UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería Informática

Innovación tecnológica en el bienestar geriátrico: Automatización de procesos usando APIs

Trabajo Fin de Grado

Grado en Ingeniería Informática

AUTOR/A: Chilet Rodrigo, Javier

Tutor/a: Fons Cors, Joan Josep

CURSO ACADÉMICO: 2023/2024

Resum

Este treball presenta una proposta de solució basada en l'arquitectura API-led, amb l'objectiu d'optimitzar les operacions diàries en la gestió de residències i d'implementar noves mesures de monitoratge de pacients. En primer lloc, es proporciona una descripció del context actual de la integració tecnològica en l'àmbit empresarial, destacant les problemàtiques i desafiaments que enfronten les residències a causa del creixement demogràfic i a l'extensa quantitat de processos manuals involucrats.

Posteriorment, s'identifiquen els problemes específics que afecten estes institucions, i es dissenya una arquitectura API-led orientada a la solució d'estos problemes. Es detallarà la implementació d'esta proposta utilitzant diverses tecnologies, incloent-hi ferramentes proporcionades per MuleSoft i Amazon Web Services.

Finalment es descriurà el procés de realització de proves utilitzant la ferramenta Postman, i es presentaran les conclusions derivades del desenrotllament del projecte.

Paraules clau: EAI, Integració d'Aplicacions, MuleSoft, Amazon Web Services, arquitectura d'integració, middleware, API-led, endpoint, residència geriàtrica.

Resumen

Este trabajo presenta una propuesta de solución basada en la arquitectura API-led, con el objetivo de optimizar las operaciones diarias en la gestión de residencias y de implementar nuevas medidas de monitoreo de pacientes. Inicialmente se proporciona una descripción del contexto actual de la integración tecnológica en el ámbito empresarial, destacando las problemáticas y desafíos que enfrentan las residencias debido al crecimiento demográfico y a la extensa cantidad de procesos manuales involucrados.

Posteriormente, se identifican los problemas específicos que afectan a estas instituciones, y se diseña una arquitectura API-led orientada a la solución de dichos problemas. Se detallará la implementación de esta propuesta utilizando diversas tecnologías, incluyendo herramientas proporcionadas por MuleSoft y Amazon Web Services.

Finalmente se describe el proceso de realización de pruebas utilizando la herramienta Postman, y se presentan las conclusiones derivadas del desarrollo del proyecto.

Palabras clave: EAI, Integración de Aplicaciones, MuleSoft, Amazon Web Services, arquitectura de integración, middleware, API-led, endpoint, residencia geriátrica.

Abstract

This work presents a solution proposal based on API-led architecture, aimed at optimizing daily operations in the management of residences and implementing new patient monitoring measures. First, it provides a description of the current context of technological integration in the business sector, highlighting the problems and challenges faced by residences due to demographic growth and the extensive number of manual processes involved.

Subsequently, the specific problems affecting these institutions are identified, and an API-led architecture designed to address these issues is proposed. The implementation of this proposal will be detailed, utilizing various technologies, including tools provided by MuleSoft and Amazon Web Services.

Finally, the process of conducting tests using the Postman tool will be described, and the conclusions derived from the project's development will be presented.

Key words: EAI, Application Integration, MuleSoft, Amazon Web Services, integration architecture, middleware, API-led, endpoint, nursing home.

Índice general

Índice de figuras

CAPÍTULO 1
Introducción

Actualmente la integración se ha erigido como un pilar fundamental para el éxito y la sostenibilidad de las organizaciones en el entorno empresarial contemporáneo. Conectar de manera eficiente procesos, sistemas y datos es esencial para mejorar la eficiencia operativa y responder de forma ágil a las demandas del mercado. La integración permite optimizar la toma de decisiones al proporcionar una visión integral y en tiempo real del negocio, facilitando un servicio al cliente más coherente y personalizado, lo que refuerza la lealtad y satisfacción de este.

No obstante, es crucial considerar que una integración mal concebida puede provocar incompatibilidades entre sistemas, brechas de seguridad y resistencia al cambio, lo que podría derivar en interrupciones operativas y costos elevados. Por lo tanto, resulta esencial implementar una arquitectura de integración bien planificada para maximizar sus beneficios y minimizar los posibles riesgos.

El presente trabajo tiene como objetivo desarrollar una solución de integración de aplicaciones en respuesta al creciente aumento de la población de la tercera edad en los últimos años. Esta iniciativa surgió del proyecto presentado durante el Hackathon organizado por la Universidad Politécnica de Valencia y DISID Corporation, en el marco de la Cátedra de Integración de Aplicaciones firmada entra ambas organizaciones. Durante el desarrollo, tanto de la Cátedra como del Hackathon, se contó con la valiosa colaboración de la Universidad CEU Cardenal Herrera, MuleSoft, Salesforce y Amazon Web Services (AWS). Esta colaboración conjunta permitió abordar de manera efectiva los retos planteados en la Cátedra.

La solución propuesta y desarrollada en este proyecto incluye la creación de un middleware de comunicaciones y la posterior configuración de los servicios proporcionados por Amazon Web Services. A lo largo del trabajo se enfatizará en el diseño y posterior implementación de la propuesta siguiendo una arquitectura API-led que se basa en tres capas. Esta arquitectura facilita la reutilización y escalabilidad de las diferentes aplicaciones mediante la creación de una red, reduciendo así el impacto de futuros cambios.

Al finalizar el desarrollo de la solución propuesta se espera haber alcanzado la automatización de diversos procesos en las residencias de ancianos, lo cual contribuirá significativamente a la mejora del cuidado de las personas de la tercera edad. Esta automatización permitirá liberar al personal geriátrico de múltiples tareas administrativas, facilitando así una mayor dedicación al cuidado directo de los residentes. Asimismo, los enfermos dispondrán de un control más exhaustivo

sobre el estado de salud de los residentes, mejorando la capacidad de respuesta y atención sanitaria. Además, se proporcionará a los familiares un acceso más detallado y actualizado a los informes médicos de los pacientes. Este acceso permitirá a los familiares estar mejor informados sobre la salud de sus seres queridos, promoviendo una comunicación más efectiva y un mayor involucramiento en el cuidado y bienestar de los ancianos.

1.1 Motivación

La informática, como disciplina, juega un papel fundamental en la sociedad contemporánea, actuando como un pilar esencial en la comunicación entre diversos servicios e industrias. Desde una edad temprana siempre me ha fascinado comprender cómo se facilita la comunicación entre diferentes sistemas y cómo se posibilita la transmisión de datos a través de estos. Este interés profundo me llevó a elegir la ingeniería informática como mi campo de estudio.

Durante mi formación universitaria he tenido la oportunidad de cursar una variedad de asignaturas que me han proporcionado el conocimiento necesario para entender en profundidad estos procesos. Estas materias abarcan desde los fundamentos teóricos hasta las aplicaciones prácticas de la informática, permitiéndome adquirir una comprensión integral de cómo se logra la interconexión entre sistemas y la gestión eficaz de los datos. La educación en esta área no solo me ha dotado de habilidades técnicas, sino que también me ha permitido apreciar la importancia de la informática en el desarrollo y mantenimiento de infraestructuras críticas en nuestra sociedad.

En consecuencia, al ser introducido a la integración de aplicaciones mediante la asignatura de último año Integración de Aplicaciones y Procesos (IAP) mi interés por la materia se incrementó significativamente. Esta asignatura combinaba el intercambio de datos con la interconexión de diferentes sistemas, lo que permitía la automatización de procesos. Además, las lecciones se caracterizaban por su enfoque práctico, alejándose de las explicaciones teóricas tradicionales. El profesorado también jugó un papel crucial al incentivarnos a continuar aprendiendo, ofreciendo la oportunidad de inscribirnos en el Hackathon previamente mencionado, que fue precedido por una cátedra diseñada para aumentar nuestros conocimientos de manera exponencial.

Mi elección para realizar las prácticas fue clara: opté por DISID, una empresa española de ingeniería de software especializada en desarrollo de software, ingeniería de datos e integración de sistemas. Durante mi estancia en la empresa, he podido apreciar la diferencia entre la teoría impartida en las clases y la práctica en el mundo real, lo que me ha permitido familiarizarme con el entorno laboral. Tuve la oportunidad de participar en proyectos reales, donde pude mejorar mis conocimientos gracias a la colaboración constante de mis compañeros. Además, adquirí un mayor dominio de las herramientas proporcionadas por MuleSoft, como Anypoint Studio y Anypoint Platform, aunque también tuve la ocasión de conocer otras herramientas de integración como Azure Logic Apps.

A lo largo de estos meses mis conocimientos han mejorado significativamente, lo que me ha llevado a presentar el proyecto desarrollado durante el Hackathon,

implementando una serie de mejoras basadas en el conocimiento adquirido durante las prácticas.

1.2 Objetivos

El objetivo principal de este trabajo es implementar la solución desarrollada durante el Hackathon basándose en el diseño técnico original con el propósito de mejorar y optimizar las áreas necesarias del proyecto incrementando así su eficiencia general. Asimismo, este proyecto representa una oportunidad para adquirir conocimientos sobre estrategias contemporáneas, al interactuar con tecnologías empleadas en el ámbito empresarial actual. La implementación de esta solución, centrada en problemas reales que enfrentan numerosas residencias geriátricas, no solo me permitirá aplicar de manera práctica los conceptos aprendidos, sino que también facilitará un aprendizaje profundo y significativo a lo largo del desarrollo del proyecto.

La solución propuesta dotará a las residencias de una serie de funcionalidades que no sólo facilitarán el trabajo al personal, sino que también mejorarán la calidad de vida de los residentes. Además, ofrecerá a los familiares de los residentes un acceso más amplio y detallado sobre el estado de salud de sus seres queridos.

Los objetivos específicos que se buscan alcanzar son los siguientes:

1. Implementar la monitorización de parámetros de bienestar de los pacientes.
2. Almacenar y consultar informes médicos.
3. Gestión integral de la base de datos de medicamentos.
4. Automatizar la gestión de existencias de medicamentos.

1.3 Estructura de la memoria

El desarrollo de este trabajo se inicia en el capítulo 2, titulado Estado del Arte. Este capítulo se dedicará a una comparación exhaustiva de las diversas tecnologías disponibles en el campo de la integración de aplicaciones. Además, se explicarán las razones que motivaron la elección de las API REST sobre otras tecnologías.

En el tercer capítulo se lleva a cabo un análisis detallado del problema actual que afecta a las residencias, el cual se abordará y resolverá a lo largo de este estudio. Se examinarán también las diversas soluciones disponibles y se justificará la selección de la solución más adecuada para enfrentar esta problemática.

Seguidamente en el capítulo 4 se presenta la propuesta de diseño destinada a soportar la funcionalidad descrita en el apartado precedente. En esta sección se explicarán las diversas tecnologías empleadas y se detallarán las distintas operaciones que se implementarán para abordar la problemática previamente expuesta.

En el transcurso del capítulo 5 se detalla el desarrollo del diseño mediante el uso del software de integración utilizado. Este capítulo proporcionará una descripción del proceso seguido destacando las herramientas, metodologías empleadas y configuraciones realizadas en los sistemas finales.

Durante el capítulo 6 se presentarán las diversas pruebas efectuadas con el propósito de garantizar el óptimo desempeño del middleware desarrollado.

Finalmente, en el último capítulo se llevará a cabo una reflexión sobre los conocimientos adquiridos a lo largo del proyecto, se evaluará si se han cumplido los objetivos establecidos al inicio del trabajo y se identificarán las posibles áreas de mejora.

CAPÍTULO 2
Estado del arte

La integración de aplicaciones hace referencia al proceso mediante el cual diversas aplicaciones independientes diseñadas para cumplir con funciones específicas se coordinan para trabajar de manera conjunta y eficiente. Este proceso implica la combinación y optimización de datos y flujos de trabajo entre diferentes aplicaciones software, permitiendo a las organizaciones modernizar sus infraestructuras tecnológicas y promover operaciones empresariales más ágiles y eficientes. La combinación resultante entre las aplicaciones no solo refuerza la cohesión de los sistemas internos de una organización, sino que también mejora la flexibilidad y la adaptabilidad en las operaciones diarias. [1]

Uno de los principales beneficios de la integración de aplicaciones es su capacidad para superar las brechas entre los sistemas tradicionales y las aplicaciones empresariales basadas en la nube, que están en constante evolución. Al interconectar datos y procesos, la integración de aplicaciones permite a las empresas armonizar diversas funciones a lo largo de toda su infraestructura. Esta cohesión no solo mejora la coordinación entre diferentes áreas operativas, sino que también incrementa la eficiencia y efectividad general de las operaciones empresariales. [2]

Aquellas empresas que no adoptan la integración de aplicaciones enfrentan numerosos desafíos que pueden afectar negativamente su rendimiento. La falta de integración de aplicaciones empresariales provoca la dispersión de datos en diferentes sistemas, dificultando la toma de decisiones estratégicas al no disponer de una visión unificada y precisa de la información. Esta dispersión fomenta la ejecución de procesos manuales y redundantes, lo que incrementa la probabilidad de errores y los costos laborales asociados. Además, la ausencia de un sistema integrado incrementa los gastos operativos y los riesgos de seguridad, ya que se deben gestionar múltiples puntos de acceso y sistemas independientes, complicando la gestión de la seguridad y exponiendo a la organización a posibles amenazas. [2] [3]

Debido a estas razones la integración de aplicaciones desempeña un papel crucial en el entorno empresarial moderno. La implementación de soluciones de integración de aplicaciones empresariales permite unificar y centralizar datos dispersos, proporcionando acceso a información coherente y precisa en tiempo real. Este enfoque minimiza los errores, optimiza la eficiencia operativa y reduce significativamente los costos asociados con la gestión de datos y procesos. Además,

mejora la capacidad de la empresa para responder y adaptarse a las fluctuaciones del mercado. Una infraestructura de integración efectiva permite automatizar procesos críticos, eliminando redundancias y asegurando que todas las partes de la empresa trabajen con la misma información. Asimismo, la visibilidad completa de las operaciones empresariales y la capacidad de realizar ajustes en tiempo real contribuyen a una mayor agilidad organizacional. [1] [2] [3]

Por último, mencionar que la integración de aplicaciones no es solo una cuestión técnica, sino también una estrategia fundamental para mejorar la competitividad y la resiliencia empresarial. Este proceso permite un mayor rendimiento, mejor servicio al cliente y proporciona una ventaja competitiva frente a otras empresas. La capacidad de adaptar y optimizar los flujos de trabajo internos fortalece la capacidad de una organización para enfrentar desafíos futuros y aprovechar nuevas oportunidades en el mercado.

2.1 Tipos de integraciones

A raíz de la explicación anterior hemos obtenido una comprensión más precisa del concepto de integración de aplicaciones. Este proceso es crucial para las empresas, ya que aquellas que no lo incorporan en su arquitectura tecnológica enfrentan diversas dificultades y desventajas frente a otras empresas. No obstante, la integración empresarial de aplicaciones presenta un amplio abanico de opciones en cuanto a los métodos de integración y los mecanismos de comunicación que deben establecerse entre los diferentes sistemas.

Existen diversas formas de llevar a cabo esta integración, cada una adaptada a las necesidades y características específicas de las organizaciones. Estas pueden incluir desde soluciones basadas en middleware hasta arquitecturas orientadas a servicios (SOA) y el uso de APIs. Además, es fundamental asegurar que los sistemas puedan comunicarse de manera eficiente y segura, lo cual implica la adopción de estándares y protocolos adecuados para garantizar la interoperabilidad y la integridad de los datos.

Antes de emprender el diseño de un middleware es esencial evaluar su necesidad y considerar si las condiciones de la organización son propicias para este desarrollo. Este análisis preliminar es fundamental para determinar la conveniencia de implementar una solución de Integración de Aplicaciones Empresariales. La decisión de adoptar esta solución requiere un examen de varios factores críticos. Los elementos que se deben considerar al decidir la implementación de una solución de integración de aplicaciones son las siguientes: [4]

- Beneficios: Las ventajas que una organización puede obtener al implementar una iniciativa EAI. Algunos de los beneficios incluyen una mayor eficiencia operativa, mayor flexibilidad, reducción de costos y mejora en la toma de decisiones.

- Barreras: Los obstáculos o desafíos que pueden dificultar la implementación del middleware. Estos pueden ser de naturaleza tecnológica, organizacional, financiera o relacionados al cambio dentro de la organización.

- Costos: Los gastos asociados a la implementación, que incluyen tanto los costos directos, como la compra de software y hardware, así como los costos indirectos, tales como la capacitación del personal y reestructuración de procesos.
- Presión externa: Las influencias externas que impulsan a una organización a adoptar la EAI. Esto puede incluir la competencia del mercado, requisitos regulatorios, demandas de los clientes o presiones de los socios comerciales.
- Marco de evaluación: El conjunto de criterios y metodologías utilizados para evaluar la efectividad y el impacto de la implementación. Un marco de evaluación adecuado permite medir los beneficios, costos y el rendimiento general de la integración.
- Infraestructura IT: Los componentes tecnológicos y sistemas de soporte necesarios para implementar el middleware. Esto incluye servidores, redes, software y otros recursos esenciales para facilitar la integración.
- Sofisticación IT: El nivel de desarrollo y capacidad tecnológica de una organización. Una mayor sofisticación implica una mayor capacidad para manejar sistemas complejos y adaptarse a nuevas tecnologías.
- Soporte: El respaldo que recibe la implementación dentro de la organización. Esto puede incluir el apoyo de la alta dirección, la disponibilidad de recursos y la aceptación por parte del personal involucrado.

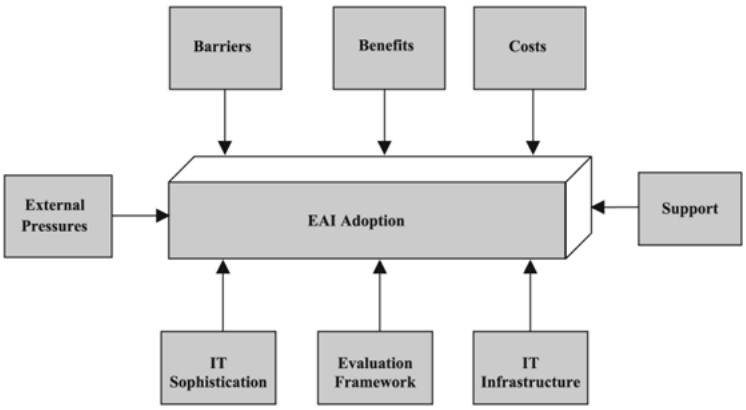

Figura 2.1: Factores clave que conducen a la adopción de EAI

Cada uno de estos factores debe ser analizado detalladamente para tomar una decisión informada sobre la implementación de una solución. La evaluación integral de estos aspectos garantiza que la organización esté preparada para enfrentar los desafíos y aprovechar los beneficios asociados.

2.1.1. Modelos de comunicación

Luego de realizar el análisis previamente mencionado cuyo propósito era evaluar la viabilidad de implementar una solución de integración de aplicaciones

empresariales para la organización, es imperativo examinar los diferentes modelos de arquitectura disponibles. Este paso es fundamental para seleccionar el modelo más apropiado que se ajuste a las necesidades específicas de la empresa.

La comprensión de estos modelos arquitectónicos es fundamental, dado que cada uno presenta diversas ventajas y desventajas en términos de escalabilidad, flexibilidad y complejidad de implementación. Por ello, es esencial llevar a cabo un análisis de cada modelo para garantizar que la solución seleccionada no solo cumpla con los requisitos actuales, sino que también facilite el crecimiento y la adaptación futuros. Además, esta selección debe evitar la incurrencia en costos excesivos o en complicaciones operativas innecesarias.

Los modelos arquitectónicos más comúnmente utilizados en la integración de aplicaciones empresariales son los siguientes: [5] [6] [7]

- Modelo punto a punto: En este modelo tradicional de comunicación entre aplicaciones, una aplicación se conecta directamente con otra a través de un canal, comúnmente denominado *pipe*. Esta comunicación puede realizarse mediante llamadas a procedimientos o mediante envío de mensajes, cada par de aplicaciones requiere de un conector específico encargado de la transformación e integración de datos. Esta estructura permite una comunicación eficaz y personalizada entre dos aplicaciones, adaptándose a las necesidades particulares de infraestructuras pequeñas.

  La Figura 2.2 ilustra claramente cómo cada aplicación está conectada directamente con otra, formando una red de conexiones uno a uno. Entre las ventajas de este enfoque se destacan su simplicidad inicial y los costos bajos al comienzo de la implementación. Sin embargo, a medida que el número de aplicaciones aumenta, también lo hace la complejidad de la estructura de comunicación de manera excepcional. Este incremento en la complejidad puede dificultar el escalamiento del sistema y generar dependencias inesperadas.

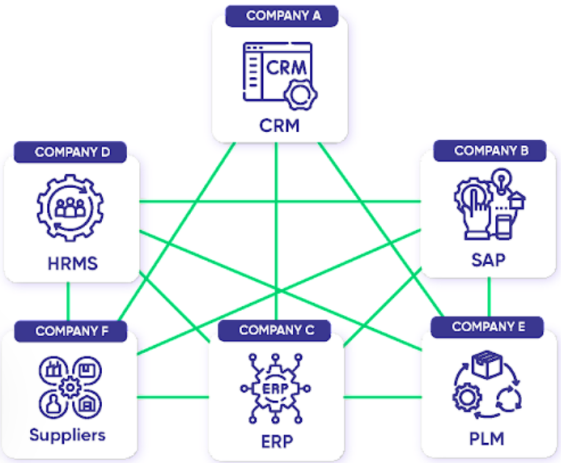

Figura 2.2: Modelo Punto a Punto

- Modelo hub and spoke: El presente modelo fue desarrollado con el propósito de solucionar las dificultades intrínsecas a la arquitectura del modelo previo. En este nuevo enfoque todos los sistemas se conectan a un sistema

central único, denominado hub o broker, mediante conectores denominados spokes, como se ilustra en la Figura 2.3.

Este modelo presenta varias ventajas significativas como el uso del hub como intermediario, que permite el desacoplamiento de los sistemas, facilitando tanto la integración como el mantenimiento. Además, la comunicación asincrónica y la gestión centralizada simplifican la administración de la integración, dado que todas las conexiones y transformaciones son manejadas desde un único punto central. No obstante, un punto crítico es la existencia de un único punto de fallo, ya que el hub puede convertirse en un cuello de botella y, en caso de fallo, esto podría tener consecuencias graves para el sistema. Adicionalmente existen desafíos relacionados con la escalabilidad y la compatibilidad tecnológica, así como limitaciones geográficas, derivados del uso del hub.

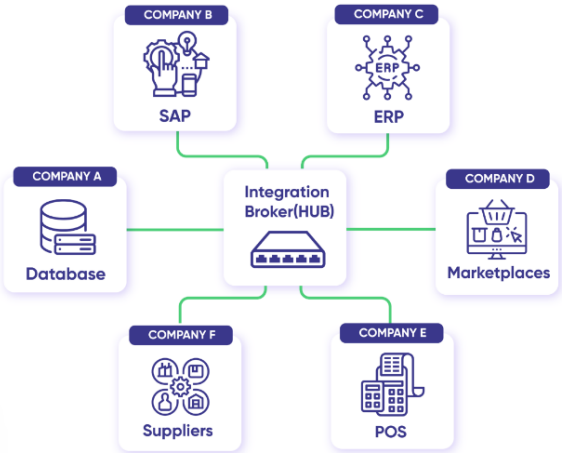

Figura 2.3: Modelo Hub and Spoke

- Modelo mediante bus de servicio empresarial: Este modelo representado gráficamente en la Figura 2.4 se fundamenta en un bus que permite la conexión de diversas aplicaciones y servicios, facilitando su interacción a través de un canal común. Aunque la lógica de enrutamiento y distribución es centralizada, el modelo permite la descentralización de componentes funcionales, que pueden ubicarse en diferentes partes de la red. Las características clave de esta topología incluyen la centralización de la lógica de integración, la descentralización de componentes funcionales, el soporte para múltiples protocolos y la abstracción de la capa de aplicación.

  Existen varias ventajas relacionadas con este modelo, como la simplificación mediante de la integración mediante una plataforma unificada, flexibilidad y escalabilidad, mantenimiento reducido, soporte para la Arquitectura Orientada a Servicios, reutilización de servicios, reducción de costos a largo plazo, y gestión centralizada de la seguridad y errores. A su vez también presenta desventajas, tales como un único punto de fallo, altos costos iniciales de implementación, posibles problemas de rendimiento debido a cuellos de botella, sobrecarga de características innecesarias y dependencias del proveedor, lo que puede resultar en elevados costos de licencias y soporte.

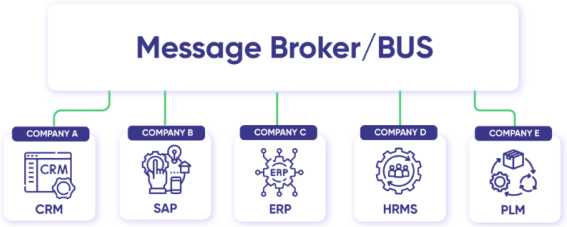

Figura 2.4: Modelo Bus de servicio empresarial (ESB)

2.1.2. Tipos de comunicación

Finalmente queremos abordar las distintas formas de interacción que existen en el ámbito de la integración y permiten la comunicación eficiente entre sistemas. Gregor Hohpe y Bobby Woolf fueron pioneros en identificar estas maneras, cada una con sus características y usos específicos. A continuación, se presentan las principales formas de interacción: [8]

- Transferencia de archivos: Este tipo de interacción, ilustrado en la Figura 2.5, es una estrategia comúnmente utilizada en organizaciones con múltiples aplicaciones independientes. En este esquema, una aplicación actúa como *Productor*, exportando datos a archivos que luego son leídos por otras aplicaciones que operan como *Consumidores*. Este proceso se realiza mediante el uso de protocolos como FTP (File Transfer Protocol) o SFTP (Secure File Transfer Protocol).

  La selección del formato de archivo adecuado es crucial, ya que frecuentemente la salida de una aplicación no es directamente comprensible por otra. Es por ello que es necesario realizar tareas de procesamiento de archivos para facilitar la integración. Adicionalmente, es fundamental considerar la frecuencia de actualización de los datos para asegurar que las aplicaciones consuman información actualizada y precisa, optimizando así el flujo de trabajo y la interoperabilidad entre sistemas.

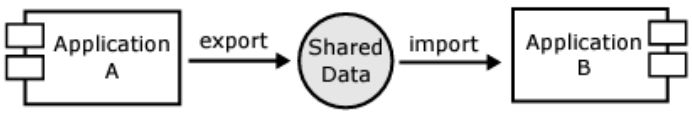

Figura 2.5: Comunicación mediante transferencia de archivos

- Base de datos compartida: Esta técnica se emplea habitualmente en organizaciones que disponen de múltiples aplicaciones independientes que operan en diferentes lenguajes y plataformas. La Figura 2.6 muestra cómo la información de diversas aplicaciones se almacena en una única base de datos, garantizando la coherencia de la información mediante el uso de sistemas de gestión de transacciones de base de datos, esto permite compartir información de manera ágil y consistente.

  Una característica fundamental de este método es su capacidad para asegurar la consistencia de los datos, dado que todas las aplicaciones acceden a una base de datos centralizada. Asimismo, para mantener la integridad

y precisión de la información en todo momento se emplea un sistema de gestión de transacciones. Además, la centralización del almacenamiento de datos permite un acceso inmediato y eficiente a la información actualizada, mejorando así el flujo de trabajo y la comunicación entre aplicaciones.

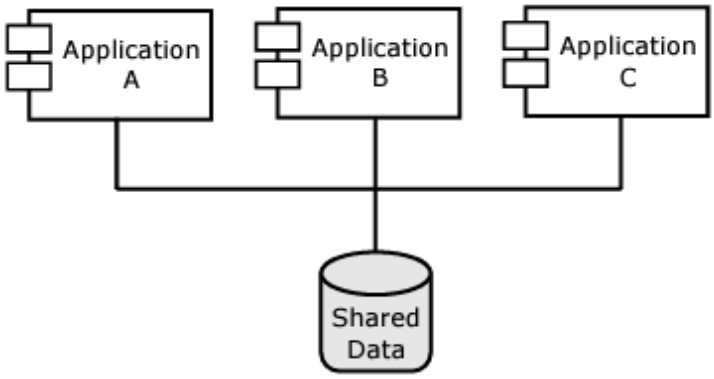

Figura 2.6: Comunicación mediante base de datos compartida

- Llamadas de procedimiento remoto: Las RPC representan una metodología eficiente para compartir datos y funcionalidades entre aplicaciones independientes que operan en distintos lenguajes y plataformas independientes que operan en distintos lenguajes y plataformas, como se observa en la Figura 2.7. Esta técnica permite desarrollar cada aplicación como un objeto o componente de gran escala con datos encapsulados, proporcionando una interfaz que facilita la interacción con otras aplicaciones.

