Universidad de Cuenca

Facultad de Ingeniería

Carrera de Computación

Arquitectura de Microservicios para Aplicaciones de Gestión Empresarial: Estrategias Modulares para Escalabilidad y Flexibilidad

Trabajo de titulación previo a la obtención del título de Ingeniero en Ciencias de la Computación

Autor:

Michael Enmanuel Peñaloza Zhañay

Director:

Miguel Ángel Zúñiga Prieto

ORCID: 0000-0001-9369-1813

Cuenca, Ecuador

2025-09-25

Resumen

Las aplicaciones empresariales modernas requieren estructuras de software que permitan su evolución, escalabilidad y mantenimiento sin afectar la operación general del sistema. Sin embargo, muchos sistemas actuales presentan limitaciones al estar basados en arquitecturas monolíticas, dificultando la adaptabilidad a nuevos procesos y tecnologías. En este contexto, la arquitectura de microservicios se presenta como una alternativa para diseñar aplicaciones empresariales más modulares y desacopladas. Este trabajo de titulación propone el diseño de una arquitectura de software basada en microservicios, aplicada a un sistema de gestión comercial. El diseño se desarrolló utilizando el método 4SRS-MSLA, el cual guía la transformación de requerimientos funcionales en componentes lógicos. Complementariamente, se utilizó la notación SoaML para modelar participantes, interfaces y contratos entre microservicios, asegurando claridad en la especificación de servicios y flujos de interacción. Como parte del proceso, se incorporaron herramientas prácticas como un diagrama de bloques y una tabla de APIs, que facilitaron la comunicación con el stakeholder y permitieron validar tempranamente los servicios esperados entre módulos. Se realizó además el diseño detallado de los microservicios mas importantes para el stakeholder y se construyó un prototipo funcional que integró dichos microservicios. A través de un caso funcional representativo se probó la interoperabilidad entre servicios y se confirmó la viabilidad técnica del diseño. La arquitectura propuesta demuestra ser una solución factible, modular y escalable, capaz de guiar el desarrollo de sistemas empresariales modernos orientados a servicios, cumpliendo así con los objetivos planteados en esta investigación.

Palabras clave del autor: interoperabilidad de sistemas, modelado de negocio, gestión contable

El contenido de esta obra corresponde al derecho de expresión de los autores y no compromete el pensamiento institucional de la Universidad de Cuenca ni desata su responsabilidad frente a terceros. Los autores asumen la responsabilidad por la propiedad intelectual y los derechos de autor.

Repositorio Institucional: https://dspace.ucuenca.edu.ec/

Abstract

Modern business applications require software structures that allow for evolution, scalability, and maintenance without affecting the overall operation of the system. However, many current systems have limitations because they are based on monolithic architectures, making it difficult to adapt to new processes and technologies. In this context, microservices architecture is presented as an alternative for designing more modular and decoupled business applications. This thesis proposes the design of a microservices-based software architecture applied to a commercial management system. The design was developed using the 4SRS-MSLA method, which guides the transformation of functional requirements into logical components. Complementarily, SoaML notation was used to model participants, interfaces, and contracts between microservices, ensuring clarity in the specification of services and interaction flows. As part of the process, practical tools such as a block diagram and an API table were incorporated, which facilitated communication with the stakeholder and allowed for early validation of the expected services between modules. The detailed design of the most important microservices for the stakeholder was also carried out, and a functional prototype was built that integrated these microservices. Through a representative functional case, the interoperability between services was tested and the technical feasibility of the design was confirmed. The proposed architecture proves to be a feasible, modular, and scalable solution, capable of guiding the development of modern service-oriented business systems, thus fulfilling the objectives set out in this research.

Author keywords: systems interoperability, business modeling, accounting management

The content of this work corresponds to the right of expression of the authors and does not compromise the institutional thinking of the University of Cuenca, nor does it release its responsibility before third parties. The authors assume responsibility for the intellectual property and copyrights.

Institutional Repository: https://dspace.ucuenca.edu.ec/

Índice de contenido

Resumen **2**

Abstract **3**

Agradecimientos **10**

Dedicatoria **11**

1 Introducción **12**
1.1 Contexto y justificación 12
1.2 Objetivos 13
1.2.1 Objetivo General 13
1.2.2 Objetivo Específico 13

2 Marco Teórico **14**
2.1 Arquitectura monolítica 14
2.2 Arquitectura orientada a microservicios 14
2.2.1 Principios arquitectónicos 15
2.3 Tecnologías para el desarrollo 16
2.3.1 Spring Boot 16
2.3.2 React 16
2.3.3 MySQL 16
2.3.4 Railway 16
2.4 Marco Histórico 17

3 Estado del Arte **18**

4 Arquitectura de microservicios **19**
4.1 Transformación de Casos de Uso en Servicios 19
4.2 Modelado de Servicios en SoaML 20

5 Diseño de la arquitectura basada en microservicios **21**
5.1 Modelado del Negocio 21
5.1.1 Proceso de ventas en el sistema comercial 22
5.1.2 Proceso de compras en el sistema comercial 24
5.1.3 Proceso de importaciones en el sistema comercial 26
5.1.4 Diagrama de flujo del dominio 28
5.2 Justificación de la elección de la arquitectura basada en microservicios tras el modelado del negocio 30
5.3 Levantamiento de Casos de Uso de primer nivel 31
5.3.1 Identificación de actores 31
5.3.2 Diagrama de casos de uso de primer nivel 32
5.4 Levantamiento de casos de uso de segundo nivel 34

5.4.1 Descomposición funcional del caso de uso Gestión de Compras 34
5.4.2 Descomposición del caso de uso de Gestión de Productos 35
5.4.3 Descomposición del caso de uso de Gestión Contable 36
5.5 Levantamiento de casos de uso de tercer nivel 37
5.5.1 Casos de uso de tercer nivel obtenidos del caso de uso Gestión de compras del sistema 38
5.5.2 Casos de uso de tercer nivel obtenidos del caso de uso Gestión de productos del sistema 39
5.5.3 Casos de uso de tercer nivel obtenidos del caso de uso Gestión de plan de cuentas 41
5.5.4 Refinamiento del caso de uso de tercer nivel 43
5.6 Separación de Componentes 44
5.6.1 Identificación de interacciones entre casos de uso 45
5.6.2 Interacciones funcionales entre casos de uso en el módulo (G.D.C) Gestión de Compras 45
5.6.3 Interacción funcional del caso de uso (G.D.P) Gestión de productos . . . 47
5.6.4 Interacción funcional del caso de uso Gestión contable 48
5.6.5 Identificacíon y clasificación de los componentes derivados de los casos de uso 50
5.7 Identificación de Microservicios 50
5.7.1 Análisis de mensajes para la comunicación de microservicios 51
5.7.2 Diagrama de Participantes de Servicio 53
5.8 Definir Interfaces y Operaciones de Servicios 54
5.8.1 Interfaz de servicio del microservicio Compras 55
5.8.2 Interfaz de servicio del microservicio Productos 56
5.8.3 Interfaz de servicio del microservicio Contabilidad 57
5.9 Especificar Contratos entre Proveedores y Consumidores 57
5.9.1 Contrato de servicio del microservicio Compras 58
5.9.2 Contrato de servicio del microservicio Productos 58
5.9.3 Contratos de servicio del microservicio Contabilidad 59
5.10 Modelar Roles que Ofrecen/Consumen Servicios 60
5.10.1 Modelado del Diagrama de Participantes de Servicios 60

6 Desarrollo del prototipo 62
6.1 Implementación del diseño arquitectónico de microservicios 62
6.1.1 Entorno de despliegue del prototipo 62
6.2 Codificación y Testeo 64
6.2.1 Estructura del backend por microservicio 64
6.2.2 Implementación de DTOs para consumo de microservicios 66
6.3 Estructura del frontend 67
6.4 Diseño de interfaces 69
6.4.1 Interfaz de historial de facturas de compra 69
6.4.2 Interfaz de creación de factura de compra 70

6.4.3 Interfaz de selección de productos para la factura de compra 71
6.4.4 Interfaz de consulta de productos 71
6.4.5 Interfaz de creación de producto 72

7 Validación funcional del prototipo 73
7.1 Caso funcional propuesto para la validación del prototipo 73
7.1.1 Objetivo del caso funcional 73
7.1.2 Funcionamiento del caso 73
7.1.3 Validaciones técnicas realizadas 75
7.1.4 Conclusión del caso funcional 75

8 Metodología resultante del caso de estudio 76
8.1 Modelado del negocio 76
8.2 Levantamiento de casos de uso 77
8.3 Refinamiento de los casos de uso 77
8.4 Identificación de interacciones entre casos de uso 78
8.5 Identificación y clasificación de componentes (4SRS) 79
8.6 Identificación de microservicios 79
8.7 Análisis de mensajes para la comunicación entre microservicios 79
8.8 Modelado con SoaML 80
8.9 Representación de la implementación física 80
8.10 Codificación, testeo y construcción del prototipo 80

9 Conclusiones, Discusión y Trabajo futuro 81
9.1 Conclusiones 81
9.2 Contribuciones 82
9.3 Discusión de resultados 83

Referencias 85

Anexo A: Tabla de Componentes para los casos de uso de Gestión de Facturas de Compra 88

Índice de figuras

Fig. 1 Especificación de microservicios utilizando el método 4SRS-MSLA. 19
Fig. 2 Diagrama de flujo del proceso de ventas en el sistema comercial 23
Fig. 3 Diagrama de flujo del proceso de compras locales 25
Fig. 4 Diagrama de flujo del proceso de importaciones 27
Fig. 5 Diagrama funcional de procesos comerciales 29
Fig. 6 Diagrama de casos de uso de primer nivel del sistema comercial 33
Fig. 7 Casos de uso de segundo nivel del caso de uso de gestión de compras . . . 34
Fig. 8 Casos de uso de segundo nivel del caso de uso de Gestión de Productos . . 35
Fig. 9 Casos de uso de segundo nivel del caso de uso de gestión contable 37
Fig. 10 Casos de uso de tercer nivel del caso de uso G.D.C.1 Gestión de compras del sistema 38
Fig. 11 Casos de uso actualizados del caso de uso G.D.P.1 Gestión de productos del sistema 40
Fig. 12 Casos de uso de tercer nivel del caso de uso G.C.1 Gestión de plan de cuentas 42
Fig. 13 Refinamiento del caso de uso G.D.C.1.1 Crear factura de compra 43
Fig. 14 Diagrama de interacción funcional del caso de uso G.D.C Gestión de Compras 46
Fig. 15 Diagrama de interacciones del caso de uso G.D.I Gestión de productos . . . 47
Fig. 16 Interacción del caso de uso G.C Gestión contable con otros módulos 49
Fig. 17 Diagrama de bloques para la identificación de microservicios 51
Fig. 18 Diagrama de servicios y participantes: interacción entre Compras, Productos y Contabilidad 54
Fig. 19 Diagrama de interfaz de servicio del microservicio Compras 55
Fig. 20 Diagrama de interfaz de servicio del microservicio Productos 56
Fig. 21 Diagrama de interfaz de servicio del microservicio Contabilidad 57
Fig. 22 Contrato de servicio del microservicio Compras 58
Fig. 23 Contrato de servicio del microservicio Productos 59
Fig. 24 Contratos de servicio del microservicio Contabilidad 60
Fig. 25 Diagrama final de arquitectura de servicios del sistema comercial 61
Fig. 26 Entorno de despliegue del prototipo con microservicios en Spring Boot . . . 63
Fig. 27 Estructura del microservicio Compras en Spring Boot 65
Fig. 28 Clase `ProductoDto` utilizada para consumir el microservicio de Productos . . 66
Fig. 29 Estructura del proyecto frontend desarrollado en React 68
Fig. 30 Listado de facturas registradas 70
Fig. 31 Formulario para registrar una nueva factura de compra 70
Fig. 32 Interfaz de selección de productos en el proceso de compra 71
Fig. 33 Listado y filtro de productos registrados 71
Fig. 34 Formulario de creación de producto 72
Fig. 35 Lista de productos recibida consumiendo API desde Compras 74
Fig. 36 Lista de productos cargada en la interfaz 75

Fig. 37 Metodología para el diseño de una arquitectura basada en microservicios . . 76

Índice de tablas

Tabla. 1 Reglas de transición desde UML (en AMPLA) hacia SoaML 21
Tabla. 2 Actores identificados en el sistema comercial 32
Tabla. 3 Campos refinados del caso de uso G.D.C.1.1 Crear factura de compra . . . 44
Tabla. 4 Tabla de mensajes identificadas entre Compras y Productos 52
Tabla. 5 Resumen de servicios ofrecidos y participantes consumidores 54
Tabla. 6 Componentes de infraestructura para la implementación física de servicios . 64
Tabla. 7 Componentes identificados en los casos de uso de Gestión de Facturas de Compra 88

Agradecimientos

Quiero expresar mis agradecimiento a todas las personas que han hecho posible la realización de esta tesis. En primer lugar, a mi familia por su amor y apoyo incondicional, especialmente a mi mamá Juana Zhañay que está en el cielo y hasta el último día siempre creyó en mí; sin ella haber formado parte de mi vida este logro no habría sido posible.

A mi director de tesis, el Ing. Miguel Ángel Zuñiga, por su dedicación, paciencia y orientación durante todo el proceso de desarrollo de este trabajo.

A la Empresa Advantech Digital, por brindarme la oportunidad de realizar este proyecto y por su colaboración durante todo el proceso.

A mis amigos y compañeros de estudio, por su apoyo, motivación y por compartir conmigo esta etapa de mi vida académica.

Finalmente, a todos los profesores de la carrera por impartir sus conocimientos y su dedicación en nuestra formación.

Dedicatoria

Quiero dedicar este trabajo a mi familia, especialmente a mi mamá Juana Zhañay, aunque ya no este a mi lado siempre la tengo presente, a mi tia Alicia Zhañay que fue la que un momento en el que todo estaba perdido vino a darme una mano y sin ella seria imposible llegar hasta donde e llegado, a mi hermana Cristina quien siempre encontró las palabras correctas para motivarme a continuar y a todo el resto de mi familia que en algún momento me dio la mano para seguir en mi carrera.

Finalmente, agradezco a todas aquellas personas que, de una u otra manera, contribuyeron al logro de este objetivo académico. Su apoyo y contribuciones, aunque no mencionadas individualmente, son igualmente valoradas y apreciadas.

1. Introducción

1.1. Contexto y justificación

En los últimos años, las aplicaciones en el dominio comercial han experimentado un crecimiento significativo, impulsado por la digitalización y la demanda de experiencias de compra más ágiles y personalizadas. Este crecimiento ha traído consigo la necesidad de desarrollar plataformas de comercio robustas y escalables capaces de gestionar un número creciente de usuarios y transacciones de manera eficiente y segura[1]. En este contexto, las arquitecturas monolíticas tradicionales, aunque eficaces en entornos más pequeños o menos dinámicos, se enfrentan a retos en cuanto a escalabilidad, flexibilidad y mantenimiento a medida que las aplicaciones crecen[2]. Por ejemplo, el escalado de una parte específica de la aplicación requiere escalar todo el sistema, lo cual puede resultar ineficiente y costoso[3].

Para enfrentar estas limitaciones, la arquitectura de microservicios ha emergido como una solución que permite descomponer aplicaciones grandes en servicios independientes que pueden evolucionar y escalar de manera selectiva y autónoma [4]. Este estilo arquitectónico propone dividir la aplicación en componentes más pequeños (microservicios), cada uno de los cuales gestiona una función específica del negocio y puede desarrollarse, desplegarse y mantenerse de forma independiente [3]. Esto facilita no solo el escalado selectivo de las aplicaciones, sino el uso efectivo de recursos [5].

A pesar del creciente interés en el estilo arquitectónico de microservicios, la implementación efectiva de esta arquitectura en aplicaciones comerciales de empresas sigue siendo un desafío complejo para empresas grandes y más aún para las pequeñas y medianas empresas (PYMES)[2]. Las organizaciones enfrentan dificultades significativas, desde la gestión de servicios distribuidos, la seguridad, la comunicación entre servicios, hasta la integración con sistemas heredados[4]. Además, la naturaleza dinámica y cambiante de los procesos comerciales provoca que las aplicaciones de gestión comercial sean construidas a la medida de las necesidades de cada empresa[6]. Es decir, la existencia de aplicaciones comerciales en el mercado no solventa las necesidades específicas de las otras empresas, requiriendo incurrir en desarrollos propios[7]. Sumado a esto, las arquitecturas monolíticas con las que se construyen estas aplicaciones generan dificultades comerciales y técnicas. Desde el punto de vista comercial, las empresas no pueden comercializar módulos independientes especializados en soportar únicamente los procesos de negocio requeridos por sus clientes, viéndose obligadas a ofrecer el sistema completo a sus clientes. Desde el punto de vista técnico, las aplicaciones tienen problemas de escalabilidad, requiriendo recursos informáticos no utilizados para todos sus módulos[4].

En este contexto, este trabajo de tesis propone el diseño arquitectónico basado en microservicios de una aplicación de gestión comercial que soporte los procesos de negocio relacionados a ventas, compras, tesorería, inventarios, bancos, contabilidad y tributación. Un enfoque basado en microservicios permitirá superar los problemas comerciales y técnicos identificados

previamente. El diseño arquitectónico propuesto en este trabajo servirá como base para que las pequeñas y medianas empresas (PYMES) construyan aplicaciones comerciales modulares y escalables. Este trabajo de titulación toma como caso de estudio el sistema de gestión comercial de la empresa Advantech, empresa cuencana que brinda servicios tecnológicos a gasolineras a nivel nacional; estos servicios incluyen la provisión de un sistema informático que soporta los procesos comerciales. El alcance de este trabajo cubrirá el diseño arquitectónico basado en microservicios de la aplicación comercial, el diseño detallado de un microservicio y el prototipo de su implementación. En donde el prototipo incluirá la implementación de funcionalidades de un microservicio identificado como prioritario y la implementación de la interacción con otro microservicio. Dejando el diseño detallado y la implementación de los microservicios y funcionalidades faltantes para etapas posteriores a este trabajo de tesis.

Desde el punto de vista metodológico, este trabajo aplicará la metodología 4SRS-MSLA (Four Step Rule Set - Microservices Oriented Logical Architecture), esta metodología propone cuatro actividades para diseñar arquitecturas basadas en microservicios[8]. La metodología propone primero modelar el dominio empresarial en casos de uso de UML (Unified Modeling Language), luego derivar un diagrama de componentes de UML para el dominio y así poder agrupar los componentes en microservicios[9]. Esto permite identificar y definir los elementos clave de la arquitectura de microservicios, asegurando una separación adecuada de las funcionalidades en módulos independientes[10]. En la segunda fase del enfoque entraría el cuarto paso donde estos componentes son identificados y se deriva su comportamiento en microservicios utilizando SoaML (Service-oriented architecture Modeling Language), lo que permite definir los canales de comunicación y contratos de servicio entre ellos[8].

1.2. Objetivos

1.2.1. Objetivo General

Diseñar una arquitectura basada en microservicios para el desarrollo de aplicaciones empresariales que respaldan los procesos comerciales, con el objetivo de mejorar la escalabilidad, flexibilidad y mantenimiento.

1.2.2. Objetivo Específico

- Obtener los requerimientos que debe satisfacer el sistema: Identificar y documentar los requerimientos funcionales y no funcionales del sistema.
- Diseñar la arquitectura de basada en microservicios del Sistema: Diseñar la arquitectura del sistema, incluyendo vistas arquitectónicas que se consideren relevantes para la comunicación con diferentes partes interesadas.
- Diseñar la arquitectura detallada de un microservicio específico: Desarrollar el diseño

detallado de uno de los microservicios propuestos, definiendo sus componentes internos.

- Probar la factibilidad de implementación del diseño arquitectónico propuesto.

2. Marco Teórico

2.1. Arquitectura monolítica

En general, la arquitectura monolítica utiliza una única tecnología de desarrollo, lo que limita la disponibilidad de herramientas adecuadas para cada tarea que el sistema necesita ejecutar. La mayoría de los aspectos involucrados, como las capas de presentación, procesamiento y almacenamiento, forman un único componente de software que debe ejecutarse en un servidor. La arquitectura monolítica ofrece ciertas ventajas y desventajas que generan confianza o inconvenientes tecnológicos. Es necesario identificar los riesgos que pueden derivar en problemas a corto o largo plazo [4].

Entre las ventajas, es fácil de desarrollar debido a su estructura sencilla, donde todos los componentes del sistema se encuentran integrados en una única aplicación. Esto permite que el proceso de desarrollo sea más directo, sin la necesidad de gestionar la comunicación entre múltiples servicios [3].

Sin embargo, entre las desventajas, estos sistemas son rígidos y difíciles de adaptar a nuevas necesidades. Incrementar su capacidad de procesamiento requiere cambiar el servidor actual por uno de mayor tamaño. Además, su tecnología es propietaria, lo que genera una gran dependencia del cliente hacia la empresa proveedora. Los costos de adquisición, renovación y soporte suelen ser elevados [4].

El nivel de complejidad de los sistemas modernos exige mejoras tanto en la producción de software como en el rendimiento. Esto implica que, con el tiempo, se han encontrado defectos inevitables en la arquitectura monolítica, los cuales eventualmente resultan contraproducentes, dando paso a arquitecturas modernas como los microservicios [4].

2.2. Arquitectura orientada a microservicios

Los microservicios adoptaron los principios y conceptos de la arquitectura orientada a servicios (SOA) y organizan una aplicación basada en servicios en un conjunto reducido de servicios de software con bajo acoplamiento. Para lograr un mayor desacoplamiento en las aplicaciones del lado del servicio, la arquitectura de microservicios sugiere dividir estas aplicaciones en varios servicios independientes enfocados en responsabilidades específicas del negocio. La aplicación del servidor se fragmenta en múltiples microservicios más detallados, cada uno encargado de cumplir con una responsabilidad empresarial particular, y se gestionan para operar en contenedores separados. Cada contenedor tiene su propia base de datos privada, la

cual no puede ser accedida por otros contenedores [11].

Las ventajas de la arquitectura de microservicios incluyen la simplificación del proceso de incorporación, la flexibilidad en el almacenamiento de datos y la posibilidad de utilizar diversos lenguajes de programación comprensibles. Además, permiten gestionar problemas relacionados con la tolerancia y el aislamiento de fallos, ofrecen escalabilidad para manejar el crecimiento de datos y la concurrencia sin afectar el rendimiento, y facilitan la monitorización de la seguridad [5].

2.2.1. Principios arquitectónicos

En esta parte se presentan los principios arquitectónicos que se tomaron como guía para comprender por qué se optó por ciertos enfoques en el diseño del sistema. Estos principios ayudan a justificar las decisiones tomadas, y a continuación se detallan los más importantes en el contexto de este trabajo.

- Encapsulamiento: Se refiere a mantener oculto el estado interno de un componente, permitiendo su acceso únicamente a través de métodos específicos. Esta práctica mejora la seguridad y evita manipulaciones no controladas desde el exterior.[12]
- Acoplamiento: Este concepto se refiere al nivel de dependencia entre componentes del sistema. Un acoplamiento bajo es deseable porque reduce el impacto de cambios en otros módulos.[13]
- Cohesión: Este principio mide qué tan enfocadas están las responsabilidades dentro de un módulo. Se busca siempre una alta cohesión, ya que mejora la claridad funcional y facilita el mantenimiento.[13]
- Don't Repeat Yourself (DRY): Promueve evitar la duplicación de lógica o estructuras de código dentro del sistema. Mantener una única fuente de verdad reduce errores y facilita los cambios.[14]
- Separación de Concerns (SoC): Consiste en dividir el sistema en partes que se ocupen de aspectos distintos del problema. Esta separación simplifica el diseño y facilita su comprensión y evolución.[12]
- Ley de Demeter: Establece que un objeto debe comunicarse sólo con aquellos componentes con los que tiene una relación directa. Esto fomenta un diseño más limpio y menos acoplado.[15]
- Keep it Simple, Stupid! (KISS): Este principio sugiere que las soluciones simples son preferibles. Diseños innecesariamente complejos son más propensos a fallos y difíciles de mantener.[12]

- Inversion of Control (IoC): Plantea que los componentes de bajo nivel no deben controlar el flujo de ejecución, sino que este debe estar gestionado por un contenedor externo o framework. Esto permite una arquitectura más flexible y menos dependiente.[16]

2.3. Tecnologías para el desarrollo

2.3.1. Spring Boot

Spring Boot es una herramienta diseñada para facilitar la creación de aplicaciones independientes y listas para producción basadas en el marco Spring. Permite ejecutar estas aplicaciones de manera directa, con configuraciones predeterminadas y flexibilidad para personalizaciones específicas [17].

2.3.2. React

React es una biblioteca desarrollada en JavaScript para construir interfaces de usuario modulares y reutilizables. Permite estructurar aplicaciones dividiéndolas en componentes jerárquicos, aptos para sitios web o aplicaciones móviles [18].

2.3.3. MySQL

MySQL es un sistema de gestión de bases de datos relacionales (RDBMS) de código abierto, ampliamente utilizado para almacenar y administrar datos estructurados. Utiliza el lenguaje SQL para gestionar la información y destaca por su fiabilidad, rendimiento, escalabilidad y soporte a transacciones ACID. Es versátil, adecuado tanto para proyectos pequeños como para aplicaciones empresariales de gran escala, y es considerado la base de datos de código abierto más popular del mundo[19].

2.3.4. Railway

Railway es una plataforma de despliegue en la nube que permite a los desarrolladores crear y administrar fácilmente aplicaciones y servicios, incluyendo bases de datos. Su principal objetivo es simplificar el ciclo de vida del desarrollo, eliminando la necesidad de configuraciones complejas o mantenimiento de servidores. Con Railway, es posible levantar una base de datos en la nube en pocos segundos y conectarla de inmediato a tu aplicación, lo que lo convierte en una opción ideal para proyectos modernos que requieren rapidez y eficiencia[20].

2.4. Marco Histórico

La arquitectura es el componente clave que permite a los sistemas adaptarse y mantener un nivel específico de servicio a lo largo de su ciclo de vida. En el ámbito de la ingeniería de software, su función principal es conectar la funcionalidad del sistema con los requisitos de calidad que este debe cumplir. Durante las últimas décadas, la arquitectura de software ha sido objeto de un estudio exhaustivo, lo que ha llevado a los ingenieros a desarrollar diversas estrategias para construir sistemas que ofrezcan funcionalidades amplias y cumplan con una variedad de requisitos [21].

La primera generación de arquitecturas orientadas a servicios (SOA) planteó requisitos que resultaron complejos y vagamente definidos, como la capacidad de descubrir servicios y los contratos asociados a estos, lo que dificultó su implementación generalizada. Los microservicios surgen como una evolución del enfoque SOA y de la computación orientada a servicios (SOC), buscando simplificar el diseño al eliminar niveles innecesarios de complejidad y priorizar la creación de servicios independientes que realicen funciones específicas de manera eficiente. Al igual que el paradigma orientado a objetos (OO), los microservicios demandan herramientas especializadas que respalden a los desarrolladores, dando lugar a la aparición de patrones de diseño propios. Aunque aún se utilizan lenguajes orientados a objetos como Java y JavaScript, también se empieza a recurrir a lenguajes funcionales para el desarrollo de microservicios [21].

La arquitectura de microservicios surgió como un paradigma para desarrollar aplicaciones mediante pequeños servicios independientes. Cada servicio ejecuta su propio proceso y se comunica utilizando mecanismos ligeros. Este enfoque se basa en principios de SOA, pero lleva estos conceptos al nivel de las aplicaciones, enfocándose en arquitecturas más detalladas y orientadas al servicio. El término ”microservicios” fue acuñado en 2011 durante un taller de arquitectura para describir ideas comunes entre los participantes sobre patrones arquitectónicos. No obstante, enfoques similares ya existían anteriormente bajo otras denominaciones, como el ”SOA de grano fino” utilizado por Netflix [21].

