitBit Smartwatch / Tracker (FitBit)
- Garmin Watch (Garmin)
- Mi Band, Amazfit GTR, Amazfit GTS, Amazfit Neo, Amazfit Bip (Xiaomi)
- Huawei Band / Watch (Huawei)
- Polar Watch (Polar)
- Accu-chek Glucometer (Roche)
- Contour Glucometer (Bayer)
- GlucoQuick Glucometer (GlucoQuick)
- Abbott Glucometer (Abbott)
- Omron BP Monitor (Omron)
- Microlife BP Monitor (Microlife)

Para cada uno de estos dispositivos se investigó la lista de mediciones que pueden registrar para desarrollar las siguientes tablas. La tabla 2 corresponde a las mediciones de dispositivos *wearables*, como lo son smartbands y smartwatches, mientras que la tabla 3 corresponde a las mediciones de glucómetros y tensiómetros digitales. Para ambas tablas, la letra X significa que

cualquier versión del dispositivo cuenta con esta medición, mientras que el nombre indica que esta medición solo está disponible para esa versión del dispositivo.

Tabla 2

Información presente en los dispositivos wearables

	Apple	Samsung	FitBit	Garmin	Mi Band	GTR	GTS	Neo	Bip	Huawei	Polar
Cantidad de pasos diarios	X	X	X	X	X	X	X	X	X	X	X
Distancia recorrida caminando/corriendo	X	X	X	X	X	X	X	X	X	X	X
Calorías quemadas	X	X	X	X	X	X	X	X	X	X	X
Tiempo de inicio de sueño	X	X	X	X	X	X	X	X	X	X	X
Tiempo de fin de sueño	X	X	X	X	X	X	X	X	X	X	X
Tiempo de sueño ligero		X	X	X	X	X	X	X	X	X	X
Tiempo de sueño profundo		X	X	X	X	X	X	X	X	X	X
Tiempo de sueño REM		X	X	X	5	2+	2+	X	U series	X	X
Ritmo cardiaco	X	X	X	X	X	X	X	X	X	X	X
Ritmo cardiaco en reposo	X	X	X	X	X	X	X	X	X	X	X
Promedio de ritmo cardiaco	X	X	X	X	X	X	X	X	X	X	X
Frecuencia respiratoria	X			X		3+	3+				
Saturación de oxígeno	Series 6	4 Series				2+	2+		U series	X	
Porcentaje de grasa		4 Series									
PAI					X	X	X	X	U series		

Tabla 3

Información presente en los glucómetros y tensiómetros digitales

	Accu-chek glucometer (Roche)	Contour glucometer (Bayer)	GlucoQuick glucometer	Abbott glucometer	Omron BP Monitor	Microlife BP Monitor
Nivel de glucosa en la sangre	X	X	X	X		
Presión sistólica					X	X
Presión diastólica					X	X

A partir de la información recolectada, se definieron las mediciones que se tendrán en cuenta para el desarrollo de Whatoko Health, teniendo en cuenta cuáles mediciones se pueden obtener de todos o casi todos los fabricantes y cuáles son indispensables para el desarrollo del proyecto. Estas mediciones se organizaron en los siguientes grupos:

- Mediciones de actividad: Calorías, Distancia, Pasos y Sesiones de ejercicio.
- Mediciones de sangre: Glucosa, Presión, Saturación de oxígeno y Colesterol.
- Mediciones de composición corporal: Porcentaje de grasa, Altura, Peso y Temperatura corporal.
- Mediciones de pulso: Pulso cardíaco y Pulso cardíaco en reposo.
- Mediciones de sueño: Ciclos del sueño.

Como se puede ver en las mediciones de sangre, se tiene contemplado el colesterol a pesar de que ningún fabricante otorga ese dato, sin embargo, es un dato que es importante para la aplicación y éste se obtiene por medio de un formulario en la aplicación que debe llenar manualmente el usuario. Además, a parte de las mediciones, de los fabricantes se puede obtener información relacionada con el usuario que también es importante para Whatoko Health como lo es información técnica del dispositivo e información básica del usuario como género, fecha de nacimiento, etc.

4.2.2 Definición de la arquitectura del proyecto

Con el fin de abordar todo el flujo de información, desde la obtención de los datos hasta el registro de los datos estructurados en la base de datos, es necesario acompañar el desarrollo del API Gateway con el desarrollo de un microservicio de fabricantes. Para poder definir la arquitectura y el flujo de información que tiene el proyecto, primero es necesario definir las características y requerimientos que necesitan tener tanto el API Gateway como el microservicio de fabricantes, según las necesidades del proyecto. Estos requerimientos son:

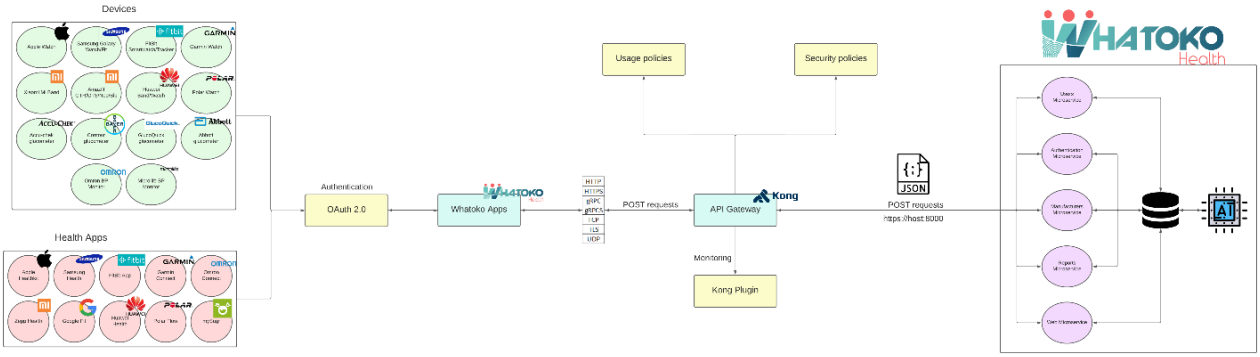
- El API Gateway debe recibir todas las llamadas API REST y HTTP de los diferentes microservicios del proyecto y unificar el flujo de información.
- El API Gateway debe recibir las mediciones que van hacia el microservicio de fabricantes, estructurarlas con modelos predefinidos y enviar los datos estructurados al microservicio.
- El API Gateway debe tener políticas de seguridad para evitar que se corrompa su funcionamiento o la información que pasa por esta.
- El API Gateway debe tener una herramienta de monitoreo para revisar estadísticas sobre la cantidad de llamadas e información que pasa por esta.
- El microservicio de fabricantes debe tener rutas para el flujo de información de las diferentes mediciones.
- Cada una de las rutas del microservicio de fabricantes debe validar la información entrante.
- El microservicio de fabricantes debe clasificar la información validada e insertarla a la base de datos.

A partir de estos requerimientos y de las aplicaciones y dispositivos encontrados en la

sección anterior, se puede realizar un diagrama de la arquitectura y flujo de información de todo el proyecto.

Figura 4

Diagrama de flujo de información de Whatoko Health



Como se puede ver en el diagrama, el flujo de información inicia con los diferentes dispositivos y aplicaciones que se seleccionaron. La aplicación móvil de Whatoko obtiene la información del usuario proveniente de estas dos fuentes por medio de un proceso de autenticación con OAuth 2.0. Además, una tercera fuente de información es la posibilidad de que el usuario ingrese información manualmente en la aplicación móvil de Whatoko. La información proveniente de estas tres fuentes se dirige hacia el API Gateway, en donde se realiza un proceso de estructuración y verificación de los datos para adecuarlos según modelos predefinidos que sean de utilidad para Whatoko. Posteriormente esta información es enviada al microservicio de fabricantes, en donde se clasifican los datos y se ingresan a la base de datos para que los demás servicios puedan consumirlos.

4.2.3 Definición de tecnologías

Para llevar a cabo el desarrollo del proyecto de una manera adecuada, es necesario definir las mejores tecnologías para desarrollarlo. A continuación, se especifican las tecnologías que se utilizaron.

4.2.3.1 API Gateway

El API Gateway es lo más importante del desarrollo, por lo que es necesario elegir una herramienta que pueda cumplir con los requerimientos establecidos. Con el fin de elegir la mejor herramienta, se desarrollaron dos pruebas de concepto. Una prueba de concepto es la puesta a prueba de una herramienta para evaluar su viabilidad en la incorporación al proyecto. Se evaluaron las herramientas Express Gateway y Kong Gateway.

La primera se terminó descartando debido a que lleva más de 3 años sin actualizaciones y los desarrolladores recomendaron no utilizarla más. Kong Gateway es un API Gateway optimizada para microservicios y arquitecturas distribuidas y se concluyó que esta cuenta con las herramientas que se necesitan para cumplir los requerimientos del proyecto, sin embargo, la versión gratuita tiene algunas limitaciones en cuanto a cantidad de llamadas mensuales y retención de los datos que, en caso de que el proyecto crezca mucho en el futuro, sería necesario adquirir la versión de pago. Para el desarrollo de este proyecto se utilizó la versión gratuita ya que brindaba lo suficiente para el alcance que se tiene.

Uno de los requerimientos más importantes del API Gateway es que estructure los datos de las mediciones provenientes de los dispositivos y los envíe al microservicio de fabricantes. La mejor forma de realizar esto con Kong Gateway es creando un plugin personalizado que realice la estructuración de los datos.

4.2.3.2 Lenguaje de programación

Javascript es un lenguaje de programación orientado a objetos que permite implementar funciones especiales en páginas web. Typescript es un lenguaje de programación que se basa en javascript. Las principales ventajas de utilizar typescript es que este agrega sintaxis adicional a javascript para una mejor integración con el editor de código, el código en typescript se convierte automáticamente en javascript al correr la aplicación y typescript utiliza inferencia de tipos para brindar mejores herramientas sin código adicional (Typescript, s.f.).