  Una de las principales ventajas del uso de RPC es que permite a una aplicación leer o modificar los datos de otra mediante una llamada, manteniendo la integridad de los datos en cada aplicación. Esto significa que cada aplicación puede modificar sus datos internos o la forma en que estos se almacenan sin afectar a las demás aplicaciones. No obstante, dado que RPC funciona como un sistema sincrónico donde las aplicaciones están directamente conectadas entre sí, existe el riesgo de que una aplicación sobrecargada ralentice todo el sistema. Además, los problemas de red pueden ralentizar o incluso causar fallos en una parte del sistema, lo que puede repercutir en el conjunto.

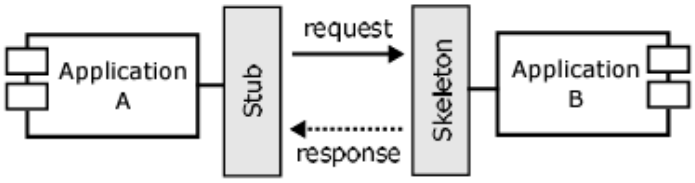

Figura 2.7: Comunicación mediante llamadas de procedimiento remoto

- Mensajería: Este tipo de integración es una solución utilizada en organizaciones con aplicaciones desarrolladas de forma independiente, ejecutadas en diferentes lenguajes y plataformas, donde se necesita compartir funcionalidades de manera reactiva ante eventos. La Figura 2.8 ilustra cómo, a

diferencia de la integración mediante RPC, la integración por mensajería puede ser asincrónica. Este enfoque permite la transferencia de paquetes de datos de manera frecuente, confiable e inmediata, utilizando formatos personalizables y adaptadores.

Un adaptador es un fragmento de código independiente de la aplicación, que abstrae el mecanismo de comunicación entre la aplicación y el mensaje. Constar con un sistema responsable de tomar y entregar mensajes de una aplicación a otra (o incluso a más de una) facilita la interoperabilidad en situaciones donde no todos los sistemas están operativos simultáneamente. No obstante, puede ocurrir que una secuencia de mensajes no se reciba en el mismo orden en que se envió ya sea porque un mensaje falló o tardó más en ser creado, en estos casos el Bus de Mensajes debe reenviar el mensaje.

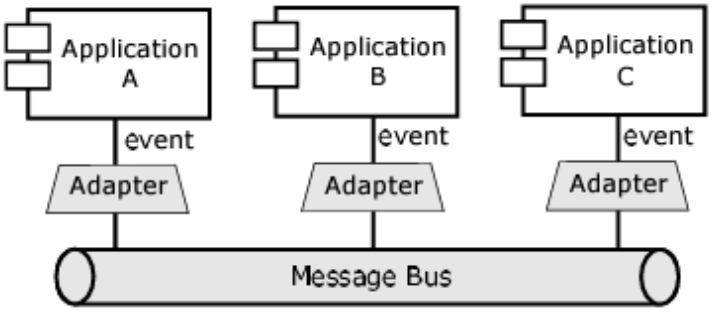

Figura 2.8: Comunicación mediante mensajería

2.2 API REST y arquitectura API-Led

A lo largo del tiempo, la integración de sistemas y aplicaciones ha experimentado una evolución considerable con el propósito de resolver las limitaciones propias de los modelos de comunicación tradicionales, tales como Punto a Punto (Point-to-Point), Hub and Spoke, y el bus de servicios empresariales (ESB). Aunque estos enfoques fueron efectivos en sus inicios, presentaban diversas limitaciones, incluyendo una complejidad creciente, problemas de escalabilidad, altos costos de mantenimiento y una notable falta de agilidad para adaptarse a los nuevos requerimientos del negocio. [9] [10]

A fin de abordar los desafíos presentes en la integración y comunicación entre sistemas se ha adoptado un enfoque que pone énfasis en las interfaces de programación de aplicaciones (APIs), especialmente las APIs REST. Las APIs REST han surgido como una solución eficaz y versátil para la interoperabilidad entre sistemas, diferenciándose de los métodos tradicionales por su capacidad de comunicación a través de una interfaz estándar basada en el protocolo HTTP. Este método simplifica la integración, dado que cualquier sistema capaz de enviar solicitudes HTTP puede interactuar con una API REST. Este enfoque resulta particularmente valioso en entornos donde es necesario que los sistemas se comuniquen a través de redes diversas y variadas. [11]

Una de las características más destacables de las APIs REST es su capacidad para presentar los datos como recursos accesibles a través de URLs específicas.

Las operaciones sobre estos recursos se realizan empleando los métodos HTTP estándar (GET, POST, PUT, DELETE). Este enfoque no solo facilita la comprensión y el uso de las APIs, sino que también potencia la interoperabilidad entre sistemas distintos. Aunque las APIs REST ofrecen numerosas ventajas, como una implementación y mantenimiento más sencillos en comparación con otros métodos, también presentan desventajas, tales como la dificultad para implementar medidas de seguridad y autorización robustas. [11]

La arquitectura basada en la conectividad dirigida por API, conocida como API-led, se establece como una metodología para estructurar las API REST con el fin de optimizar la reutilización, flexibilidad y la gestión eficiente de servicios dentro de un ecosistema tecnológico. Esta arquitectura se organiza en múltiples niveles mediante una segmentación en diversas capas, cuyo propósito es satisfacer las expectativas de flexibilidad y agilidad requerida por las grandes empresas. [12]

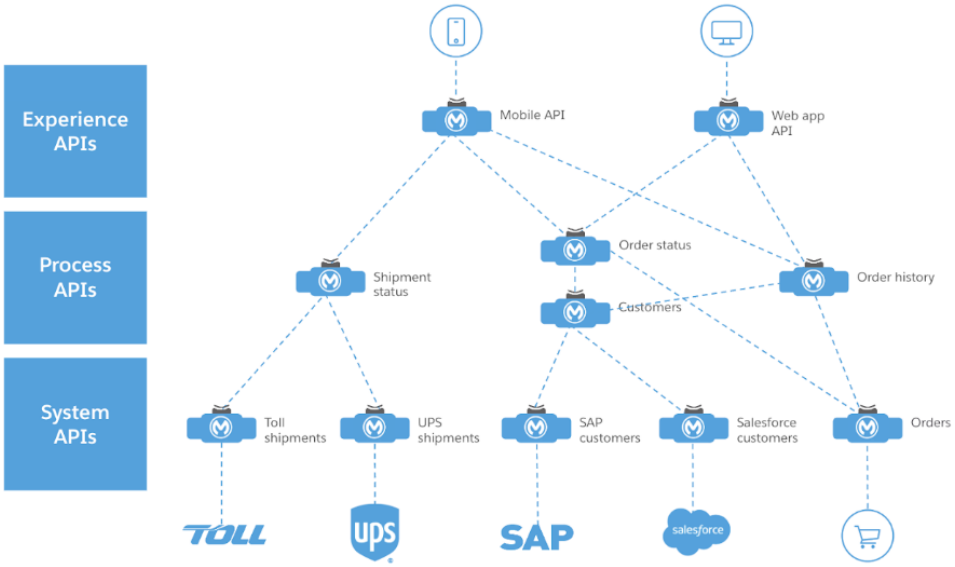

Figura 2.9: Arquitectura API-led o API-led connectivity

La arquitectura fundamentada en la conectividad dirigida por API se estructura en las siguientes capas: [12]

- Capa de Experiencia: Proporciona la interfaz de usuario final y facilita la interacción con el sistema. Esta capa implementa múltiples APIs de experiencia adaptadas a diversos dispositivos y tipos de usuario. Estas APIs permiten utilizar una API de proceso común para gestionar y manipular los datos. Además, la capa de experiencia actúa como la capa de producción tanto para el sistema como para los procesos, entregando el producto final accesible y utilizable por el usuario.

- Capa de Procesos: Las APIs presentes en esta capa encapsulan las lógicas de negocio, manteniendo separadas las interacciones con los sistemas de negocio y los canales de destino de dicha lógica. Esta separación permite que las APIs de proceso operen o interactúen con los datos dentro de un sistema específico sin depender de su procedencia ni su destino final. Este diseño facilita una mayor reutilización y una gestión más eficiente de las

lógicas de negocio, asegurando que las modificaciones en los sistemas de origen o en los canales de destino no impacten en las operaciones de negocio predefinidas.

- Capa de Sistemas: Desempeña un papel fundamental en la conexión directa con todos los sistemas centrales de registro. Aunque su funcionamiento permanece oculto para el usuario final, esta capa es crítica para mantener la conectividad de los sistemas centrales y, por ende, no es directamente accesible para los usuarios. Los cambios en esta capa son poco frecuentes, priorizándose la estabilidad y la integridad de los sistemas centrales. Además, esta capa abstrae y oculta las complejidades inherentes a los sistemas centrales, garantizando que los usuarios no se enfrenten a dichas complejidades.

La Figura 2.9 permite una comprensión más clara de la arquitectura API-led. Esta figura muestra cómo las APIs de la capa de sistemas interactúan con los sistemas finales, exponiendo datos y funcionalidades de manera segura. Además, se ilustra cómo las APIs de la capa de procesos combinan y orquestan los servicios proporcionados por las APIs de sistemas, implementando la lógica de negocio correspondiente. Finalmente se evidencia que las APIs de la capa de experiencia interactúan con los usuarios finales, ofreciendo interfaces optimizadas para diversos dispositivos y canales.

CAPÍTULO 3
Análisis del problema

En el presente capítulo se realizará un análisis integral de la situación demográfica actual con el fin de comprender el contexto en el que se desarrollará el proyecto orientado a mejorar la gestión de las residencias. Se evaluarán las tecnologías empleadas en estos establecimientos y, a partir de este análisis, se identificarán los elementos clave para el desarrollo de la solución propuesta.

Además, se efectuará un análisis comparativo de las diversas soluciones existentes en la actualidad, lo que permitirá identificar las mejores prácticas y las áreas de oportunidad para la formulación de una propuesta de valor sólida. Finalmente se detalla la metodología de trabajo que se utilizará durante el desarrollo de la solución, asegurando una implementación efectiva y coherente con los objetivos establecidos.

3.1 Situación demográfica y su impacto

El aumento en la demanda de residencias para personas mayores es un fenómeno observable a nivel global, relacionado directamente con el envejecimiento progresivo de la población. Este proceso se refleja en diversos indicadores demográficos proporcionados por organizaciones como la Organización Mundial de la Salud (OMS) y la ONU.

A nivel mundial la esperanza de vida ha experimentado un incremento significativo en las últimas décadas. En 2020 se estimaba que alrededor de mil millones de personas tenían 60 años o más, con proyecciones que sugieren un continuo aumento de esta cifra en las próximas décadas. Además, se anticipa que para 2050 la cantidad de personas mayores de 60 años se triplicará. Este crecimiento demográfico se traduce en un aumento acelerado de la población de 80 años o más, la cual se espera también se triplique en el mismo periodo. Actualmente, el número de personas mayores de 65 años ha superado al de niños menores de cinco años, un hito histórico que refleja una transición demográfica hacia una población más envejecida. [13] [15]

Este fenómeno es especialmente notable en la Unión Europea, donde en 2020 el porcentaje de población de 65 años o más representaba una parte significativa del total, marcando un notable incremento respecto a décadas anteriores. Particularmente, el grupo de personas de 80 años o más ha casi duplicado su proporción

en los últimos 20 años y paralelamente se ha observado una disminución en la proporción de jóvenes, destacándose una reducción en la cantidad de niños menores de 14 años. [14]

El envejecimiento de la población tiene importantes implicaciones para la gestión de las residencias para personas mayores. A medida que aumenta la proporción de personas de edad avanzada, la demanda de cuidados geriátricos también se incrementa planteando desafíos significativos para garantizar una atención de calidad. Las residencias para ancianos suelen estar compuestas por equipos multidisciplinarios que incluyen médicos, enfermeros, trabajadores sociales y psicólogos. Sin embargo, el personal de enfermería frecuentemente enfrenta una considerable carga de trabajo, encargándose de la atención directa y de la gestión de medicamentos. Esta sobrecarga puede llevar a errores humanos que, en ocasiones, resultan en negligencias y un deterioro de la salud de los residentes. Esto es observable en informes anuales y noticias sobre incidentes en residencias que destacan los problemas recurrentes, como la falta de respuesta oportuna del personal y errores en la administración de medicamentos.

Además de los desafíos internos en las residencias, la comunicación con los familiares de los residentes presenta dificultades significativas. A menudo, los familiares encuentran problemas para obtener información actualizada sobre el estado de salud de sus seres queridos, debido a restricciones de tiempo, distancia geográfica o limitaciones de salud. Esta falta de comunicación puede causar ansiedad y estrés en los familiares, quienes frecuentemente no reciben la información necesaria para comprender la situación de sus seres queridos.

La creciente proporción de personas mayores en la población exige una revisión de las prácticas en las residencias para ancianos. Es crucial abordar áreas clave como la optimización de la carga de trabajo del personal de enfermería y la mejora en los canales de comunicación en los familiares para asegurar una atención adecuada y eficaz en el contexto de un envejecimiento poblacional acelerado. Esta adaptación es esencial para enfrentar los retos que plantea el envejecimiento demográfico y para mejorar la calidad de vida de los residentes y sus familias en las residencias geriátricas.

3.2 Tecnologías actuales en aplicación

El conocimiento profundo de las tecnologías contemporáneas empleadas en las residencias geriátricas resulta esencial antes de proponer nuestra solución. Este entendimiento nos capacita para identificar los sistemas clave que podemos integrar de manera adecuada, así como para reconocer aquellas áreas que podrían beneficiarse de mejoras o de una migración hacia tecnologías más avanzadas.

Comprender las herramientas tecnológicas implementadas en estas instituciones garantiza que cualquier solución se incorpore sin inconvenientes y aporte un valor significativo al entorno existente. Este conocimiento no solo facilita una integración armoniosa de nuestra propuesta, sino que también nos permite evaluar la eficiencia de los sistemas actuales y proponer mejoras fundamentadas en las últimas innovaciones del sector.

Adoptando este enfoque integral aseguramos que nuestras propuestas no solo sean compatibles con los sistemas existentes, sino que también optimicen los procesos y la calidad de vida de los residentes. Así, al estar bien informados sobre las tecnologías actuales podemos diseñar una solución que responda adecuadamente a las necesidades presentes de las residencias.

Actualmente las residencias geriátricas hacen uso de un software de gestión avanzado, el cual se ha convertido en la principal herramienta tecnológica para su operación diaria. Este tipo de sistemas abarca prácticamente todos los aspectos necesarios para un funcionamiento eficiente, y resulta fundamental para la administración y optimización de diversos elementos operativos en una residencia y su importancia en el ámbito laboral es incuestionable.

El software de gestión empleado en residencias geriátricas cubre áreas clave como la administración de medicamentos, la elaboración de informes médicos y la monitorización continua de los residentes, aspectos esenciales para asegurar una atención de alta calidad. En el contexto médico, este tipo de software facilita un control detallado de los planes farmacológicos, la preservación de historiales médicos y el seguimiento permanente del estado de salud de los residentes. Además, en el ámbito de la enfermería, el software permite el monitoreo preciso y la administración eficiente de medicamentos, así como la documentación sistemática de incidentes y la evaluación constante del estado del paciente.

Asimismo, la capacidad de estos sistemas para generar informes detallados, que consolidan datos de todas las áreas, garantiza que la información médica y de cuidados esté siempre actualizada y sea fácilmente accesible para los profesionales de la salud. La centralización de la información no solo facilita la comunicación entre los diferentes miembros del equipo, sino que también mejora la coordinación de las tareas, lo que contribuye a una presentación de servicios más efectiva y cohesiva.

A pesar de las numerosas funcionalidades integradas que ofrece el software de gestión, las residencias geriátricas aún dependen en gran medida de procesos manuales, lo que limita la optimización completa de las operaciones. Aunque el software contribuye significativamente a la gestión y seguimiento de los procesos, el personal geriátrico se ve obligado a realizar muchas tareas de manera manual, lo que puede restringir la eficiencia operativa y el potencial de mejora dentro del entorno laboral de las residencias. Esta situación pone de manifiesto la necesidad de seguir desarrollando soluciones tecnológicas que no solo integren múltiples funcionalidades, sino que también automaticen procesos con el fin de maximizar la eficiencia y elevar la calidad del servicio ofrecido.

Es importante destacar que muchos de estos sistemas de software cuentan con una API propia, lo que facilita en gran medida la integración de soluciones adicionales que pueden mejorar las aplicaciones existentes. Esta capacidad de integración permite la incorporación de nuevas herramientas y tecnologías, optimizando los procesos manuales actuales y permitiendo a las residencias alcanzar niveles superiores de eficiencia operativa. [16] [17]

3.3 Estructura lógica de la solución

La importancia de definir correctamente la estructura de la información en los proyectos reside en garantizar que todos los componentes del sistema estén alineados y puedan interactuar de manera eficiente. En nuestro proyecto de desarrollo, esta estructura lógica actúa como la columna vertebral que sostiene la integración de los diferentes elementos presentes en las residencias.

Al llevar a cabo un análisis de cada uno de estos elementos y su función dentro del sistema hemos conseguido construir una estructura de información robusta, como se ilustra en la Figura 3.1. Esta estructura no solo refleja la interrelación entre los dispositivos, las acciones que estos ejecutan y los componentes de estos elementos a considerar, sino que también nos permite anticipar y mitigar problemas de interoperabilidad.

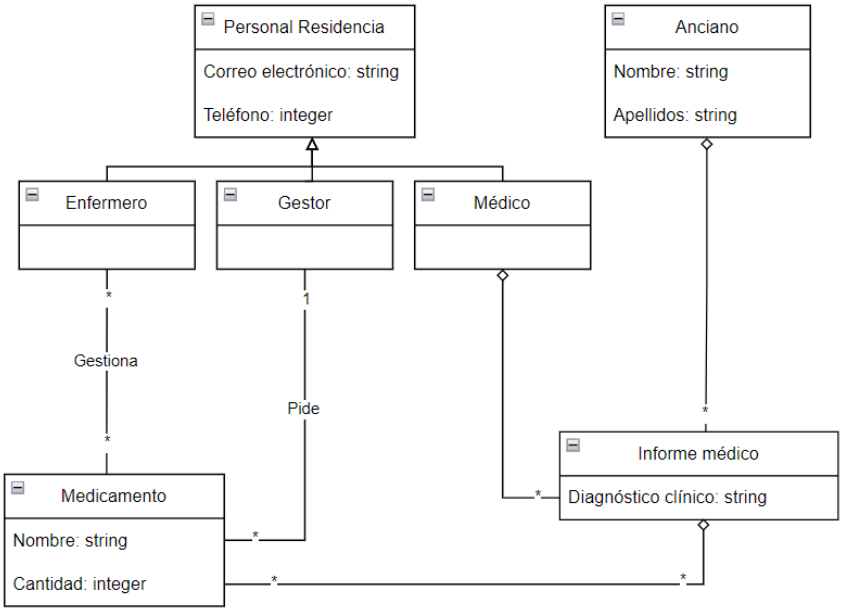

Figura 3.1: Estructura de los elementos de la solución

Este enfoque metódico y sistemático garantiza que todas las partes del sistema puedan comunicarse y operar en completa armonía, lo cual resulta fundamental para la eficiencia y el éxito del proyecto. Además, una estructura lógica bien definida contribuye significativamente a la escalabilidad y al mantenimiento de la solución, también facilita la incorporación de nuevas tecnologías y la adaptación a futuras necesidades sin comprometer la integridad del sistema.

3.4 Identificación de posibles soluciones

A lo largo del siguiente apartado abordaremos un análisis detallado de varias tecnologías de integración como son los servicios web SOAP, los middlewares de integración empresarial (EAI) y las API REST, mencionadas en la Sección 2.2. Cada una de estas tecnologías presenta un conjunto único de características, así

como ventajas y desventajas específicas que deben ser consideradas en función de las necesidades y objetivos de nuestro proyecto.

Primero explicaremos el enfoque SOAP, que se basa en un protocolo estándar para el intercambio de información estructurada en servicios web. A continuación, revisaremos el enfoque EAI, que se centra en la integración de aplicaciones empresariales a través de diversas metodologías y herramientas. Finalmente, examinaremos las APIs REST conocidas por la integración de sistemas a través de servicios basados en HTTP.

Este análisis nos permitirá tomar una decisión informada sobre cuál de estas tecnologías es la más adecuada para satisfacer nuestras necesidades específicas de integración.

3.4.1. Servicios Web SOAP

El protocolo SOAP (Simple Object Access Protocol) está diseñado para su implementación en entornos descentralizados y distribuidos, utilizando Internet y XML como medios para transmitir información tipificada entre nodos. SOAP se utiliza ampliamente en la integración de aplicaciones debido a su capacidad para operar en entornos heterogéneos y su robustez en aplicaciones críticas.

SOAP actúa como un mecanismo de intercambio de mensajes unidireccional y sin estado, permitiendo la combinación de mensajes para crear interacciones más complejas, como las de solicitud/respuesta. Este protocolo es ligero, independiente de la plataforma, del sistema operativo y del medio de transporte, y está construido sobre otros protocolos como HTTP y XML.

Existen dos tipos principales de solicitudes SOAP: la llamada a procedimiento remoto (RPC) y la solicitud de documento, donde se intercambia un documento XML completo. Un mensaje SOAP es un documento XML estándar con una estructura específica que incluye un sobre (envelope), un cuerpo (body) y una cabecera (header).

Las ventajas de SOAP incluyen su adecuación para entornos heterogéneos, la interoperabilidad para el uso de XML, la facilidad para depurar paquetes de texto plano y su robustez para aplicaciones empresariales críticas. Sin embargo, también presenta desventajas, como el mayor tamaño de los mensajes, la complejidad de su implementación, posibles problemas de seguridad, latencia en el procesamiento y la necesidad de mecanismos adicionales para mantener el estado. [18]

3.4.2. Microservicios

Los microservicios son un enfoque de software que descompone una aplicación en servicios independientes, cada uno con una función específica y gestionado de manera autónoma. Este enfoque permite una mayor flexibilidad al facilitar el desarrollo, despliegue y mantenimiento por separado de cada servicio. Las características clave de los microservicios incluyen el desacoplamiento de servicios, el uso de APIs bien definidas para la integración y la capacidad para un desa-

rrollo ágil, así como la posibilidad de escalar y desplegar cada componente de manera independiente.

Sin embargo, este enfoque presenta varios desafíos. La comunicación entre servicios puede resultar compleja, especialmente en términos de coherencia de datos y sincronización. La gobernanza de múltiples servicios requiere una gestión eficaz para mantener la cohesión y el control. Además, los microservicios dependen de metodologías de despliegue, como la contenedorización, lo que puede añadir complejidad a la infraestructura. La gestión de datos distribuidos y las transacciones también son complicadas debido a la fragmentación de datos entre servicios.

A pesar de estos desafíos, los microservicios son especialmente ventajosos para sistemas complejos que necesitan modularidad y cuando se utilizan técnicas de despliegue basadas en contenedores. [19]

3.4.3. API REST como solución

Recapitulando con lo explicado en la Sección 2.2, las API REST ofrecen una solución efectiva para la interoperabilidad entre sistemas mediante el uso del protocolo HTTP. Esto permite que cualquier sistema capaz de enviar solicitudes HTTP pueda interactuar con una API REST. Estas APIs se distinguen por su habilidad para representar datos como recursos accesibles a través de URLs específicas y para realizar operaciones utilizando los métodos HTTP convencionales. Esta metodología facilita tanto la comprensión como el uso de las APIs, mejorando la capacidad de los sistemas para trabajar conjuntamente.

La arquitectura API-led permite la organización de las APIs en diferentes capas para optimizar la reutilización, flexibilidad y gestión dentro de un ecosistema tecnológico. En esta estructura la capa de sistemas maneja la conexión con los sistemas centrales de registro, enfocándose en mantener la estabilidad y ocultar la complejidad inherente. La capa de procesos se encarga de la lógica de negocio, favoreciendo una alta reutilización y protegiendo contra los cambios en los sistemas de origen o destino. Por último, la capa de experiencia ofrece la interfaz de usuario final, adaptándose a diversos dispositivos y tipos de usuarios para asegurar una experiencia óptima.

Finalmente se ha seleccionado esta solución en lugar de los servicios web SOAP y los microservicios debido a que las API REST, al ser más ligeras y flexibles, resultan más adecuadas para nuestro contexto. Aunque los microservicios ofrecen modularidad y escalabilidad, su implementación y gestión son complejas y costosas. En contraste, la arquitectura API proporciona un equilibrio óptimo entre simplicidad, reutilización y escalabilidad, lo que la hace más manejable en contextos empresariales dinámicos.

3.5 Metodología de trabajo y control de versiones

Establecer una metodología sistemática y bien delineada es esencial para el éxito en el desarrollo software, ya que ofrece una estructura organizada que guía

a los equipos a través de todas las etapas del proyecto, facilitando así la coordinación y la colaboración. Un marco metodológico claro permite una gestión más eficaz del tiempo y los recursos, incrementa la eficiencia y la productividad, y contribuye a minimizar los riesgos propios del desarrollo de software.

Además, la gestión de versiones constituye otro elemento fundamental en el desarrollo de software. Este procedimiento garantiza que las modificaciones en el código sean registradas y administradas de manera sistemática, permitiendo a los equipos rastrear y revertir los cambios cuando sea necesario. Sin un sistema de gestión de versiones el desarrollo de software puede volverse desordenado, con el consiguiente riesgo de pérdida de código y conflictos entre distintas versiones del mismo archivo. La gestión de versiones facilita el trabajo colaborativo, dado que múltiples desarrolladores pueden trabajar simultáneamente en el mismo proyecto sin interferir entre sí, y asegura que siempre exista un registro detallado de todas las alteraciones efectuadas.

3.5.1. Scrum y Jira

En nuestro proyecto hemos adoptado la metodología ágil Scrum para optimizar la gestión y el desarrollo de las tareas. Scrum es un marco de trabajo que, aunque su aprendizaje inicial puede ser relativamente sencillo, requiere un esfuerzo considerable para su perfección y dominio. Este enfoque se basa en la ejecución de iteraciones breves denominadas Sprints, que en nuestro caso tienen una duración de una semana. Al concluir cada sprint llevamos a cabo una revisión para evaluar las tareas completadas y establecer los objetivos del siguiente sprint. Este ciclo de evaluación y ajuste nos permite mantener una visión clara y precisa del progreso del proyecto, así como adaptarnos a los cambios y ajustar la planificación en función de las necesidades emergentes. [20] [21]

Scrum se centra en la creación de valor mediante la autoorganización del equipo dentro de períodos de trabajo bien definidos. Los principales mecanismos de Scrum son el Producto Backlog, el Sprint Backlog y el Incremento. El Product Backlog es una lista dinámica que incluye características, requisitos, mejoras y correcciones necesarias para el éxito del proyecto. El Sprint Backlog, por otro lado, es una lista de los elementos del Product Backlog que el equipo ha seleccionado para completar durante el sprint en curso. El Incremento representa el producto final y funcional generado al final de cada sprint, mostrando el avance hacia los objetivos del proyecto. Además, los eventos clave de Scrum, como la planificación del sprint, las reuniones diarias, la revisión del sprint y la retrospectiva del sprint son claves para la coordinación y evaluación continua del trabajo del equipo. [20] [21]

Además, para gestionar y organizar las tareas dentro de nuestro proyecto hemos recurrido a Jira, una herramienta ágil que proporciona un tablero visual para la planificación y administración del trabajo. Un tablero Jira muestra incidencias distribuidas en columnas, donde cada columna representa una fase del flujo de trabajo que el equipo debe seguir. Este tablero ofrece una vista compartida de todas las tareas en diferentes estados: no iniciadas, en progreso y completadas. En nuestro caso, hemos utilizado un tablero de Scrum, adecuado para equipos

que trabajan en sprints, facilitando así una planificación y gestión del trabajo de manera eficiente. [22]

La estructura de nuestro proyecto se organizó en varios Sprints, con una duración total de seis semanas. Durante la primera semana nos centramos en la definición de la especificación RAML, que establece los recursos de las APIs y el formato de los datos. Posteriormente, asignamos una semana a cada funcionalidad, excepto para la segunda operativa, que requirió dos semanas o dos sprints debido a los diferentes submódulos que contiene. A lo largo de los diferentes sprints, se desarrollaron diferentes tareas, incluyendo el desarrollo de los diferentes endpoints de las capas de experiencia, proceso y sistemas, así como la configuración de los sistemas involucrados para su correcto funcionamiento.