Los microservicios se convirtieron en una tendencia clave en arquitectura de software, destacándose por facilitar el diseño y desarrollo de sistemas altamente escalables y fáciles de mantener. Este modelo divide grandes sistemas en servicios autónomos, lo que permite un desarrollo y despliegue independiente. Gracias a su enfoque en bajo acoplamiento y alta cohesión, los microservicios llevan la modularidad a un nivel superior. Aunque ofrecen ventajas como mejor mantenimiento y escalabilidad, también enfrentan desafíos derivados de los sistemas distribuidos. A pesar de sus similitudes con SOA, los microservicios presentan características únicas que los diferencian claramente de su predecesor [21].

3. Estado del Arte

Las empresas de software en todo el mundo han adoptado ampliamente la arquitectura de microservicios debido a su alta escalabilidad, facilidad de mantenimiento, flexibilidad y eficiencia en el desarrollo[22]. Compañías como Amazon, Uber, Netflix y Spotify han liderado esta transición, logrando importantes avances en la implementación de este enfoque[22]. Los microservicios se caracterizan por ser pequeños servicios independientes, cada uno ejecutándose en su propio proceso y comunicándose a través de protocolos ligeros como HTTP. Diseñados en torno a necesidades específicas del negocio, estos servicios pueden desarrollarse y desplegarse de manera autónoma [23].

La migración desde sistemas monolíticos hacia microservicios implica descomponer aplicaciones grandes en servicios más pequeños y coherentes. Este proceso se basa en el análisis del dominio, los requisitos del negocio, el código fuente y el historial de versiones para identificar posibles microservicios[24]. Sin embargo, esta transición presenta desafíos significativos, ya que requiere experiencia técnica para manejar aspectos como la escalabilidad, la independencia de los componentes, la comunicación entre servicios, el despliegue y la supervisión [23].

Un reto crítico es identificar correctamente los límites de los microservicios, especialmente en la etapa inicial de descomposición[24]. Una definición inadecuada puede resultar en sistemas más complejos y de menor calidad. Esto convierte la extracción de microservicios en un proceso subjetivo, dependiente del juicio humano y propenso a errores, lo que subraya la importancia de contar con ingenieros expertos para garantizar el éxito de la migración [23].

No obstante, es fundamental destacar que muchas de estas propuestas aún se encuentran en etapas iniciales, careciendo de especificaciones completas o evaluaciones rigurosas que validen su eficacia[25]. La investigación en migración de software, especialmente en lo relacionado con los microservicios, suele carecer de procesos sólidos de validación y verificación de sus modelos en sistemas reales. Esta limitación dificulta la generalización de los resultados obtenidos en dichos estudios[25]. Por lo tanto, es crucial llevar a cabo investigaciones empíricas más robustas y realizar implementaciones prácticas que evalúen los modelos de migración en diversos contextos reales, garantizando así su confiabilidad y aplicabilidad [23, 24].

El uso de la arquitectura de microservicios se ha convertido en una tendencia significativa para las pequeñas y medianas empresas (PYME), particularmente en el contexto de los sistemas de comercio electrónico[26]. Este estilo arquitectónico se está empleando en varias aplicaciones diseñadas para las PYME, incluidos los sistemas de gestión empresarial, las solicitudes de préstamos en línea, las plataformas de facturación, los modelos de inteligencia empresarial y los backends de comercio electrónico[27]. Los microservicios se consideran una forma de abordar los desafíos financieros y mejorar la transformación y el análisis de datos para estos [28]. Además, los microservicios se están integrando en plataformas y servicios basados en la nube, como la facturación electrónica y las funciones de geolocalización,

que son relevantes para las operaciones de comercio electrónico 679. Este enfoque apoya el creciente sector del comercio electrónico y ayuda a las PYME a mejorar su logística y gestión de la cadena de suministro a través de aplicaciones de computación en la nube [29].

4. Arquitectura de microservicios

4.1. Transformación de Casos de Uso en Servicios

Primero se toma todo el sistema y se divide en subsistemas según las funcionalidades principales. Esta separación se basa en capacidades de negocio o subdominios, siguiendo patrones como los de Domain-Driven Design (DDD) cada subsistema es un caso de uso de primer nivel. Luego, cada subsistema es refinado de forma independiente. Para ello, se identifican nuevos casos de uso en UML centrados exclusivamente en las funcionalidades del subsistema. Esta etapa permite detallar con mayor precisión los requisitos previamente identificados.[10]

Con este propósito, se aplica el método 4SRS-MSLA, ilustrado en la Figura 1, que guía la transformación lógica de los casos de uso en una arquitectura orientada a microservicios. Inicialmente, cada componente UML se especifica a partir de los casos de uso refinados. Posteriormente, se identifican los microservicios correspondientes y se deriva su comportamiento mediante los *Service Participants* definidos en SoaML. A continuación, se determinan los canales de comunicación y los contratos entre dichos participantes.[10]

El objetivo fundamental de este enfoque es lograr una representación lógica del comportamiento interno y de las interacciones entre microservicios, garantizando que todos los requisitos funcionales sean cubiertos adecuadamente por la solución derivada.[10]

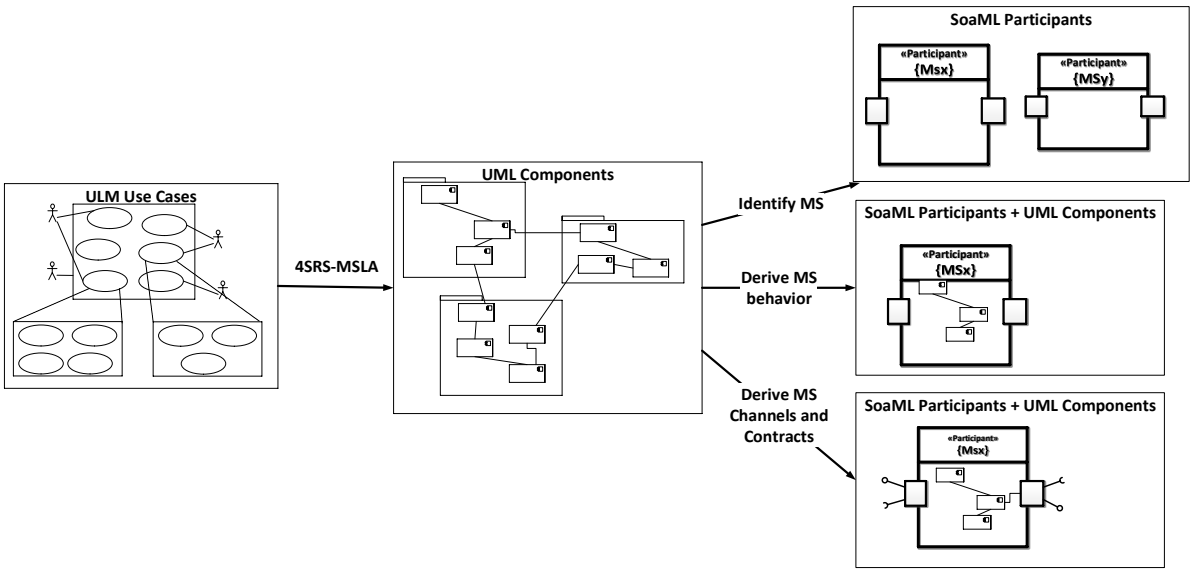

Figura 1: Especificación de microservicios utilizando el método 4SRS-MSLA.

El proceso se estructura en cuatro pasos esenciales[8]:

- Creación de Componentes: Se generan tres componentes por cada caso de uso: uno para la interfaz (i-type), otro para el modelo de datos (d-type) y otro para la lógica de control (c-type), abarcando así las funciones clave de cada microservicio.
- Eliminación de Componentes: Se revisan los componentes para eliminar redundancias. Solo se eliminan aquellos que cumplen exactamente la misma función, respetando el principio de responsabilidad única de los microservicios.
- Empaquetamiento e Identificación: Los componentes se agrupan en paquetes que representan microservicios, siguiendo la jerarquía de los casos de uso. Esto asegura coherencia con los límites definidos por los subdominios del sistema.
- Asociación entre Microservicios: Se definen los canales de comunicación entre microservicios según sus relaciones funcionales, permitiendo su colaboración sin imponer un patrón de integración específico.

4.2. Modelado de Servicios en SoaML

En esta fase, los modelos UML obtenidos mediante la aplicación de AMPLA y del método 4SRS-MSLA se utilizan para construir los diagramas SoaML y sus respectivos componentes. Este modelado permite representar tanto el comportamiento interno de los microservicios, como sus modelos de datos y los canales de comunicación existentes entre ellos.[10]

Estos aspectos se integran en distintos diagramas de SoaML, utilizando un conjunto de reglas de transición que facilitan el paso desde los elementos UML hacia sus equivalentes en SoaML. Dichas reglas se organizan en tres categorías: *Límites* (*Boundary*), *Datos* y *Comunicación*, como se muestra en la Tabla 1.[10]

Tabla 1: Reglas de transición desde UML (en AMPLA) hacia SoaML

Regla	Entrada desde UML	Salida en SoaML
1. Límite	Paquetes UML	Participantes de Servicio (*Service Participants*)
2. Límite	Paquetes UML	Diagrama de Arquitectura de Servicios
3. Límite	Componentes UML (dentro de paquetes)	Capacidades del Servicio (*Service Capabilities*)
4. Límite	Componentes tipo *i*	Aplicaciones web independientes de los participantes del servicio
1. Datos	Componentes tipo *d*	Capacidades del Servicio
1. Comunicación	Asociaciones 4SRS	Participantes del Servicio (solicitudes/servicios y puertos), Interfaces de Servicio
2. Comunicación	Secuencias UML	Interfaces de Servicio

5. Diseño de la arquitectura basada en microservicios

A continuación, de acuerdo a la metodología propuesta, se describen las actividades realizadas con el propósito de migrar una aplicación comercial monolítica a microservicios. En donde, las capacidades actuales de la aplicación brindan soporte a estaciones de servicio en las siguientes áreas de negocio: ventas, inventario, tesorería y contabilidad.

5.1. Modelado del Negocio

El propósito del modelado del negocio fue establecer una visión compartida del funcionamiento actual de la aplicación y de las empresas que hacen uso de la misma (estaciones de servicio clientes), así como definir una base sólida para el posterior levantamiento de los casos de uso. Este análisis previo no solo permitió alinear los requerimientos técnicos con las necesidades reales de los usuarios, sino también identificar fortalezas y debilidades de la aplicación actual.

Esta actividad se ejecutó realizando entrevistas y grupos focales con los stakeholders clave del sistema (expertos en el dominio y personal técnico que mantiene la aplicación actual en la empresa propietaria del software donde se ejecutó el caso de estudio, empresa que provee software a estaciones de servicio). A partir de estos encuentros y la revisión del funcionamiento de la aplicación actual, se logró identificar las principales operaciones que se realizan en cada módulo y los flujos de trabajo que las sostienen. Con el fin de facilitar la comunicación con los stakeholders, los procesos de negocio fueron descritos mediante diagramas de actividades, los cuales permitieron visualizar de forma clara las secuencias de acciones y decisiones que se toman durante la ejecución de los diferentes procesos de negocio.

El producto principal de esta actividad es el Diagrama de Flujo de Negocio (ver sección 5.1.4), que representa los procesos empresariales con sus entradas, actividades y salidas, proporcionando una visión clara de las operaciones clave del negocio.

5.1.1. Proceso de ventas en el sistema comercial

El proceso de ventas constituye uno de los pilares fundamentales en la operación de los sistemas contables empresariales, dado que involucra tanto aspectos comerciales como financieros. La Figura 2 presenta un diagrama de flujo que ilustra las etapas involucradas en una transacción de venta típica, desde la solicitud de cotización hasta la contabilización final en tesorería. Este diagrama permite visualizar los puntos de decisión críticos, las rutas divergentes según el tipo de producto y los módulos contables afectados durante el proceso.

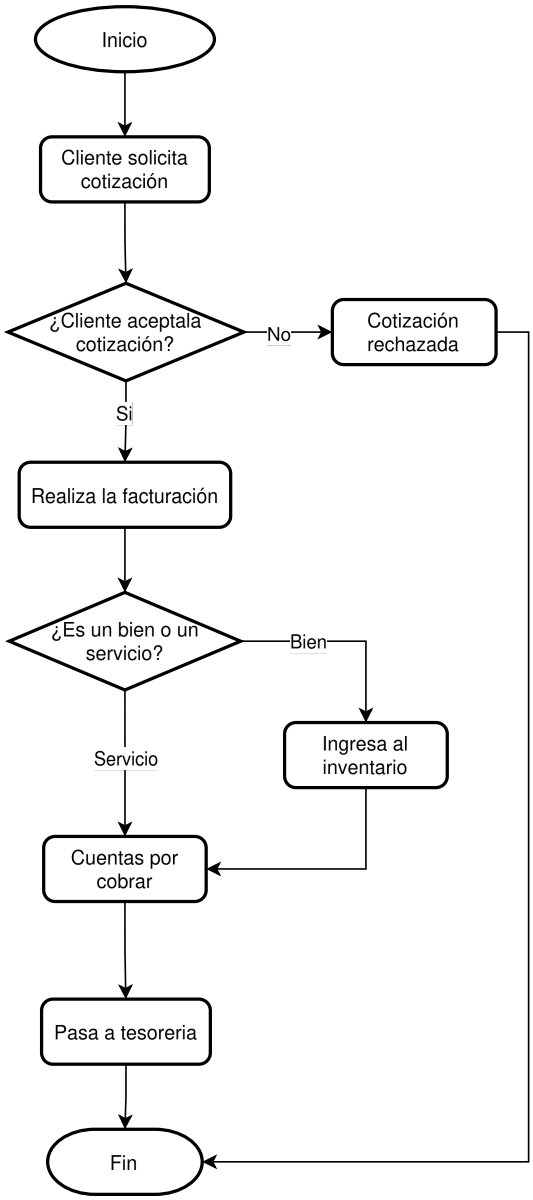

Figura 2: Diagrama de flujo del proceso de ventas en el sistema comercial

A continuación, se explican los principales elementos que intervienen en el flujo de ventas:

- Inicio: marca el punto de partida del proceso comercial, donde se inicia la interacción entre el cliente y la empresa.
- Solicitud de cotización: el cliente presenta una solicitud para conocer precios, condiciones y características de los bienes o servicios ofrecidos. Este paso no representa aún un compromiso contractual.
- Decisión del cliente sobre la cotización: se determina si el cliente acepta o rechaza la oferta presentada. En caso negativo, el proceso concluye con una cotización rechazada; en caso afirmativo, se procede con la facturación.

- Cotización rechazada: representa la terminación del proceso por decisión del cliente. Puede registrarse para fines analíticos y de retroalimentación comercial.
- Facturación: se emite la factura correspondiente, la cual documenta legalmente la transacción comercial y detalla los productos o servicios vendidos, precios, impuestos y condiciones de pago.
- Clasificación del producto: se establece si el objeto de la venta corresponde a un bien físico o a un servicio. Esta distinción determina el tratamiento posterior dentro del sistema.
- Ingreso al inventario: en el caso de bienes, el sistema actualiza el inventario registrando la salida de productos. Este paso garantiza el control físico y contable de los activos.
- Cuentas por cobrar: se registra la obligación del cliente de efectuar el pago, en caso de ventas a crédito. Este módulo representa un activo dentro de los estados financieros.
- Tesorería: una vez recibido el pago, el ingreso se refleja en tesorería, cerrando el ciclo de la venta desde el punto de vista financiero.
- Fin: el proceso concluye con el cumplimiento de todas las etapas operativas y contables involucradas.

5.1.2. Proceso de compras en el sistema comercial

El proceso de compras representa un conjunto de actividades sistematizadas orientadas a la adquisición de bienes, servicios o activos necesarios para el funcionamiento de una organización y que debera soportar la aplicación. La Figura 3 muestra el flujo típico en un sistema comercial, desde la decisión de compra hasta el registro contable y financiero. Esta representación permite visualizar los puntos de control, decisiones críticas y los módulos implicados, tales como inventario, activos fijos, cuentas por pagar y tesorería.

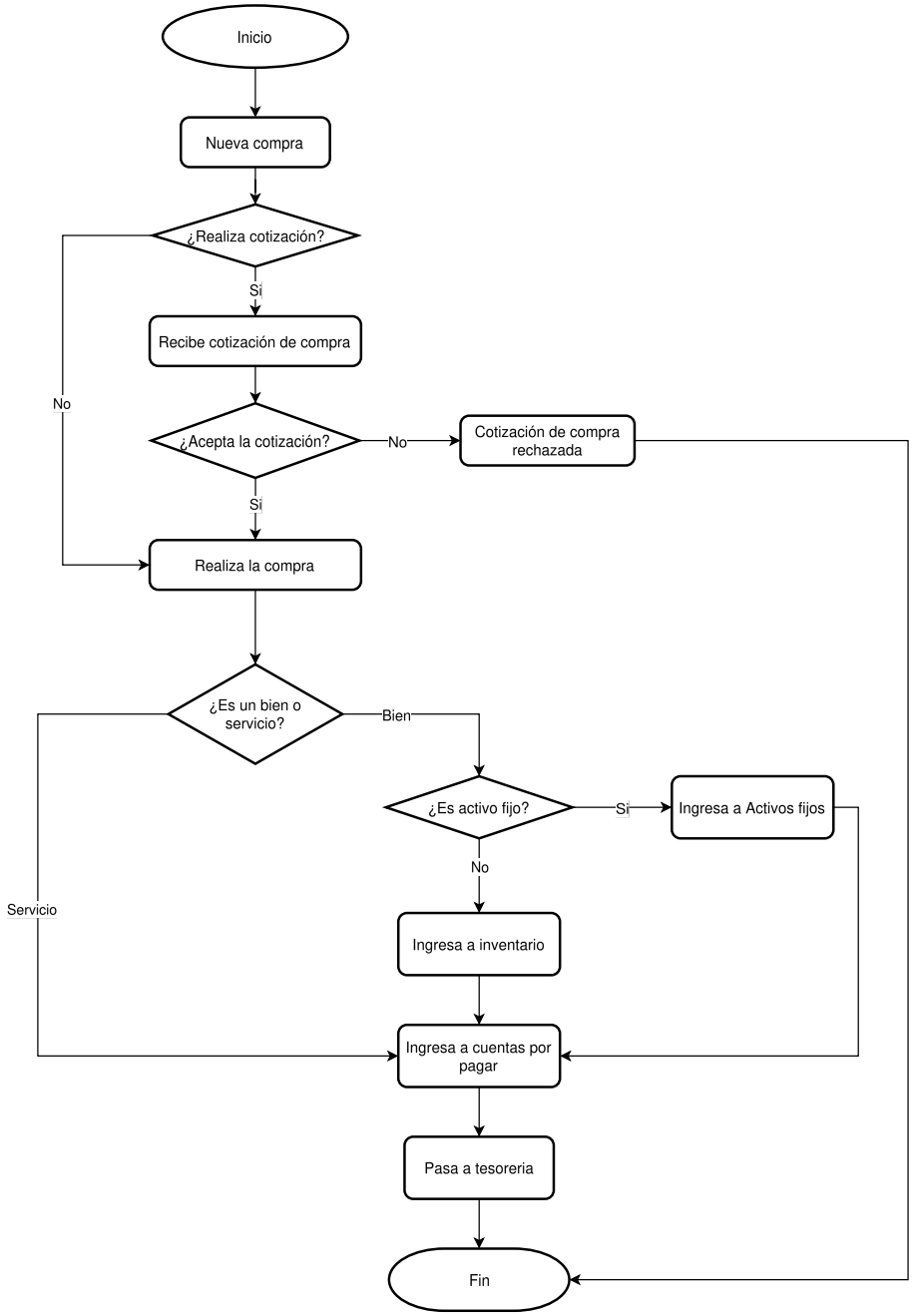

Figura 3: Diagrama de flujo del proceso de compras locales

A continuación, se describen los principales elementos de este proceso:

- Inicio: señala el punto en que se identifica una nueva necesidad de compra en la organización.
- Nueva compra: se establece formalmente el requerimiento de adquirir un bien, servicio o activo.
- ¿Realiza cotización?: determina si el proceso requiere consulta previa a proveedores.

Si no se realiza, puede continuar directamente a la compra.

- Recibe cotización de compra: se obtienen ofertas de uno o más proveedores para evaluar precios, condiciones y plazos.
- ¿Acepta la cotización?: implica una evaluación y decisión sobre la oferta recibida. Si es rechazada, el proceso concluye.
- Cotización de compra rechazada: registro de una oferta no aceptada por no cumplir condiciones comerciales o técnicas.
- Realiza la compra: formalización del compromiso de adquisición con un proveedor seleccionado.
- ¿Es un bien o servicio?: clasificación del objeto de la compra, lo cual determina su ruta de tratamiento contable.
- ¿Es activo fijo?: si es un bien, se evalúa si cumple con los criterios para ser considerado activo fijo (vida útil prolongada, inversión significativa, etc.).
- Ingresa a activos fijos: registro contable del bien como inversión de largo plazo dentro del patrimonio de la empresa.
- Ingresa a inventario: si el bien no es un activo fijo, se incorpora al inventario disponible de la organización.
- Ingresa a cuentas por pagar: se registra la obligación financiera hacia el proveedor por el bien o servicio recibido.
- Pasa a tesorería: cuando la obligación se cancela, el egreso es reflejado en el módulo de tesorería, cerrando el proceso desde el punto de vista financiero.
- Fin: indica que el proceso ha concluido, con todos los registros administrativos y contables correctamente actualizados.

5.1.3. Proceso de importaciones en el sistema comercial

El proceso de importaciones en un sistema comercial permite registrar la adquisición de bienes provenientes del extranjero, cumpliendo con aspectos legales, logísticos y financieros. Este proceso integra distintos módulos del sistema, como cuentas por pagar, activos fijos, inventario y tesorería. La Figura 4 muestra el diagrama de flujo correspondiente, que contempla tanto las decisiones clave como los registros contables necesarios en cada etapa.

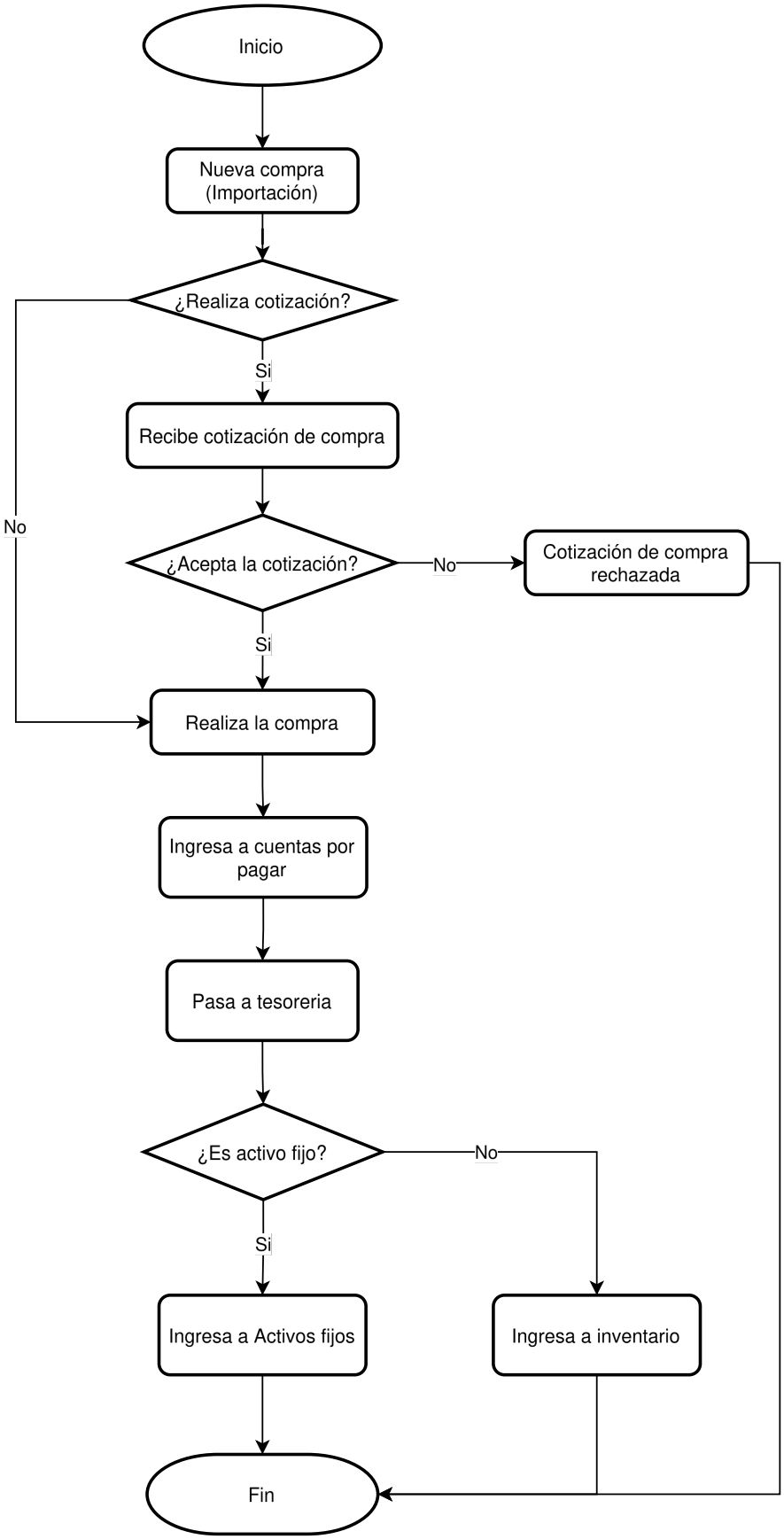

Figura 4: Diagrama de flujo del proceso de importaciones

A continuación, se describen los principales elementos de este flujo:

- Inicio: marca el comienzo del procedimiento de importación, originado por la necesidad de adquirir un bien del exterior.
- Nueva compra (importación): representa la solicitud interna de adquisición de un producto importado, iniciando el proceso formal.
- ¿Realiza cotización?: evalúa si es necesario solicitar propuestas a proveedores internacionales. En caso contrario, se procede directamente a la compra.
- Recibe cotización de compra: se obtienen los detalles de la oferta del proveedor, incluyendo precios, condiciones y términos de entrega.
- ¿Acepta la cotización?: punto de decisión sobre la aceptación de las condiciones presentadas. Si la cotización es rechazada, el proceso finaliza.
- Cotización de compra rechazada: documento que deja constancia del rechazo de la oferta por parte de la empresa.
- Realiza la compra: implica la formalización del pedido, el contrato de importación o la emisión de la orden de compra.
- Ingresa a cuentas por pagar: se registra la obligación financiera contraída con el proveedor internacional.
- Pasa a tesorería: una vez que se efectúa el pago, la salida de fondos se contabiliza en el módulo de tesorería.
- ¿Es activo fijo?: se analiza si el bien adquirido cumple los criterios para ser registrado como activo fijo (vida útil, inversión relevante, uso prolongado).
- Ingresa a activos fijos: si se clasifica como activo fijo, se registra como inversión a largo plazo.
- Ingresa a inventario: si no cumple con los criterios de activo fijo, se considera una incorporación al inventario operativo.
- Fin: el proceso concluye con el registro de la transacción y la afectación contable correspondiente.

5.1.4. Diagrama de flujo del dominio

La Figura 5 presenta un diagrama de modelo que describe las interacciones entre los principales procesos en el sistema actual de contabilidad de la empresa. Este diagrama permite identificar los módulos clave que forman parte de las operaciones, lo cual resulta esencial para estructurar una futura arquitectura basada en microservicios.

