UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería Informática

# Automatización de la Cadena de Suministro en el Sector Alimenticio Mediante Integración de Sistemas

Trabajo Fin de Grado

Grado en Ingeniería Informática

AUTOR/A: El Farh el Farh, Nas Reddine

Tutor/a: Fons Cors, Joan Josep

CURSO ACADÉMICO: 2023/2024

# Resum

En l'era digital, les empreses del sector alimentari s'enfronten a la necessitat de millorar l'eficiència i automatització de les seues cadenes de subministrament per respondre a les expectatives d'entregues ràpides i mantenir la frescor dels seus productes. Aquest repte implica la sincronització de la demanda del consumidor amb la producció i el subministrament d'aliments frescos, minimitzant el desaprofitament i optimitzant els recursos. Per abordar això, es proposa desenvolupar una solució integrada que enllace directament les compres dels clients amb la gestió d'inventari, utilitzant anàlisi predictiu per anticipar la demanda i millorar la planificació dels recursos humans i la logística. El projecte adoptarà un enfocament modular, utilitzant tecnologies avançades per a la recollida de dades en temps real, gestió d'inventaris, logística automatitzada i monitoratge ambiental mitjançant sensors. Es crearà un prototip per demostrar la viabilitat i eficàcia de la solució, permetent una gestió més eficient de la cadena de subministrament i millorant la satisfacció del client.

**Paraules clau:** Integració d'Aplicacions, Arquitectures dirigides per APIs, API Led, Cadena de Subministraments

# Resumen

En la era digital, las empresas del sector alimenticio se enfrentan a la necesidad de mejorar la eficiencia y automatización de sus cadenas de suministro para responder a las expectativas de entregas rápidas y mantener la frescura de sus productos. Este desafío implica la sincronización de la demanda del consumidor con la producción y el suministro de alimentos frescos, minimizando el desperdicio y optimizando los recursos. Para abordar esto, se propone desarrollar una solución integrada que vincule directamente las compras de los clientes con la gestión de inventario, utilizando análisis predictivo para anticipar la demanda y mejorar la planificación de los recursos humanos y la logística. El proyecto adoptará un enfoque modular, utilizando tecnologías avanzadas para la recolección de datos en tiempo real, gestión de inventarios, logística automatizada y monitoreo ambiental mediante sensores. Se creará un prototipo para demostrar la viabilidad y eficacia de la solución, permitiendo una gestión más eficiente de la cadena de suministro y mejorando la satisfacción del cliente.

**Palabras clave:** Integración de Aplicaciones; Arquitecturas dirigidas por APIs; API Led; Cadena de Suministros

# Abstract

In the digital era, companies in the food sector face the need to enhance the efficiency and automation of their supply chains to meet the expectations of fast deliveries and maintain the freshness of their products. This challenge involves synchronizing consumer demand with the production and supply of fresh foods, minimizing waste, and optimizing resources. To address this, we propose developing an integrated solution that directly links customer purchases with inventory management, using predictive analytics to anticipate demand and improve human resource and logistics planning. The project will adopt a modular approach, utilizing advanced technologies for real-time data collection, inventory management, automated logistics, and environmental monitoring through sensors. A prototype will be created to demonstrate the solution's feasibility and effectiveness, enabling more efficient supply chain management and improving customer satisfaction

**Key words:** Application Integration, API-driven Architectures, API Led, Supply Chain

# Índice general

# Índice de figuras

# Índice de tablas

# CAPÍTULO 1
# Introducción

En el contexto actual de la era digital, el sector alimenticio se enfrenta al desafío crucial de optimizar sus cadenas de suministro para satisfacer las demandas de entregas rápidas y mantener la frescura de sus productos. La eficiencia y automatización de estos procesos son fundamentales no solo para responder a las expectativas de los consumidores, sino también para minimizar el desperdicio y optimizar el uso de los recursos disponibles. Este Trabajo de Fin de Grado propone desarrollar una solución integrada que aborde estos desafíos mediante la sincronización eficaz de la demanda del consumidor con la producción y el suministro de alimentos frescos, utilizando tecnologías avanzadas y automatización.

El proyecto contempla la automatización de procesos clave, como la creación de clientes y la generación de pedidos, lo que simplificará la operativa diaria y garantizará una mayor precisión y eficiencia en la gestión de la cadena de suministro. Aunque el análisis de datos avanzados no ha sido implementado en esta etapa, se prevé su integración en fases futuras del proyecto para optimizar aún más la planificación y ejecución logística.

A través del diseño y desarrollo de un prototipo modular, el proyecto demostrará cómo la aplicación de técnicas de recolección de datos en tiempo real y sistemas automatizados de gestión de inventarios pueden transformar la cadena de suministro alimentaria, adaptándose a las fluctuaciones del mercado y las expectativas del consumidor moderno.

## 1.1 Motivación

La elección de este tema surge de una profunda convergencia entre mi experiencia personal en la industria alimentaria y mi sólida formación académica en ingeniería informática. Durante varios años, he tenido la oportunidad de trabajar en diversas facetas del sector alimentario, observando de primera mano los desafíos y complejidades que enfrentan las cadenas de suministro. Estos desafíos no solo incluyen la necesidad de mantener la frescura y calidad de los productos a lo largo de toda la cadena logística, sino también la presión constante de cumplir con las demandas de entregas rápidas y eficientes en un mercado cada vez más competitivo.

En mi experiencia laboral, he sido testigo de cómo las ineficiencias en la gestión de inventarios y la falta de sincronización en las operaciones pueden llevar a pérdidas significativas, tanto en términos de productos perecederos como de oportunidades de negocio. Estos problemas, que se agravan por la creciente demanda de transparencia y sostenibilidad por parte de los consumidores, despertaron en mí un interés genuino por explorar soluciones tecnológicas que pudieran transformar estos procesos.

La implementación de tecnologías avanzadas, como sistemas automatizados de gestión de inventarios, inteligencia artificial y análisis de datos en tiempo real, promete revolucionar la forma en que se gestionan las cadenas de suministro en la industria alimentaria. Este proyecto se enmarca en un contexto profesional en el que la industria busca no solo mejorar su eficiencia operativa, sino también cumplir con las expectativas modernas de servicio al cliente y sostenibilidad ambiental. La necesidad de reducir el desperdicio de alimentos, optimizar los recursos y ofrecer productos frescos y de alta calidad, requiere una integración inteligente de soluciones tecnológicas que estén alineadas con estos objetivos.

Desde un punto de vista académico y técnico, este proyecto me ofrece la oportunidad de aplicar y expandir mis conocimientos en áreas clave de la ingeniería informática, como la automatización, el diseño de sistemas de software y la gestión de datos. La fusión de estas habilidades con mi experiencia en la industria alimentaria crea una base sólida para abordar de manera efectiva los desafíos actuales y futuros de este sector.

Más allá de las motivaciones técnicas y profesionales, este proyecto representa también una realización personal, ya que me permite contribuir de manera significativa a un campo que no solo afecta a la economía global, sino también a la vida diaria de millones de personas. A través de este trabajo, aspiro a demostrar cómo la integración adecuada de tecnología avanzada en las cadenas de suministro puede resultar en mejoras sustanciales en términos de eficiencia, reducción de desperdicios y satisfacción del cliente. Este proyecto no solo busca ofrecer soluciones prácticas a problemas significativos, sino también sentar las bases para futuras innovaciones que puedan continuar mejorando la industria alimentaria en su conjunto.

## 1.2 Objetivos

El objetivo principal de este proyecto es desarrollar una solución avanzada para la gestión de cadenas de suministro en la industria alimentaria, que logre mejoras sustanciales en términos de eficiencia, automatización y previsión. La propuesta se enfoca en el diseño de un sistema integrado que conecte las compras de los clientes con la gestión de inventarios, aprovechando tecnologías como la inteligencia artificial y sensores IoT. Estas herramientas permitirán anticipar la demanda y monitorear en tiempo real las condiciones de los productos, lo que facilitará una toma de decisiones más precisa y efectiva.

Además de optimizar los procesos operativos actuales, la solución pretende automatizar tareas clave, como la verificación de inventarios al realizar un pedido y la gestión del registro de clientes. Este enfoque no solo reducirá la incidencia de errores comunes, como la duplicación de registros, sino que también garantizará que los pedidos se acepten únicamente cuando puedan ser cumplidos, mejorando así la fiabilidad y la eficiencia operativa.

El sistema será diseñado con características de escalabilidad y adaptabilidad, lo que permitirá su evolución conforme cambien las necesidades del sector. Esta flexibilidad facilitará la integración de nuevas tecnologías y la adaptación a variaciones en la demanda o en las condiciones del mercado, asegurando que la solución mantenga su relevancia y eficacia a lo largo del tiempo. Asimismo, la implementación de esta solución contribuirá a promover prácticas más sostenibles dentro de la industria, alineando las operaciones con las expectativas actuales de responsabilidad social y medioambiental, reduciendo el desperdicio y optimizando el uso de recursos.

Por otro lado, este proyecto representa una oportunidad para aplicar tecnologías emergentes en la gestión de cadenas de suministro, con el objetivo de desarrollar una

solución alineada con las tendencias más innovadoras de la industria. La creación de un prototipo basado en problemas reales permitirá no solo abordar necesidades inmediatas, sino también adquirir un conocimiento profundo de estas nuevas tecnologías. Este aprendizaje será fundamental para preparar al equipo para enfrentar futuros desafíos y garantizar que la solución se mantenga competitiva en un entorno empresarial dinámico.

## 1.3 Impacto esperado

Este proyecto tiene el potencial de transformar significativamente la gestión de cadenas de suministro en la industria alimentaria, ofreciendo una solución que no solo optimiza la gestión de inventarios y clientes, sino que también integra tecnologías avanzadas como la inteligencia artificial y sensores IoT para lograr una mayor eficiencia y automatización. La arquitectura escalable del sistema permitirá su expansión y adaptación a las demandas futuras del sector, facilitando la incorporación de nuevas tecnologías y mejorando continuamente la operatividad de las empresas.

Se espera que la implementación de este sistema incremente la satisfacción del cliente gracias a una cadena de suministro más ágil y automatizada, que mejora la gestión de recursos y permite anticipar la demanda de manera más precisa. Esto no solo reducirá costos operativos, sino que también mejorará la rentabilidad al optimizar el flujo de productos y minimizar el desperdicio.

**Para los Proveedores de Alimentos:** La solución permitirá una mejora sustancial en la coordinación y eficacia de sus operaciones. La capacidad de prever la demanda con mayor precisión mediante inteligencia artificial reducirá la necesidad de mantener grandes excedentes de stock, ayudando a mantener niveles óptimos de inventario. El monitoreo en tiempo real de los productos garantizará que se conserven en condiciones óptimas de frescura y calidad, lo que a su vez minimizará el desperdicio y permitirá una distribución más eficiente, alineada con la demanda real.

**Para los Consumidores:** Los beneficios incluyen una mayor disponibilidad de productos frescos y de alta calidad. La capacidad del sistema para anticipar y responder rápidamente a la demanda, junto con la automatización en la gestión de inventarios, asegurará que los productos lleguen al consumidor de manera más rápida y eficiente. Esto mejorará la experiencia de compra, fomentando hábitos de consumo más sostenibles y satisfaciendo las expectativas de los consumidores modernos.

**Impacto en la Sostenibilidad:** La reducción del desperdicio de alimentos y la mejora en la eficiencia de la distribución contribuirán significativamente a los Objetivos de Desarrollo Sostenible (ODS), en particular al ODS 12 (Producción y Consumo Responsables) y al ODS 13 (Acción por el Clima), al disminuir la huella de carbono asociada a la logística alimentaria. Además, la adopción de innovaciones tecnológicas como la inteligencia artificial y los sensores no solo mejora la infraestructura del sector, sino que también respalda el ODS 9 (Industria, Innovación e Infraestructura), promoviendo un desarrollo más sostenible y resiliente.

**Aprendizaje y Aplicación de Nuevas Tecnologías:** Otro impacto significativo de este proyecto es la oportunidad de adquirir y aplicar conocimientos en tecnologías emergentes. Al desarrollar un prototipo basado en problemas reales, el equipo no solo abordará necesidades inmediatas, sino que también obtendrá una comprensión profunda de las tendencias tecnológicas actuales en la industria. Este aprendizaje contribuirá a posicionar al equipo y a la organización en la vanguardia de la innovación, mejorando su capacidad para enfrentar futuros desafíos y mantenerse competitivos en un entorno empresarial en constante cambio.

En resumen, este proyecto no solo tendrá un impacto positivo en la eficiencia y rentabilidad de las operaciones empresariales, sino que también contribuirá al logro de metas globales de sostenibilidad y responsabilidad ambiental. Además, proporcionará una valiosa experiencia en el uso de tecnologías avanzadas, fortaleciendo la capacidad del equipo para liderar en un mercado dinámico y en evolución.

## 1.4 Estructura de la memoria

Este documento está organizado en varios capítulos diseñados para facilitar la comprensión y el seguimiento del contenido técnico y metodológico. A continuación, se detalla brevemente lo que el lector puede esperar encontrar en cada uno de los capítulos incluidos en este proyecto:

- **Introducción:** Se expone el propósito del trabajo, la motivación detrás del proyecto, y los objetivos específicos que se buscan alcanzar. Además, se ofrece una visión general sobre la importancia de la integración de aplicaciones en el contexto empresarial actual y cómo este proyecto aborda los desafíos relacionados.
- **Contexto Tecnológico:** Este capítulo explora el desarrollo histórico y las tendencias actuales en la integración de aplicaciones, desde el uso de middleware hasta la evolución de las arquitecturas API-led. Se analizan tecnologías clave como las APIs REST, los microservicios y la inteligencia artificial, justificando la elección de las mismas para este proyecto. También se abordan las futuras tendencias y desafíos que podrían impactar la integración de sistemas empresariales.
- **Análisis del Problema:** Se presenta la problemática que este proyecto pretende resolver, relacionada con la necesidad de integrar múltiples sistemas y aplicaciones en un entorno empresarial moderno. Se exploran y comparan diferentes enfoques y tecnologías de integración, seleccionando la solución más adecuada para el desarrollo del proyecto.
- **Diseño de la Solución:** Basado en el análisis realizado, se elabora un diseño detallado de la solución propuesta. Este capítulo justifica la arquitectura y la tecnología seleccionada, profundizando en los detalles técnicos de la implementación, incluyendo la estructura API-led con sus tres capas principales: sistema, proceso y experiencia.
- **Desarrollo de la Solución Propuesta:** Se describe cómo se ha implementado la solución, detallando el proceso de creación y configuración de las APIs utilizando herramientas como RAML y Anypoint Studio. Se explora el ciclo de vida de las APIs desde su diseño hasta su monitoreo y mantenimiento.
- **Pruebas e Implantación:** Se explica el proceso realizado para trasladar la solución del entorno de desarrollo al de producción, incluyendo la realización de pruebas para asegurar que el sistema funcione correctamente según lo previsto. Además, se abordan las estrategias de gestión de APIs y de datos para garantizar la seguridad y el cumplimiento normativo.
- **Conclusiones:** Se reflexiona sobre los aprendizajes obtenidos durante el desarrollo del proyecto, evaluando el cumplimiento de los objetivos establecidos. También se discuten las dificultades encontradas y cómo se resolvieron, así como las posibles futuras mejoras y líneas de trabajo a partir de los resultados obtenidos.

# CAPÍTULO 2
# Contexto Tecnológico

La integración de aplicaciones se ha convertido en un componente esencial para las empresas que buscan mejorar la interacción entre sistemas dispares. Históricamente, las organizaciones utilizaban métodos manuales o sistemas aislados que resultaban ineficientes. Con el avance de la tecnología, se desarrollaron soluciones como middleware y arquitectura orientada a servicios (SOA), facilitando la comunicación y automatización entre sistemas empresariales y aplicaciones [1].

Inicialmente, la integración se realizaba mediante técnicas que requerían una configuración manual extensa y eran difíciles de mantener. A medida que las empresas comenzaron a enfrentarse a mercados más dinámicos y globalizados, la necesidad de sistemas más flexibles y adaptables se hizo evidente. Esto llevó al desarrollo de SOA en la década de 1990, que promovía la idea de servicios de software reutilizables y fácilmente integrables [1].