El ecosistema de typescript y javascript tiene múltiples herramientas y librerías para ayudar al desarrollador. Una de estas es la librería Express, la cual facilita la configuración de una API al crear un servidor. Como el proyecto cuenta con múltiples microservicios, en donde la mayoría cuenta con una API internamente, utilizar este ecosistema facilitaría las cosas al momento del desarrollo de estos microservicios. Además, como se mencionó en la sección anterior, para el API Gateway se requiere el desarrollo de un plugin personalizado y Kong permite desarrollar plugins en únicamente 4 lenguajes: lua, go, python y javascript, con soporte directo de typescript. Por estas dos razones se utilizó el lenguaje de programación typescript para el desarrollo del proyecto.

4.2.3.3 Control de versiones

Normalmente los controladores de versión son similares en las funciones básicas al momento de llevar la trazabilidad de un proyecto, sin embargo, hay algunas opciones que cuentan con otras herramientas. En este caso se eligió GitLab como controlador de versiones ya que ésta cuenta con varias herramientas que facilitan la incorporación de DevOps al proyecto. Un ejemplo de esto son las herramientas de CI/CD (Integración y entrega continuas, por sus siglas en inglés), tales como los *pipelines*. Un *pipeline* sirve para automatizar el proceso de entrega de software, creando código, ejecutando pruebas e implementa de manera segura una nueva versión de la

aplicación cada vez que ocurren cambios. El uso de *pipelines* se utilizó principalmente en el repositorio del *backend* del proyecto, en donde se encuentran los diferentes microservicios que lo componen. Adicionalmente se tiene un repositorio con el desarrollo y despliegue del API Gateway y su plugin personalizado.

4.2.3.4 Documentación

Read The Docs es una herramienta que permite automatizar la creación, control de versiones y alojamiento de la documentación de un proyecto. Se eligió esta herramienta para la documentación del API Gateway debido a la facilidad de utilizar y por el hecho de que esta herramienta es gratuita. La documentación se vincula con el repositorio de GitLab donde está alojado el proyecto del API Gateway y cada vez que hay un nuevo *commit*, la documentación se actualiza automáticamente con los cambios.

Para la documentación del microservicio de fabricantes y los demás microservicios que se encuentran en el *backend* del proyecto, se utilizó una herramienta llamada Compodoc, la cual se utiliza para generar documentación estática de una aplicación. Esta documentación se crea automáticamente a partir de los comentarios realizados en el código durante el desarrollo. Además, esta herramienta cuenta con una página dentro de la documentación que muestra la cobertura, es decir, qué porcentaje de archivos cuentan con los comentarios adecuados para realizar la documentación. Esto resulta de utilidad para determinar en qué archivos hacen falta comentarios para poder agregarlos y tener una documentación completa.

4.2.3.5 Despliegue

Para desplegar tanto el API Gateway como el microservicio de fabricantes se utilizaron contenedores de Docker, una herramienta utilizada para automatizar el despliegue de aplicaciones. Un contenedor es un entorno portable que contiene todo lo que una aplicación necesita para correr

y realizar su trabajo, tales como dependencias y archivos de configuración. El uso de Docker resulta especialmente útil gracias a su portabilidad, ya que solo es necesario codificar los archivos para desplegar la aplicación y este despliegue se puede replicar en los computadores de los demás miembros del equipo sin pasos adicionales.

Docker se suele utilizar en conjunto con una herramienta llamada Kubernetes, la cual se utiliza principalmente para llevar control de contenedores. Para el despliegue de Whatoko Health, el líder técnico del equipo se encargó de utilizar Kubernetes en un servidor en la nube para orquestar todos los contenedores que componen el proyecto y que la aplicación y todo lo que la compone quede corriendo en un servidor.

4.3 Fase de desarrollo

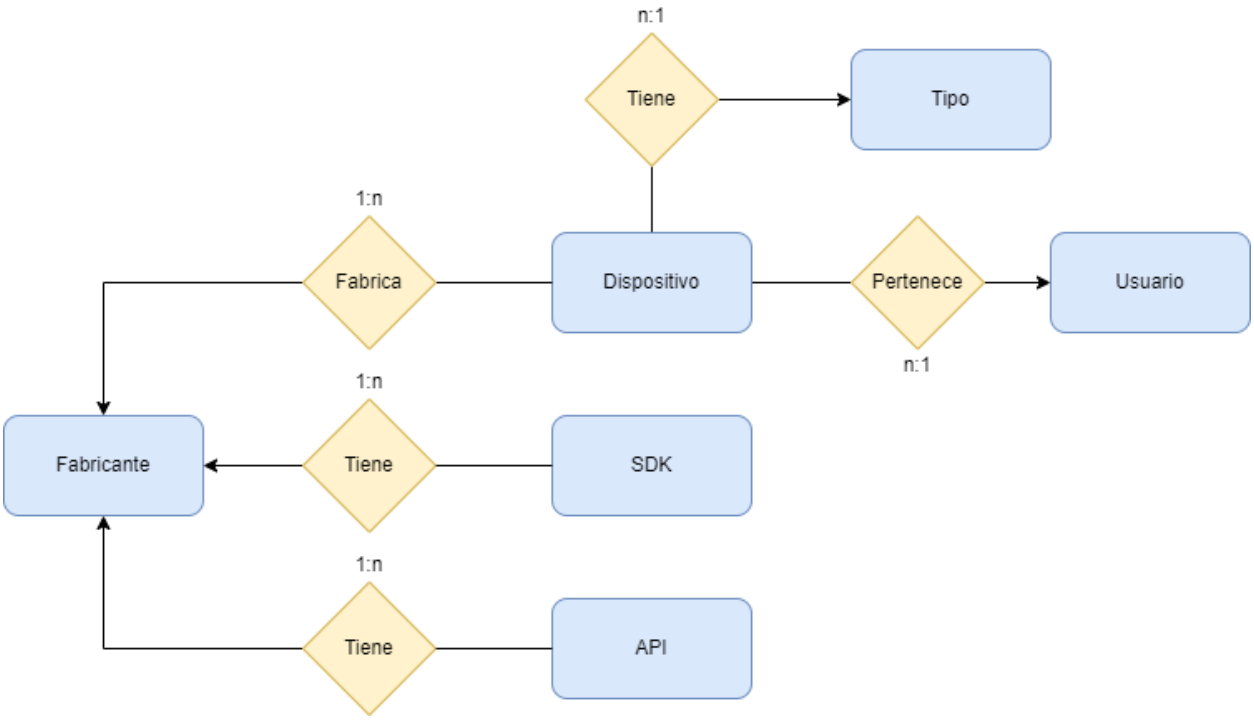
Con esta fase se da cumplimiento del cuarto y quinto objetivos específicos (Ver sección 2.2). En esta fase se realizó el desarrollo de la API de Whatoko Health y su plugin personalizado, el microservicio de fabricantes y el diseño de la base de datos de los fabricantes de dispositivos.

4.3.1 Diseño de base de datos

Whatoko Health cuenta con una extensa base de datos con información sobre usuarios, enfermedades, fabricantes de dispositivos, entre otras cosas. En esta sección se habla del desarrollo de la parte de la base de datos relacionada a los dispositivos y sus fabricantes. A continuación, se muestra el modelo entidad relación y su explicación.

Figura 5

Modelo entidad relación de la base de datos de fabricantes



Las tablas de SDK y API se utilizan principalmente para almacenar información técnica como URLs, versión, protocolos de comunicación, entre otros. Estas tablas se relacionan directamente con el fabricante, es decir que cada API y SDK corresponde a un único fabricante registrado, pero un fabricante puede contar con múltiples SDKs y APIs.

La tabla de dispositivos funciona de manera similar, en donde un dispositivo es fabricado por un único fabricante, pero este puede fabricar múltiples dispositivos. En esta tabla se guarda información técnica del dispositivo, como nombre, modelo y versión, pero también hay datos sobre la conexión con la aplicación, tales como la fecha de la primera conexión y el estado actual de la conexión. Un dispositivo tiene un único tipo, el cual puede ser *smartband*, tensiómetro, glucómetro o báscula. Toda esta sección de la base de datos relacionada con los fabricantes se conecta con el resto por medio de la relación entre el dispositivo y el usuario. Un dispositivo pertenece únicamente a un usuario, pero el usuario tiene la opción de conectar múltiples dispositivos a la aplicación.

El proceso de diseñar la base de datos se llevó a cabo utilizando las herramientas LucidChart, una herramienta para diseñar diagramas, y MYSQL Workbench, una herramienta útil para diseñar bases de datos. Este fue un proceso iterativo en donde el líder técnico del proyecto iba revisando el diseño y dando retroalimentación para cada vez tener una versión mejor. Inicialmente se tenían muchos errores y ambigüedades debido a la falta de comprensión del problema, pero a medida que se fue iterando y se fue teniendo una idea más clara del contexto y de la solución del problema, fue más sencillo llegar a un diseño de base de datos estable y útil para Whatoko Health.

4.3.2 API Gateway

Como se mencionó anteriormente, las principales tareas del API Gateway son unificar el flujo de información hacia todos los servicios y estructurar los datos de las mediciones que se dirigen hacia el microservicio de fabricantes, para esto se optó por utilizar Kong Gateway como API Gateway para Whatoko Health. Esta herramienta se puede instalar en múltiples entornos y tiene diferentes formas de instalación. En este caso se optó por utilizar Docker, una herramienta que permite automatizar el despliegue de aplicaciones en contenedores, y en este entorno, Kong se puede instalar de dos formas: con una base de datos PostgresSQL en donde se guarda toda la configuración, o sin base de datos. En el segundo caso es necesario utilizar un archivo de configuración dentro del proyecto y fue la opción que se utilizó para el desarrollo del proyecto.