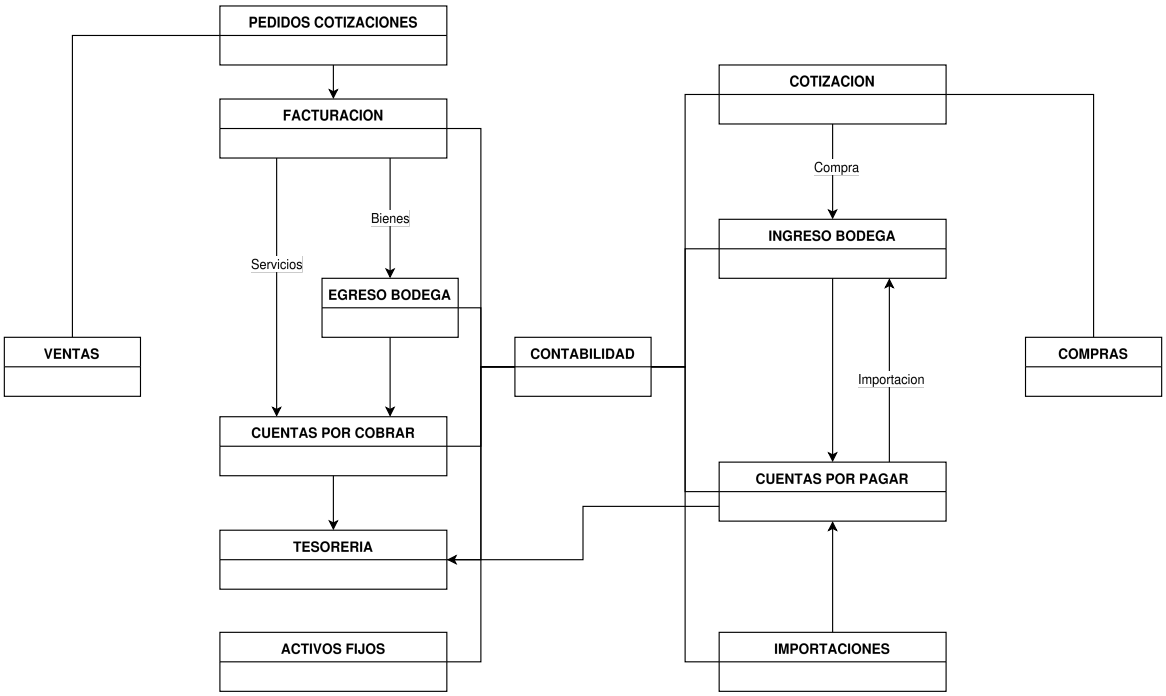

Figura 5: Diagrama funcional de procesos comerciales

A continuación, se describen los principales módulos del sistema:

- Cotización de venta: Documento que presenta a un cliente potencial los precios y condiciones de los bienes o servicios ofrecidos. No implica compromiso inmediato, pero sirve como base para formalizar una venta.
- Cotización de compra: Solicitud que un comprador envía a los proveedores para obtener información sobre los precios, plazos y condiciones de los bienes o servicios que desea adquirir.
- Ventas: Registro de las transacciones en las que la empresa entrega bienes o presta servicios a cambio de dinero u otro beneficio económico. Es una fuente clave de ingresos.
- Compras: Registro de las adquisiciones de bienes, materiales o servicios necesarios para el funcionamiento de la empresa. Representan un gasto o una inversión en activos.
- Facturación: Proceso de emisión de facturas, que son documentos legales que detallan las transacciones comerciales, especificando los productos o servicios vendidos, sus precios, impuestos aplicables y condiciones de pago.
- Inventario: Conjunto de bienes o productos que la empresa tiene en existencia para su uso, venta o transformación. Es un activo circulante que debe gestionarse eficientemente para evitar pérdidas o excesos.

- Cuentas por cobrar: Registro de las deudas que los clientes tienen con la empresa por productos o servicios adquiridos a crédito. Representan un activo en los estados financieros.
- Cuentas por pagar: Registro de las obligaciones financieras de la empresa con proveedores u otras partes por bienes y servicios adquiridos a crédito. Representan un pasivo.
- Tesorería: Área responsable de la gestión y control del efectivo, inversiones y liquidez de la empresa. Su objetivo es garantizar la solvencia para cumplir con los compromisos financieros.
- Activos fijos: Bienes tangibles que la empresa utiliza en sus operaciones durante un periodo prolongado, como edificios, maquinaria, vehículos o equipos. Representan inversiones de largo plazo.
- Contabilidad: Sistema estructurado de registro, análisis e interpretación de las transacciones financieras de una organización, con el objetivo de proporcionar información útil para la toma de decisiones y el cumplimiento de normativas.
- Importaciones: Adquisición de bienes o servicios provenientes de otros países para consumo, reventa o uso en los procesos de producción de la empresa. Implica costos adicionales como aranceles e impuestos.

5.2. Justificación de la elección de la arquitectura basada en microservicios tras el modelado del negocio

Una vez concluido el modelado del negocio, se identificaron con claridad los distintos procesos empresariales que conforman el sistema comercial, tales como ventas, compras, contabilidad, inventario, importaciones, cuentas por pagar, cuentas por cobrar y tesorería. Estos procesos, aunque interrelacionados, presentan una lógica de negocio propia, flujos independientes y objetivos específicos. Esta realidad evidenció la necesidad de adoptar una arquitectura que permitiera reflejar de forma fiel dicha modularidad operativa.

En este contexto, la arquitectura basada en microservicios fue seleccionada como solución idónea, ya que permite descomponer la aplicación en servicios independientes y especializados, alineados con los límites funcionales detectados durante el modelado. Esta decisión garantiza una mayor cohesión interna dentro de cada módulo, un bajo acoplamiento entre componentes, y facilita la evolución individual de cada servicio sin afectar el sistema en su conjunto.

Además, esta arquitectura permite escalar de manera selectiva los servicios más demandados (por ejemplo, ventas o inventario), optimizando el uso de recursos y reduciendo costos operativos. También favorece la mantenibilidad del sistema, ya que cada microservicio puede ser desarrollado, probado, desplegado y actualizado de forma autónoma. En consecuencia, la arquitectura de microservicios no solo responde a las necesidades técnicas derivadas del

análisis del dominio, sino que también aporta beneficios tangibles en términos de flexibilidad, escalabilidad y sostenibilidad del sistema a largo plazo.

5.3. Levantamiento de Casos de Uso de primer nivel

Una vez definidos los procesos del negocio, se procedió con el levantamiento de casos de uso. El objetivo de esta etapa fue identificar cómo los distintos usuarios interactúan con el sistema para cumplir tareas específicas, conectando de manera directa las necesidades operativas del negocio con las funcionalidades que debe ofrecer la solución tecnológica.

Durante esta fase, se identificaron tanto actores internos como externos, con base en los roles definidos en los procesos modelados previamente. A partir de esta identificación, se documentaron los casos de uso asociados a cada uno de ellos, considerando tanto flujos básicos como escenarios alternativos que podrían surgir durante la operación del sistema.

Para describir estas interacciones se utilizaron diagramas de casos de uso y plantillas estructuradas definidas con este propósito. Los diagramas permitieron representar de manera clara las relaciones entre los actores y los distintos casos de uso del sistema, mientras que las plantillas permitieron capturar detalles esenciales como flujos de trabajo paso a paso, condiciones previas, resultados esperados y posibles excepciones.

Como entregables, esta etapa produjo un Diagrama de Casos de Uso, que muestra gráficamente las interacciones entre los actores y el sistema, y un documento de Especificación de Casos de Uso, que describe de manera detallada los escenarios contemplados, así como las reglas que condicionan su ejecución. Estos artefactos constituyen la base para la posterior definición de componentes y servicios dentro de la arquitectura del sistema.

5.3.1. Identificación de actores

La Tabla 2 resume los actores identificados en el sistema comercial monolítico. Cada uno de ellos representa un perfil operativo dentro de la organización, con funciones específicas que definen su interacción con los módulos del sistema. Esta categorización resulta clave tanto para el modelado de casos de uso como para futuras decisiones de diseño al momento de migrar a una arquitectura basada en microservicios.

Tabla 2: Actores identificados en el sistema comercial

Actor	**Rol en el negocio**
Gestor de ventas	Supervisa la operación comercial relacionada con la venta de productos o servicios. Realiza cotizaciones, gestiona pedidos y autoriza facturación.
Gestor de compras	Encargado de iniciar procesos de adquisición, evaluar proveedores y emitir órdenes de compra. Administra las solicitudes de adquisición.
Gestor de inventarios	Controla el stock disponible, coordina el ingreso y egreso de productos, y garantiza la integridad física del inventario.
Gestor de productos	Administra la base de datos de productos. Define categorías, precios, atributos y características necesarias para su uso operativo.
Tesorero	Responsable del control de ingresos y egresos monetarios. Efectúa pagos, coordina transferencias y gestiona la caja chica y cuentas bancarias.
Contador	Administra el registro contable. Supervisa la generación de asientos, el cumplimiento tributario y la elaboración de reportes financieros.
Gestor de bancos	Realiza operaciones bancarias dentro del sistema. Gestiona conciliaciones, registros de transferencias y movimientos financieros externos.
Gestor de SRI	Ejecuta tareas relacionadas con el cumplimiento tributario: emisión de retenciones, envío de comprobantes electrónicos y declaraciones fiscales.
Administrador	Define la configuración general del sistema. Crea usuarios, asigna roles, y controla accesos según la política de seguridad organizacional.
Usuario del sistema	Representa a cualquier persona que accede al sistema para operar según el rol asignado. Utiliza funciones disponibles a través de autenticación.

5.3.2. Diagrama de casos de uso de primer nivel

La Figura 6 representa el nuevo diagrama de casos de uso de primer nivel correspondiente a los subsistemas del sistema comercial. Cada actor actúa sobre módulos específicos que agrupan funcionalidades coherentes con su rol, lo que favorece un análisis posterior orientado a la segmentación de servicios y componentes reutilizables.

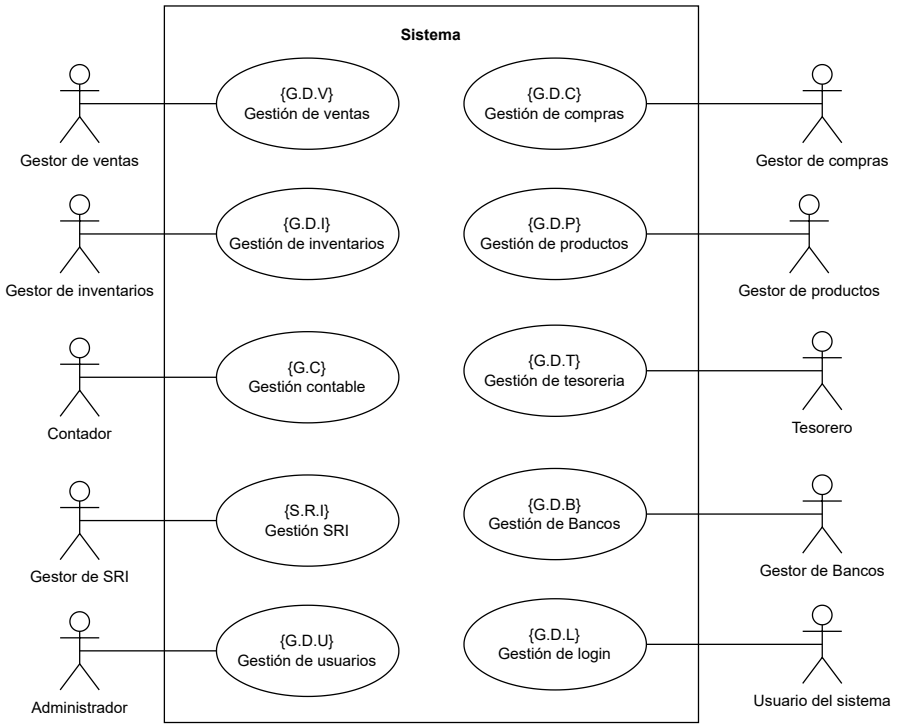

Figura 6: Diagrama de casos de uso de primer nivel del sistema comercial

La lista que se detalla a continuación expone los principales casos de uso funcionales presentes en el sistema, junto con una descripción general de su propósito:

- Gestión de ventas (G.D.V): comprende las operaciones asociadas a las actividades comerciales de salida, tales como la generación de cotizaciones, facturación y registro de ventas.
- Gestión de compras (G.D.C): permite al usuario gestionar el ciclo completo de adquisiciones, incluyendo solicitudes, órdenes de compra y relaciones con proveedores.
- Gestión de inventarios (G.D.I): orientada al control de existencias y movimientos de productos dentro de los almacenes o bodegas, incluyendo ajustes manuales y operaciones de ingreso/egreso.
- Gestión de productos (G.D.P): centraliza la administración de los datos maestros de productos, categorías, precios y configuraciones necesarias para otros procesos.
- Gestión contable (G.C): núcleo del sistema desde el punto de vista financiero, permite la creación y consulta de asientos contables, cierres contables y emisión de informes financieros.
- Gestión de tesorería (G.D.T): da soporte a la administración de la caja y flujos monetarios, tales como pagos a proveedores, recepción de ingresos y conciliación de saldos.

- Gestión de bancos (G.D.B): registra y controla todas las operaciones relacionadas con cuentas bancarias, como transferencias, débitos y conciliaciones bancarias.
- Gestión SRI (S.R.I): permite la emisión de comprobantes electrónicos, generación de retenciones y cumplimiento de los requerimientos tributarios definidos por la normativa local.
- Gestión de usuarios (G.D.U): ofrece las herramientas necesarias para el alta, edición, inactivación y control de permisos sobre los perfiles de usuario registrados en el sistema.
- Gestión de login (G.D.L): gestiona el acceso de los usuarios al sistema, aplicando políticas de autenticación y seguridad básicas para el inicio de sesión.

5.4. Levantamiento de casos de uso de segundo nivel

Los casos de uso de segundo nivel se obtienen al refinar una funcionalidad general del sistema (primer nivel). En ellos se describe, con mayor detalle, la interacción específica: cómo reacciona el software paso a paso ante una entrada del usuario o un evento externo. Este nivel de detalle es el que permite traducir un requisito funcional en una especificación técnica comprensible.

5.4.1. Descomposición funcional del caso de uso Gestión de Compras

La Figura 7 muestra la descomposición del caso de uso de compras (G.D.C) a un segundo nivel de análisis. Esta descomposición permite identificar los subprocesos clave que conforman la funcionalidad completa del módulo, en función de las responsabilidades del actor principal: el gestor de compras. Esta estructura facilitará, en etapas posteriores, la identificación de componentes y microservicios bajo el enfoque del método 4SRS-MSLA.

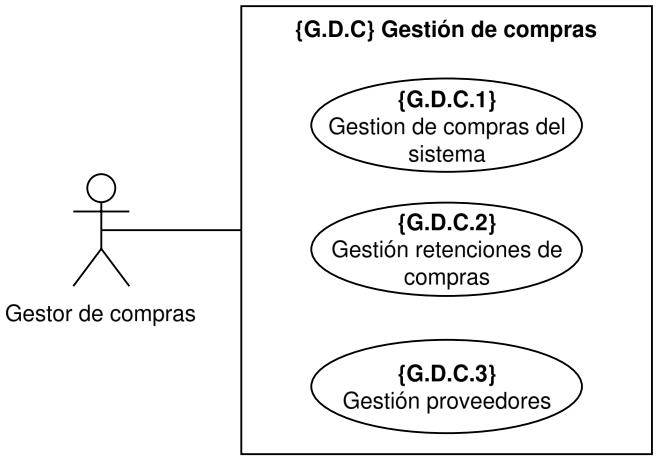

Figura 7: Casos de uso de segundo nivel del caso de uso de gestión de compras

A continuación se detallan los casos de uso específicos contenidos dentro del módulo:

- Gestión de compras del sistema (G.D.C.1): se encarga del registro y control de las órdenes de compra generadas por el área, incluyendo la validación de proveedores, fechas, productos, cantidades, precios y condiciones comerciales.
- Gestión de retenciones de compras (G.D.C.2): permite calcular, registrar y emitir los comprobantes de retención que deben generarse por obligaciones tributarias asociadas a las compras, de acuerdo con la normativa vigente.
- Gestión de proveedores (G.D.C.3): administra la información relacionada con los proveedores, incluyendo su creación, modificación, datos fiscales, contactos, cuentas bancarias y condiciones de pago establecidas.

5.4.2. Descomposición del caso de uso de Gestión de Productos

La Figura 8 presenta el desglose funcional del caso de uso Gestión de productos (G.D.P), correspondiente al segundo nivel de análisis de los casos de uso de primer nivel . En esta etapa, se identifican nuevos casos de uso orientados a la administración de atributos clave del catálogo de productos. Esta división permite observar con mayor precisión los límites lógicos del sistema y allana el camino hacia un diseño modular, particularmente útil en el contexto de migración a microservicios.

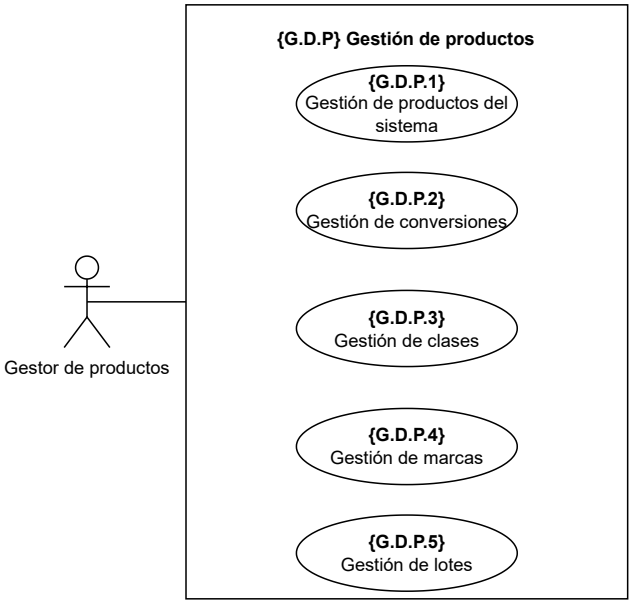

Figura 8: Casos de uso de segundo nivel del caso de uso de Gestión de Productos

A continuación, se describen las funcionalidades específicas que conforman este caso de uso:

- Gestión de productos del sistema (G.D.P.1): permite registrar, editar y consultar los productos activos dentro del sistema, incluyendo códigos, descripciones, unidades y precios.
- Gestión de conversiones (G.D.P.2): facilita la administración de equivalencias entre distintas unidades de medida de un mismo producto, lo que resulta esencial en procesos de compra, venta o inventario.
- Gestión de clases (G.D.P.3): agrupa productos por categorías funcionales o lógicas (por ejemplo, electrónicos, perecibles, repuestos), lo que optimiza la clasificación y análisis posterior.
- Gestión de marcas (G.D.P.4): permite vincular productos con marcas comerciales, manteniendo información como el proveedor principal, referencias y estrategias de posicionamiento.
- Gestión de lotes (G.D.P.5): gestiona el control por lotes de productos, habilitando el seguimiento de vencimientos, trazabilidad y agrupación por series de producción.

5.4.3. Descomposición del caso de uso de Gestión Contable

El caso de uso de Gestión contable (G.C) constituye el núcleo del sistema desde el punto de vista financiero. La Figura 9 descompone esta funcionalidad en caso de uso que responden a operaciones estructurales de la contabilidad empresarial, todas ellas bajo la responsabilidad del actor ”Contador”. Este nivel de análisis permite identificar los elementos que deben ser tratados como componentes independientes o servicios especializados en el diseño arquitectónico del sistema.

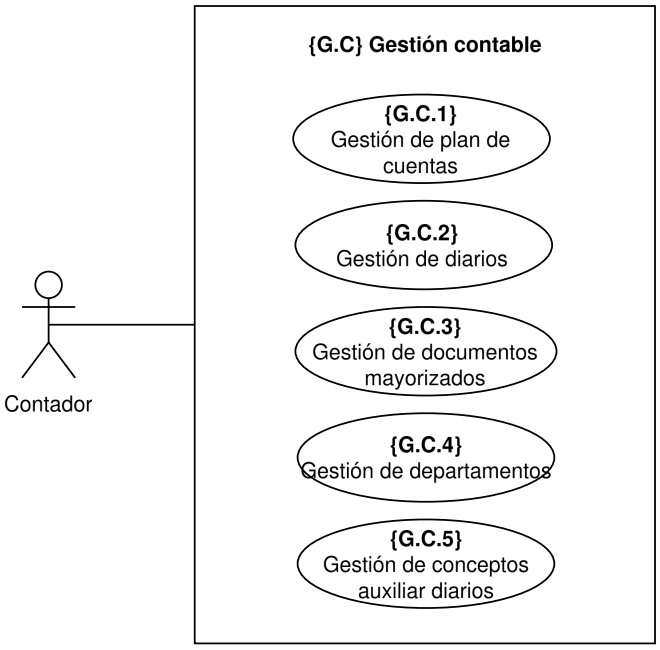

Figura 9: Casos de uso de segundo nivel del caso de uso de gestión contable

Los casos de uso identificados en este caso de uso se describen a continuación:

- Gestión de plan de cuentas (G.C.1): permite la creación y organización jerárquica de cuentas contables, definiendo su codificación, naturaleza y estructura contable.
- Gestión de diarios (G.C.2): comprende el registro manual o automático de transacciones financieras mediante asientos contables que afectan una o más cuentas del plan.
- Gestión de documentos mayorizados (G.C.3): permite consultar, validar y cerrar documentos contables que ya han sido procesados, facilitando su trazabilidad y auditoría.
- Gestión de departamentos (G.C.4): habilita la asignación contable de registros a diferentes unidades organizativas, con fines de control interno y análisis de costos.
- Gestión de conceptos auxiliar diarios (G.C.5): permite definir y mantener un catálogo de conceptos que complementan la clasificación de los asientos, facilitando reportes analíticos más detallados.

5.5. Levantamiento de casos de uso de tercer nivel

Para alcanzar un nivel de detalle que facilite la implementación, los casos de uso del sistema se refinan en un tercer nivel. Estos casos de uso granulares describen las tareas atómicas y subfuncionalidades que, en conjunto, materializan una interacción más compleja. Su propósito es definir acciones tan específicas que puedan ser asignadas y desarrolladas como funcionalidades concretas dentro de los componentes individuales del software.

5.5.1. Casos de uso de tercer nivel obtenidos del caso de uso Gestión de compras del sistema

La Figura 10 ilustra la descomposición de tercer nivel del caso de uso Gestión de compras del sistema (G.D.C.1), correspondiente al módulo de compras. En este nivel se identifican las operaciones específicas que realiza el Gestor de compras para registrar, modificar, consultar y gestionar documentos de compra. Este tipo de análisis permite establecer una base clara para la futura transformación de cada funcionalidad en servicios independientes o componentes reutilizables.

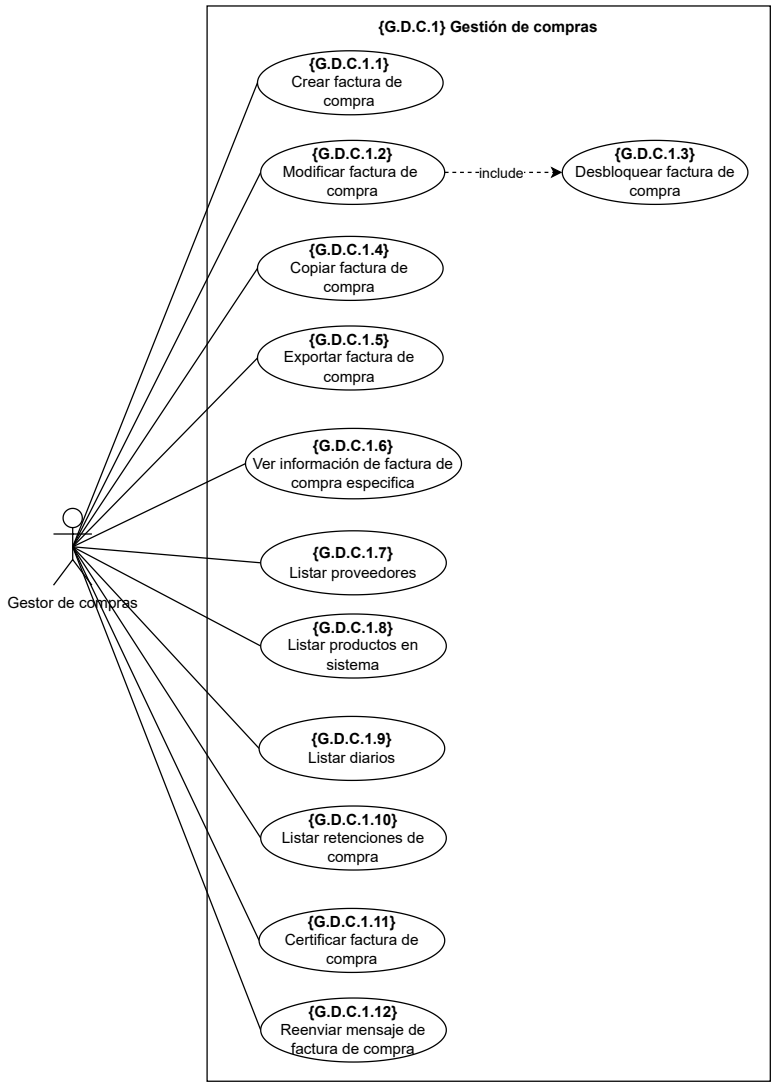

Figura 10: Casos de uso de tercer nivel del caso de uso G.D.C.1 Gestión de compras del sistema

A continuación se detallan los casos de uso incluidos en esta descomposición:

- Crear factura de compra (G.D.C.1.1): permite registrar una nueva factura en el sistema, incluyendo datos de proveedor, productos adquiridos, totales e impuestos.
- Modificar factura de compra (G.D.C.1.2): habilita al usuario para editar facturas previamente registradas. Este caso de uso requiere incluir la funcionalidad para desbloquear facturas si estas se encuentran cerradas (G.D.C.1.3).
- Desbloquear factura de compra (G.D.C.1.3): permite habilitar una factura bloqueada para su edición, asegurando que el cambio esté debidamente autorizado o controlado.
- Copiar factura de compra (G.D.C.1.4): genera una nueva factura basada en los datos de una factura ya existente, facilitando procesos recurrentes o similares.
- Exportar factura de compra (G.D.C.1.5): posibilita la descarga o envío de la factura en formatos digitales como PDF o XML, para ser almacenada o enviada por medios electrónicos.
- Ver información de factura de compra específica (G.D.C.1.6): muestra en pantalla todos los detalles de una factura seleccionada, incluyendo líneas de productos, impuestos aplicados y estado.
- Listar proveedores (G.D.C.1.7): muestra un catálogo de proveedores vinculados al sistema, para facilitar la referencia cruzada con los documentos de compra cuando sea necesario.
- Listar productos en sistema (G.D.C.1.8): despliega la lista de productos registrados en el catálogo, lo cual permite su selección durante el proceso de facturación.
- Listar diarios (G.D.C.1.9): ofrece acceso a los diarios contables vinculados a las facturas de compra para efectos de revisión o auditoría.
- Listar retenciones de compra (G.D.C.1.10): muestra las retenciones aplicadas en cada factura, facilitando el seguimiento fiscal y contable de dichas operaciones.
- Certificar factura de compra (G.D.C.1.11): emite la certificación formal de una factura cuando ha cumplido con todos los controles requeridos, quedando lista para su registro contable definitivo.
- Reenviar mensaje de factura de compra (G.D.C.1.12): permite reenviar la notificación electrónica de una factura a un proveedor o al sistema tributario, en caso de fallos o solicitudes de reenvío.

5.5.2. Casos de uso de tercer nivel obtenidos del caso de uso Gestión de productos del sistema

En la Figura 11, se presenta la versión depurada del caso de uso Gestión de productos del sistema (G.D.P.1), parte de la funcionalidad operada por el actor Gestor de productos. El diagrama refleja las acciones esenciales que permiten gestionar el ciclo de vida de los productos

registrados en el sistema, desde su creación hasta su análisis comercial. Este nivel de detalle favorece la detección de funciones repetitivas y posibles puntos de refactorización en un futuro diseño por servicios.