Además, el auge de Internet y la tecnología web a principios de los años 2000 transformó radicalmente la integración de aplicaciones. Las API (Interfaces de Programación de Aplicaciones) comenzaron a ganar popularidad como un medio para permitir que las aplicaciones de software interactuaran entre sí de manera más abierta y estándar. Las API web, en particular, se convirtieron en una herramienta crucial para construir aplicaciones escalables y mantenibles que podrían interactuar con cualquier otro sistema en línea, sin importar la plataforma subyacente [6, 7].

La introducción de tecnologías como XML y JSON, junto con protocolos de comunicación como SOAP y REST, facilitó aún más este proceso, permitiendo que las aplicaciones no solo compartan funciones sino también datos de manera estructurada y predecible. Esta evolución ha llevado a la creación de arquitecturas basadas en microservicios, donde las aplicaciones están compuestas por servicios independientes que se comunican a través de la red [2].

Este paradigma de microservicios, junto con el enfoque API-led, representa la vanguardia de la integración de aplicaciones hoy en día. Las organizaciones pueden ahora diseñar sistemas que son verdaderamente modulares, donde los cambios en una parte del sistema pueden realizarse con un impacto mínimo en los demás componentes. Este enfoque no solo mejora la agilidad y la capacidad de respuesta de las empresas sino que también reduce significativamente los costos de mantenimiento y escalabilidad. Este panorama tecnológico en constante evolución subraya la importancia de adoptar una estrategia proactiva en la integración de sistemas, garantizando que las empresas no solo se mantengan competitivas sino que también puedan capitalizar las oportunidades en un entorno de negocios que cambia rápidamente [21].

## 2.1 API REST

La API REST, basada en el principio de transferencia de estado representacional (Representational State Transfer), es un tipo de API que utiliza métodos HTTP estándar y es especialmente útil en la integración web debido a su flexibilidad y ligereza. La arquitectura REST se ha convertido en una piedra angular para el desarrollo de servicios web y microservicios, proporcionando una manera estándar y eficiente de estructurar comunicaciones entre cliente y servidor [22].

El uso de REST ha facilitado la expansión del diseño de servicios web y la arquitectura de microservicios, siendo esencial para muchos servicios en la nube modernos (Figura 2.1). Estas APIs permiten a los desarrolladores manejar las solicitudes HTTP para crear, leer, actualizar y eliminar recursos, lo que representa operaciones CRUD en las bases de datos.

Además, REST es altamente escalable, permitiendo a los servicios evolucionar con el tiempo sin interrumpir la funcionalidad del cliente. Su naturaleza sin estado significa que cada llamada a la API desde un cliente contiene toda la información necesaria para completar la solicitud. Como resultado, las aplicaciones REST pueden servir a un número mucho mayor de solicitudes y son más fáciles de escalar en ambientes de alta demanda [6].

REST ha sido adoptado ampliamente debido a su capacidad para simplificar las interacciones del backend y soportar comunicaciones entre diversas tecnologías, facilitando así la integración de sistemas heterogéneos. Esta compatibilidad es crucial para las empresas que utilizan una variedad de tecnologías y plataformas en sus operaciones diarias [23].

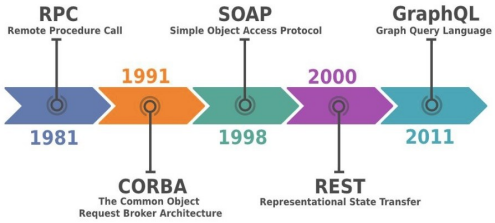


**Figura 2.1:** Cronología de la evolución de las API

En resumen, las APIs REST son un componente fundamental en la arquitectura de software moderna, permitiendo la creación de aplicaciones robustas, mantenibles y escalables que son capaces de operar sobre una variedad de plataformas y entornos.

## 2.2 Tendencias Futuras y Desafíos en la Integración de Aplicaciones

A pesar de los significativos avances tecnológicos, la integración de aplicaciones continúa enfrentando desafíos críticos que impactan directamente en la eficacia de las soluciones empresariales. Uno de los principales retos es la seguridad de los datos, especialmente relevante en un contexto donde las brechas de seguridad pueden tener consecuencias devastadoras [24]. La creciente sofisticación de los ataques cibernéticos y la mayor exposición de datos a través de múltiples APIs hacen que la seguridad sea una prioridad esencial en cualquier estrategia de integración.

En el futuro, se espera que la integración de aplicaciones explore más profundamente el uso de tecnologías emergentes como la inteligencia artificial (IA) y el aprendizaje automático (ML). Estas tecnologías prometen revolucionar la manera en que se optimizan los procesos de integración, permitiendo respuestas más rápidas y precisas a las dinámicas del mercado y a las necesidades operativas [26]. Por ejemplo, la IA puede utilizarse para predecir y prevenir problemas de rendimiento en tiempo real, mejorando la resiliencia y la eficiencia operativa.

Un desafío importante en el futuro será la integración de dispositivos IoT (Internet de las Cosas), que requerirá soluciones de integración robustas y escalables para manejar la enorme cantidad de datos generados por estos dispositivos. La arquitectura orientada a eventos y los microservicios (Figura 2.2) serán fundamentales para soportar esta integración, permitiendo a las empresas reaccionar en tiempo real a los cambios en su entorno operativo [28]. Con la expansión de IoT, la capacidad de integrar y analizar datos de múltiples dispositivos en tiempo real se convertirá en un diferenciador clave para las empresas.

Además, la aparición de tecnologías como 5G y el edge computing facilitará la creación de arquitecturas más descentralizadas, donde los datos se procesan más cerca de su origen. Esto reducirá la latencia y permitirá una toma de decisiones más rápida y eficiente, especialmente en entornos críticos como la manufactura, la salud y la logística [29]. El edge computing, en particular, permite a las organizaciones mover el procesamiento de datos a la periferia de la red, lo que es crucial para aplicaciones que requieren respuestas inmediatas, como los vehículos autónomos o la automatización industrial.

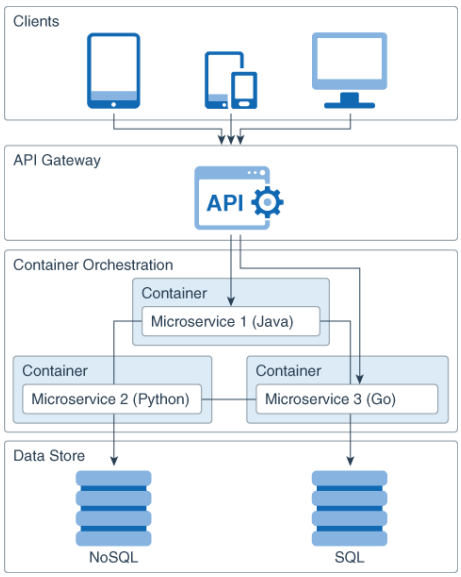


**Figura 2.2:** Arquitectura de Microservicios Facilitada por APIs

## 2.3 Arquitectura API-led

La arquitectura API-led (Figura 2.3) es una aproximación estratégica para la integración de sistemas y aplicaciones en las empresas, promoviendo la creación de conexiones significativas y reutilizables a través de interfaces de programación de aplicaciones (APIs) bien definidas. Esta metodología estructura la integración en tres capas principales, cada una con un propósito específico en la gestión y el flujo de datos [33]:

**Capa de Sistema**: Esta capa es fundamental para exponer los datos de sistemas de fondo, como bases de datos y sistemas de gestión de recursos empresariales (ERP), a través de APIs. Facilita el acceso seguro y controlado a los datos corporativos, permitiendo que otros servicios y aplicaciones consuman y utilicen estos datos sin necesidad de interactuar directamente con los sistemas subyacentes.

**Capa de Proceso**: Actúa como un orquestador entre las diversas capas de sistema y las aplicaciones que consumen sus datos. Esta capa toma los datos expuestos por las APIs de la capa de sistema y aplica la lógica de negocio necesaria para preparar y entregar la información de manera que sea útil y aplicable en los procesos empresariales. Esto incluye la transformación de datos, la implementación de reglas de negocio, y la coordinación de transacciones y datos a través de múltiples sistemas .

**Capa de Experiencia**: La última capa es donde se personaliza la interacción de los usuarios con los datos y procesos gestionados por las capas inferiores. Aquí, las APIs están diseñadas para ofrecer una experiencia de usuario optimizada para diferentes tipos de clientes, como aplicaciones móviles, portales web, y otros sistemas de interfaz de usuario. Esta capa asegura que los usuarios finales reciban una vista y funcionalidad adaptadas a sus necesidades específicas, mejorando la accesibilidad y la usabilidad de los sistemas de información empresariales.

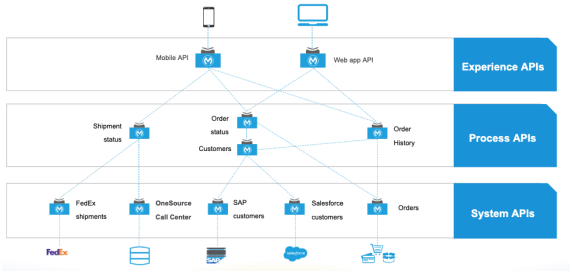


**Figura 2.3:** Arquitectura API-led dividida en sus tres capas principales

La implementación de una arquitectura API-led no solo facilita la integración eficiente y flexible de tecnologías dispares sino que también promueve la agilidad empresarial, permitiendo a las organizaciones adaptarse rápidamente a los cambios del mercado y a las nuevas oportunidades de negocio [21]. Esto es especialmente relevante en un entorno empresarial dinámico donde la capacidad de respuesta y la innovación constante son claves para mantener una ventaja competitiva.

En la práctica, las organizaciones que adoptan una arquitectura API-led pueden experimentar mejoras significativas en la escalabilidad y reutilización de sus servicios. Al desacoplar las capas de sistema, proceso y experiencia, se logra una mayor modularidad, lo que facilita las actualizaciones y el mantenimiento de los sistemas sin afectar las operaciones críticas [30].

Además, el enfoque API-led permite una integración más coherente con servicios en la nube, donde las APIs juegan un papel crucial en la interoperabilidad entre diferentes entornos y tecnologías. Este enfoque también soporta una mejor gobernanza y seguri-

dad de los datos, ya que las APIs pueden ser monitorizadas y gestionadas con mayor precisión a lo largo de todo el ciclo de vida de la aplicación [23].

## 2.4 Ciclo de Vida de las APIs

En este contexto, el ciclo de vida de las APIs (Figura 2.4) se convierte en un pilar fundamental dentro de la estrategia de integración de aplicaciones. El ciclo de vida abarca todas las etapas desde el diseño y prototipado hasta el despliegue, monitoreo, mantenimiento, versionado, y eventual retiro de las APIs [33].

- **Diseño y Prototipado**: Esta etapa inicial se centra en la definición de los recursos y métodos que estarán disponibles, alineándose con las necesidades del negocio para resolver problemas específicos.
- **Desarrollo y Pruebas**: Aquí se implementa la API y se somete a pruebas exhaustivas para asegurar su funcionalidad, seguridad, y rendimiento antes de su lanzamiento.
- **Despliegue**: La API se despliega en un entorno de producción, convirtiéndose en accesible para los usuarios, y se empieza a monitorear su uso y rendimiento.
- **Monitoreo y Mantenimiento**: En producción, la API requiere un monitoreo continuo para asegurar que sigue cumpliendo con los objetivos del negocio, además de recibir actualizaciones y mejoras.
- **Versionado y Retiro**: Finalmente, las APIs pueden requerir versiones nuevas o ser retiradas, gestionando cuidadosamente este proceso para minimizar el impacto en los usuarios.

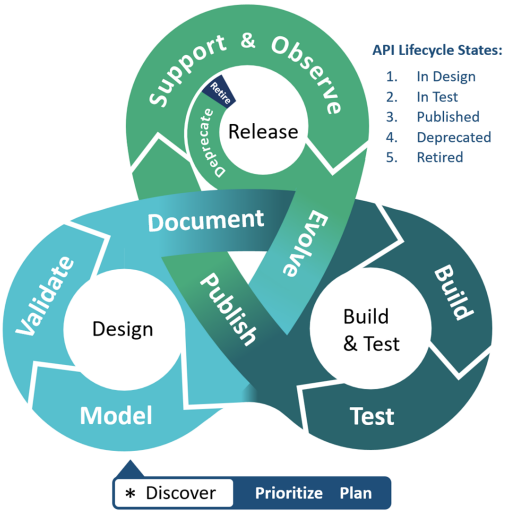


**Figura 2.4:** Ciclo de Vida de las APIs.

Este ciclo de vida es crucial en la integración de aplicaciones modernas, garantizando que las APIs sigan siendo eficientes, seguras, y relevantes a medida que evolucionan las necesidades empresariales y tecnológicas [25].

## 2.5 API Management y Gestión de Datos

La gestión de APIs (API Management) es crucial en el entorno empresarial moderno, donde las APIs se han convertido en la columna vertebral de la integración de aplicaciones, permitiendo así a las empresas controlar el acceso a sus APIs, implementar políticas de seguridad, y monitorear el uso de APIs en tiempo real para detectar y mitigar problemas antes de que afecten a los usuarios finales [30].

La gestión de datos se relaciona estrechamente con la gestión de APIs, ya que asegura que los datos que se comparten a través de APIs son consistentes, seguros y cumplen con las normativas relevantes [25]. La combinación de API Management y un robusto gobierno de datos es esencial para garantizar la integridad y seguridad de la información en un entorno donde las aplicaciones y sistemas interactúan continuamente [23].

Este enfoque también permite a las empresas cumplir con normativas globales de protección de datos, como el GDPR en Europa, asegurando que los datos sensibles se manejen de acuerdo con las regulaciones establecidas, reduciendo así el riesgo de violaciones de datos y sanciones legales [30].

## 2.6 Colas de Mensajería

En los sistemas distribuidos y en la integración de aplicaciones, las colas de mensajería (2.5) son fundamentales para gestionar la comunicación entre diferentes componentes o servicios de manera asíncrona. Una cola de mensajería es un mecanismo que permite que los mensajes enviados por un componente sean almacenados temporalmente en una cola hasta que otro componente esté listo para procesarlos [31].

Las colas de mensajería permiten desacoplar la producción y el consumo de mensajes, lo que es esencial en sistemas donde los componentes pueden operar a velocidades diferentes o donde pueden ocurrir fallos temporales. Este desacoplamiento asegura que los mensajes no se pierdan y que el sistema pueda recuperarse de fallos sin perder información crítica [31].

En la arquitectura moderna, las colas de mensajería son ampliamente utilizadas en aplicaciones que requieren alta disponibilidad, escalabilidad y resiliencia. Por ejemplo, en entornos de microservicios, cada servicio puede enviar y recibir mensajes a través de una cola de mensajería, lo que facilita la comunicación entre servicios sin necesidad de que estén directamente conectados [2].

Además, las colas de mensajería son vitales en escenarios donde es necesario manejar grandes volúmenes de datos o eventos en tiempo real, como en la monitorización de sistemas, la transmisión de datos de sensores o la ejecución de procesos empresariales complejos. La capacidad de manejar picos de carga sin comprometer la integridad de los datos hace que las colas de mensajería sean una herramienta indispensable en la arquitectura de sistemas distribuidos [31].


**Figura 2.5:** Arquitectura General de Colas de Mensajería

## 2.7 Impacto de la Inteligencia Artificial en la Integración de Aplicaciones

La inteligencia artificial está comenzando a integrarse con arquitecturas API-led y microservicios, ofreciendo nuevas capacidades en la automatización de decisiones y la optimización de flujos de trabajo. Por ejemplo, la IA puede analizar patrones en los datos de inventario para anticipar la demanda, permitiendo a las empresas ajustar su producción y distribución de manera más eficiente [27].

La integración de IA en la gestión de APIs también está permitiendo la creación de APIs más inteligentes, que pueden adaptarse a las necesidades del usuario en tiempo real y proporcionar recomendaciones personalizadas basadas en el comportamiento y las preferencias del usuario [26].