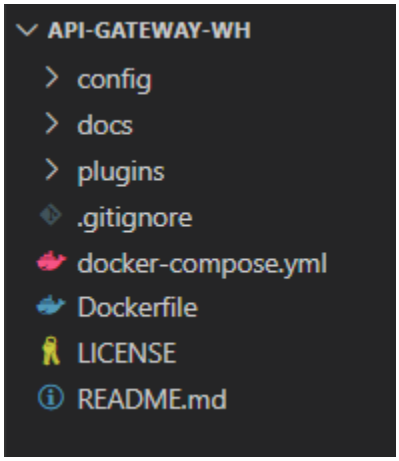
El proyecto del API Gateway se manejó en un repositorio de GitLab en donde se creaban ramas para cada nueva funcionalidad o corrección y posteriormente se juntaba todo en la rama de desarrollo. A continuación, se revisa la estructura de archivos del proyecto.

4.3.2.1 Estructura de carpetas

La estructura de archivos del API Gateway se basó en un entorno de desarrollo proporcionado por Kong para la creación de plugins personalizados (Kong Inc., 2022). En la siguiente figura se muestran los archivos y carpetas que componen el proyecto del API Gateway.

Figura 6

Estructura de archivos del API Gateway



4.3.2.1.1 Archivos Dockerfile y docker-compose

Estos archivos se utilizan para automatizar el despliegue de la aplicación en un contenedor de Docker. El archivo Dockerfile se utiliza para crear una imagen del proyecto, la cual es compuesta por una serie de instrucciones para poder desplegar el proyecto. Este archivo se encarga de instalar múltiples dependencias que tiene el proyecto, una de estas siendo el kit de desarrollo de plugins de Kong. El archivo docker-compose se encarga de crear el contenedor utilizando de base el archivo Dockerfile para crear la imagen que va dentro del contenedor. En este archivo también se establecen múltiples variables de entorno necesarias para que el kit de desarrollo de plugins de Kong funcione de manera correcta. Por último, se especifican los puertos que se deben

exponer para poder acceder a la aplicación, en este caso se utiliza el puerto 8000 para realizar las peticiones HTTP y el puerto 8001 como una API para consultar información interna de Kong.

4.3.2.1.2 Carpeta config

En la carpeta config se encuentra un único archivo llamado kong.yml. En este archivo se guarda toda la configuración de Kong con respecto a las entidades, tales como servicios, rutas y plugins. En este archivo se declaran cada uno de los servicios que componen a Whatoko Health, incluyendo el microservicio de fabricantes. Para cada servicio se especifica el nombre, la url (esta debe contener el protocolo de comunicación, el puerto y la ruta), los nombres de los plugins que se desean aplicar al servicio (en este caso únicamente se aplica el plugin desarrollado a todos los servicios) y las rutas. Este archivo de configuración permite que se creen los servicios al momento de desplegar Kong, lo cual permite unificar el flujo de información de todos los servicios para que se pueda acceder a todos desde el puerto 8000. Además, se aplica el plugin personalizado para estructurar la información que pasa por el API Gateway.

4.3.2.1.3 Carpeta docs

En la carpeta docs se encuentran todos los archivos correspondientes a la documentación del API Gateway, la cual se desplegó en Read The Docs.

4.3.2.1.4 Carpeta plugins

En esta carpeta se encuentra el plugin personalizado que se desarrolló. Hay una carpeta llamada manufacturer, en donde se encuentra el desarrollo del plugin. En caso de, en el futuro, querer desarrollar otro plugin, solo basta con crear una nueva carpeta con el nombre del plugin dentro de esta carpeta.

4.3.2.2 Desarrollo del plugin personalizado de Kong

La principal tarea del plugin personalizado de Kong es obtener el cuerpo de la solicitud, en donde vienen los datos de una medición, estructurar este cuerpo según un modelo de datos para cada tipo de medición y después reemplazar el cuerpo de la solicitud con el cuerpo estructurado. Teniendo en cuenta esto, el primer paso fue definir los modelos de datos para cada uno de los tipos de medición que se tuvieron en cuenta.

4.3.2.2.1 Modelos de datos de mediciones

Para definir los modelos inicialmente se recurrió a la información recolectada en la fase de planeación (Ver sección 4.2.1), en donde se seleccionaron las mediciones y se organizaron en grupos. Para cada una de estas mediciones fue necesario identificar las partes que componen su respectivo modelo, es decir, la información que va dentro del objeto que va en el cuerpo de la solicitud HTTP. A continuación, se mencionan las diferentes mediciones y la información que compone a cada una:

- Calorías: Esta medición está compuesta por fecha inicial, fecha final y valor de las calorías quemadas en ese período de tiempo.
- Distancia: Esta medición está compuesta por fecha inicial, fecha final y valor de distancia recorrida en metros en ese período de tiempo.
- Pasos: Esta medición está compuesta por fecha inicial, fecha final y cantidad de pasos dados en ese período de tiempo.
- Sesión de ejercicio: Esta medición está compuesta por fecha inicial, fecha final, nombre, descripción, distancia recorrida y calorías quemadas en ese período de tiempo y tipo de actividad.

- Glucosa: Esta medición está compuesta por fecha, nivel de glucosa en mg/dL, tiempo con respecto a la última comida, tipo de comida ingerida alrededor del momento de la medición, tiempo con respecto a la última vez que durmió y tipo de fluido corporal que se utilizó para medir el nivel de glucosa.
- Presión: Esta medición está compuesta por fecha, valor de presión sistólica y valor de presión diastólica.
- Saturación de oxígeno: Esta medición está compuesta por fecha y porcentaje de saturación de oxígeno en la sangre.
- Colesterol: Esta medición está compuesta por fecha y niveles de colesterol total, LDL, HDL y triglicéridos en mg/dL.
- Porcentaje de grasa: Esta medición está compuesta por fecha y porcentaje de grasa.
- Altura: Esta medición está compuesta por fecha y altura en centímetros.
- Peso: Esta medición está compuesta por fecha y peso en kilogramos.
- Temperatura corporal: Esta medición está compuesta por fecha y temperatura en grados Celsius.
- Pulso cardíaco y pulso cardíaco en reposo: Estas mediciones están compuestas por fecha y valor de pulso cardíaco.
- Sueño: Esta medición está compuesta por fecha inicial, fecha final y nombre de la etapa del sueño correspondiente al período.
- Usuario: Este modelo se compone de género, tipo de sangre, fecha de nacimiento y correo electrónico.

- Dispositivo: Este modelo se compone de id, nombre, modelo, fecha de primera conexión, variable que indica si el dispositivo está conectado o no y la descripción del estado de la conexión.

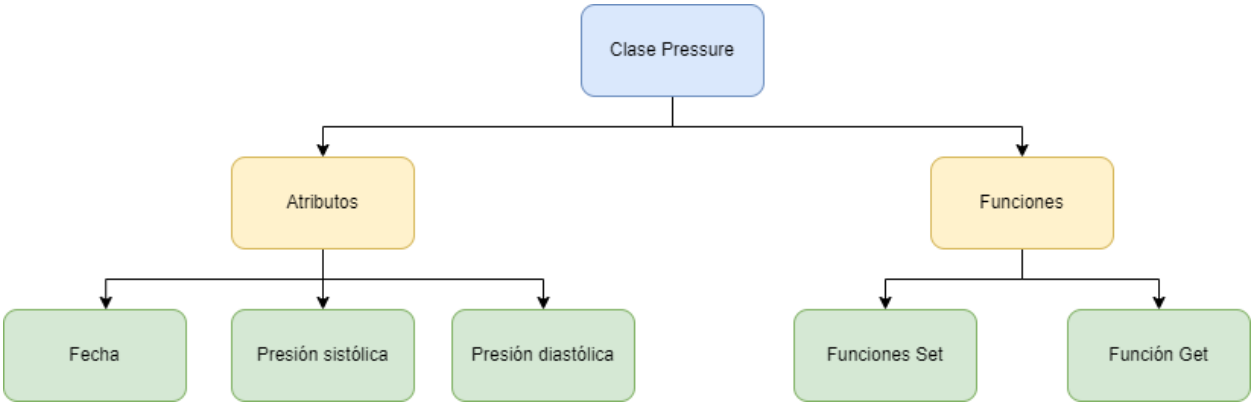
Hay que tener en cuenta que cada uno de estos modelos tiene su respectiva ruta en el microservicio de fabricantes en donde se hace la inserción de los datos en la base de datos. Teniendo claro esto y la información que compone cada uno de los modelos, se procede a desarrollar los modelos en Typescript.

4.3.2.2.2 Desarrollo de modelos en Typescript

Para este desarrollo se utilizaron interfaces. Una interfaz se utiliza para definir la estructura de una clase, es decir, esta incluye los métodos y los atributos que tiene la clase, pero no la implementación de estos. Habiendo definido la interfaz, la clase la implementa para que tenga la misma estructura y es acá en donde se define el contenido de las funciones. En la siguiente figura se ve un ejemplo de qué contiene una clase, en este caso para la medición de presión.

Figura 7

Contenido de la clase Pressure



Como se puede observar, los atributos están compuestos por la información que se definió en la sección anterior. A parte de eso, cuenta con dos tipos de funciones. En cuanto a las funciones

Set, hay una por cada atributo que tenga la clase y estas se encargan de darle un valor a cada atributo una vez se inicializa un objeto de esta clase. La función *Get* se debe llamar una vez todos los atributos tengan valor y esta se encarga de retornar un objeto con la información estructurada, la cual varía dependiendo del modelo.