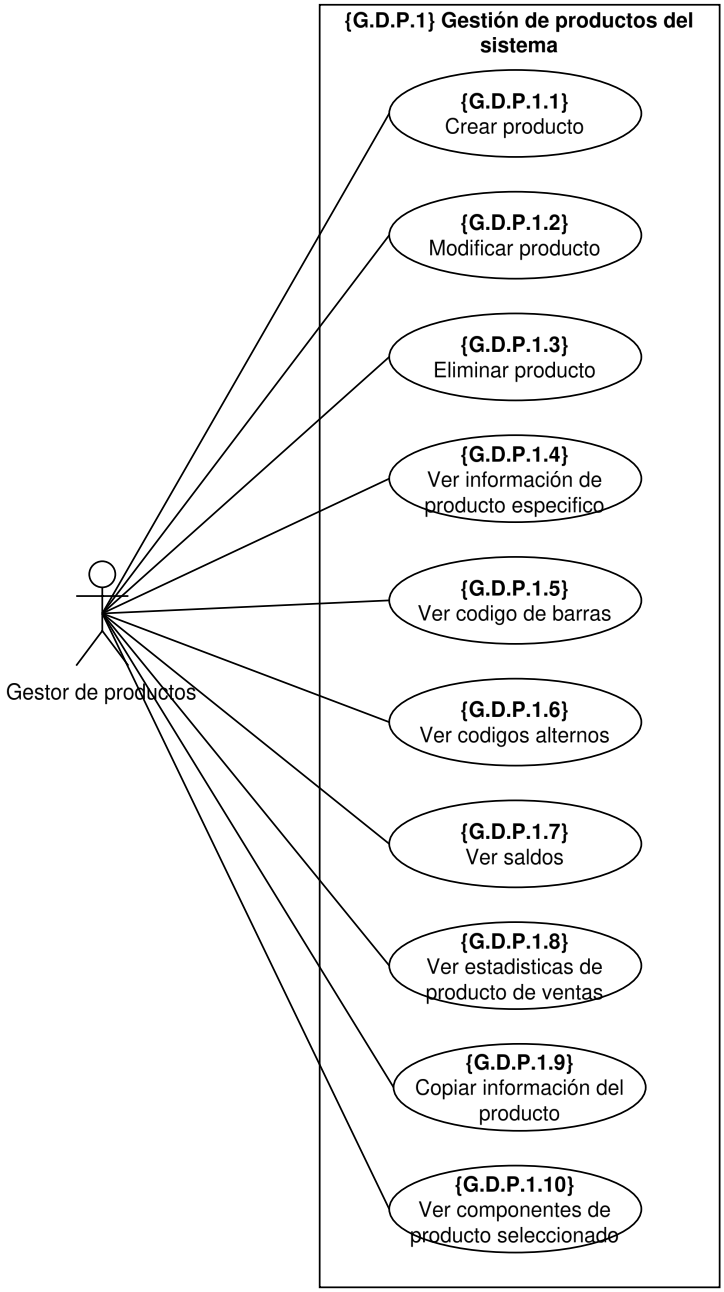

Figura 11: Casos de uso actualizados del caso de uso G.D.P.1 Gestión de productos del sistema

A continuación, se detallan los casos de uso comprendidos en esta nueva versión:

- Crear producto (G.D.P.1.1): operación básica para el registro de un nuevo artículo dentro del catálogo del sistema, con sus propiedades generales.

- Modificar producto (G.D.P.1.2): permite actualizar información de productos existentes, como nombre, presentación, precios o unidad de medida.
- Eliminar producto (G.D.P.1.3): acción que inactiva o elimina un registro de producto, dependiendo de si ya ha sido utilizado en procesos.
- Ver información de producto específico (G.D.P.1.4): consulta detallada del producto seleccionado, incluyendo historial de modificación y atributos clave.
- Ver código de barras (G.D.P.1.5): acceso al código de barras asignado al producto, utilizado comúnmente para integración con puntos de venta.
- Ver códigos alternos (G.D.P.1.6): muestra identificadores secundarios, como referencias de proveedor o códigos internos utilizados por otras áreas.
- Ver saldos (G.D.P.1.7): permite revisar las existencias actuales del producto en todas las bodegas o puntos de almacenamiento registrados.
- Ver estadísticas de producto de ventas (G.D.P.1.8): despliega un resumen de indicadores de comportamiento del producto en el mercado.
- Copiar información del producto (G.D.P.1.9): genera un nuevo producto a partir de los datos de otro existente, optimizando el registro de variantes.
- Ver componentes de producto seleccionado (G.D.P.1.10): muestra si el producto es parte de una estructura compuesta o tiene una relación con otros artículos.

5.5.3. Casos de uso de tercer nivel obtenidos del caso de uso Gestión de plan de cuentas

La Figura 12 representa los casos de uso del caso de uso Gestión de plan de cuentas (G.C.1), el cual forma parte del módulo contable general. El actor principal, el Contador, accede a diversas funcionalidades que le permiten construir, mantener y consultar la estructura contable base del sistema. Este conjunto de acciones refleja operaciones propias de la gestión financiera estructural, necesarias para el registro y análisis de transacciones.

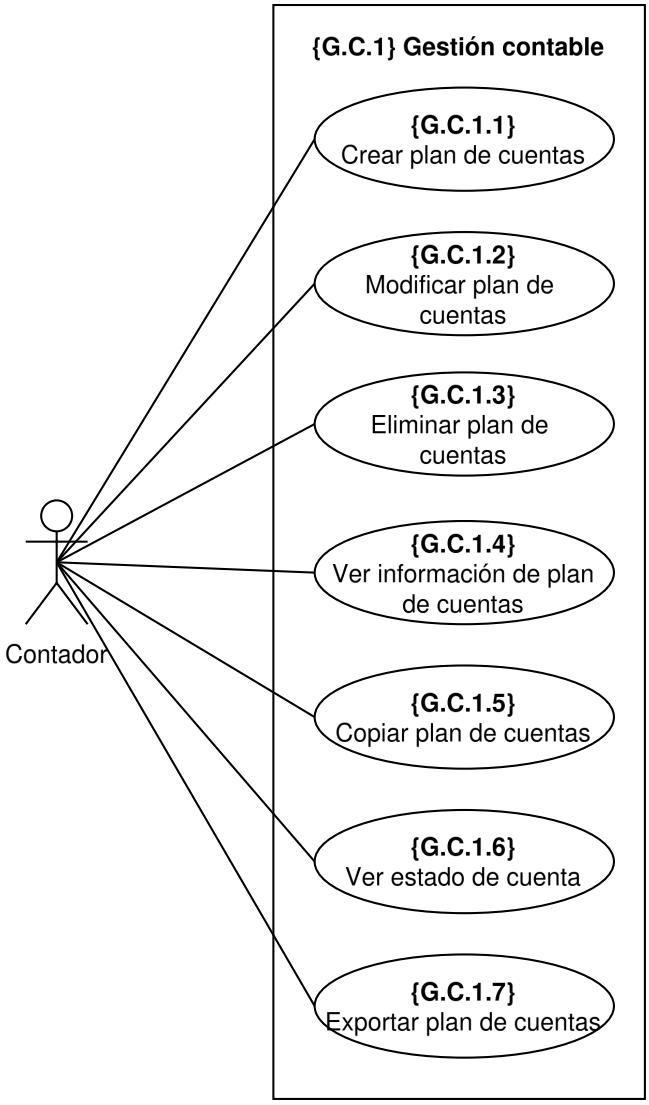

Figura 12: Casos de uso de tercer nivel del caso de uso G.C.1 Gestión de plan de cuentas

Los casos de uso identificados en este nivel son los siguientes:

- Crear plan de cuentas (G.C.1.1): permite construir una nueva jerarquía de cuentas contables, definiendo códigos, niveles, clasificaciones y tipo de cuenta.
- Modificar plan de cuentas (G.C.1.2): habilita al usuario para editar la estructura de un plan de cuentas existente, aplicando cambios en nombres, naturaleza o codificación.
- Eliminar plan de cuentas (G.C.1.3): elimina total o parcialmente un plan contable, dependiendo de si está en uso o relacionado con registros históricos.
- Ver información de plan de cuentas (G.C.1.4): consulta detallada de una cuenta o grupo de cuentas, mostrando atributos como saldo inicial, clasificación o vínculos con transacciones.

- Copiar plan de cuentas (G.C.1.5): permite duplicar un plan de cuentas completo o por secciones, útil para generar estructuras similares entre empresas o periodos fiscales.
- Ver estado de cuenta (G.C.1.6): muestra el movimiento contable acumulado de una cuenta específica, con su saldo actualizado y sus afectaciones por diarios contables.
- Exportar plan de cuentas (G.C.1.7): genera un archivo en formato externo (por ejemplo, Excel o PDF) para respaldar o compartir la estructura del plan contable.

5.5.4. Refinamiento del caso de uso de tercer nivel

Como parte del análisis detallado dentro del proceso 4SRS-MSLA, se ha realizado el refinamiento de los casos de uso de tercer nivel, como ejemplo tomamos el caso de uso Crear factura de compra (G.D.C.1.1), que forma parte del módulo de gestión de compras. Esta actividad consiste en descomponer la funcionalidad hasta identificar todos los datos necesarios para su ejecución, permitiendo entender con claridad qué información se manipula, cómo se estructura, y qué relaciones existen con otros módulos del sistema.

En la Figura 13, se muestra el resultado de este análisis en el entorno de modelado. Se observan anotaciones realizadas durante las entrevistas con los stakeholders. Estas notas no forman parte del caso de uso, sino que son utilizadas por el analista de software o ingeniero de requisitos para documentar información que será útil en etapas posteriores del desarrollo del caso de uso como se ve en la Sección 5.6 y en etapas posteriores del desarrollo del caso de uso.

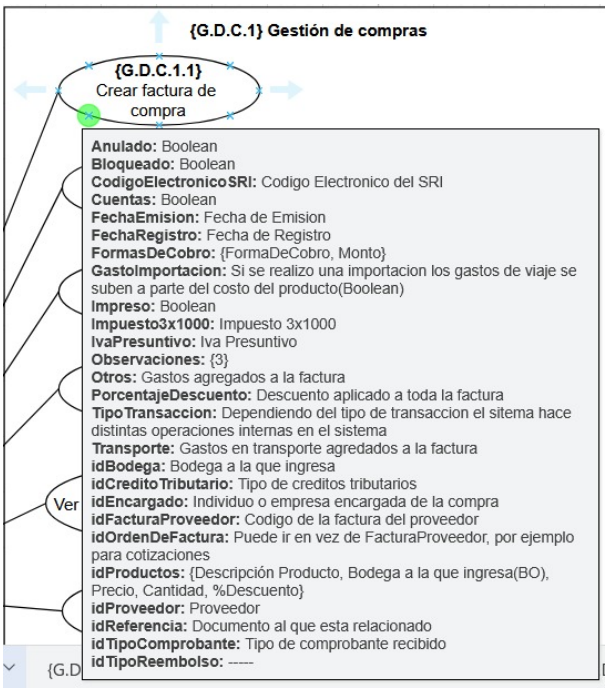

Figura 13: Refinamiento del caso de uso G.D.C.1.1 Crear factura de compra

Se observa una lista completa de campos, sus tipos de datos, y algunos indicadores sobre interacciones con otras entidades del sistema. Esta representación es clave no solo para la implementación posterior, sino también para definir los componentes ‘d‘ (datos) y ‘c‘ (control) que serán utilizados en la arquitectura modular.

La Tabla 3 resume la información más relevante asociada a esta operación, identificando el nombre del campo, su tipo de dato y una breve descripción funcional.

Tabla 3: Campos refinados del caso de uso G.D.C.1.1 Crear factura de compra

Nombre del Campo	**Tipo de Dato**	**Descripción**
Anulado	Boolean	Indica si la factura ha sido anulada.
Bloqueado	Boolean	Determina si la factura está bloqueada para edición.
CodigoElectronicoSRI	Texto	Código electrónico asignado por el SRI.
Cuentas	Boolean	Define si la factura afecta cuentas contables.
FechaEmision	Fecha	Fecha oficial de emisión de la factura.
FechaRegistro	Fecha	Fecha en que se registró la factura en el sistema.
FormasDeCobro	Lista	Métodos de cobro utilizados junto con sus montos.
GastoImportacion	Boolean	Indica si se agregan gastos de importación al producto.
Impreso	Boolean	Define si el documento fue impreso o digital.
Impuesto3x1000	Decimal	Valor correspondiente al impuesto 3x1000.
IvaPresuntivo	Decimal	Valor del IVA estimado aplicado al total.
Observaciones	Texto	Notas o comentarios agregados a la factura.
Otros	Decimal	Gastos adicionales no categorizados.
PorcentajeDescuento	Decimal	Descuento porcentual aplicado al total.
TipoTransaccion	Texto	Tipo de transacción registrada en el sistema.
Transportes	Texto	Costos asociados al transporte de la mercadería.
idBodega	Texto	Identificador de la bodega de destino.
idCreditoTributario	Texto	Código del crédito tributario usado.
idEncargado	Texto	Identificador del responsable de la factura.
idFacturaProveedor	Texto	Número o código de la factura del proveedor.
idOrdenDeFactura	Texto	Identificador de la orden que generó la factura.
idProductos	Lista	Listado de productos, cantidades y precios.
idRelacion	Texto	Documento relacionado o entidad vinculada.
idTipoComprobante	Texto	Tipo de comprobante emitido (e.g. factura, nota débito).
idTipoReembolso	Texto	Tipo de reembolso, si aplica.

5.6. Separación de Componentes

Este paso se centra en la descomposición del sistema en componentes funcionales autónomos. Para ello, se adhiere al Principio de Responsabilidad Única (SRP), asignando responsabilidades claras mediante el mapeo de casos de uso a componentes y la minimización de sus interdependencias. El método 4SRS-MLSA se utiliza para guiar esta segmentación, asegurando que los límites de cada componente estén bien definidos y alineados con los dominios

de negocio. Esto promueve tanto la independencia de los componentes como su relevancia para el negocio.

5.6.1. Identificación de interacciones entre casos de uso

Como parte del proceso posterior al refinamiento funcional, se realizó una identificación detallada de las interacciones necesarias entre casos de uso. Este análisis permite reconocer qué módulos externos interactúan con el módulo de Gestión de Compras, tanto desde la perspectiva de consumo de servicios (casos de uso que llaman a este) como desde la de proveedores (casos de uso a los que éste llama).

5.6.2. Interacciones funcionales entre casos de uso en el módulo (G.D.C) Gestión de Compras

Una vez realizado el refinamiento de los casos de uso del módulo Gestión de Compras, se identificaron las dependencias funcionales necesarias para ejecutar correctamente sus operaciones. En el contexto de una arquitectura basada en microservicios, esto implica establecer claramente con qué módulos externos debe comunicarse este caso de uso, y por qué esas interacciones son fundamentales desde una perspectiva contable.

La Figura 14 muestra un diagrama de casos de uso extendido, donde se representan a la izquierda los casos de uso que invocan servicios ofrecidos por (G.D.C), y a la derecha los casos de uso externos a los que (G.D.C) realiza llamadas para completar su lógica de negocio.

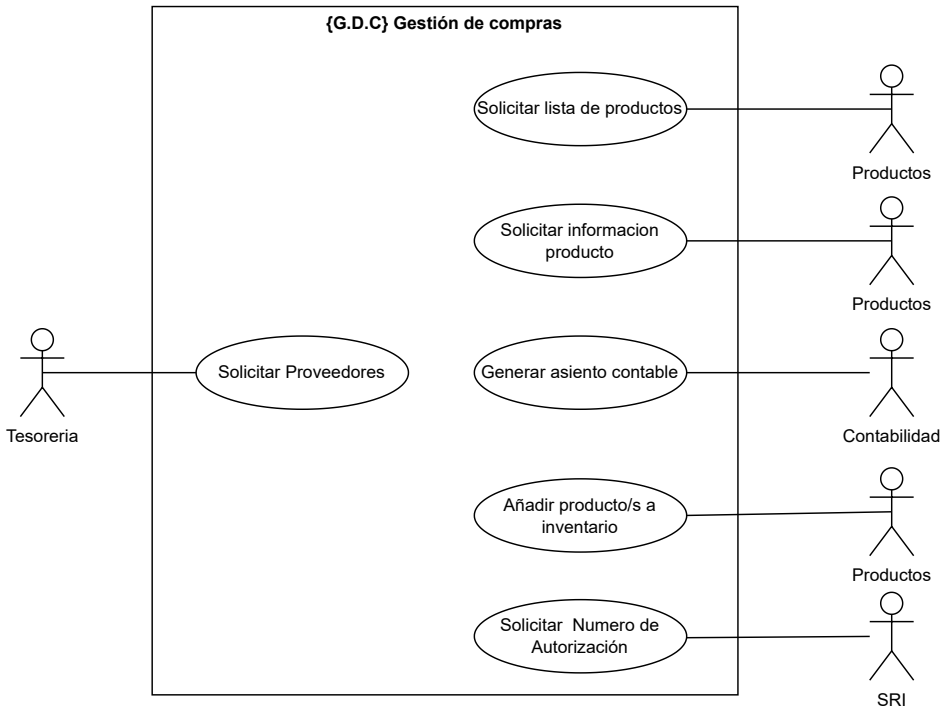

Figura 14: Diagrama de interacción funcional del caso de uso G.D.C Gestión de Compras

A continuación se describen las funcionalidades llamadas por este caso de uso y su propósito en el contexto de un sistema comercial empresarial:

- Solicitar Proveedores (Tesorería): Este método permite recuperar la lista de proveedores disponibles en el sistema. En una compra, es esencial asociar correctamente la factura a un proveedor específico, lo cual impacta directamente en la generación de cuentas por pagar y el control financiero de obligaciones.
- Solicitar lista de productos (Productos): La gestión de compras requiere acceder a productos válidos registrados previamente. Esta consulta permite garantizar que los productos ingresados en una factura estén codificados correctamente, lo cual es vital para la consistencia entre compras, inventarios y contabilidad.
- Solicitar información de producto (Productos): Este método obtiene detalles técnicos y tributarios de cada producto, como su clasificación, si aplica IVA, el tipo de unidad, entre otros. Esta información permite calcular correctamente los valores de la factura y preparar su respectiva contabilización.
- Añadir producto/s a inventario (Productos): Una vez registrada la compra, este método permite crear automáticamente las entradas de inventario asociadas. Desde el punto de vista contable, garantiza que las adquisiciones queden reflejadas no solo como un gasto, sino también como un activo disponible.

- Solicitar Número de Autorización (SRI): Toda factura de compra debe cumplir con los requisitos del Servicio de Rentas Internas. Este método asegura que el número de autorización tributaria sea válido, lo cual permite respaldar legalmente el asiento contable y evitar observaciones fiscales.
- Generar asiento contable (Contabilidad): Finalmente, esta funcionalidad permite registrar el impacto financiero de la compra. Se generan los movimientos que afectan cuentas como proveedores, inventario, IVA en compras, o gastos operativos, cumpliendo con las normativas contables y asegurando trazabilidad.

5.6.3. Interacción funcional del caso de uso (G.D.P) Gestión de productos

Una vez definidas las funcionalidades internas del caso de uso Gestión de productos, se procedió a identificar las interacciones que mantiene con otros módulos del sistema. Este análisis es esencial en el contexto de una arquitectura orientada a microservicios, donde cada servicio cumple una responsabilidad delimitada y debe interoperar de forma eficiente con los demás.

La Figura 15 presenta el diagrama de casos de uso correspondiente. En él se representan, a la izquierda, los módulos que consumen funcionalidades ofrecidas por (G.D.I), mientras que a la derecha se encuentra el módulo que recibe información proveniente de este caso de uso.

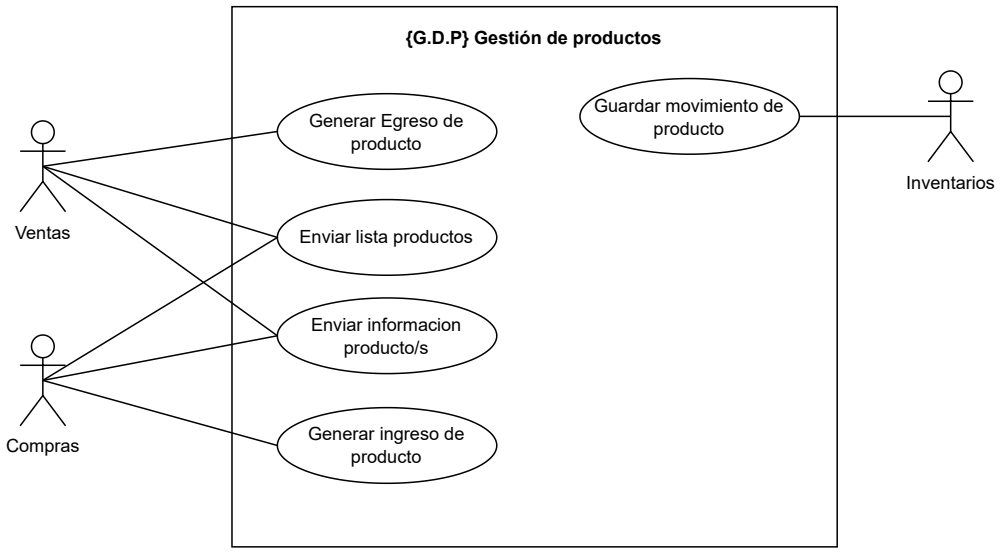

Figura 15: Diagrama de interacciones del caso de uso G.D.I Gestión de productos

A continuación se detallan los métodos identificados, junto con su propósito dentro del sistema comercial y de inventarios:

- Generar egreso de producto (Ventas): permite descontar automáticamente del inventario

los productos vendidos. Esta acción actualiza el stock y garantiza consistencia entre ventas físicas y el inventario contable.

- Enviar lista de productos (Ventas y Compras): proporciona a ambos módulos la lista de productos activos y disponibles para transacciones. Es fundamental para asegurar que los documentos de compra y venta trabajen con catálogos válidos y actualizados.
- Enviar información de producto/s (Ventas y Compras): entrega detalles técnicos y comerciales de uno o varios productos, como unidad de medida, costos, impuestos aplicables, entre otros. Esta información es necesaria para generar correctamente una factura o documento contable.
- Generar ingreso de producto (Compras): tras registrar una compra, este método permite generar automáticamente el ingreso físico al inventario. Esto garantiza que el activo adquirido esté disponible en sistema, alineando los módulos de compras e inventario.
- Guardar movimiento de producto (Inventarios): permite que el modulo de inventarios registre un nuevo movimiento (ingreso o egreso) generado por una acción del módulo de productos. Es un paso clave para mantener trazabilidad, auditoría y control del inventario desde el punto de vista contable.

5.6.4. Interacción funcional del caso de uso Gestión contable

El caso de uso Gestión contable actúa como el núcleo de integración financiera dentro del sistema. En una arquitectura contable distribuida, su rol es fundamental para garantizar la correcta generación y gestión de asientos contables, así como para facilitar la trazabilidad de todas las transacciones que afectan las finanzas de la organización.

La Figura 16 representa las interacciones que otros módulos mantienen con (G.C), indicando qué módulos hacen uso de su funcionalidad para registrar operaciones, consultar cuentas o asociar entidades contables específicas.

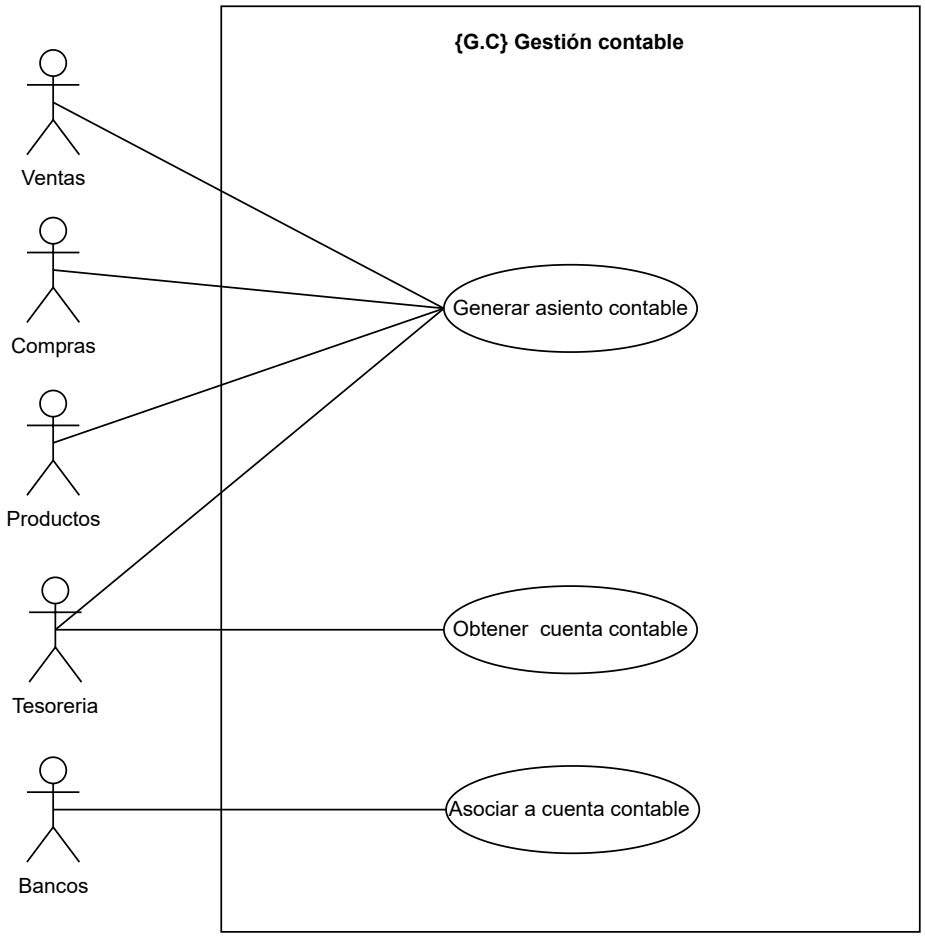

Figura 16: Interacción del caso de uso G.C Gestión contable con otros módulos

A continuación, se explican las funcionalidades identificadas y su propósito en el contexto de un sistema comercial empresarial:

- Generar asiento contable (Ventas, Compras, Productos, Tesorería): es la función central de este modulo. Permite registrar el impacto financiero de cada operación, ya sea una venta, una compra, un movimiento de inventario o un desembolso. Este método asegura que cada transacción quede reflejada en los libros contables mediante un asiento debidamente estructurado.
- Obtener cuenta contable (Tesorería): esta función permite recuperar el plan de cuentas estructurado, identificando la cuenta contable correspondiente para clasificar un pago, ingreso o movimiento financiero. Garantiza que los movimientos monetarios se asignen a las cuentas correctas.
- Asociar a cuenta contable (Bancos): facilita la vinculación de una cuenta bancaria operativa con una cuenta contable específica. Esto permite que los movimientos registrados

desde el módulo de bancos puedan integrarse automáticamente en los libros contables sin necesidad de intervención manual.

5.6.5. Identificacíon y clasificación de los componentes derivados de los casos de uso

Durante la aplicación del proceso 4SRS-MSLA, una de las actividades fundamentales consiste en la identificación y clasificación de los componentes derivados de los casos de uso. Esta etapa permite refinar la arquitectura lógica a través de una descomposición progresiva de las funcionalidades, con el objetivo de facilitar la transición hacia una arquitectura orientada a microservicios.

La tabla 7 del Anexo A presenta los componentes resultantes del análisis de los primeros casos de uso del submódulo de gestión de compras (G.D.C). Para cada caso de uso, se especifica el tipo de componente lógico involucrado —clasificado como interfaz (i), control (c) o datos (d)— junto con su descripción funcional, su representación en el sistema, el nombre propuesto del componente, así como su agrupación lógica dentro de un paquete y las relaciones existentes con otros elementos arquitectónicos.

Este proceso de refinamiento permite consolidar una base estructural sólida que guiará la identificación de microservicios, garantizando una separación clara de responsabilidades y fomentando la escalabilidad modular del sistema.