Este enfoque está facilitando el desarrollo de sistemas más resilientes y adaptativos, capaces de gestionar una mayor complejidad y de responder rápidamente a los cambios en el entorno empresarial [25]. A medida que las capacidades de IA continúan avanzando, se espera que su integración en la infraestructura de aplicaciones se convierta en un diferenciador clave para las empresas que buscan mantener una ventaja competitiva en un mercado en constante evolución [26].

## 2.8 Análisis Comparativo de Tecnologías de Integración

En este contexto, es importante comparar diferentes tecnologías de integración para entender sus ventajas y limitaciones. Por ejemplo, mientras que SOAP ofrece un enfoque estructurado y robusto para la integración de aplicaciones empresariales, REST proporciona mayor flexibilidad y simplicidad para aplicaciones web. GraphQL, por otro lado, permite una mayor eficiencia en las interacciones con la API, permitiendo a los clientes definir exactamente los datos que necesitan [23, 25].

Cada una de estas tecnologías tiene sus propios casos de uso ideales, y la elección de la tecnología adecuada depende en gran medida de los requisitos específicos del proyecto, incluyendo la necesidad de escalabilidad, seguridad, y facilidad de mantenimiento [22].

Por ejemplo, SOAP sigue siendo popular en entornos donde se requiere un alto nivel de seguridad y transacciones complejas, mientras que REST se ha convertido en el estándar de facto para la mayoría de las aplicaciones web modernas debido a su simplicidad y flexibilidad [23]. GraphQL, por su parte, está ganando terreno en aplicaciones donde la eficiencia en la transferencia de datos es crucial, ya que permite a los clientes obtener exactamente los datos que necesitan en una única solicitud [32].

## 2.9 Conclusión

La integración de aplicaciones y el desarrollo de APIs han jugado roles indispensables en la transformación digital de las empresas. La arquitectura API-led, en particular, ha sido fundamental, proporcionando un marco robusto para mejorar la interoperabilidad y la agilidad empresarial. Este enfoque ha permitido a las organizaciones adaptarse más rápidamente a las condiciones cambiantes del mercado, integrar nuevas tecnologías con mayor eficacia y mejorar la experiencia general del cliente [21].

Mirando hacia el futuro, la integración continuará siendo un campo de innovación importante, especialmente a medida que las empresas busquen aprovechar las capaci-

dades avanzadas de la IA y el ML para superar los desafíos actuales y capitalizar las oportunidades emergentes en el panorama digital global [26]. La adopción de tecnologías emergentes como IoT, 5G, y edge computing también desempeñará un papel crucial en la próxima evolución de las arquitecturas de integración, permitiendo a las empresas crear sistemas más descentralizados, resilientes y adaptativos que pueden responder de manera efectiva a las demandas de un mundo cada vez más conectado [29].

# CAPÍTULO 3
# Análisis del problema

Este proyecto aborda un desafío importante en la gestión de la cadena de suministro en la agricultura al centrarse en la optimización del proceso de cultivo y recolección de alimentos en respuesta directa a las demandas del consumidor. Este proyecto busca implementar un sistema avanzado de gestión de inventario para alinear la producción agrícola con las expectativas de compra del mercado mediante el uso de una plataforma de comercio electrónico que comercializa productos cultivados directamente en tierras propias.

Uno de los problemas actuales en la gestión tradicional de la cadena de suministro agrícola es la incapacidad para adaptar la producción y el inventario eficientemente ante fluctuaciones rápidas en la demanda del consumidor. Esto no solo incrementa el riesgo de sobreproducción y desperdicio de alimentos, sino que también puede resultar en una pérdida de ingresos y en una disminución de la satisfacción del cliente cuando los productos deseados no están disponibles. Para abordar este problema, el proyecto propone la integración de tecnologías avanzadas de gestión de inventario para analizar tendencias de compra, lo que permitirá anticipar las necesidades del mercado y ajustar automáticamente los niveles de producción e inventario en los almacenes.

Además, la logística de asignación de tareas en las fincas presenta desafíos significativos debido a la falta de planificación anticipada, afectando la eficiencia en el transporte y la organización del trabajo. Implementar un sistema que mejore la comunicación y coordinación de las actividades agrícolas puede llevar a una notable mejora en la eficiencia operativa. La introducción de sensores para monitorear las condiciones ambientales y el estado de los cultivos se propone como una solución para optimizar la calidad y rendimiento de los productos, aunque el desarrollo completo de esta tecnología puede no ser alcanzable dentro del marco temporal del proyecto.

Este enfoque global tiene como objetivo mejorar la eficiencia y reducir el desperdicio, además de aumentar la sostenibilidad de las prácticas agrícolas, alineando las operaciones con las expectativas modernas de responsabilidad social y ambiental. El objetivo del proyecto es establecer un nuevo estándar en la gestión de cadenas de suministro agrícolas que sea replicable y escalable a nivel mundial mediante la integración de estas tecnologías.

## 3.1 Especificación de Requisitos del Sistema

La especificación de los requisitos tanto funcionales como no funcionales es crucial para el éxito de cualquier proyecto tecnológico. En el contexto de nuestro sistema de gestión agrícola, estos requisitos definen cómo la plataforma debe operar para satisfacer las necesidades de los usuarios y mantener la integridad operacional bajo diferentes condiciones.

**Requisitos Funcionales:** Son aquellos que describen las interacciones específicas entre el sistema y sus usuarios, así como las operaciones internas que debe realizar el software para cumplir con las funcionalidades descritas. Por ejemplo, nuestro sistema debe permitir a los clientes realizar pedidos a través de un portal web, gestionar estos pedidos en el backend y actualizar el inventario en tiempo real según la demanda. Además, el sistema debe interactuar con sensores ambientales para monitorear las condiciones de los cultivos directamente en el campo, asegurando que se mantenga su frescura y calidad.

**Requisitos No Funcionales:** Estos requisitos se relacionan con la calidad del sistema, como el rendimiento, la seguridad, la portabilidad y la escalabilidad. Por ejemplo, el sistema debe ser capaz de manejar un alto volumen de transacciones durante los períodos pico de demanda sin degradar el rendimiento ni la experiencia del usuario. También debe garantizar la seguridad de los datos del cliente, especialmente la información personal y de pago, mediante el cumplimiento de las normativas de protección de datos aplicables.

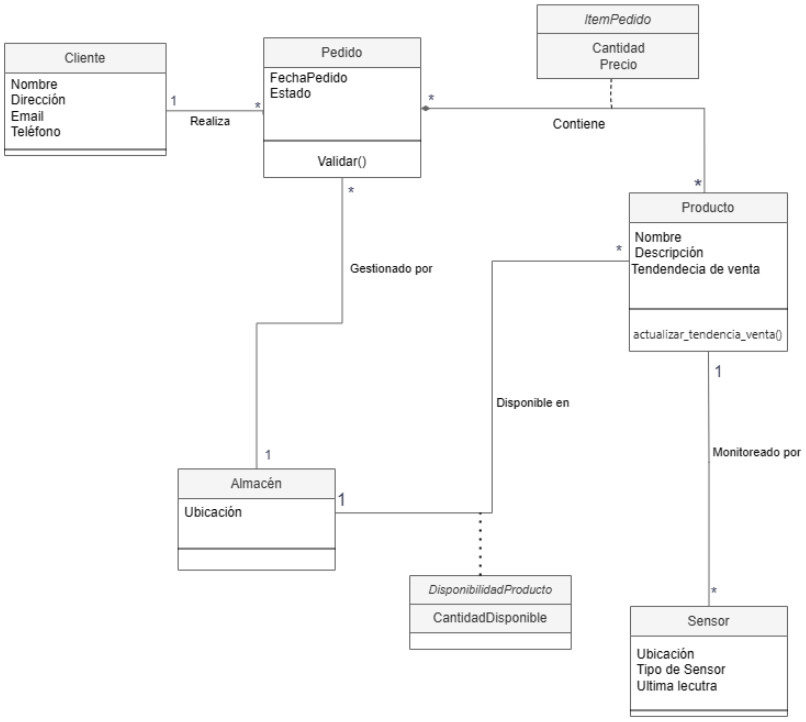


**Figura 3.1:** Diagrama UML del sistema de gestión agrícola

El diagrama UML proporcionado en la Figura 3.1 ilustra la estructura del sistema, mostrando cómo se relacionan las diferentes entidades como clientes, pedidos, productos y almacenes. Este modelo es fundamental para entender el flujo de datos y las responsabilidades de cada componente dentro del sistema.

- **Cliente**: Representa los consumidores finales. Atributos incluyen nombre, dirección, email y teléfono. Relación uno a muchos con la clase **pedido**, permitiendo a un cliente realizar múltiples pedidos.
- **Pedido**: Gestiona la información de los pedidos realizados por los clientes, incluyendo fecha y estado. Relacionada uno a muchos con **item/pedido**, reflejando que cada pedido puede incluir varios ítems. Incorpora funciones como `validar()` para asegurar el cumplimiento de los pedidos.
- **Producto e Item/Pedido**: **Producto** incluye detalles como nombre, descripción y tendencia de venta, útiles para la gestión del inventario y análisis de compra. Relaciona productos con ítems específicos en pedidos, manejando la cantidad y precio.
- **Almacén**: Esencial para la logística de inventario, gestionando ubicaciones de almacenamiento y disponibilidad de productos.
- **Sensor**: Monitorea condiciones ambientales cruciales para la calidad del producto, con atributos como ubicación, tipo y última lectura, proporcionando datos vitales para decisiones agrícolas informadas.

Este modelo es fundamental para comprender el flujo de datos y responsabilidades de cada componente, asegurando que todas las operaciones estén alineadas con los objetivos generales del proyecto y optimizando la eficiencia operativa y la satisfacción del cliente.

A partir de este análisis inicial de los requisitos y del modelo del sistema, podemos proceder a profundizar en las especificaciones técnicas y comenzar a diseñar las soluciones detalladas que abordarán los desafíos identificados durante la fase de análisis del problema. Este enfoque sistemático asegura que cada aspecto del sistema está bien definido y alineado con los objetivos generales del proyecto.

## 3.2 Posibles soluciones

Para garantizar una toma de decisiones informada y justificada, se exploraron varias alternativas tecnológicas antes de seleccionar la solución óptima. Cada opción fue evaluada en base a su capacidad para integrar análisis avanzados y gestionar eficazmente la cadena de suministro agrícola, teniendo en cuenta los costos, beneficios y desafíos asociados.

Una solución considerada fue la implementación de un **Sistema de Gestión de Recursos Empresariales** (ERP) especializado en el sector agrícola que pudiera manejar todos los aspectos de la producción, desde la siembra hasta la distribución. Este sistema prometía una integración completa con capacidades avanzadas para la planificación de recursos y la gestión de inventarios. Sin embargo, los principales inconvenientes incluían un alto costo de implementación y mantenimiento, así como la necesidad de personalizar extensamente el software para adaptarlo a las necesidades específicas del proyecto, lo que podría resultar en un aumento significativo del tiempo y el presupuesto requerido.

Otra opción era **adaptar una plataforma de comercio electrónico existente** para incluir funcionalidades adicionales que gestionaran las especificidades del ciclo agrícola y la cadena de suministro. Esto implicaría menos desarrollo desde cero y podría integrarse con sistemas de comercio electrónico ya utilizados para la venta y distribución de productos agrícolas. Aunque esta solución podría ser más económica y rápida de implementar, ofrecía menos flexibilidad para el manejo de operaciones agrícolas complejas y no proporcionaba la misma profundidad de análisis y control sobre el inventario y los procesos de producción como un sistema dedicado.

La tercera alternativa explorada fue una **plataforma basada en la nube** que integrara inteligencia artificial (IA) y aprendizaje automático para prever demandas y optimizar el inventario y los recursos de producción. Esta tecnología permitiría análisis predictivos avanzados y podría adaptarse dinámicamente a las condiciones cambiantes del mercado y el clima. Aunque altamente innovadora, esta opción requería una inversión significativa en desarrollo tecnológico y capacitación personal, y presentaba riesgos asociados a la dependencia de algoritmos para decisiones críticas de negocio.

## 3.3 Solucion propuesta

Después de considerar todas las opciones, se determinó que la solución híbrida propuesta, que combina la integración API con sistemas de gestión internos, ofrecía el mejor equilibrio entre funcionalidad, coste, tiempo de implementación y escalabilidad. Esta solución permite una integración efectiva con las plataformas existentes mientras facilita la incorporación de tecnologías avanzadas de análisis de datos y sensorización para una gestión más eficiente y proactiva del inventario y la producción agrícola.

**Fases del desarrollo del proyecto:**

1. **Planificación y Diseño:** En esta fase inicial, se definirán los requisitos detallados del sistema y se diseñará la arquitectura de la solución para integrarse de manera óptima con la aplicación de comercio electrónico existente. Se seleccionarán las tecnologías adecuadas para el desarrollo, incluyendo la configuración de la infraestructura de las APIs y la selección de plataformas para el análisis predictivo y la integración con los sensores ambientales.

2. **Desarrollo e Integración:** Durante esta fase, se adaptarán y desarrollarán los módulos del sistema para integrarse con la aplicación de comercio electrónico ya implementada, incluyendo la lógica de negocio en el backend y la integración con bases de datos y sensores. Se emplearán prácticas de programación ágil para facilitar iteraciones rápidas y adaptativas.

3. **Pruebas y Validación:** El sistema será sometido a una serie de pruebas para garantizar su funcionalidad, seguridad y rendimiento. Esto incluirá pruebas unitarias, pruebas de integración y pruebas de aceptación de usuario, asegurando que el sistema integrado cumple con los requisitos especificados y opera robustamente con la aplicación de comercio electrónico.

4. **Implementación y Despliegue:** En esta fase, el sistema integrado será implementado en un entorno de producción. Se realizarán ajustes basados en el feedback del usuario y se monitoreará el sistema para resolver cualquier problema técnico que pueda surgir durante el despliegue inicial.

**Validación y Pruebas:** Actualmente, la validación completa del sistema, incluyendo pruebas de campo para evaluar la eficacia de los sensores en el monitoreo de las condiciones de cultivo y la precisión del análisis predictivo, no se llevará a cabo inmediatamente. Sin embargo, se considera una fase importante para futuras iteraciones del proyecto. En el futuro, se planifica realizar ensayos para testear la fiabilidad del sistema integrado en diferentes escenarios operacionales. Esto asegurará que la plataforma pueda manejar variaciones en la demanda y adaptarse a cambios en las condiciones ambientales, validando así la efectividad y robustez del sistema bajo condiciones reales de uso.

Esta fase futura es crítica para asegurar que el sistema no solo funcione en condiciones teóricas o controladas sino que también sea capaz de operar eficazmente en el entorno dinámico y a veces impredecible de la agricultura moderna. La implementación de estas pruebas contribuirá a un entendimiento más profundo de cómo la tecnología puede ser optimizada para cumplir mejor con las exigencias del sector agrícola y las necesidades de los usuarios finales.

Este enfoque integral asegura que la solución no solo sea técnicamente viable sino también escalable y eficiente desde el punto de vista operacional, adaptándose a las necesidades cambiantes del mercado agrícola y las expectativas de los clientes.

## 3.4 Metodología de Trabajo

Para el desarrollo de este proyecto, se adoptó la metodología **Scrum**, adaptada a un entorno de trabajo individual. Aunque Scrum es comúnmente utilizado en equipos, su estructura flexible y iterativa también es efectiva en proyectos individuales, permitiendo una organización clara de las tareas y un seguimiento constante del progreso [34].

El proceso se dividió en varias fases claramente definidas:

1. **Investigación de Tecnologías y APIs:** Inicialmente, se realizó una exploración detallada de las tecnologías a utilizar, enfocándome en comprender y seleccionar las APIs relevantes que serían integradas en el proyecto.
2. **Especificaciones:** Se procedió a definir las especificaciones de las interfaces de las APIs. Este paso fue crucial para estructurar correctamente las APIs y facilitar su integración y mantenimiento.
3. **Desarrollo de APIs:** Con las especificaciones ya establecidas, se inició el desarrollo de las APIs. Esta etapa incluyó la codificación de las funcionalidades según lo definido en las especificaciones.
4. **Pruebas de Integración:** Cada API desarrollada fue sometida a pruebas de integración en un entorno local para asegurar su correcto funcionamiento y la integración eficaz con otros componentes del sistema.
5. **Redacción de la Memoria:** Paralelamente al desarrollo, se llevó a cabo la redacción de la memoria del proyecto, documentando los procesos, decisiones técnicas y resultados obtenidos.