Por último, para acabar con el desarrollo de los modelos es necesario contar con un controlador para cada modelo, el cual se encarga de crear una instancia de la clase correspondiente al modelo, utilizar las funciones set para establecer los valores del objeto y finalmente retornar el valor que retorna la función get, es decir, la información estructurada. Además, hay que tener en cuenta que una solicitud siempre debe ir acompañada del id del usuario y del id del dispositivo, por lo que es necesario que el controlador también se encargue de que esta información se mantenga intacta y esté en la primera posición del arreglo que retorna el controlador, como se ve en la siguiente figura.

Figura 8

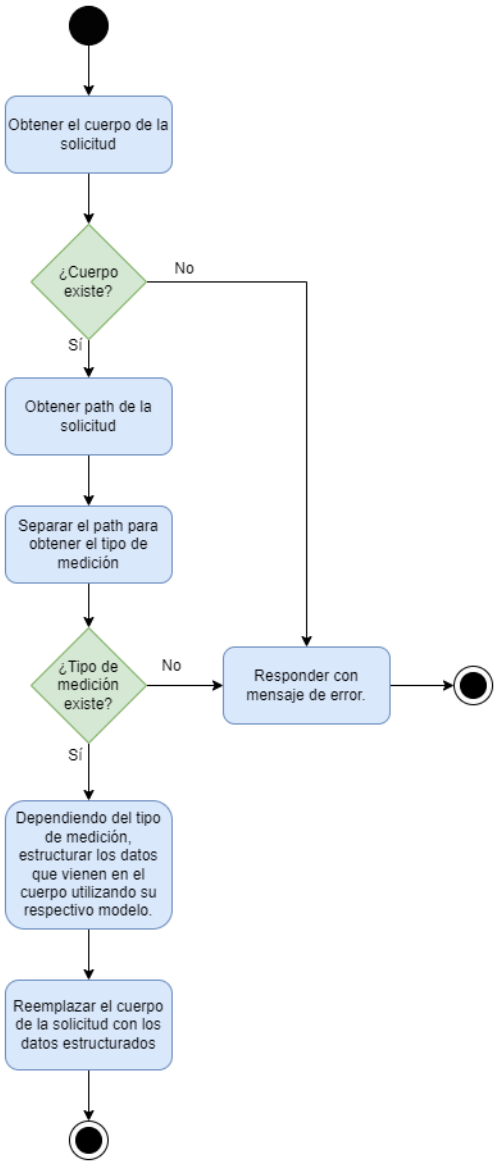
Medición de presión estructurada

4.3.2.2.3 Desarrollo del plugin personalizado

Una vez se tienen definidos y desarrollados los modelos para cada medición, se procede al desarrollo del plugin, cuyo principal propósito es obtener el cuerpo de la solicitud HTTP que pasa por el API Gateway, estructurar la información con los modelos y reemplazar el cuerpo de la solicitud con la información estructurada, este flujo se ve representado en la siguiente figura.

Figura 9

Diagrama de flujo del plugin personalizado de Kong



Para lograr este flujo, se hace uso de algunas características que tiene el *plugin developer kit* de Kong, específicamente los manejadores de fase y las funciones PDK. Los manejadores de fase son funciones que permiten implementar una lógica personalizada para que se ejecute en un punto específico del ciclo de vida del procesamiento de solicitudes. Los manejadores que existen son: *certificate*, *rewrite*, *access*, *response*, *preread* y *log*. En este caso, el que resulta de mayor utilidad es el manejador access, el cual se ejecuta para cada solicitud de un cliente antes de que se transmita al servicio, en este caso sería el microservicio de fabricantes. Este manejador es el adecuado porque permite modificar el cuerpo de la solicitud antes de que continúe su flujo. Por otro lado, están las funciones PDK, las cuales pueden ser llamadas dentro de los manejadores. Estas funciones permiten acceder y modificar características de las solicitudes, como por ejemplo obtener y/o reemplazar el *body* de la solicitud (Kong Inc., s.f.).

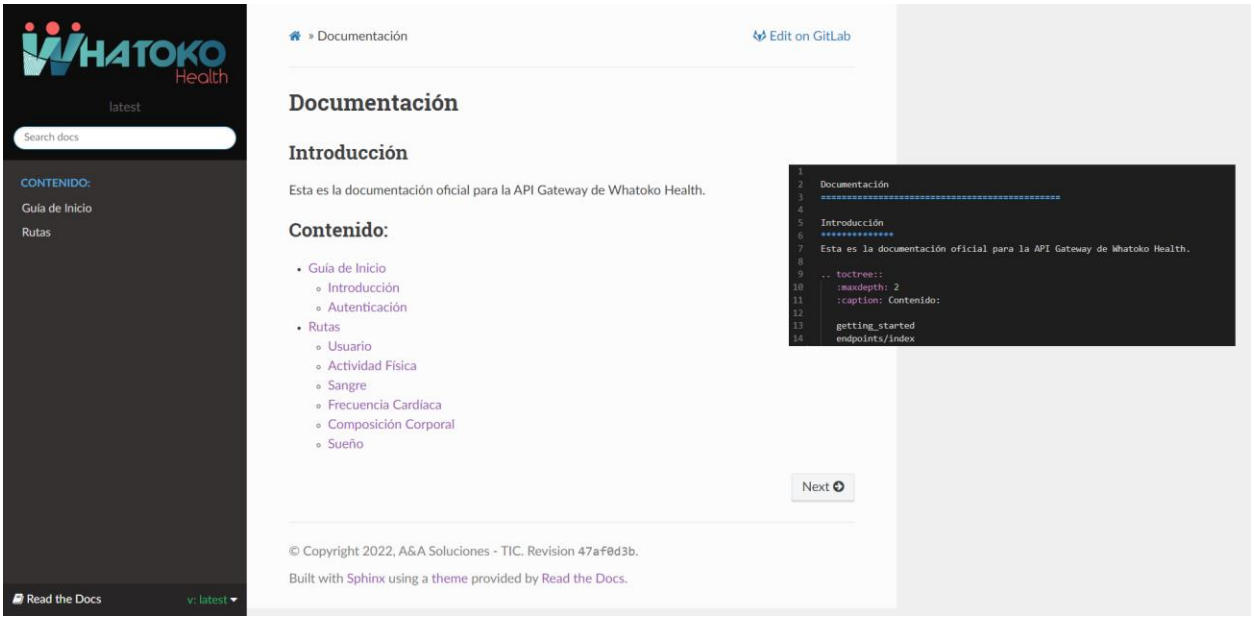
Como se ve en la figura 9, el flujo comienza obteniendo el cuerpo de la solicitud, lo cual se puede lograr utilizando la función PDK getRawBody(). Después, en caso de que la solicitud sí tenga un cuerpo adjuntado a ella, se procede a obtener el *path* o la ruta, la cual se obtiene utilizando la función PDK getPath(). A partir de esta ruta se puede obtener el tipo de medición debido a la forma en como están configuradas las rutas en Whatoko Health. Por ejemplo, la ruta para la medición de presión es “api/v1/manufacturers/pressure”. De acá se puede observar que la última parte de la ruta siempre contiene el tipo de medición. Una vez se tiene tanto el tipo de medición como el cuerpo de la solicitud, se crea una instancia de la clase correspondiente al tipo de medición y se llama la función que se encarga de estructurar los datos. Finalmente, teniendo ya el nuevo cuerpo de la solicitud, este se reemplaza utilizando la función PDK setRawBody(). Con este flujo, se garantiza que la información que llega a los servicios de Whatoko está estructurada de la manera como se requiere.

4.3.2.3 Desarrollo de la documentación del API Gateway

Para desarrollar la documentación del API Gateway se utilizó una herramienta llamada Read The Docs. Para empezar a utilizar esta herramienta primero es necesario crear una carpeta llamada docs en el proyecto e instalar Sphinx, que es la principal herramienta que utiliza Read The Docs para generar la documentación. Una vez teniendo esto instalado, se utiliza el comando "sphinx-quickstart" para generar los archivos iniciales de la documentación. Sphinx utiliza un lenguaje de marcado llamado reStructuredText, en este lenguaje se escribieron las diferentes páginas que se requirieron para la documentación. En la siguiente figura se ve un ejemplo de la página inicial de la documentación junto con el código correspondiente a la página.