5.7. Identificación de Microservicios

Durante esta fase del proceso 4SRS-MSLA, se procedió a identificar los microservicios necesarios a partir de los casos de uso de negocio previamente identificados, descritos y refinados. A diferencia de los enfoques tradicionales que utilizan diagramas de componentes UML, en este caso se optó por representar la arquitectura lógica mediante un diagrama de bloques.

Esta decisión fue motivada por la necesidad de facilitar la comunicación con los stakeholders expertos del dominio. Reduciendo la carga cognitiva que representa para los expertos el enfrentarse a una arquitectura de sistemas presentada en un lenguaje con notaciones y semántica técnica, sin contexto de negocio, y la desconexión con su lenguaje familiar. El uso de bloques permitió mantener una estructura clara y accesible, reduciendo la carga técnica sin comprometer la representación conceptual del sistema distribuido.

La Figura 17 muestra el diagrama de bloques elaborado. En él, cada bloque representa un conjunto funcional o módulo del sistema, que a su vez encapsula uno o más microservicios. Las flechas indican las solicitudes o mensajes que envía un módulo a otro para poder ofrecer los servicios de negocio de los cuales es responsable, es decir, el microservicio desde el que se origina la petición apunta hacia el microservicio del cual depende o requiere para ofrecer sus servicios de manera completa.

Además, sobre cada flecha se ha especificado la API solicitada para completar la interacción. Esta notación permite visualizar con claridad las dependencias entre servicios, identificar puntos de integración y definir rutas de comunicación necesarias en la futura implementación.

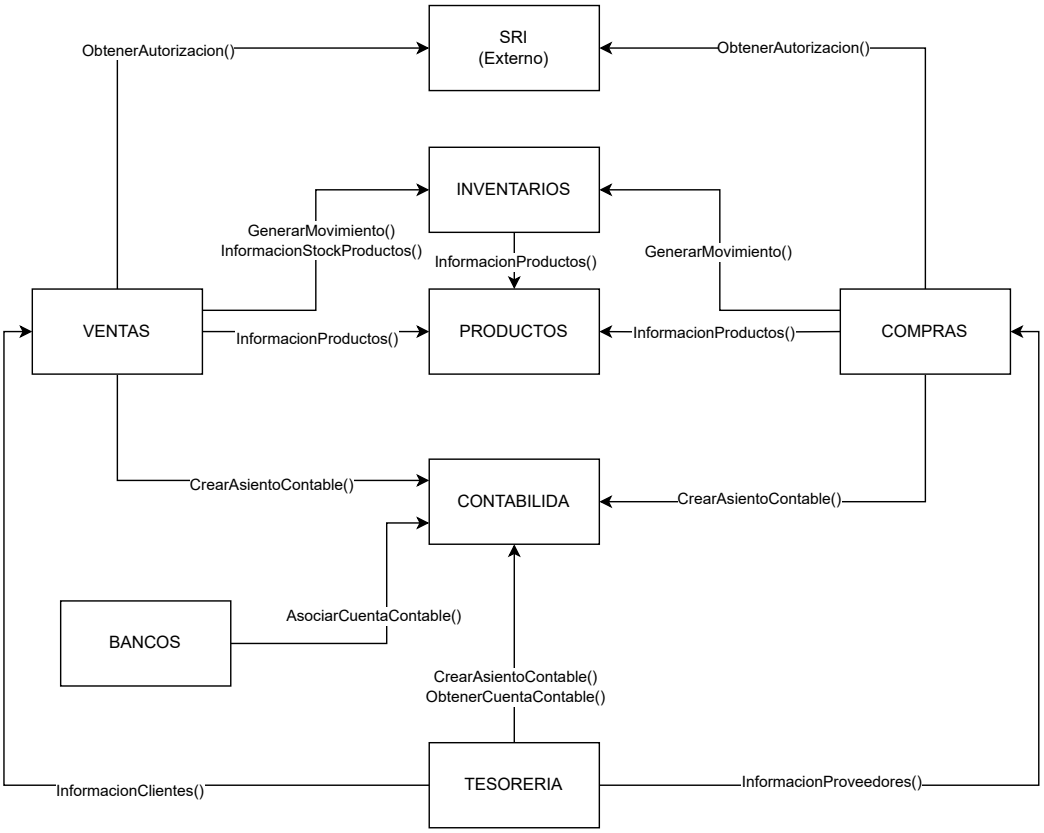

Figura 17: Diagrama de bloques para la identificación de microservicios

Este enfoque permitió establecer de manera colaborativa con los expertos de negocio las interacciones que requeridas entre los diferentes microservicio, así como los datos involucrados durante estas interacciones (tanto los correspondiente a la solicitud que hace un microservicio a otro, como los correspondientes a la respuesta esperada).

Esta información, obtenida a nivel de negocio, luego será utilizada para describir a nivel técnico el diseño arquitectónico (ej. interfaces en un diagrama de componentes descrito con UML)

5.7.1. Análisis de mensajes para la comunicación de microservicios

En la sección anterior se identificaron las solicitudes que un microservicio realiza a otro servicio. Estas solicitudes, en etapas de diseño arquitectónico posteriores, serán convertidas en operaciones de interfaces que describen los contratos de servicio que deben cumplir los microservicios involucrados en diferentes servicios de negocio.

Sin embargo, no es suficiente conocer los mensajes intercambiados entre microservicios; se

requiere conocer los datos involucrados en esos mensajes. Datos que forman parte tanto del mensaje que envía un microservicio a otro, como los datos involucrados en la respuesta esperada.

En ese contexto, para facilitar la comunicación con expertos del negocio y poder identificar esos datos sin utilizar un lenguaje técnico. (ej. interfaces o clases de uml) se utilizó una hoja en excel que permitió identificar los datos involucrados en las interacciones desde una perspectiva de negocio. La información recolectada haciendo uso de esta herramienta (tablas en excel), posteriormente será utilizada para el diseño de interfaces en un diagrama de componentes, traduciéndolo a clases que representan los datos que reciben o devuelven las operaciones previamente identificadas con el diagrama de bloques de la sección anterior.

Esta tabla fue elaborada en conjunto con el stakeholder experto del dominio y permitió obtener una visión clara sobre:

- Las operaciones que cada módulo necesita realizar sobre otros microservicios (consultas, creaciones, modificaciones, eliminaciones).
- Los datos que son enviados por el consumidor de la API.
- Las estructuras de datos que son devueltas por el proveedor del servicio.
- Las validaciones y reglas de negocio que deben cumplirse para aceptar o responder una solicitud.

Este análisis detallado fue clave para definir correctamente las responsabilidades de cada microservicio, identificar qué funcionalidades deben ser encapsuladas como servicios reutilizables, y delimitar los contratos esperados entre consumidores y proveedores.

A partir de esta tabla se identificaron patrones de consumo y dependencia que luego se reflejan en el Diagrama de Participantes, permitiendo consolidar el diseño orientado a servicios sobre una base colaborativa y validada funcionalmente.

Tabla 4: Tabla de mensajes identificadas entre Compras y Productos

	API 1	**API 2**
Origen	Compras	Compras
Destino	Productos	Productos
API	LISTA DE PRODUCTOS	OBTENER PRODUCTO
Operación	Consultar	Consultar
Datos enviados	-	Código
Datos devueltos	Lista: nombre, código, marca, unidad, tipo, precio	nombre, código, marca, unidad, tipo, precio, tipo de control
Validación	Producto esté activo y registrado	Valida existencia del producto y que esté activo

5.7.2. Diagrama de Participantes de Servicio

En esta etapa del modelado, el objetivo fue identificar de forma explícita qué actores interactúan con cada microservicio y cómo se estructuran las relaciones de consumo de servicios. Para ello, se utilizó un diagrama de participantes basado en el perfil SoaML, que permite representar de forma gráfica los roles funcionales de cada módulo dentro del sistema distribuido.

El modelado de la arquitectura de microservicios reveló una estructura compuesta por ocho participantes principales. Cada uno de estos fue concebido como un módulo funcional que representa a unidades clave del negocio, abarcando áreas como Compras, Ventas, Tesorería, Productos, Inventario, Contabilidad, Bancos y la gestión tributaria (SRI). A partir de la interacción entre estas entidades funcionales, se diseñaron nueve interfaces de servicio distintas. Estas interfaces exponen operaciones esenciales, tales como la generación de asientos contables, la consulta de productos y clientes, el control de stock, la obtención de autorizaciones fiscales y el registro de transacciones financieras. La clara delimitación de estos participantes y sus interfaces sentó las bases para una arquitectura de microservicios cohesiva, donde cada servicio opera con independencia y una responsabilidad bien definida.

Con el fin de lograr una visualización más clara y enfocada, se decidió acotar el diagrama únicamente a los microservicios de Compras, Productos y Contabilidad. Esta decisión se tomó en conjunto con los actores clave del proyecto, priorizando aquellos módulos que presentan una mayor complejidad funcional y un nivel elevado de interdependencia en esta etapa del desarrollo. De esta manera, fue posible representar de forma más detallada las interacciones entre módulos sin perder precisión ni sobrecargar el modelo.

La Figura 18 ilustra cómo el microservicio de Compras se apoya en los servicios ofrecidos por los módulos de Productos y Contabilidad. En particular, se observan las siguientes relaciones:

- Compras consume el servicio Información Productos, que le permite validar los datos de los artículos seleccionados para una orden de compra.
- Compras también invoca el servicio Creación Asiento Contable provisto por Contabilidad, a fin de generar automáticamente los registros contables correspondientes a una transacción.

Estas relaciones se modelan como contratos formales entre proveedores y consumidores, lo cual garantiza una integración coherente y facilita la evolución futura del sistema.

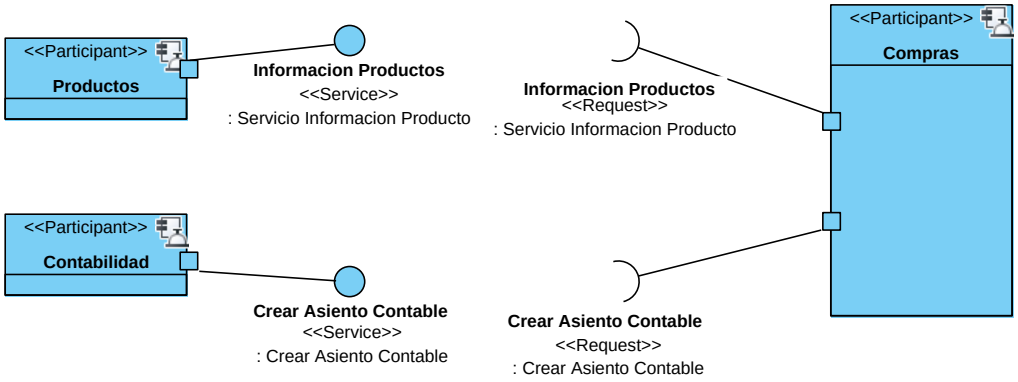

Figura 18: Diagrama de servicios y participantes: interacción entre Compras, Productos y Contabilidad

Para complementar esta visualización, la Tabla 5 resume los servicios modelados, el microservicio que los ofrece y los participantes que los consumen dentro del alcance de esta representación.

Tabla 5: Resumen de servicios ofrecidos y participantes consumidores

Servicio	**Proveedor**	**Consumidor(es)**
Información Productos	Productos	Compras
Creación Asiento Contable	Contabilidad	Compras

Este modelo contribuye a consolidar la arquitectura orientada a servicios, permitiendo una mayor trazabilidad de las responsabilidades funcionales y sirviendo como base para el diseño de contratos y validaciones en fases posteriores.

5.8. Definir Interfaces y Operaciones de Servicios

En esta fase del proceso 4SRS-MSLA se especifican las interfaces expuestas por cada microservicio y las operaciones que estos ofrecen al resto del sistema. Esta definición es clave para establecer los contratos de comunicación entre servicios y garantizar una interoperabilidad clara, formalizada y basada en estándares.

Estas interfaces se disenaron en base a la informacion recolectada con los stakeholders expertos de domnio y descrita a traves del digrama de bloques de la Figura 17 y los datos descritos desde una perspectiva de negocio como se ve muestra en al Tabla 4

Para representar esta información se utilizaron diagramas de interfaz de servicio, donde se modelan las interfaces ofrecidas por cada microservicio, los tipos de mensajes implicados y las operaciones disponibles. A continuación, se describen los diagramas correspondientes a

los microservicios de Compras, Productos y Contabilidad.

5.8.1. Interfaz de servicio del microservicio Compras

El diagrama presentado en la Figura 19 muestra las operaciones disponibles para acceder a la información de proveedores. Se expone un servicio llamado Servicio Información Proveedor, que ofrece dos métodos: ‘obtenerProveedor()‘ y ‘obtenerListaProveedores()‘. Ambos retornan datos estructurados mediante el tipo de mensaje Proveedor, que incluye atributos como identificación, nombre, dirección, correo y teléfono. Esta interfaz permite que otros módulos, como Tesorería o Contabilidad, consulten datos de proveedores activos al momento de registrar operaciones de compra o generar retenciones.

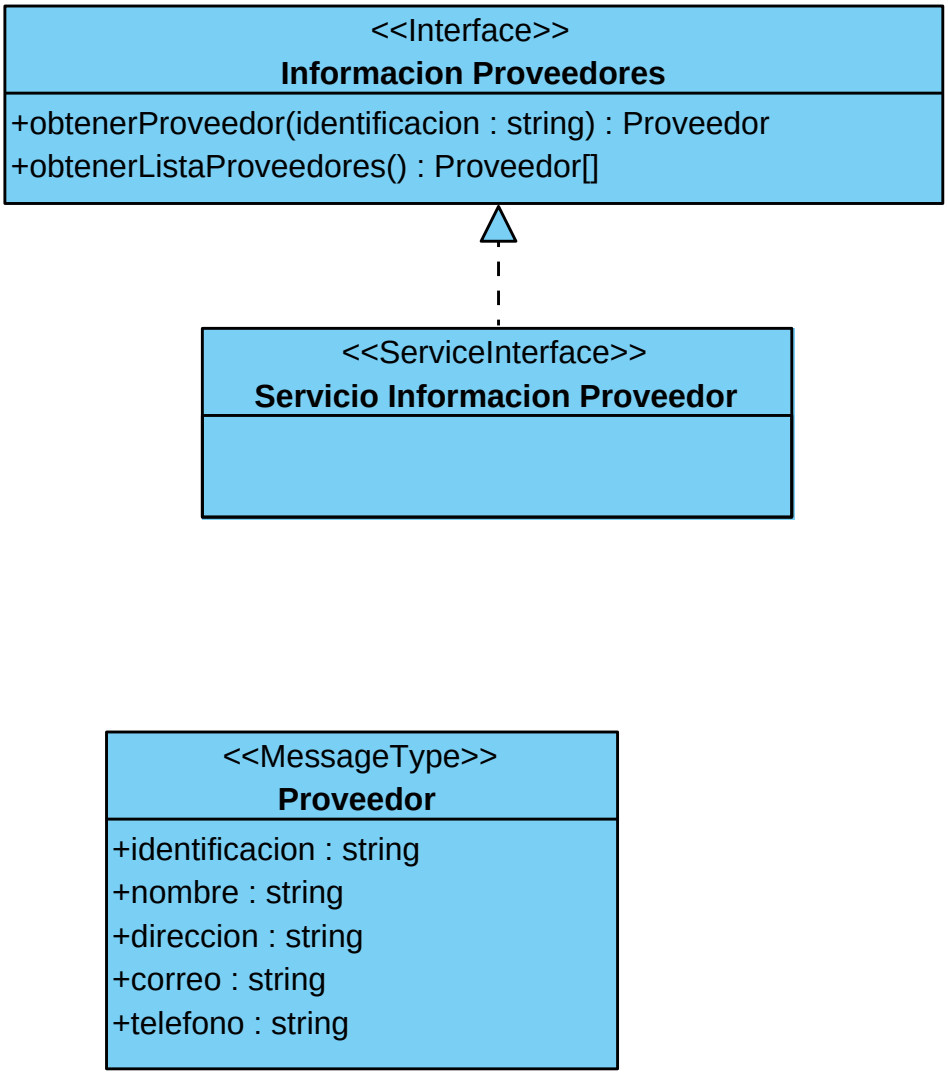

Figura 19: Diagrama de interfaz de servicio del microservicio Compras

5.8.2. Interfaz de servicio del microservicio Productos

La Figura 20 presenta la interfaz del microservicio Productos. Este módulo ofrece dos métodos clave: ‘obtenerProducto()‘ y ‘obtenerListaProductos()‘. Ambas operaciones permiten consultar datos como nombre, precio, código, cantidad y tipo de impuesto de los productos registrados. Los métodos se encapsulan en el servicio ‘Servicio Información Producto‘ y utilizan el mensaje `Producto` como tipo de retorno. Esta interfaz es consumida por los microservicios de Compras y Ventas para validar disponibilidad de productos, verificar impuestos aplicables y evitar errores de codificación en facturas.

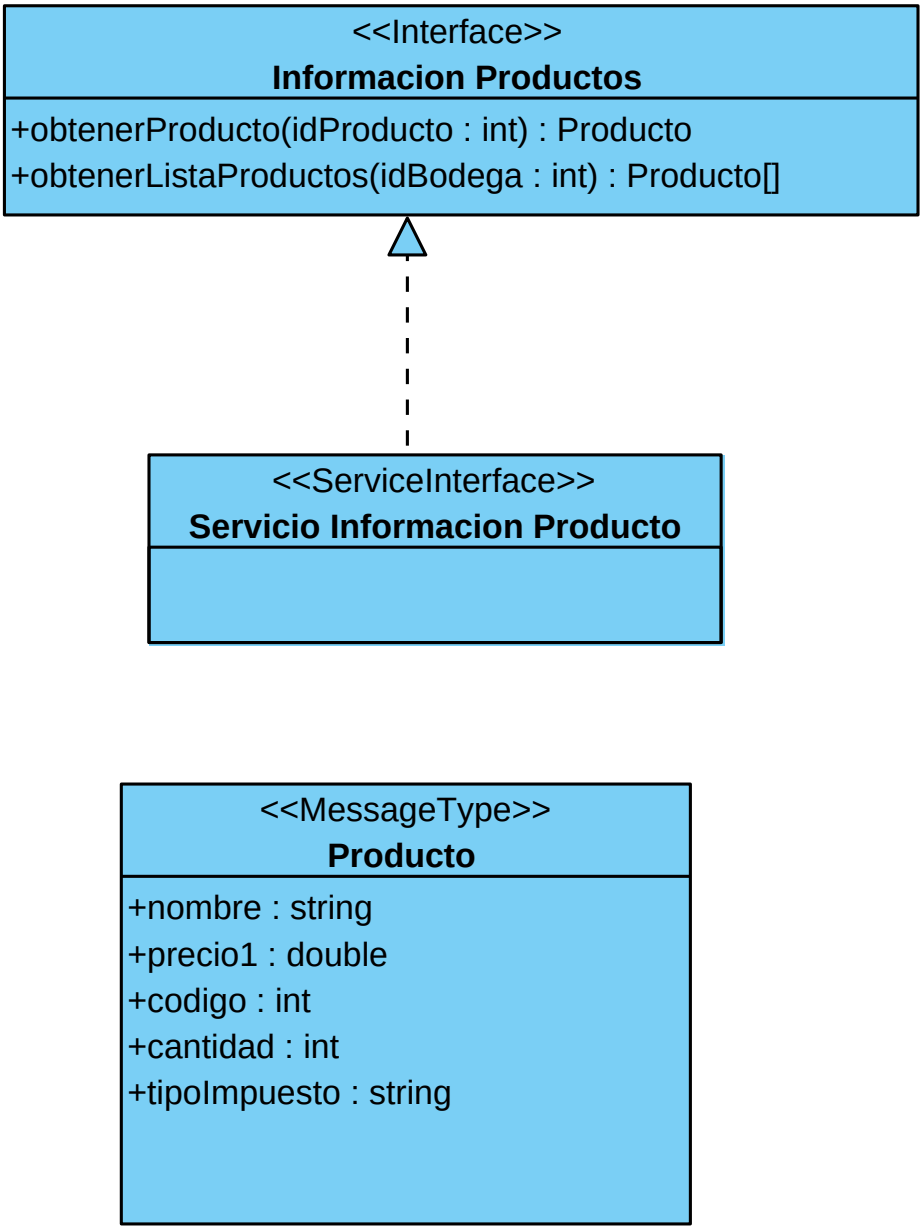

Figura 20: Diagrama de interfaz de servicio del microservicio Productos

5.8.3. Interfaz de servicio del microservicio Contabilidad

Finalmente, la Figura 21 representa las operaciones ofrecidas por el microservicio Contabilidad. Se identifican tres servicios: ‘Crear Asiento Contable‘, ‘Obtener Cuenta Contable‘ y ‘Asociar Cuenta Contable‘. Cada uno está compuesto por métodos específicos como ‘generarAsientoContable()‘ o ‘obtenerCuentaContable()‘, que reciben y devuelven estructuras complejas como `AsientoContable`, `CuentaContable` y `AsociacionBancoCuentaContable`. También se emplean enumeraciones como `TipoComprobante` y estructuras embebidas como ‘DetalleAsientoContable[]‘, necesarias para reflejar movimientos contables derivados de documentos de compra, venta o retención. Esta interfaz permite centralizar la lógica contable y garantiza integridad en los registros financieros.

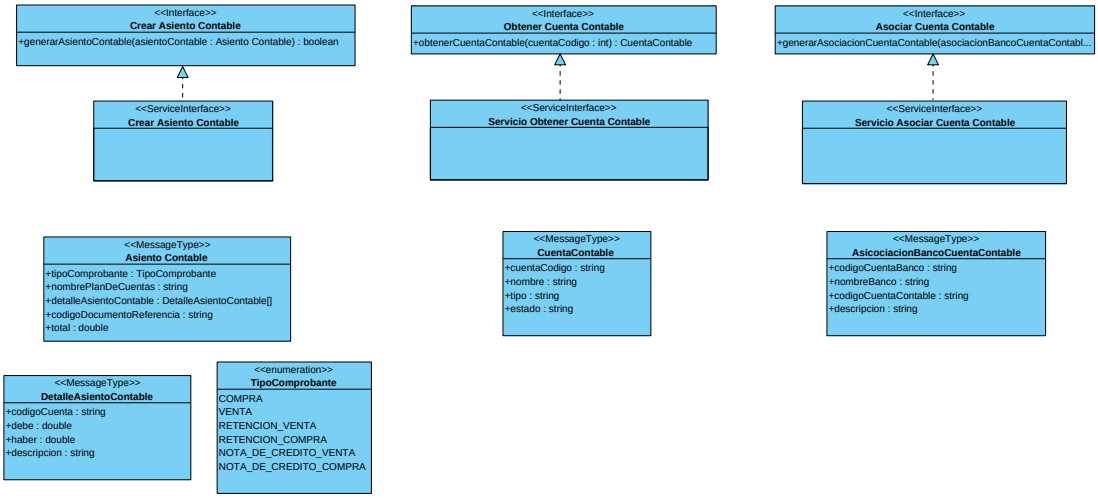

Figura 21: Diagrama de interfaz de servicio del microservicio Contabilidad

5.9. Especificar Contratos entre Proveedores y Consumidores

En esta etapa del proceso 4SRS-MSLA, se especifican los contratos de servicio entre los microservicios identificados. Un contrato de servicio define formalmente la relación entre un proveedor y un consumidor, estableciendo claramente qué servicios son ofrecidos, quién los consume y bajo qué condiciones ocurre dicha interacción.

El modelado se realiza mediante *diagramas de contrato de servicio* (Service Contract Diagrams), utilizando el perfil SoaML. Estos diagramas son fundamentales para garantizar la interoperabilidad entre microservicios, facilitar la evolución independiente de módulos y establecer acuerdos explícitos sobre el comportamiento esperado de cada parte involucrada.

A continuación, se presentan los contratos correspondientes a los microservicios de Compras, Productos y Contabilidad.

5.9.1. Contrato de servicio del microservicio Compras

La Figura 22 muestra el contrato de servicio *Información de Proveedores*, en el cual el microservicio **Compras** actúa como proveedor y un módulo consumidor (por ejemplo, Tesorería) solicita acceso a los datos de proveedores registrados en el sistema. Esta relación contractual garantiza que cualquier módulo que requiera esta información pueda consumirla mediante operaciones previamente definidas en la interfaz del microservicio.

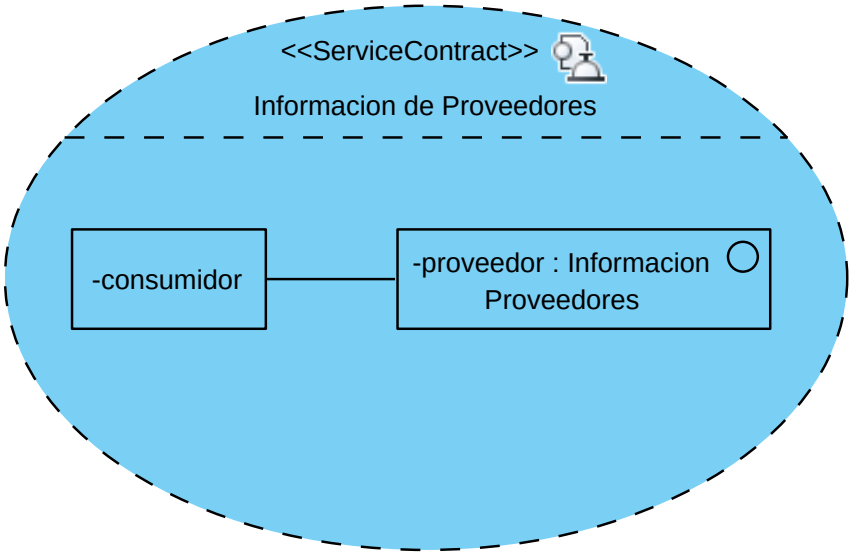

Figura 22: Contrato de servicio del microservicio Compras

5.9.2. Contrato de servicio del microservicio Productos

La Figura 23 representa el contrato de servicio *Información Productos*, donde el microservicio **Productos** ofrece servicios de consulta de datos de inventario (por ejemplo, nombres, códigos, precios) a consumidores como Compras o Ventas. Este contrato formaliza el acceso a la información de productos, permitiendo una integración segura, estructurada y desacoplada entre los módulos del sistema.

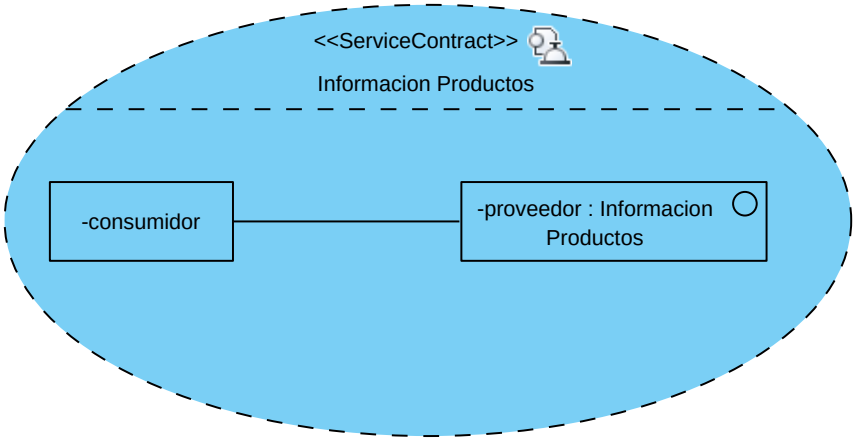

Figura 23: Contrato de servicio del microservicio Productos

5.9.3. Contratos de servicio del microservicio Contabilidad

Finalmente, la Figura 24 presenta tres contratos de servicio independientes ofrecidos por el microservicio **Contabilidad**:

- *Creación Asiento Contable*: consumido por Compras, Ventas y Tesorería, permite registrar operaciones contables de forma automatizada.
- *Información Cuenta Contable*: utilizado por Tesorería y Bancos, proporciona acceso al plan contable, clave para clasificar operaciones financieras.
- *Asociación Cuenta Contable*: habilita la vinculación de una cuenta contable con cuentas bancarias, esencial para la integración de bancos al sistema comercial.

