EDUCACIÓN
SECRETARÍA DE EDUCACIÓN PÚBLICA

TECNOLÓGICO NACIONAL DE MÉXICO

Tecnológico Nacional de México

Centro Nacional de Investigación y Desarrollo Tecnológico

Tesis de Maestría

Metodología en la determinación de la granularidad de un Microservicio

presentada por

Ing. César Merino Ibáñez

como requisito para la obtención del grado de

Maestro en Ciencias de la Computación

Director de tesis

Dr. Juan Carlos Rojas Pérez

Cuernavaca, Morelos, México. Enero de 2023.

Centro Nacional de Investigación y Desarrollo Tecnológico
Departamento de Ciencias Computacionales

Cuernavaca, Mor., 29/noviembre/2019

OFICIO No. DCC/103/2019
Asunto: Aceptación de documento de tesis
CENIDET-AC-004-M14-OFICIO

DR. CARLOS MANUEL ASTORGA ZARAGOZA
SUBDIRECTOR ACADÉMICO
PRESENTE

Por este conducto, los integrantes de Comité Tutorial del C. CÉSAR MERINO IBÁÑEZ, con número de control M21CE018, de la Maestría en Ciencias en Ingeniería de Software, le informamos que hemos revisado el trabajo de tesis de grado titulado "**METODOLOGÍA EN LA DETERMINACIÓN DE LA GRANULARIDAD DE UN MICROSERVICIO**"" y hemos encontrado que se han atendido todas las observaciones que se le indicaron, por lo que hemos acordado aceptar el documento de tesis y le solicitamos la autorización de impresión definitiva.

DR. JUAN CARLOS ROJAS PÉREZ
Director de tesis

EDUCACIÓN | TECNOLÓGICO NACIONAL DE MÉXICO
08 DIC 2022
CENTRO NACIONAL DE INVESTIGACIÓN Y DESARROLLO TECNOLÓGICO
SERVICIOS ESCOLARES
RECIBIDO

DR. RENÉ SANTAOLAYA SALGADO
Revisor 1

M.C. MARIO GUILLÉN RODRÍGUEZ
Revisor 2

C.c.p. Depto. Servicios Escolares.
Expediente / Estudiante

JGGS/ibm

Cuernavaca, Mor., 08/diciembre/2022
No. De Oficio: SAC/180/2022
Asunto: Autorización de impresión de tesis

CÉSAR MERINO IBÁÑEZ
CANDIDATO AL GRADO DE MAESTRO EN CIENCIAS DE LA COMPUTACIÓN
P R E S E N T E

Por este conducto, tengo el agrado de comunicarle que el Comité Tutorial asignado a su trabajo de tesis titulado **"Metodología en la determinación de la granularidad de un microservicio",** ha informado a esta Subdirección Académica, que están de acuerdo con el trabajo presentado. Por lo anterior, se le autoriza a que proceda con la impresión definitiva de su trabajo de tesis.

Esperando que el logro del mismo sea acorde con sus aspiraciones profesionales, reciba un cordial saludo.

A T E N T A M E N T E
Excelencia en Educación Tecnológica®
"Educación Tecnológica al Servicio de México"

EDUCACIÓN | TECNOLÓGICO NACIONAL DE MÉXICO
09 DIC 2022
CENTRO NACIONAL DE INVESTIGACIÓN Y DESARROLLO TECNOLÓGICO
SERVICIOS ESCOLARES
RECIBIDO

EDUCACIÓN | TECNOLÓGICO NACIONAL DE MÉXICO
CENTRO NACIONAL DE INVESTIGACIÓN Y DESARROLLO TECNOLÓGICO
SUBDIRECCIÓN ACADÉMICA

DR. CARLOS MANUEL ASTORGA ZARAGOZA
SUBDIRECTOR ACADÉMICO

C. c. p. Departamento de Ciencias Computacionales
Departamento de Servicios Escolares

CMAZ/RMA

Dedicatoria

Con todo mi amor y cariño a mi familia entera, por darme siempre su apoyo incondicional, pero por sobre todo a mi mamá, porque ha sido mi motivación más grande para lograr cada una de las metas que me he propuesto.

Agradecimientos

A mi mamá Lulú, por todo su apoyo, motivación y sobre todo su amor que siempre me ha mostrado. A mi hermana y abuelos por su apoyo y cariño.

Al Consejo Nacional de Ciencia y Tecnología (CONACYT), así como al Centro Nacional de Investigación y Desarrollo Tecnológico, por darme la oportunidad de realizar mis estudios y por brindarme la beca durante la maestría.

Mi más profundo agradecimiento a mi director de tesis, el Dr. Juan Carlos Rojas Pérez, por ser mi guía en esta etapa de mi formación profesional, por todo su valioso y oportuno apoyo, y por brindarme su experiencia en el campo científico que se trabajó.

A todos los doctores de la línea, el Dr. Rene, la Dra. Olivia y la Dra. Blanca, también al Mtro. Mario Guillen, porque aprendí mucho de ustedes como investigadores. Son personas que admiro mucho y valoro todos sus consejos dados durante la maestría.

A todos mis compañeros del CENIDET, gracias por siempre estar dispuestos a apoyarme y por compartir sus conocimientos, gracias por todas las aventuras que se animaron a vivir conmigo e hicieron de esta una etapa muy agradable.

Por último, gracias a todos aquellos que desde la distancia siempre me brindaron su apoyo, que me motivaron a lograr esta meta y que siempre estaban al pendiente de mí.

Resumen

La arquitectura basada en microservicios es una forma actual de diseño y desarrollo de software, la cual está siendo implementada por muchas industrias. Algunos de los principales beneficios de desarrollar sistemas con microservicios se ven reflejados en el rendimiento, escalabilidad y mantenibilidad del software, sin embargo, al ser una arquitectura relativamente nueva existen factores que deben ser atendidos para su mejora, como es el caso de la determinación del tamaño o granularidad de los microservicios.

Aunque existen diferentes formas de determinar la granularidad, en este trabajo se presenta una metodología que engloba cuatro enfoques para la determinación del tamaño adecuado de un microservicio las cuales son: el modelo de dominio, los casos de uso, el diseño de la base de datos y la declaración de objetivos.

La metodología aquí propuesta fue probada en los casos de estudio: “Cinema” y “Tea Store”, abordando y validando únicamente el enfoque de casos de uso, mediante los estándares y métricas involucrados en dicha metodología.

Abstract

Microservices-based architecture is a current form of software design and development, which is being implemented by many industries. Some of the main benefits of developing systems with microservices are reflected in the performance, scalability and maintainability of the software, however, being a relatively new architecture there are factors that must be addressed for its improvement, as is the case of determining the size or granularity of microservices.

Although there are different ways to determining the granularity, this paper presents a methodology that includes four approaches to determine the appropriate size of a microservice, which are: the domain model, the use cases, the database design and the statement of objectives.

The methodology proposed here was tested in the case studies: "Cinema" and "Tea Store", addressing and validating only the use cases approach, using the standards and metrics involved in this methodology.

Contenido

Índice de figuras

Índice de tablas

Capítulo 1

1. Introducción

Desde hace algunos años, el desarrollo tecnológico en el país ha venido en incremento, y hoy día debido a la contingencia vivida se potencializó más este incremento. Muchas pequeñas, medianas y grandes empresas están optando por dar seguimiento a sus actividades desde un ámbito de software, debido a que trabajar con aplicaciones y servicios móviles o web demostró ser eficaz para mantener en pie a dichos negocios y seguir teniendo presencia en sus respectivos sectores.

Como se menciona, el desarrollo de software está jugando un papel esencial en toda esta dinámica. El crear software de calidad a cualquier nivel requiere del uso de metodologías, estándares y la aplicación de nuevos paradigmas. Cuando se desarrolla un proyecto de software se debería diseñar con la intención de crecer en cualquiera de las intenciones para las que se realice, por lo tanto, es entendible que aumentarán sus recursos y funcionalidades.

Partiendo de esto el uso de lenguajes de programación por sí solos no garantizan desarrollos que puedan ser reutilizables y extensibles [19].

Un problema actual que es enfrentado en el desarrollo de software es el mantenimiento a los monolitos, de acuerdo con Nicola Dragoni, Saverio Giallorenzo y Larisa Safina en el artículo Microservices: yesterday, today, and tomorrow, [1], un monolito es una aplicación de software cuyos módulos no pueden ser ejecutados independientemente.

Surge después la Arquitectura Orientada a Servicios (SOA), en esta arquitectura los componentes (funcionalidades) de la aplicación están relativamente separados y se complementan unos con otros mediante un protocolo de comunicación en red [10].

Aunque SOA supone una evolución hacia arquitecturas distribuidas, los componentes siguen sin ser lo suficientemente autónomos como para ser independientes [8], por ejemplo, si el Bus de Servicio Empresarial (ESB) se convierte en un punto de falla, afectaría a toda la aplicación, pues todos los servicios se comunican a través de este Bus, así mismo, si uno de estos se ralentiza, podría ocasionar que el ESB se atasque con las solicitudes de ese servicio, produciendo el mismo efecto de falla general para todo el sistema de software. Es aquí donde surgen los Microservicios, un paradigma reciente para tratar el problema presentado por los monolitos y que ha ganado un amplio reconocimiento y aceptación por la industria de software [2]. En el trabajo de Nicola Dragoni, Saverio Giallorenzo y Larisa Safina, se define un microservicio como un proceso cohesivo e independiente que interactúa a través de mensajes, mientras que James Lewis y Martin Fowler [3], contemplan a un microservicio como un estilo de arquitectura que pone énfasis en dividir el sistema en servicios pequeños y livianos que se construyen expresamente para realizar una función comercial muy cohesiva, y es una evolución del estilo de arquitectura tradicional orientada a servicios, desde la perspectiva de Jamshidi P., Mendonca N., y Tilkov S. en Microservices, The Journey So Far and Challenges Ahead, [5], los

microservicios son componentes granulares totalmente independientes, que funcionan como partes de un programa más amplio, dichas partes también llamadas módulos, pueden ser actualizados y desplegados sin requerir un solo cambio en el resto del sistema. Es importante entender que los microservicios no están creados para ser utilizados en todos los desarrollos de software, debido a que es más probable que se utilicen en situaciones en las que los costos superan por mucho a los beneficios. Una razón por la cual no podría ser implementada esta arquitectura sería que un proyecto se desarrolle mejor de forma monolítica [9].

Sin embargo, la arquitectura basada en microservicios como tecnología relativamente nueva también presenta algunos problemas o retos asociados con la modularización y refactorización, la granularidad, fallas, recuperación y reparación automática, y la seguridad [4] entre otros. Es aquí donde surge el concepto de "*granularidad*", el cual parte de dos aspectos importantes, la división de las funcionalidades a desarrollar en un sistema y el tamaño adecuado para cada una de estas funciones, ya que deben ser del tamaño justo para no ser repetitivas [12], este tamaño dependerá de cómo sea medido.

Este trabajo se centra en conocer el problema de la granularidad en microservicios e identificar cuáles son los enfoques utilizados para determinar dichos microservicios. Razonar objetivamente para definir cuál es tamaño correcto o adecuado de un microservicio es un problema actual que ha sido abordado sin llegar a una solución definitiva o aproximada.

1.1 Antecedentes

El Centro Nacional de Investigación y Desarrollo Tecnológico (CENIDET), a través del departamento de Ciencias Computacionales (DCC), ha desarrollado algunas investigaciones en relación al tema de microservicios, una de ellas es la tesis de maestría de Francisco León Pérez [54], la cual fue la primera investigación que introdujo el concepto de casos de uso como nivel correcto de granulación de servicios web, lo cual es aplicable también a microservicios. En este trabajo se menciona que cada caso de uso es considerado un componente reusable (servicio web o microservicio) ya que satisface una capacidad o requerimiento del dominio o negocio.

Otro antecedente que se tiene, es la tesis que presenta O. H. López [6], sobre un método para migrar los Servicios Web bajo protocolo SOAP hacia microservicios basados en REST, en este trabajo también se propone que cada caso de uso identificado sea considerado un microservicio, lo cual sienta un precedente a tomar en cuenta para las intenciones de este trabajo.

Así mismo y sirviendo de apoyo para cimentar las bases de la arquitectura de desarrollo basada en microservicios, se cuenta con la tesis de Iván Daniel Joya, quien a través de su trabajo [7] da a conocer conceptos base sobre el desarrollo de microservicios, centrándose en la adaptación de métricas para el desarrollo e implementación de microservicios.

1.2 Planteamiento del problema

Una de las preguntas a tomar en cuenta a la hora de trabajar con este tipo de arquitectura es: ¿Qué criterios seguir a la hora de dividir o agrupar las funcionalidades en microservicios?, como

tal no existe una forma establecida para asignar el número de funciones o la cantidad de código que debe contener un microservicio.

Uno de los principales desafíos en el desarrollo de software basado en microservicios es justamente determinar la granularidad adecuada de los microservicios, esto actualmente lo realizan los arquitectos y desarrolladores bajo su propio criterio, produciendo en algunos casos un bajo rendimiento e inestabilidad en el sistema [11], y afectaciones a la cohesión, coherencia, entre otros atributos.

La granularidad de un microservicio se refiere al tamaño del servicio. Definir objetivamente el tamaño adecuado o preciso de un microservicio repercutirá en atributos como son: rendimiento (performace), confiabilidad (reliability), escalabilidad (scalability), mantenibilidad (maintainability) y complejidad (complexity) [2]. Determinar la granularidad de un microservicio considerando solo objetivos técnicos puede conducir a una descomposición inadecuada de funcionalidades que repercute en el valor final del proyecto.

1.3 Objetivos

1.3.1 Objetivo general

Determinar la granularidad adecuada de un microservicio por medio de una metodología que considere aspectos estructurales y de comportamiento.

1.3.2 Objetivos específicos

- Realizar una búsqueda de los enfoques que se usan para determinar la granularidad de los microservicios de un sistema de software.
- Contemplar la experiencia del arquitecto de software y los diagramas de representación de las funcionalidades (casos de uso, diagramas de clase, historias de usuario, etc.) como aspectos estructurales y de comportamiento para determinar la granularidad.
- Validar los procesos involucrados en la metodología, mediante métricas o estándares que sustenten la correcta ejecución de los pasos.

1.4 Alcances

- La metodología incluye los atributos más relevantes en la determinación de la granularidad de un microservicio, los cuales serán definidos en el desarrollo de la investigación.
- En cuanto a la arquitectura de microservicios, la metodología considera la división en módulos independientes, cómo se construyen alrededor de funcionalidades de negocio muy concretas y sus mecanismos de interacción. Sirviendo como base de entendimiento para el cumplimiento de los objetivos.
- Este trabajo aborda de forma más puntual los distintos niveles de capacidad, y el tamaño de la funcionalidad en que los microservicios pueden declararse, lo cual se suele denominar granularidad [14].

1.5 Limitaciones

- Se utiliza código sintético y código real para probar la metodología.
- Se utilizan herramientas de software libres para el desarrollo y despliegue de microservicios.

1.6 Trabajos relacionados

Para construir el estado del arte con relación al presente tema de tesis se han consultado artículos en diversas fuentes documentales. A continuación, se describen los trabajos relacionados:

Microservices: granularity vs. Performance, presentado por Shadija, Dharmendra, Rezai, Mo And Hill y Richard en 2017, este articulo explora un punto en concreto de la arquitectura de microservicios (MSA, por sus siglas en ingles), específicamente lo relacionado con la granularidad del servicio, en el cual se hace mención de su potencial para influir en la latencia de las aplicaciones. Los microservicios se declaran con distintos niveles de capacidad de operación, al tamaño de esta funcionalidad se le denomina granularidad, también entendida como, la complejidad funcional codificada en un servicio o el número de casos de uso implementados por microservicio [14]. En este mismo artículo se dice que lograr un nivel óptimo de granularidad es de sumo interés para los desarrolladores que adoptan la MSA, los factores clave considerados que contribuyen a ello son:

- **La necesidad o capacidad de la empresa:** Es típico que los desarrolladores de aplicaciones utilicen la propia funcionalidad para establecer el alcance que determina el tamaño de un microservicio.
- **Tamaño de la aplicación:** Para las aplicaciones más pequeñas, el nivel de granularidad podría ser de grano fino. En el caso de las aplicaciones más grandes, es probable que la granularidad sea de un nivel más alto y que cada microservicio se construya a partir de otros más pequeños.
- **Tamaño del equipo de desarrollo:** Tanto en el contexto del MSA, como en el Diseño Basado en Dominios, el grado de éxito de la descomposición funcional, y su posterior implementación, depende de la estructura organizativa de los equipos de desarrollo. Por lo tanto, se debe tener en cuenta el número de desarrolladores en un equipo, así como sus habilidades.
- **Diseño de la base de datos:** El modelo de la base de datos puede influir en la granularidad. Cualquier relación en las bases de datos se implementará a nivel de código, dando lugar a microservicios de grano más grueso.
- **Reutilización.** Si los servicios son de grano fino, la reutilización es posible, pero existe la sobrecarga adicional de conectar los servicios entre sí. Si los servicios son demasiado gruesos, es difícil reutilizarlos. Un balance entre ambos aspectos es lo más acertado a contemplar.

Por otro lado, Cojocaru, Uta y Oprescu en su artículo Attributes Assessing the Quality of Microservices Automatically Decomposed from Monolithic Applications [17], definen que la

evaluación basada en métricas es obligatoria para valorar los atributos de calidad de los microservicios, sin embargo, dichas formas de evaluación son escasas. La redacción contempla a la granularidad como un atributo de calidad y que esta requiere un tipo de análisis especial. Esto se debe a que no existe una definición que haya sido aceptada para calcular el tamaño idóneo de un microservicio.

En cuanto a la validación de la granularidad, el escrito plasma que cuando se preguntó sobre las métricas utilizadas en las industrias para evaluar la granularidad de los microservicios, un experto reconoció que muchos profesionales de la industria todavía se basan en las Líneas de Código (LOC, por sus siglas en inglés) [18]. Otra métrica utilizada es la cantidad de puntos finales (dirección a la que se envían las peticiones) multiplicada por la cantidad de métodos para evaluar la granularidad de los microservicios. Aunque la relación de los tamaños y los componentes es un comienzo prometedor para evaluar la granularidad, todavía no se considera el valor más acertado debido a su escasa precisión en proyectos pequeños [17].

En un estudio de mapeo sistemático realizado en el 2019 por Sara Hassan, Rami Bahsoon y Rick Kazman sobre la Transición de los Microservicios y su granularidad se concluye que, actualmente en la práctica, la adopción de microservicios carece de métodos y técnicas apropiadas que justifiquen la forma de determinación adecuada de la granularidad, esto es porque la determinación de la granularidad implica la fusión o descomposición de microservicios, pasando así a un nivel de grano más fino o más grueso. [2]

En el mismo estudio se habla sobre los marcos de clasificación del Lenguaje de Definición de Arquitecturas (ADL) [19], indicando que es necesario modelar tanto los aspectos estructurales o topológicos, como los de comportamiento de una arquitectura. Durante la práctica del modelado de microservicios debe tomarse en cuenta que los enfoques de modelado estructural (113 identificados) son casi el doble de los enfoques de comportamiento (71 identificados). Mientras que los enfoques estructurales capturan la topología y/o las dependencias entre las unidades de construcción de la arquitectura de microservicios, los enfoques de comportamiento almacenan las acciones de estas unidades en varios escenarios de tiempo de ejecución en los que opera el microservicio modelado. Por lo tanto, se puede interpretar que un enfoque de modelado sistemático y orientado a la arquitectura de microservicios necesita contemplar los aspectos estructurales y el comportamiento de la arquitectura para facilitar el razonamiento sobre la granularidad.

La figura 1 [2] muestra una clasificación del número de publicaciones y los atributos de calidad que se deben optimizar al tratar la granularidad; respondiendo así a una de las preguntas más importantes en torno a esta investigación, ¿qué atributos de calidad se tienen que tomar en cuenta al determinar la granularidad de los microservicios? Según las literaturas examinadas por el mapeo sistemático la escalabilidad y el rendimiento son los aspectos más comúnmente considerados.

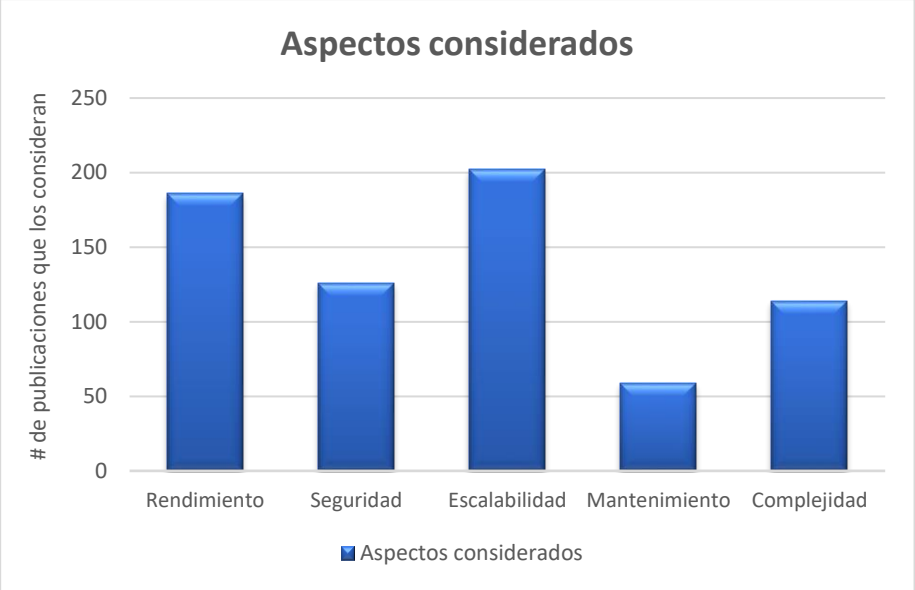

Figura 1 Adaptación de la gráfica presentada en el mapeo sistemático [2], sobre los aspectos considerados como atributos de calidad para determinar la granularidad.

La investigación titulada Microservice Ambients: An Architectural Meta-modelling Approach for Microservice Granularity [21], presentada por Sara Hassan, Nour Ali y Rami Bahsoon encontró que, con el aumento del interés de las industrias de software por los microservicios, aún hay una falta general de enfoques sistemáticos que modelen las decisiones de diseño de los microservicios, incluyendo la determinación de los límites óptimos de los microservicios entendiéndose como el nivel óptimo de granularidad. En este sentido, el tiempo de ejecución es un aspecto detectado de este problema, ya que gran parte de las incertidumbres relacionadas con la elección del nivel óptimo de granularidad y el comportamiento esperado del sistema no solo son problemas del tiempo de diseño.

Hassan, Ali y Bahsoon expresan que la heterogeneidad de las herramientas que soportan las arquitecturas de microservicios requiere flexibilidad a la hora de modelar el problema de la decisión de granularidad independiente de la tecnología, así mismo, en el reporte también se destaca que los entornos de microservicios soportan el análisis en tiempo de ejecución, la movilidad y el conocimiento de la ubicación; siendo que todo ello es importante para la adaptación de la granularidad de los microservicios con calidad. [21]

El artículo Microservices Backlog - A Model of Granularity Specification and Microservice Identification [11], desarrollado en agosto 2020, por Vera, Puerto, Astudillo y Gaona, secundados por algunas universidades de Colombia, describe *Microservice Backlog* (MB), una técnica de programación genética totalmente automática que utiliza las historias de usuario del producto para proponer un conjunto de microservicios con una granularidad óptima, permitiendo a los arquitectos de software visualizar sus métricas de diseño. Pero como punto más importante se define una nueva métrica de granularidad (GM) que combina las métricas existentes de métricas de acoplamiento, cohesión e historias de usuario asociadas.

Los autores sostienen que es difícil identificar las capacidades de negocio y los contextos delimitados que se pueden asignar a cada microservicio, en cuanto a las metodologías y técnicas éstas deben facilitar el dimensionamiento y el versionado de los microservicios, intentando un corte de servicio adecuado, es decir; se define cuán pequeño o fino deberá ser.

El trabajo de investigación analiza gráficamente la granularidad de los microservicios, comenzando por la contemplación de un conjunto de requisitos funcionales, expresados como historias de usuario dentro de una lista priorizada y caracterizada de funcionalidades que debe contener una aplicación. Se propone así, un modelo que ayuda a definir el tamaño y número de microservicios utilizando programación genética; la cual muestra los microservicios que van a formar parte de una aplicación haciendo énfasis en sus dependencias, funcionalidades y acoplamiento, cohesión y métricas de complejidad en tiempo de diseño [11].

Munezero Immaculee J., Tuheirwe-Mukasa, Benjamin Kanagwa y Joseph Balikuddembe en su investigación de 2018 titulada Partitioning Microservices: A Domain Engineering Approach [22], hablan sobre como la ingeniería de dominio identifica la información que necesita un microservicio, los servicios necesarios para su funcionalidad y de cómo esta ingeniería proporciona una descripción de los flujos de trabajo en el servicio, así pues, el Diseño Dirigido por Dominios (DDD) proporciona una serie de patrones útiles para tratar el tipo de complejidad que se encuentra en el diseño de sistemas distribuidos y con dominios grandes y complejos, dividiendo el dominio en una serie de contextos limitados. Tomando en cuenta estos hechos, la investigación sugiere una metodología detallada que utiliza patrones DDD para dividir los microservicios y así a través del DDD proporcionar una manera de establecer un buen diseño en términos de la determinación del límite apropiado.

La literatura Granularity Decision of Microservice Splitting in View of Maintainability and Its Innovation Effect in Government Data Sharing, escrita por Yan Li, Chun-Zi Wang, Ying-chao Li y Jia Su, valora cuatro aspectos a tener en cuenta durante la determinación de un microservicio. Se comienza hablando de la mantenibilidad la cual puede separarse en dos partes; una es la realización de la corrección y refinamiento del sistema cuando las demandas son estables; la otra es la expansión y la optimización cuando las demandas cambian, lo cual hace que un sistema sea mantenible. Otro aspecto es el tamaño de un sistema de microservicios, el cual es la suma del tamaño de todos los subservicios para esta literatura, este tamaño está representado por el valor ponderado de la cantidad de operaciones contenidas en la interfaz expuesta de un servicio. Se habla también del grado de acoplamiento, el cual refleja cuánto depende un servicio de otros. Si un servicio depende de otro servicio y viceversa, habrá interdependencia, que según el documento es lo que se tiene que evitar durante la práctica, aunque un menor grado de acoplamiento significa mayor mantenibilidad, en la transformación de los servicios del sistema, un grado de acoplamiento bajo constituirá un cuello de botella con las funciones a llamar de los diferentes servicios. Por último, se menciona que también se debe considerar a la cohesión, refiriéndose a la contribución de la operación del servicio a una determinada tarea, aunque resulta difícil de medirla automáticamente [23].

A continuación, se muestra la Tabla 1 donde se hace una comparación entre los aspectos encontrados en las literaturas revisadas; la perspectiva del arquitecto representa la experiencia de cada persona involucrada en el desarrollo de software, esto con el fin de que basados en su criterio se puedan tomar decisiones en cuanto al diseño del sistema. Los diagramas de representación de cada funcionalidad son entendidos como aquellos esquemas que permiten modelar una forma de operación de una función específica del sistema, como por ejemplo historias de usuario, diagramas de flujo, etc. estos servirán para conocer cada funcionalidad a implementar en el sistema, es importante que este punto se tenga muy claro para así llegar a la granularidad más adecuada. La idea de tomar en cuenta las diferentes métricas propuestas en las bibliografías consultadas, es tener en cuenta los aspectos más relevantes de ellas, y así proponer una que ayude a evaluar si se hizo una correcta división del sistema, que es uno de los objetivos de este trabajo.

Si bien el cuadro no hace mención de todas las características identificadas por estudios anteriores (las cuales fueron anteriormente mencionadas), estos cuatro puntos representan los aspectos sobre los cuales se pretende apoyar la metodología para determinar la granularidad de un microservicio.

Tyson R. Browning en el artículo Aplicación de la matriz de estructura de diseño a los problemas de descomposición e integración de sistemas: Una revisión y nuevas direcciones [27] habla sobre la matriz de estructura de diseño (MDS), la cual es una herramienta de representación y análisis para el modelado de sistemas, especialmente para fines de descomposición e integración. Una MDS muestra las relaciones entre los componentes de un sistema en un formato compacto, visual y analíticamente ventajoso.

Se pueden clasificar dos tipos de matrices, las MDS estáticas y temporales, las estáticas representan elementos del sistema que existen simultáneamente, como los componentes de la arquitectura de un producto o los grupos de una organización. En las MDS temporales, el orden de las filas y columnas indica un flujo en el tiempo: las actividades anteriores de un proceso preceden a las posteriores, y términos como "feedforward" y "feedback" cobran sentido al referirse a las interfaces. Las MDS basados en el tiempo suelen analizarse mediante algoritmos de secuenciación.

Por su parte Luciano Baresi, Martin Garriga y Alan de Renzis en su informe Identificación de microservicios mediante el análisis de interfaces [28], señala que una arquitectura de microservicios es un enfoque para desarrollar una aplicación única como un conjunto de pequeños servicios, cada uno de los cuales se ejecuta en su propio proceso y se comunica con mecanismos ligeros, a menudo una API RESTful.

1.7 Cuadro comparativo de los trabajos relacionados.

Tabla 1 Cuadro comparativo de los trabajos relacionados.

Nombre de la investigación	Referencia	Determinación de la granularidad a partir de:			
		La perspectiva del arquitecto de software	Diagramas de representación de cada funcionalidad	El tamaño de las funcionalidades	Métricas
Microservices: granularity vs. performance.	[14]	Si	No	No	No
Attributes Assessing the Quality of Microservices Automatically Decomposed from Monolithic Applications.	[17]	Si	No	No	Si
Microservice Transition and its Granularity Problem: A Systematic Mapping Study.	[2]	Si	No	No	No
Microservice Ambients: An Architectural Meta-modelling Approach for Microservice Granularity.	[21]	Si	No	No	No
Microservices Backlog - A Model of Granularity Specification and Microservice Identification.	[11]	No	Si	No	Si
Partitioning Microservices: A Domain Engineering Approach.	[22]	Si	No	Si	No
Granularity Decision of Microservice Splitting in View of Maintainability and Its Innovation Effect in Government Data Sharing.	[23]	Si	No	Si	No
Extracting SOA Candidate Software Services from an Organization's Object Oriented Models.	[10]	Si	Si	No	Si
The ENTICE Approach to Decompose Monolithic Services into Microservices.	[11]	Si	Si	No	No
Extracting Microservices from a Monolithic Application.	[12]	Si	Si	Si	No
Microservices Identification through Interface Analysis	[2]	Si	Si	Si	No

La literatura aborda el problema de la descomposición para identificar módulos, paquetes, componentes y servicios "tradicionales", principalmente por medio de técnicas de agrupación de artefactos de diseño o código fuente, basándose en estas experiencias para introducir un enfoque novedoso para razonar sobre los microservicios a partir de una especificación inicial de OpenAPI (una interfaz independiente del lenguaje y legible por la máquina para las API de REST) de las operaciones que debe ofrecer la aplicación.