Las diferentes tareas se gestionan mediante la utilización del tablero de Jira, desplazándose entre las columnas *Por Hacer*, *En desarrollo*, *Revisión*, *Pruebas* y *Finalizada*. Finalmente, la última semana se dedicó a la realización de pruebas y la depuración de errores, asegurando que cada proceso funcionará conforme a las expectativas y especificaciones definidas.

3.5.2. Gitflow

El control de versiones que hemos seguido en nuestro proyecto ha sido el modelo Gitflow. Este es un modelo de control de versiones que organiza y gestiona el desarrollo del software de manera estructurada y eficiente mediante el uso de múltiples ramas. A continuación, procederemos a describir detalladamente las distintas ramas que forman parte de este modelo y su función específica en el ciclo de vida.

La Figura 3.2 facilita una comprensión más clara y visual del flujo de trabajo y la interacción entre las diferentes ramas del modelo. A continuación, describimos cada una de estas ramas así como su función:

- Rama master: Este componente del modelo representa la línea base del código en su estado más estable, destinado al despliegue en producción. El código que reside en master ha sido sometido a pruebas y se considera completamente fiable y listo para ser lanzado al entorno de producción. Las versiones que se encuentran en esta rama han superado todas las fases de prueba y validación, asegurando su estabilidad y confiabilidad.

- Rama develop: Actúa como el núcleo para el desarrollo de nuevas funcionalidades y mejoras. A partir de develop se integran todas las nuevas características y correcciones antes de prepararlas para una versión estable. Esta rama es fundamental para la realización de pruebas integrales y para la consolidación de los cambios que, eventualmente, se incorporarán en master. Su uso facilita la organización del desarrollo en curso, permitiendo una integración continua y coherente de las nuevas características.

- Rama feature: Se origina a partir de develop y se utiliza para el desarrollo de características específicas o nuevas funcionalidades. Este tipo de ramas permite a los desarrolladores trabajar en mejoras particulares sin comprometer la estabilidad de la rama develop. Una vez que una característica ha

sido desarrollada y probada en su rama correspondiente, se fusiona de nuevo en develop, integrando los cambios de manera controlada y asegurando que el código en desarrollo se mantenga estable.

- Rama release: Generada a partir de develop cuando se está preparando una nueva versión para su lanzamiento, esta rama permite la realización de pruebas finales, corrección de errores y ajustes necesarios antes de la nueva publicación de la nueva versión. Una vez completada el proceso de pruebas y ajustes, la rama release se fusiona tanto en master, para reflejar la nueva versión en producción, como en develop, para incorporar los cambios y correcciones realizadas en la base de código en desarrollo.
- Rama hotfix: Utilizada para resolver errores críticos que requieren una solución inmediata, esta rama se crea a partir de master. Permite realizar correcciones rápidas y específicas para problemas que afectan la versión en producción. Tras implementar las correcciones necesarias, las ramas hotfix se fusionan de nuevo en master y develop. Este procedimiento garantiza que las soluciones aplicadas sean integradas tanto en el entorno de producción como en la línea de desarrollo, manteniendo la coherencia y estabilidad del software.

Para implementar Gitflow en nuestro proyecto hemos utilizado GitHub Desktop. Esta herramienta ha facilitado la gestión visual de las ramas y el seguimiento de los cambios realizados. No obstante, también se podría haber utilizado otras aplicaciones de escritorio, como SourceTree, que permiten aplicar este modelo de flujo de trabajo de manera eficaz de igual forma. [23]

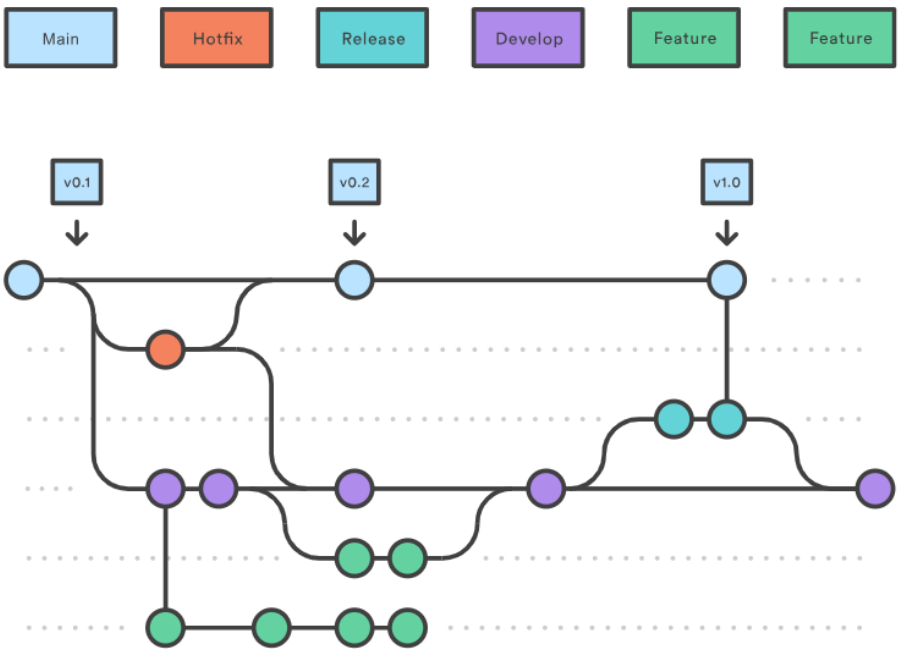

Figura 3.2: Esquema Gitflow

CAPÍTULO 4
Diseño de la solución

A continuación, procederemos a detallar el diseño propuesto para nuestra solución de middleware, que se basa en una arquitectura API-led. Esta arquitectura está compuesta por un total de 12 APIs, las cuales se distribuyen equitativamente a lo largo de las diversas capas de la estructura.

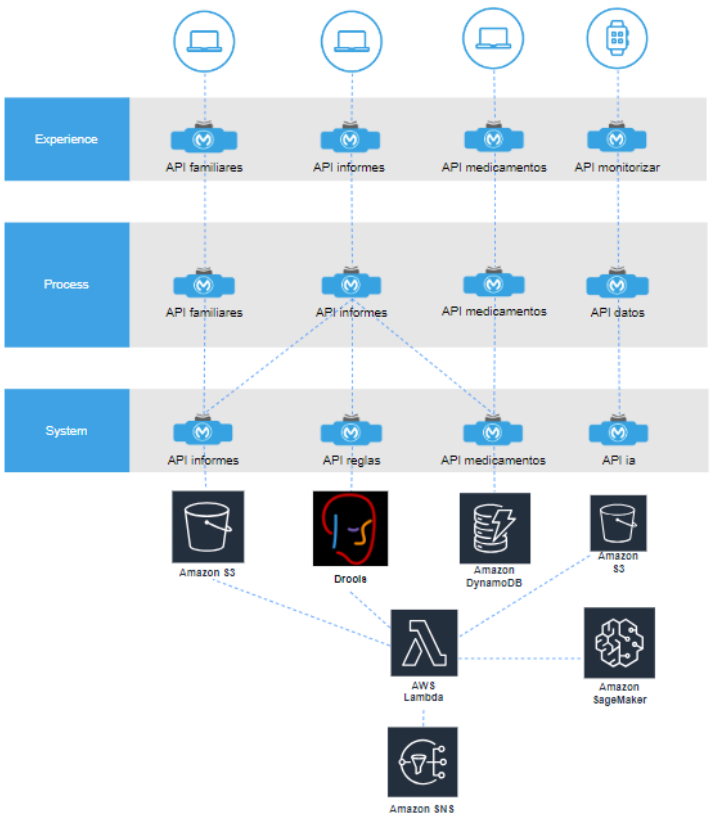

Figura 4.1: Solución arquitectura API-led

En la Figura 4.1 se puede observar la división de las APIs en capas específicas y cómo estas se conectan con múltiples sistemas finales. A lo largo de este capítulo explicaremos en detalle el funcionamiento y propósito de estos sistemas en la solución, así como su interacción con las APIs. Reconocemos que la integración de los sistemas finales puede resultar compleja como consecuencia de la interconexión entre ellos y, por lo tanto, explicaremos el motivo de esto y cómo contribuye al funcionamiento general del sistema.

En cuanto a las APIs de la capa de experiencia, estas están principalmente conectadas a los sistemas de gestión utilizados en las residencias, esto se identifica mediante el símbolo correspondiente a un monitor. Estos softwares que incluyen funcionalidades como portales para familiares, generación de informes médicos, y gestión de inventarios, se aprovecharán para incorporar nuestra solución. No obstante, hemos decidido separar el software en diferentes secciones debido a la diferencia de usuarios que accederán, es decir, a cada API de experiencia que tiene como sistema el software de gestión de residencias accedera un usuario diferente. Esta separación facilita un mejor manejo de las distintas funcionalidades y diferentes grupos de usuarios que interactúan con el sistema.

Dado que una explicación detallada de cada API individual podría resultar demasiado extensa, hemos optado por abordar el diseño en función de las operativas o funcionalidades que componen el sistema. Este enfoque nos permitirá presentar de manera más clara y concisa los aspectos clave de la solución, facilitando la comprensión de cómo cada operativa contribuye al sistema en su conjunto.

En el desarrollo de las funcionalidades, nos centraremos en los sistemas finales involucrados y en su esquema API-led. Describiremos las diversas transformaciones de datos que se llevarán a cabo, explicaremos el diseño detallado de cada API involucrada, y su diagrama de secuencia correspondiente. Además, proporcionaremos una descripción detallada de cada funcionalidad para asegurar una comprensión completa del sistema y su operativa.

Finalmente abordaremos la estrategia de registros de eventos que hemos adoptado. Explicaremos cómo se gestionan y se monitorean los eventos del middleware para garantizar un funcionamiento eficiente y un adecuado seguimiento de las actividades. Esta sección ofrecerá una visión detallada de cómo se asegura la integridad y el rendimiento del sistema mediante la correcta gestión de eventos y registros.

4.1 Monitorización de los parámetros de bienestar

La funcionalidad que proponemos ha sido meticulosamente diseñada para optimizar la atención al paciente mediante la monitorización en tiempo real de sus parámetros de bienestar, empleando avanzadas técnicas de aprendizaje automático (machine learning). Nuestro sistema está concebido para realizar un análisis continuo de datos críticos, tales como la frecuencia cardíaca, los niveles de oxígeno en sangre y la tasa de respiración, los cuales son recolectados a través de pulseras inteligentes que los pacientes portan de manera constante. Esta tecnología es capaz de detectar de forma proactiva cualquier anomalía en estos parámetros, lo que podría ser indicativo de posibles problemas de salud.

En el caso de que nuestro sistema identifique valores fuera de los rangos establecidos como normales se activa automáticamente un proceso de alerta. Este proceso envía notificaciones inmediatas al personal de enfermería, garantizando así una rápida intervención en situaciones potencialmente críticas. Los datos recogidos no solo son procesados en tiempo real, sino que también se almacenan de manera segura para futuras consultas y análisis. Esta capacidad de seguimiento continuo y personalizado proporciona a los profesionales de la salud una herra-

mienta robusta para mejorar tanto la calidad del cuidado como la prevención de emergencias.

Aunque la implementación de esta funcionalidad requiere la adquisición de pulseras inteligentes y su integración con la infraestructura tecnológica existente, consideramos que los beneficios derivados de una mejora sustancial en la atención justifican ampliamente esta inversión inicial. Las pulseras inteligentes no solo son un componente tecnológico esencial, sino también representan una herramienta crucial para garantizar el bienestar constante de los pacientes. A pesar del costo inicial, los beneficios a largo plazo que estas pulseras aportan superan con creces la inversión realizada. En primer lugar, permiten una monitorización continua, lo que reduce la necesidad de revisiones manuales frecuentes por parte del personal geriátrico, liberando así recursos humanos que pueden ser dedicados a otras tareas críticas. Además, la capacidad del sistema para detectar problemas de salud de manera anticipada facilita una intervención médica oportuna, lo que podría prevenir complicaciones graves e, incluso, salvar vidas en algunos casos. Asimismo, la integración de estas pulseras con nuestro sistema también permite la recolección de datos de salud a lo largo del tiempo, ofreciendo a los médicos una visión más completa y detallada del estado de salud de cada paciente. Esta información es invaluable para ajustar tratamientos, detectar patrones no evidentes en revisiones esporádicas y mejorar la personalización del cuidado.

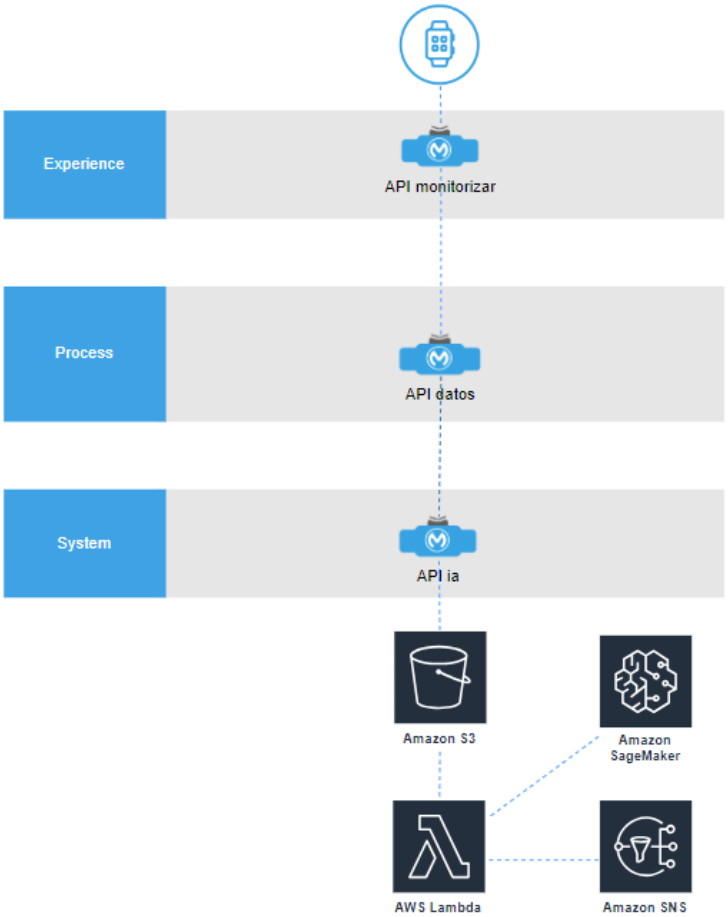

Figura 4.2: Esquema API-led de la monitorización de parámetros

Es crucial enfatizar que las pulseras están diseñadas para servir como un apoyo complementario al personal de geriatría y no como un sustituto a sus responsabilidades. Aunque estas pulseras proporcionan una valiosa herramienta para la monitorización continua de los pacientes, el personal no debe, en ninguna circustancia, despojarse de sus funciones o confiar exclusivamente en la tecnología. Las pulseras, como cualquier dispositivo tecnológico, son susceptibles a fallos y limitaciones.

Finalmente es relevante destacar que esta funcionalidad se compone de tres APIs, como se ilustra en la Figura 4.2. Estas APIs son responsables de la gestión y procesamiento de los datos del paciente, la integración con el sistema de aprendizaje automático para la detección de anomalías, y la activación y gestión del proceso de alertas. Específicamente, la primera API se encargará de recibir y gestionar los datos transmitidos por las pulseras, la segunda API realizará las transformaciones de datos necesarias, y la tercera API interactúa con el sistema de contenedores en la nube que activará mediante el servicio de computación sin servidor el modelo de aprendizaje automático y el sistema de alertas en caso de detección de valores anómalos. Estas APIs trabajarán de manera coordinada para asegurar un monitoreo efectivo y una respuesta ágil ante cualquier eventualidad, garantizando así una atención de calidad y un alto nivel de seguridad para los pacientes.

4.1.1. Sistemas involucrados

La funcionalidad descrita implica la interacción de cuatro sistemas fundamentales, cada uno desempeñando un rol crucial en el proceso de monitoreo y análisis de los parámetros recopilados por las pulseras utilizadas por los residentes. En este análisis consideramos cada uno de estos sistemas, su interrelación y su importancia dentro del flujo general de datos.

En primer lugar, el contenedor en la nube se configura como el punto inicial de almacenamiento de los datos recopilados. Este sistema se encarga de recibir y almacenar los parámetros fisiológicos, como son la frecuencia cardíaca, el número de respiraciones por minuto y el nivel de oxígeno en sangre, en un contenedor inicial. Estos parámetros permanecen en este contenedor hasta que son procesados por el modelo de aprendizaje automático. Según los resultados del análisis, los datos se trasladan desde el contenedor inicial a uno de varios contenedores específicos, cada contenedor está diseñado para alojar los datos según el tipo de anomalía identificada, las cuales pueden variar desde una condición normal (sin anomalías) hasta la detección de irregularidades en uno o más de los tres parámetros monitoreados.

El modelo de aprendizaje automático constituye el núcleo analítico del sistema, su función principal es examinar los parámetros almacenados en el contenedor inicial para detectar cualquier anomalía. Este modelo ha sido entrenado previamente con datos históricos, lo que le permite identificar patrones anómalos con un alto grado de precisión. Además, cuenta con umbrales predefinidos que mitigan la posibilidad de errores en la detección de anomalías. Una vez completado el análisis, el modelo clasifica los parámetros en función del tipo de anomalía

detectada, preparando así los datos para su reubicación en el contenedor adecuado.

El servicio de computación sin servidor juega un papel esencial en la orquestación del proceso automatizado de análisis y acción. Este servicio se activa para ejecutar el modelo de aprendizaje automático, lo que permite el análisis de los parámetros almacenados en el contenedor inicial. Tras completar el análisis, el servicio interpreta los resultados y transfiere los datos al contenedor correspondiente, en función del tipo de anomalía identificada. Adicionalmente, este servicio activa el sistema de mensajería, encargado de notificar a los dispositivos correspondientes en caso de que se detecte una anomalía significativa que requiera atención inmediata.

Por último, el sistema de mensajería se encarga de la notificación final. Una vez que el servicio de computación sin servidor ha determinado la necesidad de generar una alerta, este sistema envía un mensaje personalizado al dispositivo previamente registrado, utilizando el método de comunicación predefinido, como correo electrónico, SMS u otro. El contenido del mensaje se adapta según el tipo de anomalía detectada y está diseñado para proporcionar información clara y directa al destinatario, facilitando así una respuesta rápida y eficiente.

4.1.2. Transformación de datos

Durante esta operativa solamente se ha hecho uso de una única transformación de datos. Esta transformación consiste en adaptar el formato JSON recibido desde las pulseras. El JSON original contiene información relevante como el nombre de la persona que porta la pulsera, su frecuencia cardíaca, el nivel de oxígeno en sangre y las respiraciones por minuto.

La conversión de JSON a CSV se ha llevado a cabo mediante la utilización de la función *mapObject*, que permite recorrer cada clave y valor del objeto JSON y convertirlos en un archivo CSV. En el archivo CSV generado, la primera línea se utiliza para los encabezados de las columnas (nombrePersona, frecuenciaCardiaca, oxigenoSangre y respiracionesMinuto), mientras que la segunda línea almacena los valores correspondientes, todos ellos separados por comas.

Esta transformación se ha integrado en la API de sistemas con el objetivo de facilitar la extracción y el procesamiento de los valores necesarios para el título del objeto que se almacenará en el contenedor inicial. La elección del formato CSV responde a su compatibilidad con el modelo de aprendizaje automático, ya que este formato optimiza el procesamiento y análisis de los datos recopilados por las pulseras. Al utilizar CSV, garantizamos una mayor eficiencia en el análisis y posterior uso de los datos en el contexto del modelo de aprendizaje automático.

4.1.3. Diseño detallado y diagrama de secuencia

La arquitectura de la solución diseñada para la monitorización y análisis de la salud de los residentes está estructurada en varias capas, cada una de las cuales cuenta con APIs especializadas que desempeñan funciones específicas dentro del

sistema. A continuación, presentaremos un análisis detallado del diseño y funcionamiento de las diferentes APIs que conforman esta solución.

En primer lugar, tenemos la capa de experiencia, donde se ubica la API denominada *e-monitorizar-api*. Esta API juega un papel crucial en la interacción directa con las pulseras inteligentes que portan los residentes. Estas pulseras están diseñadas para medir en tiempo real parámetros críticos de salud, como la frecuencia cardíaca, el nivel de oxígeno en sangre y la tasa de respiración por minuto. La API se encarga de recopilar estos datos y prepararlos para su posterior envío a la capa de procesos. Para llevar a cabo esta operación, la API expone el endpoint *post:\datos*, el cual constituye el punto de entrada inicial del sistema. Este endpoint acepta solicitudes *POST* que deben incluir un cuerpo en formato JSON, conteniendo los datos vitales recolectados por las pulseras y estructurados en un objeto JSON. La importancia de este endpoint radica en que actúa como la puerta de entrada para todo el flujo de información dentro del sistema, garantizando que los datos lleguen de manera efectiva a las capas subsiguientes para su procesamiento y análisis.

Una vez que la API de experiencia ha recibido los datos a través del endpoint, estos son transferidos a la capa de procesos, donde opera la API *p-datos-api*. Esta API cumple una función análoga a la capa de experiencia, ya que se encarga de recibir los datos JSON provenientes de dicha capa y transmitirlos a la capa de sistemas para su transformación y análisis. El endpoint principal en esta capa es *post:\bienestar*, que también recibe un cuerpo de la petición en formato JSON, similar al enviado desde la capa de experiencia. Aunque esta API no realiza transformaciones significativas en los datos, actúa como un intermediario esencial que asegura la correcta transmisión de la información hacia la siguiente capa.

En la capa de sistemas, se encuentra *s-ia-api*, la cual desempeña un rol más técnico y complejo. Su función principal es transformar los datos recibidos en un formato compatible con el modelo de machine learning que utilizaremos. Dado que este modelo requiere que los datos estén en formato CSV, la API de sistema realiza la conversión de JSON a CSV. Esta transformación es crucial para que los datos puedan ser procesados por el sistema de machine learning que hemos configurado. La API *s-ia-api* expone el endpoint *post:\datosSalud*, que recibe los datos en formato JSON, y tras la conversión, almacena el archivo CSV resultante en el sistema de contenedores en la nube. Este proceso inicial de almacenamiento es necesario debido a la falta de un conector directo con el modelo de aprendizaje automático. Es por ello que se debe configurar un desencadenador que activa el servicio de computación sin servidor cada vez que se inserta un nuevo archivo en el contenedor. La función definida en este servicio de computación no solo se encarga de enviar el archivo CSV al modelo de aprendizaje automático, sino que también clasifica los resultados del análisis. Dependiendo del estado de salud identificado por la machine learning, la función organiza los archivos en contenedores específicos y en caso de que se detecte alguna anomalía, esta función prepara una notificación personalizada y activa el sistema de mensajería para que llegue a los enfermeros asignados al residente afectado.

Antes de proceder con la implementación de la funcionalidad propuesta, es fundamental comprender la interacción entre las distintas APIs presentes en nuestra solución. El diagrama ilustrado en la Figura 4.3 proporciona una representa-

ción visual detallada de la forma en que las APIs de experiencia *e-monitorizar-api*, de procesos *p-datos-api*, y de sistemas *s-ia-api* interrelacionan para garantizar la correcta recolección, transmisión, transformación y almacenamiento de datos vitales de los residentes.

Este diagrama no solo resalta el orden y la sincronización de las llamadas entre las APIs, sino que también muestra las respuestas que estas envían, asegurando así un flujo continuo de información. Además, nos permite identificar los diferentes endpoints involucrados en el proceso. La visualización de estas interacciones resulta esencial para detectar áreas de mejora en la coordinación y eficiencia del sistema.

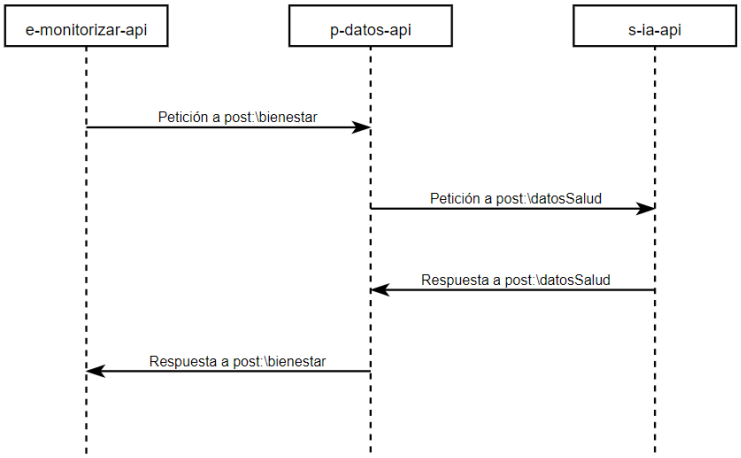

Figura 4.3: Diagrama de secuencia de la monitorización de parámetros

La Figura 4.4 representa un diagrama de secuencias que ilustra las interacciones entre los diversos sistemas finales implicados en la operativa descrita. Este diagrama abarca un total de cuatro sistemas finales, los cuales interactúan entre sí para ejecutar las operaciones previamente explicadas.

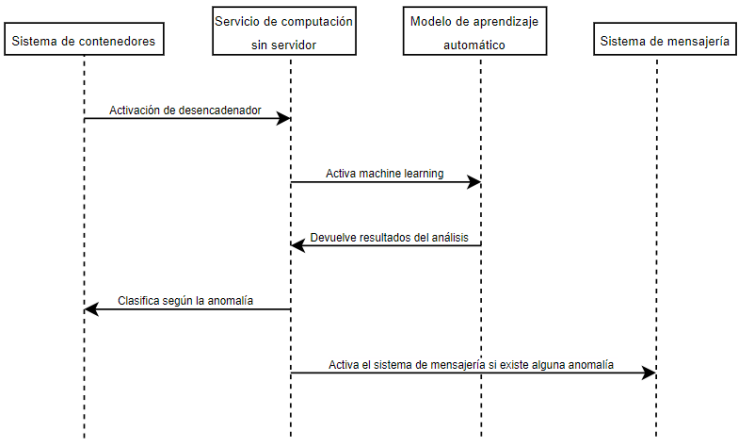

Figura 4.4: Diagrama de secuencia de los sistemas de la monitorización de parámetros

La complejidad asociada al manejo simultáneo de múltiples sistemas finales puede presentar desafíos significativos para la comprensión global del proceso. La necesidad de seguir y entender las múltiples interacciones que ocurren de manera simultánea puede dificultar una visión clara del funcionamiento general.

Debido a esta dificultad hemos optado por incluir este diagrama con el propósito de facilitar la comprensión visual de las secuencias y las interacciones entre los sistemas finales.

4.2 Gestión integral de los medicamentos

La funcionalidad que hemos desarrollado abarca tres operaciones fundamentales en la gestión del inventario de medicamentos: la adición y eliminación de medicamentos, la actualización de la cantidad disponible de un medicamento y la visualización de la lista completa de medicamentos. Hemos optado por integrar estas operaciones en una única funcionalidad, dado que, aunque operan a través de diferentes endpoints, ambas comparten la misma infraestructura de integración. Esta integración no solo facilita una administración más coherente y eficiente de los recursos, sino que también simplifica la implementación y el mantenimiento del sistema. La Figura 4.5 ilustra visualmente esta infraestructura común en la cual intervienen tres APIs distintas y un único sistema final.