Figura 10

Página principal de la documentación



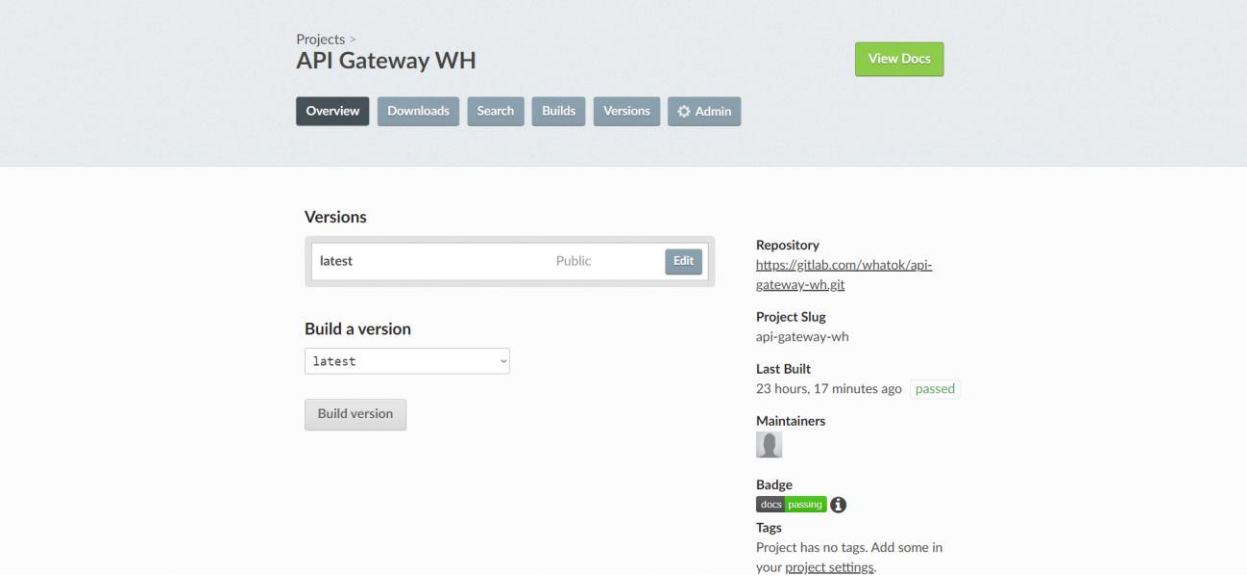
Como se puede observar, la página cuenta con varias cosas que no se especifican en el código. Esto es porque en el archivo de configuración de la documentación se especifica

información como el nombre del proyecto, autor, copyright, plantilla, entre otros. En este caso se utilizó la plantilla de Sphinx que utiliza Read The Docs.

Una vez el código está en el repositorio del proyecto, solo hace falta crear una cuenta en la página de Read The Docs y enlazarla con la cuenta del controlador de versiones que se esté utilizando para el proyecto, en este caso fue GitLab. Cuando las cuentas estén enlazadas, al momento de crear un nuevo proyecto en Read The Docs, se tiene la opción de importar cualquier proyecto público que se encuentre en la cuenta que se enlazó. Una vez el proyecto es importado correctamente, se tiene una pantalla de inicio como se ve en la siguiente figura.

Figura 11

Proyecto del API Gateway en Read The Docs

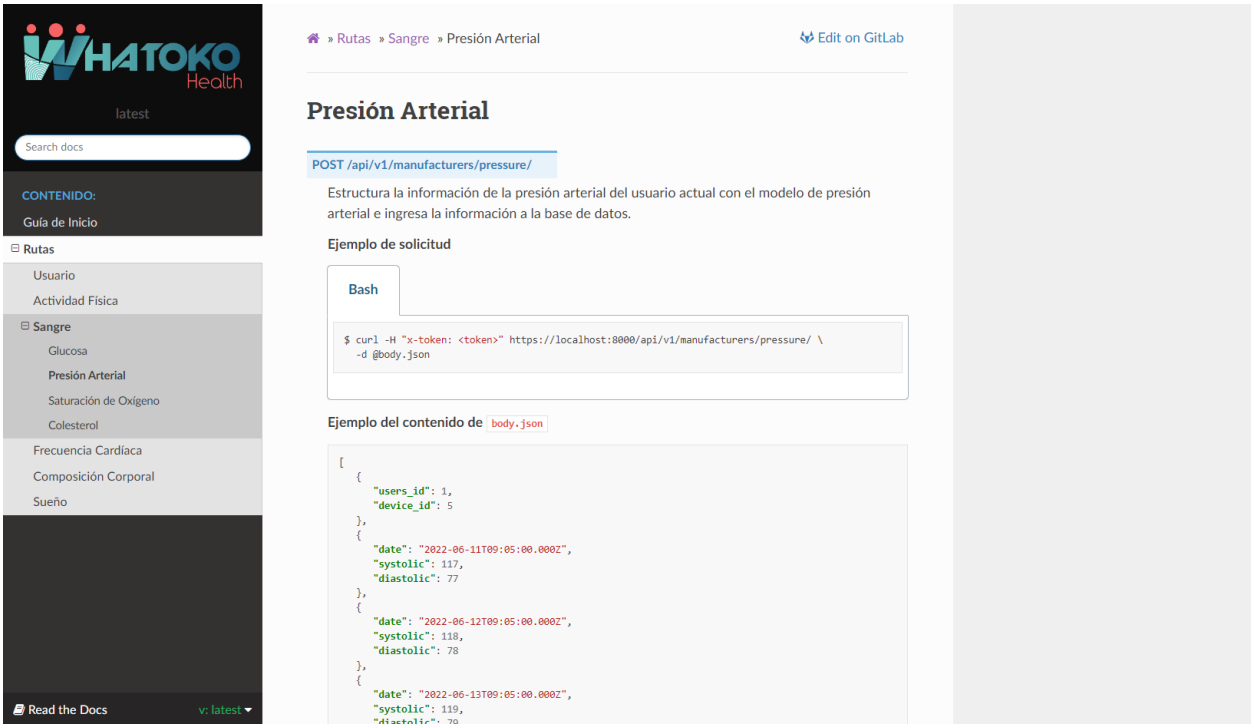


En esta pantalla se pueden ver las versiones del proyecto, se puede construir una nueva versión del proyecto, se pueden ver y modificar las configuraciones del proyecto, se tiene el enlace del repositorio que se utilizó y se puede visitar la documentación ya desplegada con el botón que

dice “View Docs”. En la siguiente figura se tiene una página de la documentación mostrando la información disponible para las rutas, en este caso para la medición de presión.

Figura 12

Documentación de la medición de presión



Finalmente, Read The Docs permite descargar la documentación utilizando el botón que se ve en la imagen en la esquina inferior izquierda. Esta puede ser cargada en PDF, HTML o Epub.

4.3.3 Microservicio de fabricantes

Como se mencionó anteriormente, las principales tareas del microservicio de fabricantes son validar la información entrante, la cual proviene de la aplicación móvil, clasificarla e insertarla en la base de datos. En esta sección se explica cómo se lograron estos requerimientos. El microservicio de fabricantes se encuentra dentro del proyecto del backend de Whatoko Health y este se manejó en un repositorio de GitLab, en donde se creaban ramas para cada nueva

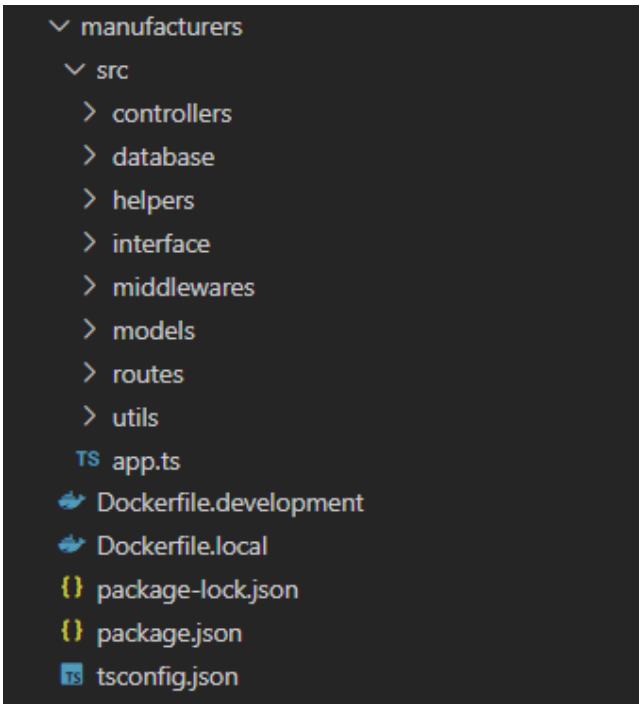
funcionalidad o corrección y posteriormente se juntaba todo en la rama de desarrollo. A continuación, se revisa la estructura de archivos del proyecto.

4.3.3.1 Estructura de carpetas

La estructura de carpetas para el microservicio se basó en el patrón de diseño MVC (Model-View-Controller), el cual se basa en que existen modelos que contienen la estructura de los datos, unas vistas que definen la interfaz del usuario y unos controladores que contienen la lógica del proyecto. Sin embargo, en este caso se modifica un poco el patrón debido a que no se necesita de interfaces de usuario para este microservicio, por lo tanto, solo se implementan modelos y controladores para manejar el flujo de información. A continuación, se revisa la estructura de carpetas del proyecto.

Figura 13

Estructura de archivos microservicio de fabricantes



4.3.3.1.1 Archivos Dockerfile

Se tienen dos archivos Dockerfile para hacer el despliegue del contenedor del microservicio. Uno de los archivos se utiliza para desplegar el contenedor en local y poder desarrollar y hacer cambios y el otro se utiliza para desplegar en el ambiente de desarrollo, que sería el servidor en la nube en donde se aloja el proyecto. En ambos archivos se copia el archivo package.json, se instalan dependencias y se construye y se corre el proyecto, sin embargo, la principal diferencia es que para desarrollo se necesitan especificar las variables de entorno como argumentos.

4.3.3.1.2 Archivos .json

Se tienen tres archivos con extensión json: package-lock.json, package.json y tsconfig.json. Los dos primeros se utilizan para llevar la configuración del proyecto, así como también las dependencias, mientras que el último contiene la configuración que se necesita para poder compilar el código de typescript en javascript al momento de construir el proyecto.

4.3.3.1.3 Carpeta controllers

En la carpeta *controllers* se encuentran los controladores del proyecto, los cuales se encargan de la parte lógica, en este caso son las inserciones a la base de datos. Se tienen 6 archivos, uno para cada tipo de dato: actividad, sangre, composición corporal, dispositivos, frecuencia cardíaca y sueño. Por ejemplo, en el archivo de sangre se tienen 3 funciones para ingresar información a la base de datos, una por cada tipo de medición: presión, saturación de oxígeno y glucosa.