La contribución de estos autores es un proceso automatizado para identificar microservicios candidatos mediante un análisis semántico ligero y agnóstico del dominio de los conceptos de la especificación de entrada con respecto a un vocabulario de referencia.

En el mapeo sistemático de la arquitectura de microservicios, realizado por Nuha Alshuqayran, Nour Ali y Roger Evans [29], se menciona a los microservicios como un estilo de arquitectura que pone énfasis en dividir el sistema en servicios pequeños y ligeros que se construyen a propósito para realizar una función de negocio muy cohesionada.

Se ha observado en la literatura que los atributos de calidad de la arquitectura de microservicios, como la modularidad, la escalabilidad, la independencia y la mantenibilidad se presentan en casi todos los artículos revisados por el mapeo.

La mayoría de los artículos del estudio se encuentran en "propuesta de solución" o de "validación de solución", con validaciones basadas únicamente en experimentos controlados en laboratorio.

Los casos de uso UML se utilizaron principalmente para para modelar la validación y las pruebas de los microservicios, mientras que los de secuencia UML se utilizaron para esbozar la comunicación entre microservicios.

Fredy Humberto Vera en su tesis Modelo Inteligente de Especificación de La Granularidad de Aplicaciones Basadas en Microservicios [30] ve como los microservicios siguen el principio de la responsabilidad simple que dice que "Se reúnan las cosas que cambian por la misma razón, y separe las cosas que cambian por diferentes razones".

Proponiendo un modelo inteligente que permite especificar la arquitectura de aplicaciones basadas en microservicios, de tal forma que a partir del modelo se pueda definir la granularidad o tamaño adecuado de cada microservicio teniendo en cuenta algunas características como la complejidad cognitiva, el tiempo de desarrollo, el acoplamiento, la cohesión, la comunicación y dependencias entre ellos, todo en tiempo de diseño.

En el documento Granularidad del servicio en los sistemas de control y automatización industrial escrito por Aydin Homay, Mário de Sousa, Alois Zoitl y Martin Wollschlaeger [31], se menciona que la granularidad del servicio puede clasificarse en dos categorías: la granularidad física, que se refiere al tamaño físico de un servicio, y la granularidad semántica, que se refiere al tamaño lógico del servicio.

Un servicio puede granularse en función de los procesos que serán manejados por el servicio. A este tipo de granularización se le llama la Granularidad de Proceso. Un servicio puede ser granulado en función de los sistemas back-end (máquinas) que cooperan con el servicio. A este tipo de granularidad se le nombra Granularidad de Máquina. Por último, también menciona que un servicio puede granularse en función del número de funcionalidades que serán expuestas al resto del sistema y a este tipo de granularización se le llama Granularidad de Funcionalidad.

Chris Richardson en su libro Microservicios Patterns [32], establece que los servicios se organizan en torno a preocupaciones de negocio más que a preocupaciones técnicas. En este libro se muestra cómo utilizar las ideas del diseño orientado al dominio (DDD) para eliminar las clases supremas, que son clases que se usan en toda la aplicación y que causan dependencias enredadas que impiden la descomposición.

También describe un proceso sencillo de tres pasos, que se muestra en la figura 2, para definir la arquitectura de una aplicación. Es importante recordar, que no es un proceso que se pueda seguir mecánicamente. Ya que es muy probable que sea iterativo e implique mucha creatividad en cuanto su forma de trabajarlo.

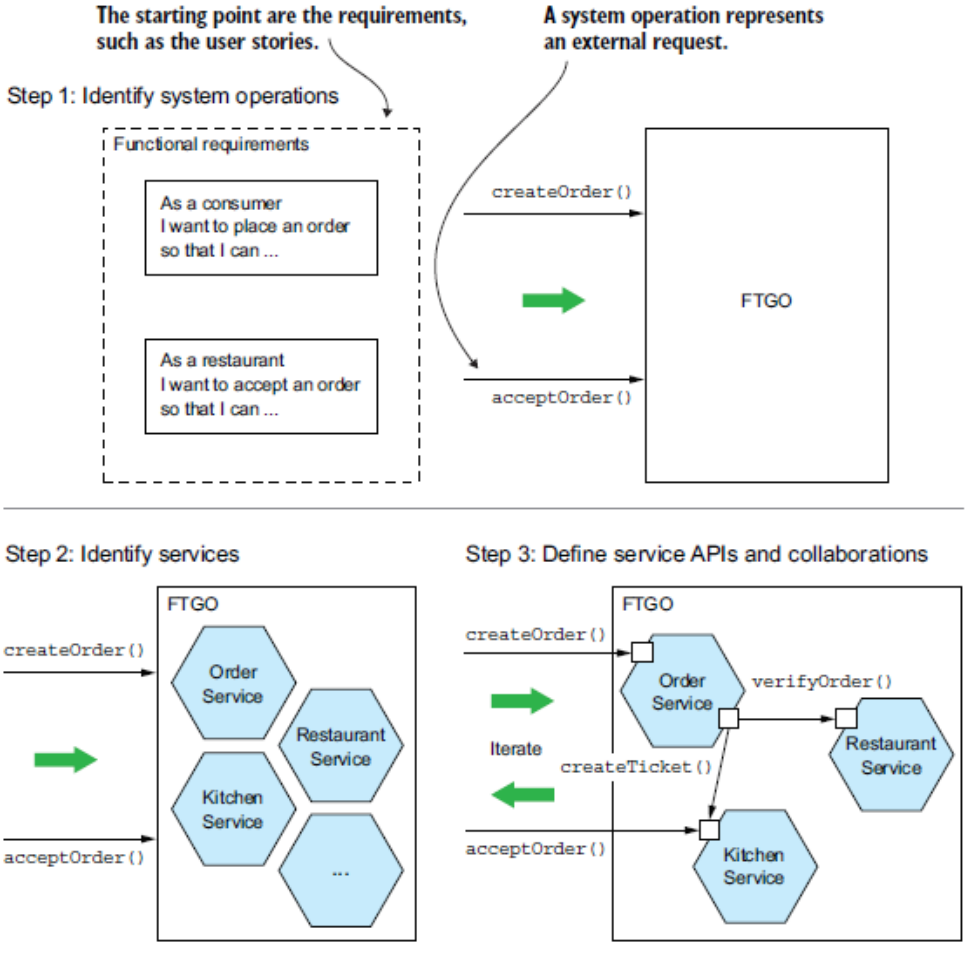

Figura 2 Proceso para definir la arquitectura de un proyecto, diagrama tomado del libro Microservicios Patterns [32].

A través de la identificación de las operaciones del sistema, la construcción de un modelo de dominio y definir las operaciones del sistema, se pueden identificar los microservicios de un sistema.

En Microservices Tenets: Agile Approach to Service Development and Deployment [33], su autor Olaf Zimmermann de la Universidad de Ciencias Aplicadas de Suiza Oriental, argumenta los despliegues de microservicios exitosos son posibles gracias a los paradigmas modernos de ingeniería de software y a los recientes avances en el desarrollo de aplicaciones web, por ejemplo, el diseño dirigido por el dominio y el desarrollo dirigido por las pruebas.

En el documento Semantic clustering: Identifying topics in source code escrito por Adrian Kuhn, Stephane Ducasse y Tudor Gırba [34], utilizan como enfoque la indexación semántica latente (LSI), una técnica de recuperación de información que localiza temas lingüísticos en un conjunto de documentos. Aplican la LSI para calcular la similitud lingüística entre los artefactos fuente (por ejemplo, paquetes, clases o métodos) y los agrupa según su similitud. Esta agrupación divide el sistema en temas lingüísticos que representan grupos de documentos con un vocabulario similar. Para identificar cómo se relacionan los clústeres entre sí, utilizan una matriz de correlación. Luego vuelven a emplear LSI para etiquetar automáticamente los clústeres con sus términos más relevantes. Y finalmente, para completar el cuadro, utilizan un mapa de visualización para analizar la distribución de los conceptos sobre la estructura del sistema.

Los autores Aydin Homay, Alois Zoitl, y Mário de Sousa, en su artículo Granularity Cost Analysis for Function Block as a Service [35] mencionan que entender si dividir un servicio en servicios más pequeños (descomposición) o combinar varios servicios en un solo servicio (composición) que añade algún valor podría ayudar a optimizar el rendimiento del servicio. Este artículo hace un intento de presentar un método basado en el análisis de la complejidad para este problema calculando el costo de la descomposición del servicio. Los autores introducen una metodología para descomponer una aplicación monolítica en microservicios candidatos basándose en técnicas de composición y descomposición de servicios. Por ejemplo, la aplicación del enfoque de contexto limitado del Diseño Dirigido por Dominios (DDD) mapeando los servicios a los contextos de negocio.

En el artículo estudio experimental para determinar la influencia de la estructura interna de servicios web en su tiempo de respuesta [52], se hace un estudio para determinar la influencia que tiene el nivel de granulación de servicios Web sobre su tiempo de respuesta como factor que afecta la eficiencia. A través de cinco casos de prueba, cuya estructura interna fue diseñada con cuatro niveles de granulación diferentes donde cada uno fue fuertemente estresado con un gran número de peticiones simultáneas y observándose la variación del tiempo de respuesta, los autores concluyen que el nivel de granulación de la estructura interna de entidades de software influye en el tiempo de respuesta, considerándose así que la mejor decisión es utilizar un grado de granulación intermedio para alcanzar un balance entre atributos de calidad de software y su tiempo de respuesta.

Capítulo 2

2. Marco teórico

2.1 Definiciones

2.1.1 Arquitecturas monolíticas

Son arquitecturas que se dividen en capas, y poseen un único ejecutable lógico, es decir que toda la aplicación se ejecuta dentro de un mismo contexto. Sus ventajas están dadas por facilidad de desarrollo, simplicidad de pruebas, sencillez de despliegue, viabilidad del escalado.

Generalmente son aplicaciones hechas a la medida por lo cual son eficientes y rápidas, sin embargo, se tratan de estructuras rígidas que carecen de flexibilidad, pues a medida que crecen comienzan a ponerse de manifestó algunos problemas [1].

2.1.2 Arquitecturas Orientadas a Servicios (SOA)

Es una arquitectura de software en la cual los distintos componentes de la aplicación se relacionan entre sí, y proporcionan servicios al resto de componentes mediante un protocolo de comunicaciones en red. SOA puede coordinar o poner en comunicación dos o más servicios [15].

2.1.3 Microservicio

Es un componente granular desacoplado, dentro de una aplicación más amplia. Constituye un enfoque de la arquitectura de software y sistemas que se basa en el concepto bien establecido de modularización, cada módulo (microservicio) es una unidad independiente de desarrollo, despliegue, operaciones, versionado y escalado; se implementa y opera con un sistema pequeño que ofrece acceso a su lógica y datos internos a través de una interfaz de red bien definida [6].

2.1.4 Arquitectura basada en microservicios

Enfoque de desarrollo de software basado en microservicios, dirigido a soluciones tecnologías que buscan soportar numerosos procesos de negocio, teniendo una dependencia relativa entre ellos. Este enfoque tiene como objetivo adaptar el software a situaciones de escalabilidad, además de proporcionar la flexibilidad para desarrollar, probar, mantener y desplegar servicios de forma independiente [24].

2.1.5 Microservitización

Concepto utilizado para hacer referencia a un desarrollo de software basado en una arquitectura de microservicios. Esta arquitectura se basa en que cada componente sea lo más independiente posible del resto. Los microservicios son debidamente acoplados y su modelo de datos independiente para cada uno, proporcionando una funcionalidad completa y de autogestión.

2.1.6 Granularidad de un microservicio

La granularidad de un servicio parte de dos conceptos importantes, la división de las funcionalidades a desarrollar en un sistema y el tamaño adecuado para cada una de estas funciones, ya que deben ser del tamaño justo para no ser repetitivas [12].

2.1.7 Interfaz de Programación de Aplicaciones (API)

Una API es un conjunto de definiciones y protocolos que se utiliza para desarrollar e integrar el software de las aplicaciones. Las API permiten que sus productos y servicios se comuniquen con otros, sin necesidad de saber cómo están implementados. Las API otorgan flexibilidad; simplifican el diseño, la administración y el uso de las aplicaciones, y proporcionan oportunidades de innovación, lo cual es ideal al momento de diseñar herramientas y productos nuevos [16].

2.1.8 Análisis estático

Es una técnica para el análisis automatizado de código fuente, que no requiere la ejecución del software analizado. El análisis del código de un microservicio antes de ser enlazado con otros microservicios puede ayudar a la identificación y corrección de vulnerabilidades sin incurrir en los costes de ejecución del código.

2.1.9 Análisis dinámico

Es una técnica para el análisis de código que requiere la ejecución del software, generalmente empleada cuando el código fuente de la aplicación no está disponible. El tipo más común de análisis dinámico consiste en Pruebas de Unidad y tiene el beneficio de validar los hallazgos del análisis estático o identificar nuevas fallas.

2.1.10 LOC

Medida definida por el número de líneas de código escritas para la entidad de software. Aunque la mayoría de las veces la granularidad se asocia al número de "Líneas de Código", no es una métrica muy descriptiva en sí misma.

2.1.11 Interfaces abiertas

Una métrica tomada de SOA que describe la granularidad de los microservicios a través del número de interfaces abiertas/expuestas. Desde una perspectiva de descripción de la carga de trabajo, el tamaño de un microservicio es cuantificable investigando el número de operaciones disponibles a través de las interfaces expuestas. Las operaciones pueden ponderarse en función de su granularidad o número de parámetros.

2.1.12 Cohesión

Representa el grado en que las operaciones proporcionadas por un modulo atienden a una sola funcionalidad [53]. Desde la perspectiva de la descripción de la carga de trabajo, la cohesión puede medirse a través de las interfaces expuestas afirmando la similitud entre los tipos de datos de los parámetros.

2.1.13 Latencia

En términos informáticos este concepto se define como, el tiempo que transcurre entre una orden y la respuesta que se produce a esa misma orden. Cada elemento informático funciona a través de estímulos eléctricos, entonces se entiende como el tiempo que tardan en realizarse todas las conmutaciones eléctricas y lógicas necesarias desde el comienzo de la acción, hasta que el ordenador muestra los resultados. Generalmente la latencia se mide en unidad de tiempo, concretamente en milisegundos o microsegundos [20].

2.1.14 Heterogeneidad

Cuando los componentes de un conjunto son distintos, pero éstos pueden comunicarse entre sí por medios comunes. Termino también relacionado con el desarrollo de sistemas flexibles, multiplataforma, dado que lo más probable es que se conecten diferentes tipos de plataformas de cómputo, las cuales deben tener resueltas las incompatibilidades que presenten entre sí.

2.1.15 Metodología

Una metodología es un conjunto integrado de técnicas y métodos que permite abordar de forma homogénea y abierta cada una de las actividades del ciclo de vida de un proyecto de desarrollo. También puede ser entendido como un proceso de desarrollo detallado y completo [25].

Las metodologías definen artefactos, roles y actividades, junto con prácticas y técnicas recomendadas. La metodología para el desarrollo de software es un modo sistemático de realizar, gestionar y administrar un proyecto para llevarlo a cabo con altas posibilidades de éxito.

2.1.16 Pasos para una propuesta metodológica

Una metodología comprende los procesos a seguir sistemáticamente para idear, implementar y mantener un producto. Entendiendo en qué consiste el método científico, en contraste con una metodología de desarrollo de software. A continuación, se establece una secuencia de actividades que siempre se verán involucradas en cualquier proyecto de desarrollo, abordadas desde un ámbito de software para fines del presente tema de tesis [25, 26]:

- Descripción de la situación o problema que se quiere tratar.
- Comprender el contenido, establecer los objetivos, alcances y tiempos del proyecto.
- Identificar el proceso de desarrollo (métodos, herramientas, lenguajes, documentación, ayudas).
- Perspectiva teórica, haciendo la documentación y análisis de toda la base de información existente sobre el tema a tratar.
- Programar el proceso, programar qué actividades deben realizarse y en qué tiempo.
- Identificación de los riegos y búsqueda preventiva de las soluciones.
- Realización del proyecto, se realizan las actividades de acuerdo al proceso establecido para alcanzar los objetivos.
- Verificación o análisis del producto obtenido.
- Se establecen conclusiones sobre todo el trabajo desarrollado.

Capítulo 3

3. Desarrollo de la solución

3.1 Selección de enfoques para determinar la granularidad.

De acuerdo con toda la literatura revisada, donde se abordan los diferentes enfoques tomados en cuenta para determinar la granularidad, aplicados en diferentes proyectos, y en función del grado de popularidad de uso de cada uno de éstos, se ha optado por la elección de cuatro enfoques. Hasta ahora, los enfoques seleccionados han reflejado ser buenos como herramientas para identificar el tamaño más adecuado de un microservicio, permitiendo una correcta modularización de un sistema, cumpliendo con los objetivos en el mejoramiento de la mantenibilidad, escalabilidad, independencia, entre otros aspectos.

Aunque los autores consultados, en muchas ocasiones manejan términos diferentes para referirse a su enfoque para determinar la granularidad, algunos de sus conceptos utilizados tienen una misma definición en términos informáticos, por lo cual estos términos fueron agrupados en una sola categoría, como es el caso de la determinación de la granularidad por modelo de dominio, que en otros registros puede encontrarse como granularidad de máquina.

Por esta razón, en la tabla 2, por cada categoría se pueden encontrar diferentes nombres, pero debe entenderse que se trata de un mismo enfoque que puede ser referenciado de diferente forma.

Tabla 2 Lista de enfoques seleccionados.

#	Enfoques	Literaturas que lo mencionan
1	Por Modelo de Dominio / Maquina	[29] [31] [32] [33]
2	Por Capacidades de negocio / Base de Datos	[27] [29] [30] [39]
Ok3	Por Declaración de Objetivos / Design Thinking	[28] [40]
4	Por Procesos / Modelado UML	[31] [29] [36]

3.2 Metodología en la determinación de la granularidad de un microservicio.

En esta sección se describe de forma detalla la propuesta de la “metodología en la determinación de la granularidad de un microservicio”. Esta metodología reúne las que se consideran las principales (mejores) características (procesos) de acuerdo a los estudios revisados.

Para la elaboración de la propuesta presentada se realizó un análisis inicial basándose en múltiples artículos que abordan la granularidad desde diferentes enfoques, por ejemplo, utilizando un modelo de dominio, por reglas o capacidades de negocio, por procesos, etc. En base a toda la información examinada, la metodología se delimitó a cuatro formas (enfoques) diferentes en que se aconseja descomponer un sistema en microservicios. Es importante

mencionar que los pasos marcados en color lila, son las actividades puramente obtenidas de los trabajos relacionados y colocadas en la metodología, mientras que los pasos marcados en color rosa, son tareas adicionales que fueron adaptadas a la metodología en las áreas de oportunidad que se encontraron en los enfoques revisados, por lo cual, esta metodología supone una mejora al aprovechar los vacíos identificados en los trabajos ya existentes y proponer formas de atacarlos.

Como se observa en la figura 3 la metodología se compone de tres partes esenciales, 1) la etapa de análisis, 2) la de descomposición por enfoque y 3) la de cierre o implementación.

- **La etapa de análisis.** Esta primera etapa se encarga de reunir toda la información posible sobre el proyecto a realizar, y se encuentran involucrados tanto los arquitectos y desarrolladores de software, como los usuarios directos e indirectos del sistema. Su principal objetivo es generar uno de los cuatro esquemas de salida abordados en la tabla 3, de la forma más detallada posible. El esquema seleccionado repercutirá directamente en el siguiente paso.
- **La etapa de descomposición por enfoque.** La segunda parte de la metodología consiste en la identificación de los candidatos a microservicios. En esta etapa se puede observar que existen cuatro diferentes caminos a seguir, cada uno de ellos representa a un enfoque seleccionado. En función del esquema construido en la etapa anterior, será la dirección que tomará para la determinación de la granularidad de los microservicios de su proyecto, es decir, que solo seguirá una de las cuatro rutas de actividades a realizar. Se sugiere la lectura completa de cada proceso antes de la selección y construcción del esquema de salida, esto con el fin de conocer más a fondo lo que se espera en cada caso y tener un acercamiento sobre las actividades que involucra la elección de cada esquema.
- **La etapa de implementación.** Todos los procesos de la etapa anterior concluyen en el paso de la implementación, como el nombre lo indica la intención de esta última etapa es llevar a cabo el desarrollo de cada microservicio identificado. Al ser una actividad que depende de la elección de plataformas de trabajo que mejor convenga a cada desarrollador, no se ahondará más sobre el tema de la implementación.

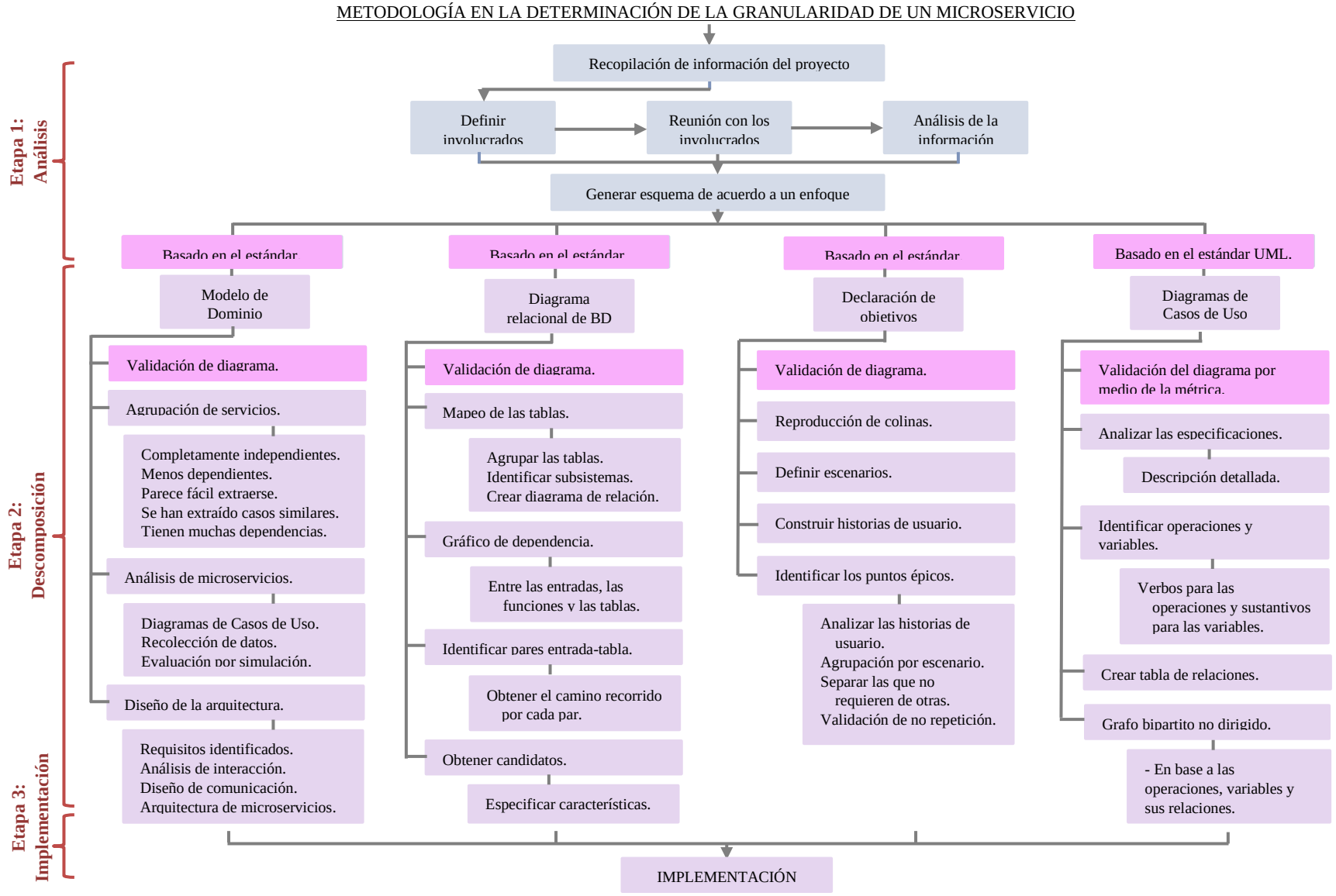

Figura 3 Metodología para determinar la granularidad de un Microservicio, elaboración propia.

3.2.1 Etapa de análisis

A continuación, se describe la primera etapa de la metodología propuesta para determinar la granularidad de un microservicio. Como se muestra en la tabla 3, en esta etapa de análisis, por parte del equipo de desarrollo se hace una comprensión total del proyecto, construyendo alguno de los esquemas de salida. El diseño del esquema es de suma importancia ya que será el punto de partida de las actividades siguientes.

Cabe mencionar que esta parte fue construida a partir de las diferentes actividades encontradas en las diversas fuentes consultadas, ajustándolas en un solo proceso, que dio como resultado esta etapa de análisis.

Tabla 3 Paso 1: Cuadro descriptivo de la etapa de análisis.

ÉTAPA	ACTIVIDAD	TAREAS	DESCRIPCIÓN
ANALISIS	Definición de involucrados	• Arquitecto / Desarrollador. • Usuarios directos. • Usuarios indirectos.	Se definen todos los involucrados en el proyecto, los usuarios directos son aquellos que realizan el proceso directamente y utilizarán el sistema. Los indirectos son aquellos que se ven involucrados en algún punto del proceso y por lo tanto tendrán alguna interacción (mínima) con el sistema, estos también involucran al personal de alto rango que estará a cargo y aprobará el proyecto.
	Recabar y analizar la información	• ¿Qué? • ¿Para qué? • ¿Quién? • ¿Cómo?	Se lleva a cabo una reunión entre el arquitecto o desarrollador y los usuarios (directos e indirectos) con el fin de obtener una alta comprensión de lo que se va a desarrollar. Se responde a las preguntas clave qué, para qué, quién y cómo.
	Planteamiento de acuerdos	• Establecer acuerdos entre los involucrados.	Se presenta lo entendido o aceptado por todos los involucrados, si es necesario se reúne más información (se regresa al paso de “Recabar y analizar información”). Se establecen acuerdos claros de lo que se va hacer y como se va hacer.
	Construcción de diagrama	• Seleccionar enfoque. • Seguir estándar.	De acuerdo a la experiencia del arquitecto de software y en general del equipo de desarrollo, se elige un enfoque a seguir y se realiza el diseño de: **modelo de dominios, diagrama relacional de una base de datos, declaración de objetivos o diagrama de casos de uso,** siguiendo el estándar establecido. ➢ Para casos de uso – Estándar UML

3.2.2 Etapa de descomposición por enfoque

Al finalizar la etapa anterior se debe contar con alguno de los esquemas de salida (modelo de dominios, diagrama relacional de una base de datos, declaración de objetivos o diagrama de casos de uso), ya que a partir de este punto se tomará un camino diferente dependiendo del esquema realizado. Para efectos de esta investigación se han delimitado cuatro enfoques principales, por lo cual si se llegara a generar algún producto (esquema) de salida que no está contemplado en la tabla 3, no podrá continuar con el proceso, ya que para cada enfoque se han definido una serie de actividades que encajan exactamente con las características que presenta cada esquema.

El título de cada figura de los enfoques presentados a continuación, indica el número de paso y a qué proceso corresponde cada diagrama. Así mismo, cada figura cuenta con números consecutivos en cada apartado, los cuales representan el orden a seguir en cada una de las actividades. Se debe tener cuidado durante el proceso de determinación de granularidad, siguiendo los pasos que corresponden a cada modelo.

El orden de aparición de cada proceso se encuentra de la siguiente manera:

1. Por Modelo de Dominios.
2. Por Diseño de la base de datos (Capacidades de Negocio).
3. Por Declaración de objetivos (Escenarios).
4. Por Diagrama de Casos de Uso (Procesos).

Nota: Recordar que todos los diagramas comienzan a partir del paso dos, ya que el paso uno fue el de la etapa de análisis, por esta razón notará que en cada primer diagrama (por enfoque), se tiene el número dos ubicado en la esquina superior izquierda.

3.2.2.1 Granularidad por Modelo de Dominio

El diseño basado en dominios (DDD) es uno de los métodos propuestos para extraer microservicios de un dominio, este es un modelo evolutivo que coloca el conocimiento del dominio en el centro del diseño. El DDD sugiere que cada contexto delimitado represente un microservicio, entendiendo que un contexto delimitado en un sentido lógico describe una parte del software donde términos, definiciones y reglas particulares se aplican de forma coherente. A continuación, se explican de forma precisa, los pasos a seguir en la determinación de la granularidad por modelo de dominio.

Paso 1:

La primera actividad para este enfoque es la validación del diagrama construido, en esta tarea se aplica un método de validación que permite verificar si se ha hecho un correcto diseño del modelo de dominios y así dar paso a la siguiente actividad de la metodología, o de lo contrario si al aplicar la validación el modelo resulta no valido, hacer los cambios necesarios hasta tener un correcto diseño de éste.

Paso 2:

Teniendo correctamente construido el modelo de dominio, el siguiente paso será la agrupación de servicios, en esta fase sólo están involucrados el arquitecto y los desarrolladores de software, quienes bajo su criterio y siguiendo el orden de prioridad de la pirámide mostrada en la figura 4, procederán a decidir que partes van juntas y cuales separadas.

En primer lugar, se identifican las partes completamente independientes del sistema, éstas son aquellas que por sí solas tienen un trabajo independiente dentro del sistema, por lo cual su nivel de relación en modelo de dominio es el más mínimo existente. Siguiendo la misma lógica se procederá a sacar a las partes que presentan muy poca dependencia de otras, por lo cual su relación se encuentra muy aproximada al nivel mínimo. Se espera que, para estos dos casos, se obtengan los mejores módulos candidatos a microservicios. En el nivel 3 y 4 estarán aquellos dominios que su extracción parece fácil y no complicaría a la arquitectura, así mismo en base a la experiencia de los desarrolladores y arquitecto, identificarán aquellos servicios que anteriormente ya han sido trabajados como módulos independientes y su despliegue ha funcionado correctamente.

Para casos más complejos se tiene como candidatos a microservicios ~~a~~ aquellos servicios que se sabe que dentro de poco tiempo pueden cambiar, en esta misma categoría están las partes que muestran demasiada dependencia de otras, vista desde el nivel de relación que presentan en el modelo de dominio.

Hasta este punto solo se ha obtenido un primer borrador con los candidatos a microservicios, por lo cual se debe esperar a la siguiente etapa de valoración de datos por cada microservicio identificado.