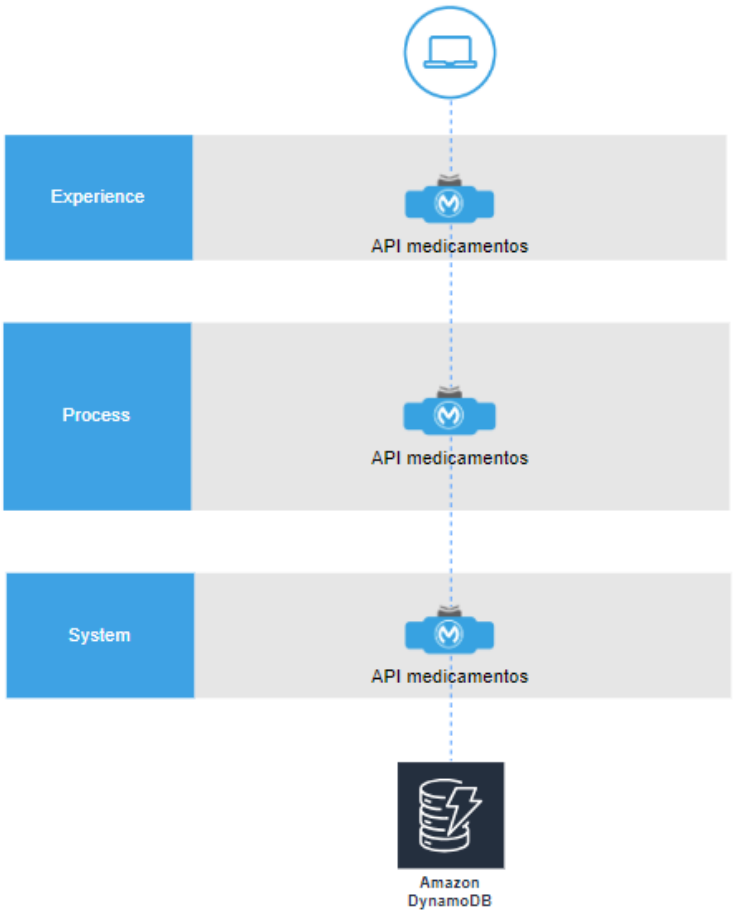

Figura 4.5: Esquema API-led de la gestión de medicamentos

El primer submódulo se encarga de la inserción y eliminación de medicamentos en la base de datos de medicamentos. La operación de inserción requiere que el usuario proporcione información clave, como el nombre común del medicamento, su número CAS (Chemical Abstracts Service), y la cantidad inicial dispo-

nible. Estos datos son procesados por el middleware antes de ser almacenados en la base de datos, lo que genera una nueva entrada en la base de datos. De manera análoga, la eliminación de medicamentos se realiza introduciendo el número CAS, que funciona como un identificador único que permite localizar y eliminar el registro correspondiente en la base de datos, garantizando que la información se mantenga precisa y actualizada.

El segundo submódulo se enfoca en la recuperación del listado de medicamentos disponibles. A través de esta funcionalidad los usuarios pueden visualizar todos los medicamentos existentes, junto con sus identificadores y cantidades respectivas. Esta funcionalidad permite que los usuarios tengan acceso a una vista completa y actualizada del inventario, lo que es esencial para la toma de decisiones informada en la gestión de los recursos farmacéuticos.

El tercer submódulo se centra en el aumento de la cantidad disponible de uno o varios medicamentos seleccionados. Una vez que el usuario identifica el medicamento o medicamentos que desea reponer y especifica la cantidad adicional necesaria, esta información es procesada y actualizada en la base de datos. Este proceso incrementa la cantidad del medicamento seleccionado asegurando que el inventario refleje con precisión las cantidades disponibles. De esta manera se optimiza la administración de los recursos farmacéuticos en las residencias, garantizando una disponibilidad continua de los medicamentos necesarios.

Estas funcionalidades se articulan para ofrecer un sistema de gestión de inventario robusto y confiable, que garantiza la integridad y disponibilidad continua de los medicamentos.

4.2.1. Sistemas involucrados

La operativa que describimos se fundamenta en la interacción con un sistema final único, específicamente una base de datos dedicada a la gestión del inventario de medicamentos de una residencia. Este sistema es esencial para garantizar un control riguroso y actualizado de los medicamentos disponibles, lo que resulta fundamental para la correcta administración de estos y la seguridad de los pacientes.

La base de datos en cuestión está diseñada para almacenar información detallada y específica sobre los medicamentos en una tabla estructurada. Esta tabla incluye campos clave como el nombre común del medicamento, su número CAS (Chemical Abstracts Service) y la cantidad de unidades disponibles. El nombre común facilita la identificación rápida del medicamento, ya que corresponde a su denominación comercial, mientras que el número CAS actúa como un identificador único y estandarizado a nivel internacional, lo que asegura la precisión en la identificación y manejo de cada medicamento.

La intervención de este sistema en la funcionalidad que describimos es crucial, ya que permite ejecutar una variedad de operaciones esenciales. Entre estas operaciones se incluyen la actualización de la cantidad disponible de un medicamento, la adición de nuevos registros en la base de datos, y la eliminación de registros obsoletos o innecesarios. Además, el sistema ofrece la posibilidad de

recuperar, en forma de lista, la totalidad de los medicamentos registrados, facilitando así una visión general y actualizada del inventario disponible.

4.2.2. Transformación de datos

A lo largo de la presente operativa realizamos diversas transformaciones de datos con el fin de asegurar una manipulación y presentación adecuadas de la información. A continuación, describimos detalladamente las transformaciones implementadas siguiendo la estructura de submódulos que hemos empleado durante la descripción de la funcionalidad.

El primer submódulo, que se centra en la eliminación e inserción de registros, abarca transformaciones específicas según la acción a ejecutar. Durante el caso de la eliminación de un registro es esencial proporcionar un identificador que permita su localización y posterior eliminación, para garantizar que este identificador sea reconocido correctamente por el sistema se lleva a cabo un proceso de tipificación explícita. Este proceso implica definir el identificador en un formato específico que la base de datos pueda interpretar y procesar adecuadamente durante la operación de eliminación. Por otro lado, al insertar un nuevo registro en la base de datos se realiza otra transformación. En este caso tanto el nombre del medicamento como el identificador son sometidos a un proceso de tipificación similar. Adicionalmente, la cantidad de medicamento a insertar se somete a una transformación mediante una técnica equivalente aplicada a datos numéricos, lo que permite convertir y formatear este valor para que sea aceptado correctamente por la base de datos. Estas transformaciones se ejecutan en la capa de sistemas justo antes de que se complete la operación de inserción o eliminación.

El segundo submódulo está dedicado a la recuperación de la lista de medicamentos. Tras realizar un escaneo de la base de datos para obtener la lista de registros, el sistema devuelve una serie de datos que incluyen la información relevante de los médicos. La API de sistema lleva a cabo una transformación inicial con el objetivo de aislar los elementos que contienen la lista específica de medicamentos, separándolos de otros datos no relevantes. Posteriormente, en la capa de proceso, reorganizamos y adaptamos estos datos a un formato más claro y accesible, diseñado específicamente para su presentación en el software de gestión de la residencia. Aunque se conserva el tipo de dato original, en este caso JSON, la estructura se modifica para facilitar la comprensión.

Por último, el tercer submódulo se centra en el aumento de la cantidad de uno o varios medicamentos. Durante este submódulo solamente se realiza una única conversión, esta consiste en la realización de una tipificación explícita para el identificador del registro a tratar.

La tipificación explícita es una técnica fundamental que permite definir los datos en un formato específico, lo que asegura que el sistema pueda interpretar y procesar dicha información de forma correcta. Asimismo, aplicamos un proceso equivalente a los datos numéricos, asegurando que estos se formateen y presenten adecuadamente para ser manejados sin errores en la base de datos.

4.2.3. Diseño detallado y diagrama de secuencia

Para abordar de manera exhaustiva el diseño detallado y la secuencia de operaciones de las APIs que conforman esta funcionalidad es necesario organizar la explicación en tres submódulos principales cuya estructura fue previamente descrita. Aunque las APIs involucradas son las mismas en todos los submódulos, la variación en los endpoints empleados en cada uno es clave para atender las necesidades funcionales particulares.

El primer submódulo se centra en la gestión de los registros de medicamentos en la base de datos, abarcando tanto la inserción como la eliminación de dichos registros. Este proceso se inicia en la capa de experiencia, donde se ubica la API *e-medicamentos-api*. Esta API proporciona la funcionalidad necesaria para gestionar los registros de medicamentos integrándose directamente con el software de gestión de la residencia. El endpoint relevante para este submódulo es *post:\medicamentos*, que recibe los datos del medicamento tales como el identificador, nombre, cantidad, y un parámetro de consulta denominado *modo*. Este parámetro es crucial, ya que determina si la operación solicitada es una inserción o una eliminación de un medicamento en la base de datos. Una vez recibida la solicitud, la API de experiencia transmite esta información a la capa subsiguiente, conocida como la capa de procesos.

En la capa de procesos la API utilizada es *p-medicamentos-api*, que también expone el endpoint *post:\medicamentos* y recibe la solicitud de la capa de experiencia. Posteriormente, esta API reenvía la petición incluyendo los datos del medicamento y el parámetro *modo* hacia la capa de sistemas. Este paso es esencial para mantener la consistencia en la transición de los datos entre las capas y asegurar que la información se transmita de manera precisa a la capa encargada de la ejecución lógica de la operación solicitada.

La capa de sistema es gestionada por la API *s-medicamentos-api*, la cual también implementa el endpoint *post:\medicamentos*. Esta API es responsable de realizar la operación solicitada, ya sea la inserción o eliminación de un medicamento. El proceso se basa en un patrón de decisión determinado por el valor del parámetro *modo* y el número de elementos presentes en el cuerpo del mensaje. Cuando se busca eliminar un registro el cuerpo del mensaje debe contener únicamente el identificador del medicamento, y el valor del parámetro *modo* debe ser borrar. En contraste, para la inserción de un fármaco el valor del parámetro *modo* debe ser insertar, y el cuerpo del mensaje debe incluir tres elementos: cantidad, identificador y nombre del medicamento. Este enfoque permite manejar ambas operaciones utilizando un solo endpoint, lo que evita la necesidad de definir múltiples endpoints para cada tipo de operación. Previo a la ejecución de estas operaciones se realiza una conversión que incluye una tipificación explícita de los elementos presentes en el cuerpo de la petición.

Es relevante destacar que un parámetro de consulta (query parameter) es un tipo de parámetro que se envía en la URL de una solicitud HTTP y se utiliza para transmitir información adicional al servidor, la cual puede ser empleada para determinar cómo procesar dicha solicitud.

La Figura 4.6 presentada en el documento ilustra la sincronización de llamadas entre las distintas APIs y cómo el parámetro de consulta es utilizado con-

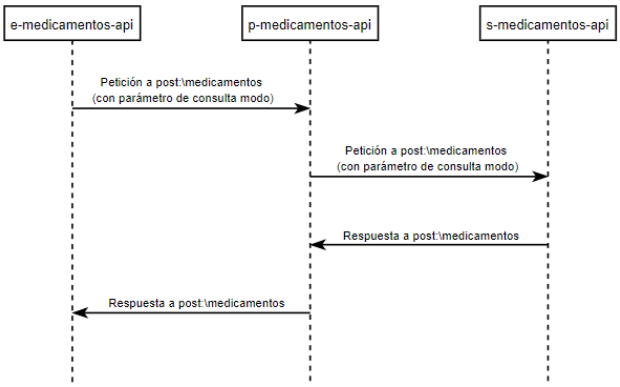

Figura 4.6: Diagrama de secuencia de la eliminación o inserción de un medicamento

sistentemente para realizar la operación deseada. Este diagrama ofrece una comprensión más profunda del funcionamiento del submódulo.

Posteriormente abordaremos el segundo submódulo, el cual facilita la recuperación de la lista de medicamentos disponibles en la base de datos. A diferencia del primer submódulo, este emplea el endpoint *get:\medicamentos* en la capa de experiencia. Este endpoint no solo sirve como la puerta de acceso al middleware para el usuario a través del software de gestión de la residencia, sino que también se encarga de propagar la petición realizada.

Cuando la petición llega a la capa de procesos, es gestionada por el mismo endpoint *get:\medicamentos*, el cual se encarga de propagar la solicitud y, al recibir la respuesta de la API de sistemas, realiza una transformación de datos. Esta transformación tiene como objetivo modificar la estructura de la respuesta para mejorar su visualización por parte del usuario, como se explicó en la Sección 4.2.2.

La capa de sistemas, por su parte, es responsable de la conexión con la base de datos. Al recuperar la lista de medicamentos se realiza una solicitud desde la capa de procesos a la capa de sistemas a través del endpoint *get:\medicamentos*. Este endpoint obtiene todos los objetos de la lista y luego realiza una conversión de datos para aislar los medicamentos de cualquier dato irrelevante que pudiera haberse incluido en la respuesta.

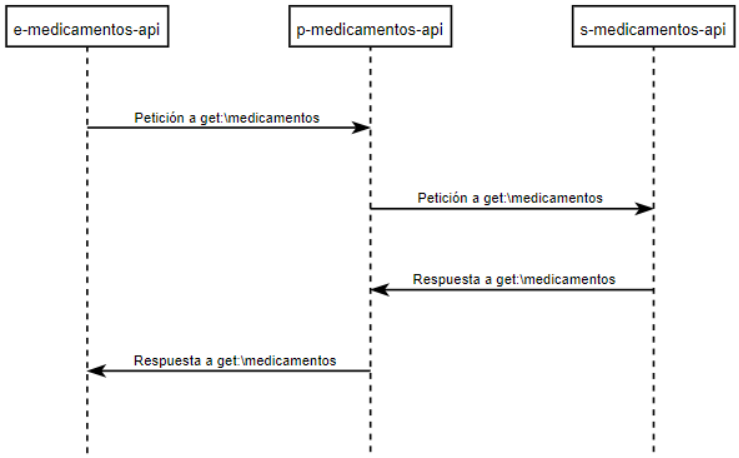

Figura 4.7: Diagrama de secuencia de la obtención de la lista de medicamentos

La Figura 4.7 muestra un diagrama de secuencia que ilustra la sincronización de llamadas y los endpoints utilizados en este submódulo, permitiendo así una visualización clara de la interacción entre las diferentes APIs.

Finalmente abordaremos el tercer submódulo, que se encarga de gestionar el incremento en la cantidad de uno o varios medicamentos almacenados en la base de datos. El punto de entrada para interactuar con este submódulo es el endpoint *put:\medicamentos*, a través del cual el usuario envía su solicitud. El cuerpo de la solicitud que llega a esta API está compuesto por un array que puede contener uno o varios objetos que representan medicamentos, cada uno de los cuales está definido por su identificador y la cantidad que se desea aumentar. La función principal de esta API es propagar la solicitud a la capa de procesos.

La capa de procesos cuenta con el endpoint *put:\medicamentos*. Sin embargo, a diferencia de la API anterior, aquí se encarga de tratar de manera individual los diferentes objetos medicamentos que recibe. Esta separación es necesaria dado que la cantidad de medicamento debe incrementarse de forma independiente. Es por ello que cada medicamento es procesado y enviado a la capa de sistemas a través del endpoint *put:\medicamentos\(id)*, donde el parámetro URI corresponde al identificador único del medicamento, conocido como su número CAS. El cuerpo de la petición en este caso contiene únicamente la cantidad que se desea aumentar para ese medicamento en particular.

En la capa de sistemas, los medicamentos son gestionados de manera individual utilizando el endpoint *put:\medicamentos\(id)*. La cantidad especificada en el cuerpo de la petición es empleada para interactuar con el conector de la base de datos. Antes de realizar esta operación, es necesario llevar a cabo una tipificación explícita tanto del identificador como de la cantidad, con el fin de garantizar que la base de datos pueda reconocer y procesar correctamente los valores.

La sincronización de llamadas entre las distintas APIs se ilustra en la Figura 4.8, la cual muestra cómo inicialmente se transmite todo el arreglo que contiene los diferentes medicamentos y las cantidades a incrementar, pero posteriormente, la petición que llega a la capa de sistemas se maneja de manera individual, incluyendo un parámetro URI específico para cada medicamento.

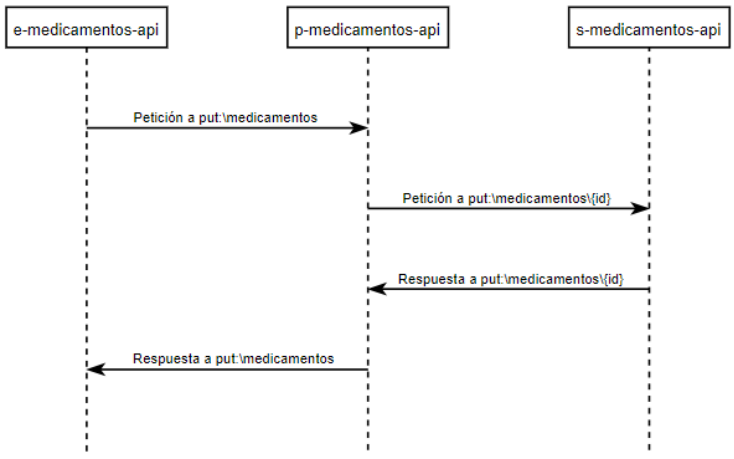

Figura 4.8: Diagrama de secuencia del aumento de la cantidad de medicamentos

4.3 Consulta de informes médicos

Hemos desarrollado esta funcionalidad con el objetivo de mejorar la comunicación con los familiares de los residentes mediante la implementación de un sistema de consulta de informes médicos a través de un software de gestión especializado o portales web habilitados por las residencias. Esta herramienta facilita el acceso a la información médica crítica permitiendo a los familiares consultar informes médicos utilizando un usuario y contraseña proporcionados por el centro. Al iniciar sesión los usuarios pueden acceder a una sección dedicada a la visualización de informes médicos, donde se detallan los diversos informes clínicos del residente, facilitando así el seguimiento de su evolución médica.

El proceso de consulta comienza con la autenticación del usuario, un paso fundamental para asegurar que solo las personas autorizadas puedan acceder a la información. Una vez autenticados, los familiares interactúan con una interfaz intuitiva que simplifica la búsqueda y visualización de los informes. Estos informes, almacenados de forma segura, se presentan de manera clara y accesible. Además, la implementación de rigurosas medidas de seguridad para proteger la privacidad y confidencialidad de la información médica garantiza que únicamente los familiares autorizados puedan acceder a los informes. Esta funcionalidad no solo mejora la comunicación entre la residencia y los familiares, sino que también facilita una gestión más eficiente de la salud de los residentes.

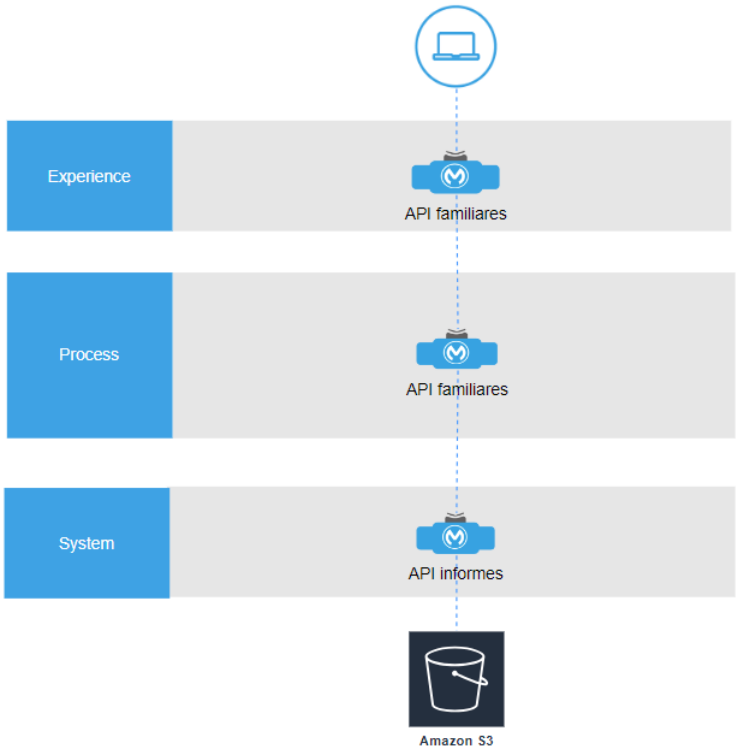

Figura 4.9: Esquema API-led de la consulta de informes

La operativa se basa en una arquitectura API-led compuesta por tres APIs, como se ilustra en la Figura 4.9. Esta arquitectura está diseñada para mejorar la eficiencia en la entrega de informes médicos, promoviendo una comunicación más efectiva y una mayor transparencia entre la residencia y los familiares.

4.3.1. Sistemas involucrados

Esta operativa que permite la recuperación de informes implica la interacción con un único sistema final. Seguidamente detallaremos el sistema implicado y su función específica dentro de este proceso.

El sistema en cuestión es un sistema de contenedores en la nube, que se configura como el punto inicial de almacenamiento para los informes médicos generados a pacientes. Este sistema desempeña un papel crucial al permitir la recuperación de dichos informes a través de la utilización de un identificador único proporcionado previamente al intento de recuperación de los informes. La clave para la recuperación exitosa radica en la utilización del identificador único, que actúa como un medio de búsqueda dentro del repositorio de datos. Este identificador facilita la localización y extracción de los informes médicos específicos que se requieren.

4.3.2. Transformación de datos

Durante el proceso que estamos describiendo se llevan a cabo diversas transformaciones de datos que optimizan la presentación y accesibilidad de la información para el usuario final. Estas transformaciones se efectúan en dos capas distintas: la capa de sistemas y la capa de procesos.

La capa de sistemas inicia con la recopilación de los nombres de todos los informes relacionados con el paciente que coincide con el usuario autenticado. Organizamos esta lista de nombres de informes en orden descendente, de modo que el informe más reciente se sitúe en la parte superior. La razón para esta transformación radica en que el informe más reciente suele ser el más relevante para el usuario, dado que refleja el estado de salud más actualizado del paciente. Esta priorización asegura que el usuario pueda acceder de manera rápida y eficiente a la información más actualizada, lo cual es crucial para una consulta precisa y efectiva.

Por otro lado, la capa de procesos lleva a cabo la segunda conversión de los datos. Durante esta transformación, los informes, que inicialmente están en formato JSON como un array, se convierten a formato XML, donde un elemento principal llamado INFORMES contiene una serie de elementos secundarios denominados INFORME. Esta conversión tiene como objetivo principal mejorar la visualización y presentación de los informes en el software de gestión de las residencias o en el portal para familiares ofrecido. El formato XML es ampliamente utilizado en entornos web debido a su capacidad para estructurar y presentar datos de manera clara y jerárquica, facilitando la interpretación y manipulación de la información por parte del navegador web. Luego de transformar los datos a un formato XML aseguramos una mayor compatibilidad con las tecnologías web y una presentación más ordenada y legible de los informes para el usuario final.

4.3.3. Diseño detallado y diagrama de secuencia

Esta sección presenta la construcción completa de la arquitectura, abordando las distintas capas que organizan la funcionalidad y gestionan el flujo de datos

entre los diversos componentes y servicios. Cada capa está equipada con APIs y endpoints específicos que facilitan la interacción y comunicación entre los sistemas involucrados.

Inicialmente en la capa de experiencia se encuentra la API denominada *e-familiares-api*. Esta API desempeña un papel crucial al gestionar la comunicación directa con el portal web ofrecido por la residencia o el software de gestión, permitiendo que los familiares consulten los diferentes informes. Su función principal es recibir solicitudes para la recuperación de informes por parte de los familiares. Cuando un usuario accede a la sección de informes, la API *e-familiares-api* recibe una solicitud de recuperación a través del endpoint *get:\informes\(id)*. El parámetro *id* se utiliza para identificar y localizar todos los informes asociados a un usuario específico. Este enfoque, que emplea parámetros URI (Uniform Resource Identifier) en las URL, no solo incrementa la seguridad al minimizar la exposición de datos sensibles, sino que también optimiza el control de acceso y reduce el riesgo de ataques de fuerza bruta. Cabe destacar que, aunque la API en esta capa no procesa directamente información, si se encarga de transmitir la petición a la siguiente capa, conocida como la capa de procesos.

En la capa de procesos encontramos la API denominada *p-familiares-api*, la cual también dispone de un endpoint *get:\informes\(id)*, similar al de la capa de experiencia. La función principal de esta API es, en un primer momento, transmitir la solicitud realizada desde la capa de experiencia, en la cual se solicitaron los informes de un paciente específico, hacia la capa de sistemas. Una vez que el conjunto de informes es recibido, esta API lleva a cabo una conversión de datos de formato JSON a XML, adaptando así la información según las necesidades de los sistemas a los que presta servicio. Esta transformación resulta esencial para garantizar la compatibilidad de los datos entre los distintos sistemas y formatos gestionados en la arquitectura. Además, esta capa actúa como intermediaria, facilitando el paso de la información desde la capa de sistemas hasta la capa de experiencia.

Finalmente, en la capa de sistemas, se encuentra la API *s-informes-api*, cuya función principal es interactuar con el sistema de contenedores en la nube donde se almacenan los informes. Esta API cuenta con el endpoint *get:\informes\(id)*, que permite acceder a los informes almacenados mediante el uso del parámetro URI *id*. Una vez que los informes son recuperados del sistema de contenedores en la nube, estos se transmiten a través de las diferentes APIs en las capas superiores, hasta llegar a la interfaz web de la residencia. Antes de continuar queremos destacar que para maximizar la eficiencia del proceso de recuperación inicialmente se obtiene los nombres de aquellos informes cuyo nombre coincide con el parámetro URI proporcionado de forma parcial, y posteriormente se realiza una ordenación descendente de estos nombres para que los informes más recientes, que se obtienen luego, se presenten primero. Así los familiares pueden visualizar los informes médicos dónde se incluyen las evaluaciones realizadas por el personal médico y la información actualizada sobre los medicamentos que el residente está recibiendo.

A través la Figura 4.10 podemos comprender de manera gráfica la sincronización de llamadas entre las diferentes APIs a lo largo de las distintas capas. Este diagrama proporciona una representación visual del flujo de datos y la interac-

ción entre los componentes, evidenciando cómo se coordinan las solicitudes y respuestas entre las APIs para asegurar un proceso fluido y eficiente de recuperación y presentación de informes.

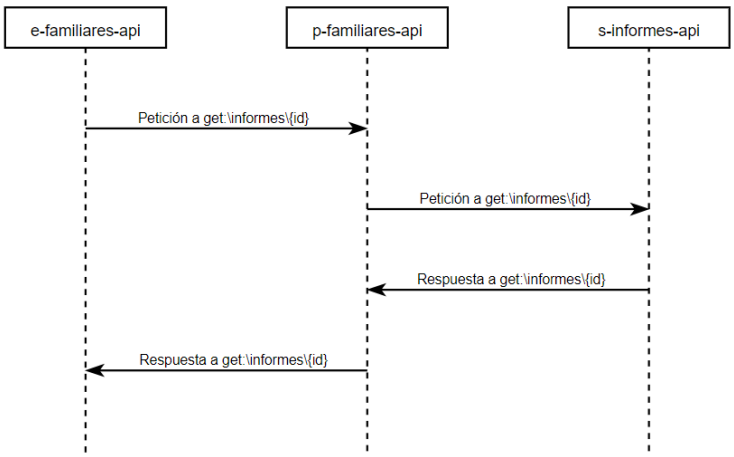

Figura 4.10: Diagrama de secuencia de la consulta de informes

4.4 Control de informes médicos y medicamentos

La funcionalidad descrita a continuación constituye el núcleo fundamental de nuestro middleware, centrado en la gestión de medicamentos recetados en informes médicos, así como en el almacenamiento y la notificación de nuevos informes, y en la alerta sobre la escasez de medicamentos, en caso de que ocurra. Tal como se ilustra en la Figura 4.11 esta funcionalidad integra diversos sistemas finales mediante cinco APIs principales, divididas entre las diferentes capas. Estas APIs son esenciales para facilitar tanto la comunicación como la automatización de las tareas involucradas en todo el proceso.

El flujo de trabajo comienza con la generación de un informe médico, el cual se almacena en un sistema en la nube. Cada informe se identificada de manera única utilizando un identificador que combina el nombre del residente, un identificador específico, y la fecha de creación del documento. Posteriormente, una de las APIs de la capa de sistemas se encarga de almacenar el informe y de notificar a los familiares del residente sobre la disponibilidad de este nuevo documento, asegurando así el acceso inmediato a la información relevante.

Simultáneamente el proceso de almacenamiento del informe, otra API se encarga de actualizar automáticamente el inventario de medicamentos, ajustando la cantidad disponible según los medicamentos recetados en el informe. Una vez que el inventario ha sido actualizado, este es evaluado por un sistema de reglas de negocio que, a través de otra API, determina si es necesario reponer algún medicamento. En caso de detectarse un nivel bajo de un medicamento en particular esta API activa un proceso de alerta que notifica a los gestores de la residencia, sugiriendo la realización de un nuevo pedido. Esta notificación, personalizada y enviada de forma inmediata a través de un sistema de mensajería, asegura que los gestores estén informados en un tiempo real, permitiéndoles tomar decisiones oportunas y efectivas.