4.3.3.1.4 Carpeta database

En esta carpeta se tiene un único archivo de configuración de la base de datos, el cual toma valores de las variables de ambiente del proyecto, como el usuario y contraseña de la base de datos,

y crea una instancia de la conexión a la base de datos con esta información. Este proceso se realiza utilizando Sequelize y una base de datos MySQL.

4.3.3.1.5 Carpetas helpers, interface y middlewares

Estas tres carpetas contienen archivos para realizar validación en las rutas del microservicio. Una de las validaciones es que se debe tener un token, el cual se obtiene del microservicio de autenticación del proyecto, para poder hacer una petición. También se tienen validaciones para verificar que la información indicada exista y poder manejar mejor los mensajes de error.

4.3.3.1.6 Carpeta models

En esta carpeta se guardan los modelos del proyecto. Hay un modelo para el servidor, el cual contiene información como la ruta predeterminada del servicio y funciones para realizar la conexión con la base de datos, para definir las rutas que se encuentran en la carpeta routes, para iniciar el servidor web y otras más para que el servicio funcione de la manera correcta. Además de ese modelo, se tienen múltiples modelos de tablas de la base de datos para poder hacer las inserciones. Cada uno de estos modelos se extiende de la clase Model de Sequelize, lo cual permite utilizar funciones para hacer inserciones de manera sencilla.

4.3.3.1.7 Carpeta routes

En la carpeta *routes* se encuentran las diferentes rutas del microservicio. Al igual que con los controladores, estas están divididas dependiendo del tipo de dato: actividad, sangre, composición corporal, dispositivo, frecuencia cardíaca y sueño. Dentro de cada archivo se crea una instancia de la clase Router, el cual es un manejador de rutas. En este caso todas las rutas que se tienen son para solicitudes POST. En la declaración de las rutas también se implementan las validaciones que se declararon en las carpetas mencionadas en la sección 4.3.3.1.5, las cuales se

encargan de verificar que el token que viene con la solicitud sea válido y que el cuerpo de la solicitud contenga la información del usuario y del dispositivo.

4.3.3.1.8 Carpeta utils

En esta carpeta se encuentra un único archivo de utilidades para los controladores. En este archivo se tienen algunas funciones para simplificar la inserción de datos, tales como calcular tiempo entre dos fechas, calcular la categoría del biotipo según el bmi del usuario y calcular el porcentaje de grasa a partir del bmi, la edad y el género del usuario.

4.3.3.1.9 Archivo app.ts

Por último, se tiene el archivo que contiene la clase del microservicio de fabricantes. En este archivo se crea una instancia de esta clase y se utiliza el modelo del servidor para inicializar el servidor web y que el servicio quede corriendo.

4.3.3.2 Desarrollo del microservicio de fabricantes

El desarrollo del microservicio inicia como el desarrollo de cualquier API en node, utilizando el comando “npm init”, el cual crea los archivos de configuración y dependencias. Posterior a esto, se creó el archivo de configuración de typescript para poder utilizar este lenguaje en lugar de javascript. Con esto, ya se tienen todos los archivos de configuración para empezar a desarrollar el microservicio.

4.3.3.2.1 Servidor web

El primer paso es gestionar la creación del servidor web y para ello se creó la carpeta de modelos y el modelo del servidor web. Este archivo contiene la definición de la clase Server, la cual cuenta con 3 atributos y 4 funciones. El primer atributo es una variable de tipo Application, la cual se utiliza para gestionar el servidor web, después se tiene una variable de tipo string, la cual contiene el puerto en el que se desplegará el servidor, y por último se tiene una variable de tipo

enum, la cual se encarga de guardar las rutas de la API. En este caso solo se tiene una ruta por defecto "/api/v1/manufacturers". En cuanto a las funciones se tiene una función para gestionar la conexión a la base de datos, una función que establece las rutas del API, una función que inicializa el servidor web y algunas de estas funciones son llamadas en una función que se encarga de la asignación de middlewares. Finalmente, en el constructor de la clase se les asigna valor a los atributos del puerto y de la aplicación y se llaman las funciones de middlewares y de rutas. Teniendo el modelo del servidor creado se procede a crear el archivo principal en la base del proyecto, el archivo app.ts. Acá se declara la clase ManufacturerService, en la cual se crea una instancia del servidor dentro del constructor de la clase y se declara una función para inicializar este servidor web. Finalmente, Se crean instancias de la clase Server y ManufacturerService y se llama la función para inicializar el servidor web.

4.3.3.2.2 Base de datos

Teniendo listo el servidor web, hace falta desarrollar la lógica del microservicio. Para esto se empieza con la configuración de la base de datos, para lo cual se creó un archivo config.ts en una nueva carpeta llamada database. En este archivo se utilizan variables de ambiente para crear y exportar una instancia de la clase Sequelize, la cual se utiliza en el modelo del servidor para gestionar la conexión.

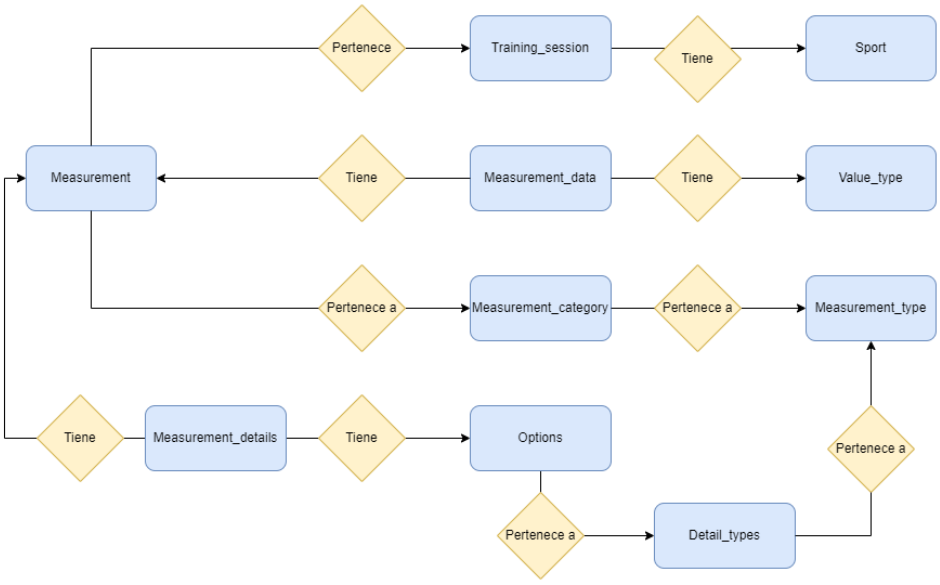
4.3.3.2.3 Rutas, modelos y controladores

Continuando con la lógica del microservicio, se pasa a desarrollar las rutas, modelos y controladores, los cuales se desarrollan de forma síncrona para cada tipo de medición. A continuación, se explica el desarrollo para las mediciones de la categoría de sangre. Inicialmente se crea un archivo en la carpeta de controladores que contiene el controlador para las mediciones de la categoría de sangre. En este archivo se tiene la clase BloodController y dentro de ella existen

3 funciones, cada una sirve para insertar datos de un tipo de medición (presión, glucosa o saturación de oxígeno) a la base de datos. A continuación, se explica cómo se realiza la inserción para la medición de presión, pero primero es necesario saber cómo se estructura la base de datos de Whatoko Health en la parte de mediciones.

Figura 14

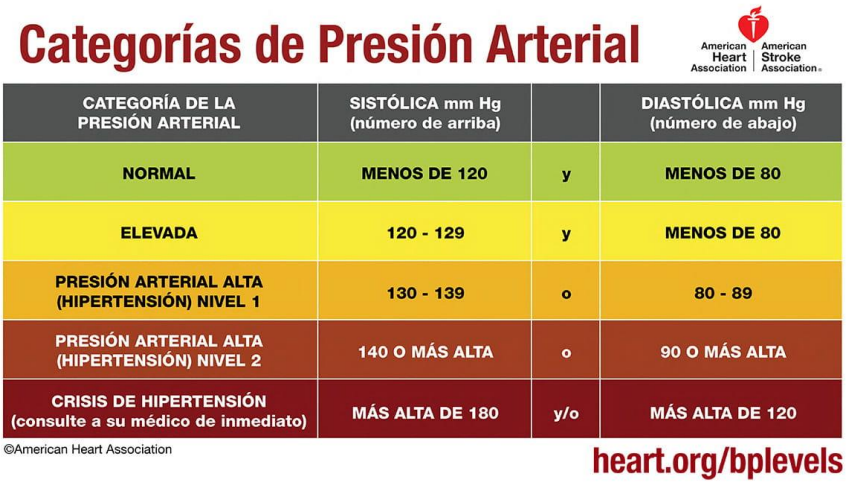
Modelo entidad relación de la sección de mediciones



Para poder insertar datos de presión se empieza buscando el tipo de medición en la tabla Measurement_type, en donde debe existir el campo “tensión arterial”. Si este campo existe, se procede a identificar la categoría de la medición. Para esto se utilizan los valores de presión sistólica y presión diastólica que vienen en el cuerpo de la solicitud, según la figura que se ve a continuación.

Figura 15

Categorías de presión arterial



Nota. Recuperado de: https://www.goredforwomen.org/es/health-topics/high-blood-pressure/understanding-blood-pressure-readings

Una vez se determina en qué categoría está la medición, se verifica que los campos de presión sistólica y presión diastólica existan en la tabla Value_types. Por último, se crea un registro en la tabla Measurement con la fecha de la medición, el id del usuario y el id de la categoría de la medición y se crean dos registros en la tabla Measurement_data con el id de la medición que se acaba de crear y con el valor y el id de la tabla Value_type, tanto para la presión sistólica como la diastólica.