Cada contrato establece una relación clara entre proveedor y consumidor, estructurando el sistema como una red de servicios desacoplados que comparten información a través de interfaces formales.

Figura 24: Contratos de servicio del microservicio Contabilidad

5.10. Modelar Roles que Ofrecen/Consumen Servicios

5.10.1. Modelado del Diagrama de Participantes de Servicios

La aplicación del método 4SRS-MSLA, en conjunto con un enfoque de modelado progresivo que abarcó diagramas de flujo, casos de uso, componentes, participantes, interfaces y contratos, permitió consolidar el diseño lógico de la arquitectura de microservicios propuesta para el sistema comercial. Este proceso sistemático no solo aseguró la coherencia, sino que también sentó las bases para una implementación robusta y escalable.

La Figura 25 presenta el **diagrama final de arquitectura de servicios**, en el cual se visualizan los módulos clave que componen el sistema y sus relaciones como proveedores o consumidores de servicios. Este modelo sintetiza todas las interacciones analizadas previamente y establece de forma clara las responsabilidades y dependencias entre microservicios.

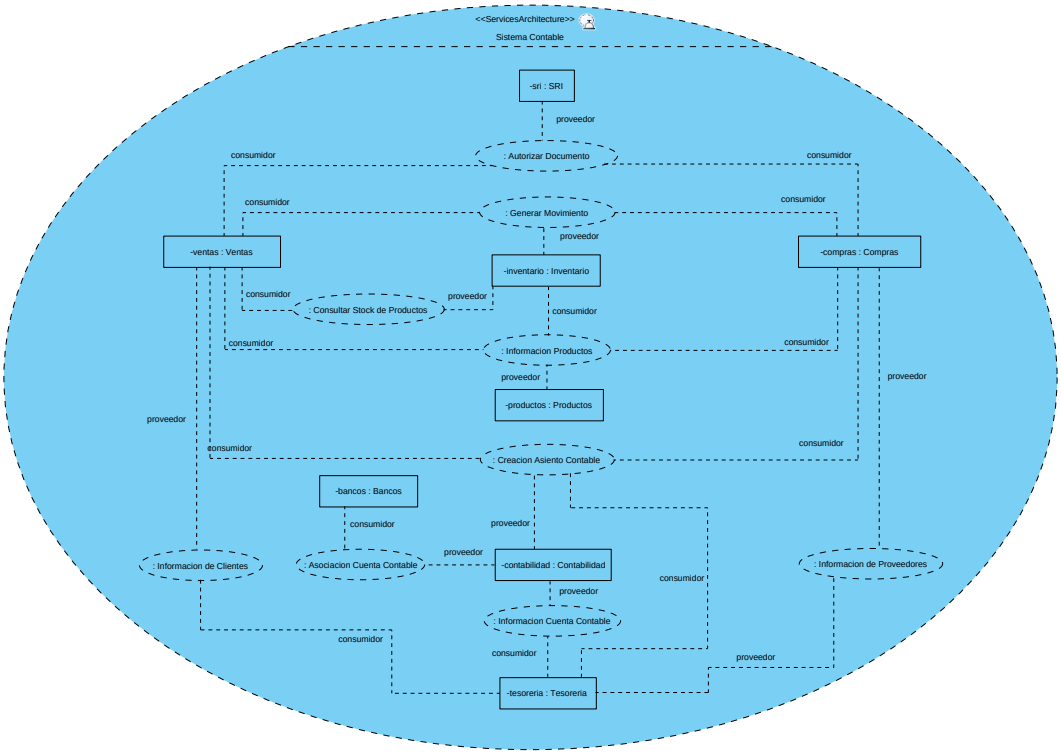

Figura 25: Diagrama final de arquitectura de servicios del sistema comercial

En el modelo se identifican los siguientes microservicios principales:

- Compras, Ventas, Tesorería, Contabilidad, Inventario, Productos, Bancos y SRI.

Cada uno de ellos expone una o más interfaces, agrupadas conceptualmente en servicios como:

- Información de Proveedores ofrecido por Compras.
- Información de Clientes ofrecido por Ventas.
- Información Productos y Consultar Stock de Productos, ofrecidos por Productos.
- Asociación Cuenta Contable e Información Cuenta Contable, ofrecidos por Contabilidad.
- Creación Asiento Contable, centralizado en Contabilidad y consumido por múltiples servicios.
- Generar Movimiento, ofrecido por Inventario.
- Autorizar Documento, provisto por el módulo SRI como interfaz con el servicio tributario nacional.

Las relaciones entre microservicios están representadas mediante flechas que indican qué módulo actúa como consumidor o proveedor, según el flujo de datos y la responsabilidad funcional. Esta vista final resume la lógica del sistema distribuido propuesto y proporciona la base para la implementación física y la orquestación de los servicios.

6. Desarrollo del prototipo

6.1. Implementación del diseño arquitectónico de microservicios

Para validar la viabilidad del diseño arquitectónico propuesto, se procedió a la construcción de un prototipo de aplicación en base al diseño arquitectónico propuesto en donde los microservicios creados fueron desplegados de forma local y en la nube. Este prototipo se desarrolló utilizando el ecosistema de Spring Boot, el cual demostró ser idóneo al ofrecer las herramientas necesarias para una implementación autónoma de cada servicio, lo que a su vez facilitó significativamente el control de versiones, las pruebas unitarias y los despliegues modulares. Los servicios fueron desplegados en una infraestructura local, con sus correspondientes bases de datos en la nube, y expuestos a través de APIs. Por lo tanto, las APIs a más de permitir la interacción entre microservicios permiten que aplicaciones externas se integren a las aplicaciones construidas en base al diseño propuesto.

6.1.1. Entorno de despliegue del prototipo

La Figura 26 muestra el diagrama de implementación correspondiente a este entorno. En esta etapa del prototipo se incluyeron tres microservicios: Compras, Productos y Contabilidad, debido a su alta interacción funcional dentro del sistema comercial.

Cada uno de estos microservicios se registra en un servidor Eureka, que actúa como servicio de descubrimiento. Esto permite que las instancias se comuniquen entre sí sin necesidad de conocer su ubicación exacta en la red, favoreciendo un entorno altamente desacoplado.

Para la centralización de configuraciones se integró un Spring Cloud Config Server, donde se gestionan las propiedades externas de cada microservicio desde un repositorio común. Esto ofrece mayor control sobre variables sensibles (como credenciales o URLs) y permite actualizaciones sin necesidad de recopilar el código.

Las bases de datos fueron desplegadas en Railway, siendo cada una desplegada en infraestructuras de la nube independientes de la otra en MySQL.

Además, se utilizó un API Gateway que opera como punto de entrada único del sistema. Este componente se encarga de enrutar las solicitudes entrantes hacia el microservicio correspondiente, aplicar políticas de seguridad, gestionar el control de acceso y realizar tareas de monitoreo.

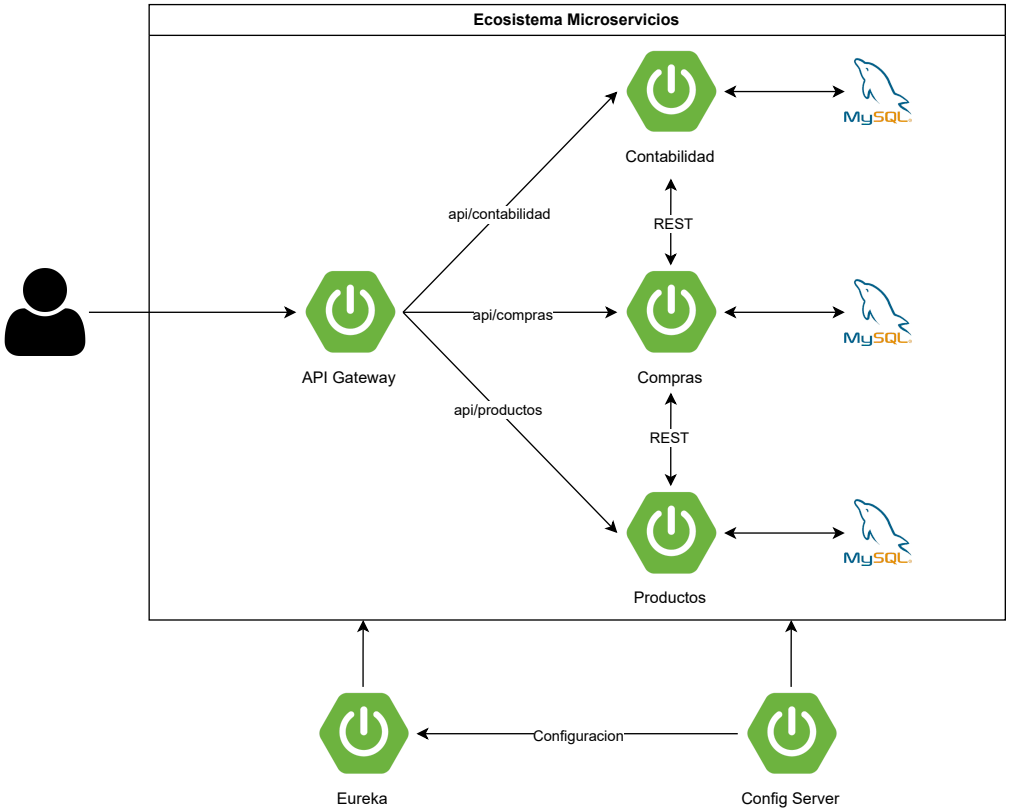

Figura 26: Entorno de despliegue del prototipo con microservicios en Spring Boot

Este entorno simula una arquitectura real en la que los servicios están desplegados de forma autónoma, interactúan por medio de API REST y son gestionados mediante mecanismos estándar de descubrimiento y configuración. Con esta base, se procedió al desarrollo parcial de las funcionalidades de cada microservicio, así como a la integración con prototipos de interfaz que consumen sus servicios a través del API Gateway.

La Tabla 6 presenta los elementos clave del entorno de despliegue, junto con las tecnologías seleccionadas y su función dentro del sistema. Cada componente fue elegido por su integración nativa con el ecosistema de Spring Boot y su capacidad para facilitar el desarrollo, escalado e interoperabilidad del sistema comercial.

Tabla 6: Componentes de infraestructura para la implementación física de servicios

Componente	Tecnología	Función
API Gateway	Spring Cloud Gateway	Punto de entrada único que enruta solicitudes al microservicio correspondiente y controla el acceso.
Eureka Server	Netflix Eureka	Facilita que los microservicios se encuentren entre sí dinámicamente, sin depender de direcciones IP estáticas, lo cual es esencial en entornos de escalado automático.
Config Server	Spring Cloud Config	Centraliza la configuración de todos los microservicios, lo que permite mantener consistencia y flexibilidad en distintos entornos.
Compras, Productos, Contabilidad	Spring Boot	Microservicios principales del prototipo. Cada uno maneja una funcionalidad específica y expone endpoints REST.
Base de datos	MySQL	Almacenamiento persistente de información para cada microservicio. Puede estar compartida o distribuida según el diseño.

6.2. Codificación y Testeo

Se implementan los microservicios siguiendo los diseños previos y se realizan pruebas exhaustivas. Se desarrollan pruebas unitarias, de integración y de rendimiento para asegurar que los servicios cumplen con los requisitos y son escalables. Herramientas como JUnit o Postman pueden ser utilizadas en esta fase.

El resultado es un Prototipo Funcional, que incluye microservicios completamente desarrollados, desplegados y probados, listos para ser integrados en un entorno de producción.

6.2.1. Estructura del backend por microservicio

La implementación del backend se realizó utilizando Spring Boot, siguiendo una estructura modular que favorece la mantenibilidad, escalabilidad y separación de responsabilidades. Cada microservicio cuenta con su propio repositorio, configuración, y arquitectura interna organizada en paquetes funcionales. Esta estructura se diseñó para reflejar de manera clara el flujo de datos desde las solicitudes HTTP hasta la persistencia y la lógica de negocio.

La Figura 27 muestra la organización del microservicio **Compras**. Esta estructura es repre-

sentativa del resto de microservicios desarrollados en el prototipo, y sigue principios comunes como: segmentación entre capa de controlador, lógica de negocio, acceso a datos y manejo de DTOs.

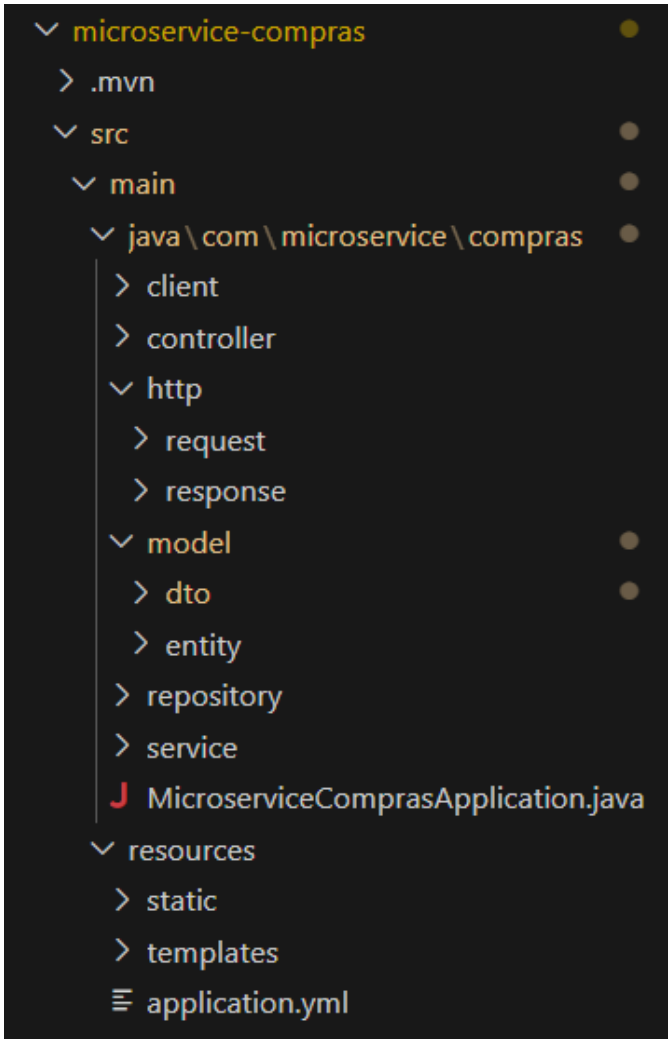

Figura 27: Estructura del microservicio Compras en Spring Boot

A continuación, se describen los paquetes principales que conforman el microservicio:

- controller: contiene las clases que manejan las solicitudes HTTP entrantes. Se encarga de enrutar los endpoints REST hacia la capa de servicio correspondiente.
- service: encapsula la lógica de negocio. Aquí se definen los procesos que combinan operaciones de acceso a datos, validaciones y reglas específicas del dominio de compras.
- repository: define las interfaces que permiten la comunicación con la base de datos, generalmente extendiendo JpaRepository u otras interfaces de Spring Data JPA.
- model.entity: contiene las entidades JPA que representan las tablas del modelo relacional y sus relaciones.

- model.dto: incluye los objetos de transferencia de datos (DTOs) utilizados para estructurar la información que se expone o se recibe a través de las APIs.
- http.request y http.response: organizan de forma separada los objetos que representan las peticiones y las respuestas en la capa web. Esta separación permite mayor claridad al estructurar los datos que provienen del cliente y los que se devuelven desde el backend.
- client: se reserva para integrar futuros consumidores de servicios externos o llamados a otros microservicios usando RestTemplate, WebClient o Feign.
- application.yml: ubicado en la carpeta resources, este archivo gestiona la configuración del microservicio, incluyendo puertos, rutas, propiedades de base de datos, nombre de instancia, y otros parámetros utilizados por Eureka y el Config Server.

Esta estructura modular no solo mejora la legibilidad del código, sino que también facilita el trabajo en equipo y la escalabilidad futura del sistema. Cada capa se encuentra bien definida, y el uso de DTOs y paquetes HTTP permite desacoplar los modelos internos de las estructuras expuestas al exterior.

6.2.2. Implementación de DTOs para consumo de microservicios

Una de las prioridades al momento de implementar el prototipo fue garantizar que las estructuras de datos utilizadas en el código del backend reflejaran fielmente lo definido previamente en el diseño lógico basado en SoaML. En particular, los mensajes definidos en los diagramas de interfaz de servicio y en los contratos de servicio fueron utilizados como base para la creación de los objetos de transferencia de datos (DTOs) en la capa de aplicación.

La Figura 28 muestra la clase ProductoDto implementada en el microservicio de Compras, que es utilizada para consumir el servicio de información de productos ofrecido por el microservicio Productos. Este DTO se encarga de recibir los datos estructurados desde el endpoint del proveedor del servicio.

```
public class ProductoDto {
    private String codigo;
    private String nombre;
    private Double precio_uno;
    private String marca;
    private String tipo;
    private Double impuesto;
}
```

Figura 28: Clase `ProductoDto` utilizada para consumir el microservicio de Productos

Como se puede observar, los campos definidos en la clase DTO coinciden exactamente con

los atributos establecidos en el mensaje Producto del modelo SoaML, presentado anteriormente en el diagrama de interfaz del microservicio Productos:

- Codigo: representa el identificador único del producto.
- Nombre: nombre comercial del producto.
- Precio uno: primer precio de referencia (por ejemplo, precio de venta).
- Marca: marca o fabricante del producto.
- Tipo: clasificación o categoría del producto.
- Impuesto: valor del impuesto aplicado al producto.

Esta correspondencia entre el diseño y la implementación demuestra la coherencia entre las etapas del desarrollo y valida la utilidad de utilizar un enfoque orientado a servicios desde el modelado temprano del sistema. Además, permite asegurar que los datos intercambiados entre microservicios siguen una estructura conocida y predefinida, lo que facilita la integración, el testing y la trazabilidad del sistema en su conjunto.

6.3. Estructura del frontend

La interfaz gráfica de usuario fue desarrollada utilizando la biblioteca React, estructurada bajo una arquitectura modular por vistas y componentes reutilizables. Esta organización permite mantener un código limpio, escalable y fácil de mantener, respetando el principio de separación de responsabilidades tanto a nivel visual como funcional.

La Figura 29 muestra la estructura de carpetas del frontend. El directorio raíz src contiene subcarpetas organizadas por tipo y propósito.

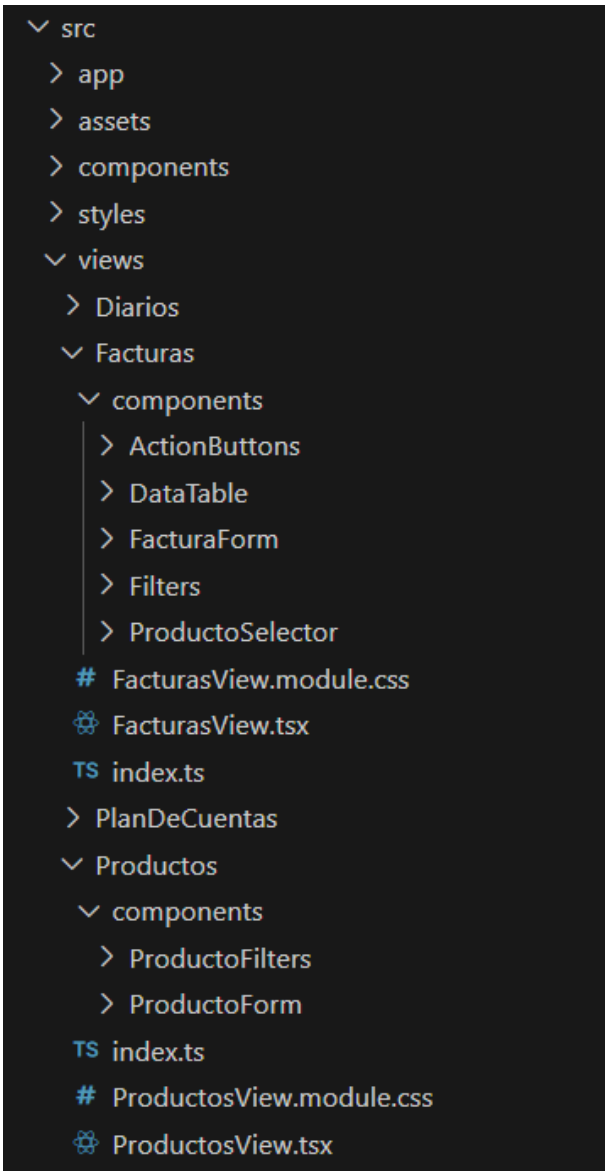

Figura 29: Estructura del proyecto frontend desarrollado en React

A continuación, se describen los elementos más relevantes:

- Views: Contiene las vistas principales del sistema, agrupadas por dominio funcional. Por ejemplo:
  - Facturas: incluye la lógica y componentes asociados a la gestión de facturas de compra. Dentro de esta vista se encuentran subcomponentes reutilizables como ActionButtons, DataTable, FacturaForm, Filters y ProductoSelector.
  - Productos: alberga la vista y formularios relacionados a la administración de productos. Incluye componentes como ‘ProductoFilters‘ y ‘ProductoForm‘.
  - Diarios y PlanDeCuentas: representan otras vistas relacionadas al dominio contable.

- Components: Carpeta general para componentes reutilizables entre distintas vistas, si bien en este caso la reutilización principal está encapsulada dentro de subcarpetas de cada vista.
- Styles: Contiene los estilos CSS locales, algunos bajo módulos como FacturasView.module.css y ProductosView.module.css, que permiten aplicar estilos scoped a cada componente.
- Assets, app e index.ts: archivos principales de configuración y recursos estáticos, típicos de un proyecto React con TypeScript.

Esta organización permite una escalabilidad natural del proyecto. Cada vista se encapsula con sus propios estilos, componentes internos y lógica asociada, favoreciendo el mantenimiento modular del sistema. Además, al estar desarrollado en TypeScript, se asegura un tipado fuerte y mayor robustez en la manipulación de datos provenientes de los microservicios.

6.4. Diseño de interfaces

El diseño de las interfaces gráficas de usuario se realizó con el objetivo de ofrecer una experiencia fluida, funcional y alineada con los procesos definidos en la arquitectura de microservicios. Estas interfaces se desarrollaron utilizando React y organizan la interacción del usuario con los microservicios a través de formularios, vistas de tablas y paneles de búsqueda.

Cada interfaz del frontend consume servicios REST expuestos por los microservicios de Compras, Productos y Contabilidad. Las solicitudes se realizan mediante llamadas HTTP, utilizando objetos DTO previamente definidos en el backend, lo que asegura una integración estructurada y coherente con los modelos de datos.

6.4.1. Interfaz de historial de facturas de compra

La Figura 30 muestra la vista donde se listan todas las facturas registradas. Esta interfaz consume el microservicio tipo GET de compras que permite obtener una lista de las facturas compras y permite filtrar por fecha, proveedor o tipo de documento. Además, incorpora botones para anular, modificar o consultar detalles.

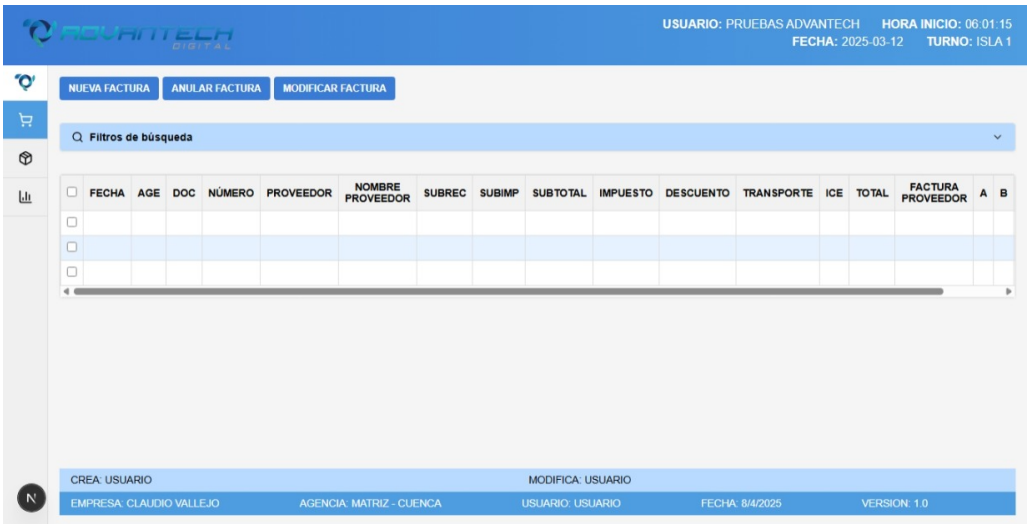

Figura 30: Listado de facturas registradas

6.4.2. Interfaz de creación de factura de compra

En la Figura 31, se visualiza el formulario para registrar una nueva factura de compra. Esta vista incluye secciones para el proveedor, los productos y los datos de pago. Al confirmar la acción, se emite una solicitud tipo POST con el detalle de los productos y totales calculados, lo que permite generar automáticamente el registro de la factura y, en paralelo, la solicitud de creación del asiento contable a través del microservicio de Contabilidad.

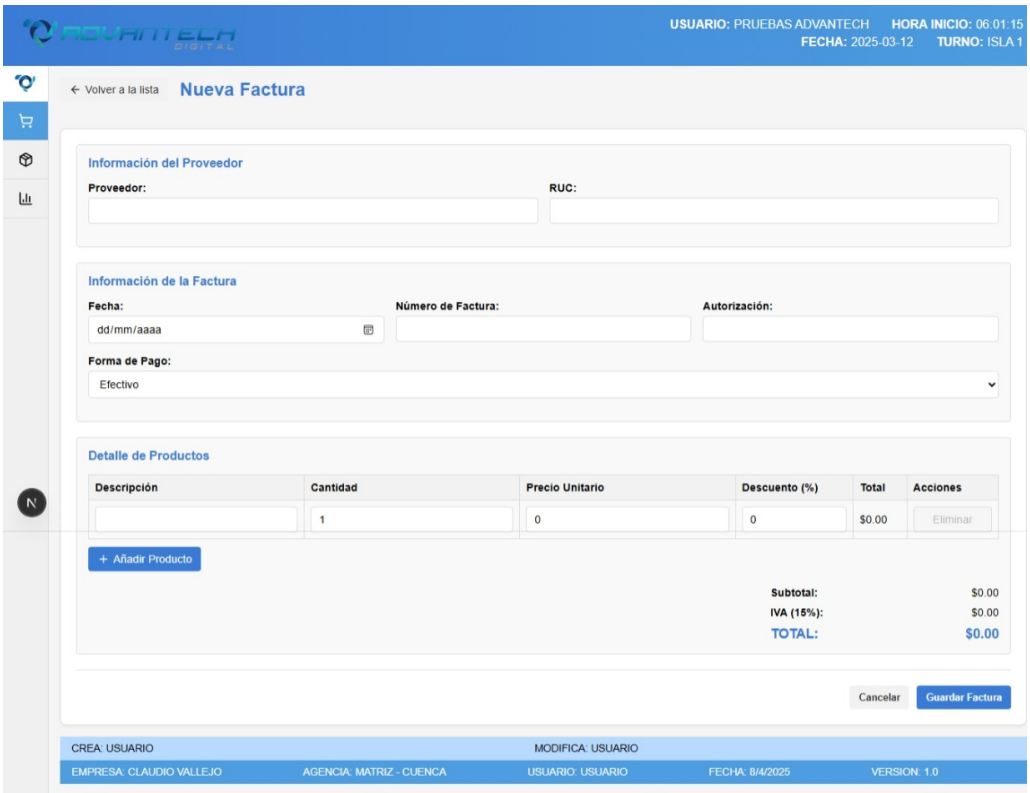

Figura 31: Formulario para registrar una nueva factura de compra

6.4.3. Interfaz de selección de productos para la factura de compra

En la Figura 32, se observa una ventana emergente que permite seleccionar uno o varios productos para una factura de compra. Esta interfaz consume el endpoint del microservicio de Productos mediante una solicitud GET, que retorna una lista de productos activos. Los datos mostrados en la tabla como código, nombre, precio, marca, tipo e IVA provienen directamente del DTO ProductoDto, cuya estructura fue definida previamente con base en el modelo SoaML.