Cada una de estas fases constituyó un sprint en el marco de Scrum, comenzando cada uno con una planificación detallada y terminando con una revisión del trabajo realizado y una reflexión sobre el proceso para identificar mejoras continuas.

Además, realicé revisiones diarias del progreso, lo que me permitió identificar y resolver rápidamente cualquier impedimento que surgiera. Esta rutina diaria fue fundamental para mantener el flujo de trabajo y asegurar que se cumplían los objetivos de cada sprint.

Para la planificación global del proyecto, utilicé una vista general de Gantt que me permitió mantener una perspectiva clara de las etapas principales y los hitos del proyecto. Esta herramienta fue útil para alinear los sprints con los plazos críticos y asegurar que el proyecto se mantuviera dentro del cronograma previsto.

La flexibilidad de Scrum fue esencial para adaptarme a los cambios y desafíos que surgieron durante el desarrollo. Además, utilicé herramientas visuales como Kanban para gestionar el flujo de trabajo y tener un seguimiento visual claro del estado de cada tarea [36].

En resumen, esta metodología no solo facilitó una ejecución coherente y eficiente del proyecto, sino que también me permitió adaptarme de manera ágil a cualquier cambio o desafío que surgiera durante el desarrollo. La iteración constante y la evaluación continua fueron clave para mejorar el producto final, asegurando que el resultado estuviera alineado con los objetivos establecidos desde el inicio del proyecto.

# CAPÍTULO 4
# Diseño de la solución

El diseño de la solución es una etapa crucial en el desarrollo de cualquier proyecto tecnológico. Una vez identificados y documentados los requisitos de la aplicación, sistema o hardware a desarrollar, es fundamental tomar una serie de decisiones estratégicas y técnicas sobre cómo llevar a cabo la solución al problema planteado. Este apartado se enfoca en el diseño de la solución, abordando tanto la arquitectura general del sistema como los detalles específicos de implementación.

Este enfoque sistemático asegura que cada aspecto del diseño está bien fundamentado y alineado con los objetivos generales del proyecto, proporcionando una base sólida para la implementación y eventual despliegue de la solución propuesta.

## 4.1 Arquitectura del sistema

El diseño de la arquitectura de nuestro sistema se fundamenta en el modelo API-led, que articula la integración y la funcionalidad a través de tres capas distintas: experiencia, proceso y sistema (Figura 4.1). Cada capa tiene un propósito definido y específico, trabajando en conjunto para crear una solución robusta y escalable que soporta tanto la interacción directa con el usuario final como la manipulación compleja de datos en el backend.

**Capa de Experiencia:** La capa de experiencia está diseñada para ser el primer punto de contacto entre el usuario y el sistema. La API principal en esta capa es `e-web-api`, que gestiona todas las interacciones relacionadas con el cliente, como realizar pedidos y registrar nuevos usuarios. Esta API captura la información del usuario y la envía a la capa de proceso para su posterior tratamiento. Su diseño permite a los usuarios una navegación intuitiva y eficiente, ofreciendo una interfaz clara para la inserción y consulta de datos personales y de pedidos.

**Capa de Proceso:** En esta capa se coordinan todas las solicitudes que vienen de la capa de experiencia. Esta capa es crucial para el manejo de la lógica de negocio y para asegurar que las operaciones necesarias se ejecuten de manera eficiente antes de devolver las respuestas a la capa de experiencia.

Dentro de esta capa, encontramos APIs claves como `p-gestion-pedidosClientes-api`, `p-analisis-predictivo` y `p-datos-sensores`. Estas APIs son responsables de procesar los datos y gestionar las interacciones entre las capas de sistema y experiencia. La api `p-gestion-pedidosClientes` maneja la lógica de los pedidos y clientes, la `p-analisis-predictivo` realiza análisis avanzados para prever tendencias de demanda, y la api `p-datos-sensores` gestiona la información proveniente de los sensores ambientales.

**Capa de Sistema:** La capa de sistema contiene las APIs que interactúan directamente con los sistemas de almacenamiento de datos. En esta capa, se incluyen APIs como `s-inventario-api`, `s-pedidosClientes-api` y `s-api-analisis`. La `s-inventario-api` gestiona las consultas de inventario, asegurando la disponibilidad y precisión de los datos de stock. La `s-pedidosClientes-api` maneja las operaciones relacionadas con los pedidos, facilitando la comunicación con el sistema de almacenamiento para la actualización en tiempo real del estado de los pedidos. Además, la api `s-api-analisis` trabaja en colaboración con herramientas de análisis predictivo para prever tendencias de demanda y comportamiento de los consumidores, permitiendo ajustes en la producción y la logística de manera eficiente.

Cada API está diseñada para responder eficientemente a las solicitudes específicas, transformando los datos necesarios y proporcionando respuestas que permiten a la siguiente capa ejecutar acciones concretas. Este diseño asegura que nuestro sistema no solo sea reactivo, sino también proactivo en la gestión de los recursos agrícolas, mejorando así la eficiencia y la efectividad de toda la cadena de suministro. Además,todos estos componentes se comunican utilizando un **formato JSON** para el intercambio de mensajes, seleccionado por su eficiencia, facilidad de uso y su capacidad para integrarse a la perfección con diversos lenguajes de programación y plataformas, asegurando así una interoperabilidad óptima y un mantenimiento simplificado del sistema.

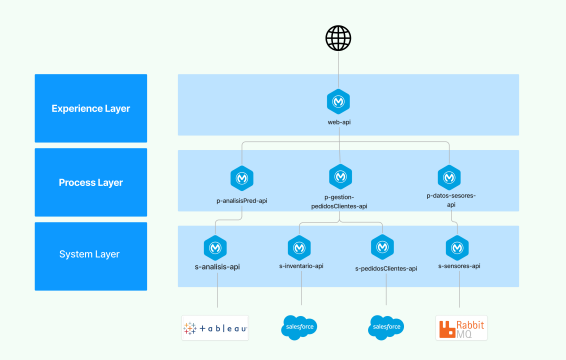


**Figura 4.1:** Diagrama de la arquitectura API-led con las tres capas principales

## 4.2 Diseño detallado

El enfoque para diseñar la solución en este proyecto se caracteriza por un flujo de comunicación claramente estructurado entre las distintas capas de la arquitectura API-led. Cada capa tiene la responsabilidad específica de hacer solicitudes a la capa inmediatamente inferior, facilitando la gestión eficiente de interacciones que van desde la creación de pedidos hasta la administración de datos de clientes y el manejo de información obtenida de sensores. Esta arquitectura está diseñada para asegurar que cada operación necesaria se ejecute de manera óptima, mejorando así la integridad y la eficiencia de todo el proceso. En el desarrollo de este flujo, la capa de experiencia es crucial, ya que gestiona las interacciones iniciales y se comunica con la capa de proceso, la cual, a su vez, interactúa con la capa de sistema para llevar a cabo efectivamente estas operaciones. Además, se considera el uso de un modelo canónico de datos para facilitar las transformaciones necesarias entre las capas, garantizando que la información fluya correctamente y sin errores a través de la arquitectura.

### Creación de pedidos

El proceso de creación de pedidos inicia en la **capa de experiencia** (Figura 4.2), donde el usuario realiza una solicitud para generar un nuevo pedido. Esta solicitud es capturada por la API correspondiente (`e-web-api`), que se encarga de enviar la información necesaria a la capa de proceso. Aquí, la API de gestión de pedidos y clientes (`p-gestion-pedidosClientes-api`) verifica la disponibilidad de los artículos solicitados consultando el inventario a través de la `s-inventario-api`. Este paso es crucial para asegurar que el stock está disponible y puede cumplir con el pedido. Una vez confirmada la disponibilidad, la solicitud es procesada para registrar el pedido en el sistema de gestión (`s-pedidosClientes-api`), que interactúa con nuestro CMS para almacenar y gestionar los pedidos eficazmente.

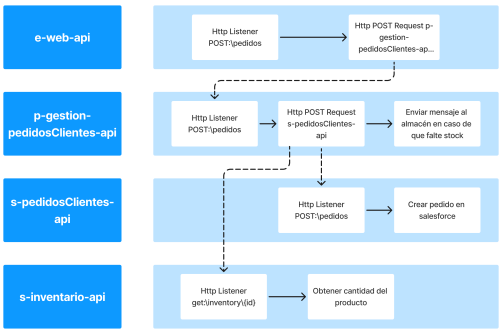


**Figura 4.2:** Diagrama de flujo del proceso de creación de pedidos

La primera tabla (Tabla 4.1) presenta las propiedades esenciales de la API utilizada para la creación de pedidos. Detalla los componentes básicos como el host, puerto, ruta y el método HTTP utilizado para las solicitudes. Esta configuración define cómo y dónde se envían las solicitudes para crear pedidos en el entorno de preproducción.

| Properties | PRE |
|---|---|
| Host | e-web-api.us-e2.cloudhub.io |
| Port | 443 |
| Path | /api/pedidos |
| Method | POST |

**Tabla 4.1:** Propiedades de la API para la creación de pedidos

La segunda tabla (Tabla 4.2) describe los tipos de datos requeridos por la API cuando se envían solicitudes de creación de pedidos a Salesforce. Especifica el tipo de dato esperado para cada campo, junto con la indicación de si su inclusión es obligatoria o no. Esto asegura que las solicitudes enviadas cumplan con los requisitos necesarios para una integración exitosa.

| Nombre del Campo | Tipo de Datos | Requerido |
|---|---|---|
| idPedido__c | String \| Null | Sí |
| cuenta__c | String \| Null | Sí |
| cantidad__c | String \| Null | Sí |
| estado__c | String \| Null | Sí |

**Tabla 4.2:** Tipos de datos requeridos para la creación de pedidos en Salesforce

La tercera tabla (Tabla 4.3) muestra cómo se mapean los campos entre diferentes APIs que intervienen en el proceso de creación de pedidos. Establece la correspondencia entre los campos utilizados por la API de la capa de experiencia, la API de gestión de pedidos y clientes en la capa de proceso, y la API que interactúa con Salesforce en la capa de sistema. Este mapeo es crucial para asegurar que los datos se transfieran correctamente entre las diferentes capas y sistemas.

| **Mapeo de Campos entre Diferentes APIs** | | |
|---|---|---|
| **e-web-api** | **p-gestion-pedidosClientes-api** | **s-pedidosClientes-api** |
| id | id | idPedido__c |
| financial_status | estado | estado__c |
| line_items | productos | |
| line_items.quantity | productos.cantidad | |
| line_items.description | productos.descripción | Descripción |
| customer.id | clienteId | cuenta__c |

**Tabla 4.3:** Mapeo de campos para la creación de pedidos

### Creación de clientes

Similar al proceso de creación de pedidos, la creación de clientes también comienza en la **capa de experiencia** (Figura 4.3). Cuando un nuevo usuario desea registrarse, la información es recopilada inicialmente y enviada por la misma API de la capa de experiencia. En la capa de proceso, se realiza una verificación para determinar si el cliente ya está registrado en el sistema, utilizando la `s-pedidosClientes-api`, que consulta nuestra base de datos en Salesforce. Si el cliente no existe, se procede a su registro, asegurando así que los datos del cliente se manejen con coherencia y eficacia.

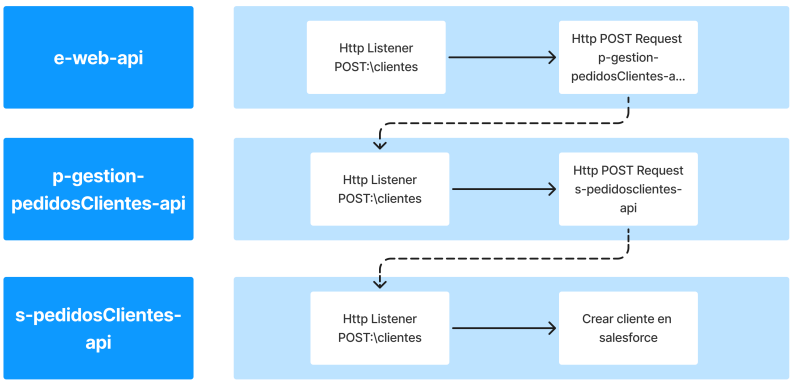


**Figura 4.3:** Diagrama de flujo del proceso de creación de clientes

A continuación se describen las tablas utilizadas para gestionar la creación de clientes: De la misma forma que en el apartado anterior (Tabla 4.1), en esta tabla (Tabla 4.4) se detallan las propiedades de la API utilizada para la creación de clientes. Se especifican el host, puerto, ruta y método HTTP, que configuran cómo se deben enviar las solicitudes al servidor.

| **Properties** | **PRE** |
|---|---|
| Host | e-web-api.us-e2.cloudhub.io |
| Port | 443 |
| Path | /api/clientes |
| Method | POST |

**Tabla 4.4:** Propiedades de la API para la creación de clientes

En cuanto a los tipos de datos requeridos, se definen en la tabla siguiente 4.5 para cada campo cuando se crea un cliente en Salesforce. Cada campo está clasificado por su tipo y si es mandatorio incluirlo en la solicitud.

| **Nombre del Campo** | **Tipo de Datos** | **Requerido** |
|---|---|---|
| name | String \| Null | Sí |
| phone | String \| Null | Sí |
| fax | String \| Null | No |
| billingStreet | String \| Null | No |

**Tabla 4.5:** Tipos de datos requeridos para la creación de clientes en Salesforce

La Tabla 4.6 ilustra cómo se mapean los campos de datos a través de las diferentes APIs involucradas en el proceso de creación de clientes. Este mapeo es vital para asegurar la integridad y coherencia de los datos transferidos entre las capas de la arquitectura.

| **Mapeo de Campos entre Diferentes APIs** | | |
|---|---|---|
| **e-web-api** | **p-gestion-pedidosClientes-api** | **s-pedidosClientes-api** |
| first_name | nombre | name |
| phone | phone | phone |
| fax | fax | fax |
| address1 | direccion | BillingStreet |

**Tabla 4.6:** Mapeo de campos para la creación de clientes

**Obtención y Gestión de Datos de Sensores**

Continuando con la automatización de procesos dentro de nuestra arquitectura, la gestión de la información recogida por los sensores ambientales es otro componente crítico. Este proceso comienza en la **capa de experiencia**, donde los datos de los sensores son capturados inicialmente. Estos datos son enviados a la capa de proceso mediante la `p-datos-sensores-api`, responsable de procesar y validar la información obtenida. La figura 4.4 ilustra este flujo de datos.

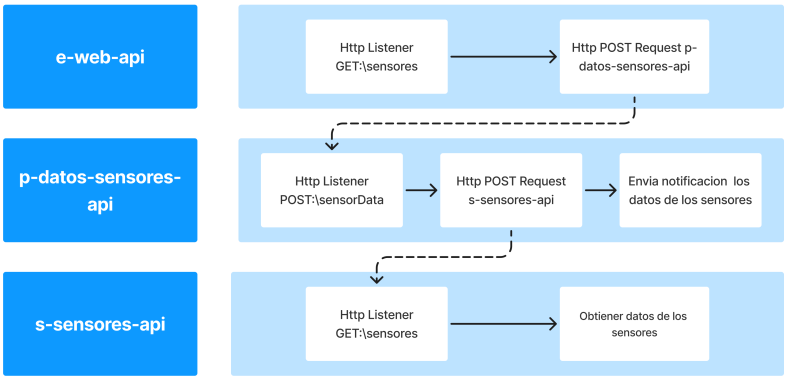


**Figura 4.4:** Diagrama de flujo de la gestión de datos de sensores

Esta API procesa la información y la transmite a la `s-sensores-api` en la capa de sistema, donde se almacenan y gestionan los datos. Esta información procesada puede activar alertas o notificaciones dependiendo de los parámetros operativos establecidos, como umbrales críticos que podrían impactar directamente la calidad de los productos almacenados.