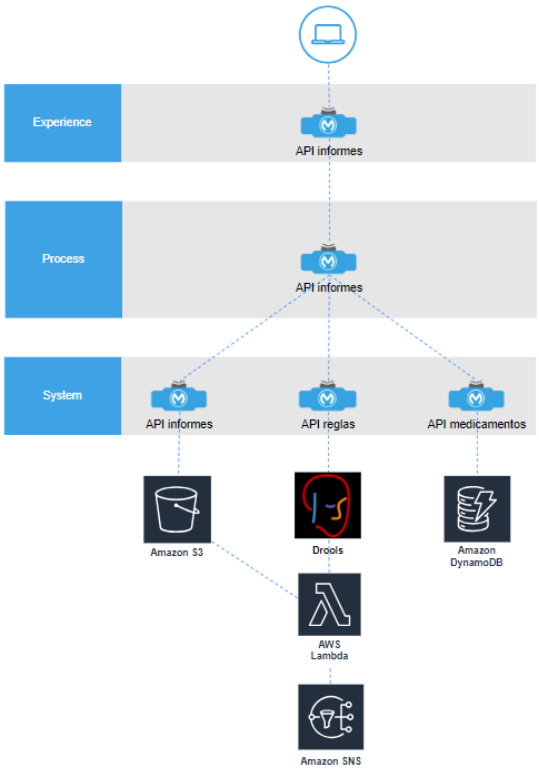

Figura 4.11: Esquema API-led del control de informes y medicamentos

La Figura 4.11 proporciona una representación detallada de cómo estas cinco APIs operan de manera interconectada, enlazando múltiples sistemas finales. Esta integración no solo garantiza una gestión eficiente y automatizada de los informes médicos y del inventario de medicamentos, sino que también mejora la capacidad de una respuesta crítica ante situaciones críticas, como la escasez de medicamentos.

4.4.1. Sistemas involucrados

La solución que estamos desarrollando integra cinco sistemas finales diferentes para llevar a cabo su operativa principal, que es la gestión eficiente de los informes médicos y la administración de medicamentos en las residencias de ancianos.

En primer lugar, el sistema de contenedores en la nube se encarga de almacenar de manera segura los informes médicos de los pacientes residentes. Estos informes, disponibles para ser consultados por los familiares como explicamos en apartados previos, proporcionan una herramienta invaluable para seguir de cerca la evolución del estado de salud de los seres queridos. Asimismo, permiten a enfermeros y médicos realizar un seguimiento detallado y continuo del progreso de los pacientes a lo largo del tiempo, lo que facilita un análisis más preciso y fundamentado de su condición.

Por otra parte, la base de datos, cuya estructura ha sido detallada en operativas anteriores, es fundamental en la gestión de los medicamentos. Esta base de datos se actualiza constantemente para reflejar la cantidad de medicamentos dis-

ponibles en la residencia, descontando aquellos que han sido recetados según lo indicado en los informes médicos. Este proceso garantiza que la información del inventario esté siempre sincronizada con la realidad, minimizando así el riesgo de desabastecimiento. Además, la base de datos proporciona datos actualizados que se integran en el sistema de reglas para la toma de decisiones sobre la gestión de medicamentos.

El sistema de reglas es responsable de aplicar una serie de normas predefinidas que pueden ser geográficas, temporales o basadas en la proximidad a otras residencias dentro del mismo consorcio. Estas reglas son cruciales para determinar la necesidad de realizar pedidos de medicamentos, con el objetivo de anticipar y prevenir posibles escaseces, de este modo se asegura un suministro adecuado y continuo de medicamentos, ajustando a las necesidades específicas de cada residencia. Asimismo, también se encargará de personalizar un mensaje para los familiares de los residentes dónde indicará la existencia de un nuevo informe médico.

El servicio de computación sin servidor desempeña un rol crucial al personalizar el mensaje que se enviará al gerente de la residencia cuando se determine la necesidad de realizar un pedido de un medicamento específico. La personalización del mensaje es esencial para garantizar que la comunicación sea clara y específica, facilitando una respuesta rápida y efectiva por parte del gerente. Además, el sistema de mensajería también se encargará de enviar una notificación a los familiares de los residentes cuando exista un nuevo informe médico a su disposición.

Finalmente, el sistema de mensajería es el encargado de entregar estos mensajes personalizados al gerente de la residencia, garantizando que la notificación se realice a través del medio de comunicación preferido o seleccionado. Este enfoque asegura que la información se transmita de manera oportuna y eficiente, lo que permite una respuesta inmediata a las necesidades de medicamentos, asegurando así un funcionamiento ininterrumpido y seguro de la residencia.

4.4.2. Transformación de datos

Durante el proceso de manejo y transformación de datos se llevan a cabo una serie de pasos meticulosos para asegurar que la información se ajuste a las necesidades específicas del sistema y los usuarios finales. A continuación, describimos en detalle los procesos implicados en la transformación y gestión de los datos.

En primer lugar, en la capa de procesos, los informes se encuentran en formato JSON. En esta etapa se realizan simultáneamente dos transformaciones de datos. La primera transformación consiste en convertir los informes en formato JSON a formato XML. Esta conversión se lleva a cabo con el propósito de facilitar el almacenamiento posterior. El formato XML, debido a su estructura jerárquica y organizada, ofrece ventajas significativas en términos de la organización y accesibilidad de los datos, lo que a su vez facilita su manejo en las etapas siguientes del procesamiento.

Simultáneamente, en la misma capa de procesos, se realiza una segunda transformación que se enfoca en extraer y aislar los datos relacionados con los medica-

mentos contenidos en los informes. El objetivo principal de esta transformación es separar la información específica de cada medicamento para permitir su tratamiento individualizado en las fases siguientes del proceso. La segregación de estos datos es crucial para una gestión más eficiente, facilitando así el análisis y manipulación específica de cada medicamento.

Posteriormente, en la capa de sistemas, durante la fase de almacenamiento, llevamos a cabo una transformación adicional. Esta transformación está orientada a preparar los argumentos necesarios para la interacción con el sistema de computación sin servidor. Concretamente se busca estructurar la información de manera que se pueda generar un mensaje adecuado para los familiares a través de la residencia. Este mensaje tiene como propósito informar a los familiares sobre la disponibilidad de un nuevo informe. La preparación de estos argumentos es esencial para garantizar que la comunicación sea clara y que los destinatarios reciban la información relevante en el momento oportuno.

Además, esta misma transformación se repite en otra API con el objetivo de proporcionar los argumentos necesarios al servicio de computación sin servidor para personalizar el mensaje destinado al gerente de la residencia. En este caso, la personalización del mensaje es fundamental, ya que se busca que el mensaje dirigido al gerente contenga información específica que sea útil para la gestión y supervisión de los medicamentos relacionados.

Finalmente es importante destacar una última transformación de datos que se realiza en relación con los medicamentos actualizados. En este proceso, se lleva a cabo una extracción precisa del nombre del medicamento y su cantidad a partir de los registros de la base de datos obtenidos tras la acción de actualizados. Esta transformación final asegura que solo la información esencial y actualizada del medicamento sea identificada y destacada, facilitando su utilización posterior en el sistema de reglas.

4.4.3. Diseño detallado y diagrama de secuencia

La funcionalidad que estamos desarrollando se implementa a través de múltiples capas interconectadas, cada una de las cuales interactúa con diversas APIs que cumplen roles específicos dentro del sistema.

En primer lugar, abordaremos la capa de experiencia, donde se encuentra la API denominada *e-informes-api*. Esta API es fundamental, pues actúa como el punto de entrada entre el sistema utilizado por el usuario, que en este caso es un software de gestión, y el middleware encargado de gestionar la comunicación entre las distintas capas del sistema. El endpoint principal expuesto por esta API es *post:\informeGeneral*, a través del cual se reciben solicitudes desde el sistema de gestión, permitiendo al usuario activar la funcionalidad requerida. Una vez que la solicitud llega al middleware, este se encarga de propagar la petición hacia la capa de procesos, donde se ejecuta la lógica necesaria para cumplir con la funcionalidad solicitada.

La capa de procesos gestiona nuevamente la solicitud a través del endpoint *post:\informeGeneral*. Este endpoint lleva a cabo dos procesos simultáneos. Inicialmente genera un nombre preliminar para el informe y se realiza una con-

versión de datos que transforma el informe de formato JSON al formato XML. Seguidamente este informe transformado se envía a una API en la capa de sistemas denominada *s-informes-api*, utilizando el endpoint *post:\informes\(id)*. Simultáneamente se ejecuta un segundo proceso que consiste en aislar y tratar individualmente los medicamentos recetados en el informe. Este proceso solo se lleva a cabo si el informe contiene medicamentos recetados, ya que existe la posibilidad de que no se prescriba ninguno. Los medicamentos son procesados uno por uno, enviándolos primero a la API *s-medicamentos-api* a través del endpoint *patch:\medicamentos\(id)*, donde se actualiza la cantidad disponible en en la base de datos. Una vez que se actualizan los datos se realiza una llamada adicional a la API *s-reglas-api* mediante el endpoint *post:\sistemaReglas*, que evalúa si es necesario notificar al gerente en función de las reglas establecidas.

Cada una de las APIs en la capa de sistemas desempeña un papel crucial en la correcta ejecución de la funcionalidad. La API *s-informes-api* recibe solicitudes en el endpoint *post:\informes\(id)*, donde *id* es un parámetro URI que identifica de manera única el informe. Esta API se encarga de recibir el informe en formato XML y de generar un nombre completo para el documento, combinando el parámetro URI con la fecha actual, lo que previene la sobreescritura accidental de informes. Además, se preparan los argumentos necesarios para un servicio de computación sin servidor, que se encargará de enviar notificaciones a los familiares una vez que el informe se haya almacenado correctamente. Por su parte, la API *s-medicamentos-api*, accesible a través del endpoint *patch:\medicamentos\(id)*, utiliza el número CAS del medicamento como identificador único para actualizar la base de datos, descontando la cantidad prescrita en el informe del stock disponible. Después de actualizar los datos, el medicamento se recupera con la información actualizada, y se filtran los datos irrelevantes antes de procesar la respuesta. Finalmente, la API *s-reglas-api* accesible a través del endpoint *post:\sistemasReglas* recibe los datos actualizados del medicamento y los pasa al sistema de reglas, que determina si es necesario enviar una notificación al gerente. En caso de ser necesario se realiza una transformación de datos para preparar los argumentos que se utilizarán en el sistema de computación sin servidor.

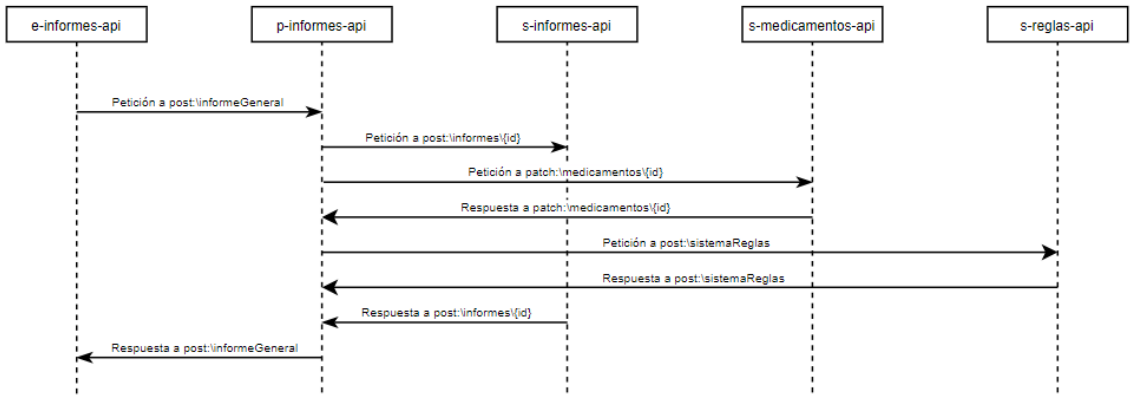

Figura 4.12: Diagrama de secuencia del control de informes y medicamentos

Al observar la Figura 4.12 podemos comprender mejor cómo se ejecutan los procesos en paralelo. La figura muestra que la llamada a *s-informes-api* se realiza inicialmente, aunque su respuesta no se recibe hasta que los otros procesos hayan

finalizado. Este diagrama ha sido diseñado para reflejar un paralelismo, ya que la respuesta de la API de informes de la capa de sistemas puede llegar en cualquier momento, incluso antes de que se realice la llamada a *s-medicamentos-api*. Sin embargo, dado que es necesario esperar a que termine el segundo proceso paralelo para devolver la respuesta a la capa de experiencia se ha decidido representar los eventos de esta manera.

Finalmente es relevante explicar que un parámetro URI (Uniform Resource Identifier) es una secuencia de caracteres que identifica un recurso dentro de una red, como un informe específico o un medicamento en nuestro caso. Este parámetro se utiliza en la URL para localizar recursos de manera eficiente y segura, permitiendo a las APIs acceder a la información adecuada sin necesidad de exponer detalles sensibles en la red. Este enfoque no solo incrementa la seguridad al minimizar la exposición de datos sensibles, sino que también optimiza el control de acceso y reduce el riesgo de ataques de fuerza bruta.

4.5 Estrategia de registros de eventos

Durante el diseño de la solución hemos decidido implementar una estrategia de gestión de registros (logs) que nos permitirá llevar un control sobre la ejecución de los diferentes flujos y facilitar la depuración de errores. La estrategia adoptada se centra en proporcionar información clave de cada flujo sin perjudicar el rendimiento del sistema.

Esta estrategia consiste en añadir registros al inicio y al final de cada flujo para marcar claramente los puntos de entrada y salida, lo que facilitará el seguimiento del ciclo de vida de cada flujo. Además, incluiremos logs antes y después de cada endpoint, así como después de realizar transformaciones de datos clave en los flujos. Estos registros, a diferencia de los demás que están en modo INFO, se registrarán en modo DEBUG. Este enfoque nos permitirá obtener un nivel de detalle mucho mayor sobre el comportamiento de la aplicación, lo cual es esencial para identificar problemas complejos y depurar errores de manera más efectiva.

La inclusión de logs en modo DEBUG es crucial para una revisión detallada de incidencias, ya que proporciona información específica sobre el punto exacto en el que ha ocurrido un error. No obstante, es importante encontrar un equilibrio, ya que una cantidad excesiva de registros podría afectar negativamente el rendimiento de la aplicación. Aunque los logs se ejecutan en hilos asíncronos, lo que minimiza el impacto puntal en el rendimiento de cada flujo, un gran número de hilos dedicados a la generación de logs podría incrementar el uso de CPU y memoria, afectando el rendimiento general del sistema. Es por ello por lo que hemos optado por incorporar únicamente los logs que consideramos imprescindibles, evitando así una degradación significativa en el rendimiento de la aplicación.

CAPÍTULO 5
Desarrollo de la solución propuesta

A lo largo de este capítulo nos enfocaremos en el proceso de implementación de la solución propuesta, proporcionando un análisis de cada etapa involucrada. Iniciaremos con una explicación de las tecnologías seleccionadas para ejecutar el diseño previamente establecido, describiendo también las configuraciones específicas aplicadas a los diversos sistemas finales. Es importante destacar que, debido a la considerable cantidad de APIs integradas en esta solución, dedicaremos una sección significativa a examinar las áreas esenciales de la implementación, de esta forma se garantiza una comprensión integral del proceso al prestar especial atención a las interacciones críticas y los desafíos técnicos que se dan durante la implementación.

5.1 Tecnologías utilizadas

A pesar de la amplia variedad de tecnologías disponibles tanto para la integración de sistemas como para el desarrollo de soluciones finales hemos optado por utilizar MuleSoft y Amazon Web Services (AWS) debido a la notable sinergia que existe entre ambas plataformas. Nuestra elección se fundamenta en la alta compatibilidad que estas herramientas ofrecen, lo que resulta en una integración más fluida y eficiente dentro de nuestra arquitectura tecnológica.

Antes de profundizar en los detalles de MuleSoft y AWS, es importante resaltar que MuleSoft es la tecnología central que hemos estado trabajando en el contexto académico en las clases de Integración de Aplicaciones (IAP), así como en las prácticas profesionales realizadas en DISID. Además, el proyecto que estamos presentando, originado en el Hackathon de integración, fue desarrollado utilizando estas tecnologías, lo que subraya su relevancia y aplicabilidad en escenarios reales.

MuleSoft es una plataforma de integración de aplicaciones y servicios que se ha consolidado como una herramienta fundamental en el ámbito empresarial por su capacidad para conectar de manera eficaz diversos sistemas y aplicaciones. MuleSoft se basa en un motor de tiempo de ejecución que actúa como un bus de servicios empresariales (ESB) desarrollado en Java, lo que habilita un intercambio fluido de datos entre aplicaciones, independientemente de las tecnologías en las que estén basadas. Este ESB posee la flexibilidad de desplegarse en cualquier

entorno, proporcionando conectividad universal y permitiendo la integración y coordinación de eventos tanto en tiempo real como por lotes. [24]

Una de las principales fortalezas de MuleSoft radica en su capacidad para habilitar la comunicación entre distintas aplicaciones, para lograrlo esto ofrece funcionalidades avanzadas como la creación y alojamiento de servicios, lo cual permite exponer y reutilizar servicios en un contenedor ligero. Esta plataforma también refuerza la seguridad en la comunicación entre servicios mediante la mediación, lo que asegura que la lógica empresarial permanezca separada de la mensajería, garantizando que las invocaciones a servicios sean independientes de la ubicación física de los mismos. [24]

Además, esta plataforma destaca en el enrutamiento de mensajes, facilitando la aplicación de reglas específicas para filtrar, agregar y reorganizar los mensajes conforme a las necesidades del entorno empresarial. En este sentido, su capacidad para transformar datos es crucial, ya que permite el intercambio de información entre diferentes formatos y protocolos de transporte. Esta funcionalidad es especialmente relevante en contextos empresariales caracterizados por una diversidad tecnológica considerable. La plataforma MuleSoft se compone de varias tecnologías entre las que destacan Anypoint Studio y Anypoint Platform, que describiremos con mayor detalle posteriormente. [24]

Por otro lado, Amazon Web Services se ha consolidado como un proveedor líder en servicios de computación en la nube, ofreciendo una extensa gama de soluciones que incluyen almacenamiento, capacidad de cómputo y gestión de bases de datos, todas bajo un modelo de pago por uso. Esta modalidad permite gestionar costos y recursos de manera óptima, lo que constituye un factor clave en la eficiencia operativa de las organizaciones. Una de las ventajas más significativas que ofrece AWS es su amplia capacidad de integración con otros servicios y herramientas, aspecto que resulta particularmente beneficioso para el desarrollo de nuestro middleware. [25]

La infraestructura global de AWS se distingue no solo por su amplitud, sino también por su asequibilidad, accesibilidad y robustas características de visibilidad y gobernanza. Estas características permiten a las organizaciones gestionar recursos de manera eficaz, garantizando la continuidad operativa incluso frente a fallos en servidores individuales o centros de datos completos. [25]

El modelo de responsabilidad compartida de AWS es otro elemento crucial, ya que asegura que la infraestructura subyacente esté protegida contra amenazas de seguridad, ataques de denegación de servicio y problemas de privacidad de datos. Sin embargo, también es importante destacar que este modelo requiere que los usuarios implementen sus propias medidas de seguridad para proteger sus aplicaciones y datos, lo que subraya la necesidad de una gestión activa y consciente por parte de los usuarios. [25]

Durante los siguientes apartados abordaremos en detalle algunos de los servicios clave de AWS que utilizaremos: Amazon DynamoDB, Amazon S3, Amazon SageMaker, Amazon Lambda y Amazon SNS. Estos servicios representan pilares fundamentales para el desarrollo y la operación de nuestra infraestructura tecnológica, permitiéndonos aprovechar al máximo las capacidades que AWS pone a nuestra disposición.

Finalmente, hay que destacar que, aunque el software de gestión de residencias utilizado actualmente ya incluye funcionalidades para realizar el inventario de medicamentos y almacenar informes, como se explicó en la Sección 3.2, hemos decidido incorporar una nueva base de datos y un sistema de contenedores en la nube para almacenar los informes. Esta decisión se fundamenta en las múltiples ventajas que ofrece el uso de sistemas basados en AWS, que no solo facilita la integración y comunicación entre los diferentes componentes de la solución, sino que también asegura una mayor cohesión con los sistemas de Amazon Web Services que hemos implementado.

5.1.1. Anypoint Platform

Anypoint Platform es una solución integral que aborda de manera efectiva la administración de integraciones complejas entre aplicaciones y servicios, independientemente de si estos operan en entornos locales, en la nube o en configuraciones híbridas. Esta plataforma se clasifica dentro del modelo iPaaS (Integration Platform-as-a-Service), lo que implica que proporciona un conjunto integral de herramientas y servicios que cubren todas las fases del ciclo de vida de las APIs.

Uno de los componentes fundamentales de esta plataforma es el Design Center, una herramienta que permite diseñar, construir y gestionar APIs de manera eficiente. Dentro de este entorno el lenguaje RAML (RESTful API Modeling Language) adquiere una importancia crucial al facilitar la definición estructurada y comprensible de las APIs. Este lenguaje permite especificar recursos, métodos, parámetros y, además, generar automáticamente la documentación necesaria. Este enfoque no solo optimiza el proceso de desarrollo, sino que también garantiza que las APIs estén bien documentadas y sean fácilmente comprensibles.

Otro elemento esencial es Anypoint Exchange, que actúa como un repositorio centralizado y colaborativo. Este espacio permite descubrir, compartir y reutilizar recursos, tales como conectores, plantillas, ejemplos de código y fragmentos de APIs. Este repositorio es fundamental para incrementar la eficiencia ya que evita la duplicación de esfuerzos y acelera significativamente el proceso de integración mediante la reutilización de componentes preexistentes.

El API Manager se presenta como la herramienta principal dentro de Anypoint Platform para la gestión centralizada de APIs. A través de esta herramienta se pueden implementar políticas de seguridad, establecer controles de acceso, monitorear el rendimiento y proteger las APIs mediante mecanismos avanzados de autenticación.

Finalmente, Anypoint Platform ofrece el Runtime Manager, un entorno robusto para desplegar, monitorear y administrar integraciones. Dentro de esta herramienta destaca CloudHub, la infraestructura en la nube que permite a las organizaciones desplegar aplicaciones e integraciones en un entorno completamente gestionado y escalable. CloudHub simplifica el proceso de despliegue al encargarse de la gestión de servidores, escalabilidad automática y alta disponibilidad, liberando a las organizaciones de la preocupación por la infraestructura subyacente.

5.1.2. Anypoint Studio

Anypoint Studio es un entorno de desarrollo integrado (IDE) de alto rendimiento. Esta plataforma ha sido diseñada con el propósito específico de facilitar la creación, implementación y gestión de soluciones de integración, asegurando una compatibilidad óptima con las tecnologías y servicios proporcionados por MuleSoft.

La interfaz gráfica de Anypoint Studio se distingue por su intuitividad, permitiendo diseñar rutas de integración utilizando la técnica de arrastrar y soltar. Este enfoque gráfico reduce la necesidad de codificación manual en cada etapa del proceso de desarrollo, lo que simplifica significativamente la construcción de flujos de integración complejos. La representación visual de la estructura de los flujos no solo facilita la comprensión inmediata del código, sino que también mejora la colaboración entre equipos de trabajo y acelera la identificación y resolución de problemas.

Adicionalmente, Anypoint Studio incorpora una amplia variedad de componentes y módulos predefinidos que pueden configurarse y combinarse para implementar transformaciones de datos y lógica de negocio de acuerdo con los requisitos específicos de cada proyecto. Una de las tecnologías que destaca es DataWeave, un lenguaje de transformación de datos de MuleSoft que está completamente integrado en Anypoint Studio. DataWeave permite a los desarrolladores realizar transformaciones de datos entre diversos formatos (como JSON, XML, CSV...) de una manera declarativa y concisa.

La integración de DataWeave no solo simplifica el proceso de transformación de datos, sino que también posibilita la escritura de estas transformaciones directamente dentro del entorno de desarrollo. Esto aprovecha las capacidades avanzadas del lenguaje y la interfaz visual de Anypoint Studio para optimizar y visualizar las operaciones de transformación de datos. Esta funcionalidad es particularmente valiosa para la creación de soluciones de integración complejas, ya que DataWeave ofrece un método robusto y eficiente para gestionar y transformar grandes volúmenes de datos entre múltiples sistemas y aplicaciones.

5.1.3. Amazon S3

Amazon Simple Storage Service (Amazon S3) es un servicio de almacenamiento de objetos altamente escalable, diseñado para ofrecer niveles de disponibilidad, seguridad y rendimiento. Este servicio es ideal para una amplia gama de casos de uso, incluyendo la gestión de datos, la administración de sitios web, el soporte a aplicaciones móviles, la realización de copias de seguridad, el archivado, la ejecución de aplicaciones empresariales, la integración con dispositivos IoT y el análisis de grandes volúmenes de datos.

Amazon S3 ofrece diversas clases de almacenamiento que se ajustan a diferentes necesidades específicas:

- S3 Standard y S3 Express One Zone: Ideal para datos críticos y de acceso frecuente con baja latencia y costos reducidos.

- S3 Standard-IA y S3 One Zone-IA: Pensado para datos de acceso poco frecuente con opciones más económicas y almacenamiento en una sola zona.
- S3 Glacier y sus variantes (S3 Glacier Instant Retrieval, S3 Glacier Flexible Retrieval, S3 Glacier Deep Archive): Diseñado para archivado a bajo costo con diferentes tiempos de recuperación según la variante.

Además, Amazon S3 incluye la funcionalidad S3 Intelligent-Tiering, que optimiza los costos mediante la migración automática de datos entre cuatro niveles de acceso, según los patrones de uso observados. [26]

Amazon S3 se desempeña como un sistema de contenedores en la nube en nuestro proyecto. Este sistema se destinada a la gestión de informes y parámetros relacionados con el bienestar de los residentes.

Hemos configurado este sistema de manera sencilla y eficiente, estableciendo diez contenedores distintos en Amazon S3, cada uno con un propósito específico que responde a las diferentes necesidades operativas identificadas. Uno de los contenedores está dedicado exclusivamente al almacenamiento de informes y otro contenedor se utiliza para almacenar inicialmente los parámetros de salud, actuando como el punto de entrada para la recopilación de estos datos cruciales. Los ocho contenedores restantes están diseñados para clasificar los datos según el tipo de anomalía detectada, permitiendo así una categorización detallada.

Además, hemos implementado una configuración adicional que incluye la creación de un desencadenador (trigger). Este desencadenador se activa cada vez que se inserta un nuevo archivo en el contenedor de parámetros de salud, lo que inicia una función del servicio de computación sin servidor. Esta función es responsable de coordinar la activación de los sistemas de aprendizaje automático y del sistema de mensajería, lo que optimiza aún más el flujo de trabajo y garantiza la rápida respuesta ante cualquier situación que requiera intervención.

5.1.4. Amazon DyanmoDB

Amazon DynamoDB es una base de datos NoSQL completamente gestionada y sin servidor, diseñada para ofrecer un rendimiento de baja latencia en milisegundos independientemente de la escala requerida. Esta tecnología se orienta a superar las limitaciones y complejidades operativas asociadas comúnmente con las bases de datos relacionales, proponiéndose como una alternativa robusta que optimiza tanto el rendimiento como la escalabilidad.

Una de las características distintivas más relevantes de DynamoDB es su modo de capacidad bajo demanda. Este permite un escalado automático y un ajuste dinámico de la capacidad en función de las cargas de trabajo, garantizando así que las tablas puedan gestionar picos de tráfico sin inconvenientes.

DynamoDB está diseñada para proporcionar un rendimiento superior, acompañado de una escalabilidad y flexibilidad significativamente mayores en comparación con las bases de datos relacionales tradicionales. Su arquitectura es compatible tanto con modelos de datos clave-valor como de documentos, lo que la hace aplicable a una amplia variedad de casos de uso. Además, DynamoDB soporta transacciones ACID (Atomicidad, Consistencia, Aislamiento y Durabilidad) y

ofrece consistencia fuerte en las lecturas, lo que la convierte en una opción confiable para aplicaciones empresariales críticas. [27]

Amazon DynamoDB se utiliza como base de datos principal para gestionar los medicamentos en nuestra solución. Esta elección permite manejar eficientemente un volumen considerable de datos y solicitudes, garantizando que la información sobre los medicamentos esté disponible de manera rápida y confiable para los residentes y el personal, independientemente de la carga de trabajo o la cantidad de usuarios.

La gestión de los diferentes medicamentos se realiza mediante una tabla en DynamoDB que contiene tres columnas principales, cada una de las cuales cumple una función específica en la organización y accesibilidad de los datos.