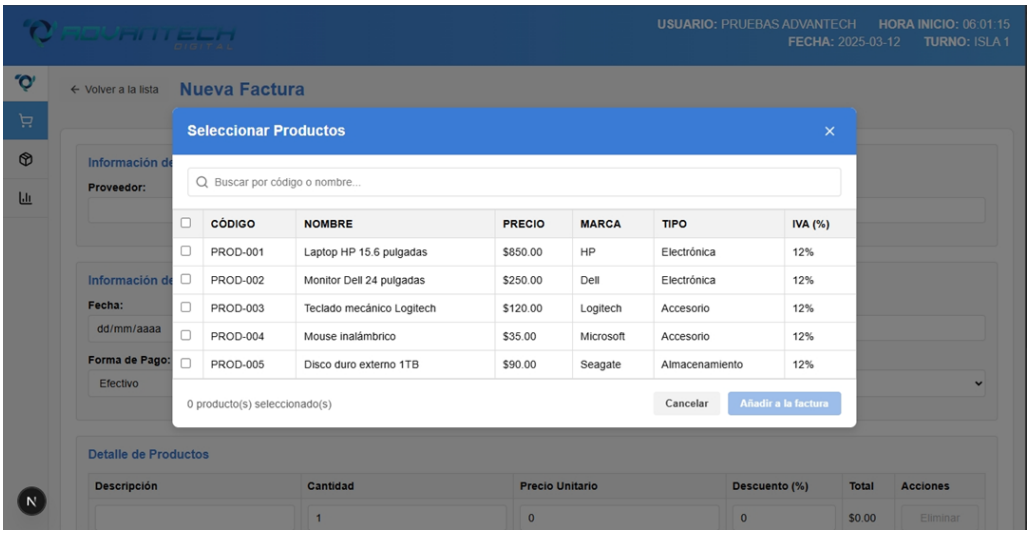

Figura 32: Interfaz de selección de productos en el proceso de compra

6.4.4. Interfaz de consulta de productos

La Figura 33 corresponde a la vista donde se listan todos los productos del sistema. Esta pantalla permite aplicar filtros por marca, tipo o código de barras. También permite acceder rápidamente a la modificación del producto.

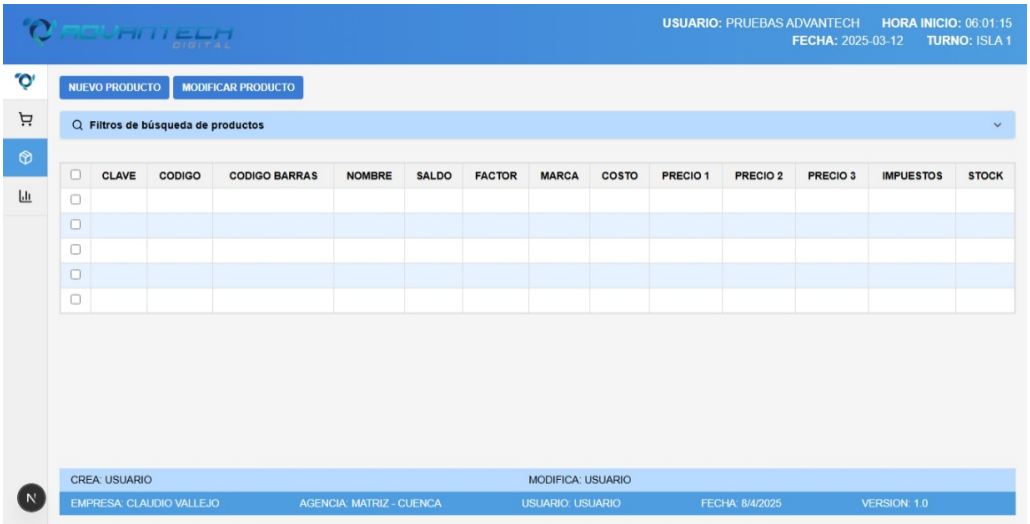

Figura 33: Listado y filtro de productos registrados

Cada una de estas interfaces refleja el diseño orientado a servicios que se definió en las fases anteriores. Su desarrollo se alinea con los contratos establecidos en los modelos SoaML y se basa en una comunicación estrictamente desacoplada, consumiendo únicamente los servicios publicados por cada microservicio.

6.4.5. Interfaz de creación de producto

La Figura 34 muestra el formulario utilizado para registrar un nuevo producto. Este formulario permite ingresar datos básicos como código, nombre, descripción, marca, tipo, precio y condiciones fiscales. Al guardar, se emite una solicitud POST, consumido por el microservicio de Productos, el cual valida y almacena la información en la base de datos.

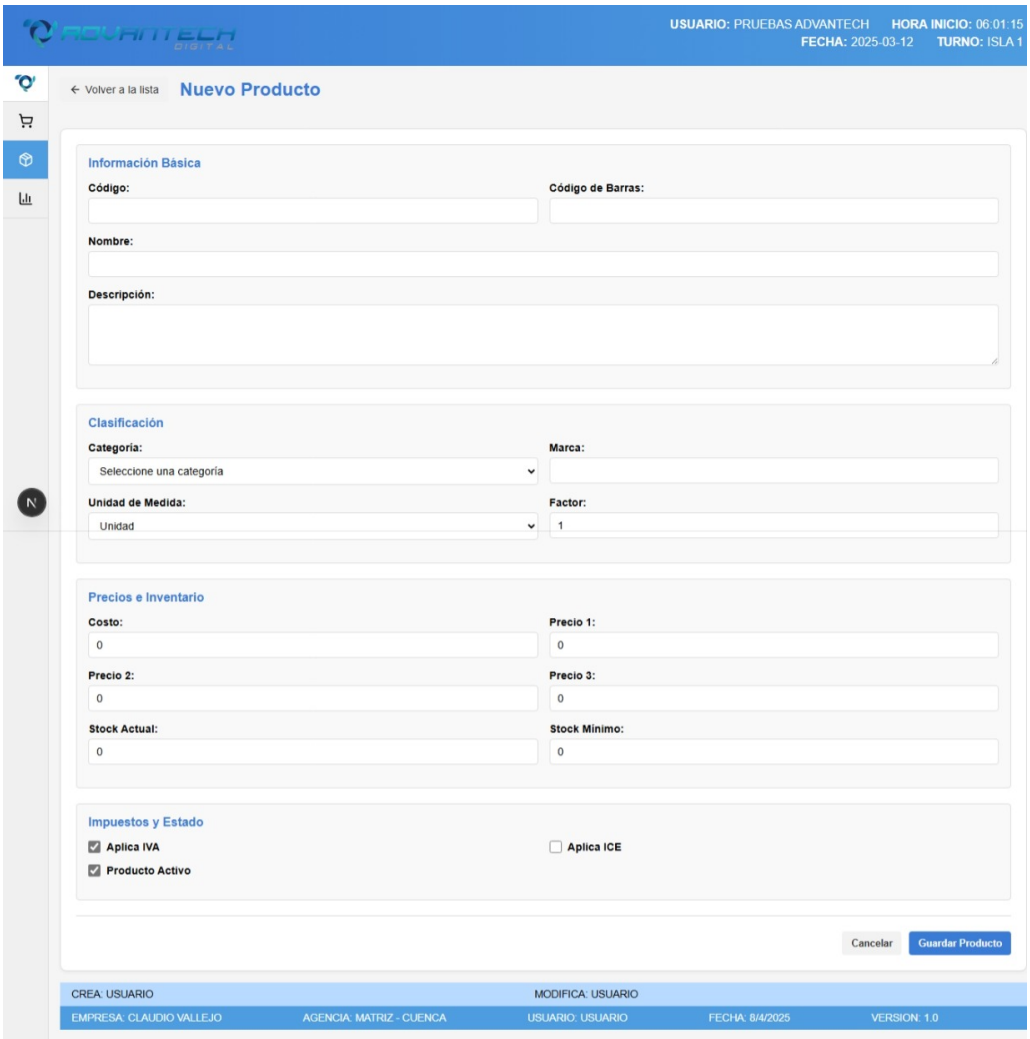

Figura 34: Formulario de creación de producto

7. Validación funcional del prototipo

Debido a restricciones de tiempo y alcance, no se implementaron todos los microservicios definidos en la arquitectura. No obstante, se desarrolló un caso funcional que permite validar la factibilidad del diseño propuesto, a través de la integración real entre los módulos de Productos y Facturas de Compra.

7.1. Caso funcional propuesto para la validación del prototipo

Debido a limitaciones de tiempo y recursos, el desarrollo del sistema completo no pudo cubrir todos los microservicios propuestos en la arquitectura. No obstante, se implementó un caso funcional representativo que permitió validar la viabilidad del diseño y comprobar el correcto funcionamiento de los componentes desarrollados.

Este caso funcional se centra en la interacción entre los microservicios de Compras y **Productos**, y simula el flujo real de una factura de compra en la que el usuario debe seleccionar productos desde una interfaz web conectada al backend.

7.1.1. Objetivo del caso funcional

El propósito principal de este caso fue demostrar que:

- Los microservicios se ejecutan de forma independiente y están desplegados de manera desacoplada.
- Las interfaces REST definidas en la arquitectura funcionan según lo modelado en los diagramas de servicio y contratos.

7.1.2. Funcionamiento del caso

El escenario comienza cuando el usuario accede al sistema y navega hacia el módulo de facturas de compra. Desde allí, se despliega un formulario que le permite registrar una nueva factura. Como parte del proceso, el usuario necesita añadir uno o más productos al detalle de la factura. Para ello, interactúa con un componente de la interfaz que abre una ventana emergente diseñada específicamente para la selección de productos.

La interfaz del usuario realiza una solicitud HTTP al microservicio de Productos a través del endpoint GET /api/detalle-factura-compra/obtener-productos. Aunque esta operación corresponde funcionalmente al microservicio de Productos, la solicitud se canaliza mediante el API Gateway implementado con Spring Boot. Gracias a esta configuración, los microservicios alojados bajo el dominio de Compras pueden acceder a los servicios de Productos de forma transparente, sin necesidad de conocer directamente su ubicación ni detalles internos. Esta

capa de enrutamiento centralizado permite mantener un sistema desacoplado, organizado y fácil de escalar. Esta petición desencadena una respuesta por parte del servicio, que entrega una lista de productos disponibles en formato estructurado. Cada producto viene representado por un objeto ProductoDto, cuya estructura fue previamente modelada en la fase de diseño lógico y que contiene atributos como código, nombre, precio, marca, tipo e impuesto.

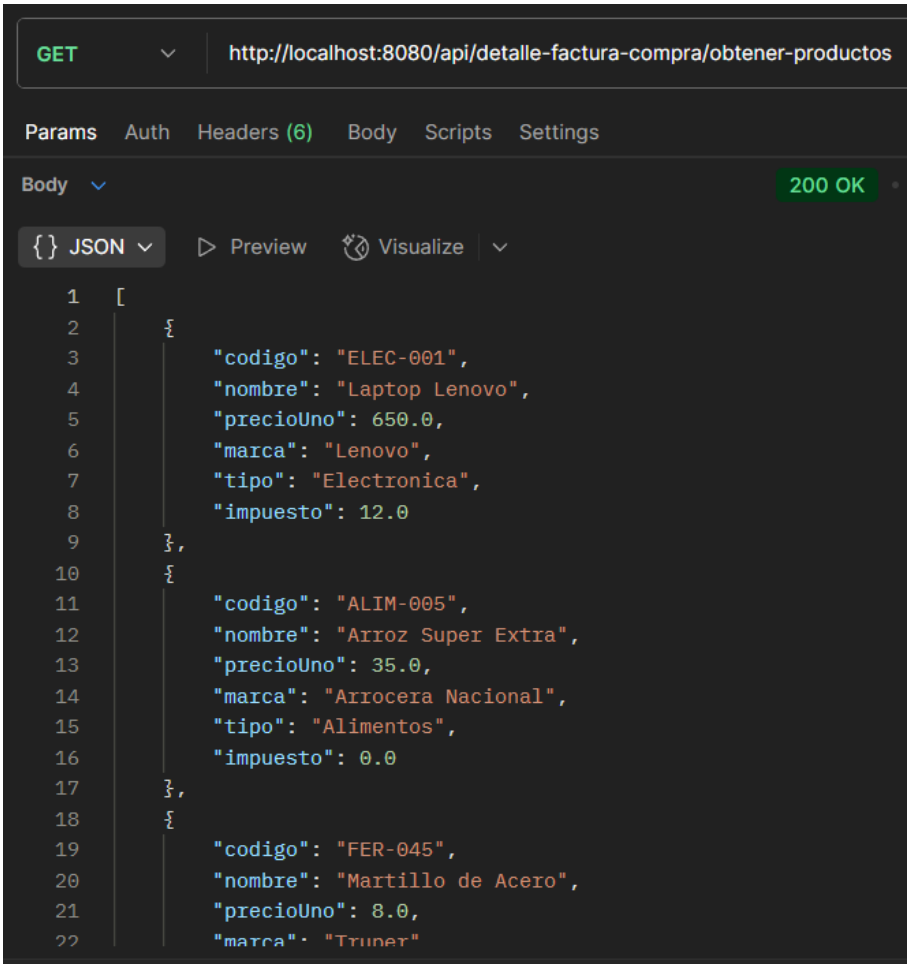

Figura 35: Lista de productos recibida consumiendo API desde Compras

La interfaz muestra esta información en una tabla, permitiendo al usuario revisar y seleccionar los productos deseados. Una vez realizada la selección, los productos se integran automáticamente al detalle de la factura, completando así la construcción del documento desde el frontend. Todo este proceso se realiza sin acoplamiento directo entre módulos, únicamente mediante consumo de servicios, validando de esta forma el enfoque orientado a microservicios definido en la arquitectura.

Figura 36: Lista de productos cargada en la interfaz

7.1.3. Validaciones técnicas realizadas

Este caso permitió validar, en un entorno real de ejecución, los siguientes aspectos clave del sistema:

- Interoperabilidad: los microservicios se comunican correctamente mediante HTTP REST.
- Desacoplamiento: Compras no conoce la lógica interna de Productos, solo su API expuesta.
- Coherencia de diseño: el DTO utilizado en la respuesta coincide con el mensaje definido en los diagramas SoaML.
- Consumo real desde el frontend: el sistema de interfaz gráfica puede conectarse a los servicios y procesar los datos recibidos.
- Aplicación de un flujo funcional: el usuario logra completar una tarea significativa usando módulos implementados.

7.1.4. Conclusión del caso funcional

Aunque el prototipo implementado representa solo una parte de la arquitectura completa, este caso funcional proporciona evidencia sólida de que el diseño es aplicable y funcional. La correcta interacción entre microservicios, el respeto de los contratos definidos y la integración fluida con el frontend demuestran que los principios arquitectónicos adoptados son viables para su escalado futuro.

Este caso puede considerarse como una prueba de concepto del enfoque orientado a microservicios, donde cada módulo puede ser desarrollado y validado de manera autónoma, respetando los acuerdos de integración definidos en fases previas del proyecto.

8. Metodología resultante del caso de estudio

La metodología resultante del caso de estudio que fue realizada en las secciones 5, 6 y 7 en las que se articulan en diez fases secuenciales que permiten transformar los procesos del negocio en una arquitectura lógica y física de microservicios. Estas fases están diseñadas para guiar el desarrollo de sistemas empresariales modernos, asegurando modularidad, escalabilidad y claridad en la transición desde sistemas monolíticos. A continuación, se describen cada una de las fases junto con sus respectivos entregables.

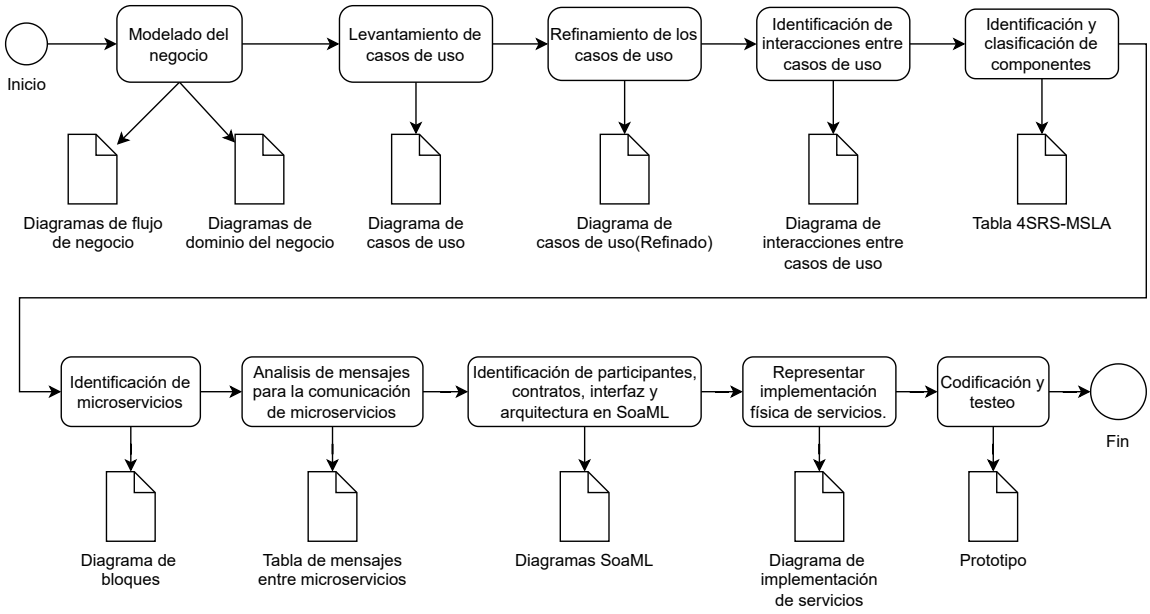

Figura 37: Metodología para el diseño de una arquitectura basada en microservicios

8.1. Modelado del negocio

Esta fase tiene como objetivo comprender los procesos fundamentales que soporta la aplicación actual desde la perspectiva de los usuarios y stakeholders. Se realizaron entrevistas, talleres y análisis de documentación para identificar cómo operan las estaciones de servicio que utilizan el sistema.

Como resultado de esta fase, se dispone de diagramas de flujo del negocio que representan las secuencias de actividades, decisiones y actores involucrados en los procesos de ventas, compras e importaciones. Además, se incorpora un diagrama del dominio del negocio que permite visualizar los conceptos fundamentales y sus relaciones dentro del sistema.

8.2. Levantamiento de casos de uso

Una vez modelado el negocio, se procede al levantamiento de los casos de uso que representan las interacciones entre los actores del sistema y las funcionalidades que este debe ofrecer. Esta fase tiene como propósito capturar los requerimientos funcionales mediante la construcción de diagramas UML, iniciando desde un nivel general y profundizando en la descomposición funcional hasta un tercer nivel.

El proceso inicia con la identificación de los casos de uso de primer nivel, los cuales representan funcionalidades globales asociadas a procesos empresariales clave, tales como gestión de ventas, compras, productos o contabilidad. A partir de estos, se desarrollan los casos de uso de segundo nivel, que descomponen las funcionalidades principales en tareas más específicas, vinculadas a los flujos normales de operación. Finalmente, se construyeron los casos de uso de tercer nivel, donde se detallan las operaciones concretas que se traducen posteriormente en funcionalidades CRUD (crear, leer, actualizar, eliminar), típicas en sistemas de gestión empresarial.

Este enfoque jerárquico permite capturar tanto la visión de alto nivel del sistema como los detalles operativos necesarios para su implementación técnica. Además, facilitó la identificación temprana de actores, flujos alternativos y condiciones excepcionales.

Esta fase permite disponer de diagramas de casos de uso desarrollados en tres niveles de profundidad, que reflejan la estructura jerárquica de las funcionalidades del sistema. Se complementa con plantillas narrativas que detallan los flujos básicos y alternos, así como con documentación estructurada que describe las condiciones previas, las reglas de negocio asociadas y los resultados esperados para cada escenario identificado.

8.3. Refinamiento de los casos de uso

Una vez estructurados los casos de uso hasta el tercer nivel, se procede al refinamiento detallado de cada uno de ellos con el fin de facilitar su futura transformación en componentes técnicos reutilizables. Esta fase no solo profundiza en los flujos operativos, sino que introduce un análisis de los datos que intervienen en cada acción, lo cual permite esclarecer las dependencias funcionales y la comunicación entre los distintos módulos del sistema.

Para cada caso de uso refinado se identifican explícitamente los datos requeridos como entradas, así como los datos producidos como salidas. Estos datos fueron descritos en detalle donde se incluyen su nombre, tipo y descripción.

Asimismo, el análisis de los datos posibilita detectar patrones de uso comunes, reglas de validación y restricciones de integridad que debían ser considerados durante el diseño de la arquitectura lógica. Este refinamiento permite consolidar una visión integral de cada funcionalidad, no solo en términos de pasos secuenciales, sino también en cuanto a su contexto de datos y a su interacción con otras funcionalidades. La identificación precisa de estos pun-

tos de conexión fue esencial para la posterior agrupación de componentes y definición de microservicios.

Esta fase proporciona diagramas de casos de uso refinados que incorporan los datos involucrados en cada funcionalidad, lo que permite identificar con claridad los puntos de entrada y salida de información. También se elaboran fichas descriptivas por cada dato, especificando su nombre, tipo, origen, destino y validaciones asociadas. Además, se establecen relaciones funcionales entre casos de uso, fundamentadas en el intercambio y reutilización de dichos datos.

8.4. Identificación de interacciones entre casos de uso

Esta fase tiene como propósito analizar las relaciones funcionales entre los distintos casos de uso refinados, con el fin de identificar cómo se comunican entre sí los módulos del sistema y qué flujos de información los vinculan. El análisis parte directamente del refinamiento detallado realizado en la fase anterior, en donde ya se habían identificado los datos que intervienen en cada funcionalidad.

Gracias a esta base, es posible determinar qué casos de uso requerían información producida por otros. Esta dependencia evidencia una interacción funcional que va más allá de una simple coincidencia temática y que, por tanto, debe ser modelada explícitamente.

Para capturar estas relaciones, se construyen diagramas de interacciones entre casos de uso, en los cuales se representó gráficamente el flujo de datos entre ellos. La dirección de cada conexión indica la dependencia funcional: si un caso de uso A consulta información generada por un caso B, se traza una flecha desde A hacia el módulo en el que se encuentra el caso de uso B y viceversa para poder tener un análisis de qué casos de uso son solicitados a otros módulos y qué módulos solicitan los casos de uso identificados en el módulo actual.

Este análisis de interacciones no solo permite comprender mejor la dinámica operativa del sistema, sino que también sirvió como insumo directo para la identificación de componentes funcionales coherentes, así como para el diseño de contratos de servicio entre módulos. Las relaciones detectadas en esta etapa serían posteriormente traducidas en canales de comunicación y dependencias entre microservicios.

El resultado de esta fase consiste en un diagrama de interacciones entre casos de uso, en el que se anotan los flujos de datos y las direcciones funcionales que representan las dependencias operativas entre procesos. También se construye un listado de dependencias cruzadas entre funcionalidades, y se identifican los datos compartidos que servirán como base para el diseño posterior de las interfaces de servicio.

8.5. Identificación y clasificación de componentes (4SRS)

Como se explica en la sección 4 los casos de uso refinados se transforman en componentes técnicos mediante la metodología 4SRS, que clasifica los elementos en componentes de tipo interfaz (i-type), datos (d-type) y control (c-type). Esto permite un mapeo claro hacia una arquitectura orientada a microservicios.

Como producto principal de esta fase se construye la tabla 4SRS-MSLA, en la cual se desglosan y categorizan los componentes identificados a partir de los casos de uso refinados. Esta clasificación distingue claramente las funcionalidades de tipo interfaz, control y datos, lo que facilita su posterior agrupación en microservicios coherentes.

8.6. Identificación de microservicios

Los componentes clasificados se agrupan en unidades lógicas que representan microservicios. Esta agrupación se realiza respetando los principios de cohesión y bajo acoplamiento, además de los límites de subdominios definidos con DDD (Domain-Driven Design).

Para abstraer un ambiente de una arquitectura basada en microservicios se utiliza un diagrama de bloques donde cada bloque referencia un módulo que ofrece servicios y con ayuda de una flecha que apunta hacia el módulo que ofrece el microservicio con un nombre que hace referencia al nombre del método que se va a implementar.

Toda esta abstracción es necesaria para tener una buena comunicación con los stakeholders donde el diagrama de bloques implementado no requiere un alto entendimiento a nivel técnico de diseño de arquitecturas basadas en microservicios.

A partir de esta fase se obtiene un diagrama de bloques que representa la agrupación lógica de los componentes identificados en microservicios. Esta representación visual permite delimitar con claridad las fronteras funcionales de cada servicio y su alineación con los subdominios del negocio.

8.7. Análisis de mensajes para la comunicación entre microservicios

Con ayuda del diagrama de bloques se analizan los flujos de comunicación entre microservicios para definir los mensajes que serán intercambiados, así como los esquemas de datos y APIs necesarias para garantizar interoperabilidad y autonomía.

En la tabla generada se tiene en cuenta el origen, destino, nombre de la API, tipo de operación, datos enviados, datos devueltos y datos a validar. Esta tabla se puede revisar con ayuda de los stakeholders ya que los campos a escribir no necesitan un conocimiento técnico para el desarrollo de software y aun así esta información es valiosa para el diseño de una arquitectura basada en microservicios.

Esta fase culmina con la elaboración de una tabla de mensajes entre microservicios. Esta información resulta esencial para el diseño de contratos y APIs que regulan la interoperabilidad entre los servicios.

8.8. Modelado con SoaML

Se utiliza la notación SoaML para representar la arquitectura orientada a servicios. Se definen los participantes de servicio, contratos, interfaces, roles consumidores y proveedores, así como las interacciones entre ellos.

Esta fase da lugar a una serie de diagramas SoaML que representan los servicios, interfaces, contratos y participantes involucrados en la arquitectura. A través de estos modelos, se define de manera formal la interacción entre los microservicios, estableciendo los roles de proveedor y consumidor, así como los puntos de conexión entre ellos. Todo esto conforma una arquitectura conceptual clara y estandarizada, orientada a servicios.

8.9. Representación de la implementación física

Esta fase permite modelar cómo se desplegarán los microservicios en entornos reales. Se asocian los servicios a contenedores, bases de datos, puertos y endpoints específicos para su futura implementación y despliegue.

Como resultado, se construye un diagrama de implementación que representa la distribución física de los servicios en la infraestructura tecnológica. Este modelo permite visualizar cómo se despliegan los microservicios en contenedores o nodos específicos, junto con sus respectivas dependencias, bases de datos, puntos de acceso y configuraciones técnicas necesarias para su ejecución.

8.10. Codificación, testeo y construcción del prototipo

Finalmente, se implementa un prototipo funcional que refleja la arquitectura definida. Se desarrollan los microservicios en Spring Boot, se construye la interfaz de usuario en React y se ejecutan pruebas para validar la interoperabilidad.

Los entregables de esta fase son el código fuente de microservicios, estructura del frontend y validación del prototipo.

9. Conclusiones, Discusión y Trabajo futuro

9.1. Conclusiones

Con el objetivo de diseñar una arquitectura basada en microservicios que respalde el desarrollo de aplicaciones empresariales, este trabajo de titulación se enfocó en definir una solución técnica que permita gestionar procesos comerciales de forma modular, escalable y mantenible. A lo largo del desarrollo del proyecto, se aplicaron metodologías de modelado orientadas a servicios y se construyeron diferentes vistas arquitectónicas para garantizar una comprensión clara del sistema tanto a nivel global como específico. Como resultado se concreto una metodología que se basa en los pasos seguidos en el caso de estudio para el diseño de una arquitectura basada en microservicos para aplicaciones de gestión empresarial. A continuación, se analiza el grado de cumplimiento de los objetivos planteados para esta investigación.