Además, ofrecemos a nuestros clientes la capacidad de acceder directamente a estos datos a través de la API de experiencia. Mediante el endpoint `GET`, como se detalla en la Tabla 4.7 , los usuarios pueden solicitar información histórica o en tiempo real sobre las condiciones monitorizadas por los sensores. En la Tabla 4.8 , se observan los tipos de datos requeridos, de la misma forma que en los apartados anteriores.

| **Propiedad** | **Valor** |
|---|---|
| Host | api-ue.ict.net |
| Puerto | 443 |
| Ruta | /api/sensores |
| Método | GET |

**Tabla 4.7:** Propiedades de la API para la obtención de datos de los sensores

| **Nombre del Campo** | **Tipo de Datos** | **Requerido** |
|---|---|---|
| sensorId | String | Sí |
| fechaHora | DateTime | Sí |
| temperatura | Float \| Null | No |
| humedad | Float \| Null | No |
| luzSolar | Float \| Null | No |
| velocidadViento | Float \| Null | No |
| humedadSuelo | Float \| Null | No |
| pH | Float \| Null | No |
| nutrientes | String \| Null | No |
| estadoCrecimiento | String \| Null | No |
| detecciónPlagas | String \| Null | No |

**Tabla 4.8:** Tipos de datos requeridos para la obtención de datos de sensores

La recopilación de datos también implica un detallado mapeo entre las diferentes APIs, como se muestra en la Tabla 4.9, asegurando una transferencia y procesamiento eficientes de los datos sensoriales a través de las capas de nuestra arquitectura.

| **Mapeo de Campos entre Diferentes APIs** | | |
|---|---|---|
| **e-web-api-sensores** | **p-api-datos-sensores** | **s-api-sensores** |
| sensorId | sensor_id | id_sensor |
| fechaHora | timestamp | datetime |
| temperatura | temp | temperature |
| humedad | humidity | hum |
| luzSolar | solar_radiation | sunlight |
| velocidadViento | wind_speed | viento |
| humedadSuelo | soil_moisture | humedad_suelo |
| pH | soil_ph | ph |
| nutrientes | nutrients | nutrient_levels |
| estadoCrecimiento | growth_stage | etapa_crecimiento |
| detecciónPlagas | pest_detection | detección_plagas |

**Tabla 4.9:** Mapeo de campos para la transferencia de datos de sensores entre diferentes APIs

Este enfoque integrado no solo mejora la eficiencia operativa, sino que también asegura una respuesta proactiva a las condiciones cambiantes del ambiente, facilitando ajustes operativos y respuestas rápidas que son vitales en el contexto agrícola.

Estos diagramas no solo ilustran el flujo operativo de cada proceso, sino que también reflejan la efectividad del modelo canónico de datos para asegurar que las transformaciones entre las capas se realicen sin errores, preservando la integridad y la precisión de los datos a lo largo de toda la arquitectura API-led. Además, este diseño no solo facilita la comprensión del flujo operativo y la funcionalidad de cada proceso, sino que también demuestra la eficacia del modelo canónico de datos en preservar la precisión y coherencia de la información a lo largo del sistema.

# CAPÍTULO 5
# Desarrollo de la solución propuesta

## 5.1 Tecnologías Utilizadas

En este proyecto se han empleado diversas tecnologías avanzadas, cada una cuidadosamente seleccionada por su capacidad para contribuir de manera específica a diferentes aspectos de la solución global. A continuación, se ofrece una descripción detallada de cada tecnología y su papel en el desarrollo del sistema:

### MuleSoft

MuleSoft es una plataforma líder en la integración de aplicaciones, datos y dispositivos, tanto en entornos locales como en la nube. Esta plataforma permite a las organizaciones conectar sistemas dispares de manera eficiente, adoptando un enfoque orientado a servicios (SOA). Utiliza un modelo basado en APIs para facilitar la conexión entre aplicaciones, datos y dispositivos, lo que ayuda a las empresas a ser más ágiles, mejorar la eficiencia operativa y ofrecer experiencias digitales transformadoras [8].

### Anypoint Platform de MuleSoft

Anypoint Platform es una solución integral proporcionada por MuleSoft que simplifica el diseño, despliegue y gestión de APIs. Esta plataforma ofrece diversas herramientas que cubren todo el ciclo de vida del desarrollo de APIs, incluyendo:

- **Anypoint Exchange:** Un repositorio centralizado donde se pueden almacenar y compartir recursos como APIs, conectores y plantillas. Esto facilita la reutilización y la colaboración entre desarrolladores, permitiendo un desarrollo más ágil y consistente [9].
- **API Manager:** Esta herramienta permite gestionar el ciclo de vida de las APIs, abordando aspectos como la seguridad, gobernanza, documentación y análisis de uso. API Manager asegura que las APIs estén seguras y se utilicen eficazmente, permitiendo a las organizaciones cumplir con las normativas y proteger los datos sensibles [10].
- **Runtime Manager:** Utilizado para desplegar y gestionar aplicaciones Mule, APIs y microservicios en cualquier entorno, incluyendo CloudHub y entornos locales. Proporciona flexibilidad operativa y escalabilidad, lo que permite a las empresas manejar aplicaciones complejas en múltiples entornos sin comprometer el rendimiento o la seguridad [11].

- **Design Center:** Ofrece herramientas para el diseño y prototipado rápido de APIs, incluyendo funcionalidades para diseñar flujos de integración que automatizan procesos de negocio de manera eficiente. Esto permite a los desarrolladores crear y probar APIs de forma ágil, asegurando que las soluciones sean robustas desde las primeras etapas del desarrollo [12].

### RAML (RESTful API Modeling Language)

RAML es un lenguaje de modelado utilizado para diseñar y documentar APIs REST de una manera legible tanto para humanos como para máquinas. Permite a los desarrolladores describir claramente la estructura de las APIs, incluyendo recursos, métodos, parámetros, respuestas y otros datos relevantes. Este enfoque facilita el entendimiento y la colaboración entre equipos técnicos y no técnicos, promoviendo una mayor coherencia y calidad en el desarrollo de APIs [13].

### Salesforce

Salesforce es una plataforma integral de gestión de relaciones con clientes (CRM) que proporciona herramientas para el manejo de ventas, servicio al cliente, marketing y más. En este proyecto, Salesforce se utiliza para gestionar la información de clientes y pedidos. La plataforma ofrece funcionalidades extensas de personalización y automatización, permitiendo adaptar el sistema a las necesidades específicas del proyecto y mejorar la eficiencia operativa. Gracias a su capacidad para integrarse con otras plataformas y su enfoque en la nube, Salesforce se ha convertido en una herramienta esencial para muchas organizaciones en la gestión de relaciones con clientes [14].

### CloudHub

CloudHub es una plataforma como servicio (PaaS) ofrecida por MuleSoft que permite desplegar y administrar aplicaciones de integración y APIs directamente en la nube. Esta plataforma facilita la gestión de cargas de trabajo, escalabilidad automática y supervisión en tiempo real, lo que permite a las empresas concentrarse en la innovación y la optimización de sus procesos sin preocuparse por la infraestructura subyacente. CloudHub es ideal para organizaciones que buscan soluciones flexibles y escalables que puedan crecer con sus necesidades [15].

### DataWeave

DataWeave es el lenguaje de programación de transformación de datos de MuleSoft, diseñado específicamente para la manipulación y la transformación de datos. Permite a los usuarios transformar datos entre múltiples formatos (como JSON, XML y CSV) de forma eficiente, con un manejo sofisticado de la lógica de datos. Esto es esencial para procesar las interacciones y las transacciones que fluyen a través de las APIs, asegurando que los datos se manipulen de manera coherente y eficiente a lo largo de todo el sistema [16].

### RabbitMQ

RabbitMQ es un software de intermediación de mensajes que implementa el protocolo AMQP (Advanced Message Queuing Protocol). Facilita la comunicación asincrónica y robusta entre componentes de software distribuidos, lo que ayuda a desacoplar la arquitectura de la aplicación y mejora la escalabilidad y la flexibilidad del sistema. RabbitMQ es especialmente útil en entornos donde la fiabilidad y la capacidad de gestionar grandes volúmenes de mensajes son cruciales para el rendimiento de la aplicación [17].

### AMQP (Advanced Message Queuing Protocol)

AMQP es un protocolo abierto estándar para la intermediación de mensajes, que permite la comunicación entre diferentes aplicaciones mediante la entrega de mensajes de manera confiable y segura. En este proyecto, AMQP es fundamental para asegurar la integridad de la comunicación entre los diferentes servicios y componentes del sistema, garantizando que los mensajes sean entregados de manera efectiva y que las aplicaciones puedan funcionar de manera desacoplada y escalable [18].

### Shopify

Shopify es una plataforma de comercio electrónico que permite a los comerciantes configurar y gestionar tiendas en línea y puntos de venta físicos. En este proyecto, Shopify se utiliza para gestionar todas las interacciones de la tienda en línea, incluyendo la presentación de productos, la gestión de inventarios, la tramitación de pedidos y el procesamiento de pagos. Shopify se integra con el sistema de gestión de pedidos mediante APIs, lo que permite una gestión fluida y eficiente de las operaciones comerciales [19].

Estas tecnologías se combinan para crear un sistema que no solo es técnicamente robusto y escalable, sino también eficiente y adaptado a las necesidades específicas del manejo de pedidos y la gestión de clientes en un entorno dinámico de comercio electrónico. La integración de estas herramientas asegura que cada componente del sistema funcione de manera armoniosa, permitiendo a la organización responder de manera ágil a las necesidades del mercado y optimizar sus operaciones de manera continua.

## 5.2 Uso de RAML para la Estructuración de APIs

La definición de las APIs se realiza a través de RAML. El proceso de desarrollo de la solución se inicia con la creación de un archivo RAML que no solo define los métodos HTTP necesarios y los tipos de datos de las solicitudes y respuestas esperadas, sino que también establece todos los endpoints necesarios para la funcionalidad de la API.

La figura 5.1 muestra la sección inicial de este archivo RAML, donde se declara el título de la API, la versión, y la URI base, junto con el tipo de contenido media que se manejará, que en este caso es JSON. Este encabezado del archivo RAML es crucial pues establece el contexto y los estándares básicos para las interacciones que se definirán más adelante.


**Figura 5.1:** Inicio de la definición del archivo RAML.

La figura 5.2 detalla cómo se define un endpoint específico dentro del mismo archivo RAML. En esta sección, se especifica el endpoint para crear un nuevo pedido, incluyendo una descripción del mismo, el tipo de solicitud esperada, y las posibles respuestas. Por ejemplo, se muestra cómo responder a una creación exitosa de un pedido con un código 201, y cómo manejar errores con un código 400, proporcionando detalles específicos sobre el tipo de error.

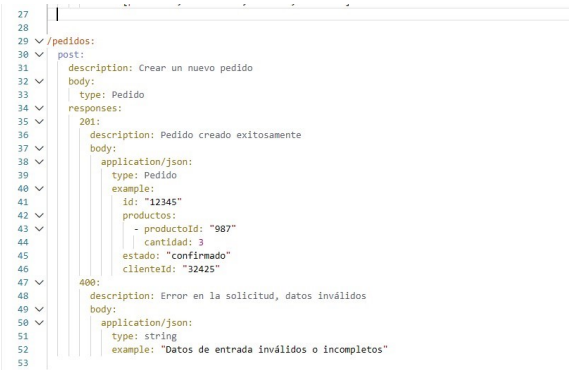


**Figura 5.2:** Especificación de un endpoint en el archivo RAML para crear un nuevo pedido.

Una vez importado el RAML en Anypoint Studio, se genera un esqueleto (Figura 5.3) que incluye flujos para cada endpoint definido en el RAML, facilitando la organización y el manejo específico de cada operación de la API. Dentro de este esqueleto, hay dos flujos destacados que cumplen funciones específicas: `e-web-api-main` y `e-web-api-console`.

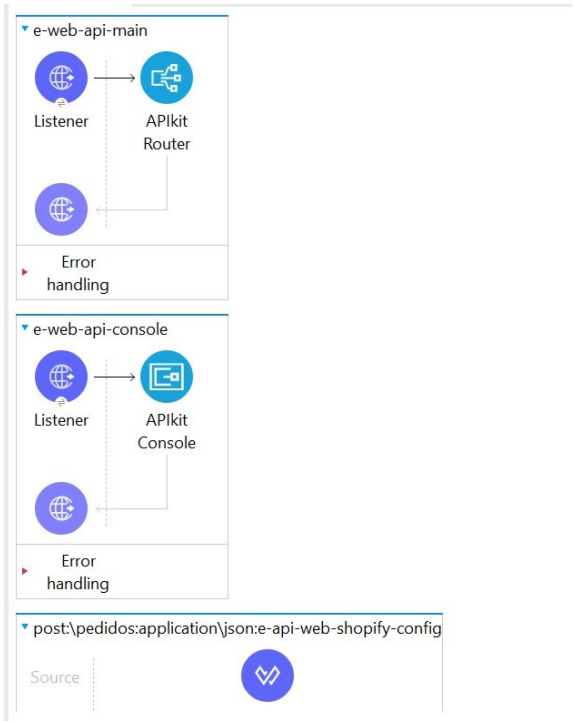


**Figura 5.3:** Esqueleto de la API generado en Anypoint Studio.

El flujo `e-web-api-main` actúa como el enrutador principal de la API. Emplea un Listener para captar solicitudes entrantes y las dirige hacia el APIkit Router. Este router encamina cada solicitud al flujo correspondiente basado en la URI y el método HTTP, garantizando un manejo adecuado de las operaciones. También incluye una gestión de errores para manejar excepciones y fallos, aumentando la robustez y fiabilidad de la API.

Por otro lado, el flujo `e-web-api-console` proporciona una consola interactiva accesible a través de un navegador web (Figura 5.4), que permite a los usuarios interactuar directamente con la API.

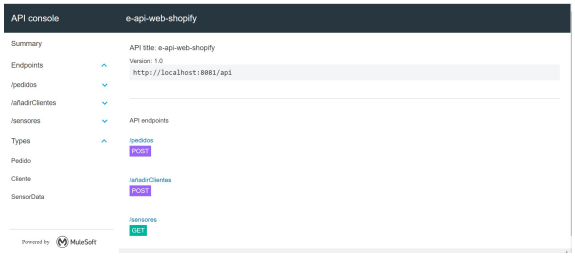


**Figura 5.4:** Consola interactiva accesible a través de un navegador.

El uso de RAML es fundamental no solo por su capacidad para estructurar de manera lógica y organizada el diseño de la API, sino también por asegurar que todas las APIs desarrolladas sigan un estándar y mantengan una documentación consistente, lo que facilita integraciones más fluidas y reduce la probabilidad de errores durante la fase de interconexión con otros sistemas o componentes del proyecto.

## 5.3 Desarrollo de las APIS

### 5.3.1. Creación de Pedidos

Cuando un cliente realiza un pedido a través de la tienda Shopify, se desencadena un proceso automatizado en nuestra API. La secuencia inicia en el flujo de la capa de experiencia de Mule, específicamente con el `post_pedidos` mostrado en la Figura 5.5. En este punto, el conector de Shopify recibe una notificación de un nuevo pedido. Acto seguido, se procede al mapeo de los datos del pedido a través de un flujo de referencia (`Add_Order_Subflow`), que se utiliza para modularizar el flujo y hacerlo más manejable al dividirlo en sub-flujos más pequeños y específicos. En esta etapa, se transforman los datos para ser enviados a través de una solicitud `POST` a la API de proceso (Figura 5.6). Este procedimiento se sigue en otros casos de uso para la comunicación entre la API de experiencia y la API de proceso.