La primera columna, denominada idMedicamento, está asociada al número CAS del fármaco. Aunque el número CAS es fundamental para identificar el tipo de medicamento, su principal función en esta columna es facilitar la categorización y el reconocimiento general del medicamento dentro del sistema. La segunda columna, etiquetada como cantidad, refleja el número de unidades disponibles para cada tipo de medicamento. Esta columna proporciona una visión precisa del inventario actual, permitiéndonos mantener un control detallado sobre la disponibilidad de cada producto, lo que es esencial para una gestión efectiva del stock. Finalmente, la tercera columna, nombreMedicamento, contiene el nombre comercial del fármaco. La inclusión de esta columna es vital para una gestión eficiente del inventario, ya que facilita la identificación rápida y precisa de los productos dentro del sistema, asegurando así una localización efectiva y la administración adecuada de los medicamentos disponibles.

5.1.5. Amazon SageMaker

Amazon SageMaker es un servicio de aprendizaje automático (ML) completamente gestionado que facilita a científicos de datos y desarrolladores la construcción, el entrenamiento y el despliegue de modelos de machine learning de forma rápida y confiable en un entorno de producción listo para ser utilizado. SageMaker proporciona una interfaz de usuario intuitiva que optimiza la ejecución de flujos de trabajo de ML y permite que sus herramientas estén disponibles en múltiples entornos de desarrollo integrados (IDEs), lo que simplifica el proceso de desarrollo y mejora la eficiencia en la gestión de proyectos de ML.

Una de las principales ventajas de SageMaker es su capacidad para almacenar y compartir datos sin la necesidad de construir y gestionar infraestructuras de servidores propias. Esta funcionalidad permite que nos enfoquemos más en el diseño y desarrollo colaborativo de flujos de trabajo de ML, reduciendo el tiempo necesario para alcanzar resultados. SageMaker también ofrece algoritmos de ML gestionados que operan eficientemente con grandes volúmenes de datos en un entorno distribuido. Adicionalmente, brinda soporte integrado para utilizar algoritmos y frameworks personalizados, ofreciendo opciones de entrenamiento distribuidas y flexibles que se adaptan a las necesidades específicas de nuestros flujos de trabajo. [28]

Esta tecnología se utiliza en nuestro proyecto con el objetivo de analizar los parámetros de salud de los residentes. Una vez que los modelos son entrenados y configurados correctamente, SageMaker se convierte en una herramienta fundamental para la toma de decisiones informadas basadas en datos.

```python
import sagemaker
from sagemaker import get_execution_role
import boto3
import pandas as pd

# Configuración del entorno de SageMaker
role = get_execution_role()
session = sagemaker.Session()
bucket = 'parametros-pulseras-pacientes'  # Reemplaza con el nombre de tu bucket

# Leer y preprocesar el archivo CSV
df = pd.read_csv(f's3://{bucket}/train_data.csv')

# Verificar que las columnas están correctamente leídas
print(df.columns)

# Función para entrenar el modelo RCF para una columna
from sagemaker import RandomCutForest

def train_rcf(column_data, column_name):
    rcf = RandomCutForest(
        role=role,
        instance_count=1,
        instance_type='ml.m5.large',
        num_samples_per_tree=100,
        num_trees=100,
        feature_dim=1  # Una característica por modelo
    )
    rcf.fit(rcf.record_set(column_data.values.reshape(-1, 1)))
    rcf_inference = rcf.deploy(
        initial_instance_count=1,
        instance_type='ml.m5.large'
    )
    print(f'Model for {column_name} trained and deployed successfully.')
    return rcf_inference

# Entrenar un modelo RCF para cada columna
models = {}
for column in df.columns:
    models[column] = train_rcf(df[column], column)

# Verificar que los modelos están desplegados y las claves están en el diccionario
print("Model keys:", models.keys())
```

Figura 5.1: Código desarrollado para crear el modelo de machine learning utilizado

La implementación de este sistema se realiza mediante el desarrollo de un script específico, como se detalla en la Figura 5.1. Este script tiene la tarea de procesar una serie de archivos de entrenamiento que contienen un total de 10.000 muestras aleatorias, las cuales representan valores considerados dentro de los parámetros normales de salud.

El script ejecuta varias funciones clave. Inicialmente genera tres endpoints distintos en SageMaker, cada uno dedicado a un parámetro de salud específico. Posteriormente entrena y despliega estos endpoints de manera individual. El objetivo de esta configuración es evaluar cada dato de manera independiente y determinar si presenta una anomalía en relación con los parámetros de salud previamente establecidos.

5.1.6. Amazon Lambda

Amazon Lambda es un servicio de computación sin servidor que ofrece la capacidad de ejecutar código sin la necesidad de administrar la infraestructura subyacente. Este servicio asume la responsabilidad completa del mantenimiento del servidor y del sistema operativo, la asignación y gestión de la capacidad, el escalado automático, y la monitorización de eventos. Al ejecutar el código en una infraestructura altamente disponible Lambda se encarga de toda la administración de los recursos computacionales, lo que permite que nos concentremos exclusivamente en el desarrollo y optimización de nuestro código. [29]

Amazon Lambda es utilizado durante la implementación y diseño del proyecto para ejecutar diversas tareas esenciales que automatizan y optimizan nuestros

procesos. Hemos desarrollado un total de tres funciones Lambda que cumplen con roles específicos dentro del sistema.

Las dos primeras funciones Lambda tienen como objetivo principal personalizar un mensaje que se enviará a los familiares o a los gerentes de la residencia, dependiendo de la función correspondiente. Este proceso es sencillo y se puede observar en la Figura 5.2, que muestra el código desarrollado para enviar un mensaje a los familiares, informándoles sobre la disponibilidad de un nuevo informe. Es importante destacar que el nombre y apellido, tal como se describió en los distintos diseños previos, son proporcionados previamente a la invocación de esta función para asegurar la personalización del mensaje.

```python
import json
import boto3

def lambda_handler(event, context):
    nombre = event['nombre']
    apellido1 = event['apellido1']
    client = boto3.client("sns")

    mensaje = f"Nuevo informe medico disponible de {nombre} {apellido1}. ¡Ya puede consultarlo en la web!"
    response = client.publish(
        TopicArn="arn:aws:sns:us-east-1:891377142630:notificarInforme",
        Message = mensaje
        )

    return mensaje
```

Figura 5.2: Función Lambda que permite la notificación a familiares

Por otro lado, hemos implementado una tercera función Lambda de mayor complejidad, la cual se activa automáticamente cada vez que se almacena un archivo en el contenedor de Amazon S3, donde inicialmente se guardan los parámetros de bienestar. Una vez activada esta función descarga el archivo recién insertado y procesa cada columna del archivo CSV, enviando los datos a su correspondiente endpoint para ser analizados por el modelo de machine learning de AWS SageMaker. Una vez obtenidos los resultados del análisis realizado por SageMaker, la función interpreta estos resultados para determinar en qué contenedor de Amazon S3 mover finalmente el archivo, dependiendo de los resultados obtenidos. Por último, la función elabora un mensaje dirigido a los enfermeros, y en caso de detectar alguna anomalía, publica este mensaje a través del sistema de mensajería, de manera similar a lo mostrado en la Figura 5.2.

5.1.7. Amazon SNS

Amazon Simple Notification Service (Amazon SNS) es una solución gestionada que optimiza la distribución de mensajes entre publicadores y suscriptores, también denominados productores y consumidores. Este servicio permite que los publicadores envíen mensajes de forma asincrónica a un tema, que actúa como un punto de acceso lógico y canal de comunicación central. Los clientes pueden suscribirse a un tema específico de Amazon SNS para recibir los mensajes publicados a través de diversos tipos de endpoints compatibles, tales como Amazon Data Firehose, Amazon SQS, AWS Lambda, HTTP, correo electrónico, notificaciones push en dispositivos móviles y mensajes de texto (SMS). [30]

Nuestra solución implementa esta tecnología avanzada de mensajería con el objetivo de enviar notificaciones importantes a familiares, enfermeros y gerentes. Este sistema asegura que la información crítica llegue de manera rápida y efectiva

a las personas adecuadas, garantizando una comunicación fluida y oportuna en situaciones clave.

Estas notificaciones llegan a los diferentes destinatarios a través de tres temas distintos, cada uno destinado a un grupo específico: uno para enfermeros, otro para familiares y un tercero para gerentes. Gracias a esto cada grupo recibe únicamente las notificaciones relevantes para su función. Mediante la asignación de diferentes correos electrónicos podemos comprobar que las distintas notificaciones se reciben correctamente.

5.1.8. Drools

Drools es un sistema de gestión de reglas de negocio (BRMS, por sus siglas en inglés) de código abierto desarrollado en Java, que ofrece una amplia variedad de funcionalidades entre las que destacan un motor de reglas de negocio, herramientas de autoría web, una aplicación para la gestión de reglas, y soporte en tiempo de ejecución para modelos basados en Decision Model and Notation (DMN). Este software facilita la separación y el razonamiento lógico sobre los datos y procesos empresariales, lo que nos permite optimizar la toma de decisiones a través de la automatización y mejora en la eficiencia de estas.

El funcionamiento de Drools se articula en dos componentes fundamentales: la Autoría y el Tiempo de Ejecución. Durante la fase de autoría nos enfocamos en el desarrollo y definición de los archivos de reglas, donde se especifican de manera precisa las condiciones y acciones que el sistema debe ejecutar. Posteriormente, en la fase de tiempo de ejecución se establece una memoria de trabajo en la que se gestionan tanto la activación como el seguimiento de las reglas previamente definidas, garantizando así que estas se apliquen correctamente en el contexto adecuado.

Una de las características más destacadas de Drools es su implementación del algoritmo de coincidencia de patrones Rete. Este algoritmo incrementa significativamente la eficiencia en la evaluación de reglas complejas y el procesamiento de grandes volúmenes de datos. La optimización lograda a través del algoritmo Rete es crucial en escenarios donde la velocidad y precisión en la toma de decisiones son de vital importancia para el éxito del negocio. [31]

En nuestro proyecto hemos optado por utilizar Drools como herramienta para determinar, a partir de la aplicación de un conjunto de reglas predefinidas, si es necesario o no notificar al gerente sobre la necesidad de realizar nuevos pedidos. Esto nos permite asegurar una gestión eficiente y oportuna de los recursos, minimizando el riesgo de desabastecimiento y optimizando la cadena de suministro.

No obstante, es importante señalar que, debido a las restricciones temporales y a la complejidad propia a la configuración de Drools, no hemos logrado integrar plenamente esta tecnología en nuestra solución actual, es por ello que hemos realizado una simulación de su funcionamiento en la solución. Esta limitación nos impide ofrecer detalles específicos sobre la configuración y personalización del sistema en nuestro caso particular.

5.2 Implementación de las funcionalidades

Antes de profundizar en las implementaciones operativas más relevantes es fundamental establecer una base común que garantice una ejecución coherente. Esto incluye la creación de RAMLs, la configuración de propiedades fundamentales, y la definición de configuraciones específicas para las llamadas entre APIs y sistemas. Asimismo, también resulta indispensable organizar los flujos y subflujos de trabajo mediante archivos de configuración de Mule.

Una vez establecidos estos elementos generales, procederemos a explorar en detalle cada una de las operativas, enfocándonos en los flujos más relevantes detallados durante el anterior capítulo, y cómo su implementación contribuye a la correcta ejecución de las operativas definidas en las APIs.

Inicialmente abordaremos la creación de RAML, un componente esencial en el diseño e implementación de APIs dentro de la plataforma MuleSoft. Anypoint Platform facilita significativamente este proceso a través del Design Center, una herramienta robusta que nos permite diseñar y especificar nuestras APIs con un alto grado de detalle.

Las especificaciones en RAML son fundamentales al permitir a MuleSoft generar automáticamente la estructura básica de la API, incluyendo todos los endpoints necesarios. Este enfoque no solo simplifica la implementación de la API al proporcionar una base sólida, sino que también garantiza que la API siga un diseño coherente y bien definido desde el principio. Por lo tanto, la creación de un RAML es crucial, ya que actúa como un plano detallado de nuestra API. Además, el RAML describe el funcionamiento de la API, permitiendo anticipar posibles problemas y optimizar su diseño.

El RAML nos brinda la capacidad de definir aspectos fundamentales de nuestras APIs, como los tipos de datos que se utilizarán en las definiciones de los endpoints y sus respectivas respuestas. Esto incluye la definición de cada endpoint con sus métodos correspondientes, así como los parámetros URI y de consulta, que son esenciales en ciertos casos para permitir que los clientes transmitan información adicional a la API.

Asimismo, RAML facilita la creación y gestión de estos elementos mediante su sintaxis clara y bien estructurada. El Design Center ofrece valiosas ayudas como sugerencias automáticas, validación en tiempo real y plantillas reutilizables, lo que simplifica enormemente el proceso de diseño y reduce la probabilidad de errores.

La Figura 5.3 ilustra la especificación que hemos utilizado en la implementación de nuestra solución para la API *s-ia-api*. Este ejemplo muestra cómo se emplea la declaración *type:* para definir un tipo de datos personalizados que se utilizará en el endpoint correspondiente. Es notable que los diferentes objetos del dato personalizado no cuentan con un *?* lo que indica que dichas datos no son opcionales. Seguidamente, el endpoint */datosSalud* se define acompañado de la especificación del método HTTP y los posibles códigos de respuesta que la API devolverá al cliente.

Todas las APIs cuentan con una especificación RAML que define su estructura y comportamiento. Sin embargo, es fundamental tener en cuenta las dependen-

cias que pudieran aparecer entre las diferentes APIs que se interconectan, ya que esto garantiza una correcta comunicación y funcionamiento en conjunto. La coordinación adecuada de estas dependencias es clave para asegurar la coherencia y eficiencia en los procesos que involucran múltiples APIs.

```
#%RAML 1.0
title: s-ia-api
mediaType:
  - application/json
protocols:
  - HTTP
types:
  datosBinestar:
    description: Objeto que representa los datos de bienestar.
    type: object
    properties:
      nombrePersonaCompleto:
        type: string
        example: Juan Sánchez
      oxigenoSangre:
        type: number
        format: int
        example: 97
      respiracionesMinuto:
        type: number
        format: int
        example: 16
      ritmoCardiaco:
        type: number
        format: int
        example: 65

/datosSalud:
  post:
    displayName: Almacenar datos para posterior procesamiento de la machine learning.
    description: Almacena los datos en Amazon S3 para que posteriormente la machine learning los analice.
    body:
      type: datosBinestar
    responses:
      "200":
        description: Información almacenada correctamente.
      "400":
        description: Solicitud incorrecta. Los parámetros proporcionados no son válidos.
      "500":
        description: Error interno del servidor.
```

Figura 5.3: RAML de la API s-ia-api

Una vez importado el RAML diseñado en nuestra API es necesario proceder con la creación de dos archivos de configuración adicionales a los generados durante la importación inicial. Estos archivos, denominados global.xml e implementation.xml, juegan un papel crucial en la organización y funcionalidad del proyecto, aportando beneficios que van más allá de la mera estructura del código.

En primer lugar, el archivo global.xml tiene la responsabilidad de centralizar las configuraciones clave de los conectores, tales como las solicitudes HTTP y las configuraciones del sistema final o del Router de Mule. Al establecer estas configuraciones en un solo lugar se facilita no solo la gestión y reutilización de componentes comunes, sino también la actualización o modificación de estas configuraciones, reduciendo significativamente la duplicación de código y simplificando el proceso de mantenimiento.

Por otro lado, el archivo implementation.xml está dedicado exclusivamente a la lógica de negocio que se ejecuta cada vez que un endpoint recibe una solicitud. Esta separación modular entre la lógica de negocio y la configuración de infraestructura permite un desarrollo más ordenado y manejable. La claridad en la organización facilita el proceso de depuración, ya que la distinción entre la lógica de negocio y las configuraciones de conexión y enrutamiento permite aislar y resolver problemas con mayor facilidad.

Al iniciar el diseño de una nueva API, uno de los pasos fundamentales después de la creación de los dos archivos de configuración adicionales es la generación de los archivos de propiedades. Estos archivos son cruciales ya que nos permiten establecer configuraciones clave, como el puerto a través del cual la API recibirá las solicitudes, así como las propiedades relacionadas con las llamadas a otras APIs y sistemas que utilizaremos en el proyecto.

```yaml
api:
  name: "e-medicamentos-api"
  port: "8090"

muleApis:
  p-medicamentos-api:
    protocol: "HTTP"
    host: "localhost"
    port: "8091"
    path: "/api"

dynamoDb:
  AccessKey: "AKIA47CRXZNTCUSZGBHL"
  SecretKey: "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"

```

Figura 5.4: Ejemplo de archivo de propiedades

La Figura 5.4 ilustra la configuración inicial del puerto de la API, junto con otros parámetros como el host, el protocolo y la base de la ruta común para la API *p-medicamentos-api*. Además, en dicha figura se presenta la configuración correspondiente a la base de datos DynamoDB. Es importante destacar que este archivo no corresponde a ninguno utilizado en la solución final del proyecto, se presenta únicamente como un ejemplo para ilustrar las diversas propiedades que puede incluir este archivo, denominado *local.yaml*.

En el contexto de despliegues, como es el caso de Cloudhub, es recomendable disponer de varios archivos de configuración adaptados a los diferentes entornos (desarrollo, pruebas, producción, etc.). Esta práctica permite una mayor flexibilidad y control al configurar la API para cada ambiente específico, asegurando así que las características y restricciones de cada entorno sean debidamente consideradas.

Aunque la sintaxis y la estructura para definir los elementos en estos archivos son consistentes, los detalles específicos, como las configuraciones de bases de datos u otros sistemas externos, pueden variar significativamente dependiendo del sistema que estemos utilizando.

Finalmente expondremos en detalle un aspecto común en todas las APIs siempre que hacemos una petición HTTP a otra API o cuando queremos utilizar un sistema final. La configuración de los diversos conectores es crucial para garantizar una comunicación eficiente y precisa entre los diferentes componentes del sistema. Estos desempeñan un papel fundamental al especificar cómo se establecen las conexiones, asegurando que la transmisión de datos sea segura y que las solicitudes lleguen correctamente a su destino.

En la Figura 5.5 se presenta un ejemplo de configuración para una llamada HTTP destinada a interactuar con la API denominada *p-medicamentos-api*. En esta figura, se puede observar que el host, el puerto y la base común de la ruta se definen utilizando una sintaxis particular *${...}* en lugar de valores explícitos.

Esto indica que dichos valores se obtienen dinámicamente desde un archivo de propiedades. Esta técnica permite que la configuración sea más flexible y reutilizable, ya que los valores reales pueden cambiar sin necesidad de modificar el código fuente.

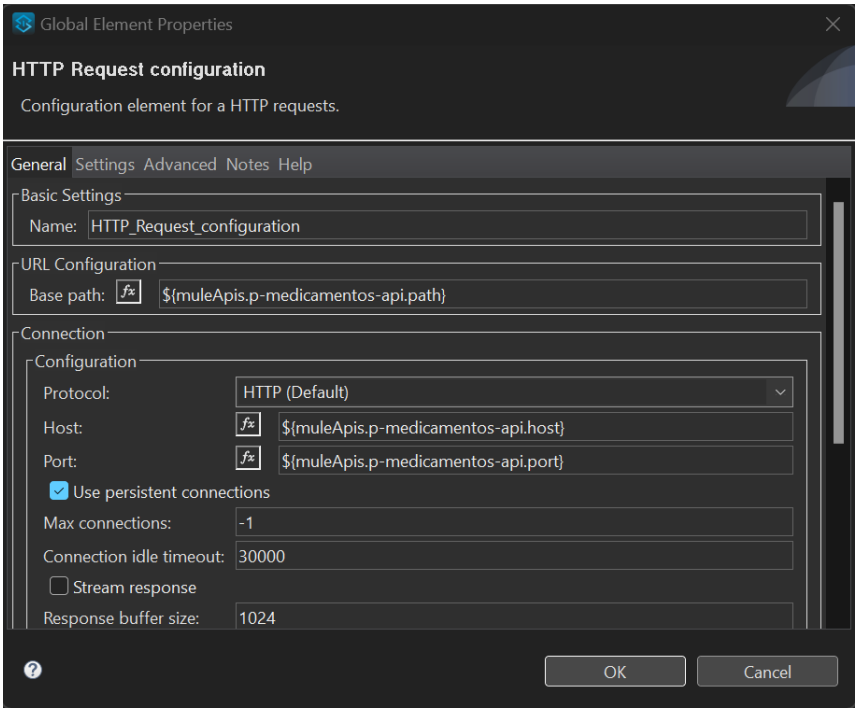

Figura 5.5: Ejemplo de archivo de configuración de una llamada HTTP

Comparando con la Figura 5.4 podemos entender mejor cómo estos valores son recuperados y aplicados en tiempo de ejecución. Este enfoque permite una configuración dinámica y adaptable según las necesidades del entorno en el que se despliega la aplicación.

5.2.1. Monitorización de los parámetros de bienestar

La implementación de la operativa descrita en la Sección 4.1.3 se caracteriza por una complejidad moderada. Las APIs de la capa de experiencia y la capa de procesos desempeñan el papel principal de propagar las solicitudes hacia la capa de sistemas sin llevar a cabo transformaciones de datos significativas. No obstante, el endpoint de la capa de sistemas presenta una complejidad mayor debido a la necesidad de convertir los datos, crear variables y almacenar los parámetros de salud que son monitoreados.

La Figura 5.6 ilustra el subflujo al que redirecciona el endpoint *post:\datosSalud* tras recibir una solicitud mediante un *Flow Reference*. Esta figura muestra el proceso detallado que se sigue para llevar a cabo las acciones descritas. En primer lugar, se crea una variable que representa el nombre del documento, este nombre se construye a partir del nombre de la persona que lleva la pulsera, el cual se obtiene del cuerpo de la solicitud, seguido de un guion y la hora en que se almacena el informe. Esta asignación de nombre se realiza previamente a la conversión de los datos al formato CSV con el objetivo de evitar una complejidad adicional en esta etapa del proceso.

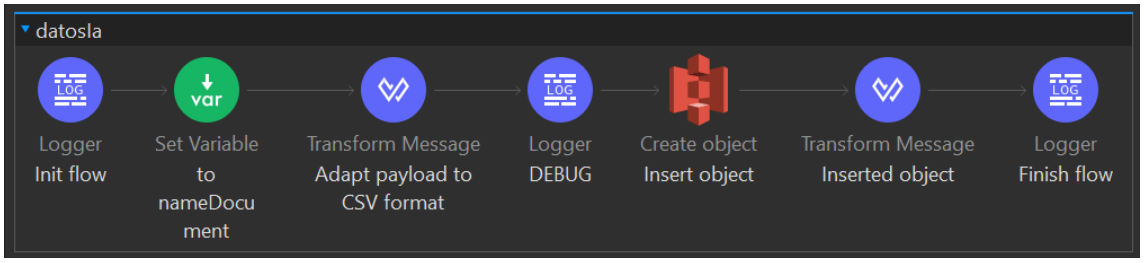

Figura 5.6: Subflujo del endpoint post:\datosSalud

Posteriormente se procede al almacenamiento de los datos mediante el conector *Create Object* de Amazon S3. Durante esta fase se especifica el contenedor en el que se almacenará el objeto, el nombre del objeto y su contenido.

Finalmente, se envía una respuesta al cliente para confirmar que la operación se ha completado de manera exitosa. Esta confirmación asegura al cliente que la solicitud ha sido procesada y que los datos han sido gestionados correctamente.

5.2.2. Gestión integral de medicamentos

A continuación, describiremos de manera detallada las implementaciones más significativas dentro de la operativa que permite una gestión integral de medicamentos. Esta operativa está estructurada en varios submódulos interconectados que facilitan la ejecución de diferentes funciones relacionadas con la administración de medicamentos.

El primer submódulo se enfoca en la inserción y eliminación de medicamentos en la base de datos. La implementación más relevante de este proceso reside en la API de sistemas, mientras que las APIs de proceso y experiencia se limitan a la propagación de las solicitudes. Dichas solicitudes se transmiten desde la capa de experiencia hasta la capa de sistemas y consisten en un mensaje que puede incluir hasta tres elementos. Sin embargo, tanto la cantidad de medicamentos como su nombre son opcionales, dependiendo del tipo de operación que se desee realizar. La operación en cuestión se determina a través de un parámetro de consulta que puede adoptar los valores borrar o insertar, definiendo así la acción que se llevará a cabo.

En el transcurso de este subflujo de la API de sistemas, tal como se ilustra en la Figura 5.7, el parámetro de consulta y el número de elementos en el cuerpo del mensaje desempeñan un papel fundamental. Inicialmente, en el flujo del endpoint se almacena el número de elementos del cuerpo del mensaje en una variable. Luego, mediante el uso de un parámetro *Choice*, el flujo se redirige según la operación requerida. La condición para insertar un medicamento requiere que el cuerpo del mensaje contenga tres elementos, y el parámetro de consulta debe estar configurado como insertar. Por el contrario, para eliminar un medicamento, el parámetro de consulta debe ser borrar, y el cuerpo del mensaje solo debe incluir un elemento: el identificador del medicamento. En caso de que estas condiciones no se cumplan se devuelve un mensaje al cliente notificando que la inserción o eliminación ha fallado.

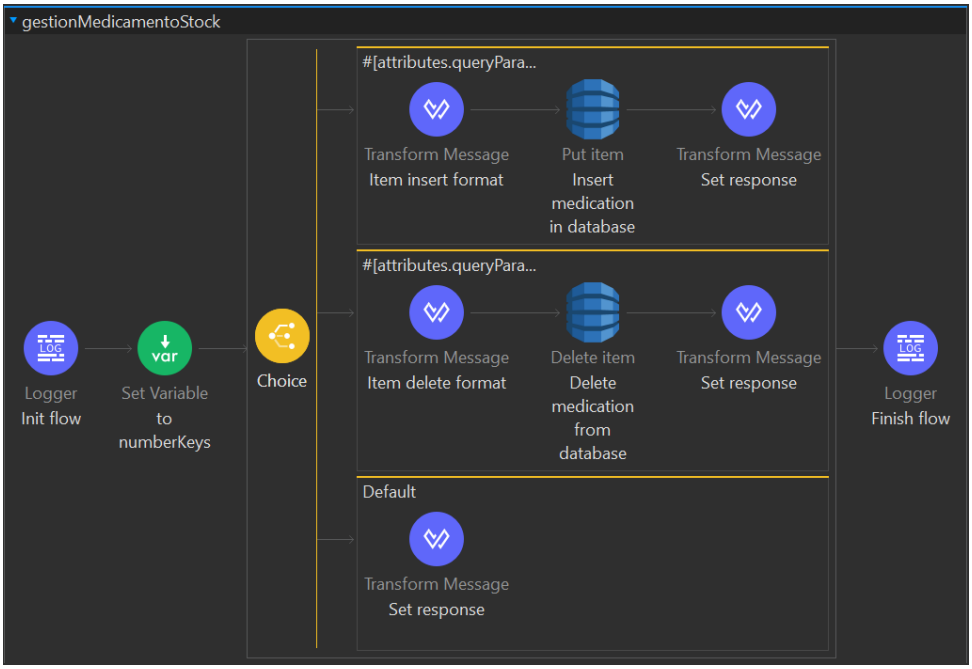

Figura 5.7: Subflujo del endpoint post:\medicamentos

Si las condiciones establecidas para la inserción o eliminación son satisfechas se procede con una tipificación explícita de los datos antes de ejecutar la acción con el fin de asegurar que la base de datos interprete correctamente los valores del cuerpo de la petición. Posteriormente se lleva a cabo la acción correspondiente, ya sea mediante la inserción utilizando el conector de Amazon DynamoDB con la operación Put Item, o mediante la eliminación con la operación Delete Item. Una vez completada la operación, se envía un mensaje al cliente confirmando que la acción se ha realizado exitosamente.

El segundo submódulo tiene como objetivo principal la recuperación de todos los medicamentos almacenados en la base de datos. Así como en el anterior submódulo la API de experiencia no resulta de particular interés, dado que su función se limita a propagar la solicitud sin llevar a cabo transformaciones o modificaciones adicionales. De manera similar, la API de procesos también se encarga de propagar la solicitud, realizando únicamente una transformación en la estructura de la respuesta detallada en el diseño de la solución.