Respecto al primer objetivo específico, “Obtener los requerimientos que debe satisfacer el sistema: Identificar y documentar los requerimientos funcionales y no funcionales del sistema”, este fue alcanzado mediante un proceso de levantamiento de información junto al stakeholder del sistema. Se utilizaron diagramas de casos de uso y especificaciones textuales para identificar las funcionalidades clave y restricciones del sistema. El resultado fue un conjunto de diagramas de casos de uso claramente documentados, que sirvieron como punto de partida para el diseño lógico del sistema. Estos casos de uso fueron clasificados en tres niveles jerárquicos: los de primer nivel representan las funcionalidades globales del sistema desde una perspectiva de negocio, los de segundo nivel detallan los procesos específicos dentro de cada caso de uso de primer nivel identificado como por ejemplo Gestión de Compras o Gestión Contable, y los de tercer nivel profundizan en acciones concretas que realiza el sistema dentro de un flujo funcional, describiendo con mayor granularidad los pasos operativos de cada proceso.

Con relación al segundo objetivo específico, “Diseñar la arquitectura de microservicios del sistema: Diseñar la arquitectura del sistema, incluyendo vistas arquitectónicas que se consideren relevantes para la comunicación con diferentes partes interesadas”, se construyó una arquitectura global del sistema, haciendo uso del método 4SRS-MSLA y del lenguaje de modelado SoaML. Durante esta fase se desarrollaron múltiples vistas arquitectónicas, como el diagrama de bloques, el diagrama de participantes y los contratos de servicio, que facilitaron la identificación de módulos, servicios expuestos, responsabilidades y relaciones de dependencia. Esta arquitectura permitió comunicar de forma clara, tanto al equipo técnico como a las partes interesadas, cómo estaría distribuido funcionalmente el sistema en términos de microservicios.

En cuanto al tercer objetivo, “Diseñar la arquitectura detallada de un microservicio específico”, se procedió a profundizar en el diseño del microservicio de Compras, seleccionándolo por su relevancia dentro del sistema. En esta fase se especificaron las interfaces ofrecidas, las

estructuras de datos (mensajes), los contratos de servicio, y se describieron los componentes internos que conforman su lógica de funcionamiento, como controladores, servicios, repositorios y DTOs. A diferencia del objetivo anterior, esta etapa se centró exclusivamente en un microservicio particular, permitiendo representar con mayor detalle su comportamiento técnico, sin incluir aún la implementación física.

El cuarto objetivo específico, “Probar la factibilidad de implementación del diseño arquitectónico propuesto”, fue abordado a través del desarrollo de un prototipo funcional. Este prototipo incluyó los microservicios de Compras, Productos y Contabilidad, desplegados mediante Spring Boot en un entorno controlado con Eureka, Config Server y API Gateway. La validación se llevó a cabo mediante un caso funcional representativo: la creación de una factura de compra que consume productos desde otro microservicio. Esta prueba permitió verificar la interoperabilidad entre servicios, el cumplimiento de los contratos definidos y la capacidad de integración del frontend con la arquitectura propuesta. Se concluyó que la arquitectura diseñada es factible de implementar y operativamente funcional.

Para concluir, respecto al objetivo general ”Diseñar una arquitectura basada en microservicios para el desarrollo de aplicaciones empresariales que respalden los procesos comerciales, con el objetivo de mejorar la escalabilidad, flexibilidad y mantenimiento”, se dictamina que el trabajo ha cumplido a cabalidad su objetivo central. La arquitectura diseñada encarna los principios de desacoplamiento, modularidad y escalabilidad, resultando en aplicaciones más sencillas de mantener y más resistentes al cambio. El prototipo validado es una prueba tangible de que la implementación de esta arquitectura es factible en contextos reales, abriendo perspectivas para su desarrollo futuro y la expansión de los microservicios faltantes.

9.2. Contribuciones

Los resultados obtenidos en este trabajo de titulación pueden aplicarse directamente al desarrollo y estructuración de sistemas empresariales modulares, en especial aquellos que requieren adaptabilidad, mantenimiento ágil y escalabilidad. La metodología para el diseño de una arquitectura basada en microservicios propuesta proporciona un marco claro y replicable para diseñar soluciones orientadas a servicios, aplicables en sectores como comercio, contabilidad, inventario y otros procesos organizacionales que requieren alta disponibilidad y evolución constante.

Una de las principales contribuciones de este trabajo es el diseño completo de una arquitectura orientada a microservicios desarrollada bajo el método 4SRS-MSLA, complementada con la notación SoaML. Este enfoque estructurado permitió establecer vistas arquitectónicas que abarcan desde el modelado de casos de uso en niveles jerárquicos hasta la definición de participantes, interfaces y contratos de servicio. Estas representaciones no solo mejoran la trazabilidad y la comunicación con las partes interesadas, sino que sirven como base sólida para futuras implementaciones en entornos reales.

Como parte del proceso de diseño colaborativo, se implementó el uso de un diagrama de bloques y una tabla de APIs como herramientas clave para facilitar el trabajo conjunto con el stakeholder experto en el dominio. Estas representaciones fueron fundamentales para representar de manera clara y accesible las relaciones funcionales entre los módulos, definir los puntos de integración y visualizar los servicios que cada microservicio debía ofrecer o consumir. Su uso contribuyó significativamente a mejorar la comunicación técnica entre el desarrollador y el experto del negocio, y sirvió como puente intermedio entre el análisis de requerimientos y el modelado formal de diagramas UML y SoaML. Esta estrategia puede ser considerada una buena práctica para futuros proyectos donde se requiera validar arquitectura junto con personas no técnicas.

Asimismo, se implementó un prototipo funcional que demuestra la viabilidad del diseño propuesto, integrando los microservicios de Compras, Productos y Contabilidad. Este prototipo permitió validar la interoperabilidad entre módulos desacoplados, así como la capacidad del frontend para consumir servicios REST mediante un API Gateway. Esta validación representa una evidencia concreta de que la arquitectura desarrollada es factible, aplicable y útil para proyectos empresariales que buscan modernizar sus sistemas monolíticos o construir soluciones distribuidas desde cero.

Otra contribución importante fue la sistematización del uso de técnicas de diseño orientadas a servicios en conjunto con herramientas modernas del ecosistema Spring Boot, como Eureka, Config Server y API Gateway. Estas tecnologías fueron empleadas para construir un entorno controlado de despliegue, capaz de reflejar el comportamiento de una arquitectura real en producción.

Finalmente, el desarrollo de una técnica escalonada de modelado que va desde el análisis de requerimientos hasta la implementación parcial demuestra un camino práctico que puede ser replicado en investigaciones futuras. Este trabajo aporta tanto a nivel metodológico como aplicado, ofreciendo un marco reutilizable para diseñar e implementar arquitecturas de microservicios en distintos dominios del software empresarial.

9.3. Discusión de resultados

En este trabajo, se logró diseñar una arquitectura orientada a microservicios aplicable a sistemas empresariales, utilizando el método 4SRS-MSLA como base para la transformación de requerimientos funcionales en una arquitectura lógica estructurada. Esta arquitectura no solo fue definida a nivel conceptual, sino que además fue parcialmente implementada y validada mediante un prototipo funcional, lo que permitió comprobar su aplicabilidad en escenarios reales. A continuación, se analizan los principales resultados alcanzados, así como su relación con trabajos previos y con la literatura especializada en microservicios.

El método 4SRS-MSLA, propuesto por Santos et al. (2019), ha demostrado ser una herramienta efectiva para guiar la derivación de una arquitectura lógica desde modelos de casos

de uso en UML, permitiendo generar componentes que posteriormente se transforman en participantes SoaML. Este enfoque facilitó una transición ordenada desde el análisis funcional hasta la definición de microservicios con sus respectivos contratos, canales de comunicación y estructuras internas. En nuestro caso, el uso de este método permitió estructurar de manera jerárquica los casos de uso (primer, segundo y tercer nivel), y segmentar correctamente el sistema en microservicios alineados con capacidades de negocio.

Comparado con otras técnicas de modelado como DDMM(Domain-Driven Design) UML(Unified Modeling Language) o Microservice DSL(Domain-Specific Language), el enfoque 4SRS-MSLA se destaca por su capacidad de mantener trazabilidad entre los requisitos funcionales iniciales y los microservicios derivados, lo cual fue especialmente útil en nuestro proceso de modelado colaborativo con el stakeholder. Esta característica de trazabilidad ha sido resaltada en estudios como el de van Amerongen (2023), quien reconoce la necesidad de métodos que soporten la especificación estructurada y completa de arquitecturas de microservicios.

Una aportación distintiva en nuestra investigación fue la inclusión de un diagrama de bloques funcionales y una tabla de APIs como herramientas intermedias entre el análisis de requerimientos y el modelado formal. Estas herramientas sirvieron como puente de comunicación entre el analista técnico y el stakeholder experto en el dominio del sistema, facilitando la validación temprana de los límites de contexto, las dependencias entre módulos y los servicios requeridos. Su uso fortaleció el modelado posterior en SoaML y permitió diseñar interfaces más coherentes con las necesidades del negocio.

Desde una perspectiva práctica, la implementación parcial de los microservicios de Compras, Productos y Contabilidad permitió verificar que los contratos definidos en los diagramas SoaML son implementables en un entorno basado en Spring Boot. Se logró validar la interoperabilidad entre servicios mediante un API Gateway, así como la separación efectiva de responsabilidades y el cumplimiento del principio de responsabilidad única (SRP).

A diferencia de otros estudios centrados exclusivamente en el diseño o la implementación de microservicios, esta investigación abarcó ambas fases, demostrando cómo una arquitectura lógica bien especificada puede facilitar una implementación funcional coherente. Se evidenció que el método 4SRS-MSLA no solo guía la segmentación técnica del sistema, sino que también permite prever interacciones clave, validar dependencias y anticipar posibles cuellos de botella en la comunicación entre módulos.

Finalmente, la propuesta desarrollada no solo es válida para el caso específico de estudio, sino que puede ser extrapolada a otros contextos donde se requiere transformar aplicaciones monolíticas en arquitecturas distribuidas o construir sistemas empresariales desde cero siguiendo principios de diseño moderno y escalable.

Referencias

[1] State University of Trade and Economics et al. “Features of Microservices Architecture in E-Commerce Systems”. In: *Mathematical machines and systems* 3 (2023), pp. 51–58. ISSN: 10289763. DOI: 10.34121/1028-9763-2023-3-51-58. (Visited on 12/06/2024).

[2] Mendoza Linares Jersson German. “Metodología de conversión de aplicaciones monolíticas a microservicios desplegable en la nube para pequeñas empresas”. In: *Repositorio Institucional - USS* (2022). (Visited on 12/06/2024).

[3] Konrad Gos and Wojciech Zabierowski. “The Comparison of Microservice and Monolithic Architecture”. In: *2020 IEEE XVIth International Conference on the Perspective Technologies and Methods in MEMS Design (MEMSTECH)*. Apr. 2020, pp. 150–153. DOI: 10.1109/MEMSTECH49584.2020.9109514. (Visited on 12/03/2024).

[4] Freddy Tapia et al. “From Monolithic Systems to Microservices: A Comparative Study of Performance”. In: *Applied Sciences* 10.17 (Jan. 2020), p. 5797. ISSN: 2076-3417. DOI: 10.3390/app10175797. (Visited on 12/03/2024).

[5] I. Asrowardi, S. D. Putra, and E. Subyantoro. “Designing Microservice Architectures for Scalability and Reliability in E-Commerce”. In: *Journal of Physics: Conference Series* 1450.1 (Feb. 2020), p. 012077. ISSN: 1742-6596. DOI: 10.1088/1742-6596/1450/1/012077. (Visited on 12/03/2024).

[6] Meng Wu, Xiangqian Ding, and Ruichun Hou. “Design and Implementation of B2B E-commerce Platform Based on Microservices Architecture”. In: *Proceedings of the 2nd International Conference on Computer Science and Software Engineering*. CSSE ’19. New York, NY, USA: Association for Computing Machinery, May 2019, pp. 30–34. ISBN: 978-1-4503-7172-8. DOI: 10.1145/3339363.3339369. (Visited on 12/03/2024).

[7] Davide Taibi et al. “Microservices in Agile Software Development: A Workshop-Based Study into Issues, Advantages, and Disadvantages”. In: *Proceedings of the XP2017 Scientific Workshops*. XP ’17. New York, NY, USA: Association for Computing Machinery, May 2017, pp. 1–5. ISBN: 978-1-4503-5264-2. DOI: 10.1145/3120459.3120483. (Visited on 12/03/2024).

[8] Nuno Santos et al. “A Logical Architecture Design Method for Microservices Architectures”. In: *Proceedings of the 13th European Conference on Software Architecture - Volume 2*. Paris France: ACM, Sept. 2019, pp. 145–151. ISBN: 978-1-4503-7142-1. DOI: 10.1145/3344948.3344991. (Visited on 12/03/2024).

[9] Nuno António de Lira Fernandes Faria dos Santos. “An agile process for modeling logical architectures: demonstration cases from large-scale software projects”. doctoralThesis. Nov. 2020. (Visited on 12/06/2024).

[10] Nuno Santos et al. “Inputs from a Model-Based Approach Towards the Specification of Microservices Logical Architectures: An Experience Report”. In: *Product-Focused Software Process Improvement*. Ed. by Xavier Franch, Tomi Männistö, and Silverio Martínez-

Fernández. Cham: Springer International Publishing, 2019, pp. 473–488. ISBN: 978-3-030-35333-9. DOI: 10.1007/978-3-030-35333-9_33.

[11] Guozhi Liu et al. "Microservices: Architecture, Container, and Challenges". In: *2020 IEEE 20th International Conference on Software Quality, Reliability and Security Companion (QRS-C)*. Dec. 2020, pp. 629–635. DOI: 10.1109/QRS-C51114.2020.00107. (Visited on 12/03/2024).

[12] Gary Ricardo Campaña Ramírez. "Aplicabilidad de los patrones arquitectura de microservicios: aplicación, aplicación-infraestructura, infraestructura, enfocado en la lógica de la compraventa de libros : módulo de aplicabilidad de los patrones de tipo aplicación - infraestructura." In: (Jan. 2025). (Visited on 06/13/2025).

[13] Edward Yourdon and Larry L. Constantine. *Structured Design: Fundamentals of a Discipline of Computer Program and Systems Design*. 1st. USA: Prentice-Hall, Inc., Jan. 1979. ISBN: 978-0-13-854471-3.

[14] David Thomas and Andrew Hunt. *The Pragmatic Programmer: Your Journey to Mastery, 20th Anniversary Edition*. Addison-Wesley Professional, July 2019. ISBN: 978-0-13-595691-5.

[15] K.J. Lieberherr and I.M. Holland. "Assuring Good Style for Object-Oriented Programs". In: *IEEE Software* 6.5 (Sept. 1989), pp. 38–48. ISSN: 1937-4194. DOI: 10.1109/52.35588. (Visited on 06/13/2025).

[16] Martin Fowler. *Inversion of Control Containers and the Dependency Injection Pattern*. https://martinfowler.com/articles/injection.html. (Visited on 06/13/2025).

[17] *Spring Boot :: Spring Boot*. https://docs.spring.io/spring-boot/index.html. (Visited on 01/14/2025).

[18] *Describir la UI – React*. https://es.react.dev/learn/describing-the-ui. (Visited on 01/14/2025).

[19] *MySQL en detalle: guía para dominar esta potente base de datos*. https://www.oracle.com/lad/mysql/wha is-mysql/. (Visited on 06/12/2025).

[20] *About Railway — Railway Docs*. https://docs.railway.com/overview/about-railway. (Visited on 06/12/2025).

[21] Nicola Dragoni et al. "Microservices: Yesterday, Today, and Tomorrow". In: *Present and Ulterior Software Engineering*. Ed. by Manuel Mazzara and Bertrand Meyer. Cham: Springer International Publishing, 2017, pp. 195–216. ISBN: 978-3-319-67425-4. DOI: 10.1007/978-3-319-67425-4_12. (Visited on 01/07/2025).

[22] Alan González Heredia et al. "Prácticas de los equipos de desarrollo de microservicios: un mapeo sistemático de la literatura". In: *ReCIBE, Revista electrónica de Computación, Informática, Biomédica y Electrónica* 10.1 (Sept. 2021), pp. C2–26. ISSN: 2007-5448. DOI: 10.32870/recibe.v10i1.218. (Visited on 06/19/2025).

[23] Hossam Hassan, Manal A. Abdel-Fattah, and Wael Mohamed. "Migrating from Monolithic to Microservice Architectures: A Systematic Literature Review". In: *International Journal of Advanced Computer Science and Applications (ijacsa)* 15.10 (Dec. 2024). ISSN: 2156-5570. DOI: 10.14569/IJACSA.2024.0151013. (Visited on 01/07/2025).

[24] Idris Oumoussa and Rajaa Saidi. “Evolution of Microservices Identification in Monolith Decomposition: A Systematic Review”. In: *IEEE Access* 12 (2024), pp. 23389–23405. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2024.3365079. (Visited on 01/07/2025).

[25] J. van Amerongen. *A Qualitative Comparison of Specification Techniques for Microservice Architectures*. http://essay.utwente.nl/94433/. Info:Eu-Repo/Semantics/bachelorThesis. Feb. 2023. (Visited on 12/03/2024).

[26] Dennys Xavier Landy Rivera. “Plataforma web de facturación como servicio”. MA thesis. E.T.S.I de Sistemas Informáticos (UPM), July 2019. (Visited on 06/19/2025).

[27] Sabina Moreno Morlón. *Sistema para la gestión de una PYME basado en microservicios*. https://oa.upm.es/76451/. info:eu-repo/semantics/bachelorThesis. Madrid, España, Oct. 2023. (Visited on 06/19/2025).

[28] Diana Regina Navarro Ruiz and Carlos Manuel Carazas Calizaya. “Aplicación web de solicitud de préstamos online para PYMEs”. In: *Universidad Peruana de Ciencias Aplicadas (UPC)* (July 2024). (Visited on 06/19/2025).

[29] Mohamed Amarani Darraz. “Análisis e implantación de una arquitectura back-end basada en microservicios sobre una plataforma cloud”. Bachelor thesis. Universitat Politècnica de Catalunya, Oct. 2021. (Visited on 06/19/2025).

Anexos

Anexo A: Tabla de Componentes para los casos de uso de Gestión de Facturas de Compra

Tabla 7: Componentes identificados en los casos de uso de Gestión de Facturas de Compra

ID del Caso de Uso	Nombre del Caso de Uso	Clasificación	Eliminación Local	Descripción del Componente	Eliminación Global	Renombrar	Representa a	Nombre del Componente	Paquete	Componente Relacionado
G.D.C.1.1	Crear factura de compra	i-type	No aplica	Pantalla inicial diseñada para ingresar nuevos ...	No aplica	No aplica	Interfaz de usuario o sistema	i_crear_factura_de_compra	Compras	c_crear_factura_de_compra
G.D.C.1.1	Crear factura de compra	d-type	No aplica	Modelo de base de datos encargado de almacenar ...	No aplica	No aplica	Modelo de datos	d_crear_factura_de_compra	Compras	c_crear_factura_de_compra
G.D.C.1.1	Crear factura de compra	c-type	No aplica	Conjunto de reglas que validan los campos oblig...	No aplica	No aplica	Lógica de negocio	c_crear_factura_de_compra	Compras	i_crear_factura_de_compra
G.D.C.1.2	Modificar factura de compra	i-type	No aplica	Ventana interactiva que permite al usuario loca...	No aplica	No aplica	Interfaz de usuario o sistema	i_modificar_factura_de_compra	Compras	c_modificar_factura_de_compra
G.D.C.1.2	Modificar factura de compra	d-type	No aplica	Estructura que gestiona los cambios aplicados a...	No aplica	No aplica	Modelo de datos	d_modificar_factura_de_compra	Compras	c_modificar_factura_de_compra
G.D.C.1.2	Modificar factura de compra	c-type	No aplica	Mecanismo que asegura que las modificaciones cu...	No aplica	No aplica	Lógica de negocio	c_modificar_factura_de_compra	Compras	i_modificar_factura_de_compra
G.D.C.1.3	Desbloquear factura de compra	i-type	No aplica	Interfaz que habilita la opción de liberar fact...	No aplica	No aplica	Interfaz de usuario o sistema	i_desbloquear_factura_de_compra	Compras	c_login_usuario (MS_Usuarios)
G.D.C.1.3	Desbloquear factura de compra	d-type	No aplica	Registro auxiliar que permite actualizar el est...	No aplica	No aplica	Modelo de datos	d_desbloquear_factura_de_compra	Compras	c_login_usuario (MS_Usuarios)
G.D.C.1.3	Desbloquear factura de compra	c-type	No aplica	Proceso que evalúa condiciones para permitir la...	No aplica	No aplica	Lógica de negocio	c_desbloquear_factura_de_compra	Compras	c_login_usuario (MS_Usuarios)
G.D.C.1.4	Copiar factura de compra	i-type	No aplica	Formulario sencillo que asiste en la duplicació...	No aplica	No aplica	Interfaz de usuario o sistema	i_copiar_factura_de_compra	Compras	c_copiar_factura_de_compra
G.D.C.1.4	Copiar factura de compra	d-type	No aplica	Tabla temporal o lógica que respalda la duplica...	No aplica	No aplica	Modelo de datos	d_copiar_factura_de_compra	Compras	c_copiar_factura_de_compra
G.D.C.1.4	Copiar factura de compra	c-type	No aplica	Función que clona datos existentes y ajusta ref...	No aplica	No aplica	Lógica de negocio	c_copiar_factura_de_compra	Compras	i_copiar_factura_de_compra
G.D.C.1.5	Exportar factura de compra	i-type	No aplica	Vista que permite seleccionar formatos y destin...	No aplica	No aplica	Interfaz de usuario o sistema	i_exportar_factura_de_compra	Compras	c_exportar_factura_de_compra
G.D.C.1.5	Exportar factura de compra	d-type	No aplica	Contenedor de datos con formato adaptable para ...	No aplica	No aplica	Modelo de datos	d_exportar_factura_de_compra	Compras	c_exportar_factura_de_compra
G.D.C.1.5	Exportar factura de compra	c-type	No aplica	Procedimiento que transforma y prepara los dato...	No aplica	No aplica	Lógica de negocio	c_exportar_factura_de_compra	Compras	i_exportar_factura_de_compra
G.D.C.1.6	Ver información de factura de compra específica	i-type	No aplica	Sección dedicada a mostrar los detalles complet...	No aplica	No aplica	Interfaz de usuario o sistema	i_ver_información_de_factura_de_compra_específica	Compras	c_ver_información_de_factura_de_compra_específica
G.D.C.1.6	Ver información de factura de compra específica	d-type	No aplica	Vista estructurada que organiza y presenta la i...	No aplica	No aplica	Modelo de datos	d_ver_información_de_factura_de_compra_específica	Compras	c_ver_información_de_factura_de_compra_específica
G.D.C.1.6	Ver información de factura de compra específica	c-type	No aplica	Algoritmo que recupera y organiza la informació...	No aplica	No aplica	Lógica de negocio	c_ver_información_de_factura_de_compra_específica	Compras	i_ver_información_de_factura_de_compra_específica
G.D.C.1.7	Listar proveedores	i-type	No aplica	Pantalla de consulta con acceso rápido al lista...	No aplica	No aplica	Interfaz de usuario o sistema	i_listar_proveedores	Compras	c_listar_proveedores
G.D.C.1.7	Listar proveedores	d-type	No aplica	Relación tabular con atributos esenciales de ca...	No aplica	No aplica	Modelo de datos	d_listar_proveedores	Compras	c_listar_proveedores
G.D.C.1.7	Listar proveedores	c-type	No aplica	Consulta optimizada que obtiene la lista actual...	No aplica	No aplica	Lógica de negocio	c_listar_proveedores	Compras	c_listar_proveedores
G.D.C.1.8	Listar productos en sistema	i-type	No aplica	Vista integrada donde se muestran todos los pro...	No aplica	No aplica	Interfaz de usuario o sistema	i_listar_productos_en_sistema	Compras	c_listar_productos_en_sistema(MS_Productos)
G.D.C.1.8	Listar productos en sistema	d-type	No aplica	Catálogo de productos gestionado por el módulo ...	No aplica	No aplica	Modelo de datos	d_listar_productos_en_sistema	Compras	c_listar_productos_en_sistema(MS_Productos)
G.D.C.1.8	Listar productos en sistema	c-type	No aplica	Rutina que reúne los artículos disponibles con ...	No aplica	No aplica	Lógica de negocio	c_listar_productos_en_sistema	Compras	i_listar_productos_en_sistema(MS_Productos)
G.D.C.1.9	Listar diarios	i-type	No aplica	Ventana que presenta los diarios contables asoc...	No aplica	No aplica	Interfaz de usuario o sistema	i_listar_diarios	Compras	c_listar_diarios (MS_Contabilidad)
G.D.C.1.9	Listar diarios	d-type	No aplica	Registro contable con claves, fechas y valores ...	No aplica	No aplica	Modelo de datos	d_listar_diarios	Compras	c_listar_diarios (MS_Contabilidad)
G.D.C.1.9	Listar diarios	c-type	No aplica	Lógica que vincula movimientos contables con su...	No aplica	No aplica	Lógica de negocio	c_listar_diarios	Compras	c_listar_diarios (MS_Contabilidad)
G.D.C.1.10	Listar retenciones de compra	i-type	No aplica	Interfaz diseñada para acceder al historial de ...	No aplica	No aplica	Interfaz de usuario o sistema	i_listar_retenciones_de_compra	Compras	c_listar_retenciones (MS_Contabilidad)
G.D.C.1.10	Listar retenciones de compra	d-type	No aplica	Base de datos estructurada que almacena porcent...	No aplica	No aplica	Modelo de datos	d_listar_retenciones_de_compra	Compras	c_listar_retenciones (MS_Contabilidad)
G.D.C.1.10	Listar retenciones de compra	c-type	No aplica	Procedimiento que recopila todas las retencione...	No aplica	No aplica	Lógica de negocio	c_listar_retenciones_de_compra	Compras	c_listar_retenciones (MS_Contabilidad)
G.D.C.1.11	Certificar factura de compra	i-type	No aplica	Panel que guía al usuario en el proceso de vali...	No aplica	No aplica	Interfaz de usuario o sistema	i_certificar_factura_de_compra	Compras	c_certificar_factura_de_compra
G.D.C.1.11	Certificar factura de compra	d-type	No aplica	Campo específico que almacena el estado de cert...	No aplica	No aplica	Modelo de datos	d_certificar_factura_de_compra	Compras	c_certificar_factura_de_compra
G.D.C.1.11	Certificar factura de compra	c-type	No aplica	Algoritmo de validación final que asegura el cu...	No aplica	No aplica	Lógica de negocio	c_certificar_factura_de_compra	Compras	i_certificar_factura_de_compra
G.D.C.1.12	Reenviar mensaje de factura de compra	i-type	No aplica	Ventana operativa que permite reenviar notifica...	No aplica	No aplica	Interfaz de usuario o sistema	i_reenviar_mensaje_de_factura_de_compra	Compras	c_reenviar_mensaje_de_factura_de_compra
G.D.C.1.12	Reenviar mensaje de factura de compra	d-type	No aplica	Log o registro de control que guarda el histori...	No aplica	No aplica	Modelo de datos	d_reenviar_mensaje_de_factura_de_compra	Compras	c_reenviar_mensaje_de_factura_de_compra
G.D.C.1.12	Reenviar mensaje de factura de compra	c-type	No aplica	Proceso que gestiona la reemisión de mensajes g...	No aplica	No aplica	Lógica de negocio	c_reenviar_mensaje_de_factura_de_compra	Compras	i_reenviar_mensaje_de_factura_de_compra