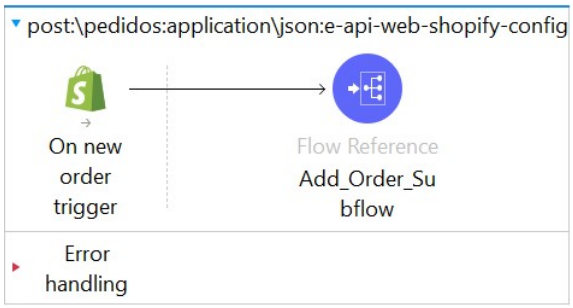


**Figura 5.5:** Inicio del flujo crear pedidos

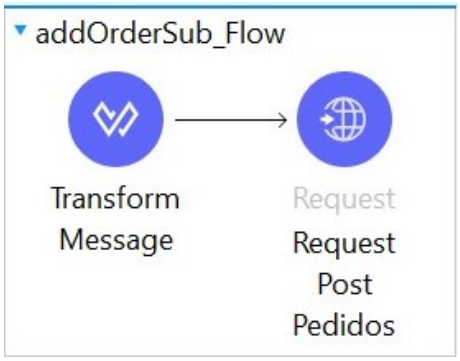


**Figura 5.6:** Flujo que hace para añadir pedido

Una vez llegan los datos a la api de proceso inica el sub flujo (`gestionarPedidoSubFlow`) (Figura 5.7), dentro del mismo se puede obsrva que el primer conector es un flow refe-recnce que lleva al sub flujo implementationSub_flow (Figura 5.8 ) en el que se establece la variable Productos, almacenando el array de productos recibidos. Posteriormente, se inicia un bucle For Each que recorre cada producto individualmente, tratando cada uno de manera aislada. Para cada producto, se crea una variable local que almacena informa-ción específica del producto en tratamiento. Esta información se utiliza para realizar una solicitud a la API de inventario, enviando el ID del producto como parámetro URI para obtener la cantidad disponible del producto específico.

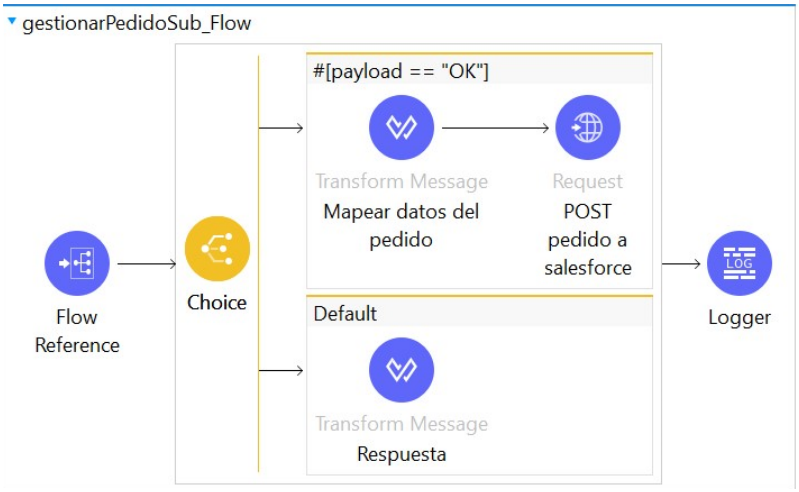


**Figura 5.7:** Transformación de datos y envio a siguiente capa

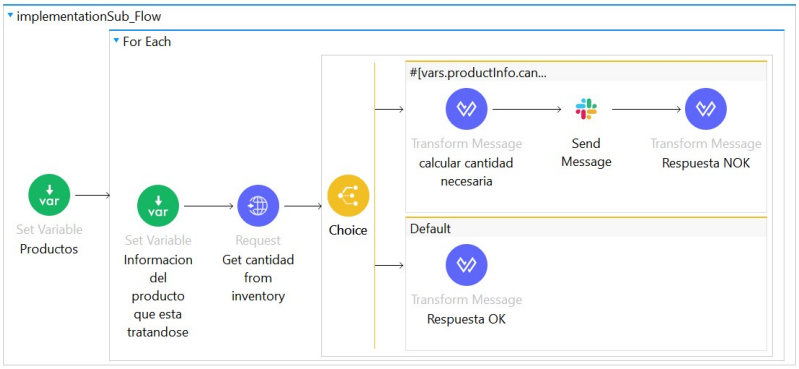


**Figura 5.8:** Transformación de datos y envio a siguiente capa

En la API de inventario (Figura 5.9), se realiza una consulta a Salesforce para obtener la cantidad del producto por su ID. Una vez obtenida, se mapea para devolver el nombre, la cantidad y el ID del producto.

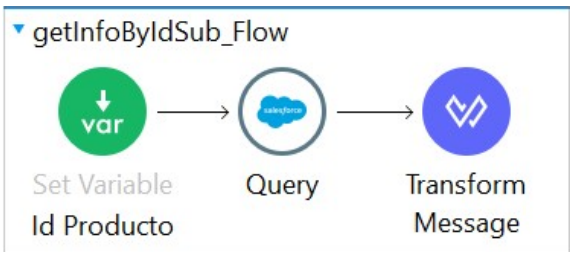


**Figura 5.9:** Flujo obtener cantidad de un producto de salesforce

Una vez obtenida la cantidad del producto, volvemos al flujo implementationSub_flow (Figura 5.8) y se accede al componente Choice que evalúa si la cantidad solicitada excede la cantidad en stock. Si la cantidad disponible es menor que la solicitada, se prepara un mensaje en el componente Transform Message, que posteriormente se envía al almacén a través de la aplicación de mensajería Slack, notificando sobre la insuficiencia del stock y posteriormente establece en el payload el valor "NOK".

Si la cantidad en stock es suficiente, el flujo establece en el payload el valor .ºK", indicando que el producto puede ser satisfactoriamente agregado al pedido. Este resultado se propaga de vuelta al flujo principal a través del `Flow Reference`.

Al regresar al flujo principal(gestionarPedidoSub_Flow), se realiza una verificación final del estado del pedido con otro componente Choice. Si el resultado es"Not OK", se envía un mensaje al cliente indicando que su pedido estará disponible pronto. Si el resultado es "OK", se procede a mapear los datos del pedido en un componente Transform Message y se realiza una solicitud POST a la API de Salesforce para registrar el pedido en el sistema.

En esta API (Figura 5.10), se mapean los datos para que coincidan con los campos definidos en Salesforce y se realiza la operación de creación de registros en el objeto de pedidos de Salesforce.

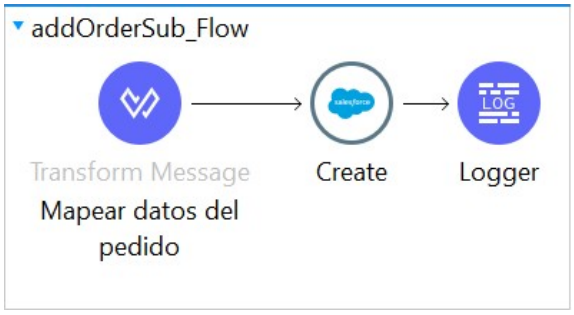


**Figura 5.10:** Flujo añadir pedido a saleforce

### 5.3.2. Creación De clientes

El proceso de registro de clientes en nuestro sistema se inicia con una notificación de Shopify que informa sobre un intento de registro por parte de un cliente. Este evento activa el flujo en la capa de experiencia, específicamente en el endpoint `post ñadirClientes` (Figura 5.11), configurado para captar estas notificaciones.

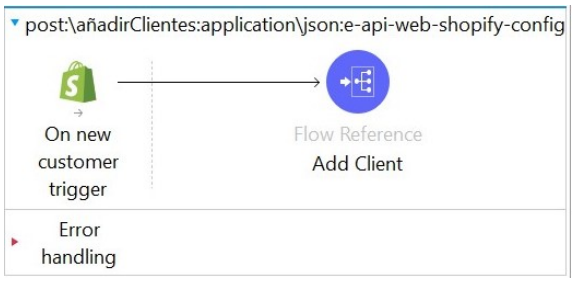


**Figura 5.11:** Flujo añadir Cliente

Una vez recibida la notificación, el flujo procede a ejecutar el sub-flujo `registrarClienteSub_Flow`, que está dedicado específicamente al manejo de la lógica para añadir nuevos clientes (Figura 5.12). Inicialmente, se utiliza un componente de `Transform Message` para mapear los datos del cliente recibidos de Shopify a un formato compatible con nuestra API de procesos, adaptando campos como el nombre, detalles de contacto y otros datos esenciales. Además, se crea una variable con los datos del cliente para preservar la información, ya que después de realizar la solicitud a la API de sistemas, el payload original se sobrescribe. En esta misma solicitud, se realiza un GET a la API de Salesforce utilizando el ID del cliente y el nombre para comprobar si ya existe.

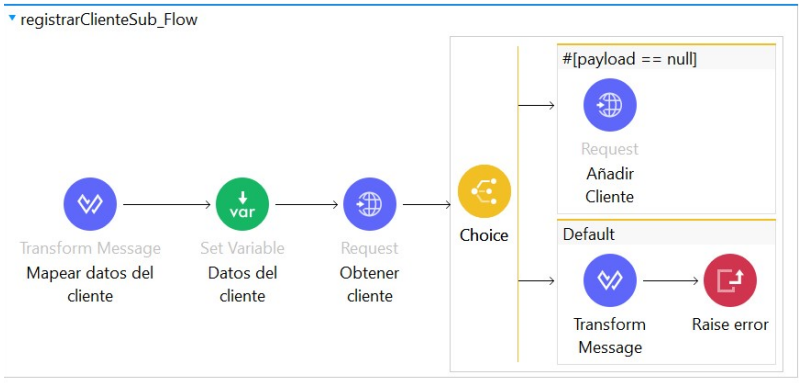


**Figura 5.12:** Flujo aregistrarClienteSubFlow

Una vez en la API de sistemas de Salesforce (5.13), este flujo realiza una consulta a Salesforce y mapea la respuesta. Tras esto, se vuelve al flujo principal (`registrarCliente_Flow`) y si la respuesta devuelta por la API de sistemas es null, esto indica que el cliente no existe y, por lo tanto, se procede a crearlo realizando un POST a la API de Salesforce y en caso contrario provoca un error.

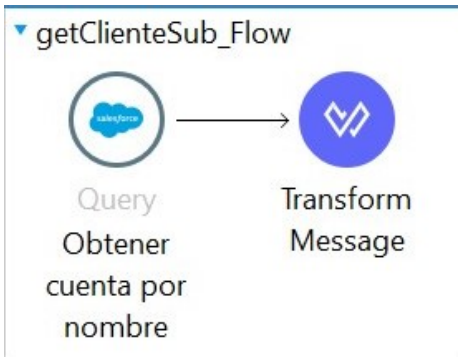


**Figura 5.13:** Obtener Cliente a Salesforce

En la API de sistemas que gestiona clientes y pedidos de Salesforce, se mapean los datos para que coincidan con los de Salesforce y así poder crear el registro. Finalmente, se utiliza el conector `Create` de Salesforce, referenciando al objeto Clientes para registrar al mismo (Figura 5.14).

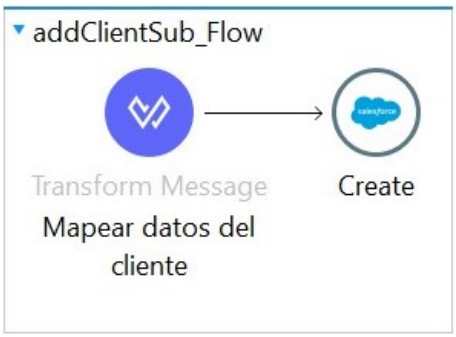


**Figura 5.14:** Añadir Cliente a Salesforce

Estos flujos aseguran que el registro de clientes se maneje de manera eficiente, verificando primero la existencia del cliente para evitar duplicados y utilizando la capacidad de Salesforce para gestionar la información de clientes de manera centralizada.

### 5.3.3. Obtener datos de sensores

En el marco de nuestra solución, los sensores juegan un papel vital al monitorear continuamente variables ambientales y enviar esta información a una cola de mensajes administrada por RabbitMQ, utilizando el conector AMQP. Este enfoque asincrónico garantiza la eficiencia y la confiabilidad en la entrega y manejo de los datos recolectados.

La configuración de RabbitMQ es central para garantizar la alta disponibilidad y la resistencia de los datos en tránsito. Se han implementado diversas colas y intercambios para manejar eficientemente el flujo de mensajes desde múltiples fuentes sensoriales, asegurando que los mensajes no se pierdan incluso en casos de picos de datos o fallos temporales. Cada cola se configura con políticas de reintento y expiración de mensajes, lo que permite un control detallado sobre cómo y cuándo se manejan los mensajes.

En particular, la cola denominada "sensores"(Figura 5.15) está diseñada para recibir y mantener los mensajes hasta que sean procesados por el componente consumidor adecuado. Esta cola es fundamental para asegurar que no haya pérdida de datos y que los mensajes se manejen eficientemente, actuando como un buffer entre la generación de datos y su procesamiento.

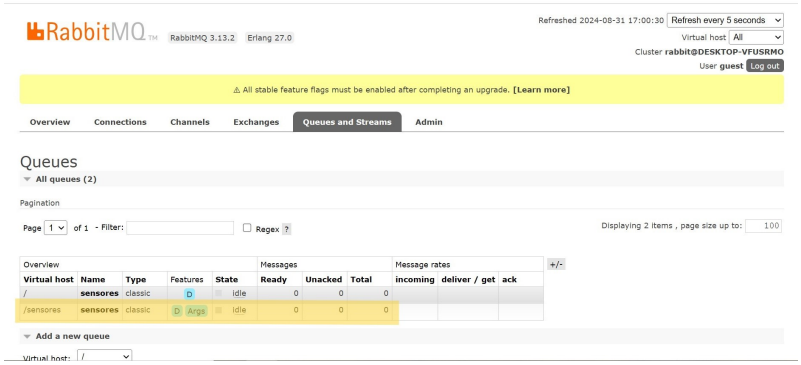


**Figura 5.15:** Imahe de la cola sensores

Una vez que los datos son emitidos por los sensores, son capturados por el Listener AMQP de nuestra API de sistemas (Figura 5.16), que actúa como un puente entre los sensores y el procesamiento interno. Este conector facilita la recopilación eficiente de las lecturas, encaminando la información hacia procesos subsecuentes sin retrasos significativos.


**Figura 5.16:** Flujo que espera a que se envie un mensaje a la cola

Tras la recepción de los datos, la API de sistemas procesa y envía esta información a la capa de proceso(Figura 5.17). Aquí, el flujo se encarga de transformar y encaminar estos datos hacia un canal de alertas en Slack, equipando al equipo con datos críticos como el identificador del sensor y la marca temporal de las mediciones, lo que permite un seguimiento preciso y en tiempo real de las condiciones ambientales.

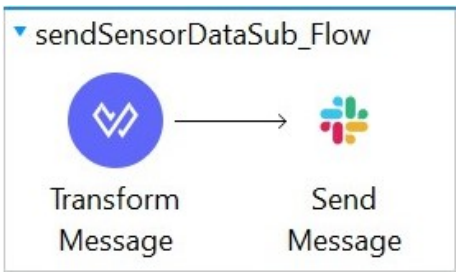


**Figura 5.17:** Flujo para enviar mensaje por slack

Además, se ofrece a los clientes la posibilidad de consultar directamente estos datos a través de la API de experiencia. Utilizando el endpoint `get` (Figura 5.18), los usuarios pueden solicitar información histórica o en tiempo real de los sensores, facilitando una interacción activa y directa con los datos recopilados (5.18). Esta capacidad interactiva no solo mejora la experiencia del usuario sino que también potencia la utilidad práctica de la información recogida, permitiendo ajustes operativos y respuestas rápidas a las condiciones cambiantes.

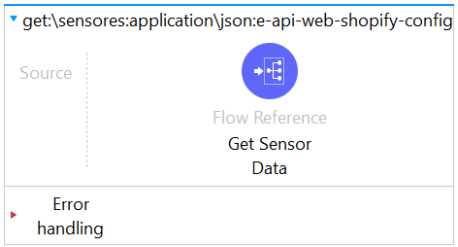


**Figura 5.18:** Flujo para iniciar proceso obtener datos sensores

Esta integración de RabbitMQ no solo optimiza el flujo de trabajo de datos, sino que también subraya nuestro compromiso con una infraestructura robusta y escalable, capaz de soportar las demandas de un entorno dinámico y cargado de datos.

### 5.3.4. Módulo de IA para Análisis Predictivo

Dentro del marco del proyecto, se contempló la integración de una solución de inteligencia artificial (IA) enfocada en el análisis predictivo, aunque este componente no se llevó a cabo por falta de recursos y conocimientos especializados. La idea fundamental detrás de esta iniciativa era utilizar algoritmos de aprendizaje automático para analizar los datos históricos de los pedidos y realizar predicciones precisas sobre futuras demandas de productos.