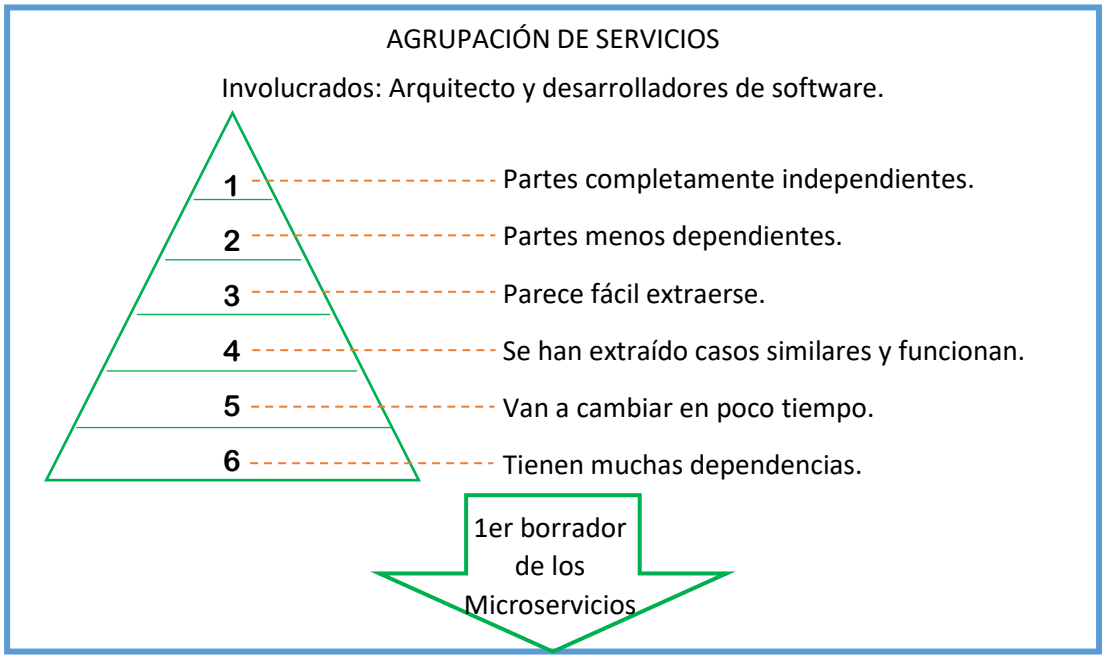

Figura 4 Paso 2 Granularidad por Modelo de Dominio.

Paso 3:

Por cada servicio identificado anteriormente se obtendrán los requisitos funcionales de cada uno de ellos. Para esto se realizarán casos de uso por cada candidato, y así obtener los datos que se necesitan en cada servicio. Se recomienda simular ejemplos utilizando la forma "Given – When - Then" (dado, cuando, entonces). Y así explicar cada requisito por microservicio en tarjetas o algún medio de presentación, se tiene que ser detallado en esta actividad e incluir todas las tareas y subtareas involucradas (figura 5).

Paso 4:

En la figura 6 se explican los pasos a seguir para el diseño de la arquitectura por microservicios, el objetivo es aclarar cómo el microservicio interactúa y se comunica con el resto del sistema desde una perspectiva técnica.

Los requisitos creados en el paso anterior se utilizarán para comprender cuando el microservicio necesita interactuar con otras partes del sistema. A partir de esto se debe analizar si la comunicación debe realizarse mediante una comunicación asíncrona o mediante comunicación síncrona.

En esta fase también se seleccionan las herramientas que se van a utilizar en la implementación. Durante la implementación se tiene que hacer un esfuerzo para que el nuevo microservicio sea más comprensible y estructurado que el código de un monolito. Y también esforzarse por establecer una infraestructura que pudiera ser reutilizada cuando se desarrollen otros microservicios.

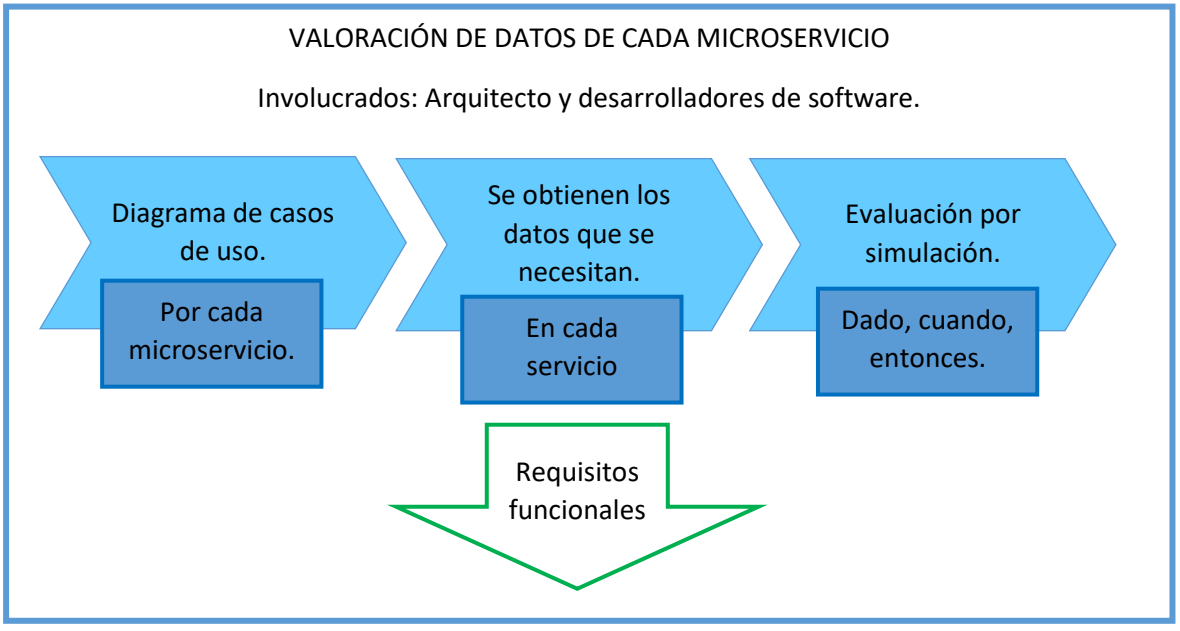

Figura 5 Paso 3 Granularidad por Modelo de Dominio.

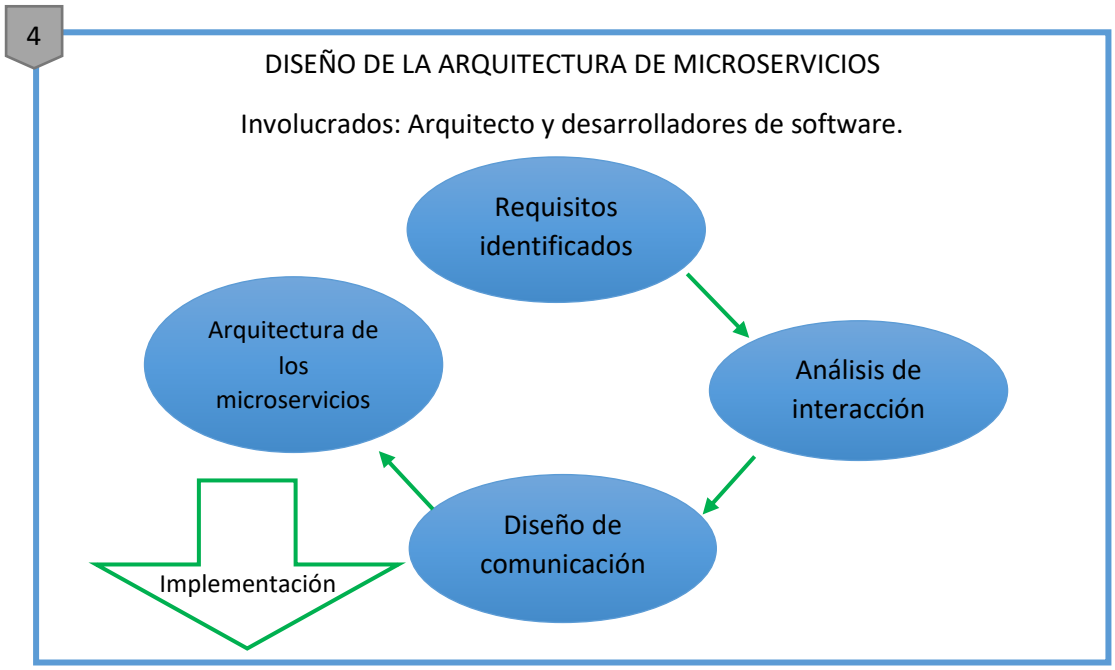

Figura 6 Paso 4 Granularidad por Modelo de Dominio.

3.2.2.2 Granularidad por Capacidades de Negocio (Base de Datos).

Si se pretende determinar la granularidad por capacidades de negocio el proceso a seguir se encuentra en las figuras 7, 8 y 9, cabe recalcar que, para este enfoque como producto de salida de la etapa de análisis, se debe obtener el diseño relacional de una base de datos.

Paso 1:

Como primera actividad se tiene la validación del diagrama relacional, se validará si se ha realizado un correcto diseño del diagrama relacional de una base datos, dando pie a la siguiente actividad en la metodología, o si es necesario, hacer los ajustes correspondientes hasta tener un diagrama relacional valido (figura 7).

Paso 2:

El siguiente paso de esta segunda fase, es mapear las tablas de la base de datos (BD) en subsistemas. Cada subsistema representa a un área de negocio. Básicamente se analizan las relaciones que existen entre las tablas y los subsistemas para conocer como dependen entre ellas. Para esto se construirá un diagrama de relación como apoyo (figura 8).

Paso 3:

En este punto se crea un gráfico de dependencia, donde el primer nivel representa a las entradas del sistema, los siguientes niveles serán las funciones de negocio (son los métodos que se esperan del sistema) y en el último nivel se tendrá a las tablas de la BD.

Paso 4:

En la figura 9 se observa la continuación de este proceso, la actividad siguiente será la identificación de pares entre entradas y tablas, y los caminos existentes entre ellos, tomando en cuenta por cuales funciones se pasa, construyendo así un conjunto de pares de relaciones.

Paso 5:

En cuanto a la identificación de candidatos a microservicios se debe analizar lo siguiente, para cada par distinto del paso anterior (entrada-tabla), se inspeccionarán las funciones que existen (en la ruta de cada par), el objetivo de este paso es identificar qué reglas de negocio dependen realmente de la tabla de la BD y esas operaciones deben describirse en forma textual como reglas. Por lo tanto, para cada par (entrada-tabla) define a un candidato a microservicio.

En este punto sólo restaría la implementación de cada microservicio identificado, no olvidándose que la selección de protocolos de comunicación y forma de almacenamiento de datos son parte importante previo a la implementación.

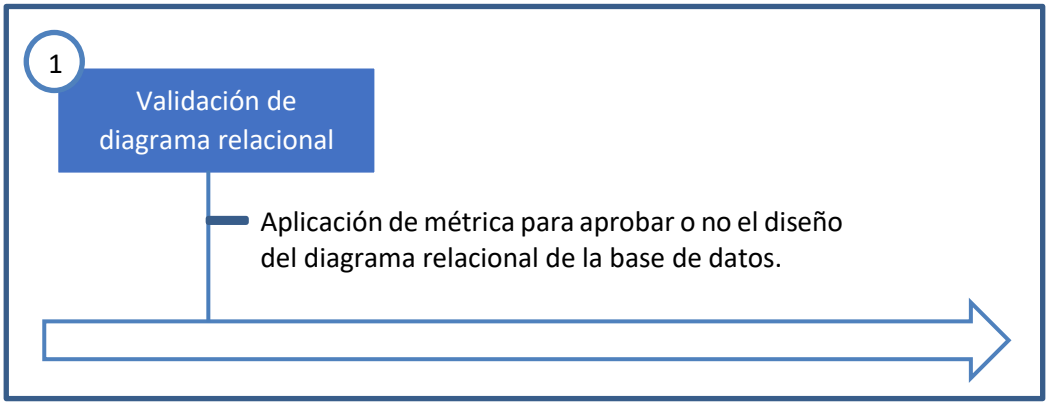

Figura 7 Paso 1 Granularidad por diagrama relacional de base de datos.

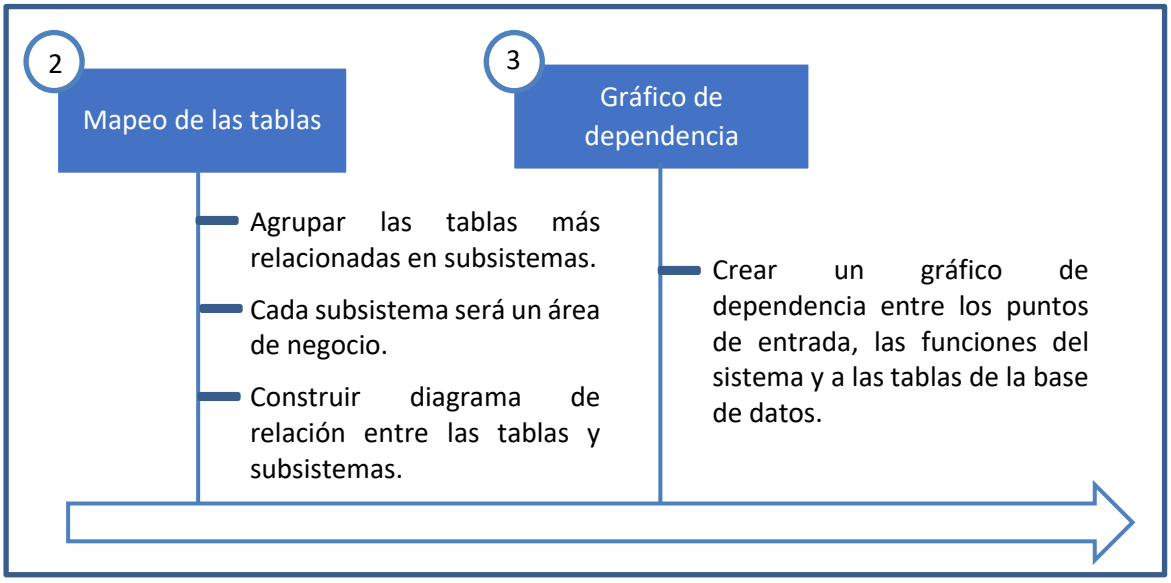

Figura 8 Paso 2 y 3 Granularidad por Base de Datos (Capacidades de Negocio).

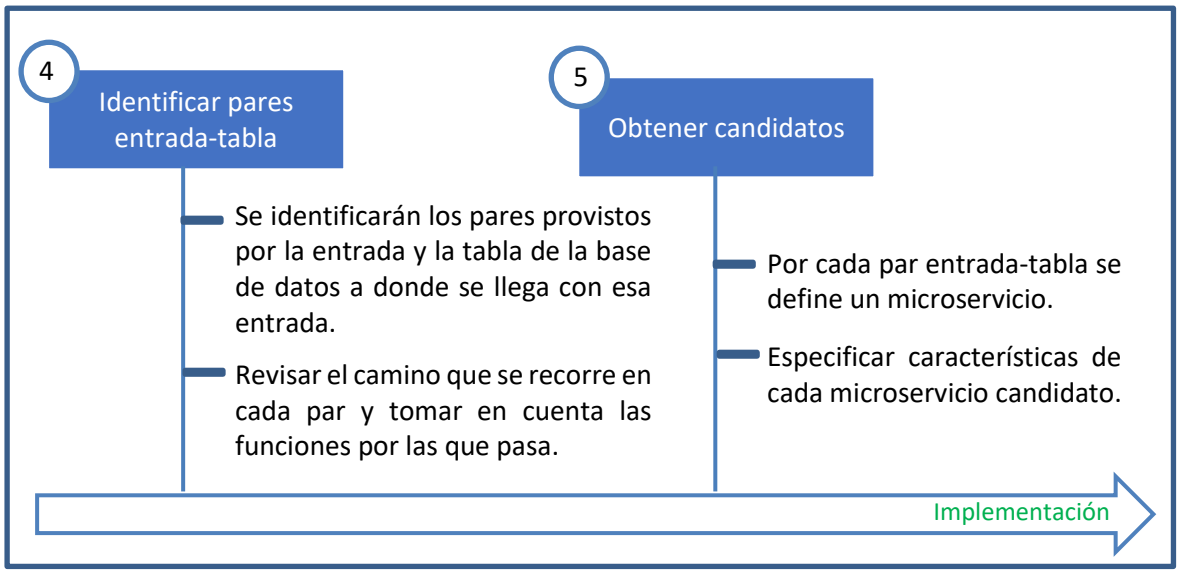

Figura 9 Paso 4 y 5 Granularidad por Base de Datos (Capacidades de Negocio).

3.2.2.3 Granularidad por declaración de objetivos (Design Thinking).

El pensamiento de diseño es un proceso para visualizar la experiencia del usuario en su totalidad. Es importante señalar que, en lugar de centrarse en una característica, se centra en la experiencia del usuario. El pensamiento de diseño es útil para definir el alcance del trabajo en unidades funcionales utilizables y liberables. Los diseños implementados facilitan la descomposición funcional y la identificación de microservicios.

Para seguir esta vertiente (figura 10), se deben tener muy claros los objetivos del proyecto, o del sistema en este enfoque los objetivos definen qué se va hacer, quién lo va hacer y cómo se va hacer. Cada objetivo representa una "Colina", estas colinas informan sobre la intención del líder y permite que los equipos de trabajo usen su propia experiencia sobre cómo interpretar e implementar una solución.

Paso 1:

Este enfoque comienza con la validación de los objetivos identificados, por medio de un método de validación se evaluará que se haya hecho una correcta declaración de objetivos, continuando con la aplicación de la metodología, o en su defecto, hacer los cambios necesarios hasta tener los objetivos correctamente declarados.

Paso 2:

Este paso se centra en la reproducción de colinas. La reproducción de las colinas proporciona un resumen de lo que el equipo va a realizar. Estas reproducciones establecen el alcance (lanzamiento de un producto) que se tendrá en cierto periodo de tiempo. La reproducción 0 es cuando el equipo ha completado el dimensionamiento, y se compromete con los patrocinadores de la empresa en cuanto a los resultados que quiere conseguir. Se realizan semanalmente con la participación de los equipos de desarrollo y los usuarios patrocinadores.

Paso 3:

Se pasa entonces a la definición de escenarios. Un escenario es un flujo de trabajo único a través de una experiencia de usuario, y da pie a las historias de usuario y el contexto utilizado de las mismas. Los escenarios capturan el escenario "tal cual" y se convierte en un escenario "a mejorar".

Paso 4:

A continuación, se sigue con la creación de las historias de usuario, las cuales representan un requisito autónomo y codificable que puede desarrollarse en poco tiempo (en relación a la dimensión del proyecto), recordando siempre ser expresadas en términos de experiencia de usuario.

Paso 5:

Los puntos épicos buscan delimitar la división de los servicios del sistema, y se comienza con el análisis de las historias de usuario, donde se construyen descripciones detalladas del modo de operación de cada historia, con esta información se podrá hacer una agrupación de aquellas que pueden reutilizarse varias veces en un escenario, se obtendrán así agrupaciones independientes de historias que funcionen adecuadamente, por último, se tiene que validar que no existan historias repetidas en las agrupaciones. Pasando así a la etapa de implementación.

Figura 10 Pasos Granularidad por Escenarios

3.2.2.4 Granularidad por diseño de Casos de Uso.

Las especificaciones de casos de uso están ampliamente aceptadas como una forma de describir los requisitos funcionales de un sistema. A continuación, se presenta la forma de abordar los microservicios cuando lo que se ha construido son diagrama de casos de uso como resultado de la primera etapa de esta metodología.

Un caso de uso describe cómo los usuarios (actores) emplean un sistema para alcanzar un objetivo concreto.

Paso 1:

Esta actividad no está contemplada en el proceso original propuesto por Tyszberowicz, Heinrich y Liu [48], sin embargo, ha sido incorporada como primera tarea, con el objetivo que antes de obtener los candidatos a microservicios el diagrama de casos de uso pueda ser validado a través de la aplicación de la métrica de complejidad de Sabharwal, Kaur y Sibal [42], y así ahorrar tiempos y recursos en caso de que sea necesario hacer ajustes, lo cual es una mejora al proceso original consultado. La explicación detalla sobre la aplicación de la métrica se encuentra en la sección 3.3 de este documento.

Paso 2:

Siguiendo la estructura de la figura 11, se comienza haciendo un análisis a los escenarios de los casos de uso, para eso se realizan sus descripciones de forma detalla en los cuadros de descripción que se usan en el estándar UML para casos de uso.

Paso 3:

Posteriormente se identifican las operaciones del sistema y las variables de estado, a partir del análisis de las descripciones de cada caso de uso, los verbos encontrados representan a las operaciones que tendrán lugar en las funciones del sistema, mientras que los sustantivos o frases sustantivas halladas son representan las variables primitivas, tanto de instancia como de clase y también representan las variables de referencia, también de instancia y de clase.

Paso 4:

Estas operaciones y variables son registradas, en una tabla de operaciones/relaciones, cada celda de la tabla indica si la operación escribe en la variable de estado, la lee o no la escribe ni la lee y las relaciones serán las que se dan entre cada operación del sistema y las variables de estado que la operación utiliza.

Paso 5:

Para la descomposición, se construye un grafo bipartito no dirigido (G) cuyos vértices representan las variables de estado y las operaciones del sistema. Una arista conecta una operación (op) con una variable de estado (v) si y sólo si op lee el valor de v o lo actualiza. Además, asignamos un peso a cada arista de G, dependiendo de la naturaleza de la conexión. Una conexión de lectura tiene un peso menor (1) y una conexión de escritura tiene un peso

mayor (2). Esta elección tiende a agrupar los datos con las operaciones que los modifican, prefiriendo las interfaces de lectura. Una interfaz de escritura involucra activamente a ambos subsistemas, por lo que tiene un acoplamiento fuerte, Como resultado se obtiene un gráfico que identifica clústeres claramente separados. Cada uno de estos clústeres representa ~~a~~ un candidato a microservicio.

Paso 6:

Como último paso debe definirse la tecnología de comunicación y almacenamiento de datos para posteriormente proceder a realizar la implementación.

Figura 11 Pasos Granularidad por Casos de Uso.

3.3 Métrica para la validación del enfoque de casos de uso.

Los modelos creados durante la fase de análisis de requisitos del desarrollo de software tienen una influencia significativa en la calidad general del producto de software. La evaluación temprana y objetiva de los atributos de calidad de estos modelos ayudan en el rediseño de la arquitectura, ahorrando costos y tiempo.

Es bien conocido que las métricas son un buen indicador de la calidad del software. Los diferentes atributos de calidad que definen los productos, procesos y proyectos de software se miden cuantitativamente a través de métricas. La complejidad estructural de varios modelos de análisis, como el diagrama ER, los diagramas de clase, el diagrama de casos de uso, creados durante las fases iniciales del desarrollo del software, se pueden utilizar como indicadores de la calidad de un sistema. La medición de la complejidad también ayuda a estimar el esfuerzo del software, el mantenimiento y la evaluación de los componentes del diseño [42].

El diagrama de casos de uso representa el comportamiento del sistema, las métricas derivadas del diagrama de casos de uso pueden actuar como un medio eficaz para evaluar la complejidad del sistema. A continuación, se presenta una métrica para medir la complejidad del software en función de las relaciones de inclusión y extensión en el diagrama de casos de uso propuesta por Sabharwal, Kaur y Sibal [42]. La complejidad del diagrama de casos de uso se define en términos de relaciones de dependencia entre diferentes casos de uso y la asociación entre el actor y los casos de uso en el diagrama de casos de uso.

Esta métrica se calcula utilizando el proceso que se indica a continuación:

1. Primero se necesita contar con uno o más diagramas de casos de uso del proyecto en cuestión.
2. Documentar los casos de uso usando la plantilla de caso de uso y los actores usando la plantilla de actor (Anexo A).
3. Generar una matriz de casos de uso de disparadores (M_{trig}).
   Es una matriz n por m donde n son casos de uso principales y actores y m son casos de uso principales y dependientes. Los casos de uso principales son aquellos que son llamados directamente por un actor, mientras que los dependientes son aquellos que se llaman a través de otro caso de uso. Una entrada a_{ij} en la matriz M_{trig} es el valor de dependencia que muestra que para el caso de uso/actor i, el caso de uso desencadenado es j. El valor de a_{ij} es la fuerza de la dependencia/asociación que existe entre i y j.
   Dependiendo de las relaciones entre el actor y los casos de uso principales o la dependencia entre los casos de uso, los valores de 1, 2 o 3 se asignan en estas matrices de la siguiente manera:
   - Cuando un caso de uso principal está conectado por asociación (uses o include) con un actor (lógico externo o humano) siempre será llamado, a la celda de la matriz se le asigna la fuerza más alta de 3.
   - Cuando un caso de uso principal está conectado (a un subcaso de uso o actor lógico interno) por incluir dependencia siempre será llamado, a la celda de la matriz se le asigna una fuerza de 2.
   - Cuando un caso de uso (principal) está conectado por extender la dependencia puede llamarse o no, a la celda de la matriz se le asigna la fuerza mínima de 1.
4. Generar matriz de casos de uso de iniciador (M_{init}).
   Es una matriz de *n* por *m* donde n son todos los casos de uso y m son los actores y los casos de uso principales. A través de esta matriz se puede identificar qué actores o

casos de uso son iniciadores de un caso de uso. Una entrada a_{ij} en la matriz M_{init} es el valor de dependencia que muestra que para i-ésimo caso de uso, el caso de uso/actor iniciador es j. El valor de a_{ij} es la fuerza de la dependencia/asociación que existe entre i y j.

5. Encontrar el efecto de iniciador combinado (CIE) de la matriz de casos de uso de iniciador (M_{init}) y el efecto de activación combinado (CTE) de la matriz de casos de uso de activación (M_{trig}).
    - IE = Es el efecto acumulativo de todas las dependencias de casos de uso en un i-ésimo caso de uso debido a sus casos de uso/actores iniciadores.

$$IE(i) = \sum_{j=1}^{m}[(celdas(i,j)\ de\ la\ fila\ (i)\ de\ M_{init}\ teniendo\ fuerza\ 3)\ x\ (número\ de\ escenarios) + \sum_{j=1}^{m}(\ celdas(i,j)\ de\ la\ fila\ (i)\ de\ M_{init}\ teniendo\ fuerza\ 2) + (número\ de\ celdas\ (i,j)\ de\ la\ fila\ (i)\ de\ M_{init}\ teniendo\ fuerza\ 1)^0] \quad (1)$$

    - CIE = Es la suma de todos los IE en la matriz Minit.
      El efecto de iniciador combinado (CIE) para el diagrama de casos de uso es:

$$CIE = \sum_{i=1}^{n} IE(i)$$
$$Donde\ i\ es\ el\ caso\ de\ uso/actor \quad (2)$$

    - TE = Es el efecto acumulativo de todas las dependencias de casos de uso en el caso de uso de itlt debido a los casos de uso desencadenados por él.

$$TE(i) = \sum_{j=1}^{m}[\big(celdas(i,j)\ de\ la\ fila\ (i)\ de\ M_{trig}\ teniendo\ fuerza\ 3\big)\ x\ (número\ de\ escenarios) + \sum_{j=1}^{m}(\ celdas(i,j)\ de\ la\ fila\ (i)\ de\ M_{trig}\ teniendo\ fuerza\ 2) + (número\ de\ celdas\ (i,j)\ de\ la\ fila\ (i)\ de\ M_{trig}\ teniendo\ fuerza\ 1)^0] \quad (3)$$

    - CTE = Es la suma de todos los TE en la matriz Mtrig.

$$CTE = \sum_{i=1}^{n} TE(i)$$
$$Donde\ i\ es\ el\ caso\ de\ uso/actor \quad (4)$$

6. Calcular la complejidad del diagrama de casos de uso como suma de CTE y CIE.

$$C_{usecasediagram} = CTE + CIE \quad (5)$$

7. Calcular la complejidad del sistema general como suma de la complejidad de todos los diagramas de casos de uso del sistema

$$C_{sys} = \sum_{k=1}^{n} [C_{usecasediagram}(k)]$$

Donde n es el número total de diagramas de casos de uso.

(6)

La utilidad de la métrica de complejidad discutida en este trabajo es aceptable sólo cuando se valida. Esta métrica utiliza las propiedades del marco de trabajo de Briand [44], para cotejar los resultados obtenidos y así establecer una resolución en cuanto al diseño del diagrama de casos de uso, estas propiedades se explican a continuación.

Tabla 4 Marco de propiedades de Briand [44].

Property No.	Property Description
#1	Non Negativity
#2	Null Value
#3	Symmetry
#4	Module Monotonicity
#5	Disjoint Module Additivity

Propiedad #1 No Negatividad: Complejidad del sistema mayor a cero, es decir Complejidad (S) ≥ 0. La complejidad del sistema, se calcula asignando pesos a las relaciones de asociación y dependencia que siempre son positivas, por lo que la complejidad no puede ser negativa y siempre debe ser positiva.

Propiedad #2 Valor nulo: Complejidad del sistema es nula, Complejidad (S) = 0. Si no existen relaciones en el diagrama de casos de uso, entonces la complejidad del sistema será nula.

Propiedad #3 Simetría: Para la Complejidad (S) = Complejidad (S_{-1}). La medida de complejidad en el trabajo propuesto no es sensible a la dirección de la relación. No depende de la convención utilizada para representar las relaciones. Por lo tanto, la medida de complejidad seguirá siendo la misma independientemente de la convención seguida.

Propiedad #4 Módulo Monotonicidad: La complejidad del sistema es mayor o igual a la suma de las complejidades de dos subsistemas que no tienen relaciones compartidas (salientes del subsistema general). Complejidad (S) ≥ [Complejidad (S1) + Complejidad (S2)].

Propiedad #5 Aditividad del módulo disjunto: La complejidad del sistema es igual a la suma de las complejidades de sus dos subsistemas disjuntos.

Capítulo 4

4. Fase de pruebas

4.1 Plan de pruebas

El plan de pruebas está orientado hacia la implementación de la "metodología en la determinación de la granularidad de un microservicio", fueron consideradas una serie de tareas divididas en tres fases, las cuales van de la mano con las actividades marcadas por la metodología, aunando actividades previas y posteriores a la aplicación de la metodología.

4.1.1 Objetivo del plan de pruebas

El siguiente plan de pruebas tiene como objetivo mostrar y mantener el orden y desarrollo de las actividades relacionadas a la implementación y evaluación de la metodología propuesta para determinar la granularidad de un microservicio, a través de la estructuración, documentación y análisis de cada tarea asociada.

4.1.2 Alcance del plan de pruebas

Con la implementación del plan de pruebas se busca abordar de forma completa la metodología para determinar la granularidad siguiendo un enfoque, arrojando como resultado sólo el diseño estructural de una arquitectura basada en microservicios.