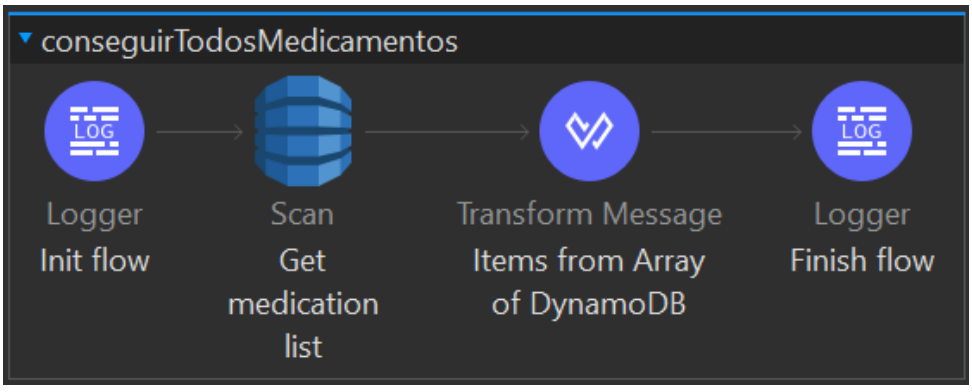

Figura 5.8: Subflujo del endpoint get:\medicamentos

El subflujo correspondiente al endpoint de la API de sistemas, representado en la Figura 5.8 se distingue por una implementación relativamente simple. Durante la implementación empleamos el conector Scan de Amazon DynamoDB para obtener una lista completa de los medicamentos almacenados en la base de datos. Sin embargo, es importante señalar que, junto con estos medicamentos, se recupera también un conjunto de datos adicionales que no son relevantes para la respuesta final. Es por ello que se lleva a cabo una transformación posterior de los datos, en la cual se filtra y selecciona únicamente la lista de medicamentos pertinente que se devolverá al cliente como respuesta.

En última instancia, tenemos el tercer submódulo cuya función principal es permitir el incremento simultáneo de la cantidad de uno o varios medicamentos. A diferencia de los submódulos anteriores, en este caso se detalla la implementación tanto de la capa de procesos como de la capa de sistemas para una comprensión integral del funcionamiento.

La Figura 5.9 ilustra el subflujo correspondiente al endpoint *put:\medicamentos* en la capa de procesos. Dado que el cuerpo de la solicitud está compuesto por una lista que puede incluir uno o varios medicamentos es necesario tratar el incremento de la cantidad de manera individualizada. Es por ello que en la capa de procesos hemos implementado un patrón de tipo *For Each*, que nos permite gestionar de manera separada cada objeto medicamento conformado por un identificador único y la cantidad a incrementar. Durante la ejecución del *For Each* inicialmente almacenamos el identificador del medicamento en una variable específica, que posteriormente utilizamos como parámetro URI en la solicitud a la capa de sistemas. Una vez procesados todos los medicamentos a través del *For Each* se procede a enviar una respuesta consolidada al cliente, indicando que la operación ha sido completada.

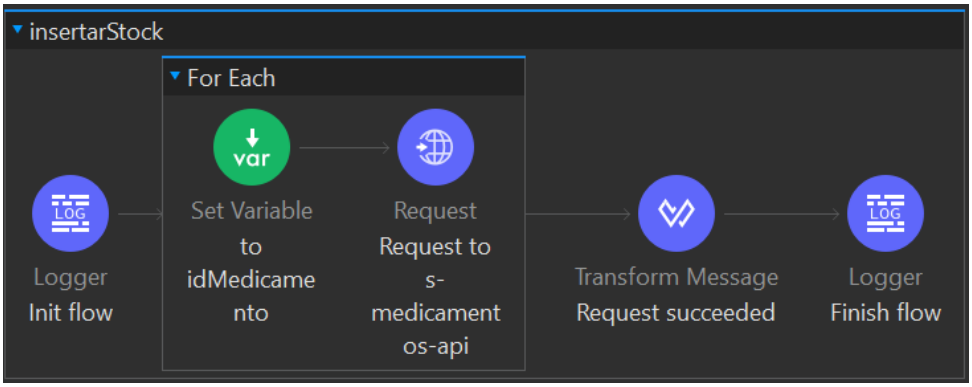

Figura 5.9: Subflujo del endpoint put:\medicamentos

Por otro lado, la Figura 5.10 detalla el subflujo del endpoint en la capa de sistemas, que se encarga del incremento de la cantidad de los medicamentos. Este subflujo crea una variable específica con el fin de tipificar explícitamente el valor de la cantidad asegurando su correcta interpretación por parte de la base de datos. Posteriormente, se procede a la actualización del valor del medicamento utilizando el conector Update Item de Amazon DynamoDB. Finalmente, se genera y envía una respuesta al cliente, confirmando la correcta ejecución de la operación de incremento de la cantidad de medicamentos.

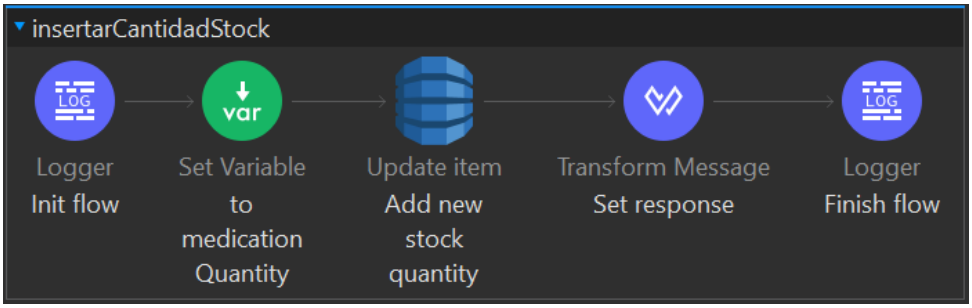

Figura 5.10: Subflujo del endpoint put:\medicamentos\(id)

5.2.3. Consulta de informes médicos

Durante la fase de diseño detallado de la operativa para la recuperación de una lista de informes de un paciente específico, observamos que la complejidad en las capas de experiencia y de proceso no era significativa. La capa de experiencia se limitaba a propagar la solicitud HTTP recibida, mientras que la capa de procesos, además de propagar dicha solicitud, realizaba una conversión de datos, tal como se describe en la Sección 4.3.2. Por lo tanto, nuestra atención se centra en la capa de sistemas, la cual presenta una complejidad considerablemente mayor.

El subflujo de *get:\informes\(id)* representado en la Figura 5.11, inicialmente almacena el valor del parámetro URI en una variable. Este valor se emplea para obtener una lista de nombres de informes que coinciden, parcialmente, con el ID proporcionado. La razón de esto radica en que, durante el proceso de almacenamiento, los informes se guardan en una estructura que también incluye la fecha de realización del informe. Aunque el parámetro URI no contiene la fecha utilizamos este conector para recuperar todos los nombres de informes que presenten una coincidencia parcial con dicho parámetro.

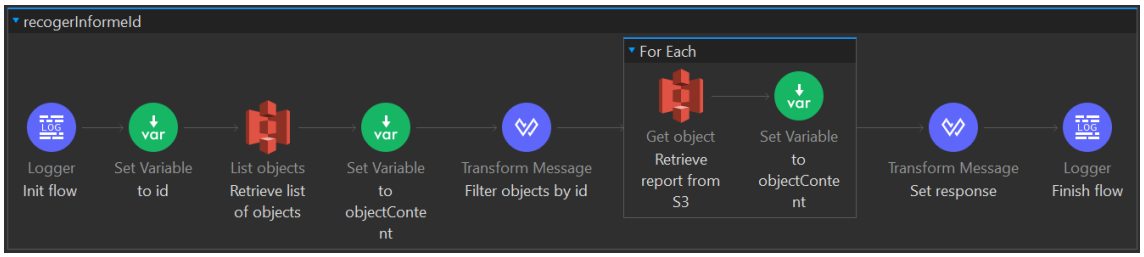

Figura 5.11: Subflujo del endpoint get:\informes\(id)

Posteriormente generamos una variable que contiene una lista vacía. Este paso es necesario porque, aunque disponemos de un array de nombres de informes no podemos utilizarlo directamente para obtener los informes completos, ya que debemos procesarlos de manera individual. Es por ello que empleamos un patrón de iteración *For Each* que recorre el arreglo elemento por elemento. Una limitación de este patrón es que no modifica la respuesta de la petición, lo que nos obliga a almacenar los informes procesados en una nueva variable.

Teniendo como objetivo resolver esta limitación creamos una variable antes de iniciar el bucle *For Each* y, en cada iteración, actualizamos esta variable añadiendo el informe correspondiente a la lista. Antes de proceder con la obtención del contenido de los informes mediante el conector de Amazon S3 (Get Object),

reorganizamos el array de nombres de informes para asegurar que el informe más reciente sea el primero en la lista.

Finalmente asignamos el valor de la variable que contiene la lista de informes procesados al resultado que se devolverá al cliente. Este enfoque nos permite optimizar la recuperación de datos y asegurar que los informes se presenten de manera eficiente y en el orden adecuado.

5.2.4. Control de informes médicos y medicamentos

La explicación de esta implementación se centra en la capa de procesos, así como en la implementación de los diferentes endpoints de las capas de sistemas. Esto se debe a que la capa de experiencia no resulta de especial interés en este contexto, ya que su función principal es la simple propagación de la petición.

La Figura 5.12 ilustra el subflujo correspondiente al endpoint de procesos, denominado *insertarInformeMedicamentos*. Este subflujo, según lo establecido en la fase de diseño, ejecuta dos procesos en paralelo utilizando el patrón *Scatter-Gather*. Este patrón permite dividir el flujo en dos ramas simultáneas que luego convergen para generar una respuesta unificada al cliente.

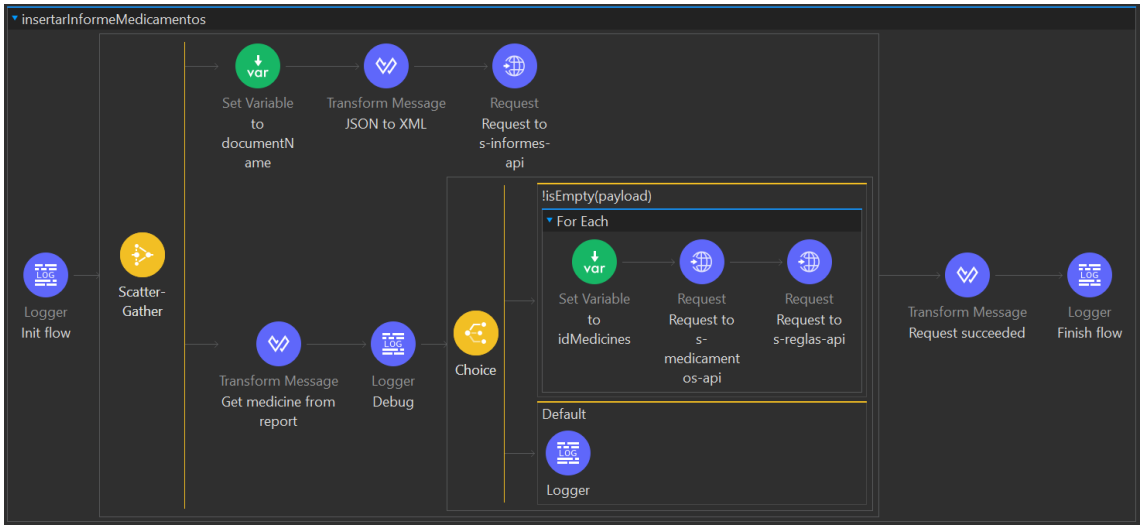

Figura 5.12: Subflujo del endpoint post:\informeGeneral

El flujo paralelo superior se encarga de crear una variable que unifica el nombre y el identificador del paciente. Seguidamente transforma el formato de datos del informe de JSON a XML, tras esta transformación se realiza una solicitud a la API de sistemas para almacenar el informe utilizando la variable previamente creada como parámetro URI en dicha petición.

Por otro lado, el flujo paralelo inferior se ocupa de extraer la lista de medicamentos recetados en el informe. Antes de proceder, se verifica si dicha lista contiene elementos, lo que permite evitar llamadas innecesarias en caso de que no existan medicamentos que procesar. En caso de que la lista contenga medicamentos, se utiliza el patrón *For Each* para enviarlos individualmente primero a la API *s-medicamentos-api* y luego a *s-reglas-api*.

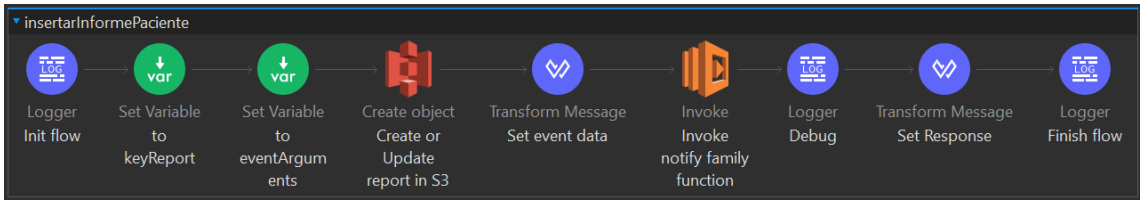

Figura 5.13: Subflujo del endpoint post:\informes\(id)

La Figura 5.13 representa el flujo correspondiente al proceso de almacenamiento de los informes recibidos en formato XML. Este proceso incluye dos pasos preliminares fundamentales antes del almacenamiento del informe. En primer lugar, se obtiene el parámetro URI al que se añade la marca temporal del informe, con el objetivo de evitar la sobreescritura de informes con nombres similares. En segundo lugar, se preparan los argumentos necesarios para notificar a los familiares. Una vez completados estos pasos previos el informe se almacena utilizando el conector Create Object de Amazon S3, y la carga útil se modifica para asignar los argumentos preparados. Posteriormente, se activa la función de Amazon Lambda, que se encarga de preparar el mensaje a enviar y activar el sistema de mensajería de Amazon SNS.

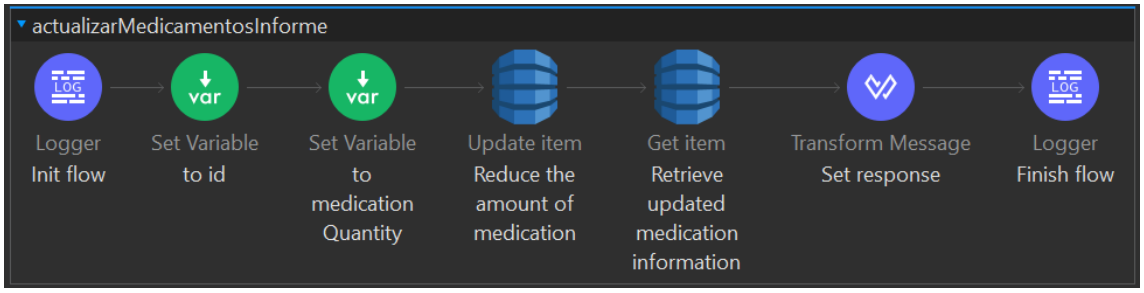

Figura 5.14: Subflujo del endpoint patch:\medicamentos\(id)

El flujo inferior se presenta en la Figura 5.14 y comienza con una llamada a la API *s-medicamentos-api*, cuya función es actualizar la cantidad de medicamentos restando la cantidad recetada en el informe. Inicialmente se realiza una tipificación explícita para que Amazon DynamoDB reconozca correctamente los valores del identificador y la cantidad. A continuación, se emplea el conector Update Item de Amazon DynamoDB para actualizar la cantidad de medicamentos. Este proceso implica especificar el nombre de la base de datos a la que se desea acceder e introducir el identificador del medicamento, que se encuentra como parámetro URI de la petición. Finalmente se resta la cantidad recetada a la cantidad actual mediante una expresión de actualización. El dato actualizado se recupera posteriormente para su uso por el sistema de reglas. La última transformación de datos asigna el valor y el nombre del medicamento actualizado a la respuesta que se devolverá a la petición inicial.

Por último, continuando con el flujo paralelo inferior descrito en la Figura 5.12, tras recibir la respuesta de la API *s-medicamentos-api*, se realiza una llamada a la API *s-reglas-api*, a través del endpoint *post:\sistemaReglas*. La Figura 5.15 muestra el subflujo correspondiente a este endpoint que tal como se describió durante la fase de diseño esta API utiliza un motor de reglas denominado Drools para determinar si es necesario realizar una notificación al gerente sobre la nece-

sidad de realizar un nuevo pedido de un fármaco específico. No obstante, debido a limitaciones temporales y de complejidad, no se ha implementado este sistema en totalidad. En su lugar, se ha optado por una simulación, que decide si realizar la notificación o no dependiendo de si la cantidad actualizada del medicamento es inferior a 100 unidades, como se puede observar con el primer patrón *Choice*. Si se cumple esta condición, se preparan los argumentos para la función de AWS Lambda, la cual personaliza el mensaje y activará el sistema de mensajería.

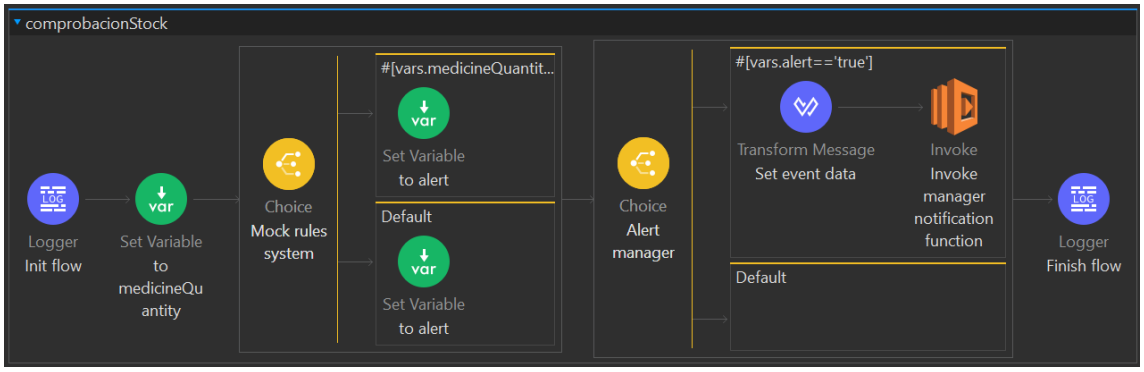

Figura 5.15: Subflujo del endpoint post:\sistemaReglas

CAPÍTULO 6
Pruebas

A lo largo del presente capítulo nos centraremos en la evaluación detallada de las operativas que hemos implementado en el capítulo anterior. El propósito fundamental de esta fase es comprobar que todas las funcionalidades del proyecto se desempeñen de manera adecuada y eficiente. Este proceso de evaluación, conocido comúnmente como fase de pruebas, reviste una importancia crucial, ya que nos brinda la oportunidad de identificar y corregir posibles errores antes de que el proyecto sea entregado a los usuarios finales.

Durante esta fase, llevaremos a cabo una serie de pruebas sistemáticas sobre las distintas funcionalidades del sistema. Estas pruebas estarán orientadas a verificar que cada funcionalidad opere conforme a las especificaciones previstas. El objetivo de estas pruebas es asegurar que el sistema, en su totalidad, satisfaga los requisitos especificados y funcione sin problemas en el entorno previsto. Este enfoque metódico nos permitirá detectar y abordar cualquier anomalía o deficiencia antes de la implementación final del proyecto.

6.1 Pruebas de integración con Postman

Postman es una plataforma diseñada para facilitar la creación, prueba y gestión de APIs. La principal función de Postman es permitir probar y gestionar colecciones de APIs, tanto para el Frontend como para el Backend de sus aplicaciones. La herramienta permite crear y ejecutar peticiones HTTP de manera gráfica, lo que simplifica la verificación de que los servicios web estén funcionando correctamente. Este proceso es crucial para identificar y corregir errores. [32]

Además, Postman ofrece la posibilidad de organizar funciones y módulos en carpetas dentro de colecciones, lo que facilita la gestión y el acceso a diferentes partes de los servicios web, permitiendo gestionar el ciclo de vida completo de una API, desde su conceptualización hasta su mantenimiento y documentación. Una característica destacada de Postman es su capacidad para trabajar con entornos colaborativos. Este permite a los equipos compartir información y coordinar esfuerzos de manera eficiente. Asimismo, la plataforma facilita la definición y gestión de variables de entorno, lo que resulta útil cuando se trabaja con múltiples entornos o proyectos simultáneamente. [32]

A continuación, realizaremos un proceso de pruebas utilizando Postman, con el propósito de validar y optimizar las operativas desarrolladas. Este proceso es fundamental para garantizar no solo el correcto funcionamiento de las funcionalidades, sino también para asegurar que cumplen con los requisitos de rendimiento establecidos, especialmente en términos de eficiencia y velocidad.

Inicialmente se revisará cada solicitud generada, evaluando tanto la respuesta obtenida como la naturaleza de la ejecución. Además de monitorear los tiempos de respuesta, se verificará la efectividad de las operaciones, asegurándonos de que las acciones realizadas se reflejan correctamente en los sistsmeas correspondientes.

Un aspecto clave del proceso de pruebas es la utilización correcta del host que vamos a utilizar, dado que el desarrollo se realiza en un entorno local, el host en todas las peticiones, será siempre *localhost*.

6.1.1. Monitorización de los parámetros de bienestar

La primera operativa permite la monitorización de los parámetros de bienestar de un paciente. Observando la Figura 6.1 vemos un ejemplo de petición para probar esta funcionalidad. La petición que vemos en la imagen se realiza directamente a la API de la capa de experiencia y utiliza el método *POST*. El cuerpo del mensaje incluye el nombre del paciente y los diferentes parámetros mencionados durante el diseño. La petición que hemos realizado cuenta con un valor anormal para ritmo cardíaco con el objetivo de que esto desencadene la activación del sistema de notificaciones, lo que nos permitirá conocer si este funciona correctamente.

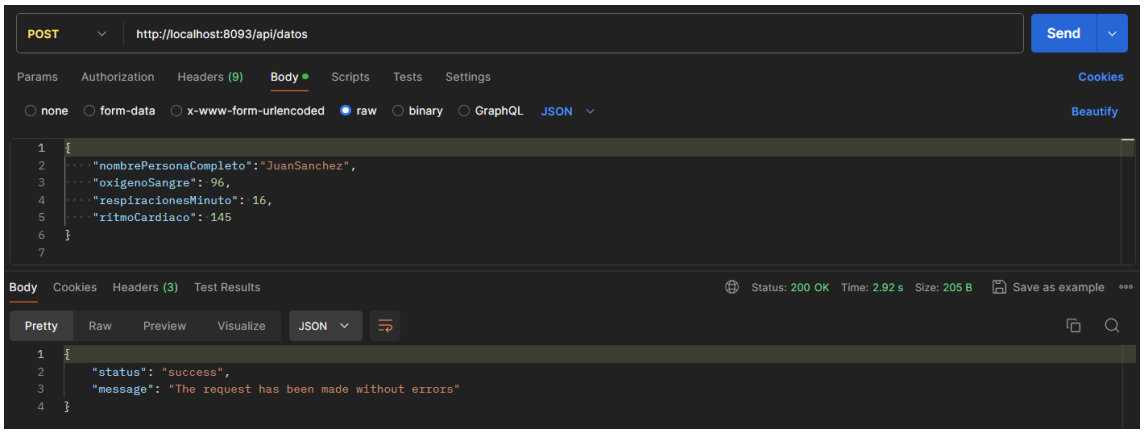

Figura 6.1: Petición y respuesta del análisis de parámetros de bienestar

Una vez enviada la petición, como se observa en la Figura 6.1, se ha obtenido una respuesta en 2.92 segundos con un código 200 OK. Este código de respuesta confirma que la operación fue exitosa. Este éxito también se refleja en el contenido de la respuesta, que indica que la operación se realizó correctamente.

Adicionalmente, al detectar una anomalía en el paciente y tras modificar el valor del ritmo cardíaco por uno anómalo acorde a personas de la tercera edad, se generó una notificación a través de Amazon SNS. Esto se puede observar en la

Figura 6.2, donde se ha recibido un correo electrónico de un remitente identificado claramente como Amazon SNS. Este correo detalla el nombre del paciente y la naturaleza de la anomalía detectada, proporcionando información crítica que, en un entorno real, permitiría a la enfermera responsable del paciente actuar de manera rápida y efectiva.

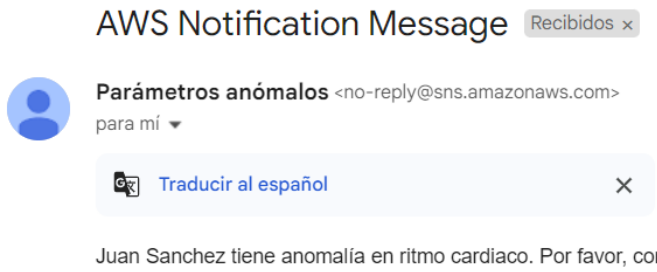

Figura 6.2: Notificación recibida como respuesta al análisis del paciente

6.1.2. Gestión integral de medicamentos

El siguiente paso en nuestro proceso de pruebas consiste en evaluar la operativa destinada a la gestión integral de la base de datos de las residencias, la cual se divide en tres submódulos. Durante esta fase inicial nos enfocamos en el primer submódulo, que permite la inserción y eliminación de medicamentos en Amazon DynamoDB.

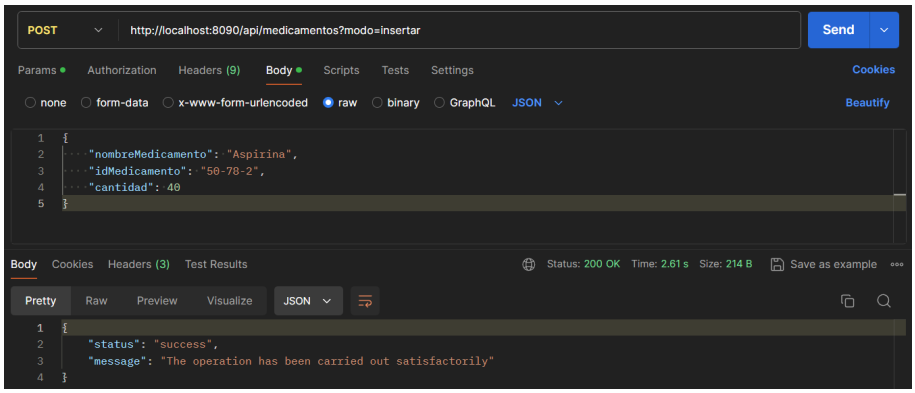

Figura 6.3: Petición y respuesta de la inserción de un nuevo medicamento

La Figura 6.3 muestra la solicitud realizada a la capa de experiencia con la finalidad de insertar un nuevo medicamento. La Figura permite ver que el parámetro de consulta se ajusta en modo insertar, introduciéndose en la solicitud precedido de un ? para indicar que se trata de un parámetro de consulta. El cuerpo de la solicitud incluye tres elementos esenciales para asegurar que la inserción se realice de manera exitosa. Luego de enviar la petición, se recibe una respuesta con un código 200 OK, con un tiempo de espera de 2.61 segundos, indicando que la operación se completó satisfactoriamente. De igual manera, en caso de haber ocurrido algún problema, habríamos recibido un código 404, indicando una petición mal formulada, o un código 500, señalando un error en el servidor.

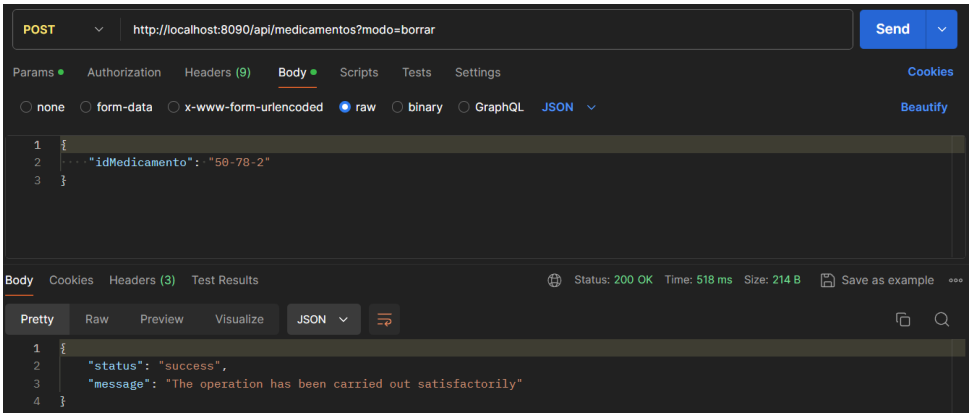

Figura 6.4: Petición y respuesta de la eliminación de un medicamento

Por otro lado, la Figura 6.4 presenta el proceso de eliminación de un medicamento. Este procedimiento es esencialmente el mismo al de la inserción, ya que tanto el puerto como el endpoint son los mismos. La diferencia radica en el valor del parámetro de consulta y en el contenido del cuerpo de la petición. En este caso, la operación de eliminación requiere únicamente el elemento idMedicamento. El tiempo de respuesta para esta operación fue de 516 ms, significativamente menor en comparación con la inserción, lo que sugiere que el proceso de eliminación en Amazon DynamoDB es más rápido.