La implementación propuesta de la IA habría requerido la recopilación y el análisis de grandes volúmenes de datos de ventas, producciones pasadas y variables contextuales como el clima y las condiciones del mercado. Se habrían utilizado técnicas de minería de datos para identificar patrones y tendencias que influyen en la demanda de productos.

Los modelos de aprendizaje automático, como las redes neuronales o los modelos de regresión, se habrían entrenado con estos datos para prever las demandas futuras. Estas predicciones habrían sido integradas en un sistema dinámico de gestión de recursos, ajustando automáticamente las órdenes de producción y las asignaciones de recursos en tiempo real.

El principal desafío para implementar esta solución habría sido la calidad y la cantidad de los datos disponibles. Para que los modelos predictivos funcionen eficazmente, necesitan datos históricos extensos y precisos. Además, la integración de la IA en los procesos existentes habría requerido una transformación digital significativa y la adopción de una cultura organizativa que abrace la innovación tecnológica y el cambio.

En conclusión, aunque la implementación de la IA no se materializó en el marco temporal del proyecto, representa una dirección estratégica valiosa para el futuro. Explorar esta tecnología podría ofrecer ventajas competitivas sustanciales, permitiendo a la empresa responder mejor a las dinámicas del mercado y optimizar su operativa interna.

# CAPÍTULO 6
# Implantación y pruebas

## 6.1 Implantación de la Configuración de Conexiones

Para integrar y gestionar las conexiones con las diferentes tecnologías dentro de nuestra aplicación en Mule, utilizamos un archivo de propiedades global denominado `global.properties`. Este archivo contiene todas las credenciales y parámetros necesarios para establecer conexiones seguras con servicios externos, como Shopify, Salesforce, RabbitMQ y Slack. Mantener estas credenciales fuera del código fuente no solo mejora la seguridad, sino que también facilita la gestión centralizada y la actualización de las mismas sin necesidad de modificar el código de la aplicación.

**Archivo de Propiedades Global**

A continuación, se muestra un ejemplo del contenido del archivo `global.properties`, donde se incluyen las configuraciones específicas para Shopify:

```
shopify:
  username: "credenciales"
  password: "credenciales"
  baseUri: "https://XXXXX.myshopify.com"
```

Este archivo también puede incluir configuraciones para otros servicios, como Salesforce, AMQP y Slack, de la siguiente manera:

```
salesforce:
  username: "credenciales"
  password: "credenciales"
  securityToken: "token"


AMQP:
  username: "credenciales"
  password: "credenciales"
  host: "hostname"

slack:
  token: "credenciales"
  channel: "nombre-del-canal"
  baseUri: "https://login.slack.com"
```

En Anypoint Studio, estas propiedades se referencian directamente en la configuración de los conectores correspondientes. Por ejemplo, la configuración del conector Shopify podría verse de la siguiente manera (ver Figura 6.1), donde se utilizan las propiedades definidas para autenticar y establecer la conexión:

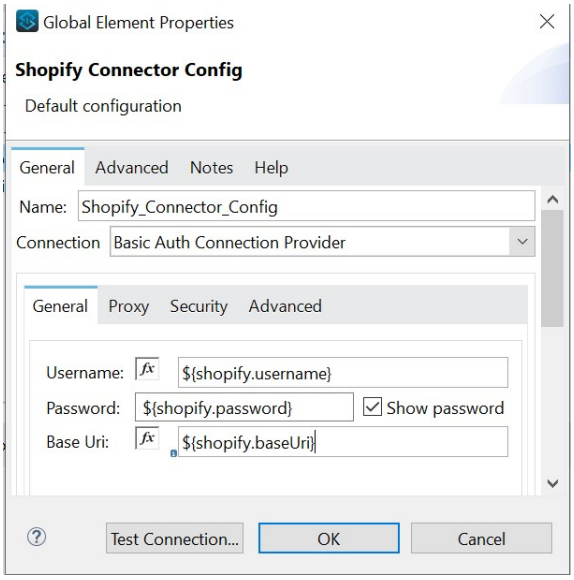


**Figura 6.1:** Configuración del Conector de Shopify en Anypoint Studio

Esta forma de manejar las configuraciones garantiza que cualquier actualización de las credenciales pueda realizarse en un solo lugar sin necesidad de tocar el código base, aumentando así la seguridad y simplificando el mantenimiento del sistema.

De manera similar, la configuración de otros conectores (como Salesforce, AMQP y Slack) se realiza utilizando las propiedades definidas en `global.properties`. Esto permite una gestión centralizada y consistente de todas las conexiones externas que la aplicación Mule necesita para funcionar correctamente.

## 6.2 Despliegue de la Solución

Durante la fase de desarrollo, todas las pruebas y validaciones se realizaron de manera local, asegurando un control detallado y una respuesta rápida ante cualquier incidencia o necesidad de ajuste. Esto permitió un desarrollo ágil y eficiente, facilitando iteraciones rápidas y un entendimiento profundo de la funcionalidad de cada componente de la solución.

Posteriormente, la solución fue desplegada en el entorno **sandbox** de **CloudHub**. Este entorno sandbox permite realizar pruebas en un entorno de producción simulado, asegurando que todas las integraciones funcionen correctamente antes de un despliegue en producción. El despliegue se realizó utilizando un solo *worker*, lo que significa que se asignó un solo núcleo de procesamiento dedicado a la aplicación. Este enfoque simplificado fue suficiente para gestionar la carga de trabajo actual, manteniendo un balance entre costos y rendimiento.

El entorno de **CloudHub** fue configurado para utilizar la versión más reciente del runtime de **Mule**, específicamente la versión 4.7.2, que ofrece las últimas funcionalidades y mejoras en seguridad y rendimiento. La elección de esta versión del runtime garantiza compatibilidad con las más recientes mejoras de la plataforma y una mayor eficiencia en la ejecución de los flujos de integración.

Además, se optó por configuraciones estándar sin necesidad de IPs estáticas, dado que la arquitectura diseñada no requiere de conexiones entrantes que justifiquen esta configuración. La gestión y monitorización de la aplicación se realizó a través de la consola de **Runtime Manager**, la cual permite visualizar en tiempo real los logs generados

por la aplicación. Esta característica es esencial para realizar un seguimiento detallado y un diagnóstico rápido en caso de incidencias.

En la Figura 6.2, se muestra el estado del despliegue en el entorno **sandbox** de **CloudHub**, donde se puede ver que la aplicación se encuentra en estado "Started", indicando que el despliegue fue exitoso. Además, en la Figura 6.3 , se presenta la consola de logs, la cual permite revisar detalladamente cada operación realizada por el sistema, facilitando la detección y resolución de posibles problemas durante la ejecución.

Este despliegue en **CloudHub** no solo proporciona una alta disponibilidad y escalabilidad, sino que también permite una gestión y monitorización centralizada, facilitando la operación y el mantenimiento continuo de la aplicación sin necesidad de infraestructura física propia.

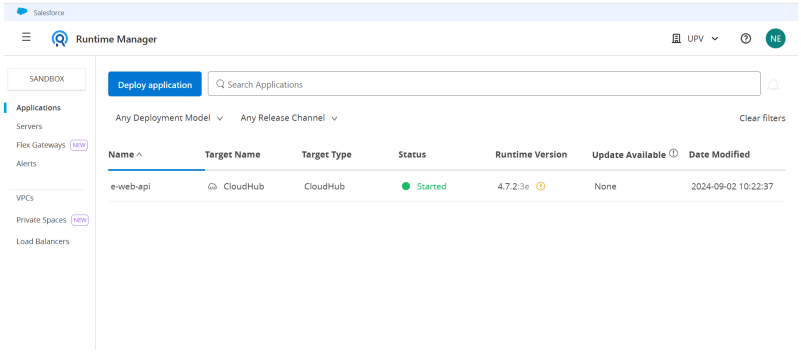


**Figura 6.2:** Despliegue en CloudHub sandbox mostrando el estado de la aplicación.

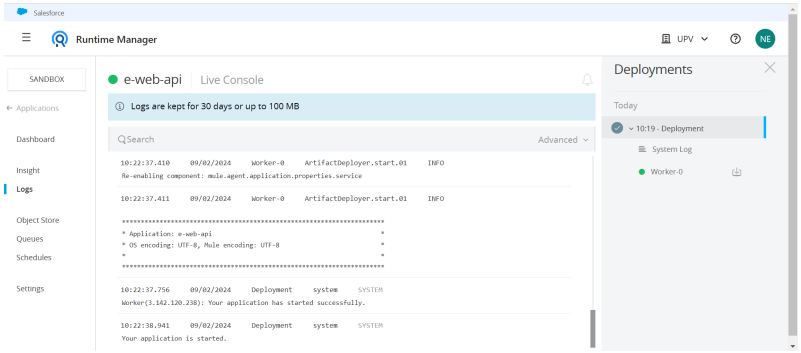


**Figura 6.3:** Consola de logs en CloudHub para monitorización en tiempo real.

## 6.3 Pruebas

Para realizar las pruebas de funcionamiento entre las APIs y entre las APIs y sistemas externos, inicialmente se configuró el entorno utilizando `localhost*`. Este enfoque permitió una verificación y ajuste detallados del comportamiento de las APIs en un entorno de desarrollo controlado, empleando Anypoint Studio. Posteriormente, para emular un entorno más próximo a la producción y evaluar la interacción entre componentes bajo condiciones operativas reales, los procesos se desplegaron en CloudHub, utilizando un solo worker para gestionar todas las solicitudes.

Los procesos de prueba para la creación de clientes y pedidos se inician cuando un usuario realiza una acción en Shopify, ya sea registrándose como nuevo cliente o realizando un pedido. Se realiza una verificación meticulosa en cada etapa de estos flujos para asegurar que los datos se procesen y almacenen correctamente.

Para la validación de estos procesos, se utilizó Postman para enviar solicitudes POST a las APIs desplegadas en CloudHub. Se observaron las respuestas a estas solicitudes para confirmar la correcta ejecución de las acciones y el adecuado registro de los datos en Salesforce. Las capturas de pantalla a continuación muestran las solicitudes POST realizadas con éxito en Postman, donde se pueden apreciar las estructuras de los cuerpos de las solicitudes y las respuestas recibidas. Estas respuestas indican que tanto la creación de clientes como la de pedidos se han completado satisfactoriamente, con un código de estado HTTP 201 que confirma la creación exitosa de los recursos (Figuras 6.4 y 6.5).

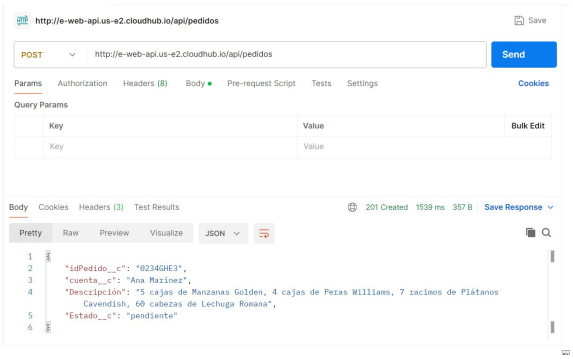


**Figura 6.4:** Ejemplo de solicitud POST utilizando Postman para probar la adición de pedidos.

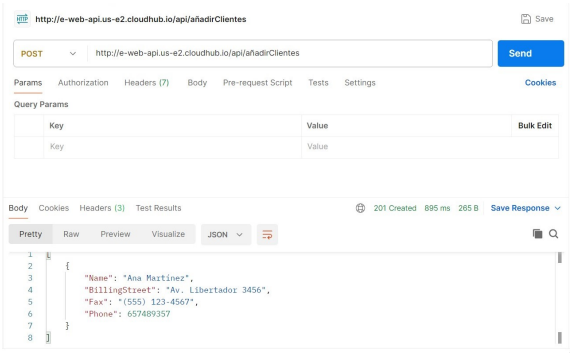


**Figura 6.5:** Ejemplo de solicitud POST utilizando Postman para probar la creación de clientes.

[0]*El término `localhost` hace referencia a la dirección IP local del equipo de desarrollo, usualmente `127.0.0.1`. Este ambiente es ampliamente utilizado por los desarrolladores para realizar pruebas de aplicaciones web y servicios de forma aislada, sin necesidad de conectividad con redes externas, facilitando la detección y corrección de errores en etapas tempranas del desarrollo.

Estas imágenes confirman el funcionamiento correcto del sistema al interactuar con Shopify y Salesforce (Figuras 6.6 y 6.7). Se demuestra que tanto el cliente como el pedido han sido creados con éxito en Salesforce, lo que evidencia la eficacia del flujo de datos a través de las capas de la arquitectura y valida la implementación en CloudHub.

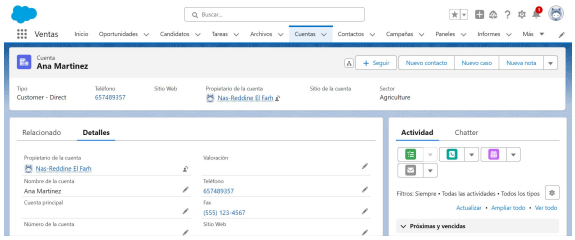


**Figura 6.6:** Ejemplo de cliente añadido en Salesforce.

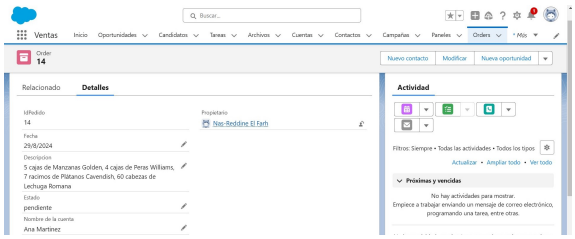


**Figura 6.7:** Ejemplo de pedido añadido en Salesforce.

Para probar el sistema de sensores en nuestro entorno, se estableció un mecanismo mediante el cual los datos recolectados por sensores ambientales se envían continuamente a una cola de RabbitMQ, independientemente de las interacciones del cliente. Esta implementación asegura la recopilación y transmisión automatizada de datos hacia el canal de alertas en Slack, facilitando la monitorización constante de variables críticas.

La prueba específica se inicia cuando un cliente realiza una solicitud GET a través de nuestra API e-web-api-us-e2.cloudhub.io/api/sensores. Esta acción dispara el envío de datos recientes, que han sido acumulados a través de RabbitMQ, hacia el canal de alertas de Slack. Además de las alertas automáticas configuradas para dispararse por eventos o umbrales predefinidos, esta solicitud permite la verificación manual y en tiempo real del estado actual y el funcionamiento del sistema de sensores .

La siguiente figura (Figura 6.8) ilustra la respuesta obtenida tras realizar la solicitud GET mencionada, mostrando un mensaje de éxito y orientando al usuario hacia el canal de alertas en Slack, donde se pueden visualizar las alertas generadas:

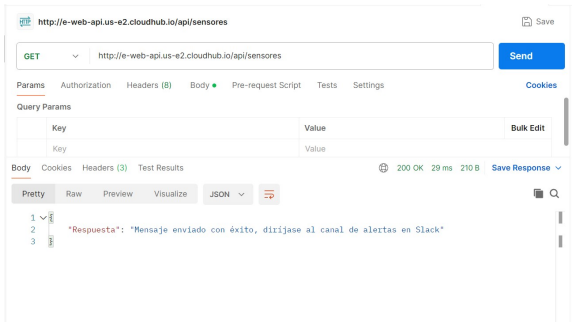


**Figura 6.8:** Respuesta de Postman al al solicitar datos de sensores

Esta configuración no solo prueba la efectividad del flujo de datos desde los sensores hasta las interfaces de usuario final, sino que también demuestra la robustez del sistema en la gestión de alertas en tiempo real y la integración con herramientas de comunicación como Slack. En la figura 6.9 se observa directamente el mensaje en el canal de alertas de Slack, confirmando la recepción y visualización de los datos por parte de los equipos pertinentes.