4.1.3 Fase 1

La primera fase contempla la ejecución de cuatro actividades las cuales se centran en la determinación y documentación del proyecto de prueba, las cuales son:

- **Definición de los escenarios de pruebas:** esta tarea está enfocada en consultar y analizar ejemplos de repositorios de microservicios, esto con el fin de seleccionar proyectos ya implementados y funcionales, sirviendo como puntos de comparación su arquitectura actual y la arquitectura que se obtiene a partir de la metodología, permitiendo así, obtener algunas conclusiones sobre los resultados que arroja la metodología propuesta.
- **Recopilación de la información del escenario de pruebas:** se procede a recabar toda la información posible sobre el proyecto utilizado como ejemplo, esta información debe estar orientada como si se fuera hacer la construcción de un sistema desde cero, pues la metodología gira en torno al desarrollo de un proyecto real, por lo cual los datos proporcionados o recopilados deben ser lo más detallados posible. Como producto de esta actividad se debe tener un documento con el levantamiento de requerimientos del cliente.
- **Análisis de la información:** se comienzan a seguir los pasos marcados por la primera etapa de la metodología, que es el "análisis", se lleva a cabo la definición de involucrados, identificación de roles, reuniones y comprensión total del proyecto, dando

como resultado un documento de análisis, que concentrará un planteamiento de acuerdos del trabajo a desarrollar.

- **Determinación del enfoque a seguir:** de acuerdo al análisis de la información del caso de prueba, se define que enfoque o enfoques se seguirán para la determinación de la granularidad, esta tarea producirá un esquema de análisis como resultado (modelo de dominio, diagrama relacional de una BD, declaración de objetivos o diagrama de casos de uso), el cual o cuales serán en función del esquema a seguir.

4.1.4 Fase 2

La fase dos, es el desarrollo de las pruebas, esta fase está enfocada en la aplicación de la metodología de acuerdo al enfoque seleccionado. Aunque se involucran solo dos tareas, es importante recordar que por cada tarea hay varias actividades a desarrollar:

- **Implementación de la metodología:** Se llevan a cabo las actividades correspondientes al enfoque que se haya elegido, es importante seguir cada actividad indicada, así como realizar los modelos correspondientes solicitados en cada parte del proceso, cabe aclarar que también, se pueden originar e implementar nuevas estrategias durante la ejecución de actividades.
- **Verificación de cumplimiento de los requerimientos:** A través de un mapeo, se hace una validación de las metas planteadas por el proyecto y del modelo arquitectónico que se construyó, esta actividad tiene como objetivo comprobar que hayan sido contemplados todos los puntos solicitados por el cliente, y en caso contrario hacer los ajustes necesarios para cubrirlos en su totalidad.

4.1.5 Fase 3

La tercera fase está enfocada en el análisis de los resultados, pero no solo de los resultados arrojados del proyecto de prueba, si no también esta fase está orientada a la inspección del proceso seguido durante la implementación de la metodología. Las tareas involucradas son:

- **Comparación entre el proceso realizado y la metodología propuesta**: Cuando se ha finalizado con la aplicación de la metodología de acuerdo al enfoque seleccionado y se han obtenido una serie de resultados, se procede a hacer una comparación entre las actividades propuestas a realizar es decir la serie de pasos descritos en la metodología y lo que en verdad fue realizado durante la práctica, para esto se va haciendo una comparación y análisis exhaustivo entre cada paso ejecutado y cada paso propuesto, generando observaciones y conclusiones encontradas por cada aspecto.
- **Contraste entre la arquitectura original del proyecto de ejemplo y la arquitectura resultante de la metodología aplicada:** Identificar las situaciones que se están presentando y establecer conclusiones.
- **Retrospectiva general:** Se analizan las conclusiones obtenidas en la tarea anterior, y se lleva a cabo una retrospectiva de todo lo realizado y lo propuesto, teniendo a bien hacer los cambios necesarios para la mejora general del modelo presentado. Cada detalle, cada

cambio, tiene que ser sustentado en base a toda la documentación generada hasta el momento, dejando muy en claro su intención o propósito.

4.2 Definición de los escenarios de pruebas.

Para la definición de los escenarios a utilizar, se realizó una inmersión por internet, revisando los diferentes ejemplos de repositorios de microservicios, primero se obtuvieron todos aquellos proyectos que contaban con la información necesaria para entender el contexto general del sistema, como los requisitos de usuario, levantamiento de requerimientos, etc. Al final solo se contemplaron a los repositorios que presentaban información más detallada sobre el entorno general del proyecto.

Dentro de los ejemplos consultados se encuentran:

Tabla 5 Ejemplos de proyectos con Microservicios.

#	Nombre	Descripción	Link de información
1.	Cinema	Se reestructura el sistema de boletería, dulcería y catálogo de proyección de un cine en microservicios.	https://medium.com/@cramirez92/build-a-nodejs-cinema-microservice-and-deploying-it-with-docker-part-1-7e28e25bfa8b
2.	TeaStore	TeaStore emula una tienda web básica para té y suministros de té generados automáticamente.	https://github.com/Descartes-Research/TeaStore
3.	E-Commerce App	Aplicación de comercio electrónico nativa de la nube ficticia que utiliza una arquitectura de microservicios impulsada por Spring Cloud, Docker.	https://github.com/venkataravuri/e-commerce-microservices-sample#readme
4	Restaurant	Aplicación para la gestión de un restaurante, que incluye los escenarios de menú, pedidos, entrega y pagos.	C. Richardson (2019). Microservices Patterns. Whit examples in Java. Manning Publications Co. Shelter Island, NY 11964.

En base a toda la información consultada, y para dar seguimiento a la siguiente tarea del plan de pruebas, se optó por utilizar el ejemplo del sistema para la administración de la boletería, dulcería y catálogo de proyección de un cine (proyecto “Cinema”), así como el proyecto “TeaStore” que emula una tienda web básica para té y suministros de té, como escenarios para el desarrollo de pruebas de la metodología en la determinación de la granularidad de un microservicio.

4.3 Caso de prueba Cinema

4.3.1 Recopilación de la información

A continuación, se presenta la información encontrada hasta el momento sobre el proyecto "Cinema" [45], a través de un levantamiento de requerimientos.

El departamento de TI de un Cine tiene la tarea de estructurar un sistema, el cual se encargue de la gestión de las áreas de: taquilla, dulcería, café central y proyección, se solicita que este sistema sea construido utilizando la arquitectura de microservicios.

Requerimientos funcionales:

1) El sistema será operado por los empleados generales que desempeñan un rol de forma mensual, en función del rol son las actividades que podrán desempeñar a través del sistema. Los roles asignados son:
    - Empleado de taquilla
    - Empleado de dulcería
    - Empleado de café central
    - Empleado de piso
2) Existe otro rol: Coordinador de proyección, esta función es permanente, sus funciones son diferentes a la demás.
3) El último rol es el de gerente, es de cargo permanente.
4) Se requiere de un modo de autenticación, por lo cual es asignado un usuario y contraseña, para tener acceso al sistema.
5) El empleado general podrá ver las diferentes áreas a las que tiene acceso, que son: taquilla, dulcería, café central y piso. El empleado podrá elegir el área que le fue asignada, para su operación.
    - Taquilla: El área de taquilla se encargará de la venta de boletos, debe incluirse la venta en línea y los métodos de pago en efectivo, con tarjeta de crédito/débito y mercado pago (app). También se llevará a cabo la venta de tarjetas denominadas "miembro destacado", las cuales manejan descuentos en la compra de boletos y área de dulcería. Por lo cual debe contemplarse que en la venta de boletos también se haga la lectura de dichas membresías, para la aplicación de los descuentos cuando sea necesario.
    - Dulcería: La dulcería se encarga de la venta de snacks, bebidas y artículos de colección. Se debe contemplar la lectura de membresías para la aplicación de descuentos.
    - Café central: Venta de cafés, postres y alimentos.
    - Piso: Validación de boletos para acceso a las salas.
6) En el caso del Coordinador de proyección, no se mostrará en su pantalla las opciones de acceso de las demás áreas, esto también aplica para las demás áreas, ya que no podrán tener acceso a nada de lo que el Coordinador de proyección maneje.

- Coordinación de proyección: Esta área se encarga de la gestión de proyección de películas, lo cual incluye la asignación de salas y horarios a cada una de las películas para su proyección. El cine cuenta con 7 salas generales, 2 salas ejecutivas y 1 sala de pantalla gigante, y su horario de apertura al público es de las 10:00 a 23:59 hrs de lunes a domingo. Con esta información el coordinador debe poder realizar la calendarización de proyección de películas. Cabe mencionar que esta área debe estar relacionada con el área de taquilla, ya que la venta de boletos depende por completo de la gestión del catálogo de proyección de películas.

7) Para el gerente, al ingresar al sistema podrá ver todas las opciones tanto de empleado general como de coordinación de proyección, por lo cual su cuenta es la única que tiene acceso a todas las funcionalidades del sistema.
    - Gerencia: La gerencia cuenta con una función adicional, la cual se encarga de poder dar de alta y baja las películas que forman parte del catálogo del mes, contemplando los datos de nombre, cast, director, sinopsis y duración. Esta información debe estar definida para que el coordinador de proyección lleve a cabo la asignación de salas y horarios.

Requerimientos no funcionales:

1) El sistema debe contar con la suficiente seguridad para que solo los empleados tengan acceso a él.
2) El sistema debe estar disponible las 24/7.
3) El sistema debe poder ser operado desde computadoras de escritorio, computadora portátil y dispositivo móvil (smartphone).
4) Si el sistema se queda sin acceso a internet debe seguir operando con normalidad.
5) Eventualmente al sistema se le añadirán nuevas características, nuevas funciones y secciones, por lo cual debe quedar listo para ser implementadas.
6) Los colores predominantes a utilizar son: #4F242F, #FFE6EB, #6E6E6E

4.3.2 Análisis de la información

En base al análisis de la información presentada sobre el proyecto “Cinema” [45], se realizó un cuadro que identifica los roles, áreas y funciones participantes en el sistema. Así mismo se colocaron los involucrados en cada función y una sección de comentarios donde se colocaron notas importantes.

Tabla 6 Tabla de analisis de requerimientos para el proyecto Cinema.

ACTOR	ROL	AREAS DE ACCESO	FUNCIONES	OTROS INVOLUCRADOS	COMENTARIOS
Gerente	1) Gerente	Gerencia	• Alta de películas en el catálogo. • Baja de películas del catálogo.	NA	Datos para el alta: Nombre, cast, director, sinopsis y duración.
		Coordinación de proyección	• Asignación de salas a películas del catálogo. • Asignación de horarios a películas.	Coordinador de proyección	7 salas generales 2 salas VIP 1 sala pantalla mega Horario: lunes a domingo de 10:00 a 23:59 hrs.
		Taquilla	• Venta de boletos. • Venta de membrecías. • Lectura de membrecías para aplicación de descuentos.	Empleados de taquilla	Se realizan ventas de boletos en línea. Cobro: efectivo, tarjeta y mercado pago.
		Dulcería	• Venta de snacks. • Venta de bebidas. • Venta de artículos de colección. • Lectura de membrecías para aplicación de descuentos.	Empleados de dulcería	Cobro: efectivo, tarjeta y mercado pago.

		Café central	• Venta de cafés. • Venta de postres. • Venta de alimentos.	Empleados de café central	Cobro: efectivo, tarjeta y mercado pago.
		Piso	• Escaneo de boletos para acceso a las salas.	Empleados de piso	
Coordinador de proyección	1) Coordinador de proyección	Coordinación de proyección	• Asignación de salas a películas del catálogo. • Asignación de horarios a películas.	Catálogo de películas	7 salas generales, 2 salas VIP, 1 sala pantalla mega Horario: lun- domingo 10:00 a 23:59 hrs.
Empleado general	2) Empleado de taquilla	Taquilla	• Venta de boletos. • Venta de membresías. • Lectura de membresías para aplicación de descuentos.	Gerente	Se realizan ventas de boletos en línea. Cobro: efectivo, tarjeta y mercado pago.
	3) Empleado de dulcería	Dulcería	• Venta de snacks. • Venta de bebidas. • Venta de artículos de colección. • Lectura de membrecías para aplicación de descuentos.	Gerente	Cobro: efectivo, tarjeta y mercado pago.
	4) Empleado de café central	Café central	• Venta de cafés. • Venta de postres. • Venta de alimentos.	Gerente	Cobro: efectivo, tarjeta y mercado pago.
	5) Empleado de piso	Piso	• Escaneo de boletos para acceso a las salas.	Gerente	

4.3.3 Determinación del enfoque a seguir.

Como se ha visto en secciones anteriores, las funciones del sistema para el proyecto en cuestión, están representadas desde el punto de vista de los usuarios del sistema, que son: gerente, coordinador de proyección y empleado general. Estos usuarios desde ahora conocidos como actores, pueden tener diferentes instancias, como es el caso del empleado general, pues desempeñar el rol de empleado de taquilla, empleado de dulcería, empleado de café central y empleado de piso, cada rol con sus propias funciones definidas. De este modo, y dado que el diagrama de casos de uso muestra la relación que existe entre los actores y sus requisitos o expectativas del sistema, sin representar las acciones que tienen lugar o ponerlas en un orden lógico, se optó por utilizar el enfoque de Casos de Uso de la metodología como estrategia a seguir para el caso de prueba.

4.4 Diagrama de casos de uso.

El diagrama de casos de uso, es adecuado para representar claramente las funciones y/o objetivos más importantes de un sistema. Los casos de uso son un modelo de representación del software incorporado a UML, para la especificación de requisitos funcionales, modela la funcionalidad del sistema tal como la perciben los agentes externos, denominados actores, según se menciona Francisco García P. y Alicia García H, en su presentación Fundamentos de la vista de los Casos de Uso [46].

Para la elaboración del diagrama de casos de uso se ha seguido el estándar UML, basado en la norma ISO/IEC 19501 elaborada por el Comité Técnico ISO/IEC/TC JTC1, Tecnología de la Información, Subcomité SC 7, Ingeniería de Software y Sistemas, en colaboración con el Grupo de Gestión de Objetos (OMG). La cual está estrechamente relacionada con los trabajos de normalización de Open Distributed (ODP), cuyo marco de coordinación está en las Recomendaciones X.901-904 del UIT-T | ISO/IEC 10746 [47].

La figura 12 muestra el diagrama de casos de uso construido para el proyecto “Cinema”.

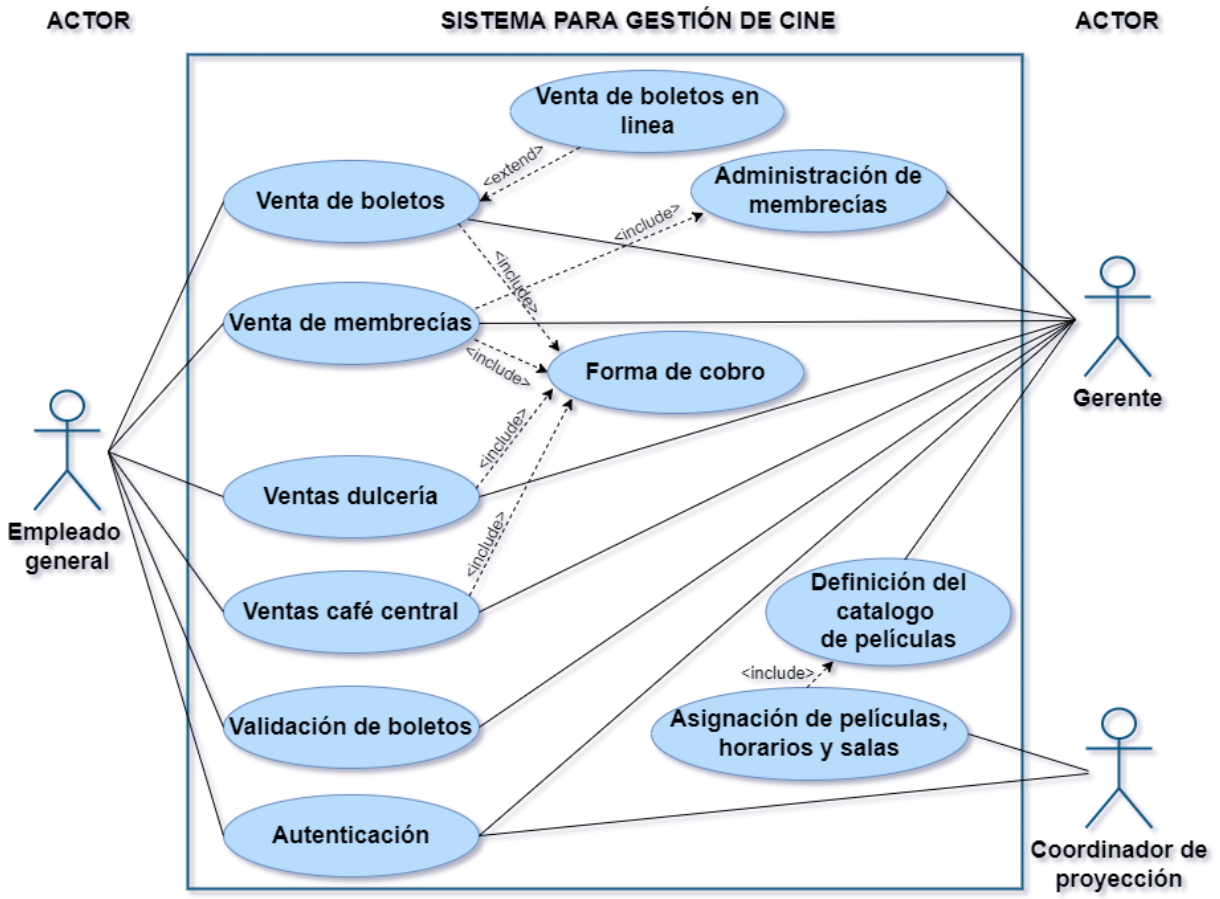

Figura 12 Diagrama de Casos de Uso proyecto "Cinema" elaboración propia.

4.5 Aplicación de la metodología por enfoque de Casos de Uso.

4.5.1 Paso 1: Validación del diagrama de casos de uso.

Esta sección se centra en la aplicación de la métrica de Sabharwal, Kaur y Sibal [42], para medir la complejidad del software en función de las relaciones de dependencia y asociación en el diagrama de casos de uso. La complejidad del diagrama de casos de uso se define en términos de relaciones de dependencia entre diferentes casos de uso y la asociación entre el actor y los casos de uso en el diagrama de casos de uso.

La métrica de caso de uso se calcula utilizando el procedimiento que se indica a continuación:

1. Dibuje el diagrama de casos de uso para el software a desarrollar.
2. Documente los casos de uso usando la Plantilla de caso de uso y los actores usando la Plantilla de actor.
3. Genere la matriz de casos de uso de disparadores (M_{trig}).
4. Genere matriz de casos de uso de iniciador (M_{init}).
5. Encuentre el efecto de iniciador combinado (CIE) de la matriz de casos de uso de iniciador (M_{init}) y el efecto de activación combinado (CTE) de la matriz de casos de uso de activación (M_{trig}).

6. Calcular la complejidad del diagrama de casos de uso como suma de CTE y CIE.
7. Calcular la complejidad del sistema general como suma de la complejidad de todos los diagramas de casos de uso del sistema.

Paso 1: La figura 12, presentada anteriormente, representa al diagrama de casos de uso construido para el proyecto “Cinema”.

Paso 2: Para derivar la métrica de complejidad basada en casos de uso, los casos de uso se documentan mediante una plantilla de caso de uso y los actores se documentan mediante una plantilla de actor. A continuación, se muestra la descripción de escenarios caso de uso “Autenticación” y del actor “Gerente” (tabla 7 y 8), utilizando las plantillas de Sabharwal, Kaur y Sibal [42], los cuadros de descripción restantes pueden ser encontrados en el Anexo B de este documento.

Descripción de casos de uso:

Tabla 7 Descripción del caso de uso Autenticación.

1.	**ID:**	**001**
2.	**Nombre del Caso de Uso:**	Autenticación
3.	**Descripción:**	A través de la validación de un nombre de usuario y contraseña se da acceso al sistema.
4.	**Tipo:**	Principal
5.	**Número de escenarios:**	1
6.	**Vector iniciador:**	Empleado general, coordinador de proyección, gerente.
7.	**Vector de activación:**	NA

Descripción de actores:

Tabla 8 Descripción del actor "Gerente".

1.	**ID del actor:**	001
2.	**Nombre del actor:**	Gerente
3.	**Descripción del actor:**	Empleado de más alto rango, tiene acceso a todas las funcionalidades del sistema con permisos especiales.
4.	**Vector de activación:**	CU001, CU002, CU005, CU006, CU007, CU008, CU009, CU010 y CU011.

Número de escenarios de los casos de uso principales.

Tabla 9 No. de escenarios por cada caso de uso.

Caso de uso principal	N° de escenarios
001	1
002	2
005	1
006	1
007	2
008	1
009	1
010	2
011	1

Paso 3: Matriz de disparadores de casos de uso (M_{trig}).

Tabla 10 Matriz de disparadores de casos de uso del proyecto "Cinema".

		Casos de uso principales									Casos de uso dependientes		
		001	002	005	006	007	008	009	010	011	003	004	TE
Actores	Gerente	3	3	3	3	3	3	3	3	3	0	0	36
	Empleado general	3	3	3	0	3	3	3	0	0	0	0	24
	Coordinador de proyec.	3	0	0	0	0	0	0	0	3	0	0	6
Casos de uso	001	0	0	0	0	0	0	0	0	0	0	0	0
	002	0	0	0	0	0	0	0	0	0	1	2	3
	005	0	0	0	2	0	0	0	0	0	0	2	4
	006	0	0	0	0	0	0	0	0	0	0	0	0
	007	0	0	0	0	0	0	0	0	0	0	2	2
	008	0	0	0	0	0	0	0	0	0	0	2	2
	009	0	0	0	0	0	0	0	0	0	0	0	0
	010	0	0	0	0	0	0	0	0	0	0	0	0
	011	0	0	0	0	0	0	0	2	0	0	0	2
												CTE =	79

TE = Efecto acumulativo de todas las dependencias de casos de uso en un i-ésimo caso de uso debido a los casos de uso desencadenados por él.
CTE = Efecto de activación combinado = **79**

Paso 4: Matriz de iniciadores de casos de uso (M_{init}).

Tabla 11 Matriz de iniciadores de casos de uso del proyecto "Cinema".

		Actores			Casos de uso principales									
		Gere.	Emp. Gen.	Coor. Proy.	001	002	005	006	007	008	009	010	011	IE
Casos de uso	001	3	3	3	0	0	0	0	0	0	0	0	0	9
	002	3	3	0	0	0	0	0	0	0	0	0	0	12
	003	0	0	0	0	1	0	0	0	0	0	0	0	1
	004	0	0	0	0	2	2	0	2	2	0	0	0	8
	005	3	3	0	0	0	0	0	0	0	0	0	0	6
	006	3	0	0	0	0	2	0	0	0	0	0	0	5
	007	3	3	0	0	0	0	0	0	0	0	0	0	12
	008	3	3	0	0	0	0	0	0	0	0	0	0	6
	009	3	3	0	0	0	0	0	0	0	0	0	0	6
	010	3	0	0	0	0	0	0	0	0	0	0	2	8
	011	3	0	3	0	0	0	0	0	0	0	0	0	6
													CIE =	79

IE = Efecto acumulativo de todas las dependencias de casos de uso en un i-ésimo caso de uso debido a sus casos de uso/actores iniciadores.
CIE = Efecto de iniciador combinado = **79**

Los valores de 1, 2 y 3 presentes en la matriz de disparadores y la matriz de iniciadores, fueron asignados de la siguiente manera:

- El caso de uso conectado por asociación con un actor, siempre será llamado a la celda de la matriz con la fuerza más alta de 3 puntos.
- El caso de uso conectado por incluir dependencia, siempre será llamado a la celda de la matriz con una fuerza de 2 puntos.
- El caso de uso conectado por extender la dependencia, se llamará a la celda de la matriz con una fuerza mínima de 1 punto.

Paso 5, 6 y 7: Complejidad del diagrama de casos de uso:

Tabla 12 Calculo de la complejidad del proyecto "Cinema".

FORMULA	DESCRIPCIÓN	SUSTITUCIÓN
$CIE = \sum_{i=1}^{n} IE(i)$	Sumatoria de todos los efectos acumulativos de las dependencias de casos de uso en un i-ésimo caso de uso debido a sus casos de uso/actores iniciadores.	CIE = 9 + 12 + 1 + 8 + 6 + 5 + 12 + 6 + 6 + 8 + 6 = **79**
$CTE = \sum_{i=1}^{n} TE(i)$	Sumatoria de todos los efectos acumulativos de las dependencias de casos de uso en un i-ésimo caso de uso debido a los casos de uso desencadenados por él.	CTE = 36 + 24 + 3 + 2 + 4 + 2 + 2 + 2 = **79**
$C_{usecasediagram} = CTE + CIE$	Complejidad del diagrama de casos de uso sumando CTE y CIE.	$C_{usecasediagram} = 79 + 79 = \mathbf{158}$
$C_{sys} = \sum_{k=1}^{n} [C_{usecasediagram}(k)]$	Calcular la complejidad del sistema general sumando la complejidad de todos los diagramas de casos de uso del sistema.	Como solo hay un diagrama de casos de uso la complejidad total es la obtenida anteriormente (158).

4.5.1.1 Interpretación de resultados de la métrica.

Para la interpretación de resultados se utilizaron las propiedades del marco de trabajo de Briand [44], explicadas anteriormente.

1) **Non Negativity:** El resultado de 158 puntos de complejidad del sistema de prueba, cumple con la propiedad de no negatividad, lo cual es lo correcto para este tipo de sistemas, donde las relaciones de asociación y dependencia deben ser siempre positivas por lo cual la complejidad no puede ser negativa, se comprueba entonces que el sistema cumplirá con este criterio.
2) **Null Value:** Si no existen relaciones en el diagrama de casos de uso, entonces la complejidad será de 0 puntos, y la complejidad del sistema será nula. En este caso como en el diagrama de casos de uso si existen relaciones entre los distintos casos de uso, la complejidad es de 158 puntos, lo que valida así la propiedad de valor nulo en el sistema.
3) **Symmetry:** En cuanto a la propiedad de simetría la medida de complejidad para este caso de prueba no es sensible a la dirección de la relación. Es decir que no depende de la convención utilizada para representar las relaciones. Por lo tanto, la medida de complejidad de 158 seguirá siendo la misma independientemente de la convención seguida cumpliendo con la propiedad de simetría.

4) **Module Monotonicity:** Para validar la propiedad de monotonicidad, se realizó una división del diagrama general de casos de uso en dos subsistemas, la figura 13 y 14 representan a los subsistemas obtenidos, estos subsistemas arrojaron los valores de 136 y 12 puntos de complejidad, teniendo una suma de 148 puntos, la cual, al ser comparada contra los 158 puntos del diagrama general de casos de uso, evidencia el cumplimiento de la cuarta propiedad de Briand, la cual explica que la complejidad del sistema es mayor o igual a la suma de las complejidades de dos subsistemas que no tienen relaciones compartidas.

Figura 13 Subsistema 1 obtenido del diagrama general de casos de uso.

Figura 14 Subsistema 2 obtenido del diagrama general de casos de uso.

5) **Disjoint Module Additivity:** Al intentar evaluar la propiedad de aditividad del módulo disjunto, se pudo notar que en este caso de prueba hay una interacción total de los casos de uso con los actores involucrados, es decir, que no se pueden obtener subsistemas disjuntos del diagrama general de casos de uso presentado, por lo cual esta quinta propiedad no puede ser comprobada para este ejemplo.

A través de la aplicación de la métrica y con el análisis anterior sobre los resultados obtenidos, se puede decir que este diagrama de casos de uso cumple con los criterios de complejidad recomendadas por Briand, se considera un constructo valido para su uso.

4.5.2 Paso 2: Análisis de las especificaciones

En esta actividad se realizaron las descripciones detalladas de los casos de uso, a través de cuadros de información, a continuación, en la tabla 13 se muestra la descripción del caso de uso "Autenticación", las descripciones restantes las encuentra en el Anexo C.

Tabla 13 Descripción del caso de uso Autenticación.

1.	**ID:**	**CU1**		
2.	**Nombre del Caso de Uso:**	Autenticación		
3.	**Autor:**	César Merino Ibáñez	**Última Modificación:**	César Merino
4.	**Fecha de Creación:**	24/marzo/2022	**Fecha de la Última Modificación:**	24/Marzo/2022
5.	**Actores:**	Empleado general Coordinador de proyección Gerente		
6.	**Descripción:**	A través de la validación de un nombre de usuario y contraseña se da acceso al sistema.		
7.	**Precondiciones:**	Registro previo de los usuarios, asignación de nombre de usuario y contraseña.		
8.	**Postcondiciones:**	Se ingresa al sistema, mostrando las funciones acordes al tipo de rol del usuario.		
9.	**Escenario Principal de éxito:**	1. Se inicia el sistema. 2. El usuario ingresa su nombre de usuario. 3. El usuario ingresa su contraseña. 4. El sistema valida el nombre de usuario y contraseña. 5. El sistema da acceso a sus funciones de acuerdo a los datos detectados.		
10.	**Escenario Alterno (2):**	- -		
11.	**Escenario Alterno (3):**	- -		
12.	**Escenario de Fracaso:**	1. El sistema no inicia. 2. Se ingresa un nombre de usuario invalido. 3. La contraseña ingresada es incorrecta. 4. No se da acceso al sistema.		
13.	**Prioridad:**	Alta		
14.	**Notas:**			

4.5.3 Paso 3: Identificación de operaciones y variables.

Se llevo a cabo la identificación de verbos y sustantivos en las descripciones de los casos de uso, estos verbos y sustantivos o frases sustantivas funcionan como las operaciones y variables de estado del sistema, la tabla 14 presenta una síntesis de la información obtenida.

Tabla 14 Identificación de verbos y sustantivos (operaciones y variables de estado) en las descripciones de los casos de uso.

OPERACIONES Y VARIABLES IDENTIFICADAS	NOTAS
Autenticación de usuarios.	Acceder al sistema.
Registro de usuarios.	
Validación de boletos.	Para acceder a las salas.
Escaneo de boletos.	
Venta de boletos.	
Impresión de boletos.	
Seleccionar asientos.	
Venta de boletos en línea.	Se requiere un correo electrónico.
Venta de membrecías.	
Activar membrecías.	
Registro de membrecías.	
Realizar cobro	
Aplicación de descuentos	Solo en compra de boletos y dulcería.
Validación de membrecías.	
Seleccionar forma de cobro.	Efectivo, tarjeta o mercado pago.
Ventas en dulcería.	
Ingresar orden.	
Ventas en café central.	
Ingresar orden.	
Coordinación de proyección.	
Asignar horarios.	Contemplar tiempo de limpieza.
Asignar salas.	
Administrar catálogo de películas.	
Registrar películas.	
Dar de baja películas.	

4.5.4 Paso 4: Creación de tabla de operaciones y relaciones.

Para la construcción de la tabla de operaciones y relaciones se utilizó la información provista en el paso anterior, siendo entonces identificada la relación que existe entre las operaciones y variables del sistema, si se trata de una relación de lectura, de escritura o ninguna de las dos. En la tabla 15 se puede observar de forma precisa el resultado de la ejecución de esta prueba.

Tabla 15 Tabla de operaciones y relaciones del proyecto "Cinema".