La mayoría de las mediciones tienen un proceso de inserción similar al de la presión, a excepción de la glucosa y las sesiones de ejercicio. La parte superior del diagrama, que incluye las tablas Training_session y Sport, se utiliza únicamente para realizar inserciones de sesiones de ejercicio del usuario. La parte inferior del diagrama, que incluye las tablas Measurement_details, Options y Details_types, se utiliza únicamente para la medición de glucosa, en donde una medición de glucosa tiene detalles que se relacionan con una opción de un tipo de detalle. Por ejemplo, un tipo de detalle es la fuente de la muestra y una opción de este tipo es el plasma.

Cabe resaltar que para poder hacer las inserciones es necesario tener, en la carpeta de modelos, un modelo para cada una de las tablas que se mencionaron previamente. Estos modelos son clases que se extienden de la clase Model de Sequelize, lo cual permite poder realizar estas inserciones de una manera más sencilla. Una vez se tienen creadas las funciones que se encargan de hacer la inserción a la base de datos en el controlador y los modelos de las tablas de la base de datos, se agregan las rutas a las que se hacen las solicitudes HTTP. Para el ejemplo de las mediciones de sangre, se agregan las rutas "/pressure", "/glucose" y "/oxygen-saturation". Cada una de estas es de tipo POST y están programadas para que cuando se llamen, se redirija el flujo a la función correspondiente del controlador.

4.3.3.2.4 Validaciones y mensajes de error

En Whatoko Health se tiene un microservicio de autenticación que se encarga de generar un token cada vez que un usuario se registra o inicia sesión. Cada vez que se hace una solicitud a una ruta del microservicio de fabricantes se verifica si el token que viene en el *header* de la solicitud es válido y pueden ocurrir 3 escenarios: que el token sea válido, que el token sea válido, pero ya expiró, o que el token sea inválido. A continuación, se tienen los mensajes de error para cuando el token expiró o cuando el token es inválido.

Figura 16

Mensaje de error de token expirado

Figura 17

Mensaje de error de token inválido


```
{
    "ok": false,
    "message": "Token no valido",
    "error": {
        "name": "JsonWebTokenError",
        "message": "invalid signature"
    }
}
```


Estas validaciones del token se realizaron con ayuda del paquete jsonwebtoken en Node, el cual tiene funciones para realizar estas validaciones y retornar mensajes de error específicos. Teniendo las validaciones del token listas, se procede a validar el cuerpo de la solicitud. Para todas las mediciones, el cuerpo de la solicitud debe tener un objeto que contenga tanto el id del usuario como el id del dispositivo que se utilizó para tomar la medición. Para esto se crearon funciones que verifican que en el cuerpo estén los ids y que los ids pertenezcan a usuarios y dispositivos existentes en la base de datos. Por ejemplo, a continuación, se ve el mensaje de error cuando en el cuerpo de la solicitud va un id de usuario que no existe. Un mensaje similar aparece cuando ocurre lo mismo con el id del dispositivo.

Figura 18

Mensaje de error de id de usuario inválido


```
{
    "error": {
        "errors": [
            {
                "value": 10000,
                "msg": "El usuario con id: 10000 no existe",
                "param": "[0].users_id",
                "location": "body"
            }
        ]
    },
    "code": 400,
    "message": "wrong validation"
}
```

4.3.3.2.5 Documentación

Para la documentación del microservicio de fabricantes se utilizó una herramienta de documentación automática. Este tipo de herramientas se encarga de generar una documentación de un proyecto a partir de los comentarios que se encuentran dentro de los archivos de código. Por ejemplo, un comentario hecho en una función indica en la documentación el uso de esta función, así como también sus parámetros. Para el *backend* del proyecto de Whatoko Health, el cual contiene todos los microservicios que componen el proyecto, se utilizó el paquete compodoc. Para utilizarlo únicamente se necesita instalar la dependencia, crear un archivo de configuración que indique dónde están los archivos de código del proyecto y, finalmente, utilizar el comando para iniciar compodoc.

Si el proceso se realizó de forma correcta, se generan una serie de archivos HTML con la documentación que se creó automáticamente. Cada clase del proyecto tiene su propia página en donde hay información como descripción, de qué clase se extiende, atributos, funciones y localización del archivo, así como también el código fuente correspondiente a la clase. La herramienta compodoc también genera una página en donde se muestra la cobertura de los comentarios, es decir, qué porcentaje de las funciones, clases y atributos están comentados de la manera correcta. Esto resulta especialmente útil para verificar qué tan completa está la documentación para posteriormente proceder a completarla.

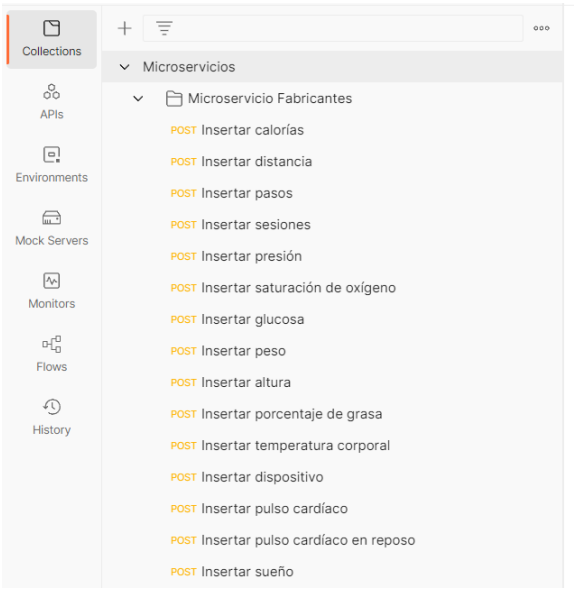
4.4 Fase de pruebas

Con esta fase se da cumplimiento del sexto objetivo específico (Ver sección 2.2). En esta fase se realizaron pruebas y validación con el fin de verificar el correcto funcionamiento de la API Gateway.

Para verificar que el funcionamiento de la API Gateway y del microservicio de fabricantes fuese correcto se realizaron pruebas de tipo funcional en Postman, la cual es una herramienta útil para realizar pruebas de APIs. Para este caso se creó una colección, la cual es un grupo de solicitudes con un formato y características predefinidas. En esta colección se encuentra una solicitud para cada una de las rutas que existen en el microservicio de fabricantes, además, cada solicitud también contó con su respectivo *header* y *body* necesarios para que funcionen correctamente. A continuación, se tiene una figura de la colección en la plataforma Postman.

Figura 19

Colección del microservicio de fabricantes

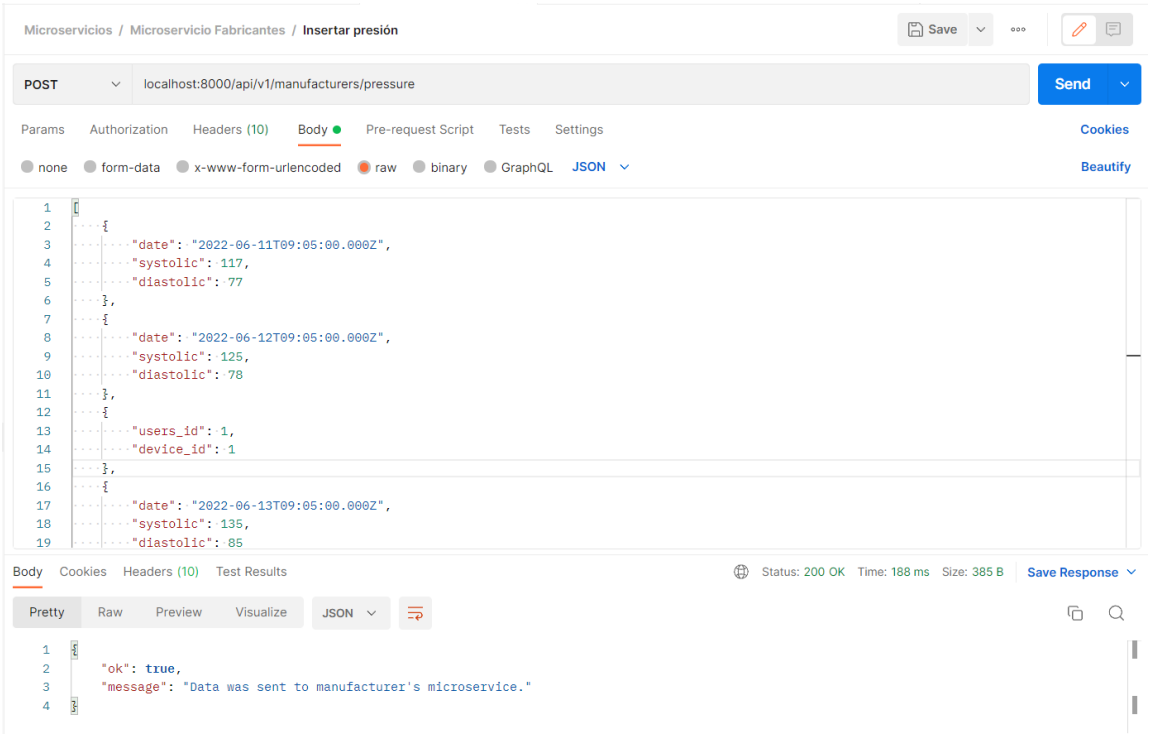


Para realizar las pruebas funcionales se fue accediendo a cada una de las solicitudes de la colección para probar cada uno de los posibles escenarios de respuesta a los que se podía llegar. Primero se verificó que las validaciones funcionaran correctamente ingresando un token incorrecto y un token expirado. En estos casos se obtuvieron respuestas como las de las figuras 16 y 17 para todas las solicitudes. Con esto se supo con certeza que las validaciones con respecto al token

funcionan correctamente. Posteriormente se pusieron a prueba las validaciones con respecto al cuerpo de las solicitudes. Para esto se utilizó un token que sí fuera válido y se fueron haciendo pequeños cambios en el cuerpo de las solicitudes como agregar un id de usuario no válido o agregar un id de dispositivo no válido. Para estos casos se obtuvieron respuestas similares a la figura 18 dependiendo de la variable que se modificara. Con esto se verificó que las validaciones con relación al cuerpo de las solicitudes también estaban funcionando de manera correcta. Finalmente, las últimas pruebas funcionales que se realizaron fueron para verificar el escenario en que la solicitud tuviera todo correcto y se hiciera la inserción de las mediciones en la base de datos. En la siguiente figura se tiene la interfaz de Postman con la solicitud a la ruta de presión de la colección, con su respectivo body y mensaje de respuesta satisfactorio.