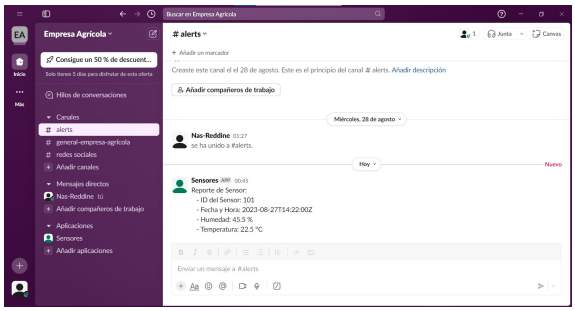


**Figura 6.9:** Respuesta de la API al solicitar datos de sensores

# CAPÍTULO 7
# Conclusiones

Después de haber completado mi Trabajo de Fin de Grado, enfocado en la automatización de la cadena de suministro en el sector alimenticio mediante la integración de sistemas, he llegado a varias conclusiones que me gustaría compartir.

Primero, me ha quedado claro lo importante que es experimentar directamente con tecnologías avanzadas. Durante el proyecto, tuve la oportunidad de profundizar en cómo las nuevas tendencias en tecnología de la información, especialmente en integración de aplicaciones, pueden realmente marcar la diferencia en la eficiencia y funcionalidad de los sistemas empresariales. La integración es más que solo conectar aplicaciones; se trata de mejorar la coordinación entre ellas para que todo funcione de manera más fluida y efectiva.

El proceso comenzó con un análisis teórico para entender las herramientas y tecnologías disponibles. Esta fase fue esencial porque, sin una buena base, sería imposible diseñar algo que realmente funcione. Después de entender bien qué necesitaba el sector alimenticio, pasé a diseñar la arquitectura del sistema. Este diseño tenía que ser flexible y escalable, de modo que no solo resolviera los problemas actuales, sino que también pudiera adaptarse a futuras necesidades.

Cuando llegó el momento de implementar, usé herramientas como MuleSoft y Salesforce para poner en práctica todo lo que había planeado. La verdad es que no fue un camino fácil; hubo varios desafíos técnicos, pero gracias a una buena planificación y a la capacidad de adaptarme a los problemas, logré que todo funcionara. Lo mejor de esta arquitectura es que no solo es eficiente ahora, sino que también está preparada para futuras expansiones o mejoras, lo cual es un gran plus.

El uso de la metodología Scrum fue muy útil. Trabajar de manera ágil me permitió ajustar y mejorar la solución a medida que avanzaba. Además, este proyecto me dio una visión muy práctica de cómo se gestionan los proyectos en un entorno real, subrayando la importancia de la colaboración y la comunicación en el equipo.

En resumen, este trabajo me enseñó lo crucial que es integrar bien las aplicaciones para mejorar procesos empresariales. También aprendí que tener una arquitectura sólida desde el principio es clave para el éxito de cualquier proyecto. Esta experiencia no solo me ha servido para reforzar mis conocimientos técnicos, sino que también me ha preparado para enfrentar futuros desafíos en el mundo laboral, donde la integración y automatización de sistemas son cada vez más importantes.

En definitiva, lo que me llevo de este proyecto es que, con una buena planificación y un enfoque flexible, se pueden lograr grandes cosas. Esta es una lección que pienso aplicar en cualquier proyecto que enfrente en el futuro.

## 7.1 Trabajos Futuros

A lo largo del desarrollo de este proyecto, han surgido varias ideas y oportunidades que podrían explorarse en futuros trabajos para ampliar y mejorar la solución presentada.

Una de las principales áreas de investigación futura es la implementación de un sistema de inteligencia artificial (IA) para realizar análisis predictivos, algo que en este proyecto solo se abordó de manera teórica. La integración de modelos de aprendizaje automático podría permitir analizar datos históricos de ventas, condiciones climáticas y otros factores relevantes para predecir la demanda de productos. Esto ayudaría a optimizar la producción y la gestión de inventarios, reduciendo el desperdicio y mejorando la satisfacción del cliente.

Otra dirección interesante para futuros trabajos es la expansión de la arquitectura API-led utilizada en este proyecto. Aunque ha demostrado ser eficaz, hay margen para mejorarla mediante la implementación de microservicios más especializados en las capas de experiencia, proceso y sistema. Esta evolución permitiría una mayor modularidad y escalabilidad, facilitando el mantenimiento y la actualización de la solución a medida que evolucionan las necesidades del negocio.

La trazabilidad en la cadena de suministro es otra área que podría beneficiarse de futuras investigaciones, especialmente mediante la integración de tecnologías de blockchain. Implementar blockchain podría proporcionar una manera segura e inmutable de rastrear productos desde su origen hasta el consumidor final, mejorando la transparencia y la confianza en la cadena de suministro.

También existe la posibilidad de automatizar el control de calidad mediante la incorporación de sensores adicionales que monitoreen en tiempo real la calidad de los productos durante la producción, el almacenamiento y el transporte. Este avance permitiría realizar ajustes inmediatos para mantener la frescura y calidad de los productos, basándose en datos en tiempo real.

Por último, aunque el sistema de alertas actual es funcional, futuros trabajos podrían enfocarse en desarrollar algoritmos más sofisticados para gestionar las alertas de manera más proactiva. El sistema podría priorizar alertas según la gravedad de las condiciones reportadas por los sensores y prever posibles problemas antes de que ocurran, mejorando así la capacidad de respuesta y mitigación.

## 7.2 Relación con los Estudios

Este proyecto tiene una relación directa y significativa con mis estudios de Ingeniería Informática, ya que me ha permitido aplicar muchos de los conceptos y habilidades adquiridos a lo largo de la carrera para abordar un problema complejo desde una perspectiva técnica y organizada.

Uno de los aspectos clave de este proyecto ha sido el diseño y desarrollo de software, una de las competencias fundamentales de la Ingeniería Informática. Durante el desarrollo, apliqué técnicas avanzadas de diseño de software, como la arquitectura API-led, que aprendí en mis estudios. Esta capacidad para diseñar sistemas escalables y eficientes fue crucial para el éxito del proyecto.

Además, la integración de sistemas, otra área central en Ingeniería Informática, fue un componente esencial de este proyecto. Utilicé MuleSoft para implementar APIs que conectan diferentes sistemas y tecnologías, como Shopify, Salesforce y sensores externos.

Esta experiencia práctica me permitió aplicar la teoría aprendida en clase en un contexto real y entender mejor las complejidades de la integración de sistemas.

La gestión de proyectos también jugó un papel importante en este trabajo. La adopción de la metodología Scrum, que se ha convertido en un estándar en la industria del software, me permitió gestionar el proyecto de manera ágil, adaptándome a los cambios y asegurando la entrega de un producto funcional en tiempo y forma. Esta experiencia reforzó mis habilidades en la planificación y ejecución de proyectos complejos.

La seguridad informática fue otro aspecto crucial en el desarrollo del proyecto. La protección de datos sensibles, especialmente en la gestión de información a través de APIs, requirió la implementación de prácticas de seguridad sólidas, basadas en el conocimiento adquirido durante mis estudios. Esto aseguró la integridad y seguridad del sistema en todas las etapas del proyecto.

Finalmente, aunque no llegué a implementar la inteligencia artificial para el análisis predictivo, los conocimientos adquiridos en análisis de datos y machine learning durante la carrera sentaron las bases para comprender cómo esta tecnología podría integrarse en futuros proyectos. Este proyecto me permitió explorar cómo estos conceptos pueden aplicarse en un entorno real.

# Bibliografía


[1] Thomas Erl. *Service-Oriented Architecture: Concepts, Technology, and Design*. Prentice Hall, 2005.

[2] Sam Newman. *Building Microservices: Designing Fine-Grained Systems*. O'Reilly Media, Inc., 2015. Disponible en https://www.oreilly.com/library/view/building-microservices/9781491950340/

[3] Ken Schwaber. *Scrum: The Art of Doing Twice the Work in Half the Time*. Random House, 2020.

[4] Kenneth S. Rubin. *Essential Scrum: A Practical Guide to the Most Popular Agile Process*. Addison-Wesley Professional, 2012.

[5] David J. Anderson. *Kanban: Successful Evolutionary Change for Your Technology Business*. Blue Hole Press, 2010.

[6] Roy Thomas Fielding. *Architectural styles and the design of network-based software architectures*. Doctoral dissertation, University of California, Irvine, 2000. Disponible en https://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm

[7] Cesare Pautasso, Olaf Zimmermann, y Frank Leymann. RESTful Web Services: Principles, Patterns, and Constraints. In *Proceedings of the 17th International Conference on World Wide Web*, ACM, 2008. Disponible en https://www2008.org/papers/pdf/p959-pautassoA.pdf

[8] MuleSoft Documentation. Disponible en: https://docs.mulesoft.com/.

[9] Anypoint Platform Overview. MuleSoft. Disponible en: https://www.mulesoft.com/platform/enterprise-integration.

[10] API Manager. MuleSoft. Disponible en: https://www.mulesoft.com/platform/api/api-manager.

[11] Runtime Manager. MuleSoft. Disponible en: https://www.mulesoft.com/platform/runtime-manager.

[12] Design Center. MuleSoft. Disponible en: https://www.mulesoft.com/platform/design-center.

[13] RAML. Disponible en: https://raml.org/.

[14] Salesforce CRM. Salesforce. Disponible en: https://www.salesforce.com.

[15] CloudHub. MuleSoft. Disponible en: https://www.mulesoft.com/platform/cloudhub.

[16] DataWeave. MuleSoft. Disponible en: https://docs.mulesoft.com/dataweave/.

[17] RabbitMQ Documentation. Disponible en: https://www.rabbitmq.com/documentation.html.

[18] AMQP Documentation. Disponible en: https://www.amqp.org/.

[19] Shopify Help Center. Disponible en: https://www.shopify.com.

[20] Departamento de la Guerra de los Estados Unidos. Comunicado de prensa emitido el 16 de febrero de 1946. Consultado en http://americanhistory.si.edu/comphist/pr1.pdf.

[21] Martin Fowler, y James Lewis. *Microservices: A Definition of This New Architectural Term*. Disponible en: https://martinfowler.com/articles/microservices.html, 2016.

[22] Leonard Richardson, Mike Amundsen, y Sam Ruby. *RESTful Web APIs*. O'Reilly Media, 2013.

[23] Stefan Tilkov. *A Brief Overview of REST*. Disponible en: https://www.infoq.com/articles/rest-introduction/, 2015.

[24] OWASP Foundation. *OWASP API Security Project*. Disponible en: https://owasp.org/www-project-api-security/, 2018.

[25] Mehdi Medjaoui, Erik Wilde, Ronnie Mitra, y Mike Amundsen. *Continuous API Management: Making the Right Decisions in an Evolving Landscape*. O'Reilly Media, 2018.

[26] Stuart Russell, y Peter Norvig. *Artificial Intelligence: A Modern Approach*. 4ta edición, Pearson, 2021.

[27] Ian Goodfellow, Yoshua Bengio, y Aaron Courville. *Deep Learning*. MIT Press, 2016.

[28] Jayavardhana Gubbi, Rajkumar Buyya, Slaven Marusic, y Marimuthu Palaniswami. *Internet of Things (IoT): A Vision, Architectural Elements, and Future Directions*. Future Generation Computer Systems, 29(7), 1645-1660, 2013.

[29] Mahadev Satyanarayanan. *The Emergence of Edge Computing*. Computer, 50(1), 30-39, 2017.

[30] Mike Amundsen, y Erik Wilde. *RESTful Web APIs*. O'Reilly Media, 2013.

[31] Gregor Hohpe, y Bobby Woolf. *Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions*. Addison-Wesley Professional, 2012.

[32] Olaf Hartig. *Querying the Web of Data with GraphQL*. 2018.

[33] MuleSoft. *API-Led Connectivity: The Key to Agile Digital Transformation*. Disponible en: https://www.mulesoft.com/resources/api/what-is-api-led-connectivity, 2020.

[34] Ken Schwaber. *Scrum: The Art of Doing Twice the Work in Half the Time*. Random House, 2020.

[35] Kenneth S. Rubin. *Essential Scrum: A Practical Guide to the Most Popular Agile Process*. Addison-Wesley Professional, 2012.

[36] David J. Anderson. *Kanban: Successful Evolutionary Change for Your Technology Business*. Blue Hole Press, 2010.

# APÉNDICE A
# Objetivos de Desarrollo Sostenible

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 |

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

El desarrollo sostenible se ha convertido en un tema de crucial importancia en la agenda global, y es fundamental que cualquier proyecto, independientemente de su enfoque, considere cómo puede contribuir a este objetivo global. Este Trabajo de Fin de Grado se ha planteado dentro de un contexto donde la automatización y la eficiencia en la cadena de suministro no solo mejoran los resultados empresariales, sino que también pueden generar impactos positivos en varios aspectos del desarrollo sostenible.

## A.1 Relación del proyecto con los ODS

### ODS 4: Educación de calidad

Aunque en primera instancia podría parecer que el proyecto tiene una relación indirecta con el ODS 4, en realidad, la mejora en los procesos empresariales a través de la integración y automatización puede llevar a una optimización significativa en la formación y educación en sectores industriales específicos. Por ejemplo, al implementar soluciones tecnológicas avanzadas, se abre la puerta a la capacitación continua de los empleados en nuevas tecnologías, fomentando un entorno de aprendizaje constante. Este enfoque contribuye indirectamente a la educación de calidad, especialmente en áreas técnicas y de gestión de operaciones.

### ODS 8: Trabajo decente y crecimiento económico

El ODS 8 está estrechamente relacionado con este proyecto, ya que uno de los objetivos fundamentales es mejorar la eficiencia y la productividad de la cadena de suministro en el sector alimenticio. Al automatizar procesos y mejorar la integración de sistemas, se crean condiciones de trabajo más seguras y eficientes, lo que contribuye a un entorno laboral más estable y productivo. Además, la optimización de los recursos y la reducción de desperdicios promueven un crecimiento económico más sostenible y equitativo, lo que es esencial para el cumplimiento de este objetivo.

### ODS 9: Industria, innovación e infraestructuras

Este proyecto se alinea de manera significativa con el ODS 9, ya que la implementación de una arquitectura API-Led y la integración de sistemas avanzados directamente promueve la innovación y el desarrollo de infraestructuras tecnológicas en la industria alimentaria. La capacidad de integrar tecnologías emergentes, como el Internet de las Cosas (IoT) y potencialmente la inteligencia artificial, posiciona este proyecto en la vanguardia de la innovación industrial, fomentando una industria más moderna, resiliente y sostenible.

### ODS 11: Ciudades y comunidades sostenibles

La mejora en la gestión de la cadena de suministro tiene un impacto directo en la sostenibilidad de las ciudades y comunidades. Al optimizar los procesos de logística y reducir el desperdicio de alimentos, el proyecto contribuye a una menor huella ecológica y a un uso más eficiente de los recursos. Esto, a su vez, mejora la calidad de vida en las comunidades donde se implementan estas mejoras, apoyando el desarrollo de ciudades más sostenibles.

### ODS 12: Producción y consumo responsables

El ODS 12 está directamente relacionado con los objetivos de este proyecto. Al enfocar la automatización en la reducción del desperdicio y en la optimización de la cadena de suministro, se promueve un consumo y una producción más responsables. La integración de sensores y tecnologías predictivas, aunque no implementadas en esta fase del proyecto, tiene el potencial de minimizar el desperdicio de alimentos y recursos, alineándose con los principios de este objetivo.

### ODS 13: Acción por el clima

Aunque la relación del proyecto con el ODS 13 es más indirecta, no deja de ser importante. La optimización de la cadena de suministro y la reducción del desperdicio tienen un impacto positivo en la reducción de la huella de carbono de las operaciones empresariales. Menos desperdicio significa menos recursos desperdiciados y, por tanto, una menor necesidad de producir y transportar bienes, lo que contribuye a la reducción de emisiones de gases de efecto invernadero.

En conclusión, este proyecto tiene una relación significativa con varios de los Objetivos de Desarrollo Sostenible. Aunque no todos los ODS están directamente implicados en el trabajo realizado, es evidente que los principios de sostenibilidad, eficiencia y mejora continua que guían este proyecto contribuyen a un futuro más sostenible. La adopción de tecnologías avanzadas y la optimización de procesos no solo benefician a las empresas y a sus empleados, sino que también tienen un impacto positivo en las comunidades y en el medio ambiente. Por lo tanto, este proyecto representa un paso hacia la integración de prácticas sostenibles en la industria alimentaria y más allá, apoyando los esfuerzos globales por un desarrollo más equitativo y responsable.