VARIABLES / OPERACIONES	Datos de usuario	Datos de autenticación	Boletos	Asientos	Datos de membrecía	Ticket de venta	Datos de cobro	Productos dulcería	Productos de café central	Datos horarios	Datos películas	Datos salas
Autenticación de usuarios.		Lee										
Registro de usuarios.	Escribe	Escribe										
Validación de boletos.			Lee									
Escaneo de boletos.			Lee									
Venta de boletos.			Escribe	Escribe		Escribe						
Impresión de boletos.			Escribe									
Seleccionar asientos.				Escribe								
Venta de boletos en línea.			Escribe	Escribe		Escribe						
Venta de membrecías.					Lee	Escribe						
Activar membrecías.					Escribe							
Registro de membrecías.					Escribe							
Realizar cobro							Lee					
Aplicación de descuentos					Lee		Lee					
Validación de membrecías.					Lee							
Seleccionar forma de cobro.							Lee					
Ventas en dulcería.						Lee						
Ingresar orden.								Escribe				
Ventas en café central.						Lee						
Ingresar orden.									Escribe			
Coordinación de proyección.											Lee	
Asignar horarios.										Escribe	Lee	

Asignar salas.											Lee	Escribe
Administrar catálogo de películas.											Escribe	
Registrar películas.											Escribe	
Dar de baja películas.											Escribe	

4.5.5 Paso 5: Descomposición, Grafo Bipartito.

La última prueba consistió en la elaboración de un grafo bipartito, en el cual los vértices están señalados como las operaciones y variables de estado, mientras que las conexiones son las relaciones de lectura o escritura que existen entre cada operación y variable, es importante hacer notar que cuando una conexión va de operación a operación no se coloca ningún peso, pero cuando la conexión es entre una operación y una variable debe asignársele un peso a dicha conexión, si la conexión es con una variable de escritura, esta tendrá un peso mayor que cuando se trate de una de lectura, en este caso se utilizó un peso de un punto para las relaciones de lectura y dos puntos para las de escritura.

En la figura 15 se puede observar la construcción del grafo para el proyecto de “Cinema”, del lado izquierdo se muestra el diagrama antes de identificar los microservicios, y al lado derecho se presenta también el diagrama, pero con una clara identificación de los candidatos a microservicios (óvalos de colores). Aunque pareciera que cada microservicio responde varias funciones, esto no es así, es importante aclarar que las líneas representan la relación de lectura o escritura; mientras que los textos de color negro representan las operaciones y los textos azules las variables de estado que se utilizan en cada operación. El nombre del microservicio será dado por la operación con más relaciones, en este caso están identificados en letra itálica remarcada, por lo tanto, para el caso de este proyecto se consideran como microservicios:

- Autenticación de usuarios.
- Venta de boletos.
- Venta de boletos en línea.
- Validación de boletos.
- Registro de membresías.
- Venta de membresías.
- Realizar cobro.
- Ventas en dulcería.
- Ventas en café central.
- Coordinación de proyección.
- Administrar catálogo de películas.

El grafo fue realizado de forma manual apoyándose de la tabla de operaciones y relaciones construida en la prueba anterior, aunque esta actividad también puede realizarse con ayuda de un software específico que construya grafos.

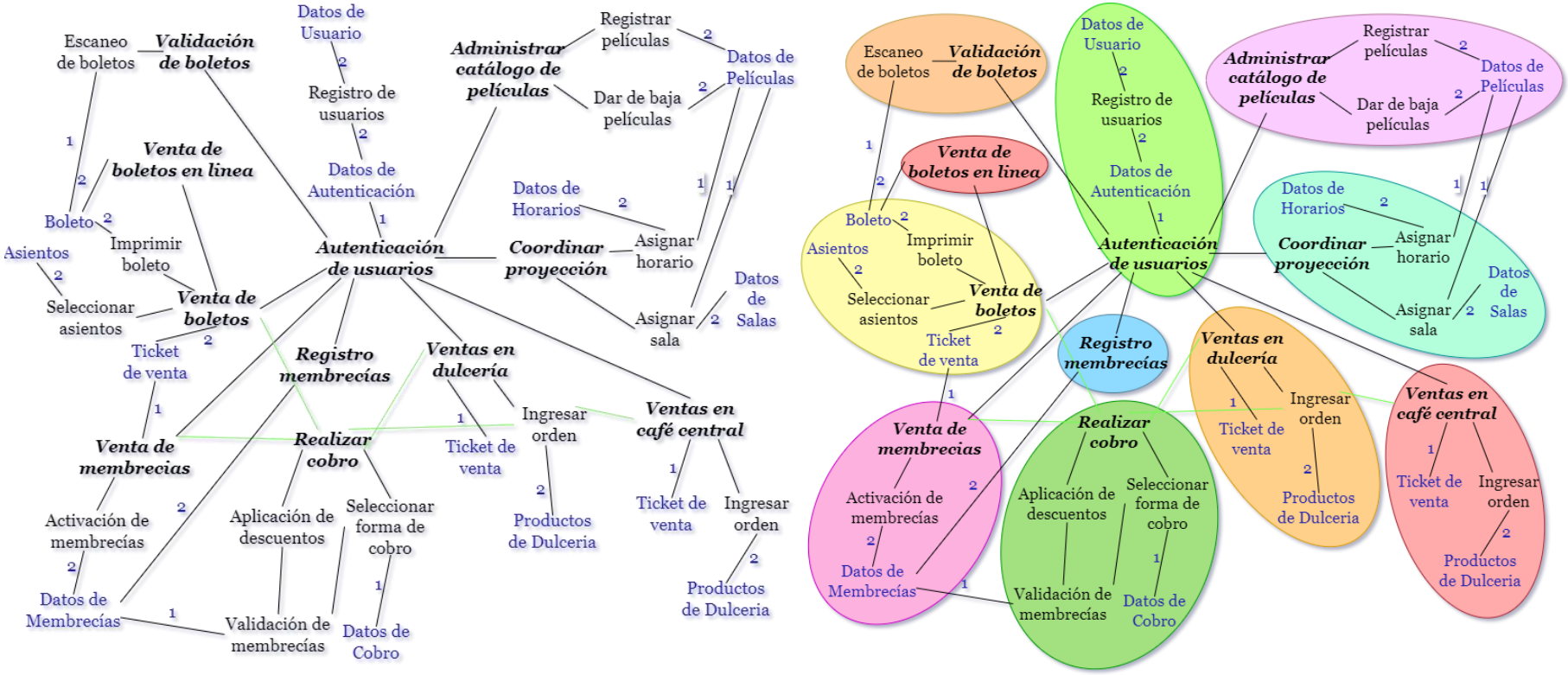

Figura 15 Identificación de Microservicios proyecto "Cinema".

4.6 Verificación de cumplimiento de los requerimientos.

Para la validación de los requerimientos se realizó un mapeo de los objetivos planteados por el cliente, a través de un checklist se fue corroborando que hayan sido contemplados todos los requerimientos del sistema en el modelo diseñado.

Tabla 16 Verificación de cumplimiento de los requerimientos del cliente.

#	Requerimiento	Considerado
1	El sistema será operado por los empleados generales del cine, coordinador de proyección y gerente.	✓
2	Se requiere de un modo de autenticación.	✓
3	Empleado general con acceso a taquilla, dulcería, café central y piso.	✓
4	Coordinador de proyección con acceso a la coordinación de proyección.	✓
5	El gerente tiene acceso a todas las áreas.	✓
6	Venta de boletos en taquilla.	✓
7	Venta de boletos en línea.	✓
8	Aceptación de pago en efectivo, con tarjeta y mercado pago.	✓
9	Venta en taquilla de membresías	✓
10	Lectura de membresías para aplicación de descuentos en taquilla.	✓
11	Venta de snacks, bebidas y artículos de colección en dulcería.	✓
12	Lectura de membresías para aplicación de descuentos en dulcería.	✓
13	Venta de cafés, postres y alimentos en café central.	✓
14	Validación de boletos para acceso a las salas.	✓
15	El coordinador de proyección puede asignar las salas y horarios a cada una de las películas para su proyección.	✓
16	La gerencia se encarga de poder dar de alta y baja las películas.	✓
17	Si el sistema se queda sin acceso a internet debe seguir operando con normalidad.	✓
18	Los colores predominantes a utilizar son: #4F242F, #FFE6EB, #6E6E6E.	✓

La tabla 16 muestra el cumplimiento de los requerimientos del cliente para el proyecto “Cinema”, si alguno o algunos de los requerimientos no hubiesen sido considerados, se procedería a rediseñar el modelo (diagrama de casos de uso para este ejemplo) tomando en cuenta las funciones faltantes.

4.7 Análisis de los resultados del caso de prueba “Cinema”.

La metodología propuesta comienza con la recopilación de la información, para este caso la mayor parte de la información fue adquirida a partir del repositorio de donde se obtuvo el ejemplo del proyecto “Cinema” [45], sin embargo, para tener un mejor acercamiento a la realidad se complementó este paso realizando una entrevista a un empleado general de un cine, obteniendo así una serie de requerimientos. Justamente el levantamiento de requerimientos funcionales y no funcionales, es uno de los puntos donde se debe prestar mucha atención dentro de las actividades de recopilación de información, pues a partir de la narrativa de las especificaciones del cliente, y la experiencia que tenga el arquitecto de software para identificar los requerimientos atómicos será el rumbo que tomará la arquitectura resultante. Si desde un inicio hay diferentes perspectivas de lo que involucra cada requerimiento, aunque los demás pasos de la metodología se apliquen de forma sistemática, al final cada arquitecto de software tendrá una forma distinta de granular un sistema, en este punto lo que se identifica es que se debería trabajar en una forma de sistematizar esta actividad, algo que se ha sugerido en los trabajos relacionados es el análisis sintáctico y semántico de los requerimientos del cliente.

La construcción de la tabla mostrada en el paso de análisis de la información representa una forma muy completa de analizar la información, pues a través de ella se contemplan los puntos clave del sistema, y además fue de gran ayuda en la ejecución de los pasos siguientes.

Se ha notado que, aunque esta metodología busca ser lo más objetiva posible, siempre existirá una parte humana y sobre todo profesional que intervendrá en la aplicación de la misma, como es el caso de la construcción del diagrama de salida de la etapa de análisis. Como ya se ha mencionado el diagrama de salida de la etapa de análisis definirá la ruta a seguir en la aplicación de la metodología, por lo cual un factor determinante en esta tarea es el criterio del arquitecto de software en base a la experiencia que tenga en el diseño de los diagramas propuestos (modelo de dominio, base de datos, declaración de objetivos o diagrama de casos de uso), es decir, que el arquitecto debe pensar con qué tipo de diagrama tiene más experiencia, y en base a su respuesta elegir y modelar.

Algo que no estaba contemplado en la metodología inicialmente propuesta era el seguimiento de un estándar para la construcción de los diagramas de cada enfoque, en este caso para la construcción del diagrama de casos de uso se utilizó el estándar UML, basado en la norma ISO/IEC 19501. Incluir estándares de diseño de los diagramas implementados en la metodología es un aspecto de suma importancia que hasta el momento no ha sido mencionado por los autores en sus procesos, como se mostró en los ajustes de la metodología de la sección 4, el seguimiento del estándar UML fue un aspecto introducido en el enfoque de casos de uso de esta metodología.

El primer paso del enfoque de casos de uso fue la validación del diagrama de casos de uso, para esto se aplicó la métrica de complejidad de Sabharwal, Kaur y Sibal [42], y se validó con el marco de propiedades de Briand [44], según el estudio de ambos trabajos, un diagrama de casos de uso será aceptable cuando su complejidad cumpla con al menos las cuatro primeras

propiedades de las cinco del marco de Briand, donde las propiedades de No negatividad, Valor Nulo, Simetría y la de Modulo de Monotonicidad son obligatorias, mientras que la de Aditividad del módulo disjunto, puede estar presente o no, esto es porque en consonancia con la propiedad de Monotonicidad esta última propiedad se justifica intuitivamente por el hecho de que el cierre transitivo de un grafo compuesto por varios subgrafos disjuntos es igual a la unión de los cierres transitivos de cada subgrafo tomado de forma aislada [44].

La identificación de operaciones y variables se realizó mediante un análisis manual, aunque se encontró software que pueden llevar a cabo esta tarea, de momento solo están disponibles para redacciones en inglés (p. ej., easyCRC [49], TextAnalysisOnline [50]), lo cual involucra cuestiones de semántica y redacción, esto supone una buena práctica para futuros trabajos, incluso haciendo el análisis de los mismos cuadros de ejemplo para hacer una comparativa entre los resultados manuales y los automatizados.

Al igual que la identificación de operaciones y variables del sistema, la construcción del grafo bipartito no dirigido se llevó a cabo de forma manual, en este punto se pudo notar a nivel práctico que el peso asignado a cada arista tal y como lo marca la metodología no causa un impacto a nivel de la identificación de microservicios, aunque si pudiera tener una gran importancia a nivel de desarrollo de cada microservicio, pues indica cuando una variable es calculada y utilizada en un mismo microservicio, o cuando un microservicio necesita de un valor final de otro microservicio para su funcionamiento. Se decidió que la asignación de pesos permanezca en la metodología final, pues estos son de ayuda durante la implementación, además que dan a conocer especificaciones de funcionamiento a los arquitectos de software.

Otro factor notado durante la elaboración del grafo fue que, aunque si existen clústeres claramente separados, de primer instancia resulta un poco complicado identificar cada uno de estos, por lo cual la creación del grafo debe ser lo más organizada posible, utilizando elementos que faciliten una comprensión visual del grafo, como el no cruzar aristas (conexiones), utilizar un color para identificar las operaciones y otro para las variables y acomodar las partes relacionadas juntas, de esta forma será más fácil la identificación de candidatos a microservicios.

4.7.1 Contraste entre la arquitectura original y la arquitectura resultante propuesta.

La figura 16 representa a la arquitectura original utilizada para desarrollo del proyecto “Cinema”, aunque el autor no especifica qué forma de descomposición utilizó para la identificación de microservicios, se pueden analizar algunos puntos a partir de la información presentada, cabe señalar que, la elección de este caso de estudio fue en función de la información general del proyecto, pues para fines de la aplicación de la metodología los requerimientos del cliente son los activos más importantes.

Como se mencionó anteriormente, para ejecutar una prueba lo más cercana posible a la realidad, se complementó la información del proyecto realizando una entrevista a un empleado de un cine. Para fines de este análisis, se hizo el contraste solo abarcando los mismos puntos referentes de información entre el sistema original y la arquitectura obtenida a partir de la metodología.

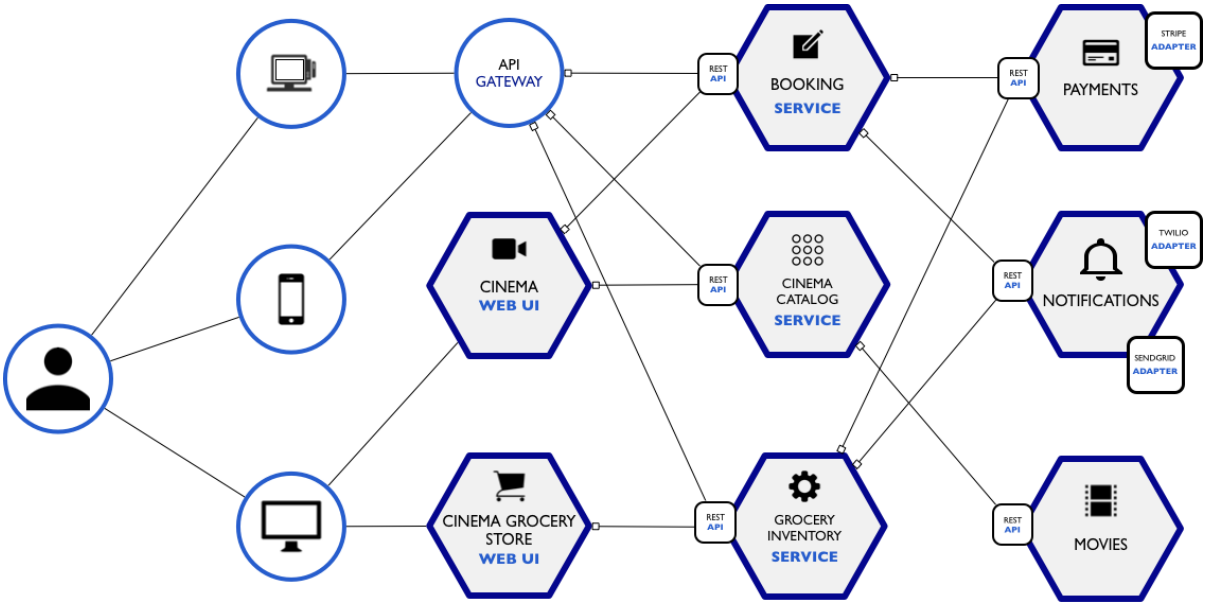

Figura 16 Arquitectura original del proyecto "Cinema", tomada del repositorio de Cristian Ramírez.

Para esta comparativa solo se tomaron en cuenta a los microservicios identificados a partir de los requerimientos del proyecto original involucrados en el caso de prueba, es decir, que los microservicios que surgieron a partir de la complementación de información que se hizo por medio de la entrevista no fueron tomados en cuenta, esto con el fin de obtener un contraste más equitativo entre la arquitectura original y la resultante por la metodología.

Como se puede observar la arquitectura original tiene la identificación de los siguientes servicios:

- Reservaciones (compras en línea).
- Catálogo de películas.
- Inventario de dulcería.
- Pago
- Notificaciones
- Películas

Mientras que la arquitectura arrojada por el seguimiento de la metodología propuesta (figura 3), dio como resultado:

- Venta de boletos.
- Venta de boletos en línea.
- Realizar cobro.
- Ventas en dulcería.
- Coordinación de proyección.
- Administrar catálogo de películas.

El servicio "Reservaciones" (de la arquitectura original) para la nueva arquitectura fue dividida en los microservicios: "Venta de boletos" y "Venta de boletos en línea", por su parte los

servicios “Catalogo de películas” y “Películas” de la arquitectura original ahora se encuentra distribuido en los microservicios “Administración del catálogo de películas” y “Coordinación de proyección” respectivamente.

El servicio “Inventario de dulcería” ahora está contenido dentro de la nueva arquitectura con el nombre “Ventas dulcería” y aplica lo mismo para el caso del servicio “Pago” el cual en la arquitectura propuesta está identificado como “Realizar cobro”. En lo que respecta al servicio “Notificaciones” de la arquitectura original, este no está siendo considerado en el nuevo diseño.

De acuerdo con los nuevos requerimientos considerados, los microservicios adicionales identificados a partir de las pruebas de este proyecto fueron:

- Autenticación de usuarios.
- Validación de boletos.
- Registro de membresías.
- Venta de membresías.
- Ventas en café central.

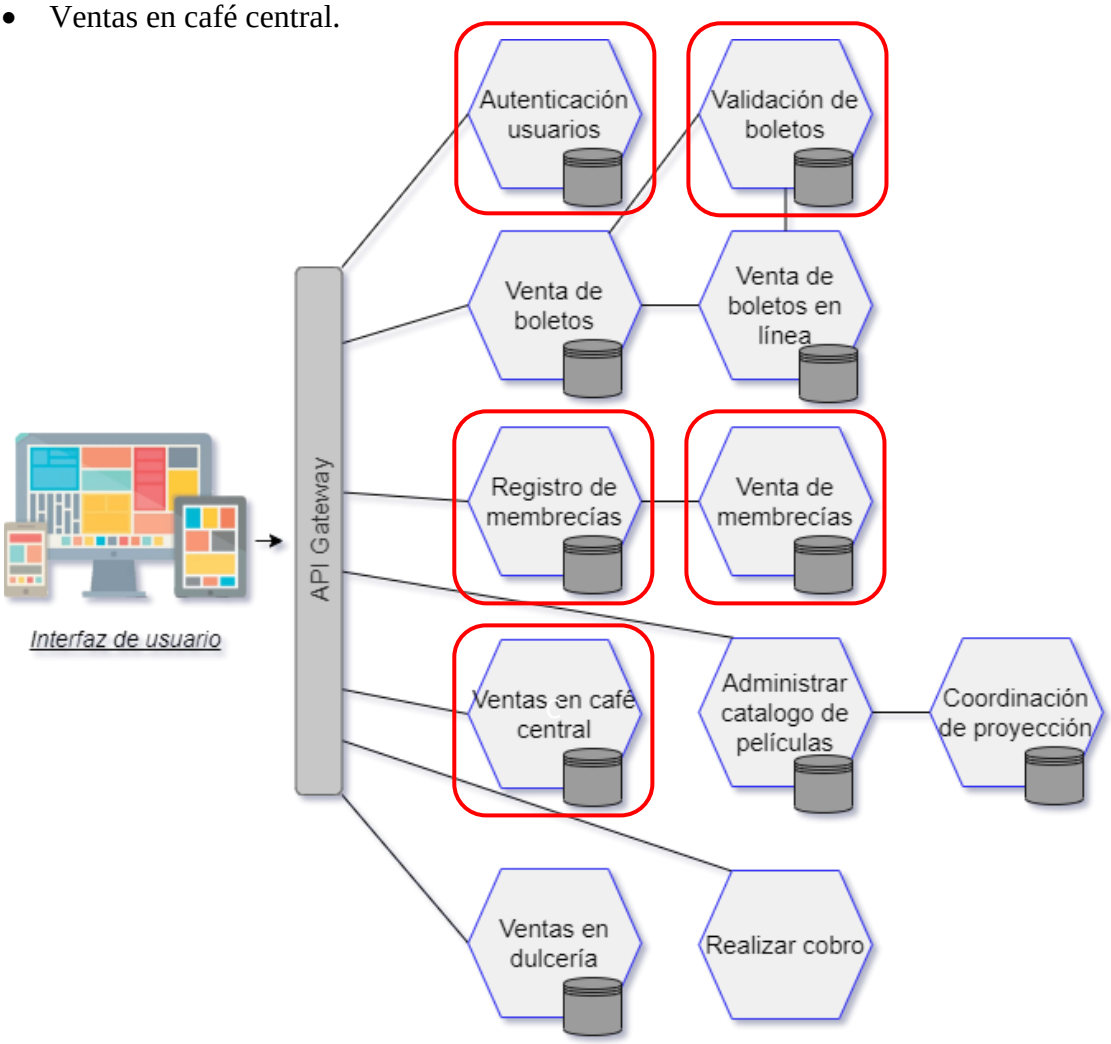

Figura 17 Arquitectura resultante por la metodología para el proyecto “Cinema”.

Al realizar un mapeo entre la arquitectura obtenida con la metodología y el diagrama de casos de uso, se puede notar que por cada caso de uso identificado existe un microservicio único que lo atiende. En la figura 18 se muestra como cada caso de uso es identificado como un microservicio, en algunos casos hay una variación en el nombre, pero el contexto general es el mismo.

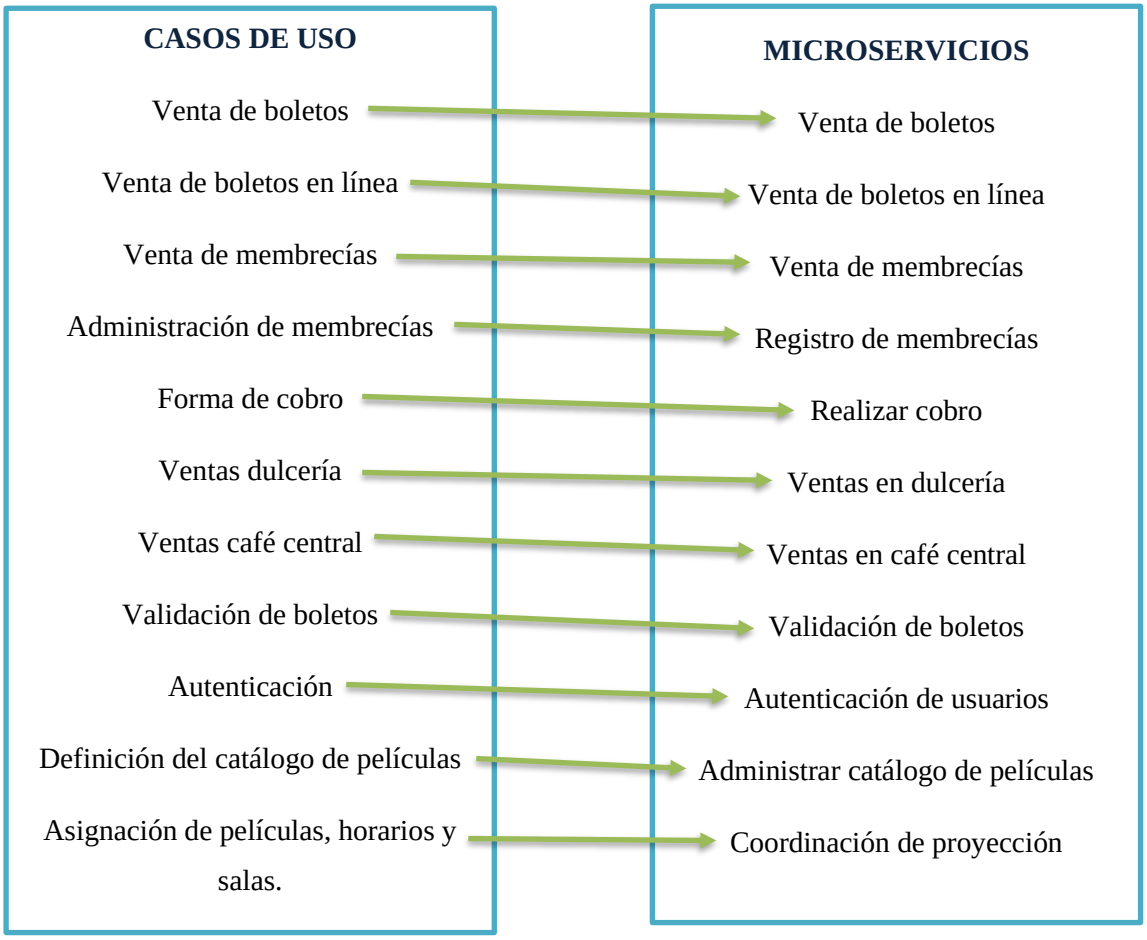

Figura 18 Mapeo entre casos de uso y microservicios.

5.7.2 Otra forma de comprobar el enfoque de casos de uso.

Otra forma que se puede utilizar para validar la correcta modularización de un sistema en microservicios a través del enfoque de casos de uso, es mediante un análisis del grafo bipartito construido durante la aplicación de la metodología, este análisis se basa en comprobar cómo cada línea secuencial de operaciones en el grafo, se traduce como un candidato a microservicio, esto permite constatar que cada microservicio identificado solo atiende a las operaciones que le permiten llevar a cabo su función, la cual debe ser única.

Se utilizó el grafo obtenido en el proyecto “Cinema” (que sirvió cómo caso de prueba), para realizar esta comprobación. Como se observa en la figura 19, se realizó una adaptación del grafo,

donde solo se dejaron las operaciones identificadas, quitando a las variables involucradas, ya que para este caso solo se toman en cuenta las conexiones entre las operaciones del sistema.

Cada línea secuencial de operaciones identificada fue marcada con un color distinto, las cuales se explican a continuación:

- De color verde claro hay dos operaciones dependientes; el "Registro de usuarios" la cual ayudará a la "Autenticación de usuarios" para acceder a las demás funciones del sistema.
- En color café esta la operación de "Escaneo de boletos" para su "Validación" y así dar acceso a las salas.
- De azul petróleo está la operación de "Venta de boletos en línea" la cual, al llevarse a cabo, inicia a la de "Imprimir boleto".
- En rojo esta la operación de "Venta de boletos", esta operación es la que se lleva a cabo desde la taquilla, y hace un llamado a "Seleccionar asientos", para posteriormente "Imprimir boletos".
- La operación de "Activación de membrecías" depende de la "Venta de membrecías", por eso se encuentran juntas identificadas de color azul claro.
- En color negro esta la operación de "Registrar membrecía" que no posee más operaciones asociadas.
- Aunque la operación de "Realizar cobro" pareciera tener diferentes líneas secuenciales, en realidad da la opción de "Seleccionar la forma de cobro" y hacerlo directamente, o bien "Aplicar descuento" haciendo una "Validación de membrecía", para que, en ambos casos, se termine esta función con la operación de "Imprimir ticket".
- De color naranja se encuentra la operación "Ventas dulcería", la cual invoca a "Ingresar orden".
- Mientras que de color verde obscuro esta "Ventas en café central" e "Ingresar orden".
- En color rosa se identifica la operación "Coordinar proyección", con la cual se puede "Asignar horario" o bien "Asignar sala" a una película para su proyección.
- La línea secuencial de la operación "Administrar catálogo de películas" que sirve para ejecutar a "Registrar películas" o "Dar de baja películas", está identificada de color azul marino.

Cabe mencionar que en la figura 19, los nombres de los microservicios se encuentran identificados con letra itálica.

Figura 19 Lineas secuenciales de operaciones para el proyecto "Cinema".

A través de este ejemplo claramente se puede observar cómo cada candidato a microservicio identificado por la metodología, corresponde justamente con una sola línea secuencial de operaciones; cabe señalar que estas líneas secuenciales atienden a una única función del sistema, lo cual a su vez cumple con la premisa del diagrama de casos de uso, la cual especifica que cada caso de uso atiende a una sola funcionalidad, esta situación de alguna forma también sustenta lo mencionado por algunos autores, que dicen que cada microservicio es un caso de uso. En términos generales, con esta otra comprobación, se puede reafirmar que la metodología propuesta para granular un sistema en microservicios utilizando el enfoque de casos de uso, cumple correctamente con los objetivos trazados en una arquitectura por microservicios.

4.8 Caso de prueba Tea Store

4.8.1 Recopilación de la información

A continuación, se presenta el caso de prueba “TeaStore”, este ejemplo fue desarrollado siguiendo cada paso en orden de la metodología propuesta.

TeaStore es una tienda en línea de té y servicios relacionados con el té [51]. Sus productos están clasificados en categorías. Para las compras en línea, la tienda admite una descripción general de los productos que incluye imágenes de vista previa para cada categoría y presenta una cantidad configurable de productos por página. Todas las páginas de TeaStore muestran una barra de encabezado de descripción general e incluyen el menú de categorías y el pie de página. Como contenido principal, muestra los productos de la categoría seleccionada, incluida la información resumida del producto y la imagen de vista previa.

Cada producto se puede ver en una página de producto separada que contiene información detallada, una imagen grande y anuncios de otros artículos de la tienda. Además del encabezado, el pie de página y la lista de categorías habituales, esta página incluye una imagen detallada del producto (proporcionada por el Servicio de proveedor de imágenes), una descripción y un precio. La página también contiene un panel de publicidad que sugiere tres productos en los que el usuario podría estar interesado. Los productos anunciados son proporcionados por el Servicio de recomendación y se seleccionan según el producto visto.

Todos los productos se pueden colocar en un carrito de compras y los usuarios pueden ordenar el carrito de compras actual. El usuario puede optar por modificar la cesta de la compra en cualquier momento. La página del carrito de compras enumera todos los productos incluidos actualmente en el carrito junto con cierta información del producto y la cantidad. La vista del carrito de compras también muestra anuncios de productos, que son, nuevamente, proporcionados por el servicio de Recomendador por separado y seleccionados según el contenido del carrito de compras.