El segundo submódulo está diseñado para recuperar la lista de los diferentes medicamentos almacenados en DynamoDB. La Figura 6.5 muestra cómo se realiza la petición *GET* pertinente a la misma API utilizada por el submódulo anterior. La respuesta de esta solicitud es un array que contiene los distintos medicamentos existentes en la base de datos, presentados en un formato JSON, que facilita la visualización y manipulación de los datos recuperados. El tiempo de respuesta registrado para esta operación es de 530 ms, demostrando la eficiencia del proceso en términos de velocidad.

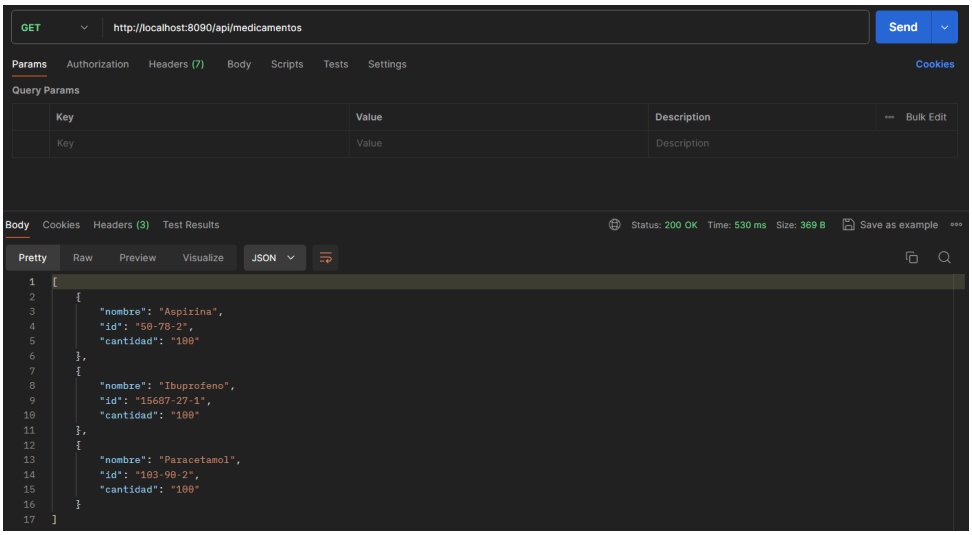

Figura 6.5: Petición y respuesta de la obtención de medicamentos

El tercer submódulo del sistema nos permite actualizar las cantidades disponibles de uno o varios medicamentos. Esta funcionalidad se ilustra en la Figura

6.6, donde se observa que el cuerpo del mensaje está compuesto por un array que contiene diferentes objetos, cada uno de estos objetos incluye dos elementos clave: el identificador del medicamento y la cantidad que se desea incrementar. Una vez enviada la solicitud para actualizar las cantidades el sistema responde con un código de estado 200 OK, indicando que la operación se realizó con éxito. El tiempo de respuesta fue de 770 ms, y el cuerpo del mensaje de respuesta confirma que el incremento solicitado en las cantidades se llevó a cabo correctamente.

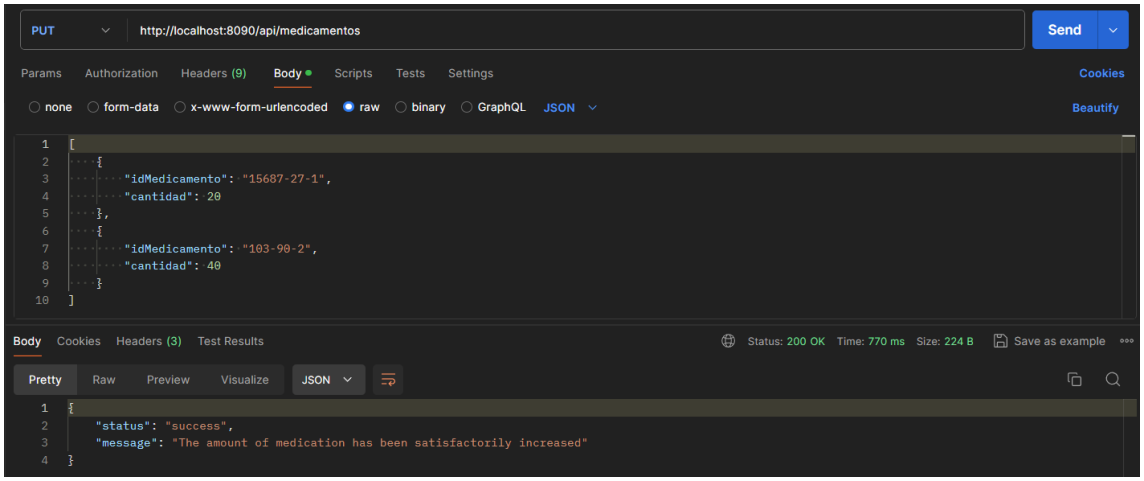

Figura 6.6: Petición y respuesta de la actualización de la cantidad de medicamentos

6.1.3. Consulta de informes médicos

Seguidamente realizamos las pruebas de la funcionalidad que permite recuperar los informes de los medicamentos asociados a un paciente. Esto se logra mediante una solicitud realizada a la API denominada *e-familiares-api*, como se muestra en la Figura 6.7. La solicitud incluye un parámetro URI específico para recuperar los distintos informes.

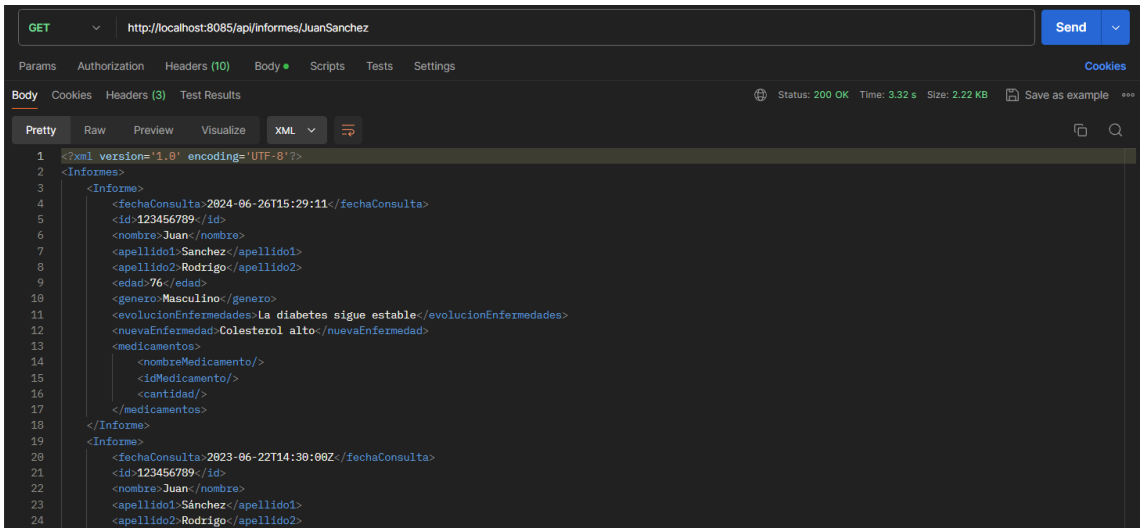

Figura 6.7: Petición y respuesta de la obtención de los informes médicos

La petición resulta exitosa, evidenciado por el código de respuesta 200 OK, con un tiempo de respuesta de 3.32 segundos. El cuerpo de la respuesta contie-

ne los informes en formato XML, lo que facilita la lectura y procesamiento de la información. Este formato permite una visualización clara, destacando que los informes están ordenados de manera descendente basándose en el campo fechaConsulta, lo que facilita la rápida identificación de los registros más recientes.

6.1.4. Control de informes médicos y medicamentos

Finalmente, abordamos la operativa destinada a la creación de un nuevo informe, diseñada para automatizar varios procesos clave en la gestión de medicamentos y la comunicación con los familiares de los residentes. Este proceso comienza con una solicitud *POST*, a la API de experiencia *e-informes-api*, como se ilustra en la Figura 6.8. El cuerpo del mensaje de esta solicitud incluye la estructura del informe a insertar. Una vez enviada la petición, se recibió un código de respuesta 200 OK con un tiempo de respuesta de 7.55 segundos, indicando que la solicitud fue procesada exitosamente. Además, se genera una notificación confirmando la creación del nuevo informe.

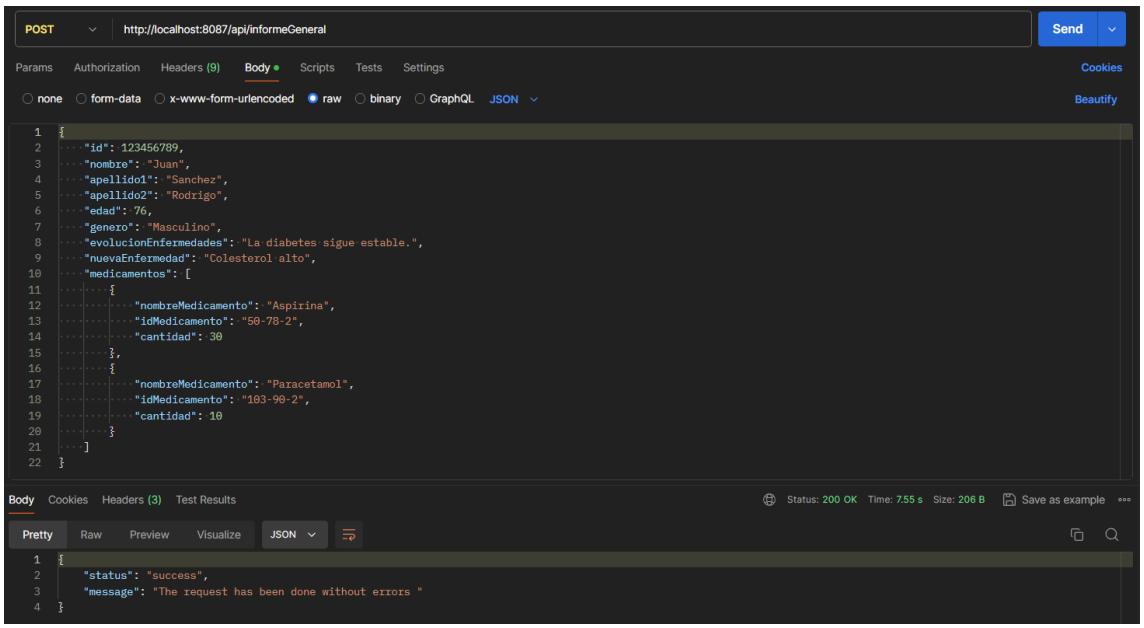

Figura 6.8: Petición y respuesta de la creación de un nuevo informe

Esta notificación, representada en la Figura 6.9, informa a los familiares del residente sobre la creación del nuevo informe, invitándolos a acceder al portal web para obtener detalles sobre el estado de salud del residente y los medicamentos recetados.

Figura 6.9: Notificación recibida como respuesta a la inserción del informe

Por otro lado, previo al envío de la solicitud, se ajustó la cantidad de medicamentos en el sistema para activar otra notificación, esta vez dirigida al gerente. Como se muestra en la Figura 6.10 tras la prescripción de 30 aspirinas al paciente, la cantidad restante en el inventario se redujo a un nivel críticamente bajo, lo que podría ocasionar una escasez. La notificación alerta al gerente sobre esta situación, permitiéndole tomar las medidas necesarias para evitar la falta de suministros en la residencia.

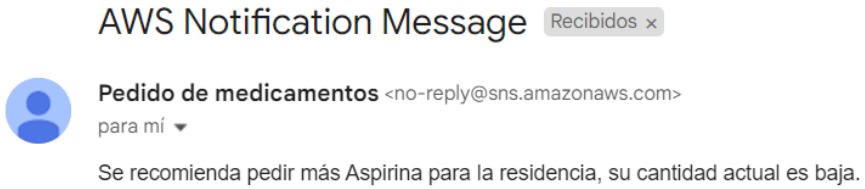

Figura 6.10: Notificación recibida como respuesta a la baja cantidad de un medicamento

La recepción de estas notificaciones nos brinda la posibilidad de verificar que los procesos se han llevado a cabo de manera exitosa. Estas notificaciones son indicativas tanto del correcto almacenamiento del informe como de la actualización efectiva de la cantidad de medicamentos, además de la implementación adecuada del sistema de reglas. Podemos afirmar esta conclusión debido a que, en caso de haberse producido algún error en los otros sistemas involucrados, estas notificaciones no se habrían generado.

CAPÍTULO 7
Conclusiones

El presente trabajo expone el desarrollo de un middleware fundamentado en una arquitectura API-led, diseñado para integrar un total de siete sistemas en el contexto de las residencias geriátricas. El propósito primordial de este proyecto era optimizar las tareas diarias en las residencias mediante la integración de estos sistemas con el fin de reducir la carga de trabajo del personal y abordar los problemas derivados del envejecimiento de la población.

En primer lugar, se logró implementar con éxito el middleware propuesto cumpliendo con los diversos objetivos específicos establecidos al inicio del proyecto. La solución se desarrolló diseñando un total de 12 APIs REST organizadas en capas, las cuales fueron implementadas y probadas en conjunto mediante funcionalidades. Además, se utilizó la metodología ágil SCRUM y el control de versiones Gitflow con el objetivo de seguir una planificación y gestión de las diferentes tareas de una forma eficiente.

Durante el desarrollo se presentaron varias problemáticas, siendo una de las más desafiantes la configuración del sistema de machine learning Amazon SageMaker, dada la falta de conocimiento previo sobre esta tecnología. No obstante, este y otros desafíos fueron superados gracias a la investigación exhaustiva y al control meticuloso de los distintos pasos realizados en la implementación. Otro desafío relevante fueron las transformaciones de datos, que se resolvieron eficazmente mediante el uso de Dataweave Interactive, herramienta que permitió observar las transformaciones en tiempo real.

El conocimiento adquirido a través de los estudios cursados durante la carrera resultó fundamental para el desarrollo de la solución. Estos estudios proporcionaron una base sólida sobre transformaciones de datos, arquitecturas de sistemas y tipos de peticiones, así como su comunicación. En particular, la asignatura de Integración de Aplicaciones, cursada en cuarto año, desempeñó un papel crucial, despertando un interés en esta área específica de la informática que inicialmente era desconocida y que condujo a la realización de este trabajo y a la realización de prácticas empresariales en este ámbito.

En conclusión, este trabajo resalta la importancia de la integración de aplicaciones en las organizaciones, como ejemplifica el caso de las residencias geriátricas. Gracias a este middleware basado en una arquitectura API-led, se ha conseguido reducir la carga de trabajo del personal, permitiéndoles centrarse en la salud y bienestar de los residentes. Este avance representa un paso significati-

vo hacia la optimización de las tareas en el ámbito de las residencias de ancianos, cumpliendo con la premisa inicial del proyecto así como con los diferentes objetivos planteados.

7.1 Trabajos futuros

Durante el transcurso de este proyecto hemos logrado demostrar la funcionalidad y eficacia de la solución propuesta en el ámbito de la gestión de residencias. A través de un enfoque sistemático y riguroso hemos diseñado una arquitectura escalable y adaptable que no solo cumple con los objetivos establecidos, sino que también ha creado una base sólida para el desarrollo de futuras mejoras que optimicen aún más la propuesta.

La arquitectura que hemos implementado permite continuar perfeccionando y ampliando las funcionalidades originalmente planteadas. Este enfoque, además de garantizar la adaptabilidad a las necesidades actuales, también abre nuevas posibilidades para incorporar avances tecnológicos y metodológicos.

A continuación, detallamos algunas de las posibles mejoras que podrían incrementar el valor y la efectividad del proyecto en su conjunto:

- Configuración de Drools: Una de las mejoras que consideramos más relevante para el proyecto es la implementación adecuada de la configuración del sistema de reglas Drools. Este sistema no ha podido ser configurado de manera óptima durante el desarrollo del proyecto debido a la complejidad temporal asociada con su implementación. Sin embargo, creemos firmemente que una configuración adecuada de Drools mejoraría exponencialmente la eficiencia de la solución propuesta ya que permitiría proporcionar respuestas más precisas y efectivas en el proceso de gestión de pedidos de medicamentos.

- Establecimiento de un modelo canónico: Durante el desarrollo del proyecto hemos utilizado una amplia variedad de datos provenientes de diversas fuentes, como información médica o datos sobre medicamentos. Una forma de integrar eficazmente estos datos y mejorar la interoperabilidad entre sistemas proponemos la implementación de un modelo canónico de datos para registros electrónicos de salud (EHR). Un modelo canónico de datos proporciona una estructura unificada que facilita el intercambio de información entre diferentes sistemas médicos. Estandarizar el formato y la estructura de los datos asegura que la información pueda ser comprendida y procesada de manera coherente sin importar su origen. Esto simplifica la comunicación entre sistemas heterogéneos y reduce la complejidad de traducir y entender datos provenientes de diferentes fuentes.

- Seguridad y protección de datos: Los datos médicos son altamente sensibles, por lo que es esencial implementar medidas de seguridad para prevenir su fuga y uso indebido. Algunas medidas recomendadas incluyen el cifrado de datos, una autenticación robusta y el control de acceso, monitoreo constante, actualización regular de sistemas y establecimiento de polí-

ticas claras. Estas acciones pueden ayudar a proteger la privacidad de los pacientes y garantizar el cumplimiento de normativas legales y éticas.

- Establecimiento de una librería común de errores: La creación de una librería común de errores permitiría a los usuarios tener una retroalimentación clara y consistente cuando se produce un error, mejorando la experiencia de uso. Esto no solo facilita el diagnóstico de problemas, sino que también mejora la capacidad de los usuarios para tomar medidas correctivas, reduciendo frustración y mejorando la experiencia en general.
- Inclusión de nuevas operativas: Se podría considerar el desarrollo de nuevas soluciones que optimicen diversos procesos operativos. Algunos de los procesos que podrían beneficiarse son el registro de nuevos pacientes y la administración de las habitaciones disponibles en las residencias.

Bibliografía

[1] ¿Qué es la integración de aplicaciones? IBM. IBM - United States https://www.ibm.com/es-es/topics/application-integration.

[2] GORKHALI, Anjee; XU, Li Da. *Enterprise application integration in industrial integration: a literature review*. Journal of Industrial Integration and Management, 2016, vol. 1, no 04, p. 1650014.

[3] LINTHICUM, David S. *Enterprise application integration*. Addison-Wesley Professional, 2000.

[4] THEMISTOCLEOUS, Marinos. *Justifying the decisions for EAI implementations: a validated proposition of influential factors*. Journal of Enterprise Information Management, 2004, vol. 17, no 2, p. 85-104.

[5] SOOMRO, Tariq Rahim; AWAN, Abrar Hasnain. *Challenges and future of enterprise application integration*. International Journal of Computer Applications, 2012, vol. 42, no 7, p. 42-45.

[6] GALKIN, Ossi, et al. Enterprise Application Integration Architecture Recovery. *Tesis de Maestría*, 2019

[7] Entender la integración empresarial de aplicaciones: las ventajas del ESB para la EAI MuleSoft. MuleSoft. https://www.mulesoft.com/es/resources/esb/enterprise-application-integration-eai-and-esb.

[8] ROSA-SEQUEIRA, Fernando; BASTO-FERNANDES, Vıtor; FRANTZ, Rafael Z. *Enterprise application integration: Approaches and platforms to design and implement solutions in the cloud*. Advances in Engineering Research, 2018.

[9] The Evolution of Integration: From Point-to-Point to API-Centric. LinkedIn. https://www.linkedin.com/pulse/evolution-integration-from-point-to-point-api-centric-sivasena-reddy-csi3f.

[10] Evolution of Application Integration and API First Approach. Medium. https://medium.com/@chandramanohar/evolution-of-application-integration-and-api-first-approach-15b35f4f305f.

[11] PATNI, Sanjay. *Pro RESTful APIs*. Apress, 2017.

[12] SHARMA, Kirti, et al. *API LED Connectivity Approach*. En 2022 International Conference on Cyber Resilience (ICCR). IEEE, 2022. p. 1-3.

[13] Demography of Europe - An ageing population. (s.f.). Instituto Nacional de Estadística (INE). https://www.ine.es/prodyser/demografia_UE/bloc-1c.html?lang=es.

[14] Envejecimiento y salud. (s.f.). World Health Organization (WHO). https://www.who.int/es/news-room/fact-sheets/detail/ageing-and-health.

[15] El número de personas mayores de 65 años en el mundo ya supera al de niños menores de cinco: el envejecimiento en nueve gráficos. elEconomista.es. (2019, 1 de octubre). https://www.eleconomista.es/economia/noticias/10114752/10/19/El-numero-de-personas-mayores-de-65-anos-en-el-mundo-ya-supera-al-de-ninos-menores-de-cinco.html.

[16] Centros de mayores ResiPlus. (s.f.). ResiPlus. https://addinformatica.com/software-gestion-centros-mayores/.

[17] Funciones y módulos de nuestro software GdR. (s.f.). Gestión de Residencias.. https://www.gestionderesidencias.es/funciones/.

[18] SUDA, Brian. *Soap web services*. Retrieved June, 2003, vol. 29, p. 2010.

[19] INDRASIRI, Kasun; SIRIWARDENA, Prabath. *Microservices for the Enterprise*. Apress, Berkeley, 2018.

[20] Qué es la Metodología SCRUM y Características grupoaspasia.com. (s.f.). grupoaspasia.com. https://grupoaspasia.com/es/glosario/metodologia-scrum/.

[21] ¿En qué consiste Scrum? AWS. (s.f.). Amazon Web Services, Inc. https://aws.amazon.com/es/what-is/scrum/.

[22] Una breve introducción a los tableros de Jira. Atlassian. (s.f.). Atlassian. https://www.atlassian.com/es/software/jira/guides/boards/overview#what-is-a-jira-board.

[23] Gitflow Workflow. Atlassian Git Tutorial. (s.f.). Atlassian. https://www.atlassian.com/git/tutorials/comparing-workflows/gitflow-workflow#:~:text=What%20is%20Gitflow?,lived%20branches%20and%20larger%20commits.

[24] What is MuleSoft?. (s.f.). Salesforce. https://www.salesforce.com/mulesoft/what-is-mulesoft/#:~:text=So%20what%20is%20MuleSoft?,easy-to-use%20platform..

[25] ¿Qué es AWS?. Computación en la nube con Amazon Web Services. (s.f.). Amazon Web Services, Inc. https://aws.amazon.com/es/what-is-aws/.

[26] What is Amazon S3? - Amazon Simple Storage Service. Amazon Web Services, Inc. (s.f.). Amazon Web Services, Inc. https://docs.aws.amazon.com/AmazonS3/latest/userguide/Welcome.html.

[27] What is Amazon DynamoDB? - Amazon DynamoDB. Amazon Web Services, Inc. (s.f.). Amazon Web Services, Inc. https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/Introduction.html.

[28] What is Amazon SageMaker? - Amazon SageMaker. Amazon Web Services, Inc. (s.f.). Amazon Web Services, Inc. https://docs.aws.amazon.com/sagemaker/latest/dg/whatis.html.

[29] What is AWS Lambda? - AWS Lambda. Amazon Web Services, Inc. (s.f.). Amazon Web Services, Inc. https://docs.aws.amazon.com/lambda/latest/dg/welcome.html.

[30] What is Amazon SNS? - Amazon Simple Notification Service. Amazon Web Services, Inc. (s.f.). Amazon Web Services, Inc. https://docs.aws.amazon.com/sns/latest/dg/welcome.html.

[31] Drools Rules Engine. Medium. https://medium.com/@dixitsatish34/drools-rules-engine-bffc7ef9077c.

[32] ¿Qué es Postman? Características y Ventajas. Assembler Institute. https://assemblerinstitute.com/blog/que-es-postman/.

APÉNDICE A
Objetivos de desarrollo sostenible

A.1 Grado de relación del trabajo con los Objetivos de Desarrollo Sostenible (ODS)

Grado de relación del trabajo con los Objetivos de Desarrollo Sostenible (ODS).

Objetivos de Desarrollo Sostenible	**Alto**	**Medio**	**Bajo**	**No procede**
ODS 1. **Fin de la pobreza.**				X
ODS 2. **Hambre cero.**				X
ODS 3. **Salud y bienestar.**	X			
ODS 4. **Educación de calidad.**				X
ODS 5. **Igualdad de género.**				X
ODS 6. **Agua limpia y saneamiento.**				X
ODS 7. **Energía asequible y no contaminante.**				X
ODS 8. **Trabajo decente y crecimiento económico.**	X			
ODS 9. **Industria, innovación e infraestructuras.**	X			
ODS 10. **Reducción de las desigualdades.**				X
ODS 11. **Ciudades y comunidades sostenibles.**				X
ODS 12. **Producción y consumo responsables.**				X
ODS 13. **Acción por el clima.**				X
ODS 14. **Vida submarina.**				X
ODS 15. **Vida de ecosistemas terrestres.**				X
ODS 16. **Paz, justicia e instituciones sólidas.**				X
ODS 17. **Alianzas para lograr objetivos.**				X

A.2 Reflexión sobre la relación del TFG/TFM con los ODS y con el/los ODS más relacionados.

Nuestro proyecto se alinea de manera clara y directa con tres Objetivos de Desarrollo Sostenible (ODS): *Salud y Bienestar* (ODS 3), *Trabajo Decente y Crecimiento Económico* (ODS 8), e *Industria, Innovación e Infraestructura* (ODS 9). Esta alineación se vuelve evidente al examinar el propósito y los beneficios que nuestro trabajo ofrece en el contexto de la atención geriátrica.

En primer lugar, el ODS 3 busca garantizar una vida sana y promover el bienestar para todos en todas las edades, este objetivo se ve impactado significativamente por la creación de este middleware. Nuestro proyecto se centra en mejorar la calidad de atención para los adultos de la tercera edad en residencias geriátricas, un grupo particularmente vulnerable que requiere en ciertos casos cuidados especializados y personalizados. La implementación de nuestro middleware permite automatizar y optimizar numerosas tareas manuales y repetitivas realizadas por el personal geriátrico, lo que reduce su carga administrativa. Esto permite al personal dedicar más tiempo y atención a las interacciones directas con los residentes, lo cual es esencial para su bienestar tanto físico como emocional.

El segundo ODS relevante es el ODS 9, centrando en promover la construcción de infraestructuras resilientes, la industrialización inclusiva y sostenible, y el fomento de la innovación. El uso de la arquitectura API-led en el desarrollo de este middleware representa un claro ejemplo de cómo la tecnología puede impulsar la eficiencia y la innovación en las organizaciones. Esta arquitectura facilita la integración ágil y eficiente de diversos sistemas dentro de la infraestructura tecnológica de la organización, permitiendo que esta evolucione a lo largo del tiempo. Las APIs desarrolladas no solo posibilitan una integración inmediata y eficiente, sino que también establecen una base sólida para futuras innovaciones. Este aspecto es crucial en un entorno donde la tecnología avanza rápidamente y la capacidad de adaptarse a nuevas tecnologías es fundamental.

El ODS 8 que busca promover el crecimiento económico sostenido, inclusivo y sostenible, así como el empleo pleno, productivo y decente para todos, también tiene relación con nuestro proyecto. Aunque esta conexión podría parecer menos directa en comparación con los ODS explicados antes, es igualmente significativa. La implementación de esta solución proporciona al personal geriátrico una herramienta que alivia la carga de trabajo, especialmente en lo que respecto a tareas repetitivas y administrativas. Gracias a reducir el tiempo destinado a estas tareas los trabajadores pueden enfocarse más en actividades como el cuidado directo de los residentes.

Además, la mejora en la eficiencia y en la organización del trabajo contribuye a un entorno laboral más productivo y seguro. Esto provoca que el personal geriátrico sienta menos presión y pueda desempeñar sus funciones de manera más efectiva, lo que fomenta un ambiente de trabajo más positivo y colaborativo. Asimismo, al mejorar la calidad de los servicios ofrecidos por las residencias geriátricas mediante el uso de la tecnología genera un impacto económico positivo al atraer a más posibles clientes. Sin lugar a duda residencias más eficientes y

con personal más capacitado y motivado resultan más competitivas, ofreciendo mejores servicios respecto a las demás.