Figura 20

Solicitud satisfactoria a ruta de presión



El último paso de la prueba fue verificar que las mediciones presentes en el cuerpo de la solicitud se hayan insertado en la base de datos con su respectiva categoría. En la siguiente figura se ven los registros creados en la tabla de mediciones con su respectiva fecha, id del usuario y categoría de medición según los valores de presión sistólica y diastólica, los cuales se encuentran registrados en la tabla de datos de medición.

Figura 21

Inserciones de presión en la tabla de mediciones de la base de datos

←T→	id 1	date	start	stop	created_at	updated_at	users_id	training_sessions_id	measurement_categories_id
Edit Copy Delete	406378	2022-06-14 10:51:00	NULL	NULL	2022-08-31 16:05:56	2022-08-31 16:05:56	1	NULL	20
Edit Copy Delete	406377	2022-06-14 09:05:00	NULL	NULL	2022-08-31 16:05:56	2022-08-31 16:05:56	1	NULL	19
Edit Copy Delete	406376	2022-06-13 09:05:00	NULL	NULL	2022-08-31 16:05:56	2022-08-31 16:05:56	1	NULL	18
Edit Copy Delete	406375	2022-06-12 09:05:00	NULL	NULL	2022-08-31 16:05:56	2022-08-31 16:05:56	1	NULL	17
Edit Copy Delete	406374	2022-06-11 09:05:00	NULL	NULL	2022-08-31 16:05:56	2022-08-31 16:05:56	1	NULL	16

5. Conclusiones

En este documento se presentó el trabajo que se realizó durante la práctica empresarial con la empresa A&A Soluciones – TIC, la cual se centró en el desarrollo de un canal de comunicación entre las mediciones obtenidas en la aplicación móvil, a partir de dispositivos y aplicaciones de salud, y la base de datos de Whatoko Health.

Durante el desarrollo de la práctica hubo retos que dificultaron el desarrollo. Principalmente, la falta de comprensión del contexto del problema durante las primeras semanas de la práctica resultó en la necesidad de iterar las actividades más veces de las esperadas. Sin embargo, gracias a la ayuda del equipo de la empresa se pudo superar este reto y tener el contexto lo más claro posible para continuar con el desarrollo. A lo largo de las diferentes fases del proyecto se fue dando cumplimiento a cada uno de los objetivos específicos, desde los relacionados con la planeación del proyecto hasta los relacionados con el desarrollo y pruebas, por medio del desarrollo de un microservicio de fabricantes, el cual se encargó de realizar las inserciones de información a la base de datos del proyecto, y un API Gateway con su respectivo plugin personalizado, el cual se encargó de estructurar las mediciones provenientes de la aplicación de Whatoko Health para poder guardarlas con una estructura determinada.

La justificación del proyecto se basó en brindar ayuda en cuanto al control y prevención de enfermedades para pacientes con problemas de diabetes y/o hipertensión. Aunque el proyecto por sí solo no cumple con este objetivo, ya que este es solo una parte de un proyecto más grande, al conectarlo con el ecosistema de Whatoko Health sí se adecúa perfectamente y se complementa con las otras partes del proyecto para cumplir los objetivos planteados en la justificación. Este proyecto permite que haya un canal de comunicación directo entre los dispositivos que toman la

información del paciente y la base de datos, la cual se utiliza posteriormente para alimentar herramientas que brindan ayudas a los usuarios de la aplicación.

En cuanto el trabajo a futuro hay que tener en cuenta que, a medida que se agreguen nuevos dispositivos a la aplicación, es necesario hacer una revisión de los modelos planteados durante el proyecto para verificar que los datos provenientes de los nuevos dispositivos se adecúen a los modelos implementados en el API Gateway. Además, es importante realizar otro tipo de pruebas, como pruebas de carga, cuando todo el proyecto esté montado en el servidor para verificar otras características del API Gateway.

Por otro lado, la participación en la práctica empresarial permitió experimentar cómo es un ambiente de trabajo real, en donde se pudieron afianzar conocimientos adquiridos durante el programa de Ingeniería de Sistemas e Informática, pero además adquirir nuevos conocimientos y habilidades en temas tanto de desarrollo como de planeación y trabajo en equipo.

Finalmente, se concluyó que tanto los objetivos del proyecto como los objetivos personales se lograron con el desarrollo de la práctica empresarial. Tener una prueba de un ambiente profesional antes de finalizar la carrera permitió tener una visión más clara de cómo los conocimientos y experiencias adquiridos durante la carrera se traducen al mundo laboral.

Referencias Bibliográficas

(No lleva número de capitulo e inicia en hoja nueva)

Amazon Web Services. (s.f.). Obtenido de What is DevOps?: https://aws.amazon.com/devops/what-is-devops

American Cancer Society. (22 de Abril de 2020). Obtenido de Telemedicina y telesalud: https://www.cancer.org/es/tratamiento/tratamientos-y-efectos-secundarios/su-equipo-de-tratamiento/telemedicina-telesalud.html#written_by

Atlassian. (s.f.). Obtenido de What is version control?: https://www.atlassian.com/git/tutorials/what-is-version-control

Awad, M. (2005). The University of Western Australia. Obtenido de A Comparison between Agile and Traditional Software Development Methodologies: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.464.6090&rep=rep1&type=pdf

Centers for Disease Control and Prevention. (16 de Diciembre de 2021). Obtenido de What is Diabetes?: https://www.cdc.gov/diabetes/basics/diabetes.html

GCFGlobal. (s.f.). Obtenido de What is wearable technology?: https://edu.gcfglobal.org/en/wearables/what-is-wearable-technology/1/

Healthline. (2022). Obtenido de The Best Diabetes Apps of 2022: https://www.healthline.com/health/diabetes/top-iphone-android-apps

IBM. (19 de Agosto de 2020). Obtenido de Application Programming Interface (API): https://www.ibm.com/cloud/learn/api

IBM. (6 de Abril de 2021). Obtenido de REST APIs: https://www.ibm.com/cloud/learn/rest-apis

International Diabetes Federation. (2021). IDF Diabetes Atlas.

Kong Inc. (30 de Junio de 2022). GitHub. Obtenido de Kong JS PDK development environment: https://github.com/Kong/docker-kong-js-pdk

Kong Inc. (s.f.). Kong Docs. Obtenido de https://docs.konghq.com

LINKLINKS LTD. (2021). App Store. Obtenido de Blood Pressure & Glucose Pal: https://apps.apple.com/us/app/blood-pressure-glucose-pal/id895314221

Liu, K., Xie, Z., & Or, C. K. (2020). Effectiveness of Mobile App-Assisted Self-Care Interventions for Improving Patient Outcomes in Type 2 Diabetes and/or Hypertension: Systematic Review and Meta-Analysis of Randomized Controlled Trials. JMIR Publications.

MacMillan, A. (7 de Mayo de 2020). Live Strong. Obtenido de 5 apps to monitor your blood pressure when you can't see your doc regularly: https://www.livestrong.com/article/13726355-blood-pressure-app/

Microsoft technical documentation. (6 de Enero de 2022). Obtenido de Microservices architecture style: https://docs.microsoft.com/en-us/azure/architecture/guide/architecture-styles/microservices

Nginx. (s.f.). Obtenido de API Gateway: https://www.nginx.com/learn/api-gateway/

Oracle. (s.f.). Obtenido de What is IoT?: https://www.oracle.com/internet-of-things/what-is-iot/

Osakidetza. (21 de Diciembre de 2021). Obtenido de Hipertensión arterial en la diabetes: https://www.osakidetza.euskadi.eus/enfermedad-hta/-/hipertension-arterial-en-la-diabetes/

Pacienzia, J., & Maida, E. G. (Diciembre de 2015). Biblioteca digital de la Universidad Católica Argentina. Obtenido de Metodologías de desarrollo software: https://repositorio.uca.edu.ar/bitstream/123456789/522/1/metodologias-desarrollo-software.pdf

Pulido Rozo, C., & Valencia Collazos, M. R. (2017). SISTEMA PARA EL AUTO MONITOREO DE LA GLUCEMIA Y TENSIÓN ARTERIAL. Bogotá, Cundinamarca, Colombia.

Red Hat. (10 de Junio de 2020). Obtenido de What is an SDK?: https://www.redhat.com/en/topics/cloud-native-apps/what-is-SDK

Scrum. (s.f.). Obtenido de What is Scrum?: https://www.scrum.org/resources/what-is-scrum

Typescript. (s.f.). Obtenido de What is Typescript?: https://www.typescriptlang.org

World Health Organization. (25 de Agosto de 2021). Obtenido de Hypertension: https://www.who.int/news-room/fact-sheets/detail/hypertension

World Health Organization. (11 de Junio de 2021). Obtenido de Cardiovascular diseases (CVDs): https://www.who.int/news-room/fact-sheets/detail/cardiovascular-diseases-(cvds)