Para realizar el pedido, el usuario deberá proporcionar información personal sobre la dirección de facturación y los datos de pago. La tienda también admite la autenticación y el inicio de sesión del usuario. Los usuarios registrados pueden ver su historial de pedidos después de iniciar sesión.

4.8.2 Análisis de la información

Requerimientos funcionales identificados:

- Los productos deben estar clasificados en categorías.
- El sistema muestra una descripción general de los productos, incluyendo imágenes de vista previa para cada categoría.
- Se debe poder configurar la cantidad de productos presentados por página.
- Todas las páginas del sistema deben tener:
  - Una barra de encabezado de descripción general.
  - Un menú de categorías.
  - El pie de página del mapa del sitio.
- Como contenido principal, mostrar los productos de la categoría seleccionada, incluida la información resumida del producto y la imagen de vista previa.
- Cada producto se puede ver en una página de producto separada que contiene información detallada como descripción y precio del artículo, una imagen grande y anuncios de otros artículos de la tienda.
- La página debe contener un panel de publicidad que sugiere tres productos en los que el usuario podría estar interesado.
- Los productos por comprar se colocan en un carrito de compras, el cual podre ser ordenado en el momento que lo desee el cliente.
- El cliente puede modificar la cesta de la compra en cualquier momento.

- El carrito de compras enumera todos los productos incluidos actualmente en el carrito, junto con cierta información del producto y la cantidad.
- En la vista del carrito de compras se muestra anuncios de productos recomendados, los cuales son proporcionados según el contenido del carrito de compras e historial del cliente.
- El sistema también admite la autenticación y el inicio de sesión del usuario, mediante un registro previo de sus datos.
- Los usuarios registrados pueden ver su historial de pedidos después de iniciar sesión.
- Para realizar el pedido, en el caso de un usuario no registrado, el cliente deberá proporcionar información para envío del producto, datos de pago y datos de facturación en caso que lo requiera.

4.8.3 Determinación del enfoque a seguir.

Al igual que con el ejemplo “Cinema”, y para fines de que las pruebas siguieran una misma línea de actividades, se decidió seguir trabajando con el enfoque de casos de uso, aprovechando así, las adecuaciones realizadas en la metodología en este enfoque.

4.9 Diagrama de casos de uso.

El diagrama de casos de uso fue construido siguiendo el estándar UML, basado en la norma ISO/IEC 19501. La figura 12 muestra el diagrama de casos de uso construido para el proyecto “Tea Store”.

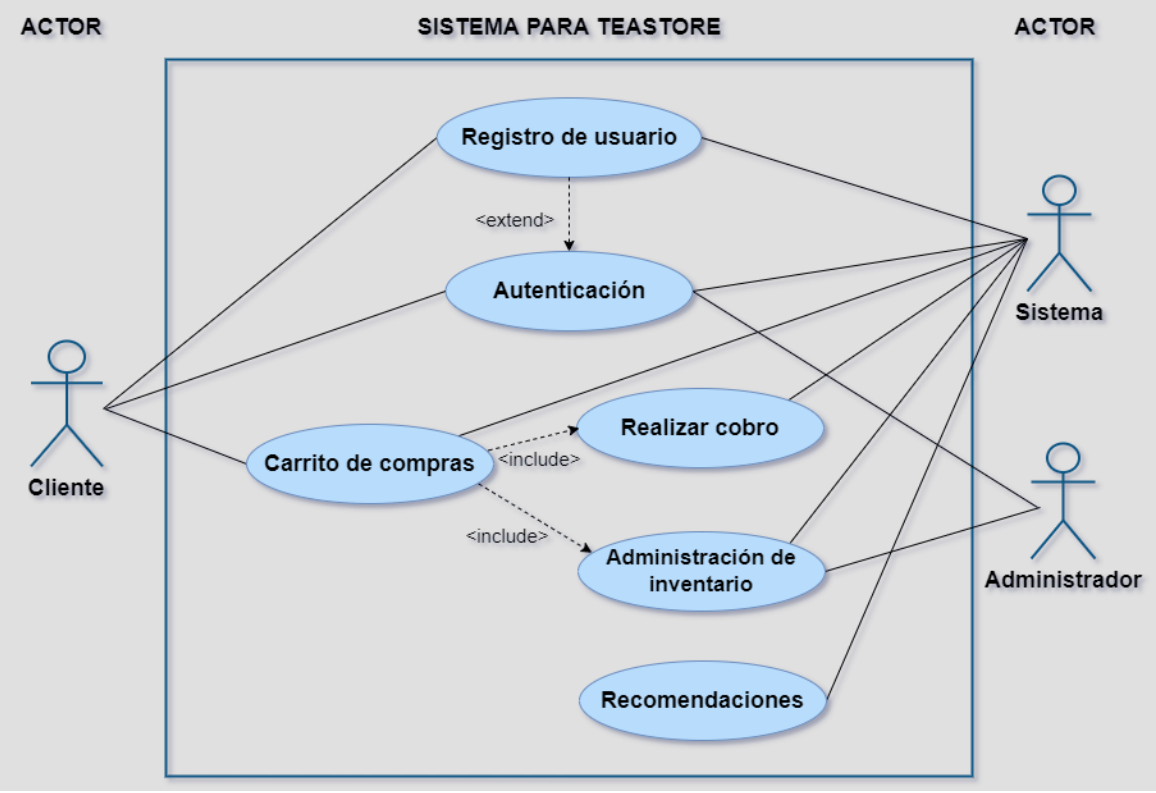

Figura 20 Diagrama de casos de uso para el proyecto "TeaStore", elaboración propia.

4.10 Aplicación de la metodología por enfoque de Casos de Uso.

4.10.1 Paso 1: Validación del diagrama de casos de uso.

Paso 1: La figura 20 representa al diagrama de casos de uso utilizado para la aplicación de la métrica de complejidad.

Paso 2: A continuación, en la tabla 17 se describe el caso de uso "Registro de usuario" utilizando la plantilla de Sabharwal, Kaur y Sibal [42], es importante mencionar que las descripciones de los casos de uso y la de los actores restantes, pueden ser encontradas en los anexos D de este documento.

Tabla 17 Plantilla del caso de uso Registro de usuario.

1.	**ID:**	**001**
2.	**Nombre del Caso de Uso:**	Registro de usuario
3.	**Descripción:**	El cliente puede registrarse para realizar una compra, almacenando su información en el sistema.
4.	**Tipo:**	Principal
5.	**Número de escenarios:**	1
6.	**Vector iniciador:**	Cliente, Sistema
7.	**Vector de activación:**	CU2

Número de escenarios de los casos de uso principales.

Tabla 18 No. de escenarios por cada caso de uso.

Caso de uso principal	N° de escenarios
001	1
002	2
003	4
004	2
005	3
006	2

Paso 3: Matriz de disparadores de casos de uso (M_{trig}).

Tabla 19 Matriz de disparadores de casos de uso del proyecto "TeaStore".

		Casos de uso principales						Casos de uso dependientes	
		001	002	003	004	005	006	NA	TE

Actores	Cliente	3	3	3	0	0	0	- -	21
	Sistema	3	3	3	3	3	3	- -	42
	Administrador	0	3	0	0	3	0	- -	15
Casos de uso	001	0	1	0	0	0	0	- -	1
	002	0	0	0	0	0	0	- -	0
	003	0	0	0	0	0	0	- -	0
	004	0	0	0	2	2	0	- -	4
	005	0	0	0	0	0	0	- -	0
	006	0	0	0	0	0	0	- -	0
								CTE =	83

E = Efecto acumulativo de todas las dependencias de casos de uso en un i-ésimo caso de uso debido a los casos de uso desencadenados por él.
CTE = Efecto de activación combinado = 83

Paso 4: Matriz de iniciadores de casos de uso (M_{init}).

Tabla 20 Matriz de iniciadores de casos de uso del proyecto "TeaStore".

		Actores			Casos de uso principales						
		Cliente	Sistema	Admin	001	002	003	004	005	006	IE
Casos de uso	001	3	3	0	0	0	0	0	0	0	6
	002	3	3	3	1	0	0	0	0	0	19
	003	3	3	0	0	0	0	0	0	0	24
	004	0	3	0	0	0	2	0	0	0	8
	005	0	3	3	0	0	2	0	0	0	20
	006	0	3	0	0	0	0	0	0	0	6
										CIE =	83

IE = Efecto acumulativo de todas las dependencias de casos de uso en un i-ésimo caso de uso debido a sus casos de uso/actores iniciadores.
CIE = Efecto de iniciador combinado = 83

Paso 5, 6 y 7: Complejidad del diagrama de casos de uso:

Tabla 21 Calculo de la complejidad del proyecto "TeaStore".

FORMULA	DESCRIPCIÓN	SUSTITUCIÓN
$CIE = \sum_{i=1}^{n} IE(i)$	Sumatoria de todos los efectos acumulativos de las dependencias de casos de uso en un i-ésimo caso de uso debido a sus casos de uso/actores iniciadores.	CIE = 6 + 19 + 24 + 8 + 20 + 6 = 83
$CTE = \sum_{i=1}^{n} TE(i)$	Sumatoria de todos los efectos acumulativos de las dependencias de casos de uso en un i-ésimo caso de uso debido a los casos de uso desencadenados por él.	CTE = 21 + 42 + 15 + 1 + 4 = 83
$C_{usecasediagram} = CTE + CIE$	Complejidad del diagrama de casos de uso sumando CTE y CIE.	$C_{usecasediagram} = 83 + 83 = 166$
$C_{sys} = \sum_{k=1}^{n} [C_{usecasediagram}(k)]$	Calcular la complejidad del sistema general sumando la complejidad de todos los diagramas de casos de uso del sistema.	Como solo hay un diagrama de casos de uso la complejidad total es la obtenida anteriormente (166).

4.10.1.1 Interpretación de resultados de la métrica.

Al aplicar dicha métrica se observó el cumplimiento de las propiedades de Briand [44]:

Non Negativity: El resultado de 166 puntos de complejidad del sistema de prueba, cumple con la propiedad de no negatividad, pues las relaciones de asociación y dependencia deben ser siempre positivas, se comprueba entonces que el sistema cumple con este criterio.

Null Value: Si no existen relaciones en el diagrama de casos de uso, entonces la complejidad será de 0 puntos, pero en este caso como en el diagrama de casos de uso si existen relaciones entre los distintos casos de uso, hay una complejidad de 166, lo que valida así la propiedad de valor nulo en el sistema.

Symmetry: La medida de complejidad para este caso de prueba tampoco es sensible a la dirección de la relación. Por lo tanto, la medida de complejidad de 166 seguirá siendo la misma independientemente de la convención seguida, cumpliendo así con la propiedad de simetría.

Module Monotonicity: Para validar esta propiedad, se realizó una división del diagrama general de casos de uso en dos subsistemas, las figuras 21 y 22 representan a los subsistemas obtenidos, estos subsistemas arrojaron los valores de 136 y 30 puntos de complejidad, teniendo una suma de 166 puntos, la cual, al ser comparada contra los 166 puntos del diagrama general de casos de uso, evidencia el cumplimiento la cuarta propiedad de Briand, la cual explica que la complejidad del sistema general es mayor o igual a la suma de las complejidades de dos subsistemas.

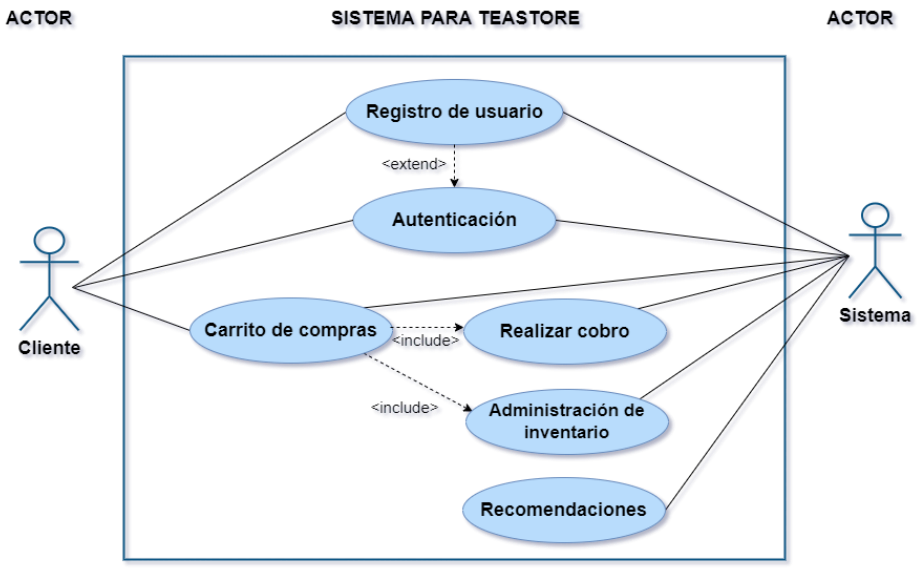

Figura 21 Subsistema 1 obtenido del diagrama general de casos de uso del proyecto TeaStore.

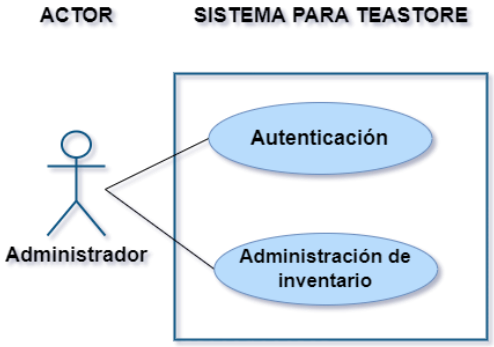

Figura 22 Subsistema 2 obtenido del diagrama general de casos de uso del proyecto TeaStore.

Disjoint Module Additivity: Al no haber una interacción total de los casos de uso con los actores involucrados, es decir, que no se pueden obtener subsistemas disjuntos del diagrama general de casos de uso presentado, esta quinta propiedad no puede ser comprobada para este ejemplo.

De forma general, esta situación sugiere que se ha realizado un correcto diseño del diagrama de casos de uso, dando paso así a la aplicación del paso 2 de la metodología.

4.10.2 Paso 2: Análisis de las especificaciones

Para el análisis de las especificaciones se realizaron las descripciones de los casos de uso, la descripción detallada del caso de uso “Registro de usuario” se presenta en la tabla 22, en el anexo E puede encontrar los cuadros de descripción restantes.

Tabla 22 Descripción del caso de uso Registro de usuario.

1.	**ID:**	**CU1**		
2.	**Nombre del Caso de Uso:**	Registro de usuario		
3.	**Autor:**	César Merino Ibáñez	**Última Modificación:**	César Merino
4.	**Fecha de Creación:**	04/agosto/2022	**Fecha de la Última Modificación:**	04/agosto/2022
5.	**Actores:**	Sistema Cliente		
6.	**Descripción:**	El cliente puede registrarse para realizar una compra, almacenando su información en el sistema.		
7.	**Precondiciones:**	Se solicita información personal básica al cliente.		
8.	**Postcondiciones:**	Se genera un usuario y contraseña para el cliente.		
9.	**Escenario Principal de éxito:**	6. Se inicia el sistema. 7. El cliente selecciona la opción de registrarse. 8. El cliente ingresa los datos solicitados. 9. El sistema registra los datos correctamente. 10. El sistema manda mensaje de confirmación de registro.		
10.	**Escenario Alterno (2):**	- -		
11.	**Escenario Alterno (3):**	- -		
12.	**Escenario de Fracaso:**	5. El usuario ingresa datos erróneos. 6. El usuario ingresa los datos incompletos. 7. El sistema no puede guardar la información.		
13.	**Prioridad:**	Alta		
14.	**Notas:**			

4.10.3 Paso 3: Identificación de operaciones y variables.

Se llevo a cabo la identificación de verbos y sustantivos en las descripciones de los casos de uso, estos verbos y sustantivos o frases sustantivas funcionan como las operaciones y variables de estado del sistema, la tabla 23 presenta una síntesis de la información obtenida.

Tabla 23 Identificación de verbos y sustantivos (operaciones y variables de estado) en las descripciones de los casos de uso.

OPERACIONES Y VARIABLES IDENTIFICADAS	NOTAS
Registro de usuarios.	Nombre, fecha de nacimiento, dirección.
Autenticar usuarios.	
Confirmar registro.	
Mostrar perfil del cliente.	
Mostrar historial de compras del cliente.	
Añadir artículo al carrito.	Seleccionar productos.
Confirmar pedido.	
Imprimir ticket de compra.	
Solicitar datos por no registro.	Nombre, dirección, medio de pago, datos de facturación (opcional).
Realizar cobro.	Mandar mensaje del estatus del pedido.
Quitar artículo del carrito.	
Buscar artículo.	
Dar de alta artículo.	
Modificar datos del artículo.	
Confirmar cambios en el artículo.	
Dar de baja artículo.	
Confirmar baja del artículo.	
Verificar artículos seleccionados.	
Verificar artículos comprados.	
Revisar que artículos se han comprado juntos.	
Recomendar artículos.	

4.10.4 Paso 4: Creación de tabla de operaciones y relaciones.

Para la construcción de la tabla de operaciones y relaciones se utilizó la información provista en el paso anterior, siendo entonces identificada la relación que existe entre las operaciones y variables del sistema, si se trata de una relación de lectura, de escritura o ninguna de las dos. En la tabla 24 se puede observar de forma precisa el resultado de la ejecución de esta prueba.

Tabla 24 Tabla de operaciones y relaciones del proyecto "Tea Store".

VARIABLES / **OPERACIONES**	Información de registro	Nombre de usuario	Contraseña	Confirmación de registro	Perfil del cliente	Historial de compras	Funciones del sistema	Artículo	Carrito de compras	Pedido	Ticket de compra	Datos por no registro	Medio de pago	Estatus del pago	Artículos recomendados
Registrar usuarios.	Escribe	Escribe	Escribe	Lee									Escribe		
Autenticar usuarios.		Lee	Lee										Lee		
Confirmar registro.	Lee			Lee											
Mostrar perfil del cliente.	Lee				Escribe										
Mostrar historial de compras.					Lee	Escribe									
Añadir artículo al carrito.							Lee	Escribe	Escribe						
Confirmar pedido.									Lee	Lee					
Imprimir ticket.									Lee	Lee	Escribe				
Solicitar datos por no registro.												Escribe			
Realizar cobro.										Lee		Lee	Lee	Escribe	
Quitar artículo del carrito.							Lee	Lee	Escribe						
Buscar artículo.								Lee							
Dar de alta artículo.								Escribe							
Modificar datos del artículo.								Escribe							
Confirmar cambios en el artículo.								Lee							

OPERACIONES	Información de registro	Nombre de usuario	Contraseña	Confirmación de registro	Perfil del cliente	Historial de compras	Funciones del sistema	Artículo	Carrito de compras	Pedido	Ticket de compra	Datos por no registro	Medio de pago	Estatus del pago	Artículos recomendados
Dar de baja artículo.								Escribe							
Confirmar baja del artículo.								Lee							
Verificar artículos seleccionados.								Lee							
Verificar artículos comprados.										Lee					
Revisar que artículos se han comprado juntos.										Lee					
Recomendar artículos.															Escribe

4.10.5 Paso 5: Descomposición, Grafo Bipartito.

La última prueba consistió en la elaboración de un grafo bipartito, en el cual los vértices están señalados como las operaciones y variables de estado, mientras que las conexiones son las relaciones de lectura o escritura que existen entre cada operación y variable.

En la figura 14 se puede observar la construcción del grafo para el proyecto de “TeaStore”, del lado izquierdo se muestra el grafo original, es decir, antes de identificar los microservicios, y al lado derecho se presenta el mismo diagrama, pero con una clara identificación de los candidatos a microservicios (óvalos de colores).

Al igual que con el caso de prueba anterior (proyecto Cinema), el nombre de cada microservicio será dado por la operación que tenga más operaciones dependientes asociadas, en la figura 14 los nombres de los microservicios se identifican en letra itálica, bold:

- Autenticar usuarios.
- Registrar usuarios.
- Realizar cobro.
- Carrito de compras.
- Administrar inventario.
- Recomendar artículos.

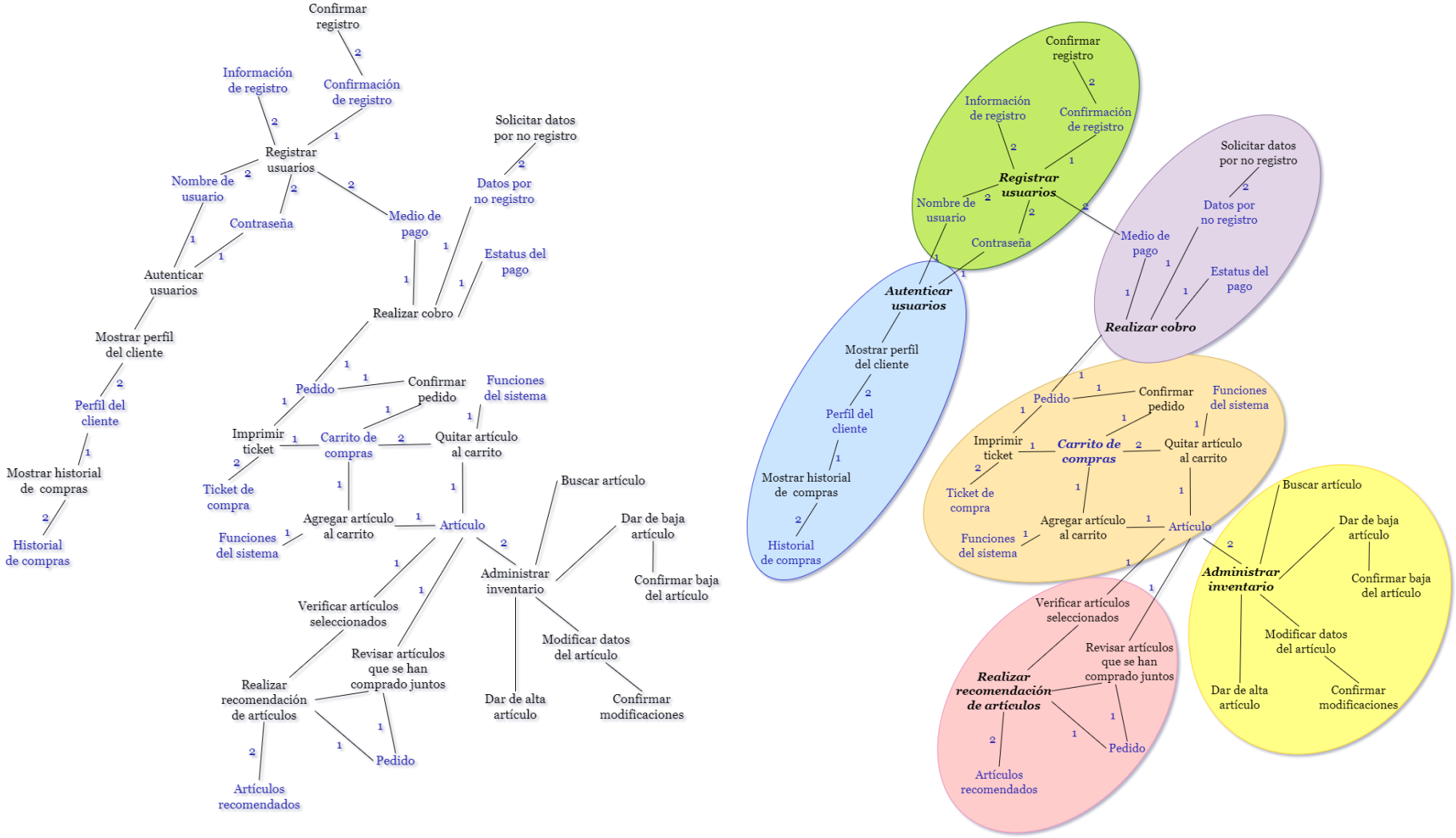

Figura 23 Identificación de Microservicios proyecto "TeaStore".

4.11 Verificación de cumplimiento de los requerimientos.

Para la validación de los requerimientos se realizó un mapeo de los objetivos planteados por el cliente, a través de un checklist se fue corroborando que hayan sido contemplados todos los requerimientos del sistema en el modelo diseñado.

Tabla 25 Verificación de cumplimiento de los requerimientos del cliente.

#	Requerimiento	Considerado
1	Los productos deben estar clasificados en categorías.	✓
2	Como contenido principal, mostrar los productos de la categoría seleccionada, incluida la información resumida del producto y la imagen de vista previa.	✓
3	Cada producto se puede ver en una página de producto separada que contiene información detallada como descripción y precio del artículo, una imagen grande y anuncios de otros artículos de la tienda.	✓
4	La página debe contener un panel de publicidad que sugiere tres productos en los que el usuario podría estar interesado.	✓
5	Los productos por comprar se colocan en un carrito de compras, el cual podre ser ordenado en el momento que lo desee el cliente.	✓
6	El cliente puede modificar la cesta de la compra en cualquier momento.	✓
7	El carrito de compras enumera todos los productos incluidos actualmente en el carrito, junto con cierta información del producto y la cantidad.	✓
8	En la vista del carrito de compras se muestra anuncios de productos recomendados, los cuales son proporcionados según el contenido del carrito de compras e historial del cliente.	✓
9	El sistema también admite la autenticación y el inicio de sesión del usuario, mediante un registro previo de sus datos.	✓
10	Los usuarios registrados pueden ver su historial de pedidos después de iniciar sesión.	✓
11	Para realizar el pedido, en el caso de un usuario no registrado, el cliente deberá proporcionar información para envío del producto, datos de pago y datos de facturación en caso de que lo requiera.	✓

4.12 Análisis de los resultados del caso de prueba “Tea Store”.

De acuerdo a la metodología propuesta, se comenzaron las pruebas con las actividades de recolección y análisis de información del proyecto. Posteriormente se pasó a la construcción del diagrama de casos de uso, la cual fue realizada siguiendo el estándar UML como lo marca la metodología en este enfoque, inmediatamente que se terminó de realizar el diagrama de casos de uso se procedió a su validación utilizando la métrica de complejidad de Sabharwal, Kaur y Sibal [42]. Aunque como tal no se tiene una unidad de medida para el puntaje arrojado por la métrica de complejidad, algo interesante notado en este paso, fue que para este caso de prueba (que de primera vista luce más chico) el puntaje de complejidad obtenido fue de 166 puntos,

este valor es más alto que el obtenido en el caso de prueba “Cinema” con 158 puntos, siendo que este último proyecto (Cinema) cuenta con 11 microservicios identificados mientras que el de “TeaStore” cuenta con solo 6 microservicios. Aunque es muy pronto para sacar conclusiones a partir de solo dos pruebas, se puede ver que posiblemente existe una relación entre el puntaje de complejidad y el tamaño del proyecto. Pues al menos con estos casos de prueba, se puede decir que a mayor puntaje de complejidad el proyecto es más chico (en cantidad de microservicios), por el contrario, con menor puntaje de complejidad el proyecto tiene una perspectiva más amplia al contemplar más microservicios.

Para analizar las especificaciones de los casos de uso se construyeron los cuadros de descripción de cada caso de uso, a través de los cuales se identificaron la operaciones y variables del sistema y se creó así la tabla de relaciones. Con esta información se construyó el grafo bipartito y por último se identificaron a los candidatos a microservicios.

4.12.1 Contraste entre la arquitectura original y la arquitectura resultante propuesta.

El caso de prueba de la “Tea Store”, según sus autores [51], presenta una arquitectura identificada con 5 microservicios y un servicio de Registro, como se puede observar en la Figura 24. Los microservicios identificados por sus autores son: WebUI, Proveedor de imágenes, Autenticación, Persistencia y Recomendador. Hay que recordar que además de los cinco microservicios, la TeaStore tiene un servicio “Registro”, el cual no forma parte de la aplicación (que se está probando), pero es un servicio de soporte necesario.

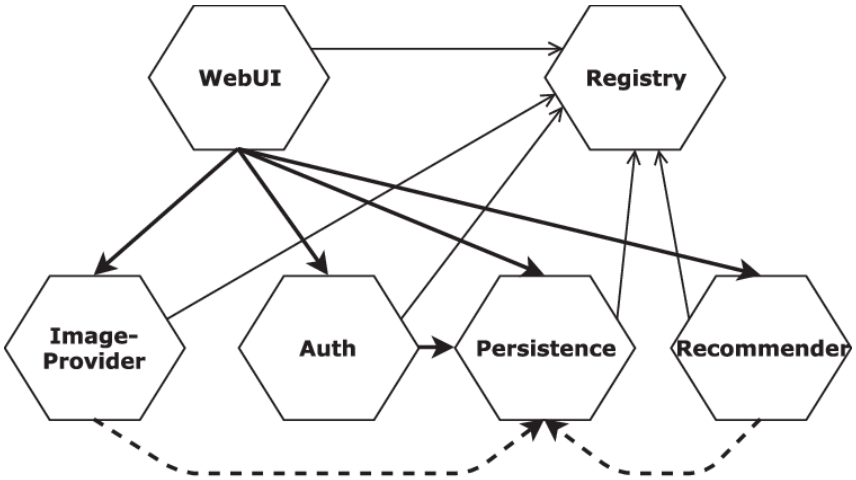

Figura 24 Arquitectura del proyecto "Tea Store" [51].

Por otro lado, a través de la aplicación de la metodología se obtuvo la arquitectura mostrada en la figura 25, en esta arquitectura propuesta, se puede observar que para el proyecto “TeaStore” se plantean 6 microservicios, cada uno de ellos respondiendo a una funcionalidad del sistema, de tal forma que:

- El microservicio “Registrar usuarios”, sirve para almacenar la información del cliente como su nombre, dirección y su medio de pago, así mismo en esta función se crea su nombre de usuario y contraseña.

- Por su parte el microservicio “Autenticar usuarios”, como su nombre lo indica servirá para validar el nombre de usuario y contraseña del cliente, dando acceso así a su perfil, donde se muestra su historial de compras.
- El microservicio “Realizar cobro” se encarga de tomar los datos de pago del cliente registrado, o en su defecto si se trata de un cliente no registrado, el microservicio en cuestión le solicita dichos datos de pago, para así proceder a realizar el cobro, al final siempre se le informa al cliente sobre el estatus de su pago.
- “Carrito de compras” es el microservicio encargado de procesar los pedidos, a través de él, se realiza la selección de artículos para agregarlos al carrito o bien quitarlos en caso que el cliente lo requiera, este mismo microservicio solicita la confirmación del pedido e imprime el ticket de compra.
- El microservicio “Realizar recomendación de artículos” se encarga de revisar los artículos seleccionados para compra, y en base a ellos y a pedidos anteriores revisa que artículos se han comprado juntos, recomendando así aquellos que no haya seleccionado el cliente.
- En lo que respecta a la alta y baja de artículos del inventario, así como de la actualización de su información, es decir de la gestión general del inventario, el microservicio encargado es “Administrar inventario”.

De acuerdo al análisis anterior, se puede ver que con los microservicios identificados el sistema cumple con todos los requerimientos solicitados, por lo cual, la arquitectura resultante por medio de la metodología, es una propuesta confiable (al menos hasta este punto) para definir los microservicios a partir de la información del proyecto “Tea Store”.

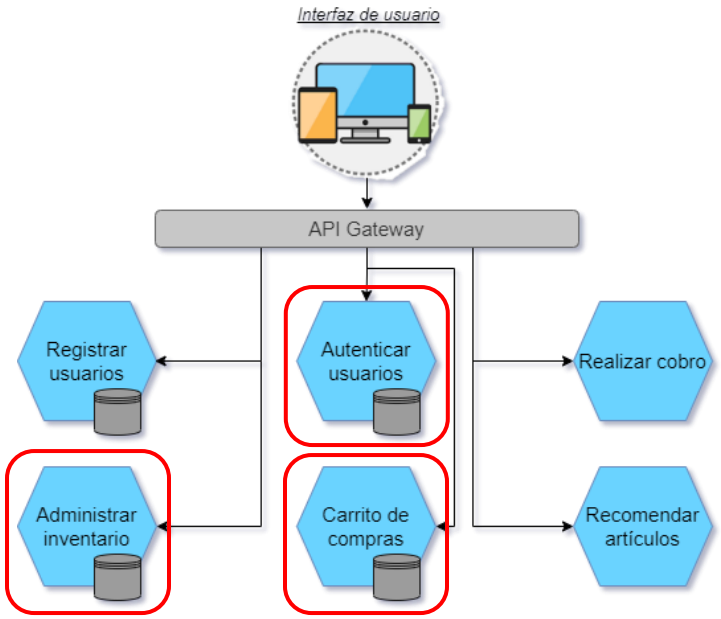

Figura 25 Arquitectura de microservicios propuesta para el proyecto "Tea Store”.

4.12.2 Otra forma de comprobar el enfoque de casos de uso.

Con el grafo obtenido en el paso 5, se realizó una adaptación para dejar solo las operaciones identificadas, quitando las variables, ya que solo se tomaron en cuenta las conexiones entre las operaciones del sistema “Tea Store”. En la figura 26 cada línea secuencial de operaciones identificada fue marcada con un color distinto:

- En color morado hay tres operaciones dependientes; pues al “Autenticar usuarios” se muestra el perfil del cliente y su historial de compras.
- De color verde esta la operación de “Registrar usuarios” que a su vez llama a la operación de “Confirmar registro”.
- La línea secuencial de operaciones correspondiente a “Realizar cobro” esta identificada de color rojo.
- Mientras que de color azul están las operaciones “Imprimir ticket”, “Agregar articulo al carrito”, “Quitar artículo del carrito” que dependen de “Carrito de compras”.
- De color café esta la operación “Realizar recomendación de artículos” que depende de “Verificar artículos seleccionados” y “Revisar artículos que se han comprado juntos”.
- Por último, en color rosa esta la operación “Administrar inventario” que causa dependencia en las operaciones “Buscar artículo”, “Dar de baja artículo”, “Modificar datos de artículo” y “Dar de alta artículo”.

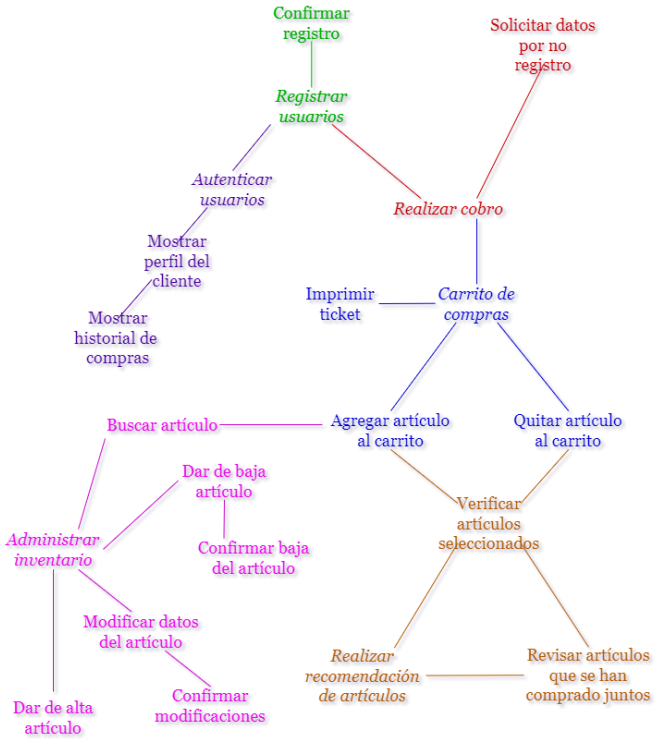

Figura 26 Comprobación de las lineas secuenciales de operaciones.

Como se observa, cada línea secuencial de operaciones corresponde con un microservicio identificado (el nombre de cada microservicio se encuentra en letra itálica), al igual que con el ejemplo “Cinema”, esta es otra forma de validar, de alguna forma, la correcta división de un sistema en microservicios.

Capítulo 5

5. Conclusiones y trabajos futuros.

5.1 Conclusiones

La definición de la granularidad adecuada es un aspecto fundamental en el desarrollo de aplicaciones basadas en microservicios. En estos últimos años desarrolladores y arquitectos de software han mostrado su preocupación por conocer cuál es la mejor forma de identificar el tamaño de los microservicios de un sistema, y es que, este tema ha sido abordado por diferentes estudios, donde se proponen distintas formas de obtener la granularidad de un microservicio.

En granularidad, decidir que cada operación de un sistema sea considerada un microservicio no es lo óptimo. Por ejemplo, si una aplicación que ofrece 100 operaciones tuviera 100 microservicios, esto no es lo adecuado, debido a diferentes factores que se verían afectados a corto o largo plazo, como la latencia, el rendimiento y la gestión de un sistema distribuido en general. El llegar a una granularidad que se considere óptima tiene que ver con las características de la aplicación, el equipo de desarrollo, los requisitos, y los recursos disponibles, pues para cada caso es distinta la forma de partición de un sistema, esto debido a que en la arquitectura por microservicios aún no se definen estándares o protocolos que seguir, lo cual deja a criterio de cada desarrollador aceptar si se trata de una correcta aplicación de dicha arquitectura.

Como ya se ha mencionado, a lo largo de esta investigación se encontraron múltiples formas en que la granularidad de los microservicios ha sido abordada, sin embargo, es importante hacer notar la ausencia de técnicas o métodos que permitan evaluar la arquitectura resultante de cada una de estas formas de determinación de granularidad, pues a pesar de las pruebas realizadas la mayoría de los autores se limitan a decir que su forma de descomposición es la adecuada porque para fines de su proyecto los microservicios creados son funcionales, lo cual no es una forma valida de sustentar sus métodos, como lo sería por ejemplo el uso de métricas o estándares, esta situación hace que sigan apareciendo más enfoques para determinar la granularidad de un microservicio, pero al no sustentarse estos métodos no son tomados en cuenta para ser estandarizados.

Está claro que, al gestionar microservicios más grandes, las pruebas pueden ser más lentas y tediosas, sobre todo cuando se comienza desde el diseño de la arquitectura pasando por la implementación y hasta llegar a la evaluación de los resultados. Determinar el número apropiado de microservicios y su impacto en el despliegue continuo es un tema de investigación interesante pero pocos trabajos abordan estas cuestiones de forma completa.

A pesar de que se han propuesto estudios interesantes, todavía hay pocas propuestas concretas y aplicables que pudieran ser aceptadas por organismos internacionales. Se necesita más investigación y pruebas que tomen en cuenta métricas, estándares de diseño, atributos de calidad, etc., para proponer patrones de diseño, buenas prácticas, métodos o herramientas que puedan estandarizarse para definir la granularidad adecuada de los microservicios.

En esta tesis se diseñó una metodología de tipo “Top-Down” que ayuda en la determinación de la granularidad de los microservicios, la cual se construyó a partir de métodos propuestos por diferentes autores, y en específico este trabajo se centró en realizar pruebas sobre dicha metodología, de forma general se puede decir que la metodología es clara al explicar los pasos para saber cómo proceder ante una descomposición en microservicios (al menos desde el enfoque de casos de uso). A continuación, se presenta una retrospectiva general de todo el proceso de aplicación de la metodología, que justamente llevó a realizar ajustes en los procesos propuestos por los autores, y también apoyo en la identificación de áreas de mejora para trabajos futuros.

Empezando por la etapa de análisis de la metodología, fue importante agregar métodos de recabación de requerimientos, como entrevistas, la observación, creación de bocetos, etc. que para este caso fueron los principales medios de obtención de datos. Así mismo para esta fase de análisis de la información fue importante aclarar que normas o estándares deben seguirse para diseñar cada diagrama de salida, para fines del diagrama de casos de uso, se propone seguir es el estándar UML, basado en la norma ISO/IEC 19501. Dado que no se realizaron pruebas con los otros enfoques, de momento no se pueden mencionar estándares para el modelado de los demás diagramas.

Para la fase dos, que se enfoca en la descomposición del sistema en los diferentes microservicios, la retrospectiva obtenida está centrada en el enfoque de casos de uso, aquí es donde como primer paso se añadió una validación al diagrama de casos de uso, dicha validación es llevada a cabo a partir de una métrica que permite medir la complejidad de un proyecto de software en base a su diagrama de casos de uso, la cual es descrita en el apartado 3.3 de este documento de tesis, esta validación permite conocer si hay una correcta construcción del diagrama de casos de uso continuando así con la aplicación de la metodología, o en su defecto si a través de la aplicación de la métrica el diagrama de casos de uso se considera un constructo no valido, se proceda al rediseño de dicho diagrama. En este sentido es importante aclarar que en ninguno de los trabajos consultados sobre la métrica de complejidad se habla sobre la interpretación que se les da a los puntajes obtenidos durante la aplicación de la métrica, este puntaje solo ayuda a validar el cumplimiento de las propiedades del marco de Briand [22], por lo cual queda en el aire entender específicamente el significado de los puntos de complejidad de la métrica de Sabharwal, Kaur y Sibal [20].

Como se realizó en cada caso de prueba, otra forma de validación para este enfoque es mediante la comprobación de que cada candidato a microservicio sigue solo una línea secuencial de operaciones, esto se hizo utilizando el grafo bipartito construido durante la aplicación de la metodología, de esta forma sí en efecto por cada candidato a microservicio solo hay una secuencia de operaciones entonces se puede decir que el microservicio responde a una sola funcionalidad, lo cual es lo ideal para esta arquitectura y fue justamente lo que sucedió en los casos de prueba “Cinema” y “TeaStore”

Durante las pruebas solo fueron tomados en cuenta los requerimientos funcionales, pues estos fueron los únicos que se usaron durante la ejecución de los pasos de la metodología. Sin embargo, también es importante hacer notar que a partir de los requerimientos funcionales se puede dar respuesta a los no funcionales, esto se pudo entender dado qué no siempre es necesario que un software esté terminado para poder evaluarlo, pues desde el diseño se pueden llevar a cabo actividades que permitan evaluar si lo que se está construyendo está tomando en cuenta a los requerimientos no funcionales.

Por último, tomando en cuenta los objetivos planteados por la presente tesis, se puede observar cómo se da un correcto cumplimiento a cada uno de estos. El objetivo general consistió en diseñar una metodología que considerara tanto los aspectos estructurales y de comportamiento, para determinar la granularidad de un microservicio, dicho objetivo fue abordado de manera exitosa, resultado de ello se tiene a la metodología presentada en este trabajo, que, si bien está basada en trabajos previos realizados, esta contiene adecuaciones que suponen mejores resultados. La metodología propuesta, remarca que la experiencia de un arquitecto de software siempre estará presente en el diseño de una arquitectura para un sistema, siendo esta experiencia un aspecto estructural para definir la granularidad de los microservicios, pues como se explicó anteriormente, se espera que, si un arquitecto de software tiene más experiencia diseñando diagramas de casos de uso, se incline por seguir ese enfoque.

Como se puede notar la metodología utiliza al diagrama de casos de uso, al modelo de dominio, a las historias de usuario y al diagrama relacional de una BD, como puntos de partida para identificar a los candidatos a microservicios, de esta forma se cumple con el objetivo que establece utilizar los diagramas de representación de las funcionalidades como aspectos de comportamiento para determinar la granularidad. En cuanto al objetivo que considera el uso de métricas para la determinación de la granularidad de un microservicio, en la metodología se incorporó la métrica de complejidad de Sabharwal, Kaur y Sibal [20], la cual, valida al diagrama de casos de uso, indicando si se realizó una correcta construcción o no de este diagrama.

En términos generales, a través de la presente tesis se puede comprobar el cumplimiento de los objetivos trazados por el tema “Metodología en la determinación de la granularidad de un microservicio”, los cuales se encargan de ayudar a los arquitectos de software en la definición objetiva del tamaño adecuado de un microservicio, cuidando el rendimiento, la confiabilidad, escalabilidad, mantenibilidad y complejidad de un sistema de software, a su vez que se entiende que el tamaño no es necesariamente una dimensión totalmente apegada a la granularidad de los microservicios.

5.2 Trabajos futuros

Dentro de los trabajos futuros identificados, uno de ellos se basa en definir que normas, estándares o patrones de diseño se pueden seguir para construir cada uno de los diagramas que no fueron probados de la metodología (modelo de dominio, diagrama relacional de una base de datos y declaración de objetivos), como se observa en la metodología presentada en la sección

3.2, en el último paso de la fase de análisis se indica que ahí será incluido el estándar que se debe seguir para la construcción del diagrama elegido.

Por otro lado, es bueno esclarecer qué representan por si solos los puntos obtenidos en la métrica de complejidad de Sabharwal, Kaur y Sibal [20], ya que no se hace la aclaración de la unidad de medida que están representando, si a mayor puntaje mayor complejidad, si a menor puntaje la complejidad es menor, etc. en si se requiere conocer cómo influye este puntaje en la forma de entender la complejidad de un sistema.

Ahora bien, respecto a los enfoques que no fueron probados (por modelo de dominio, por base de datos y por declaración de objetivos), se espera que en trabajos futuros estos puedan ser demostrados, incluso que los casos de prueba aquí abordados sean ejecutados en todos los enfoques, para tener así una comparativa de los resultados arrojados por cada uno, analizando así, si se llega a una misma arquitectura que sería lo ideal porque de esta forma se validarían entre ellos, o en su defecto si hay diferencias entre las arquitecturas resultantes, de tal manera que se pueda validar cada uno de los enfoques, como se hizo con el de casos de uso, que cabe mencionar fue una situación adicional al alcance de la tesis, pero que se consideró importante abordarla, aunque como se vio a lo largo de este documento aún hay muchas áreas de oportunidad que trabajar en el campo de los microservicios.

Por último, en conjunto con los revisores y director de tesis, se propone que el grafo bipartito no dirigido que se construye en el enfoque de casos de uso, sea cambiado por un modelo estandarizado más conocido y sencillo, como lo es el diagrama de Venn o Euler, incluso algún otro que facilite entender la forma en que se dividen los microservicios de un sistema. Para esto es importante hacer un análisis de aquellos modelos que puedan construirse a partir de la tabla de relaciones del paso cuatro del enfoque de casos de uso.

Bibliografía

[1] N. Dragoni, S. Giallorenzo, A. Lluch Lafuente, M. Mazzara, F. Montesi, R. Mustafin, L. Safina, “Microservices: yesterday, today, and tomorrow,” In: Mazzara M., Meyer B. (eds) Present and Ulterior Software Engineering. Springer, Cham. https://doi.org/10.1007/978-3-319-67425-4_12, 2017.

[2] S. Hassan, R. Bahsoon, R. Kazman, “Microservice Transition and its Granularity Problem: A Systematic Mapping Study,” Software: Practice and Experience, 2019.

[3] J. Lewis and M. Fowler. Microservices. http://martinfowler.com/articles/microservices.html, March 2014.

[4] A. Pereira Vale, G. Márquez, H. Astudillo, E. B. Fernandez, “Security Mechanisms Used in Microservices-Based Systems: A Systematic Mapping,” XLV Latin American Computing Conference (CLEI), IEEE Explore, 2019.

[5] P. Jamshidi, C. Pahl, N. C. Mendonca, J. Lewis y S. Tilkov. “Microservices, The Journey So Far and Challenges Ahead”. Publicado por la IEEE sociedad informática, mayo/junio,2018

[6] O. H. López, Método para Migrar Servicios Web bajo protocolo SOAP hacia Microservicios basados en REST. Tesis de maestría. Cuernavaca: CENIDET, 2018.

[7] I. D. Joya de la Cruz, “Selección y adaptación de Métricas para Microservicios”, Propuesta de Tesis, Cuernavaca: CENIDET, junio 2020

[8] D. Roldan Martínez, P. J. Valderas Aranda y V. Torres Bosch. “Microservicios Un enfoque integrado”. Microservicios, conceptos básicos. España. RA-MA Editorial, 2018

[9] IBM Cloud. "IBM Support. Obtenido de" https://www.ibm.com/downloads/cas/ODGVKQE7, s.f

[10] D. Shadija, M. Rezai and R. Hill, “Towards an Understanding of Microservices”, University of Huddersfield, UK, September 2017

[11] F. H. Vera-Rivera, E. G. Puerto-Cuadros, H. Astudillo y C. M. Gaona-Cuevas, Microservices Backlog - A Model of Granularity Specification and Microservice Identification**,** Conference paper, 13 September 2020 https://link.springer.com/chapter/10.1007%2F978-3-030-59592-0_6

[12] V. Martínez, Granularidad de Microservicios, junio 2020, Obtenido de https://medium.com/@vandresmartinez/granularidad-de-microservicios-3d58f3002120#:~:text=La%20granularidad%20de%20los%20servicios%20es%20un%20aspecto%20fundamental%20al,problemas%20y%20dif%C3%ADcil%20de%20mantener.

[13] C. Richardson, Microservice Architecture, What are microservices?, 31 Dec 2018. Consultado en https://microservices.io/

[14] Shadija, Dharmendra, Rezai, Mo and Hill, Richard. Microservices : granularity vs. performance. In: Companion Proceedings of the 10th International Conference on Utility and Cloud Computing - UCC '17 Companion. 2017.

[15] J. C. Canto. Arquitectura web: SOA vs. Microservices, aplicaciones y diferencias. 2018 Obtenido de https://www.bilib.es/actualidad/blog/noticia/articulo/arquitectura-web-soa-vs-microservices-aplicaciones-y-diferencias/

[16] Florencia del Medico. ¿Qué es una API y cómo funciona?. 2020. Obtenido de https://maplink.global/es/blog/como-usar-apis/

[17] M. D. Cojocaru, A. Uta, A. M. Oprescu. Attributes Assessing the Quality of Microservices Automatically Decomposed from Monolithic Applications. 2019 18th International Symposium on Parallel and Distributed Computing (ISPDC)

[18] J. Bogner, S. Wagner, A. Zimmermann, ”Automatically measuring the maintainability of service and microservice-based systems - a literature review”. in 27th International Workshop on Software Measurement, Gothenburg, Sweden, 2017.

[19] N. Medvidovic and R. N. Taylor. 2000. A Classification and Comparison Framework for So.ware Architecture Description Languages. IEEE Trans. So.w. Eng. 26, 1 (Jan. 2000), 70–93

[20] J. A. Castillo. Qué es la latencia en informática y cómo medirla. 3 enero, 2019. Obtenido de https://www.profesionalreview.com/2019/01/03/latencia-en-informatica/

[21] S. Hassan, N. Ali y R. Bahsoon. Microservice Ambients: An Architectural Meta-modelling Approach for Microservice Granularity. IEEE International Conference on Software Architecture. 2017.

[22] J. Munezero Immaculee, Tuheirwe-Mukasa, B. Kanagwa y J. Balikuddembe, Partitioning Microservices: A Domain Engineering Approach, 2018.

[23] Y. Li, C. Wang, Y. Li and J. Su, Granularity Decision of Microservice Splitting in View of Maintainability and Its Innovation Effect in Government Data Sharing. School of Management, Xi’an Polytechnic University, China, Junio 2020.

[24] J. M. Ortega Candel. Tecnologías para arquitecturas basadas en microservicios: Patrones y soluciones para aplicaciones desplegadas en contenedores. Junio 2020. Obtenido de https://books.google.com.mx/books?id=5xfzDwAAQBAJ&pg=PA39&dq=arquitectura+basada+en+Microservicios&hl=es&sa=X&ved=2ahUKEwjXo_enmY7xAhULca0KHZqaCiEQ6AEwAnoECAMQAg#v=onepage&q&f=false

[25] Maida, EG, y J. Pacienzia. Metodologías de desarrollo de software [en línea]. Tesis de Licenciatura en Sistemas y Computación. Facultad de Química e Ingeniería “Fray Rogelio Bacon”. Universidad Católica Argentina, 2015. Obtenido de https://repositorio.uca.edu.ar/bitstream/123456789/522/1/metodologias-desarrollo-software.pdf

[26] Posgrados IBERO. ¿Qué es la metodología de la investigación?. Mayo 2020. Obtenido de https://blog.posgrados.ibero.mx/metodologia-de-investigacion/

[27] T. R. Browning, Applying the design structure matrix to system decomposition and integration problems: a review and new directions. IEEE Transactions on Engineering Management 48(3) (Aug 2001)

[28] L. Baresi, M. Garriga y A. De Renzis, "Microservices identification through interface analysis," in European Conference on Service-Oriented and Cloud Computing - Lecture Notes in Computer Science., Sep. 2017, vol. 10465 LNCS, pp. 19–33, doi: 10.1007/978-3-319-67262-5_2.

[29] N. Alshuqayran, N. Ali y R. Evans, "A Systematic Mapping Study in Microservice Architecture," 2016, Accessed: Sep. 13, 2017. [Online]. Available: http://dspace.brunel.ac.uk/bitstream/2438/14968/1/FullText.pdf.

[30] F. H. Vera, "Modelo Inteligente De Especificación De La Granularidad De Aplicaciones Basadas En Microservicios". Universidad Francisco De Paula Santander. Cali - Colombia 2021

[31] A. Homay, M. de Sousa, A. Zoitl y M. Wollschlaeger, "Service Granularity in Industrial Automation and Control Systems," in 2020 25th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA), Sep. 2020, pp. 132–139, doi: 10.1109/ETFA46521.2020.9212048.

[32] C. Richardson (2019). "Microservices Patterns". Whit examples in Java. Manning Publications Co. Shelter Island, NY 11964.

[33] O. Zimmermann, "Microservices Tenets: Agile Approach to Service Development and Deployment", Universidad de Ciencias Aplicadas de Suiza Oriental, Overview and Vision Paper, SummerSoC 2016

[34] A. Kuhn, S. Ducasse y T. Gırba, "Semantic clustering: Identifying topics in source code", LISTIC, University of Savoie, France, January 2007

[35] A. Homay, A. Zoitl, M. de Sousa, M. Wollschlaeger y C. Chrysoulas, Granularity Cost Analysis for Function Block as a Service, Conference Paper · July 2019 disponible en https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.70.5280&rep=rep1&type=pdf

[36] O. Adwan, R. Yousef y M. A. M. Abushariah. Extracting SOA Candidate Software Services from an Organization's Object Oriented Models. Universidad de Jordan. Article in Journal of Software Engineering and Applications, August 2014.

[37] G. Kecskemeti, A. C. Marosi and A. Kertesz. The ENTICE approach to decompose monolithic services into microservices. Institute for Computer Science and Control, Hungarian. Conference Paper, July 2016.

[38] E. Axelsson Y E. Karlkvist. "Extracting Microservices from a Monolithic Application Developing a Migration Strategy". University of Gothenburg, Sweden 2019.

[39] A. Levcovitz, R. Terra y M. T. Valente. “Towards a Technique for Extracting Microservices from Monolithic Enterprise Systems”. Universidad Federal de Minas Gerais, Universidad Federal de Lavras, Brazil. Article, May 2016

[40] International Technical Support Organization. “Microservices from Theory to Practice: Creating, Applications in IBM Bluemix Using the Microservices”, Approach. First Edition, August 2015

[41] H. Vural. “Does Domain-Driven Design Lead to Finding the Optimal Modularity of a Microservice?”. Information & Communication Technology Department, Havelsan Inc., 06510 Ankara, Turkey. March 2, 2021.

[42] S. Sabharwal, P. Kaur y R. Sibal, “Empirical and Theoretical Validation of a Use Case Diagram Complexity Metric”, University of Delhi, INDIA, 2017.

[43] B. Bernárdez, A. Durán Toro, y M. Genero, “Empirical Evaluation and Review of a Metrics-Based Approach for Use Case Verification”. Journal of Research and Practice in Information Technology. España, 2014.

[44] L. C. Briand, S. Morasca and V. R. Basili, "Property-based software engineering measurement," in *IEEE Transactions on Software Engineering*, vol. 22, no. 1, pp. 68-86, Jan. 1996, doi: 10.1109/32.481535.

[45] C. Ramirez. “How to build a NodeJS cinema microservice and deploy it with docker”, 2017, disponible en https://medium.com/@cramirez92/build-a-nodejs-cinema-microservice-and-deploying-it-with-docker-part-1-7e28e25bfa8b

[46] F. J. García y A. García. “Fundamentos de la vista de Casos de Uso”. Departamento de Informática y Automática. Universidad de Salamanca. 2017. Disponible en https://repositorio.grial.eu/bitstream/grial/1155/1/UML%20-%20Casos%20de%20uso.pdf

[47] ISO/IEC 19501:2005. “Unified Modeling Language Specification”. Versión 1.4.2, formal/05-04-01, enero 2005.

[48] S. Tyszberowicz, R. Heinrich, B. Liu and Z. Liu. “Identifying Microservices Using Functional Decomposition”, Southwest University, China, 2018.

[49] A. Raman and S. S. Tyszberowicz. The EasyCRC tool. In ICSEA, pages 52–57. IEEE, 2007.

[50] Text analysis online. http://textanalysisonline.com/ textblob-noun-phrase-extraction. Accessed: April 2018.

[51] J. von Kistowski, S. Eismann, N. Schmitt, A. Bauer, J. Grohmann and S. Kounev, "TeaStore: A Micro-Service Reference Application for Benchmarking, Modeling and Resource Management Research," 2018 IEEE 26th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS), 2018, pp. 223-236, doi: 10.1109/MASCOTS.2018.00030.

[52] R. Santaolaya, L. Guadarrama, O. Fragoso, B. Valenzuela, J. C. Rojas. “Experimental Study To Determine The Influence Of The Internal Structure Of Web Services In Its

Response Time". DYNA New Technologies, 2019. [10 p.]. DOI: https://doi.org/10.6036/NT9111

[53] L. Rodríguez Cortés, "UF1305 - Programación con lenguajes de guion en páginas web", Elearning, ISBN: 978-84-16424-30-6, España, 2017.

[54] F. León Pérez. "Generación de Servicios Web desde Marcos Orientados a Objetos a partir de Clases Colaborativas en Casos de Uso (MOOaSW)". Tesis de maestría, Centro Nacional de Investigación y Desarrollo Tecnológico, Cuernavaca Morelos, 2009.

Anexo A - Plantillas de caso de uso y actores de la métrica de complejidad.

Tabla 26 Plantilla de los casos de uso métrica de complejidad.

ID del caso de uso:	Se asigna un número único a cada caso de uso en el diagrama de casos de uso.
Nombre del caso de uso:	Se asigna un nombre en lenguaje natural a cada caso de uso en el diagrama de casos de uso.
Descripción del caso de uso:	Se proporciona una breve descripción del caso de uso que define el requisito que representa, en lenguaje natural.
Tipo:	Principal o dependiente. • El caso de uso principal es el caso de uso que está relacionado con el actor por una relación de asociación, es decir, el caso de uso solo puede llamarse a través del actor. • El caso de uso dependiente es el caso de uso llamado por el caso de uso principal a través de la dependencia de inclusión o extensión.
Número de escenarios:	Representa el número de escenarios en el caso de uso.
Vector iniciador:	Conjunto de actores y casos de uso que pueden llamar a este caso de uso a través de relaciones de asociación o dependencia.
Vector de activación:	Conjunto de casos de uso que este caso de uso puede llamar como resultado de las dependencias de inclusión/extensión.

Tabla 27 Plantilla del actor métrica de complejidad.

ID del actor:	Se asigna un identificador alfanumérico a cada actor del diagrama de casos de uso.
Nombre del actor:	Se asigna un nombre en lenguaje natural a cada actor del diagrama de casos de uso.
Descripción del actor:	Se da una breve descripción del actor en lenguaje natural.
Vector de activación:	Conjunto de casos de uso que puede llamar este actor a través de una relación de asociación.

Anexo B - Descripción de caso de uso y actores con la plantilla de la métrica de complejidad.

Plantillas casos de uso:

Tabla 28 Descripción del caso de uso Venta de boletos.

1.	**ID:**	**002**
2.	**Nombre del Caso de Uso:**	Venta de boletos
3.	**Descripción:**	Se realiza la venta de boletos para las funciones del cine, si se presenta membrecía se aplica un descuento.
4.	**Tipo:**	Principal
5.	**Número de escenarios:**	2
6.	**Vector iniciador:**	Empleado general, gerente
7.	**Vector de activación:**	CU 004

Tabla 29 Descripción del caso de uso Venta de boletos en línea.

1.	**ID:**	**003**
2.	**Nombre del Caso de Uso:**	Venta de boletos en línea.
3.	**Descripción:**	Se realiza la venta de boletos por internet, no aplica membrecía.
4.	**Tipo:**	Dependiente
5.	**Número de escenarios:**	1
6.	**Vector iniciador:**	NA
7.	**Vector de activación:**	CU 002

Tabla 30 Descripción del caso de uso Forma de cobro.

1.	**ID:**	**004**
2.	**Nombre del Caso de Uso:**	Forma de cobro
3.	**Descripción:**	El pago puede ser realizado en efectivo, tarjeta o por mercado pago.
4.	**Tipo:**	Dependiente
5.	**Número de escenarios:**	3
6.	**Vector iniciador:**	CU 002, CU 005, CU007, CU008
7.	**Vector de activación:**	NA

Tabla 31 Descripción del caso de uso Venta de membrecias.

1.	**ID:**	**005**
2.	**Nombre del Caso de Uso:**	Venta de membrecías.
3.	**Descripción:**	En la taquilla se realiza la venta de tarjetas de “Miembro frecuente”.
4.	**Tipo:**	Principal
5.	**Número de escenarios:**	1
6.	**Vector iniciador:**	Empleado general, gerente
7.	**Vector de activación:**	CU 004, CU 006

Tabla 32 Descripción del caso de uso Administración de membrecías.

1.	**ID:**	**006**
2.	**Nombre del Caso de Uso:**	Administración de membrecías.
3.	**Descripción:**	Se lleva a cabo el alta de tarjetas de membrecía a través de sus folios.
4.	**Tipo:**	Principal.
5.	**Número de escenarios:**	1
6.	**Vector iniciador:**	Gerente
7.	**Vector de activación:**	NA

Tabla 33 Descripción del caso de uso Venta de dulcería.

1.	**ID:**	**007**
2.	**Nombre del Caso de Uso:**	Venta de dulcería.
3.	**Descripción:**	Se lleva a cabo la venta de snacks, bebidas, golosinas y artículos de colección en la dulcería.
4.	**Tipo:**	Principal.
5.	**Número de escenarios:**	2
6.	**Vector iniciador:**	Empleado general, gerente
7.	**Vector de activación:**	CU 004

Tabla 34 Descripción del caso de uso Venta de cafe central.

1.	**ID:**	**008**
2.	**Nombre del Caso de Uso:**	Venta de café central.
3.	**Descripción:**	Se lleva a cabo la venta de café en sus diferentes presentaciones y comida.
4.	**Tipo:**	Principal.
5.	**Número de escenarios:**	1
6.	**Vector iniciador:**	Empleado general, gerente
7.	**Vector de activación:**	CU 004

Tabla 35 Descripción caso de uso Validación de boletos.

1.	**ID:**	**009**
2.	**Nombre del Caso de Uso:**	Validación de boletos.
3.	**Descripción:**	A través del sistema se hace un escaneo de los boletos, para permitir la entrada a las salas de proyección.
4.	**Tipo:**	Principal.
5.	**Número de escenarios:**	1
6.	**Vector iniciador:**	Empleado general, gerente
7.	**Vector de activación:**	NA

Tabla 36 Descripción del caso de uso Definición del catálogo de películas.

1.	**ID:**	**010**
2.	**Nombre del Caso de Uso:**	Definición del catálogo de películas.
3.	**Descripción:**	El usuario registra el nombre, cast, director, sinopsis y duración de las películas disponibles. O da de baja las que ya no estarán disponibles.
4.	**Tipo:**	Principal
5.	**Número de escenarios:**	2
6.	**Vector iniciador:**	Gerente
7.	**Vector de activación:**	NA

Tabla 37 Descripción del caso de uso Asignación de películas, horarios y salas.

1.	**ID:**	**011**
2.	**Nombre del Caso de Uso:**	Asignación de películas, horarios y salas.
3.	**Descripción:**	El usuario hace la asignación de horario y salas, a las películas disponibles en el catálogo.
4.	**Tipo:**	Principal
5.	**Número de escenarios:**	1
6.	**Vector iniciador:**	Gerente, coordinador de proyección.
7.	**Vector de activación:**	CU 010

Plantillas de los actores:

Tabla 38 Descripción del actor "Empleado general".

1.	**ID del actor:**	002
2.	**Nombre del actor:**	Empleado general
3.	**Descripción del actor:**	El empleado general es un usuario de bajo rango que se encarga de las áreas de: taquilla, dulcería, café central y piso.
4.	**Vector de activación:**	CU001, CU002, CU005, CU007, CU008, CU009

Tabla 39 Descripción del actor "Coordinador de proyección".

1.	**ID del actor:**	003
2.	**Nombre del actor:**	Coordinador de proyección.
3.	**Descripción del actor:**	Empleado con dedicación exclusiva a la asignación de salas y horarios a cada una de las películas para su proyección.
4.	**Vector de activación:**	CU011

Anexo C - Cuadros de descripción detallados de los casos de uso proyecto Cinema.

Tabla 40 Descripción del caso de uso Venta de boletos.

1.	**ID:**	**CU2**		
2.	**Nombre del Caso de Uso:**	Venta de boletos		
3.	**Autor:**	César Merino Ibáñez	**Última Modificación:**	César Merino
4.	**Fecha de Creación:**	24/marzo/2022	**Fecha de la Última Modificación:**	24/Marzo/2022
5.	**Actores:**	Empleado de taquilla Gerente Sistema		
6.	**Descripción:**	Se realiza la venta de boletos para las funciones del cine, si se presenta membrecía se aplica un descuento.		
7.	**Precondiciones:**	El usuario de taquilla o el gerente se autenticaron (CU1).		
8.	**Postcondiciones:**	Se imprimen los boletos vendidos.		
9.	**Escenario Principal de éxito:**	1. El usuario entra al apartado de taquilla. 2. Un cliente solicita un boleto. 3. El usuario muestra asientos disponibles. 4. El cliente selecciona asiento. 5. Se realiza el cobro, se pasa al CU4. 6. El sistema completa la venta.		
10.	**Escenario Alterno (2):**	4. El cliente selecciona asiento. 7.1 El cliente presenta membrecía. 7.2 El usuario escanea membrecía. 7.3 El sistema aplica el descuento.		
11.	**Escenario Alterno (3):**	- -		
12.	**Escenario Alterno (4):**	- -		
13.	**Escenario de Fracaso:**	1. El usuario no tiene acceso a las funciones de taquilla. 2. No hay lugares disponibles en la sala. 3. Membrecía invalida. 4. El sistema no completa la venta.		
14.	**Prioridad:**	Alta		
15.	**Notas:**			

Tabla 41 Descripción del caso de uso Venta de boletos en línea.

1.	**ID:**	**CU3**		
2.	**Nombre del Caso de Uso:**	Venta de boletos en línea		
3.	**Autor:**	César Merino Ibáñez	**Última Modificación:**	César Merino
4.	**Fecha de Creación:**	24/marzo/2022	**Fecha de la Última Modificación:**	24/Marzo/2022
5.	**Actores:**	Sistema Comprador		
6.	**Descripción:**	Se realiza la venta de boletos por internet, no aplica membrecía.		
7.	**Precondiciones:**	El comprador debe contar con tarjeta de crédito/débito o mercado pago y un correo.		
8.	**Postcondiciones:**	Se mandan los boletos al correo electrónico.		
9.	**Escenario Principal de éxito:**	1. El cliente accede al sitio web. 2. El cliente selecciona “Compra de boletos”. 3. El cliente rellena los campos y selecciona asientos. 4. El cliente ingresa datos para pago. 5. El sistema procesa el pago, CU4. 6. El sistema manda los boletos al correo electrónico del cliente.		
10.	**Escenario Alterno (2):**	- -		
11.	**Escenario Alterno (3):**	- -		
12.	**Escenario Alterno (4):**	- -		
13.	**Escenario de Fracaso:**	1. Sitio web caído. 2. No hay asientos disponibles. 3. Los datos ingresados son inválidos. 4. Medio de pago inválido. 5. El sistema no puede procesar la compra.		
a	**Prioridad:**	Alta		
15.	**Notas:**	Solo pago con tarjeta o mercado pago.		

Tabla 42 Descripción del caso de uso Forma de cobro.

1.	**ID:**	**CU4**		
2.	**Nombre del Caso de Uso:**	Forma de cobro		
3.	**Autor:**	César Merino Ibáñez	**Última Modificación:**	César Merino
4.	**Fecha de Creación:**	24/marzo/2022	**Fecha de la Última Modificación:**	24/Marzo/2022
5.	**Actores:**	Sistema Empleado de taquilla Gerente		

6.	**Descripción:**	El pago puede ser realizado en efectivo, tarjeta o por mercado pago.
7.	**Precondiciones:**	El cliente debe contar con efectivo, tarjeta de crédito/débito o mercado pago.
8.	**Postcondiciones:**	Se efectúa el pago.
9.	**Escenario Principal de éxito:**	1. El cliente paga en efectivo. 2. El usuario procesa el pago. 3. El sistema valida el pago. 4. El pago es efectuado correctamente.
10.	**Escenario Alterno (2):**	1. El cliente paga con tarjeta. 2. El usuario procesa el pago. 3. El sistema valida el pago. 4. El pago es efectuado correctamente.
11.	**Escenario Alterno (3):**	1. El cliente paga con mercado pega. 2. El usuario procesa el pago. 3. El sistema valida el pago. 4. El pago es efectuado correctamente.
12.	**Escenario Alterno (4):**	- -
13.	**Escenario de Fracaso:**	1. El medio de pago es invalido. 2. Se presenta otra forma de pago ajena a las consideradas. 3. El pago no puede procesarse.
a	**Prioridad:**	Alta
15.	**Notas:**	

Tabla 43 Descripción del caso de uso Venta de membrecias.

1.	**ID:**	**CU5**		
2.	**Nombre del Caso de Uso:**	Venta de membrecías.		
3.	**Autor:**	César Merino Ibáñez	**Última Modificación:**	César Merino
4.	**Fecha de Creación:**	24/marzo/2022	**Fecha de la Última Modificación:**	24/Marzo/2022
5.	**Actores:**	Empleado de taquilla Gerente Sistema		
6.	**Descripción:**	En la taquilla se realiza la venta de tarjetas de “Miembro frecuente”.		
7.	**Precondiciones:**	El usuario de taquilla y el gerente se autenticaron (CU1).		
8.	**Postcondiciones:**	Se entrega la tarjeta de membrecía.		

9.	**Escenario Principal de éxito:**	1. El usuario entra al apartado de taquilla. 2. Un cliente solicita una membrecía. 3. El usuario activa tarjeta. 4. Se realiza el pago, se pasa al CU04. 5. El sistema completa la venta.
10.	**Escenario Alterno (2):**	- -
11.	**Escenario Alterno (3):**	- -
12.	**Escenario Alterno (4):**	- -
13.	**Escenario de Fracaso:**	1. El usuario no tiene acceso a las funciones de taquilla. 2. No hay membrecías disponibles. 3. Folio de la membrecía invalido. 4. El sistema no puede completar la venta.
14.	**Prioridad:**	Alta
15.	**Notas:**	

Tabla 44 Descripción del caso de uso Administración de membrecías.

1.	**ID:**	**CU6**		
2.	**Nombre del Caso de Uso:**	Administración de membrecías.		
3.	**Autor:**	César Merino Ibáñez	**Última Modificación:**	César Merino
4.	**Fecha de Creación:**	24/marzo/2022	**Fecha de la Última Modificación:**	24/Marzo/2022
5.	**Actores:**	Gerente Sistema		
6.	**Descripción:**	Se lleva a cabo el alta de tarjetas de membrecía a través de sus folios.		
7.	**Precondiciones:**	El gerente se autenticó (CU1) y cuenta con los folios de las membrecías.		
8.	**Postcondiciones:**	Se dan de alta los folios de las membrecías.		
9.	**Escenario Principal de éxito:**	1. El gerente entra a la administración de membrecías. 2. El gerente presiona dar de alta nueva membrecía. 3. El gerente ingresa datos de la membrecía. 4. El sistema guarda los datos. 5. El sistema lleva la administración de las membrecías.		
10.	**Escenario Alterno (2):**	- -		
11.	**Escenario Alterno (3):**	- -		
12.	**Escenario Alterno (4):**	- -		
13.	**Escenario de Fracaso:**	1. El gerente no tiene acceso a las funciones de administración de membrecías 2. No hay membrecías para dar de alta. 5. La administración de membrecías no se puede llevar a cabo.		

14.	**Prioridad:**	Alta
15.	**Notas:**	

Tabla 45 Descripción del caso de uso Venta de dulcería.

1.	**ID:**	**CU7**		
2.	**Nombre del Caso de Uso:**	Venta de dulcería.		
3.	**Autor:**	César Merino Ibáñez	**Última Modificación:**	César Merino
4.	**Fecha de Creación:**	24/marzo/2022	**Fecha de la Última Modificación:**	24/Marzo/2022
5.	**Actores:**	Empleado de dulcería Gerente Sistema		
6.	**Descripción:**	Se lleva a cabo la venta de snacks, bebidas, golosinas y artículos de colección en la dulcería.		
7.	**Precondiciones:**	El usuario y el gerente se autentican (CU1).		
8.	**Postcondiciones:**	Se imprime el ticket de venta.		
9.	**Escenario Principal de éxito:**	1. El usuario entra al apartado de dulcería. 2. Un cliente acude a la dulcería a comprar un artículo. 3. El usuario ingresa la orden al sistema. 4. Se realiza el pago, se pasa al CU4. 5. El sistema completa la venta.		
10.	**Escenario Alterno (2):**	3. El usuario ingresa la orden al sistema. 3.1 El cliente presenta su membrecía. 3.2 El usuario escanea la membrecía. 3.3 El sistema aplica el descuento correspondiente.		
E	**Escenario Alterno (3):**	- -		
12.	**Escenario Alterno (4):**	- -		
13.	**Escenario de Fracaso:**	1. El usuario no tiene acceso a las funciones de dulcería. 2. No hay en existencia el articulo solicitado. 3. Membrecía invalida. 4. El sistema no completa la venta.		
14.	**Prioridad:**	Alta		
15.	**Notas:**	- -		

Tabla 46 Descripción del caso de uso Venta de cafe central.

1.	**ID:**	**CU8**		
2.	**Nombre del Caso de Uso:**	Venta de café central.		
3.	**Autor:**	César Merino Ibáñez	**Última Modificación:**	César Merino
4.	**Fecha de Creación:**	24/marzo/2022	**Fecha de la Última Modificación:**	25/Marzo/2022
5.	**Actores:**	Empleado de café central Gerente Sistema		
6.	**Descripción:**	Se lleva a cabo la venta de café en sus diferentes presentaciones y comida.		
7.	**Precondiciones:**	El usuario y el gerente se autentican (CU1).		
8.	**Postcondiciones:**	Se imprime el ticket de venta.		
9.	**Escenario Principal de éxito:**	1. El usuario entra al apartado de café central. 2. Un cliente acude al café a comprar. 3. El usuario ingresa la orden al sistema. 4. Se realiza el pago, se pasa al CU4. 5. El sistema completa la venta.		
10.	**Escenario Alterno (2):**	- -		
E	**Escenario Alterno (3):**	- -		
12.	**Escenario Alterno (4):**	- -		
13.	**Escenario de Fracaso:**	1. El usuario no tiene acceso a las funciones de café central. 2. No hay en existencia el articulo solicitado. 3. El sistema no completa la venta.		
14.	**Prioridad:**	Alta		
15.	**Notas:**	- -		

Tabla 47 Descripción caso de uso Validación de boletos.

1.	**ID:**	**CU9**		
2.	**Nombre del Caso de Uso:**	Validación de boletos.		
3.	**Autor:**	César Merino Ibáñez	**Última Modificación:**	César Merino
4.	**Fecha de Creación:**	25/marzo/2022	**Fecha de la Última Modificación:**	25/Marzo/2022
5.	**Actores:**	Empleado de piso Gerente Sistema		

6.	**Descripción:**	A través del sistema se hace un escaneo de los boletos, para permitir la entrada a las salas de proyección.
7.	**Precondiciones:**	El usuario se autentica (CU1).
8.	**Postcondiciones:**	Se valida el boleto.
9.	**Escenario Principal de éxito:**	1. El usuario entra al apartado de piso. 2. El cliente presenta sus boletos. 3. El usuario escanea el boleto en el sistema. 4. El sistema valida el boleto.
10.	**Escenario Alterno (2):**	- -
E	**Escenario Alterno (3):**	- -
12.	**Escenario Alterno (4):**	- -
13.	**Escenario de Fracaso:**	1. El usuario no tiene acceso a las funciones de piso 2. El boleto es invalido. 3. El sistema rechaza el boleto.
14.	**Prioridad:**	Alta
15.	**Notas:**	El boleto puede ser digital o impreso.

Tabla 48 Descripción del caso de uso Definición del catálogo de películas.

1.	**ID:**	**CU10**		
2.	**Nombre del Caso de Uso:**	Definición del catálogo de películas.		
3.	**Autor:**	César Merino Ibáñez	**Última Modificación:**	César Merino
4.	**Fecha de Creación:**	25/marzo/2022	**Fecha de la Última Modificación:**	25/Marzo/2022
5.	**Actores:**	Gerente Sistema		
6.	**Descripción:**	El usuario registra el nombre, cast, director, sinopsis y duración de las películas disponibles. O da de baja las que ya no estarán disponibles.		
7.	**Precondiciones:**	El usuario cuenta con los datos de las películas a registrar.		
8.	**Postcondiciones:**	Se da de alta o baja a las películas correspondientes.		
9.	**Escenario Principal de éxito:**	1. El usuario entra al apartado de definición de catálogo. 2. El usuario ingresa los datos de la película. 3. El usuario guarda la información. 4. El sistema valida el registro.		

10.	**Escenario Alterno (2):**	1. El usuario entra al apartado de definición de catálogo. 2. El usuario selecciona la película a dar de baja. 3. El usuario acepta dar de baja la película. 4. El sistema da de baja los datos de la película.
E	**Escenario Alterno (3):**	- -
12.	**Escenario Alterno (4):**	- -
13.	**Escenario de Fracaso:**	1. El usuario no tiene acceso a las funciones de definición de catálogo de películas. 2. El usuario no ingresa los datos completos solicitados. 3. El sistema no puede realizar el registro o la baja.
14.	**Prioridad:**	Alta
15.	**Notas:**	- -

Tabla 49 Descripción del caso de uso Asignación de películas, horarios y salas.

1.	**ID:**	**CU11**		
2.	**Nombre del Caso de Uso:**	Asignación de películas, horarios y salas.		
3.	**Autor:**	César Merino Ibáñez	**Última Modificación:**	César Merino
4.	**Fecha de Creación:**	25/marzo/2022	**Fecha de la Última Modificación:**	25/Marzo/2022
5.	**Actores:**	Gerente Coordinador de proyección Sistema		
6.	**Descripción:**	El usuario hace la asignación de horario y salas, a las películas disponibles en el catálogo.		
7.	**Precondiciones:**	Debe haber películas registradas en el catálogo.		
8.	**Postcondiciones:**	Se obtiene la cartelera a publicar.		
9.	**Escenario Principal de éxito:**	1. El usuario entra al apartado de asignación de películas, horarios y salas. 2. El usuario realiza la asignación de horarios a las películas. 3. El usuario realiza la asignación de salas. 4. El sistema valida las asignaciones.		
10.	**Escenario Alterno (2):**	- -		
11.	**Escenario de Fracaso:**	1. El usuario no tiene acceso a las funciones de asignación. 2. El sistema detecta asignaciones cruzadas. 3. El sistema no puede realizar la asignación por factores no contemplados.		
12.	**Prioridad:**	Alta		
13.	**Notas:**	- -		

Anexo D - Descripción de caso de uso y actores con la plantilla de la métrica de complejidad.

Tabla 50 Plantilla del caso de uso Autenticación.

1.	**ID:**	**002**
2.	**Nombre del Caso de Uso:**	Autenticación
3.	**Descripción:**	A través de la validación de un nombre de usuario y contraseña se puede iniciar sesión en el sistema.
4.	**Tipo:**	Principal
5.	**Número de escenarios:**	2
6.	**Vector iniciador:**	Cliente, Sistema, Administrador.
7.	**Vector de activación:**	NA

Tabla 51 Plantilla del caso de uso Carrito de compras.

1.	**ID:**	**003**
2.	**Nombre del Caso de Uso:**	Carrito de compras
3.	**Descripción:**	Un cliente realiza la compra de un artículo de la tienda de té en línea.
4.	**Tipo:**	Principal
5.	**Número de escenarios:**	4
6.	**Vector iniciador:**	Cliente
7.	**Vector de activación:**	CU4, CU5

Tabla 52 Plantilla del caso de uso Realizar cobro.

1.	**ID:**	**004**
2.	**Nombre del Caso de Uso:**	Realizar cobro
3.	**Descripción:**	El sistema procede a validar y realizar el cobro.
4.	**Tipo:**	Principal
5.	**Número de escenarios:**	2
6.	**Vector iniciador:**	Sistema, CU3
7.	**Vector de activación:**	NA

Tabla 53 Plantilla del caso de uso Administración de inventario.

1.	**ID:**	**005**
2.	**Nombre del Caso de Uso:**	Administración de inventario.
3.	**Descripción:**	Se lleva a cabo el alta, baja y edición de los artículos a publicarse en la TeaStore.
4.	**Tipo:**	Principal
5.	**Número de escenarios:**	3
6.	**Vector iniciador:**	Sistema, Administrador, CU3
7.	**Vector de activación:**	CU 004, CU 006

Tabla 54 Plantilla del caso de uso Recomendaciones.

1.	**ID:**	**006**
2.	**Nombre del Caso de Uso:**	Recomendaciones
3.	**Descripción:**	El sistema recomendará tres productos en los que el usuario podría estar interesado, esto a partir del contenido del carrito de compras y de su historial.
4.	**Tipo:**	Principal.
5.	**Número de escenarios:**	2
6.	**Vector iniciador:**	Sistema
7.	**Vector de activación:**	NA

Descripción de actores:

Tabla 55 Descripción del actor "Cliente".

1.	**ID del actor:**	001
2.	**Nombre del actor:**	Cliente
3.	**Descripción del actor:**	Un cliente es un usuario externo que desea realizar alguna compra a través de la TeaStore.
4.	**Vector de activación:**	CU001, CU002, CU003

Tabla 56 Descripción del actor "Sistema".

1.	**ID del actor:**	002
2.	**Nombre del actor:**	Sistema
3.	**Descripción del actor:**	El sistema se refiere al sitio web que se va a desarrollar para la venta de artículos de té.
4.	**Vector de activación:**	CU001, CU002, CU003, CU004, CU005, CU006

Tabla 57 Descripción del actor "Administrador".

1.	**ID del actor:**	003
2.	**Nombre del actor:**	Administrador
3.	**Descripción del actor:**	El administrador es un usuario interno del sistema que se encarga de la administración del inventario de la TeStore.
4.	**Vector de activación:**	CU002, CU005

Anexo E - Cuadros de descripción detallados de los casos de uso proyecto Tea Store.

Tabla 58 Descripción del caso de uso Autenticación.

1.	**ID:**	**CU2**		
2.	**Nombre del Caso de Uso:**	Autenticación		
3.	**Autor:**	César Merino Ibáñez	**Última Modificación:**	César Merino
4.	**Fecha de Creación:**	04/agosto/2022	**Fecha de la Última Modificación:**	08/agosto/2022
5.	**Actores:**	Cliente Sistema Administrador		
6.	**Descripción:**	A través de la validación de un nombre de usuario y contraseña se puede iniciar sesión en el sistema.		
7.	**Precondiciones:**	Registro previo de los clientes ya administrador.		
8.	**Postcondiciones:**	Al ingresar al sistema se muestra el perfil del cliente y su historial de compras.		

9.	**Escenario Principal de éxito:**	1. Se inicia el sistema. 2. El cliente ingresa su nombre de usuario. 3. El cliente ingresa su contraseña. 4. El sistema valida el nombre de usuario y contraseña. 5. El sistema da acceso a sus funciones de acuerdo con los datos detectados.
10.	**Escenario Alterno (2):**	1. Se inicia el sistema. 2. El administrador ingresa su nombre de usuario. 3. El administrador ingresa su contraseña. 4. El sistema valida el nombre de usuario y contraseña. 5. El sistema da acceso a sus funciones de acuerdo con los datos detectados.
11.	**Escenario Alterno (3):**	- -
12.	**Escenario de Fracaso:**	1. El sistema no inicia. 2. Se ingresa un nombre de usuario invalido. 3. La contraseña ingresada es incorrecta. 4. No se da acceso a la cuenta.
13.	**Prioridad:**	Alta
14.	**Notas:**	

Tabla 59 Descripción del caso de uso Carrito de compras.

1.	**ID:**	**CU3**		
2.	**Nombre del Caso de Uso:**	Carrito de compras		
3.	**Autor:**	César Merino Ibáñez	**Última Modificación:**	César Merino
4.	**Fecha de Creación:**	04/agosto/2022	**Fecha de la Última Modificación:**	08/agosto/2022
5.	**Actores:**	Cliente Sistema		
6.	**Descripción:**	Un cliente realiza la compra de un artículo de la tienda de té en línea.		
7.	**Precondiciones:**	El cliente inicia sesión en el sistema.		
8.	**Postcondiciones:**	Se realiza la compra exitosamente.		

9.	**Escenario Principal de éxito:**	7. El cliente inicia sesión en la TeaStore. 8. El cliente elige el producto(s). 9. El cliente se dirige al carrito de compras. 10. El cliente selecciona pagar. 11. El sistema solicita confirmar pedido. 12. El cliente confirma la compra. 13. El sistema imprime ticket de compra.
10.	**Escenario Alterno (2):**	1. El cliente abre la página de la TeaStore. 2. El cliente elige el producto(s). 3. El cliente se dirige al carrito de compas. 4. El cliente selecciona pagar. 5. El sistema solicita más información. 10.1 Dirección de envío. 10.2 Datos de pago. 10.3 Datos de facturación (en caso de que lo requiera). 6. El sistema solicita confirmar el pedido. 7. El cliente confirma la compra. 8. El sistema imprime ticket de compra.
11.	**Escenario Alterno (3):**	1. El cliente inicia sesión en la TeaStore. 2. El cliente elige el producto(s). 3. El cliente se dirige al carrito de compras. 4. El cliente desea agregar artículos. 4.1 El cliente regresa al catálogo. 4.2 El cliente agrega artículo(s). 5. El cliente se dirige al carrito de compras. 6. El cliente selecciona pagar. 7. El sistema solicita confirmar pedido. 8. El cliente confirma la compra. 9. El sistema imprime ticket de compra.
12.	**Escenario Alterno (4):**	1. El cliente inicia sesión en la TeaStore. 2. El cliente elige el producto(s). 3. El cliente se dirige al carrito de compras. 4. El cliente desea quitar artículos. 4.1 El cliente elimina artículo(s). 5. El cliente se dirige al carrito de compras. 6. El cliente selecciona pagar. 7. El sistema solicita confirmar pedido. 8. El cliente confirma la compra. 9. El sistema imprime ticket de compra.
13.	**Escenario de Fracaso:**	6. El usuario no puede iniciar sesión. 7. No se puede acceder a la TeaStore. 8. Datos de domicilio inválidos. 9. Datos de facturación inválidos 10. El sistema no completa la venta.

14.	**Prioridad:**	Alta
15.	**Notas:**	

Tabla 60 Descripción del caso de uso Realizar cobro.

1.	**ID:**	**CU4**		
2.	**Nombre del Caso de Uso:**	Realizar cobro		
3.	**Autor:**	César Merino Ibáñez	**Última Modificación:**	César Merino
4.	**Fecha de Creación:**	08/agosto/2022	**Fecha de la Última Modificación:**	08/agosto/2022
5.	**Actores:**	Sistema		
6.	**Descripción:**	El sistema procede a validar y realizar el cobro.		
7.	**Precondiciones:**	El cliente debe contar con tarjeta de crédito o débito y confirmar que desea hacer el pedido.		
8.	**Postcondiciones:**	Se efectúa el cobro.		
9.	**Escenario Principal de éxito:**	5. El cliente logueado confirma el pedido. 6. El sistema toma los datos de su registro. 7. El sistema valida el pago. 8. El pago es efectuado correctamente. 9. El sistema manda mensaje de “Pago exitoso”.		
10.	**Escenario Alterno (2):**	5. El cliente selecciona pagar. 6. El sistema solicita los datos de cobro. 7. El cliente confirma el pedido. 8. El sistema valida el pago. 9. El pago es efectuado correctamente. 10. El sistema manda mensaje de “Pago exitoso”.		
11.	**Escenario Alterno (3):**	- -		
12.	**Escenario Alterno (4):**	- -		
13.	**Escenario de Fracaso:**	4. El sistema no puede obtener los datos de cobro del cliente registrado. 5. El medio de pago es invalido. 6. El pago no puede procesarse.		
14.	**Prioridad:**	Alta		
15.	**Notas:**			

Tabla 61 Descripción del caso de uso Administración de inventario.

1.	**ID:**	**CU5**		
2.	**Nombre del Caso de Uso:**	Administración de inventario.		
3.	**Autor:**	César Merino Ibáñez	**Última Modificación:**	César Merino
4.	**Fecha de Creación:**	08/agosto/2022	**Fecha de la Última Modificación:**	08/agosto/2022
5.	**Actores:**	Sistema Administrador		
6.	**Descripción:**	Se lleva a cabo el alta, baja y edición de los artículos a publicarse en la TeaStore.		
7.	**Precondiciones:**	Se tiene la información de los artículos a dar de alta, a dar de baja o a modificar.		
8.	**Postcondiciones:**	Se actualiza la información en la TeaStore.		
9.	**Escenario Principal de éxito:**	6. El administrador accede al apartado de “Inventario” del sistema. 7. Se da de alta un producto nuevo. 7.1 Se registran sus datos. 8. Se confirma el alta del artículo. 9. El sistema guarda la información. 10. Se actualizan los datos en la página de la TeaStore.		
10.	**Escenario Alterno (2):**	1. El administrador accede al apartado de “Inventario” del sistema. 2. Se da de baja un artículo. 2.1 Se elimina manualmente un artículo. 2.2 Se elimina un artículo por la venta de este. 3. El sistema guarda la información. 4. Se actualizan los datos en la página de la TeaStore.		
11.	**Escenario Alterno (3):**	1. El administrador accede al apartado de “Inventario” del sistema. 2. Se modifica la información de un artículo. 2.1 Se busca el artículo deseado. 2.2 Se selecciona la opción “Modificar datos”. 2.3 Se realizan los cambios. 3. Se confirman los cambios realizados. 4. El sistema guarda la información. 5. Se actualizan los datos en la página de la TeaStore.		
12.	**Escenario Alterno (4):**	- -		
13.	**Escenario de Fracaso:**	4. No se tienen todos los datos del artículo a dar de alta. 5. No se encuentra el artículo que se desea modificar. 11. El sistema no puede dar de alta, baja o modificar un artículo.		

14.	**Prioridad:**	Alta
15.	**Notas:**	

Tabla 62 Descripción del caso de uso Recomendaciones.

1.	**ID:**	**CU6**		
2.	**Nombre del Caso de Uso:**	Recomendaciones		
3.	**Autor:**	César Merino Ibáñez	**Última Modificación:**	César Merino
4.	**Fecha de Creación:**	08/agosto/2022	**Fecha de la Última Modificación:**	08/agosto/2022
5.	**Actores:**	Sistema		
6.	**Descripción:**	El sistema recomendará tres productos en los que el usuario podría estar interesado, esto a partir del contenido del carrito de compras y de su historial.		
7.	**Precondiciones:**	Haber seleccionado algún artículo para su compra, o en el caso de un usuario registrado haber generado compras anteriormente.		
8.	**Postcondiciones:**	Se muestran recomendaciones de productos.		
9.	**Escenario Principal de éxito:**	1. El sistema verifica que artículos está comprando el cliente. 2. En base al artículo seleccionado el sistema revisa cuales se han comprado juntos. 3. El sistema recomienda tres artículos en base a los resultados anteriores.		
10.	**Escenario Alterno (2):**	1. El sistema verifica que artículos ha comprado un cliente registrado. 2. En base a su historial de compras el sistema revisa cuales artículos se han comprado junto a los que el cliente ya adquirió. 3. El sistema recomienda tres artículos en base a los resultados anteriores.		
11.	**Escenario Alterno (3):**	- -		
12.	**Escenario de Fracaso:**	1. El cliente no ha seleccionado algún artículo para comprar. 2. El cliente registrado no ha realizado compras. 3. No se han comprado artículos similares al seleccionado. 4. El sistema no da recomendaciones de artículos que le pueden interesar al cliente.		
13.	**Prioridad:**	Alta		
14.	**Notas:**			